

Apprendre le CSS

CSS est un langage complexe et puissant. Il nous permet d'ajouter de la mise en forme à la conception de nos pages et il nous permet de partager les styles d'éléments entre toutes les pages. Avant de pouvoir découvrir toutes ses fonctions, il y a quelques aspects du langage que nous devons comprendre.

D'abord, il est crucial de comprendre exactement comment les styles sont rendus. Précisément, nous avons besoin de savoir comment les différents types de sélecteurs fonctionnent et comment l'ordre de ces sélecteurs peut affecter la façon dont nos styles sont rendus. Nous voulons également comprendre quelques valeurs de propriétés courantes qui apparaissent souvent en CSS, en particulier celles qui traitent de la couleur et de la taille.

La Cascade

Nous allons commencer par revoir comment les styles sont rendus en regardant ce qui est connu, comme la cascade, en étudiant quelques exemples. Dans CSS, tous les styles cascadedu haut de la feuille de style à la fin de celle-ci, ce qui permet d'avoir des styles différents, en ajoutant ou modifiant des propriétés et leur valeurs tout au long de la feuille de style.

Par exemple, si nous définissons que tous éléments de paragraphe au sommet de notre feuille de style ont une couleur de fond `orange` et une taille de police à 24 pixels, puis plus bas dans notre feuille de style, nous sélectionnons tous éléments de paragraphe de nouveau et cette fois redéfinissons leur couleur d'arrière plan à `green`, que va t'il se passer ?

```
1 p{
2   background: orange;
3   font-size: 24px;
4 }
5
6 p {
7   background : green;
8 }
```

Parce que le sélecteur de paragraphe qui définit la couleur d'arrière plan `green` vient après le sélecteur de paragraphe qui définit la couleur d'arrière plan à `orange`, il aura la priorité dans la cascade. Tous les paragraphes apparaîtront avec un fond `green`. La taille de police restera 24 pixels parce que le second sélecteur de paragraphe n'a pas été redéfini à une nouvelle taille de police.

Propriétés de cascade

La cascade fonctionne également avec les propriétés à l'intérieur des sélecteurs individuels. Encore une fois, par exemple, disons nous choisissons tous les éléments de paragraphe et mis leur couleur d'arrière plan à `orange`. Ensuite directement sous

la propriété de fond `orange`, nous ajoutons une autre propriété de couleur de fond avec la valeur `green` comme on le voit ci-dessous.

```
1 p {  
2   background: orange;  
3   background: green;  
4 }
```

Parce que la déclaration de couleur de fond `green` vient après la déclaration de couleur de fond `orange`, le navigateur va passer outre le fond `orange` et comme auparavant, nos paragraphes apparaissent avec un fond `green`.

Tous les styles seront toujours appliqués en cascade à partir du haut de notre feuille de style jusqu'au bout de notre feuille de style. Il y a cependant des moments où la cascade ne s'applique pas si bien. Cela va se produire lorsque différents types de sélecteurs sont utilisés et la spécificité de ces sélecteurs vont briser la cascade. Jetons un coup d'oeil.

Le calcul de spécificité

Chaque sélecteur CSS a un poids de spécificité. Le poids de spécificité d'un sélecteur, ainsi que sa mise place dans la cascade, identifie la façon dont ses styles seront rendus :

Dans notre première leçon nous avons parlé de trois différents types de sélecteurs, par type, class ou ID. Chacun de ces sélecteurs a un poids de spécificité différente.

- Le sélecteur de type a le poids de spécificité le plus bas et est titulaire d'une valeur de point de `0-0-1`.
- Le sélecteur de classe a un poids de spécificité moyenne et contient une valeur de point de `0-1-0`.
- Enfin, le sélecteur ID a un poids le plus élevé de spécificité et contient une valeur de point de `1-0-0`.

Comme nous pouvons le voir les points de spécificité sont calculés en utilisant trois colonnes. La première colonne compte le sélecteurs d'ID, la deuxième colonne compte les sélecteurs de class, et la troisième colonne compte le sélecteurs de type.

Ce qui est important de noter ici est que le sélecteur d'ID a un poids de spécificité plus élevé que le sélecteur de classe, et le sélecteur de classe a une spécificité plus élevée poids que le sélecteur de type.

