RÉALISER UN SITE WEB AVEC HTML5 ET CSS3

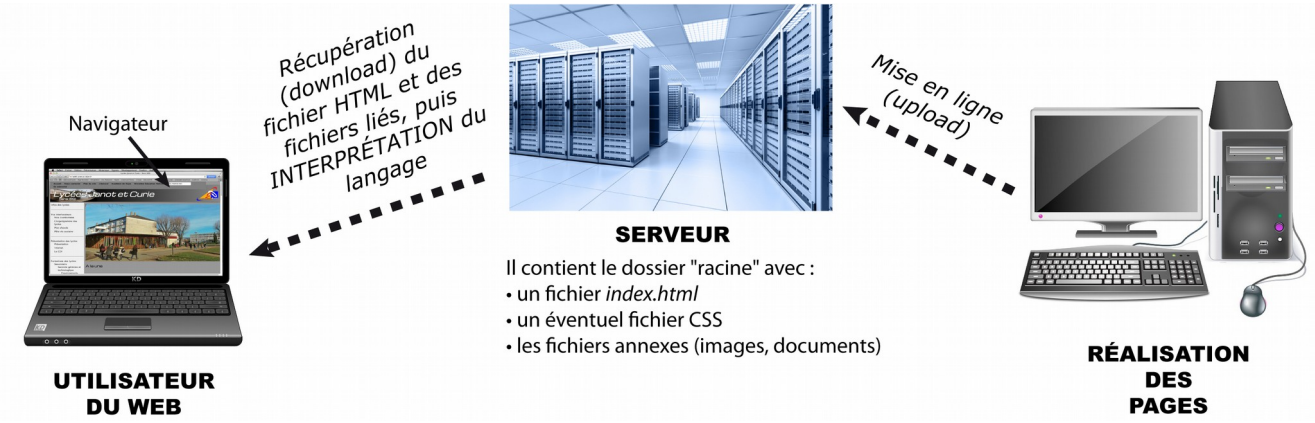
(Cours réalisé à partir du travail original de S. MOGENIER et E. CHAVANTON)

1. Un site internet « moderne » : accessible et adaptable.

Un site « web » moderne doit concilier plusieurs caractéristiques : être accessible à tous (quel que soit le navigateur , quelle que soit la personne, quel que soit le média de visualisation), être facile à référencer par les moteurs de recherche (sémantique de qualité) et facile à mettre à jour ou faire évoluer.
La solution existe : HTML (HyperText Markup Language) pour le contenu (l'information brute), aujourd'hui dans sa version 5, et CSS (Cascading Style Sheet) pour la mise en page et le design (CSS3 aujourd'hui).

2. Qu'est-ce qu'un site internet ?

Il se compose d'un ensemble de pages (qui sont des fichiers texte enregistrés au format HTML de suffixe .htm) liées entre elles par des **liens hypertexte**.
Un fichier HTML décrit le contenu d'une page du site ; le navigateur (Firefox, IE, Safari, Opera, etc.) télécharge puis « lit » ce fichier, en interprète les instructions et reconstitue la page (en téléchargeant aussi les fichiers liés : images, fontes, etc.)
Chaque navigateur peut interpréter à sa façon le code HTML, ce qui explique les différences de rendu.
Le **W3C** (World Wide Web Consortuim) essaie de normaliser les instructions pour limiter les écarts d'interprétation. Le langage évoluant, certaines instructions ne sont pas encore implémentées dans tous les navigateurs.



3. Où placer ses fichiers ?

Toutes les pages, images, feuilles de style doivent être placées dans un dossier unique (dossier « racine ») pour une organisation simple et cohérente.
Le fichier HTLM, ouvert automatiquement par un navigateur (lorsque le site est sur un serveur distant en particulier), doit être intitulé `index.htm` ; tous les autres documents seront reliés à ce fichier de référence par des liens.

