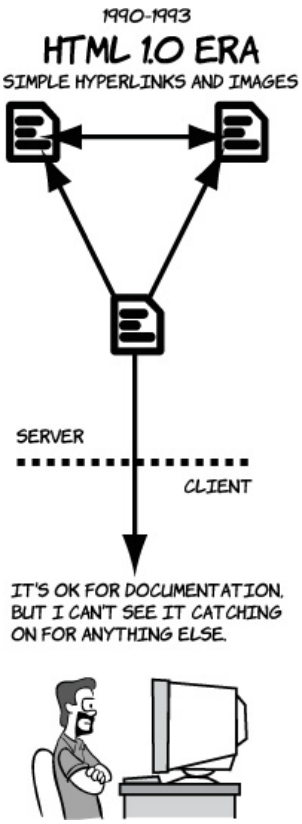
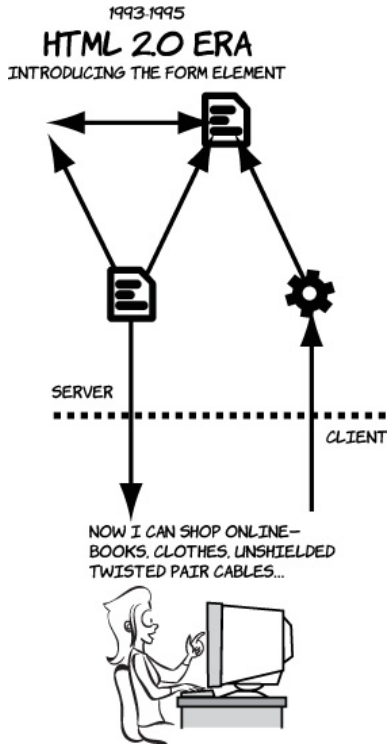
HTML/CSS

HTML est un langage de description de documents hypertextes. Les documents hypertextes sont constitués de titres, de paragraphes, de listes, d'images, etc... et, surtout, de liens vers d'autres documents hypertextes ; ce sont les liens qui constituent la partie hypertexte.

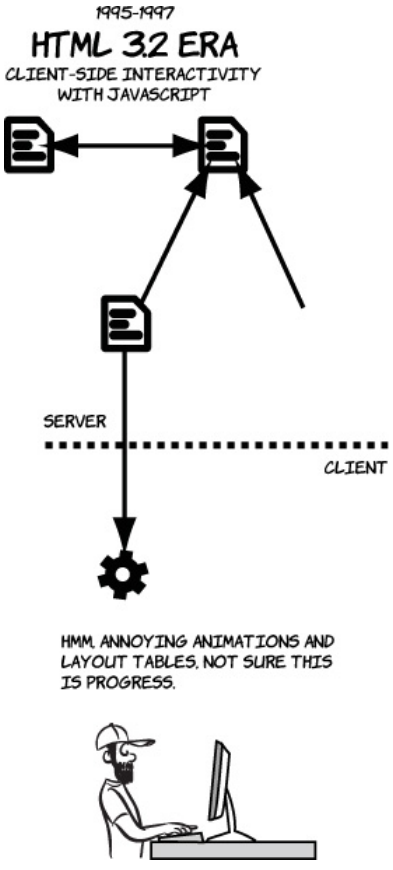
Les différentes version HTML



Lien sur des pages avec textes et images et liens vers d'autres pages. Aucune interaction à part les liens. HTML a été développé par Tim Berners-Lee au CERN de Genève pour faciliter la communication entre physiciens sur les accélérateurs de particules.



Possibilité d'utiliser la balise <form> pour entrer des données.



Utilisation de javascript pour animer la page, ici des actions peuvent s'effectuer côté client mais problèmes de compatibilités dûs à la guerre des navigateurs et des différentes versions Javascript.

D'où la création du W3C[1], consortium proposant de normaliser les besoins en publiant plusieurs « *drafts* » (brouillons),puis passent au stade « *Proposed Recommandations* » et un vote est proposé pour être repassé en *draft* ou passer en *Recommandation* sur lesquelles les éditeurs de logiciels devront se baser.

[1] World Wide Web Consortium - Tim Berners-Lee en est le dirigeant

Normalisation Javascript ECMAScript 5,6,7

Les pages WEB sont de véritables application (SPA) grâce à la normalisation des moteurs Javascript et l'éco-système extrêmement foisonnant.

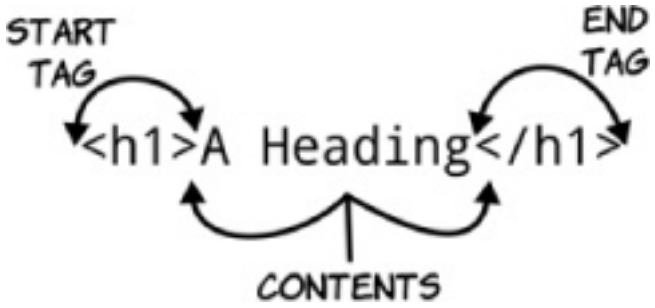
XHTML1.0 : HTML respecte les règles XML qui consiste à vérifier la cohérence de la page web à l'aide d'une « grammaire » DTD. ça permet de mieux détecter les erreurs mais ce régime est un peu draconien de plus IE ne gère pas correctement le XHTML.

L'élément <!DOCTYPE> indique que le document actuel appartient à un type spécifique de HTML

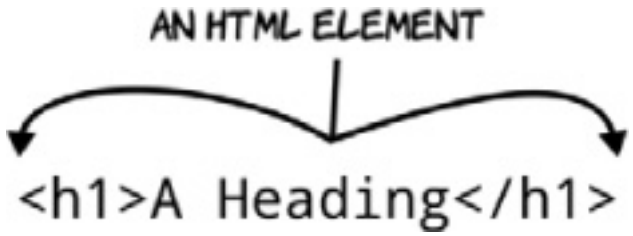
HTML5	<!DOCTYPE html>
HTML 4.01	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd";>
XHTML 1.0	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

HTML4

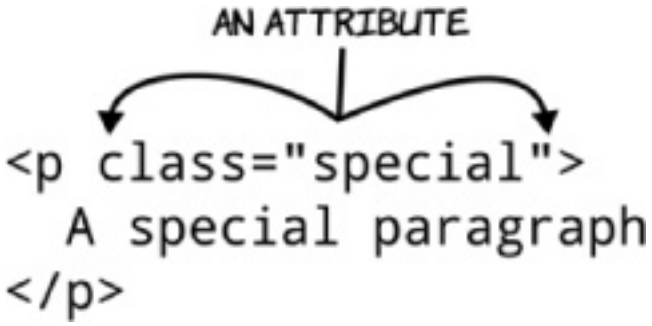
LES BALISES (TAGs)



HTML ELEMENT



ATTRIBUT D'UN ELEMENT (ID , CLASS)



Les éléments peuvent contenir du texte mais aussi d'autres éléments . L'élément contenant étant le **parent**.

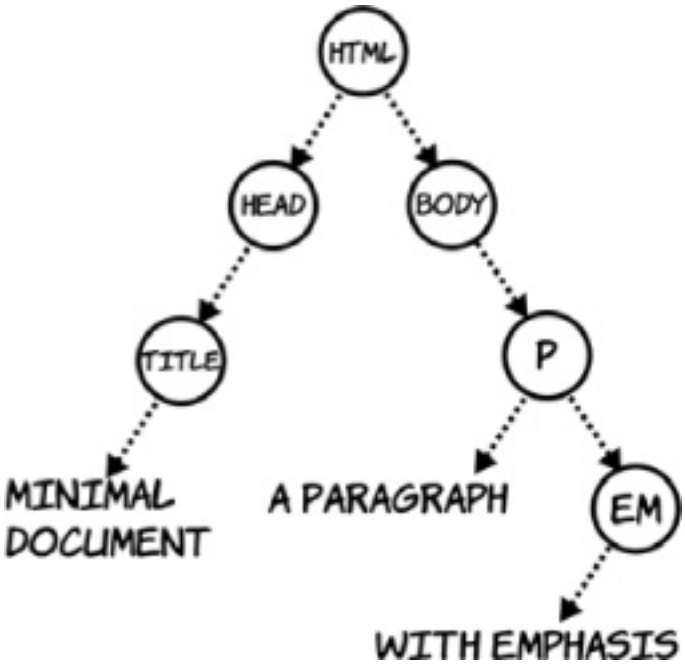
Testons nos scripts HTML avec l'éditeur ligne suivant

https://fr.w3docs.com/tools/editor

Un document HTML est un arbre partant du tag <html> avec 2 enfants <head> et <body>.
<head> pour les méta-données (metadata -> données à propos des données) exemple

<body> pour le contenu de la page

```
<html>
<head>
        <title>Minimal document</title>
</head>

<body>
<p>A paragraph
        <em>with emphasis</em>
</p>
</body>
</html>
```



Une balise doit toujours être fermée par un slash …….

