

HTML/CSS

Langues et encodage

La déclaration de la langue se fait dans la balise html : `<html lang="fr">`

Nous déclarons ici que la langue est le français, ce qui sert au référencement (ex : Google vous propose les pages en français ou une traduction) et aux synthétiseurs vocaux.

L'en-tête : encodage

Le type d'encodage des caractères de la page

`<meta charset="encodage"/>`

ATTENTION : doit être cohérent avec l'encodage physique de la page. Pour le français on dispose des encodages suivants :

iso-8859-1 : encodage classique pour les langues de l'Europe occidentale (aussi appelé LaOn-1) : `<meta charset=" iso-8859-1 "/>`

iso-8859-15 : même encodage comportant quelques caractères supplémentaires comme le signe €... : `<meta charset=" iso-8859-15 "/>`

utf-8 : encodage pour les caractères de la majorité des langues mondiales :
`<meta charset=" utf-8 "/>`

Images et liens

Balise ``. Ses principaux attributs sont :

`src` : emplacement du fichier source de l'image

`width` : largeur

`height` : hauteur

`alt` : texte qui apparaît comme info bulle de l'image ou lorsque l'image ne s'affiche pas

```

```

Lien vers une autre page - même site - même répertoire

```
<a href="page1.html">Titre du lien</a>
```

Lien vers une autre page - même site - autre répertoire

```
<a href="../../rep1/sousrep2/page1.html">Titre</a>
```

Liens vers une page d'un autre site

```
<a href="http://www.autresite/chemin/page.html">Un site</a>
```

Lien vers une autre page - nouvelle fenêtre du navigateur

```
<a href="exemples/page1.html" target="_blank">Lien</a>
```

Lien vers un fichier

```
<a href="mydoc.pdf">Document PDF</a>
```

```
<a href="http://www.autresite/mydoc.pdf">Document PDF</a>
```

Lien vers un fragment de document

Définition d'un fragment :

```
<a id="ancre12">Sommet</a>
```

Accès au fragment :

```
<a href="#ancre12">Lien</a>
```

```
<a href="http://www.autresite/page1.html#ancre12">Lien</a>
```

Lien sous forme d'image vers une autre page

```
<a href="index.html"></a>
```

CSS

- **CSS** = Cascading Style Sheets = Feuilles de style en cascade

- Ensemble de règles
- Qui sélectionnent les éléments HTML
- Qui leur associent des caractéristiques de style

- 3 versions : CSS 1 et CSS 2 puis CSS 3

- **Séparation du contenu et de la présentation :**

- Conserver la lisibilité des documents même avec des navigateurs ne supportant pas les CSS
- Permettre leur consultation avec d'autres supports (assistants personnels, synthèse vocale, navigateurs braille...)
- Améliorer l'accessibilité des documents
- Améliorer l'impression des documents

Voici la structure d'une règle de style qui s'applique à la balise table en indiquant l'épaisseur, la forme et la couleur de la bordure. Cette règle de style va être interprétée par le navigateur de l'utilisateur qui applique par la suite les transformations à des éléments. Une règle se décompose ainsi en trois éléments :

Sélecteur – Le sélecteur est en fait une balise HTML, telle que <h1>, <table> ou autres, à laquelle on applique un traitement de style.

Attribut - Un attribut indique quelles caractéristiques d'une balise HTML on veut changer telle que la couleur, taille, décoration etc. Les attributs se trouvent entre les accolades et sont séparés entre eux par un point-virgule.

Valeur - Chaque attribut possède une valeur. Ainsi, par exemple, l'attribut **color** possède des valeur comme « red » ou « #F10000 » etc.

- Une règle de CSS se compose de :

- Un sélecteur

- Une déclaration entre accolades { }, composée de propriétés séparées par des ;

- § Une propriété est constituée de

- § Un mot-clé, suivi de ":"

- § La (les) valeur(s) associée(s)

Exemple (pour appliquer une taille de 18 pixels aux titres de niveau 1), la règle est :

```
h1 { font-size:18px;}
```

- Les commentaires dans les feuilles de style s'écrivent entre `/*` et `*/`

Sélecteur = « lien » entre le document HTML et la feuille de style (ou les feuilles de style). Permet au CSS d'identifier (sélectionner) le(s) élément(s) au(x)quel(s) appliquer une règle.

