# Introduction Web
# Cours

1ère année

## Institut Galilée.

Mathieu Lacroix[1]

1. E-mail : mathieu.lacroix@univ-paris13.fr,
Page Web : http://www.lipn.univ-paris13.fr/~lacroix/

# 1. Web et HTML

Ce chapitre présente le langage de balises HTML ainsi que l'interaction entre le client et le serveur lors de la consultation d'une page Web. Ce chapitre concerne uniquement les pages Web statiques, c'est-à-dire des pages Web dont le contenu ne varie pas (sauf modification du code source HTML) et est le même pour tous les utilisateurs (par opposition aux pages de sites Web dynamiques tels que les webmails, les forums de discussion, moteurs de recherche, etc).

## 1.1 Principe du Web

Cette partie explique succintement l'interaction entre le client et le serveur lors de la consultation d'une page Web statique.

### 1.1.1 Définitions

Voici une liste de définitions des principaux termes liés au Web statique.

— *Réseau* : ensemble de machines (ordinateurs, imprimantes, modems routeurs, ...) pouvant communiquer entre elles. Pour être en mesure de communiquer, ces machines doivent être reliées entre elles soit par des câbles, soit par des ondes (wifi). Un ordinateur relié à un modem (box) correspond à un réseau. De même, tous les ordinateurs de l'institut Galilée forment un réseau.

— *Internet* : obtenu par contraction de "Interconnexion Networks" (interconnexion de réseaux). Internet est un réseau connectant entre eux différents réseaux existants. Internet a pour vocation de relier tous les ordinateurs du monde. Internet permet de faire communiquer deux machines connectées sur des réseaux différents. Mais Internet n'est qu'un réseau, c'est-à-dire qu'il permet juste d'envoyer des données (0 ou 1) entre deux ordinateurs.

— *Web* : application permettant de consulter à l'aide d'un programme spécifique appelé *navigateur*, des pages d'un site (ensemble de pages reliées entre elles). Si le navigateur et le site se trouvent sur des ordinateurs différents, alors le Web utilise l'Internet pour la communication entre ces deux ordinateurs. Le Web n'est donc qu'une application utilisant Internet. D'autres applications sont le courrier électronique, la messagerie instantanée, etc.

— *Navigateur* : Le navigateur est un logiciel permettant l'affichage de pages d'un site Web. Différents navigateurs existent : Google Chrome (58.9%), Apple Safari (13.7%), Mozilla Firefox (5.2%) et Microsoft Edge (5.0%) sont différents exemples de navigateurs.

— *Langage HTML* : langage de balises permettant de spécifier dans une page Web quelles parties correspondent à un titre, un paragraphe, quelles images insérer dans la page, etc.

Le Web est donc une application permettant d'afficher, dans un navigateur, des fichiers texte contenant du code HTML. Ces fichiers se trouvent généralement sur un ordinateur différent et le Web utilise alors l'Internet pour transmettre les informations entre les deux ordinateurs : celui possédant le fichier, appelé *serveur*, et celui sur lequel est utilisé le navigateur, appelé *client*.

La principale caractéristique du Web est la possibilité de passer d'un fichier à l'autre grâce aux hyperliens. Ce sont ces hyperliens qui permettent par exemple de passer de la page d'accueil d'un compte facebook aux pages contenant les photos, les messages, etc.

### 1.1.2 Qu'est-ce qu'une adresse Web (url) ?

Pour afficher une page Web, il faut indiquer le fichier contenant le code HTML de cette page. Ces informations sont données sous forme d'*adresse Web* ou *url* (Uniform Resource Locator, littéralement "localisateur uniforme de ressource") en anglais. Une adresse Web est de la forme :

```
nom_du_protocole://adresse_du_site/chemin_du_repertoire/fichier
```

— *protocole* : définit les règles pour un type de communication entre deux ordinateurs. Pour le Web, le protocole est `http`.

— *adresse_du_site* : correspond à l'adresse du serveur sur lequel se trouve le fichier que vous désirez consulter. Cette adresse correspond au numéro IP de la machine (ex : 157.240.21.39) ou à une chaîne de caractères (ex : www.facebook.com) qui est appelée *nom de domaine*. Un nom de domaine est associé à un numéro IP unique.
— *chemin_du_repertoire* : correspond au répertoire où se trouve le fichier sur le serveur.
— *fichier* : nom du fichier que vous voulez consulter.

Voici différents exemples d'adresses Web :

```
http://lipn.univ-paris13.fr/~lacroix/Documents/Web/interaction.html
http://157.240.21.39/index.php
http://www.facebook.com/
```

On remarque que pour la dernière adresse donnée en exemple, il n'y a pas de fichier spécifié. Dans ce cas, le fichier est alors le fichier `index.html` ou `index.php`. Comme le nom de domaine `www.facebook.com` est associé au numéro IP 157.240.21.39, les deux dernières adresses sont équivalentes !

### 1.1.3 Que se passe-t-il lorsque je saisis une adresse Web (`fichier.html`) ?

Lorsque je clique sur un hyperlien ou que je saisis dans mon navigateur une adresse Web dont le fichier porte l'extension `.html`, l'affichage de la page Web se fait de la manière suivante :

1. le client (navigateur) envoie un message au serveur pour lui demander de lui envoyer le code HTML contenu dans le fichier indiqué par l'url,
2. le serveur répond en envoyant le contenu du fichier demandé,
3. le navigateur interprète le code HTML reçu et affiche la page Web.

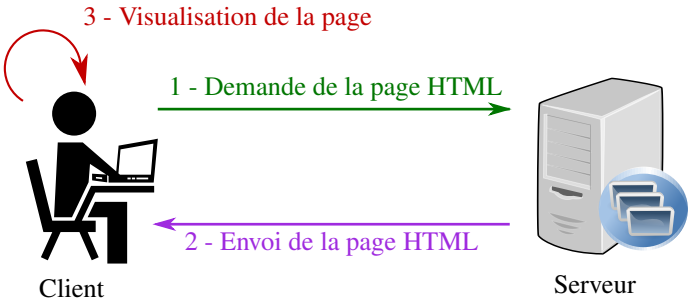


FIGURE 1.1 – Schéma de requête en HTML [1]

De manière plus précise, pour demander au serveur de lui envoyer le code du fichier HTML, le client (mon navigateur) va envoyer au serveur un message du type :

```
GET /~lacroix/Documents/Web/interaction.html HTTP/1.1
Host: lipn.univ-paris13.fr
```

Ce code correspond alors à la requête adressée au serveur `lipn.univ-paris13.fr` d'envoyer (`GET`) le code du fichier `interaction.html` se trouvant dans le dossier `/~lacroix/Documents/Web/` selon le protocole `HTTP (version 1.1)`. Généralement, la requête est suivie de quelques informations supplémentaires qui ne sont pas nécessaires. Le client peut dire sur quel type de système d'exploitation il tourne (windows, linux, mac), quelle est la taille en octets de la requête envoyée, etc.

Le serveur recevant la requête envoie à son tour un message. Celui-ci ressemble alors à

```
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html lang="fr">
<head>
```

1. Image construite à partir d'une icone créée par Freepik sur www.flaticon.com.

```
7     <title>Interaction client/serveur</title>
8     <meta charset="utf-8">
9   </head>
10  <body>
11  <h1> Interaction client/serveur lors de la saisie d'une url (statique) </h1>
12  ...
```

La première ligne indique que le serveur a accepté la réponse. La deuxième ligne spécifie que le serveur envoie du code HTML. Le reste correspond au code HTML du fichier `interaction.html`.

Le navigateur recevant le code HTML interprète les balises et affiche alors la page Web. Pour l'exemple, on obtient :

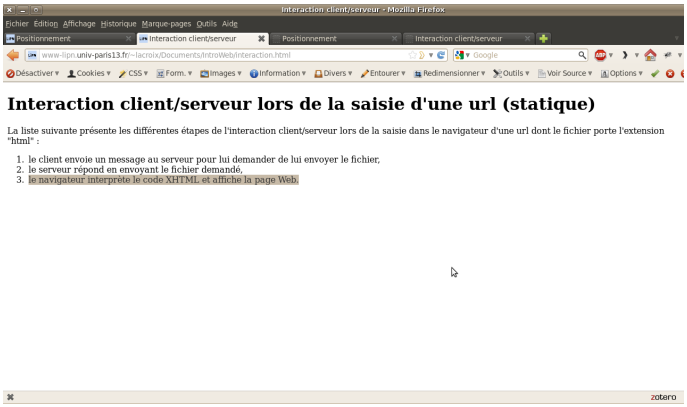


FIGURE 1.2 – Visualisation de la page Web avec le navigateur

Certaines pages Web nécessite du contenu additionnel (images, scripts javascript, fichiers CSS, etc) pour s'afficher correctement. Dans ce cas, lors de la réception du code HTML, le client effectue au besoin de nouvelles requêtes pour récupérer les fichiers additionnels. Le nombre de requêtes que fait le client dépend du nombre de fichiers nécessaires pour afficher la page Web. À titre d'exemple, l'affichage de la page d'accueil du site `leMonde.fr` nécessite 108 requêtes ! Plus ce nombre est important, plus la communication entre le serveur et le client est importante, et plus le navigateur met du temps à afficher la page.