<HEAD>

<title> le titre qui apparait en haut du navigateur </title>

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

<meta name="Author" content="J. Smith">
<meta name="keywords" lang="fr" content="AFPA, webdev, brest " >[2]
<meta name="description " lang="en" content="course, HTML" >

<meta charset="UTF-8">

Les américains ont défini les caractères représentés par des nombres
Ce sont les caractères ASCII

0 NUL	32 espace	64 @	96 `
1 SOH	33 !	65 A	97 a
2 STX	34 "	66 B	98 b
3 ETX	35 #	67 C	99 c
4 EOT	36 $	68 D	100 d
5 ENQ	37 %	69 E	101 e
6 ACK	38 &	70 F	102 f
7 BEL	39 '	71 G	103 g
8 BS	40 (	72 H	104 h
9 HT	41)	73 I	105 i
10 LF	42 *	74 J	106 j
11 VT	43 +	75 K	107 k
12 FF	44 ,	76 L	108 l
13 CR	45 -	77 M	109 m
14 SO	46 .	78 N	110 n
15 SI	47 /	79 O	111 o
16 SLE	48 0	80 P	112 p
17 CS1	49 1	81 Q	113 q
18 DC2	50 2	82 R	114 r
19 DC3	51 3	83 S	115 s
20 DC4	52 4	84 T	116 t
21 NAK	53 5	85 U	117 u
22 SYN	54 6	86 V	118 v
23 ETB	55 7	87 W	119 w
24 CAN	56 8	88 X	120 x
25 EM	57 9	89 Y	121 y
26 SIB	58 :	90 Z	122 z
27 ESC	59 ;	91 [	123 {
28 FS	60 <	92 \	124 |
29 GS	61 =	93]	125 }
30 RS	62 >	94 ^	126 ~
31 US	63 ?	95 _	127 ■

[2] Plus nécessaire depuis 10 ans. google n'a jamais utilisé le meta *keyword*

Le problème se pose pour les autres caractères dans le monde (accents français, n tilde pour l'espagnol ou le breton, et alphabets asiatiques

D'ou l'invention de Unicode.

Unicode prend 2 octets au lieu d 1 seul octet pour l'ASCII

UTF-8 permet d'utiliser l'ASCII quand c'est possible puis l'UNICODE si nécessaire.

Il existe des Entités HTML pour représenter certains caractères unicode mais c'est déconseillé car peut lisible, il vaut déclarer l'encodage de votre éditeur qui permet l'écriture Unicode (ou UTF-8).

é

è
…etc…

Base

créons un fichier index.html

```
<html>
    <body>
        mon texte
    </body>
</html>
```

Toujours après un tag ouverture => un tag de fermeture

Structure d'une page head

```
<html>
    <head>
        <title> Titre de notre page </title>
    </head>
    <body>
        mon texte
    </body>
</html>
```

Paragraphes et Titres dans <body>

```
<html>
    <head>
        <title> Titre de notre page </title>
    </head>
    <body>
        <h1>Tite de niveau 1</h1>
        <p>Paragraphe 1</p>
        <p>Paragraphe 2</p>
        <p>Paragraphe 3</p>

    </body>
</html>
```

Line break

```
<br/>
```

Lien

```
<a href="http://reddit.com" target="_blank">le site
reddit.com</a>
```

Entêtes et paragraphes

```
<html>
<head>
    <title>
      Headings and
      implicit structure
    </title>
</head>
<body>
    <h1>The main heading</h1>
    <p>Main introduction</p>
    <h2>First section</h2>
    <p>Section introduction</p>
    <h3>Subsection heading 1.1</h3>
    <p>Subsection 1.1</p>
    <h2>Second section</h2>
    <p>Section introduction</p>
    <h3>Subsection heading 2.1</h3>
    <p>Subsection 2.1</p>
    <h4>Sub-subsection</h4>
    <p>Subsection 2.1.1</p>
</body>
</html>
```

Listes:

non numérotées :

```
<ul>
    <li>List item</li>
    <li>List item</li>
    <li>List item</li>
</ul>
```

numérotées :

```
<ol>
    <li>List item</li>
    <li>List item</li>
    <li>List item</li>
</ol>
```

Balise Table :

```
<table>
   <caption>Passagers du vol 377</caption>

   <tr>
       <th>Nom</th>
       <th>Âge</th>
       <th>Pays</th>
   </tr>
   <tr>
       <td>Carmen</td>
       <td>33 ans</td>
       <td>Espagne</td>
   </tr>
   <tr>
       <td>Michelle</td>
       <td>26 ans</td>
       <td>États-Unis</td>
   </tr>

</table>
```

Saisie de donnée avec <form><input>

En html4 pour envoyer des données il y a une balise qui permet la saisie d'éléments , il s'agit de la balise input qui peut être répétée et regroupée dans une balise <form>

Exemple

```
<!DOCTYPE html>
<html>
<body>

<h2>The method Attribute</h2>

<p>Ce formulaire sera soumis en utilisant la méthode GET:</p>

<form action="test.php" target="_blank" method="GET">
  <label for="fname">Prénom:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Nom:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>

<p>Après l'envoi, vous remarquerez que les valeurs du formulaire
sont visibles dans la barre d'adresse du nouvel onglet du
navigateur.</p>

</body>
</html>
```

DANS test.php

```
<?php
echo 'Bonjour ' . htmlspecialchars($_GET["fname"]) . ' ' .
htmlspecialchars($_GET["lname"]) ;
?>
```

action: spécifie le programme qui va traiter les données saisies (en backend)

target: type d'affichage de la réponse

- _blank : nouvelle page
- _self : dans la même page

method : GET ou POST

- GET : déconseillé car les attributs sont limités en taille.

POST : passe par le protocole http , pas de limite (ou uniquement imposé par le programme de réponse)

`<input type=…`

Différent types en HTML4 (assez limité)

text :	texte alphanumérique
radio :	boutons radio
checkbox :	cases à cocher
submit :	bouton d'envoi des données
file :	envoi de fichier
hidden :	permet d'envoyer des élément cachés utilisés par le programme receveur

Type file:

```
<label for="avatar">Choisissez une photo de profil:</label>

<input type="file"
       id="avatar" name="avatar"
       accept="image/png, image/jpeg">
```

Les élément neutres et

Tous les élément ne sont pas marqués sémantiquement.

```
<div> de type block permet de regrouper des éléments
<span> changer une partie de texte dans un paragraphe par exemple.

<div class="person">
    <p class="full_name">
        <span class="first_name">
             Rob
         </span>
        <span class="surname">
            Crowther
         </span>
     </p>
    <p class="hometown">
        London
     </p>
</div>
```

Liens et ressources:

```
<p>Use
  <a href="http://www.google.com/">
    the Google
  </a>.
</p>
```

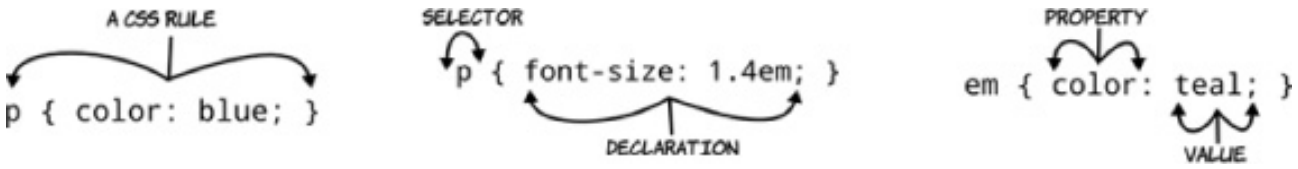
Permet de se connecter à un site web

```
<p>Go to
  <a href="pages/links-3.html">
    another page
  </a>.
</p>
```

Ou sur une page locale à partir de la racine du site.

```
<a href="#section-1>plus bas </a>
…
<div id="section-1">textes…</div>
```