Pour éviter tout problème avec les liens, on utilisera uniquement des liens relatifs (et non pas absolus) dans le code HTML.
Exemple : si dans le code de la page *index.htm* on a l'instruction suivante : `<a href='images/photo1.jpg'>`, le lien mènera vers l'image intitulée « *photo1.jpg* » qui se trouve dans le dossier « *images* » qui doit être dans le même dossier *index.htm*, quel que soit l'endroit où se trouve ce dossier. C'est un lien relatif (relatif à l'emplacement de *index.htm*). Les liens absolus seront réservés aux liens externes (par exemple `<a href='http://www.tonsiteweb.com/index.htm'>`.

4. Le principe

La conception d'un site web est partagée en deux taches :

- La réalisation du **contenu** du site : les informations proprement dites, ce qui est lisible et informatif (y compris les images). Ces informations sont dans le fichier HTML.
- La réalisation du **design** du site. Ces informations sont dans le (ou les) fichier(s) CSS.

Site Web sans CSS appliqué

Site Web avec CSS

5. Le contenu du site : fichier HTML

5.1. Structure du fichier HTML

Exemple : Un fichier HTML5

```
<!DOCTYPE html>

<html>

<head> <!-- N'apparaît pas sur la page web -->
  <meta charset="utf-8" />
  <title> Page 1 </title>    <!—titre qui s'affiche dans l'onglet-->
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>


<body>
  <div id="wrapper">    <!-- L'enveloppe de la partie centrale de ma page -->
    <header>
      <h1>Mon Site Web</h1>
    </header>
    <article>
      <p>Hello World</p>
      <p class='centre'>C'est mon site !</p>
      <p><a href='page2.htm'>Cliquez ici </a>pour voir la page 2</p>
      <img src="photo.jpg" width="200" height="230" alt='Photo'>
    </article>
    <footer>© 2013</footer>
  </div>
</body>
</html>
```

Doctype ***Balises en paires*** ***Commentaire*** ***Head*** ***Paragraphe*** ***Body*** ***Attributs*** ***Balises orphelines***

La 1ère information à donner en début de document HTML est le **doctype** : il annonce au navigateur quelle version de HTML est utilisée. `<!DOCTYPE>` pour du HTML version 5.

Le document doit se poursuivre par `<html>` qui comporte deux blocs :
• le bloc `<head>` qui contient les informations pour le navigateur et qui n'apparaîtront pas sur la page web (L'encodage qui précise le jeu de caractère utilisé (accents, glyphes spéciaux), es « méta balise » qui sont les infos destinées aux moteurs de recherche, la désignation des ressources externes (feuilles de style CSS, fontes, etc.))

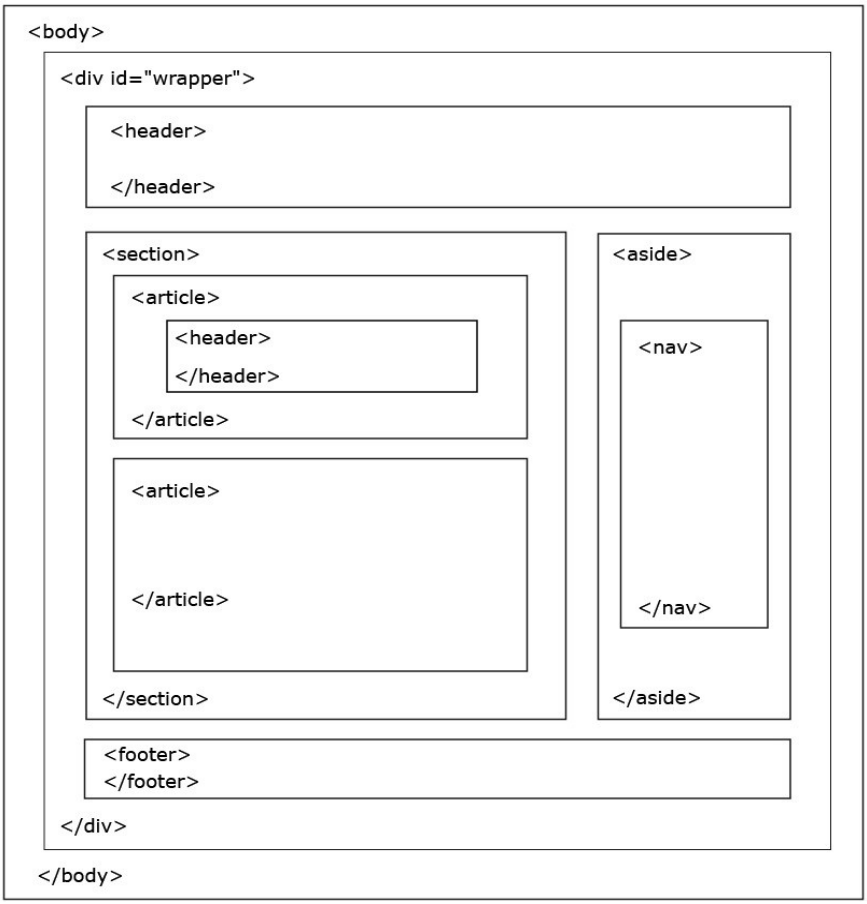
• le bloc `<body>` qui contient tout le contenu affiché sur la page web.