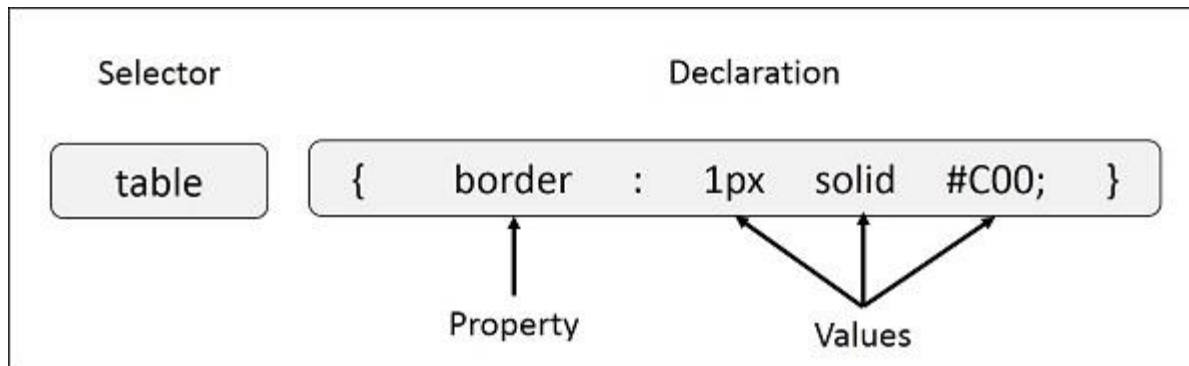
3 types simples de sélecteurs :

- les **balises**
- les **classes**
- les **identificateurs**

Toutes les balises HTML peuvent servir de sélecteurs à des règles CSS.

Exemples :

- Pour agir sur les paragraphes => `p { font-size: 14px; }`
- Pour agir sur les liens => `a { font-family: tahoma, arial, verdana; }`



Sélecteur (peut être suivi d'autres sélecteurs)

Comme par exemple :

`h1, h2, p, blockquote {`

`article > p > a {`

`h1 + p {`

Par contre chaque exemple signifie une autre logique d'attribution des règles de styles

La dernière valeur d'un attribut est toujours suivie d'un point-virgule

`h1 { color: maroon; margin-left: 40px; }`

Après les sélecteurs il faut toujours mettre les accolades

Les attributs sont toujours suivis d'un double point

La description se termine toujours par l'accolade de fermeture

Les balises

Chaque élément a une double identité :

- Sa **structure** (mot clé, attributs standards, ...)
- Son **rendu** (ou apparence), qui est défini par défaut pour chaque navigateur (et qui peut être modifié par CSS).

2 grands mode de rendu des éléments

Les rendus de type **block**

Les rendus de type **inline**

Cette typologie dicte le comportement en terme de **positionnement** et **d'affichage**

Ce sont :

Des blocs dans les documents – Exemples : **paragraphes, listes...**

Apparaissent les uns en dessous des autres

Ont des dimensions et des marges externes ou internes fixées par défaut, à l'exception des blocs `<div>`

Sont positionnables (avec les feuilles de style CSS)

Ils peuvent :

1) Contenir d'autres blocs sauf les blocs de paragraphes (`<p>`) et de titres (`<h1>`, `<h2>`, ... `<h6>`) qui ne peuvent contenir d'autres blocs

2) Contenir des éléments inline

Exemples :

<code><h1></h1></code>	<code><table></table></code>
<code><h2></h2></code>	<code></code>
...	<code></code>
	<code><blockquote></blockquote></code>
	<code><dl></dl></code>
<code><h6></h6></code>	<code><div></div></code>
<code><p></p></code>	etc.

Les éléments de type Block

`<header>`

`<nav>`

`<article>`

`<section>`

`<aside>`

`<footer>`

Les éléments inline

Apparaissent au fil du texte, ils ne sont pas placés les uns au dessus des autres (ils restent à l'emplacement défini).

- 1) N'ont pas de marges internes ou externes par défaut
- 2) Ne sont pas dédiés à un positionnement précis (même si cela est possible avec les CSS)
- 3) Servent à modifier, enrichir des portions de textes, apporter du sens

Les éléments inline :

- 1) Ne peuvent contenir que des éléments inline (→ pas de block)
- 2) Un élément inline doit être contenu dans un élément de type block

Exemples :

`<a> ... `

` ... `

``

` ... `

`<i> ... </i>`

` ... `

` ... `

Les éléments inline

Note : en HTML4, certaines balises définissaient un rendu visuel (ex : `` = texte en gras. Ce n'est plus le cas en HTML5.

b (avant : bold) ne signifie pas « en gras », mais stylistiquement décalé sans avoir une importance supplémentaire (sinon, utiliser strong)

i (avant : italic) ne signifie pas « en italique », mais « dans une voix ou une humeur alternative » sans emphase ni importance particulière (sinon, utiliser em)

hr (avant : ligne horizontale) prend un sens sémantique de séparation entre paragraphes (potentiellement rendu comme une ligne horizontale)

small ne signale plus un texte en petite taille, mais caractérise des éléments « en petits caractères » comme un copyright ou une référence écrite « en petits caractères »

Les structures universelles

DIV et **SPAN** = éléments pour structurer des documents Web (en association avec les CSS)

div = élément de type bloc (il y a toujours un saut de ligne après le bloc)

span = élément en ligne (les éléments se trouvent sur la même ligne)

N'apportent aucune contrainte de présentation, ils sont « **neutres** » à cet égard. Ils servent à « **ajouter** » de la structure.