CSS



Une règles CSS (Cascade Style Sheet) est constituée d'un sélecteur, d'une liste de déclarations avec le séparateur « : » le tout englobé dans 2 accolades

Si un élément du fichier HTML correspond au sélecteur de cet élément les déclarations sont appliquées à cet élément

```
<!DOCTYPE html>
<html>
<head>
 <title>Simple CSS Example</title>
  <style>
  body {
    font-family: "Komika Hand";
    font-size: 250%;
   }
  p {
    color: blue;
    font-size: 1.4em;
  }
  em {
     color: teal;
  }
  </style>
</head>
<body>
  <p>A paragraph
    <em>with emphasis</em>
  </p>
</body>
</html>
```

```
<p style="color: red;">Another paragraph</p>
```

On peut directement appliquer un style dans un élément (déconseillé).Dans cet exemple le style est inclus dans la page web mais il est préférable de l'extraire dans un fichier externe puis de le lier ainsi (dans l'entête html <head>)

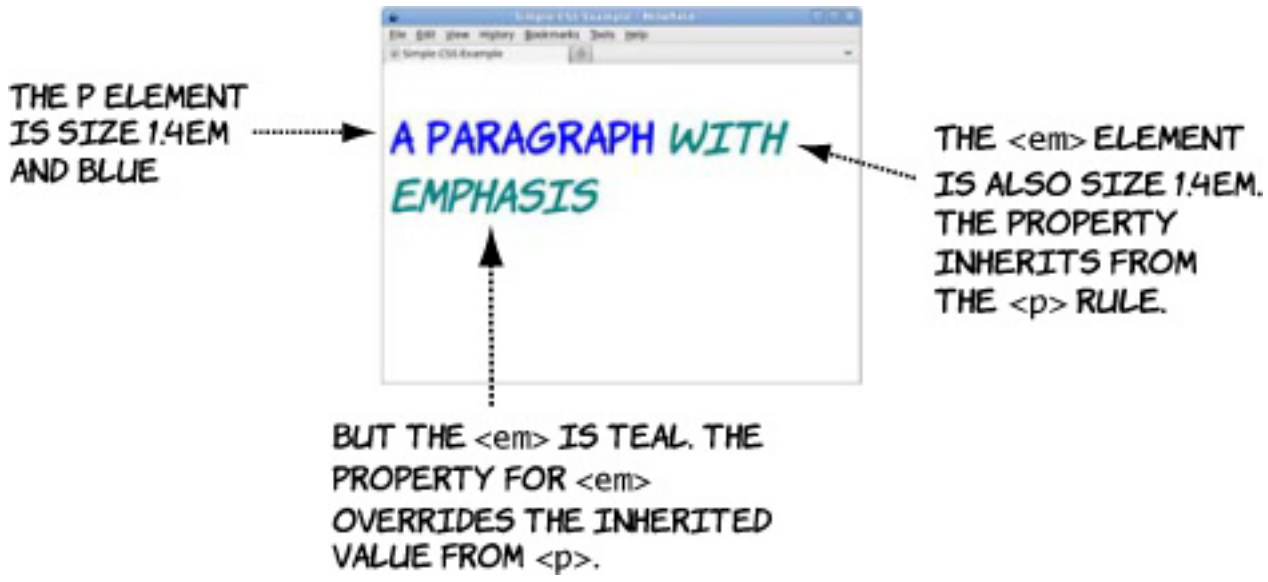
```
<link href="/bundles/common/css/master-30373b45.css" type="text/
css" rel="stylesheet" >
```

Mais il peut y avoir combinaison des deux (CASCADING)

```
<head>
    <title>CSS Cascade 1</title>
    <link href="style-1.css" rel="stylesheet">
    <style>
    p {
        color: black;
        background-color: white;
    }
    </style>
</head>
```

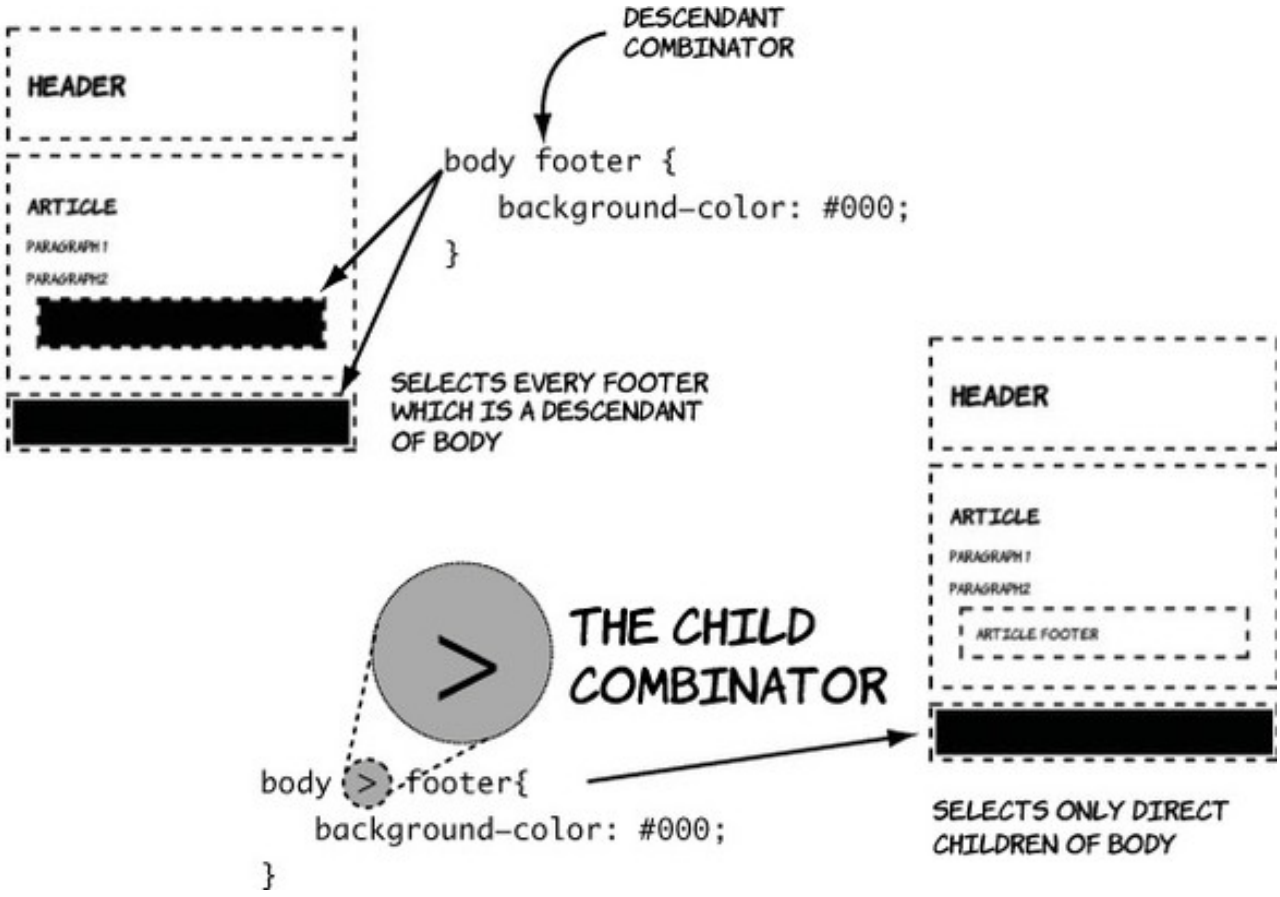
Héritage

```
p    { color: blue; font-size: 1.4em; }
em   { color: teal; }
```



Le combinateur >

```
<header>
  <h1>Header</h1>
</header>
<article>
  <h1>Article</h1>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
  <footer>Article footer</footer>
</article>
<footer>
  Body footer
</footer>
```



LES SELECTEURS

tags html
classes
id

```
<p>le paragraphe </p>
<p class="post">un post</p>
<p id="para-1">paragraphe1</p>
```

CSS:

```
p          { color: red        }
.post      { color: green      }
#para-1    { color: silver     }
```

Il est possible de combiner les classes dans un seul élément, un élément peut avoir un seul id et une ou plusieurs classes.

```
<p id="monid" class="ma-classe1 maclasse-2"> para……</p>
```

Tester ceci

```
<p id="monid" class="ma-classe1 maclasse-2" style="color:red" >
para……</p>
```

Propriétés et valeurs

Couleurs en hexadécimal

https://fr.w3docs.com/apprendre-html/les-couleurs-html.html

tailles

px	Pixels	Length in pixel units. The actual size is determined by monitor resolution.
pt	Points	A measure from typography, equivalent to 1/72 of an inch.
cm	Centimeters	Absolute length in centimeters.
em	Ems	Size of the capital M in the current font.
%	Percentage	Length as a proportion of the size of the element's parent.

