

Alex Bortolotti

# RELOOKER SON THÈME

---

Apprenez les bases du code,  
gagnez en autonomie et  
personnalisez n'importe  
quel thème WordPress



Alex Bortolotti

# Relooker son thème

*À Marina et aux personnes qui me soutiennent.*

# Table des matières

<b>Introduction : Déroulez le tapis rouge à vos visiteurs .....</b>	<b>7</b>
---	----------

<b>Chapitre 1 : Prérequis avant de foncer dans le code.....</b>	<b>11</b>
---	-----------

Chapitre 1.1 : Que voulez-vous faire et pourquoi ? .....	12
--	----

Chapitre 1.2 : Ne mettez pas les mains dans le cambouis (enfin pas tout de suite).....	19
--	----

Chapitre 1.3 : Comment fonctionne un site internet .....	31
--	----

<b>Chapitre 2 : Le HTML, la structure des pages web.....</b>	<b>39</b>
--	-----------

Chapitre 2.1 : La syntaxe du HTML .....	40
---	----

Chapitre 2.2 : La structure des pages web.....	45
--	----

Chapitre 2.3 : Les balises HTML de la section « head » .....	50
--	----

Chapitre 2.4 : Les balises HTML de la section « body » .....	61
--	----

Chapitre 2.5 : Les balises HTML de la section « body » – Partie 2.....	67
--	----

Chapitre 2.6 : Créer des formulaires en HTML.....	73
---	----

Chapitre 2.7 : Les balises génériques.....	86
--	----

<b>Chapitre 3 : Le CSS, la mise en forme des sites web .....</b>	<b>95</b>
--	-----------

Chapitre 3.1 : Les principes du langage CSS.....	96
--	----

Chapitre 3.2 : Les principes du langage CSS – Partie 2 .....	106
--	-----

Chapitre 3.3 : Les propriétés CSS – Mettre le texte en forme .....	117
--	-----

Chapitre 3.4 : Les propriétés CSS – Bordures et arrière-plans.....	130
--	-----

Chapitre 3.5 : Les propriétés CSS – Dimensionner et ajuster les éléments .....	140
--	-----

Chapitre 3.6 : Les propriétés CSS – Positionner et gérer l’affichage des blocs .....	147
--	-----

Chapitre 3.7 : Le responsive et 7 bonnes pratiques en CSS.....	155
--	-----

## **Chapitre 4 : Comment fonctionne un thème WordPress ..... 169**

Chapitre 4.1 : La structure de fichiers d'un thème WordPress.....	170
Chapitre 4.2 : Introduction au langage PHP .....	192
Chapitre 4.3 : Les fonctions usuelles des thèmes WordPress.....	211
Chapitre 4.4 : La boucle WordPress et ses fonctions associées .....	223
Chapitre 4.5 : Les modèles de page personnalisés .....	233
Chapitre 4.6 : Le fichier functions.php et les hooks.....	237
Chapitre 4.7 : Les menus et les widgets dans WordPress .....	248

## **Chapitre 5 : Comment relooker son thème en pratique ..... 258**

Chapitre 5.1 : Relooker son thème en direct, la fausse bonne idée .....	259
Chapitre 5.2 : Utiliser un thème enfant, la pratique indispensable .....	266
Chapitre 5.3 : Simuler des modifications avec l'inspecteur de code.....	279
Chapitre 5.4 : Enregistrer ses personnalisations dans l'éditeur de code.....	295
Chapitre 5.5 : Gagnez en productivité avec ces 9 outils de développement .....	310
Chapitre 5.6 : 11 Erreurs à éviter absolument en relookant un thème .....	324
Chapitre 5.7 : Comment mettre son nouveau thème en ligne.....	335

## **Chapitre 6 : 17 modifications à copier/coller pour se lancer ..... 348**

Chapitre 6.1 : 10 Astuces CSS pour relooker son thème .....	349
Chapitre 6.2 : 7 Astuces WordPress pour relooker son thème.....	368

## **Conclusion : Vers l'infini et au delà..... 392**

7 Bonnes adresses pour aller plus loin.....	394
Glossaire .....	395
Remerciements .....	400

**RELOOKER SON THÈME**

# **INTRODUCTION**

**DÉROULEZ LE TAPIS ROUGE  
À VOS VISITEURS**

---

**Découvrez pourquoi  
vous devez lire ce guide  
et comment il va vous aider  
à progresser**

# Introduction

Cela fait longtemps que je crois que le travail d'un webmaster s'apparente à celui d'un maître d'hôtel. Par là, j'entends que **chaque site doit proposer une expérience exceptionnelle à ses visiteurs** (ses hôtes).

C'est le meilleur moyen pour qu'ils passent à l'action, qu'ils reviennent à l'avenir et qu'ils en parlent autour d'eux.

Cette expérience est principalement transmise par le design, l'ergonomie et le contenu de votre site.

Avec WordPress, il est désormais beaucoup plus simple de publier du contenu. Ce n'est d'ailleurs pas un hasard s'il est installé sur 24 % des sites dans le monde.

Concernant le design et l'ergonomie, cela est un peu plus subtil.

Certes, des milliers de thèmes WordPress sont disponibles gratuitement et d'autres proposés à la vente sur des boutiques mais au fond, tout le monde désire que son site soit unique.

Ce que vous écrivez/publiez/proposez est unique.

**Votre site et surtout l'expérience qu'il procure doit l'être aussi.** Sinon pourquoi prêterait-on attention à vous ?

Pour atteindre cette singularité, installer et configurer un thème WordPress ne suffit pas.

Il faut toujours procéder à des ajustements, des adaptations et même à des ajouts pour en faire quelque chose d'unique.

Et pour y parvenir, il va falloir ouvrir le capot de votre thème WordPress et mettre les mains dans le cambouis (enfin dans le code).

*Jusqu'à présent, consulter le code de votre thème vous impressionnait sûrement. Une fois que vous aurez achevé la lecture de ce guide, vous serez beaucoup plus sûr de vous.*

En effet, Relooker son Thème va vous transmettre les bases du code pour personnaliser n'importe quel thème WordPress.

Cela vous permettra d'apporter la touche personnelle qui manque tant à votre site tout en enchantant ses visiteurs.

## **Pourquoi avoir écrit ce guide**

Après avoir créé le blog WP Marmite pour partager « des bonnes recettes » autour de WordPress, j'ai constaté que beaucoup de personnes avaient besoin d'aller plus loin que les options par défaut de leur thème.

Souvent, on a besoin de faire des modifications ici et là pour finaliser son site. On peut aussi aller plus loin en ajoutant des choses pour rendre son site plus efficace.

Le problème est qu'il ne faut pas faire n'importe quoi en modifiant les fichiers de son thème WordPress car on prend le risque de tout casser et rendre son site inaccessible.

Il faudra ensuite faire appel à un développeur pour tout remettre sur pied, ce qui n'est pas gratuit.

J'avais à coeur de faire quelque chose pour aider un maximum de personnes à se doter de ces compétences. C'est ainsi que l'idée de Relooker son Theme est née.

De plus, **connaître les bases du code et le fonctionnement d'un site internet est à mon sens indispensable aujourd'hui** (c'est un peu comme une voiture, on n'a pas vraiment besoin de savoir comment cela fonctionne exactement mais lorsqu'on le sait, on en fait un bien meilleur usage).

*Bref, que vous soyez webmaster, blogueur ou créateur de site débutant, Relooker son Thème vous guidera pas à pas afin de vous faire gagner en autonomie avec votre site.*

## **Comment lire ce guide**

Selon votre profil, il y a plusieurs façons de tirer parti du contenu de Relooker son Thème.

Si vous n'avez aucune notion en code, il vaut mieux lire Relooker son Thème du début à la fin en faisant les exercices au fur et à mesure. Cela vous permettra de vous former progressivement sans vous perdre.

Dans le cas où vous possédez déjà quelques notions, vous pouvez piocher des informations dans chaque chapitre. Vous complétez ainsi vos connaissances selon les besoins de vos projets.

À vous de voir ce qui vous convient le mieux.

Pensez aussi à y revenir de temps en temps pour bien intégrer les concepts abordés au fil des chapitres.

*Pour acquérir ces nouvelles compétences, la lecture ne suffit pas. Vous devez passer à l'action et vous entraîner régulièrement, que ce soit sur un projet réel ou fictif. C'est la clé de votre réussite.*

Sans pratique, le contenu de ce guide ne vous sera d'aucune utilité.

Ceci étant posé, je vous propose de débiter la lecture en passant au premier chapitre.

**RELOOKER SON THÈME**

# **CHAPITRE 1**

## **PRÉREQUIS AVANT DE FONCER DANS LE CODE**

---

**Posez les bases de votre projet  
de relooking de thème WordPress  
et comprenez comment fonctionne  
un site internet**

## Chapitre 1.1

# Que voulez-vous faire et pourquoi ?

*Comment personnaliser l'apparence de mon thème WordPress ? Quels fichiers dois-je modifier ?  
Quels outils dois-je utiliser ? Comment est-ce que tout cela fonctionne ?!*

*Au secours ! Je ne m'y retrouve plus !*

C'est probablement ce qu'il se passe dans votre tête en ce moment. De plus, il y a fort à parier que débiter la lecture doit faire émerger des dizaines d'autres questions.

Vous devez être pressé d'entrer dans le vif du sujet.

Vous avez soif de connaissance. Vous désirez enfin comprendre la logique qui se cache derrière tout le code que renferme votre thème WordPress.

Rassurez-vous, nous allons y venir.

Pour que votre projet de personnalisation de thème WordPress soit une réussite, nous allons devoir prendre un peu de recul afin de poser les bases.

Lorsque l'on rénove un appartement, on ne fonce pas tête baissée avec le premier pot de peinture venu. On prend soin de poncer soigneusement les boiseries, de décoller l'ancien papier peint, de lisser les murs, de lessiver les plafonds, etc (oui, oui, ça sent le vécu).

Ensuite, on applique une peinture adaptée au plafond avec la bonne méthode. Idem pour les murs et les boiseries. On laisse sécher sans faire de courants d'air, puis on enchaîne avec une seconde couche et une troisième si nécessaire.

C'est exactement la même chose avec un thème WordPress. Si on se rue dans le code sans préparer son travail, on va droit dans le mur (surtout lorsque l'on a peu d'expérience).

Au cours de ce premier chapitre, nous allons voir comment préparer votre projet pour que le relooking de votre thème se fasse le plus rapidement et efficacement possible.

## Établir votre plan d'action

Si vous avez lu les guides que je propose gratuitement sur [WP Marmite](#), vous savez que j'adore faire prendre un crayon et une feuille de papier afin de poser les choses au clair.

Autant être franc, vous n'allez pas y échapper avec Relooker son Thème ;)

Partons du principe que vous disposez d'un site web équipé de WordPress. Après avoir passé pas mal de temps à rechercher un thème WordPress, vous l'avez finalement trouvé.

Ce thème correspond presque au site que vous voulez mettre en place. Et c'est bien ce « presque » qui pose problème.

Comme me l'a confié une lectrice :

*“Aucun thème n'est parfaitement adapté à ce que l'on veut en faire : ils vont du trop simple (ne possèdent pas les fonctions voulues, le look ne plaît pas trop), au trop complexe (super look mais beaucoup trop de fonctionnalités pour l'usage prévu).” — Noelle Y.*

Bref, il y a toujours quelque chose à modifier sur un thème WordPress.

C'est une situation que je connais bien avec WP Marmite. Pour tout vous dire, j'ai une liste de modifications à apporter au blog longue comme le bras.

Si on s'écoute, on peut y passer des heures.

Malgré tout, c'est ici que se trouve le piège.

**Si vous vous attardez sur des détails insignifiants, vous ne vous occuperez pas de ce qui compte pour votre site.**

En tant qu'auteur de ce guide, je pourrais vous encourager à faire en sorte que votre thème relooké soit parfait mais ce n'est pas la meilleure chose à faire.

Au lieu de cela, je compte vous aider à dresser *une liste des modifications à accomplir classées par priorité*, c'est ce que j'appelle « La liste TOP » (Tâches et Objectifs Prioritaires).

Grâce à cette liste, vous aurez une meilleure vision de ce qu'il faut faire sur votre site et vous pourrez vous dire : « Ça, c'est super urgent. » et « Cela peut attendre ».

En complément de Relooker son Thème, vous trouverez dans le dossier Ressources un modèle de liste TOP vierge.

Avant de vous y précipiter, continuez la lecture pour découvrir...

## **Comment remplir votre liste TOP**

Étant donné que vous désirez personnaliser votre site, vous devez certainement avoir plusieurs pistes d'améliorations à l'esprit.

Pour vous aider à organiser vos idées, suivez les trois étapes de ce que j'appelle « la moulinette à idées ».

Quand vous aurez fait cet exercice, vous aurez une vision précise de ce que vous devez accomplir sur votre site.

## **1. Notez vos idées en vrac sur une feuille**

L'important ici est de bien définir ce que vous voulez faire. Ne soyez pas vague.

Comme le disait Boileau :

*Ce que l'on conçoit bien s'énonce clairement,  
et les mots pour le dire arrivent aisément.*

Par exemple, on peut avoir :

- Créer un modèle de page sans menu ni barre latérale
- Augmenter la taille du texte dans mes articles et mes pages
- Modifier l'apparence des boutons de partage

Je vous encourage à parcourir votre site pour lister tout ce qui pourrait être amélioré.

Pour l'instant, ne vous posez pas de questions concernant ces modifications. Notez-en un maximum.

## **2. Utilisez le mot magique « pour » à la fin de chaque idée**

Le mot « pour » est effectivement magique car il va vous aider à clarifier les choses.

Le but est de trouver une bonne raison de procéder à chaque modification. Si vous ne trouvez pas de justification, c'est que votre idée n'est probablement pas si pertinente.

En reprenant nos trois exemples, on obtient :

- Créer un modèle de page sans menu ni barre latérale *pour proposer mes services*
- Augmenter la taille du texte dans mes articles et mes pages *pour gagner en lisibilité*
- Modifier l'apparence des boutons de partage *pour les harmoniser avec le reste du site*

*Rappelez-vous bien que chaque modification ou ajout doit être utile aux lecteurs de votre site ou vous aider à atteindre un objectif précis.*

Attention : N'ajoutez pas des éléments parce que vous trouvez ça beau. Gardez en tête que votre site a un objectif et que rien ne doit aller à son encontre.

### **3. Donnez une priorité à chaque modification**

C'est ici que la liste TOP va être utile. Lorsque vous aurez terminé de lister les modifications à accomplir sur votre site, classez-les par priorité dans votre liste TOP.

En ouvrant la liste TOP vierge, vous verrez qu'elle se compose de 3 colonnes :

1. Impératif
2. Utile
3. Superflu

Ajoutez ensuite toutes les modifications dans votre liste TOP en choisissant la colonne la plus appropriée.

De grâce, soyez honnête et ne mettez pas tout dans « Impératif ». Tout n'est pas **vraiment** urgent. Le monde ne s'écroulera pas si vous ne changez pas la couleur de tel ou tel élément.

Au lieu de ça, utilisez plutôt les justifications que vous avez trouvées auparavant (grâce au mot « pour »).

On pourrait classer nos trois exemples de la manière suivante :

- [Impératif] Créer un modèle de page sans menu ni barre latérale pour proposer mes services
- [Utile] Augmenter la taille du texte dans mes articles et mes pages pour gagner en lisibilité
- [Superflu] Modifier l'apparence des boutons de partage pour les harmoniser avec le reste du site

Pourquoi ai-je procédé ainsi ? Étudions ces choix un par un.

1. Créer un modèle de page sans distractions va permettre aux lecteurs de se concentrer uniquement sur le contenu de la page qu'ils liront, en l'occurrence la description d'un service. L'objectif est qu'ils vous contactent, pas qu'ils changent de page. Il est donc impératif de créer ce modèle de page.
2. Avoir du texte de plus grande taille va rendre votre site plus lisible. Personne n'aime coller ses yeux près d'un écran, toutefois si l'augmentation est légère cela n'est pas indispensable. C'est pour cela que j'ai placé cette idée dans la colonne « Utile ».
3. Même si beaucoup de monde vante les boutons de partage (moi y compris), en réalité peu de monde les utilise. Changer leur couleur ne sera juste qu'un effet de style mais n'apportera rien de concret aux lecteurs. C'est donc du superflu.

## Que faire avec votre liste TOP ?

Une fois que vous aurez mis sur pied votre liste TOP grâce à la moulinette à idées, votre plan d'action va se dérouler sous vos yeux.

Comme vous pouvez vous en douter, vous allez devoir réaliser les tâches impératives en priorité. Si vous avez encore du temps, accomplissez les tâches utiles. Et seulement si toutes vos autres tâches sont terminées, acquittez-vous des tâches superflues.

En chemin, de nouvelles idées de modifications apparaîtront sûrement. Passez-les à travers la moulinette à idées afin de les placer dans la bonne colonne de votre liste TOP.

*Vous constaterez que la liste ne se terminera jamais.  
Il y aura toujours des choses à faire.*

Ne vous laissez pas piéger par votre liste TOP. Il se peut que vous vous rendiez compte qu'une tâche impérative soit en fait utile ou superflue. N'hésitez pas à la changer de colonne pour gagner en sérénité.

Rappelez-vous que seule la colonne « Impératif » est vraiment importante.

Passons maintenant à la seconde partie de ce chapitre dans laquelle nous allons découvrir les manières de relooker un thème WordPress.

Vous verrez qu'il ne faut pas toujours foncer dans le code pour effectuer vos modifications.

## Chapitre 1.2

# **Ne mettez pas les mains dans le cambouis (enfin pas tout de suite)**

Je me rappellerai toujours de mon professeur de mathématiques de troisième. Elle s'appelait M<sup>me</sup> Robbe. C'était une femme toute menue mais d'une énergie étonnante.

Elle avait les calculatrices en horreur car elle pensait (à juste titre) que cela nous empêchait de réfléchir.

Si je vous parle d'elle, c'est parce que je me souviens d'une de ses phrases favorites :

*Un bon matheux, c'est un bon feignant.*

Elle voulait dire par là qu'il ne fallait pas chercher midi à quatorze heures lorsque l'on devait résoudre un problème. Il fallait aller au plus simple.

Une fois que cela est réglé, on passe à la suite et on n'en parle plus.

Et bien figurez-vous que l'on peut procéder de la même manière pour la personnalisation d'un thème WordPress !

Cela ne sert à rien de vouloir modifier le code alors qu'il est possible d'aller dans les options pour procéder aux changements désirés.

C'est à mon sens le conseil le moins technique de ce guide mais aussi celui qui est le plus important :

*Allez au plus simple, toujours.*

J'imagine que vous avez autre chose à faire que de passer des heures à personnaliser votre thème WordPress.

Dans ce chapitre, nous allons voir comment procéder pour relooker votre thème de la meilleure façon qui soit, en fonction de ce que vous désirez accomplir.

Commençons sans plus attendre.

Tout d'abord, vous devez...

## **Utiliser les options disponibles**

Bien que je ne connaisse pas le thème que vous utilisez sur votre site, il propose certainement plusieurs options vous permettant d'en modifier l'apparence.

Cela peut paraître bête mais si vous pouvez changer une couleur, un logo ou une image d'arrière-plan sans aller dans le code, il faut le faire.

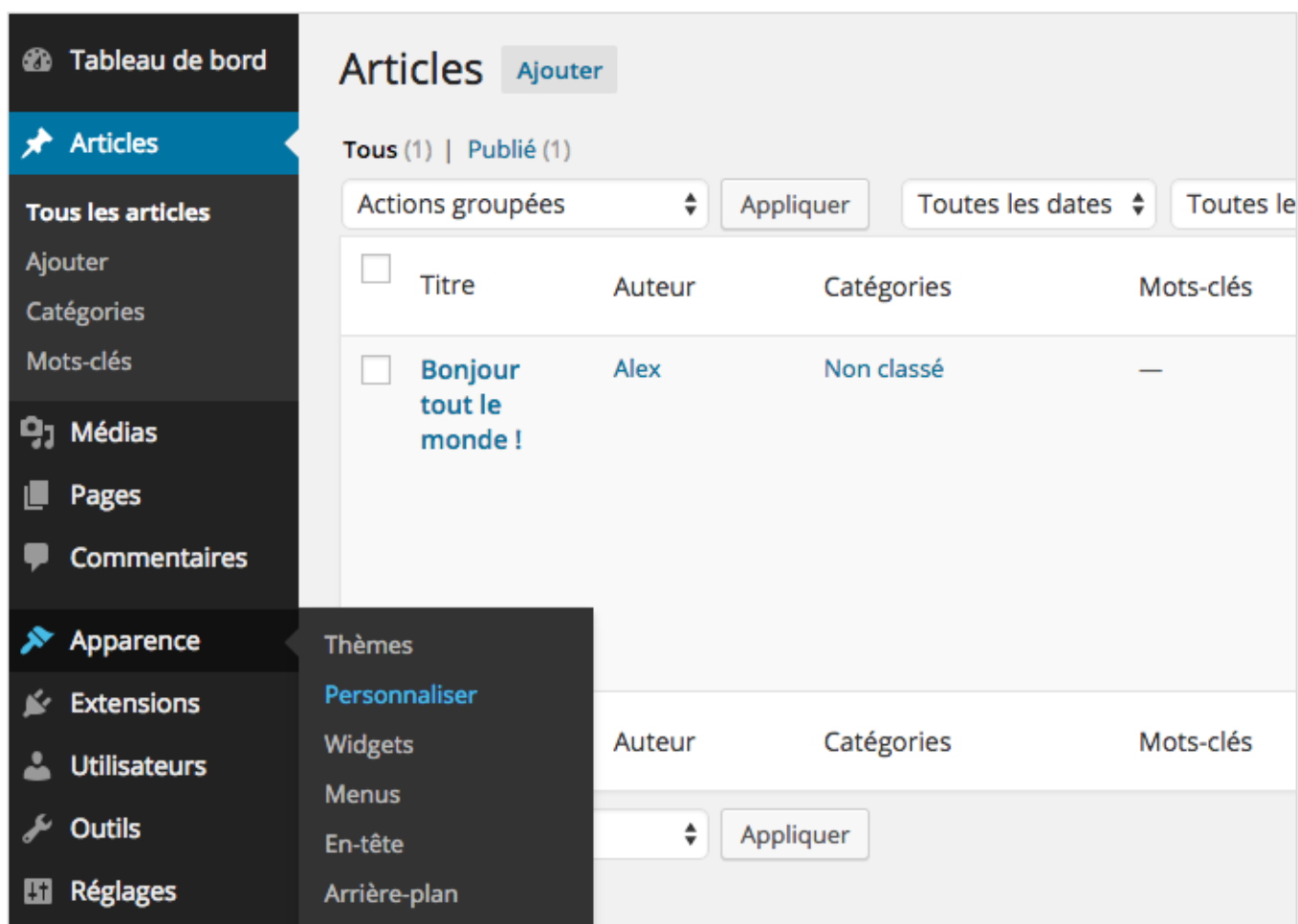
On revient exactement à ce dont je vous parlais au sujet des bons matheux.

Dans la plupart des thèmes WordPress, il y a deux possibilités pour modifier les options disponibles.

### **1. Le menu Personnaliser**

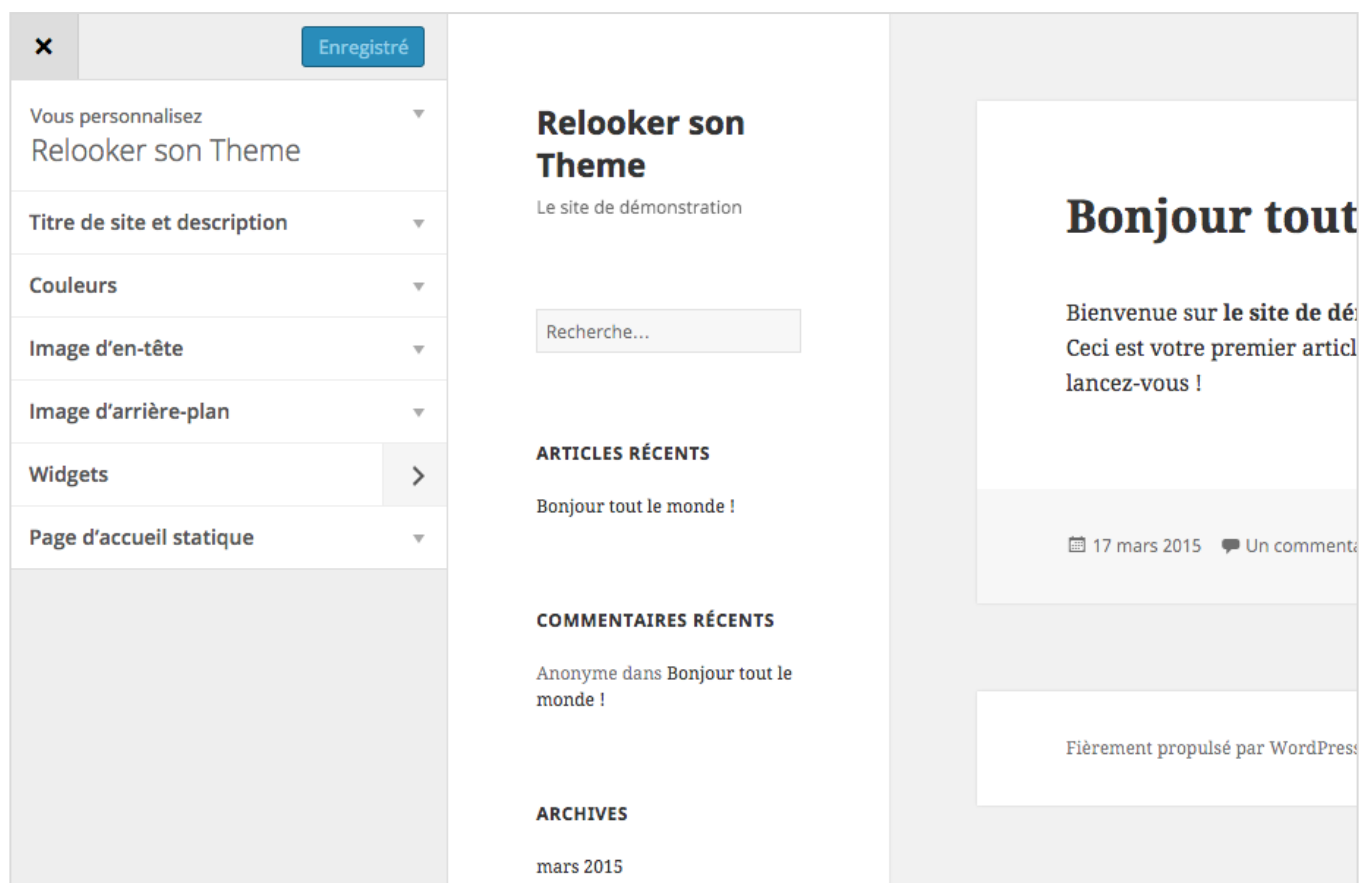
Derrière ce mot barbare se cache un outil bien pratique qui permet de visualiser immédiatement vos personnalisations.

Il faut savoir que tous les thèmes n'utilisent pas pleinement cet outil. Pour savoir si c'est le cas du vôtre, il faut aller dans *Apparence > Personnaliser*.



Si vous constatez qu’il est possible d’agir sur les couleurs, le logo ou les polices d’écriture, votre thème utilise bien le menu Personnaliser.

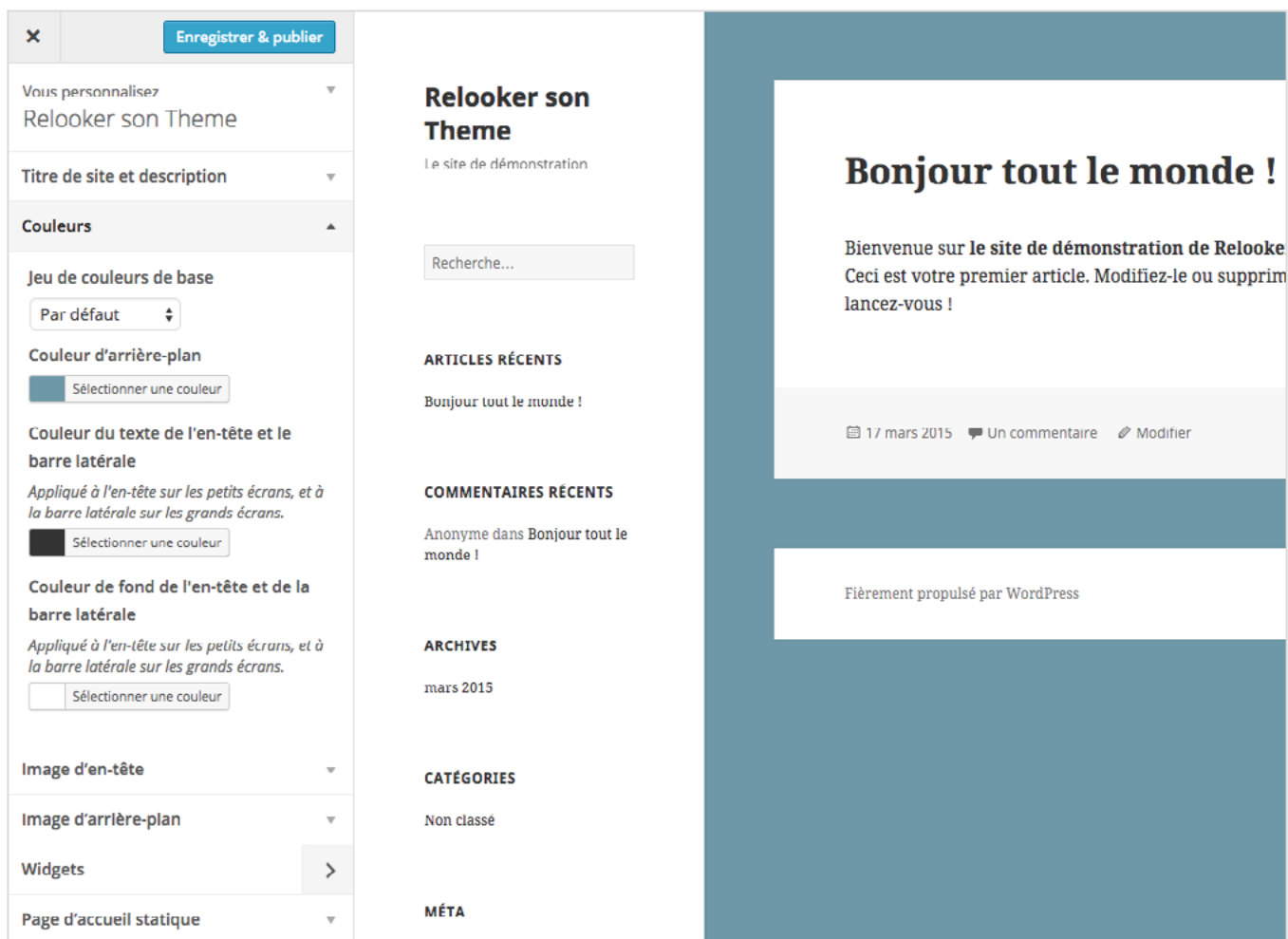
À titre d’exemple, regardons le thème par défaut de WordPress :  
Twenty Fifteen.



En plus des deux options citées auparavant, on trouve :

- Couleurs
- Image d'en-tête
- Image d'arrière-plan
- Widgets

En cliquant sur *Couleurs*, plusieurs options vous seront proposées pour modifier les couleurs utilisées par le thème.



Dans le cas du thème par défaut, il est possible de définir :

- Un jeu de couleurs (habillage prédéfini)
- La couleur d'arrière-plan
- La couleur de texte de l'en-tête et de la barre latérale
- La couleur de fond de l'en-tête et de la barre latérale

*Bien entendu, les options disponibles dans le menu Personnaliser varient en fonction des thèmes. Certains en proposent plus que d'autres.*

En général, le menu Personnaliser est assez simple à prendre en main. Cliquez sur l'élément que vous désirez modifier et laissez-vous guider.

On note aussi la présence d'un élément *Widgets* au sein du menu Personnaliser.

Cela vous permettra d'ajouter, gérer et supprimer des widgets comme il est possible de le faire en allant dans *Apparence > Widgets*.

La seule différence est que vous verrez les changements s'afficher dans l'aperçu de droite.

## 2. Les pages d'options du thème

Si le menu Personnaliser de votre thème est peu fourni, il se peut que l'auteur du thème ait préféré créer une page d'options comme il en existe dans les extensions.

Cette page d'options peut se trouver dans le menu *Apparence* ou directement dans le menu principal.

Si vous ne la trouvez pas, référez-vous à la documentation de votre thème.

Selon les thèmes WordPress, la page d'options sera plus ou moins garnie. Tout dépend de ce que le développeur du thème veut vous permettre de modifier.

*Attention : Ce n'est pas parce qu'un thème propose peu d'options qu'il s'agit d'un mauvais thème. Dans la plupart des cas, il s'agit souvent de l'inverse.*

*Le faible nombre d'options permet de faire en sorte que le thème ne soit pas trop dénaturé et conserve fière allure. Il faut en effet éviter **le syndrome du sapin de Noël** à tout prix.*

Une fois que vous aurez exploré les options proposées par votre thème, que ce soit par le menu Personnaliser ou la page d'options, il y a des chances pour que vous restiez sur votre faim.

Si ce que vous voulez modifier ne s’y trouve pas, vous allez devoir ouvrir le capot et mettre les mains dans le cambouis code.

## **Modifier le code de votre thème WordPress**

Si vous vous êtes procuré ce guide, c’est que vous désirez aller plus loin que les options proposées nativement par votre thème.

Il ne faut cependant pas foncer tête baissée car vous risquez de faire plus de mal que de bien. Nous avons vu cela dans la première partie de ce chapitre.

Normalement, vous devriez avoir votre liste TOP à proximité. **Vous savez donc exactement ce que vous devez faire.**

*Vos modifications doivent être apportées avec méthode et rigueur.*

Par exemple, nous verrons dans la suite de ce guide qu’il ne faut pas modifier directement le code de votre thème WordPress car cela comporte des risques.

Avant de nous attarder sur la technique, nous devons nous munir des bons outils.

Tout bon artisan possède des outils performants, il doit en être de même pour l’apprenti développeur que vous êtes.

## **2 outils indispensables pour commencer**

*C’est en forgeant que l’on devient forgeron.*

Il n’y a pas plus vrai que ce proverbe. On n’apprend pas à marcher du premier coup. Idem pour le vélo, le patin à glace, la guitare, la natation... Bref, vous voyez ce que je veux dire.

La programmation n'échappe pas à la règle.

Même si Relooker son Thème n'a pas pour objectif de vous transformer en « une bête du code », vous allez devoir vous entraîner un minimum avant de pouvoir plonger dans le grand bain et personnaliser votre thème.

*En parlant de plongeon, figurez-vous que je n'ai appris à plonger qu'en août 2012. Pourquoi ? Et bien, tout simplement parce que j'avais peur. En voyant mon cousin de 10 ans plonger dans une rivière avec 4 mètres de profondeur, je me suis dit qu'il y avait un problème...*

*Ni une ni deux, je me suis mis à m'entraîner (avec plus ou moins de succès d'ailleurs, les plats étaient au rendez-vous...). J'ai poursuivi mon entraînement à la piscine jusqu'à parvenir à faire des plongeurs corrects.*

*Je n'ai rien lâché, il était hors de question d'arrêter avant d'y arriver.*

La morale de cette histoire est qu'il faut s'entraîner un minimum avant de pouvoir obtenir des résultats satisfaisant.

Vous allez devoir faire de même avec le code. Rassurez-vous, vous n'aurez pas de plaques rouges sur la poitrine ; vous devrez juste faire les exercices associés à Relooker son Thème.

Pour débiter votre entraînement, le premier outil dont vous allez avoir besoin est...

## **L'éditeur de code**

Cet outil va vous permettre d'ouvrir et de modifier les fichiers de n'importe quel thème WordPress.

En fait, il s'agit une sorte d'éditeur de texte (comme le Bloc-notes) qui possède des fonctions supplémentaires pour que vous puissiez programmer plus rapidement.

Il en existe des dizaines, que ce soit sur Windows ou sur Mac. Certains sont payants et d'autres gratuits.

Pour débiter, je vous conseille d'utiliser l'éditeur **Brackets**. Celui-ci a l'avantage d'être gratuit, en français et disponible sur Windows et sur Mac.

Brackets est d'ailleurs l'éditeur que nous allons utiliser dans les exercices accompagnant ce guide.

Dans le dossier *Ressources* fourni avec Relooker son Thème, vous trouverez le tutoriel d'installation de Brackets ainsi qu'une présentation afin de bien le prendre en main (fiche n°1).

Note : Si vous utilisez déjà un éditeur de code, vous pouvez le conserver. Je conseille Brackets aux personnes qui partent de zéro.

Intéressons-nous à un autre outil indispensable pour travailler plus rapidement, il s'agit de...

## **L'inspecteur de code**

Une fois que vous disposerez d'un éditeur de code, je vous recommande de vous munir d'un inspecteur de code pour éviter de perdre du temps.

Je m'explique.

Jusque-là vous disposez de votre site et d'un éditeur de code. Lorsque vous modifierez un fichier, vous l'enregistrerez puis vous actualiserez la page dans le navigateur pour observer les résultats.

Logique me direz-vous.

Imaginez maintenant que la modification apportée n'ait pas l'effet escompté. Vous allez devoir modifier le fichier à nouveau, l'enregistrer et actualiser votre page pour vérifier si cela a bien fonctionné. Est-ce que c'est bon cette fois ?

Non ?! Eh bien rebelote ...

À la longue, cela est très pénible.

Heureusement, un inspecteur de code vous aidera à **modifier les pages web en direct**.

Un inspecteur de code fonctionne de la façon suivante :

- Vous lancez l'outil
- Vous modifiez le code de la page
- Les modifications se répercutent devant vos yeux

Une fois que vous aurez trouvé la bonne modification, il ne vous restera plus qu'à la reporter dans les fichiers de votre thème WordPress.

Attention : L'inspecteur de code ne sauvegarde pas les modifications qu'il apporte à une page web. Vous les perdrez dès que la page sera actualisée.

Pour résumer, **un inspecteur de code est un moyen de tâtonner** afin de trouver le code qui produit l'effet escompté.

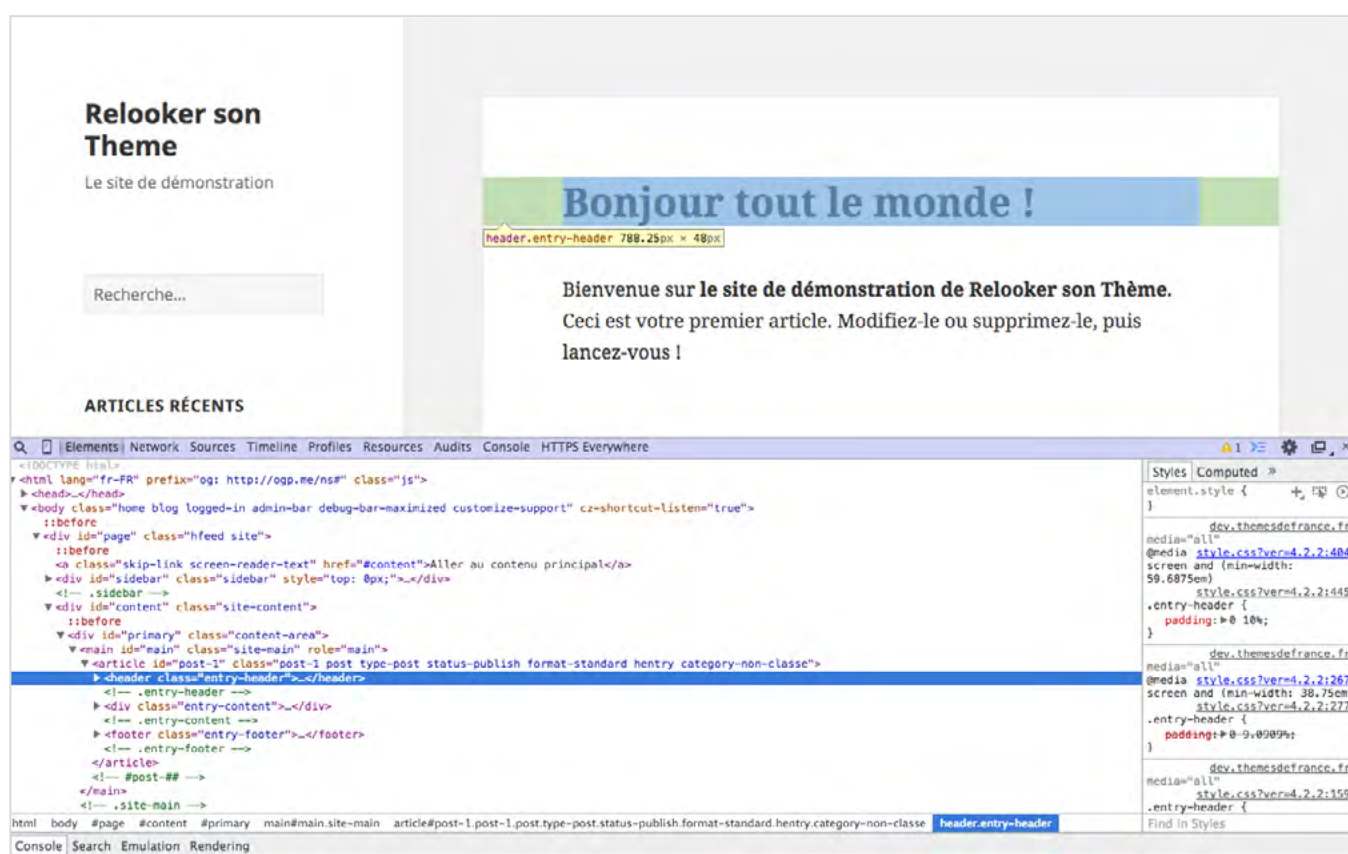
Aujourd'hui, tous les bons navigateurs embarquent un inspecteur de code. Vous n'avez donc pas besoin d'en installer un.

Voici comment le lancer pour chaque navigateur :

- **Google Chrome** : Clic droit sur une page web puis *Procéder à l'inspection de l'élément*.
- **Mozilla Firefox** : Clic droit sur une page web puis *Examiner l'élément*.

- **Safari** : Allez dans *Préférences*, cliquez sur l'onglet *Avancées* puis cochez la case *Afficher le menu Développement* dans la barre des menus. Pour afficher l'inspecteur de code, faites un clic droit sur une page web et sélectionnez *Inspecter l'élément*.
- **Internet Explorer** : Pressez la touche F12 pour lancer l'inspecteur de code sur une page web.

Concrètement, l'inspecteur de code se greffe en bas de votre navigateur comme sur l'image ci-dessous :



*Inspecteur de code du navigateur Chrome.*

## Récapitulons

À présent, vous savez que modifier le code de votre thème WordPress doit être la dernière chose à faire pour personnaliser votre thème.

Vous devez toujours vérifier s'il est possible d'effectuer une modification avec les options proposées par le thème sur lequel vous travaillez.

Vous savez également quels outils employer pour éditer le code de n'importe quel thème WordPress. En deux mots, vous êtes prêt à vous lancer.

Toutefois, avant de vous plonger dans le code, vous devez apprendre à le lire et à le comprendre. C'est ce que nous allons étudier dans les trois prochains chapitres de ce guide.

Avant cela, je veux vous poser une question.

Savez-vous comment fonctionne un site internet ?

Si vous avez des doutes, lisez la dernière partie de ce chapitre.

## Chapitre 1.3

# Comment fonctionne un site internet

Si vous lisez ceci, vous avez probablement quelques doutes ou interrogations sur le fonctionnement des sites internet.

Des millions de personnes les consultent tous les jours sans savoir comment ils apparaissent sur leurs écrans.

C'est un peu comme utiliser une voiture sans comprendre comment elle fonctionne.

Tant que vous êtes un utilisateur lambda, c'est très bien. En revanche quand vous désirez réparer ou améliorer quelque chose, mieux vaut savoir ce que l'on fait.

C'est exactement la même chose pour un site web.

Sans ces notions de base, vous n'allez pas pouvoir faire le lien entre les chapitres, ni appliquer correctement le contenu de ce guide.

*Il est capital de comprendre comment s'affiche un site internet sur votre écran, d'où viennent les pages web et de quoi elles se composent.*

Étant donné que vous lisez ce guide, vous devez certainement connaître le blog **WP Marmite**. Lorsque vous le consultez, vous utilisez un logiciel que l'on appelle un *navigateur web*.

Il y a de fortes chances que vous utilisiez un de ces cinq navigateurs :

- **Google Chrome**
- **Mozilla Firefox**
- **Safari d'Apple**
- **Opera**
- **Internet Explorer de Microsoft**

Le rôle d'un navigateur web est d'interpréter le code informatique avec lequel sont conçus les sites internet afin de les afficher à l'écran.

Ce code informatique est principalement constitué de ces deux langages :

- HTML
- CSS

Un site peut utiliser d'autres langages informatiques comme le Javascript ou le Flash mais nous n'aborderons pas cela dans ce guide.

Retenez simplement qu'un navigateur interprète le code HTML et CSS d'un site pour l'afficher.

Étudions ces langages d'un peu plus près...

## **HTML**

Le HTML est un langage permettant de représenter la structure et le contenu des pages web.

C'est en quelque sorte l'ossature d'un site. Ce sur quoi tout le reste va reposer.

Si on reprend l'exemple de WP Marmite, le texte des articles est écrit en HTML, tout comme les titres, les menus, etc.

# CSS

Le CSS est un langage utilisé pour gérer l'apparence des pages web.

Avec le CSS vous allez pouvoir modifier les couleurs de votre site, la position des éléments, les polices d'écriture, les espacements, etc.

Le CSS permet d'habiller la structure HTML pour la rendre plus agréable visuellement.

## Un exemple concret

Afin de vous donner une idée de ce à quoi peut ressembler un site dépourvu de CSS, voici à quoi ressemble WP Marmite avec CSS :



Vous reconnaissez sûrement le blog que vous avez l'habitude de visiter (enfin j'espère :)).

On retrouve l'en-tête qui contient deux menus, le logo, l'encart d'inscription à la newsletter et les derniers articles.

En désactivant le CSS, voici ce que cela donne :



- [À propos](#)
- [Blog](#)
- [Podcast](#)
- [Cuisitors](#)
- [Contact](#)

[Menu](#)

- [Accueil](#)
- [Tutoriels WordPress](#)
- [Thèmes](#)
- [Plugins](#)
- [Études de Cas](#)
- [Ressources](#)



## Créez votre site avec WordPress dès aujourd'hui

Comme plus de 10000 personnes, découvrez comment :

- Trouver le thème WordPress idéal
- Installer WordPress étape par étape
- Traduire votre thème en français

Entrez votre email ci-dessous et cliquez sur "Recevoir les infos" :

- [Dégustez les dernières recettes](#)



## Tutoriel WordPress SEO

[WordPress SEO : Le plugin indispensable à toute installation WordPress](#)

Avec plus de 17 millions de téléchargements, WordPress SEO est un des 10 plugins les plus téléchargés de tous les temps. Si autant de sites lui...

[Lire la suite](#)

Le moins que l'on puisse dire est que WP Marmite prend un tout autre visage. En effet, seul le code HTML est interprété par le navigateur.

### **Sans CSS, on perd toute la mise en forme !**

Plus de couleurs d'arrière-plan, ni de menus placés à droite de l'en-tête ni de joli encart de newsletter.

Les contenus de la page sont justes empilés comme sur un vulgaire tas de bois.

Autant vous dire que le blog n'aurait pas le même succès s'il ressemblait à ça !

Maintenant que vous savez à quoi servent le HTML et le CSS, vous vous posez probablement cette question...

## **Pourquoi utiliser deux langages ?**

C'est vrai après tout. On pourrait être tenté de croire qu'utiliser deux langages complique les choses. Après tout, qui dit deux langages dit deux langages à apprendre.

Il y a certes deux langages à apprendre mais contrairement aux idées reçues, il est plus simple de procéder ainsi.

Comme nous venons de le voir, le HTML et le CSS ont une utilité différente. L'un sert à structurer un site et l'autre à le mettre en forme.

Cette séparation a été faite pour simplifier la vie des créateurs de sites internet.

*Auparavant, le style des pages était inclus dans chaque fichier HTML. Imaginez-vous avoir un site de plusieurs milliers de pages.*

*Si pour une raison quelconque vous deviez modifier la taille de la police des textes, vous n'auriez pas eu d'autre choix que d'éditer chaque page afin de répercuter cette modification sur tout le site !*

Avec le CSS, une simple modification suffit pour changer l'apparence d'un élément instantanément sur tout un site.

C'est donc un gain de temps incroyable !

Vous allez certainement me dire :

*Ok Alex, le HTML et le CSS travaillent en équipe pour afficher les pages web mais d'où vient le code qui est interprété par mon navigateur ?*

Effectivement, le code HTML et CSS des sites web vient bien de quelque part. C'est ce que nous allons voir en étudiant...

## **L'architecture client/serveur**

Lorsque votre site est en ligne, les fichiers qui le composent ne sont pas situés sur votre ordinateur mais chez un hébergeur.

Cela permet à votre site d'être accessible en permanence (si votre site était hébergé sur votre ordinateur, vous devriez le laisser allumé 24h/24).

L'espace de stockage fourni par votre hébergeur va vous servir à placer les fichiers qui composent votre site internet. **On dit qu'ils sont situés sur un serveur.**

Nous voilà déjà avec le premier élément de notre couple client/serveur.

Mais qui est le client ?

Eh bien c'est vous, moi ou n'importe quel autre visiteur.

Plus précisément, **le client est l'ordinateur qui va demander à afficher votre site.**

Pour mieux comprendre, je vous propose une petite mise en situation :

Imaginons que vous désiriez lire un blog, prenons-en un au hasard :  
WP Marmite !

Lorsque vous tapez "wpmarmite.com" dans votre navigateur web, la page d'accueil de WP Marmite va être demandée à mon serveur.

Techniquement, **un client (vous) vient d'effectuer une requête à mon serveur.**

Une fois que le serveur aura reçu votre requête, il va effectuer plusieurs opérations pour générer une page web en HTML et CSS qu'il enverra ensuite au client.

Le navigateur situé sur l'ordinateur client va interpréter le code de cette page web pour l'afficher convenablement et vous permettre de parcourir le site.

*Si on regarde comment cela fonctionnent de plus près, on s'aperçoit que c'est un peu plus complexe mais, pour le moment, retenez que lorsque vous demandez une page à un serveur celui-ci vous l'envoie en HTML/CSS pour que votre navigateur l'affiche convenablement.*

## **Et WordPress dans tout ça ?**

J'ai volontairement omis de parler de WordPress pour rester dans les fondamentaux.

Comme cela a été mentionné auparavant, les choses sont un peu plus complexes que la façon dont je les ai décrites.

Vous avez certainement hâte de commencer à relooker votre thème mais n'allons pas trop vite.

Il est important de maîtriser la théorie ainsi que la pratique de base avant d'aller plus loin. Nous étudierons la manière dont un site sous WordPress fonctionne dans le chapitre 4.

Pour le moment, attardons-nous sur le langage qui structure les sites internet : le langage HTML.

**RELOOKER SON THÈME**

# **CHAPITRE 2**

## **LE HTML, LA STRUCTURE DES PAGES WEB**

---

**Créez une page web basique  
et découvrez les balises HTML  
de base structurant  
n'importe quel site internet**

## Chapitre 2.1

# La syntaxe du HTML

Comme nous venons de le voir, le HTML est un des deux langages informatiques utilisés par les navigateurs pour afficher une page web.

Et comme tout langage, le HTML possède une syntaxe qui lui est propre. Regardons de plus près comment le HTML se rédige.

## Le HTML, un langage balisé

Pour reprendre ce qui est écrit dans le titre, le HTML est ce que l'on appelle un langage balisé.

Balisé veut dire que ce langage est composé de balises.

**Les balises sont un moyen d'indiquer au navigateur web la nature des éléments à afficher.** Par exemple, un paragraphe de texte possède une balise spécifique, tout comme le titre d'une page ou encore une image.

*À chaque type d'élément correspond une balise.*

Pour être plus concret, regardez par exemple le code correspondant à un paragraphe de texte en HTML :

```
<p>Texte de mon paragraphe</p>
```

Ceci est un élément HTML de type paragraphe. Il est constitué d'une balise d'ouverture de paragraphe (<p>), du contenu de l'élément (Texte de mon paragraphe) et d'une balise de fermeture de paragraphe (</p>).

On dit que le texte du paragraphe est *balisé* car il est situé entre les deux balises.

Dans la plupart des cas, les éléments HTML fonctionnent de cette façon :

Balise d'ouverture + contenu + balise de fermeture

Soit en HTML :

```
<balise>Contenu</balise>
```

Attention : Les balises ne peuvent pas se combiner n'importe comment. Une balise ouverte doit toujours être fermée.

Par exemple, ceci est correct :

```
<balise1>
  <balise2></balise2>
</balise1>
```

Alors que cela est faux :

```
<balise1>
  <balise2></balise1>
</balise2>
```

Dans le contre-exemple, la balise fermante `</balise1>` se retrouve à l'intérieur de `balise2` alors que la balise ouvrante `<balise1>` ne s'y trouve pas.

*Les balises ne doivent pas s'enchevêtrer.*

On peut corriger le contre-exemple avec le premier exemple ou de la manière suivante :

```
<balise2>
  <balise1></balise1>
</balise2>
```

Retenez bien que les balises doivent être fermées au bon endroit sinon votre page web ne s'affichera pas correctement.

## Il y a cependant des exceptions

Vous avez probablement remarqué que « dans la plupart des cas » a été mentionné avant de partager la syntaxe d'une balise en HTML.

Il faut savoir qu'il existe des balises qui ne possèdent pas de balises fermantes. Vous allez me dire : Mais pourquoi une telle bizarrerie ?

La réponse est simple, c'est parce que la balise fermante est implicite. Autrement dit, il n'y a pas besoin de l'écrire.

À titre d'exemple, on peut citer la balise `<br>` qui sert à faire des sauts de ligne.

Étant donné que cette balise n'encadre aucun contenu, il n'y a aucune raison d'ajouter une balise fermante. On se retrouverait avec quelque chose du genre :

```
<br></br>
```

Il faut donc l'écrire de la manière suivante :

```
<br>
```

C'est tout de même plus simple n'est-ce pas ?

Note : Ne vous inquiétez pas si vous ne retenez pas les balises présentées pour le moment, nous allons les étudier dans la suite de ce chapitre. L'important est que vous compreniez le fonctionnement des balises.

## N'oublions pas les attributs

En HTML, chaque élément peut être doté d'un ou plusieurs attributs.

Un attribut permet d'apporter de l'information supplémentaire à l'élément auquel il est affecté.

Voici un élément HTML (fictif) doté d'un attribut :

```
<balise attribut="valeur">Contenu de l'élément</balise>
```

Comme vous pouvez le voir, **l'attribut se situe toujours dans la balise ouvrante** et jamais dans la balise fermante (cette fois il n'y a pas d'exception).

Un attribut doit posséder la syntaxe suivante :

```
Nom de l'attribut + signe égal (=) + guillemet (") +  
Valeur de l'attribut + guillemet (")
```

Les éléments sans balises fermantes peuvent également posséder des attributs :

```
<balise attribut="valeur">
```

# Récapitulons

Nous venons d'étudier les concepts de base du HTML. C'est étonnant mais tous les sites web reposent sur la combinaison de balises HTML.

Les pages web combinent juste un grand nombre de ces balises, ce qui peut présenter une certaine complexité.

Dans la partie suivante, nous allons voir comment organiser les bonnes balises afin de créer une page web basique.

## Chapitre 2.2

# La structure des pages web

Dans la partie précédente nous avons vu que le HTML était un langage balisé. À présent nous allons découvrir comment les pages web sont structurées.

Pour être franc, vous n'aurez pas besoin de faire cela lorsque vous modifierez un thème WordPress MAIS il est important de savoir comment une page web est construite pour ne pas faire n'importe quoi.

Voici le code HTML de base pour créer une page web basique en HTML :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Titre de la page</title>
  </head>
  <body>
    <p>Bonjour</p>
    <!-- Ceci est un commentaire -->
  </body>
</html>
```

Regardons ce code de plus près...

# doctype

```
<!doctype html>
```

La balise doctype est un peu spéciale car elle débute par un point d'exclamation. Cette balise sert à dire au navigateur que ce qu'il est en train de lire est un document de type HTML (doctype signifie « document type » soit « type de document » en français).

Grâce à cette balise, le navigateur pourra mieux interpréter le contenu présent au sein de la page.

Dans le cas du code ci-dessus, il s'agit de la balise doctype d'un document en HTML 5 (la dernière version du langage HTML).

Chaque version possède une balise doctype qui lui est propre. Ne soyez pas étonné si vous voyez d'autres balises doctype dans votre thème.

Par exemple, voici la balise doctype d'une page web en HTML 4 :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
    "http://www.w3.org/TR/html4/strict.dtd">
```

La balise doctype du HTML 5 est plus séduisante, n'est-ce pas ?

## html

```
<html lang="fr">  
  
</html>
```

La balise html est la balise de base d'un document HTML. **Rien ne peut se trouver à l'extérieur de cette balise sauf la balise doctype.**

Notez que la balise `html` comporte un attribut `lang` qui va indiquer au navigateur la langue du contenu de la page web.

Notre page est en français donc nous lui attribuons la valeur `fr`. Une page en anglais aurait la valeur `en` pour « english ».

## head

```
<head>
```

```
</head>
```

La balise `head` sert à inclure des informations concernant la page **qui ne seront pas affichées à l'écran** par le navigateur.

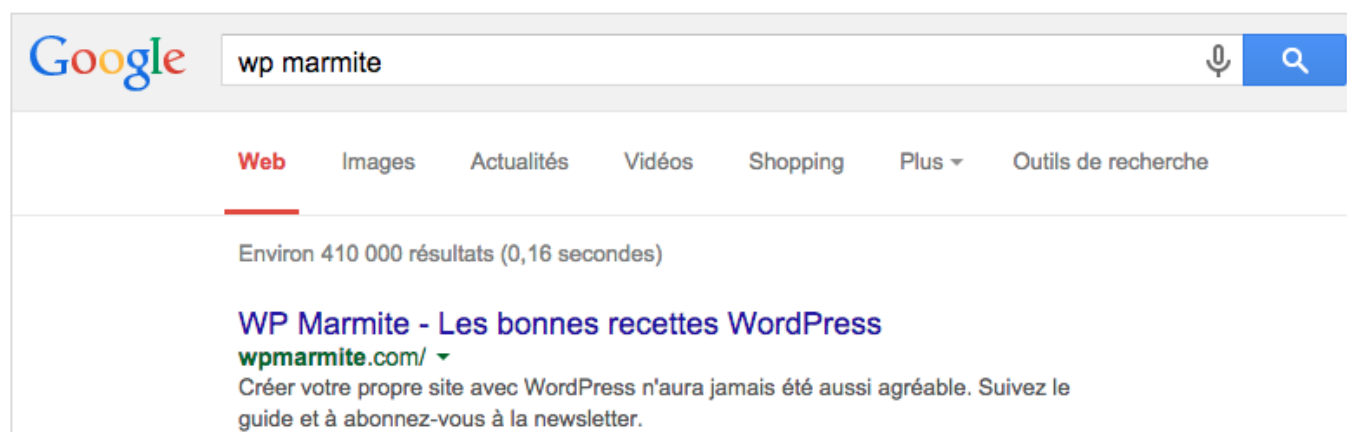
On appelle ces informations des métadonnées.

Vous avez par exemple le titre de la page web :

```
<title>Titre de la page</title>
```

Le titre sera utilisé par les moteurs de recherche dans les pages de résultats. Il est donc capital que chaque page possède une balise `title`.

Vous pouvez voir comment Google utilise le titre de la page d'accueil de WP Marmite :



L'autre métadonnée présente dans la balise head est la balise meta charset :

```
<meta charset="utf-8">
```

Cette balise (auto-fermante) permet de définir l'encodage des caractères de la page web.

L'attribut charset doit avoir la valeur utf-8 afin que les caractères accentués soient affichés convenablement.

Nous aborderons d'autres métadonnées à insérer dans la balise head dans la suite de ce chapitre.

## **body**

```
<body>  
  
</body>
```

À l'opposé de la balise head, la balise body embarque tout ce qui sera visible sur votre page web, c'est à dire le contenu.

Dans l'exemple présenté en haut de cette page, une balise de paragraphe (que vous connaissez déjà) est incluse avec le texte « Bonjour ».

```
<p>Bonjour</p>
```

Du texte figure également entre des balises spéciales :

```
<!-- Ceci est un commentaire -->
```

Comme vous pouvez le lire, il s'agit d'un commentaire. C'est à dire que le navigateur va ignorer tout ce qui sera écrit entre les balises de type commentaire.

Les commentaires sont utiles lorsque vous voulez laisser des indications sur ce que vous avez fait dans votre code.

Attention : Même si les commentaires ne sont pas visibles lorsque l'on consulte une page web, ils le sont lorsque l'on affiche le code source de la page. Ne mettez donc pas des mots de passe ou d'autres informations confidentielles en commentaire.

## Récapitulons

Toutes les pages web possèdent la même structure. Elles sont composées d'un ensemble de balises agencées d'une certaine manière.

On peut retenir la présence de deux sections pour tout document HTML, à savoir une section HEAD (qui regroupe des métadonnées) et une section BODY (qui comprend le contenu).

Avant de passer à la suite du guide, je vous propose de passer à la pratique avec le premier exercice de ce chapitre consacré au HTML. Pour cela, rendez-vous dans le dossier *Exercices > Chapitre 2 - HTML* et ouvrez le fichier PDF de l'exercice 1.

## Chapitre 2.3

# Les balises HTML de la section « head »

Logiquement, vous venez de créer votre première page web. Le résultat est pour le moins minimaliste mais vous êtes dans la bonne direction.

Vous devez vous rappeler qu'un document HTML est composé de deux éléments principaux : la section head et la section body (l'en-tête et le corps de la page).

Dans cette partie nous allons voir les balises de la section head. Si vous vous souvenez bien ce que nous avons vu précédemment, les visiteurs ne peuvent pas voir les informations contenues dans l'en-tête sur la page web.

Cependant sans les informations de la section head, le navigateur serait incapable d'afficher correctement votre page web.

Étudions de plus près les balises HTML de la section head.

## La balise title

Commençons avec une des balises que vous connaissez déjà. Il s'agit de la balise `title`.

Nous avons vu qu'elle servait à indiquer le titre de la page web au navigateur (au niveau de l'onglet) ainsi qu'aux moteurs de recherche.

Voici la syntaxe de cette balise :

```
<title>Titre de ma page</title>
```

Il faut impérativement que chaque page comporte une balise `title`. Sans cette balise, votre site ne sera pas optimisé pour le référencement.

Note : Dans le cas de WordPress, la balise `title` est insérée par le thème ou par une extension de référencement.

## Les balises meta

Nous avons déjà croisé une balise `meta` précédemment. Vous verrez qu'il existe d'autres balises `meta` possédant des fonctions bien spécifiques.

### La balise meta charset

```
<meta charset="utf-8" >
```

Comme nous avons pu le voir, la balise `meta charset` sert à définir la norme de codage d'une page web.