5.2. Balises de contenu

Une page web peut être considérée comme un empilement de « *boîtes* » ou *divisions* ; chacune d'elle est délimitée par des balises. Ces balises structurent le document. Elles vont par deux (sauf exceptions comme `<img>` par exemple), pour ouvrir et fermer une section. Les balises ouvrantes et fermantes se différencient par le « / »).

Chacune apporte des informations hiérarchiques précises et joue un rôle précis dans la logique du site. La **sémantique** (le sens) de chaque balise doit être respectée. Elles permettent un référencement précis, une organisation rigoureuse et logique et facilitent la "lisibilité" du code.

Exemple de structure :



Remarque : il s'agit de l'organisation des différentes « boîtes » matérialisées par les balises (un rectangle délimite la portée des balises sémantiques), mais **pas** de leur **disposition**, qui, elle, relève du langage CSS !

Les principales balises structurantes sont :
• <header></header> : regroupe l'introduction à un contenu (titres par ex.)
• <aside></aside> : éléments qui ne sont pas en rapport avec le contenu informatif du site. Par exemple des menus, de la publicité.
• <nav></nav> : contient les éléments dédiés à la navigation dans le site (liens internes).
• <section></section> : regroupe un contenu relatif à une même thématique ou fonctionnalité.
• <article></article> : élément de contenu qui pourrait être extrait du site sans perdre de sens; c'est le "contenu brut" qui apporte une information.
• <footer></footer> : "pied de page" (liens, crédits, mentions légales etc.)
• : balise générique qui délimite une "boîte" quand aucune autre balise ne se justifie.

Les balises peuvent contenir des **attributs** qui sont un peu comme des options. Par exemple, pour différencier des balises génériques, on peut leur affecter un **attribut** via une **id** avec la syntaxe .

A l'intérieur de ces balises « structurantes », le contenu est lui-même organisé sous forme de balises.
Il y a deux familles de balises : les balises de type « **bloc** » (block) et les balises de type « **ligne** » (inline). Comprendre leurs différences est fondamental pour la mise en page ultérieure avec le langage CSS.

Balise de type « ligne » (inline)	***Balise de type « bloc » (block)***
<a> : désigne un lien hypertexte.	: conteneur générique
<img> : inclut une image dans le document	<h1>, <h2>,…,<h6> : les six niveaux de titres, hiérarchiquement organisés entre eux. Un titre <h2> sera affecté à un titre de niveau hiérarchique inférieur à un titre <h1>
: une partie de texte importante	, : listes et éléments de liste
<emph> : une partie de texte moyennement importante	<p> : paragraphe de texte.