Attention, ces éléments n'ont pas de sens particulier, ils sont neutres également sur le plan de la sémantique. Par conséquent, ils ne doivent pas remplacer systématiquement les autres éléments qui sont d'ailleurs nécessaires pour un bon traitement par les moteurs de recherche.

Sélecteur universel

```
* {  
    margin:0;  
    padding:0;  
}
```

Ce symbole s'applique à **tous les éléments** de la page (à utiliser avec précautions). Il est souvent utilisé pour « mettre les compteurs à zéro », ici les marges intérieures et extérieures doivent être à zéro par défaut.

Appliquer un style à une balise

```
p {  
    font-size:12px;  
}
```

Tous les textes contenus dans des paragraphes auront une taille de 12 pixels.

Identifier une balise par son attribut class

```
img.presentation {  
    border : 1px solid red;  
}
```

Toutes les images ayant l'attribut class="presentation" auront une bordure rouge (les autres non)

```
.presentation {  
    border : 1px solid red;  
}
```

Ici la bordure pourra être appliquée à différents éléments HTML et pas simplement aux images. Privilégier cette syntaxe plutôt que la première pour plus de souplesse.

Identifier une balise par son attribut id

```
#page {  
    background-color : black;  
}
```

L'élément **(unique)** ayant l'attribut `id="page"` aura une couleur de fond noire.

Un même style pour différentes balises

```
p.introduction, blockquote {  
    font-weight : bold;  
}
```

Tous les paragraphes ayant l'attribut `class="introduction"` ET toutes les citations seront en gras.

Préciser une balise contenue dans une autre

```
.introduction a {  
    text-decoration: underline;  
}
```

Tous les liens contenus DANS le(s) éléments ayant l'attribut `class="introduction"`

Préciser une balise qui suit une autre

```
h1+p {  
    font-style:italic;  
}
```

Le paragraphe situé directement APRES un titre de niveau 1 sera en italique

Cibler seulement le descendant direct d'un élément

Les sélecteurs

```
#menu > li > a {  
    background-color: #000000;  
}  
  
<ul id="menu">  
    <li><a href="#">Lien</a></li>  
    <li><a href="#">Lien</a></li>  
    <li><a href="#">Lien</a>  
        <ul>  
            <li><a href="#">Lien niveau 2</a></li>  
            <li><a href="#">Lien niveau 2</a></li>  
        </ul>  
    </li>  
</ul>
```

Tous les éléments **a** qui sont les descendants directs d'un élément **li** lui-même directement contenu dans un élément ayant l'identifiant **menu**.

Cibler les éléments en fonction de leur attribut

```
a[title]{  
    border-bottom: 1px dotted #cccccc;  
}
```

Tous les liens qui ont un « title » (infobulle) seront soulignés en pointillés gris

```
input[type="text"]{  
    border-bottom: 1px solid #cccccc;  
}
```

Tous les champs (zone de saisie d'un formulaire) de type text (pas les types email, tel,,)

Les **attributs** : permettent de préciser une valeur dans les balises ouvrantes ou uniques, structure en paires : **nom**="valeur"

<p **id**="unique">Paragraphe unique</p>
<p **class**="special">Paragraphe spécial</p>

Apostrophes doubles " (le plus courant) ou simples sont optionnelles mais (fortement) recommandées

Attributs standards : **id**="id432" ou **class**="typeUrgent"

Attention :

- **XML** et **XTML** : case sensitive ; **HTML5** : non
- Recommandation : noms d'attributs en minuscules

CSS – Unités de mesure

Unit	Description	Example
%	Defines a measurement as a percentage relative to another value, typically an enclosing element.	<code>p {font-size: 16pt; line-height: 125%;}</code>
cm	Defines a measurement in centimeters.	<code>div {margin-bottom: 2cm;}</code>
em	A relative measurement for the height of a font in em spaces. Because an em unit is equivalent to the size of a given font, if you assign a font to 12pt, each "em" unit would be 12pt; thus, 2em would be 24pt.	<code>p {letter-spacing: 7em;}</code>
ex	This value defines a measurement relative to a font's x-height. The x-height is determined by the height of the font's lowercase letter x.	<code>p {font-size: 24pt; line-height: 3ex;}</code>
in	Defines a measurement in inches.	<code>p {word-spacing: .15in;}</code>
mm	Defines a measurement in millimeters.	<code>p {word-spacing: 15mm;}</code>
pc	Defines a measurement in picas. A pica is equivalent to 12 points; thus, there are 6 picas per inch.	<code>p {font-size: 20pc;}</code>
pt	Defines a measurement in points. A point is defined as 1/72nd of an inch.	<code>body {font-size: 18pt;}</code>
px	Defines a measurement in screen pixels.	<code>p {padding: 25px;}</code>
vh	1% of viewport height.	<code>h2 { font-size: 3.0vh; }</code>
vw	1% of viewport width	<code>h1 { font-size: 5.9vw; }</code>
vmin	1vw or 1vh, whichever is smaller	<code>p { font-size: 2vmin;}</code>

CSS – Unités de mesure

Les unités de mesure les plus importantes sont - %, **EM** et **PX**. Même s'il y en a d'autres, la plupart du temps on va utiliser ces trois mais chacune a une utilisation bien précise avec ses forces et ses inconvénients.

