CSS3

Gaëlle Loosli

Automne 2012 - v1.0

Plan

CSS - Cascading Style Sheets

Pourquoi le CSS ?

- langage de style qui définit la présentation des documents HTML
- offre plus d'options et se montre plus précis et sophistiqué d'HTML
- est pris en charge par tous les navigateurs actuels

Principe : séparer le fond et la forme

- HTML sert à structurer le contenu, CSS sert à formater un contenu structuré.
- contrôle de la présentation de plusieurs documents par une seule feuille de style
- présentations différentes appliquées à des types de médias différents (à l'écran, à l'impression, etc.)

CSS - Mise en place

Utiliser le CSS

- le code CSS est placé dans un/plusieurs fichiers CSS à part
- le lien vers vers ses fichiers est fait dans l'entête du fichier HTML
- une modification du CSS affecte le rendu visuel de toutes les pages l'utilisant

Modifier l'apparence d'une balise

L'apparence de chaque balise du langage HTML peut-être personnalisée dans le CSS.

Mettre le titre principal en gros et rouge, centré

```
h1 {
  color: red;
  font-size: 24px;
  text-align: center;
}
```

Concepts de base

Balise muette

Sert à identifier des éléments que l'on souhaite afficher de manière particulière, sans utiliser une balise HTML pré-définie

Identifier visuellement les mots clefs

Dans le fichier HTML

```
<p> Mon joli <span class=motclef> texte </span> d'exemple</p>
```

Dans le fichier CSS

```
.motclef {
   font-size: 110%;
}
```

Concepts de base

Les blocs et divisions

La structuration du contenu se fait de manière globale à l'aide de blocs (balises div), dans lesquels sont regroupées des choses qui vont ensemble d'un point de vue sens.

Exemple pour un blog

- le bloc **article** est composé d'un titre, d'un auteur, d'un *contenu*, d'une date...
- le bloc **contenu** est composé d'un ou plusieurs paragraphes
- le bloc **publications** est composé d'un ensemble d' *articles*
- le bloc **menu** est composé d'une série de liens sur les différentes pages du site
- le bloc **pied de page** est composé du copyright, le la date de dernière mise à jour...
- le bloc **page** est composé d'un *menu*, de *publications* et d'un *pied de page*

```
<div id=page>
  <div id=menu>...</div>
  <div id=publications>
    <div class=article>
      <div class=contenu>...</div>
    </div>
    ...
  </div>
</div>
```

Concepts de base

La mise en forme des bloc

Dans le fichier CSS, on positionne et décore les bloc à l'aide de leur identifiant (id) ou classe (class).

Reprise de l'exemple

style_blog.css

```
#page{
  background-color: #7A95C7;
}
#menu{ ... }
#publication{ ... }
.article{ ... }
.contenu{ ... }
```

```
<div id=page>
  <div id=menu>...</div>
  <div id=publications>
    <div class=article>
      <div class=contenu>...</div>
    </div>
    ...
  </div>
</div>
```

class ou id ?

On utilise le mot clef *id* pour un bloc unique dans la page et il est associé au symbole # dans le CSS Le mot clef *class* est utilisé pour les blocs qui peuvent être utilisés plusieurs fois, et est associé au symbole . dans le CSS.

Modifier l'apparence des blocs

Polices

- font-family ("Arial Black", Arial, Verdana, serif, ...)

Tailles

- font-size (12px, 80%, small, medium...)
- font-weight (bold, bolder, lighter, normal)
- font-style (italic, oblique)

Décorations

- text-decoration (underline, overline, line-trough, blink, none)
- font-variant (small-caps, normal)
- text-transform (uppercase,lowercase,capitalize,none)

Modifier l'apparence des blocs

Alignement

- text-align (left,right,center, justify)
- vertical-align (top, middle, bottom)
- line-height (% ou px)
- text-indent (px)
- white-space (césure : normal, nowrap, pre)