Les points de spécificité

Les points de spécificité sont intentionnellement représentés avec un trait d'union, leurs valeurs ne sont pas calculées en base 10. Le sélecteurs de classe n'a pas pas une valeur de point de 10, et sélecteurs d'ID n'a pas une valeur de point de 100. au

lieu , ces points doivent être lus respectivement comme 0-1-0 et 1-0-0. Nous regarderons bientôt, quand nous combinerons des sélecteurs, pourquoi ces valeurs de point sont ainsi libellées avec des traits d'union.

Plus haut sera le poids de spécificité d'un sélecteur, plus on donnera la priorité à ce sélecteur quand un conflit de style arrive. Par exemple, si un élément de paragraphe est choisi utilisant un sélecteur de type en un endroit et un sélecteur d'ID dans un autre, le sélecteur d'ID aura la priorité sur le sélecteur de type indépendamment d'où le sélecteur d'ID apparaît dans la cascade.

HTML

```
1 <p id="food">...</p>
```

CSS

```
1 #food{
2     background: green;
3 }
4
5 p {
6     background: orange;
7 }
```

Ici nous avons un élément de paragraphe avec un `id` qui a comme valeur d'attribut *food*. Au sein de notre CSS, ce paragraphe est sélectionné par deux différents types de sélecteurs: un sélecteur de type et un sélecteur d'ID. Bien que le sélecteur de type vienne après le sélecteur d'ID dans la cascade, le sélecteur d'ID est prioritaire sur le sélecteur de type car il a un poids de spécificité plus élevée et par conséquent le paragraphe apparaîtra avec un fond *green*.

Il est particulièrement important de se rappeler des poids de spécificité des différents types de sélecteurs. Parfois les styles peuvent ne pas apparaître sur les éléments comme vous le prévoyez... Quand c'est ainsi il y'a de bonnes chances que ce soient les poids des sélecteurs qui enfreignent la cascade, et qui font que nos styles ne vont pas apparaître correctement.

Comprendre comment la cascade et la spécificité fonctionnent est un gros morceau ! Pour l'instant, regardons de quelle façon on peut être un peu plus précis avec nos sélecteurs en les combinant. Gardez à l'esprit que si nous combinons sélecteurs, nous allons également changer leur spécificité.

En pratique

Actuellement, notre site Web n'a presque pas de style. Commençons par styler quelques éléments du fichier `index.html`.

- Faire en sorte que les éléments de titre aient une taille qui s'échelonne de 10px en 10px sachant que le titre le plus petit aura une taille de 15px

```
1 h1{ font-size: 65px }
2 h2{ font-size: 55px }
3 h3{ font-size: 45px }
4 h4{ font-size: 35px }
5 h5{ font-size: 25px }
6 h6{ font-size: 15px }
```

- Tous les éléments de type ancre doivent être de couleur verte.

```
1 a{
2   color: green;
3 }
```

- On attribuera un identifiant "carte" à l'élément d'ancre de la section "cartes". Cet élément doit être de couleur orange.

```
1 #carte{
2   color: orange;
3 }
```

La combinaison des sélecteurs

Jusqu'à présent nous avons regardé comment utiliser différents types de sélecteurs individuellement, mais nous avons aussi besoin de savoir comment utiliser ces sélecteurs ensemble. En combinant les sélecteurs nous pouvons être plus précis sur quel élément ou groupe d'éléments nous aimerions sélectionner.

Par exemple, disons que nous voulons sélectionner tous les éléments de paragraphe qui résident dans un élément avec une valeur d'attribut de classe de `hotdog` et mettre leur couleur d'arrière plan à `brown`. Toutefois, si un de ces paragraphes arrive à avoir la valeur d'attribut de classe de `moutarde`, nous voulons définir sa couleur d'arrière plan à `jaune`. Notre HTML et CSS ressembleront à ce qui suit:

HTML

```
1 <div class="hotdog">
2   <p>...</p>
3   <p>...</p>
4   <p class="moutarde">...</p>
5 </div>
```

CSS

```
1 .hotdog p {
2   background: brown;
3 }
4 .hotdog p.moutarde {
5   background: yellow;
6 }
```

Lorsque les sélecteurs sont combinés Ils doivent être lus de droite à gauche. Le sélecteur le plus à droite, juste avant l’accolade d’ouverture, est connu comme le *sélecteur clé*. Le sélecteur clé identifie exactement à quels éléments les styles seront appliqués. N'importe quel sélecteur vers la gauche de la sélection servira à préqualifier le ou les sélecteurs.