• Les éléments des balises de type **bloc** sont par défaut placés l'un **sous** l'autre par le navigateur, de manière séquentielle (c'est-à-dire l'un après l'autre dans l'ordre de leur apparition dans le code). C'est le cas par exemple des balises de paragraphe <p> : deux paragraphes successifs délimités par cette balise seront affichés l'un **au dessous** de l'autre. Les éléments de type bloc peuvent recevoir des attributs sur leur aspect (hauteur, largeur, marges, marges internes appelées *padding*) et de choisir relativement facilement un placement particulier sur la page web apparente.
• Les éléments de type **ligne** sont affichés les uns **à côté** des autres, sur la même ligne (dans la limite de la place disponible) et ne peuvent pas être pourvus d'attributs relatifs à leur largeur ou hauteur.

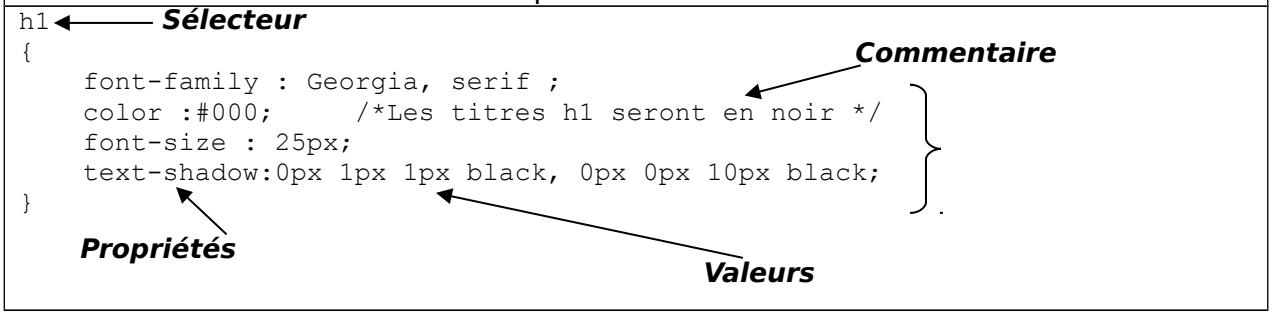
Rappel : les instructions données dans le fichier HTML ne comportent ***pas*** d'informations sur le **design** (mise en page, apparence, couleurs, typographie etc.) ; ce n'est pas son rôle, qui doit se limiter au **contenu informatif**. Certaines balises anciennes (par ex.) sont donc à proscrire et ne sont plus utilisées.

6. L'aspect du site : le design (fichier CSS)

En l'absence de précisions sur le design (mise en page, couleurs, fontes etc.), les pages d'un site sont toutes semblables et sans intérêt graphique. Les **feuilles de style en cascade** (CSS) permettent de spécifier tous les attributs de mise en page **sans toucher au contenu**.

L'intérêt est multiple : on pourra personnaliser des feuilles de style suivant l'utilisation fixe (ordinateur de bureau) ou nomade (smartphone, tablette), suivant l'humeur du jour ou les spécificités temporaires (période de soldes pour un site de vente par exemple).
Les CSS listent les propriétés et les attributs de chaque balise utilisée par le code HTML.
Le fichier qui contient ce informations est ouvert, lu et stocké en mémoire par le navigateur en début d'interprétation du fichier HTML.

Exemple de fichier CSS3



6.1. L'hérédité

Dans les feuilles de style en cascade (CSS), toute balise (qualifiée d'*enfant*) hérite des propriétés de la balise **à l'intérieur de laquelle elle se trouve "enfermée"** (celle-ci étant la balise "*parent*"). Si l'on veut une **autre propriété**, il faut la préciser dans le sélecteur de la balise enfant.
Ex. : si on définit la fonte `{font-family: 'Georgia';}` dans le sélecteur *body*, tout le texte de la page sera en *Georgia* sans qu'il soit nécessaire de le préciser pour les autres balises, puisque ces dernières sont "à l'intérieur" des balises `<body>`.
Si on veut une autre fonte pour les paragraphes par exemple, il faudra créer un sélecteur `p {font-family: 'Times', 'Garamond', serif}` qui "prendra la main" sur cette propriété.