Borders

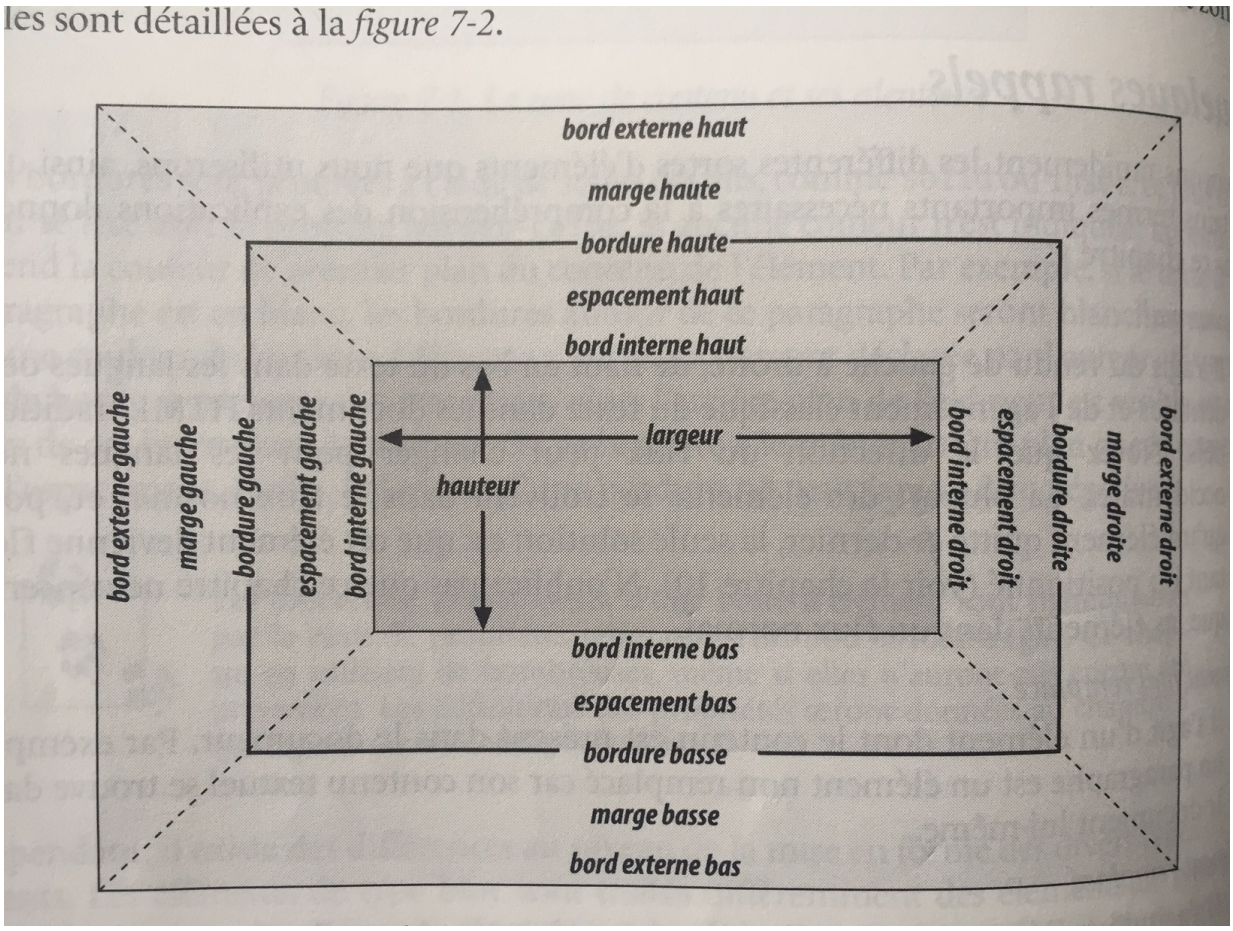
Un border (encadrement d'un block) a une largeur , un type et une couleur

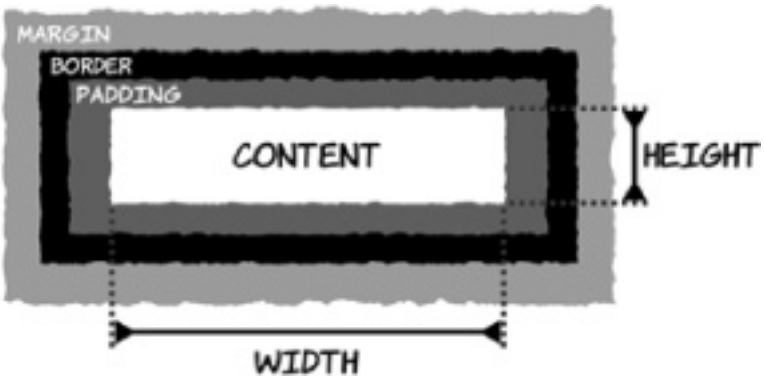
```
.one {
     border-width: 5px;
     border-style: solid;
     border-color: #999;
}
```

Backgrounds

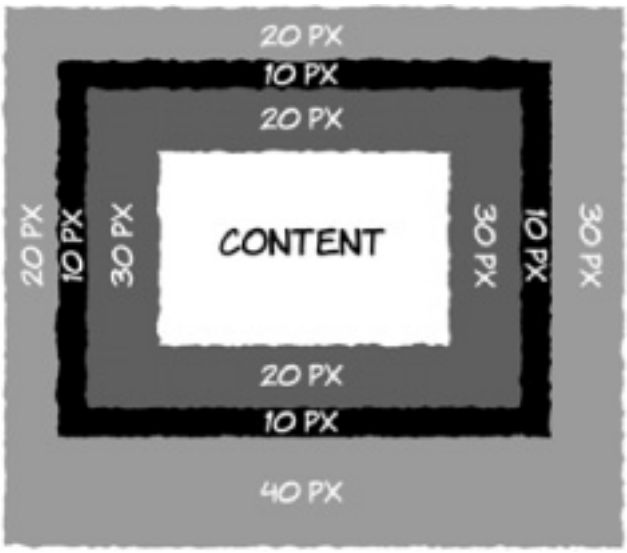
background-color	red, #f00, rgb(255,0,0)	Any valid color.
background-image	url(background.png)	A link to an image.
background-repeat	repeat, no-repeat, repeat-x, repeat-y	Should the background image tile across the background, or only appear once?
background-position	top left, 100px 200px, 50% 50%	Where should the first background image be placed?

BOX MODEL





padding: 20px 30px;
border-width: 10px;
margin: 20px 30px 40px 20px;



Les modes d'affichage block , inline et inline-block

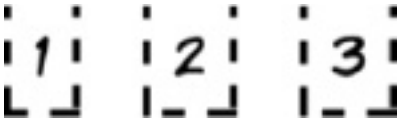
```
<div>1</div>
<div>2</div>
<div>3</div>

div {
    width: 2.5em;
    height: 2.5em;
    margin: 0.5em;
    padding: 0.5em;
    border: 5px dashed black;
}
```

S'affichera en mode par default ainsi car div est un élément de type **bloc**

Ajout dans le style

div { display: inline ;}

S’affiche ainsi :



Les éléments sont sur la même ligne mais bien plus petit . Les propriétés width et height ne s’appliquent pas aux éléments inline. Chaque à maintenant la taille de son contenu plus les marges spécifiées.

Pour cela utiliser `inline-block`

Saisie de donnée avec <form><input>

En html4 pour envoyer des données il y a une balise qui permet la saisie d'éléments , il s'agit de la balise input qui peut être répétée et regroupée dans une balise <form>

exemple

```
<!DOCTYPE html>
<html>
<body>

<h2>The method Attribute</h2>

<p>This form will be submitted using the GET method:</p>

<form action="action_page.php" target="_blank" method="get">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>

<p>After you submit, notice that the form values is visible in the address bar of the new browser tab.</p>

</body>
</html>
```

action: spécifie le programme qui va traiter les données saisies (en backend)

target: type d'affichage de la réponse
- _blank : nouvelle page
- _self : dans la même page
- _parent _top _framename : ne pas utiliser frame abandonnés et iframe déconseillés

method : GET ou POST

GET : déconseillé car les attributs sont visibles dans la barre d'URL et sont limités en taille et empêche les attributs cachés (voir plus bas input).

