



CSS

FONCTIONNEMENT & BONNES PRATIQUES



NICOLAS LETHUILLIER

Ingénieur développeur
Spécialiste front-end

Intervenant sur :

- Java
- JavaScript
- Angular
- CSS

nlethuillier@gmail.com

PRÉREQUIS

HISTOIRE QU'ON PARLE TOUS DE LA MÊME CHOSE

- ✓ Connaître **HTML**
- ✓ Avoir déjà travaillé sur un **projet web**
- ✓ Disposer d'un **IDE** digne de ce nom
Indice : Notepad++ n'est pas un IDE digne de ce nom
- ✓ Il n'est pas nécessaire d'avoir le moindre **bon goût ou sens esthétique**
(voir ci-après « De quoi on ne parlera pas aujourd'hui »)

SOMMAIRE



Introduction

- Utiliser le CSS à bon escient, ce qu'on évoquera et ce qu'on n'évoquera pas



Intégration

- Mise en place, branchement avec le HTML, syntaxe et bonnes pratiques



Basiques

- Mise en forme de texte, de blocs



Notions intermédiaires

- Agencement des éléments (layout)



Concepts avancés

- Pseudo-machins, polices, flexboxes, responsive design



Mastering CSS

- Pour sauver le monde avec classe et panache



Pour aller plus loin

- Compatibilité des navigateurs, préprocesseurs, conventions



Références

- Quelques URL intéressantes



INTRODUCTION

UTILISER LE CSS À BON ESCIENT,
CE QU'ON ÉVOQUERA
ET CE QU'ON N'ÉVOQUERA PAS

INTRODUCTION

RAPPELS CONCERNANT LE DÉVELOPPEMENT WEB (FRONT)

HTML



STRUCTURE
l'information

CSS



MET EN FORME
le contenu

JAVASCRIPT



DYNAMISE
la page

INTRODUCTION

RAPPELS CONCERNANT LE DÉVELOPPEMENT WEB (FRONT)

HTML



« Ma page est composée d'un en-tête, d'un pied, d'un menu, d'articles, eux-mêmes composés d'un titre, d'un auteur et d'un contenu textuel. »

CSS



« Mon en-tête est bleu, le premier article a un liseré doré, le nom des auteurs est en blanc sur fond rouge, le pied disparaît sur mobile. »

JAVASCRIPT



« Quand le visiteur clique sur un article, il apparaît en plein écran, puis le fichier fanfare.mp3 est joué en continu à plein volume. »

INTRODUCTION

RAPPELS CONCERNANT LE DÉVELOPPEMENT WEB (FRONT)

Ce qu'il ne faut PAS faire

HTML



« Ma page est uniquement composée d'un seul `div`.
J'utilise des `button` au lieu des `input` car j'aime bien leur design. »



CSS



« Le pied de page est affiché au-dessus du contenu en-tête.
Les listes ne doivent avoir aucune bordure et doivent s'afficher en Arial 48 gras. »



JAVASCRIPT



« Le premier article a un liseré double et le pied de page disparaît en mobile.
Les tableaux possèdent une ligne de séparation vide au milieu, pour la lisibilité. »



INTRODUCTION

RAPPELS CONCERNANT LE DÉVELOPPEMENT WEB (FRONT)

css



Le moteur CSS
du navigateur
est souvent utilisé à

15%

de ses capacités*



**On utilise trop souvent
JavaScript pour faire des choses
dont CSS seul est capable**

→ **Performances en berne**

JAVASCRIPT



Le moteur JavaScript
du navigateur
est souvent utilisé à

150%

de ses capacités*

INTRODUCTION

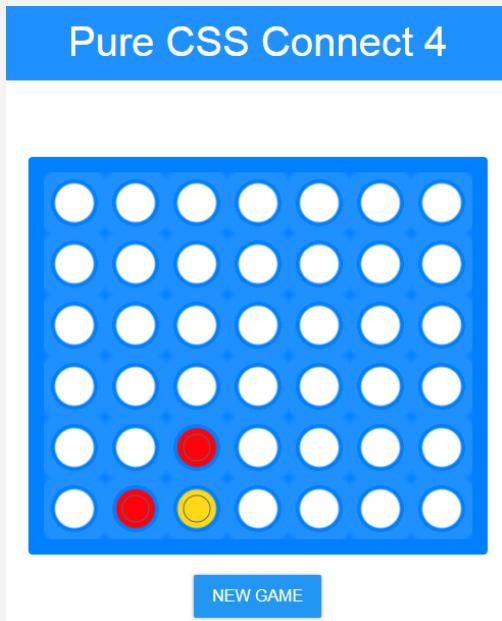
POURQUOI NE SAIT-ON PAS FAIRE DU CSS ?