Le premier sélecteur combiné ci dessus, `p.hotdog` inclut deux sélecteurs: un sélecteur de classe et un sélecteur de type. Ces deux sélecteurs sont séparés par un espace. Le sélecteur clé est le sélecteur de type qui va cibler les éléments de paragraphe. Et parce que ce sélecteur de type est préqualifié avec le sélecteur de classe `hotdog`, le sélecteur combiné complet ne sélectionnera que les éléments de paragraphe qui résident dans un élément avec une valeur d'attribut de classe `hotdog`.

Le second sélecteur ci dessus, `.hotdog p.moutarde`, comprend trois sélecteurs : deux sélecteurs de classe et un sélecteur de type. La seule différence entre le deuxième sélecteur et le premier sélecteur est l'addition du sélecteur de classe de `moutarde` à l'extrémité du sélecteur de type, après le point. Parce que le nouveau sélecteur de classe, `moutarde`, se situe sur le chemin le plus vers la droite du sélecteur combiné, il est le sélecteur clé, et tous les sélecteurs individuels qui lui sont soumis sont les “prequalifier” en anglais dans le texte.

L’espaces dans les sélecteurs

Dans le sélecteur combiné précédente, `.hotdog p.moutarde`, il y a un espace entre le sélecteur de classe `hotdog` et le sélecteur de type de paragraphe mais pas entre le sélecteur de type de paragraphe et le sélecteur de classe `moutarde`. L'utilisation ou l'omission des espaces fait une grande différence dans les sélecteurs.

Comme il n'y a pas d'espace entre le sélecteur de type de paragraphe et le sélecteur de classe `moutarde` la sélection sera uniquement faite sur les éléments de paragraphe qui ont la classe `moutarde`. Si le sélecteur de type de paragraphe a été retiré, et le sélecteur de classe `moutarde` avait des espaces sur les deux côtés, cela ciblerait les éléments avec la classe `moutarde` mais pas seulement les paragraphes.

La bonne pratique est de ne pas préfixer un sélecteur de classe avec un sélecteur de type. En général nous voulons sélectionner un élément quelconque d'une classe donnée, pas seulement un type d'élément. En suivant cette bonne pratique, notre nouveau sélecteur combiné va donc s'écrire `.hotdog .moutarde`. La lecture du sélecteur combiné de droite à gauche, fait que l'on va viser les paragraphes avec une valeur d'attribut de classe `moutarde` qui résident dans un élément avec la valeur d'attribut de classe `.hotdog`.

Différents types de sélecteurs peuvent être combinés pour cibler un élément donné sur une page. Alors que nous continuons d'écrire différents sélecteurs combinés, nous allons voir toute la puissance apportée par cette technique. Avant de faire cela, cependant, nous allons jeter un oeil à la façon dont la combinaison des sélecteurs va modifier la spécificité et le poids d'un sélecteur.

Spécificité dans les sélecteurs combinés

Lorsque les sélecteurs sont combinés, donc les poids de spécificité des sélecteurs individuels le sont aussi. Ces poids de spécificité combinées peuvent être calculés en comptant chaque type de sélection différente dans un sélecteur combiné.

Si l'on regarde nos sélecteurs combinés précédent, le premier, `.hotdog p`, avait un sélecteur de classe et un sélecteur de type. Sachant que la valeur du point d'un sélecteur de classe est 0-1-0 et la valeur du point d'un sélecteur de type est 0-0-1, la valeur totale du point combinée est de 0-1-1, trouvé en additionnant chaque type de sélecteur.

Le second sélecteur `p.moutarde .hotdog`, avait deux sélecteurs de classe et un sélecteur de type. Ensemble, le sélecteur a une valeur de spécificité de 0-2-1. Le 0 dans la première colonne est pour zéro sélecteurs d'ID, les 2 dans la deuxième colonne est pour deux sélecteurs de classe, et le 1 dans la dernière colonne est pour un sélecteur de type. En comparant les deux sélecteurs, le second sélecteur, avec ses deux classes, a une valeur nettement plus élevée de poids de spécificité et en tant que telle elle l'emporte dans la cascade. Si nous devions retourner l'ordre de ces sélecteurs au sein de notre feuille de style, plaçant le sélecteur de pondération supérieure au dessus du sélecteur pondération inférieure comme indiqué ci-dessous, l'apparence de leurs styles ne serait pas affecté raison du poids de spécificité de chaque sélecteur.

```
1 .hotdog p.moutarde{
2   background: yellow;
3 }
4 .hotdog p{
5   background: brown;
6 }
```

En général nous voulons toujours garder un oeil sur les poids de spécificité de nos sélecteurs. Plus nos poids de spécificité augmenteront, plus il sera probable que notre cascade sera brisée.