POST : passe par le protocole http , pas de limite (ou uniquement imposé par le programme de réponse)

`<input type=…`

Différent types en HTML4 (assez limité)

text : texte alphanumérique
radio : boutons radio
checkbox : cases à cocher
submit : bouton d'envoi des données
file : envoi de fichier
hidden : permet d'envoyer des élément cachés utilisés par le programme receveur

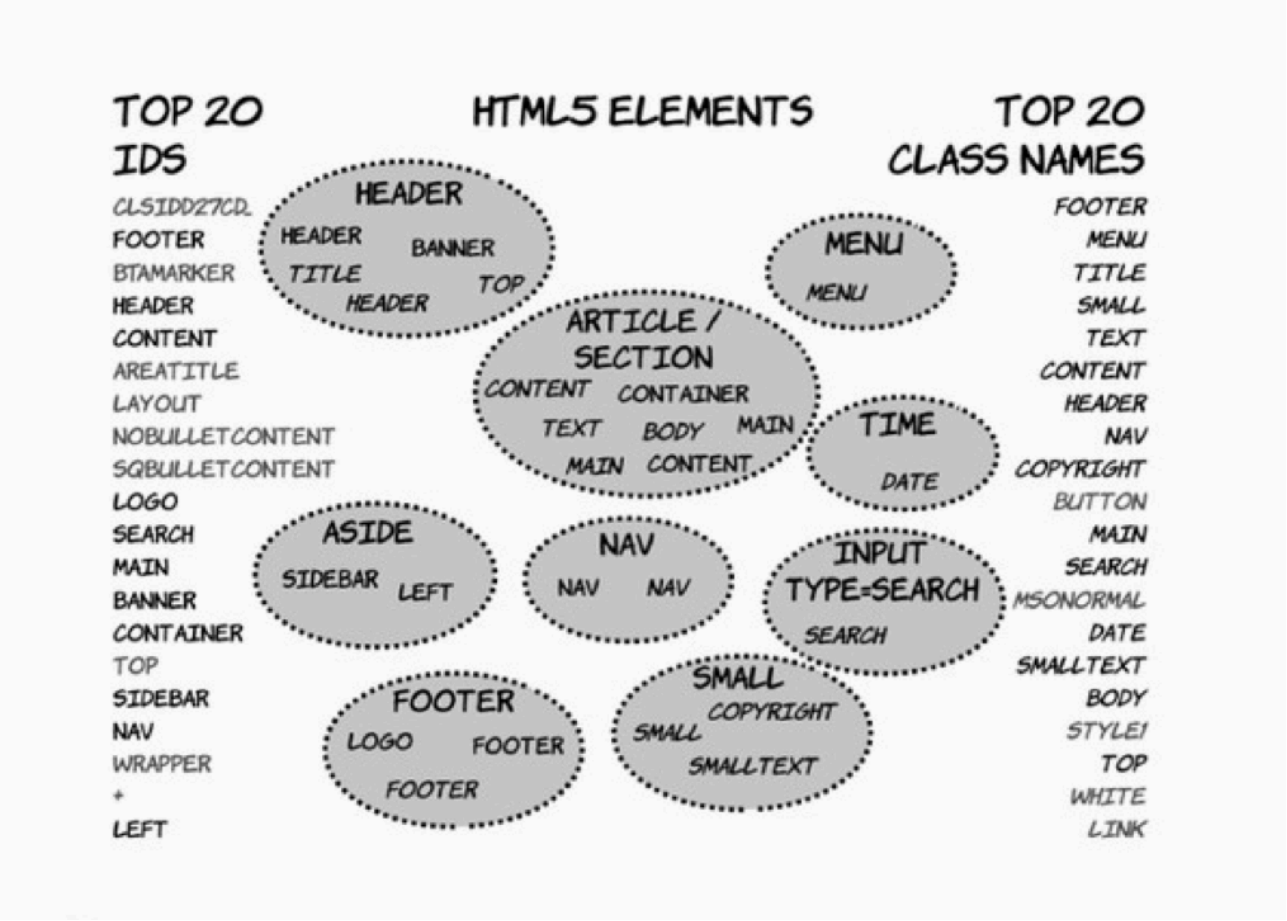
HTML5 : Sémantique

Les élements HTML sont censés avoir un sens, c'est ce à quoi ont pense quand on parle de sémantique. Les élément HTML4 n'ont pas vraiment de sens.

Par exemple la nouvelle balise <address> a plus de signification que , ça ne change en rien l'affichage mais ça donne plus d'indications au système qui va « parser » cette page web, c'est utile par exemple aux convertisseurs text-to-speech ou à une application de mashup web.

C'est en analysant les attributs **class** et **id** de nombreux sites web que les concepteurs HTML5 ont défini de nouvelles balises voyant que certaines **class** ou **id** se répétaient très souvent <ol id=nav>   …

En faisant des statistiques sur des millions de pages ils ont fait remonter ces informations (id et class) :



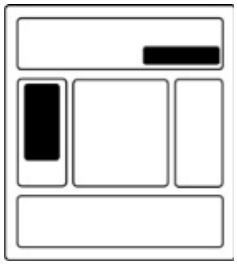
<header>

<hgroup> regroupement de tags d'entête

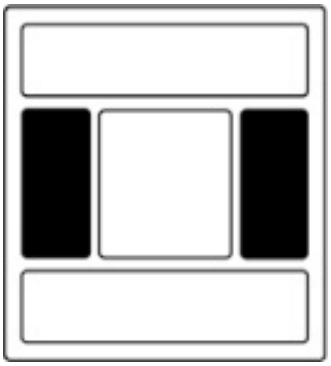
<section>

<hgroup>

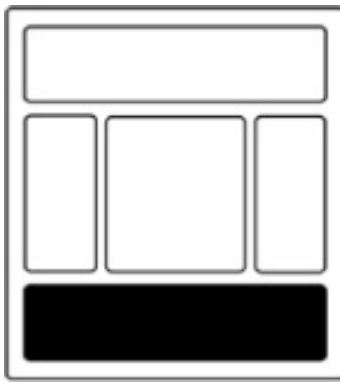
<nav>



<aside>



<footer>



<small> souvent dans footer ou aside , on y trouve copyright, mentions légales ..etc..

<time>

exemple

```
<time datetime=« 2011-06-01">today</time>
<time datetime="2011-06-01T18:00:00+01:00">6 o'clock on 1/6/2011</
time>
```

<place>

<figure> image avec légende.

<fieldset>   séparateur de formulaires (border + légende)

exemple

```
<fieldset style="colour" title="Other form elements">
     <legend>Example</legend>
     <label>
     Upload file
          <input type="file" name="name">
          </label>
</fieldset>
<input type="submit">
```

Retour sur la balise <input> en HTML5

Type HTML5 rajoutés

number, range, date, time, email, url

Chercher de la documentation et tester ces types.

La validation

required, min, max, pattern (regexp),
les types email et url font l'objet d'une validation

nouveaux type (pas encore tout à fait validés sur tous les navigateurs) :

color
meter
progress

Le placeholder:

indication dans le input

<input type=« email" placeholder="email@example.com">

Les médias:

images:

```
<img src=« dust-puppy.svg" alt="An image of Dust Puppy">

<img src="dust-puppy.svg" width="252px" height="356px" alt="An
image of Dust Puppy">
```

Indiquer la taille des images : svg OUI !!! jpg,png,gif `-> INTERDIT PAR MOI !!!`

Les images peuvent être embarquées si elle sont en base64.
https://fr.w3docs.com/tool/
https://fr.w3docs.com/tools/image-base64

Audio/ Videos:

Les sons et videos sont stockés en format compressé. Ils doivent pour cela être encodés dans ce format puis décodés. Le logiciel qui permet l'encodage et le décodage est appelé **codec.** Ces codecs sont généralement reconnus par l'extension des fichiers.

exemple: mp3 pour l'audio et mpeg2 pour la video (noté .mp4)

format ogg / ogv (firefox)
web (google video)

		WAV	OGG	MP3	AAC	WebM
Browser support quick check: audio codecs		8	5	5	5	8
		3.5	3.5	~	~	4
		~	~	9	9	[*]
		10.5	10.5	~	~	11.1
		4	[**]	4	4	[**]