R *Les fichiers nécessaires pour afficher une page Web peuvent se trouver sur différents serveurs. Dans ce cas, le navigateur demande à chaque serveur les fichiers dont il a besoin.*

## 1.2 Le langage HTML

Le langage HTML est un langage de balises permettant de décrire la structure de la page Web, c'est-à-dire quelle partie correspond à un titre, un paragraphe, un tableau, etc. Chaque balise commence par le caractère "<" et se termine par le caractère ">". Ces balises ne sont pas affichées par le navigateur car elles ne correspondent pas à du texte. Elles servent uniquement à décrire les éléments d'une page Web.

Les balises vont généralement par paire : la balise ouvrante et la balise fermante. Cette dernière se distingue de la balise ouvrante par le caractère "/" qui est ajouté juste après le caractère "<". Ces balises servent à indiquer que ce qui est entre la balise ouvrante et la balise fermante correspond à un élément de la page. Par exemple, le code

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

indique que tout se qui se trouve entre la balise  et la balise  correspond à un paragraphe.

Certaines balises ne sont cependant pas utilisées par paire. Ces balises sont appelées balises *auto-fermantes*. C'est le cas lorsqu'il n'y a pas rien à mettre entre la balise ouvrante et la balise fermante. On utilise alors la notation `<balise/>`. Par exemple, la balise `<br/>` crée un retour à la ligne.

Il est parfois nécessaire d'ajouter des attributs aux balises pour spécifier certaines informations. Ceci se fait de la manière suivante :

```
<balise attribut1="valeur1" attribut2="valeur2"> ... </balise>
<balise attribut1="valeur1" attribut2="valeur2" />
```

Les attributs sont écrits en minuscules et la valeur de l'attribut est entre guillemets. On peut avoir plusieurs attributs séparés par des espaces. Par exemple, l'insertion d'une image se fait grâce à la balise `<img/>`. Pour indiquer quelle image afficher, il faut spécifier la source de l'image grâce à l'attribut `src` : `<img src="image.jpg"/>`.

R *Comme pour beaucoup de langages informatiques, mettre plusieurs espaces d'affilée ou sauter des lignes ne modifie absolument rien. Profitez-en pour écrire un code agréable à lire ! De plus, vous pouvez ajouter des commentaires pour faciliter la compréhension de votre code HTML de la manière suivante :*

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

R *Le symbole '<' correspondant à l'ouverture d'une balise en HTML, il n'est pas possible d'utiliser ce symbole comme un caractère normal. Pour insérer ce symbole, il faut utiliser le codage spécial `&lt;`. Ainsi, si je souhaite écrire un paragraphe contenant "1 < 2" en HTML, je dois alors écrire `<p> 1 &lt; 2 </p>`. De la même manière, si je veux insérer le symbole &, il faut alors insérer `&amp;`.*

### 1.2.1 Principales balises HTML

**Titres** Il existe en HTML six niveaux de titres différents donnés par les balises `h1`, `h2`, `h3`, `h4`, `h5` et `h6`. La balise : `h1` définit un titre de partie, `h2` de sous-partie, `h3` de sous-sous-partie, etc. Par exemple, le début de la table des matières de ce cours serait représentée à l'aide du code HTML suivant :

```
<h1> Internet, Web et HTML </h1>
<h2> Introduction </h2>
  <h3> Définitions </h3>
  <h3> Qu'est-ce qu'une adresse Web (url) ?</h3>
  <h3> Que se passe-t-il lorsque je saisis une adresse Web ?</h3>
```

**Paragraphes** Un paragraphe est compris entre les balises `<p>` et `</p>`

**Listes ordonnées et non ordonnées** Les listes ordonnées (numérotées) sont définies avec la balise `ol` et les listes non-ordonnées (à puces) avec la balise `ul`. Chaque item de la liste est défini par les balises `<li>` et `</li>`. Les différents items sont sont entourés des balises `<ol>` et `</ol>` ou `<ul>` et `</ul>` selon le type de la liste. Par exemple, la liste de la page 7 peut être représentée par le code HTML suivant :

```
<ol>
  <li>le client (navigateur) envoie un message au serveur pour lui
      demander de lui envoyer le code HTML contenu dans le fichier
      indiqué par l'url,</li>
  <li>le serveur répond en envoyant le contenu du fichier demandé,</li>
  <li>le navigateur interprète le code HTML reçu et affiche la page Web.</li>
</ol>
```

**Saut de lignes** Un saut de ligne est représenté par la balise auto-fermante `<br />`.

**Ligne horizontale** Une ligne horizontale est dessinée grâce à la balise auto-fermante `<hr />`.

**Mise en valeur de texte** Le texte mis en valeur est compris entre les balises `<em>` et `</em>`, celui fortement mis en valeur entre les balises `<strong>` et `</strong>`.

**Insertion d'images** Une image peut être insérée grâce à la balise auto-fermante `<img />`. Cette balise doit obligatoirement contenir les deux attributs suivants :

- `src` : indique le nom du fichier image ainsi que son chemin. Ce chemin peut être *relatif*, c'est-à-dire défini par rapport à l'endroit où se trouve la page Web ou *absolu*. Si l'image se trouve sur le même site que la page Web, il est préférable d'utiliser le chemin relatif.
- `alt` : texte alternatif décrivant l'image, utilisé si l'image n'existe plus ou pour les navigateurs textes par exemple.

Voici un exemple de code HTML insérant le fichier image "logoIG.png" qui se trouve dans le répertoire Images contenu dans le répertoire où se situe la page Web contenant le code :

```
<img src="Images/logoIG.png" alt="logo de l'institut Gaglilée" />
```

**Hyperliens** Les hyperliens (appelés aussi liens hypertextes) permettent de naviguer d'une page Web à une autre. Un hyperlien est défini par la balise `a`. L'élément cliquable (texte ou image par exemple) doit être compris entre la balise ouvrante `<a>` et la balise fermante `</a>`. Il est nécessaire de donner en attribut (attribut *href*) la page qui doit s'ouvrir lors du clic sur le lien. Pour pouvoir, à partir du fichier `fichier1.html`, accéder en cliquant au fichier `fichier2.html`, il faut ajouter dans le code HTML du `fichier1.html` le code

```
<a href="chemin_du_fichier2/fichier2.html"> cliquez ici pour accéder à
fichier2.html </a>
```

Là encore, le chemin du fichier HTML peut être relatif ou absolu.

**Tableaux** En HTML, un tableau (balise `table`) est codé comme un ensemble de lignes (balises `tr`), chaque ligne contenant un ensemble de cellules (balises `td`). Le tableau

| | | |
|---|---|---|
| Cellule 1 | Cellule 2 | Cellule 3 |
| Cellule 4 | Cellule 5 | Cellule 6 |

est alors codé en HTML de la manière suivante :

```
<table>
  <tr>
    <td> Cellule 1 </td> <td> Cellule 2 </td> <td> Cellule 3 </td>
  </tr>
  <tr>
    <td> Cellule 4 </td> <td> Cellule 5 </td> <td> Cellule 6 </td>
  </tr>
</table>
```

Généralement, les cellules de la première ligne et/ou colonne d'un tableau contiennent les noms des données (ex : nom, age, etc) de la colonne. On parle alors de cellules d'en-tête du tableau. En HTML, une cellule d'en-tête est définie à l'aide des balises `<th>` et `</th>` au lieu des balises `<td>` et `</td>`.

**Articles** Un article, défini par la balise `<article>`, regroupe un contenu ayant son propre sens indépendamment du reste des autres éléments de la page (article de journal, commentaire, etc).

**Sections** Une section, définie par la balise `<section>`, regroupe plusieurs éléments (articles, etc) ayant une même thématique.

**En-têtes** Un en-tête de page, de section ou d'article est délimité par `<header>` et `</header>`.

**Pieds de page** Un pied de page regroupe des informations de bas de page, de section ou d'article. Il est délimité par `<footer>` et `</footer>`.

**Menu de navigation** Défini par la balise `<nav>`, il regroupe des hyperliens (menu de navigation).

**Partie principale** La partie principale de la page Web, qui est unique, est délimitée par `<main>` et `</main>`.

### 1.2.2 Structure d'un document HTML

Pour être valide, le document HTML doit respecter la structure suivante :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title> TITRE OBLIGATOIRE </title>
    <meta charset="utf-8" />
  </head>
  <body>
    ... le code HTML de la page ...
  </body>
 </html>
```

Les lignes 1 et 2 indiquent que le document est un document HTML en français. La balise `<head>` (lignes 3-6) contient un ensemble d'informations non affichables tel que l'encodage du fichier (balise `<meta/>`) ou le titre de l'onglet du document (balise `<title>` **obligatoire**). L'ensemble des informations à afficher (paragraphes, titres, etc) se trouvent entre `<body>` et `</body>`.

### 1.2.3 Représentation d'un document HTML sous forme d'arbre

Tout document HTML peut être représenté sous forme d'arbre. Cet arbre sera par la suite utilisé pour définir certaines règles d'écriture ou de mise en page à l'aide du code CSS. Cette construction sous forme d'arbre est relativement intuitive et peut être vue comme un arbre généalogique. Les sommets de l'arbre sont des balises HTML. Il existe une flèche d'une balise, disons `balise1`, à une autre, disons `balise2`, si `balise2` est comprise entre `<balise1>` et `</balise1>` et qu'il n'y a pas une balise comprise entre `<balise1>` et `</balise1>` contenant la balise `balise2`. Bien que cette notion semble abstraite et difficile, elle est très facile à comprendre. Pour cela, considérons le document HTML suivant.