6.2. La syntaxe des sélecteurs

Syntaxe générale : `selecteurCSS {proprieteCSS: valeurCSS;}`

Indication des selecteurs :

`p, h1, h2 {font-size : 2px;}`	Les trois sélecteurs `p`, `h1` et `h2` ont les mêmes propriétés
`header p {font-size : 2px;}`	La propriété concernera les balises `<p>` qui se trouvent à l'intérieur de la balise `<header>`
`#debut p {font-size : 2px;}`	La propriété concernera les balises `<p>` qui se trouvent à l'intérieur de la balise `<div id="debut">`
`h1+p {margin-top : 5px;}`	Sélecteur adjacent. La propriété s'applique uniquement à la balise `<p>` qui **suit immédiatement** une balise `<h1>`.

Classes	
`p.penche {font-style : italic;}`	La propriété concernera les balises `<p class="penche">`, et pas les autres.
`.penche {font-style : italic;}`	Toutes les balises à qui on ajoute `class="penche"` seront concernées : par ex. `<h3 class="penche">` ou `<p class="penche">`
Pseudo-classes	
`a:hover {color : yellow;}`	Les liens `<a>` se colorent en jaune au survol.
`a:link {text-decoration : none;}`	Le lien pas encore visité n'a pas de soulignement.
`a:visited {font-style : bold;}`	Les liens déjà visités apparaissent en gras.

(Les attributs Id et Class fonctionnent de la même manière. La seule différence est que Id est attribué de manière unique à une balise. La distinction entre les deux intervient si on fait du Javascript).

6.3. Quelques propriétés et valeurs

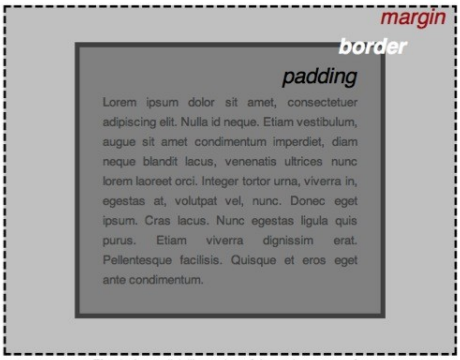
6.3.1. Propriétés des sélecteurs généraux

Propriété	**Valeurs**
`border-style`	`Solid, dotted, dashed, double...`
`border-top-style(bottom, left, right)`	Idem
`border-color`	idem
`padding`	*4 valeurs* (top right bottom left)
`margin`	Idem
`margin-top (bottom, left, right)`	
`background-image`	url(……);
`background-color`	*couleur*
`float`	Right, left

Exemple :

```
article img {                    /* toutes les images de la section
        margin-top: 30px;           article auront une marge supérieure
        margin-bottom: 20px;        de 30 pixels et inférieure de 20 px */
}
```

Remarque 1 : si les marges sont différentes autour d'un élément, on spécifie les quatre. Exemple : `{padding :1px 2px 5px 6px;}` signifie que la marge interne *haute* sera de 1px, la *droite* de 2px, la *basse* de 5px et la *gauche* de 6px (sens horaire).



Remarque 2 : **Padding, border et margin** :
La "boîte" ou conteneur est la partie sombre du centre. Le **padding** et l'espace entre le contenu et la bordure du conteneur, **border** est la bordure elle-même (son épaisseur est paramétrable) et **margin** est l'espace créé autour de la boïte.

6.3.2. Proprietés des sélecteurs de type texte

Propriété	**Valeurs**
font-family	'Georgia', 'Times', serif, sans…
font-size	unité
font-style	Normal, italic, oblique
font-weight	Normal, bold, light
width	*unité*
text-align	Left, right, center, justify
text-decoration	None, underline, overline...
background-color	*couleur*
line-height	*unité*
text-shadow	*couleur*, *longueur*, none
list-style-type (puces)	Disc, square,…, none

Exemple :

```
#principal h1 {
        font-family: 'Merienda', 'Georgia', sans;
        font-size:20px;
        text-align:center; }
```

6.3.4. Les unités

Unité relative	**Définition**
%	Valeur en pourcentage
em	Proportion de la taille de la fonte courante. Ex. : 1.2em vaut 1,2 fois la taille standard.
Unité absolue	
px	Pixel
pc (pica)	1 pc égale 12 pt
pt (point)	1 pt = 1/72è de pouce