PX : c'est une unité fixe qui se réfère au pixel. La taille des éléments est donnée en nombre de pixels qu'occupe un élément sur un des axes (vertical ou horizontal). Ici il faut également savoir que plus il y a de pixels dans un écran, plus il faudra de pixels

pour les éléments traités afin de garder les bonnes proportions.

De plus, si tous les éléments dans une page web sont donnés en **px**, s'il y un changement il faut refaire tous les indicateurs de taille un par un et les adapter aux nouvelles valeurs demandées.

EM : il s'agit d'une unité relative qui indique la taille des éléments en relation à une valeur de référence qui est définie soit via les paramètres de navigateur de l'utilisateur, soit via la feuille de style en amont de la structure CSS. La force de cette approche est le fait que les éléments traités avec **em** gardent leur proportion par rapport à l'ensemble. De plus, pour changer les valeurs, il suffit de changer la valeur de référence. L'utilisation de **em** est plus que recommandée (surtout pour le texte).

% : le pourcentage ressemble dans son action à celui de **em** mais s'applique principalement à des éléments autres que le texte (même si cela n'est pas exclu). Pourcentage est un excellent outil pour garder les bonnes proportions des éléments.

Pts	Px	Em	Porcentage
6pt	8px	0.5em	50%
7pt	9px	0.55em	55%
7.5pt	10px	0.625em	62.5%
8pt	11px	0.7em	70%
9pt	12px	0.75em	75%
10pt	13px	0.8em	80%
10.5pt	14px	0.875em	87.5%
11pt	15px	0.95em	95%
12pt	16px	1em	100%
13pt	17px	1.05em	105%
13.5pt	18px	1.125em	112.5%
14pt	19px	1.2em	120%
14.5pt	20px	1.25em	125%
15pt	21px	1.3em	130%
16pt	22px	1.4em	140%
17pt	23px	1.45em	145%
18pt	24px	1.5em	150%
20pt	26px	1.6em	160%
22pt	29px	1.8em	180%
24pt	32px	2em	200%
26pt	35px	2.2em	220%
27pt	36px	2.25em	225%
28pt	37px	2.3em	230%
29pt	38px	2.35em	235%
30pt	40px	2.45em	245%
32pt	42px	2.55em	255%
34pt	45px	2.75em	275%
36pt	48px	3em	300%

Les unités de mesures en CSS (pour exprimer des largeurs, marges, taille de police...):

px (absolu),

pt (absolu),

em (relatif),

% (relatif)

rem (relatif-absolu)

```
html {
```

```
font-size: 100%;
```

```
/* Pour les vieux navigateurs*/
```

```
}
```

```
body {
```

```
font-size: 0.95em;
```

```
/* Choisir la taille de police  
par défaut */
```

```
}
```

```
h1 {
```

```
font-size: 16px;
```

```
/* équivaut à 1em */
```

```
}
```

Les classes HTML = familles d'éléments librement définies

```
<p id="p1" class="maclasse">...</p>  
<span id="s2" class="maclasse">...</span>
```

Sélection en CSS : nom de la classe préfixé d'un point. Exemple :

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

Les classes peuvent être restreintes à un élément

- Le point est précédé de l'élément auquel la classe peut être appliquée
- La classe ne s'applique qu'aux éléments correspondants `p.maclasse {color : green;}`
- La règle s'appliquera à tout le paragraphe de classe `maclasse`
- Mais PAS aux autres éléments même s'ils sont de la classe `maclasse` (`div`, `h1`, `span`, `table` ou autre)

Pseudos classes

"Une **pseudo-classe** CSS est un mot-clé ajouté au sélecteur pour indiquer un état particulier de l'élément qui doit être sélectionné. Par exemple, **:hover**, appliquera le style quand l'utilisateur survolera l'élément visé par le sélecteur."

<https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes>

Liens

```
a{  
    color: #000000;  
}  
a:hover {  
    background-color: #000000;      /* Couleur de fond au survol */  
    color: #ffffff;  
}
```

Tous les liens auront un fond noir et une couleur de police blanche au survol.