Superposition de styles avec classes multiples

Une façon de garder les poids de spécificité de nos sélecteurs bas est d'être aussi modulaire que possible, de partager les styles similaires entre éléments. Et une façon d'être aussi modulaire que possible est de faire des superpositions de styles différents en utilisant plusieurs classes sur les éléments.

Les éléments au sein du HTML peuvent avoir comme valeur d'attribut class plus d'une classe aussi longtemps que chaque valeur sera séparée par un espace. Avec cela, nous pouvons placer certains styles sur tous les éléments d'une sorte tout en plaçant d'autres styles que sur des éléments spécifiques de ce genre.

Nous pouvons lier les styles que nous voulons ré-utiliser grâce à une classe et ajouter une couche de styles supplémentaires avec une autre classe.

Par exemple, disons que nous voulons que tous nos boutons aient une taille de police de 16 pixels, mais nous voulons que la couleur de fond de nos boutons de fond puisse varier en fonction de l'endroit où ces boutons seront utilisés. Nous pouvons créer quelques classes et les ajouter sur un élément nécessaire pour appliquer les styles souhaités.

HTML

```
1 <a class="btn btn-danger">...</a>
2 <a class="btn btn-success">...</a>
```

CSS

```
1 .btn{
2   font-size: 16px;
3 }
4 .btn-danger{
5   background: red;
6 }
7 .btn-success{
8   background: green;
9 }
```

Ici vous pouvez voir deux éléments d'ancrage avec de multiples valeurs d'attribut de classe. La première classe, `btn`, est utilisée pour appliquer une taille de police de 16 pixels à chacun des éléments. Ensuite, le premier élément d'ancrage utilise une classe supplémentaire de `btn-danger` pour appliquer la couleur d'arrière plan `red` tandis que le deuxième élément d'ancrage utilise une classe supplémentaire de `btn-success` pour appliquer la couleur d'arrière plan `green`. Nos styles ici sont propres et modulaire.

Avec l'utilisation de plusieurs classes, nous pouvons superposer autant de styles que nous souhaitons sur nos éléments tout en gardant notre code simple et nos poids de spécificité faible. Tout comme la compréhension de la cascade et le calcul de spécificité, la superposition est une pratique qui va prendre du temps à être maîtrisée entièrement, mais cela ira de mieux en mieux au fur et à mesure de chaque leçon.

En pratique

- Tous les éléments ancre du menu doivent être de couleur grise.

```
1 nav a{  
2   color: gray;  
3 }
```

- On attribuera une classe "btn" à tous les éléments d'ancres des sections "horaires" et "cartes". Ces éléments avec la classe "btn" doivent être de couleur rouge.

```
1 .btn{  
2   color: red;  
3 }
```

- On attribuera à tous les éléments de titre 4 contenus dans une classe "btn" la couleur bleu turquoise -aqua-.

```
1 .btn h4{  
2   color: aqua;  
3 }
```

- On attribuera à l'élément de classe "btn" de la section "programme" une autre classe "btn-primary". Cette classe ajoutera un fond de couleur marron.

```
1 .btn-primary{  
2   background-color: maroon;  
3 }
```

Valeurs de propriétés CSS

Nous avons déjà utilisé une poignée de valeurs de propriété CSS commune, comme les valeurs de couleur avec les mots clés `red` et `green`. Nous allons prendre le temps maintenant de revenir sur certaines des valeurs de propriété précédemment utilisées, ainsi que d'explorer quelques unes des valeurs de propriété les plus courantes que nous allons bientôt utiliser, plus précisément, nous allons examiner les valeurs des propriétés qui se rapportent aux couleurs et aux mesures de longueur.

Les couleurs

















Toutes les valeurs de couleur dans CSS sont définies sur un espace de couleur sRGB (standard rouge, vert et bleu). Ces couleurs sont formées en mélangeant les canaux

rouge (R), vert (G) et bleu (B) ensemble à la façon dont les téléviseurs et les moniteurs génèrent toutes les différentes couleurs qu'ils affichent. En mélangeant différents niveaux de rouge, vert et bleu, nous pouvons créer des millions de couleurs et trouver presque toutes les couleurs que nous voulons. Actuellement il y a quatre manières principales pour représenter les couleurs sRGB au sein du CSS: *les mots clés*, *la notation hexadécimale*, *le RGB* et *les valeurs HSL*.