- « position: absolute; top: 437px; margin-left: 1.2cm; »
→ Ça y est, mon bloc est à la bonne place ☺
- CSS, comme d'autres langages, permet d'arriver à ses fins en faisant n'importe quoi
- Apprendre à faire du CSS correct ET propre garantit :
 - Sa compatibilité sur un maximum de navigateurs et d'écrans
 - La maintenabilité de votre application
 - Sa pérennité (versions de navigateur, tailles d'écrans)

INTRODUCTION

CSS = MCGYVER

- Certains développeurs artistes réalisent des images avec des div et un peu de CSS... voire des jeux entiers, sans une seule ligne de JavaScript !
- Exemple : **Puissance 4**



<https://codepen.io/finnhvman/pen/xXpzVN>

Des radio button sont cachés derrière chaque colonne, la fin de partie est détectée via des règles CSS qui ciblent les radio button cochés adjacents.

164 lignes de HTML
349 lignes de CSS
0 ligne de JavaScript

INTRODUCTION DE QUOI ON NE PARLERA PAS AUJOURD'HUI

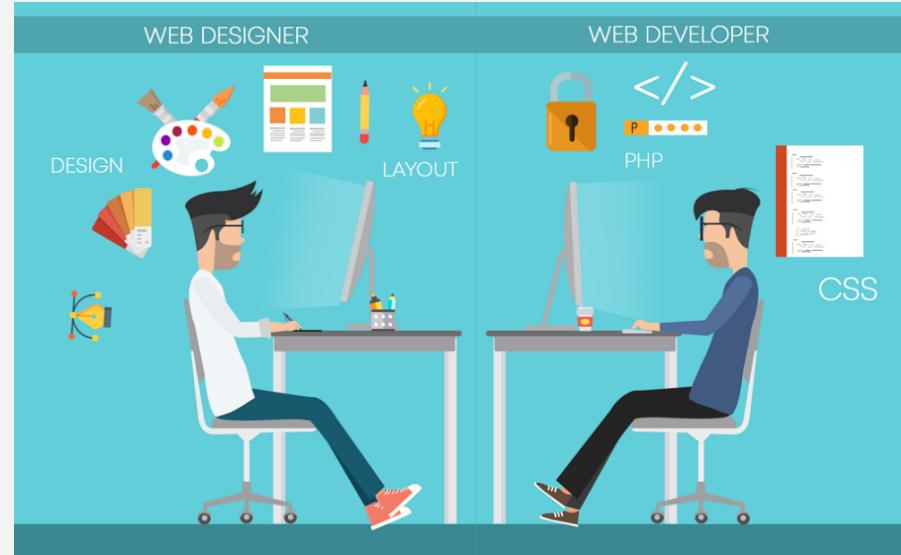
- Le web design
 - conception d'un site internet (ergonomie, utilisabilité, accessibilité).

Expert Photoshop

Se met à la place de l'utilisateur
Produit des maquettes
(PSD, PNG, JPG)

Skills

Connaît la largeur du « m »
en Helvetica 12pt
implémentation Windows 98



Expert dev

Se met à la place du dev qui passera après lui
Produit du code
(HTML, CSS, JavaScript)

Skills

Sait quand utiliser des `div`
et quand utiliser des `span`
(enfin en théorie)

Des questions ?





INTÉGRATION

MISE EN PLACE,
BRANCHEMENT AVEC LE HTML,
SYNTAXE ET BONNES PRATIQUES

INTÉGRATION

COMMENT INTÉGRER DU CSS DANS SA PAGE ?

Là, c'est le moment où vous trouvez que je vous prends pour des novices, mais **back 2 basics**, on a dit !

