# Technologies du web

2 - CSS

# Séparation des pouvoirs

Le HTML définit la structuration des données

Le CSS définit la présentation des données

Le DOM (Document Object Model) est formé à partir de l’arborescence des balises HTML

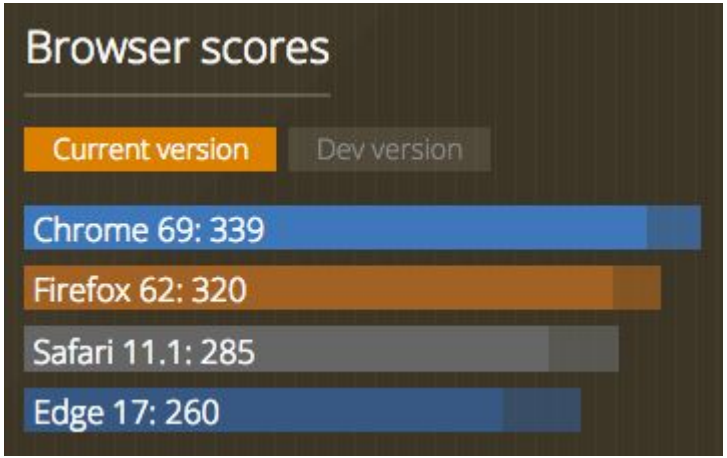
# Cascading Style Sheet

LA CASCADE !!!

# Cascading Style Sheet

Standardisation progressive, la norme CSS3 n’est pas encore supportée totalement par les navigateurs

https://**caniuse.com**



# CSS3

Depuis 1999, c’est lent. Trèèèès lent.

CSS permet de définir l’aspect de la page. Chaque élément balisé peut-être décoré (i.e: tout)

L’aspect peut varier
- selon les navigateurs (compatibilité et implémentations)
- selon les médias (taille de l’écran, type d’interface)

# CSS

Manipuler la couleur, la police, la taille, l’emplacement, etc.

# Inclure du CSS

Directement dans la balise avec **l'attribut style** => bad

Dans le head dans une **balise style** => meh

Dans le head via un link vers un (ou plusieurs) fichier(s) css dédié(s) => ok

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

# Syntaxe

```
sélecteur {
    propriété1: valeur;
    propriété2: valeur;
}
```

Sélecteurs :

**balise** (p, div, nav, etc.)
**.classe** (.bouton-ok, .lien-externe, .texte-intro, etc.)
**#id** (#menu, #contenu, etc.)

# Les balises

Exemples :

(provient du CSS appliqué par défaut par Firefox)

```
h1 {
  display: block;
  font-size: 2em;
  font-weight: bold;
  margin-block-start: .67em;
  margin-block-end: .67em;
}
```

```
table {
  display: table;
  border-spacing: 2px;
  border-collapse: separate;
  box-sizing: border-box;
  text-indent: 0;
  overflow-wrap: normal;
}
```

# Les classes

<balise class="nom-classe"> … </balise>

L’attribut **class** permet d’appliquer un style commun à une ou plusieurs balises différentes

<p **class="ombrage"**>...</p>

```
.ombrage {
        text-shadow: 1px 1px 2px pink;
}
```

# Les identifiants

**Unique** (!)

<balise id="nom-identifiant"> … </balise>

Attribut **id** d’une balise, permet de rendre l’élément unique et identifiable

<div **id="main-content"**> … 

```
#main-content {
        font-size: large;
}
```

# Groupes

Appliquer le même style à plusieurs sélecteurs

```
a, .blue, #logo {
    color: blue;
}
```

# Les attributs

Cibler une balise selon ses attributs

```
table[align="left"] { (recherche exacte)
  float: left;
}

a[href~="example.com"] { (recherche floue)
  color: red;
}
```

# Les états

Selon l’état de l’élément :

```
a:link {
        color: blue;
}

a:visited { color: darkblue; }

a:hover { color: darkblue; text-decoration: underline; }
```

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

# Combinaisons

Pour affiner la précision, des combinaisons sont possibles :

```
div.ombrage {
    text-shadow: 1px 1px 2px pink;
    display: inline-block;
}
```

# Héritage

Le style est parfois hérité par les enfants depuis l’élément parent. C’est le cas des polices, tailles de textes, couleurs notamment.

Pour forcer l’héritage des autres propriétés : **inherited**

Pour revenir à la valeur par défaut : **initial**

# Surcharge

En cas de définition concurrentes, c’est la dernière interprétée qui sera prise en compte.

Dans un même fichier ce sera la plus basse.

Dans une même page ça sera celle définie par le dernier fichier ou bien au plus proche de la balise (LA CASCAAAAADE)

# Sélecteurs multiples

Les styles se combinent (wouuuh)

```
a { color: blue }
.external { font-weight: bold; }
```

```
<p><a href="page.html" class="external">Coucou</a></p>
```

# Parenté

Les sélecteurs de famille

Tous les liens situés dans un élément de liste
**li a** { color: blue; }

Tous les liens dans un élément avec la classe menu
**.menu a** { color: blue; }

Seulement les liens enfants au premier dégré d’un élément de liste
**li > a** { color: blue; }

# Media Queries

```
@media print {

    /* règles spéciales imprimantes */

}

@media screen and (min-width: 200px) and (max-width: 640px) {

    /* règles petits écrans */

}
```

# CSS

Permet de gérer :

- L’apparence
- La taille
- Le positionnement

# Couleurs

color: blue;
background-color: #87a800
background-image: url("image_url.jpg");

Les couleurs peuvent être exprimées par leur :