Pour écouter sur IE,Firefox, chrome, safari

```
<audio id="myaudio" controls>
    <source src="myaudio.mp3" type="audio/mp3">
    <source src="myaudio.ogg" type="audio/ogg">
    No audio support!
</audio>
```

HTML5 creation du tag video

Pour lire les vidéos auparavant on passait par flash

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/
    swflash.cab#version=6,0,40,0"
 width="320" height="240"
 id="myvideoname">
    <param name="movie"
      value="myvideo.swf">
    <param name="quality" value="high">
    <param name="bgcolor" value=#ffffff>

    <embed href="myvideo.swf
     quality="high" bgcolor="#ffffff"
           width="320" height="240"
           name="myvideoname"
           type="application/x-shockwave-flash"
           pluginspage="http://www.macromedia.com/
             go/getflashplayer">
    </embed>
</object>
```

Il fallait utiliser un **plug-in**

plug-in: extension qui permet navigateur de rendre un contenu par un programme externe.

Mais suite à une déclaration de Steve Jobs annonçant qu'il ne supporterait pas flash sur iPhone, flash fut petit à petit abandonné[3].

[3] https://flowplayer.com/blog/the-rise-and-fall-of-flash

Une video c’est un conteneur et un codec

Le **conteneur** est un format de fichier qui contient les flux audio et vidéo, l’information codec et les métadonnées.

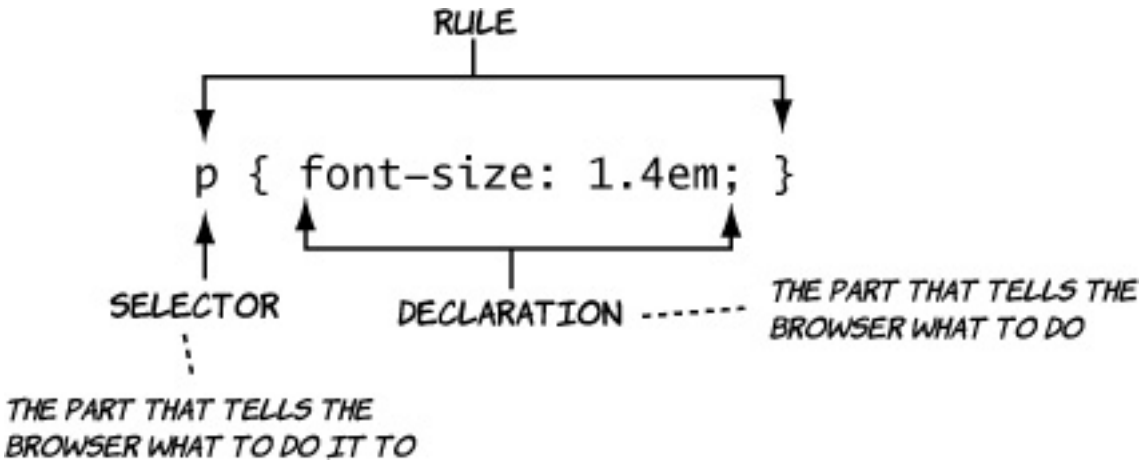
Un **codec** (acronyme de codage-décodage) est un algorithme de compression / décompression d’un signal audiovisuel numérique.

Le codec est un programme qui se trouve sur la machine (il y en a des centaine). Le codec le plus courant est le mpeg4 ou H.264 créé par Apple .

Actuellement la plupart du temps on trouve ces fichiers video avec le suffixe .mp4.

Aujourd’hui HTML5 permet de s’affranchir des plug-in et il est recommandé d’encoder en MP4/h.264, AAC (Advance Audio Encoding)

CSS



Il y a 3 façons d'insérer une feuille de style

feuille de style externe nommée unnom.css et importée dans la section head de la page HTML ainsi

```
<link href='/chemin/de/monfichier/unnom.css rel='stylesheet' type='text/css' />
```

Déclarer un tag <style> directement dans la page html (déconseillé)

```
<head>
<style>
body { color:red ;}
</style>
</head>
```

Appplique l'attribut style directement dans un élément (encore plus déconseillé)

```
<div style="color:red;font-size:16px'>lorem ipsum.....</div>
```

Nous verrons plus tard que des actions JQuery insèrent directement ces directives css dans les éléments.

Typographie:

- font-family
- font-size
- font-style
- font-weight
- color

font-family: 'police voulue','web safe fonts','generic'

police voulue : cherche la police dans votre PC , ou sera inséré par CSS

web safe font : police théoriquement universelle pour tous les navigateurs (genre Verdana, Helvetica…etc) on peut en mettre plusieurs à la suite.

generic : dernier recours, cinq options : serif, sans serif, monospace , cursive, fantasy (en general : serif)

Polices google:

CSS permet d'insérer des polices google (fonts.google.com)

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset= "utf-8">
        <link rel="stylesheet" href="cours.css">
        <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Open+Sans|Roboto">
    </head>

    <body>
        <h1>Un titre de niveau 1</h1>

        <p>Un premier paragraphe</p>
        <p id="p2">Un autre paragraphe</p>
    </body>
</html>
```

```css
body{
    font-family: "Open sans", Georgia, serif;
}

#p2{
  font-family: Roboto, Verdana, sans-serif;
}
```

fonts.google.com

https://www.dafont.com/fr/

Utiliser une police externe avec CSS

Comme pour les videos il faut plusieurs type de fichiers pour exploiter les polices sur les différents navigateurs

Pour ça il faut uploader une police choisie sur DAFONT par exemple puis générer le ou les fichiers nécessaire avec un site très utile : https://www.fontsquirrel.com/tools/webfont-generator

- Télécharger une police sur Dafont
- la passer dans Fontsquirrel
- télécharger le fichier et décompresser
- On aura des fichiers de type .woff reconnus aujourd'hui
- puis insérer l'exemple css fourni

```
/*! Generated by Font Squirrel (https://www.fontsquirrel.com) on August 22, 2020 */

@font-face {
   font-family: 'quadrantaregular';
   src: url('quadranta-regular-webfont.woff2') format('woff2'),
        url('quadranta-regular-webfont.woff') format('woff');
   font-weight: normal;
   font-style: normal;
}
```

Puis utiliser cette font-family, exemple:

```
h1 { font-family: 'quadrantaregular' ;}
```

TP: Tester avec l'exemple fourni par Fontsquirrel

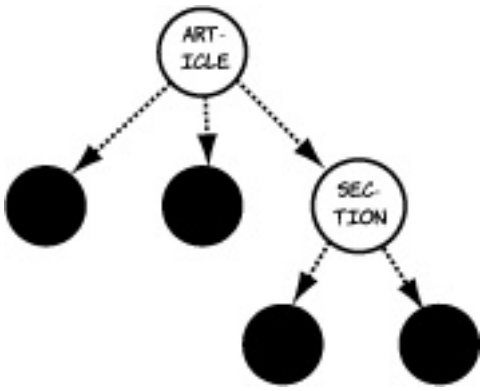
CSS3 ajoute l'utilisation de combinateurs et de pseudo-classes

Combinateur

Considérons

```
<article>
        <p></p>
        <p></p>
        <section>
                <p></p>
                <p></p>
        </section>
</article>
```

Donne le DOM suivant



si on applique le style

article p { color : red; }

l'espace après article indique que le style s'appliquera aux sélecteur « P » descendants de « article ».

mais si on utilise le combinateur greater-than ainsi

article > p {color : red }

le style sera appliqué aux enfants directs de « article »

Combinateur de voisin direct

h1 + p { background-color: #000; }

Elément directement situé après et seulement celui-là

Général sibling

h1 ~ p { background-color: #000; }

Pseudo-classes

ul li:first-child
ul li:last-child

```
li:last-child {
  background-color: #000;
}
li:last-child a {
  background-color: #fff;
}
```

Le n-ième élément

nth-child(1), :nth-child(2), :nth-child(3)

Tous les second éléments (ou pairs) . Utilisation du pattern « 2n"

```
li:nth-child(2n) {
  background-color: #000;
```

Pour les éléments impairs

```
li:nth-child(2n-1) {
        background-color: red ;
}
```

raccourcis

```
i:nth-child(odd) {
  background-color: #f00;
}
li:nth-child(even) {
  background-color: #006;
}
```

Selection des éléments par leur attributs

```
<span typeinfo="sport">page des sports</span>
<span typeinfo="economie">page economie</span>
```

style :

```
[type info]             {  font-size: 24px; }
[type info=sport]    { colour:red ; }
```

expression régulière

selection des élément possédant l'attribur itemprop dont la propriété commence par org

```
[itemprop^="org"] {
  outline: 4px dashed black;
  display: block;
}
```

Positions avec CSS

On connait déjà les élément de type <block> et <inline>

<inline-block> est un compromis entre les deux.