6.3.5. Quelques propriétés spécifiques à CSS3

Ombres		
Box-shadow	2px 2px 2px black inset;	Les deux premières valeurs sont le décalage horizontal et vertical (obligatoires), les deux autres sont le flou et la couleur (facultatifs) ; inset crée une ombre interne à la boîte (externe par défaut).
Text-shadow	2px 2px 2px black;	idem
Coins arrondis		
Border-radius	10px 10px 10px 10px	Rayons en partant de l'angle en haut à gauche, puis dans le sens horaire.
Transparence		
opacity	0.5	La valeur doit être comprise entre 0 (totalement transparent) et 1 (opaque).

6.3.6. Les fontes autres que les fontes de l'utilisateur

Le navigateur peut télécharger, à l'ouverture de la page, une ou plusieurs fontes qui ne sont pas présentes dans le système de l'utilisateur. Ceci nécessite donc une connexion internet, ce qui en limite l'utilisation en « interne ».
Exemple : **http://www.google.com/webfonts**
Après avoir choisi la fonte, cliquer sur "use" et recopier les lignes de code HTML (entre les balises `<head>`) et CSS (dans la feuille de style), comme indiqué sur le site.

6.4. La mise en page

Le positionnement des conteneurs est au centre des préoccupations en CSS. Il en existe plusieurs : dans le flux, relatif, absolu, fixe et flottant.

6.4.1. Le positionnement « dans le flux » :

Par défaut, chaque nouveau élément défini par une balise de type « bloc » (`<p>` par ex.) succède au précédent en se plaçant juste au dessous. Les blocs occupent toute la largeur possible dans leur conteneur.
Chaque élément de type « ligne » se place sur la même ligne que ce qui l'entoure.
On dit que les conteneurs se placent dans le flux courant : les uns après les autres en fonction de l'ordre avec lequel ils apparaissent dans le fichier HTML.

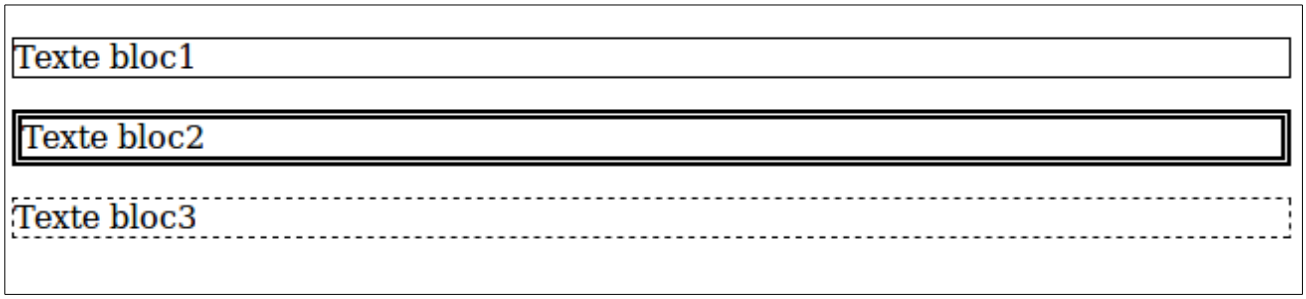
Exemples :
Considérons le fichier HTML avec le contenu suivant :

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8" />
		<title>Test positionnement</title>
		<link rel="stylesheet" type="text/css" href="styles.css">
	</head>

	<body>
		<p id="paragraphe1">Texte bloc1</p>
		<p id="paragraphe2">Texte bloc2</p>
		<p id="paragraphe3">Texte bloc3</p>
	</body>
</html>
```

Afin de mieux visualiser les « blocs », on complète le fichier CSS avec :