Également pour les liens, pseudos classes :

a:active – lien qui est activé : quand l'internaute clique dessus

a:focus - élément sélectionné (ex:champ de formulaire)

a:link - liens qui n'ont pas été visités (peu utilisés)

a:visited – tous les liens qui ont été visités (peu utilisés)

Autres pseudo-classes

Il existe beaucoup de pseudo-classes, voici les plus utilisées :

- :first-letter** – Sélectionne la première lettre d'un élément
- :first-line** – Première ligne
- :first-child** – Sélectionne le premier élément enfant

(également **:last-child** et **:nth-child(5)** pour le 5ième élément)

Formater la première lettre de tous les titres de niveau 2 :

```
h2:first-letter {  
    font-size : 30px;  
    color: #ccc;  
}
```

Les identificateurs ne peuvent porter que sur un seul élément, unique et identifié (de fait) dans un document HTML

Le sélecteur d'un identifiant est écrit avec un dièse (#) comme préfixe
`#monidentificateur { ... }`

Exemple :

```
#grandtitre {  
    font-weight: bold;  
}
```

Côté HTML :

```
<h1 id="grandtitre">Mon titre</h1>
```

Utiliser des classes ou des identificateurs ?

- Lorsque c'est possible, lorsqu'un élément est identifié de manière unique dans un document, on privilégie l'utilisation d'un identificateur : le code est ainsi plus facile à lire et à maintenir.
- Les classes sont adaptées pour des éléments génériques, (éventuellement) appelés à être utilisés plusieurs fois dans un même documents : « types » de paragraphes, éléments redondants (mise en valeur de portions de texte) ...

4 façons d'incorporer les directives de style au HTML

- Données incorporées dans les balises

```
<p style="color:red; font-weight:bold;">
```

Rouge gras

```
</p>
```

- Données incorporées dans l'en-tête du document

```
<head>
```

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

```
    h4 { color:green }
```

```
  </style>
```

```
</head>
```

- Feuille externe attachée ou liée

```
<head>
```

```
  <link rel="stylesheet" type="text/css" href="monstyle.css">
```

```
</head>
```

Les règles de style se trouvent dans un fichier monstyle.css auquel le document fait appel ; il est possible et parfois utile de lier plusieurs feuilles de styles successivement.

- (Feuille de style importée)

Embedded CSS - The <style> Element

```
<!DOCTYPE html>
<html>
  <head>
    <title> Encrage des styles </title>
    <style type = "text/css" media = "all">
      body { background-color: linen; }
      h1 { color: maroon; margin-left: 40px; }
    </style>
  </head>
  <body>
    <h1> Ici c'est un titre </h1>
    <p> Ici c'est un paragraphe </p>
  </body>
</html>
```

Inline CSS - The *style* Attribute

```
<!DOCTYPE html>
<html>
  <head>
    <title> Encrage des styles </title>
  </head>
  <body>
    <h1 style = "color:#333;"> This is inline CSS </h1>
    <h2> Ici c'est un titre </h2>
    <p> Ici c'est un paragraphe </p>
  </body>
</html>
```


External CSS - The <link> Element

```
<!DOCTYPE html>
<html>
  <head>
    <title> Encrage des styles </title>
    <link type = "text/css" href = "monstyle.css" media = "all" />
  </head>
  <body>
    <h1> Ici c'est un titre </h1>
    <p> Ici c'est un paragraphe </p>
  </body>
</html>
```

Imported CSS - @import Rule

```
<!DOCTYPE html>
<html>
  <head>
    @import "monstyle.css";
  </head>
  <body>
    <h1> Ici c'est un titre </h1>
    <p> Ici c'est un paragraphe </p>
  </body>
</html>
```

L'héritage

En appliquant un style « **par défaut** » à un élément englobant, dans le cas où ce style n'est pas redéfini, les éléments « **enfants** » de body, et les « **enfants des enfants** » hériteront de ce style

```
<div style="color:red; font-weight:bold;">
```

Rouge gras

```
<div>
```

Rouge gras aussi

```
</div>
```

```
</div>
```

La div englobée hérite des propriétés de style

Note : définir un style sur body = l'appliquer par défaut à tous les éléments de la page

NB : Certaines propriétés de style ne sont pas transmises de l'élément parent à l'élément enfant, c'est le cas de margin, padding (et d'autres propriétés de bloc). Il y a aussi certains bugs de navigateurs anciens

```
<!doctype html>
```

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

```
<head>
```

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

```
<title>Page d'exemple</title>
```

```
</head>
```

```
<body>
```

```
<p>Hello world</p>
```

```
</body>
```

```
</html>
```

L'imbrications des éléments

```
<!doctype html>
```

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

```
<head>
```

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

```
<title>Page d'exemple</title>
```