Un élément block a une hauteur , une largeur , un espacement , des marges

Un élement inline se trouve dans une ligne de texte mais n'a pas de hauter, largeur padding ou marge. Un inline-block combine les deux, il est dans une ligne de texte mais peut avoir une hauteur, largeur, padding, une marge ?

LE RESPONSIVE
Commencer par MOBILE FIRST !!

Devant la multiplication des tailles d'écrans il fallut adapter nos sites et applications web pour être lisibles pour différents type d'ordinateurs, smartphone, tablettes de différentes tailles, Desktop , machine texte-to-speech …etc…

On créa pour palier à ces problèmes le **responsive web**

Le responsive web c'est une seule url pour tous types d'appareils. Attention certains éditeurs préfèrent parfois développer un site web (une url différente) par type d'appareil, ça dépend de leurs contraintes.

Un site responsive nécessite CSS3 et HTML5

En s'efforçant d'adapter et de donner au site une présentation fluide, votre site a plusieurs avantages :

une présentation qui s'adapte aux différentes tailles d'écran
un site plus rapide

Pour cela fut créé une instruction CSS très utilisée en CSS3

media-queries

Une media-querie est un type de règle CSS qui limite la portée d'un style en fonction de facteurs définis par la requête. Chaque requête de média spécifie un type de média et un ensemble d'expressions qui sont vérifiées par le navigateur. Les types de médias possibles sont notamment : **screen** pour les écrans numériques, **print** pour les pages imprimées, et **all** pour tous les types de médias. Les expressions sont plus détaillées et comprennent des instructions telles que la largeur maximale ou l'orientation.

Type de Média

```
@media screen {
          p { font-family: sans-serif }
}

@media screen :     initialise la requête css déclarant quel média
                    on vise (ici un écran)
p {…} :             CSS appliqué aux paragraphes
```

Ici on applique la police sans-serif aux paragraphes qui seront affichés seulement sur un écran.

Type de médias CSS2

```
screen
     Écrans
handheld
     Périphériques mobiles ou de petite taille
print
     Impression
aural (CSS 2.0) / speech (CSS 2.1)
     Synthèses vocales
braille
     Plages braille
embossed
     Imprimantes braille
projection
     Projecteurs (ou présentations avec slides)
tty
     Terminal/police à pas fixe
tv
     Téléviseur
all
     Tous les précédents
```

Apport de CSS3

La philosophie des Media Queries en CSS3 est d'offrir un panel de critères plus vaste et plus précis, à l'aide de propriétés et de valeurs numériques, ainsi que de combinaisons multiples de ces mêmes critères. Le but est de cibler plus finement les périphériques de destination en fonction de leurs capacités intrinsèques.

L'écriture de ces requêtes est relativement explicite (en anglais) : une *media query* est une **expression dont la valeur est toujours vraie ou fausse**

Les opérateur logiques sont : **and**, **only** et **not** et peuvent être combinés avec des expressions numériques.

L'objectif de la conception réactive est d'éviter ce qu'Ethan Marcotte[4] appelle le "jeu à somme nulle" qui consiste à redessiner un site web pour tous les appareils et fenêtres possibles. Pour éviter cela, vous devez identifier les limites des endroits où vous modifierez votre mise en page pour répondre aux besoins du contexte changeant. Lorsque le site sur lequel vous travaillez passe de la largeur d'un appareil mobile à la largeur d'un desktop, à quel moment change-t-il ou "**break**" ?

[4] inventeur du responsive design

La clé de l'utilisation des requêtes médias dans une conception adaptée réside dans leur capacité à servir le CSS en fonction de la largeur de la fenêtre de visualisation, qui est la largeur de la fenêtre du navigateur.

Les requêtes " sont ce qu'on appelle des expressions, et ce sont les paramètres que le navigateur vérifie.

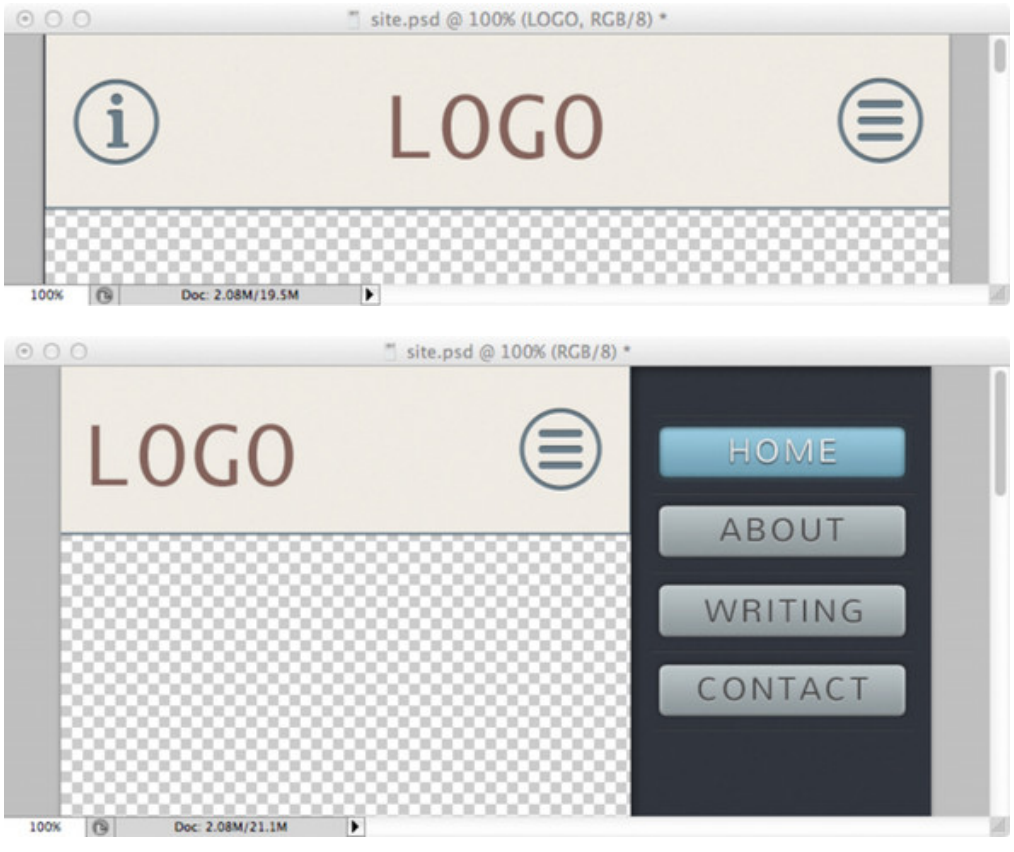
En utilisant les requêtes média, vous pouvez servir le CSS en fonction de la largeur de la fenêtre de visualisation ou de la largeur du périphérique.

Pour appliquer un CSS basé sur une fenêtre de 400 pixels ou moins, vous utiliserez une requête média comme celle-ci :

```
@media (max-width : 400px) { ... }

@media (max-device-width: 400px) { ... }
```

DESIGN MOBILE FIRST

Dans le header dans lequel on peut trouver logo , l'accès aux autres infos se fait par un bouton (exemple ici bouton info et bouton hamburger qui indique qu'il y a probablement un menu au touch sur ce bouton