```
#paragraphe1
{	border-style: solid;
	border-width: 1px;
}

#paragraphe2
{	border-style: double;
	border-width: 5px;
}
#paragraphe3
{
	border-style: dashed;
	border-width: 1px;
}
```

On peut visualiser alors les blocs obtenus et leur position :



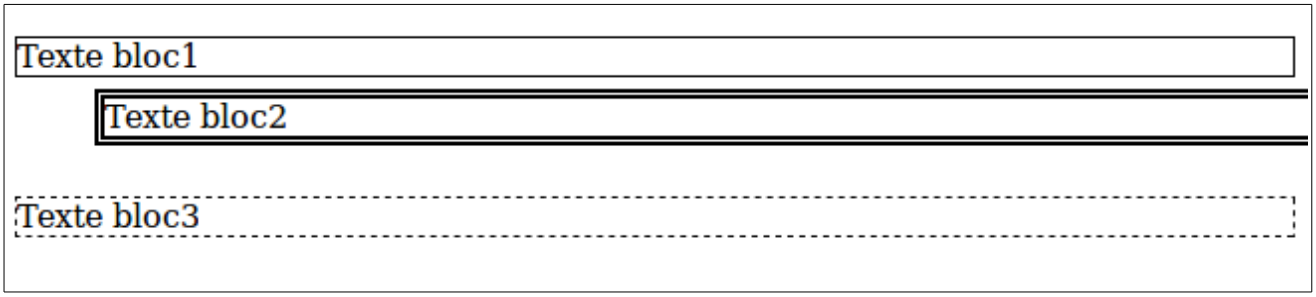
Les 3 blocs se retrouvent l'un **en dessous** de l'autre, dans l'ordre dans lequel ils apparaissent dans le fichier HTML, et ils occupent la largeur totale de la place disponible.

● Le positionnement « relatif » :

Le conteneur déclaré « relatif » peut être décalé de la position qu'il aurait **normalement dans le flux courant**. Il faut donc comprendre « relatif à la place qu'il occuperait normalement dans le flux ».
Exemple :

`#paragraphe2`
`{`
`  border-style:solid;`
`  border-width: 1px;`
`  position: relative;`
`  left: 40px;`
`  bottom: 20px;`
`}`
	Le conteneur appelé « paragraphe2 » sera décalé de 40 pixels vers la gauche (par rapport à la marge gauche du **conteneur parent**) et de 20 pixels vers le haut, par rapport à la place qu'il aurait occupé si cette position relative n'avait pas été définie.

On obtient :



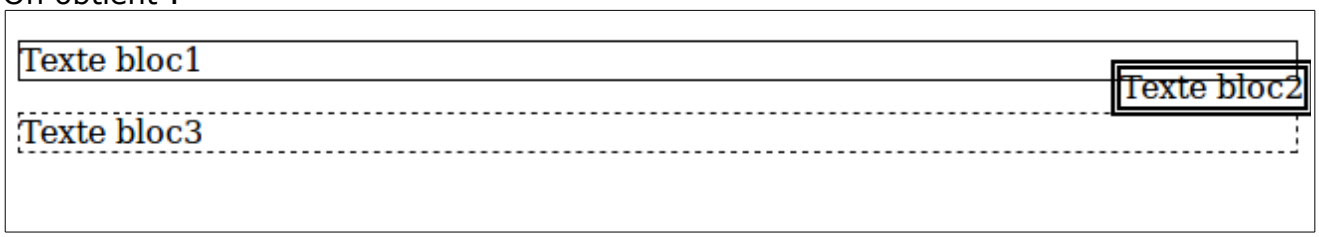
● Le positionnement « absolu» :

Cette fois-ci, on va pouvoir réellement positionner un conteneur là où l'on veut sur la page, sachant que le point de coordonnées 0,0 est en haut à gauche de l'écran.

Exemple :

`#paragraphe2`
`{`
`  border-style:solid;`
`  border-width: 1px;`
`  position: absolute;`
`  right: 0px;`
`  top: 10px;`
`}`
	Le conteneur appelé « paragraphe2» sera affiché en bas à droite de son conteneur parent.
	Ce qui ne veut pas forcément dire en bas à droite de l'écran ! Tout dépend du conteneur parent dans lequel il se trouve...

On obtient :



6.4.2. Le positionnement « flottant » :

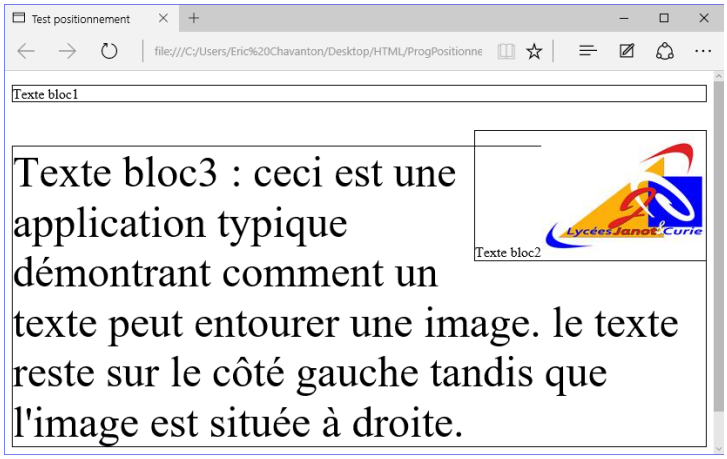
Le conteneur est « **retiré** » du flux normal pour prendre place à gauche (*left*) ou à droite (*right*) du conteneur qui le contient. L'élément qui le suit dans le code HTML occupera alors l'espace laissé libre, en épousant sa forme.
Pour bien démontrer l'aspect pratique de se positionnement, nous allons prendre l'exemple d'une image autour de laquelle on souhaite mettre du texte,

Exemple :