1

```
<head>
  ...
  <link type="text/css" rel="stylesheet" href="https://monsite.fr/style/style.css" />
  ...
</head>
<body>
  ...
```

OUI !

2

```
<style>
  h2 { color: red; }
  .corps { margin-top: 20px; }
</style>
```

Très rare

3

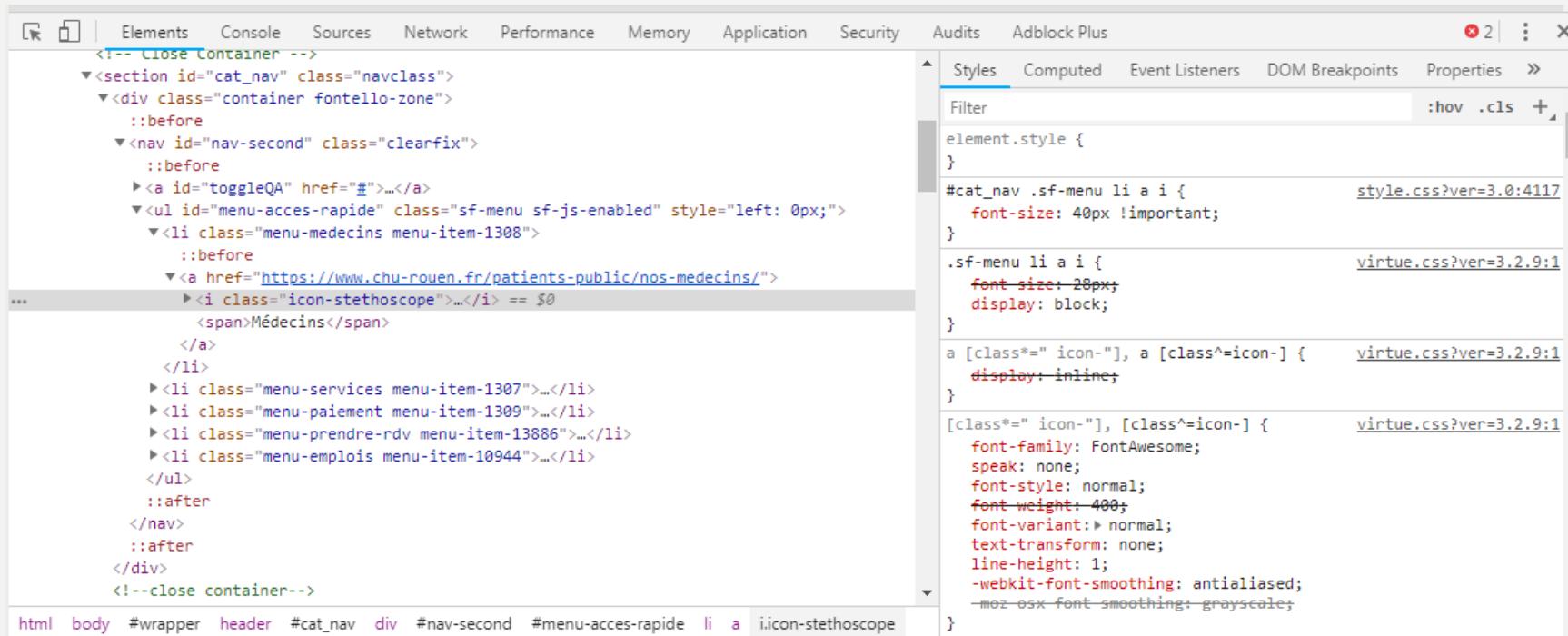
```
<div style="background-color: #000; color: #fff">ABC</div>
```

Uniquement généré par JS
Interdit d'écrire ça !