```
1  <!DOCTYPE html>
2    <html lang="fr">
3      <head>
4        <title> TITRE OBLIGATOIRE </title>
5        <meta charset="utf-8" />
6    </head>
7    <body>
8      <h1> Voici un titre <strong> avec des mots importants ! </strong> </h1>
9      <p> Voici un premier paragraphe. Il contient notamment <a href="doc.html">
10         un lien vers le fichier doc.html </a>. </p>
11     <p> Le deuxième paragraphe contient des mots à mettre en <em>valeur</em>.
12         Nous avons ensuite une liste non ordonnée : </p>
13     <ul>
14       <li> Premier point : encore <strong> <a href="link.html"> un lien </a>
15       </strong> </li>
16       <li> Deuxième point : une autre liste
17         <ul>
18           <li> avec un item </li>
19           <li> et un autre </li>
20         </ul>
21       </li>
22       <li> Finalement, un dernier point </li>
23     </ul>
24   </body>
25  </html>
```

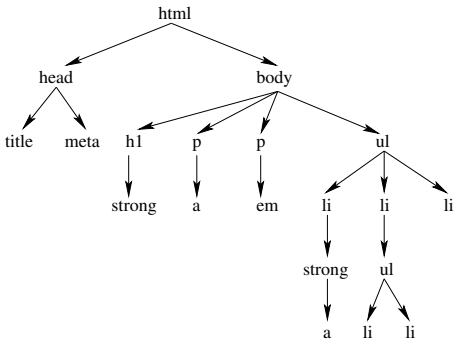


FIGURE 1.3 – Arbre de représentation du document HTML

L'arbre représentant ce code HTML est donné dans la figure 1.3. Il y a une flèche de `html` à `body` car la balise `body` se trouve entre `<html>` et `</html>` et il n'existe pas de balise contenant `body` qui est contenue dans `html`.

Cet arbre permet d'indiquer différentes relations entre les balises, similaires à celles définies pour les arbres généalogiques :

— Une balise `balise1` est dite *enfant* d'une balise `balise2` s'il existe une flèche de `balise1` à `balise2`. Par exemple la balise `h1` est enfant de `body`.
— Une balise `balise1` est dite *parent* d'une balise `balise2` si `balise2` est enfant de `balise1`. Par exemple la balise `body` est parent de `h1`.
— Une balise `balise1` est dite *descendant* d'une balise `balise2` si en prenant successivement les parents de `balise1`, on arrive sur `balise2`. Par exemple, la balise `title` est un descendant de `html` car `head` est parent de `title` et `html` est parent de `title`.
— Une balise `balise1` est dite *ascendant* d'une balise `balise2` si `balise2` est descendant de `balise1`.

R *Par définition, tout parent est un ascendant et tout enfant un descendant.*

**Attention :** Dans l'arbre de représentation, les enfants d'une balise sont ordonnés de gauche à droite suivant l'ordre dans lequel ils apparaissent dans le document HTML. Ainsi, la balise `head` est à gauche de la balise `body` puisque `head` apparaît avant `body` dans le document.

### 1.2.4 Différences entre balises de type `block` et balises de type `inline`

Les différentes balises que l'on peut mettre dans le corps de la page peuvent être classées en deux groupes : balises de type `block` et balises de type `inline`.

Les (principales) balises de type `inline` sont : `br`, `em`, `strong`, `img` et `a`. Ces balises n'introduisent pas de saut de ligne dans le flux du document (elles ne commencent pas sur une nouvelle ligne) et prennent uniquement la place dont elle ont besoin.

Les autres balises sont de type `block`. Un retour à la ligne est inséré avant et après chaque élément (créant ainsi un « bloc » de contenu). Ils prennent la largeur de leurs conteneurs (largeur de la page par défaut).

### 1.2.5 Règles d'écriture d'un fichier HTML

Afin d'être valide, un document HTML doit vérifier un certain nombre de règles. Ces règles ont pour but d'afficher correctement une page Web mais permettent également une meilleure visibilité du site sur le Web. De plus, elles permettent de rendre le site accessible pour tous les types de navigateurs (navigateurs textes par exemple). Il est primordial de suivre ces règles !

**Non imbriquation de balises** Il n'est pas possible d'imbriquer deux balises, c'est-à-dire d'ouvrir une première balise, puis une seconde, puis fermer la première et fermer la seconde. Le code

```
<p>Paragraphe avec texte en <strong>gras</p></strong>
```

n'est donc pas valide ! Il faut écrire à la place

```
<p>Paragraphe avec texte en <strong>gras</strong></p>
```

**Enfants et parents possibles** Certaines balises sont interdites comme enfants ou parents d'une autre balise. Ces règles découlent généralement du bon sens. Voici quelques exemples :

— un titre ne peut pas être fils de paragraphe et inversement,
— un paragraphe ne peut pas être fils d'un autre paragraphe,
— les seuls fils possibles pour les balises `ul` et `ol` sont les balises `li`,
— les balises de type `inline`, excepté la balise `a`, ne peuvent être parents d'une balise de type `block`.

Pour connaître la liste des enfants et des parents possibles pour une balise, vous pouvez regarder sur le Web différentes références telles que celle située à l'adresse :

`https://developer.mozilla.org/fr/docs/Web/HTML/Element`.

R *Même si le navigateur affiche correctement une page Web, cela ne signifie pas que celle-ci est valide. Il faut vérifier un code HTML à l'aide d'un validateur HTML tel que* `https://validator.w3.org/`.

# 2. Mise en page avec CSS

Le langage HTML permet de spécifier la structure d'une page Web (titres, paragraphes, sections, etc). Le langage CSS (pour Cascading Style Sheets) permet de mettre en forme une page Web en précisant comment doivent être affichés les différents éléments de la page.

## 2.1 Écriture de code CSS

La mise en page avec le langage CSS se fait grâce à un ensemble de *règles*, chacune étant définie de la manière suivante :

```
selecteur
{
  propriete : valeur;
}
```

Le sélecteur est l'élément dont on souhaite modifier l'apparence. La propriété (d'affichage) est celle que l'on veut modifier avec la nouvelle valeur. Par exemple, si l'on souhaite que les titres de niveau 1 soient affichés en rouge, il suffit d'écrire la règle :

```
h1
{
  color : red;
}
```

*Une règle peut contenir plusieurs propriétés. Dans un fichier CSS, il peut y avoir plusieurs règles ayant le même sélecteur.*

Les règles CSS sont écrites dans un fichier (avec l'extension `.css`) distinct du fichier HTML [1]. Pour que ce fichier soit pris en compte lors de l'affichage de la page Web, il faut indiquer le nom du fichier CSS dans le fichier HTML. Ceci se fait grâce à la balise auto-fermante `link` qui doit être contenue entre les balises `<head>` et `</head>`. La balise `link` s'utilise de la manière suivante :