Les couleurs mots clés

Les valeurs des couleurs par mots clés sont les noms (comme le *red*, *green* ou *blue*) qui sont liées à une couleur donnée. Ces noms des mots clés et leurs couleurs correspondantes sont déterminées par la spécification CSS. Presque toutes les couleurs les plus courantes, ainsi que quelques bizarreries, ont leurs noms de mots clés.

Une liste complète de ces noms des mots clés se trouvent dans la [spécification CSS](#).

Color	Name	Hex Values	RGB Values	HSL Values
	black	#000000	rgb(0, 0, 0)	hsl(0, 0%, 0%)
	silver	#c0c0c0	rgb(192, 192, 192)	hsl(0, 0%, 75%)
	gray	#808080	rgb(128, 128, 128)	hsl(0, 0%, 50%)
	white	#ffffff	rgb(255, 255, 255)	hsl(0, 100%, 100%)
	maroon	#800000	rgb(128, 0, 0)	hsl(0, 100%, 25%)
	red	#ff0000	rgb(255, 0, 0)	hsl(0, 100%, 50%)
	purple	#800080	rgb(128, 0, 128)	hsl(300, 100%, 25%)
	fuchsia	#ff00ff	rgb(255, 0, 255)	hsl(300, 100%, 50%)
	green	#008000	rgb(0, 128, 0)	hsl(120, 100%, 25%)
	olive	#808000	rgb(0, 255, 0)	hsl(120, 100%, 50%)
	lime	#00ff00	rgb(128, 128, 0)	hsl(60, 100%, 25%)
	yellow	#ffff00	rgb(255, 255, 0)	hsl(60, 100%, 50%)
	navy	#000080	rgb(0, 0, 128)	hsl(240, 100%, 25%)
	blue	#0000ff	rgb(0, 0, 255)	hsl(240, 100%, 50%)
	teal	#008080	rgb(0, 128, 128)	hsl(180, 100%, 25%)
	aqua	#00ffff	rgb(0, 255, 255)	hsl(180, 100%, 50%)

Ici nous appliquons un fond `maroon` à tous les éléments avec la valeur d'attribut de classe `task` et un fond `yellow` à tous les éléments avec la classe `count`.

```
1 .task {  
2   background : maroon;  
3 }  
4 .count {  
5   background : yellow;  
6 }
```

Alors que les valeurs de couleur mots clés sont simples par nature, ils offrent des options limitées et ne sont donc pas le choix de valeur de couleur la plus populaire.

Couleurs en hexadécimales

Les valeurs de couleur en hexadécimal sont constitués d'un dièse, ou hash, #, suivi d'un chiffre à trois ou six caractère. Les chiffres utilisent les chiffres 0 à 9 et les lettres a à f, majuscules ou minuscules. Ces valeurs correspondent aux canaux rouge, vert et bleu.

Dans la notation à six caractères, les deux premiers caractères représentent le canal rouge, les troisième et quatrième caractères représentent le canal vert, et les deux derniers caractères représentent le canal bleu. Dans la notation à trois caractères, le premier caractère représente le canal rouge, le deuxième caractère représente le canal vert, et le dernier caractère représente le canal bleu.

Si dans la notation à six caractères les deux premiers caractères sont une paire de correspondance, les troisième et quatrième caractères sont une paire de correspondance, et les deux derniers caractères sont une paire de correspondance, le chiffre à six caractères peut être ramené à un chiffre à trois caractères. A cela le caractère répété de chaque paire doit être utilisé une fois faire. Par exemple, une nuance d'orange représenté par la couleur hexadécimal `#FF6600` pourrait également être écrit comme `#f60`.

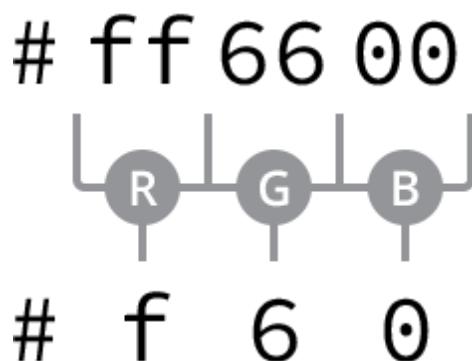


Fig 3 Six caractères peuvent être écrits en tant que valeurs trois caractères lorsque les canaux rouge, vert et bleu contiennent chacun un caractère répétitif.