INTÉGRATION

KNOW YOUR F12

- Si vous travaillez sur un projet web,
votre meilleure amie est la touche **F12** de votre clavier



INTÉGRATION

KNOW YOUR F12

The screenshot shows the 'Styles' tab in the Chrome DevTools. It lists several CSS rules for an element. A red box highlights the first rule: '#cat_nav .sf-menu li a i { font-size: 40px !important; }'. Another red box highlights the third rule: '[class*=" icon-"], [class^="icon-"] { font-family: FontAwesome; speak: none; font-style: normal; font-weight: 400; font-variant: normal; text-transform: none; line-height: 1; -webkit-font-smoothing: antialiased; -moz-osx-font-smoothing: grayscale; }'. A third red box highlights the fifth rule: '* { outline: 0 !important; }'. The right side of the interface shows the source files and line numbers for each rule.

```
element.style {
}
#cat_nav .sf-menu li a i {
    font-size: 40px !important;
}
.sf-menu li a i {
    font-size: 28px;
    display: block;
}
a [class*=" icon-"], a [class^="icon-"] {
    display: inline;
}
[class*=" icon-"], [class^="icon-"] {
    font-family: FontAwesome;
    speak: none;
    font-style: normal;
    font-weight: 400;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.sf-menu, .sf-menu * {
    list-style: none;
}
*:not(strong):not(b) {
    font-weight: normal !important;
}
* {
    outline: 0 !important;
}
*, :after, :before {
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    box-sizing: border-box;
}
```

- L'onglet **Styles** présente le CSS appliqué sur l'élément sélectionné

Une propriété appliquée car l'élément correspond à la règle

Le fichier et la ligne où apparait la règle

Une propriété surchargée par une autre
(cf. juste après : « Les priorités »)

INTÉGRATION

KNOW YOUR F12

The screenshot shows the Chrome DevTools Styles tab. At the top, there are tabs for Styles, Computed, Event Listeners, DOM Breakpoints, and Properties. The Styles tab is active. Below the tabs is a toolbar with a dropdown set to ':hov' (highlighted with a blue border), a 'Filter' input field, and a '+' button. To the right of the toolbar are checkboxes for Force element state: :active, :focus, :focus-within, :hover, and :visited. The main area displays a list of CSS rules. A red box highlights the first rule: '#cat_nav .sf-menu li a i { font-size: 40px !important; }'. Another red box highlights the second rule: '.sf-menu li a i { font-size: 20px; display: block; }'. A third red box highlights the third rule: 'a [class*=" icon-"], a [class^="icon-"] { display: inline; }'. A fourth red box highlights the fourth rule: '[class*=" icon-"], [class^="icon-"] { font-family: FontAwesome; speak: none; font-style: normal; font-weight: 400; font-variant: normal; text-transform: none; line-height: 1; -webkit-font-smoothing: antialiased; -moz-osx-font-smoothing: grayscale; }'. A fifth red box highlights the sixth rule: '.sf-menu, .sf-menu * { list-style-type: none; }'. A sixth red box highlights the eighth rule: '*:not(strong):not(b) { font-weight: normal !important; }'. A seventh red box highlights the ninth rule: '* { outline: 0 !important; }'. To the right of each highlighted rule, its source file and line number are shown: style.css?ver=3.0:4117, virtue.css?ver=3.2.9:1, virtue.css?ver=3.2.9:1, virtue.css?ver=3.2.9:1, virtue.css?ver=3.2.9:1, style.css?ver=3.0:3862, style.css?ver=3.0:27.

- L'onglet **Styles** présente le CSS appliqué sur l'élément sélectionné

Le bouton `:hov` vous permet de simuler différents états (survol, focus, etc.)