Elle possède un attribut :

- **charset** : norme de codage de la page dont la valeur par défaut est UTF-8

Sur certains thèmes WordPress la balise `meta charset` peut prendre cette forme :

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" >
```

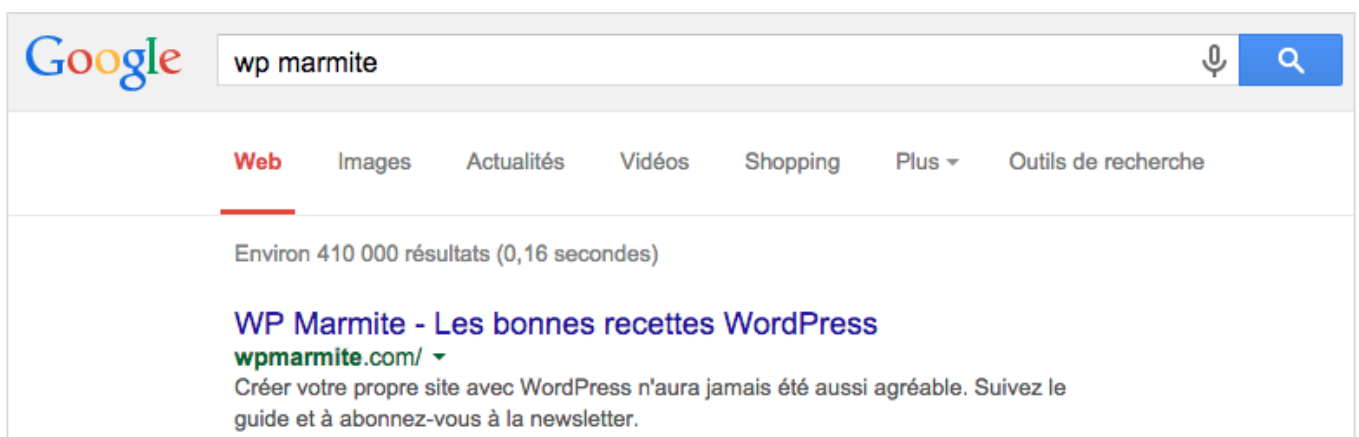
Il s'agit de la syntaxe utilisée dans les versions antérieures au HTML 5. J'ai préféré vous en parler afin que vous ne trouviez pas cela étrange au cas où vous tombiez dessus.

## La balise meta description

```
<meta name="description" content="Description de ma page web pour les moteurs de recherche" >
```

La balise meta description est une balise très importante. C'est de cette balise que les moteurs de recherche vont se servir pour décrire vos pages dans les résultats de recherche.

Voyez plutôt :



Cette balise possède deux attributs :

- **name** : afin de définir le type de la balise (ici il s'agit de description)
- **content** : qui va inclure le contenu de la description de votre page (la limite est de 156 caractères)

Note : Dans WordPress, il est possible de gérer le contenu des balises meta description (et title) grâce à **l'extension WordPress SEO** (cette extension est également très utile dans d'autres aspects du référencement).

## La balise meta robots

```
<meta name="robots" content="index, follow, archive" >
```

La balise meta robots permet d'indiquer aux moteurs de recherche ce qu'ils doivent faire de la page, à savoir l'indexer, suivre les liens présents et mettre la page en cache.

Cette balise possède deux attributs :

- **name** : identification du type de la balise, ici robots
- **content** : définition des paramètres d'indexation (il faut indiquer l'un des deux paramètres pour obtenir le comportement souhaité) :
- *index / noindex* : Indexer ou non la page dans les moteurs de recherche
- *follow / nofollow* : suivre ou ne pas suivre les liens présents sur la page
- *archive / noarchive* : mettre ou ne pas mettre la page en cache des moteurs de recherche
- *none* : ne pas indexer, ni suivre, ni mettre la page en cache. Cela est équivalent à noindex, nofollow, noarchive

## La balise meta viewport

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

La balise meta viewport permet d'optimiser l'affichage d'une page web sur les appareils mobiles. Sans elle, le contenu d'une page web sera mal affiché (textes trop petits et non mis à l'échelle).

Cette balise possède deux attributs :

- **name** : identification du type de la balise, ici viewport

- **content** : définition des paramètres d’affichage sur mobile (les paramètres recommandés par Google sont ceux indiqués dans l’exemple ci-dessus).

## Les autres types de balises meta

Il existe d’autres balises meta pouvant se trouver dans la section head des pages web.

Certaines d’entre elles sont incluses par WordPress ou des extensions afin de transmettre des informations aux moteurs de recherche ou aux réseaux sociaux.

Il faut savoir qu’il existe des balises totalement inutiles.

C’est le cas de la balise meta keywords qui était auparavant utilisée par les moteurs de recherche pour détecter les mots-clés associés à une page (et donc mieux la référencer).

Aujourd’hui, les moteurs de recherche analysent directement le contenu des pages et n’ont donc plus besoin de cette balise. Il est donc inutile voire nuisible de s’en servir.

Passons maintenant à un autre type de balise.

## Les balises link

En parcourant la section head d’un site web, vous remarquerez certainement des balises link. **Ces balises link ont pour but de lier un document à une page web.**

Cette balise est surtout connue car c’est elle qui permet d’associer un fichier CSS à une page web. Rappelez-vous, ce sont les feuilles de style CSS qui permettent de donner une apparence particulière à un site.

Nous allons voir dans les exemples suivants que l'on peut l'employer dans d'autres cas.

## La balise `link` d'association de fichier CSS

```
<link rel="stylesheet" type="text/css" media="screen"
href="http://monsite.com/wp-content/themes/montheme/style.
css" >
```

Pour associer un fichier CSS à une page web, il faut inclure la balise `link` avec certains attributs. Voyons à quoi chacun d'eux correspond :

- **rel** : permet de définir le rôle de la balise `link`. La valeur de l'attribut est `stylesheet` pour *feuille de style*, c'est à dire feuille de style CSS.
- **type** : spécifie le format du fichier lié. Comme il s'agit d'un fichier CSS, on indique `text/css`.
- **media** : indique le type de support sur lequel le fichier CSS doit être utilisé. Cet attribut peut être doté des valeurs suivantes (liste non exhaustive) :
  - *screen* : écrans d'ordinateur
  - *print* : aperçu avant impression et impression
  - *all* : pour tous les types d'appareils
- **href** : adresse web du document lié, ici le fichier CSS.

## La balise `link` d'indication de flux RSS

```
<link rel="alternate" type="application/rss+xml"
title="Flux RSS de mon site" href="http://monsite.com/
feed/" >
```

Les flux RSS sont un moyen de rendre votre contenu accessible à vos lecteurs sans qu'ils ne se rendent directement sur votre site. Ils utilisent ce

que l'on appelle des agrégateurs pour rassembler le contenu de plusieurs sites à un seul endroit.

Il est donc important pour un site d'indiquer à ces agrégateurs comment lister les derniers articles publiés sur votre site. C'est la balise `link` ci-dessus qui va effectuer ce travail.

Étudions les attributs qui la composent :

- **rel** : permet de définir le rôle de la balise `link`. La valeur de l'attribut est `alternate` car cela permet d'accéder au contenu du site de façon alternative.
- **type** : format du fichier lié. Les flux RSS sont structurés au format XML donc on indique `application/rss+xml`.
- **title** : nom de votre flux RSS. Il pourra être utilisé par les agrégateurs de contenus.
- **href** : adresse web du flux RSS.

## La balise `link` d'association de favicon

```
<link rel="icon" href="http://monsite.com/favicon.ico">
```

Un favicon est une petite image utilisée par les navigateurs pour représenter votre site au niveau des onglets. Généralement, il s'agit du logo du site ou d'un pictogramme le représentant.

Si vous cherchez à créer votre propre favicon, vous pouvez vous rendre sur [ce site](#). Le code des balises `link` vous est fourni mais voici à quoi correspondent les attributs :

- **rel** : permet de définir le rôle de la balise `link`. La valeur de l'attribut est `icon` car il s'agit du favicon.
- **href** : adresse web du favicon.

D'autres attributs peuvent également être présents :

- **type** : spécifie le format du favicon. Selon le format de l'image on pourra trouver : `image/png` ou `image/jpeg`.
- **sizes** : indique les dimensions du favicon (exemple : 16x16).

Note : Depuis la version 4.3, WordPress permet d'ajouter un favicon sans extension. Pour cela il suffit de se rendre dans *Réglages > Général*.

## La balise `link` d'indication de l'adresse principale

```
<link rel="canonical" href="http://monsite.com"/>
```

Parfois, il est possible d'accéder à une même page par différentes adresses web (ou URL). Le problème est que les moteurs de recherche voient ces pages comme des pages différentes alors que ce n'est pas le cas.

Cela peut amener à une pénalisation du site pour duplication de contenu.

Pour éviter ce genre de désagrément, il faut préciser sur chaque page l'adresse de la page principale (on appelle cela une url canonique).

Les attributs de ce type de balise `link` sont les suivants :

- **rel** : permet de définir le rôle de la balise `link`. La valeur de l'attribut est `canonical` car on désire indiquer l'adresse principale d'une page (canonique).
- **href** : adresse web de la page principale.

# Les balises script

Le quatrième type de balise pouvant figurer dans la section head d'une page web est la balise `script`. **Elle est employée pour inclure du code Javascript dans les pages web.**

Le Javascript (aussi appelé JS) est un langage web utilisé pour apporter des fonctionnalités supplémentaires aux pages web. Par exemple, les diaporamas s'en servent pour leurs animations.

Nous n'allons pas étudier le Javascript dans ce guide mais vous devez savoir reconnaître les balises qui en ajoutent.

Vous devez savoir qu'il y a deux façons d'inclure du code Javascript :

1. En écrivant directement le code Javascript entre les balises

```
<script type="text/javascript">CODE_JAVASCRIPT</script>
```

2. En indiquant le lien vers un fichier Javascript dans l'attribut `src`

```
<script type="text/javascript" src="LIEN_VERS_FICHER_JS"></script>
```

Contrairement aux balises précédentes, la balise `script` n'est pas auto-fermante et ce, même si elle est utilisée de la seconde manière.

Les attributs de la balise `script` sont les suivants :

- **type** : spécifie le type de code inséré dans la page. Ici nous avons `text/javascript` car il s'agit de code Javascript. Cet attribut est facultatif.
- **src** : adresse du fichier Javascript à inclure (`src` signifie source).

Note : Même si cela est possible, il n'est pas recommandé d'inclure tous ses scripts dans l'en-tête de votre page HTML. Cela peut ralentir le temps de chargement. Pour régler ce problème, placez les balises `script` dans le pied de page du site (footer), juste avant la balise de fermeture `body`. WordPress possède des fonctions pour faire cela correctement.

## Les balises style

```
<style type="text/css">CODE_CSS</style>
```

Le dernier type de balise que vous trouverez dans la section `head` d'une page web est la balise `style`.

Cette balise est similaire à la balise `script` sauf qu'au lieu d'insérer du code Javascript, elle permet d'inclure du code CSS (nous aborderons le CSS dans le chapitre 3 de ce guide).

Il n'est pas recommandé d'utiliser les balises `style` car comme nous l'avons vu, il ne vaut mieux pas intégrer directement du CSS au sein d'une page web.

Toutefois, certains thèmes WordPress l'utilisent pour inclure des styles personnalisés.

Au niveau des attributs, cela reste simple :

- **type** : spécifie le type de code inséré dans la page. Ici nous avons `text/css` car il s'agit de code CSS. Cet attribut est facultatif.

## Récapitulons

Au cours de cette partie, nous avons passé en revue les types de balises pouvant apparaître dans la section `head` des pages web.

Cela devrait vous permettre de mieux appréhender ce qui se trouve dans les en-têtes des pages web.

Rassurez-vous, vous n'avez pas à apprendre tout cela par coeur, cela n'aurait pas de sens.

Servez-vous en de référence si vous souhaitez déchiffrer ce que contient l'en-tête de votre thème WordPress (ou d'un site en général).

Avant de passer aux balises de la section body, reprenez l'exercice que nous avons commencé pour y ajouter quelques balises découvertes dans cette partie avec l'exercice 2.

## Chapitre 2.4

# Les balises HTML de la section « body »

Après avoir étudié ce que pouvait contenir l'en-tête d'une page web, je vous propose à présent de passer au contenu !

Dans cette partie, nous allons examiner les différentes balises qu'il est possible de trouver dans le corps d'un document HTML, c'est à dire dans la section body d'une page web.

Vous avez déjà rencontré quelques-unes de ces balises en parcourant ce guide et probablement davantage dans l'éditeur visuel de WordPress lorsqu'il est en mode texte.

Après avoir parcouru cette partie, vous comprendrez à quoi elles correspondent.

Allez, on attaque sans plus tarder !

## La balise de paragraphe

```
<p>Ceci est un paragraphe</p>
```

Commençons par une balise qui doit vous sembler familière puisque nous l'avons abordée précédemment dans ce chapitre.

La balise de paragraphe est une des balises les plus courantes en HTML. Elle permet de structurer le texte d'une page. Vous éviterez ainsi les gros blocs de texte que personne n'aime lire.

# Les balises de titres

```
<h1>Ceci est un titre</h1>
<h2>Ceci est un sous-titre</h2>
<h3>Ceci est un sous-sous-titre</h3>
<h4>Ceci est un sous-sous-sous-titre</h4>
<h5>Vous voyez où je veux en venir ?</h5>
<h6>Stop ! On ne va pas plus loin.</h6>
```

Pour organiser l'information, les paragraphes ne suffisent pas. Lorsque qu'on rédige quelque chose, il faut un titre et des sous-titres pour hiérarchiser les différentes parties (voire d'autres sous-titres pour organiser les sous-parties).

Eh bien, il est possible de faire cela en HTML !

La balise `h1` détermine le titre principal d'une page web. Afin de soigner votre référencement, veillez à ce qu'une seule balise `h1` soit présente par page.

Pour revenir à WordPress, **il ne faut pas inclure de balise `h1` dans le contenu des pages et des articles**. Si votre thème est bien conçu, il doit déjà l'inclure au niveau du titre des pages et des articles.

Attention : Ne confondez pas la balise de titre `h1` avec la balise `title`. La balise `title` sert à afficher le titre d'une page pour les moteurs de recherche (et est située dans la section `head`) alors que la balise `h1` affiche le titre d'une page dans le contenu (dans la section `body`).

Vous devez utiliser les balises de titre par ordre d'importance, c'est à dire `h2`, `h3`, `h4`, `h5` et `h6`. La balise `h7` n'existe pas.

Sachez que **vous n’êtes pas obligé d’utiliser toutes ces balises**. Cela dépend du contenu que vous avez à publier. Veillez juste à les utiliser dans le bon ordre (de 1 à 6 et non l’inverse).

## Les balises de listes

Si vous suivez WP Marmite régulièrement, vous n’avez pas pu passer à côté des articles sous forme de liste.

Créer des listes est un moyen original de présenter des informations aux visiteurs de votre site (il faut savoir qu’il est plus pratique de lire une liste qu’une longue énumération dans un paragraphe).

Le HTML vous permet de créer deux types de listes. Commençons par...

### Les listes désordonnées

```
<ul>
  <li>Un élément de la liste</li>
  <li>Un autre élément de la liste</li>
  <li>Le dernier élément de la liste</li>
</ul>
```

Comme son nom l’indique, une liste désordonnée présente des éléments sans ordre particulier.

Elles se composent toujours de la manière suivante :

- Ouverture de la balise `ul` avec `<ul>`
- Insertion de plusieurs balises `li` ( `<li>` ) contenant le texte de l’élément (sans oublier de refermer la balise avec `</li>`)
- Fermeture de la balise `ul` avec `</ul>`

## Les listes ordonnées

```
<ol>
  <li>Premier élément de la liste</li>
  <li>Deuxième élément de la liste</li>
  <li>Troisième élément de la liste</li>
</ol>
```

À l'inverse de ce que l'on vient de voir, il est possible de créer des listes ordonnées (avec un classement numérique du type « 1, 2, 3... »).

Ces listes se construisent presque comme les listes désordonnées, seule la balise de base change. Pour une liste ordonnée, il faut utiliser la balise `ol` au lieu de la balise `ul`.

1. Ouverture de la balise `ol` avec `<ol>`
2. Insertion de plusieurs balises `li` ( `<li>` ) contenant le texte de l'élément (sans oublier de refermer la balise avec `</li>`)
3. Fermeture de la balise `ol` avec `</ol>`

Avez-vous vu que la liste ci-dessus est une liste ordonnée ?

## La balise de lien

```
<a href="http://wp-marmite.com" title="WP Marmite : Les  
bonnes recettes WordPress" >Lien vers WP Marmite</a>
```

Les liens sont la base du web, c'est par eux que l'on navigue entre les pages. Nous les utilisons tous les jours.

Regardons comment ils se construisent :

- Ouverture de la balise de lien avec `<a`

- Ajout de l'attribut `href` avec une adresse en valeur (j'ai mis l'adresse de WP Marmite dans l'exemple)
- Ajout de l'attribut `title` pour spécifier une description de la page liée (de moins en moins utilisé)
- On referme la balise d'ouverture avec `>`
- Écriture de l'ancre, c'est à dire le texte qui sera lié (dans l'exemple il s'agit de « Lien vers WP Marmite »)
- Fermeture de la balise `a` avec `</a>`

Et hop ! Vous avez un beau lien :)

## Lien vers un nouvel onglet

Je suis certain que vous avez déjà cliqué sur des liens qui se sont ouverts dans de nouveaux onglets.

Pour créer des liens de ce type, tout ce que vous avez à faire est d'inclure un nouvel attribut nommé `target` à la balise de lien (balise `a`) et lui attribuer la valeur `_blank`.

Voici la structure du lien présenté précédemment sauf que celui-ci s'ouvrira dans un nouvel onglet :

```
<a href="http://wp-marmite.com" title="WP Marmite : Les  
bonnes recettes WordPress" target="_blank" >Lien vers  
WP Marmite</a>
```

## Récapitulons

Tout au long de cette partie, nous avons vu quelques balises pouvant figurer dans la section `body` d'une page web. Il en existe d'autres bien entendu mais il faut procéder par étapes.

Pour découvrir ces balises grandeur nature, vous pouvez [cliquer sur ce lien](#).

Avant de découvrir de nouvelles balises, passons à la pratique avec un nouvel exercice. Pour cela reportez-vous à l'exercice 3 du chapitre 2.

## Chapitre 2.5

# Les balises HTML de la section « body » – Partie 2

À présent, intéressons-nous à d'autres balises pouvant figurer dans la section body des pages web.

Étant donné que vous venez de faire l'exercice 3, j'imagine que vous êtes impatient de découvrir de nouvelles balises.

Commençons avec...

## La balise image

```

```

En regardant le code qui la compose, on peut voir que la balise image est auto-fermante contrairement aux autres balises de la section body que nous avons vues jusqu'à présent.

La balise image est composée de plusieurs attributs :

- **src** : adresse du fichier image (jpg, png, gif...)
- **alt** : description de l'image qui s'affichera si l'image ne peut pas l'être (et décrira aussi l'image pour les moteurs de recherche)
- **width** : largeur de l'image en pixels
- **height** : hauteur de l'image en pixels

Indiquer la largeur et la hauteur des images n'est pas obligatoire mais cela est recommandé afin d'optimiser la vitesse d'affichage d'un site.

## Les balises de mise en forme

```
<strong>Ceci est du texte en gras</strong> et <em>cela du  
texte en italique</em>.
```

```
<blockquote>Voilà une belle citation.</blockquote>
```

```
La balise `br` sert à faire<br>des sauts de lignes.
```

```
La balise `hr` à insérer une ligne : <hr>
```

Les balises de mise en forme sont utilisées pour mettre du texte en gras, en italique, insérer une citation, faire un saut de ligne ou encore insérer une ligne horizontale.

Voici les correspondances de ces balises :

- **strong** : mettre du texte en gras
- **em** : mettre du texte en italique
- **blockquote** : insérer une citation
- **br** : insérer un saut de ligne
- **hr** : insérer une ligne horizontale (hr pour « horizontal rule » en anglais)

# Les balises de tableaux

```
<table>
  <tr>
    <td>WooThemes</td>
    <td>60 thèmes</td>
  </tr>
  <tr>
    <td>Elegant Themes</td>
    <td>87 thèmes</td>
  </tr>
  <tr>
    <td>Themeforest</td>
    <td>5376 thèmes</td>
  </tr>
</table>
```

Pour construire un tableau en HTML, vous aurez besoin de combiner plusieurs balises. La balise de base `table` contiendra l'intégralité du tableau.

Vous devrez ensuite insérer autant de balises `tr` que vous désirez de lignes dans votre tableau.

Pour finir, ajoutez des balises `td` à l'intérieur des balises `tr` pour indiquer le nombre de cellules à insérer dans chaque ligne. Le code du tableau ci-dessus comporte 3 lignes et 2 colonnes.

Il arrive parfois d'avoir besoin de mettre en avant l'en-tête des colonnes d'un tableau. Voici comment modifier le tableau précédent pour donner un titre aux colonnes :

```

<table>
  <caption>Comparatif des boutiques</caption>
  <tr>
    <th>Boutique</th>
    <th>Nombre de thèmes</th>
  </tr>
  <tr>
    <td>WooThemes</td>
    <td>60 thèmes</td>
  </tr>
  <tr>
    <td>Elegant Themes</td>
    <td>87 thèmes</td>
  </tr>
  <tr>
    <td>Themeforest</td>
    <td>5376 thèmes</td>
  </tr>
</table>

```

Nous pouvons remarquer que les colonnes qui possèdent un en-tête (au niveau de la première ligne) sont entourées de balises `th` au lieu des balises `td` que l'on utilise pour les cellules normales.

Pour résumer, pour créer un tableau vous avez besoin des balises suivantes :

- **table** : balise de tableau
- **tr** : ligne d'un tableau (doit être contenu dans une balise `table`)
- **td** : cellule d'un tableau (doit être contenu dans une ligne)
- **caption** : nom de votre tableau (facultatif)

- **th** : en-tête de tableau (facultatif et doit être contenu dans la première ligne)

Note : D'autres balises peuvent être insérées dans un tableau HTML mais elles ne sont pas indispensables.

## La balise `iframe`

```
<iframe src="http://www.youtube.com/embed/wj7cU3Mr7xg"
frameborder="0" allowfullscreen width="640"
height="360"></iframe>
```

Cette balise est un peu particulière car elle sert à inclure du contenu venant d'une autre page web.

La balise `iframe` n'est généralement pas recommandée ; elle pose en effet des problèmes de duplication de contenu (et donc de référencement) car une page est affichée une seconde fois sur une autre.

Toutefois, les gros sites comme Youtube et Facebook se servent de cette balise pour que les webmasters puissent intégrer des vidéos et les plugins sociaux comme les boutons « J'aime ».

Jetons un œil aux attributs de la balise `iframe` :

- **src** : adresse de la page à afficher (la page source)
- **frameborder** : affichage ou non d'un cadre autour de l'`iframe`.  
Indiquez 0 pour non, 1 pour oui.
- **allowfullscreen** : autorise la mise en plein écran. Cet attribut est de type booléen, c'est-à-dire soit vrai soit faux. Le fait qu'il n'ait pas de valeur associée indique au navigateur qu'il doit utiliser la valeur par défaut : vrai donc 1.
- **width** : largeur de la balise `iframe`

- **height** : hauteur de la balise `iframe`

## Récapitulons

Nous avons maintenant terminé d'étudier les principales balises HTML permettant de mettre le contenu en forme dans la section `body` d'une page web.

Vous pouvez voir ces balises en action [en cliquant sur ce lien](#).

Dans la suite de ce chapitre, nous allons découvrir les balises permettant de créer des formulaires et de structurer le contenu des pages web.

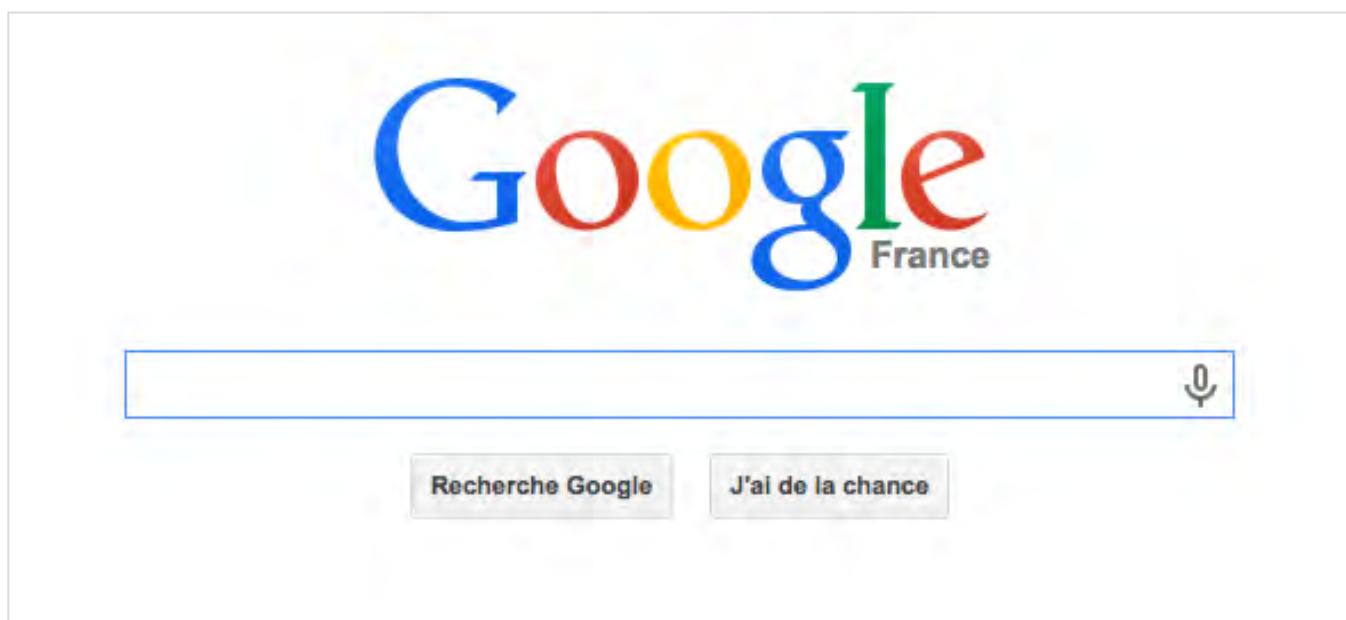
Avant de consulter la suite, allez dans le dossier des exercices et faites l'exercice 4 pour pratiquer, c'est très important.

## Chapitre 2.6

# Créer des formulaires en HTML

Dans cette sixième partie, nous allons étudier un genre de balise bien spécifique. Il s'agit des balises pour créer des formulaires.

Si le mot formulaire ne vous dit rien, sachez que vous en remplissez tous les jours. Par exemple, la page d'accueil de Google possède un formulaire :



Vous remplissez aussi des formulaires lorsque que vous vous inscrivez sur un site, contactez quelqu'un, écrivez un commentaire, répondez à des sondages, etc.

Pour résumer, **à chaque fois que vous tapez du texte sur le web, vous utilisez un formulaire.**

Au cours de cette partie, nous allons voir par quels éléments se constituent les formulaires. Ensuite, nous étudierons les éléments qu'il est possible d'ajouter pour les enrichir davantage.

N'ayez pas peur, nous allons procéder étape par étape :)

## Le formulaire de base

```
<form method="post" action="">
  <label for="prenom">Quel est votre prénom ?</label><br>
  <input type="text" name="prenom" id="prenom"><br>
  <input type="submit" value="Envoyer">
</form>
```

Décryptons la syntaxe d'un formulaire. Tout d'abord, nous voyons qu'il emploie la balise `form` (pour formulaire). Cette balise `form` est dotée de deux attributs :

- **method** : détermine la façon dont les données seront gérées lorsque l'utilisateur validera le formulaire. Cet attribut peut être doté de deux valeurs :
  - *get* : Les données du formulaire seront transmises dans l'url de la page et limitées à 255 caractères. Ce qui est peu pratique et peut poser des problèmes de sécurité.
  - *post* : Les données transmises seront invisibles et illimitées, c'est donc la méthode à privilégier.
- **action** : adresse de la page qui devra traiter les données du formulaire. Laissez vide signifie que la page sur laquelle se trouve le formulaire traitera les résultats.

*Notez que le traitement des données n'est pas possible en HTML, il faut utiliser un autre langage.*

Les formulaires que nous créerons dans les exercices seront inactifs mais ce n'est pas grave. L'important est de vous apprendre à reconnaître les balises constituant les formulaires et vous communiquer leur rôle.

Nous allons oublier la balise `label` de l'exemple ci-dessus quelques instants pour nous focaliser sur...

## La balise `input`

```
<input type="text" name="prenom" id="prenom">
```

La balise `input` permet d'insérer un champ de saisie dans un formulaire. C'est à dire un endroit où vous pouvez entrer des données (comme sur la page d'accueil de Google).

Jetons un œil aux attributs :

- **type** : spécifie le type du champ. Dans notre exemple, il s'agit du type texte (`text`). Nous allons aborder les principaux types ci-dessous.
- **name** : permet de donner un nom à la balise `input`, ce qui permettra à la page de traitement du formulaire de l'identifier. Il faut donc que la valeur de l'attribut `name` soit unique pour chaque élément de formulaire.
- **id** : définit un identifiant (unique également) qui permet de styler le champ en CSS. Cet identifiant est indispensable pour associer un champ avec une balise `label` que nous allons aborder ensuite.

Note : La particularité de la balise `input` est que vous ne pouvez n'y entrer qu'une seule ligne de texte.

## Autres valeurs de l'attribut "type"

L'attribut `type` de la balise `input` peut posséder d'autres valeurs en fonction de la nature des données qu'il va recevoir. Voyons quels sont les principaux types de données acceptés :

- **text** : vous le connaissez déjà, il s'agit du type le plus courant, le type texte.
- **password** : ici, il s'agit d'un mot de passe. Les caractères seront masqués par le navigateur.
- **hidden** : champ caché qui contient des données utiles au formulaire.
- **email** : champ destiné à contenir un email.
- **tel** : spécifie un numéro de téléphone.
- **url** : champ destiné à recevoir une adresse web (`http://`), FTP (`ftp://`) ou email (`mailto:`)

L'avantage d'utiliser les types `email`, `tel` ou `url` est que lorsque les visiteurs de votre site se trouvent sur un appareil mobile, le bon clavier leur sera directement proposé.

Avouez qu'il n'y a rien de plus fastidieux que de taper un numéro de téléphone avec un clavier texte classique.

## Attributs supplémentaires

```
<input type="text" name="prenom" id="prenom"
placeholder="Entrez votre prénom" size="40"
maxlength="25">
```

La balise `input` présentée précédemment ne contient que le minimum pour fonctionner. Il est possible de l'enrichir grâce à d'autres attributs :

- **placeholder** : affiche une indication dans le champ de saisie qui s'effacera dès que l'on commencera à taper.
- **size** : définit la taille du champ de saisie. Notez que vous pourrez le personnaliser plus précisément grâce au CSS.
- **maxlength** : définit le nombre maximum de caractères pouvant être tapés par l'utilisateur. Cet attribut est très utile pour limiter les erreurs de saisie des utilisateurs.
- **value** : permet d'attribuer une valeur par défaut. Particulièrement utile pour les champs de type `hidden`. Je ne l'ai pas inclus dans l'exemple ci-dessus car cela aurait désactivé l'attribut `placeholder`.

## La balise `label`

```
<label for="prenom">Quel est votre prénom ?</label><br>
<input type="text" name="prenom" id="prenom">
```

Rappelez-vous, tout à l'heure nous avons ignoré la balise `label`. En fait, cette balise sert à décrire un champ de saisie.

**Pour cela, il faut que la valeur de l'attribut `for` soit identique à la valeur de l'attribut `id` du champ auquel on veut l'associer.**

Dans l'exemple, on voit très bien que la valeur de l'attribut `for` est `prenom` car la valeur de l'attribut `id` de la balise `input` est `prenom`.

La balise `label` (ou libellé) est donc associée au champ de texte `prenom`.

Cette balise est là pour aider l'utilisateur à compléter le formulaire. Elle permet d'indiquer un texte descriptif lié au champ à remplir.

Note : Il n'est pas obligatoire d'inclure une balise `label` dans un formulaire. Il est fort possible que vous n'en trouviez pas dans certains formulaires.

## Une balise `input` spéciale

```
<input type="submit" value="Envoyer">
```

Le formulaire présenté tout à l'heure présentait une balise `input` d'un type que nous n'avons pas encore évoqué.

Lorsqu'une balise `input` possède un attribut de type `submit`, il s'agit en fait du bouton de validation du formulaire.

L'attribut `value` permet de définir le texte qui se trouve dans ce bouton. Dans l'exemple, le bouton contient le texte « Envoyer ».

## Deux autres balises `input` particulières

C'est promis, après j'arrête de vous parler de balises `input` ! En ce qui concerne ces deux-là, je suis certain que vous les connaissez déjà, enfin que vous vous en êtes déjà servi.

## Les boutons radio

```
<form method="post" action="">
  Quel type de thème préférez-vous ?<br>
  <p>
    <input type="radio" name="theme"
value="gratuit">Gratuit<br>
    <input type="radio" name="theme" value="premium"
checked>Premium
  </p>
  <input type="submit" value="Envoyer">
</form>
```

On insère des boutons radio dans un formulaire lorsque l'on veut que l'utilisateur fasse **un seul choix** parmi plusieurs propositions.

Explorons les attributs de ce type d'élément :

- **type** : doit obligatoirement contenir la valeur `radio` pour qu'il puisse être identifié en tant que tel par le navigateur.
- **name** : c'est une sorte d'identifiant commun pour grouper les choix que l'on propose à l'utilisateur. Cet attribut doit être le même pour tous les choix d'un même groupe (sinon l'utilisateur pourra en cliquer plusieurs, ce n'est pas ce que nous voulons).
- **value** : doit toujours comporter quelque chose. C'est ce qui sera envoyé à la page de traitement lors de la validation du formulaire.
- **checked** : coche une case par défaut. Cet attribut est facultatif. On peut aussi l'écrire de la forme `checked="checked"`, c'est à vous de voir mais cela revient au même.

On doit ensuite préciser ce que représente chaque bouton à l'utilisateur. C'est pour cela que j'ai indiqué Gratuit et Premium à la suite des balises `input` (sinon on aurait uniquement vu les deux boutons radio).

Comme tout élément de formulaire, il est possible de rajouter des balises `label` afin de guider les utilisateurs. Voici le même formulaire avec des balises `label` :

```
<form method="post" action="">
  Quel type de thème préférez-vous ?<br>
  <p>
    <input type="radio" name="theme" id="gratuit"
value="gratuit"><label for="gratuit">Gratuit</label><br>
    <input type="radio" name="theme" id="premium"
value="premium" checked><label for="premium">Premium</
label>
  </p>
  <input type="submit" value="Envoyer">
</form>
```

Attention : Veillez à ce que chaque attribut `id` possède une valeur unique. Cela est valable sur toute page web. Nous aborderons cela plus en détail dans la partie suivante.

Note : Le formulaire contient une balise `p` (balise de paragraphe) et une balise `br` (balise de saut de ligne) pour la mise en forme.

## Les cases à cocher

```
<form method="post" action="">
  Dans quelle boutique avez-vous déjà acheté un thème ?
  <p>
    <input type="checkbox" name="boutique"
value="woothemes">WooThemes<br>
    <input type="checkbox" name="boutique"
value="elegantthemes">Elegant Themes<br>
    <input type="checkbox" name="boutique"
value="themeforest">Themeforest
  </p>
  <input type="submit" value="Envoyer">
</form>
```

On se sert des cases à cocher pour que l'utilisateur sélectionne ce qu'il désire. C'est à dire rien, une ou plusieurs réponses.

Le principe est le même que pour les boutons radio, l'attribut name doit être identique pour tous les choix présentés sinon ils ne seront pas reconnus comme appartenant au même ensemble.

La seule différence avec les boutons radio se situe au niveau de l'attribut type. Il doit posséder la valeur checkbox.

Voici le même formulaire avec des balises label :

```

<form method="post" action="">
    Dans quelle boutique avez-vous déjà acheté un thème ?
    <p>
        <input type="checkbox" name="boutique"
id="woothemes" value="woothemes"><label
for="woothemes">WooThemes</label><br>
        <input type="checkbox" name="boutique"
id="elegantthemes" value="elegantthemes"><label
for="elegantthemes">Elegant Themes</label><br>
        <input type="checkbox" name="boutique"
id="themeforest" value="themeforest"><label
for="themeforest">Themeforest</label>
    </p>
    <input type="submit" value="Envoyer">
</form>

```

## La balise zone de texte

```

<form method="post" action="">
    <label for="retourclient">Comment ce guide va-t-il
vous aider dans votre projet ?</label><br>
    <textarea cols="50" rows="10" name="retourclient"
id="retourclient"></textarea><br>
    <input type="submit" value="Envoyer">
</form>

```

La zone de texte est l'élément dont il faut se servir pour donner plus de liberté aux utilisateurs.

Grâce à la balise `textarea` (littéralement zone de texte en anglais), les utilisateurs pourront écrire plusieurs lignes de texte (ce qui n'est pas possible avec la balise `input`).

Cette balise possède deux attributs que nous n'avons pas encore abordés :

- **cols** : le nombre de colonnes que la zone de texte peut afficher
- **rows** : le nombre de lignes que la zone de texte peut afficher

Il est possible d'inclure d'autres attributs comme `placeholder`, `maxlength` et surtout `name` qui est indispensable pour que la zone de texte puisse être reconnue par la page qui va traiter les données du formulaire.

## La balise liste déroulante

```
<form method="post" action="">
  <label for="articles">Quels articles préférez-vous sur
  WP Marmite ?</label><br>
  <select name="articles" id="articles">
    <option value="selections">Les sélections de
    thèmes</option>
    <option value="tutowp">Tutoriels WordPress</option>
    <option value="pluginswp">Tutoriels plugins
    WordPress</option>
    <option value="interviews">Interviews</option>
    <option value="conseils">Conseils</option>
  </select>
  <input type="submit" value="Envoyer">
</form>
```

Se servir des listes déroulantes est une bonne alternative aux boutons radio lorsque les choix sont trop nombreux.

Elles se construisent de la façon suivante (j'omets volontairement de parler de la balise `label` que vous connaissez déjà) :

La balise `select` va encadrer tous les choix possibles. Chaque choix est matérialisé par une balise `option`.

L'attribut `name` de la balise `select` est indispensable. Son rôle est le même que pour les balises précédentes (vous vous en souvenez, n'est-ce pas ?).

L'attribut `value` des balises `option` doit obligatoirement comporter quelque chose. La valeur de la réponse choisie sera envoyée à la page de traitement lorsque le formulaire sera validé.

## Grouper les éléments dans un formulaire

Voilà, vous connaissez désormais les éléments de formulaires les plus courants.

Toutefois, avant de vous laisser vous exercer, il faut que je vous précise quelque chose.

Nous avons vu que les balises `label` servaient à décrire la réponse attendue dans l'élément auquel elles sont associées. **Sachez qu'il est possible de grouper plusieurs éléments au sein d'un formulaire.**

C'est le rôle de la balise `fieldset`.

À la manière de la balise `label`, il est possible d'associer une description à ce groupe d'éléments. C'est précisément le rôle de la balise `legend`.

Voici un exemple avec ces balises en action :

```
<form method="post" action="">
  <fieldset>
    <legend>État civil</legend>
    <label for="nom">Entrez votre nom</label>
    <input type="text" name="nom" id="nom" ><br>
    <label for="prenom">Entrez votre prénom</label>
    <input type="text" name="prenom" id="prenom"><br>
  </fieldset>
  <input type="submit" value="Envoyer">
</form>
```

Bien entendu, la balise `fieldset` n'est utile que dans le cas de formulaires assez longs (dans les formulaires de commande par exemple).

Il n'y a pas de raison de l'utiliser pour un formulaire de contact doté de trois ou quatre champs.

## Récapitulons

Nous voilà enfin arrivés au terme de cette (longue) partie sur les formulaires. Vous pouvez dès maintenant comprendre le code des formulaires de n'importe quel site web. Pas mal n'est-ce pas ?

Profitez-en pour faire une petite pause, vous la méritez bien.

Pour bien intégrer toutes ces balises, regardez [les exemples de code en cliquant ici](#) et entraînez-vous avec l'exercice 5 du chapitre 2.

Lorsque vous vous sentirez prêt, poursuivez la lecture pour finaliser ce chapitre consacré au langage HTML.

## Chapitre 2.7

# Les balises génériques

Tout au long de ce chapitre, nous avons abordé une grande partie des balises qu'il est possible de rencontrer sur une page web (et donc dans un thème WordPress).

Ces balises permettent d'insérer plusieurs éléments (images, liens, textes, titres, tableaux, etc.) afin d'organiser au mieux l'information pour les internautes.

Il y a cependant quelques balises dont je ne vous ai pas encore parlé. Il s'agit des balises génériques.

Avant de décrire ces balises, nous devons faire un point sur une notion très importante qui concerne toutes les balises que nous avons vues jusqu'à présent.

## Les balises de type block et inline

En HTML, chaque balise possède un type d'affichage par défaut qui est soit `block`, soit `inline`.

Étudions ces deux types de balises en détail pour bien comprendre leur fonctionnement.

### Les balises de type block

Les balises de type `block` possèdent les caractéristiques suivantes :

- Elles s'affichent les unes en dessous des autres

- Elles peuvent contenir d'autres éléments de type `block`, de type `inline` ainsi que du texte sans aucune balise
- Leur largeur par défaut correspond à la largeur de la balise `block` où elles sont contenues
- Il est possible de leur attribuer des dimensions précises

Rien de tel qu'un petit dessin pour mieux comprendre comment ce type de balise fonctionne :



Dans ce schéma, toutes les balises présentes sont de type `block`. On peut voir qu'elles s'affichent les unes en dessous des autres.

La première balise bleue n'a pas été redimensionnée. Elle s'étend donc sur toute la largeur de l'écran. Il en est de même pour la balise noire qui se trouve à l'intérieur.

La balise orange est dimensionnée pour faire la moitié de la balise noire.

Les balises vertes contenues dans la balise orange n'ont pas été redimensionnées, elles s'affichent donc l'une au-dessus de l'autre et s'étendent sur toute la largeur de la balise parent.

Sans le savoir, vous connaissez déjà des balises de type `block`. On peut notamment citer :

- `h1`, `h2` ... `h6`
- `p`, `blockquote`, `hr`
- `ol`, `ul`, `li`
- `form`

Pour vous en convaincre, j'ai intégré ces balises [sur cette page](#) et en leur appliquant une bordure pour vous que vous puissiez vérifier leur comportement.

Note : Retenez que les balises `block` correspondent à des boîtes. Elles s'empilent les unes sous les autres et il est possible de les mettre les unes dans les autres comme des poupées russes.

## Les balises de type `inline`

À l'opposé, les balises de type `inline` possèdent ces caractéristiques :

- Elles s'affichent les unes à la suite des autres (de gauche à droite comme du texte)
- Elles peuvent contenir des éléments de type `inline`, du texte ou rien du tout
- Leur largeur est définie par rapport à ce qu'elles contiennent

Quelques balises de type `inline` :

- `a`, `strong`, `em`
- `img`

- `input`, `label`, `select`, `textarea`
- `iframe`.

Pour avoir quelque chose de plus concret et visuel, découvrez toutes ces balises [sur cette page](#).

## **Pourquoi ces deux types de balises ?**

En combinant ces deux types de balises, il est possible de créer des pages web complexes.

La structure d'une page web se définit par l'imbrication des balises de type `block` (rappelez-vous, ce sont comme des boîtes qui s'encastrent).

Il est ainsi possible de pouvoir définir des blocs qui correspondent à l'entête du site, au menu, au contenu principal, au pied de page, etc.

Chacun de ces éléments peut contenir d'autres blocs s'il y a besoin de sous-éléments mais aussi des balises `inline` et du texte.

Regardez par exemple les balises `block` de la page d'accueil de WP Marmite. On peut voir que certains blocs sont inclus dans d'autres et contiennent les éléments du site :



Bien entendu, tous ces blocs figurent à l'intérieur de la section body étant donné qu'ils sont visibles dans le navigateur.

Maintenant que vous comprenez mieux comment les balises permettent de définir la structure d'une page web. Intéressons-nous aux balises par défaut des types block et inline.

## La balise div

```
<div></div>
```

La balise div est certainement la balise que vous trouverez le plus souvent au sein des pages web. Elle est la balise de type block de base.

**Avec les balises `div`, vous allez pouvoir définir la structure de vos pages web.**

Dans le cas d'un thème WordPress, vous allez pouvoir modifier la structure des modèles de pages : page d'accueil, page des articles, page par défaut ... (nous aborderons cela dans le chapitre 5).

La balise `div` possède évidemment toutes les propriétés des balises `block` abordées précédemment.

Attention : On ne doit pas utiliser de balise `div` lorsque l'on peut employer une autre balise de type `block`. Par exemple pour un titre, il faut utiliser une des balises de titre (`h1`, `h2`... `h6`). Sans cela, les moteurs de recherche comprendraient moins bien le contenu de vos pages.

## La balise `span`

```
<span></span>
```

La balise `span` est la balise de type `inline` de base. Elle sert à grouper des éléments de type `inline` et à identifier certaines portions au sein d'un texte (ces portions pourront ensuite être personnalisées grâce à du CSS).

La balise `span` possède toutes les propriétés des balises `inline` abordées précédemment.

Attention : On ne doit pas utiliser de balise `span` lorsque l'on peut employer une autre balise de type `inline`. Par exemple pour mettre du texte en italique, il faut utiliser la balise `em`.

# Les balises introduites en HTML 5

Dans la partie 2 de ce chapitre sur la structure des pages web, nous avons vu que le HTML 5 était la dernière version du langage HTML.

De nouvelles balises de type `block` ont été introduites pour mieux structurer les pages web. On peut notamment citer :

- **header** : en-tête du site ou d'un bloc (en-tête d'article, de commentaire, de produit, etc.). Cette balise ne peut pas être placée dans une autre balise header
- **footer** : pied de page du site ou d'un bloc (fin d'article, d'une présentation de produit, etc.). Cette balise doit contenir des informations à propos de la balise où elle est située (copyright, articles relatifs, etc.)
- **nav** : cette balise doit contenir le ou les menus d'un site
- **section** : cette balise permet de regrouper un contenu thématique
- **aside** : cette balise permet d'insérer du contenu relatif au contenu principal (on peut se servir de cette balise pour inclure une barre latérale par exemple)
- **article** : cette balise doit inclure du contenu comme des articles de blog, des sujets de forum ou encore des commentaires.

## Les points communs à toutes les balises HTML

À ce niveau du guide, il est essentiel de préciser que chaque balise HTML peut posséder un attribut `class` et un attribut `id`.

Nous connaissons déjà l'attribut `id`, utilisé dans les formulaires. Il s'avère que cet attribut possède une autre fonction.

*L'attribut `class` et l'attribut `id` servent à distinguer les éléments d'une page web en vue de leur appliquer un style avec le langage CSS.*

Nous allons aborder cela plus en détails dans le chapitre suivant.

En attendant, **retenez qu'il ne peut pas exister d'éléments possédant un attribut `id` dont la valeur est identique.**

Un identifiant doit être unique, une classe pas forcément.

Par exemple, le code suivant n'est pas correct car deux identifiants sont identiques :

```
<p id="note">Note : Ceci est une note.</p>
<p id="note">Note : Ceci est une autre note.</p>
```

Pour appliquer le même style à deux paragraphes, il faut utiliser l'attribut `class` :

```
<p class="note">Note : Ceci est une note.</p>
<p class="note">Note : Ceci est une autre note.</p>
```

## Récapitulons

Vous venez de terminer le premier gros chapitre de Relooker son Thème. Logiquement, vous ne devez plus être autant impressionné par le code des pages web qu'au début de votre lecture.

Vous êtes désormais capable de visualiser la structure des blocs (rappelez-vous les poupées russes) et de comprendre leur rôle.

Avant de passer au chapitre suivant, je vous encourage à étudier le code HTML de vos sites favoris (que les sites soient construits ou non avec WordPress n'a aucune importance).

Quel que soit votre navigateur, vous devriez avoir la possibilité d'afficher le code source d'une page. Cherchez les balises dont nous avons parlé et étudiez la structure de ces sites. C'est un excellent exercice.

Quand vous vous sentirez prêt, passez au test de fin de chapitre (toujours dans le dossier *Exercices*). Cela va vous permettre de vous exercer à manipuler le langage HTML.

**RELOOKER SON THÈME**

# **CHAPITRE 3**

## **LE CSS, LA MISE EN FORME DES SITES WEB**

---

**Découvrez comment le langage CSS  
fonctionne afin d'embellir  
l'apparence des pages  
d'un site internet**

## Chapitre 3.1

# Les principes du langage CSS

Au cours du précédent chapitre, nous avons vu comment structurer une page web, maintenant ajoutons un peu de couleur à tout cela.

Même si votre thème WordPress possède déjà un ou plusieurs fichiers CSS, il est important de comprendre le fonctionnement de ce langage avant de procéder à des modifications.

Rappelez-vous, un fichier CSS est lié à un fichier HTML via la balise `link` possédant la syntaxe suivante :

```
<link href="LIEN_VERS_LE_FICHIER_CSS" rel="stylesheet"
media="screen" type="text/css">
```

C'est bon pour vous ? Alors commençons avec...

## Les sélecteurs, les propriétés et les valeurs

```
sélecteur{
    propriété:valeur;
}
```

Ces trois lignes résument à elles seules, une grande partie des principes qui régissent le CSS.

Cette instruction CSS se compose de trois éléments :

- **sélecteur** : le sélecteur permet d'identifier un ou plusieurs éléments au sein d'une page web (par exemple les paragraphes, les liens, une balise `div` particulière etc.)
- **propriété** : la propriété CSS à appliquer à l'élément désigné par le sélecteur
- **valeur** : valeur de la propriété CSS à appliquer

Si cela vous semble flou, voici un exemple pour mieux assimiler ce concept.

Imaginons que nous voulions que tous les paragraphes d'une page soient écrits en rouge. Le code CSS correspondant est :

```
p{  
    color:red;  
}
```

Le sélecteur du paragraphe est `p`, cela tombe bien car la balise HTML d'un paragraphe porte le même nom.

La propriété à modifier est `color` pour la couleur du texte et la valeur associée est `red` pour le rouge.

**Voir cet exemple en action**

Bien sûr, plusieurs propriétés peuvent être utilisées pour chaque sélecteur. Par exemple, si nous voulons que les paragraphes d'une page soient rouges et possèdent une taille de 22 pixels il faut écrire cela :

```
p{
  color:red;
  font-size:22px;
}
```

[Voir cet exemple en action](#)

*Notez que chaque propriété doit se terminer par un point-virgule ;. Une exception peut toutefois être faite pour la dernière propriété. Dans l'exemple précédent, la propriété font - size peut ne pas avoir de point-virgule.*

Au niveau de la mise en page, il est possible d'écrire toutes les instructions les unes à la suite des autres. Pour des raisons de lisibilité, le code est souvent agencé comme vous le verrez dans Relooker son Thème. **Cette mise en page est appelée l'indentation.**

Il existe de nombreux sélecteurs en CSS, passons en revue ceux que l'on retrouve régulièrement...

## Les sélecteurs de balises

Dans les exemples précédents, nous avons parlé du sélecteur p. Il permet d'appliquer des instructions CSS à tous les paragraphes d'une page web.

En fait, à chaque balise HTML correspond un sélecteur CSS. Grâce à cela, vous appliquerez un style particulier aux éléments d'une page web.

Voici quelques-uns de ces sélecteurs :

- **body** : appliquer un style à toute la page web
- **h1, h2 ... h6** : pour les titres
- **p** : pour les paragraphes

- **a** : pour les liens
- **blockquote** : pour les citations
- **img** : pour les images
- **ul** : pour les listes désordonnées
- **ol** : pour les listes ordonnées
- **li** : pour les éléments de liste
- **table** : pour les tableaux

La liste continue pour toutes les autres balises du langage HTML...

Si vous vous servez d'un de ces sélecteurs, il faut bien comprendre que le style que vous leur donnerez s'appliquera à TOUS les éléments spécifiés par le sélecteur sur votre page web.

Par exemple, si vous donnez un style à la balise `img`, cela s'appliquera à TOUTES les images.

La bonne pratique consiste à donner des styles généraux aux éléments puis d'affiner ensuite. Nous allons voir comment y arriver avec...

## Les sélecteurs de classes et d'identifiants

Nous avons vu à la fin du module précédent que chaque balise HTML peut posséder un attribut `class` et un attribut `id`. Je vous rappelle la syntaxe :

```
<div class="maclasse" id="monidentifiant">Contenu de la  
balise div.</div>
```

*Souvenez-vous qu'il ne peut pas y avoir plusieurs balises dotées du même identifiant alors que cela est possible pour les classes.*

Cette règle existe car **un identifiant, comme son nom l'indique, sert à identifier un seul élément HTML**. Avoir deux balises avec le même identifiant ne permettrait plus de pouvoir identifier correctement le bon élément.

C'est un peu comme si vous aviez deux personnes pour une seule carte d'identité, ce n'est pas possible.

Par contre, **plusieurs éléments HTML peuvent avoir la même classe**.

Par exemple dans le cas d'un blog listant cinq articles, les balises `article` contenant chaque article possèdent la même classe de façon à ce que les mêmes styles leur soient appliqués.

**Un élément HTML peut aussi posséder plusieurs classes.**

Toujours dans le cas des balises `article` d'un blog, les classes suivantes sont ajoutées par WordPress : `post post-XXX type-post status-publish format-standard has-post-thumbnail hentry`.

Cela permet de cibler des articles possédant une caractéristique précise dans le code.

Voici comment représenter les classes et les identifiants dans notre fichier CSS :

```
.maclasse{
    /* Propriétés à appliquer à la classe "maclasse" */
}
#monidentifiant1{
    /* Propriétés à appliquer à l'identifiant
"monidentifiant1" */
}
#monidentifiant2{
    /* Propriétés à appliquer à l'identifiant
"monidentifiant2" */
}
```

On utilise :

- Un point suivi du nom de la classe pour une classe.
- Un dièse suivi du nom de l'identifiant pour un identifiant.

**Retenez-bien ces deux définitions car il arrive que cela nous joue parfois des tours.**

Note : Vous venez de voir comment inclure des commentaires en CSS dans l'exemple ci-dessus.

## Bien connaître la convention de nommage

Vous devez savoir que les classes et les identifiants ne peuvent pas porter n'importe quel nom. Il est possible d'utiliser :

- Des lettres minuscules (les majuscules sont autorisées mais cela se fait peu)
- Des chiffres (de 0 à 9)
- Des tirets -
- Des underscores \_ (également connus sous le nom de « tirets bas »)

Vous ne devez pas utiliser:

- Des caractères accentués
- Des caractères spéciaux
- Un chiffre en tant que premier caractère
- Un tiret en tant que premier caractère

## Les sélecteurs plus complexes

### Le sélecteur descendant : X Y

Nous avons vu les façons les plus communes d'attribuer un style CSS à des éléments HTML (ciblage d'un élément via son type de balise, sa classe ou son identifiant).

Il est néanmoins possible de cibler plus précisément des éléments à l'aide de sélecteurs plus complexes.

*Par la suite, nous désignerons des sélecteurs quelconques par les lettres X, Y et Z. Ils peuvent donc correspondre à un sélecteur de balise, à une classe ou à un identifiant.*

Il est possible de placer des sélecteurs les uns à la suite des autres afin de désigner des éléments plus précisément. Par exemple :

```
#monidentifiant .maclasse{  
    /*Code CSS*/  
}
```

Un tel enchaînement signifie que le style sera appliqué aux éléments ayant une classe `maclasse` si et seulement s'ils sont situés dans un élément possédant un identifiant `monidentifiant` :

```
<div id="monidentifiant">
    <div class="maclasse">Le code CSS est appliqué.</div>
    <div class="maclasse">Le code CSS est appliqué.</div>
    <div class="autreclasse">Le code CSS n'est pas
appliqué !</div>
</div>
```

**Voir cet exemple en action**

On peut enchaîner plusieurs sélecteurs et obtenir quelque chose du genre :

```
ul li a{
    /*Code CSS*/
}
```

Cela vous permettra d'attribuer un style aux liens placés dans une liste désordonnée (comme c'est le cas pour les menus) :

```
<ul>
    <li><a href="#" title="LIEN_1">Le code CSS sera
appliqué à ce lien</a></li>
    <li><a href="#" title="LIEN_2">Le code CSS sera
appliqué à ce lien</a></li>
    <li><a href="#" title="LIEN_3">Le code CSS sera
appliqué à ce lien</a></li>
</ul>
```

**Voir cet exemple en action**

## Le sélecteur associatif : XY ou XZ

On peut donner un style unique à un couple de sélecteurs :

- **X.maclasse** : sélecteur X possédant une classe maclasse
- **X#monidentifiant** : sélecteur X possédant un identifiant monidentifiant

Concrètement cela peut donner :

```
a.maclasse{
    color:green;
}
```

Avec ce code, tous les liens possédant la classe maclasse seront écrits en verts.

[Voir cet exemple en action](#)

## Le sélecteur de survol : X:hover

Il est possible de définir un style à appliquer lorsque l'on survole un élément avec la souris. Par exemple :

```
a: hover{
    color:red;
}
```

Ce code fera en sorte que les liens deviennent rouges au survol de la souris.

[Voir cet exemple en action](#)

# Récapitulons

Arrêtons-nous ici pour les sélecteurs. Il en existe d'autres mais le but de Relooker son Thème est de vous transmettre les bases. Ce que nous avons vu concerne l'essentiel des règles CSS.

L'objectif de cette première partie est de vous faire comprendre comment fonctionne le CSS dans le cas général.

Avant de passer à la suite, je vous propose de tester vos nouvelles connaissances au sein d'un d'exercice (faites l'exercice 1 du chapitre sur le CSS).

## Chapitre 3.2

# Les principes du langage CSS – Partie 2

Dans la partie précédente nous avons étudié la syntaxe générale du CSS. À présent nous allons aborder les deux notions fondamentales du langage CSS.

**Connaître ces principes est capital afin de bien comprendre comment le CSS fonctionne.**

Une fois que vous les aurez assimilés, vous pourrez vous rendre compte de vos erreurs (nous en faisons tous) et les corriger afin d'obtenir le rendu que vous désirez avec votre code CSS.

Commençons tout de suite par...

## L'héritage

Comme nous avons pu le voir dans le chapitre consacré au HTML, les balises peuvent s'intégrer les unes aux autres indéfiniment. Cela permet de créer la structure d'une page web.

La balise de base est `html` qui contient les balises `head` et `body` qui contiennent elles mêmes d'autres balises.

La balise `body` en particulier peut renfermer des dizaines de balises en fonction de la complexité de la page web.

**Toutes ces balises forment ce que l'on appelle une arborescence.**

Vous connaissez probablement ce mot via la structure des dossiers de votre ordinateur.

Et bien là, c'est la même chose sauf qu'au lieu d'avoir des dossiers parents qui contiennent des dossiers enfants, on a des balises parents qui contiennent des balises enfants.

Voici comment nous pourrions représenter une arborescence simple pour une page web :



Dans la précédente partie, nous avons appris que les balises HTML pouvaient servir de sélecteurs (par exemple p est le sélecteur des paragraphes).

Imaginons que vous vouliez que tout le texte sur votre site soit écrit en gris.

Si vous appliquez ce que nous avons étudié jusqu'à présent, il serait logique d'écrire des instructions CSS pour afficher tous les éléments qui affichent du texte (p, ul, ol, span, etc. ) en gris.

Vous devinez bien que passer tous les sélecteurs en revue peut prendre un certain temps... De plus on peut oublier certains sélecteurs.

Heureusement, la notion d'héritage va pouvoir nous simplifier la vie.

*En CSS, la notion d'héritage veut qu'une balise parent peut transmettre certains styles CSS à ses balises enfants.*

Pour revenir à l'exemple de couleur de texte, on peut associer la couleur gris à la balise body :

```
body{  
    color:grey;  
}
```

Ainsi, tous les éléments enfants de la balise body (paragrophes, titres, listes, etc.) seront affichés en gris.

*Grâce à l'héritage, il n'y a pas besoin de définir des propriétés pour chaque élément puisque les éléments enfants les héritent de leurs parents.*

**Voir cet exemple en action**

Notez toutefois que l'héritage ne fonctionne pas pour toutes les propriétés CSS. Seules certaines propriétés relatives aux polices d'écritures et aux couleurs se transmettent.

Imaginez le bazar si cela devait fonctionner pour les arrière-plans, chaque balise enfant aurait le même arrière-plan et il faudrait tout changer sauf celui de la balise body.

# L'importance

Si je vous parle de cascade, vous allez vous dire que je ne dois pas avoir les idées très claires en écrivant ce guide. Rassurez-vous, il n'en est rien :)

L'acronyme CSS signifie « Cascading Style Sheets » ce qui équivaut à « Feuilles de style en cascade ».

On emploie le mot « cascade » pour montrer que certaines règles CSS ont plus d'importance que d'autres.

Les règles qui auront le plus de poids seront appliquées par le navigateur alors que les autres seront ignorées.

**La cascade sert donc à déterminer quelles propriétés seront appliquées à un élément et, par conséquent, lui donneront son apparence finale.**

Décortiquons ensemble cette fameuse cascade pour découvrir les règles qui seront appliquées en priorité par le navigateur.

## L'origine des règles

Les styles CSS peuvent provenir de différentes sources. Vous le ne savez peut-être pas mais **votre navigateur possède des styles CSS par défaut.**

Ces styles sont appliqués lorsque les fichiers CSS d'un site sont incomplets (ou absents). Vous avez pu voir ces styles dans les exemples du chapitre consacré au HTML.

Chaque navigateur possède ses propres styles par défaut, d'où la différence de rendu que l'on peut observer selon celui que l'on utilise.

Les styles CSS par défaut des navigateurs ont une faible importance.

Lorsqu'une page web embarque du code CSS, celui-ci prend le dessus par rapport aux styles par défaut.

Enfin, ce n'est pas tout à fait ça.

**En réalité, le navigateur combine toutes les instructions CSS qu'il peut trouver et applique celles qui ont le plus d'importance.**

Revenons à l'exemple précédent. Nous avons vu que le contenu de la page s'affichait en gris grâce à la notion d'héritage. Par contre, si nous ajoutons un lien, on peut voir qu'il ne s'affiche pas en gris mais en bleu :



Par défaut, l'héritage ne s'applique pas aux liens. Comme aucun style n'a été défini dans le fichier CSS associé à cette page, le style de lien du navigateur s'applique.

**Voir cet exemple en action**

*Pour résumer, les styles CSS définis sur une page web ont plus d'importance par rapport aux styles par défaut des navigateurs.*

Maintenant, il faut savoir qu'il existe 3 manières d'intégrer du code CSS sur une page web.

Ces méthodes n'ont pas la même importance. Commençons par étudier...

## L'importance des styles externes

```
<link href="style.css" type="text/css" rel="stylesheet"
media="screen">
```

Derrière ce nom quelque peu ingrat se cache pourtant la manière la plus commune d'utiliser le CSS.

C'est d'ailleurs la méthode utilisée dans tous les exemples donnés précédemment, à savoir le chargement d'un fichier CSS dans la section head via une balise `link`.

Si vous vous rappelez ce que nous avons abordé précédemment, l'utilisation de fichiers CSS distincts de la page HTML est la raison d'être du CSS.

En effet, **il faut séparer le fond de la forme.**

Les styles externes ont une faible importance mais cette méthode a toujours plus de poids que les styles par défaut du navigateur.

## L'importance des styles embarqués

```
<style>
.maclasse{
    color:blue;
}
</style>
```

Comme vous pouvez le voir ci-dessus, sans le savoir vous connaissiez déjà ce qu'étaient les styles embarqués.

Ils correspondent à l'insertion de code CSS au sein d'une balise `style` dans la section `head` d'une page web.

Bien que les styles embarqués possèdent la même importance que les styles externes, **il n'est pas recommandé de les employer** car cela irait à l'encontre de la distinction du fond et de la forme.

On peut toutefois en trouver dans certains thèmes WordPress pour des raisons particulières (nous verrons pourquoi très bientôt).

## L'importance des styles en ligne

```
<body style="color:green;">Contenu de la page en vert.</body>
```

Le dernier moyen d'intégrer du code CSS dans une page web est de le placer directement dans les balises HTML via l'attribut `style`.

Cette méthode a l'avantage d'avoir une importance plus élevée que les deux précédentes, toutefois cela perturbe grandement la séparation du fond et de la forme.

Ici tout est fusionné. Ce n'est clairement pas le chemin à suivre.

Les styles en ligne peuvent toutefois être utilisés de temps en temps pour mettre du texte en forme (via l'éditeur de WordPress notamment) mais cela ne doit pas aller plus loin.

**Voir cet exemple en action**

Plaçons-nous à présent à l'intérieur d'un fichier CSS pour étudier l'importance des instructions. Elle est déterminée grâce à...

## La spécificité

L'intérieur des feuilles de style renferme aussi une autre partie de la cascade (c'est-à-dire d'autres niveaux d'importance).

En rédigeant nos instructions CSS, nous pouvons employer différents sélecteurs. Des identifiants, des classes, des éléments (p, a etc) mais aussi des pseudo-classes (:hover pour le survol, etc).

Chacun de ces sélecteurs possède un poids qui lui est propre :

- Les éléments : 1
- Les classes et les pseudo-classes : 10
- Les identifiants : 100

En ajoutant les poids des sélecteurs d'une règle CSS, vous pourrez en déterminer le poids total, c'est à dire son importance.

Prenons un exemple :

```
h2{
    color:green;
}
.article h2{
    color:red;
}
#main .article h2{
    color:blue;
}
```

Calculons le poids de chacune de ces règles :

- h2 est un élément donc le total est de : 1

- Il y a ensuite une classe (`.article`) et un élément (`h2`) donc :  
 $10 + 1 = 11$
- Enfin il y a un identifiant (`#main`), une classe (`.article`) et un élément (`h2`), ce qui nous fait :  $100 + 10 + 1 = 111$

La troisième instruction a le plus d'importance. Elle sera appliquée par le navigateur car il s'agit de l'instruction la plus spécifique (l'élément `h2` sera donc affiché en bleu).

**Voir cet exemple en action**

Note : Une bonne pratique est d'utiliser des règles générales puis de spécifier si cela est nécessaire. Il serait ridicule de complexifier des choses alors que cela n'est pas nécessaire.

## L'ordre d'affichage

Si des instructions CSS placées dans un fichier externe possèdent la même spécificité, il faut que le navigateur affiche tout de même quelque chose.

Dans ce cas, l'ordre d'affichage sera pris en compte. Par exemple :

```
h2{  
    color:blue;  
}  
h2{  
    color:red;  
}
```

Une page web dotée du code CSS ci-dessus affichera les titres de niveau 2 en rouge (`color : red ;`) car il s'agit de la dernière instruction.

**Voir cet exemple en action**

*Retenez donc qu'en cas d'égalité au niveau de l'importance, les règles écrites en dernier seront plus importantes que les précédentes.*

Tout à l'heure, nous avons vu que les styles externes et les styles embarqués possédaient une importance équivalente.

Cela est vrai sur le papier mais leur ordre d'affichage dans la section head d'une page web donnera plus d'importance à l'un ou à l'autre.

*En général, les styles embarqués sont affichés après les styles externes.* Les instructions des styles embarqués ont donc plus d'importance par rapport à celles des styles externes.

Toutefois, si des styles embarqués ne sont pas définis, le navigateur se rabattra sur les styles externes, puis sur les styles par défaut. C'est ça la cascade !

## **En dernier recours, utilisez !important**

```
h2{  
    color:red !important;  
}
```

Si jamais vous n'avez pas la possibilité de définir des instructions avec une plus grande importance que les instructions existantes, il reste une solution.

En ajoutant `!important` juste après la valeur d'une propriété, **vous serez assuré que cette instruction sera prise en compte par le navigateur.**

Cela fonctionnera même si d'autres instructions sont définies plus loin dans le fichier CSS ou par l'intermédiaire de styles en ligne.

**Voir cet exemple en action**

Attention à ne pas abuser de cette méthode. Utilisez-la seulement lorsque vous ne pouvez pas faire autrement.

**Dans la plupart des cas, donner une meilleure spécificité à ses instructions CSS et les inclure en fin de fichier suffit amplement.**

## Récapitulons

Au cours de cette partie, nous avons vu que le CSS est régi par deux grands concepts, à savoir l'héritage et l'importance.

Nous pouvons résumer cela en trois points :

- Pour afficher une page web, un navigateur analyse toutes les instructions CSS d'une page web et applique celles qui ont le plus d'importance.
- Si aucune règle n'a été définie pour les éléments d'une page, les styles par défaut du navigateur seront appliqués.
- Lorsque l'on rédige du CSS, il faut toujours donner l'importance minimale (mais suffisante) pour obtenir le rendu désiré.

L'ensemble de ces règles compose ce que l'on appelle la cascade. Dans tous les cas, le navigateur trouvera un moyen d'afficher les éléments d'une page web.

Pour tester ces nouvelles connaissances théoriques, ouvrez l'exercice 2 associé à ce chapitre puis passez à la suite.

## Chapitre 3.3

# Les propriétés CSS – Mettre le texte en forme

Nous y voilà ! Fini la théorie, nous allons pouvoir entrer dans le concret en découvrant les propriétés CSS qui vont pouvoir vous aider à changer l'apparence de votre thème WordPress.

Note : Vous n'avez pas à apprendre cela par coeur, cette partie et celles qui vont suivre vont vous aider à comprendre ce qui se trouve dans le code CSS de votre thème.

**J'espère que vous n'avez pas éludé les deux premières parties car elles sont très importantes.**

Sans elles, il se peut que vous ne compreniez pas pourquoi votre site affiche quelque chose que vous n'attendiez pas.

Maintenant que cela est posé, passons aux propriétés.

Je vous rappelle la syntaxe d'une instruction CSS :

```
sélecteur{  
    propriété1:valeur1;  
    propriété2:valeur2;  
    /* idem pour la suite */  
}
```

Commençons avec les propriétés CSS qui concernent...

# Les polices d'écriture

## font-family

```
font-family:"Lucida Grande", Verdana, Arial, sans-serif;
```

La propriété font - family va vous permettre de **définir les polices d'écriture à utiliser** sur votre site ainsi que pour différents éléments.

Vous pouvez voir ci-dessus que plusieurs polices figurent en tant que valeur :

1. "Lucida Grande"
2. Verdana
3. Arial
4. sans-serif

Quand le navigateur va lire cette propriété, il va essayer d'afficher la première police ("Lucida Grande").

Si elle n'est pas disponible, le navigateur va tenter d'afficher la seconde et ainsi de suite jusqu'à temps qu'il trouve une police disponible.

Note : Il faut mettre des guillemets au nom d'une police si elle est composée de plusieurs mots. Vous avez pu voir que c'était le cas pour "Lucida Grande".

[Voir cette propriété en action](#)

## font-size

```
font-size:14px;
```

La propriété `font-size` permet de définir la taille de la police. La valeur pourra être spécifiée en plusieurs unités :

- **px** : taille en pixels.
- **em** : taille proportionnelle à la taille par défaut du navigateur (un texte en 2em sera donc deux fois plus grand)
- **%** : pourcentage de la taille de la police de l'élément parent.

On peut également attribuer des valeurs relatives pour la propriété `font-size` :

- **xx-large** : encore plus grand
- **x-large** : très grand
- **large** : grand
- **medium** : normal
- **small** : petit
- **x-small** : très petit
- **xx-small** : encore plus petit
- **larger** : plus grand que la taille de la police de l'élément parent
- **smaller** : plus petit que la taille de la police de l'élément parent

[Voir cette propriété en action](#)

## font-weight