```
<link href="fichierCSS.css" rel="stylesheet"/>
```

L'attribut `href` permet de spécifier les nom et chemin du fichier CSS. L'attribut `rel` (obligatoire) indique que le fichier est un fichier de code CSS.

## 2.2 Les principales propriétés et leurs valeurs

Seules les principales propriétés et valeurs sont présentées ici. Pour une liste exhaustive, se référer au site `https://developer.mozilla.org/fr/docs/Web/CSS/Reference`.

### 2.2.1 Modification de la police

**font-family** permet de spécifier la police qui doit être utilisée. Les valeurs possibles sont les différentes polices existantes telles que Times, Arial, Verdana, sans-serif, Script, Courrier, "Times New Roman", etc. Il est conseillé de mettre comme valeur plusieurs polices au cas où le navigateur ne possède pas les premières polices. (*Valeur par défaut : dépend du navigateur*)

*Il est possible d'importer d'autres font que celles supportées par défaut par le navigateur grâce à la règle `@font-face`, c.f. `https://developer.mozilla.org/fr/docs/Web/CSS/@font-face` pour plus d'information.*

1. Il existe d'autres moyens d'associer du code CSS à un code HTML qui ne sont pas abordés dans ce cours.

**font-size** indique la taille du texte. La taille peut être spécifiée de manière absolue en donnant sa valeur en pixels ou en utilisant les valeurs `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`. La taille peut également être donnée de manière relative, c'est-à-dire par rapport à la taille de l'élément parent. Dans ce cas, la valeur est donnée en pourcentage ou en unité "em" (1em correspondant à 100%). (*Valeur par défaut :* `medium` *sauf pour les titres*)

**font-style** permet de spécificier si le texte est en italique (valeur : `italic`) ou non (valeur : `normal`). (*Valeur par défaut :* `normal` *sauf pour la balise* `em` *où la valeur est* `italic`)

**font-weight** permet de spécificier si le texte est en gras (valeur : `bold`) ou non (valeur : `normal`). Il est également possible de mettre comme valeur un nombre entre 100 et 900. (*Valeur par défaut :* `normal` *sauf pour la balise* `strong` *où la valeur est* `bold` *et pour les titres où la valeur est spécifiée par un nombre*)

**text-align** permet de spécifier l'alignement horizontal du texte. Cette propriété s'utilise uniquement pour les balises de type `block`. Les différentes valeurs sont :
- alignement à gauche (`left`),
- alignement à droite (`right`),
- centré (`center`),
- texte justifié (`justify`)

(*Valeur par défaut : dépend du navigateur et du sens de lecture*)

**text-decoration** permet de spécificier si le texte est souligné (valeur : `underline`) ou sans décoration (valeur : `none`). (*Valeur par défaut :* `none` *sauf pour la balise* `a` *où la valeur est* `underline`)

**color** permet de spécifier la couleur d'affichage du texte. La valeur de cette propriété peut être donnée sous trois formes :
- nom de la couleur (en anglais) : `red`, `blue`, `green`, `fuschia`, `gray`, `olive`, `purple`, etc. Environ 140 couleurs sont reconnus par les navigateurs modernes,
- code hexadécimal de la couleur : composé de trois chiffres hexadécimaux représentant les tons de Rouge, de Vert et de Bleu, il permet de définir plus de 16 millions de couleurs. La couleur indigo correspond par exemple au code hexadécimal #4B0082.
- code RVB : permet de définir les composantes de rouge, vert et bleu en indiquant leur proportion en pourcentage ou en notation absolue (nombre compris entre 0 et 255). La couleur indigo correspond au code RVB `rgb(75,0,130)`.

Les deux dernières formes sont équivalentes et permettent de définir plus de 16 millions de couleurs. (*Valeur par défaut : dépend du navigateur*)

Un exemple interactif se trouve à l'adresse `https://codepen.io/mathieulacroix/pen/xawyzL`.

### 2.2.2 Dimensions

Il est possible de spécifier les dimensions du contenu d'une balise de type `block`. Ceci se fait à l'aide des propriétés suivantes :

**width** permet de spécifier la largeur du contenu de la balise. Cette taille peut être exprimée de manière absolue en pixels, ou de manière relative en pourcentage. (Dans ce cas, la largeur de l'élément est égale à la largeur de l'élément parent multiplié par le pourcentage.) `width` peut également prendre la valeur `auto`. Dans ce cas, la valeur est égale à la taille maximum possible que peut prendre l'élément. Cette valeur dépend des marges, de la taille de la bordure (cf. plus loin) et de la taille de l'élément parent. (*Valeur par défaut :* `auto`)

**height** permet de spécifier la hauteur du contenu de la balise. La hauteur peut être exprimée en pixels ou pourcentage, ou par la valeur `auto`. Dans ce cas, la valeur correspond à la hauteur minimum nécessaire pour afficher l'élément. (*Valeur par défaut :* `auto`)

**min-height** permet de spécifier la hauteur minimum du contenu de la balise. La hauteur du contenu de la balise de type `block` associée à cette propriété sera égale au maximum entre la valeur donnée pour cette propriété et la hauteur nécessaire pour afficher correctement le contenu. Cette propriété sera utile par la suite pour s'assurer que les différentes parties (menu, en-tête, partie principale, pied de page) d'une page Web s'affichent correctement. (*Valeur par défaut :* `0`)

Ces propriétés ne s'utilisent pas pour les éléments de type `inline` qui, par défaut, n'ont pas de dimensions. La seule exception est la balise `img` pour laquelle les deux premières propriétés peuvent être utilisées pour indiquer la largeur et la hauteur de l'image.

### 2.2.3 Bordures et marges

Il est possible de spécifier pour chaque balise une bordure et des marges intérieures et extérieures. La figure 2.1 montre à quoi correspondent les bordures et les marges intérieures et extérieures d'un élément.

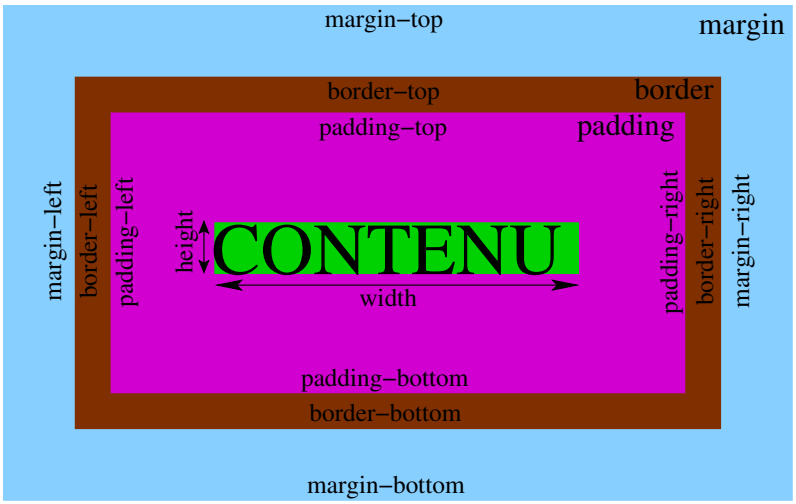


FIGURE 2.1 – Dimensions, bordures et marges [2]

**border** permet de spécifier la bordure d'un bloc. La valeur est constituée de trois éléments : la taille de la bordure, le style et la couleur. La taille est spécifiée par `thin`, `medium`, `thick` ou par une valeur en pixels ou en `em`. Le style vaut `solid` (trait plein), `double` (double trait), `inset` ou `outset` (effet 3D) ou `none` (pas de bordure). La couleur est spécifiée de la même manière que pour l'attribut `color`. La bordure d'un paragraphe correspondant à un trait plein de 1 pixel de couleur bleu est alors donnée par :

```
p
{
  border: 1px solid blue;
}
```

Cette propriété prend la valeur `none` s'il n'y a pas de bordure. (*Valeur par défaut :* `none`)

Si l'on souhaite donner différentes valeurs pour les bordures haut, bas, gauche et droite, il faut alors utiliser les propriétés `border-top`, `border-bottom`, `border-left` ou `border-right`.

**border-radius** permet de définir des coins arrondis pour la bordure d'un élément. La valeur indique la courbure de l'angle et est exprimée en pixels (px), en taille relative (em) ou en pourcentage.

**margin** permet de définir la taille de la marge extérieure en pixels (px), en taille relative (em) ou en pourcentage. Si l'on souhaite donner différentes tailles pour les marges extérieures haut, bas, gauche et droite, il faut alors utiliser les propriétés `margin-top`, `margin-bottom`, `margin-left` ou `margin-right`. La valeur de la taille d'une marge peut également être la valeur `auto`. Dans ce cas, on laisse le navigateur déterminer automatiquement cette taille. Si les marges gauches et droites sont déterminées par le navigateur, alors ce dernier centre l'élément. (Il faut cependant que la largeur (`width`) de l'élément ait été fixée.) Donc, pour centrer les tableaux dans une page Web, il suffit d'ajouter dans le code CSS :

```
table
{
```

2. Image basée sur une image provenant du cours de Marcel BOSC.