Une propriété appliquée car l'élément correspond à la règle

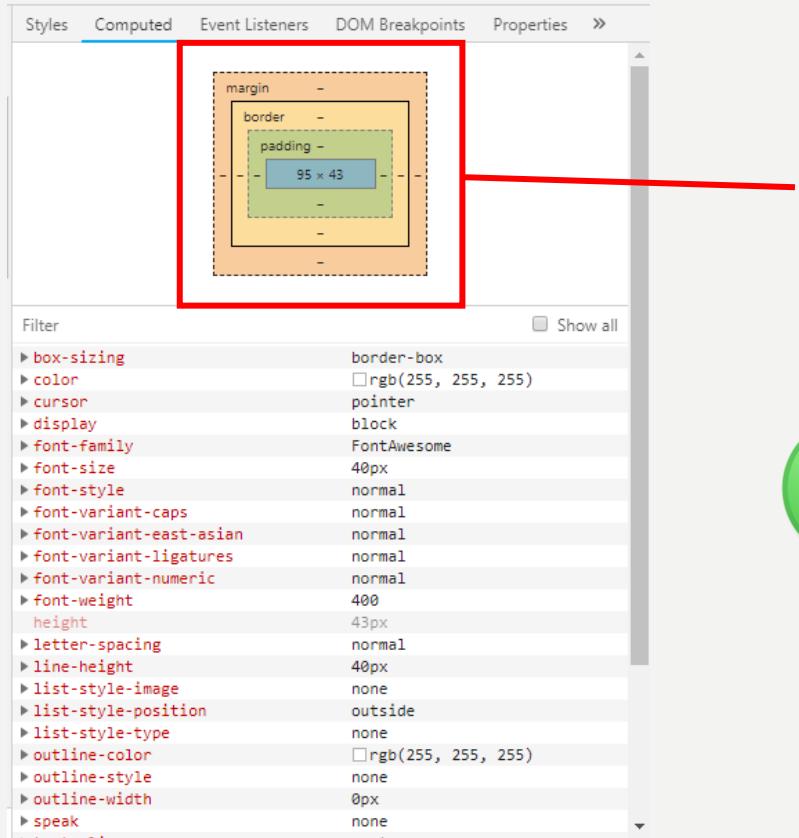
Le fichier et la ligne où apparait la règle

Une propriété surchargée par une autre
(cf. juste après : « Les priorités »)

INTÉGRATION

KNOW YOUR F12

- L'onglet **Computed** présente le CSS appliqué sur l'élément sélectionné... mais classé par propriété



Si vous redimensionnez la fenêtre
alors que l'inspecteur (F12) est ouvert,
Chrome affiche la taille actuelle de la page.
→ Pratique pour tester les media queries !

INTÉGRATION

ID, CLASSES, ATTRIBUTS

- Le HTML contient des **identifiants** (id), des **classes** (class) et des **attributs**.

```
▼<ul id="menu-acces-rapide" class="sf-menu sf-js-enabled" style="left: 0px;">
  ▼<li class="menu-medecins menu-item-1308">
    ::before
    ▼<a href="https://www.chu-rouen.fr/patients-public/nos-medecins/">
      ▶<i class="icon-stethoscope">...</i>
      <span>Médecins</span>
    </a>
  </li>
  ▶<li class="menu-services menu-item-1307">...</li>
  ▶<li class="menu-paiement menu-item-1309">...</li>
  ▶<li class="menu-prendre-rdv menu-item-13886">...</li>
  ▶<li class="menu-emplois menu-item-10944">...</li>
</ul>
```

LES CLASSES

caractérisent l'élément.

Un élément peut en avoir plusieurs et plusieurs éléments peuvent avoir la même classe.

LES IDENTIFIANTS

identifient l'élément.

Un élément ne peut avoir qu'un identifiant, qui doit être unique dans toute la page.

LES ATTRIBUTS

précisent le comportement d'un élément.

Important : « class » et « id » sont des attributs comme les autres.

INTÉGRATION

ID, CLASSES, ATTRIBUTS

- Ces **classes**, **identifiants** et **attributs** vont nous servir pour sélectionner les éléments.

LES CLASSES

caractérisent l'élément.

Un élément peut en avoir plusieurs et plusieurs éléments peuvent avoir la même classe.



```
.item-de-menu {  
    color: red;  
    background: yellow;  
}
```

LES IDENTIFIANTS

identifient l'élément.

Un élément ne peut avoir qu'un identifiant, qui doit être unique dans toute la page.



```
#top-menu {  
    position: fixed;  
    top: 0;  
}
```

LES ATTRIBUTS

précisent le comportement d'un élément.