```
font-weight:bold;
```

Avec la propriété `font-weight`, on va pouvoir mettre plus ou moins une police en gras. Les valeurs suivantes peuvent lui être attribué :

- **bold** : gras par défaut
- **normal** : texte par défaut (pas de gras)
- **bolder** : plus gras que gras

- **lighter** : moins gras que gras
- une valeur entre 100 et 900 (par tranche de 100) sachant que :
- **100** = lighter
- **400** = normal
- **700** = bold
- **900** = bolder

Note : Toutes les polices ne prennent pas en charge toutes les nuances de « gras ».

[Voir cette propriété en action](#)

## font-style

```
font-style:italic;
```

Donne un style italique ou normal à la police d'écriture. Les valeurs possibles sont :

- **normal** : style par défaut
- **italic** : mettre la police en italique

[Voir cette propriété en action](#)

## font-variant

```
font-variant:small-caps;
```

Cette propriété est un peu moins utilisée. Elle sert à mettre le texte en petites majuscules. `font-variant` n'accepte que 2 valeurs possibles :

- **normal** : affichage par défaut
- **smallcaps** : toutes les minuscules sont transformées en majuscules de petite taille

[Voir cette propriété en action](#)

## line-height

```
line-height:28px;
```

La propriété `line-height` correspond à la hauteur de ligne. Il est important de ne pas spécifier une valeur trop faible afin de ne pas perturber la lisibilité.

Il est possible d'indiquer une valeur en pixels, en em ou en pourcentage.

[Voir cette propriété en action](#)

## font

```
font:italic small-caps bold 22px/28px "Lucida Grande",  
Verdana, Arial, sans-serif;
```

`font` est une propriété un peu spéciale car elle permet de rassembler toutes les propriétés concernant les polices en une seule propriété. C'est une sorte de « meta propriété » si vous voulez.

Si vous l'utilisez, sachez que vous devez spécifier la liste des valeurs dans l'ordre suivant pour que cela fonctionne :

font:

- font-style
- font-variant
- font-weight
- font-size/line-height
- font-family

Note : Seules les valeurs de font-size et font-family sont obligatoires pour la propriété font.

## Les textes

Après avoir exploré les propriétés CSS relatives aux polices, tournons-nous vers les propriétés relatives aux textes.

### color

```
color:#ff0000;
```

Nous avons déjà croisé la propriété color précédemment dans ce guide mais je suis resté bref sur les valeurs qu'il est possible de lui attribuer.

La façon la plus courante d'attribuer une couleur est de prendre sa valeur hexadécimale comme dans l'exemple ci-dessus.

On peut également fournir les formats de valeurs suivantes :

- nom de la couleur en anglais : red, green, blue
- couleur en RGB : rgb(255, 0, 0), rgb(0, 255, 0), rgb(0, 0, 255)
- couleur en HSL : hsl(0, 100%, 50), hsl(120, 100%, 50), hsl(240, 100%, 50)

- couleur en RGBA et HSLA : Même chose que RGB et HSL sauf qu'un paramètre supplémentaire définit un niveau d'opacité.

Vous retrouverez tous ces formats en détail dans la fiche n°2 sur les couleurs en CSS dans le dossier *Ressources* associé au guide.

**Voir cette propriété en action**

## **text-align**

```
text-align:center;
```

La propriété `text-align` permet d'aligner le texte à gauche, au centre ou à droite. Elle peut prendre les valeurs suivantes :

- **left** : alignement à gauche
- **center** : centré
- **right** : alignement à droite
- **justify** : étend le texte sur toute la ligne

**Voir cette propriété en action**

## **text-decoration**

```
text-decoration:underline;
```

La propriété `text-decoration` vous permettra d'ajouter une décoration à votre texte.

Concrètement il s'agit de souligner, de barrer et de surligner le texte.  
Explications avec les différentes valeurs :

- **underline** : valeur la plus utilisée, qui permettra de souligner le texte
- **overline** : place une ligne au dessus du texte, c'est du soulignement à l'envers si on veut ou du surlignage
- **line-through** : barrer le texte
- **none** : affichage par défaut

[Voir cette propriété en action](#)

## text-transform

```
text-transform:capitalize;
```

Cette propriété est utile pour transformer les textes. Les valeurs possibles de la propriété `text-transform` sont :

- **capitalize** : mettre des majuscules à tous les mots
- **lowercase** : mettre tout le texte en minuscules
- **uppercase** : mettre tout le texte en majuscules
- **none** : affichage par défaut

[Voir cette propriété en action](#)

## text-indent

```
text-indent:20px;
```

La propriété `text-indent` permet d'ajouter une indentation sur la première ligne d'un bloc de texte (par exemple un paragraphe).

On peut lui spécifier une valeur avec une longueur précise (voir toutes les unités de mesure) ou un pourcentage correspondant à une portion de la largeur de l'élément parent.

Note : Si la valeur est négative, le texte sera décalé vers la gauche.

[Voir cette propriété en action](#)

## **letter-spacing**

```
letter-spacing: 2px;
```

La propriété `letter-spacing` correspond à l'espacement entre les lettres. Elle est généralement définie en pixels mais les autres unités de mesure sont autorisées.

Utilisez la valeur `normal` pour revenir à l'espacement par défaut.

Note : Il est possible d'utiliser des valeurs négatives pour rapprocher les lettres.

[Voir cette propriété en action](#)

## **word-spacing**

```
word-spacing: 10px;
```

La propriété `word-spacing` est similaire à la précédente sauf que cette fois, il s'agit de l'espacement entre les mots et non entre les lettres.

Les règles sont les mêmes concernant les unités de mesure et les valeurs négatives. Utilisez `normal` pour revenir à l'espacement par défaut.

[Voir cette propriété en action](#)

## **vertical-align**

```
vertical-align:middle;
```

`vertical-align` sert à définir l'alignement vertical du texte. Cette propriété peut prendre les valeurs suivantes :

- **baseline** : la valeur par défaut
- une longueur dans une des unités de mesure du CSS (voir la fiche n°3 dans les ressources)
- valeur en pourcentage (%)
- **sub** : affiche le texte en indice
- **super** : affiche le texte en exposant
- **top** : le haut de l'élément est aligné sur le haut de l'élément le plus haut de la ligne
- **bottom** : le bas de l'élément est aligné sur le bas de l'élément le plus bas de la ligne
- **text-top** : alignement le plus haut possible
- **text-bottom** : alignement le plus bas possible
- **middle** : texte affiché au milieu

[Voir cette propriété en action](#)

# Les listes

Nous avons vu dans le chapitre consacré au HTML qu'il est possible d'afficher du texte sous forme de liste (ordonnée ou désordonnée).

Faisons le tour des propriétés permettant de les personnaliser :

## list-style-type

```
list-style-type:disc;
```

La propriété `list-style-type` permet de définir les marqueurs de listes (les types d'éléments par lesquels elles débiteront)

Valeurs pour les listes désordonnées (balises `ul`) :

- **disc** : une simple puce, il s'agit de la valeur par défaut
- **circle** : comme une puce sauf que le contenu est vide
- **square** : un carré
- **none** : rien du tout

[Voir cette propriété en action](#)

Valeurs pour les listes ordonnées (balises `ol`) :

- **decimal** : 1,2,3... valeur par défaut
- **decimal-leading-zero** : 01, 02, 03...
- **lower-alpha** : a, b, c... (lettres minuscules)
- **lower-roman** : i, ii, iii... (chiffres romains)
- **upper-alpha** : A, B, C... (lettres majuscules)
- **upper-roman** : I, II, III... (chiffres romains majuscules)

- **none** : rien du tout

[Voir cette propriété en action](#)

## list-style-position

```
list-style-position:inside;
```

La propriété `list-style-position` permet d'inclure ou non les marqueurs de listes avec le contenu. Les valeurs possibles sont :

- **inside** : les marqueurs seront avec le contenu
- **outside** : les marqueurs seront hors du contenu

[Voir cette propriété en action](#)

## list-style-image

```
list-style-image:url('monimage.png');
```

Vous avez probablement remarqué que certains marqueurs de listes étaient différents. En effet, il est possible d'inclure des images au lieu des marqueurs proposés par défaut.

- **url** : adresse de l'image à utiliser en tant que marqueur
- **none** : ne pas utiliser d'image

[Voir cette propriété en action](#)

## list-style

```
list-style:circle inside url('imageliste.jpg');
```

À la manière de la propriété `font` vue précédemment, `list-style` permet de rassembler les trois propriétés relatives aux listes en une seule propriété.

L'ordre suivant doit être respecté :

```
list-style : - liststyle-type - liststyle-position -  
liststyle-image
```

Note : Les valeurs par défaut sont `disc` `outside` `none`.

## Récapitulons

Voilà, nous avons terminé le premier sous-chapitre consacré aux propriétés CSS. Nous avons découvert les propriétés liées au texte.

Certaines d'entre elles agissent sur la police d'écriture, d'autres sur les mots et l'alignement et d'autres sur les listes.

Pour mieux les appréhender, je vous invite à faire le troisième exercice associé à ce chapitre pour tester vos nouvelles connaissances.

## Chapitre 3.4

# Les propriétés CSS – Bordures et arrière-plans

Continuons de passer en revue les différentes propriétés CSS que vous allez pouvoir trouver dans des thèmes WordPress et plus généralement dans beaucoup de sites internet.

Ces propriétés vont vous permettre de colorer, agrémenter et enrichir les éléments d'une page web.

## Les bordures

Vous le savez peut-être déjà, mais il est possible de spécifier une bordure pour n'importe quelle balise HTML. Voici les propriétés qui vous y aideront :

### **border-width**

```
border-width:2px;
```

La propriété `border-width` permet de définir l'épaisseur de la bordure d'une balise. Elle peut prendre les valeurs suivantes :

- épaisseur en pixels, comme dans l'exemple ci-dessus
- **thin** : une bordure fine
- **medium** : une bordure moyenne qui est la valeur par défaut
- **thick** : une bordure plus épaisse

## border-color

```
border-color:blue;
```

Maintenant que votre bordure a une épaisseur, il faut lui donner une couleur sinon elle restera en noir (la couleur par défaut). Comme nous avons pu le voir pour la propriété `color`, il est possible d'attribuer plusieurs formats de couleurs.

Le format hexadécimal étant le plus courant (par exemple `#999999` pour du gris).

La fiche n°2 dans les Ressources fournies avec Relooker son Thème détaille l'ensemble des formats de couleurs acceptés en CSS.

## border-style

```
border-style:solid;
```

La dernière propriété indispensable à toute bordure est `border-style`. Elle permet d'indiquer le style de bordure à adopter. Les valeurs possibles sont :

- **solid** : une ligne continue
- **dotted** : en pointillés
- **dashed** : des traits les uns à la suite des autres
- **double** : une double bordure
- **groove** : crée une sorte de cadre
- **ridge** : comme groove mais avec les couleurs inversées
- **inset** : une autre sorte de cadre en deux couleurs
- **outset** : comme inset mais avec les couleurs inversées
- **none** : aucune bordure n'est appliquée

## border

```
border:2px solid blue;
```

Écrire à chaque fois les 3 propriétés précédentes peut se révéler fastidieux. La propriété `border` permet de les combiner au sein d'une seule propriété sur le même principe que les propriétés `font` et `list-style`.

Vous devez suivre cette syntaxe pour que cela fonctionne correctement :

```
border: border-width border-style border-color;
```

On peut constater que la valeur de la propriété `border` est composé de 3 « sous-valeurs » issues des propriétés `border-width`, `border-style` et `border-color`.

Il est possible d'omettre des sous-valeurs, dans ce cas les valeurs par défaut des propriétés correspondantes seront appliquées.

[Voir cette propriété en action](#)

## Gérer les bordures plus finement

Les quatre propriétés cités auparavant servent à appliquer une bordure sur tous les côtés d'un bloc, c'est à dire en haut, à droite, en bas et à gauche.

Si vous désirez n'appliquer qu'une ou plusieurs bordures sur les quatre, c'est possible.

Voici les propriétés à utiliser pour la propriété `border` :

- **border-top** : appliquer un style à la bordure supérieure
- **border-right** : appliquer un style à la bordure de droite

- **border-bottom** : appliquer un style à la bordure inférieure
- **border-left** : appliquer un style à la bordure de gauche

Même chose pour les trois autres propriétés :

- **border-top-width** : épaisseur de la bordure supérieure
- **border-right-width** : épaisseur de la bordure de droite
- **border-bottom-width** : épaisseur de la bordure inférieure
- **border-left-width** : épaisseur de la bordure de gauche
- **border-top-style** : style de la bordure supérieure
- **border-right-style** : style de la bordure de droite
- **border-bottom-style** : style de la bordure inférieure
- **border-left-style** : style de la bordure de gauche
- **border-top-color** : couleur de la bordure supérieure
- **border-right-color** : couleur de la bordure de droite
- **border-bottom-color** : couleur de la bordure inférieure
- **border-left-color** : couleur de la bordure de gauche

[Voir cette propriété en action](#)

Comme le montrent ces quelques exemples, vous pouvez constater qu'une fois que l'on connaît les principes de base, il est possible de faire tout ce que l'on veut en terme de bordure.

## **border-radius**

```
border-radius:5px;
```

Le CSS 3 (la dernière version du CSS) a introduit quelques propriétés dont `border-radius`. Elle permet d'arrondir les angles des bordures (par défaut, des angles droits sont présents).

En donnant une seule valeur, cela arrondira les quatre coins. En indiquant une valeur composée de 4 sous-valeurs, vous pourrez intervenir sur chacun des coins.

```
border-radius:5px 10px 15px 20px;
```

Dans cet exemple, les valeurs correspondent dans l'ordre :

- **5px** : coin en haut à gauche
- **10px** : coin en haut à droite
- **15px** : coin en bas à droite
- **20px** : coin en bas à gauche

[Voir cette propriété en action](#)

## Gérer les arrondis plus finement

De la même façon que pour les bordures, il est possible de gérer l'arrondi des coins un par un :

- **border-top-left-radius** : arrondi du coin en haut à gauche
- **border-top-right-radius** : arrondi du coin en haut à droite
- **border-bottom-right-radius** : arrondi du coin en bas à droite
- **border-bottom-left-radius** : arrondi du coin en bas à gauche

## Les arrière-plans

Grâce au CSS, il est possible d'attribuer un arrière-plan à n'importe quel élément HTML. Pour cela, il est possible d'utiliser les propriétés suivantes :

### background-color

```
background-color:blue;
```

Cette propriété permet d'attribuer une couleur d'arrière-plan. Indiquez simplement un code couleur pour obtenir la couleur souhaitée.

Reportez vous au guide des couleurs (fiche n°2) fourni dans les ressources pour connaître les valeurs que peut accepter cette propriété.

[Voir cette propriété en action](#)

## **background-image**

```
background-image:url('image.png');
```

De la même façon, il est possible d'utiliser une image en arrière-plan. Il faut placer l'adresse de l'image (complète ou relative) comme indiqué dans l'exemple.

[Voir cette propriété en action](#)

## **background-repeat**

```
background-repeat:no-repeat;
```

Il faut savoir que par défaut, une image est répétée horizontalement et verticalement au sein d'un élément. Par exemple, si vous avez une petite image en arrière-plan de la balise body, l'image sera reproduite sur tout le site.

Il arrive parfois que cette répétition soit défavorable, utilisez les valeurs suivantes pour contrôler la répétition :

- **repeat** : l'image sera reproduite horizontalement et verticalement (valeur par défaut)

- **repeat-x** : l'image ne sera reproduite que horizontalement
- **repeat-y** : l'image ne sera reproduite que verticalement
- **no-repeat** : l'image ne sera pas reproduite (elle ne sera affichée qu'une seule fois)

Voir cette propriété en action

## background-size

```
background-size:cover;
```

En imaginant que vous vouliez mettre une image en arrière-plan, comment faire pour qu'elle s'adapte à plusieurs types d'écrans ?

La propriété background-size est là pour vous aider. Elle peut se voir attribuer les valeurs suivantes :

- **100px 100px** : indique la largeur et la hauteur avec les unités de mesures acceptées par le CSS.
- **50%** : donne un ou deux pourcentages (s'il n'y en a qu'un le second sera défini automatiquement (valeur "auto")).
- **cover** : l'image d'arrière-plan s'adaptera à la largeur de l'élément HTML auquel elle est associée.
- **contain** : l'image d'arrière-plan étendra sur toute la hauteur de l'élément.

Voir cette propriété en action

## background-attachment

```
background-attachment:fixed;
```

La propriété `background-attachment` sert à définir si l'image d'arrière-plan définie par `background-image` doit bouger ou non lorsque l'utilisateur fait défiler la page. Vous pouvez y attribuer les valeurs suivantes :

- **scroll** : l'image d'arrière-plan défile avec la page (valeur par défaut).
- **fixed** : l'image d'arrière-plan reste fixe par rapport à la page.

[Voir cette propriété en action](#)

## background-position

```
background-position:fixed;
```

Cette propriété sert à attribuer les coordonnées à partir desquelles l'image d'arrière-plan doit être affichée.

Note : Les coordonnées sont définies par rapport à deux axes (un horizontal et un autre vertical : l'abscisse et l'ordonnée). Ces deux axes ont toujours pour origine le coin supérieur gauche de l'élément HTML concerné (l'origine étant l'endroit où les deux axes se croisent).

Vous pouvez définir la position de votre image de plusieurs façons :

- En combinant les mots-clés `left`, `top`, `center`, `bottom` et `right` qui signifient respectivement « gauche », « haut », « centre », « bas » et « droite ». Le premier mot définit la position horizontale et le second la position verticale. Vous pourrez obtenir par exemple :
- **left top** : placer l'image en haut à gauche
- **right bottom** : placer l'image en bas à droite
- **center top** : centrer l'image en haut

- **0px 0px** : en donnant des valeurs en pixels ou toute autre unité de mesure du CSS, dans ce cas c'est la même règle, la première valeur correspond à la position horizontale et la seconde à la verticale.
- **10% 50%** : en donnant un pourcentage. 50% correspondant à une image centrée.

Il est possible d'aller plus loin en mélangeant les pourcentages et les valeurs données avec des unités.

[Voir cette propriété en action](#)

## background-origin

```
background-origin:content-box;
```

Le CSS 3 a apporté cette nouvelle propriété afin de définir l'origine des axes horizontaux et verticaux à partir desquels l'image d'arrière-plan sera positionnée (nous en avons parlé avec `background-position`).

`background-origin` peut prendre les valeurs suivantes :

- **border-box** : l'origine sera basée sur la bordure extérieure.
- **padding-box** : l'origine sera basée sur le padding relatif à l'élément HTML (tout sauf la bordure). Il s'agit de la valeur par défaut.
- **content-box** : l'origine sera basée sur le contenu (exclusion de la bordure et de la zone de padding).

[Voir cette propriété en action](#)

## background-clip

```
background-clip:border-box;
```

Une autre propriété issue de CSS 3 est `background-clip`. Cette propriété correspond à la surface sur laquelle l'arrière-plan va s'étendre. Voici les valeurs qu'on peut lui attribuer :

- **`border-box`** : l'arrière-plan va s'étendre sur tout l'élément HTML, bordure comprise.
- **`padding-box`** : l'arrière-plan va s'étendre sur tout à l'exception de la bordure (valeur par défaut).
- **`content-box`** : l'arrière-plan ne va s'étendre que sur le contenu (exclusion de la bordure et de la zone de padding).

[Voir cette propriété en action](#)

## background

Comme pour la propriété `border`, il est possible de rassembler toutes les propriétés relatives aux arrière-plans dans une seule propriété. Il s'agit de la propriété `background`.

Voici la syntaxe de cette propriété (rentrez bien les valeurs dans l'ordre sinon les valeurs par défaut seront appliquées) :

```
background: color position size repeat origin clip  
attachment image;
```

## Récapitulons

Au cours de cette partie, nous avons vu quelles propriétés utiliser pour enrichir l'apparence des éléments d'une page web grâce aux bordures et aux arrière-plans.

Avant de passer à la suite, je vous propose de passer à la pratique avec l'exercice 4.

## Chapitre 3.5

# Les propriétés CSS – Dimensionner et ajuster les éléments

Jusqu'à présent, nous avons parcouru les propriétés servant à styler les contenus (polices, textes etc.) et nous avons commencé à styler les éléments HTML avec les propriétés de bordures et d'arrière-plans.

Il nous reste cependant quelque chose d'essentiel à aborder : comment dimensionner et positionner les balises HTML au sein d'une page.

Rappelez-vous, en HTML il existe des éléments de type `block`. Les propriétés CSS que nous allons aborder dans cette partie vont vous permettre d'apprendre à les transformer pour que le contenu des pages web ressemble à quelque chose.

Vous êtes prêt ? Alors on attaque avec...

## Les dimensions

### **width**

```
width:960px;
```

Le CSS permet d'attribuer aux balises de type `block` (que l'on peut appeler un bloc) une largeur grâce à la propriété `width`. Vous pouvez lui affecter les valeurs suivantes :

- **960px** : une valeur en pixel ou tout autre unité de mesure valide en CSS.
- **100%** : un pourcentage qui correspondra au pourcentage de la largeur du bloc parent (le bloc dans lequel le bloc à styler est placé).
- **auto** : la largeur est calculée automatiquement par le navigateur (valeur par défaut).

## height

```
height:400px;
```

La propriété `height` est en quelque sorte la petite sœur de `width` puisqu'il s'agit ici de contrôler la hauteur d'un bloc. Les types de valeurs que cette propriété accepte sont les mêmes :

- **400px** : une valeur en pixel ou tout autre unité de mesure valide en CSS.
- **50%** : un pourcentage qui correspondra au pourcentage de la hauteur du bloc parent (le bloc dans lequel le bloc à styler est contenu).
- **auto** : la hauteur est calculée automatiquement par le navigateur (valeur par défaut).

[Voir ces propriétés en action](#)

## Largeur/hauteur maximale et minimale

Pour chacune des deux propriétés présentées ci-dessus existe deux propriétés servant à définir une largeur maximale et minimale ainsi qu'une hauteur maximale et minimale.

On obtient donc :

- **max-width** : Largeur maximale
- **min-width** : Largeur minimale
- **max-height** : Hauteur maximale
- **min-height** : Hauteur minimale

Ces propriétés acceptent les mêmes valeurs que `width` et `height`.

Si vous ne cernez pas encore bien l'intérêt de ces propriétés, vous verrez qu'elles sont particulièrement utilisées pour créer des sites responsives (c'est à dire qui s'adaptent à toutes les tailles d'écrans). Nous aborderons cela dans la partie 7 de ce chapitre.

[Voir ces propriétés en action](#)

## Les marges

Nous avons à présent des blocs dimensionnés mais ils sont toujours collés les uns aux autres.

Souvenez-vous du module sur le HTML, les balises de type `block` s'affichent les unes en dessous des autres.

De même, les éléments contenus dans d'autres éléments sont automatiquement placés en haut à gauche de leurs éléments parents.

À la manière des marges dans un logiciel de traitement de texte, on peut attribuer des marges à un bloc. Nous allons voir qu'il existe deux types de marges en CSS.

### **margin : les marges externes**

```
margin:40px;
```

La propriété `margin` correspond aux marges extérieures d'un bloc. Cela signifie qu'il est possible d'ajouter de l'espace (une marge) au delà de la bordure.

Voici les types de valeur que l'on peut attribuer à la propriété `margin` :

- **40px** : une valeur en pixel ou tout autre unité de mesure valide en CSS.
- **50%** : un pourcentage qui se base sur les dimensions du bloc parent (le bloc dans lequel le bloc à styler est situé).
- **auto** : les marges sont calculées automatiquement par le navigateur (valeur par défaut).

En écrivant le code donné ci-dessus, l'élément aura une marge extérieure de 40 pixels tout autour de lui. C'est à dire en haut, à droite, en bas et à gauche.

Il est possible de spécifier des marges pour chaque côté avec les propriétés suivantes :

- **margin-top** : marge supérieure
- **margin-right** : marge de droite
- **margin-bottom** : marge inférieure
- **margin-left** : marge de gauche

Nous avons déjà vu qu'il était possible de rassembler plusieurs propriétés au sein d'une seule. C'est également le cas pour `margin`. On peut donner des marges différentes pour chaque côté de la façon suivante :

```
margin:10px 20px 30px 40px;
```

Dans cet exemple, la marge supérieure sera de 10 pixels, celle de droite sera de 20 pixels, la marge inférieure sera de 30 pixels et celle de gauche sera de 40 pixels.

Note : Lorsqu'il y a quatre marges d'indiquées, la première correspond toujours à la marge supérieure puis on tourne dans le sens des aiguilles d'une montre.

On peut aussi ne donner que deux sous-valeurs à la propriété `margin` :

```
margin:10px 20px;
```

Dans ce cas, les marges supérieures et inférieures seront de 10 pixels et les marges latérales seront de 20 pixels.

Il y a aussi une configuration avec trois sous-valeurs :

```
margin:10px 20px 30px;
```

Ici, la marge supérieure sera de 10 pixels, la marge inférieure de 30 pixels et les marges latérales seront de 20 pixels.

Pour spécifier une marge nulle, il suffit d'indiquer 0 ou 0px pour chacune des valeurs.

[Voir cette propriété en action](#)

Passons désormais au second type de marge...

## **padding : les marges internes**

```
padding:40px;
```

La propriété `padding` correspond aux marges intérieures d'un bloc.

Elles se trouvent entre la bordure et le contenu de l'élément. Cette propriété peut recevoir les valeurs suivantes :

- **40px** : une valeur en pixel ou tout autre unité de mesure valide en CSS.
- **50%** : un pourcentage qui correspondra au pourcentage des dimensions du bloc parent (le bloc dans lequel le bloc à styler est contenu).

Note : Il n'est pas possible d'attribuer de valeur `auto` à la propriété `padding`.

Comme pour la propriété `margin`, `padding` se décline en sous-propriétés afin de mieux cibler le côté sur lequel on veut travailler :

- **`padding-top`** : marge intérieure supérieure
- **`padding-right`** : marge intérieure de droite
- **`padding-bottom`** : marge intérieure inférieure
- **`padding-left`** : marge intérieure de gauche

Lorsque l'on désire rassembler ces quatre propriétés, on peut écrire :

```
padding:10px 20px 30px 40px;
```

Et toujours comme `margin`, il est possible de ne spécifier que deux ou trois sous-valeurs :

```
padding:10px 20px;
```

Dans ce cas, les marges internes supérieures et inférieures seront de 10 pixels et les marges internes latérales seront de 20 pixels.

[Voir cette propriété en action](#)

# box-sizing

```
box-sizing:content-box;
```

La propriété `box-sizing` sert à déterminer comment la taille des éléments est calculée. par le navigateur. Elle peut comprendre deux valeurs :

- **content-box** : seules les marges internes sont incluses dans la largeur (valeur par défaut).
- **border-box** : les marges internes (padding) et les bordures seront incluses dans la largeur (width).

Pour mieux comprendre, cliquez sur le lien ci-dessous afin de constater les différences.

[Voir cette propriété en action](#)

## Récapitulons

Ici s'achève cette partie sur la gestion des blocs au sein d'une page HTML. J'imagine que vous devez mieux saisir comment les éléments d'une page web sont dimensionnés et agencés.

Je vous propose maintenant de passer à la pratique avec l'exercice 5 associé à ce chapitre.

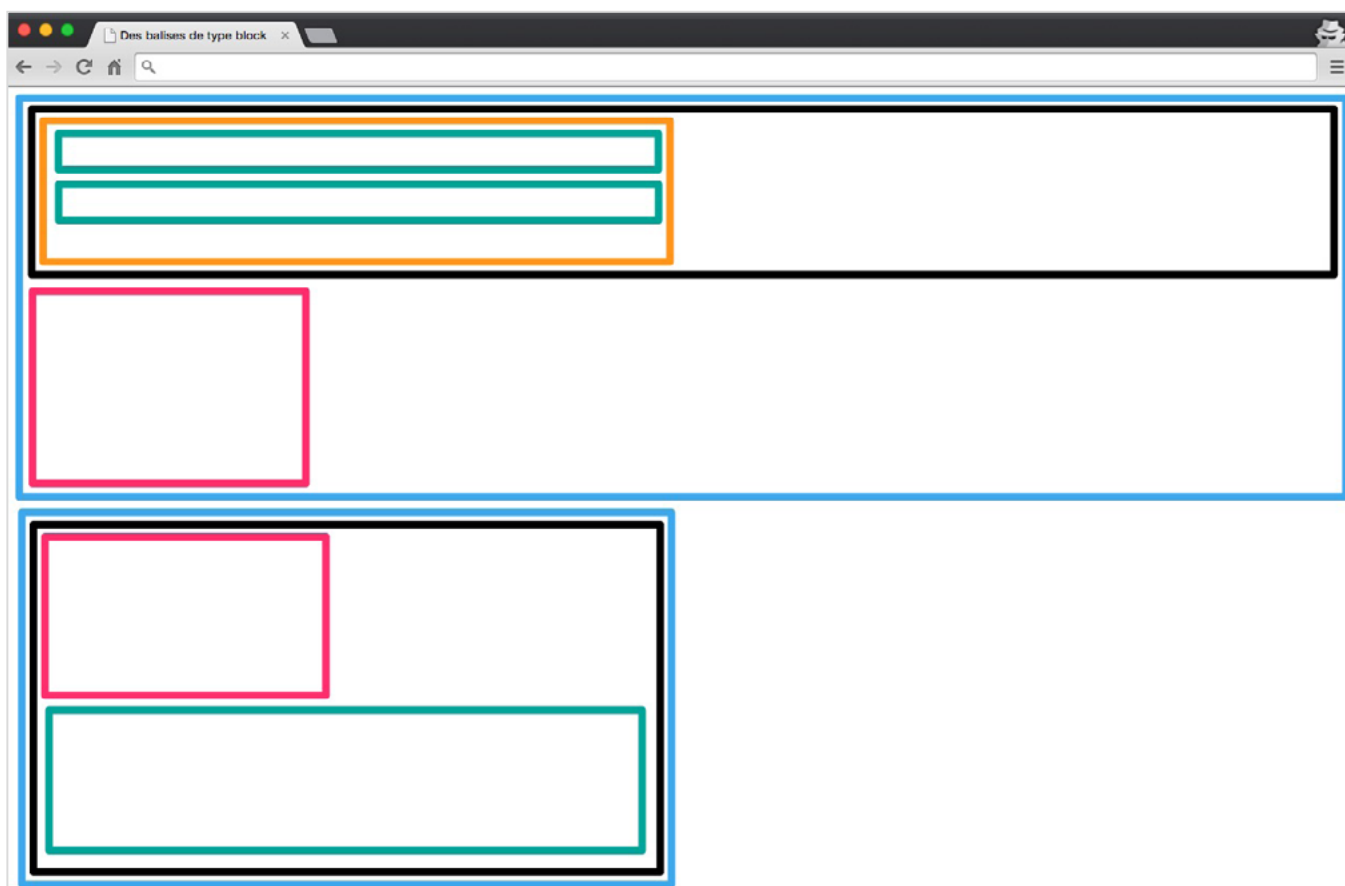
## Chapitre 3.6

# Les propriétés CSS – Positionner et gérer l’affichage des blocs

Au cours de ce chapitre, nous avons vu comment styler les éléments HTML. Vous savez également leur donner une certaine taille ainsi que des marges (les fameux `margin` et `padding`).

En revanche, nous ne savons toujours pas les positionner au sein d’une page web. Pour l’instant, ils ne gardent que leur positionnement par défaut, c’est à dire qu’ils sont les uns en dessous des autres.

Pour illustrer cela, j’avais précédemment utilisé cette image :



Nous allons voir comment faire pour positionner les éléments HTML exactement comme nous le voulons.

## Le positionnement

Commençons par la propriété phare du positionnement en CSS, il s'agit de...

### float

```
float:right;
```

La propriété `float` est couramment utilisée pour mettre en place des sites dotés de plusieurs colonnes (un blog avec une barre latérale par exemple).

Concrètement, la propriété `float` sert à placer un bloc le plus à droite ou à gauche possible (selon la valeur attribuée) du bloc parent.

**On dit que le bloc est sorti du flux.** C'est à dire qu'il est sorti du flux d'affichage par défaut (les uns à la suite des autres de haut en bas).

[Voir cette propriété en action](#)

La propriété `float` peut prendre les valeurs suivantes :

- **right** : le bloc sera placé le plus à droite possible du bloc parent.
- **left** : le bloc sera placé le plus à gauche possible du bloc parent.
- **none** : le bloc doit être positionné normalement (aucun flottement).

[Voir une application concrète](#)

### clear

```
clear:both;
```

Comme on a pu le voir dans un des exemples cités auparavant, le texte (ou tout autre élément HTML) a tendance à se glisser entre les éléments flottants.

Il est possible d'éviter cela grâce à la propriété `clear`.

Voyons les valeurs qu'il est possible d'attribuer à la propriété `clear` :

- **both** : Les éléments qui suivront seront placés sous les éléments flottants [Voir un exemple concret](#)
- **right** : Les éléments qui suivront seront placés après les éléments flottants à droite [Voir un exemple concret](#)
- **left** : Les éléments qui suivront seront placés après les éléments flottants à gauche [Voir un exemple concret](#)
- **none** : Les éléments qui suivront seront placés par défaut (ils remonteront).

## position

```
position: relative;
```

La propriété `position` sert à donner un comportement spécifique à un élément HTML.

Cela peut être assez déroutant à première vue mais ne vous inquiétez pas, vous allez mieux comprendre grâce aux exemples :)

- **static** : Ici aucune difficulté, car il s'agit du positionnement par défaut (empilement classique)
- **relative** : Il sera possible de déplacer un élément par rapport à sa position par défaut.
- **absolute** : Il sera possible de déplacer un élément par rapport à la fenêtre du navigateur.

- **fixed** : Il sera possible de déplacer un élément par rapport à la fenêtre du navigateur et restera à l'écran lorsque la page défilera.

Pour déplacer les blocs (qu'ils soient en position relative, absolue ou fixée), il faut utiliser les propriétés suivantes :

- **left** : Déplace l'élément vers la droite (car ajoute de l'espace à gauche).
- **top** : Déplace l'élément vers la bas (car ajoute de l'espace en haut).
- **right** : Déplace l'élément vers la gauche (car ajoute de l'espace à droite).
- **bottom** : Déplace l'élément vers la haut (car ajoute de l'espace en bas).

Attention : ce sont des propriétés et non des valeurs. Vous devez attribuer des valeurs à ces propriétés afin de déplacer les éléments comme bon vous semble. Cela peut être en pixels ou dans toute autre unité utilisée en CSS.

[Voir cette propriété en action](#)

## z-index

```
z-index:100;
```

Avec toutes ces histoires de positionnement, vous imaginez bien qu'il doit y avoir des cas où des blocs peuvent se superposer.

Comment faire si l'on veut qu'un bloc s'affiche par dessus un autre ? Et bien il faut utiliser la propriété `z-index`.

En plus de la longueur et de la largeur, il existe un axe dans le sens de la profondeur (la 3<sup>e</sup> dimension).

La propriété `z-index` permet de positionner un bloc sur cet axe suivant la valeur qui lui sera attribuée. Plus le nombre sera important, plus il sera au-dessus de la pile (ou au premier plan si vous préférez).

Attention : la propriété `z-index` ne fonctionne que si un bloc est positionné, cela veut dire que la propriété `position` doit être appliquée à l'élément avec `relative`, `absolute` ou `fixed` pour valeur.

Retenez également que :

- L'absence de `z-index` équivaut à `z-index : 0 ;`, c'est à dire la valeur par défaut.
- Si deux blocs ont le même `z-index`, le bloc se trouvant en dernier dans le code se retrouvera au premier plan par rapport à l'autre (on retrouve la notion d'importance).
- Il est possible de donner un `z-index` négatif.

[Voir cette propriété en action](#)

## L'affichage

Il existe encore d'autres petites subtilités concernant l'affichage en CSS. Vous pourrez les rencontrer dans le code de thèmes WordPress. Il donc est important de les étudier pour savoir à quoi elles correspondent.

### **display**

```
display: block;
```

Voilà une propriété presque « magique » car **elle permet de définir comment doit se comporter un élément HTML.**

Rappelez-vous, nous avons étudié les balises de type `block` (paragraphes, titres, `div` etc) et les balises de type `inline` (liens, images, `span` etc).

Ces deux types possèdent chacun leurs propres comportements.

Et bien figurez-vous que la propriété `display` permet de changer le comportement par défaut d'un élément HTML. Elle peut prendre pour valeur :

- **block** : L'élément se comportera comme un élément de type `block`.
- **inline** : L'élément se comportera comme un élément de type `inline`.
- **inline-block** : L'élément se placera comme un élément de type `inline` MAIS se comportera comme un élément de type `block`.
- **none** : L'élément ne s'affichera pas.

Note : La propriété `display` peut se voir attribuer d'autres valeurs mais il est préférable de ne pas les aborder pour dans l'essentiel.

[Voir cette propriété en action](#)

## overflow

```
overflow: hidden;
```

Il arrive parfois que le contenu d'un bloc soit trop important pour sa taille. Il faut donc gérer les dépassements éventuels. C'est là que la propriété `overflow` entre en scène.

Inspectons les valeurs qu'il est possible de lui attribuer :

- **hidden** : Le contenu qui dépasse sera caché.
- **visible** : Tout ce qui dépasse sera visible.

- **scroll** : Le contenu reste à l'intérieur du bloc mais un ascenseur sera ajouté pour que l'on puisse tout voir.
- **auto** : Si le contenu dépasse, un ascenseur sera ajouté. Si le bloc contient des éléments flottants, cela permet de ramener la taille du bloc à celle de son contenu.

[Voir cette propriété en action](#)

## visibility

```
visibility:hidden;
```

La propriété `visibility` sert à masquer ou non un élément HTML. Voyons les valeurs que cette propriété peut accepter :

- **visible** : L'élément est affiché et bien visible.
- **hidden** : L'élément est masqué.

Note : À la différence de `display:none;`, un élément masqué conserve ses proportions. Il devient juste invisible.

[Voir cette propriété en action](#)

## opacity

```
opacity:0.5;
```

Que faire si l'on ne veut pas qu'un élément disparaisse entièrement ? Qu'il soit plus ou moins transparent ?

Et bien il suffit d'appliquer la propriété `opacity` pour attribuer un niveau de transparence.

Les valeurs possibles sont comprises entre 0 et 1. 0 correspondant à invisible et 1 à visible. Par défaut, les éléments sont visibles.

[Voir cette propriété en action](#)

## Récapitulons

Félicitations, vous venez de parcourir une grande partie des propriétés CSS !

Je suis conscient qu'il n'est pas possible de tout retenir après une seule lecture. C'est normal. Cela viendra avec la pratique.

Veillez bien à inspecter le code des exemples fournis avec chaque propriété. Cela est indispensable pour bien comprendre.

Avant de passer à la suite, il est temps de vous exercer. Ouvrez l'exercice 6 du chapitre sur le CSS afin de prendre en main les propriétés dont nous avons parlé dans cette partie.

## Chapitre 3.7

# **Le responsive et 7 bonnes pratiques en CSS**

D'ici quelques minutes, vous aurez terminé la lecture de ce chapitre consacré au langage CSS. Vous pourrez alors mettre vos nouvelles connaissances à l'épreuve au sein d'un test.

Avant cela il nous faut aborder quelque chose d'incontournable aujourd'hui : la gestion du responsive. Nous verrons ensuite quelques bonnes pratiques à adopter lorsque vous écrirez votre propre code CSS.

## **Un site responsive est indispensable**

Comment faire en sorte qu'un site soit accessible sur la pléthore d'appareils qui existent aujourd'hui (tablettes, téléphones, etc.) ?

Avec un site responsive pardi !

Bien que cela ne soit pas le seul moyen, c'est à ce jour la manière la plus commune d'optimiser un site pour les appareils mobiles.

Si un site est responsive, Google pourra même le favoriser dans ses pages de résultats si la recherche est effectuée à partir d'un téléphone ou d'une tablette.

Étant donné que l'utilisation et le nombre de ces appareils est croissant, vous avez tout intérêt à ce que votre site soit responsive.

La quasi totalité des thèmes WordPress disponibles à l'heure actuelle sont responsives. Toutefois, il se peut que vous ayez besoin de « relooker » l'apparence de certains éléments à la version mobile de votre site.

Alors comment faire ? En fait, ce n'est pas très compliqué, il faut utiliser...

## Les media queries

Les média quoi ?!

Les « medias queries ». Ça signifie les « requêtes de média » en anglais.

*Derrière ce nom pompeux se cache un principe tout simple : celui d'appliquer des instructions CSS dans certaines conditions.*

Par exemple, pour qu'un site s'affiche bien sur un téléphone on peut définir des instructions CSS pour les appareils qui possèdent une largeur d'écran spécifique.

En fait, **l'idée est d'adapter l'affichage du site selon l'espace dont il dispose.**

Il existe deux moyens d'intégrer ces media queries sur un site web :

1. Dans un ou plusieurs fichiers CSS qui seront chargés dans la section head des pages web.
2. Dans le fichier CSS principal

Nous allons nous focaliser sur cette seconde manière de procéder car il s'agit de la plus simple à gérer (vous aurez toujours un seul fichier CSS).

Voyons à quoi une media query ressemble et auscultons-là :

```
@media screen and (max-width: 800px){  
  body{  
    color:#ecf0f1;  
    background-color:#3498db;  
  }  
}
```

[Voir cet exemple en action](#)

## Structure des media queries

Comme le montre l'exemple précédent, une media query se compose :

- Du préfixe `@media` pour indiquer qu'il s'agit bien d'une media query.
- D'une ou plusieurs conditions spécifiant si les instructions doivent être appliquées.

Dans notre cas, il y a deux conditions :

1. `screen` pour indiquer que le CSS doit être appliqué aux écrans (la destination)
2. `(max-width: 800px)` pour indiquer que le CSS s'applique uniquement lorsque la taille de la fenêtre est inférieure à 800 pixels.

Ces deux conditions sont reliées par l'opérateur `and` (« et » en anglais) afin de spécifier que les règles comprises entre les accolades s'appliquent lorsque le site est affiché sur un écran ET que la taille de la fenêtre fait moins de 800 pixels.

Il est possible d'employer les opérateurs suivants au sein d'une media query :

- `and` : et (opérateur de combinaison des conditions)
- `only` : seulement (opérateur de sélection)

- `not` : non (opérateur de restriction)

Note : Il est recommandé d'utiliser l'opérateur `only` pour une meilleure compatibilité avec les anciens navigateurs.

Vous vous demandez peut-être comment votre site peut-il s'afficher autrement que sur un écran (`screen`) ? Les mots-clés suivants peuvent être utilisés en tant que destination :

- `screen` : styles pour écrans (ordinateurs, tablettes, téléphones...)
- `print` : styles d'impression
- `all` : styles pour toutes les destinations

Pour résumer, voici la syntaxe générale d'une media query :

```
@media [not ou only ou rien] [destination] and  
(condition1) [and (condition2) etc.] {  
    /* Code CSS */  
}
```

## Les conditions des media queries

En ce qui concerne les conditions liées aux largeurs d'écrans vous pouvez utiliser les mots-clés suivants :

- `max-width` : Largeur maximale de la fenêtre (le code s'appliquera sous le nombre de pixels indiqués)
- `min-width` : Largeur minimale de la fenêtre (le code s'appliquera au dessus du nombre de pixels indiqués)

*Il existe d'autres mots-clés pour ajouter d'autres conditions comme la hauteur, la couleur et même*

*l'orientation de l'appareil mais cela correspond à une utilisation plus avancée des media queries.*

Notez que pour que les mots-clés `max-width` et `min-width` fonctionnent sur un maximum d'appareils, votre site doit posséder la balise suivante dans sa section `head` :

```
<meta name="viewport" content="width=device-width,  
initial-scale=1">
```

Pour vous en dire plus, il faut savoir que les smartphones possèdent chacun un mode d'affichage différent.

Par là je veux dire que 1000 pixels de large sur un ordinateur quelconque ne correspond pas forcément à 1000 pixels sur le navigateur web de votre iPhone ou téléphone Android...

La balise `meta viewport` permet de dire au navigateur mobile que le nombre de pixels de l'écran doit correspondre au nombre de pixels du navigateur et que le zoom doit être réinitialisé.

Cette configuration fera en sorte que vos media queries fonctionnent sur un maximum d'appareils.

## **Exemples de media queries**

Pour vous guider dans le monde des media queries, voici quelques exemples accompagnés de commentaires décrivant leur finalité.

```
/* Styles destinés uniquement aux écrans (mot-clé "only")
inférieurs à 1024 pixels de large */
@media only screen and (max-width: 1024px){
    /* Code CSS */
}

/* Styles destinés à l'impression */
@media print{
    /* Code CSS */
}

/* Styles destinés aux écrans plus larges que 500 pixels
et moins larges que 1000 pixels */
@media screen and (min-width: 500px) and (max-width:
1000px){
    /* Code CSS */
}

/* Styles destinés uniquement aux écrans plus larges que
800 pixels */
@media only screen and (min-width: 800px){
    /* Code CSS */
}
```

Pour tester la « responsivité » d'un site, vous pouvez utiliser le service **Piresponsive**.

Concluons maintenant ce chapitre consacré au CSS avec...

# 7 bonnes pratiques CSS à adopter immédiatement

Jusqu'à présent nous avons vu beaucoup de choses et j'aimerais vous donner des précisions sur certains points.

Cela va vous permettre d'écrire du code CSS optimisé et vous éviter quelques maux de tête :)

Commençons avec la première bonne pratique. Il s'agit de...

## 1. Combinez les sélecteurs

Afin de ne pas écrire du code redondant, il est possible de grouper des sélecteurs afin que les mêmes propriétés soient appliquées à un ensemble de sélecteurs.

Par exemple, le code suivant :

```
h1{
    color:blue;
}
h2{
    color:blue;
}
h3{
    color:blue;
}
h4{
    color:blue;
}
```

Peut devenir :

```
h1, h2, h3, h4{  
    color:blue;  
}
```

C'est bien plus simple n'est-ce pas ?

Tout ce que vous avez à faire, c'est de séparer vos sélecteurs par des virgules.

## 2. Créez des sélecteurs simples

Vous souvenez-vous de ce dont nous avons parlé concernant la spécificité des sélecteurs ?

Nous avons vu que plus un sélecteur est spécifique, plus il possède d'importance. Nous avons même abordé une méthode qui donne un poids de 100 aux identifiants, 10 aux classes et 1 aux autres éléments.

D'un côté, il faut qu'un sélecteur soit précis pour que les propriétés qui lui sont associées s'appliquent mais d'un autre côté, des sélecteurs trop précis pourront vous porter préjudice car :

- Cela augmente la complexité de votre feuille de style (elle sera plus difficile à lire).
- Il sera plus compliqué de spécifier davantage en cas de besoin.
- Votre feuille de style aura une taille plus importante et sera donc plus longue à charger.

Remplacez donc ce genre de code :

```
body div#content p{  
    font-size:14px;  
}
```

par :

```
#content p{  
    font-size:14px;  
}
```

En effet, les éléments `body` et `div` n'apportent pas grand chose en terme de spécificité par rapport à l'identifiant `#content`.

### 3. Utilisez des commentaires

Même si cela peut prendre un peu de place, il est toujours utile d'insérer quelques commentaires pour expliciter votre code.

Vous verrez que cela est particulièrement utile lorsque vous modifierez le code de votre thème plusieurs semaines ou mois après l'avoir écrit.

Vous serez même reconnaissant envers l'auteur du thème s'il a inclus des commentaires pour vous signaler que telles instructions CSS s'appliquent à l'en-tête, au contenu, au portfolio, etc.

Par exemple, cela peut donner :

```
/* Sidebar */  
.sidebar{  
  
}  
  
/* Articles seuls*/  
.post{  
  
}
```

N'allez tout de même pas dans l'extrême en commentant chaque règle CSS comme vous avez pu le voir dans les exemples de ce chapitre.

## 4. Indentez correctement votre code

Tout le code CSS que vous avez pu consulter dans Relooker son Thème est indenté.

Que cela signifie-t-il ?

**Un code indenté est rédigé de manière à ce qu'il soit lisible par des humains** (c'est à dire par vous et moi).

Votre navigateur peut tout à fait lire du code compressé car c'est un programme informatique.

Nous sommes un peu plus long à la détente et avons besoin de clarté pour bien comprendre le code (que ce soit en CSS ou dans un autre langage).

Alors comment indenter votre code CSS ?

Voici une instruction générique pour vous servir de modèle :

```
.monselecteur{
    propriété1:valeur1;
    propriété2:valeur2;
    propriété3:valeur3;
    /* Etc. */
}
```

Décortiquons cette instruction. Nous avons :

- Le sélecteur et la première accolade
- À chaque règle CSS doit correspondre une ligne débutant par une tabulation (ou 4 espaces)

- L'accolade fermante au tout début d'une nouvelle ligne

Avouez que c'est beaucoup plus lisible que ce code :

```
.monselecteur{propriété1:valeur1;propriété2:valeur2;  
propriété3:valeur3;/* Etc. */}
```

Imaginez un fichier CSS composé de centaines d'instructions rédigées ainsi avec des dizaines de propriétés. Gare aux maux de tête.

Note : Si jamais vous tombez sur ce genre de fichier, [ce site](#) pourra vous aider à l'indenter.

## 5. Factorisez vos classes

Derrière ce mot qui doit vous rappeler vos cours de mathématiques se cache un concept tout simple : ne pas réinventer la roue.

En effet, admettons que vous avez besoin de créer des styles de boutons de différentes couleurs et de différentes tailles.

Au lieu d'utiliser quasiment le même code dans chaque classe associée à un bouton, pourquoi ne pas créer une classe bouton contenant les règles principales et d'autres classes contenant toutes les spécificités ?

Voilà ce que ça donnerait :

```
.bouton{
    padding:10px 15px;
    font-size:14px;
    color:white;
    cursor:pointer; /* Affichage du curseur sous forme de
main */
}

.bouton-bleu{
    background-color:blue;
}

.bouton-rouge{
    background-color:red;
}
```

En HTML, le code de vos boutons serait le suivant :