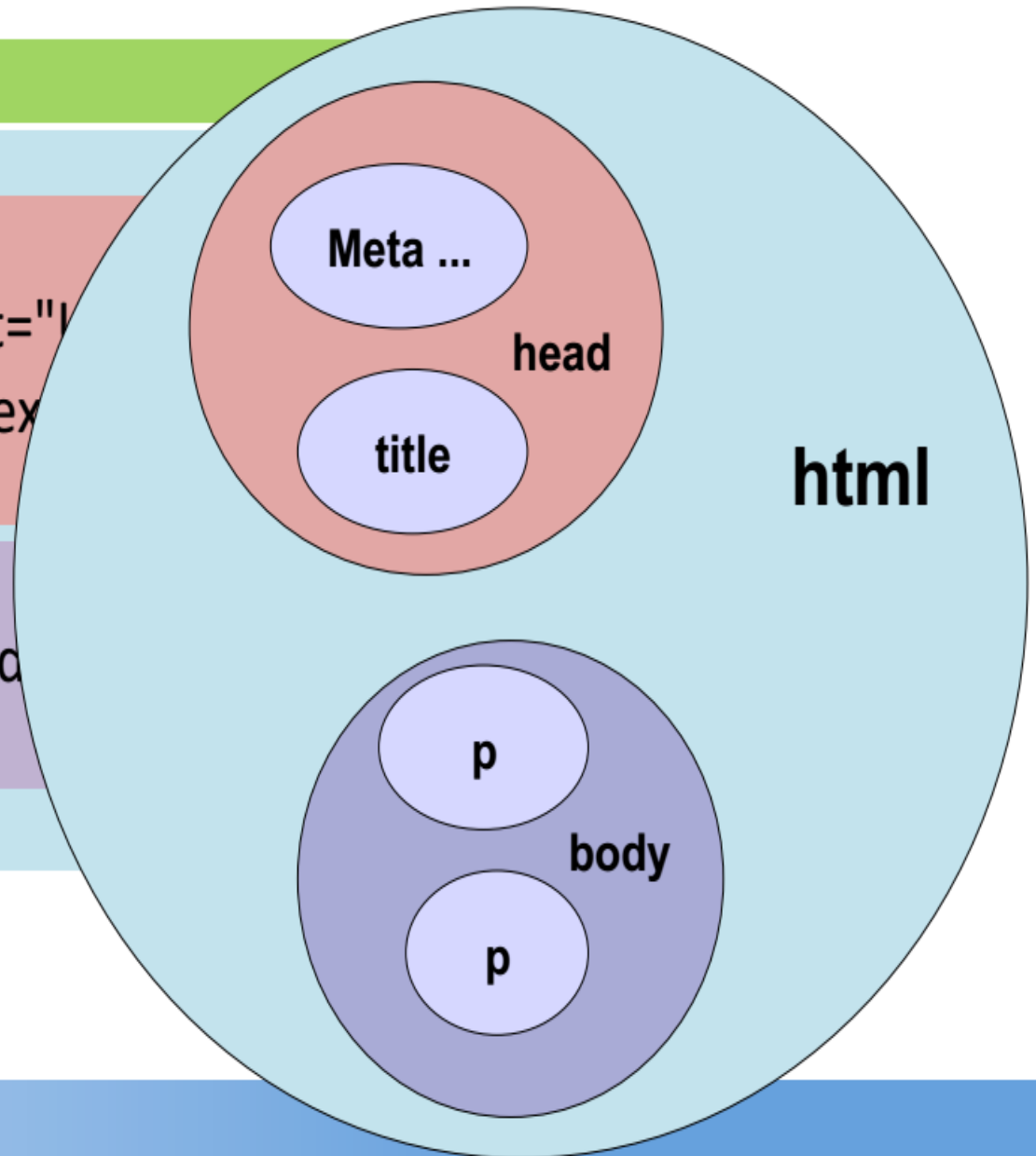
```
</head>
```

```
<body>
```

```
<p>Hello world</p>
```

```
</body>
```

```
</html>
```



CSS – héritage des règles

Un autre élément auquel il faut faire attention est la logique de l'héritage des règles de style. Comme il y a plusieurs manières d'intégrer les règles de style à un document Web, il y a également plusieurs manières de réécritures.

- Toute feuille de style **inline** prend la plus haute priorité. Ainsi, il remplacera toute règle définie dans les balises **<style> ... </ style>** ou les règles définies dans tout fichier de feuille de style externe.
- Toute règle définie dans les balises **<style> ... </ style>** remplacera les règles définies dans tout fichier de feuille de style externe.
- Toute règle définie dans un fichier de feuille de style externe prend la priorité la plus basse et les règles définies dans ce fichier ne seront appliquées que si deux règles au-dessus ne s'appliquent pas.

http://sitelec.org/download.php?filename=formation_web/liste_css.pdf

Positionnement "dans le flux"