```
    width : 500px;
    margin-left : auto;
    margin-right : auto;
}
```

Pour les éléments de type `inline`, seules les marges extérieures gauche et droite peuvent être définies. (*Valeur par défaut : dépend des côtés (haut, bas, gauche, droite) et des éléments*)

*Dans certains cas, les marges verticales (`margin-top` et `margin-bottom`) de deux éléments consécutifs sont* fusionnées *: l'espace entre les bordures de ces deux éléments correspond au maximum de leur marge extérieure, et non à leur somme. Cette fusion de marges peut également avoir lieu avec un élément et son premier enfant, et un élément avec son dernier enfant. Pour plus d'information, lire l'article :*
*`https://www.alsacreations.com/article/lire/629-fusion-des-marges.html`.*

**padding** permet de définir la taille de la marge intérieure en pixels (px), en taille relative (em) ou en pourcentage. Si l'on souhaite donner différentes tailles pour les marges intérieures haut, bas, gauche et droite, il faut alors utiliser les propriétés `padding-top`, `padding-bottom`, `padding-left` ou `padding-right`.

Pour les éléments de type `inline`, seules les marges intérieures gauche et droite peuvent être définies. (*Valeur par défaut :* `0` *sauf pour la marge gauche pour les items d'une liste*)

**Dimensions et marges** La propriété `box-sizing` détermine comment sont prises en compte les marges intérieures (`padding`) et les bordures dans les dimensions. Si la valeur est `content-box`, les valeurs `height` et `width` correspondent à la taille du contenu uniquement. Si la valeur est `border-box`, les valeurs `height` et `width` tiennent compte des dimensions du contenu, des marges intérieures et de la taille de la bordure.

À titre d'exemple, si un élément a une `width` de 100 pixels, une marge intérieure de 20 pixels et une bordure de 5 pixels, la largeur prise par cet élément dans la page est de 150 pixels si `box-sizing` est égal à `content-box` et de 100 pixels autrement. Dans les deux cas, il faut ajouter les marges extérieures.

Un exemple interactif se trouve à l'adresse `https://codepen.io/mathieulacroix/pen/PdKLxq`.

### 2.2.4 Modification du fond

Il est possible de modifier la page Web en choisissant une couleur de fond (ou arrière-plan), ou une image de fond, pour chaque balise.

**background-color** permet de spécifier une couleur de fond. Les valeurs possibles sont les mêmes que pour la propriété `color` plus la valeur `transparent`. (*Valeur par défaut :* `transparent`)

**background-image** permet de spécifier une image de fond. La valeur est alors `url("image")` où `image` correspond aux nom et chemin du fichier image. Cette propriété prend la valeur `none` s'il n'y a pas d'image de fond. (*Valeur par défaut :* `none`)

**background-repeat** permet d'indiquer, lorsque l'image de fond est plus petite que la taille de l'élément, si l'image doit être répétée ou non. Les différentes valeurs possibles sont :
- `repeat-x` : l'image doit être répétée horizontalement,
- `repeat-y` : l'image doit être répétée verticalement,
- `repeat` : l'image doit être répétée horizontalement et verticalement,
- `no-repeat` : l'image ne doit pas être répétée.

(*Valeur par défaut :* `repeat`)

**background-position** permet de définir la position de l'image de fond par rapport à l'élément. On donne deux valeurs pour indiquer l'emplacement horizontal (`left`, `right` ou `center`) et vertical (`top`, `bottom` ou `center`). Il est également possible de donner une distance par rapport aux bords en pixels ou en pourcentage.

Un exemple interactif se trouve à l'adresse `https://codepen.io/mathieulacroix/pen/rZzRvY`.

### 2.2.5 Modifications liées aux listes et tableaux

La première propriété est liée aux tableaux, les deux suivantes aux listes.

**border-collapse**

permet de spécifier si les cellules d'un tableau sont collées (valeur `collapse`) ou non (valeur `separate`). (*Valeur par défaut :* `separate`)

**list-style-type**

définit le type de puces utilisées dans les listes non ordonnées : carré (valeur `square`), cercle (valeur `circle`), cercle plein (valeur `dic`) ou aucune puce (valeur `none`). Cette propriété permet également de spécifier la numérotation d'une liste ordonnée : chiffres décimaux 1,2,.. (valeur `decimal`), chiffres romains en majuscules (valeur `upper-roman`), chiffres romains en minuscules (valeur `lower-roman`). (*Valeur par défaut :* `decimal` *pour les listes ordonnées et* `disc` *pour les listes non ordonnées*)

**list-style-image**

permet d'utiliser une image pour représenter la puce d'une liste non ordonnée. La valeur est donnée par `url("fichierImage")` où "fichierImage" représente le nom de l'image et son chemin. Cette propriété prend la valeur `none` s'il n'y a pas d'image de fond. Si une image est spécifiée, alors aucune puce n'est affichée en plus de l'image, quelle que soit la valeur de la propriété `list-style-type`. (*Valeur par défaut :* `none`)

### 2.2.6 Modification du type de la balise

Il est possible de transformer une balise de type `block` en une balise de type `inline` et inversement. Pour cela, il faut utiliser la propriété `display` qui prend les valeurs suivantes :

- `block` : l'élément devient de type `block`,
- `inline` : l'élément devient de type `inline`,
- `inline-block` : l'élément est de type `inline`, mais on peut lui appliquer les propriétés réservées aux balises de type `block`, permettant par exemple de spécifier une hauteur et une largeur.
- `none` : l'élément n'est pas affiché. Ceci revient à supprimer complètement l'élément dans le code HTML.

(*Valeur par défaut : correspondant au type de la balise*)

### 2.2.7 Spécification de l'affichage en cas de dépassement

L'espace nécessaire pour afficher le contenu d'un élément peut parfois être supérieur aux dimensions (largeur et hauteur) de cet élément. Dans ce cas, il est possible de spécifier comment afficher le contenu qui dépasse de l'élément avec la propriété `overflow`. Si cette propriété a pour valeur `visible`, tout le contenu dépassant de l'élément sera affiché (Il y a alors un risque de chevauchement). Si elle a pour valeur `hidden`, le contenu dépassant de l'élément ne sera pas affiché. Si la valeur est `auto`, une barre de défilement sera ajoutée si cela est nécessaire. Dans ce cas, le contenu ne dépassera pas les dimensions de l'élément. Il faudra utiliser la barre de défilement pour voir tout le contenu. (*Valeur par défaut :* `visible`)

À titre d'exemple, considérons le code HTML suivant

```
<body>
  <ul>
  <li> Premier point </li>
  <li> Deuxième point </li>
  <li> Troisième point </li>
  <li> Quatrième point </li>
  </ul>
</body>
```

associé au code CSS suivant

```
ul
{
  width : 100px;
  height : 80px;
  border : 1px solid red;
}
```

Clairement, la liste non ordonnée ne peut pas s'afficher entièrement dans un espace de 100 * 80 pixels. La gestion de l'affichage de la liste dépend alors de la propriété `overflow`, comme montré dans la figure 2.2.

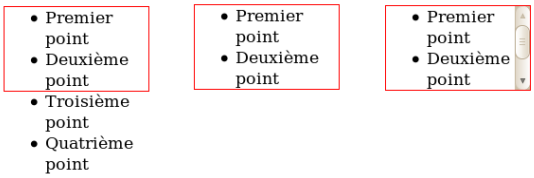


FIGURE 2.2 – Affichage de la liste avec la propriété `overflow` égale à `visible`, `hidden` puis `auto`.

## 2.3 Classes et identifiants

### 2.3.1 Utilisation

Jusqu'à présent, les sélecteurs étaient uniquement des balises et permettaient de modifier l'apparence de tous les éléments d'un même type (paragraphe, titre, etc). Pour modifier l'apparence de seulement quelques éléments d'un même type, on utilise les classes. Une classe est un nom qui peut servir de sélecteur en CSS (la classe doit être précédée du caractère '.'). On indique ensuite dans le code HTML quels éléments appartiennent à cette classe en utilisant l'attribut `class` selon `<balise class="nom_classe">`. Ces derniers sont alors modifiés par les règles associées à cette classe dans le CSS.

Pour afficher en bleu uniquement certains paragraphes, on leur ajoute l'attribut `bleu` et on ajoute dans le code CSS la règle suivante :

```
.bleu
{
  color : blue;
}
```

Il est également possible certaines fois d'utiliser les identifiants plutôt que les classes. Pour spécifier qu'un élément a pour identifiant `nomId`, il suffit de rajouter dans la balise HTML l'attribut `id` selon `<balise id="nomId">`. Les spécifications de l'identifiant en CSS se font de la même manière que pour une classe, excepté que le nom de l'identifiant doit être précédé du caractère '#' plutôt que du point.
Bien que les identifiants et les classes soient similaires, il existe une différence fondamentale : chaque identifiant peut être associé à un unique élément, c'est-à-dire qu'il ne peut y avoir deux balises dans un fichier HTML ayant la même valeur de l'attribut `id`. L'identifiant sert justement à désigner un élément unique. L'identifiant peut être utilisé dans les url (pour indiquer un endroit spéceficque de la page Web) et en javascript.

R *Un élément peut appartenir à plusieurs classes. L'attribut `class` contient alors les noms des différentes classes (séparés par des espaces) auxquelles appartient l'élément : `<balise class="classe1 classe2 classe3">`.*

Un exemple interactif se trouve à l'adresse `https://codepen.io/mathieulacroix/pen/eLGzYd`.

### 2.3.2 Les pseudo-classes

Une pseudo-classe est un mot-clé qui peut être ajouté à un sélecteur afin d'indiquer l'état spécifique dans lequel l'élément doit être pour être ciblé par la règle CSS. Le sélecteur avec la pseudo-classe s'écrit : `sélecteur:pseudo_classe`.

Il existe beaucoup de pseudo-classes dont les quatre plus importantes sont :
— `:hover` indique que l'élément doit être survolé par la souris.
— `:nth-child(n)` indique que l'élément doit être le $n^e$ enfant de son parent.
— `:active` indique que l'élément doit être cliqué avec la souris (dans le CSS, `sélecteur:active` doit être défini avant `sélecteur:hover` pour que la règle soit prise en compte).

— `:visited` indique que l'utilisateur doit déjà avoir visité ce lien (cette pseudo-classe ne s'applique qu'aux hyperliens).

Un exemple interactif pour `:nth-child(n)` se trouve à l'adresse : `https://codepen.io/mathieulacroix/pen/zJENxZ`.

### 2.3.3 Balises HTML span et div

Il est parfois nécessaire d'appliquer une classe ou un identifiant à une portion de code HTML qui n'est pas délimitée par des balises. Dans ce cas, on délimite cette portion de code en ajoutant une balise universelle qui n'a pas de signification HTML. Il existe deux types de balises universelles : la balise `<span>` de type `inline` et la balise `<div>` de type `block`.

Si l'on souhaite écrire en bleu les noms de villes d'un texte HTML, il suffit alors de délimiter chaque nom de ville par la balise `<span>` avec l'attribut `class="bleu"` (en supposant que le CSS contienne la règle définie précédemment) :