Les paires de caractères sont obtenus en convertissant 0 à 255 en une base 16, ou format hexadécimal. Le calcul est un peu délicat, mais il est utile de savoir que 0 est égal à noir et f est égal à blanc.

Les millions de couleurs hexadécimales

Il y a des millions de couleurs hexadécimales, plus de 16,7 millions pour être exact. Voici comment ...

Il y a 16 options pour chaque caractère dans une couleur hexadécimale, 0 à 9 et A à F. Avec les caractères groupés par paires, il y a 256 options de couleur par paire (16 multiplié par 16, ou 16 carré).

Et avec trois groupes de 256 options de couleur nous avons un total de plus 16,7 millions de couleurs (256 multiplié par 256 multiplié par 256, ou 256 au cube).

Pour créer le même `maroon` et `yellow` nos couleurs de fond d'avant, nous pourrions remplacer les valeurs de couleur mots clés avec des valeurs hexadécimales de couleur, comme on le voit ici.

```
1 .task {  
2   background : #800000;  
3 }  
4  
5 .count {  
6   background : #ff0;  
7 }
```

Les valeurs de couleur hexadécimales sont devenues assez populaires parce qu'elles offrent un grand nombre d'options de couleur. Elles sont, cependant, un peu difficile de travailler, surtout si vous n'êtes pas trop familier avec l'hexadécimal.

Heureusement Adobe a créé [Adobe Kuler](#), une application gratuite qui fournit une roue de couleur pour nous aider trouver couleur que nous voulons et la valeur hexadécimale correspondante.

En outre, la plupart des applications de retouche d'image, comme Adobe Photoshop ou The Gimp offrent la possibilité de trouver les valeurs de couleur hexadécimales. On peut aussi citer les outils de développement web fournis par les navigateurs.

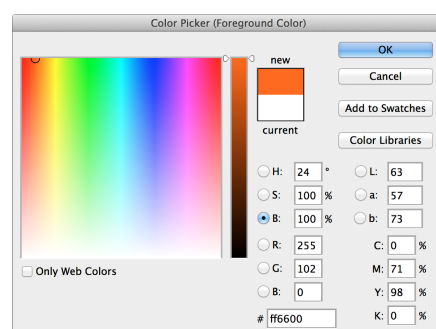


figure 3

L'outil sélecteur de couleur dans Adobe Photoshop affiche l'hexadécimal et les valeurs des couleurs RVB

Couleurs RGB & RGBA

Les valeurs de couleurs RVB sont exprimées en utilisant la fonction `rgb(n,n,n)`, qui signifie rouge, vert et bleu. La fonction accepte trois valeurs séparées par des virgules, dont chacun est un nombre entier entre 0 et 255. Une valeur de 0 sera un noir pur et une valeur de 255 sera un blanc pur.

Comme on pouvait s'y attendre, la première valeur dans la fonction `rgb` représente le canal rouge, la deuxième valeur représente le canal vert, et la troisième valeur représente le canal bleu.

Si nous devons recréer la nuance d'orange d'avant comme une valeur de couleur RVB, il serait représenté comme `rgb(255, 102, 0)`.

En utilisant les mêmes couleurs de fond `maroon` et `yellow`, nous pourrions remplacer les valeurs de mots clés ou couleur hexadécimales avec des valeurs de couleur RGB.

```
1 .task {
2   background : rgb(128, 0, 0);
3 }
4
5 .count {
6   background : rgb(255, 255, 0);
7 }
```

Les valeurs des couleurs RVB peuvent également inclure un canal alpha, ou canal de transparence en utilisant la fonction `rgba()`. La fonction `rgba()` nécessite une quatrième valeur, qui doit être un nombre décimal compris entre 0 et 1. Une valeur de 0 crée une couleur totalement transparente, signifie qu'elle sera invisible, et une valeur de 1 crée une couleur complètement opaque. Toute valeur décimale entre 0 et 1 créera une couleur semi-transparente.

Si nous voulions notre nuance d'orange à apparaître 50% opaque, nous utiliserons une valeur de couleur RGBA de `rgba(255, 102, ?, 0.5)`.

Nous peut également modifier l'opacité de nos couleurs de fond `maroon` et `yellow`.