```
<a href="#" class="bouton bouton-bleu">Mon bouton bleu</a>
```

Cette méthode permet d’avoir un code plus léger et donc plus lisible. Vous pouvez appliquer cela à tous les éléments qui se différencient que par quelques règles (boutons, titres, colonnes de contenu, etc.)

## 6. Servez-vous des propriétés génériques

Rappelez-vous de ce que disait ma professeur de mathématiques : « Un bon matheux, c’est un bon feignant ».

Il ne faut pas se compliquer la vie, lorsque l’on peut faire les choses simplement. Pensez donc à utiliser les « meta propriétés » lorsque cela est pertinent.

Concrètement, utilisez :

- font au lieu de font-family, font-size, font-style, font-weight, etc.
- background au lieu de background-color, background-image, background-position, etc.
- border au lieu de border-width, border-style et border-color.
- margin au lieu de margin-top, margin-right, margin-bottom et margin-left.
- padding au lieu de padding-top, padding-right, padding-bottom et padding-left.
- Etc.

Parfois, appliquer une propriété spécifique s'avère plus pratique (et plus lisible) donc ne vous en privez pas non plus. C'est un équilibre à trouver.

Ce conseil à pour but de vous éviter d'écrire des lignes de code qui pourraient être condensées.

## **7. N'appliquez le suffixe !important qu'en dernier recours**

Vous avez déjà pu lire cette recommandation dans le chapitre mais je tiens à le répéter.

Avant d'utiliser le suffixe !important sur une règle CSS, vérifiez qu'il n'est pas possible de spécifier davantage le sélecteur de votre instruction.

Retenez qu'il est important de ne pas utiliser !important :)

## **Prêt pour le test ?**

Bravo, vous venez de découvrir les bases du langage CSS. Que de chemin parcouru depuis le début de ce chapitre !

À présent, il est temps de voir ce que vous avez retenu grâce au test de fin de chapitre. Bien sûr vous pouvez garder le guide à portée de main :)

**RELOOKER SON THÈME**

# **CHAPITRE 4**

## **COMMENT FONCTIONNE UN THÈME WORDPRESS**

---

**Assimilez la logique de WordPress,  
comprenez comment se structurent  
les thèmes et appréhendez  
le langage PHP**

## Chapitre 4.1

# La structure de fichiers d'un thème WordPress

Vous l'avez probablement déjà constaté, un thème WordPress se compose de plusieurs fichiers.

À quoi correspondent-ils ? Qu'y a-t-il à l'intérieur ? Comment fonctionnent-ils ? C'est ce que nous allons découvrir dans cette première partie.

## De quoi se compose un thème WordPress ?

Lorsque vous téléchargez un thème (premium ou gratuit peu importe), vous allez retrouver un certain nombre de fichiers fondamentaux.

Pour n'en citer que quelques-uns, vous aurez :

- `header.php`
- `footer.php`
- `sidebar.php`
- `index.php`
- `single.php`
- `page.php`
- `archive.php`
- `comments.php`
- `functions.php`
- `style.css`

Vous vous demandez probablement : « Que sont ces fichiers PHP ? » et « Où sont les fichiers HTML ? ».

Pour vous répondre, **le PHP est le langage de programmation que WordPress utilise pour fonctionner**. On le retrouve donc au sein des thèmes WordPress.

Le PHP va permettre à votre thème WordPress de récupérer et d'afficher des informations à partir de votre base de données (contenu des articles, des pages ou encore des options, etc.).

Dans la partie suivante, nous étudierons les bases de PHP pour vous permettre de faire vos premières modifications au sein de votre thème WordPress.

Concernant le code HTML, rassurez-vous il est toujours là. Il est juste situé dans les fichiers PHP dont je viens de vous parler.

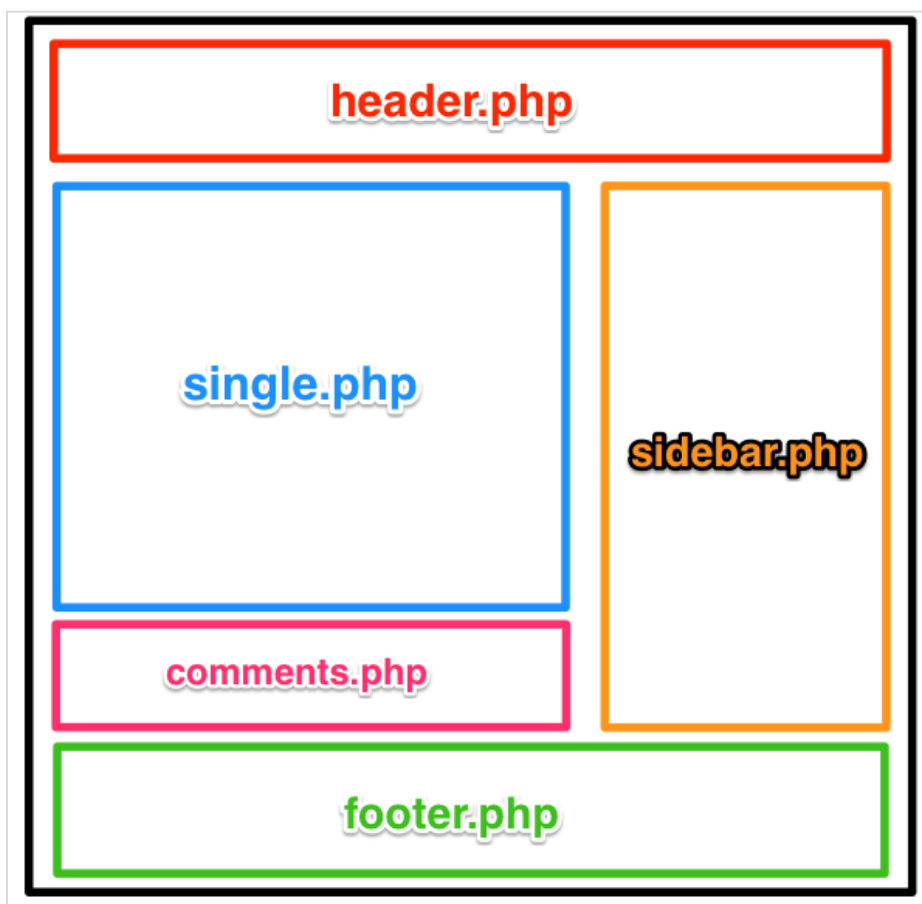
Rappelez-vous du petit site que vous avez créé au cours des tests de connaissance des deux précédents chapitres.

Chacune des pages est constituée d'un seul fichier (accueil, blog, article et contact). Pour un thème WordPress, c'est différent : **les pages sont découpées en plusieurs morceaux et sont ensuite reconstituées à la manière d'un puzzle** à chaque fois qu'un visiteur cherche à afficher une page.

Par exemple, la page qui affiche un article utilise :

- `header.php` qui contient toute la section head et l'en-tête du site
- `single.php` qui correspond au contenu de l'article
- `comments.php` qui affiche les commentaires
- `sidebar.php` qui affiche la barre latérale
- `footer.php` qui contient le pied de page

On peut résumer cela avec l'image suivante :



Pourquoi un tel enchevêtrement de fichier ? Bien que cela puisse paraître compliqué à première vue, cette méthode simplifie beaucoup de choses.

Souvenez-vous de ce que vous avez appris avec l'utilisation d'un seul fichier CSS. Le style est séparé de la structure pour que les modifications de style se répercutent sur tout le site.

Eh bien il s'agit de la même chose avec les thèmes WordPress.

Imaginez que vous désirez modifier les icônes des réseaux sociaux et que vous devez le faire sur chaque page de votre site. Nous sommes d'accord pour dire que cela peut s'avérer fastidieux si votre site possède des dizaines voire des centaines de pages...

Si vos icônes sont placées dans l'en-tête de votre thème WordPress, il vous suffira de modifier le fichier `header.php` et le tour sera joué.

Vos nouvelles icônes s'afficheront immédiatement sur toutes les pages de votre site. En effet, chaque page du site appelle le fichier `header . php`.

Ainsi chaque thème WordPress est constitué de plusieurs fichiers qui sont assemblés à chaque fois qu'un visiteur demande à afficher une page de votre site.

*Tous les fichiers ne sont donc pas utilisés systématiquement, cela dépend du type de page demandé par le visiteur.*

N'importe quel thème fonctionne sur ce principe. Après il est possible qu'il utilise des fichiers supplémentaires en fonction de sa complexité, cela dit le principe reste le même.

Examinons maintenant...

## **Le rôle des fichiers d'un thème**

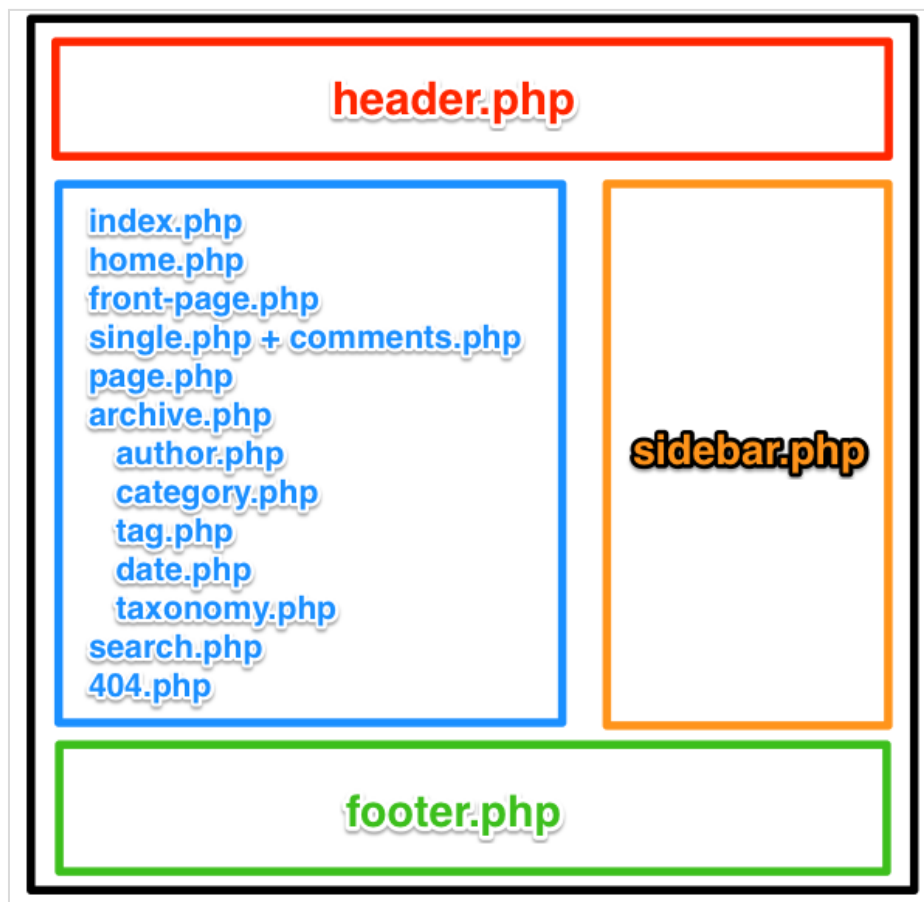
Nous avons abordé brièvement le nom de certains fichiers pouvant figurer dans un thème WordPress mais il peut y en avoir d'autres.

Voici une liste un peu plus complète (mais non exhaustive) des fichiers que vous pourrez trouver dans un thème WordPress et ce qu'ils contiennent :

- `header . php` : section head et l'en-tête HTML du thème
- `footer . php` : pied de page
- `sidebar . php` : barre latérale où l'on peut trouver les widgets
- `single . php` : afficher un article seul
- `page . php` : afficher une page
- `singular . php` : afficher une publication seule (article, page ou autre)
- `front - page . php` : afficher la page d'accueil
- `home . php` : afficher les articles en page d'accueil si `front - page . php` n'existe pas

- `404.php` : afficher la page d'erreur
- `search.php` : afficher les résultats de recherche
- `archive.php` : afficher le contenu demandé si les fichiers suivants ne sont pas présents
- `author.php` : afficher les articles d'un auteur
- `category.php` : afficher les articles d'une catégorie
- `tag.php` : afficher les articles d'une étiquette (nouveau nom des mots-clés)
- `date.php` : afficher les articles correspondant à une date (jour, mois ou année)
- `taxonomy.php` : afficher les articles d'une taxonomie
- `comments.php` : afficher les commentaires
- `functions.php` : fichier servant à ajouter des fonctionnalités (toujours chargé)
- `index.php` : affiche la page demandée en dernier recours si les fichiers nécessaires ne sont pas présents
- `style.css` : Styles CSS associés au thème (toujours chargé)

Nous sommes d'accord, il y a pas mal de fichiers... L'image ci-dessous permet de visualiser comment chacun d'eux sont utilisés :



Je n'ai pas pu inclure les fichiers `style.css` et `functions.php` mais **sachez qu'ils sont chargés systématiquement dans n'importe quel thème WordPress.**

Cette représentation n'est pas parfaite, j'en conviens. Il y a toujours des exceptions.

Par exemple, la page d'accueil (ou une autre page) peut ne pas avoir de barre latérale. Dans ce cas, `sidebar.php` ne figurerait pas dans le schéma.

*L'auteur d'un thème peut inclure d'autres fichiers pour apporter des fonctionnalités supplémentaires ou inclure de nouveaux fichiers pour que la modification du contenu soit plus aisée.*

Vous avez maintenant une meilleure idée du rôle des fichiers se trouvant dans un thème WordPress. Voyons maintenant...

# Comment fonctionnent les fichiers d'un thème WordPress

Lorsque vous êtes dans l'administration de votre site (dans le répertoire *wp-admin*) vous pouvez publier des articles, des pages et parfois d'autres types de contenus.

En appuyant sur « Publier » ou « Enregistrer », **ce que vous avez entré dans l'éditeur de texte est envoyé dans la base de données de votre site.**

Ensuite, quand un visiteur vient sur votre site, son navigateur envoie une requête à votre serveur qui, par l'intermédiaire de WordPress (et PHP), va chercher le contenu demandé dans la base de données.

Les fichiers PHP nécessaires à l'affichage de la page sont chargés et le contenu précédemment récupéré est intégré aux bons endroits. Cela génère ensuite une page en HTML et CSS qui est envoyée au navigateur par le serveur pour que le visiteur puisse la voir.

Bien sûr, cela se passe extrêmement vite. Il est tout de même utile de comprendre comment WordPress affiche une page.

Revenons aux fichiers des thèmes WordPress (*index.php*, *single.php*, *page.php*, etc.). Comme cela a été indiqué, ils ne sont pas tous utilisés pour le chargement de chaque page.

En fait, tout dépend du type de la page demandée par le visiteur.

Si un visiteur clique sur le lien d'une catégorie, la liste des derniers articles de cette catégorie va s'afficher (via le fichier *category.php*).

Il est logique que le fichier *single.php* (qui sert à afficher tout le contenu d'un article) ne soit pas utilisé sur la page d'une catégorie car l'agencement peut ne pas être le même (par exemple, les images à la une peuvent ne pas s'afficher de la même façon).

On peut visualiser les fichiers destinés à être appelés par WordPress grâce à...

## La hiérarchie des templates

Il y a effectivement une logique derrière les fichiers chargés par le thème lorsqu'une page est demandée. Nous avons vu que le fichier correspondant à un type de page est utilisé par le thème.

Mais comment faire s'il n'existe pas ?

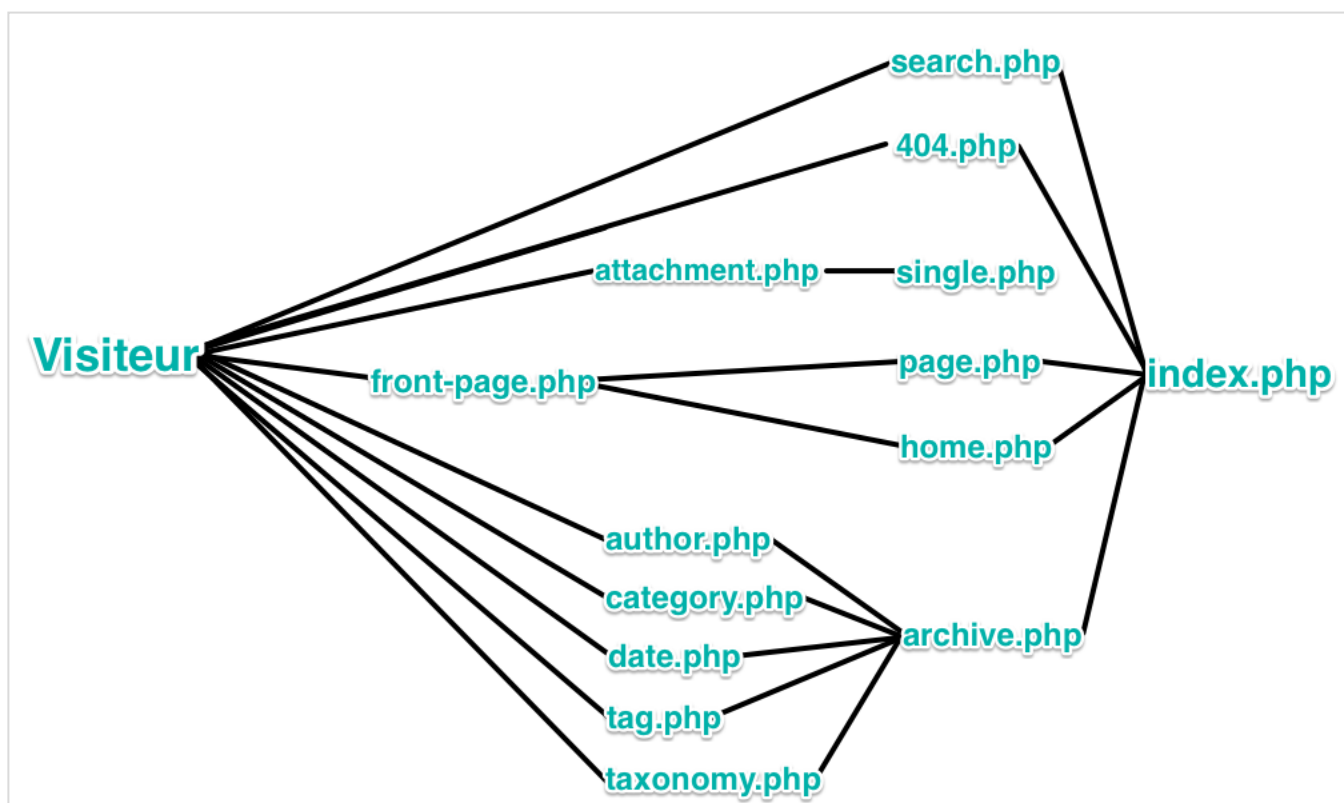
Pour reprendre l'exemple d'une page catégorie, si `category.php` n'existe pas, WordPress va essayer de charger `archive.php`, un fichier plus générique mais qui peut tout de même faire le travail.

En dernier recours, si `archive.php` n'existe pas, WordPress utilisera `index.php` pour afficher la page catégorie.

*La logique est faite de manière à ce qu'il y ait toujours un fichier de secours pour afficher la page demandée. En l'occurrence, ce fichier est `index.php`.*

**Le thème utilisera toujours le fichier le plus haut dans la hiérarchie s'il existe.**

Voici comment on peut résumer simplement cette hiérarchie :



Il faut lire ce schéma de gauche à droite. Si le type de page demandé est absent, WordPress essaiera de charger le suivant et à défaut, il utilisera `index.php`.

Vous retrouverez bien ce que nous avons dit pour l’affichage d’une page catégorie.

Si vous êtes attentif, vous devez avoir remarqué qu’un nouveau fichier figure sur le schéma. Il s’agit de `attachment.php`.

Ce fichier sert à afficher un média lié à un article (généralement une image). Cela correspond à la « Page du fichier attaché » lorsque vous insérez une image dans une publication.

Après avoir examiné ce schéma et son fonctionnement, on peut naturellement se demander si WordPress pourrait fonctionner en n’utilisant que `index.php`.

Eh bien oui, c’est le cas. En réalité, **un thème WordPress peut se composer de deux fichiers au minimum** : `index.php` et `style.css`.

En pratique, il vaut mieux utiliser plus de fichiers pour gérer la mise en page de son thème plus finement.

Ne nous arrêtons pas ici, en réalité la hiérarchie des templates est un peu plus élaborée que le schéma précédent. Enfin rassurez-vous, la logique est exactement la même.

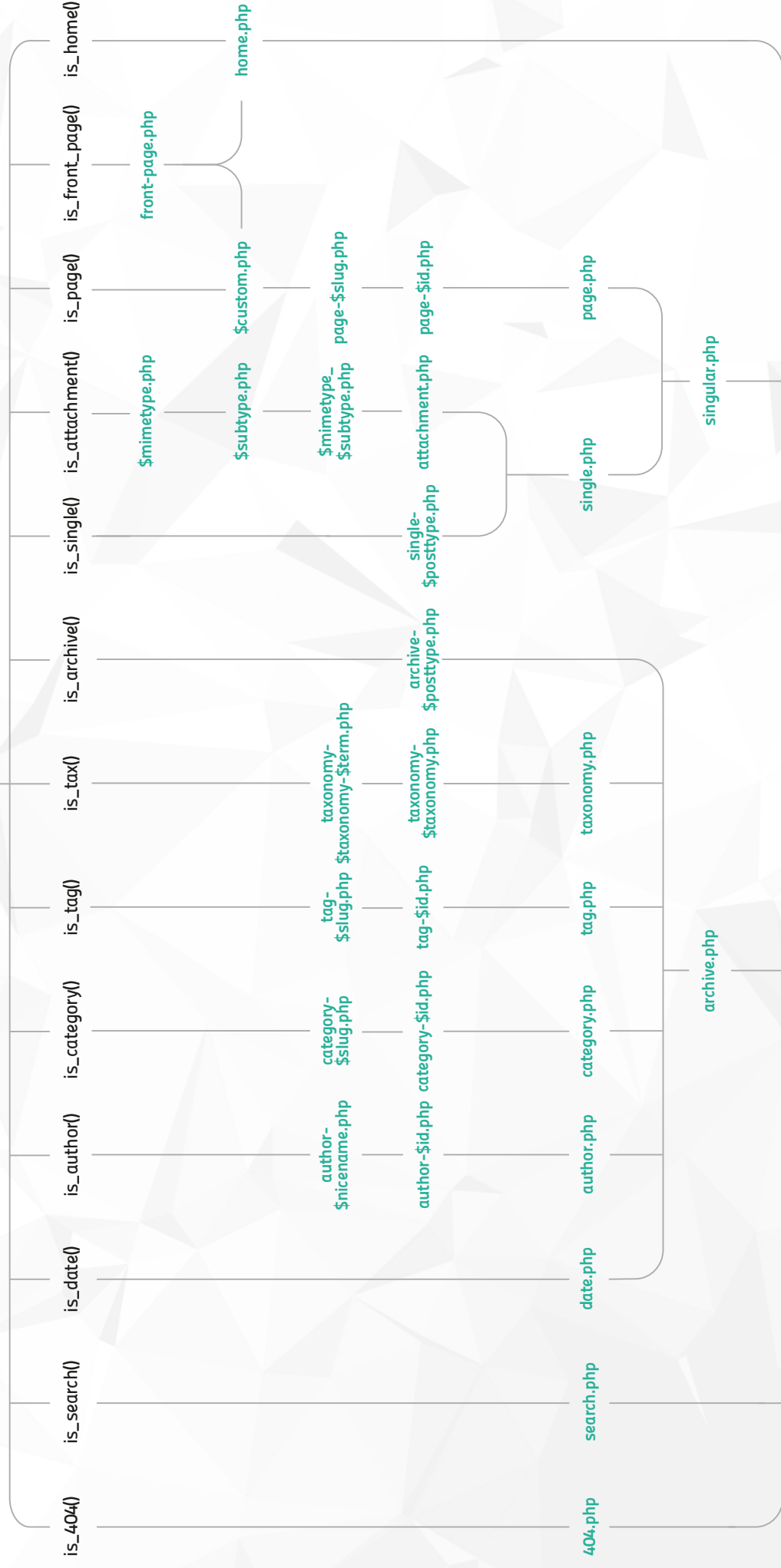
## **La hiérarchie des templates en intégralité**

J'ai préféré ne pas vous présenter l'intégralité de la hiérarchie des templates de WordPress d'entrée de jeu pour ne pas vous submerger d'informations.

L'idée était de vous faire comprendre le principe puis d'approfondir. Il est maintenant temps de découvrir le schéma en totalité :

# WORDPRESS : LA HIÉRARCHIE DES TEMPLATES

Quelle type de page ?



Comme vous pouvez le voir, le schéma est un peu plus chargé, de nouveaux fichiers se sont glissés dans la hiérarchie des templates.

Parlons de chacun d'eux en détail.

Note : Le schéma ci-dessus est disponible dans le dossier Ressources fourni avec Relooker son Thème.

## **Affichage des publications seules**

Nous avons vu que `single.php` est utilisé pour afficher les articles seuls mais il est possible de spécifier davantage ou au contraire, d'être plus générique.

Voici quels fichiers WordPress cherchera à charger pour afficher des publications individuellement selon leur type :

- `single-[posttype].php` : sera utilisé pour afficher les articles d'un type particulier. Si vous avez un type de contenu `portfolio`, vous pourrez utiliser `single-portfolio.php`. `single-post.php` sera utilisé pour afficher exclusivement un article classique
- `single.php` : fichier générique servant à afficher n'importe quel type de publication seule (sauf les pages)
- `singular.php` : sera utilisé pour n'importe quel type de publication seule (pages incluses)
- `index.php` : sera utilisé si aucun des fichiers précédents ne sont présents

## Affichage des fichiers attachés

Il est plutôt rare de trouver ces fichiers de templates au sein des thèmes. Il est toutefois utile de les connaître si vous avez besoin d'un affichage particulier pour un type de fichier précis.

- `[mimetype].php` : sera utilisé pour des fichiers d'un type spécifique. Par exemple il peut y avoir `image.php`, `video.php` ou encore `text.php`.
- `[subtype].php` : sera utilisé pour afficher des fichiers d'un sous-type spécifique. Ainsi vous pourrez utiliser `png.php` pour une image au format PNG ou `zip.php` pour une archive au format ZIP.
- `[mimetype]_[subtype].php` : sera utilisé pour afficher des médias de nature particulière. Par exemple : `image_png.php` pour une image au format PNG ou `application_pdf.php` pour un fichier PDF ([cliquez ici](#) pour voir la liste complète des mime-types et subtypes).
- `attachment.php` : fichier générique servant à afficher un média associé à un article (images, fichiers PDF, Word, etc.).
- `single.php` : sera utilisé pour n'importe quel type de publication seule (sauf les pages)
- `singular.php` : sera utilisé pour n'importe quel type de publication seule (pages incluses)
- `index.php` : sera utilisé si aucun des fichiers précédents ne sont présents

## Affichage des pages

Comme pour les articles, il est possible de cibler plus finement l'apparence des pages grâce aux fichiers de templates suivants :

- `[custom].php` : WordPress vous permet de créer vos propres modèles de page et de les attribuer à une page (nous allons aborder cela par

la suite). Le choix du nom de ces modèles de page est libre (*custom* signifie *personnalisé*).

- `page - [slug] . php` : le slug correspond à l'identifiant texte que l'on donne à une page (que l'on retrouve dans l'adresse de la page). Par exemple pour la page qui possède le slug `contact`, on peut créer le fichier `page - contact . php`.
- `page - [id] . php` : dans WordPress, chaque page possède un identifiant numérique (id). En créant un fichier nommé `page - 3 . php`, la page d'identifiant 3 n'utilisera pas la mise en page par défaut de `page . php` mais celle que vous aurez définie.
- `page . php` : fichier générique servant à afficher les pages
- `singular . php` : sera utilisé pour n'importe quel type de publication seule (pages incluses)
- `index . php` : sera utilisé si aucun des fichiers précédents ne sont présents

## **Affichage des pages auteurs**

- `author - [nicename] . php` : au lieu de prendre l'identifiant numérique, on se base sur l'identifiant texte d'un auteur. Exemple : `author - admin . php`. Attention, l'identifiant texte ne correspond pas toujours à l'identifiant de connexion
- `author - [id] . php` : affichera une page particulière en fonction de l'identifiant numérique de l'auteur, par exemple `author - 1 . php`
- `author . php` : fichier générique servant à afficher les pages auteurs
- `archive . php` : sera utilisé pour afficher n'importe quel type d'archive
- `index . php` : sera utilisé si aucun des fichiers précédents ne sont présents

## Affichage des pages de catégories

- `category-[slug].php` : Affichera les articles d'une catégorie selon son identifiant texte (slug). Exemple : `category-voyages.php`.
- `category-[id].php` : Affichera les articles d'une catégorie en fonction de son identifiant numérique (id), par exemple `category-4.php`.
- `category.php` : fichier générique servant à afficher les pages de catégories
- `archive.php` : sera utilisé pour afficher n'importe quel type d'archive
- `index.php` : sera utilisé si aucun des fichiers précédents ne sont présents

## Affichage des pages d'étiquettes

- `tag-[slug].php` : Affichera les articles relatifs à une étiquette (nouveau nom des mots-clés) selon son identifiant texte (slug). Exemple : `tag-france.php`.
- `tag-[id].php` : Affichera les articles relatifs à une étiquette en fonction de son identifiant numérique (id), par exemple `tag-8.php`.
- `tag.php` : fichier générique servant à afficher les pages d'étiquettes
- `archive.php` : sera utilisé pour afficher n'importe quel type d'archive
- `index.php` : sera utilisé si aucun des fichiers précédents ne sont présents

## Affichage des pages de taxonomies

Une taxonomie est un moyen de classer les articles ou les types des contenus personnalisés (appelés CPT en anglais pour Custom Posts Types).

*Les deux taxonomies par défaut de WordPress sont les catégories et les étiquettes (nouveau nom des mots-clés).*

Il est possible d'en créer d'autres en installant des extensions ou en créant une extension (certains thèmes en contiennent aussi).

Pour toutes les taxonomies à l'exception des catégories et des étiquettes, WordPress utilisera le fichier `taxonomy.php`. D'autres fichiers peuvent être créés pour avoir un ciblage plus précis.

- `taxonomy-[taxonomy]-[term].php` : Affichera les contenus relatifs à un terme sous-jacent d'une taxonomie. Par exemple, si vous avez une classification "typeprojet" et que vous faites des logos, vous pourrez utiliser le fichier `taxonomy-typeprojet-logo.php` pour lister les projets étant dans « Logo ».
- `taxonomy-[taxonomy].php` : Affichera les contenus relatifs à une taxonomie particulière. Exemple : `taxonomy-typeprojet.php` si l'on a créé une classification "typeprojet" pour un type de contenu « portfolio ».
- `taxonomy.php` : fichier générique servant à afficher les pages de taxonomies
- `archive.php` : sera utilisé pour afficher n'importe quel type d'archive
- `index.php` : sera utilisé si aucun des fichiers précédents ne sont présents

## **Affichage des pages de date**

WordPress permet d'utiliser une mise en page particulière pour les pages archives datées. Même si cela est peu utilisé au sein des thèmes, cela est tout de même possible.

- `date.php` : sera utilisé pour afficher les pages d'archives datées (par jour, mois ou année)
- `archive.php` : sera utilisé pour afficher n'importe quel type d'archive
- `index.php` : sera utilisé si aucun des fichiers précédents ne sont présents

## Affichage des pages d'archives

Les fichiers relatifs aux auteurs, aux catégories, aux étiquettes, aux dates et aux taxonomies sont tous prioritaires par rapport à `archive.php`.

Il reste cependant un fichier de template d'archive dont nous n'avons pas encore parlé. Il s'agit de :

- `archive-[post type].php` : Ce fichier permettra de définir une mise en page particulière pour les archives d'un type de contenu personnalisé. Par exemple, si vous avez un type de contenu « portfolio ». Vous pouvez créer le fichier `archive-portfolio.php` pour lister les archives de votre portfolio.
- `archive.php` : sera utilisé pour afficher n'importe quel type d'archive
- `index.php` : sera utilisé si aucun des fichiers précédents ne sont présents

## Affichage de la page d'accueil

Concernant la page d'accueil, deux cas peuvent se présenter. Par défaut, WordPress affiche la liste des derniers articles sur la page d'accueil.

Le fichier de template en charge de personnaliser l'apparence de la page des articles est `home.php` :

- `home.php` : sera utilisé pour afficher la page des articles
- `index.php` : sera utilisé si `home.php` est absent

Note : `home.php` sera utilisé pour agencer la page des derniers articles qu'elle soit en page d'accueil ou non (voir dans Réglages > Lecture)

Si un thème possède un fichier `front - page . php`, celui-ci sera utilisé en priorité pour afficher la page d'accueil.

- `front - page . php` : sera utilisé pour afficher la page d'accueil en priorité
- `home . php` : sera utilisé si `front - page . php` est absent et que le site est paramétré pour afficher les derniers articles en page d'accueil
- `page . php` (ou les fichiers de templates supérieurs) : seront utilisés pour afficher la page définie en tant que page d'accueil dans *Réglages > Lecture*
- `index . php` : sera utilisé si aucun des fichiers précédents ne sont présents

## **Affichage de la page d'erreur**

- `404 . php` : sera utilisé pour afficher la page d'erreur
- `index . php` : sera utilisé si `404 . php` est absent

## **Affichage des pages de résultats de recherche**

- `search . php` : sera utilisé pour afficher les pages de résultats de recherche
- `index . php` : sera utilisé si `search . php` est absent

Voilà, cette fois nous avons parcouru tous les fichiers de template relatifs à la hiérarchie des templates.

*Comme nous l'avons vu, ils sont tous optionnels. Chaque auteur de thème utilise les fichiers de template qui lui semblent nécessaires.*

En revanche, vous êtes libre de créer de nouveaux fichiers de template pour apporter des mises en page supplémentaires à un thème.

**L'important ici n'est pas de retenir tous les fichiers de template mais de connaître comment la hiérarchie des templates fonctionne.**

## **Autres fichiers pouvant être présents dans un thème WordPress**

Pour finir cette première partie, attardons-nous sur quelque chose qui risque fort d'arriver dans les thèmes que vous serez amené à relooker.

*D'autres fichiers seront présents et n'auront rien à voir avec ceux dont nous aurons parlé jusqu'à présent.*

### **Fichiers PHP**

Les auteurs de thèmes WordPress intègrent généralement d'autres fichiers PHP en plus des fichiers de template et fichiers de structure (header . php, etc.).

Ces fichiers peuvent posséder plusieurs rôles :

- **Fichiers renfermant des shortcodes, des widgets, des déclarations de types de contenus personnalisés, de taxonomies, etc.**

Les bonnes pratiques de WordPress recommandent plutôt de placer ce genre de choses dans des extensions mais vous pourrez en trouver dans des thèmes.

- **Fichiers renfermant des fonctionnalités nécessaires au thème.**

Ils sont généralement placés dans des dossiers pouvant se nommer framework, inc, includes, admin ou d'autres noms exotiques.

Ces fichiers peuvent contenir le code permettant d'afficher des pages d'options et d'autres fonctions liées au thème.

Il ne vaut mieux pas toucher à ces fichiers lorsqu'on modifie un thème (sauf si l'on sait ce que l'on fait).

- **Fichiers décomposant les fichiers de template existants.**

Comme `header.php` et `sidebar.php` permettent de décomposer la structure d'un thème, d'autres fichiers peuvent servir à constituer un fichier de template.

On peut imaginer que `single.php` puisse appeler le fichier `content.php` pour afficher le contenu d'un article classique ou `content-video.php` pour afficher le contenu d'un article de format vidéo.

Nous allons voir dans la suite de ce chapitre que des fichiers PHP peuvent en appeler d'autres. Un auteur peut donc « découper » son thème en plusieurs petits fichiers pour que ce soit plus simple à gérer.

## **Fichiers images**

Un thème WordPress peut avoir besoin d'images pour s'afficher correctement (images d'arrière-plan, icônes, etc).

Elles sont généralement situées dans un dossier nommé `img` ou `images`.

## **Fichiers JS**

Les fichiers possédant une extension en `.js` sont des fichiers Javascript. Le Javascript est un langage informatique utilisé par beaucoup de sites internet. Il permet d'apporter des fonctionnalités supplémentaires et notamment de l'interactivité à un site.

Par exemple si votre thème possède un diaporama (également appelé slider ou slideshow), il y a de fortes chances qu'il utilise le javascript pour le faire défiler.

Les fichiers Javascript peuvent être placés dans un dossier `js` ou dans d'autres dossiers, il n'y a pas vraiment de règle.

Note : Nous n'aborderons pas la modification des fichiers Javascript dans Relooker son Thème.

## Fichiers de traduction

Les thèmes WordPress contiennent généralement des fichiers de traductions. Ils possèdent les extensions `.po` et `.mo` (et parfois `.pot`) et sont habituellement situés dans un dossier `languages` ou `lang`.

*Si vous cherchez à traduire votre thème en français, je vous recommande [cet article de WP Marmite](#).*

## Fichiers de polices

Dans certains cas, des thèmes peuvent embarquer des polices d'écriture. Des bibliothèques d'icônes peuvent aussi être intégrées comme des polices.

On peut notamment citer FontAwesome, Typicons, Icomoon, etc.

Ces polices se trouvent souvent dans un dossier `font s` mais cela peut varier.

Vous les reconnaîtrez grâce aux extensions de fichiers `.eot`, `.svg`, `.ttf` et `.woff`.

## Récapitulons

Au cours de cette première partie, nous avons pu voir qu'un thème WordPress peut embarquer un nombre variable de fichiers. Tout dépend de la structure que l'auteur a voulu lui attribuer.

Vous pouvez retenir que :

- Certains fichiers portent un nom précis car ils possèdent un rôle particulier au sein du thème.
- Bien qu'un thème peut fonctionner avec seulement deux fichiers, on retrouve généralement les mêmes fichiers de base dans tous les thèmes.
- Des fichiers complémentaires peuvent figurer dans un thème pour lui apporter des fonctionnalités spécifiques.

Dans la partie suivante, nous allons nous attarder sur le langage PHP. Rappelez-vous, il s'agit du langage web qu'utilise WordPress pour fonctionner.

Nous allons voir quelles sont les bases de ce langage et comment un thème l'utilise pour afficher son contenu.

Avant de poursuivre votre lecture, je vous invite à effectuer l'exercice associé à cette première partie du chapitre 4.

## Chapitre 4.2

# Introduction au langage PHP

Maintenant que vous avez une meilleure idée des fichiers que l'on peut trouver dans un thème WordPress, nous allons pouvoir regarder ce qui se trouve à l'intérieur.

Comme nous l'avons vu, en dehors du fichier `style.css` la majorité des fichiers d'un thème sont des fichiers PHP.

Pour rappel, le PHP est le langage qu'utilise WordPress pour fonctionner. Il est donc normal que l'on retrouve du code PHP au sein des thèmes WordPress.

Le but de cette partie est de vous enseigner les bases du PHP pour que vous puissiez mieux appréhender le contenu des fichiers composant les thèmes WordPress.

*Nous en avons déjà parlé, le but de Relooker son Thème n'est pas de vous transformer en programmeur web chevronné mais de vous transmettre les bases pour vous faire gagner en autonomie avec votre thème WordPress.*

Commençons par le commencement, avant de savoir courir il faut apprendre à marcher n'est-ce pas ?

## La structure des fichiers PHP

À la différence du HTML ou du CSS, du code PHP doit toujours figurer entre des balises PHP.

Voici à quoi elles ressemblent :

```
<?php // Balise ouvrante
```

```
/* CODE PHP */
```

```
?> // Balise fermante
```

*Important : La balise de fermeture n'est pas de la forme **php?>**.*

Dans le cas des fichiers contenant exclusivement du PHP (pas de code HTML), une balise PHP ouvrante est présente `<?php` mais la balise fermante n'est pas obligatoire. C'est le cas du fichier `functions.php`.

## Les commentaires

L'exemple ci-dessus présente du texte précédé de deux barres obliques `//` (slash en anglais) et encadré par les balises `/*` et `*/`.

**Cela correspond à des commentaires, c'est à dire du texte qui sera ignoré par PHP.**

Les commentaires sont très utiles pour comprendre le fonctionnement du code.

On peut écrire des commentaires de deux manières :

- Sur une seule ligne avec `//`
- Sur plusieurs lignes avec `/*` et `*/`

```
<?php /* Les balises de type "slash étoile" et  
"étoile slash" servent à commenter  
plusieurs lignes */
```

```
// Le double slash ne commente qu'une seule ligne  
?>
```

# Les instructions PHP

Le langage PHP est composé de plusieurs instructions qui, les unes à la suite des autres, permettent d'effectuer des traitements précis.

*Chaque instruction doit impérativement s'achever par un point-virgule : ;.*

En cas d'oubli, une erreur se produira et votre site ne s'affichera pas convenablement.

Par exemple, voici une instruction destinée à afficher du texte :

```
<?php  
echo "Bonjour, comment allez-vous ?";  
  
// Cela affichera "Bonjour, comment allez-vous ?" à  
l'écran (sans les guillemets)  
?>
```

## Les variables

Le PHP peut « se souvenir » de différentes informations, pour cela il les stocke en mémoire dans ce que l'on appelle des variables.

On les reconnaît facilement car elles débutent par le signe \$ (dollar).

Elles peuvent contenir toutes sortes de valeurs, du texte, un nombre, quelque chose de vrai ou faux, un tableau, etc.

```
<?php
/* Variable $prenom initialisée à Alexandre
    Remarquez la présence du point-virgule en fin
    d'instruction
*/
$prenom = "Alexandre";

// Variable $departement
// Attention, ne mettez pas d'accents ou de caractères
spéciaux dans les noms de vos variables
$departement = 10;

// Variable de type booléenne, la valeur est soit vraie,
soit fausse
$blogueur = true; // true = vrai, false = faux
?>
```

## Les constantes

Certains thèmes et extensions se servent de valeurs non variables dans leur code. On appelle cela des constantes.

On peut se servir de constantes pour stocker un numéro de version, un chemin vers un répertoire précis, etc.

Par définition, la valeur d'une constante ne peut être modifiée (sinon elle serait une variable).

Voici comment les constantes sont déclarées :

```
<?php
// Constante MON_THEME_VERSION initialisée à 1.0.0
define('MON_THEME_VERSION', '1.0.0');
?>
```

Le nom d'une constante doit comporter des lettres majuscules et éventuellement des soulignés (\_).

L'avantage des constantes est que l'on peut accéder à leur valeur à partir de n'importe où. Dans le thème, une extension ou une de leurs fonctions.

Voici le code à utiliser pour les utiliser :

```
<?php
// Affichage de la constante
echo MON_THEME_VERSION;
?>
```

Comme vous pouvez le voir, il suffit d'insérer le nom de la constante pour que PHP la remplace automatiquement par la valeur qui lui est associée.

## Les tableaux

Parfois, on peut avoir besoin de retenir plusieurs valeurs d'un coup. Le PHP nous permet de le faire grâce à des tableaux.

Imaginons que vous ayez besoin de stocker des noms de classes CSS dans un tableau. Voici ce que vous devez écrire :

```
<?php
$classes = array("classe_a", "classe_b", "classe_c");
?>
```

Si vous avez besoin d'afficher « une case » d'un tableau, vous pouvez utiliser l'instruction `echo` de la manière suivante :

```
<?php
echo $classes[0]; // Affichera classe_a
echo $classes[1]; // Affichera classe_b
echo $classes[2]; // Affichera classe_c
?>
```

Cela peut sembler étrange mais la première case d'un tableau est identifiée par le numéro 0 et ainsi de suite. Gardez cela à l'esprit lorsque vous travaillerez avec des tableaux.

Comme vous pouvez le voir, il faut indiquer le numéro de la case du tableau que l'on veut afficher entre des crochets.

Il existe aussi un autre type de tableau. Celui que nous venons d'étudier possède un index généré automatiquement. C'est à dire que `classe_a` est à la case 0, `classe_b` à la case 1, etc.

On peut définir un tableau doté de ses propres index de la façon suivante :

```
<?php
$pages = array(
    "home" => "Bienvenue sur l'accueil",
    "about" => "À Propos",
    "contact" => "Contactez-moi",
);
?>
```

Si l'on veut afficher le contenu de la case identifiée par `home`, il faut procéder de la même manière que le tableau précédent mais en fournissant l'index `home` au lieu de 0, c'est à dire :

```
<?php
echo $pages["home"];
// Cela va afficher : Bienvenue sur l'accueil
?>
```

## Les fonctions

PHP utilise aussi des fonctions pour accomplir des traitements. En fait, **une fonction est un ensemble d'instructions qui réalisent une ou plusieurs opérations précises.**

Cela évite ainsi la réécriture des mêmes instructions à plusieurs endroits. Appeler une fonction permet de réduire et simplifier le code.

Il existe deux types de fonctions :

- celles qui retournent une valeur
- celles qui ne retournent pas de valeur

Les fonctions peuvent recevoir des variables en paramètre. Ces variables pourront ensuite être utilisées pour effectuer les traitements désirés. **On appelle ces variables des arguments.**

Prenons un exemple simple, la fonction carrée. Lorsqu'on lui donne un nombre, la fonction carrée retourne ce nombre au carré, c'est à dire ce nombre multiplié par lui même (4 au carré est égal à 16).

En PHP, cela donne :

```

<?php
// Déclaration de la fonction
// $nombre doit être une variable de type numérique (c'est
le seul argument de la fonction)

function carre($nombre){
    // On retourne $nombre multiplié par $nombre
    return $nombre * $nombre;
}

// Création d'une variable $age et initialisation à 4
$age = 4;

// Utilisation de la fonction "carre" et affichage de son
résultat grâce à l'instruction "echo"
echo carre($age);
?>

```

Dans cet exemple, le echo (instruction servant à afficher quelque chose à l'écran) indiquera 16 (la valeur de retour de la fonction `carre()`, soit 4 multiplié par 4).

Les points à retenir dans cet exemple sont :

- Une fonction doit être déclarée, sinon elle ne peut pas être utilisée
- Ce qu'on donne à la fonction (les arguments) doivent avoir une nature correspondant au traitement pour lesquelles ils seront utilisés (il est difficile d'obtenir le carré d'un morceau de texte). *Notez qu'il existe des fonctions pour tester la nature d'une variable*
- Une fois qu'une fonction est déclarée, on peut l'utiliser autant que l'on veut

- L'instruction `echo` sert à afficher quelque chose à l'écran (texte, nombre, etc.)

Dans notre exemple, la fonction `carre()` retourne une valeur grâce à `return`. On doit ensuite afficher cette valeur grâce à `echo`.

Nous aurions pu la transformer en une fonction qui ne retourne pas de valeur en y intégrant l'instruction `echo`. Cela donnerait :

```
<?php
// Déclaration de la fonction autrecarre
// $nombre doit être une variable de type numérique (c'est
le seul argument de la fonction)

function autrecarre($nombre){
    // On affiche $nombre multiplié par $nombre
    echo $nombre * $nombre;
}

// Utilisation de la fonction "autrecarre" avec 10 en
paramètre
autrecarre(10);

// Cela affichera 100
?>
```

Les points à retenir dans cet exemple sont :

- Il n'est pas obligatoire que les arguments d'une fonction soient contenus dans des variables (on peut indiquer directement une valeur en argument)
- Deux fonctions ne peuvent pas porter le même nom

Note : Il est possible de définir des variables à l'intérieur d'une fonction. Toutefois, il ne sera pas possible de les utiliser en dehors de la fonction.

Les thèmes WordPress utilisent des dizaines de fonctions pour afficher les pages d'un site. Rassurez-vous, vous n'aurez pas à en créer mais juste les utiliser. Nous les découvrirons dans la partie suivante.

## L'instruction echo

Nous avons vu que echo servait à afficher du contenu à l'écran. On peut lui passer du texte, un nombre ou une variable pour les afficher :

```
<?php
// Affichage de texte
echo "Bonjour, ceci est du texte";

// Affichage d'un nombre
echo 4;

// Affichage d'une variable
$var = "Coucou les amis";
echo $var;
?>
```

On peut aussi lui demander d'afficher la valeur de retour d'une fonction :

```
function cube($nombre){  
    return $nombre * $nombre * $nombre;  
}  
  
echo cube(3); // Affichera 27  
?>
```

Il est possible de complexifier un peu :

```
<?php  
$aube = 10;  
echo "La ville de Troyes se situe dans le département  
numéro " . $aube;  
// Affichera : La ville de Troyes se situe dans le  
département numéro 10  
?>
```

Ici, la variable \$aube sera affichée à la suite du texte grâce au caractère de liaison ..

On peut aussi inclure des fonctions qui possèdent une valeur de retour :

```
<?php  
echo "Le cube de 3 est " . cube(3);  
// Affichera : Le cube de 3 est 27  
?>
```

L'instruction echo peut également effectuer des calculs :

```
<?php
echo 4 * 5 - 2;
// Affichera 18

$var1 = 7;
$var2 = 9;
echo $var1 - $var2;
// Affichera -2

echo carre(4) + cube(3);
// Affichera 43
?>
```

Dans un thème WordPress, vous pourrez voir que echo peut être utilisé pour afficher du code HTML. Par exemple :

```
<?php echo "<p>Ceci est un paragraphe affiché avec echo</p>"; ?>
```

Vous vous demandez probablement comment afficher le caractère " étant donné que l'on l'utilise déjà pour délimiter ce que l'on désire afficher.

Si vous tentez d'écrire cela :

```
<?php
echo "<p>Ceci est un paragraphe affiché avec "echo"</p>";
// Cette instruction provoquera une erreur
?>
```

Vous obtiendrez une erreur car echo affiche ce qui est contenu entre les deux premiers guillemets. Le reste de l'instruction n'est donc pas cohérent (echo"</p>").

Pour afficher des guillemets sans provoquer d'erreur, il faut insérer ce que l'on appelle un « anti slash », c'est à dire une barre oblique inversée.

Cet anti slash indique qu'il faut ignorer la spécificité du caractère qui le suit.

Le code suivant est correct :

```
<?php
echo "<p>Ceci est un paragraphe affiché avec \"echo\"</p>";
// Cette instruction ne provoquera pas d'erreur car les
signes " précédés de \ sont considérés comme de simples
caractères
?>
```

## Les conditions

En PHP (et plus largement en programmation), il est nécessaire de faire des tests.

Est-ce que le visiteur peut consulter le contenu de cette page ? Est-ce qu'un article possède une image en miniature ? Est-ce que cet article est classé dans cette catégorie ? Etc.

Vous trouverez ce genre de tests dans tous les thèmes WordPress. Alors à quoi ressemblent-ils ? Étudions l'exemple ci-dessous :

```
<?php
// Déclaration d'une variable $age
$age = 20;
if($age >= 18){ // Si $age est supérieur ou égal à 18
    echo "Accès autorisé"; // On affiche que l'accès est
    permis
}
else{ // Sinon on indique le contraire
    echo "Accès interdit aux moins de 18 ans!";
}
?>
```

Que peut-on retenir ?

- À chaque fois que vous verrez un `if` suivi d'un `else` ou même un `if` seul, c'est qu'il y a un test. Des instructions spécifiques seront exécutées fonction du résultat (ici utilisation de l'instruction `echo`).
- Les instructions liées au `if` et au `else` doivent être contenues dans des accolades.

Note : Les accolades sont facultatives dans le cas où une seule instruction suit le `if` ou le `else` (il est toutefois recommandé de les utiliser pour gagner en clarté).

Dans l'exemple ci dessus, on teste si l'âge est supérieur ou égal (opérateur `>=`) à 18. Si c'est le cas on affiche un message, sinon on affiche autre chose.

C'est une sorte de bifurcation si vous voulez. Deux possibilités sont offertes mais une seule doit être exécutée par PHP.

La structure `if/else` peut utiliser les opérateurs suivants pour comparer ou évaluer des valeurs :

- `<` : inférieur
- `<=` : inférieur ou égal
- `==` : égal
- `!=` : différent
- `>=` : supérieur ou égal
- `>` : supérieur
- `||` : ou
- `&&` : et
- `!` : non

Les trois derniers opérateurs ne servent pas à comparer des valeurs mais à les évaluer. On peut les utiliser avec des variables booléennes (`true/ false`) ou des fonctions renvoyant une valeur booléenne. Par exemple :

```
<?php
$toto = true;
$titi = false;

// Si $toto est vrai ET que $titi est vrai alors...
if($toto == true && $titi == true){
    echo "Tout est bon !";
}

// Si $toto est vrai OU que $titi est vrai alors...
if($toto == true || $titi == true){
    echo "Allez ça passe !";
}

// Si $toto est faux alors...
```

```
if(!$toto){  
    echo "Toto a tout faux !";  
}  
?>
```

Il existe d'autres façons d'écrire des conditions en PHP, on peut citer celle-ci :

```
<?php  
$toto = true;  
// Si $toto est vrai  
if($toto) :  
    echo "Toto a raison";  
else :  
    echo "Toto a tort";  
endif;  
?>
```

Dans cet exemple, les accolades disparaissent au profit de `:` et de `endif;` à la fin de la structure conditionnelle. Si le `else` n'est pas nécessaire, on peut écrire :

```
<?php  
$toto = true;  
// Si $toto est vrai  
if($toto) :  
    echo "Toto a raison";  
endif;  
?>
```

# Les boucles

Les boucles sont la seconde structure que l'on peut trouver dans n'importe quel langage de programmation. Je me souviens encore d'un de mes professeurs lorsque j'étais en DUT Informatique qui nous disait :

*Avec les boucles et les conditions on peut régler n'importe quel problème !*

Comme son nom l'indique, une boucle est quelque chose qui se répète. Tout simplement.

Par exemple, WordPress en utilise une pour afficher des articles sur une page blog. Cela fonctionne de cette façon, tant qu'il n'y a pas X articles, il en affiche un nouveau. Dès que le nombre est atteint, la boucle s'arrête et le reste de la page est affiché.

Nous verrons ça plus en détail dans la suite de ce chapitre. Contentons-nous d'un exemple basique pour l'instant :

```
<?php
// Initialisation d'un compteur
$compteur = 5;

while($compteur > 0){ // Tant que $compteur est supérieur
à 0
    echo $compteur . "<br>"; // On affiche sa valeur et
une balise "br" pour sauter une ligne.
    $compteur = $compteur - 1; // On soustrait 1 à la
valeur du compteur
} // Et on referme la boucle
?>
```

L'exemple ci-dessus affichera :

```
5
4
3
2
1
```

Vous vous demandez probablement pourquoi 0 n'est pas affiché. Eh bien, la condition de la boucle indique « tant que \$compteur est supérieur à 0 » (`$compteur > 0`).

Lorsque \$compteur vaut 1, il est affiché (première instruction) puis sa valeur passe à 0 (seconde instruction).

Ensuite, la boucle regarde à nouveau si \$compteur est supérieur à 0. Ce n'est plus le cas donc, la boucle s'achève.

Comme pour les conditions, il existe une autre syntaxe pour les boucles de type `while`. La voici :

```
<?php
$compteur = 5;

while($compteur > 0):
    echo $compteur . "<br>";
    $compteur = $compteur - 1;
endwhile;
?>
```

Les accolades disparaissent aussi et les instructions sont comprises entre :  
et `endwhile`;

# Récapitulons

Nous en avons terminé avec les bases du PHP. Cette leçon est plutôt minimaliste (il y aurait encore beaucoup à dire) mais elle vous permettra de mieux comprendre ce langage de programmation.

Cette partie vous aura également permis d'appréhender le vocabulaire de base du PHP (une variable, une fonction, une condition `if`, une boucle `while`, etc.) nécessaire à la bonne compréhension de la suite du guide.

Ces connaissances vous permettront d'être moins perdu lorsque vous consulterez les fichiers d'un thème WordPress.

Avant de passer à la suite, testez-vous avec le quiz associé à cette partie. Rendez-vous dans le dossier WP des exercices pour vous entraîner.

## Chapitre 4.3

# Les fonctions usuelles des thèmes WordPress

Continuons notre découverte du contenu des fichiers composant les thèmes WordPress.

Logiquement, si vous ouvrez un fichier d'un thème vous devriez savoir quel est son rôle (grâce à son nom) et comprendre les grandes lignes du code qu'il contient.

Pour vous aider à affiner votre analyse des fichiers d'un thème, nous allons étudier les fonctions que vous pourrez y trouver.

Pour rappel, une fonction correspond à du code que l'on appelle et qui effectue un traitement précis. Ce traitement peut retourner ou non un résultat.

Nous avons par exemple étudié les fonctions *carre* et *cube* dans la partie précédente.

Commençons sans plus tarder par les...

## Fonctions d'inclusion de fichiers

Dans la première partie de ce chapitre, nous avons vu que les thèmes WordPress se décomposent en plusieurs fichiers pour faciliter leur construction et personnalisation.

Par exemple, le fichier `single.php` (qui affiche individuellement les articles) a besoin d'inclure l'en-tête du site, éventuellement une barre latérale et le pied de page.

Si nous jetons un œil à un thème, on peut voir que les fichiers de templates utilisent tous des fonctions similaires.

On peut citer :

- `get_header()` pour inclure l'en-tête de la page (soit le fichier `header.php`).
- `get_footer()` pour inclure le pied de page (soit le fichier `footer.php`).
- `get_sidebar()` pour inclure la barre latérale (soit le fichier `sidebar.php`) à chaque fois où elle est nécessaire.

Pour simplifier, cela fonctionne de la façon suivante :

```
<?php
get_header(); // insertion de l'en-tête

/* Affichage du contenu de la page (boucle WordPress que
nous allons aborder dans la suite de ce chapitre) */

get_sidebar(); // insertion de la barre latérale

get_footer(); // insertion du pied de page
?>
```

Vous devriez retrouver ces fonctions en ouvrant les fichiers `category.php`, `archive.php` ou `index.php` de la plupart des thèmes WordPress. Faites le test si vous avez un thème sous la main.

Si vous avez besoin de plusieurs styles d'en-tête, de pied de page ou de plusieurs barres latérales, vous pouvez donner un argument à ces trois fonctions.

Par exemple, si vous avez besoin d'un en-tête particulier pour la page d'accueil, vous pouvez créer le fichier `header - home . php` et utiliser le code `get_header ( ' home ' ) ;` pour l'inclure automatiquement.

Nous verrons comment charger certains fichiers en fonction de la page à afficher dans la suite de cette partie.

Jusqu'ici, ces fonctions sont assez simples. Il suffit de les insérer pour inclure les fichiers nécessaires (il va de soi que ces fichiers doivent exister pour être inclus).

Voyons maintenant d'autres fonctions d'inclusion de fichiers que l'on peut trouver dans les thèmes WordPress.

## **get\_template\_part()**

WordPress met à disposition la fonction `get_template_part ( )` afin d'inclure des fichiers. Il est possible de l'utiliser de deux façons :

```
<?php
// En donnant directement le nom du template
get_template_part('monfichier');
// WordPress cherchera à inclure "monfichier.php"

// En procédant en deux temps
get_template_part('contenu', 'video');
// WordPress cherchera à inclure le fichier "contenu-
video.php"
// S'il n'existe pas, il tentera d'inclure "contenu.php"
?>
```

Si le fichier à inclure se situe dans un dossier, on peut l'inclure de la façon suivante :

```
<?php
// En donnant directement le nom du template
get_template_part('mondossier/monfichier');
// WordPress cherchera à inclure "monfichier.php" à partir
de "mondossier"
?>
```

Attention : Aucune erreur ne sera déclenchée si le fichier à charger n'existe pas. Veillez à bien spécifier le nom d'un fichier existant.

## **include(), include\_once(), require(), require\_once()**

Ces quatre fonctions ne sont pas issues de WordPress mais de PHP. La bonne pratique consiste à utiliser `get_template_part()` mais vous pourrez trouver ces quatre fonctions dans certains thèmes. Je dois donc vous les présenter :

- `include()` : inclut un fichier (un avertissement sera déclenché si le fichier n'existe pas)
- `include_once()` : comme `include()` mais n'inclut pas le fichier si cela est déjà fait
- `require()` : inclut un fichier (une erreur sera déclenchée si le fichier n'existe pas)
- `require_once()` : comme `require()` mais n'inclut pas le fichier si cela est déjà fait

Note : La différence entre l'avertissement et l'erreur est qu'en cas d'erreur, le site ne fonctionnera plus.

Si votre thème possède une de ces fonctions, vous verrez que le nom entier du fichier doit être spécifié en argument. Par exemple :

```
<?php
include_once( 'monfichier.php' );

// Si le fichier est dans un sous-dossier
include_once( 'mondossier/monfichier.php' );
?>
```

## Afficher des informations liées au site

Il peut parfois être utile de récupérer des informations relatives à son site pour les afficher à un endroit précis.

Pour ce faire, vous pouvez utiliser les fonctions suivantes.

### **bloginfo()**

Il est possible de passer divers arguments à la fonction `bloginfo()` pour obtenir des informations relatives à son site mais les trois principaux sont :

- `name` : Nom du site défini dans *Réglages > Général* (ex : WP Marmite).
- `description` : Slogan du site défini dans *Réglages > Général* (ex : Tirez le meilleur de WordPress).
- `admin_email` : Adresse email de l'administrateur du site défini dans *Réglages > Général*

Il existe plus d'une quinzaine d'arguments possibles mais d'autres fonctions les remplacent désormais.

On peut notamment citer...

## home\_url()

`home_url()` est une fonction qui renvoie l'adresse de votre site. Pour afficher l'adresse sur votre site ou dans un lien, vous devez impérativement utiliser l'instruction `echo`.

Imaginons que vous ayez besoin d'afficher un lien avec l'adresse de votre site cela donnerait :

```
<a href="<?php echo home_url(); ?>">Retour à l'accueil</a>
```

L'avantage de cette fonction est que l'on peut lui donner un chemin en argument. Si vous avez besoin de créer un lien vers la page « Contact » située à l'adresse `http://monsite.com/contact`, il faut utiliser le code suivant :

```
<a href="<?php echo home_url('contact'); ?>">Contactez-moi :)</a>
```

## get\_stylesheet\_directory\_uri() et get\_template\_directory\_uri()

Derrière ces noms à rallonge se cachent des fonctions essentielles de WordPress. En effet, **ce sont elles qui vont permettre de récupérer l'adresse du dossier du thème.**

Cela est très pratique pour récupérer l'adresse du fichier `style.css`. Étant donné que le fichier `style.css` se trouve à la racine d'un thème, il y a juste à écrire ceci pour afficher son adresse :

```
<?php echo get_stylesheet_directory_uri() . "/style.css";  
?>
```

Pourquoi deux fonctions me direz-vous ? Et comment savoir quelle est celle à utiliser ?

Dans le chapitre suivant, nous allons parler d'un terme que vous connaissez probablement. Il s'agit des thèmes enfants.

*Pour rester concis, un thème enfant est une sorte de sous-thème qui permet de personnaliser un thème d'origine (que l'on appelle thème parent) sans l'altérer.*

En présence d'un thème enfant, la fonction `get_stylesheet_directory_uri()` retournera l'adresse du thème enfant alors que la fonction `get_template_directory_uri()` retournera l'adresse du thème parent.

Dans un thème WordPress, on peut avoir besoin de la première pour récupérer l'adresse du thème enfant (et lier la feuille de style) ou de la seconde pour obtenir l'adresse du thème parent à coup sûr (et faire un lien vers un dossier d'images par exemple).

*Si aucun thème enfant n'est employé, les deux fonctions retourneront la même chose, à savoir l'adresse du thème.*

Continuons notre découverte des fonctions usuelles de WordPress.

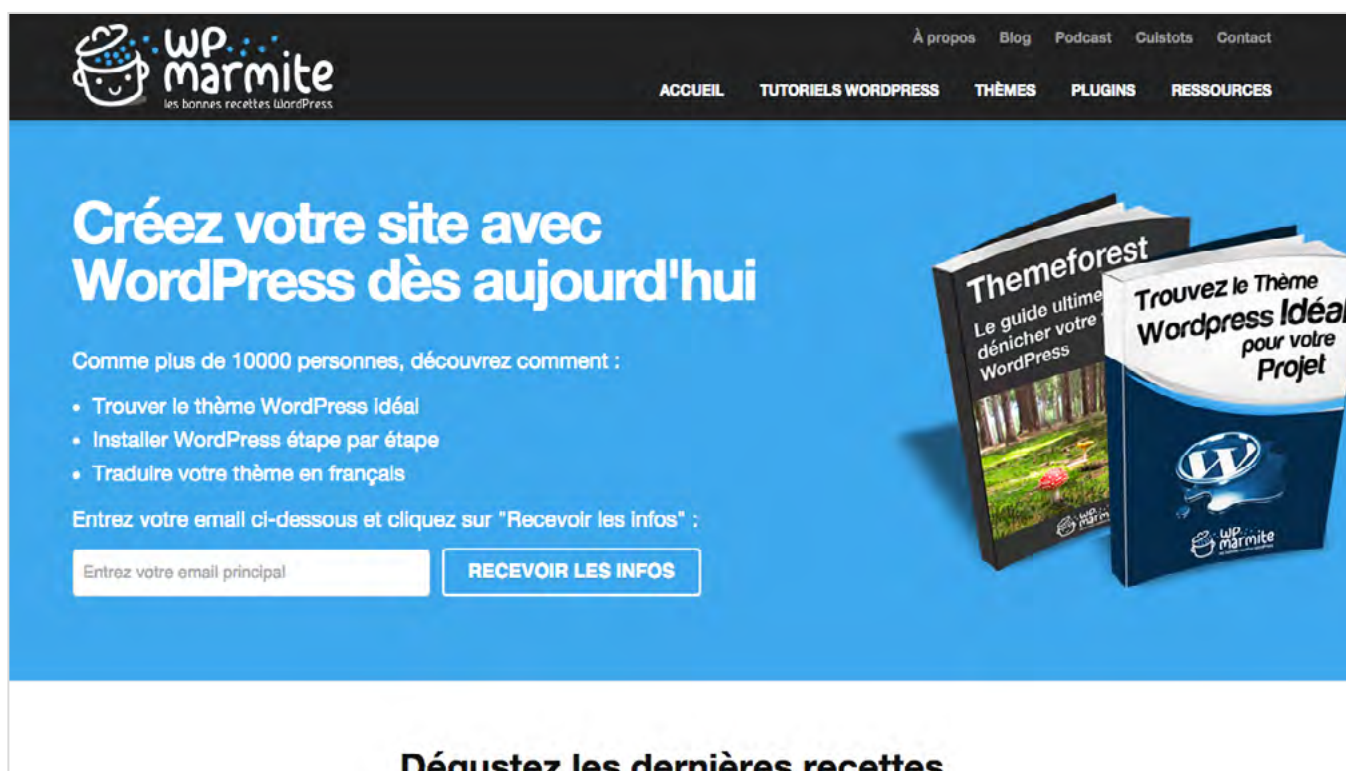
## **Les marqueurs conditionnels**

Dans la partie précédente, nous avons vu qu'il était possible de faire des tests afin d'accomplir ou non certains traitements (structure `if/else`).

Si la condition est validée, alors on exécute le code associé au `if` sinon on exécute le code associé au `else`.

À partir de là, on peut imaginer toute une ribambelle de tests à placer un peu partout dans un thème pour afficher des choses spécifiques en fonction de la page où l'utilisateur se trouve.

Par exemple sur WP Marmite, la page d'accueil possède un en-tête plus imposant que les autres pages :



Il existe plusieurs solutions pour réaliser cela, mais j'ai décidé de placer le marqueur conditionnel `is_front_page()` dans le fichier `header.php` pour tester si le visiteur est sur la page d'accueil.

Si c'est le cas, on affiche le formulaire sinon on affiche l'en-tête classique. Le code que j'ai utilisé ressemble à ceci :

```
<?php
if(is_front_page()){
    // Code HTML/PHP de l'en-tête avec le formulaire
}
else{
    // Code HTML/PHP de l'en-tête classique
}
?>
```

Il existe une quantité d'autres marqueurs conditionnels pour exécuter du code sous certaines conditions.

Avant de vous en citer quelques-uns, je veux que vous reteniez qu'**ils renvoient toujours deux sortes de valeurs : soit vrai, soit faux** (true ou false).

Sans cela, ils ne pourraient pas être utilisés par la structure if/else.

## Liste non exhaustives de marqueurs conditionnels

- `is_single()` : Vaut vrai si l'on se trouve sur un article seul. *Il est possible de donner l'identifiant numérique, texte (slug) ou le titre d'un article en paramètre*
- `is_page()` : Vaut vrai si l'on se trouve sur une page. *Il est possible de donner l'identifiant numérique, texte (slug) ou le titre d'une page en paramètre*
- `is_category()` : Vaut vrai si l'on se trouve sur une page de catégorie. *Il est possible de donner l'identifiant numérique, texte (slug) ou le titre d'une catégorie en paramètre*
- `is_tag()` : Vaut vrai si l'on se trouve sur une page d'une étiquette (nouveau nom des mots-clés) *Il est possible de donner l'identifiant numérique, texte (slug) ou le titre d'une étiquette en paramètre*

- `is_author ( )` : Vaut vrai si l'on se trouve sur une page d'un auteur. *Il est possible de donner l'identifiant numérique, texte (nicename) ou pseudo d'un auteur en paramètre*
- `is_tax ( )` : Vaut vrai si l'on se trouve sur une page d'une taxonomie. *Il est possible de donner l'identifiant numérique, texte (slug) ou le titre d'une taxonomie en paramètre*
- `is_date ( )` : Vaut vrai si l'on se trouve sur une page d'archive liée à une date. Les fonctions `is_year ( )`, `is_month ( )` et `is_day ( )` permettent respectivement de cibler les page d'archives annuelles, mensuelles et journalières
- `is_archive ( )` : Vaut vrai si l'on se trouve sur une page d'archive (voir la hiérarchie des templates dans la première partie de ce chapitre)
- `is_attachment ( )` : Vaut vrai si l'on se trouve sur une page d'un média
- `is_search ( )` : Vaut vrai si l'on se trouve sur une page de type résultat de recherche
- `is_page_template ( )` : Vaut vrai si l'on se trouve sur le modèle de page donné en paramètre (on doit donner le nom complet du fichier, ex: `template-accueil.php`)
- `is_404 ( )` : Vaut vrai si l'on se trouve sur la page 404
- `is_home ( )` : Vaut vrai si l'on se trouve sur la page des articles
- `is_front_page ( )` : Vaut vrai si l'on se trouve sur la page d'accueil
- `is_singular ( )` : Vaut vrai si l'on se trouve sur une publication seule (article, page ou autre type de contenu). *Il est possible de donner l'identifiant numérique, texte (slug) ou le titre d'un article en paramètre*
- `is_preview ( )` : Vaut vrai si l'on se trouve sur l'aperçu d'un article, d'une page ou d'un autre type de contenu
- `is_user_logged_in ( )` : Vaut vrai si le visiteur est connecté au site
- `current_user_can ( 'manage_options' )` : Vaut vrai si l'utilisateur est administrateur

Comme cela a été indiqué, il existe **d'autres marqueurs conditionnels** afin de mettre en place des tests plus poussés.

Cependant ceux présentés ci-dessus vous permettront de pouvoir créer des mises en page spécifiques dans un bon nombre de cas.

Note : Les deux derniers marqueurs conditionnels sont particulièrement utiles pour tester des choses sans que les visiteurs ne puissent s'en apercevoir.

## Les fonctions de traduction

Un bon thème WordPress se doit d'être traduisible. Cela signifie que l'auteur a dû penser à inclure des fonctions dédiées à la traduction du thème.

Ces fonctions seront ensuite utilisées par des logiciels comme **PoEdit** pour créer des fichiers de traduction en français, allemand, espagnol, etc.

WordPress lira ensuite ces fichiers pour intégrer la traduction à la place du texte original. Les visiteurs pourront ainsi consulter un site dans leur langue.

Sans les fonctions dont nous allons parler ci-dessous, tout cela serait impossible.

Voici donc les principales fonctions de traduction de WordPress avec leurs arguments :

- `__( $texte, $domaine )` : Retourne la chaîne `$texte` traduite
- `_e( $texte, $domaine )` : Affiche la chaîne `$texte` traduite (pas besoin d'utiliser l'instruction `echo`)

L'argument `$domaine` correspond à un identifiant texte associé au thème à traduire. Cela aidera WordPress à sélectionner le bon fichier de traduction pour trouver et remplacer les chaînes à traduire.

Il existe d'autres fonctions de traduction permettant de traduire des chaînes plus complexes (avec des pluriels, des variables, etc) mais cela s'éloigne un peu de la personnalisation de thème.

Vous pouvez [consulter cet article](#) pour aller plus loin.

## Récapitulons

Au cours de cette partie, nous avons vu quelles sont les principales fonctions qu'il est possible de trouver dans un thème WordPress.

Ces fonctions permettent notamment :

- d'appeler d'autres fichiers pour afficher une page en intégralité
- de récupérer des informations liées au site
- de tester certaines conditions pour afficher un contenu spécifique
- de traduire un thème

En incluant ces fonctions dans les fichiers de base d'un thème, on peut déjà entrevoir un bon nombre de possibilités.

Tout est bon pour vous ? Dans ce cas, testez vos nouvelles connaissances avec l'exercice 3 de ce chapitre. Pour cela, rendez-vous dans le dossier *Exercices > Chapitre 4 - WP* associé à Relooker son Thème.

Continuez ensuite votre apprentissage des thèmes avec un concept fondamental : la boucle WordPress.

## Chapitre 4.4

# La boucle WordPress et ses fonctions associées

Poursuivons notre exploration de WordPress avec la boucle WordPress.

La boucle WordPress ?

Diantre ! Qu'est ce que c'est que ça ?!

Procédons méthodiquement.

Nous avons vu qu'une boucle était un ensemble d'instructions qui se répétaient un certain nombre de fois.

*WordPress utilise une boucle pour afficher ses articles, ses pages et éventuellement des éléments de portfolio, des services, etc.*

Si l'on applique cela **à la page blog de WP Marmite**, elle se construit de cette façon :

- Affichage de l'en-tête grâce à la fonction `get_header()` ;
- La boucle WordPress permet d'afficher les 5 derniers articles du blog
- Affichage de la barre latérale grâce à la fonction `get_sidebar()` ;
- Affichage du pied de page grâce à la fonction `get_footer()` ;

Il en va de même pour les pages de catégories, de recherche, etc.

Même les pages et les articles seuls sont affichés de la même manière. La seule différence est que la boucle ne tourne qu'une seule fois (car il y a qu'un seul contenu à afficher).

Au niveau du code, voici à quoi ressemble la boucle WordPress :

```
<?php
// Si au moins un article correspond on lance la boucle
(sinon on passe au "else")
if ( have_posts() ) :

    // Tant qu'il y a des articles et que cela ne dépasse pas
    le nombre indiqué dans Réglages --> Lecture
    while ( have_posts() ) : the_post();

        /* Code permettant l'affichage des informations liées à un
        article, une page, etc. */

        // La boucle est terminée
        endwhile;

        // Si aucun article ne correspond
        else:
        // On affiche ce message en HTML
        ?>

        <p>Aucun article n'a été trouvé.</p>

    <?php // Fin de la condition (du if)
    endif; ?>
```

Note : Au sein des thèmes, le code de la boucle WordPress peut varier sensiblement. Néanmoins, vous pourrez toujours la repérer grâce au `if(haveposts()):while....`

Bien sûr, si un thème utilise le code présenté ci-dessus, rien ne serait affiché car la boucle contient uniquement un commentaire PHP.

Il faut donc peupler cette boucle d'un mélange de code HTML et PHP pour afficher le contenu des pages.

Cela nous amène donc à étudier...

## Les fonctions de la boucle WordPress

Dans cette boucle, vous allez retrouver les mêmes fonctions d'un thème à l'autre. Elles se mêlent au code HTML afin d'afficher diverses informations comme le titre d'un article, son adresse, son contenu, etc.

Examinons ces fonctions plus en détail :

- `the_title()` : Le titre de l'article (ou de la page)
- `the_excerpt()` : L'extrait
- `the_content()` : Le contenu
- `the_date()` : Date de publication
- `the_time()` : Date et/ou heure de publication
- `the_permalink()` : Adresse de l'article (url)
- `the_post_thumbnail()` : Image à la une (balise `img` avec l'image)
- `comments_number()` : Nombre de commentaire de l'article
- `the_category()` : Catégorie(s) de l'article
- `the_tags()` : Étiquettes de l'article
- `the_author()` : Nom de l'auteur
- `the_ID()` : Identifiant numérique de l'article

*En plaçant (ou retirant) judicieusement certaines d'entre-elles dans le code HTML de la boucle, vous relookerez la structure d'un thème très rapidement.*

Les fonctions précédentes afficheront directement la valeur demandée. Pour récupérer et stocker les valeurs dans des variables, ajoutez le préfixe `get_` aux fonctions ci-dessus. Par exemple `get_the_content()` renverra le contenu HTML d'une publication.

Dans certains thèmes, il arrive que le code de la boucle WordPress ne soit pas affiché directement.

L'auteur du thème peut décider de placer dans un ou plusieurs fichiers séparés inclus grâce à la fonction `get_template_part()` que nous avons étudié précédemment. C'est notamment le cas du thème par défaut de WordPress : TwentyFifteen).

## **Étude de la boucle WordPress issue du thème Response**

Pour un client, j'ai dû installer et relooker **le thème Response** de la boutique Organic Themes.

Voici comment se structure la boucle WordPress de ce thème pour l'affichage d'un article seul (la boucle a été simplifiée pour l'exemple) :

```

<?php // Début de la boucle. On teste si l'article en
question existe ?>
<?php if (have_posts()) : while (have_posts()) : the_
post(); ?>

    <div class="article">

        <?php // Affichage du titre avec "the_title()"
dans une balise "h1" ?>
        <h1 class="headline"><?php the_title(); ?></h1>

        <?php // Métadonnées de l'article ?>
        <div class="postauthor">

            <?php // Lien de l'auteur et date de
publication affiché à gauche ?>
            <p class="left"><i class="icon-user"></
i> &nbsp;<?php _e("Posted by", 'organicthemes'); ?>
<?php the_author_posts_link(); ?> <?php _e("on",
'organicthemes'); ?> <?php the_time(__("F j, Y",
'organicthemes')); ?></p>

            <?php // Lien vers les commentaires ainsi que
leur nombre affiché à droite ?>
            <p class="right"><i class="icon-
comment"></i> &nbsp;<a class="scroll" href="<?php
the_permalink(); ?>#comments"><?php comments_number(__
("Leave a Comment", 'organicthemes'), __("1 Comment",
'organicthemes'), '% Comments'); ?></a></p>

        </div>

```

```

        <?php // Contenu de l'article ?>
        <?php the_content(__("Read More",
'organicthemes')); ?>

        <?php // Lien pour modifier l'article (il
s'affichera uniquement aux utilisateurs concernés) ?>
        <?php edit_post_link(__("(Edit)",
'organicthemes'), '', ''); ?>
        <div class="clear"></div>

</div>

<?php // Autres métadonnées ?>
<div class="postmeta">

        <?php // Catégories de l'article ?>
        <p class="left"><i class="icon-reorder"></i>
        &nbsp;<?php _e("Category:", 'organicthemes'); ?> <?php
the_category(', '); ?></p>

        <?php // Si l'article est marqué par des
étiquettes, on les affiche ?>
        <?php if ( has_tag() ) { ?>
                <p class="right"><i class="icon-tags"></i>
                &nbsp;<?php _e("Tags:", 'organicthemes'); ?> <?php the_
tags(' '); ?></p>

        <?php } ?>
</div>

<?php // Fin de la boucle ?>

```

```
<?php endwhile; else: ?>

    <?php // Message d'erreur si aucun article ?>
    <p><?php _e("Sorry, no posts matched your criteria.",
'organictthemes'); ?></p>
<?php endif; ?>
```

Comme vous pouvez le constater, le code PHP doit impérativement se situer entre les balises `<?php` et `?>`. Dans le cas contraire, une erreur se produira et le site ne s'affichera pas.

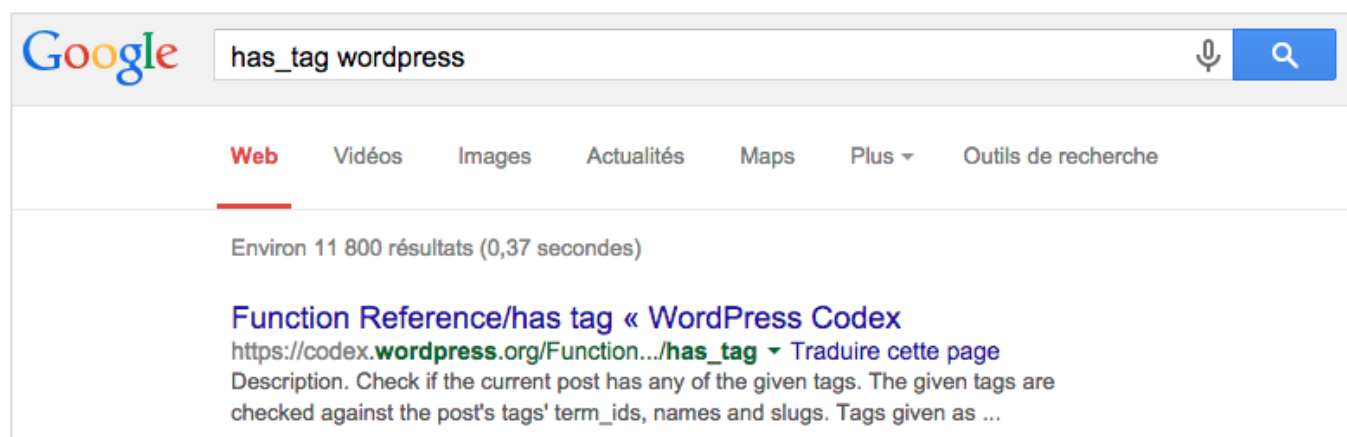
## Que font les fonctions que nous n'avons pas abordées

Il se peut que certaines fonctions que nous n'avons pas étudiées soient présentes au sein d'une boucle.

Dans ce cas, je vous invite à copier son nom et à la rechercher sur Google accompagnée du mot « WordPress ».

Si c'est une fonction de WordPress, le premier résultat s'affichera et vous pourrez en savoir plus (vous devrez peut-être traduire la page pour avoir des informations en français).

C'est par exemple le cas de la fonction `has_tag( )`. Voilà ce que l'on peut trouver :



En consultant la page, on peut se rendre compte qu'il s'agit d'un marqueur conditionnel qui permet de savoir si des étiquettes (des « tags » en anglais) sont associés à une article.

Quand on regarde le code, c'est logique car dans la condition, on utilise la fonction `the_tags()` pour les afficher. La condition permet donc de ne pas exécuter cette fonction pour rien.

Dans le cas contraire où aucun résultat concret n'est retourné par Google, c'est qu'il s'agit d'une fonction créée par l'auteur du thème.

Il faut alors chercher dans les fichiers du thème l'endroit où elle a été déclarée pour voir ce qu'elle contient.

*Attention tout de même à ne pas vous y perdre. Votre but est de relooker votre thème selon ce que vous avez défini dans votre liste TOP, pas de décortiquer votre thème.*

## Les arguments des fonctions de la boucle

Comme vous pouvez le voir, les fonctions de la boucle ne sont pas utilisées telles quelles. Il est possible de leur transmettre des arguments pour affiner ce qu'elles vont retourner.

Dans notre exemple, on peut voir que la fonction `the_time()` est affichée sous la forme suivante : `the_time(__('F j, Y', 'organicthemes'))`.

Ici, il y a deux choses à prendre en compte.

Tout d'abord, on constate que la fonction de traduction `__( )` est présente.

Dans la partie précédente nous avons vu que le premier argument correspond à la chaîne à traduire et que le second est une sorte d'identifiant qui relie toutes les chaînes de traduction de ce thème.

La fonction `__( )` va donc renvoyer le texte `F j, Y` à la fonction `the_time( )` si le thème n'est pas traduit.

À quoi peuvent bien correspondre ces lettres ?!

En fait, il s'agit du format de date à utiliser.

En effet, le format change selon les langues. C'est pour cela qu'il est contenu dans une chaîne de traduction.

Encore faut-il connaître les formats de date et d'heure mis à disposition. Vous pourrez en savoir plus sur la signification de chaque lettre en vous rendant [sur cette page du Codex de WordPress](#).

Note : Tous les thèmes ne sont pas codés avec les bonnes pratiques recommandées par WordPress. C'est le cas de cet exemple.

Pour information, l'auteur du thème aurait dû récupérer le format de date par défaut défini dans l'administration pour éviter à l'utilisateur de traduire le format pour sa langue.

Cela aurait donné `the_time( get_option( 'date_format' ) );` (beaucoup plus lisible n'est-ce pas ?).

Continuons.

Lorsque l'on regarde la fonction `comments_number( )`, on peut voir que ses paramètres sont trois fonctions `__( )` contenant chacune un texte. Cela couvre les cas :

- où l'article ne possède pas de commentaire ("Leave a comment" soit "Écrivez un commentaire")
- possède un seul commentaire ("1 Comment")
- possède plus d'un commentaire ("% Comments")

Ainsi chaque fonction de la boucle et par extension de WordPress peut accueillir des arguments.

Pour les connaître, faites une simple recherche sur Google afin d'en savoir plus. Personne ne connaît toutes les fonctions de WordPress. On en découvre tous les jours.

Par exemple, savez-vous qu'il existe une fonction qui met un « P » majuscule lorsque l'on écrit « Wordpress » ?

Et bien oui, il s'agit de `capital_P_dangit()`. [Voici sa page de description](#) sur le Codex de WordPress (la documentation de WordPress).

## Récapitulons

Voilà, nous avons fait le tour des principales fonctions de la boucle WordPress. Comme cela a été dit, il en existe d'autres mais vous allez devoir les découvrir en faisant vos propres recherches. C'est ça l'autonomie.

Prenez un peu de temps pour intégrer cela. Je suis conscient que toutes ces fonctions peuvent faire beaucoup en une seule fois. Rome ne s'est pas faite en un jour.

D'ailleurs, comme je vous l'ai dit, le but de Relooker son Theme n'est pas de devenir un pro du code mais d'en maîtriser les bases.

Lorsque vous serez prêt, ouvrez l'exercice 4 lié à ce chapitre pour tester vos connaissances (attention, vous allez devoir effectuer des recherches pour obtenir des réponses).

## Chapitre 4.5

# Les modèles de page personnalisés

Au cours des précédentes parties de ce chapitre, nous avons vu que chaque type de page emploie un fichier de template précis. La hiérarchie des templates aide notamment à s'y retrouver.

Toutefois, **il y a des cas où l'on souhaite bénéficier d'un agencement particulier** sur une page classique. C'est là qu'interviennent les modèles de page personnalisés.

Grâce à ces modèles de page, on peut par exemple créer ce que l'on appelle des pages d'atterrissage, *des landing pages* en anglais, c'est à dire des pages destinées à proposer d'effectuer une action précise aux visiteurs.

On peut aussi créer une page d'accueil différente du reste du site (par exemple sans barre latérale).

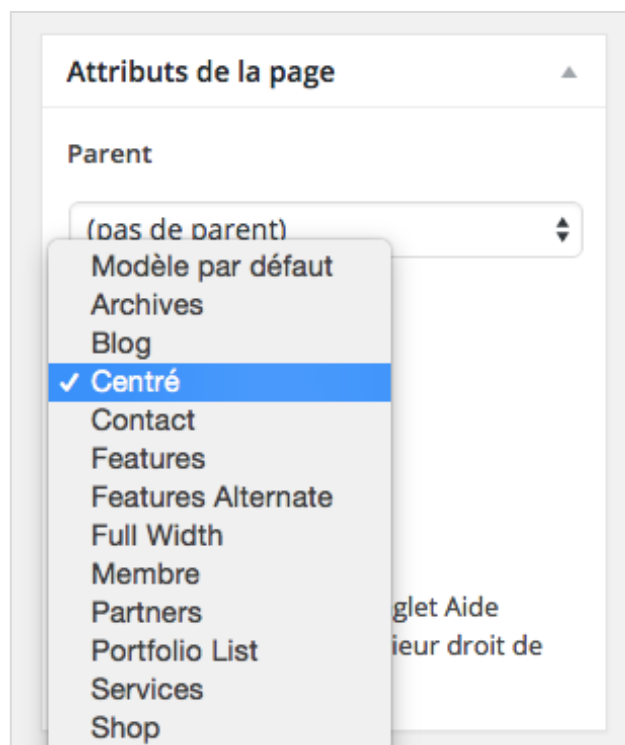
Il est également possible de créer une page dotée d'une structure autre que celle du thème que l'on utilise.

Bref, beaucoup de choses sont possibles.

Dans WordPress, il est assez simple d'appliquer un modèle à une page. Il suffit de se rendre dans l'encart *Attributs de la page* et de sélectionner le modèle de son choix dans le menu déroulant *Modèles*.

En résumé, on peut faire afficher ce que l'on veut à une page si un modèle lui est attribué (et que ce modèle embarque le code nécessaire).

Voici un aperçu des modèles de page proposés par le thème WordPress de WP Marmite, plus ceux que j'ai rajouté ensuite :



Selon votre thème, vous aurez plus ou moins de modèles de page à votre disposition. Généralement, ils se trouveront à la racine de votre site mais ils peuvent également être placés dans un sous-dossier.

Si les modèles de page disponibles ne répondent pas à vos besoins, vous pouvez en créer un. Il suffit de suivre quelques règles.

Commençons par étudier...

## La structure d'un modèle de page

Au niveau du code, un modèle de page possède du code HTML et PHP comme tous les autres fichiers de templates.

Il y a cependant un impératif, le nom du template doit être précisé en commentaire au début du fichier.

Concrètement, il faut ajouter le code suivant :