```
paragraphe2
{
      border-style:solid;
      border-width: 1px;
      float: right;
}
```

Le conteneur appelé « paragraphe2» sera affiché à droite de son conteneur parent.

On obtient :



6.4.3. Le positionnement à l'aide de l'instruction display

L'instruction **display** permet une grande finesse de placement dans la page.
Elle accepte les attributs inline, block ou inline-block.

• Elle permet par exemple d'appliquer des marges à une image, ce qui n'est pas possible par défaut puisque la balise <img> est de type « ligne » (« inline »).
Exemple :

```
img {
      display: block; /* l'image est maintenant de type « block » */
      margin: 0px 20px 20px 30px;
      box-shadow: 2px 2px 10px black;
}
```

• L'attribut `inline-block` permet de cumuler les avantages du positionnement en ligne et de la possibilité d'ajouter des marges ; très utile par exemple pour un menu en ligne ou bien pour juxtaposer deux blocs côte-à-côte.

Exemple :

```
li {
      display: inline-block;
      width: 25%;
      height: 35px;
      line-height: 35px;
      font-size: 17px;
      text-align: center;
}
```

Normalement, les éléments de liste devraient se trouver les uns au-dessous des autres puisque `<li>` est de type **bloc**. Mais là, elles se placeront les unes *à côté* des autres. Elles occuperont chacune 25% de la largeur du conteneur parent (`width : 25%`) ; cette valeur a été choisie parce que la page contient 4 articles de menus sur la largeur :

En conclusion : On peut noter que cette partie de la conception en HTML/CSS est la **base de la mise en page** d'un site. Il est absolument nécessaire de maîtriser ces différentes notions, et surtout de se rappeler que les règles appliquées se font EN FONCTION du **conteneur parent** !
Il est donc absolument indispensable, lors de la création d'une page d'un site, de réfléchir au préalable à l'agencement des différents blocs qui constitueront la page.

5. Quelques liens utiles

http://www.w3.org
Le site du World Wide Web Consortium : la référence de la standardisation du web.

http://validator.w3.org/
Pour vérifier si le site créé respecte bien les standards du W3C.

http://html5doctor.com/
Le glossaire (glossary) recense les balises de HTML5.

http://www.w3schools.com/css3/default.asp
Les références des sélecteurs CSS

http://www.alsacreations.com/
Tutoriels et articles intéressants.

http://www.proftnj.com/RGB3.htm
Pour choisir des couleurs et connaître leur code.

http://fr.openclassrooms.com
Des tutoriels et des cours, entre autres sur HTML et CSS.