- nom (blue, red, black, darkgreen, lightgrey, etc.)
- RGB/A : rgb(rouge, vert, bleu) rgba(rouge, vert, bleu, transparence)
  **rgb(255, 0, 0)** **rgb(100%, 0%, 0%)** **rgba(255, 0, 0, 0.5)**
- valeur hexadécimale : 3 bits, rouge vert bleu
  .sixdigit { color: **#0099CC;** } (alternative : .threedigit { color: #09C; })

Voir : https://en.wikipedia.org/wiki/Web_colors#CSS_colors

# Texte

Police :
font-size : 1.3em / 16px / 70% / …
font-weight : bold / normal
font-style : italic / oblique / normal
text-transformation : capitalize / lowercase / uppercase
text-decoration : none / underline / overline / line-through / blink

Alignement :
text-align : left / center / right / justify / start / end
vertical-align : baseline / middle / sup / super / top / bottom
text-indent : 3px

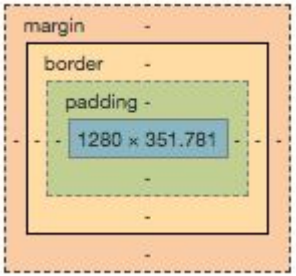
# Tailles

**px** : valeur absolue

**em** et **%** : en fonction du parent

Voir https://www.w3.org/Style/Examples/007/units.fr.html

# Chaque balise possède sa boite

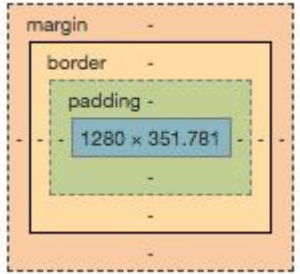


# Taille

- Taille fixe
  - **width**: 42px / 97%
  - **height**
- Taille contrainte
  - **min-**width: 42px / 97%
  - **max-**width
  - **min-**height
  - **max-**height

# Marges

- Extérieures :
  - **margin**-top
    -bottom
    -left
    -right




margin: 10px; == margin: 10px, 10px, 10px, 10px;
margin: 10px, 5px; == margin: 10px, 5px, 10px, 5px;
margin : 10px, 5px, 0; == margin: 10px, 5px, 0, 5px;

# Centrer grâce aux marges

margin: **auto**;
display: block;

(& parent’s display: block;)

Pratique pour les images

# Marges

Bordures :

- border-width: 2px;
- border-color: blue;
- border-radius: 5px;
- border-style: solid / dotted / dashed / double / …
- **border: 1px solid black;** (combo)
- border-collapse: collapse; (pour les tables)

# Marges

- Intérieures

**padding**-left
-top
-right
-bottom

# Overflow

Que faire en cas de dépassement d’une taille fixée ?

**overflow**: auto / scroll / visible / hidden

# Balise boîte ou balise en ligne

- **block** : occupe toute la largeur disponible du block parent, suivi d’un saut de ligne
- **inline** : se trouve dans un block, occupe la place nécessaire, n’est pas suivi d’un saut de ligne

# Balise boîte ou balise en ligne

**Block** : <body>, , <p>, <table>, <h*N*>, <tr>, <ul>, <ol>, <form>, …

**InLine** : , <img>, <a>, <b>, 

Display : force le type d’affichage de la balise

- inline
- block
- inline-block

# Limitations des balises inline

Il n’est pas possible de contrôler les marges verticales ni les dimensions d’une balise inline.

Les propriétés width, height, padding-top, padding-bottom, margin-top, margin-bottom ne fonctionneront pas comme prévu

Voir : https://alligator.io/css/display-inline-vs-inline-block/

# Positionnement CSS

Par défaut les éléments sont positionnés selon le **flux courant**, c'est à dire suivant l'ordre dans lequel les balises apparaissent dans le code HTML.

Chaque élément est positionné en fonction des éléments frères qui l'entourent.

# Positionnement CSS

- relatif : position normale relative au **flux**
- absolu : retirée du flux, la balise est positionnée par rapport à son parent direct
- fixe : retirée du flux, la balise est positionnée par rapport à la page (viewport)
- sticky : position relative au flux, mais décalée

{ position: relative / absolute / fixed / sticky }

{ top, right, bottom, left: 42px; }

# Positionnement CSS

Boite flottante

Troisième type de positionnement échappant au flux

Flotte à gauche ou à droite de l’élément parent

Le contenu parent entoure le flottant

{ float: right / left; }

{clear: right / left }

# Bonnes pratiques de positionnement

Garder les éléments dans le flux, n’utiliser les positions relatives et absolues qu’en cas d'extrême nécessité

→ laisser le navigateur gérer la présentation

Préférer l’utilisation des marges

Éviter les tables, utiliser le positionnement inline-block

# Flexbox

Flex permet d’agencer des blocs automatiquement

display: **flex**;  (la base)
flex-direction: row / column / row-reverse / column-reverse
flex-wrap: nowrap / wrap / wrap-reverse
justify-content: flex-start / flex-end / **center** / …  (axe X)
align-items: stretch / baseline / center / flex-start / flex-end  (axe Y)
align-content: stretch / center / flex-start / flex-end / …

Voir : https://flexbox.help/

# Bonnes pratiques

Préférer les classes aux id

Utiliser les balises html ayant du sens plutôt que de simples 

Utiliser les balises html pour leur fonction et non pas selon leur aspect

Préférer l’héritage et les combinaisons de règles pour éviter la duplication

Une même balise peut porter plusieurs classes

# Ressources

Les unités (em, px, pt, cm, in, …) :
https://www.w3.org/Style/Examples/007/units.fr.html

Les couleurs (rgb, hex, hsl, …) :
https://en.wikipedia.org/wiki/Web_colors#CSS_colors

Le positionnement :
https://www.alsacreations.com/article/lire/533-initiation-au-positionnement-en-css-partie-1.html

Exemples :
https://www.w3schools.com/css/css_examples.asp