```
<?php
/* Template Name: Mon modèle */
?>
```

Grâce à cela, WordPress reconnaîtra les modèles de page et pourra les insérer dans le menu déroulant au niveau de l'administration.

Les fichiers destinés à être des modèles de pages peuvent porter n'importe quel nom sauf ceux associés à la hiérarchie des templates de WordPress.

Pour s'y retrouver, il est préférable de les appeler `template-machin.php`.

Par exemple, le template qu'utilise WP Marmite pour afficher du contenu centré sans barre latérale s'appelle `template-centre.php`.

Attention tout de même à ne pas donner des noms trop longs à vos modèles de page car l'espace est limité dans l'administration.

Maintenant que cela est posé, occupons nous du...

## Contenu d'un modèle de page

Au niveau de ce que l'on peut faire figurer dans un modèle de page, cela peut paraître étonnant mais c'est totalement libre.

On peut insérer le code HTML et PHP de notre choix. Bien entendu, il est préférable que votre modèle de page reprenne le style de votre site pour conserver une certaine cohérence mais ce n'est pas obligatoire.

Généralement, il est conseillé de partir du contenu d'un fichier de template existant, par exemple `page.php` ou `index.php` puis de procéder à des modifications.

Il est possible de :

- Appeler un autre fichier d'en-tête à la place de celui par défaut

- Ne pas insérer de barre latérale latérale et centrer le contenu
- Ajouter des styles CSS particuliers pour ce modèle de page

En fait, vous pouvez jouer avec le HTML, le CSS et le PHP pour créer la mise en page dont vous avez besoin.

Vous n'avez même pas besoin d'insérer la boucle WordPress dans les modèles de page. Dans ce cas, il sera impossible d'afficher le contenu de la page ou des données qui y sont associées.

Cela peut sembler étrange mais la boucle n'est pas toujours nécessaire.

## Récapitulons

Dans WordPress, les modèles de page personnalisés permettent de donner une apparence supplémentaire aux pages.

Certains thèmes en proposent par défaut et il est possible de créer ses propres modèles de page. Nous verrons cela plus en détail dans le chapitre suivant.

En attendant, rendez-vous dans le dossier d'exercices de ce chapitre pour voir si vous avez bien retenu tout ce qui concerne l'aspect théorique des modèles de page.

## Chapitre 4.6

# Le fichier `functions.php` et les hooks

Si vous avez déjà parcouru les fichiers d'un thème WordPress, vous avez certainement remarqué la présence d'un fichier nommé `functions.php`.

Ce fichier est spécial car il ne correspond pas à une zone précise du site comme `header.php` ni à une mise en page donnée comme `category.php`.

## Rôle du fichier `functions.php`

Comme son nom l'indique, le fichier `functions.php` va contenir toutes sortes de fonctionnalités relatives au thème dans lequel il est situé.

**En fait, c'est comme si une extension WordPress était incluse dans le thème.**

Note : Une extension WordPress (aussi appelée « plugin » en anglais) permet d'ajouter de nouvelles fonctionnalités à un site WordPress. On peut en installer en se rendant dans Extensions > Ajouter.

En ouvrant le fichier `functions.php` d'un thème, vous ne trouverez pas de structure HTML accompagnée de PHP mais majoritairement du code PHP.

Cela peut s'avérer plus complexe à comprendre que ce qui a été abordé précédemment car on entre dans la véritable programmation.

*Comme vous le savez, le but de ce guide n'est pas de vous faire devenir un développeur, cependant nous allons voir comment décrypter au maximum le contenu du fichier `functions.php`.*

Avant de voir à quoi servent les fonctions que l'on peut trouver dans ce fichier, nous devons passer par une...

## **Introduction aux hooks**

Vous ne vous en doutez certainement pas mais ces fameux hooks sont la colonne vertébrale de WordPress.

Sans eux, WordPress n'aurait pas connu le succès qu'il a aujourd'hui.

*En fait, les hooks permettent d'exécuter du code personnalisé partout où ils sont placés.*

Ils se situent à des endroits stratégiques et n'attendent que d'être utilisés par des thèmes et extensions pour personnaliser le fonctionnement et le comportement de WordPress.

Tout comme dans WordPress, des hooks peuvent figurer dans les thèmes et les extensions. Cela permet à d'autres extensions de pouvoir agir sur ces derniers.

En clair, cela ouvre un immense éventail de possibilités aux développeurs.

Pour vous donner une idée, si on agit sur le hook lié au contenu, on peut ajouter automatiquement un encart au début de chaque article (par exemple une publicité).

Un autre hook peut permettre de placer un favicon différent selon la page où le visiteur se trouve.

*Sans eux il serait impossible de personnaliser WordPress, ses thèmes et ses extensions.*

À présent que vous savez à quoi servent les hooks, nous allons voir qu'ils se divisent en deux catégories. Tout d'abord, il y a...

## **Les actions**

Les actions sont les hooks les plus simples à comprendre. Ils permettent d'exécuter une fonction PHP aux endroits où ils sont placés.

Voici comment se présente une action dans le code :

```
do_action( 'mon_action' );
```

La fonction `do_action( )` permet de créer une action et le texte passé en argument (entre guillemets) lui confère un nom.

Nous venons donc de déclarer une action "mon\_action".

WordPress embarque des centaines de hooks, vous n'aurez donc pas à en définir. L'important ici est de vous en faire comprendre le fonctionnement.

Pour utiliser un hook action, il faut procéder en deux étapes :

1. Déclarer une fonction
2. L'associer à l'action de son choix

La première étape est assez simple car vous savez déjà à quoi ressemble une fonction.

En reprenant l'exemple du favicon, la fonction suivante permet d'afficher un favicon différent si le visiteur se trouve sur la page d'accueil :

```
function rst_custom_favicon(){
    // Si le visiteur est sur la page d'accueil, on
    affiche "homefavicon.ico"
    if(is_front_page()):
        echo "<link rel='icon' href='http://monsite.com/
homefavicon.ico'>";
    else:
        echo "<link rel='icon' href='http://monsite.com/
favicon.ico'>";
    endif;
}
```

Note : La fonction débute par le préfixe rst (pour Relooker son Thème) car il est important de choisir un nom qui soit unique. Si une fonction possède le même nom qu'une autre, PHP ne saura pas laquelle choisir et une erreur se produira. Veuillez donc à créer des fonctions avec un préfixe pour éviter ce genre de désagrément.

Ensuite, la seconde étape consiste à lier la fonction que l'on a créé à l'action que l'on désire.

Pour cela, il va falloir utiliser la fonction `add_action()`.

Dans notre cas, étant donné qu'il faut ajouter le favicon dans la section head du site, nous allons utiliser l'action nommée `wp_head`.

*Si le thème que vous utilisez est bien conçu, l'action `wp_head` est insérée via la fonction `wp_head()` placée à la fin de la section head du fichier `header.php` de votre thème.*

L'action `wp_head` permet d'ajouter toutes sortes de choses dans la section head d'un site (feuilles de styles CSS, fichiers Javascript, balises meta, etc.). Elle fait partie des actions les plus populaires.

Voici comment utiliser la fonction `add_action()` pour relier la fonction `rst_custom_favicon` à l'action `wp_head` :

```
add_action( 'wp_head' , 'rst_custom_favicon' );
```

Vous pouvez constater que le premier argument correspond au nom de l'action et que le second est le nom de la fonction.

Si la fonction `rst_custom_favicon` et l'appel à `add_action` étaient placés dans le fichier `functions.php`, il y aurait une favicon différente sur la page d'accueil de notre site.

Si jamais vous désirez « désactiver » une fonction liée à une action, il existe une fonction `remove_action()` qu'il faut utiliser de la manière suivante :

```
remove_action( 'wp_head' , 'rst_custom_favicon' );
```

Le premier argument est le nom de l'action et le second est le nom de la fonction à désactiver.

Cela aurait pour effet de ne plus exécuter la fonction `rst_custom_favicon` au niveau de l'action `wp_head`. Aucune favicon ne sera affichée.

*Un thème bien conçu doit aussi comporter la fonction `wp_footer()` juste avant la balise `</body>`. Cette fonction permet de placer le hook `wp_footer` dans le pied de page et ainsi de pouvoir charger des scripts (en Javascript principalement).*

Passons maintenant au second type de hook...

## Les filtres

De leur côté, les filtres ne permettent pas d'ajouter ou d'exécuter des morceaux de code. **Ils permettent d'altérer un résultat ou un contenu.**

Si vous voyez un jour une fonction `apply_filters()`, cela signifie qu'un filtre est actif sur une variable.

Comme pour les actions, il faut procéder en deux temps :

1. Déclarer une fonction qui « filtre » une variable
2. Associer cette fonction au filtre adéquat

Pour reprendre l'exemple de la publicité insérée au début des articles, nous allons créer une fonction `rst_top_post_ad`.

```
function rst_top_post_ad( $content ){
    // Contenu HTML de la publicité
    $ad = '<div class="top_post_ad"></div>';
    // On place le contenu de la pub devant le contenu et
on en fait un nouveau contenu
    $content = $ad . $content;

    // On retourne le contenu modifié
    return $content;
}
```

Comme vous pouvez le voir, la fonction `rst_top_post_ad` possède un argument nommé `$content`. Il s'agit du contenu HTML de l'article pour lequel on va afficher une publicité.

On constate aussi que la fonction doit retourner quelque chose avec l'opérateur `return`.

Cela est logique. Étant donné qu'un filtre doit modifier quelque chose, la fonction qui lui est associée ne peut pas rester muette.

Cette association passe par la fonction `add_filter()`.

Le code à employer pour notre exemple est le suivant :

```
add_filter( 'the_content' , 'rst_top_post_ad' );
```

Vous l'aurez compris, l'argument `the_content` de la fonction `add_filter()` est le filtre qui agit sur le contenu des articles.

Le second argument est la fonction qui agit sur notre filtre.

Si jamais vous désirez « annuler » un filtre ajouté par WordPress, un thème ou une extension, il existe la fonction `remove_filter()` qu'il faut utiliser de la manière suivante :

```
remove_filter( 'the_content' , 'rst_top_post_ad' );
```

Comme pour `remove_action`, le premier argument est le nom du hook à cibler et le second le nom de la fonction à désactiver.

Il y a des chances pour que les hooks soient encore assez obscurs pour le moment. Étudions quelques exemples que vous allez pouvoir trouver dans le...

## Contenu du fichier `functions.php`

Nous avons vu au début de cette partie que le fichier `functions.php` confère des fonctionnalités à un thème.

Pour fonctionner, ce fichier se sert du système de hooks (actions et filtres).

Afin de mieux comprendre leur fonctionnement, voici un exemple issu du thème WordPress Twenty Fifteen :

```
// On teste si la fonction que l'on veut associer au hook
n'existe pas déjà afin d'éviter de déclencher une erreur
si elle est déjà définie dans le thème enfant
if ( ! function_exists( 'twentyfifteen_setup' ) ) :

// Si c'est bon, on crée la fonction
function twentyfifteen_setup() {
    // On charge le fichier de traduction
    load_theme_textdomain( 'twentyfifteen', get_template_
directory() . '/languages' );

    // On active la création de liens RSS dans la section
`head` du site
    add_theme_support( 'automatic-feed-links' );

    // On active la gestion des titres par WordPress
    add_theme_support( 'title-tag' );

    // On active les images à la une
    add_theme_support( 'post-thumbnails' );

    // On définit la taille des images à la une
    set_post_thumbnail_size( 825, 510, true );

    // On déclare le menu principal et secondaire
    register_nav_menus( array(
```

```

        'primary' => __( 'Primary Menu',
'twentyfifteen' ),
        'social'  => __( 'Social Links Menu',
'twentyfifteen' ),
    ) );

    // On active le HTML 5 pour les éléments créés par
    WordPress (formulaire de recherche, les commentaires, etc.
    add_theme_support( 'html5', array(
        'search-form', 'comment-form', 'comment-list',
'twentyfifteen', 'caption'
    ) );

    // On active les formats d'article
    add_theme_support( 'post-formats', array(
        'aside', 'image', 'video', 'quote', 'link',
'twentyfifteen', 'status', 'audio', 'chat'
    ) );

    // On récupère la couleur par défaut dans des
    variables
    $color_scheme = twentyfifteen_get_color_scheme();
    $default_color = trim( $color_scheme[0], '#' );

    // On active l'arrière-plan personnalisé
    add_theme_support( 'custom-background', apply_filters(
'twentyfifteen_custom_background_args', array(
        'default-color'      => $default_color,
        'default-attachment' => 'fixed',
    ) ) );

```

```

        // On ajoute un style CSS à l'administration
        add_editor_style( array( 'css/editor-style.css',
        'genericons/genericons.css', twentyfifteen_fonts_url() )
        );

// La déclaration de la fonction `twentyfifteen_setup` est
terminée
}
// On referme la condition
endif;

// Il ne reste plus qu'à associer la fonction à l'action
`after_setup_theme`
add_action( 'after_setup_theme', 'twentyfifteen_setup' );

```

Pour aller plus loin, voici une liste de ce que vous pourrez voir dans le fichier `functions.php` :

- Chargement de fichiers supplémentaires (contenant d'autres fonctionnalités)
- Chargement de scripts Javascript et styles CSS
- Chargement de polices d'écriture
- Déclaration de zones de widgets
- Déclaration de menus
- Chargement des fichiers de traduction
- Déclaration de tailles d'images par défaut
- Déclaration de formats d'articles (image, vidéo, citation, etc.)
- Déclaration de types de contenus (portfolio, services ou autre)
- Déclaration de taxonomies supplémentaires
- Déclaration de pages d'options du thème
- Définition de constantes liées au thème

Bien entendu, cela ne peut pas être exhaustif car il est possible de faire des milliers de choses. C'est tout le pouvoir des hooks !

Note : Cette partie n'a pas vocation à vous enseigner la gestion des hooks car cela pourrait constituer un guide à part entière. Le but est de vous transmettre les grands principes pour vous aider à décrypter le contenu du fichier `functions.php`.

## Récapitulons

Le fichier `functions.php` permet d'apporter les fonctionnalités essentielles pour qu'un thème WordPress fasse correctement son travail.

Sans lui, les menus ne pourraient pas être gérés depuis l'administration, les traductions ne seraient pas effectuées et les images à la une ne s'afficheraient pas.

Pour fonctionner, ce fichier se base sur le système de hooks de WordPress. Il s'agit d'un moyen de personnaliser WordPress sans en modifier directement le code.

En pratique, cela permet d'associer une fonction à un hook pour :

- Afficher ou exécuter du code à un endroit précis (hooks de type action)
- Altérer du contenu ou des variables (hooks de type filtre)

Il existe des centaines de hooks à disposition. Vous pourrez en trouver une liste des plus communs dans la fiche n°4 du dossier Ressources fourni avec le guide.

Il est maintenant temps de tester vos nouvelles connaissances avec un nouvel exercice. Ouvrez l'exercice 6 lié à ce chapitre pour voir ce que vous avez retenu.

## Chapitre 4.7

# Les menus et les widgets dans WordPress

Dans la partie précédente, nous avons parlé du fonctionnement du fichier `functions.php` et de ce que l'on peut y trouver.

Pour finir ce chapitre consacré à WordPress, nous allons voir comment les menus et les widgets se matérialisent à l'intérieur d'un thème.

Cela vous permettra de mieux pouvoir les identifier et pourquoi pas de les personnaliser voire d'en créer de nouveaux.

Commençons par...

## Les menus

Aujourd'hui, tous les thèmes WordPress proposent un menu que l'on peut gérer directement depuis l'administration.

On peut y intégrer des pages, des catégories et des liens personnalisés.

Pourtant, cela serait impossible sans la présence de quelques lignes de code.

## Déclarer des menus

Dans la partie précédente, vous avez pu remarquer la présence d'une fonction `register_nav_menus()` dans le code donné en exemple.

Comme vous pouvez vous en douter, elle sert à enregistrer plusieurs emplacements de menus.

Examinons-la de plus près.

```
register_nav_menus( array(
    'primary' => __( 'Primary Menu', 'twentyfifteen' ),
    'social'   => __( 'Social Links Menu', 'twentyfifteen' ),
) );
```

Nous pouvons voir que la fonction `register_nav_menus()` ne possède qu'un seul argument de type tableau (`array`).

C'est dans ce tableau que doivent être déclarés les emplacements de menus à enregistrer.

On peut voir que chaque emplacement possède un identifiant qui lui est propre. Dans notre cas, il s'agit de `primary` et `social`. Il est important que ces identifiants soient en minuscules (ils peuvent éventuellement comporter des `_`).

À chaque identifiant (chaque index du tableau) est associé une description qui permettra de représenter l'emplacement de menu dans l'administration de WordPress.

Comme nous voulons que tout soit traduit, il faut utiliser la fonction `__( )` pour renvoyer la bonne traduction en fonction de la langue du site.

*Le code ci-dessus doit être compris dans une fonction devant être appelée avec le hook `after_setup_theme`. Cette action est déclenchée une fois que les fichiers du thème sont chargés par WordPress.*

Si un thème n'emploie qu'un seul menu, vous pourrez trouver la fonction `register_nav_menu()` (sans `S` à la fin).

Cette fois, passer un tableau en argument n'est plus nécessaire. Cette fonction a seulement besoin de deux arguments, un identifiant et une description.

Voici comment on pourrait utiliser la fonction `register_nav_menu()` pour enregistrer l'emplacement du menu principal d'un thème :

```
register_nav_menu( 'primary' , __( 'Primary Menu', 'un_theme' ) );
```

Bien entendu, `register_nav_menu()` doit être placé dans une fonction appelée avec le hook `after_setup_theme` sinon cela pourra ne pas fonctionner correctement.

## Afficher des menus

Maintenant qu'un ou plusieurs emplacements de menus sont déclarés, il est possible de leur associer un menu via l'administration (dans *Apparence > Menus*).

Cependant, ils ne s'afficheront pas encore sur le site. En effet, il faut utiliser une fonction dans le thème pour dire : « Affiche-moi tel menu à tel emplacement ».

C'est précisément le rôle de la fonction `wp_nav_menu()`.

Comme cela est indiqué **dans le Codex**, cette fonction reçoit un tableau en argument.

Ce tableau peut recevoir de multiples arguments (16 exactement). Tous sont facultatifs mais le premier est recommandé. Nous allons seulement nous intéresser aux principaux :

- `theme_location` : Correspond à l'identifiant de l'emplacement du menu (`primary` ou `social` si on reste sur les exemples cités précédemment)
- `container` : Balise HTML dans laquelle doit se situer le menu. `div` sera utilisée par défaut, l'autre choix possible est `nav` pour être optimisé en HTML 5
- `container_class` : Classe CSS à ajouter à la balise contenant le menu
- `container_id` : Identifiant CSS à ajouter à la balise contenant le menu
- `menu_class` : Classe CSS à ajouter à la balise `ul` du menu (il est possible d'en ajouter plusieurs en séparant les classes par des espaces)
- `menu_id` : Identifiant CSS à ajouter à la balise `ul` du menu
- `depth` : Profondeur des sous-menus autorisée

Pour que cela soit plus concret à vos yeux, voici un exemple assez simple.

Admettons qu'un emplacement de menu en tête a été créé dans le fichier `functions.php` et qu'un menu lui a été associé dans l'administration. Voici le code de la fonction `wp_nav_menu()` à placer dans le fichier `header.php` du thème :

```
<?php
    wp_nav_menu( array(
        'theme_location' => 'entete',
        'container' => 'nav',
        'container_id' => 'main_nav_container',
        'menu_class' => 'menu-entete',
        'depth' => 2,
    ) );
?>
```

Cela affichera le code suivant (volontairement simplifié pour l'exemple) :

```
<nav id="main_nav_container">
    <ul class="menu-entete">
        <li><a href="">Accueil</a></li>
        <li><a href="">À Propos</a></li>
        <li><a href="">Contact</a></li>
    </ul>
</nav>
```

Dans certains thèmes, on peut voir que la fonction `wp_nav_menu()` est appelée dans une instruction `if` contenant la fonction `has_nav_menu()`.

Cette fonction permet de tester si un emplacement existe, si c'est le cas on exécute la fonction `wp_nav_menu()` pour afficher le menu qui lui est associé. On évite ainsi d'obtenir une erreur si l'emplacement n'existe pas.

On peut donc optimiser notre code de la façon suivante :

```

<?php
// Si un emplacement "entete" existe...
if (has_nav_menu('entete')){
// Alors on affiche le menu associé
    wp_nav_menu( array(
        'theme_location' => 'entete',
        'container' => 'nav',
        'container_id' => 'main_nav_container',
        'menu_class' => 'menu-entete',
        'depth' => 2,
    ) );
}
?>

```

Bien entendu, il est possible d’afficher des menus beaucoup plus complexes. Ce n’est toutefois pas l’objet de ce guide.

Ce que nous avons passé en revue vous aidera à mieux manipuler les menus au sein de WordPress.

Découvrons maintenant comment sont gérés...

## Les widgets

Même si un thème ne propose pas de widgets en plus de ceux proposés par WordPress, il est quasi certain qu’une ou plusieurs zones de widgets soient présentes.

Une zone de widgets est un emplacement où l’on peut placer des widgets en se rendant dans *Apparence > Widgets*.

Généralement, ces zones correspondent à la barre latérale (que l’on appelle aussi *sidebar*) et au pied de page d’un site. On peut aussi retrouver des zones de widgets à des endroits plus exotiques.

Voyons comment...

## Créer une zone de widgets

Pour que WordPress affiche le menu *Apparence > Widgets*, il faut qu'une zone de widget existe.

Comme pour les menus, il existe une fonction pour cela. Il s'agit de `register_sidebar()`.

Étudions son fonctionnement en visualisant le code utilisé par le thème Twenty Fifteen :

```
function twentyfifteen_widgets_init() {
    register_sidebar( array(
        'name'          => __( 'Widget Area',
'twentyfifteen' ),
        'id'            => 'sidebar-1',
        'description'   => __( 'Add widgets here to appear
in your sidebar.', 'twentyfifteen' ),
        'before_widget' => '<aside id="%1$s"
class="widget %2$s">',
        'after_widget'  => '</aside>',
        'before_title'  => '<h2 class="widget-title">',
        'after_title'   => '</h2>',
    ) );
}
add_action( 'widgets_init', 'twentyfifteen_widgets_init'
);
```

Tout d'abord, on constate que la fonction `register_sidebar()` est comprise dans une autre fonction (`twentyfifteen_widgets_init()`) qui est associée au hook de type action `widgets_init`.

Concrètement, on peut créer une zone de widgets une fois que les widgets par défaut ont été initialisés (d'où le nom `widgets_init`).

Maintenant que cela a été posé, intéressons-nous aux arguments de la fonction `register_sidebar()`.

Si vous êtes observateur, vous avez dû constater qu'il n'y a en fait qu'un seul argument. Il s'agit d'un tableau comprenant plusieurs entrées (d'où la présence de `array`).

Voici de quoi ce tableau peut se constituer (toutes les valeurs sont facultatives) :

- `name` : Nom de la zone de widgets
- `id` : Identifiant de la zone de widgets
- `description` : Description de la zone de widgets (sera affiché dans l'administration)
- `class` : Classe CSS à ajouter à la zone de widgets **au niveau de l'administration** (pas sur le site)
- `before_widget` : Code HTML à insérer avant chaque widget
- `after_widget` : Code HTML à insérer après chaque widget
- `before_title` : Code HTML à insérer avant chaque titre de widget
- `after_title` : Code HTML à insérer après chaque titre de widget

Pour reprendre l'exemple précédent, on peut dire que la zone de widgets créée par la fonction `twentyfifteen_widgets_init()` possédera les caractéristiques suivantes :

- `Nom` : `Widget Area` en anglais (cela sera traduit si le site est en français)
- `Identifiant` : `sidebar-1`
- `Description` : `Add widgets here to appear in your sidebar.` (sera également traduit grâce à la fonction `__( )`)

- Les widgets seront placés dans des balises `aside` avec un identifiant et des classes personnalisées
- Les titres des widgets seront compris dans des balises `h2` et dotés d'une classe `widget-title`.

## Afficher une zone de widgets

À présent, vous savez désormais comment créer une zone de widgets dans un thème WordPress.

Toutefois, il faut encore l'afficher pour que les visiteurs puissent voir les widgets qui y sont placés.

Pour ce faire, il faut appeler la fonction `dynamic_sidebar()` et lui donner l'identifiant de la zone de widgets que vous désirez afficher.

Dans le cas de l'exemple donné auparavant, cela donnerait :

```
<?php dynamic_sidebar('sidebar-1'); ?>
```

On retrouve généralement la fonction `dynamic_sidebar()` dans le fichier `sidebar.php` des thèmes. Elle sera souvent placée dans une balise HTML avec un identifiant ou une classe spécifique afin de pouvoir styliser la zone de widgets convenablement.

## Récapitulons

Dans cette dernière partie, nous avons étudié comment les menus et les zones de widgets sont créés au sein d'un thème WordPress.

Les fonctions nécessaires commencent à vous faire toucher du bout des doigts la programmation de thèmes WordPress.

Avant de passer au test de fin de chapitre, je vous invite à tester vos connaissances avec le quiz lié à cette partie.

Poursuivez ensuite avec le test final afin de mettre en pratique les concepts et techniques abordés tout au long de ce chapitre.

**RELOOKER SON THÈME**

# **CHAPITRE 5**

## **COMMENT RELOOKER SON THÈME EN PRATIQUE**

---

**Apprenez à personnaliser  
un thème WordPress  
avec les bonnes méthodes  
et les outils adéquats**

## Chapitre 5.1

# Relooker son thème en direct, la fausse bonne idée

La première fois que l'on désire relooker un thème, on peut avoir deux états d'esprit.

Soit on est très hésitant car on ne veut rien casser, soit on est atteint par ce que j'appelle le syndrome du chien fou, c'est à dire que l'on a envie de tout modifier, voir comment tout fonctionne, etc.

Malheureusement dans tous les cas, la plupart des gens ne savent pas qu'il est malvenu de personnaliser un thème WordPress directement.

## 3 raisons de ne pas le faire

Tout d'abord, lorsqu'on modifie les fichiers PHP de son thème, **on prend le risque de rendre son site inaccessible.**

Un oubli de point-virgule, une parenthèse ou une accolade mal refermée et pan ! Au lieu de voir votre site, vos visiteurs verront une belle page blanche.

Avouez que ce n'est pas le genre d'expérience que vous désirez leur procurer.

Si vous êtes plus attentif et que le code que vous avez ajouté ne souffre d'aucune erreur pouvant faire planter votre site, il y a encore le risque que le résultat ne soit pas tel que vous l'attendez.

Vous risquez donc de faire de mauvaises modifications et **d'afficher des choses non pertinentes à vos visiteurs.**

Si cela peut vous rassurer, personne ne réussit ses modifications du premier coup.

*N'importe quel développeur, amateur ou confirmé procède par itérations successives jusqu'à ce qu'il trouve la bonne solution.*

Vous devez apprendre à faire de même.

Le dernier risque à prendre en compte est que **le thème WordPress que vous voulez personnaliser, qu'il soit gratuit ou payant, peut être mis à jour par son auteur.**

Dès lors, que vous ayez fait deux ou cinquante modifications n'importe peu, elle seront toutes perdues si vous cliquez sur le bouton « Mettre à jour ».

Nous sommes d'accord pour dire que ce n'est pas ce que vous désirez.

Ces trois raisons nous imposent de trouver des solutions pour relooker un thème en toute sécurité.

Nous allons voir comment procéder dans la suite de cette partie ainsi que dans l'ensemble de ce chapitre.

La première question que l'on doit se poser est la suivante : Faut-il...

## **Travailler en local ou sur son serveur**

Vous ne le savez peut-être pas, mais il est possible d'installer WordPress sur votre ordinateur.

Pour cela, il est nécessaire d'installer un logiciel qui va simuler un serveur et lui associer une base de données afin de fournir tout ce dont WordPress a besoin pour fonctionner.

*Le principal avantage de travailler en local est qu'il n'y a pas besoin de connexion internet, puisque le site est installé sur votre ordinateur.*

Une fois qu'un fichier est modifié, il y a juste à sauvegarder et à actualiser son navigateur pour voir le résultat.

En clair, il n'y a pas besoin de transférer le fichier que l'on vient de modifier sur son site de test sur son serveur pour voir les résultats.

Un problème se pose toutefois lorsque l'on travaille en local.

*Dans certains cas, les modifications que vous effectuerez sur votre thème ne seront pas reproduites lorsque vous les mettrez en place sur votre serveur.*

Cela peut sembler étrange mais cela m'est arrivé à plusieurs reprises.

Quelle en est la cause ?

L'environnement pardi !

N'ayez crainte, je ne vais pas vous parler de réchauffement climatique.

Non, dans ce contexte un environnement correspond à une configuration. Pour un serveur, cela rassemble l'ensemble de ses paramètres.

Chaque hébergeur configure ses serveurs comme bon lui semble. Ils ont tous un environnement différent.

C'est également le cas pour votre installation en local. **Certains paramètres peuvent varier** et cela peut produire des problèmes, d'où ma réserve quant au fait de travailler ainsi.

Un développeur pourra certainement faire face à ce type d'imprévu mais il est inutile faire prendre ce genre de risque au débutant que vous êtes.

*Après tout vous désirez juste relooker votre thème, pas chercher la petite bête pour que votre thème modifié en local fonctionne correctement sur votre site.*

Afin d'éviter de vous confronter à ce genre de problème, il vaut mieux travailler dans un sous-dossier de votre serveur.

De plus, grâce à un éditeurs de code comme Brackets accompagné du client FTP Cyberduck, il est possible de travailler sur un site de test installé sur son serveur sans avoir besoin de transférer manuellement des fichiers. Tout se fait automatiquement.

Note : Pour rendre votre site de test privé, installez l'extension **Private Only**.

La seconde question à se poser est la suivante : Doit-on...

## **Dupliquer son site ou travailler sur une installation vierge**

Lorsque vous relookez un thème, vous vous situez forcément dans l'un de ces deux cas de figure :

1. Vous créez un site en partant de zéro
2. Vous changez le design d'un site (refonte)

Si vous êtes dans le premier cas, vous n'aurez rien à dupliquer donc la réponse est assez simple : Installez WordPress dans un sous-répertoire de votre site et travaillez dessus.

Sur le site accessible aux visiteurs, vous pouvez mettre en place une page d'attente, il existe des extensions pour cela. **WP Maintenance** par exemple

Une fois que le site sera opérationnel, vous pourrez le passer en ligne.

En revanche, si vous disposez déjà d'un site, il est plus difficile de répondre.

Étant donné que vous allez travailler uniquement sur le thème, on peut être tenté de se dire qu'installer un nouveau site WordPress dans un sous-répertoire fera largement l'affaire.

Pourtant, ce n'est pas si simple.

*En procédant de la sorte, vous ne verrez pas comment le thème prend en charge le contenu, ni comment les fonctionnalités apportées par les extensions sont affichées.*

Vous pourriez alors installer quelques extensions et publier du contenu factice mais rien ne garantira que tout s'affichera correctement sur votre site.

C'est pourquoi il est préférable de créer une copie de son site dans un sous-répertoire de son serveur pour procéder au relooking de son thème.

Dupliquer un site WordPress est assez fastidieux si l'on s'y prend manuellement (copier les fichiers, dupliquer une nouvelle base de données, changer certaines valeurs dans la base etc.).

Heureusement, il existe des extensions WordPress qui permettent de gagner du temps. Dans la suite de cette partie, nous allons voir...

# Comment dupliquer son site avec l'extension Duplicator

Après avoir testé toutes les extensions populaires pouvant dupliquer un site, je me suis arrêté sur **Duplicator**.

Contrairement aux autres extensions que j'ai pu tester, le principal avantage de Duplicator est qu'il n'y a pas besoin de réinstaller WordPress pour procéder à la création de la copie d'un site.

Le seul point « ennuyeux » est qu'il faut créer une nouvelle base de données chez votre hébergeur.

Note : Si vous ne disposez que d'une seule base de données, vous allez devoir migrer votre site manuellement en dupliquant les tables de votre bases. Ce cas étant assez exceptionnel, **rendez-vous sur cette page pour en savoir plus**.

Si vous pouvez créer une autre base, consultez le tutoriel de l'extension Duplicator fourni avec Relooker son Thème (fiche n°5) pour voir comment faire.

## Récapitulons

Il faut prendre ses précautions lorsque l'on se lance dans la personnalisation d'un thème WordPress.

Quelque soit le projet, il faut toujours commencer par travailler sur un site de développement au sein de son serveur qui soit inaccessible aux visiteurs.

Cela permet de tester, améliorer, peaufiner son site en toute tranquillité. Pendant ce temps, la version « officielle » continue de fonctionner pour les visiteurs.

En procédant ainsi, vous aurez l'assurance que vos modifications auront bien l'effet escompté lorsque vous mettrez le thème relooké en ligne.

## Chapitre 5.2

# **Utiliser un thème enfant, la pratique indispensable**

S'il n'y avait qu'un seul concept à retenir de ce guide, cela serait celui des thèmes enfants/parents.

Encore trop peu de personnes savent qu'il faut systématiquement utiliser un thème enfant lorsque l'on relooke un thème WordPress.

Heureusement, vous êtes en train de lire ce guide. Sans le savoir, vous venez d'entrer dans le groupe de ceux qui savent qu'un thème enfant est indispensable pour personnaliser un thème WordPress.

## **Mais au fait, de quoi s'agit-il exactement ?**

Chez les êtres humains, lorsque l'on veut en savoir plus sur les enfants, on regarde souvent ce qu'il se passe chez les parents.

C'est exactement la même chose chez les thèmes WordPress.

Puisqu'il existe des thèmes enfants, c'est que des thèmes parents doivent se trouver dans les parages.

La seule différence par rapport à nous est qu'un thème enfant ne possède qu'un seul thème parent.

Du coup, à quoi correspond ce fameux thème parent ?

Et bien, c'est tout simplement le thème que vous avez choisi d'utiliser pour votre site.

Vous l'avez peut-être téléchargé sur le répertoire officiel des thèmes de WordPress, acheté sur Themeforest ou dans une autre boutique.

En fait, peu importe où vous vous êtes procuré votre thème. Il s'agit d'un thème parent.

De temps en temps, l'auteur met à jour son thème pour corriger divers bugs, apporter de nouvelles choses, etc. Vous recevez donc une notification dans l'administration vous invitant à faire cette mise à jour.

*Si par malheur vous avez modifié le code des fichiers de votre thème, vous pourrez dire adieu à toutes vos améliorations une fois la mise à jour effectuée.*

En effet, une mise à jour supprime l'ancienne version pour installer la nouvelle.

WordPress ne se soucie pas des modifications que vous avez pu faire. Il remplace les fichiers du thème par les nouveaux, point final.

Afin d'éviter cela, on pourrait se dire qu'il suffit de ne plus mettre son thème à jour pour régler le problème.

C'est à mon sens un choix assez risqué.

En effet, si l'auteur du thème corrige une faille de sécurité permettant à des pirates de prendre le contrôle de votre site, vous n'en bénéficierez pas.

Votre site deviendra donc une cible de premier choix.

Il a donc fallu trouver une solution et c'est là que les thèmes enfants entrent en scène.

# Comment fonctionne un thème enfant

Comme ce que vous avez pu lire auparavant le laisse penser, **les thèmes enfants permettent de personnaliser un thème WordPress (le thème parent) sans l'altérer directement.**

Le thème parent peut donc être mis à jour sans problème car les modifications sont placées dans le thème enfant.

Au niveau de la structure, un thème enfant se compose au minimum de deux fichiers :

- `style.css` : qui contiendra le code CSS pour personnaliser l'apparence du site
- `functions.php` : qui chargera le fichier CSS du thème parent et contiendra éventuellement des fonctionnalités supplémentaires

Bien entendu, il est possible d'inclure d'autres fichiers au sein d'un thème enfant. C'est d'ailleurs grâce à cela que l'on va pouvoir personnaliser le thème parent.

*En fait, tout fichier de template présent dans le thème enfant sera utilisé à la place du fichier portant le même nom dans le thème parent.*

Par fichier de template, j'entends :

- les fichiers de la hiérarchie des templates (voir le schéma dans les ressources)
- les fichiers de base d'un thème (`header.php`, `footer.php`, `sidebar.php`, `comments.php`, etc.)
- les fichiers chargés grâce à la fonction `get_template_part()`

Par exemple, si votre thème enfant comporte un fichier `single.php` pour l’affichage des articles, le fichier `single.php` du thème parent sera ignoré par WordPress.

**Pour faire une modification, il suffit de dupliquer un fichier du thème parent dans le thème enfant et de le personnaliser.**

Note : Les fichiers chargés par les fonctions d’inclusion issues de PHP (étudiées dans la partie 3 du chapitre 4) ne seront pas pris en compte dans le thème enfant.

Concernant le fichier `functions.php`, celui du thème enfant ne remplace pas celui du thème parent. Si c’était le cas, les fonctionnalités du thème parent seraient supprimées et le thème ne pourrait plus s’afficher.

WordPress charge donc les fichiers `functions.php` du thème enfant et du thème parent.

Autre point à retenir, **le fichier `functions.php` d’un thème enfant est chargé avant celui du thème parent.**

Cela a son importance car si le thème parent est bien conçu, il sera possible de redéfinir des fonctions créées dans le thème parent au sein du thème enfant.

Si vous regardez ce que contient le fichier `functions.php` d’un thème, vous trouverez certainement une condition du genre :

```
if ( ! function_exists( 'nom_de_fonction' ) ) :  
    function nom_de_fonction(){  
        // Code de la fonction  
    }  
endif;
```

La fonction `function_exists` permet de tester si une fonction existe et le caractère `!` signifie la négation (cela a été abordé dans la partie 2 du chapitre 4).

*Cette condition permet donc de tester si une fonction n'existe pas.*

Si elle n'existe pas, la fonction est déclarée. En revanche, si elle existe déjà dans le thème enfant, le fichier `functions.php` étant chargé avant, la fonction ne sera pas déclarée à nouveau dans le thème parent.

C'est tout ce que vous avez besoin de savoir concernant le fonctionnement des thèmes enfants. Voyons maintenant...

## **Comment créer un thème enfant**

Nous avons beaucoup parlé des thèmes enfants mais nous n'avons pas encore vu comment en créer un.

Étant donné qu'un thème enfant vierge ne comporte que deux fichiers, vous allez voir que c'est assez simple. Nous allons procéder en 3 étapes.

### **1. Créer le dossier de votre thème enfant**

Ici, rien de compliqué. Tout ce que vous avez à faire, c'est de créer un dossier dans lequel vous placerez les fichiers du thème enfant.

Vous pouvez le faire sur votre ordinateur ou dans le dossier `/wp-content/themes/` de votre site de test (pas sur le site accessible aux visiteurs, rappelez-vous la partie précédente).

Pour être certain de le reconnaître, si votre thème s'appelle `toto`, nommez-le `toto-enfant` ou `toto-child` si vous préférez l'anglais.

### **2. Créer le fichier `style.css`**

Dans votre dossier, créez un fichier `style.css` et insérez-y le code suivant :

```
/*  
Theme Name:      Toto Enfant  
Description:     Thème enfant du thème Toto  
Author:          Créateur du thème Toto  
Author URI:      http://monsite.com  
Template:        toto  
Text Domain:     toto-enfant  
*/
```

Regardons de plus près de quoi se compose ce code et ce que vous devez modifier pour que votre thème enfant soit fonctionnel.

- **Theme Name** : Nom de votre thème enfant
- **Description** : Donnez plus d'informations au sujet de votre thème enfant
- **Author** : Indiquez votre nom
- **Author URI** : Inscrivez l'adresse de votre site
- **Template** : Nom du dossier du thème parent (attention, la plupart des dysfonctionnements des thèmes enfants viennent de là)
- **Text Domain** : Identifiant permettant de repérer les chaînes de traduction ajoutées. Cela correspond au second paramètre des fonctions `__( )` et `_e( )`. Si vous ne vous servez pas de ces fonctions, vous pouvez supprimer cette ligne (par contre votre thème enfant ne sera pas traduisible)

Il existe d'autres paramètres mais il est inutile de les présenter ici. Le but de Relooker son Thème est d'aller à l'essentiel.

Comme cela a été indiqué, il faut absolument que la valeur associée à `Template` : corresponde au nom du dossier du thème parent.

Dans le cas contraire, vous obtiendrez le message suivant :

### Thèmes endommagés

Les thèmes suivants sont installés, mais incomplets. Les thèmes doivent avoir au moins une feuille de style et un modèle.

Nom	Description	
Theme Enfant TwentyFifteen	Le thème parent est manquant. Merci d'installer le thème parent « twenty-fifteen ».	<a href="#">Supprimer</a>

Quand votre fichier `style.css` sera opérationnel, vous pourrez passer à la dernière étape, à savoir...

## 3. Créer le fichier `functions.php`

Toujours dans votre dossier, créez un fichier `functions.php` et ajoutez-y le code suivant :

```
<?php

function rst_enqueue_styles() {
    wp_enqueue_style( 'parent-style', get_template_
directory_uri() . '/style.css' );
}
add_action( 'wp_enqueue_scripts', 'rst_enqueue_styles' );
```

Étudions ces lignes de code.

À première vue, nous pouvons voir qu'une fonction `rst_enqueue_styles()` est déclarée et associée au hook action `wp_enqueue_scripts`.

Pour vous en dire un peu plus, ce hook permet de dire à WordPress de charger des styles CSS ou scripts Javascript dans le thème.

Dans notre cas, il s'agit de charger le fichier `style.css` du thème parent.

*En effet, charger ce fichier est indispensable sinon le thème enfant ne pourra pas reprendre les styles CSS du parent. Le thème enfant ne peut pas s’afficher correctement sans lui.*

L’avantage avec cette méthode est qu’il n’y a rien à modifier. Cela fonctionnera pour n’importe quel thème enfant (vous pouvez néanmoins changer le nom de la fonction `rst_enqueue_styles()`).

Note : Auparavant, une autre méthode était utilisée pour charger les styles du thème parent au niveau du fichier `style.css` du thème enfant mais ce n’est plus une bonne pratique.

Félicitations, vous savez désormais créer un thème enfant. Pour vous entraîner, vous trouverez un thème enfant créé pour le thème par défaut Twenty Fifteen dans les ressources fournies avec le guide.

Découvrons maintenant un...

## **Exemple d’utilisation d’un thème enfant**

Pour vous montrer comment relooker un thème aisément avec un thème enfant, je vous propose d’imaginer la situation suivante.

Admettons que vous ayez besoin de personnaliser l’en-tête de votre site en ajoutant un menu supplémentaire.

Si vous vous rappelez ce que nous avons vu dans les chapitres précédents, nous avons besoin de plusieurs choses :

- Déclarer le menu dans le fichier `functions.php`
- Afficher le menu dans `header.php`
- Styler le tout en CSS dans le fichier `style.css`

Une fois que vous aurez créé votre thème enfant, il vous faudra ouvrir le fichier `functions.php` pour y ajouter le code ci-dessous :

```
function rst_top_menu(){
    register_nav_menu('topmenu' => 'Menu du haut');
}
add_action( 'after_setup_theme' , 'rst_top_menu' );
```

Ça y est, l'emplacement de menu est déclaré. Ajoutons maintenant le code pour l'afficher.

Commençons par dupliquer le fichier `header.php` du thème parent et plaçons-le dans le thème enfant. Il ne reste plus qu'à l'ouvrir pour y afficher notre nouveau menu.

Si on se reporte à ce qui a été abordé dans la partie 7 du chapitre 4, nous pouvons utiliser le code suivant :

```
<?php
    wp_nav_menu( array(
        'theme_location' => 'topmenu',
        'container' => 'nav',
        'container_id' => 'top_nav_container',
        'depth' => 1,
    ) );
?>
```

Ensuite, il ne reste plus qu'à créer le menu dans l'administration de WordPress et à styler le tout dans le fichier `style.css`.

On peut notamment se servir de l'identifiant de la balise contenant le menu pour créer des sélecteurs efficaces :

```
#top_nav_container{
    /* Code de la balise contenant le menu */
}

#top_nav_container ul{
    /* Code du menu */
}

#top_nav_container ul li{
    /* Code des éléments du menu */
}

#top_nav_container ul li a{
    /* Code des liens des éléments du menu */
}
```

Vous verrez avec l'inspecteur de code que WordPress ajoute des classes au menu et à ses éléments. Vous pouvez vous en servir pour personnaliser plus finement votre menu.

Voilà, ceci n'est qu'un exemple rapide mais il est possible d'aller beaucoup plus loin avec un thème enfant. Une fois que vous maîtrisez le principe la seule limite est votre imagination.

Terminons maintenant avec une question fréquemment posée par les personnes ne connaissant pas le système des thèmes enfants.

## **Que faire si l'on a modifié directement un thème parent**

Et oui, ce sont des choses qui arrivent. On est parfois pressé de faire des modifications et l'on bidouille un thème WordPress directement.

Quand on se rend compte de son erreur suite à la découverte du principe des thèmes enfants ou à la mise à jour de son thème (et à la perte de toutes les modifications effectuées), on désire se couvrir pour la suite.

Malheureusement, il est difficile de se souvenir de toute ce que l'on a fait dans son thème...

Il n'y a hélas pas de méthode miracle. Cela va être fastidieux.

Il faut procéder de la façon suivante :

- Récupérer la version originale de votre thème (celle que vous avez relookée)
- Comparez les fichiers un à un pour voir ce qui a été modifié (**cet outil** pourra être utile pour repérer les différences entre le contenu de deux fichiers)
- Commencez par les fichiers `style.css`, `functions.php` et les fichiers que vous vous souvenez avoir modifié
- Répercutez les modifications dans un thème enfant
- Installez le thème parent et le thème enfant
- Activez le thème enfant dans l'administration de WordPress

Ceci n'est pas un exercice facile mais il permet de se mettre à l'abri et de pouvoir mettre à jour votre thème dans le futur sans avoir de sueurs froides.

Attention : Des incompatibilités entre le thème enfant et le nouveau parent peuvent survenir. Faites le tour de votre site après chaque mise à jour de thème.

## **La seule entorse que vous pouvez faire à la règle**

Et oui, il y a bien une exception. Toutefois, il y a de grandes chances pour qu'elle ne s'applique pas à votre cas.

*En clair, si vous avez acheté votre thème cela ne vous concerne pas. Idem si vous l'avez téléchargé gratuitement sur le répertoire des thèmes de WordPress.*

Le seul moment où il ne faut pas utiliser de thème enfant est lorsque vous démarrez un projet avec ce que l'on appelle un « starter theme », c'est à dire un thème de démarrage.

Ce sont des thèmes dépouillés (avec peu de code CSS) que l'on peut utiliser **pour créer un thème sur mesure**. Cela permet de partir d'une bonne base.

On peut notamment citer :

- **Underscores**
- **HTML5Blank**
- **Bones**
- **Naked WordPress**

Ces thèmes n'ayant pas vocation à être mis à jour, il n'y a aucune raison de vouloir leur associer des thèmes enfants.

Encore une fois, cela est un cas de figure bien précis. Votre projet se situe certainement dans le cas général. Vous devrez donc passer par un thème enfant.

## Récapitulons

Lorsque vous relookez un thème, utiliser un thème enfant est indispensable pour ne pas se retrouver tout penaud après une mise à jour.

Cela permet aussi de mieux visualiser les modifications que l'on a apporté à son thème WordPress.

Concernant leur fonctionnement, il faut retenir que :

- Chaque fichier de template présent dans un thème enfant remplace celui du thème parent
- Les fichiers `functions.php` du thème enfant et du parent sont chargés par WordPress (celui de l'enfant l'est en premier)
- Les styles CSS définis dans le thème enfant viennent s'ajouter aux styles existants

Voilà, vous avez toutes les cartes en main pour commencer à relooker un thème. Continuez la lecture pour entrer davantage dans le concret.

## Chapitre 5.3

# Simuler des modifications avec l'inspecteur de code

Si vous travaillez sur votre premier relooking de thème, à ce stade vous devriez avoir installé un éditeur de code (Brackets ou un autre), avoir un site de développement opérationnel et une liste TOP à faire.

Vous savez aussi qu'utiliser un thème enfant est indispensable pour relooker un thème WordPress.

Vous avez également acquis des bases en HTML, CSS et sur le fonctionnement de WordPress.

*Autant dire que vous avez parcouru un sacré chemin depuis le début de la lecture de ce guide !*

Seulement voilà, maintenant il faut passer à l'action. L'heure de mettre les mains dans le cambouis est arrivée.

**Vous n'allez plus travailler sur des exemples fictifs. Vous allez vous occuper d'un thème WordPress important à vos yeux.**

Pour vous éviter de passer des heures à chercher comment faire ceci ou cela, nous allons apprendre à nous servir de l'inspecteur de code.

Nous avons parlé de l'inspecteur de code dans le chapitre 1. Si vous avez bonne mémoire, vous devez vous rappeler qu'il sert à interagir en direct avec une page web.

Cela évite de faire des aller-retours incessants entre son éditeur de code et son navigateur.

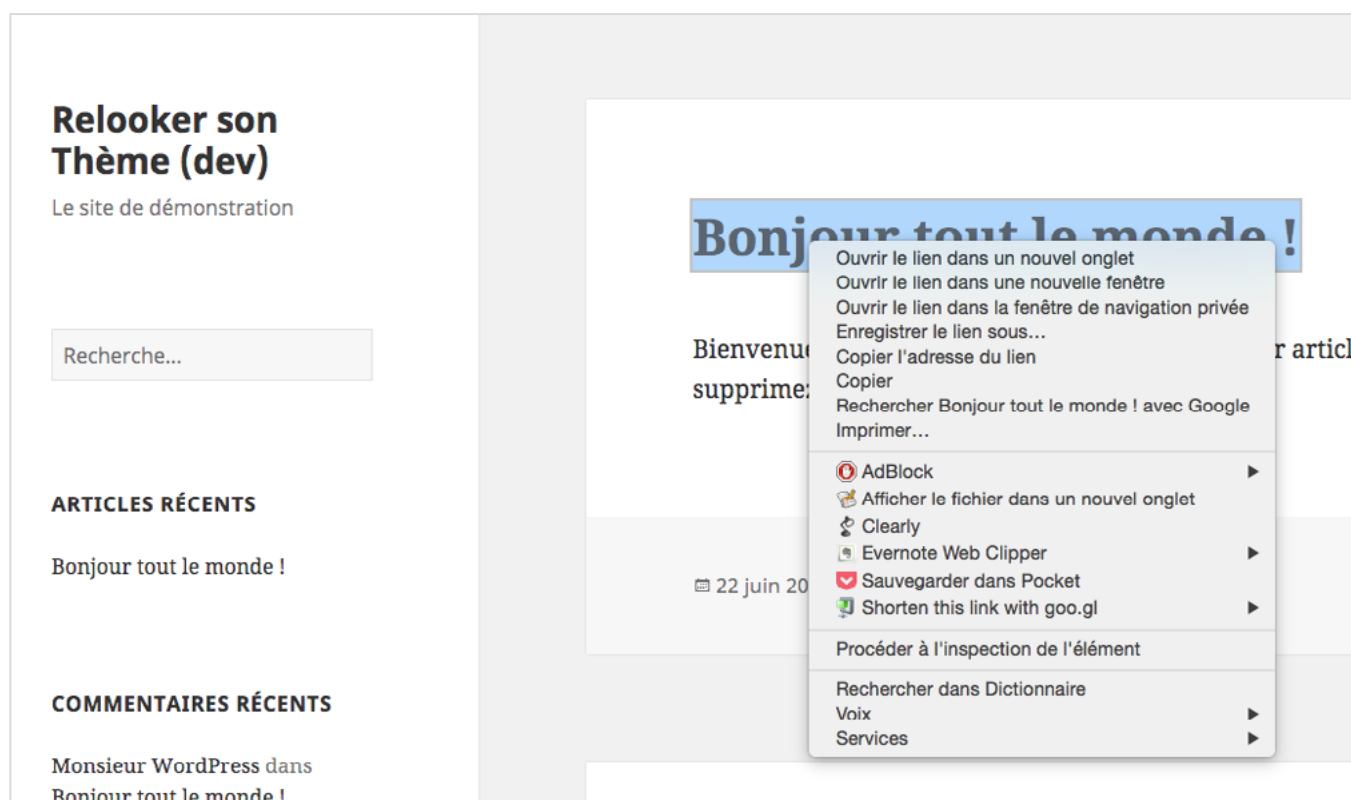
*Dans le cas d'un relookeur de thème débutant, l'inspecteur de code sert principalement à tester du code CSS sur des éléments HTML.*

Regardons ce qu'il est possible de faire avec l'inspecteur de code du navigateur Chrome. Bien sûr, il est possible de faire de même avec les inspecteurs des autres navigateurs.

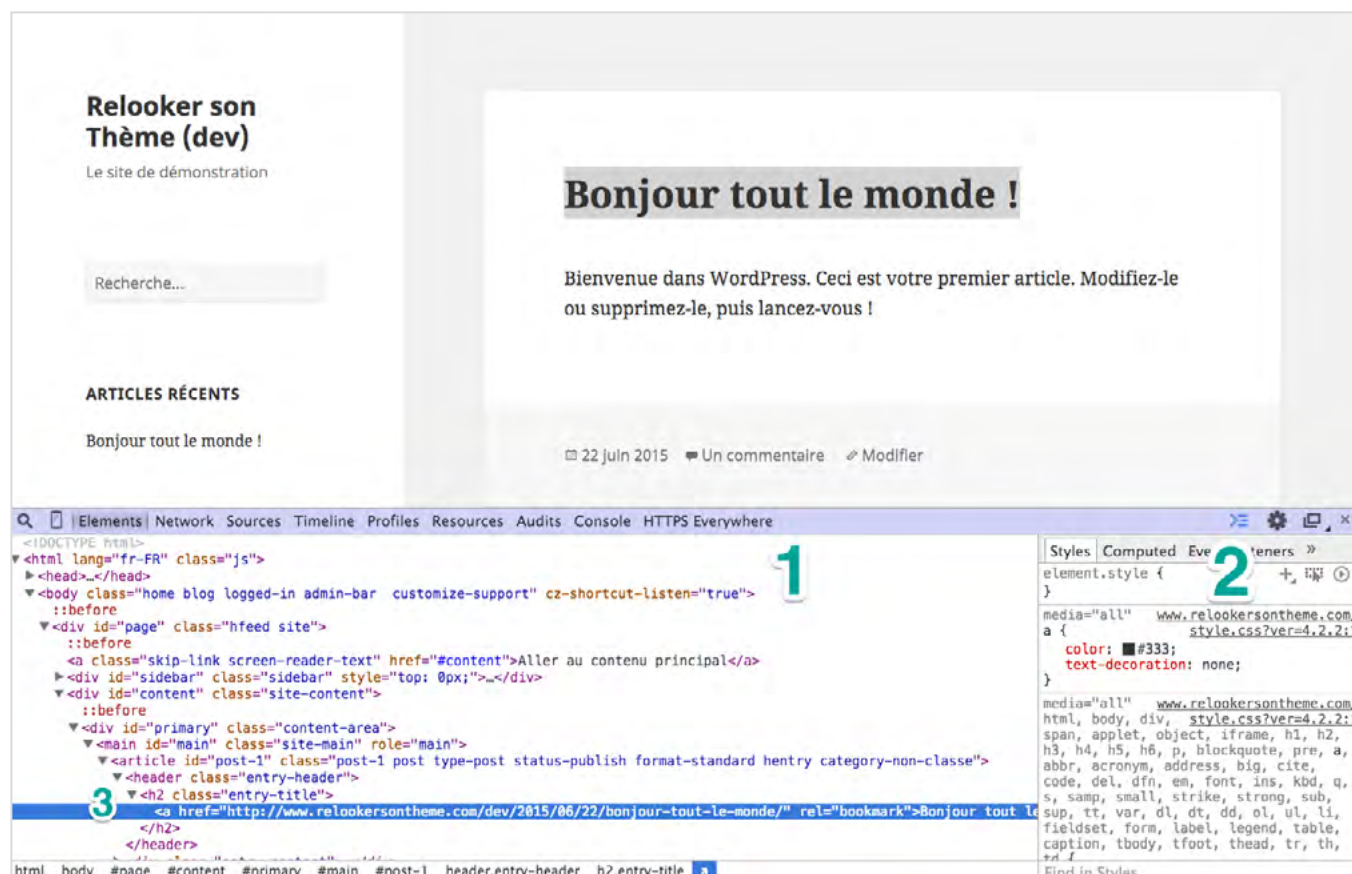
## Structure de l'inspecteur de code

Pour étudier de quoi se compose un inspecteur de code, nous allons utiliser le thème Twenty Fifteen, un des thèmes par défaut de WordPress.

Commençons par faire un clic droit sur un titre d'article et sélectionnons *Procéder à l'inspection de l'élément* (d'autres intitulés sont employés selon les navigateurs).



L'inspecteur de code va s'afficher en bas de la page, voici ce que vous obtiendrez :



Étudions les deux principales zones dont l'inspecteur se compose :

1. **Le code HTML de la page** : On y retrouve les balises et leurs attributs (identifiants, classes et autres). Tout est coloré pour que la lecture du code soit simplifiée. L'arborescence des balises est clairement visible, c'est ce que l'on appelle le DOM. Il est possible d'ouvrir et de fermer des balises grâce aux flèches situées à leur gauche.
2. **Le code CSS associé aux éléments** : Quand un élément est sélectionné, le code CSS qui lui est appliqué s'affiche à droite. On peut incorporer de nouvelles règles pour chaque élément HTML et donc prévisualiser les résultats sans éditeur de code.

Dans l'image ci-dessus, le point numéro 3 montre que la balise lien du titre de l'article « Bonjour tout le monde ! » est sélectionnée.

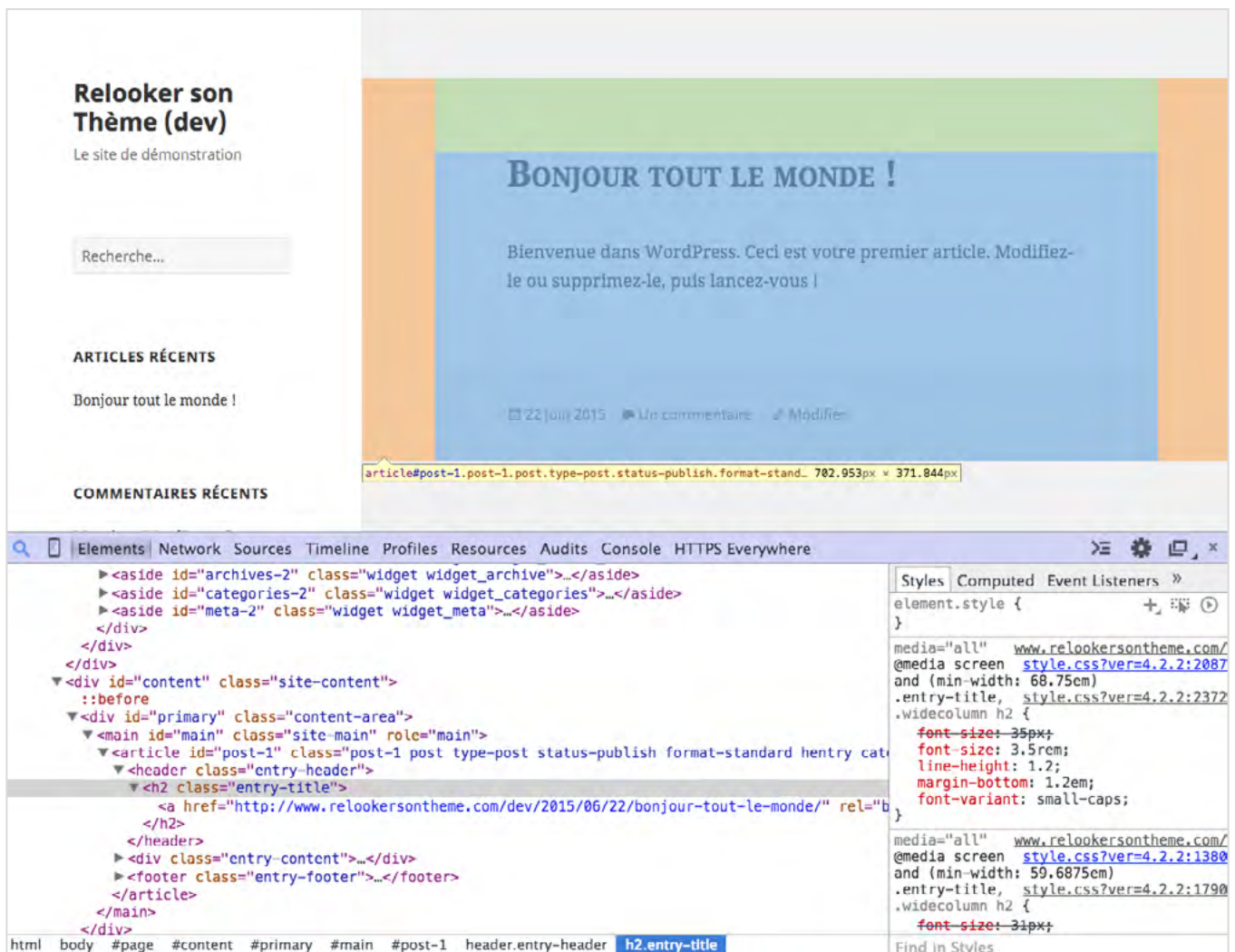
On découvre que le code CSS qui lui est associé concerne tous les liens de la page (sélecteur `a`) et que cela concerne la couleur (`#333`) et la non décoration du texte (suppression du soulignement par défaut des liens).

Note : En bas de la partie HTML, on voit l'enchevêtrement des balises, classes et identifiants. Cela vous sera très utile pour créer des sélecteurs plus précis que ceux qui existent déjà pour que votre code CSS s'applique.

## Trouver un élément grâce à l'inspecteur de code

Une fois l'inspecteur de code lancé, il est possible d'en savoir plus sur n'importe quel élément assez facilement.

En cliquant sur la loupe en haut à gauche de l'inspecteur, puis en passant la souris au dessus de la page, vous verrez les éléments se matérialiser à l'écran :



Que voit-on à l'écran ?

- En bleu, on retrouve le contenu de l'élément
- En vert, on a les marges internes
- En orange, on trouve les marges externes

Une petite infobulle indique le type d'élément (balise `article`), l'identifiant (`#post-1`), les classes associées ainsi que les dimensions du bloc.

Il ne reste plus qu'à cliquer dessus pour obtenir les informations relatives au CSS sur la droite de l'inspecteur (ce qui n'a pas été fait dans la capture d'écran).

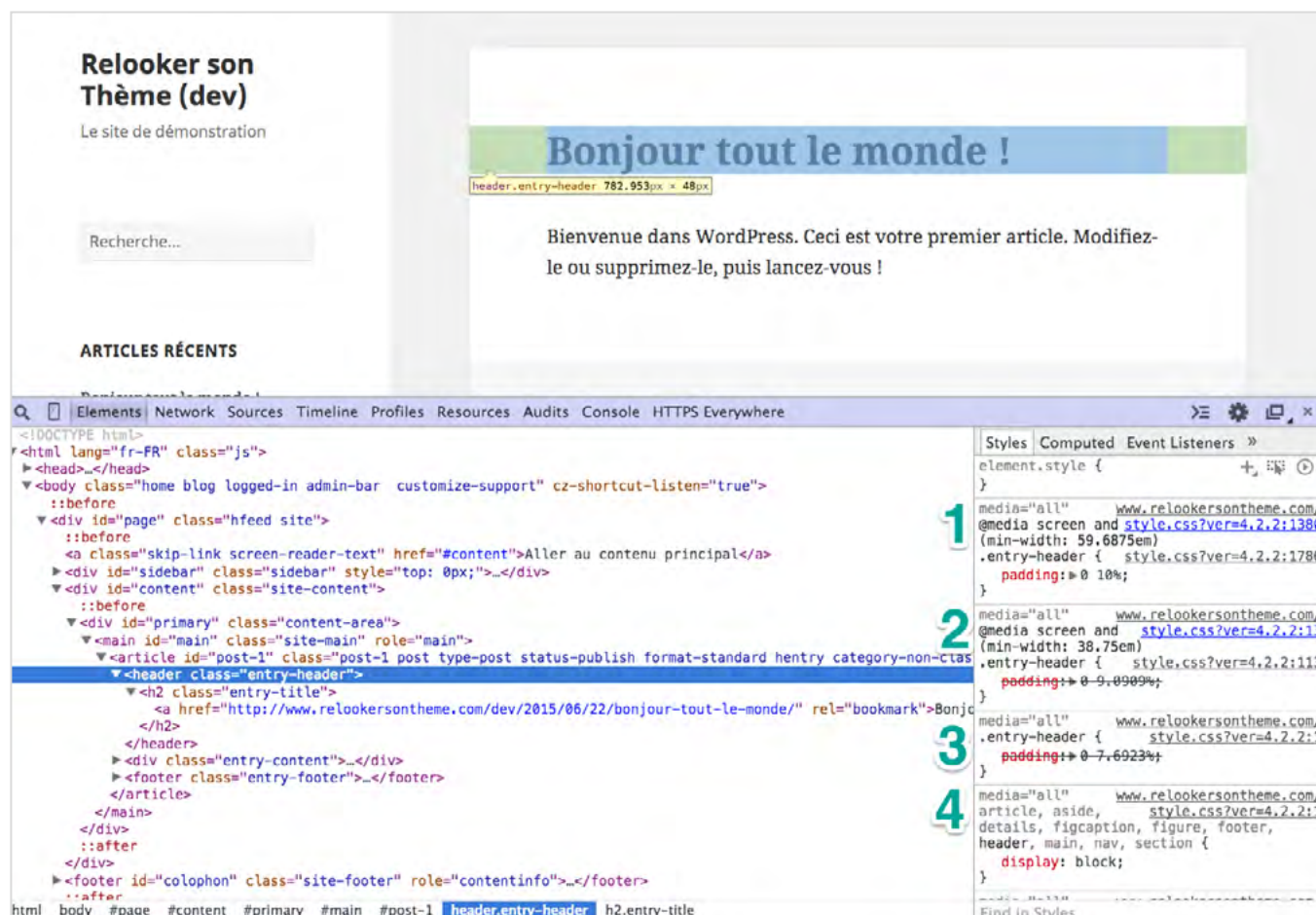
Au delà de la loupe, vous pouvez cliquer sur la barre située en bas de l'inspecteur pour naviguer entre les éléments de la page.

Il est également possible de procéder à un nouveau clic droit puis *Procéder à l'inspection de l'élément* pour afficher des informations relatives à un nouvel élément.

## Étudier les propriétés de chaque élément

Nous avons vu que les propriétés CSS appliquées à un élément s'affichent sur la droite de l'inspecteur une fois celui-ci sélectionné.

En faisant défiler le contenu, on peut voir tous les styles associés à un élément s'afficher par importance dans l'onglet *Styles*. Cela signifie que les sélecteurs les plus importants s'afficheront en haut :



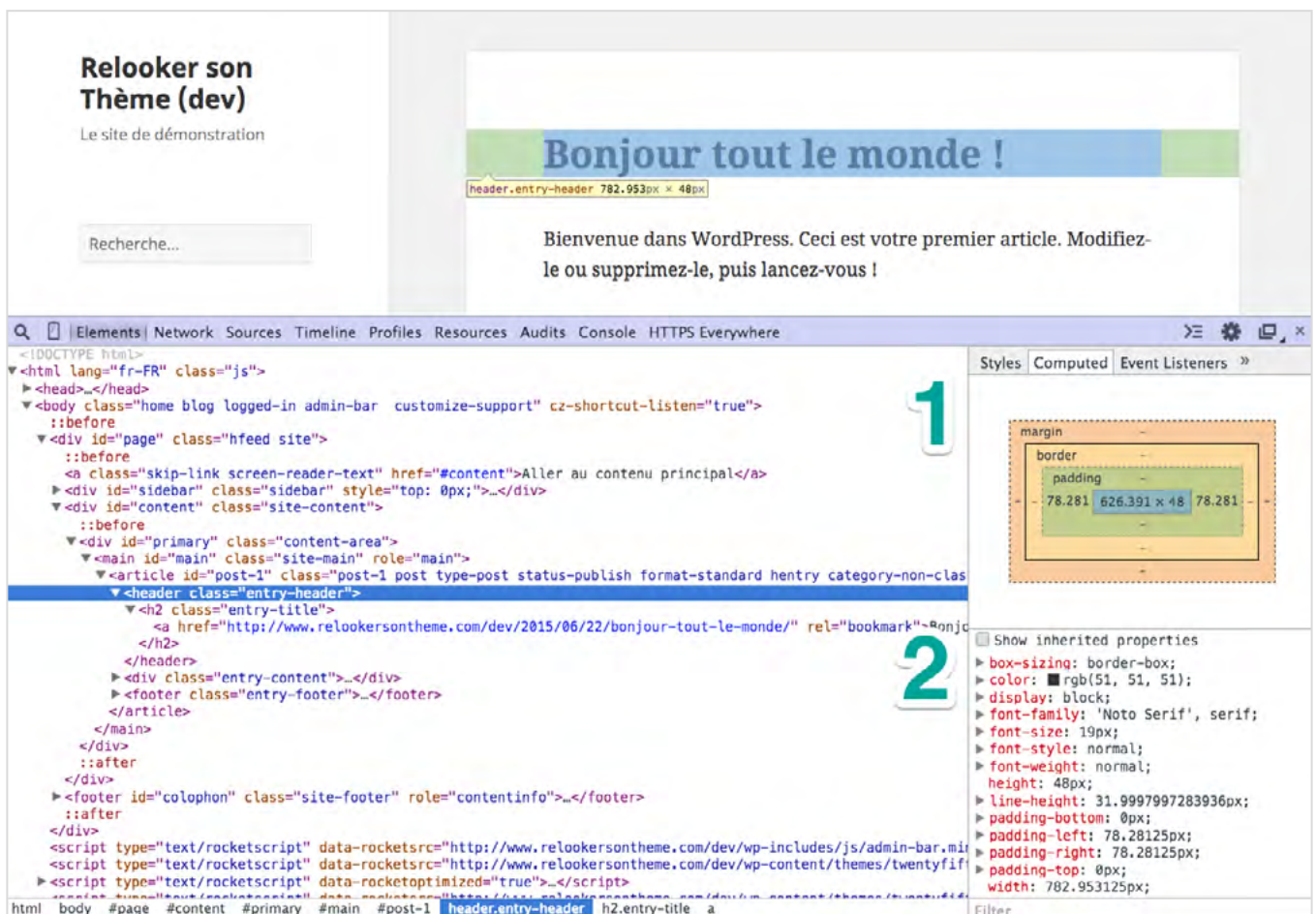
En se basant sur l'exemple ci-dessus on constate que :

1. Le padding s'applique car la règle est la plus importante (et la media query correspond à la largeur de la fenêtre)

2. Le padding n'est pas appliqué car la largeur de la fenêtre est trop importante
3. Le padding par défaut pour la classe `.entry-header` n'est pas appliqué car la media query du point 1 correspond mieux
4. La propriété `display` s'applique à l'élément header car aucune propriété plus importante n'est présente

Ce mode de visualisation est intéressant mais ce n'est pas pratique pour voir toutes les règles CSS d'un seul coup d'œil.

L'onglet *Computed* nous permet d'obtenir ce mode de visualisation :



1. Schéma de visualisation du contenu, des marges internes, des bordures et des marges externes. Passer la souris sur le schéma met les zones correspondantes en surbrillance au niveau de la page web.

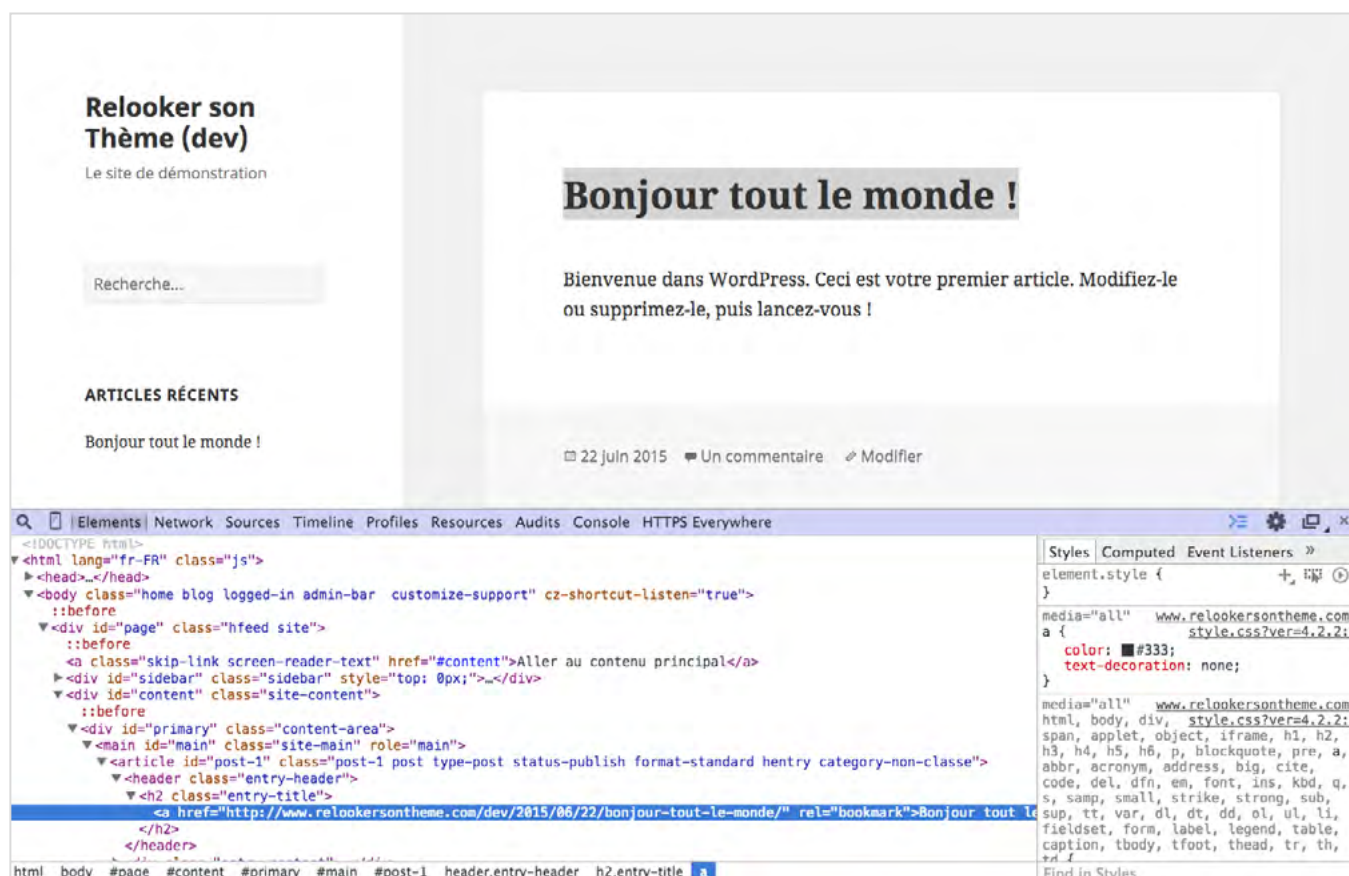
2. Résumé des règles CSS s'appliquant à un élément classées par ordre alphabétique. Cocher la case *Show inherited properties* affichera toutes les propriétés CSS existantes ainsi que leurs valeurs.

## Personnaliser un élément avec l'inspecteur de code

Jusqu'à présent, nous n'avons fait qu'observer les attributs d'une page web. Il est maintenant temps de procéder à votre première personnalisation.

Imaginons que nous désirons changer le style des titres articles. Que devons-nous faire ?

Tout d'abord, il faut inspecter le code lié à un des titres. Nous arrivons sur une balise de lien contenue dans une balise h2.



Si nous tentions d'ajouter du code CSS au niveau du sélecteur `a`, il s'appliquerait à tous les liens du site (sauf si un sélecteur plus puissant applique d'autres règles).

Ce n'est donc pas une bonne solution car nous désirons cibler uniquement les titres des articles.

Voyons ce qu'il se passe lorsque l'on clique sur la balise h2 dans laquelle est contenue ce lien :

The screenshot shows a web browser displaying a blog post. The page has a sidebar on the left with the title 'Relooker son Thème (dev)' and a search bar. The main content area shows a post titled 'Bonjour tout le monde !' with a date of '22 juin 2015' and a comment count of 'Un commentaire'. The browser's DevTools inspector is open at the bottom, showing the HTML structure and the CSS styles for the selected element, `h2.entry-title`.

**HTML Structure (Inspector):**