Le code suivant définit la couleurs de fond `maroon` opaque à 25% et laisse la couleurs de fond `yellow` opaque à 100%.

```
1 .task {
2   background : rgba(128, 0, 0, 0.25);
3 }
4
5 .count {
6   background : rgba(255, 255, 0, 1);
7 }
```

Les valeurs des couleurs RVB sont de plus en plus populaire, surtout en raison de la possibilité de créer des couleurs semi-transparentes en utilisant la fonction RGBA.

Couleurs HSL & HSLa

Les valeurs chromatiques HSL sont évaluées en utilisant la fonction `hsl()`, qui s'exprime en teinte, saturation et luminosité. Dans les parenthèses, la fonction accepte trois valeurs séparées par des virgules un peu comme `rgb()`. La première valeur, la teinte, est un nombre sans unité de 0-360. Les chiffres 0 à 360 représentent la roue des couleurs, et la valeur identifie le degré d'une couleur sur la roue des couleurs.

Les deuxième et troisième valeurs, la saturation et la luminosité, sont des valeurs de pourcentage de 0 à 100%. La valeur de saturation indique la valeur de la nuance de la teinte, 0 étant niveaux gris et 100% étant totalement saturée. La luminosité identifie si la teinte est sombre ou claire, la valeur de teinte 0 étant complètement noir et 100% étant complètement blanc.

Nos couleurs de fond `maroon` et `yellow` peuvent également être exprimées sous forme valeurs de couleur HSL, comme indiqué ici.

```
1 .task{
2   background : hsl(0, 100%, 25%);
3 }
4
5 .count {
6   background : hsl(60, 100%, 50%);
7 }
```

Les valeurs de couleur HSL, comme RGBA, peuvent également comprendre un alpha, ou transparence, par l'utilisation de la fonction `hsla()`. Le comportement du canal alpha fonctionne tout comme celle de la fonction `rgba()` avec une quatrième valeur décimale entre 0 et 1 doit être ajoutée à la fonction d'identifier le degré d'opacité.

Notre nuance d'orange comme couleur HSLA fixé à 50% opaque serait représentée comme `hsla(24, 100%, 50%, .5)`.

Si l'on reprend la couleurs de fond `maroon` opaque à 25% et la couleurs de fond `yellow` opaque à 100% cela ressemblera à ce qui suit avec des valeurs de couleur HSLa.

```
1 .task{
2   background : hsla(0, 100%, 25%, .25);
3 }
4
5 .count {
6   background : hsla(60, 100%, 50%, 1);
7 }
```

L'utilisation de la notation des couleurs en HSL est récente dans le langage CSS. En raison de cela et de son support récent dans les navigateurs, il est pas aussi largement utilisé.

Pour le moment les valeurs hexadécimales de couleur restent les plus populaires car ils sont largement pris mais quand un canal alpha pour la transparence est nécessaire, les valeurs des couleurs RGBA sont préférés. Ces préférences peuvent changer à l'avenir, mais pour l'instant nous allons utiliser les valeurs hexadécimales et de couleurs RGBA.

En pratique

- Transformer les couleurs concernant en hexadécimal à l'aide du tableau ci-dessus
- Transformer en RGBA la couleur des classes "btn" et "btn-primary". Mettre une opacité de 50% sur la classe "btn-primary"

```
1 .btn{
2   color: rgba( 255, 0, 0, 1);
3 }
4 .btn-primary{
5   background: rgba( 128, 0, 0, .5);
6 }
```

Longueurs

Les valeurs de longueur en CSS sont similaires aux couleurs qu'il y a une poignée de différents types de valeurs pour exprimer la **longueur**. Les valeurs de longueur peuvent se libeller sous deux formes différentes, absolues et relatives, dont chacune utilisent différentes unités de mesure.

Nous allons nous en tenir aux valeurs simples les plus courantes à l'heure actuelle, sachant que les valeurs plus complexes fourniront beaucoup plus de possibilités que nous n'en avons besoin pour l'instant.

Longueurs absolue

Les valeurs de longueur absolues sont les valeurs de longueur simples, comme ils sont fixés à une mesure physique, comme pouces, centimètres ou millimètres. L'unité absolue de mesure la plus populaire est connue sous le nom de pixel et est représenté par la notation d'unité `px`.