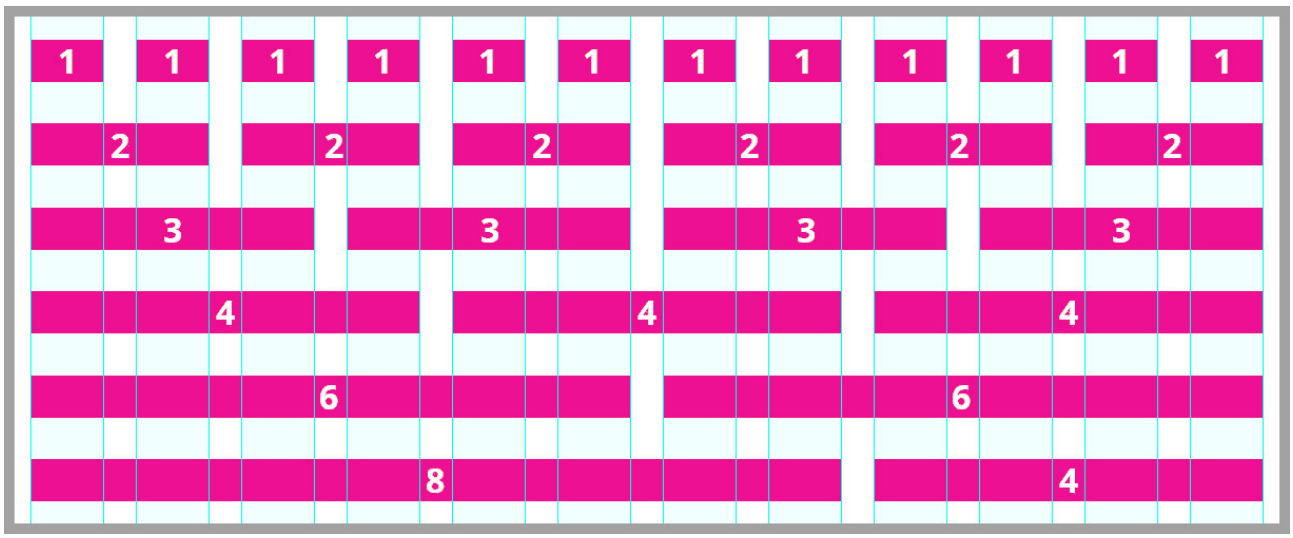


Utiliser Bootstrap

Le framework Bootstrap a été créé en 2011 par Mark Otto et Jacob Thornton, les développeurs de Twitte. Bootstrap est une collection d'outils HTML, CSS, JavaScript. Aujourd'hui, Bootstrap est devenu le framework front-end le plus populaire pour développer des projets responsive et mobile-first sur le web. La version actuelle est Bootstrap 4.

L'élément primordial en terme de responsive utilisé dans **bootstrap** est le système de grilles.

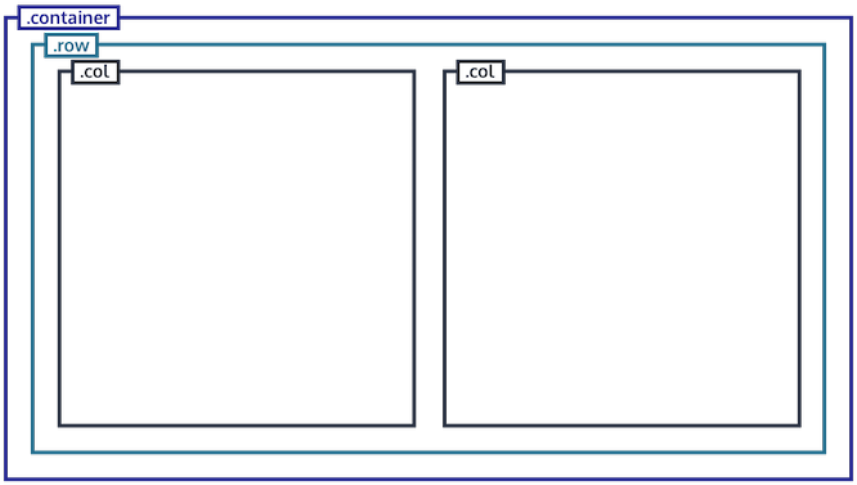
Le système est basé sur 12 colonnes que vous pouvez répartir comme vous désirez, exemple



12 colonne de 1 6 colonnes de 2 , 1/3 2/3 …etc…

Le système de grille Boostrap contient 4 classes :

- **xs** – pour les smartphones
- **sm** – pour les tablettes
- **md** – pour les ordinateurs classiques
- **lg** – pour les ordinateurs les plus grands



Exemple

```
<div class="container">
   <div class="row">
      <div class="col-xs">
         Première colonne
      </div>
      <div class="col-xs">
         Deuxième colonne
      </div>
   </div>
</div>
```

pas nécessaire de mettre de classe nombre si on désire simplement séparer en 2 la page

Autres exemple avec des tailles de colonnes différentes

```
<div class="container">
   <div class="row">
      <div class="col-xs-8">
         Première colonne
      </div>
      <div class="col-4-xs">
         Deuxième colonne
      </div>
   </div>
</div>
```

Threads

Traditionnellement Javascript s'éxécute sur une seul contexte d'éxécution appelé **thread.**

Les navigateurs sont devenus multithread pour être plus rapides , responsive, et plus résilient contre le mauvais code. mais javascript s'execute toujours sur un seul thread.

Pour palier à ce problème les web workers ont été créés.

Single-thread : le navigateur effectue une chose à la fois, s'il lance une fonction il attend qu'elle finisse et ne peut répondre aux événements (click,scroll…etc…)

Un PC a un seul processeur fonctionne de la même manière seulement il execute si rapidement les instructions qu'on a le sentiment qu'il execute plusieurs traitements à la fois.

Normalement c'est la cas pour Javascript mais une erreur ou un script mal intentionné peut planté le navigateur.

Les web-workers apporté par HTML5 permettent de coder en multithread

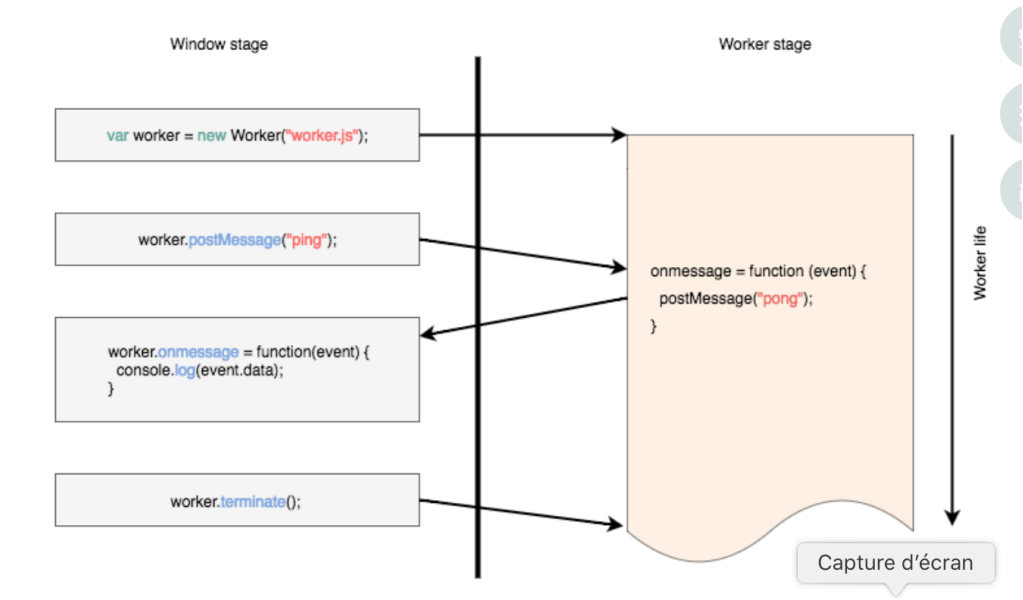
Un *Web Worker* est donc une façon d'appeler un script JavaScript qui va s'exécuter dans un Thread différent de celui du script qui le crée. Il est possible pour les deux scripts de communiquer ensemble via l'envoi et la réception de messages. Un *Web Worker* n'est cependant pas léger et n'est pas à utiliser à tort ou à travers. La création est un processus coûteux, notamment en mémoire. Il peut cependant grandement soulager l'affichage et la mise à jour d'une page web si vous avez à réaliser des calculs complexes ou des opérations régulières qui peuvent ne rien changer à votre page. Typiquement, si vous voulez faire des vérifications régulières sur une API et comparer vos données en cache à vos données affichées pour savoir ce que vous allez modifier, ce peut être une bonne idée d'utiliser un *Web Worker*.

exemple simple

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Worker example</title>
  </head>
  <body>
    <button onClick="ping()">Ping</button>
    <script>
      var worker = new Worker('worker.js');
      worker.onmessage = function (event) {
        console.log(event.data);
      }
      var ping = function() {
        worker.postMessage("ping");
      }
    </script>
  </body>
</html>
```

worker.js

```
// worker.js
onmessage = function(event) {
  console.log("message received in the worker");
  postMessage("pong");
}
```



tester (ne fonctionne pas en mode file://)

installation d’un serveur web pour test
npm install --global http-server
npx http-server