```
<!DOCTYPE html>
<html lang="fr-FR" class="js">
  <head>...</head>
  <body class="home blog logged-in admin-bar customize-support" cz-shortcut-listen="true">
    <div id="page" class="hfeed site">
      <a class="skip-link screen-reader-text" href="#content">Aller au contenu principal</a>
      <div id="sidebar" class="sidebar" style="top: 0px;">...</div>
      <div id="content" class="site-content">
        <div id="primary" class="content-area">
          <main id="main" class="site-main" role="main">
            <article id="post-1" class="post-1 post type-post status-publish format-standard hentry cate...>
              <header class="entry-header">
                <h2 class="entry-title">
                  <a href="http://www.relookersontheme.com/dev/2015/06/22/bonjour-tout-le-monde/" rel="bc...>
                </h2>
              </header>
            </article>
          </main>
        </div>
      </div>
    </div>
  </body>
</html>
```

**CSS Styles (Inspector):**

```
element.style {
}

media="all" www.relookersontheme.com/
@media screen style.css?ver=4.2.2:2087
and (min-width: 60.75em)
.entry-title, style.css?ver=4.2.2:2372
.widecolumn h2 {
  font-size: 35px;
  font-size: 3.5rem;
  line-height: 1.2;
  margin-bottom: 1.2em;
}

media="all" www.relookersontheme.com/
@media screen style.css?ver=4.2.2:1380
and (min-width: 59.6875em)
.entry-title, style.css?ver=4.2.2:1790
.widecolumn h2 {
  font-size: 35px;
}
```

Déjà, nous pouvons voir que la balise h2 possède la classe `.entry-title`. Apparemment, cette classe est appliquée à tous les titres d'articles, cela est nettement plus intéressant pour en personnaliser l'apparence que le sélecteur `a`.

Sur la droite, on constate que plusieurs règles sont définies pour les titres. On peut citer la taille de la police (`font-size`), la hauteur de ligne (`line-height`) et une marge externe inférieure (`margin-bottom`).

Note : La propriété `font-size` barrée signifie qu'une autre propriété est considérée comme plus importante. Effectivement, la règle `font-size: 3.5rem;` est définie après `font-size: 35px;` donc elle prend le dessus.

## Ajouter une règle CSS

En cliquant sous `margin-bottom`, on peut ajouter la propriété de notre choix à la classe `.entry-title`. Dans ce cas précis, la règle `font-variant: small-caps;` a été ajoutée :



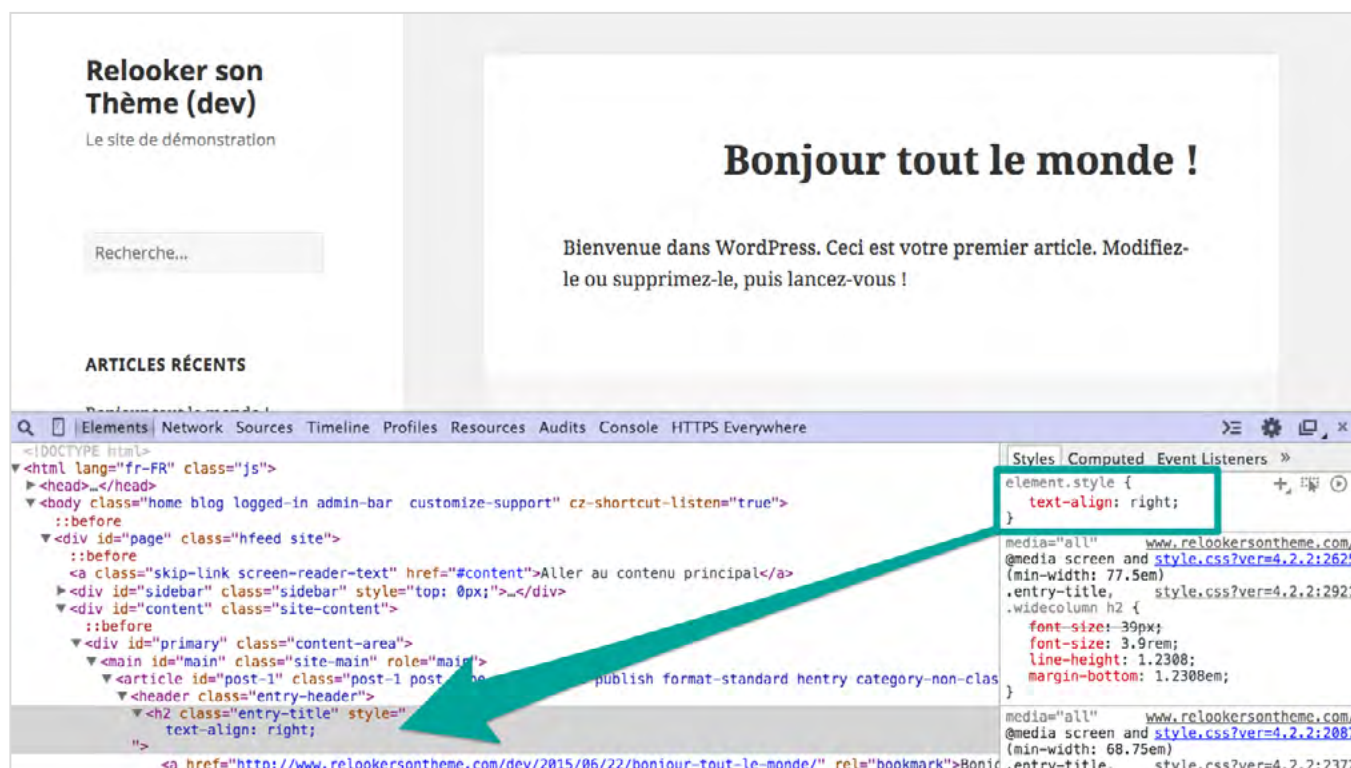
Attention, d'après l'inspecteur ce code ne sera valide que pour une largeur d'écran plus grande que 68,75em, c'est à dire 68,75 fois plus grand que la taille de la police de cet élément (c'est ce qu'indique la media query `@media screen and (min-width: 68.75em)`).

Pour que la propriété `font-variant` soit appliquée pour tous les titres quelque soit la largeur de l'écran, il faudra inclure la propriété suivante dans le fichier `style.css` du thème enfant :

```
.entry-title{
    font-variant: small-caps;
}
```

## Ajouter une règle à un seul élément

Si vous désirez ajouter du code à un élément en particulier (sans vous soucier des classes et identifiants), vous pouvez ajouter une règle au niveau de `element.style` :



Faire cela est équivalent à ajouter un attribut `style` à une balise HTML. On voit d'ailleurs que c'est le cas, car l'inspecteur rajoute effectivement cet attribut.

## Ajouter une règle selon un état

L'inspecteur permet aussi d'ajouter des règles CSS selon l'état d'un élément (active, hover, focus et visited).

Le plus commun étant l'état de survol hover. Une fois qu'un élément est sélectionné, il faut cliquer sur la seconde icône en haut à droite (juste après `element.style`) puis cliquer sur l'état de son choix pour le simuler.



Étant donné qu'aucune règle CSS n'est appliquée par le thème Twenty Fifteen lors du survol de la balise h2, j'en ajoute une nouvelle en cliquant sur la petite croix (la première icône) pour créer un sélecteur auquel il faut rajouter `: hover`.

La propriété `font-style` est ensuite ajoutée et sa valeur définie à `italic` pour que le texte soit mis en italique lors du survol d'une balise `h2` dotée d'une classe `.entry-title`.

En décochant `:hover` et en survolant le titre, on se rendrait compte qu'il se mettrait automatiquement en italique.

## Désactiver une règle CSS

L'inspecteur permet également de désactiver des règles CSS pour voir comment une page s'affiche sans elles.

Pour ce faire, il suffit de survoler une règle et de décocher la case qui apparaît sur la gauche.

Dans l'exemple ci-dessous, la couleur des liens a été désactivée :



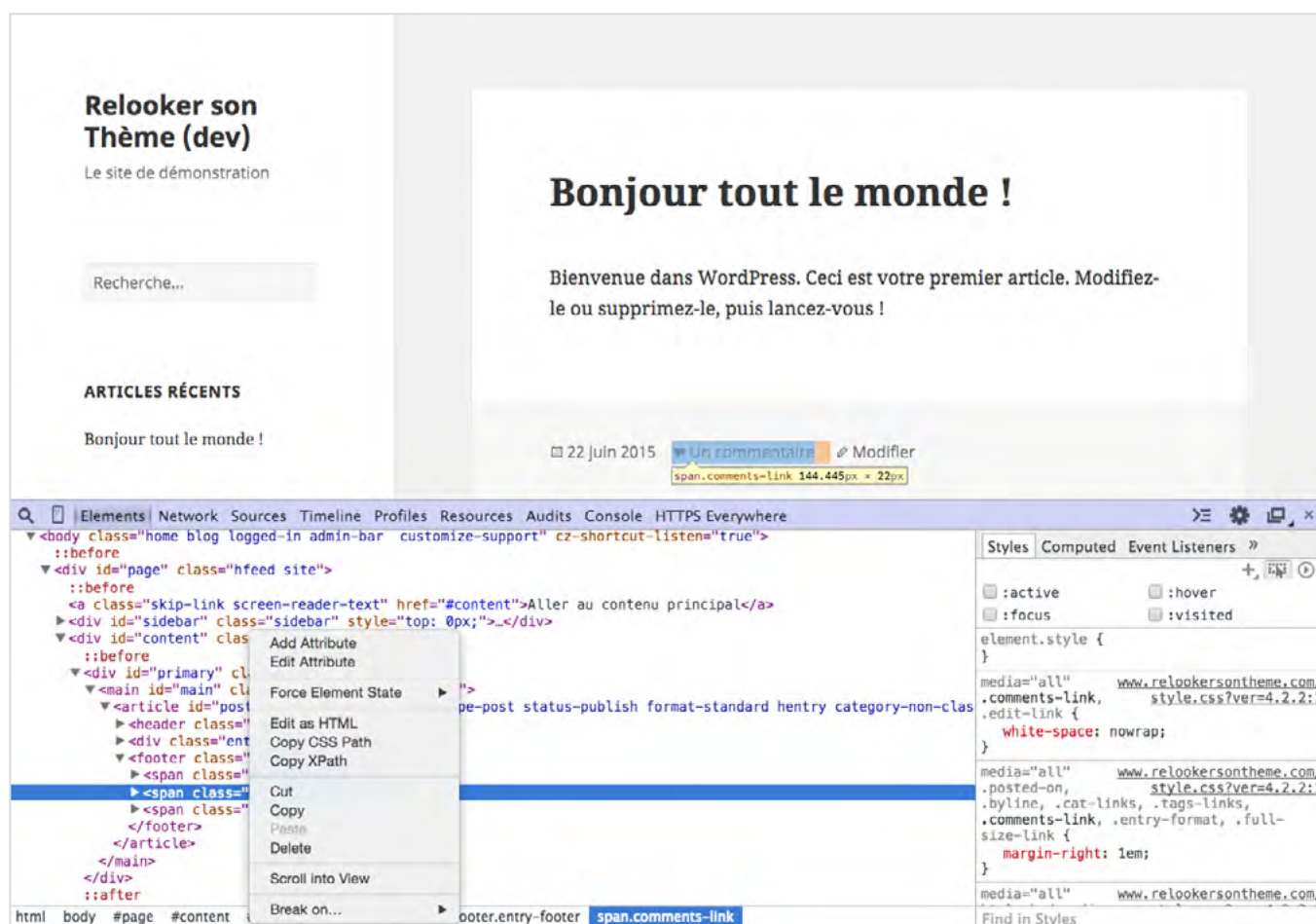
On se retrouve donc avec la couleur par défaut, c'est à dire le bleu compris dans les styles du navigateur.

*N'oubliez pas que toutes ces modifications ne seront pas enregistrées. Elles servent uniquement à tester des règles CSS. Il faut les répercuter dans le fichier `style.css` du thème enfant.*

## Personnaliser le code HTML via l'inspecteur

Voyons à présent comment modifier la structure HTML d'une page grâce à l'inspecteur.

Tout commence à partir d'un clic droit sur un élément. Le menu suivant apparaît :



Les principales actions à retenir sont :

- **Add Attribute** : Ajouter un nouvel attribut (identifiant, classe ou autre)
- **Edit Attribute** : Modifier l'attribut sur lequel le clic droit a été effectué
- **Force Element State** : Forcer l'affichage d'un état (active, hover, focus ou visited)
- **Edit as HTML** : Modifier le code HTML de l'élément
- **Copy CSS Path** : Copier le chemin CSS (idéal pour avoir un sélecteur précis)
- **Cut / Copy** : Couper / Copier un élément
- **Delete** : Supprimer un élément
- **Scroll into View** : Afficher l'élément à l'écran (cela sera sans effet si l'élément est déjà à l'écran)

En modifiant le code HTML, vous allez pouvoir tester de nouvelles choses avant de les enregistrer dans votre éditeur de code.

Par exemple, si vous désirez voir ce qu'une classe peut avoir comme impact sur un élément, il suffit d'utiliser *Add Attribute* ou *Edit Attribute* pour l'ajouter.

Vous pouvez également inclure du contenu factice grâce à *Edit as HTML* ou encore visualiser votre site sans un élément grâce à *Delete*.

Encore une fois, cela n'est que temporaire. Si les modifications vous plaisent, il faudra les répercuter dans votre thème enfant pour les conserver.

## Récapitulons

L'inspecteur de code est un outil indispensable à toute personne désirant gagner du temps. Grâce à lui vous deviendrez littéralement un alchimiste des pages web.

Vous pourrez étudier et personnaliser les règles CSS de votre site en toute tranquillité.

Si quelque chose ne vous convient pas, vous n'aurez qu'à actualiser pour retrouver la page d'origine.

Une fois que vous aurez trouvé la configuration HTML/CSS dont vous aurez besoin, il faudra la sauvegarder grâce à l'éditeur de code. Nous allons voir comment procéder en détail dans la partie suivante.

## Chapitre 5.4

# Enregistrer ses personnalisations dans l'éditeur de code

Au début de ce guide, vous avez été invité à installer un éditeur de code afin de pouvoir faire les exercices associés aux précédents chapitres.

À présent, nous allons voir comment utiliser l'éditeur de code pour relonger un thème WordPress.

Ce que vous vous apprêtez à lire est en fait une étude de cas partageant le déroulement de la personnalisation d'un thème.

Note : Quelques différences peuvent exister si vous utilisez un autre éditeur de code mais les principes restent les mêmes.

Pour cette étude de cas, nous allons relonger le thème Twenty Fifteen en ajoutant une image au dessus du nom du site. Voilà ce que nous allons chercher à obtenir :



Bien sûr, cela devra fonctionner sur toutes les tailles d'écrans.

Prêt ? Alors...

## Revenons aux fondamentaux

Pour placer cette image au dessus du titre, il faut déjà créer un thème enfant.

Nous n'allons pas revenir sur la nécessité d'un thème enfant car nous en avons amplement parlé dans la seconde partie de ce chapitre.

Par contre, nous allons voir comment créer un thème enfant sur votre site de test. Après tout, nous sommes dans le chapitre du passage à la pratique.

*Si vous n'avez pas créé votre site de test, vous prenez le risque de perturber la navigation sur votre site.*

## Utiliser le bon logiciel pour se connecter à son serveur

Pour installer WordPress, vous avez certainement utilisé un logiciel FTP pour vous connecter à votre serveur.

Un des plus connus est FileZilla mais après en avoir testé plusieurs, je vous recommande plutôt d'utiliser **CyberDuck**.

Le principal avantage de CyberDuck par rapport à FileZilla est qu'il renvoie automatiquement un fichier sur le serveur dès qu'il est sauvegardé.

L'éditeur de code Brackets permet d'installer des extensions pour se connecter à un serveur mais après plusieurs tests, il s'est avéré qu'elles ne fonctionnent pas encore de manière optimale.

*Le couple Brackets/CyberDuck est la meilleure solution gratuite, en français et disponible sur Windows et Mac que j'ai pu trouver au moment de la rédaction de ce guide.*

Si vous êtes sur Mac et que vous désirez vous connecter à votre serveur via un éditeur de code, je vous recommande grandement **Coda** (il faudra néanmoins déboursier 99\$).

Logiquement, Brackets doit déjà être installé sur votre ordinateur (si vous avez fait les exercices des chapitres 2 et 3).

Note : Si vous n'avez pas encore installé Cyberduck, vous trouverez un tutoriel dans les ressources liées au guide (fiche n°6).

Une fois Cyberduck installé et configuré, connectez-vous à votre serveur pour y visualiser la liste des fichiers de votre site.

À présent, nous allons pouvoir...

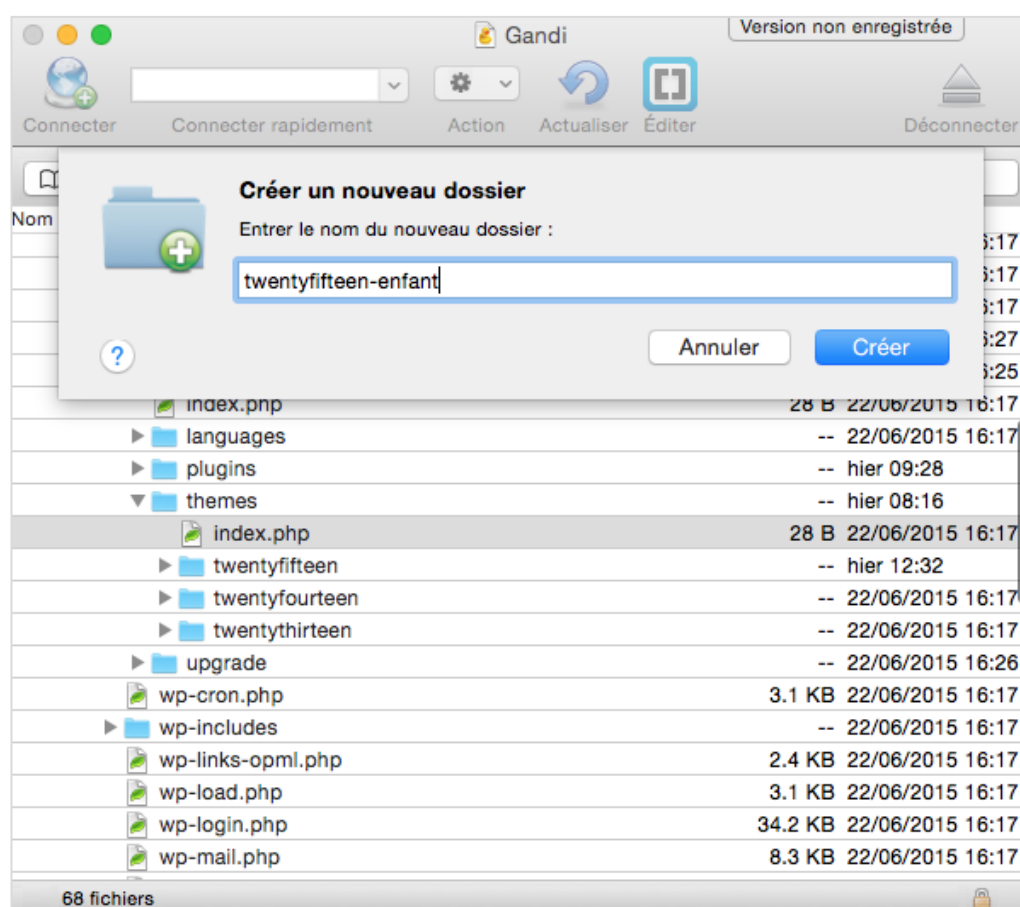
## Créer le thème enfant

Lorsque vous avez créé la copie de votre site avec l'extension Duplicator, vous l'avez fait dans un nouveau dossier.

Pour créer le thème enfant du thème Twenty Fifteen, il faut se rendre dans ce dossier puis aller dans `/wp-content/themes/`.

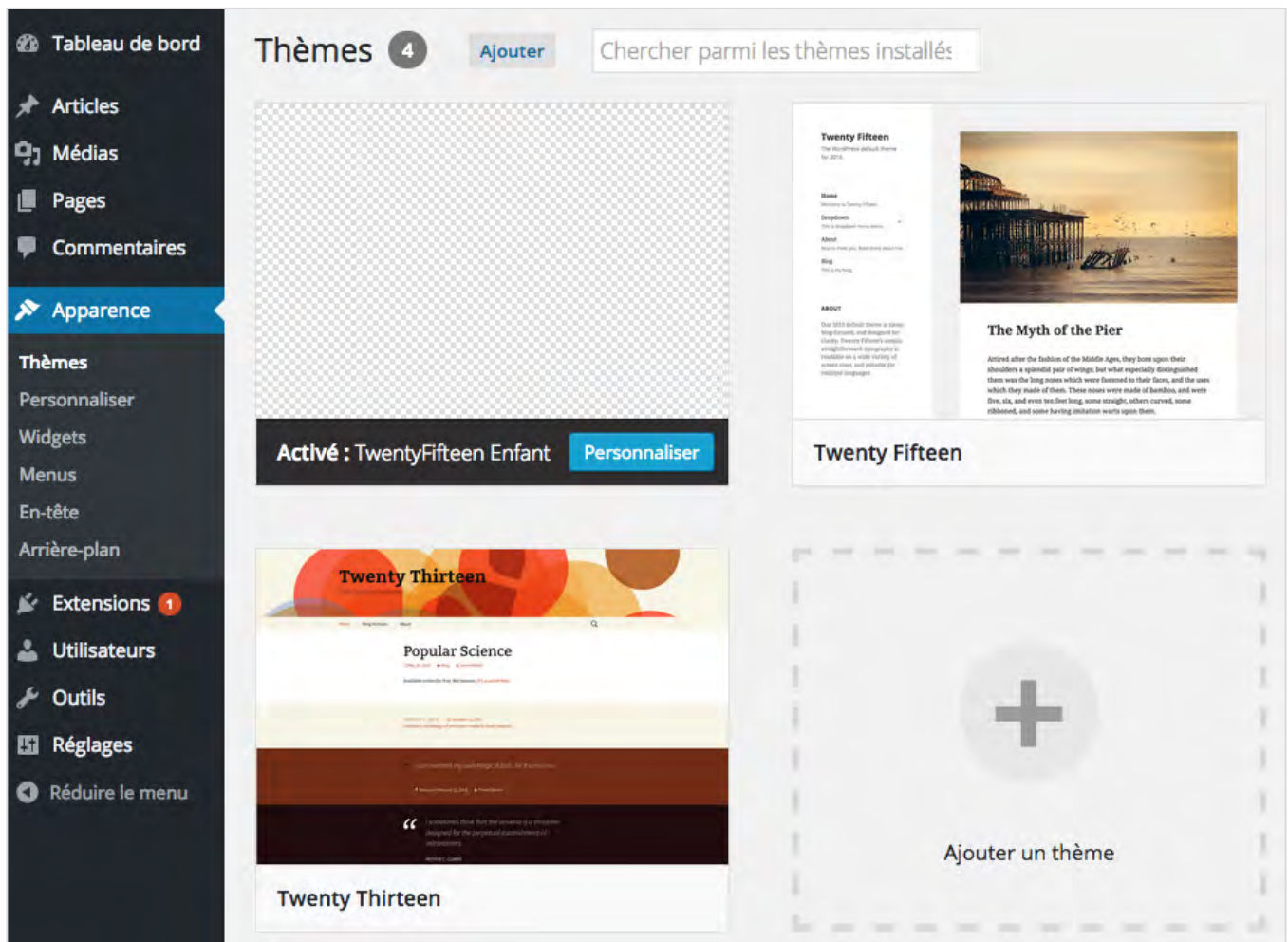
La liste des thèmes WordPress présents sur le site s'affiche, il ne restera plus qu'à créer un nouveau dossier pour y placer les fichiers du thème enfant.

Après un clic droit sur le fichier `index.php` et la sélection de *Nouveau dossier*, il faut entrer le nom du dossier du thème enfant :



Concernant le contenu de base d'un thème enfant, je vous laisse vous reporter à la partie 2 de ce chapitre.

Quand votre thème enfant sera créé sur votre serveur, activez-le en vous rendant dans *Apparence > Thèmes* :



Note : Pour afficher une image d'illustration à un thème, il suffit d'inclure un fichier nommé `screenshot.png` de 880x660 pixels à la racine du thème. Cela est bien entendu facultatif.

Une fois le thème enfant activé, il ne reste plus qu'à...

## Trouver le fichier à modifier

Mettre la main sur le bon fichier est le premier défi auquel doit faire face tout relooker de thème.

Plusieurs cas de figure se présentent en fonction de la personnalisation que l'on désire effectuer.

Dans le cas de l'insertion d'une image au dessus du titre d'un site basé sur Twenty Fifteen, c'est assez simple.

En effet, nous avons appris dans le chapitre relatif à WordPress que cela se situait dans le fichier `header.php`.

*Pour chaque modification, examinez d'abord les fichiers de template d'un thème WordPress (hiérarchie des templates et fichiers de structure).*

Dans d'autres cas, cela peut s'avérer plus complexe :

- Le contenu a pu être ajouté par le biais d'un hook, auquel cas il faudra désactiver ce hook avec la fonction `remove_action()` et associer une autre fonction à ce hook dans le fichier `functions.php` du thème enfant
- Si vous désirez modifier des fichiers PHP n'étant pas des fichiers de template, il faudra redéfinir ces fichiers (et certainement d'autres) dans le thème enfant tout en faisant en sorte qu'ils soient chargés à partir du thème enfant.

Heureusement, **la plupart des modifications simples pourront se faire grâce aux fichiers de template.**

Une fois le fichier repéré, il faut...

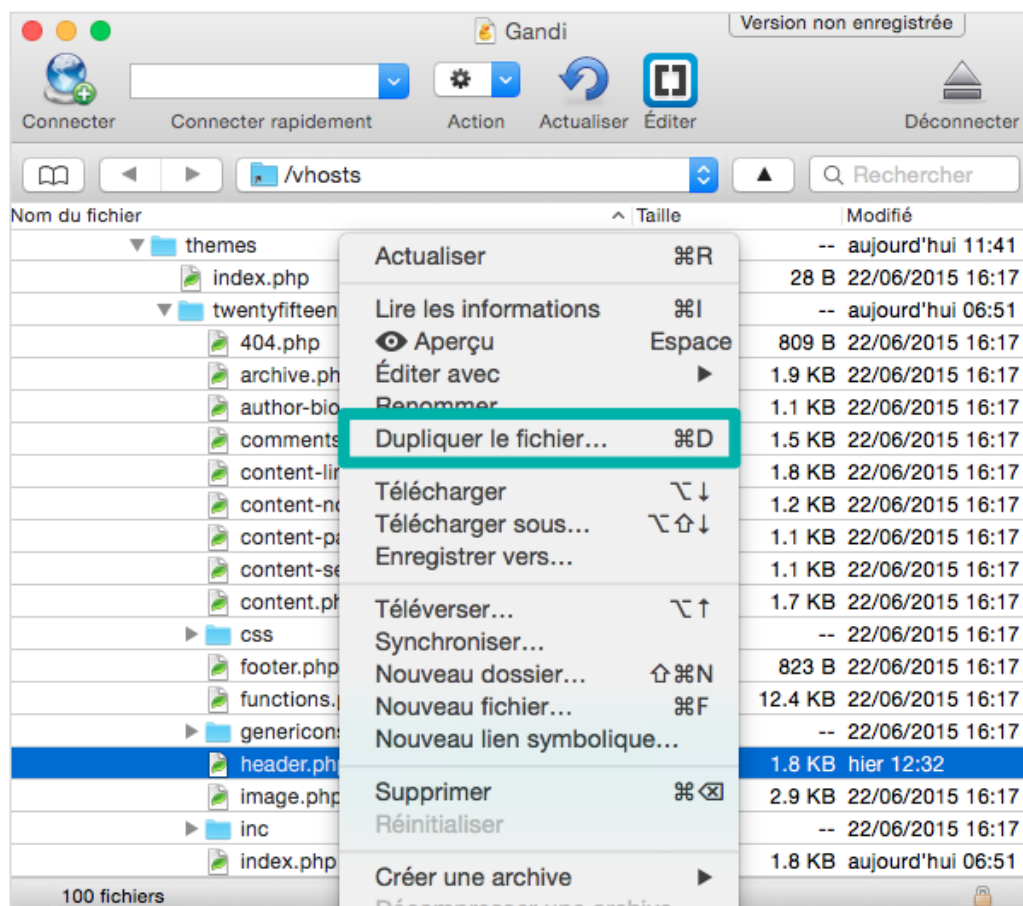
## Effectuer les modifications

Comme nous l'avons vu, le fichier de template que l'on veut personnaliser doit se trouver dans le thème enfant.

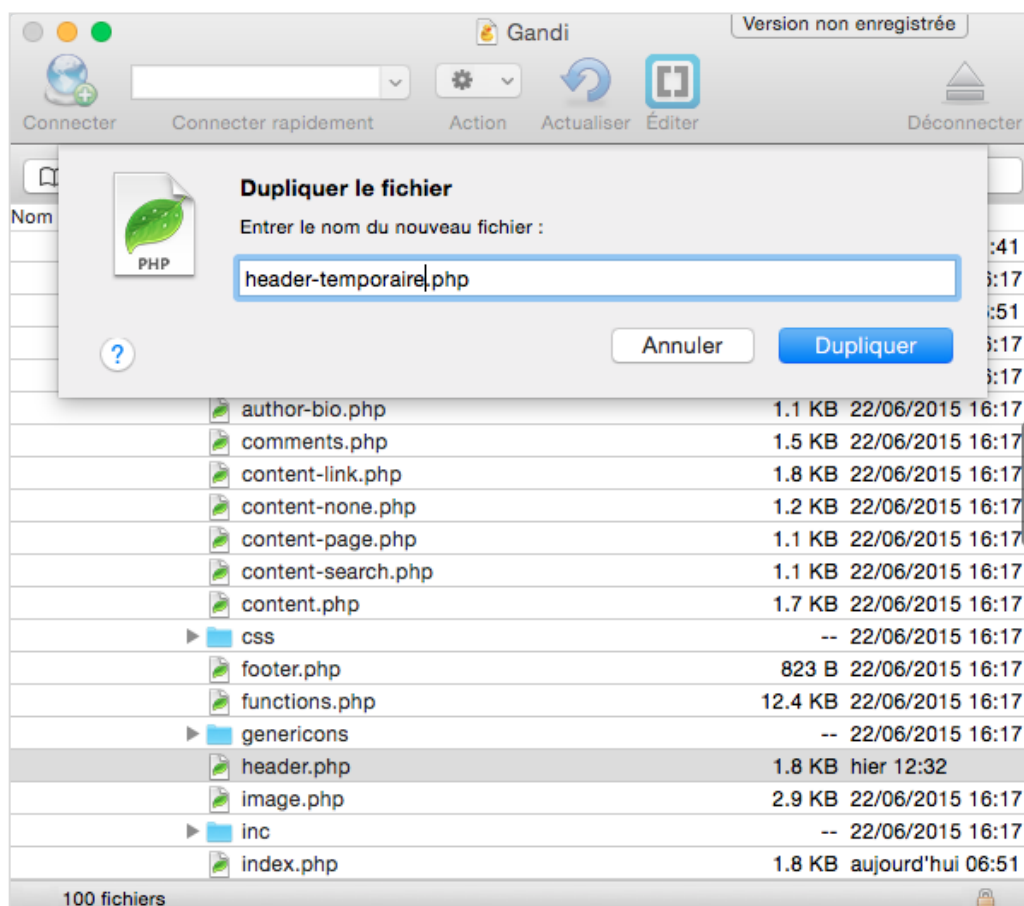
Il serait possible de créer un fichier `header.php` dans le thème enfant puis d'y copier le contenu de celui du thème parent mais Cyberduck permet de dupliquer des fichiers facilement.

## Dupliquer un fichier dans le thème enfant

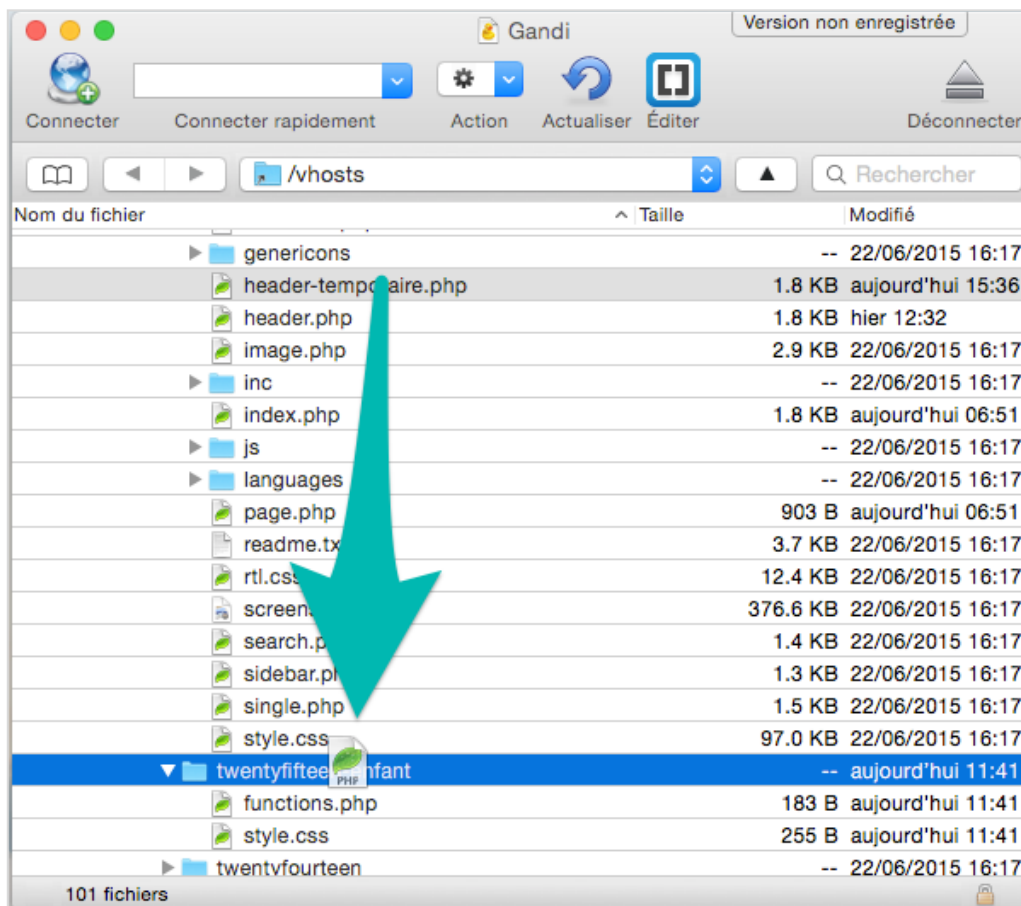
Pour ce faire, il suffit de faire un clic droit sur le fichier en question puis de cliquer sur *Dupliquer le fichier* :



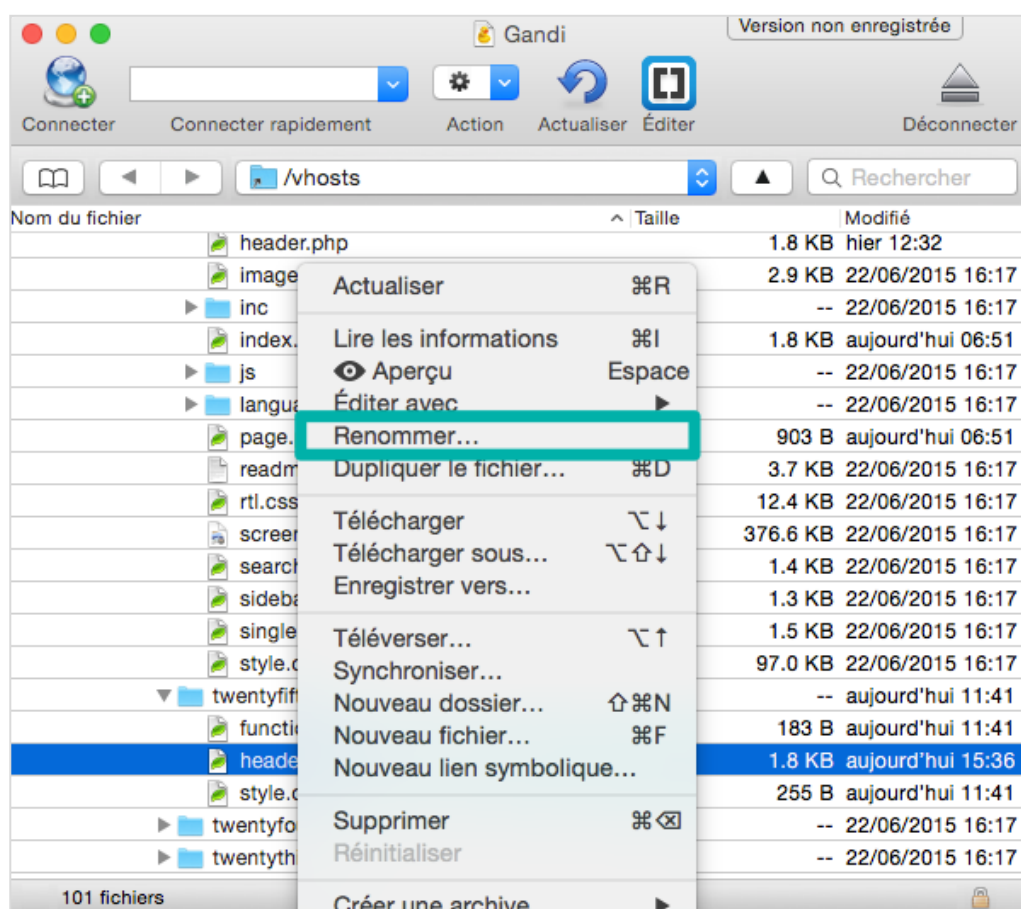
Il sera demandé de donner un nouveau nom au fichier. En effet, deux fichiers ne peuvent pas avoir le même nom dans un dossier. Il est nécessaire d'indiquer un nom de fichier temporaire :



Une fois le fichier dupliqué, l'étape suivante est de le déplacer dans le thème enfant par un glisser déposer :



Lorsque le fichier sera arrivé à destination, il ne reste plus qu'à lui redonner son nom d'origine, dans notre cas `header.php`. Faites simplement un clic droit sur le fichier et sélectionnez *Renommer* :



À présent, il ne reste plus qu'à effectuer les modifications pour obtenir le résultat désiré.

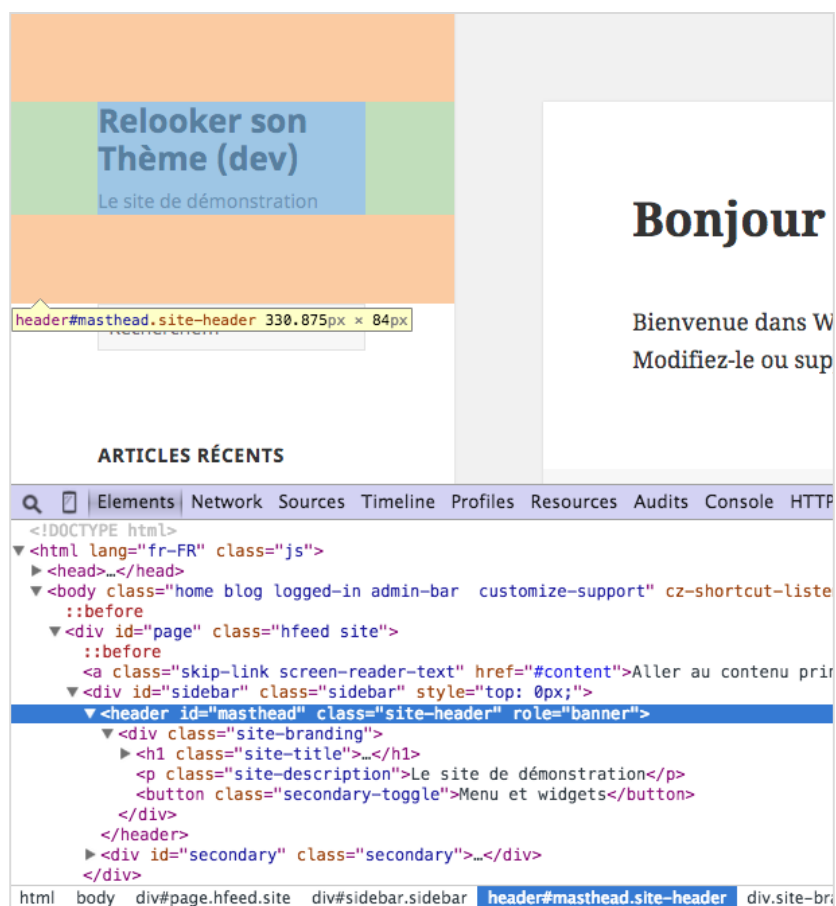
## Éditer un fichier avec Brackets

Si vous avez bien suivi le tutoriel d'installation et de configuration de Cyberduck, vous vous rappelez **qu'il n'y a qu'à double cliquer sur le fichier à modifier pour lancer Brackets**.

Quand le fichier à personnaliser est ouvert, il faut trouver l'endroit où effectuer ses modifications.

Pour éviter de chercher à l'aveuglette, il est préférable d'utiliser l'inspecteur de code sur le site pour repérer les balises HTML proches de l'endroit où l'on veut ajouter ou supprimer du code.

Avec notre exemple d'image à insérer au dessus du titre du site basé sur Twenty Fifteen, voici ce que l'inspecteur nous montre :



On peut voir que :

- La balise header contient toutes les informations de l'en-tête
- La balise div dotée de la classe .site-branding contient le titre h1, le paragraphe du slogan et une balise bouton (pour le menu responsive)

Afin de placer l'image avant le titre, voici ce qu'il est possible de faire :

```
19 <script src="<?php echo esc_url( get_template_directory_uri() ); ?>/js/html5.js">
20 </script>
21 <![endif]-->
22 <?php wp_head(); ?>
23 </head>
24 <body <?php body_class(); ?>>
25 <div id="page" class="hfeed site">
26 <a class="skip-link screen-reader-text" href="#content"><?php _e( 'Skip to
content', 'twentyfifteen' ); ?></a>
27
28 <div id="sidebar" class="sidebar">
29 <header id="masthead" class="site-header" role="banner">
30 <div class="site-branding">
31
32 |
33
34 <?php
35 if ( is_front_page() && is_home() ) : ?>
36 <h1 class="site-title"><a href="<?php echo esc_url( home_url(
37 '/' ) ); ?>" rel="home"><?php bloginfo( 'name' ); ?></a></h1>
38 <?php else : ?>
39 <p class="site-title"><a href="<?php echo esc_url( home_url(
40 '/' ) ); ?>" rel="home"><?php bloginfo( 'name' ); ?></a></p>
41 <?php endif;
42
43 $description = get_bloginfo( 'description', 'display' );
44 if ( $description || is_customize_preview() ) : ?>
45 <p class="site-description"><?php echo $description; ?></p>
46 <?php endif;
47
48 <button class="secondary-toggle"><?php _e( 'Menu and widgets',
'twentyfifteen' ); ?></button>
49 </div><!-- .site-branding -->
50 </header><!-- .site-header -->
51
52 <?php get_sidebar(); ?>
53 </div><!-- .sidebar -->
54
55 <div id="content" class="site-content">
```

Notre balise `img` se trouve bien à l'endroit souhaité. Par contre, l'adresse de l'image n'a pas encore été spécifiée.

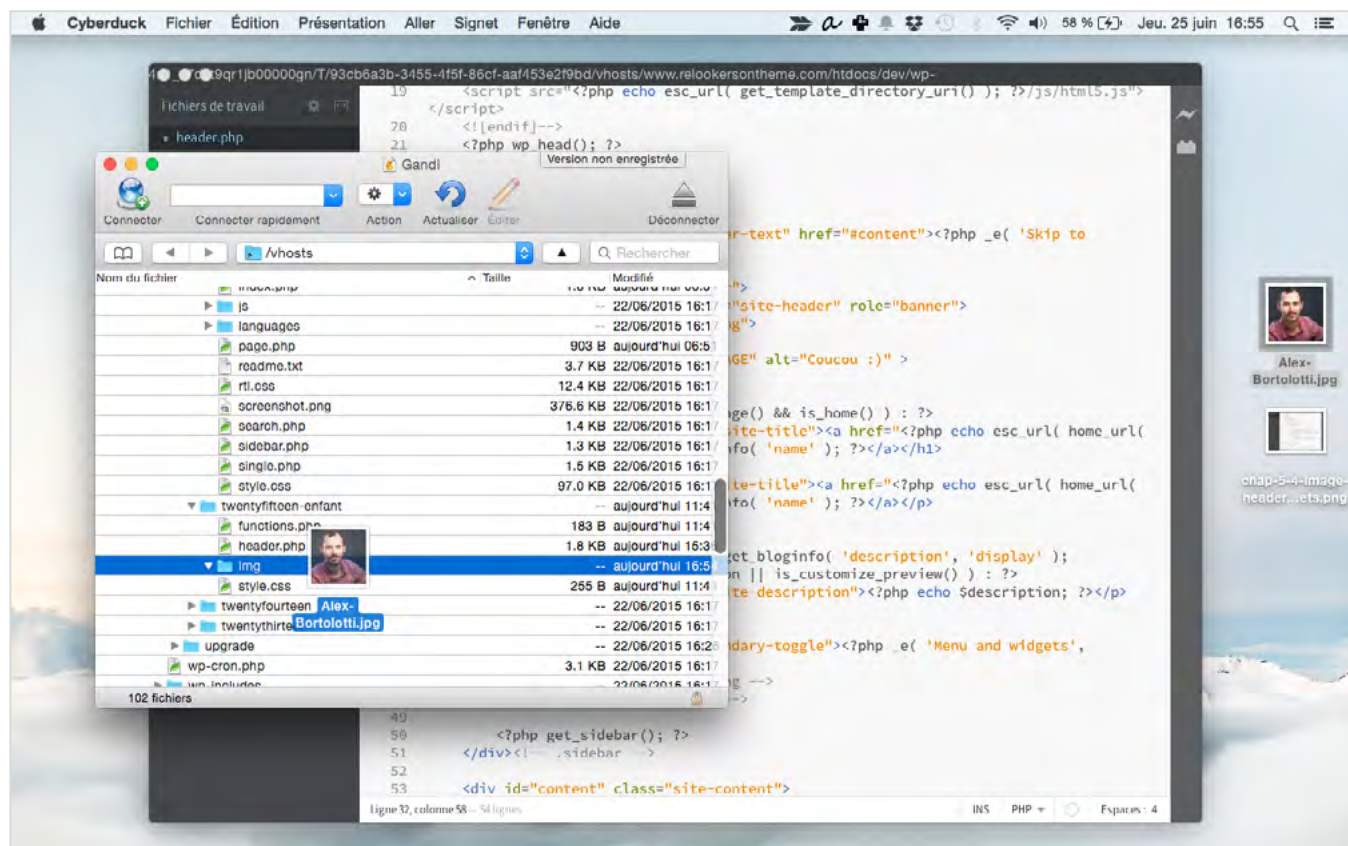
Ici nous avons deux possibilités :

1. Soit on envoie l'image dont on a besoin dans la bibliothèque de médias de WordPress puis on colle l'adresse de l'image dans la balise `img`
2. Soit on intègre l'image directement dans le thème car elle ne sera pas utilisée dans des publications

Nous allons partir sur la seconde solution.

La bonne pratique consiste à créer un dossier `img` ou `images` dans le thème enfant et d'y envoyer les images liées au thème.

La création de dossier a été abordée précédemment donc je ne reviendrai pas sur ce point. Il ne reste donc qu'à glisser déposer l'image dans ce dossier :



Il faut maintenant intégrer l'adresse de l'image dans la balise `img` créée auparavant.

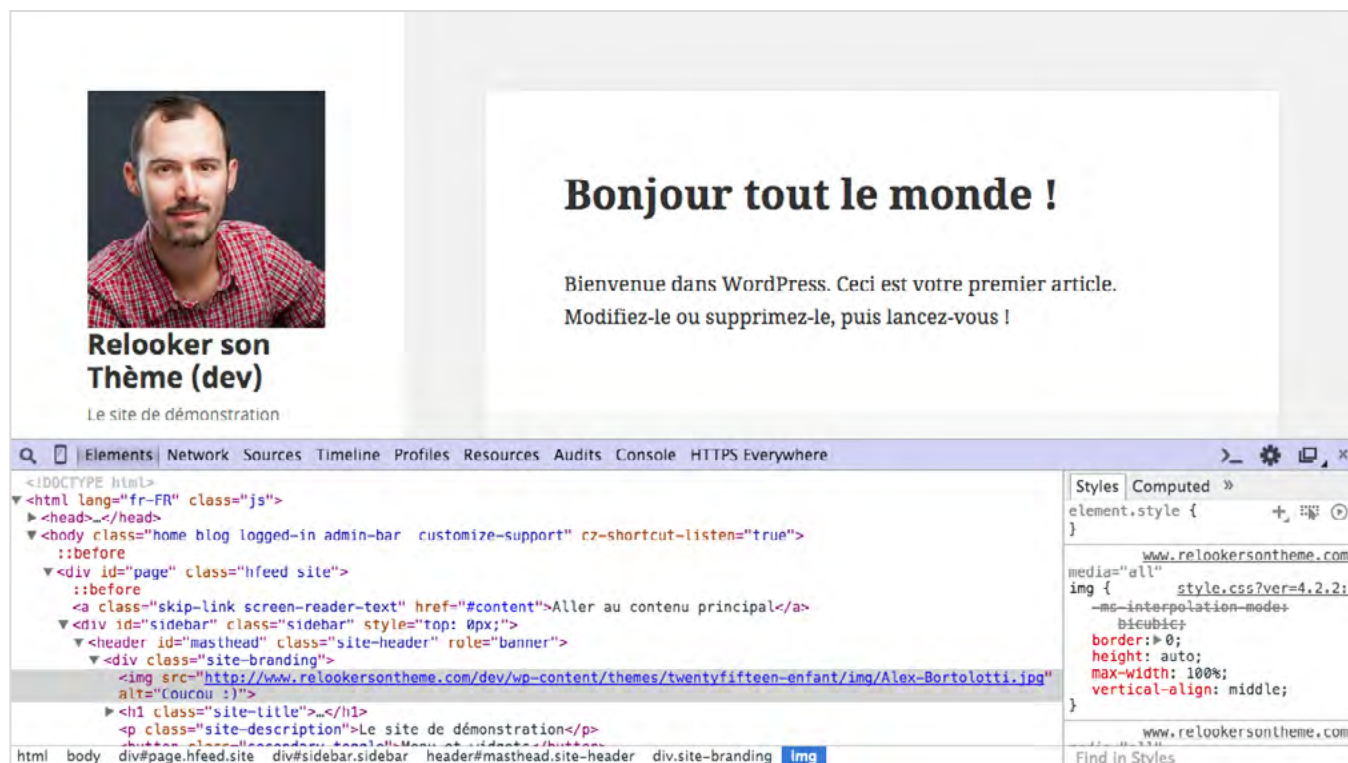
Pour ce faire, nous pouvons utiliser la fonction de WordPress qui permet d'afficher l'adresse du thème courant. Ce qui nous donne le code suivant :

```

```

Il ne reste plus qu'à sauvegarder le fichier en pressant **CTRL + S** (ou **CMD + S** sur Mac). Aussitôt, Cyberduck va détecter les modifications et les répercuter sur le serveur.

La modification est effectuée. En actualisant la page voici ce que l'on obtient :



Bien sûr, ce n'est qu'un début. On pourrait améliorer cela en :

- ajoutant un lien vers la page d'accueil sur l'image
- insérant une marge inférieure pour séparer l'image du titre
- optimisant l'affichage pour le responsive

Une fois qu'un élément est inséré, il ne reste plus qu'à le styliser en CSS. Pour cela, il faut évidemment modifier le fichier `style.css` du thème enfant.

*Après quelques aller-retours entre Brackets et le navigateur ainsi que des tests via l'inspecteur de code, on arrive généralement à ses fins.*

Des recherches de fonctions WordPress sur Google et un retour dans les chapitres précédents de Relooker son Thème sont également possibles pour obtenir les résultats désirés.

## Récapitulons

Dans cette partie, nous avons étudié en détail comment effectuer des modifications au sein d'un thème enfant grâce au tandem Brackets/Cyberduck.

Avec le temps, vous verrez que les manipulations présentées deviendront de plus en plus banales. C'est juste une question d'entraînement.

Veillez cependant à vérifier si les modifications que vous désirez effectuer ne sont pas possibles via les options du thème, il serait dommage de perdre du temps pour rien.

## Chapitre 5.5

# Gagnez en productivité avec ces 9 outils de développement

Dans les deux précédentes parties, nous avons vu comment personnaliser un thème WordPress grâce à l'inspecteur de code, l'éditeur Brackets et le client FTP Cyberduck.

Si vous comptez travailler avec WordPress régulièrement, il est possible d'aller un peu plus vite grâce à d'autres outils de développement.

Ils ne sont pas indispensables mais il ne faut tout de même pas les ignorer.

On peut classer ces outils en trois catégories, les extensions WordPress, les extensions Brackets ainsi que les extensions de navigateur (Chrome, Firefox, etc).

Pour chaque catégorie, vous allez découvrir trois outils de développement. Bien sûr, il en existe bien d'autres. J'ai préféré vous présenter ceux qui vous seront les plus utiles.

Commençons par les...

## Extensions de développement WordPress

Vous savez certainement que WordPress ne serait pas WordPress sans ses extensions. En effet, elles permettent de rajouter des dizaines de fonctionnalités à un site.

Il existe aussi des extensions WordPress dont le but est de simplifier la vie des développeurs. En voici trois qui sont particulièrement utiles :

## What The File

Vous rappelez-vous ce que nous avons fait pour ajouter une image au dessus du titre d'un site équipé de Twenty Fifteen dans la partie précédente ?

Nous avons dû *deviner* dans quel fichier il fallait faire la modification.

Il faut avouer que la devinette était assez simple car l'image devait se placer dans l'en-tête...

*What The File* est une extension WordPress qui vous évitera de passer des heures à chercher dans quel fichier faire vos modifications.

Une fois installée, l'extension *What The File* ajoute un menu en haut à droite de la barre d'outils de WordPress :



En le survolant, on peut tout de suite voir les fichiers de template utilisée par la page sur laquelle on se trouve.

Même si cette extension pourrait être améliorée pour prendre en compte certaines exceptions, elle donne un bon coup de pouce dans la majorité des cas.

**En savoir plus sur What The File**

## Simply Show IDs

Au cours du chapitre 4, nous avons vu que des IDs, c'est à dire des identifiants, étaient utilisés par WordPress.

Note : Les identifiants dont nous parlons n'ont rien à voir avec les identifiants que l'on retrouve en HTML et en CSS.

Ces identifiants servent à répertorier les publications, les catégories, les étiquettes et aussi les médias, les commentaires et les utilisateurs.

Par exemple, si l'on veut créer une page de catégorie spécifique pour une catégorie donnée, la hiérarchie des templates nous dit que l'on peut créer un fichier `category-[ID].php`.

On peut aussi se servir des identifiants dans les marqueurs conditionnels. Par exemple, pour savoir si le visiteur se trouve sur la page d'identifiant 4 on peut écrire le code suivant :