Important : « class » et « id » sont des attributs comme les autres.



```
[href] {  
    color: red;  
    text-decoration: underline;  
}
```

INTÉGRATION

LES SÉLECTEURS

- Je peux aussi écrire :

```
[href] {
    color: red;
    text-decoration: underline;
}

a[title="Cliquez ici !"] {
    font-size: 20px;
}

p.very-important.advice {
    border: 2px solid red;
    color: green;
}

#top-bar.fixed {
    position: fixed;
    top: 0;
}
```

Tous les éléments (vide = *) qui ont un attribut href

Tous les liens (a) qui ont un attribut title
avec la valeur « Cliquez ici ! »

Tous les paragraphes (p) qui ont la classe « very-important »
ET la classe « advice »

L'élément d'identifiant « top-bar » lorsqu'il a la classe « fixed »
(par exemple ajoutée/retirée en JS)

INTÉGRATION

LES SÉLECTEURS

- Je peux sélectionner un élément en fonction d'une partie de l'un de ses attributs :

```
[title~="flower"] {  
    border: 5px solid yellow;  
}  
  
[class|= "top"] {  
    background: yellow;  
}  
  
[class^= "top"] {  
    background: yellow;  
}  
  
[class$= "test"] {  
    background: yellow;  
}  
  
[class*= "te"] {  
    background: yellow;  
}
```

Tous les éléments qui ont un attribut `title` dont la valeur contient le mot « flower »

Tous les éléments qui ont un attribut `class` dont la valeur commence par le mot « top »

Tous les éléments qui ont un attribut `class` dont la valeur commence par la chaîne « top »

Tous les éléments qui ont un attribut `class` dont la valeur finit par la chaîne « test »

Tous les éléments qui ont un attribut `class` dont la valeur contient la chaîne « te »

INTÉGRATION

LES SÉLECTEURS

- Je peux sélectionner un élément en fonction de ses parents ou de ses prédecesseurs :

```
#top-bar .item-menu a {  
    color: red;  
}  
  
body > div {  
    background-color: yellow;  
}  
  
a + img {  
    border: 2px solid blue;  
}  
  
a ~ img {  
    border: 2px solid blue;  
}  
  
.header div > span + i + a ~ img {  
    width: 12px;  
}
```

Tous les liens (a)
présents dans un élément avec la classe « item-menu »
lui-même dans l'élément d'identifiant « top-bar »

Tous les div
enfants immédiats de body

Toutes les images (img)
qui viennent juste après un lien (a)

Toutes les images (img)
qui viennent après un lien (a)

Toutes les images (img) qui viennent après un lien (a),
lui-même juste après un i, lui-même juste après un span
enfant immédiat d'un div, lui-même contenu dans
un élément avec la classe « header »

INTÉGRATION

LES SÉLECTEURS

- Chaque règle a une valeur de priorité qui détermine si elle surcharge ou si elle est surchargée en cas de conflit.
- Cette valeur est calculée selon le tableau suivant :

Sélecteur	Score
Type de l'élément (a, div, h1)	0,0,0,1
Classe, attribut, pseudo-classe	0,0,1,0
Identifiant	0,1,0,0
!important, inline	1,0,0,0

donc pour la règle :

```
.header div > span + [href] ~ img {
```

le score est de :

0,0,2,3

INTÉGRATION

LES SÉLECTEURS

Sélecteur	Score
Type de l'élément (a, div, h1)	0,0,0,1
Classe, attribut, pseudo-classe	0,0,1,0
Identifiant	0,1,0,0
!important, inline	1,0,0,0

```
.header div > span + [href] ~ img {
```

0,0,2,3

```
#menu .menu-item a {
```

0,1,1,1

```
span + img {
```

0,0,0,2

```
span + img.fullwidth {
```

0,0,1,2

```
#header .menu-item > .menu-item a {
```

0,1,2,1

INTÉGRATION

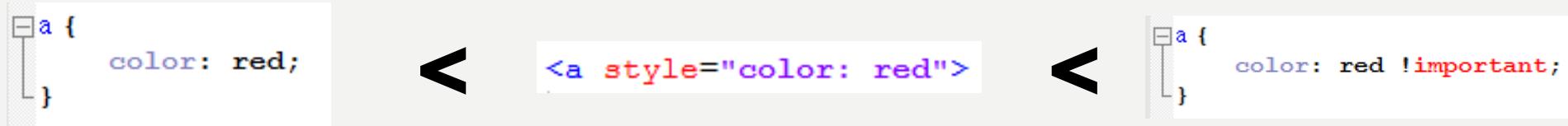
LES SÉLECTEURS

Sélecteur	Score
Type de l'élément (a, div, h1)	0,0,0,1
Classe, attribut, pseudo-classe	0,0,1,0
Identifiant	0,1,0,0
!important, inline	1,0,0,0