```
<p><span class="bleu">Paris</span>, <span class="bleu">Lyon</span> et
<span class="bleu">Marseille</span> sont les trois plus grandes villes de
France.</p>
```

## 2.4 Les différents sélecteurs et modifications en cascades

### 2.4.1 Combinaisons de sélecteurs

Jusqu'à présent, les sélecteurs étaient des balises (e.g., `p`, `h1`, etc), des balises avec des pseudo-classes, des classes ou des identifiants. Il est cependant possible de créer des sélecteurs par combinaison. Voici les principales combinaisons :

| Combinaison | Explication |
|---|---|
| E | correspond à tout élément E. |
| E F | correspond à tout élément F qui est un descendant de l'élément E. |
| E > F | correspond à tout élément F qui est un enfant de l'élément E. |
| E.maClasse | correspond à tout élément E appartenant à la classe "maClasse". |
| E#myid | correspond à tout élément E dont l'identifiant est "myid". |
| E, F | les modifications s'appliquent aux éléments E et aux éléments F. |

Il est possible d'utiliser ces différentes combinaisons entre elles pour spécifier davantage la cible des modifications. Ainsi, les propriétés associées au sélecteur `#abc p.rouge > a:hover` s'appliquent uniquement aux liens survolés par la souris qui sont enfants de paragraphes appartenant à la classe `"rouge"`, eux-même descendants d'élements dont l'identifiant est `"abc"`. De même, les propriétés associées au sélecteur `p > em.rouge.Italique , h1 .bleu` s'appliquent uniquement aux éléments mis en valeur par la balise `em` appartenant aux classes `"rouge"` et `"Italique"` et enfants de paragraphes, ou alors aux élements appartenant à la classe `"bleu"` et descendants de `h1`.

(!) *Les sélecteurs `p.rouge` et `p .rouge` sont différents ! Le premier n'applique les modifications d'apparence que pour les paragraphes appartenant à la classe `"rouge"`, alors que le second modifie l'apparence des éléments appartenant à la classe `"rouge"` et descendants d'un paragraphe.*

(R) *Le sélecteur `*` est un sélecteur qui représente n'importe quel élément. Il peut être combiné avec d'autres sélecteurs. Le sélecteur `h1 > *` applique la règle à tous les fils de `h1`. Il peut également servir seul pour modifier le comportement de tous les éléments par rapport à une propriété : `* { box-sizing : border-box}`.*

Un exemple interactif se trouve à l'adresse `https://codepen.io/mathieulacroix/pen/zJPvgR`.

### 2.4.2 Styles en cascade

Suivant les sélecteurs, plusieurs règles peuvent s'appliquer à un même élément. Lorsque les propriétés définies dans ces différentes règles sont toutes différentes, elles s'appliquent toutes à l'élément. C'est ce que l'on appelle le style en cascade. Ainsi, si dans le code HTML, on a :

```
<div id="monId"> <p class="rouge"> ... </p> </div>
```

et que le fichier CSS associé contient les règles

```
p
{
  font-size : small;
}

.rouge
{
  color : red;
}

#monId p
{
  text-align : right;
}
```

le paragraphe sera alors écrit en rouge, aligné à droite et avec la taille `small`. En effet, les différentes règles s'appliquent car le paragraphe est un paragraphe, il appartient à la classe `"rouge"` et est également un paragraphe ayant un ascendant dont l'identifiant est `"monId"`.

### 2.4.3 Priorité des sélecteurs

Les styles en cascade permettent d'utiliser plusieurs règles pour un même élément. Que se passe-t-il si deux règles indiquent deux valeurs différentes pour une même propriété ? Par exemple, si l'on a toujours le même code HTML que précédemment mais que le fichier CSS contient les règles

```
.rouge
{
  color : red;
}

#monId p
{
  color : green;
}
```

le paragraphe s'affiche-t-il en rouge ou en vert ? Afin qu'il n'y ait aucune ambiguïté, les règles s'appliquent selon une priorité. Ainsi, si deux règles s'appliquent à un même élément avec deux valeurs différentes pour une même propriété, alors la valeur de la propriété utilisée pour l'élément sera celle donnée par la règle de priorité la plus élevée.

La priorité d'une règle est donnée par la priorité du sélecteur associé. Rapidement, plus le sélecteur est spécifique, plus sa priorité est élevée. Dans notre exemple, `#monId p` est plus spécifique que `.rouge`, ce qui implique que le paragraphe sera écrit en vert.

De manière plus précise, la priorité entre deux sélecteurs est déterminée de la manière suivante. On associe à chaque sélecteur un nombre à trois chiffres :

— le chiffre des centaines correspond au nombre d'identifiants dans le sélecteur,
— le chiffre des dizaines correspond au nombre de classes et pseudo-classes dans le sélecteur,
— le chiffre des unités correspond au nombre d'élements (autres qu'identifiants, classes et pseudo-classes) dans le sélecteur.

Par exemple, le sélecteur `#monId p` est associé au nombre 101 et `.rouge` au nombre 010. Le sélecteur `#monId div.classe1 h2#monId2.classe2 em a` est associé au nombre 224. Le sélecteur qui possède le plus grand nombre est prioritaire. En cas d'égalité, la priorité est donnée à la dernière règle apparaissant dans le fichier CSS.

Un exemple interactif se trouve à l'adresse `https://codepen.io/mathieulacroix/pen/PdOzjV`.

## 2.5 Héritage

### 2.5.1 Valeur par défaut et héritage

Si aucune valeur n'est précisée pour une propriété dans le CSS, la valeur pour cette propriété est celle par défaut. Ainsi, si la propriété `border` n'est pas spécifiée dans le CSS, il n'y a aucune bordure à l'affichage puisque la propriété `border` associée à n'importe quel élément prend la valeur par défaut `none`.

Cependant, les valeurs de certaines propriétés peuvent être héritées, c'est-à-dire que pour ces propriétés (telles que `color`), si aucune valeur n'est spécifiée pour un élément, la valeur est alors celle de l'élément parent plutôt que la valeur par défaut. Ainsi, si le code CSS contient juste

```
body
{
  color : red;
}
```

alors tout sera écrit en rouge (exception faite des hyperliens). En effet, pour chaque élément qui n'est pas un lien, aucune valeur n'est précisée pour la propriété `color`. La valeur associée à cette propriété est alors égale à la valeur de `color` pour l'élément parent.

Considérons le code HTML

```
<p class="rouge"> Du texte <em> mis en valeur </em> puis <strong> du texte
fortement mis en valeur. </strong> </p>
```

et le code CSS

```
.rouge { color : red; }

em { color : blue; }
```

La première règle s'applique au paragraphe mais elle ne s'applique pas au texte qui se trouve entre les balises `<em>` et `</em>`, ni à celui se trouvant entre les balises `<strong>` et `</strong>`. Les parties `Du texte` et `puis` sont donc écrits en rouge grâce à la première règle. Le texte entre les balises `<em>` et `</em>` est écrit en bleu car seule la deuxième règle du CSS s'applique. Finalement, pour le texte compris entre les balises `<strong>` et `</strong>`, aucune règle ne s'applique. Puisque la propriété `color` peut être héritée, la valeur de `color` pour ce texte est donc égale à celle de son parent. Comme l'élément parent de `strong` est `p.rouge`, et que la valeur associée à `color` est `red`, la valeur de `color` pour la balise `strong` est `red` et le texte compris entre `<strong>` et `</strong>` est donc écrit en rouge.

Les propriétés qui peuvent être héritées sont les propriétés relatives à la police de caractères (section 2.2.1) et les propriétés `border-collapse` et `list-style`. Les autres propriétés présentées dans ce cours ne peuvent pas être héritées.

R *Les liens sont écrits en bleu car le navigateur ajoute automatiquement la règle* `a { color:blue; }` *au tout début du code CSS. La propriété* `color` *ne peut donc pas être héritée pour les liens. De la même manière, le navigateur ajoute automatiquement que le texte mis en valeur par la balise* `em` *est écrit en italique et que celui mis en valeur par la balise* `strong` *est écrit en gras. De plus, les valeurs de* `font-weight` *et* `font-size` *pour les titres sont aussi ajoutés.*

R *La propriété* `text-decoration` *ne suit pas les mêmes règles que les autres. En effet, dès qu'une valeur de* `text-decoration` *est spécifiée pour un élément, il n'est plus possible d'enlever la décoration pour les descendants.*

### 2.5.2 Valeur `inherit`

Toutes les propriétés acceptent également la valeur `inherit`. Dans ce cas, la valeur pour cette propriété est exactement la même que celle de l'élément parent. Cette valeur est possible même pour les propriétés qui ne peuvent a priori pas être héritées.

## 2.6 Positionnement

Le positionnement des éléments sur la page Web est réalisé en CSS de plusieurs manières. On présente ici le positionnement via la propriété `position` et le positionnement par grille. La propriété `float` et les `flexbox` ne sont pas présentés.

### 2.6.1 Propriété `position`

Les éléments peuvent être positionnés différemment suivant les valeurs de la propriété `position`.

#### Positionnement statique

Par défaut, les éléments d'une page Web sont affichés dans l'ordre où ils apparaissent dans le code HTML. Cet ordre est appelé *flux* de la page HTML. Si la page HTML contient