```
<?php
if(is_page(4)){
    // Code à exécuter
} ?>
```

Bref, si l'on veut cibler quelque chose de précis dans un relooking, on aura certainement besoin d'un identifiant.

Par contre, la question à se poser est : comment trouver ce fichu identifiant ?

Il est possible de le trouver sans extension mais cela est peu pratique.

Installer l'extension *Simply Show IDs* permet de les découvrir beaucoup plus rapidement.

Suite à son activation, cette extension ajoutera une colonne au niveau des listes d'articles, de pages, de catégories, etc pour indiquer chacun des identifiants. Voyez plutôt :

Articles [Ajouter](#)

Options de l'écran

Aide

Tous (3) | Publiés (3)

Chercher dans les articles

Actions groupées

Appliquer

Toutes les dates

Toutes les catégories

Filtrer

3 éléments

<input type="checkbox"/> Titre	Auteur	Catégories	Étiquettes		Date	ID
<input type="checkbox"/> Merci d'être arrivé jusqu'ici, vous êtes génial !	Alex	Non classé	—	0	Il y a 1 minute Publié	6
<input type="checkbox"/> L'entraînement est la clé de la réussite	Alex	Non classé	—	0	Il y a 1 minute Publié	4
<input type="checkbox"/> Bonjour tout le monde !	Alex	Non classé	—	1	22/06/2015 Publié	1
<input type="checkbox"/> Titre	Auteur	Catégories	Étiquettes		Date	ID

Actions groupées

Appliquer

3 éléments

[En savoir plus sur Simply Show IDs](#)

## Regenerate Thumbnails

Cette dernière extension n'est pas dédiée au développement à proprement dit mais elle vous sauvera la vie lorsque vous mettrez en ligne un nouveau thème ou un thème relooké.

Lorsqu'un nouveau thème est activé, les images à la une qu'il utilise ne sont presque jamais aux mêmes dimensions que le thème précédent.

Dès lors, des problèmes d'affichage peuvent survenir.

Des soucis peuvent également se produire lorsque l'on ajoute de nouvelles tailles d'images (nous verrons comment faire dans le chapitre suivant). Les images précédemment envoyées ne peuvent pas exister dans la nouvelle taille.

L'extension *Regenerate Thumbnails* permet de régénérer les miniatures de toutes les images envoyées sur un site (et cela dans toutes les tailles spécifiées par le thème).

Après l'avoir installée et activée, tout ce que vous avez à faire est de vous rendre dans *Outils > Regen. Thumbnails* et cliquer sur le bouton *Regenerate All Thumbnails*.

### Regenerate Thumbnails

Please be patient while the thumbnails are regenerated. This can take a while if your server is slow (inexpensive hosting) or if you have many images. Do not navigate away from this page until this script is done or the thumbnails will not be resized. You will be notified via this page when the regenerating is completed.

66.7%

Abort Resizing Images

#### Debugging Information

Total Images: 3  
Images Resized: 2  
Resize Failures: 0

1. "roca-cristaux" (ID 34) was successfully resized in 1,899 seconds.
2. "youth-570881\_1280" (ID 16) was successfully resized in 2,063 seconds.

Il est également possible de régénérer des images individuellement en passant par la bibliothèque de médias.

**En savoir plus sur Regenerate Thumbnails**

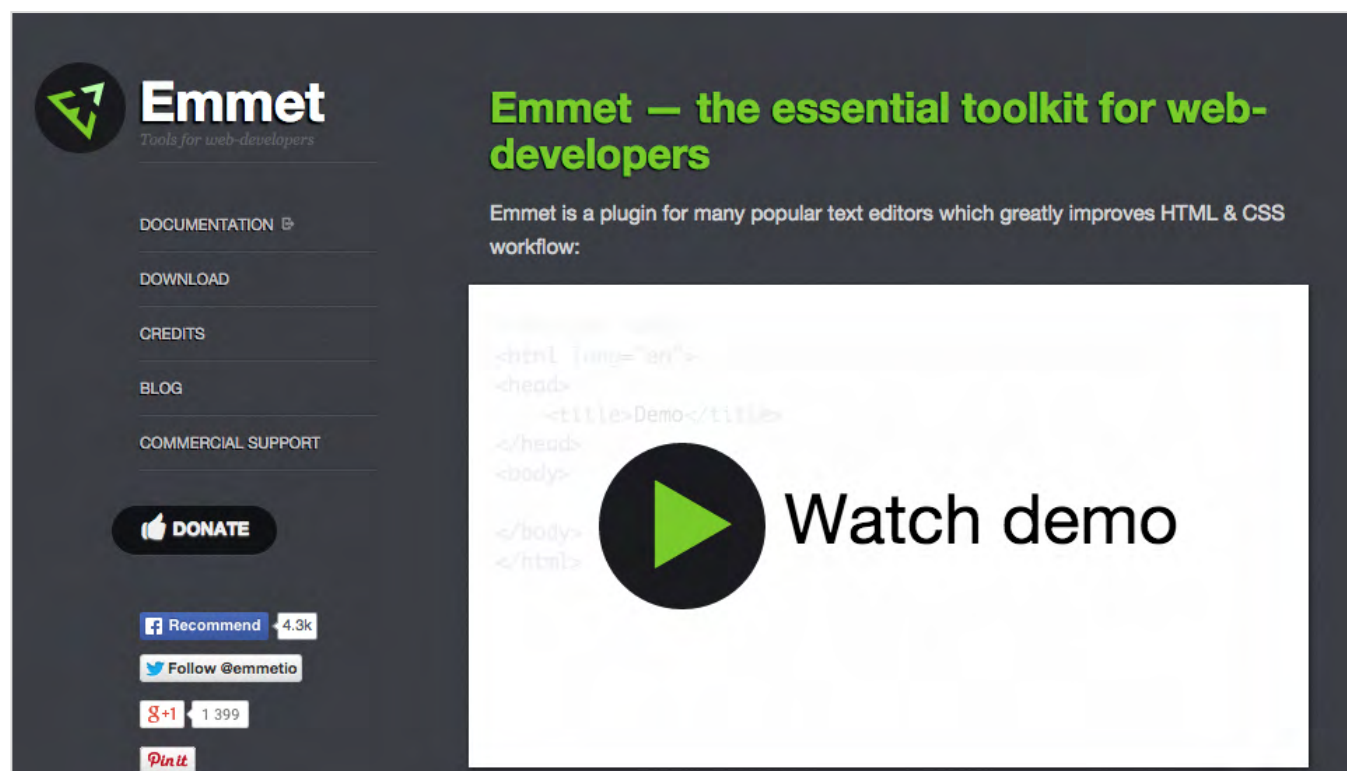
Passons maintenant aux...

## Extensions Brackets

Comme pour un bon nombre d'éditeurs de code, il est possible d'installer des extensions à Brackets pour écrire du code plus aisément.

La première extension à installer se nomme...

### Emmet



Derrière ce nom étrange se cache une extension exceptionnelle. Que diriez-vous si vous pouviez écrire du code HTML sans vous soucier des balises fermantes ?

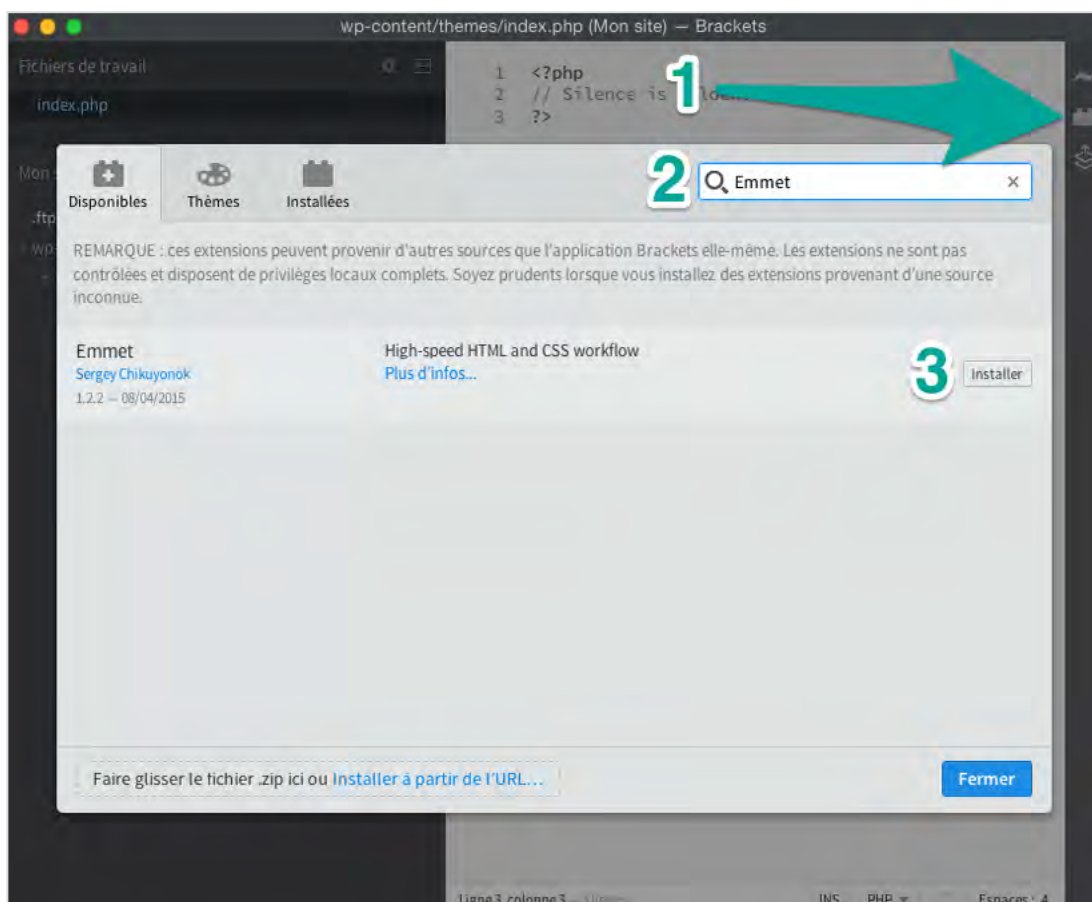
Avec Emmet, taper ce genre de code : `ul#maliste>li*4>a` puis presser la touche *Tab* (à gauche de la touche *A*) permettra d'écrire ceci :

```
<ul id="maliste">
  <li><a href=""></a></li>
  <li><a href=""></a></li>
  <li><a href=""></a></li>
  <li><a href=""></a></li>
</ul>
```

Plutôt sympa n'est-ce pas ?

En fait cela reprend presque l'écriture des sélecteurs CSS. Basez-vous sur l'exemple ci-dessus pour décupler votre créativité.

Pour installer Emmet dans Brackets, ouvrez le gestionnaire d'extensions, recherchez *Emmet* dans le champ de recherche et procédez à l'installation :



Note : Emmet est destiné aux personnes ayant de bonnes connaissances en HTML et CSS. Cette extension ne remplace pas l'apprentissage, elle sert juste à aller plus vite.

## En savoir plus sur Emmet

### Indent Guides

Cette petite extension ne paye pas de mine mais elle vous empêchera de vous perdre dans votre code.

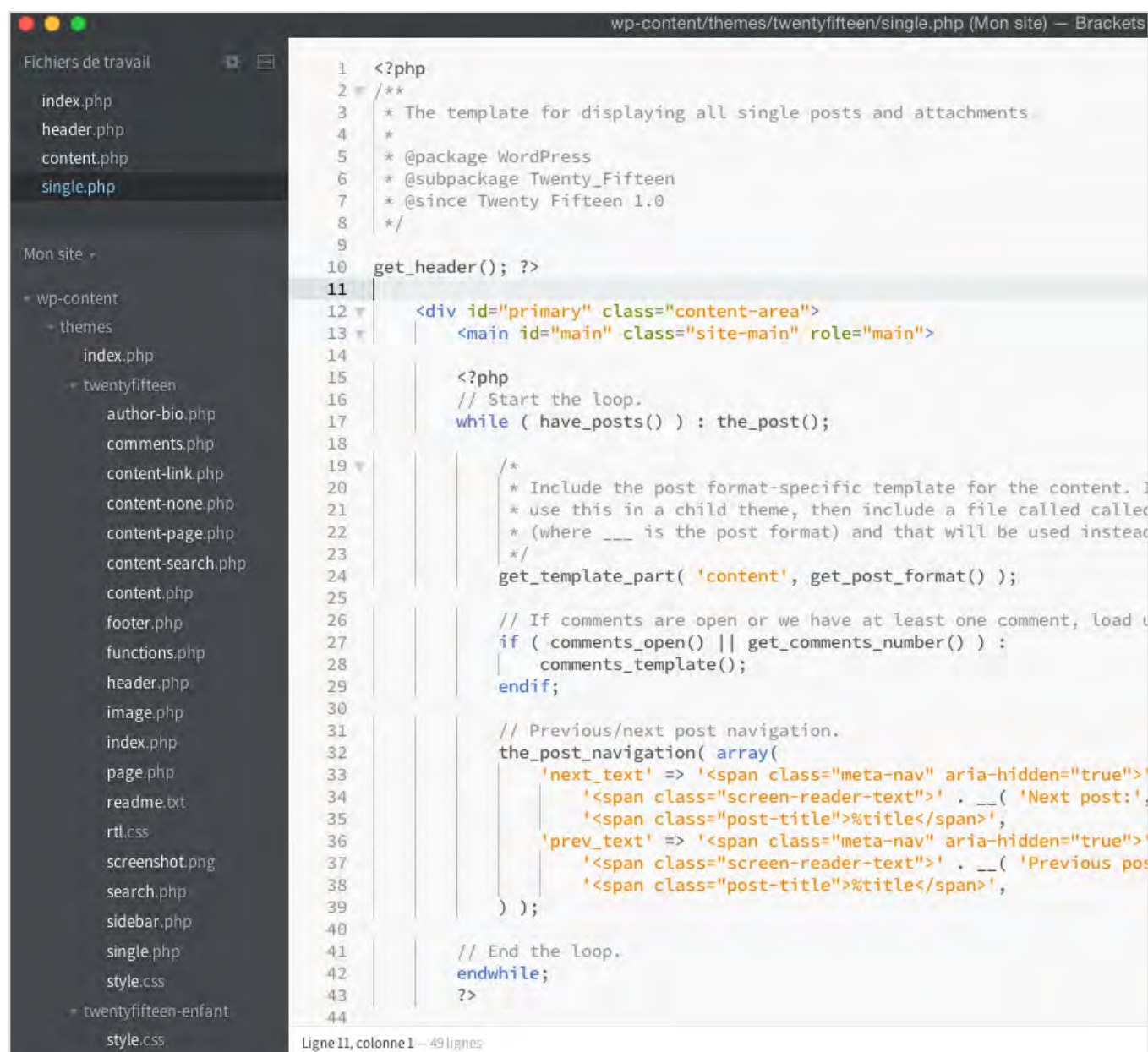
Nous avons étudié ce qu'était l'indentation, à savoir la mise en page du code pour le rendre lisible aux humains.

Grâce à la touche *Tab*, il est possible de décaler le code qui se trouve à l'intérieur d'une balise pour bien faire comprendre qu'il est compris à l'intérieur de cette balise. Par exemple :

```
<div class="parent">
  <div class="enfant">
    <div class="petit-enfant"></div>
  </div>
  <div class="enfant">
    <div class="petit-enfant"></div>
  </div>
</div>
```

Pour mieux visualiser cette hiérarchie, vous pouvez installer l'extension *Indent Guides* via le gestionnaire d'extensions de Brackets.

Une fois installée, l'extension affichera des lignes verticales entre les balises ouvrantes et fermantes, le début et la fin des conditions et des boucles ainsi que sur toute l'étendue des fonctions. Voyez plutôt :



```
1 <?php
2 /**
3  * The template for displaying all single posts and attachments
4  *
5  * @package WordPress
6  * @subpackage Twenty_Fifteen
7  * @since Twenty Fifteen 1.0
8  */
9
10 get_header(); ?>
11
12 <div id="primary" class="content-area">
13 <main id="main" class="site-main" role="main">
14
15 <?php
16 // Start the loop.
17 while ( have_posts() ) : the_post();
18
19 /*
20  * Include the post format-specific template for the content. If you
21  * use this in a child theme, then include a file called called
22  * (where ___ is the post format) and that will be used instead.
23  */
24 get_template_part( 'content', get_post_format() );
25
26 // If comments are open or we have at least one comment, load u
27 if ( comments_open() || get_comments_number() ) :
28     comments_template();
29 endif;
30
31 // Previous/next post navigation.
32 the_post_navigation( array(
33     'next_text' => '<span class="meta-nav" aria-hidden="true">
34         '<span class="screen-reader-text">' . __( 'Next post:' );
35         '<span class="post-title">%title</span>',
36     'prev_text' => '<span class="meta-nav" aria-hidden="true">
37         '<span class="screen-reader-text">' . __( 'Previous pos
38         '<span class="post-title">%title</span>',
39     ) );
40
41 // End the loop.
42 endwhile;
43 ?>
44
```

Note : Pour activer l'extension, il faut aller dans *Affichage > Indent Guides*.

*Indent Guides* vous aidera également à structurer correctement votre code.

Sans cette rigueur, vous risquez de ne plus rien comprendre à ce que vous avez fait si vous consultez votre code dans trois mois.

## Brackets WordPress Hint

Il y a fort à parier que vous ne connaissez pas l'ensemble des fonctions de WordPress, rassurez-vous c'est le cas de la majorité des gens, moi y compris.

Heureusement, l'extension *Brackets WordPress Hint* permet de suggérer des fonctions et des hooks lorsque l'on écrit du code.

Après avoir installé l'extension, une liste déroulante s'affichera à chaque fois que vous commencerez à taper un mot dans le code PHP :



```
28
29 ▼   <div id="sidebar" class="sidebar">
30 ▼       <header id="masthead" class="site-header" role="banner">
31 ▼           <div class="site-branding">
32               <?php
33               if ( is_front_page() && is_ ) : ?>
34                   <h1 class="site-title"> is_checkout() L( home_url( '/'
'<a href="' . home_url( '/'
'</a></h1> is_checkout_pay_page()
35                   <?php else : ?> is_child_theme() L( home_url( '/'
36                   <p class="site-title"> is_comments_popup()
'<a href="' . home_url( '/'
'</a></p> is_customize_preview
37                   <?php endif; is_date()
38                   is_day()
39                   $description = get_bloginfo( 'description' );
40                   if ( $description || is_customize_preview() ) :
41                       <p class="site-description"> is_dynamic_sidebar()
'<?php echo $description; ?></p>
42                   <?php endif;
43               ?>
44               <button class="secondary-toggle"><?php _e( 'Menu and widgets', 'twentyfif
45               </div><!-- .site-branding -->
46           </header><!-- .site-header -->
47
48       <?php get_sidebar(); ?>
49   </div><!-- .sidebar -->
```

*Brackets WordPress Hint* risque de vous faire découvrir un grand nombre de fonctions et de hooks. Si vous êtes curieux, vous pouvez en savoir plus sur ces fonctions en les recherchant sur Google.

## Extensions de navigateur

Pour finir, attardons-nous sur trois extensions de navigateurs. Ces extensions vont permettre d'aller un peu plus loin que ce qui est proposé par l'inspecteur de code.

La première est...

## ColorZilla

Êtes-vous déjà tombé sur une couleur que vous avez voulu utiliser sur votre site ? Le soucis c'est que sans le code de la couleur, on ne peut rien faire.

*ColorZilla* est une extension pour **Chrome** et **Firefox** qui vous permettra de récupérer n'importe quelle couleur issue d'une page web.

Après avoir cliqué sur la pipette en haut à droite du navigateur (quand l'extension sera installée) puis sélectionné *Pick Color From Page*, la couleur de l'élément survolé s'affichera en haut de la page.

Par exemple, le bleu utilisé sur WP Marmite est le suivant :



En plus de capturer les couleurs des pages web, *ColorZilla* possède un sélecteur de couleur (*Color Picker*) et un synthétiseur de couleurs (*Webpage Color Analyser*) qui vous permettra d'obtenir la liste des couleurs principales d'une page.

## WhatFont

Toujours pour satisfaire votre curiosité, l'extension *WhatFont* va permettre de découvrir quelles polices d'écriture sont utilisées par une page web.

Suite à son installation, il suffit de cliquer sur l'icône placée en haut à droite du navigateur et de cliquer sur un texte pour obtenir des informations sur la police d'écriture :



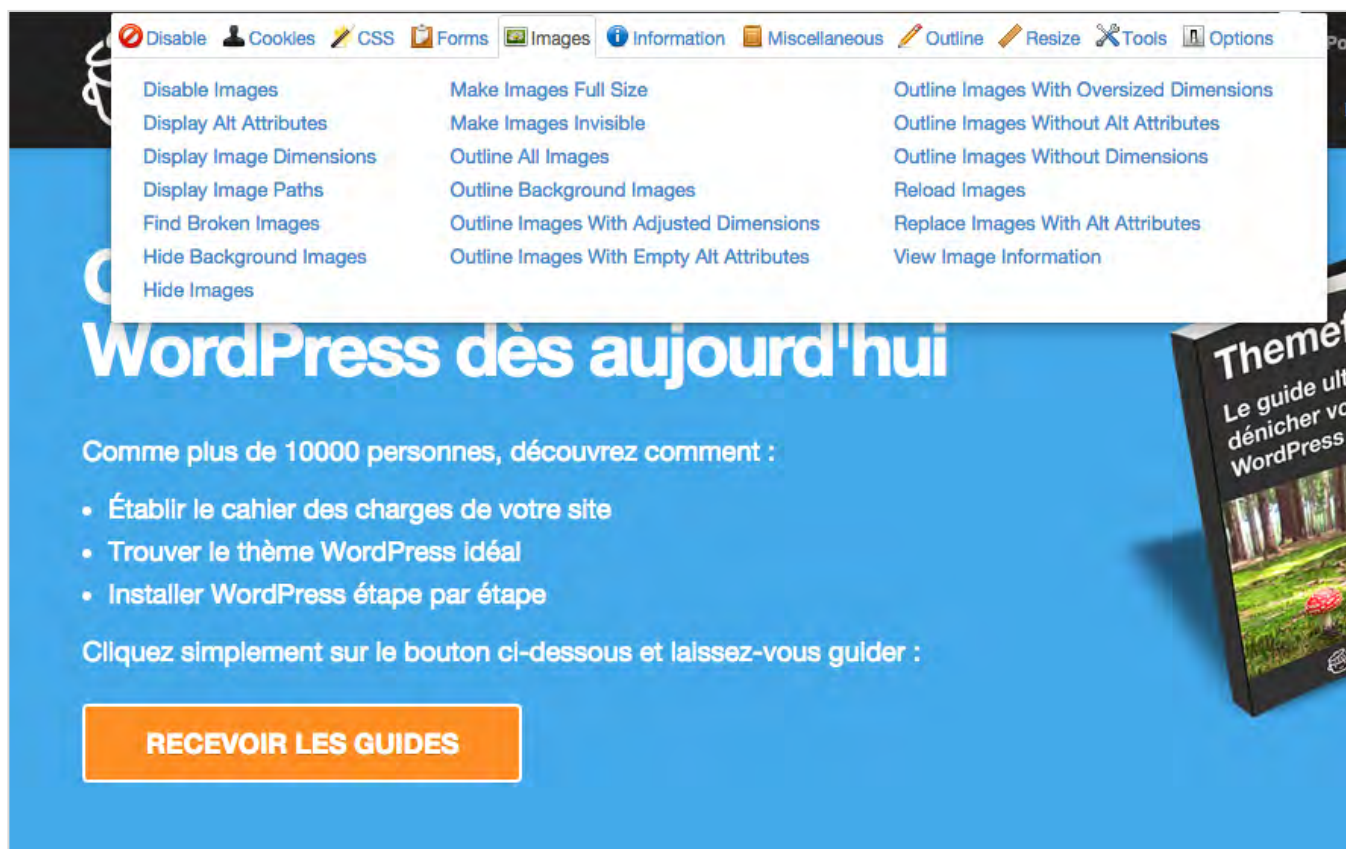
*WhatFont* est disponible sur [Chrome](#). Pour Firefox, un bookmarklet est disponible [sur le site officiel](#).

## Web Developer

La dernière extension de cette sélection est un peu plus complexe à manipuler, cela dit elle permet un grand nombre de choses.

Web developer permet notamment de :

- Désactiver / Afficher certains styles CSS
- Désactiver / Mettre en avant certaines images
- Afficher des informations particulières
- Mettre en avant certains types d'éléments
- Redimensionner le navigateur à des tailles prédéfinies



Le seul inconvénient est que *Web Developer* est en anglais. Pour vous aider, voici quelques traductions :

- **Display** : Afficher
- **Disable** : Désactiver
- **Outline** : Mettre en avant

- **View** : Voir
- **Replace** : Remplacer
- **Hide** : Cacher
- **Find** : Trouver
- **Reload** : Recharger

Web developer est disponible sur **Chrome** et **Firefox**.

## Récapitulons

De nombreuses extensions existent afin d'être assisté lorsque l'on veut créer ou modifier du code.

Quelque soit leur nature, ces extensions permettent de gagner en productivité et de découvrir de nouvelles choses.

Toutefois, il faut les utiliser à partir d'un certain niveau de maîtrise. C'est un peu comme si on fournissait des outils haut de gamme à un ouvrier débutant. Il faut déjà se faire la main et ensuite monter en puissance.

## Chapitre 5.6

# 11 Erreurs à éviter absolument en relookant un thème

Quand on se lance dans la personnalisation d'un thème pour la première fois, on peut commettre quelques erreurs.

Certaines sont bénignes et d'autres ont plus de conséquences, pourtant il est important de les connaître afin d'arriver à vos fins le plus rapidement possible.

Après avoir parcouru cette liste, vous serez paré pour commencer à faire vos premières retouches.

Allez, débutons avec une erreur qui permet justement de ne pas voir d'autres erreurs. Il s'agit de...

## 1. Ne pas activer le mode de débogage

Lorsque l'on développe avec WordPress et que l'on touche aux fichiers PHP, on n'est pas à l'abri de faire des erreurs. C'est pourquoi il faut activer ce que l'on appelle le *mode debug*.

Pour ce faire, il faut ouvrir le fichier `wp-config.php` situé à la racine du site et passer la constante `WP_DEBUG` à `true` (elle se situe presque à la fin du fichier).

Grâce à cela, les erreurs PHP s'afficheront en haut de la page. Si cela vous arrive, ne paniquez pas. Il est possible de les corriger.

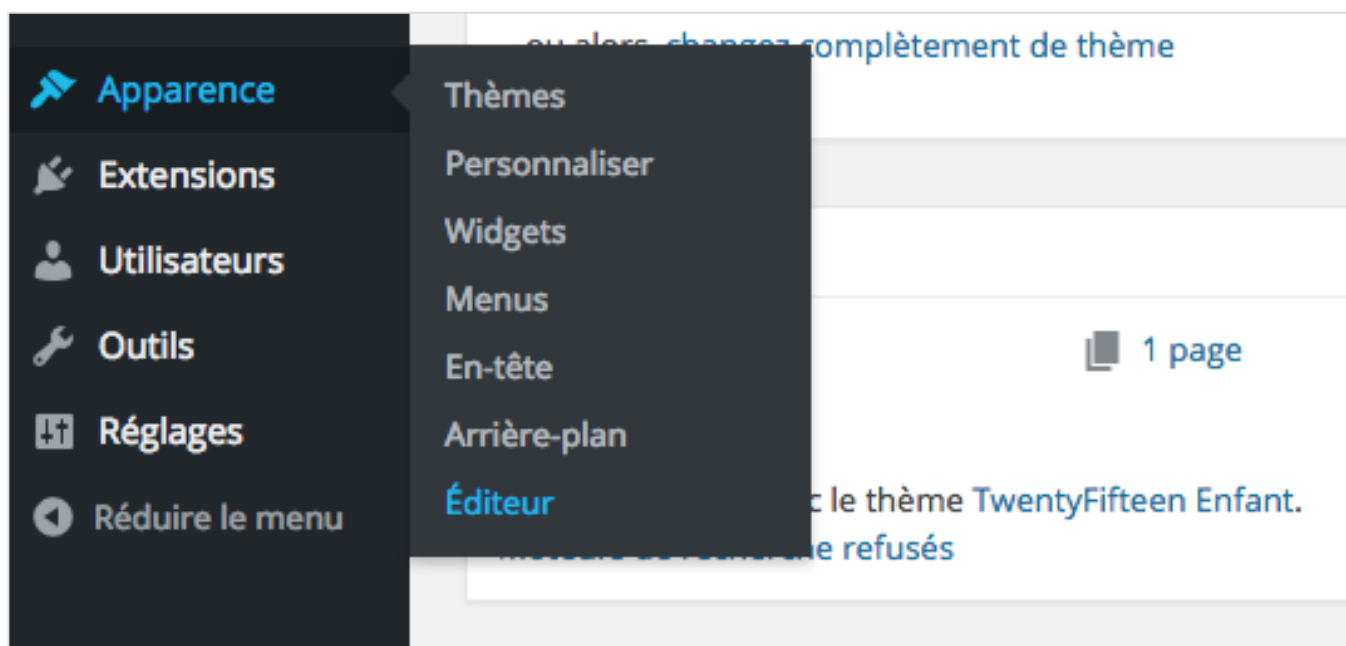
Si par exemple vous obtenez une erreur similaire à *Parse error: syntax error*, c'est probablement que vous avez oublié un point-virgule (;), une accolade, un marqueur `endif` ; ou `endwhile` ;.

Cela se joue parfois à pas grand chose. Il faut être vigilant.

*Si vos modifications ne concernent que le CSS, vous n'avez pas besoin d'activer le mode debug.*

## 2. Utiliser l'éditeur de thème de WordPress

Vous avez peut-être remarqué que WordPress embarquait un éditeur pour personnaliser le code de ses thèmes. Il se trouve dans *Apparence > Éditeur* :



Pourtant, il ne faut l'utiliser sous aucun prétexte.

Pourquoi donc me direz-vous ?

C'est très simple. Imaginez-vous en train d'éditer un fichier PHP et manque de chance, vous omettez d'inclure un point-virgule (;).

Que va-t-il se passer ?

Comme nous l'avons vu juste avant, une erreur se produira. **Sauf que vous n'aurez plus accès à l'éditeur pour la corriger !**

*C'est dans ces moments là que l'on regrette d'être passé par l'éditeur de thème de WordPress.*

Veillez donc à relooker votre thème à partir d'un éditeur de code installé sur votre ordinateur.

Pour prévenir d'éventuelles bêtises et par mesure de sécurité, il est préférable de désactiver l'éditeur de thème (et d'extension) en incluant la constante suivante dans le fichier `wp-config.php`:

```
define('DISALLOW_FILE_EDIT', true);
```

### **3. Mettre des !important partout dans son CSS**

Quand on débute avec le code CSS, on a tendance à vouloir chercher la solution de facilité en utilisant le mot-clé `!important` à tout bout de champ.

Après tout, ça marche n'est-ce pas ?

Bien sûr mais à la longue, cela sera plus contre productif qu'autre chose.

Comment faire si vous désirez apporter une nouvelle modification trois mois plus tard ?

En ayant utilisé un bon sélecteur dès le départ, on s'évite pas mal d'ennuis par la suite.

Alors certes, ce n'est pas toujours facile à trouver. Il faut rechercher les classes et identifiants nécessaires pour afficher ce que l'on veut où l'on veut, etc.

*Croyez-moi, vous ou la personne qui fera de nouvelles modifications vous remerciera plus tard.*

Pensez également à mettre des commentaires, que ce soit dans le code CSS ou PHP, même si cela peut vous sembler trivial.

Quand on sait ce que des lignes de code effectuent, cela évite pas mal de questionnements.

## **4. Ne pas insérer de préfixes à ses fonctions**

Nous en avons déjà parlé dans les chapitres précédents mais il est important d'insister.

Si jamais vous devez créer vos propres fonctions, ajoutez-leur un préfixe. Il faut faire cela pour vous assurer que leurs noms soient uniques.

*En effet, si deux fonctions ont le même nom, WordPress ne saura pas laquelle utiliser et déclenchera une erreur.*

Choisir un préfixe n'est pas compliqué, vous pouvez prendre vos initiales ou celles de votre site suivi du caractère `_`.

Pour vous donner une idée, le site de test que j'ai créé à l'occasion de l'écriture de Relooker son Thème possède des fonctions préfixées par `rst_` (exemple `rst_enqueue_styles()`).

## **5. Ne pas utiliser de thème enfant**

Là aussi, cela devrait être de l'acquis. Il faut tout de même insister car très peu de monde utilise le système des thèmes enfants pour relooker un thème.

Et là, attention les dégâts en cas de mise à jour du thème parent...

Je compte sur vous pour diffuser la bonne parole autour de vous !

## 6. Ne pas utiliser correctement la hiérarchie des templates

Personnaliser un thème WordPress à l'aveuglette peut faire naître des maux de tête, il faut le savoir.

Plus sérieusement, vous gagnerez du temps à utiliser les fichiers de la hiérarchie des templates pour créer des agencements sur mesure.

Petit conseil si vous créez des modèles de page personnalisés, ne nommez pas vos fichiers `page-quelquechose.php` mais plutôt `template-quelquechose.php`.

En regardant bien la hiérarchie des templates, vous verrez que si une page possède le *slug* `quelquechose`, le modèle de page sera utilisé alors que vous ne l'aurez pas voulu car le nom correspondra au format `page-[slug].php`.

Avec le second type de nom, vous éviterez ce genre de désagrément.

## 7. Ajouter trop de fonctionnalités dans un thème

Si vous prenez goût au PHP, vous pouvez être pris de frénésie et créer des fonctions pour toutes sortes de choses (insérer des codes de suivi, créer des shortcodes, etc.).

Dans le cas où cela ne s'apparente pas précisément au relooking d'un thème, il faut mieux placer cela dans ce que l'on appelle *une extension de fonctionnalités*.

Cette extension devra contenir tous les ajouts non liés à l'apparence (et donc au thème).

La question à se poser pour savoir si une modification doit être incluse ou non dans le thème est la suivante :

*Si je désactive le thème, ai-je encore besoin de cette fonctionnalité ?*

Si la réponse est positive, la fonctionnalité doit se trouver dans l'extension. Elle doit être placée dans le thème dans l'autre cas.

Cela dépasse légèrement le cadre de ce guide mais voici ce qu'il faut faire pour créer une extension de fonctionnalités :

- Créez un dossier nommé `monsite-fonctionnalites` dans le dossier `wp-content/plugins/`
- Créez un fichier nommé `monsite-fonctionnalites.php` à l'intérieur
- Ajoutez le code ci-dessous au début du fichier :

```
<?php
/**
 * Plugin Name:      Extension de fonctionnalités
 * Description:      Cette extension permet d'inclure
des fonctionnalités précises sur mon site sans craindre
de les perdre suite à un changement de thème.
 * Version:         1.0.0
 * Author:           Entrez votre nom
 * Author URI:       http://votresite.com/
 * License:          GPL-2.0+
 * License URI:      http://www.gnu.org/licenses/gpl-
2.0.txt
 */
```

- Ajoutez vos fonctionnalités comme si vous étiez dans le fichier `functions.php`

Note : N'oubliez pas d'activer l'extension pour mettre l'extension en service.

## 8. Ne pas se servir des hooks à disposition

De plus en plus de thèmes incluent des hooks pour se laisser personnaliser plus aisément, il ne faut pas les ignorer.

Dans un thème, les deux utilisations les plus courantes des hooks sont :

- L'insertion de contenu à des endroits précis avec des actions (avant le titre d'un article, après le contenu d'une page, etc.)
- La personnalisation de certains textes via des filtres (nom du titre de la page blog, noms d'éléments sur la page d'accueil, etc.)

Au lieu de dupliquer ces fichiers dans le thème enfant et d'ajouter le code nécessaire, il est plus rapide de créer une fonction et de l'associer à un hook dans le fichier `functions.php` (ou dans l'extension de fonctionnalités).

## 9. Ne pas utiliser la documentation

Le [Codex](#) et le [Code Reference](#) sont là pour donner plus d'informations aux créateurs de thèmes et d'extensions sur le fonctionnement de WordPress.

Il est important de s'y rendre régulièrement si l'on veut en savoir plus sur une fonction ou un hook par exemple.

Par exemple, pour en savoir plus sur la fonction `get_the_content()`, tapez « `get_the_content` WordPress » dans Google et la page du Codex devrait arriver en première position.

Certaines pages du Codex sont traduites en français. Dans le cas contraire, utilisez Google traduction pour déchiffrer tout cela.

## 10. Ne pas demander de l'aide

La communauté WordPress est formidable, il y a toujours quelqu'un pour donner un coup de main.

Il va de soi que les gens ne vont pas travailler sur votre site à votre place mais la plupart des membres de la communauté se feront un plaisir de vous aider.

Avant de demander de l'aide, faites des recherches au préalable. Si la question peut être résolue en 2 minutes sur Google, vous comprenez bien que cela est assez pénible d'y répondre...

*Vous devez acquérir le réflexe de rechercher des solutions aux problèmes auxquels vous êtes confronté avant de faire appel à la communauté.*

Cette précision étant apportée, voici comment obtenir de l'aide :

- **Le forum de support de l'association WordPress FR**
- Le hashtag **#WPAide** sur Twitter
- L'espace de discussion de la communauté **sur Slack (demandez une invitation ici)**

## 11. Ne pas indenter son code

La dernière erreur à ne pas commettre est d'avoir un code illisible.

Sur le moment, vous ne voyez probablement pas en quoi cela peut gêner mais si vous reprenez ces fichiers dans six mois, vous changerez certainement d'avis.

*Indenter son code permet d'avoir une structure claire et aisément compréhensible.*

Si un jour vous travaillez en équipe (ou si c'est déjà le cas), vos collègues vous remercieront d'avoir fait les choses correctement.

Vous pouvez aussi voir ça d'une façon différente : si votre code est propre, ils ne viendront pas vous poser des questions tous les quarts d'heure ;)

## **Bonus : Coller du code PHP n'importe comment**

Sur Internet, on trouve parfois des morceaux de code PHP pour effectuer des traitements particuliers au sein d'un site. Cela est intéressant mais il ne faut pas les inclure n'importe où.

### **Erreur PHP 1 : Inclure une balise PHP ouvrante dans un bloc de code PHP**

Lorsque l'on copie du code sur un site, ce code peut inclure une balise PHP ouvrante : `<?php`.

Si vous collez ce code directement dans le fichier `functions.php`, une erreur se produira car deux balises ouvrantes se succéderont. Par exemple :

```
<?php
/* Contenu du fichier functions.php */

<?php // Contenu du morceau de code trouvé sur un site
```

Pour éviter ce problème, vous devez simplement supprimer la balise ouvrante du code que vous voulez incorporer.

## Erreur PHP 2 : Placer du code PHP à l'extérieur de balises PHP

À l'inverse de l'exemple précédent, certains sites partagent des morceaux de code sans inclure de balises PHP.

En copiant ce code dans un fichier de votre thème comme page .php, cela sera sans effet. Par exemple :

```
<h1>echo the_title();</h1>
```

Rappelez-vous toujours que du code PHP doit se trouver entre des balises PHP. Dans le cas contraire, cela ne fonctionnera pas. Le code suivant est correct :

```
<h1><?php echo the_title(); ?></h1>
```

## Erreur PHP 3 : Placer du code PHP dans une fonction existante

Parfois, certaines fonctions PHP des thèmes peuvent être assez longues. On les trouve généralement dans le fichier `functions.php`.

Si vous désirez inclure des morceaux de code trouvés sur internet, prenez garde à ne pas les copier dans une fonction existante, sinon attention à la catastrophe. Voici ce que cela peut donner :

```
<?php
function ma_super_fonction(){
/* Code de la fonction */

function une_autre_fonction(){
    /* Code de l'autre fonction */
}
}
```

Une fonction ne peut pas en inclure d'autres. Elle doivent bien être séparées. Il faut plutôt procéder de cette manière :

```
<?php
function ma_super_fonction(){
/* Code de la fonction */
}
function une_autre_fonction(){
    /* Code de l'autre fonction */
}
```

## Récapitulons

Travailler avec WordPress est passionnant mais cela demande d'adopter quelques bonnes pratiques afin de ne pas s'égarer.

Les bons outils, les bonnes pratiques et le bon état d'esprit vous permettront de progresser rapidement.

En cas de blocage, vous pouvez toujours faire appel à la communauté WordPress francophone pour arriver à vos fins.

## Chapitre 5.7

# **Comment mettre son nouveau thème en ligne**

Et voilà, le grand jour est arrivé. Vous êtes venu à bout de votre liste TOP (ou en tout cas des points importants).

Il est maintenant temps de présenter votre thème relooké aux visiteurs de votre site.

Malgré l'empressement que vous pouvez éprouver, il ne faut pas installer votre nouveau thème WordPress sans prendre quelques précautions.

*En effet, il faut fournir une excellente expérience aux personnes qui prennent du temps de visiter votre site (rappelez-vous de l'introduction de ce guide).*

S'ils ne passent pas un bon moment, il ne reviendront probablement pas et ne parleront pas de votre site autour d'eux.

Bien entendu, ce n'est pas ce que nous voulons.

La première question à se poser est :

## **Comment être sûr qu'un site s'affiche bien sur tous les navigateurs ?**

Lorsque vous faites vos modifications, vous ne voyez votre site que par un seul navigateur.

Toutefois, les gens qui seront amenés à le consulter pourront le faire à partir de navigateurs web totalement différents.

Certains utiliseront Chrome sur un Mac, d'autres Internet Explorer (forcément sous Windows), d'autres personnes seront habituées à Firefox, Safari ou même au navigateur norvégien Opera.

Il est donc nécessaire de tester votre site avec un maximum de navigateurs.

Pour ce faire, il est possible d'installer les principaux navigateurs (notamment grâce au site **Browse Happy**).

C'est à mon sens le minimum à faire pour s'assurer que son site fonctionne bien sur la majorité des navigateurs.

Note : Étant donné que votre site est privé, vous devrez vous connecter à votre site de test sur chaque navigateur.

Si vous voulez aller plus loin, vous pouvez utiliser les outils suivants :

- **IE Net Renderer** : Génère des captures d'écran pour les différentes versions d'Internet Explorer
- **BrowserShots** : Génère des captures d'écran pour les autres navigateurs sur tous les systèmes d'exploitation
- **BrowserStack** : Génère des captures d'écran et permet de tester des sites en direct sur tous les navigateurs et sur tous les systèmes d'exploitation. Ce service est limité à une certaine durée d'utilisation, il faut payer ensuite.

Note : Ces outils permettent de tester des sites accessibles au public. Désactivez temporairement la restriction du site pendant vos tests.

Si vous utilisez un thème WordPress récent, logiquement son auteur l'a optimisé pour la plupart des navigateurs. Vous n'aurez juste qu'à vérifier si vos modifications (dans le thème enfant) ne chamboulent pas tout.

La seconde question à se poser en matière de mise à l'épreuve d'un site est :

## **Comment s'assurer qu'un site est bien responsive ?**

Nous sommes confrontés à un problème similaire à ce que nous venons de voir avec les navigateurs : tout le monde n'a pas la dernière tablette ou le dernier téléphone à la mode.

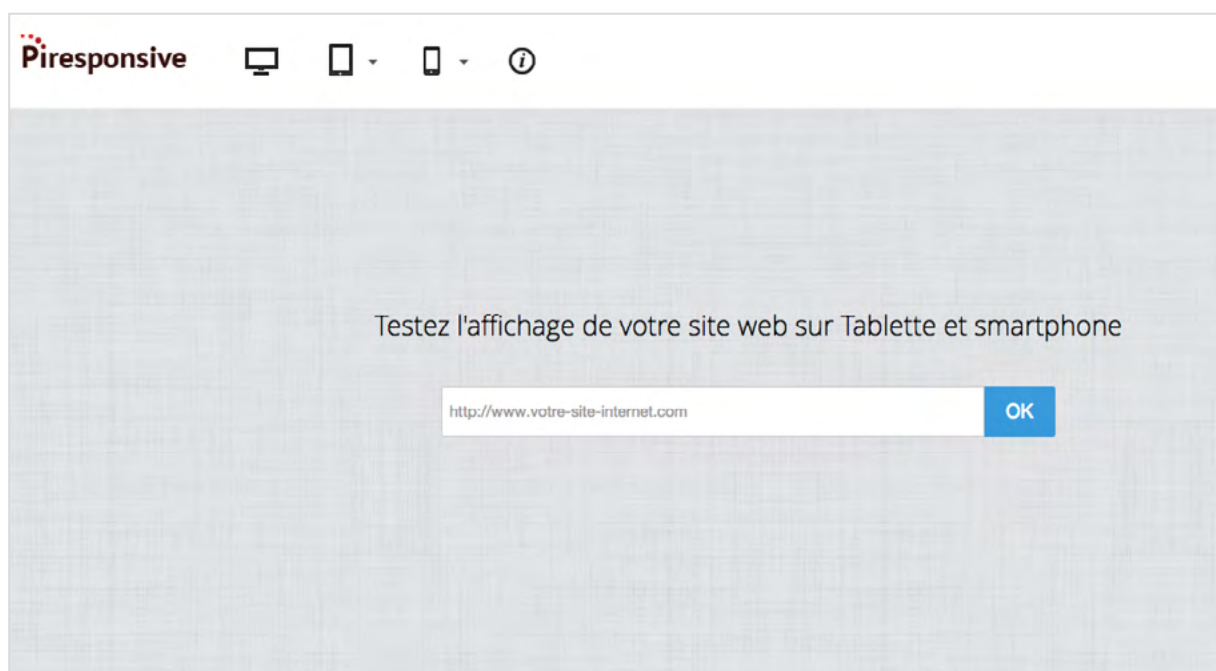
Par conséquent, il va être difficile de tester l'affichage sur ces appareils.

Pour répondre à cette problématique, deux solutions sont possibles.

### **1. Utiliser un site de test du responsive**

Il existe des dizaines de sites web permettant de tester l'affichage d'un site selon différentes tailles d'écran.

Si vous voulez en utiliser un en français, vous pouvez utiliser **Piresponsive** :



Note : Là encore, il faudra rendre son site temporairement public pour utiliser ce genre d'outils.

Tout ce que vous avez à faire est d'entrer l'adresse du site ou de la page que vous désirez tester et cliquer sur les icônes de tablettes ou de mobile

Un bouton est également disponible pour alterner entre l'affichage portrait et paysage.

Les outils de ce genre sont parfaits pour tester l'efficacité des media queries. Bien entendu, il est possible d'utiliser l'inspecteur de code pour tenter de corriger ce qui ne va pas.

Il est toutefois possible d'aller plus loin avec...

## 2. L'émulateur mobile de Chrome

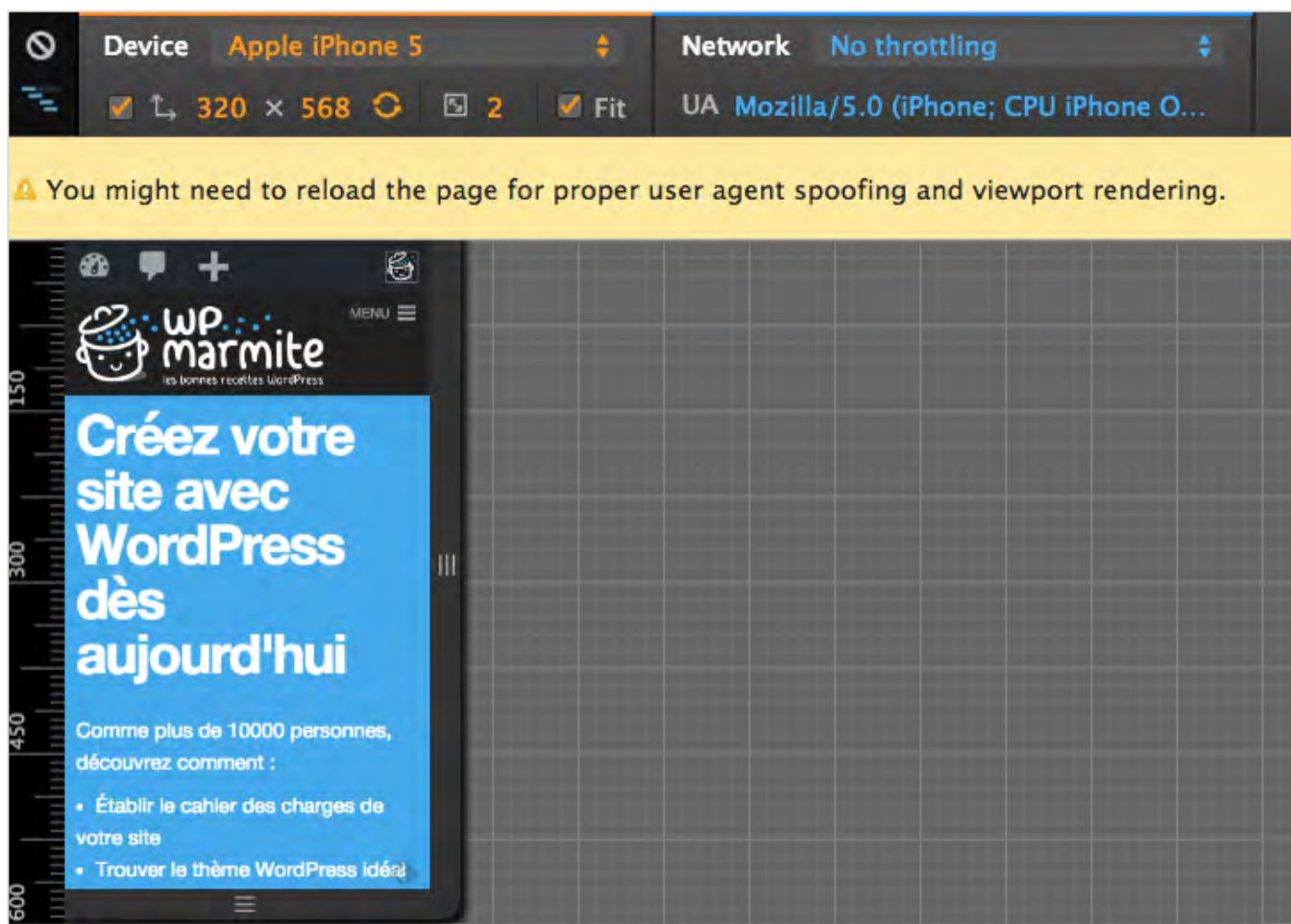
Un des avantages de l'inspecteur de code de Google Chrome est qu'il embarque un émulateur d'appareils mobiles (tablettes et téléphones).

Pour le lancer, ouvrez tout d'abord l'inspecteur de code de Chrome et cliquez sur l'icône ressemblant à un téléphone en haut à gauche (juste au dessus du code) :



Note : L'avantage de l'émulateur est que vous pouvez continuer de travailler sur votre site de test en toute tranquillité. Il n'y a pas besoin de désactiver la restriction appliquée au site.

Une fois l'émulateur lancé, voici ce que vous devriez obtenir :



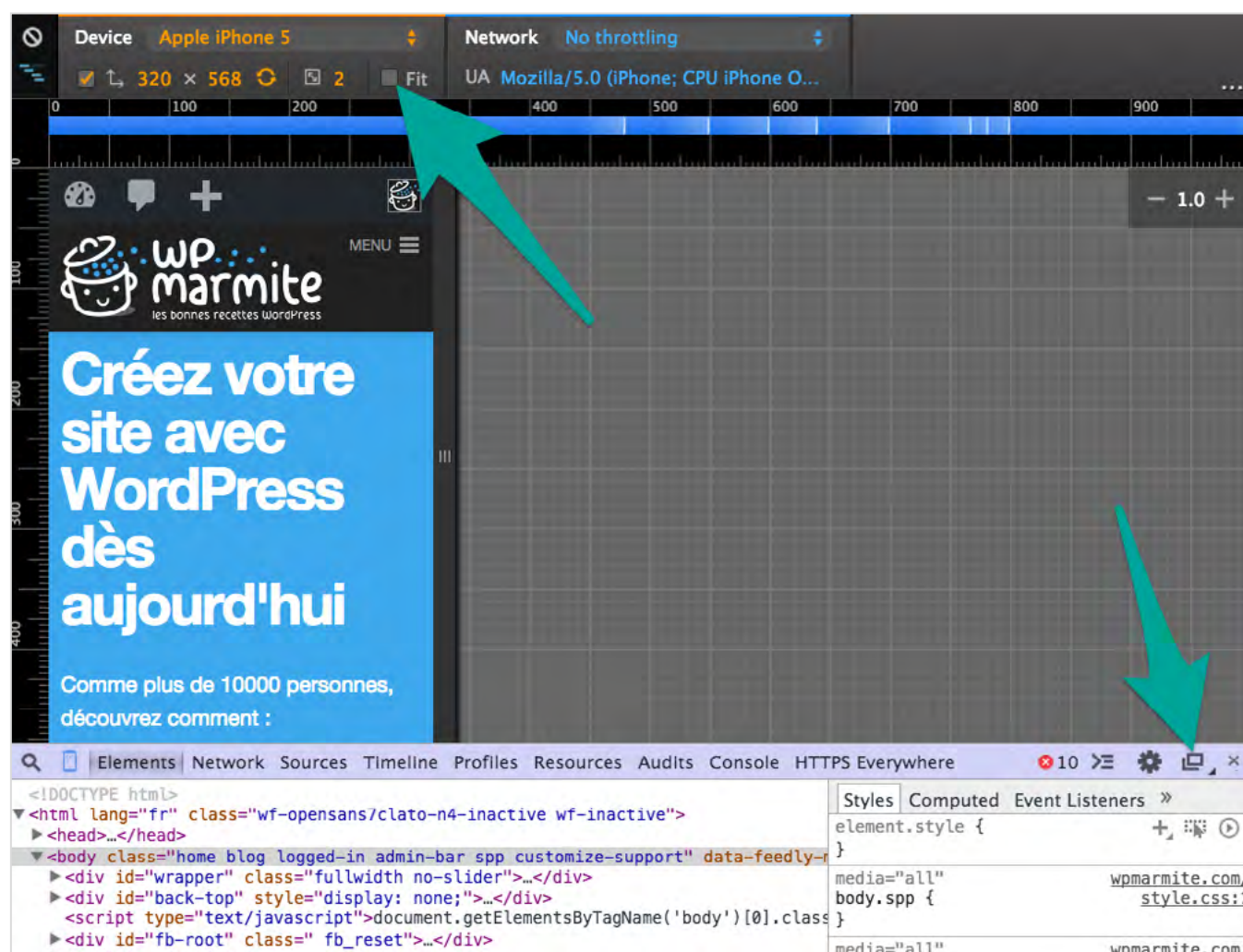
La notification en jaune vous demandera de recharger la page pour obtenir un affichage correct.

**La zone rédigée en orange concerne les types d'appareils** (« devices ») que vous désirez tester. Vous verrez que la liste déroulante est très bien fournie.

Les autres paramètres (taille d'écran et densité de pixels) seront automatiquement insérés en fonction de l'appareil que vous testerez.

Vous pouvez passer du mode portrait au mode paysage en cliquant sur le bouton situé à droite de la taille d'écran.

Pour y voir plus clair, il vaut mieux décocher la case *Fit* puis détacher l'inspecteur de code de la page afin d'éviter toute déformation :

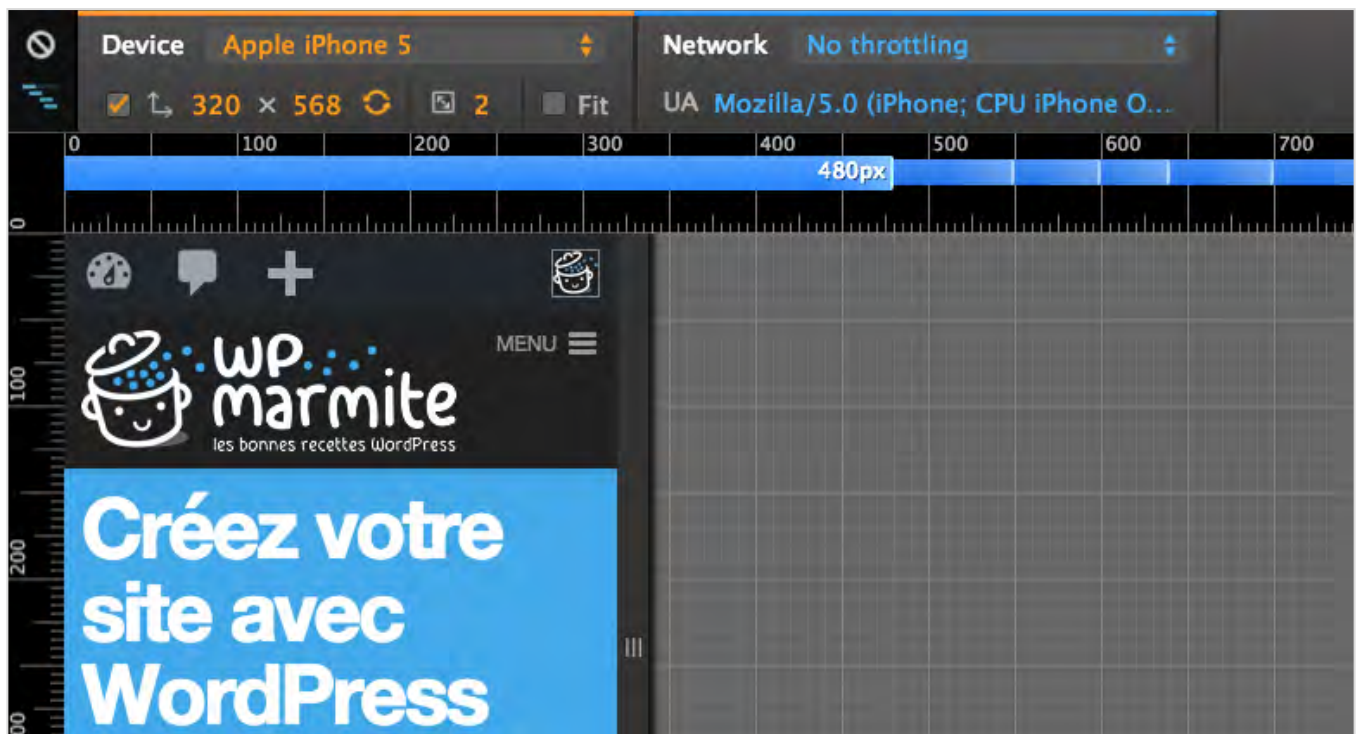


Il est bien sûr possible de continuer d'utiliser l'inspecteur pour faire ses tests CSS et HTML.

**La zone rédigée en bleu concerne la simulation de vitesse de connexion.**

Cela correspond à un usage assez avancé mais vous pouvez faire quelques tests pour voir comment un site s'affiche si la connexion est lente.

La dernière chose à connaître concernant l'émulateur de Chrome est le rôle de la barre bleue en haut de l'écran :



Si l'on y prête bien attention, on se rend compte que chaque délimitation correspond à une media query (480px, 550px, 600px, etc).

En cliquant sur chacune d'entre elle, on peut voir comment réagit le site. C'est très utile pour procéder aux derniers ajustements.

À présent, vous devriez être arrivé à un résultat satisfaisant sur votre site de test. Il est maintenant temps de partager vos derniers ajustement avec le public.

Il ne faut cependant pas faire cela directement. Voyons comment...

## Tester un thème relooké en toute discrétion

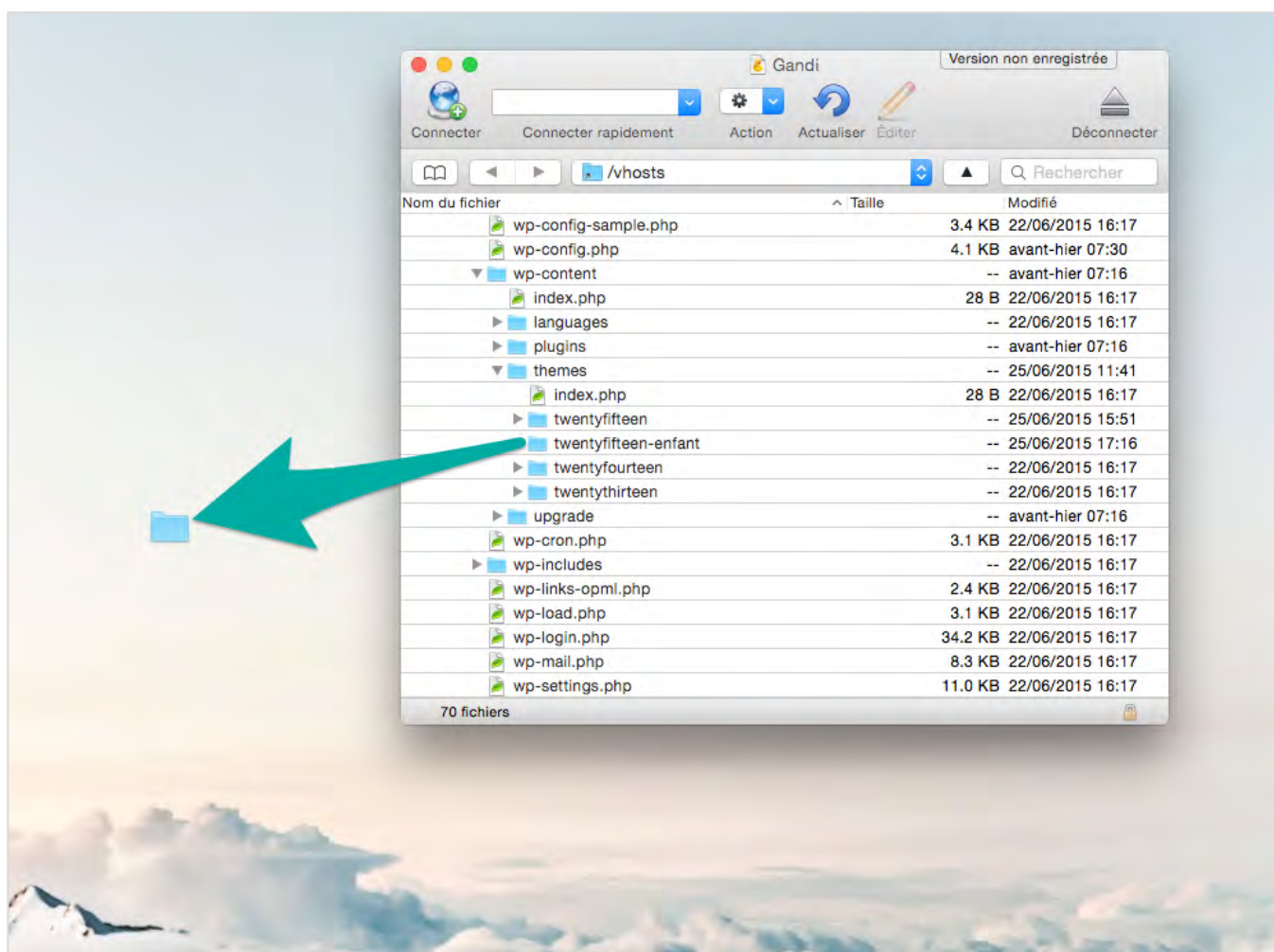
Avez-vous déjà rêvé de voir sans être vu ? Et bien cela est possible grâce à **l'extension Theme Test Drive**.

Cette extension permet de voir un thème WordPress en tant qu'administrateur alors que les visiteurs voient le thème activé par défaut.

Dès que vous aurez installé l'extension, il faudra rapatrier le thème enfant que vous avez créé (et le parent si vous avez changé de thème) sur le site de test au niveau du site principal.

Pour cela, connectez-vous à votre serveur via un client FTP (Cyberduck par exemple) et rendez-vous dans le dossier `/wp-content/themes/` de votre site de test.

Glissez déposez le dossier ou les dossiers nécessaires sur votre bureau (dans l'exemple ci-dessous, il s'agit uniquement du thème enfant) :



Puis renvoyez-le dans le dossier `/wp-content/themes/` de votre site (celui que les visiteurs voient).

Rendez-vous ensuite dans *Apparence > Theme Test Drive* pour configurer l'extension :

**Tableau de bord**

- Articles
- Médias
- Pages
- Commentaires
- Apparence**
- Thèmes
- Personnaliser
- Widgets
- Menus
- En-tête
- Arrière-plan
- Theme Test Drive**
- Extensions
- Utilisateurs
- Outils
- Réglages
- Réduire le menu

## Theme Test Drive

### Usage

1. Select a theme to preview live on the site from the box below (lists all installed themes).
2. Enable Theme Test drive
3. Once you have customized the theme to my liking, disable Theme Test Drive and enable the theme in WordPress. It will keep all previous settings. (sometimes depending on the theme setting are not saved. I am still investigating this)

Notes: Only administrator will be able to see the selected theme in Theme Test Drive. The site will show your normal theme to everyone else.

Additionally you may add "?theme=xxx" to your blog url, where xxx is the slug of the theme you want to test.

TwentyFifteen Enfant Theme Test Drive is Disabled.

You can specify the level of users to have access to the selected theme preview. By default it is set to 10 (admin only). Level 7 are editors, level 4 are authors and level 1 are contributors. The access level is ignored for accessing the site with ?theme=xxx parameter.

10 Access level

Disabling: If you wish to stop using Theme Test Drive, press *Disable* button. Alternatively, disabling this plug-in should also do the trick.

**Disable Theme Drive** **Enable Theme Drive**

L'auteur a inséré beaucoup de texte mais tout ce que vous avez besoin de savoir est qu'il faut :

- Sélectionner le thème WordPress enfant que vous avez créé dans le menu déroulant
- Cliquer sur le bouton *Enable Theme Drive* afin activer ce thème pour les administrateurs

Parcourez ensuite le site à la recherche de nouveaux problèmes d'affichage et notez-les.

Procédez ensuite aux corrections sur le site de test et recommencez l'opération jusqu'à ce que vous soyez satisfait.

Si tout est bon pour vous, vous pouvez désactiver la prévisualisation avec le bouton *Disable Theme Drive* des options de Theme Test Drive.

Avant de procéder à l'activation (cela doit vous démanger), vous devez lire ce qui suit...

## Activer son thème relooké : les deux cas de figure

Nous en avons peu parlé jusqu'à présent mais il y a deux types de relooking de thème WordPress :

1. Soit vous relookez le thème qui est déjà en ligne sur votre site (vous êtes aussi dans ce cas si vous venez d'installer WordPress)
2. Soit vous relookez un nouveau thème (qui remplacera votre thème actuel)

Dans le premier cas, c'est assez simple. Vous avez uniquement travaillé sur un thème enfant qui viendra ajouter un coup de frais à votre thème actuel.

**Vous pouvez donc procéder à l'activation de votre thème enfant.** Faites-le quand même lorsque votre site reçoit un minimum de trafic (pas mal de tests on été réalisés mais on ne sait jamais).

Dans le second cas, c'est un peu plus complexe.

En effet, si vous installez un nouveau thème (avec son thème enfant), **il faut prendre garde au fait que certaines choses pourront être impactées** lors du passage au nouveau thème.

Par exemple, si vous avez utilisé les shortcodes de votre ancien thème WordPress, ils ne vont plus fonctionner car le nouveau ne les possède pas.

Si vous désirez les conserver, vous allez devoir les trouver dans le code de votre ancien thème et les placer dans le thème enfant du nouveau (sur le site de test dans un premier temps, cela va de soi).

Autre exemple, si vous avez fait des ajouts de fonctionnalités dans le thème enfant de l'ancien thème et que vous désirez les conserver (ajout de scripts,

de types de contenus, etc), il va falloir les copier dans le nouveau thème enfant (ou dans une extension de fonctionnalités).

*Si vous êtes dans cette situation, vous devriez voir l'intérêt de créer une extension de fonctionnalités pour tout ce qui n'est pas lié au thème.*

Pour finir, installez une extension permettant d'afficher une page d'attente (comme **WP Maintenance**).

Cela permettra de faire patienter les visiteurs pendant l'installation du nouveau thème.

## **Que faire après avoir activé son thème relooké**

Que ce soit pour un nouveau thème ou uniquement l'installation d'un thème enfant, il faut être vigilant et procéder à quelques vérifications.

### **1. Vérifier que les menus et les widgets s'affichent bien**

Si ce n'est pas le cas, allez dans *Apparence > Menus* et cochez la ou les cases d'attributions d'emplacement de menus pour choisir vos menus.

Ensuite allez dans *Apparence > Widgets* et glissez déposez les anciens widgets qui ont dû être placés dans la zone *Widgets désactivés* (en bas de la page).

### **2. Régénérer les images à la une**

Si vous avez modifié la taille des images à la une, il va falloir les régénérer pour éviter d'avoir des articles dotés de problèmes d'affichage.

Nous avons parlé de l'extension **Regenerate Thumbnails** dans la partie 5 de ce chapitre. Il est maintenant temps de vous en servir.

### **3. Vérifier la configuration du thème**

Allez sur la page d'options du thème (ou dans le menu *Personnaliser*) pour vérifier que tout est en ordre.

Dans le cas d'un nouveau thème, reportez les paramètres indiqués sur le site de test pour retrouver la même apparence.

Certains thèmes WordPress proposent un outil d'import/export de leur configuration, servez-vous en pour gagner du temps.

### **4. Parcourez le site à la recherche d'éventuels problèmes**

En toute logique, si vous avez bien testé le thème sur le site de démonstration, il ne devrait pas y avoir de gros problèmes.

Peut-être qu'une ou deux corrections seront nécessaires ici et là mais rien de catastrophique.

### **5. Célébrez cela avec vos visiteurs**

Quand vous estimerez que le site sera prêt, désactivez la page d'attente pour rendre définitivement votre nouveau site accessible au public.

Vous pouvez rédiger un article pour l'annoncer à vos visiteurs et/ou diffuser la nouvelle sur vos réseaux sociaux (envoyer un email à vos clients/abonnés est aussi une bonne idée).

Les internautes pourront vous faire part d'éventuels dysfonctionnement. il faudra alors les corriger sur votre site de test puis les reporter sur votre site principal.

*Retenez qu'il faut toujours faire des modifications sur son site de test. Il ne faut pas prendre le risque de rendre votre site principal indisponible.*

# Récapitulons

Après tout ce travail, vous arrivez enfin au bout du tunnel : votre thème WordPress relooké va être mis en ligne.

Il ne faut pas faire cela brutalement au risque de perturber la navigation des visiteurs actuels (s'il y en a). Pour cela, il faut procéder avec méthode puis vérifier pas à pas que tout fonctionne bien.

Si vous prévoyez d'apporter d'autres modifications sur ce thème à l'avenir, n'oubliez pas de tester cela sur le site de test. Si cela est concluant, transférez ces ajouts sur le site principal.

Une fois que tout sera bon, procédez à une sauvegarde de votre site par mesure de sécurité.

**RELOOKER SON THÈME**

# **CHAPITRE 6**

## **17 MODIFICATIONS À COPIER/COLLER POUR SE LANCER**

---

**Commencez à relooker un thème  
dès aujourd'hui en adaptant  
des morceaux de code  
à votre projet**

## Chapitre 6.1

# 10 Astuces CSS pour relooker son thème

Débutons notre série de morceaux de code avec des astuces CSS afin d'aller plus loin que ce qui a été abordé dans le chapitre 3.

Vous verrez que ce langage possède quelques subtilités qui peuvent s'avérer extrêmement pratiques.

Commençons par voir comment...

## 1. Centrer un bloc grâce à la propriété `margin`