- Par défaut, le navigateur construit l'affichage au fur et à mesure que les descriptions des éléments lui parviennent : un bloc est placé en dessous du précédents, les éléments inline sont placés les uns à côté des autres.
- Ces éléments sont alors dits "dans le flux", sa position dépend alors de ses propres marges et des marges internes du conteneur et des éléments adjacents.

```
div {  
  width: 600px ;  
  padding-top: 20px ;  
  border: solid 1px black;  
}  
  
p {  
  margin-left: 20px ;  
  margin-bottom: 20px ;  
  width: 300px;  
  border: solid 1px black;  
}
```

```
<div>  
  <p>paragraphe</p>  
  <p>paragraphe</p>  
</div>
```

paragraphe


paragraphe

Positionnement relatif

- Le positionnement relatif permet d'inscrire un contenu en flux normal, puis de le décaler horizontalement et/ou verticalement. Le contenu suivant n'est pas affecté par ce déplacement, qui peut donc entraîner des chevauchements.
- Les propriétés `top`, `right`, `left`, `bottom`, permettent de fixer la position relative.

```
<div>
<p id="premier">paragraphe</p>
<p id="second">paragraphe</p>
</div>
```

```
...
div {
  width : 600px ;
  padding-top : 20px ;
  border: solid 1px black;
}
p#premier {
  margin-left : 20px ;
  width: 300px;
  border: solid 1px black;
}
p#second {
  margin-left : 20px ;
  width: 300px;
  border: solid 1px black;
  position: relative;
  left: 4px;
  bottom: 22px;
}
```



paragraphe
paragraphe

Autre exemple : le décalage est relatif à la position normale de l'élément dans le bloc parent

```
.decale {  
  position: relative;  
  bottom: 5px;  
  border: solid 1px black;  
}  
...  
<p>Un paragraphe avec  
<span class="decale">un  
&eacute;l&eacute;ment  
d&eacute;cal&eacute;.</span>  
du reste du texte.</p>  
...
```

Un paragraphe avec un élément décalé du reste du texte.

- La position de l'élément est déterminée de manière absolue dans son conteneur parent positionné le plus proche, ou à défaut, dans la fenêtre du navigateur
- On utilise la propriété position, avec la valeur absolute, pour positionner un élément de manière absolue.
- Les propriétés top, right, left, bottom, permettent alors de fixer la position.

```
#boite1 {  
  position: relative;  
  width: 300px;  
  border: solid 1px black;  
}  
#boite2 {  
  position: absolute;  
  top: 10px;  
  right: 30px;  
  border: solid 1px black;  
}  
...  
<div id="boite1">  
  <p>Boite 1</p>  
  <div id="boite2">Boite 2</div>  
</div>
```

Boite 1 avec son contenu son contenu

Boite 2

Positionnement fixe

Le positionnement fixe est très comparable au positionnement absolu, sauf que l'élément fixe reste à sa place sur l'écran même lorsque l'utilisateur fait défiler le contenu.

Un élément fixe est comme « ancré » à sa place.

On utilise la propriété position, avec la valeur fixed.

Les propriétés top, right, left, bottom, permettent alors de définir la position.

Positionnement flottant

- On utilise la propriété float, avec la valeur left ou right, pour positionner un élément de manière flottante.
- L'élément prend place à gauche (ou à droite) dans l'élément conteneur.
- L'élément suivant occupera la place laissée libre.
- La boîte 1 se place de manière flottante à droite, la boîte 2 occupe l'espace restant.

Positionnement flottant

```
#boite1 {
  float: right;
  width: 100px;
  border: solid 1px black;
}
#boite2 {
  border: solid 1px black;
}
```

```
<div id="boite1">Boite 1</div>
<div id="boite2">Boite 2</div>
```

Boite 2 avec son contenu	Boite 1 avec son contenu
---	-----------------------------

Exemple avec 2 boîtes flottantes

```
<div id="boite1">Boite 1</div>
<div id="boite2">Boite 2</div>
<div id="boite3">Boite 3</div>
```

```
...
#boitel {
    float: right;
    width: 100px;
    border: solid 1px black;
}
#boite2 {
    float: left;
    width: 100px;
    border: solid 1px black;
}
#boite3 {
    border: solid 1px black;
}
```

[illegible]

Remarque : attention, si dans le flux du document, la boîte 3 était inscrite avant les autres boîtes, celle-ci, non positionnée, prendrait toute la largeur du document. Les autres boîtes occuperaient donc leur emplacement « flottant », mais en dessous.

```
<div id="boite3">Boite 3</div>
<div id="boite1">Boite 1</div>
<div id="boite2">Boite 2</div>
```

Positionnement flottant

```
#boite1 {
    float: right;
    width: 100px;
    border: solid 1px black;
}
#boite2 {
    float: left;
    width: 100px;
    border: solid 1px black;
}
#boite3 {
    border: solid 1px black;
}
```

Boite 3 avec son contenu son contenu son contenu son contenu son contenu son
contenu son contenu son contenu son contenu son contenu son contenu son contenu
son contenu son contenu son contenu son contenu son contenu son contenu son
contenu son contenu son contenu son contenu son contenu son contenu son contenu
son contenu son contenu son contenu son contenu son contenu son contenu