```
<h1> titre </h1>
<p> paragraphe avec du texte <em> mis en valeur </em> et du texte <strong>
fortement mis en valeur </strong> </p>
```

Le navigateur affiche d'abord le titre, puis le paragraphe. Dans le paragraphe, il affiche d'abord le texte mis en valeur puis le texte mis fortement en valeur. De plus, il revient à la ligne avant et après tout élément de type `block`. Par ailleurs, chaque élément de type `block` prend par défaut toute la largeur qu'il peut, c'est-à-dire toute la largeur de son conteneur.

Tous les objets positionnés selon le flux sont dits positionnés de manière statique. La valeur de la propriété `position` est alors `static`. C'est la valeur par défaut de cette propriété.

#### Positionnement Relatif

La valeur `relative` de la propriété `position` permet de déplacer l'élément par rapport à la position qu'il avait dans le flux. Le déplacement est spécifié par les propriétés `top`, `bottom`, `left` et `right` qui prennent comme valeur une distance donnée en pixels ou en pourcentage. Ce déplacement ne modifie pas l'affichage des autres éléments du flux. De plus, l'élément garde les mêmes dimensions et marges que s'il avait été positionné avec la valeur `static`.

Considérons le code HTML

```
<body>
<p> Paragraphe 1 </p>
<ul>
  <li> Item 1 </li>
  <li> Item 1 </li>
</ul>
<p> Paragraphe 2 </p>
```

et le code CSS

```
body
{
  width : 200px;
  height : 200px;
  border : 2px dashed black;
  padding : 0;
  margin : 50px;
}

ul
{
  border : 1px solid silver;
  margin : 0;
  background-color : silver;
}

p
{
```

```
    background-color : #717171;
    border : 1px solid black;
```

La figure 2.3 montre l'affichage obtenu lorsque la liste non ordonnée est affichée avec la propriété `position:static` puis avec `position:relative` et les propriétés `left:50px;` et `top:50px;`.

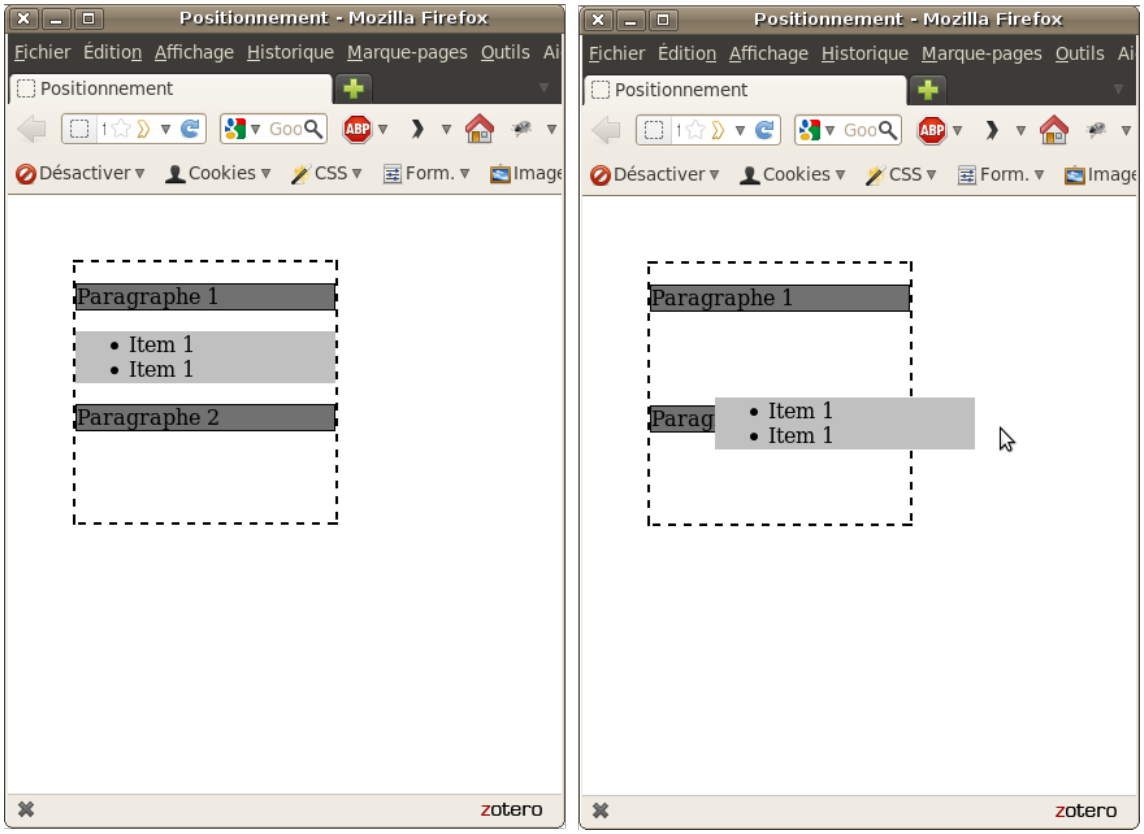


FIGURE 2.3 – Propriété `position` égale à `static` puis `relative`.

Un exemple interactif se trouve à l'adresse `https://codepen.io/mathieulacroix/pen/PdOGaz`.

### Positionnement Absolu

Lorsqu'un élément a `position:absolute;`, il est d'abord retiré du flux. Tous les éléments du flux sont ensuite affichés. Finalement, l'élément positionné de manière absolue est ajouté par la suite. Par défaut, cet élément est placé à l'endroit où il aurait été placé dans le flux. Pour modifier son placement, on utilise les propriétés `top`, `bottom`, `left` ou `right` qui permettent de définir la distance par rapport au premier ascendant qui n'est pas positionné de manière statique. Si tous les ascendants sont positionnés de manière statique, les distances sont données par rapport aux bords du navigateur.

La largeur par défaut d'un élément positionné de manière absolue est égale au minimum pour afficher le contenu de l'élément.

La figure 2.4 montre l'affichage obtenu lorsque la liste non ordonnée est affichée avec les propriétés `position:absolute;`, `right:10px;` et `bottom:10px;`. Dans le premier cas, tous les autres éléments sont positionnés de manière statique. La liste est donc positionnée par rapport aux bords du navigateur. Dans le second cas, la balise `body` est positionnée de manière relative (sans déplacement). Le positionnement de la liste se fait alors par rapport aux bords de la balise `body`.

Deux exemples interactifs se trouvent aux adresses :
— `https://codepen.io/mathieulacroix/pen/mjZxVx`,
— `https://codepen.io/mathieulacroix/pen/BPXaMe`.

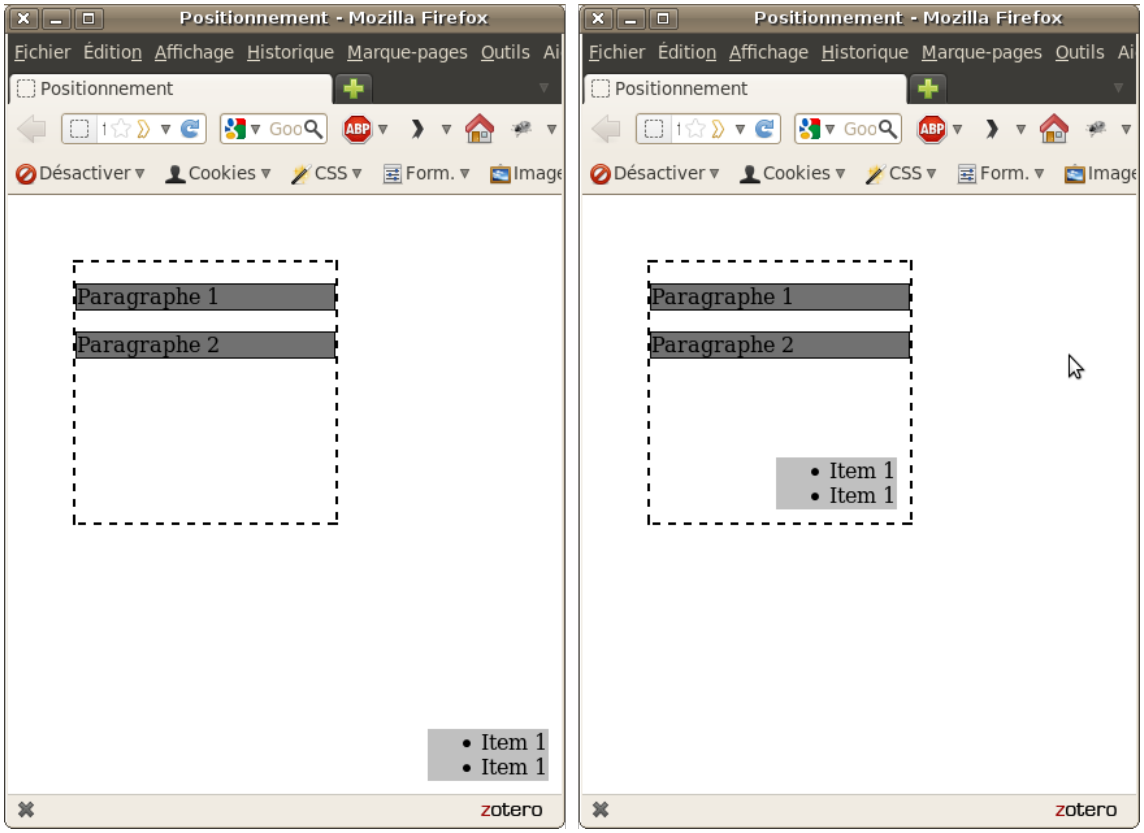


FIGURE 2.4 – Propriété `position` pour `body` égale à `static` puis `relative`.

#### Positionnement fixe

Le positionnement fixe (`position:fixed;`) d'un élément est équivalent au positionnement absolu de cet élément, à la différence que le positionnement se fait toujours par rapport aux bords du navigateur. De plus, l'élément reste à la même position, même en cas de défilement de la page Web.

Un exemple interactif se trouve à l'adresse `https://codepen.io/mathieulacroix/pen/EebZxg`.

### 2.6.2 Disposition en grille

Il est possible de créer une mise en page à l'aide de grilles (ou tableaux). Concrètement, il s'agit de modifier un élément (appelé par la suite *conteneur*) pour qu'il se comporte comme un tableau et de ranger ensuite ses enfants dans les cases de ce tableau. Ceci se fait uniquement en CSS sans modifier le HTML.

On indique que le conteneur a un comportement de grille à l'aide de la propriété `display:grid`. On indique ensuite le nombre de lignes et de colonnes à l'aide des propriétés `grid-template-rows` et `grid-template-columns`. La valeur pour chacune de ces propriétés est la liste des tailles (largeur ou hauteur) des lignes (ou colonnes).

Par exemple, pour définir un conteneur (avec la classe `conteneur`) ayant 3 lignes de 100 pixels et 4 colonnes (les deux premières de 200 pixels, les deux autres de 100 pixels), on écrit alors le code CSS