```
.blocacentrer{  
    margin: 0 auto;  
}
```

Centrer du texte est assez simple avec la propriété `text-align`. Pour des blocs, cela est plus compliqué (surtout quand on ne connaît pas cette astuce toute simple).

En fait, il s'agit de définir des marges externes latérales automatiques pour le bloc à centrer. Il sera ainsi à égale distance des côtés de son bloc parent, c'est-à-dire centré.

Note : le bloc parent doit avoir une largeur définie en pixels pour que cela fonctionne (avec `width` ou `max-width`).

[Voir cet exemple en action](#)

## 2. Changer l'apparence du curseur de la souris

Dans certains cas, on peut avoir besoin de changer le curseur de la souris. Cela tombe bien car une propriété CSS existe justement pour ça.

Voici quelques valeurs pouvant être associées à la propriété `cursor` :

- **auto** : curseur par défaut
- **pointer** : curseur de type lien
- **text** : curseur de type sélection de texte
- **zoom-in** : curseur zoom +
- **zoom-out** : curseur zoom -
- **none** : pas de curseur
- **move** : curseur de déplacement

```
.monbloc{  
    cursor: pointer;  
}
```

[Voir cet exemple en action](#)

### 3. Styliser les éléments de formulaires plus précisément

Nous n'avons pas étudié en détail comment attribuer des styles aux éléments de formulaires dans le chapitre sur le CSS. Cependant, vous devriez y parvenir à l'aide de classes et d'identifiants.

Si vous avez déjà essayé d'appliquer un style à une balise `input`, vous avez dû constater que le formulaire pouvait ne pas s'afficher correctement (le bouton de validation étant compris dans une balise `input`, les styles lui sont également appliqués).

Il est heureusement possible de cibler les éléments de formulaire par type (les champs de texte, les champs emails, etc). Voici les sélecteurs à employer :

- **`input[type="text"]`** : cible les champs de type texte
- **`input[type="password"]`** : cible les champs de type mot de passe
- **`input[type="email"]`** : cible les champs de type email
- **`input[type="url"]`** : cible les champs de type adresse web (url)
- **`input[type="tel"]`** : cible les champs de type numéro de téléphone
- **`input[type="checkbox"]`** : cible les cases à cocher
- **`input[type="radio"]`** : cible les boutons radio
- **`input[type="file"]`** : cible les boutons de chargement de fichier
- **`input[type="submit"]`** : cible les boutons de validation

```
input[type="text"], input[type="email"]{
    font-size: 0.8em;
    border:2px solid #efefef;
}

input[type="email"]{
    border-color:#a7e9ff;
}

input[type="submit"]{
    font-size:16px;
    color:white;
    background-color:#00a498;
    border:none;
    padding:10px;
}
```

[Voir cet exemple en action](#)

## 4. Utiliser les pseudo-classes

Sans le savoir, vous connaissez déjà une pseudo-classe. Il s'agit du sélecteur `:hover`.

Une pseudo-classe correspond à un état d'un élément. Vous vous rappelez sûrement que `:hover` **correspond à l'état de survol** d'un élément par la souris.

Les autres pseudo-classes n'ont pas été abordées dans le chapitre sur le CSS pour ne pas vous en dire trop d'un coup mais vous devez savoir qu'il en existe davantage.

## 4.1 Pseudo-classes associées aux liens

Il est possible d'utiliser les pseudo-classes sur tous les types d'éléments mais les suivantes sont surtout employées pour les liens (balises a) :

- **:link** : Cible les balises a possédant un attribut href n'ayant pas été visité
- **:visited** : État visité d'un lien (l'utilisateur a déjà visité le lien renseigné en href)
- **:hover** : Survol de l'élément par la souris
- **:active** : Lien en train d'être cliqué

```
a:link{
    color:green;
}
a:visited{
    color:#aaa;
}
a:hover{
    color:white;
    background:#ddd;
}
a:active{
    color:red;
}
```

[Voir cet exemple en action](#)

## 4.2 Pseudo-classes associées aux éléments de formulaires

Lorsque l'on complète un formulaire, on interagit avec les éléments. Par exemple, on tape du texte dans un champ de saisie.

Dans ce cas, la pseudo-classe : focus sera très utile pour styliser les champs en cours de complétion. Cela permet de mettre en valeur le champ que le visiteur est en train de remplir.

Par exemple :

```
input[type="text"]{  
    border:4px solid transparent;  
}  
input[type="text"]:focus{  
    background:#eee;  
    border-color:#bbb;  
}
```

Note : Il est préférable d'ajouter une bordure transparente à l'élément initial et d'appliquer une coloration uniquement au focus. Dans le cas contraire, un décalage se produira. Testez et vous verrez.

[Voir cet exemple en action](#)

## 4.3 Pseudo-classes pour sélectionner certains éléments

Un bloc peut contenir divers éléments destinés à être stylisés différemment.

Pour y parvenir, on peut utiliser des classes (qu'il faudra insérer dans le HTML) ou employer les pseudo-classes suivantes :

- **:first-child** : Sélectionne le premier élément enfant
- **:last-child** : Sélectionne le dernier élément enfant
- **:nth-child()** : Sélectionne certains éléments enfants (en fonction des paramètres qui lui auront été transmis)

Voici quelques exemples basés sur le code HTML suivant :

```
<ul id="maliste">
  <li>Élément A</li>
  <li>Élément B</li>
  <li>Élément C</li>
  <li>Élément D</li>
  <li>Élément E</li>
  <li>Élément F</li>
  <li>Élément G</li>
</ul>

/* Le premier élément de la liste aura une taille de
1.5em (A dans notre exemple) */
#maliste li:first-child{
  font-size:1.5em;
}

/* Le dernier élément de la liste sera bleu (G dans notre
exemple)*/
#maliste li:last-child{
  color:blue;
}
```

```
/* Le 4e élément de la liste sera en italique (D dans
notre exemple) */
#maliste li:nth-child(4){
    font-style:italic;
}

/* Les éléments pairs seront en gras (B, D et F dans notre
exemple) */
#maliste li:nth-child(2n){
    font-weight:bold;
}

/* Les éléments pairs seront en gras (B, D et F dans notre
exemple) */
#maliste li:nth-child(even){
    font-weight:bold;
}

/* Les éléments impairs seront soulignés (A, C, E et G
dans notre exemple) */
#maliste li:nth-child(2n + 1){
    text-decoration:underline;
}

/* Les éléments impairs seront soulignés (A, C, E et G
dans notre exemple) */
#maliste li:nth-child(odd){
    text-decoration:underline;
}
```

```
/* Les 3 premiers éléments seront en gris (A, B et C dans
notre exemple) */
#maliste li:nth-child(-n+3){
    color: grey;
}

/* Tous les éléments sauf les 3 premiers seront décalés
vers la droite (D, E, F et G dans notre exemple) */
#maliste li:nth-child(n+4){
    padding-left: 20px;
}

/* Tous les 3 éléments auront du jaune en arrière-plan (A,
D et G dans notre exemple) */
#maliste li:nth-child(3n+1){
    background-color:yellow;
}
```

[Voir cet exemple en action](#)

## 5. Ajouter une image d'arrière-plan

Placer une image d'arrière-plan au niveau d'un site peut sembler trivial mais cela comprend quelques subtilités selon le rendu que l'on désire obtenir.

Par exemple, voici le code à utiliser pour placer une image d'arrière-plan qui s'adaptera à la largeur de l'écran :

```
body{
    background-image: url("img/monimage.jpg");
    background-repeat: no-repeat;
    background-position: center center;
    background-attachment: fixed;
    background-size: cover;
}
```

[Voir cet exemple en action](#)

Attention : Certaines zones de l'image peuvent être masquées.  
N'utilisez pas d'images incluant du texte.

## 6. Masquer des éléments rapidement sur les appareils mobiles

En optimisant un site pour le responsive, vous aurez peut-être envie de dissimuler des éléments sur les appareils mobiles.

Pour ce faire, utilisez la propriété `display` avec la valeur `none` dans une media query.

Par exemple, pour masquer une barre latérale :

```
@media screen and (max-width: 600px){
    .sidebar{
        display:none;
    }
}
```

[Voir cet exemple en action](#)

## 7. Optimiser les images pour les écrans Retina

Pour améliorer la qualité d’affichage de ses appareils, la société Apple a lancé des écrans équipés de la technologie Retina.

Concrètement, cela signifie que **ces écrans affichent plus de pixels sur une même surface par rapport à un écran classique**. Le rendu est donc beaucoup plus fin et la qualité de l’image est considérablement améliorée.

Note : les écrans Retina sont présents à partir de l’iPhone 4, les iPad de troisième et quatrième génération ainsi que certains MacBook, MacBook Pro et iMac.

Dans la plupart des cas, une image doit être deux fois plus grande pour s’afficher correctement sur un écran Retina.

Il est possible de gérer cela en CSS grâce à une media query utilisant des paramètres particuliers.

Voici le code complet que l’on pourrait utiliser pour afficher le logo d’un site :

```
.logo{
    background-image:url("img/logo.png");
}

@media
(-webkit-min-device-pixel-ratio: 2),
(min-resolution: 192dpi) {
    .logo{
        background-image:url("img/logo@2x.png");
    }
}
```

Pour s’y retrouver, on ajoute le suffixe @2x à la fin du nom des images pour savoir qu’elles sont destinées à être affichées sur des écrans Retina.

Note : Si vous désirez que toutes les images de votre site soient optimisées pour les écrans Retina, vous pouvez installer l’extension **WP Retina 2x**.

## 8. Styliser un tableau

Dans certains cas, on peut avoir besoin d’insérer un tableau dans son contenu.

Nous avons vu qu’un tableau se constituait des éléments `table`, `tr`, `th` pour les cellules de l’en-tête et `td` pour les cellules constituant le contenu du tableau.

Certains tableaux pourront être accompagnés des balises `thead`, `tbody` et `tfoot` représentant respectivement l’en-tête, le corps et le pied de page.

Le code suivant permet de styliser un tableau simplement :

```
table{
    border-collapse: collapse;
    border-spacing: 0;
    empty-cells: show;
    border: 1px solid #bbb;
}

thead{
    background-color: #d3d3d3;
    color:#111;
    text-align: left;
    vertical-align: bottom;
```

```

}

th,
td{
    padding: 1em 1.5em;
    border-left: 1px solid #bbb;
    margin: 0;
    overflow: visible;
}

td{
    background-color: transparent;
}

tr:nth-child(2n+1){
    background-color: #f2f2f2;
}

```

[Voir cet exemple en action](#)

## 9. Mettre un élément du menu en valeur

Quel que soit le type de site que l'on possède, il s'avère parfois nécessaire de mettre un élément du menu en avant.

Par exemple, cela peut être un lien vers une page de commande, une page de connexion, d'enregistrement, etc.

En se basant sur le thème Twenty Fifteen, tentons de styliser le bouton menant vers la page de commande :

# Relooker son Thème (dev)

Le site de démonstration

---

Accueil

---

A Propos

---

Commander

---

La première chose à faire est de regarder quels sont les classes et l'identifiant de l'élément que l'on désire styliser grâce à l'inspecteur de code (WordPress en ajoute automatiquement aux éléments de menus) :

# Relooker son Thème (dev)

Le site de démonstration

---

Accueil

---

A Propos

---

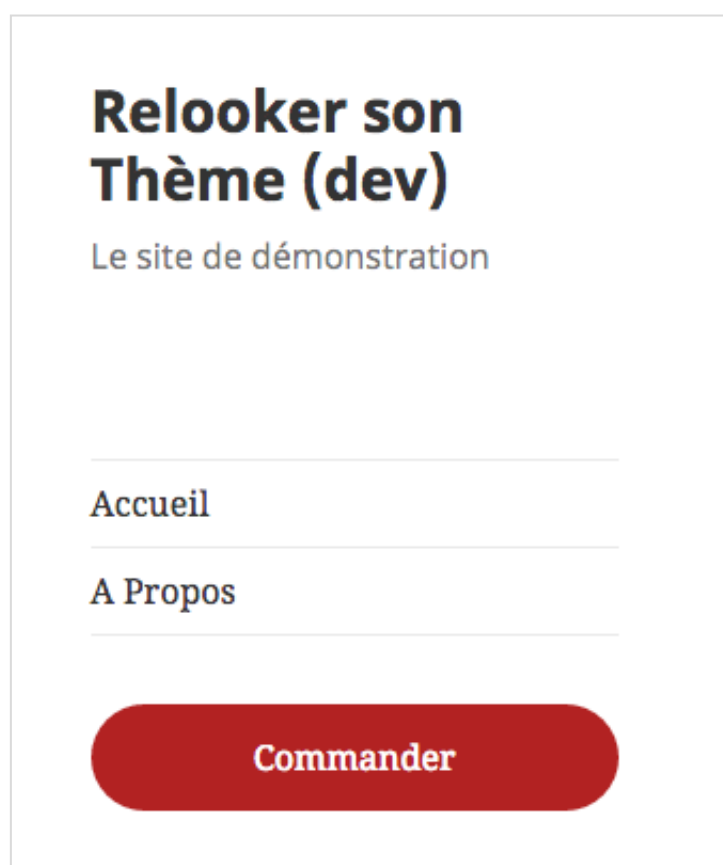
Commander

`li#menu-item-11.menu-item.menu-item-type-custom`

Grâce à cela, on peut rédiger le code suivant :

```
/* On style l'apparence du lien de l'élément "Commander"
*/
#menu-item-11 a {
    background-color: #b22222;
    color: white;
    padding: 0.8em;
    text-align: center;
    margin-top: 2em;
    border-radius: 50px;
}
/* Sans oublier l'effet de survol */
#menu-item-11 a:hover {
    background-color: #910A0A;
}
/* Et en éliminant la bordure inférieure de la liste "ul"
*/
.main-navigation ul {
    border-bottom: none;
}
```

Ces quelques lignes de code permettent d'obtenir le résultat suivant :

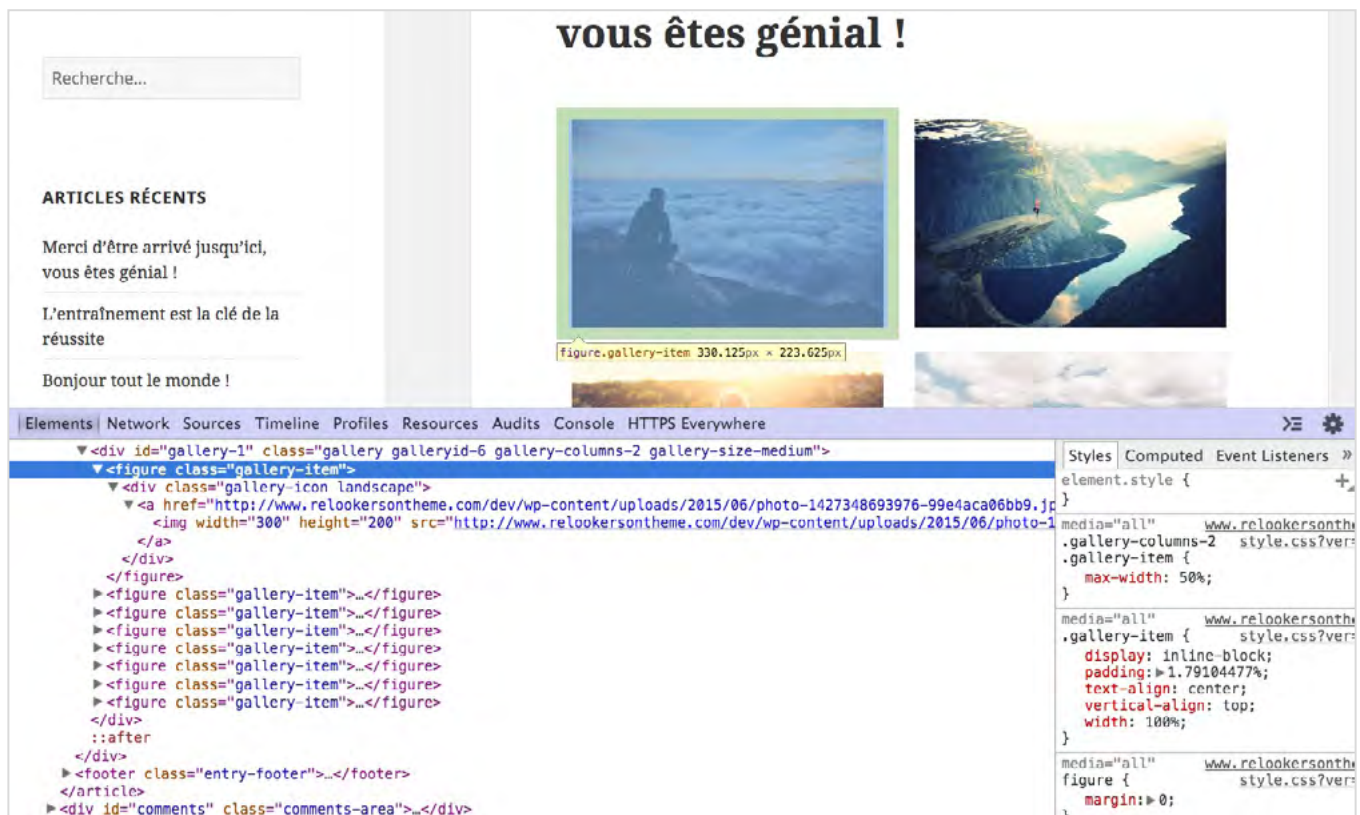


Les visiteurs ne pourront désormais plus passer à côté de cette page de commande.

## 10. Styliser une galerie d'images

Peu de personnes le savent mais WordPress permet nativement de créer des galeries d'images (cliquez sur *Ajouter un média* puis *Créer une galerie*).

Certains thèmes sont plus optimisés que d'autres pour les prendre en charge. Dans l'exemple qui va suivre, nous allons voir quels sont les classes et les identifiants présents dans les galeries.



La capture d'écran ci-dessus permet de voir de quoi se compose une galerie. Étudions les classes et identifiants qui les composent :

- **#gallery-1** : Identifiant de la galerie (pour styliser uniquement celle-ci)
- **.gallery** : Classe ciblant toutes les galeries
- **.galleryid-6** : Classe ciblant toutes les galeries de la publication d'ID 6
- **.gallery-columns-2** : Classe ciblant toutes les galeries en deux colonnes
- **.gallery-size-medium** : Classe ciblant toutes les galeries possédant des images en taille moyenne
- **.gallery-item** : Classe ciblant chaque élément de la galerie (contient l'image et une éventuelle description)
- **.gallery-icon** : Classe ciblant l'élément contenant l'image
- **.landscape** : Classe décrivant l'orientation de l'image (*landscape* vaut pour paysage et *portrait* pour portrait)

Si des descriptions sont ajoutées, on retrouvera :

- **.gallery-caption** : Classe ciblant la description associée aux images

- **#gallery-1-13** : Identifiant d'une description d'une image donnée (ici de l'image d'ID 13)

Bien entendu, il est possible (et grandement recommandé) de combiner ces sélecteurs pour obtenir une mise en page originale.

Tentons de transformer la mise en page de galerie en deux colonnes du thème Twenty Fifteen en collection de polaroids.

Une solution serait de procéder ainsi :

```
/* Ajoutons une couleur d'arrière-plan pour mieux faire
ressortir les images */
.gallery-columns-2 {
    padding: 2em;
    background-color: #F0F8FF;
}
/* Le style polaroid est appliqué aux galeries en deux
colonnes */
.gallery-columns-2 .gallery-item {
    border: 1px solid #555;
    background-color: #fff;
    margin-bottom: 30px;
    padding-bottom: 50px;
    max-width: 45%;
}
.gallery-columns-2 .gallery-item:nth-child(2n+1){
    margin-right: 10%;
}
```

Ce qui nous donne :



Certes, cela reste basique mais cela donne une bonne idée de ce qu'il est possible d'accomplir avec les galeries.

## Récapitulons

Les quelques astuces que vous avez pu découvrir dans cette première partie ne sont que la partie émergée de l'iceberg.

Avec une bonne idée, les bons sélecteurs et les bonnes propriétés, il est possible d'aller beaucoup plus loin.

Continuons la découverte de nouvelles astuces dans la partie suivante en nous focalisant cette fois sur WordPress.

## Chapitre 6.2

# 7 Astuces WordPress pour relooker son thème

Après avoir abordé dix astuces CSS, nous allons découvrir sept autres astuces liées à WordPress.

Pour les mettre en place, vous allez devoir ajouter des lignes de code dans le thème enfant créé au préalable.

N'oubliez pas de procéder à vos tests sur un site qui n'est pas accessible au public.

Commençons par voir comment...

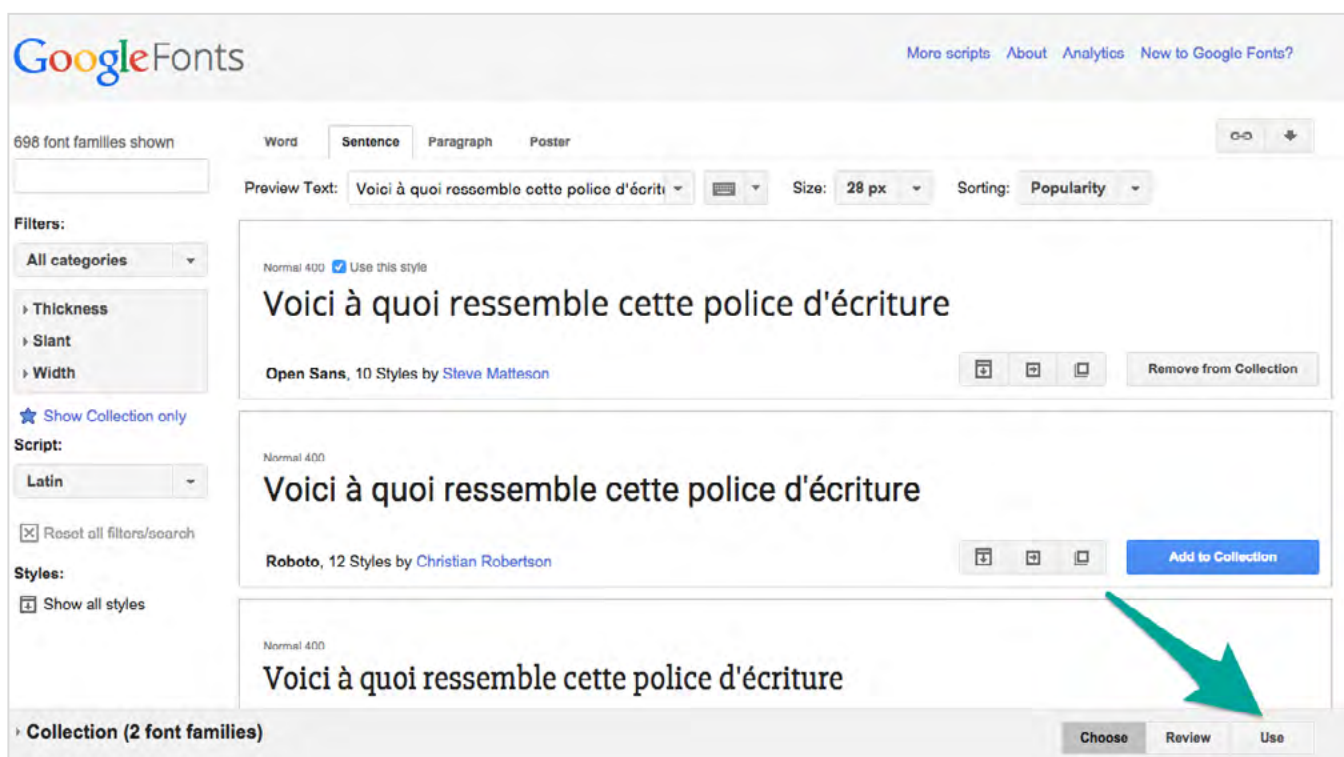
## 1. Intégrer une nouvelle police d'écriture avec Google Fonts

Si la police d'écriture d'un thème ne vous convient pas et qu'il n'est pas possible d'en changer via les options, vous pouvez vous tourner vers les polices d'écriture fournies par Google.

En allant sur [Google Fonts](#), vous bénéficierez de plusieurs centaines de polices d'écriture gratuitement.

Une fois que vous aurez trouvé la ou les polices d'écriture qui conviendront à votre site, cliquez sur le bouton *Add to Collection* pour les sauvegarder.

En bas à droite de la fenêtre de Google Fonts, cliquez sur le bouton *Use* :



Trois choses vous seront ensuite demandées :

1. Choisir parmi les variations offertes par chaque police (ce qui sera disponible via la propriété font-weight). Dans la plupart des cas, choisissez *normal* et *bold* (plus vous en cochez, plus votre site mettra de temps à se charger)
2. Choisir le type de jeu de caractères (seulement utile si vous avez un site utilisant le cyrillique, le grec ou le vietnamien). *Il y a de fortes chances pour que vous n'ayez rien à faire.*
3. Copier le code pour l'intégrer sur votre site.

*Concernant l'intégration, plusieurs méthodes sont proposées mais aucune ne correspond aux bonnes pratiques de WordPress.*

Il est vrai que l'on pourrait simplement copier la balise `link` fournie par Google Fonts dans le fichier header .php de son thème enfant.

Cette méthode n'est pas correcte car WordPress « ne saura pas » que ces polices d'écriture auront été chargées (il ne sera donc pas possible de les manipuler si besoin).

Pour faire cela proprement, **il faut passer par un de ces fameux hooks.**

Tout d'abord, copiez simplement l'adresse de chargement des polices choisies (la valeur de l'attribut href de la balise link) :

```
http://fonts.googleapis.com/  
css?family=Open+Sans:400,700|Pacifico
```

Puis, placez cette adresse dans le code suivant et ajoutez le tout dans le fichier `functions.php` du thème enfant :

```
// Fonction de chargement de polices Google Fonts  
function rst_load_fonts(){  
    wp_register_style( 'rst-fonts' , 'http://fonts.  
googleapis.com/css?family=Open+Sans:400,700|Pacifico' );  
    wp_enqueue_style( 'rst-fonts' );  
}  
add_action( 'wp_enqueue_script' , 'rst_load_fonts' );
```

Note : Utilisez le préfixe de votre choix à la place de `rst`.

Et voilà ! Maintenant vous pouvez appliquer une nouvelle police d'écriture aux éléments de votre site. Par exemple :

```
body{
    font-family: "Open Sans", "Arial", sans-serif;
}
h1,h2,h3,h4,h5,h6{
    font-family: "Pacifico", "Verdana", cursive;
}
```

## 2. Ajouter du contenu avant ou après les articles

Assez régulièrement, on peut avoir envie de placer du contenu avant ou après ses articles.

Certains sites ajoutent un encart de publicité directement au début de l'article, d'autres proposent leur produit ou une inscription à une newsletter aux personnes qui viennent de terminer leur lecture.

*Bref, les possibilités sont nombreuses. Elles varient en fonction du projet et des objectifs du site.*

Voyons comment faire cela avec n'importe quel thème WordPress.

Si vous avez bien retenu ce qui a été abordé au sujet de la hiérarchie des templates, il va falloir dupliquer le fichier `single.php` dans le thème enfant et insérer le code HTML nécessaire pour afficher ce que l'on désire avant ou après la fonction `the_content()`.

Le problème est que dans certains thèmes, la fonction `the_content()` ne figure pas toujours dans le fichier `single.php`.

Autre problème, si vous voulez afficher des contenus différents en fonction des articles, l'insertion des conditions va surcharger le code du fichier.

Il est donc préférable de travailler à nouveau avec les hooks. Rassurez-vous, nous allons voir comment faire.

Placez ce code dans le fichier `functions.php` de votre thème enfant et adaptez-le à votre situation :

```
// Fonction définissant l'encart de publicité à insérer au
dessus du contenu
function rst_ad_before_content( $content ) {

    // Si nous sommes bien dans un article (et dans la
    boucle principale)
    if ( is_single() && is_main_query() ){

        // On place le code HTML de la publicité dans une
        variable
        $pub = '<div class="pubtop"><a href="#"></a></
div>';

        // On place le contenu de la publicité avant celui
de l'article
        $content = $pub . $content;

    }

    // On retourne le contenu intégral (publicité +
article)
    return $content;
}

add_filter( 'the_content', 'rst_ad_before_content' );
```

Une fois le code inséré, il ne vous restera plus qu'à appliquer le code CSS nécessaire pour faire flotter l'encart à droite ou à gauche de vos articles.

Maintenant voyons ce qu'il faut faire pour afficher du contenu après les articles mais en affichant un encart différent selon les catégories.

Voici le code à employer :

```
// Fonction définissant l'encart à insérer sous le contenu
function rst_insert_after_content( $content ) {

    // Si nous sommes bien dans un article (et dans la
    boucle principale)
    if ( is_single() && is_main_query() ){

        // On place le code HTML de l'encart dans une
        variable
        $encart = '<div class="encart"><h4>Merci d\'avoir
        lu cet article.</h4></div>';

        // Si l'article est dans la catégorie d'ID 5
        if( in_category(5) ){
            // On remplace le contenu de la variable par
            autre chose
            $encart = '<div class="encart"><h4>Besoin
            d\'aide dans ce domaine ? <a href="http://monsite.com/
            contact/">Contactez-moi</a>.</h4></div>';
        }

        // On place le contenu de l'encart après celui de
        l'article
        $content = $content . $encart;
    }
}
```

```
    }  
    // On retourne le contenu intégral (article + encart)  
    return $content;  
}  
add_filter( 'the_content', 'rst_insert_after_content' );
```

La fonction est certes un peu plus longue mais le résultat est là. Vous pouvez modifier cela à volonté pour afficher ce que vous désirez avant et après le contenu (et cela sur les types de page de votre choix grâce aux marqueurs conditionnels).

Note : Veillez à bien utiliser des guillemets simples pour englober votre code ou texte, et à rendre ce caractère inopérant lorsque vous l'utilisez comme apostrophe (on dit que ce caractère est « échappé »). Pour ce faire, utilisez simplement un anti-slash devant, comme ceci \ '.

### 3. Créer un modèle de page personnalisé

Nous avons vu dans le chapitre consacré à WordPress qu'il est possible de créer des modèles de pages personnalisés.

Pour cet exemple, nous allons voir **comment créer une page de contact embarquant une carte Google Maps** (à la place de l'image à la une) dans le thème Twenty Fifteen.

Note : Le formulaire devra être intégré dans le contenu de la page grâce à une extension comme **Contact Form 7** ou équivalent.

Procédons étape par étape :

- Dupliquer le fichier page . php
- Renommer ce fichier en template-contact . php
- Ajouter l'en-tête nécessaire :

```
<?php
/* Template Name: Contact */
?>
```

- Supprimer les commentaires
- Dupliquer le fichier content - page . php en content - contact . php dans le thème enfant et le charger à partir de template-contact . php
- Obtenir le code d'intégration (balise `iframe`) d'une carte via **Google Maps** (il faut cliquer sur l'engrenage en bas à droite puis sur *Partager ou intégrer la carte*).
- Placer cet `iframe` dans une balise `div` dotée d'une classe `carte` (cela est important pour rendre la carte responsive) comme ceci :

```
1  <?php
2  /**
3   * The template used for displaying page content
4   *
5   * @package WordPress
6   * @subpackage Twenty_Fifteen
7   * @since Twenty_Fifteen 1.0
8   */
9  ?>
10
11 <article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
12
13     <div class="carte">
14         <iframe src="https://www.google.com/maps/embed?
15         pb=!1m14!1m12!1m3!1d42472.72329281884!2d4.0763055999999995!3d48.2923647999999994!2m3!1f0!2f0!3f0!3m2!1i1024!2i76
16         8!4f13.1!5e0!3m2!1sfr!2sfr!4v1436249235878" width="800" height="600" frameborder="0" style="border:0"
17         allowfullscreen></iframe>
18     </div>
19
20     <header class="entry-header">
21         <?php the_title( '<h1 class="entry-title">', '</h1>' ); ?>
22     </header><!-- .entry-header -->
23
24     <div class="entry-content">
25         <?php the_content(); ?>
26     </div>
```

- Rendre la carte responsive grâce au code CSS suivant trouvé grâce à une recherche sur Google :

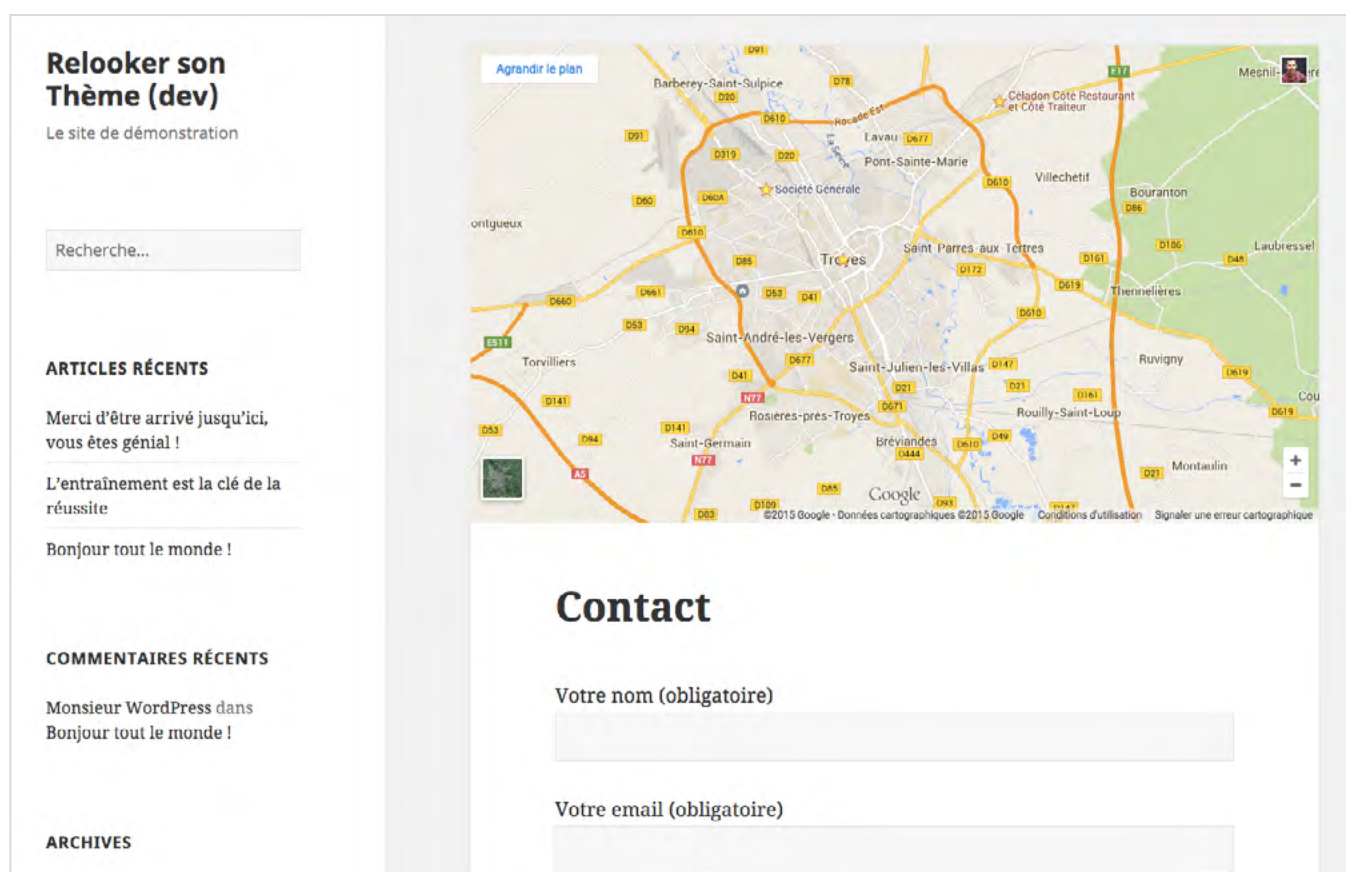
```
.carte {
    position: relative;
    width: 100%;
    padding-bottom: 56.25%; /* Format 16:9 */
    height: 0;
}
.carte iframe {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
}
```

- Optimiser l’affichage en insérant la règle `margin-bottom: 3em;` au sélecteur `.carte` et

```
.page-template.page-template-template-contact article.page
{
    padding-top: 0;
}
```

- Sauvegarder le tout et appliquer le nouveau modèle de page à la page Contact.

Toutes ces étapes ont permis d’arriver au résultat suivant :



Vous trouverez le code complet de ce thème enfant dans les ressources associées au chapitre 6 (fichier `theme-enfant-contact.zip`).

Note : Ce modèle de page n'est qu'un exemple, il est possible d'en créer de plus simples et de plus complexes.

## 4. Corriger les erreurs de traduction

Note : Pour apprendre à traduire un thème WordPress, vous pouvez [utiliser cet article](#).

Il arrive parfois que des thèmes WordPress soient mal traduits. Ils possèdent bien un fichier `.po` pour générer une traduction mais il peut manquer des termes.

Étant donné que ces termes ne sont pas présents dans le fichier .po, il faut les réintégrer.

Deux cas de figure peuvent se présenter.

## **1. La chaîne à traduire est comprise dans une fonction de traduction**

Ce premier cas est le plus simple à régler. L'erreur du fichier .po vient du fait que l'auteur du thème ne l'a pas actualisé suite à la mise à jour du thème.

Dans le menu, cliquez sur *Catalogue > Mettre à jour depuis les sources* et le tour sera joué.

Il ne restera plus qu'à traduire, sauvegarder et mettre les fichiers de traduction à jour dans le thème.

Attention : Dans la plupart des cas, il faudra placer les fichiers de traduction dans le thème parent. Veillez à garder une copie en cas de mise à jour du thème et éventuellement prévenir le créateur du thème que vous pouvez lui transmettre votre traduction à jour.

## **2. La chaîne à traduire n'est pas comprise dans une fonction de traduction**

Dans ce cas, l'auteur du thème est en cause car il n'a pas suivi les bonnes pratiques de WordPress, à savoir : rendre son thème traduisible.

Avant de faire le travail à sa place, contactez-le. S'il parle anglais, vous pouvez vous baser sur le modèle suivant s'il manque quelques termes :

Hello,

I'm using the [Nom du thème] WordPress theme and I noticed that you forgot to internationalize some terms. Here is the list of untranslatable terms:

- [Nom du fichier] - [numéro de ligne]
- [Nom du fichier] - [numéro de ligne]

Thanks for your help  
[Votre prénom]

Passez sa réponse [dans le traducteur Google](#) pour comprendre ce qu'il compte faire.

Soit il mettra le thème à jour prochainement (et vous avez de la chance), soit vous allez devoir insérer vous-même les fonctions de traduction dans le thème enfant.

Reportez-vous au chapitre 4 partie 3 pour voir comment fonctionnent les fonctions de traduction et à [cet article](#) pour aller plus loin.

## 5. Créer des shortcodes pour agrémenter son contenu

Si vous utilisez WordPress depuis un certain temps, vous avez dû découvrir ce que sont les shortcodes.

Certains thèmes WordPress en proposent par défaut mais c'est surtout le cas d'extensions.

Par exemple, Contact Form 7 fournit un shortcode pour insérer des formulaires. Il ressemble à ceci :

```
[contact-form-7 id="836" title="Contact"]
```

En étudiant la structure de ce shortcode, on constate qu'il se compose :

- **d'un nom** : `contact-form-7`
- **de deux attributs** : `id` et `title` possédant chacun une valeur

Le tout étant encadré par des crochets `[ ]`.

*Les shortcodes sont utiles pour insérer du contenu dans ses publications sans avoir à le saisir à chaque fois.*

Une fois qu'un contenu est placé dans un shortcode, on peut insérer son shortcode où l'on veut. C'est une sorte de raccourci en fait.

Si vous ajoutez régulièrement un message à la fin de vos articles, créer un shortcode peut être la solution pour vous éviter d'écrire la même chose à chaque fois.

## **Exemple de shortcode simple**

Sur WP Marmite, le shortcode `[abonnement_youtube]` a été créé pour inclure un bouton d'abonnement à la chaîne Youtube du blog.

Cela évite de reprendre systématiquement le code d'intégration à partir de Youtube.

Créer un shortcode n'est pas très compliqué, voici ce que cela donne pour le shortcode Youtube de WP Marmite (dans le fichier `functions.php`) :

```
function wpm_youtube_subscription(){
    return '<script src="https://apis.google.com/js/
platform.js"></script>
    <div class="g-ytsubscribe" data-channel="WPMarmite"
data-layout="full" data-count="default"></div>';
}
add_shortcode( 'abonnement_youtube', 'wpm_youtube_
subscription' );
```

On peut voir que le shortcode se compose d'une fonction (`wpm_youtube_subscription()`) où l'on indique ce qui devra être retourné par le shortcode (avec l'instruction `return`).

Attention : Les guillemets doivent être doubles `"` dans le code retourné car ceux de l'instruction `return` sont simples.

Ensuite, la fonction `add_shortcode()` est appelée pour déclarer le shortcode à WordPress. Le premier argument correspond au nom du shortcode (ce qui sera entre crochets) et le second au nom de la fonction associée à ce shortcode.

## Exemple de shortcode simple avec attributs

Imaginons que nous ayons besoin de personnaliser le shortcode précédent afin de changer rapidement la chaîne Youtube associée au bouton.

Au lieu de créer un shortcode à chaque fois, on pourrait donner un attribut au shortcode afin de spécifier l'identifiant de la chaîne.

L'idéal serait d'obtenir ceci :

```
[abonnement_youtube_a chaîne="WPMarmite"]
```

La fonction ressemblerait à cela :

```
function wpm_youtube_subscription($atts, $content = null)
{
    // Récupération de l'attribut "chaîne" dans un
    // tableau d'attributs, si il est absent sa valeur sera
    // "WPMarmite"
    $atts = shortcode_atts(array(
        'chaîne' => 'WPMarmite'
    ), $atts);

    // On retourne le code en incluant l'attribut
    // (attention aux guillemets)
    return '<script src="https://apis.google.com/js/
platform.js"></script>
<div class="g-ytsubscribe" data-channel="' .
$atts['chaîne'] . '" data-layout="full" data-
count="default"></div>';
}
// Déclaration du shortcode
add_shortcode( 'abonnement_youtube_a', 'wpm_youtube_
subscription' );
```

Grâce à ce shortcode, si vous voulez que vos lecteurs s'abonnent à la chaîne Youtube de Minute Papillon, vous devrez juste écrire :

```
[abonnement_youtube_a chaîne="languedepub2"]
```

languedepub2 étant l'identifiant de la chaîne Youtube de Minute Papillon (excellente chaîne d'ailleurs).

Nous pourrions aller beaucoup plus loin avec les shortcodes mais ce que nous avons vu est déjà un bon point de départ.

## 6. Personnaliser les métadonnées des articles

Sur la plupart des blogs, les articles sont accompagnés de ce que l'on appelle des métadonnées. C'est à dire des informations complémentaires comme :

- La date de publication (ou de mise à jour)
- L'auteur
- La ou les catégories de l'article
- Les étiquettes de l'article
- Le nombre de commentaires

Pour toutes sortes de raisons, on peut désirer modifier ces métadonnées car on ne les trouve pas pertinentes.

Par exemple, **si vous êtes seul à publier sur votre blog, le lien vers votre page auteur ne sert à rien**. Il faudrait mieux rediriger les internautes qui cliquent dessus sur la page *À Propos* ou sur votre compte Twitter ou Facebook.

Si votre thème WordPress ne vous permet pas de personnaliser cette zone comme vous l'entendez, il va falloir aller dans le code.

Pour cet exemple, nous allons utiliser le thème Twenty Fifteen. Vous allez devoir adapter ce qui va suivre à votre situation.

*La première chose à faire est de trouver quel fichier il faut dupliquer dans son thème enfant pour arriver à ses fins.*

Dans Twenty Fifteen, quand on ouvre `single.php` on voit que rien ne fait référence à des métadonnées mais on constate qu'un fichier `content.php` est appelé.

En l'ouvrant, on se rend compte qu'une fonction `twentyfifteen_entry_meta()` est appelée dans la balise footer de l'article. C'est intéressant car c'est précisément à cet endroit que les métadonnées sont affichées.

Il faut donc trouver où est déclarée cette fonction. Le premier endroit où chercher est le fichier `functions.php`. Malheureusement, elle ne se situe pas ici... Il faut donc continuer à chercher.

Après quelques recherches supplémentaires, on peut se rendre compte que le fichier `template-tags.php` situé dans le dossier `inc` contient notre fonction `twentyfifteen_entry_meta()`.

Étant donné qu'un test est fait pour savoir si cette fonction existe ou non (`if ( ! function_exists( 'twentyfifteen_entry_meta' ) ) :`), nous allons pouvoir la redéfinir sans problème dans le thème enfant.

Commençons par dupliquer cette fonction dans le fichier `functions.php` du thème enfant. En regardant ce qu'il y a dedans, le moins que l'on puisse dire est que cette fonction possède un code assez complexe.

Note : Pour rechercher rapidement une fonction parmi les fichiers d'un thème, vous pouvez utiliser simplement la fonction recherche de votre ordinateur. Placez-vous dans le dossier de votre thème et entrez le nom de la fonction recherchée précédée de `function` . Concrètement, si vous cherchez où est déclarée la fonction `hopla_boom()`, recherchez simplement `function hopla_boom`. Vous devriez rapidement trouver. Utilisez ensuite le raccourci CTRL + F (ou CMD + F sur Mac) pour trouver la fonction au sein du fichier précédemment trouvé.

## Étude de la fonction `twentyfifteen_entry_meta()`

Comme on ne se laisse pas impressionner par le code, on va le décomposer pour essayer de comprendre ce qu'il fait. Au début de la fonction, on trouve ce code :

```
if ( is_sticky() && is_home() && ! is_paged() ) {  
    printf( '<span class="sticky-post">%s</span>', __(  
'Featured', 'twentyfifteen' ) );  
}
```

Un test est fait (instruction `if`) avec plusieurs marqueurs conditionnels :

- `is_sticky()` : Est-ce que l'article est mis en avant ?
- `is_home()` : Est-ce que l'on est sur la page des articles ?
- `! is_paged()` : Est-ce que l'on n'est pas sur une page interne (page 2, 3, 4, etc)

Si ces conditions sont vérifiées, on peut afficher la balise `span` qui montrera la mise en avant de l'article (la fonction `printf` est similaire à l'instruction `echo`).

Cela n'a rien à voir avec l'auteur donc on laisse ce code intact et on continue notre étude.

Le test suivant concerne les formats d'articles, cela ne nous intéresse toujours pas. On poursuit.

Les classes et les fonctions semblent être liées aux dates. Aucune référence à l'auteur par contre, c'est le cas de la suite. On voit le code suivant :

```

if ( 'post' == get_post_type() ) {
    if ( is_singular() || is_multi_author() ) {
        printf( '<span class="byline"><span class="author
vcard"><span class="screen-reader-text">%1$s </span><a
class="url fn n" href="%2$s">%3$s</a></span></span>',
            _x( 'Author', 'Used before post author
name.', 'twentyfifteen' ),
            esc_url( get_author_posts_url( get_the_
author_meta( 'ID' ) ) ),
            get_the_author()
        );
    }
    /* Autres instructions concernant les catégories
et et les étiquettes */
}

```

Pour expliquer ce code, il faut comprendre le fonctionnement de la fonction `printf()`.

Comme je vous l'ai dit, elle est similaire à l'instruction `echo` car elle permet d'afficher quelque chose. La différence est qu'elle permet de placer des paramètres à certains endroits.

Ces endroits sont matérialisés par `%1$s`, `%2$s` et `%3$s`. La fonction `printf()` y placera respectivement le résultat des arguments 1, 2 et 3 suivant la chaîne de caractères placée en premier.

Dans notre cas nous avons :

- `%1$s`: `_x( 'Author', 'Used before post author name.', 'twentyfifteen' )` (fonction de traduction)
- `%2$s`: `esc_url( get_author_posts_url( get_the_author_meta( 'ID' ) ) )` (affichage du lien de la page auteur)

- %3\$s : `get_the_author ( )` (affichage du nom de l'auteur)

Certes, cela aurait pu être écrit autrement mais il faut prendre le thème tel qu'il est.

Pour remplacer le lien associé à l'auteur il suffit de mettre le lien de son choix au niveau l'argument n°2.

Attention : Cette modification concerne les blogs n'ayant qu'un seul auteur, sinon tous les auteurs auront le même lien. Pour aller plus loin, vous pouvez toujours regarder du côté de la fonction qui permet de retourner dynamiquement la valeur de `user_url` : `get_the_author_meta( 'user_url' )`.

Pour aller plus loin, il aurait été possible de jouer avec les marqueurs conditionnels pour afficher le nom de l'auteur dans certains cas.

De même, en supprimant des morceaux de code de cette fonction, on pourrait masquer des métadonnées non désirées (les étiquettes par exemple).

*L'important est de chercher à comprendre à quoi correspond un code, une fonction, etc pour arriver à obtenir le rendu souhaité.*

## 7. Ajouter de nouvelles tailles d'images

Un des ajouts les plus courants que l'on peut faire au sein d'un thème WordPress est l'ajout d'une ou plusieurs nouvelles tailles d'images.

Il faut savoir que lorsque vous envoyez une image sur votre site, plusieurs autres images sont créées afin d'obtenir des miniatures de différentes tailles.

Le thème WordPress que vous utilisez en définit certainement pour bénéficier d'images de tailles spécifiques à certains endroits.

Voici comment procéder pour ajouter une nouvelle taille d'image à votre site :

- Ouvrez le fichier `functions.php` de votre thème enfant
- Entrez le code suivant :

```
function rst_new_image_sizes() {  
    // Ajoute une nouvelle taille nommée "miniature-blog"  
    de 300x300 pixels qui pourra être rognée (paramètre true  
    à la fin)  
    add_image_size( 'miniature-blog', 300, 300, true );  
}  
add_action( 'after_setup_theme', 'rst_new_image_sizes' );
```

- Utilisez le code suivant pour afficher les images de cette taille sur votre site :

```
if ( has_post_thumbnail() ) {  
    the_post_thumbnail( 'miniature-blog' );  
}
```

Pour que cela fonctionne, il faut bien évidemment que des images soient envoyées APRÈS que la fonction `add_image_size()` ait ajouté une nouvelle taille d'image.

Afin de générer des images sous ce format à partir des images précédemment envoyées, vous pouvez utiliser l'extension **Regenerate Thumbnails** présentée dans le chapitre 5.

## Redimensionner automatiquement des images pour son contenu

Lorsque l'on ajoute des images à un article ou à une page, elles sont rarement de la bonne taille. Pour éviter de redimensionner manuellement vos images, vous pouvez utiliser ce morceau de code :

```
function rst_article_width() {
    // Définit une nouvelle taille d'image de 600 pixels
    de large (la hauteur sera proportionnelle et aucun
    recadrage ne sera effectué)
    add_image_size( 'largeur-article', 600 );
}
add_action( 'after_setup_theme', 'rst_article_width' );

// Cette fonction permet d'ajouter la taille d'image
fraîchement définie dans la liste des tailles d'images à
insérer dans son contenu
function rst_image_sizes( $sizes ) {
    $addsizes = array(
        'largeur-article' => 'Format article'
    );
    $newsizes = array_merge($sizes, $addsizes);
    return $newsizes;
}
add_filter( 'image_size_names_choose', 'rst_image_sizes'
);
```

Il faut bien sûr l'adapter et le placer dans le fichier `functions.php` de votre thème enfant.

Cela vous permettra d'intégrer vos images directement à partir du menu déroulant *Tailles* :



Vous pourrez dire adieu aux redimensionnements des images avant de les envoyer :)

## Récapitulons

Ces 7 exemples de personnalisation de thème WordPress ont pour but de vous apporter des clés supplémentaires afin d'aller plus loin dans vos relookings.

Vous pouvez bien entendu les reprendre, les adapter et les utiliser sur les thèmes de votre choix.

Si jamais vous êtes confronté à un problème, les moteurs de recherche sont là pour vous aider à trouver des solutions. La communauté WordPress sera également ravie de vous aider.

**RELOOKER SON THÈME**

# **CONCLUSION**

**VERS L'INFINI  
ET AU DELÀ**

---

**Félicitations,  
derniers conseils,  
glossaire et remerciements**

# Conclusion

Et voilà, vous arrivez au terme de Relooker son Thème. Félicitations !

Le but de ce guide était de vous prendre par la main afin de vous de transmettre les bases du code pour personnaliser n'importe quel thème WordPress.

Si vous avez bien lu les différents chapitres et fait les exercices fournis avec le guide, vos compétences ont incontestablement dû évoluer.

*Au minimum, vous devriez être capable d'ajouter vos propres styles CSS à un thème WordPress. Au mieux, vous avez commencé à jouer avec les fonctions de WordPress et ses fichiers de template.*

Vous devriez également être mieux organisé. Relooker son Thème vous a appris à élaborer votre projet de personnalisation de thème et à vous servir des bons outils pour travailler convenablement.

Pensez à parcourir le guide régulièrement. Vous y trouverez des informations que vous n'aurez pas forcément bien comprises lors de votre première lecture.

N'essayez pas de tout réussir du premier coup. Entraînez-vous régulièrement, testez de nouvelles choses et continuez de faire avancer vos projets. Vous verrez qu'une connaissance (même basique) du code fera de vous une personne plus confiante.

*Quand vous aurez terminé votre premier relooking de thème, j'aimerais beaucoup découvrir le travail que vous aurez effectué (de quoi vous êtes parti et les résultats que vous avez obtenus).*

Pour cela, **rendez-vous sur cette page** et complétez le formulaire. J'ai vraiment hâte de découvrir le fruit de votre travail.

Merci beaucoup d'avoir lu Relooker son Thème et bonne chance pour la suite.

Alex

# 7 Bonnes adresses pour aller plus loin

Relooker son Thème n'a pas la prétention d'être un guide exhaustif sur la personnalisation de thèmes WordPress. C'est un sujet tellement vaste !

Pour vous permettre d'approfondir votre apprentissage, vous pouvez consulter les sites suivants :

## Sites et blogs

- **Codex WordPress** : Documentation officielle de WordPress
- **Forum de WordPress Francophone** : Trouvez de l'aide auprès de la communauté francophone
- **WP Channel** : Blog de tutoriels WordPress d'Aurélien Denis
- **SeoMix** : Blog orienté WordPress et référencement de Daniel Roch
- **Boiteaweb** : Blog WordPress et sécurité de Julio Potier
- **GeekPress** : Blog WordPress de Jonathan Buttigieg
- **GenerateWP** (en anglais) : Très utile pour générer du code pour créer des menus, des zones de widgets et beaucoup d'autres choses.

Sans oublier **WP Marmite** pour retrouver régulièrement des tutoriels et articles détaillés sur WordPress.

# Glossaire

Pour vous aider davantage dans la compréhension du vocabulaire associé aux thèmes WordPress, ce lexique regroupe les termes les plus courants.

## A

**About page** : Page à propos

**Accent color** : Couleur principale grâce à laquelle vous pourrez habiller un thème

**AJAX animation** : Animation réalisée à l'aide du langage Javascript

## C

**Child theme** : Thème enfant. Sorte de sous-thème permettant de faire des modifications en toute sécurité (les mises à jour du thème principal n'effaceront pas vos modifications)

**Color picker** : Sélecteur de couleur (outil permettant de choisir une couleur)

**CSS** : Acronyme de *Cascading Style Sheets* soit Feuilles de style en cascade. Ce type de fichier embarque les styles à appliquer aux pages web

**Custom post type** : Type de contenu personnalisé (CPT en est l'acronyme). Cela correspond à des types de contenus autres que les pages et articles

**Custom Widgets** : Widgets créés sur mesure pour un thème

## D

**Demo content** : Contenu de démonstration pour recréer la démo facilement sur son site

**Drag and drop builder** : Constructeur de page en glisser-déposer

## F

**Font Awesome** : Ensemble d'icônes pouvant être insérées à plusieurs endroits du thème (cela varie selon le thème)

**Footer** : Pied de page du thème. Cette zone comprend généralement une zone de widgets, une ligne de texte de crédits et parfois un menu

## G

**Google fonts** : Ensemble des polices d'écriture fournies par Google. Certains thèmes les intègrent par défaut

## H

**Header** : En-tête du thème. Cette zone contient généralement le logo et le menu principal

**Homepage** : Page d'accueil

**HTML** : Acronyme de *Hypertext Markup Language*. Ce langage web sert à décrire la structure et le contenu des pages web

## I

**Infinite Scroll** : Défilement infini. Les articles se chargeront automatiquement lorsque vous arriverez en bas de page

## L

**Landing page** : Page d'atterrissage. Ce type de page est destiné à pousser le visiteur à accomplir une action précise (inscription, achat, etc.)

## M

**Masonry layout** : Mise en page sous forme de blocs qui s'agencent automatiquement quel que soit leur taille

**Mega menu** : Menus déroulants géants. On peut y inclure beaucoup plus de choses que dans de simples menus déroulants

## P

**Parallax** : Intégration des images de façon à créer une animation dynamique lors du défilement

**PHP** : Acronyme de *Hypertext Preprocessor*. Il s'agit d'un langage de programmation web. Le PHP est notamment utilisé pour faire fonctionner WordPress

**Portfolio** : Thème proposant un portfolio (pour présenter des projets)

**Post format support** : Plusieurs formats d'articles sont supportés par le thème (vidéo, audio, galerie, citation, lien, etc)

**Pricing Tables** : Tableaux de prix

**PSD** : Fichier(s) graphique(s) à l'origine du design du thème (avant qu'il soit transformé en thème WordPress) conçu avec le logiciel Photoshop

## R

**Responsive** : Thème WordPress qui s'adapte à tous les types d'écrans (ordinateur de bureau, ordinateur portable, tablette tactile, téléphones, etc.)

**Retina** : Thème optimisé pour les écrans de haute résolution (iPads, iPhones, certains ordinateurs de chez Apple et d'autres constructeurs)

**RTL stylesheet** : Feuille de style optimisée pour les langages se lisant de droite à gauche comme l'arabe ou l'hébreu

## S

**Search Engine Optimized (SEO)** : Optimisé pour le référencement (ne croyez pas l'auteur sur parole)

**Shortcode** : Morceau de code que l'on insère dans le contenu (article, page, etc) pour insérer une mise en forme ou un élément précis

**Shortcode generator** : Générateur de shortcodes pour vous aider à créer des shortcodes

**Sidebar** : Barre latérale dans laquelle on place des widgets. Vous trouverez plus d'information sur les sidebars [dans cet article](#)

**Sidebar manager** : Gestionnaire de barres latérales. Permet de créer et supprimer des barres latérales à afficher à des endroits spécifiques

**Slug** : Identifiant texte d'un contenu ou d'une taxonomie. C'est souvent le slug que l'on retrouve dans l'adresse des pages

**Smooth scroll** : Défilement fluide des pages

**Snippet** : Morceau de code

**Slider** : Diaporama animé

**Slideshow** : Voir Slider

**Sticky menu** : Menu qui reste à l'écran malgré le défilement de la page

## T

**Taxonomie** : Moyen de classer des publications. Par défaut, WordPress possède deux taxonomies : les catégories et les étiquettes

**Template** : Fichier d'un thème WordPress servant à afficher une page (ou une partie d'une page)

**Translation ready** : Thème possédant les fichiers de traduction (.po) demandés par un logiciel de traduction comme PoEdit. Découvrez comment traduire un thème [dans cet article](#)

## W

**Wide & Boxed Layout** : Possibilité de mettre le thème en mode plein écran ou cadré

**Widget** : Encart de contenu que l'on retrouve généralement dans les barres latérales et les pieds de page. On peut cependant les utiliser pour structurer des pages dans certains thèmes. Il faut aller dans *Apparence* > *Widgets* pour les agencer

**Widget ready** : Thème proposant une ou plusieurs zones widgetisées (barre latérale, pied de page ou autre)

**Widgetized homepage** : Page d'accueil widgetisée (qui utilise les widgets)

**WooCommerce ready** : Thème optimisé pour le plugin WooCommerce

**WPML ready** : Thème compatible avec le plugin multilingue WPML. Découvrez les meilleurs plugins multilingues [dans cet article](#)

# Remerciements

Rédiger Relooker son Thème fut un travail colossal. J'avais toutefois à coeur d'aider un maximum de personnes à gagner en autonomie grâce à ce guide.

Je tiens à remercier chaleureusement les personnes qui m'ont fait confiance en se procurant la version avant-première de Relooker son Thème. Je n'aurais pas pu écrire le guide sans eux :

## **Merci aux clients de l'offre avant-première**

Lise L.	Nadia Robert	Christian G.
Pascal D.	Levasseur B.	Valéry G.
Sébastien L.	Samson Bourdarias	Pascal B.
Andre Cochois	Guillaume D.	Mael R.
Denis A.	Renaud L.	Xavier R.
Jean-Louis E.	Norbert B.	Sylvie D.
Patrick G.	Marc M.	Jean-Joseph L.
Nicolas Menard	Alain T.	Jany Loiseau
Gerard M.	Claude L.	Olivier B.
Denis F.	Sylvain P.	Pascal L.
Jean-Pierre E.	Gabriel C.	Louis De C.
Julie V.	Benoit Tostain	Pierre G.
Guy De C.	Nora B.	Eric C.
Eric Verschueren	Francis A.	Stéphane B.
Olivier Beck	Emmanuel N.	Remi B.
Bernard K.	Philippe M.	Frédéric D.
Christophe De Z.	Charly D. P.	Christian M.

Mélanie L.	Jean-Paul R.	Michel C.
Severine Melchiorre	Antoine Le G.	Guillaume V.
Claire Janssens	Maryse W.	Sebastien S.
Jérôme G.	Jean-Charles L.	Sacha T.
Jean-Claude C.	Bernard S.	André C.
Clomilla M.	Michel B.	Pierre R.
Jean-Alain F.	Jean-Louis A.	Louis B.
Jean-Francois R.	Catherine D.	Anthony D.
Nicole M.	Florian H.	Céline V.
Joël G.	Lydia R.	Johan C.
Yves L.	Béatrice D.	Marie-Christine H.
Jean-Claude M.	Vincent De B.	Roselyne B.
Xavier Bascot	Georges Beaulieu	Nicolas L.
Christophe A.	Richard M.	Christophe B.
Denis O.	Denis L.	Anouc A.
Nathalie Gatien David	Pauline L.	Marietta N.
Charles B.	Marc D.	Patrick B.
Nicolas W.	Philippe N.	Maguy D.
Jean-Pierre Cousserans	Corinne C.	Manuel P.
Charles T.	Farah J.	Sylvain Jubé
Alexandre Sevrin	Jean-Luc P.	Stéphane Chevrier
Ahd Razik C.	Michel J.	Léo M.
Jean-Michel B.	Isabelle D.	Robert Theyssens
Deborah M.	Lucien S.	Michel D.
Pierre C.	Francis L.	Mylene M.
Renaud B.	Sophie P.	
Olivier T.	Régis H.	

Merci également aux personnes ayant participé à la relecture de Relooker son Thème :

- **Christophe Benoît**
- **Anne-Catherine Guervel de CyberEntraide**
- **Jean-Rémi Larcelet**
- **Daniel Roch de SEOMix**
- **Lou de Notuxedo**
- **Geoffrey Crofte de Creative Juiz**
- **Aude Youcom**
- Samuel Van Hoecke

2015 - Alexandre Bortolotti - Tous droits réservés

Relooker son Thème est un guide proposé par **WP Marmite**, le blog qui vous aide à tirer le meilleur de WordPress.

Merci de ne pas diffuser ni reproduire ce guide sans permission. Si vous avez des questions sur le contenu, contactez-moi à l'adresse suivante : **[alex@wpmarmite.com](mailto:alex@wpmarmite.com)**. J'essaie de répondre à tous les emails que je reçois.