Couleurs et fonds

- color (#000000, rgb(20,20,0))
- background-color
- background-image (url)
- background-attachment (fixed, scroll)
- background-repeat (repeat, repeat-x, repeat-y, no-repeat)
- background-position (% ou px par rapport au coin haut-gauche, ou position : top, center, bottom, left, right)

Modifier l'apparence des blocs

Taille

- width (%, px ou auto)
- height
- min-height, max-height
- min-width, max-width
- margin
- margin-top, margin-bottom, margin-left, margin-right,
- padding
- padding-top, padding-bottom, padding-left, padding-right

Modifier l'agencement des blocs

Affichage

- display (none, block, inline)
- visibility (none, hidden, visible)
- clip : rect()
- overflow (visible, hidden, scroll, auto)

Modifier l'agencement des blocs

Positionnement

- float (left, right, non)
- clear (left, right, both, none)
- position (absolute, fixed, relative, static)
- top (px, %)
- bottom
- left
- right
- z-index

Et ce qui n'est pas un bloc

Bordures

- border-width
- border-color
- border-style (none, hidden, solid, double, dashed, dotted, inset, outset, ridge)
- border-left...

Listes

- list-style-type (disc, circle, square, none, decimal, upper-roman, lower-roman,...)
- list-style-position (inside, outside)
- list-style-image (url)

Propriété display

Valeurs possibles

- bock : l'élément génère une boîte type *block*, avec un saut de ligne avant et après
- inline : l'élément génère une boîte type *inline*, sans saut de ligne ni avant ni après
- none : l'élément de génère aucune boîte

Example de code

```
.monbloc
{
  display:block;
}
```

Propriété position

Valeurs possibles

- static : comportement par défaut, affiche les élément dans l'ordre du fichier source
- absolute : l'élément est positionné relativement à son premier ancêtre positionné (non static), à défaut, le navigateur
- relative : l'élément est positionné relativement à sa position normale
- fixed : l'élément est positionné par rapport à la fenêtre du navigateur
- inherit : valeur hérité de l'élément parent

Example de code

```
h2
{
  position:absolute;
  left:100px;
  top:150px;
}
```

Positionner

Propriétés utiles

- top, left, right, bottom : donne une distance à respecter entre l'élément et son voisinage, selon le type de positionnement et la direction donnée
- z-index : permet d'ordonner les éléments en cas de superposition
- overflow (auto, hidden, scroll, visible) précise le comportement de l'élément si le contenu dépasse la taille allouée dans la mise en page.

Groupes

Grouper des sélecteurs aux propriétés identiques

```
h1
{
  color:green;
}
h2
{
  color:green;
}
```

Peut se résumer

```
h1, h2
{
  color:green;
}
```

Imbrication

Spécifier le comportement d'un sélecteur selon son contexte

```
p
{
  color:blue;
  text-align:center;
}
.marked
{
  background-color:red;
}
.marked p
{
  color:white;
}
```

Pseudo-classes

Syntaxe

```
selecteur.classe:pseudo-classe
{
  propriete:valeur;
}
```

Exemples

- a :link : lien non visité
- a :visited : lien visité
- a :hover : lien survolé
- a :active : lien sélectionné

Pseudo-élements

Syntaxe

```
selecteur.classe:pseudo-element
{
  propriete:valeur;
}
```

Exemples

- :first-line : s'applique à la première ligne de l'élément, si l'élément est de type block
- :first-letter : s'applique à la première lettre de l'élément, si l'élément est de type block
- :before : ajoute un contenu avant l'élément
- :after : ajoute un contenu après l'élément

Contenu

Contenu

La propriété content sert à insérer un contenu généré avant ou après un élément, via les pseudo-élements *:before* ou *:after*.

- counter : le contenu ajouté est un compteur
- attr(attribute) : le contenu ajouté est la valeur d'un attribut du sélecteur
- string : ajoute un texte
- open-quote et close-quote : ajoute des guillemets ouvrant ou fermant
- url(str) : ajoute un média (image, son, vidéo, ...)

Ajouter l'adresse du lien entre parenthèses après le lien

```
a:after
{
content: " (" attr(href) ")";
}
```