```
1 .conteneur{
2   display : grid;
3   grid-template-rows : 100px 100px 100px;
4   grid-template-columns : 200px 200px 100px 100px;
5 }
```

R *On peut utiliser `repeat` pour repéter plusieurs fois un motif dans la séquence de taille. Ainsi, la ligne 3 du code précédent aurait pu être remplacée par `grid-template-rows : repeat(3,100px);`*

*Il est possible de définir un espace entre les colonnes et/ou lignes grâce aux propriétés* `row-gap` *et* `column-gap`*. Sur les navigateurs moins récents, il faut utiliser les anciens noms* `grid-row-gap` *et* `grid-column-gap`*.*

Pour spécifier les tailles, il existe une nouvelle unité de mesure : `fr` (pour free space). Cette unité permet de partager l'espace restant dans le conteneur en autant de `fr` qu'indiqué. Par exemple, la ligne `grid-template-columns : 1fr 3fr 200px;` crée 3 colonnes. La dernière a une largeur de 200 pixels. Les deux premières colonnes se partagent l'espace libre restant (largeur du conteneur - 200 pixels) sachant que la deuxième colonne est 3 fois plus large que la première.

On place ensuite les enfants du conteneur dans des zones (ensembles de cases formant un rectange) de la grille. Pour faciliter le placement, on nomme les zones dans le CSS du conteneur grâce à la propriété `grid-template-areas`. La valeur de cette propriété est une suite de chaînes de caractères, une par ligne de la grille. Chaque chaîne contient pour chaque case de la ligne le nom de la zone à laquelle appartient cette case (symbole `.` si la case n'appartient à aucune zone). On indique ensuite pour chaque enfant du conteneur dans quelle zone il s'affiche grâce à `grid-area:nom_de_la_zone`.

À titre d'exemple, considérons le code html

```
<div class="conteneur">
  <p>Zone 1</p>
  <p>Zone 2</p>
  <p>Zone 3</p>
  <p>Zone 4</p>
  <p>Zone 5</p>
</div>
```

Ajoutons dans la règle CSS précédente (sélecteur `.conteneur`) les propriétés

```
row-gap:20px;
column-gap:20px;
grid-template-areas:
   "zone1 zone1 zone2 zone2"
   "zone3 zone4 zone4 ."
   "zone3 zone4 zone4 zone5";
```

On a alors créé une grille contenant cinq zones dont la première correspond aux deux premières cases de la première ligne. Ajoutons dans le code CSS les règles attribuant à chaque enfant du conteneur une zone (premier enfant dans la première zone, etc)

```
  p:nth-child(1){grid-area:zone1;}
  p:nth-child(2){grid-area:zone2;}
  p:nth-child(3){grid-area:zone3;}
  p:nth-child(4){grid-area:zone4;}
  p:nth-child(5){grid-area:zone5;}
```

On obtient alors le rendu de la figure 2.5 [3].

*Il n'est pas obligatoire de spécifier la propriété* `grid-template-rows`*. Chaque ligne prend alors la hauteur minimale nécessaire pour afficher le contenu de la ligne.*

Deux exemples interactifs se trouvent aux adresses :
— `https://codepen.io/mathieulacroix/pen/mGqaae`,
— `https://codepen.io/mathieulacroix/pen/aabmGL`.

## 2.7 Responsive design

Un site Web Responsive est un site Web qui permet d'être consulté de manière optimale (expérience utilisateur satisfaisante) quel que soit l'appareil (ordinateur, tablette, smartphone, etc). La mise en page (placement des éléments, taille de la police, etc) d'un site Responsive varie selon la taille de l'écran. Cela est possible en utilisant la règle CSS

3. Du css supplémentaire pour la couleur de fond et le style d'écriture des paragraphes est nécessaire pour obtenir ce rendu.

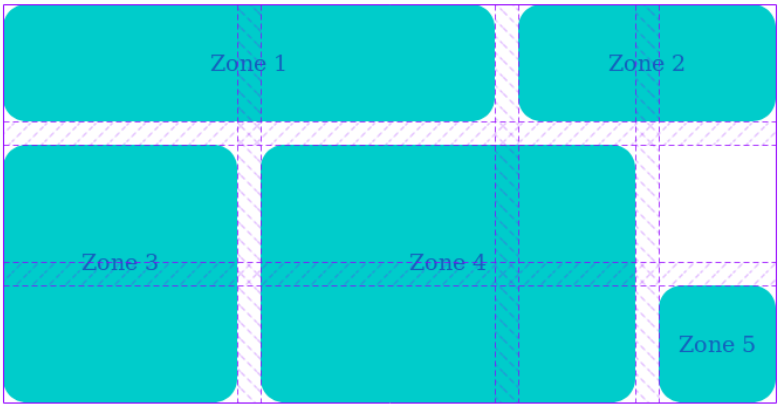


FIGURE 2.5 – Rendu de la disposition en grille. Les lignes et hachures en violet proviennent du module développement Web de Firefox.

```
@media (condition) {
  /* Règles CSS qui sont prises en compte si la condition est vérifiée */
}
```

La condition est définie par `max-width` ou `min-width` (ou une conjonction des deux) suivi d'une taille en pixels. Les règles CSS entre les accolades sont prises en compte si et seulement si l'écran (ou la fenêtre du navigateur) satisfait cette condition.

Cela permet en autres de modifier l'agencement de la page Web selon la taille d'écran. Si l'écran est petit, alors les éléments doivent s'afficher les uns en dessous des autres (comportement normal d'affichage). Si l'écran est assez grand, alors on peut créer une grille pour afficher certains éléments les uns à côté des autres.

Pour que cela fonctionne sur tous les appareils, il faut également ajouter dans le `<head>` du document HTML la balise

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Autrement, il y a des problèmes de zoom sur la page (c.f. `https://developer.mozilla.org/fr/docs/Mozilla/Mobile/Balise_meta_viewport` pour plus d'informations).

Un exemple interactif se trouve à l'adresse `https://codepen.io/mathieulacroix/pen/oPoRVY`.

## Références

Vous pouvez trouver de nombreux cours et tutoriels sur le Web. Voici quelques liens intéressants :

— le site du Mozilla pour les développeurs (`https://developer.mozilla.org/fr/docs/Web`),
— Alsacréations (`http://www.alsacreations.com/`),
— CSS Zen Garden (`http://www.csszengarden.com/`) montrant de nombreux exemples de fichiers CSS (pour un même fichier HTML).