Pixels

Le pixel est égal à 1 / 96e de pouce, il y a donc 96 pixels dans un pouce. La mesure exacte d'un pixel, cependant, peut varier légèrement entre les dispositifs de visualisation à haute densité et faible densité.

Les pixels existent depuis un certain temps et sont couramment utilisés avec une poignée de propriétés différentes. Le code ci-dessous utilise des pixels pour définir la taille de la police de tous les paragraphes à 14px *pixels*.

```
1 p {  
2   font-size: 14px;  
3 }
```

Avec l'évolution du paysage des matériels informatiques, les nombreux modèles d'écrans et leurs tailles différentes, les pixels ont perdu une partie de leur popularité. Comme une unité de mesure absolue, elles ne fournissent pas suffisamment de flexibilité. Les pixels sont, cependant digne confiance et très pratique pour commencer à gérer les tailles en CSS.

Longueurs relatives

Outre les valeurs de longueur absolue, il y a aussi les valeurs relatives. Les valeurs de longueur relatives sont un peu plus compliquées, car ils ne se base pas sur une unité fixe de mesure, ils comptent sur la longueur d'autre mesure.

Pourcentages

Pourcentages, représentés par l'unité %, notation de sont une des valeurs les plus populaires des unités relatives. Les pourcentages de longueurs sont définies par rapport à la longueur d'un autre objet. Par exemple, pour régler la largeur `width` d'un élément à 50%, il faut connaître la largeur de son élément parent, l'élément est emboîté dedans, et ensuite identifier 50% de la largeur de l'élément parent.

```
1 .col {  
2   width: 50%;  
3 }
```

Ici nous avons fixé la largeur de l'élément avec la valeur d'attribut de classe `col` à 50%. Ce "50%" sera calculé par rapport à la largeur du parent de l'élément. Les pourcentages sont extrêmement utiles pour régler la hauteur et la largeur des éléments et construire la mise page d'une page Web. Nous allons compter sur eux souvent pour nous aider dans ces domaines.

Em

L'unité `em` est également une valeur relative très populaire. L'unité de `em` est représenté par l'unité *em* et sa longueur est calculée sur la base la taille d'un élément de font.

Une unité *em* simple est équivalente à la taille police d'un élément. Ainsi, par exemple, si un élément a une taille de police de 14 pixels et une largeur `width` définie sur 5em, la largeur serait égale 70 pixels (14 pixels multiplié par 5).

```
1 .banner {  
2   font-size: 14px;  
3   width: 5em;  
4 }
```

Lorsque la taille de police n'est pas explicitement indiquée pour un élément, l'unité *em* sera rapport à la taille de police de l'élément parent le plus proche ayant une taille de police définie.

L'unité *em* est souvent utilisée pour le texte, les tailles de police, les espacements autour du texte y compris les marges externes et internes.

L'alternative : rem

Il s'agit d'une nouvelle unité qui signifie "*root em*". Elle n'est donc pas proportionnelle à son parent mais à la taille spécifiée pour la balise html.

Si on souhaite utiliser cette unité, On n'oubliera donc pas de réinitialiser la propriété font-size de son tag HTML comme le montre l'exemple ci-dessous.

```
1 html {  
2   font-size: 100%;  
3 }
```

Ou encore mieux

```
1 html {  
2   font-size: 62.5%;  
3 }
```

La taille par défaut du texte étant de 16px, ce qui est un peu grand, on diminue dès le départ cette valeur à 10px ($16 \times 62.5/100 = 10$).

Ensuite toutes les valeurs en « *rem* » définies dans notre feuille de style seront le pourcentage de cette valeur initiale.

```
1 html { font-size: 62.5%; } /* 1rem vaut 10px */  
2 body { font-size: 1.4rem; } /* 1.4 rem vaut 14px */  
3 h1 { font-size: 2.4rem; } /* 2.4 rem vaudra 24px */
```

On le voit dans l'exemple ci-dessus c'est quand même beaucoup plus simple.

Ces unités de mesures relatives (em et rem) ne servent pas uniquement à déterminer la taille de vos textes, elles sont aussi très utiles pour fixer la taille des margin et des padding. Ainsi si votre internaute décide de zoomer les pages de votre site, les proportions des espacements entre les éléments seront conservés.

Il y a beaucoup plus d'unités absolues et relatives de mesure que celles mentionnées ici. Cependant, ces quatre unités -pixels, pourcentages, em et rem- sont les plus populaires et celles que nous allons utiliser principalement.