- On peut forcer la priorité en ajouter le mot-clé « **!important** »
→ Toutefois, c'est une mauvaise pratique, car elle freine la maintenabilité

```
.item-menu a {  
    color: red !important;  
}
```

- Le style d'un attribut `style` dans le HTML a toujours priorité sur le CSS, sauf en présence de « **!important** » (seul usage accepté)



- Maîtriser les priorités permet de limiter drastiquement l'usage de « **!important** »

INTÉGRATION

QUELQUES BONNES PRATIQUES

- Les identifiants et classes s'écrivent en **kebab-case**

.my-beautiful-div



.my_beautiful_div



.myBeautifulDiv



- N'utiliser des identifiants que lorsque c'est vraiment nécessaire
→ Peut vite entraîner des conflits, surtout en programmation orientée composants !
- Ne pas « sur-prioriser »

.my-beautiful-div



div.my-beautiful-div



- Utiliser des termes clairs, facilement identifiables (ex. « heading-menu » plutôt que « hd-mn »)

Des questions ?





TP : CSS DINER

SELECTORS! SELECTORS EVERYWHERE!

TP : CSS DINER

<http://flukeout.github.io/>

Select the meals for names that contain 'obb'



CSS Editor

style.css

HTML Viewer

table.html

```
1 Type in a CSS selector  
2 {  
3 /* Styles would go here. */  
4 }  
5 /*  
6 Type a number to skip to a level.  
7 Ex → "5" for level 5  
8 */  
9  
10  
11  
12
```

```
1 <div class="table">  
2   <bento for="Robbie">  
3     <apple />  
4   </bento>  
5   <bento for="Timmy">  
6     <pickle />  
7   </bento>  
8   <bento for="Bobby">  
9     <orange />  
10    </bento>  
11  </div>
```



BASIQUES

MISE EN FORME DE TEXTE,
D'ÉLÉMENTS

BASIQUES

INTRODUCTION

- La partie « Basiques » de cette formation est découpée en deux axes, qui abordent chacun un panel étendu de propriétés CSS :

1

Mise en forme de texte

- color
- font-size
- font-family
- font-weight
- font-style
- text-transform
- text-decoration
- letter-spacing
- text-shadow

2

Mise en forme d'éléments

- background
- border
- outline
- border-radius
- box-shadow
- padding/margin
- Eléments HTML

BASIQUES

MISE EN FORME DE TEXTE

- La propriété `color` permet de changer la couleur du texte.

```
.my-block {  
    color: #fa52c4;  
}
```

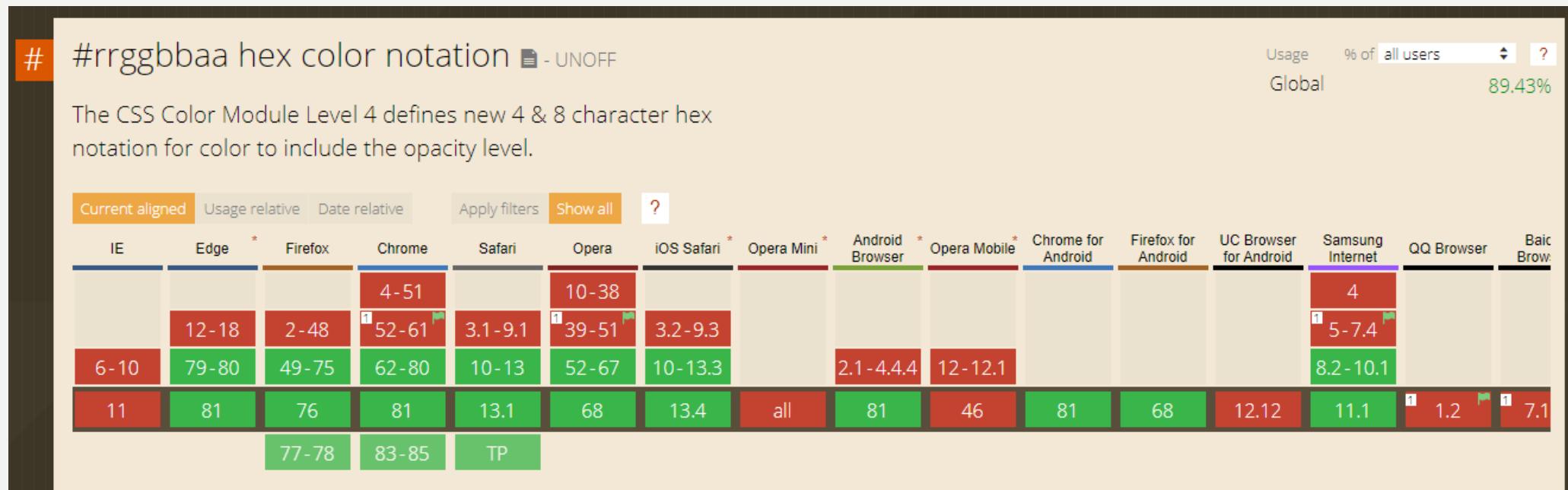
- Facile ! Cependant, quelques règles...