Boite 2 avec son contenu

Boîte 1 avec son contenu	
--------------------------	--

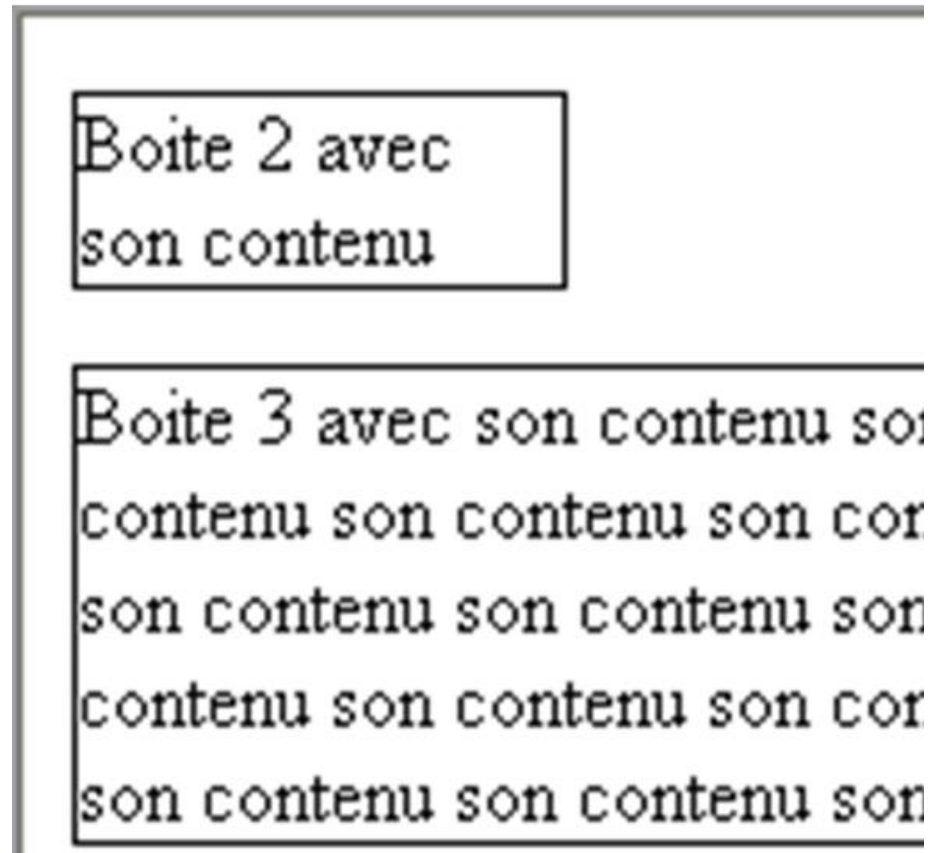
Positionnement flottant

- La propriété clear permet d'interdire le « voisinage » d'un élément « flottant ».
- Elle accepte 3 valeurs
 - **left** : interdit le « voisinage » d'un élément « flottant » à gauche
 - **right** : interdit le « voisinage » d'un élément « flottant » à droite
 - **both** : interdit le « voisinage » d'un élément « flottant » des deux cotés

```
#boite2 {
    float: left;
    width: 100px;
    border: solid 1px black;
}

#boite3 {
    clear: both;
}

<div id="boite2">Boite 2</div>
<div id="boite3">Boite 3</div>
```



Par défaut, si l'attribut média n'est pas spécifié, les directives s'appliquent pour tous les médias (all)

@-règles

@media

- Indique un bloc de règles qui ne s'appliquent qu'aux médias spécifiés.
- Cette règle est utilisable dans les feuilles de style liées ou importées, ou dans les directives fixées dans l'en-tête des documents HTML.

```
@media print {  
    body {  
        background-color: #ffffff;  
        color: #000000;  
    }  
}
```

Il est possible d'indiquer une liste de médias en les séparant par des virgules

- all Tous médias
- aural Parole et synthétiseurs de sons
- braille Interface tactiles braille
- embossed Impression braille
- handheld Petits dispositifs ou dispositifs tenus en main
- print Impression
- projection Projection
- screen Ecrans d'ordinateurs
- tty Teletypes, terminaux...
- tv TV

Erreurs classiques

- L'élément `div` ne remplace pas tous les éléments : Il vaut mieux privilégier les éléments HTML ayant une valeur sémantique.
- Il n'est pas indiqué de créer des divisions « à outrance » dans les documents.
- Tous les éléments n'ont pas besoin de « `class` ». Multiplier les classes et par conséquent les attributs d'éléments HTML (pour les rattacher à une classe) rend le code beaucoup plus lourd. Il est préférable d'utiliser la sélection hiérarchique plutôt que de vouloir « typer » tous les éléments par des classes.