	Codes en minuscules	Ne pas utiliser les alias	R=G=B ?
	<code>color: #FA52C4;</code>	<code>color: red;</code>	<code>color: #cccccc;</code>
	<code>color: #fa52c4;</code>	<code>color: #ff0000;</code>	<code>color: #ccc;</code>

BASIQUES

MISE EN FORME DE TEXTE

- La propriété `color` peut prendre une valeur de type `#rrrggbbaa`.
- Cependant, ce n'est pas encore supporté par tous les navigateurs :



BASIQUES

MISE EN FORME DE TEXTE

- La propriété `font-size` permet de changer la taille du texte.

```
.my-block {  
    font-size: 16px;  
}
```

LES UNITÉS

px

pixels

pt

points

em

% du font-size du parent

- Facile ! Cependant, quelques règles...

	Utiliser px au lieu de pt	% équivaut à em	0 superflu
	<code>font-size: 12pt;</code>	<code>font-size: 120%;</code>	<code>font-size: 0.9em;</code>
	<code>font-size: 16px;</code>	<code>font-size: 1.2em;</code>	<code>font-size: .9em;</code>

BASIQUES

MISE EN FORME DE TEXTE

- La propriété **font-family** permet de changer la police du texte.

```
.my-block {  
    font-family: Economica, Georgia, serif;  
}
```

- **font-family** est interprété par le navigateur (client), donc les polices disponibles sont celles qui sont installées sur votre OS.
- On indique souvent plusieurs polices, en finissant par une police générique, au cas où les autres ne seraient pas chargées.
- Les génériques : **serif**, **sans-serif**, **monospace**, **cursive**, **fantasy**, **system-ui**

BASIQUES

MISE EN FORME DE TEXTE

- Lorsque la police a des espaces dans son nom, on utilise des guillemets (*double quotes*).

```
.my-block {  
    font-family: "Times New Roman", Georgia, serif;  
}
```

- Je sais, on a dit qu'on ne ferait pas de web design...

STOP THE MADNESS BAN COMIC SANS

In 1995 Microsoft released the font Comic Sans originally designed for comic book style talk bubbles containing informational help text. Since that time the typeface has unfortunately been improperly used in countless contexts from restaurant signage to college exams to medical information. These widespread abuses of printed type threaten to erode the very foun-



dations upon which centuries of typographic history are built. The movement to ban Comic Sans is comprised of concerned individuals who desire to preserve the historical integrity of typography in its various forms. While we recognize the font may be appropriate in a few specific instances, our position is that the only effective means of ending the epidemic of abuse is to completely ban Comic Sans.

Comic Sans est la Kardashian des polices.

Elle a de nombreux fans, mais si vous voulez que les professionnels vous prennent au sérieux, n'évoquez même pas son nom.

BASIQUES

MISE EN FORME DE TEXTE

- La propriété **font-style** permet de changer le style de la police (sans blague).
- Cette propriété ne sert que pour mettre le texte en italique.

```
.my-block {  
    font-style: italic;  
}
```

- La propriété **font-weight** permet de changer l'épaisseur de la police (gras / pas gras... en gros).

```
.my-block {  
    font-weight: 700;  
}
```

Les valeurs vont de **100** à **900**.

Le nombre de variantes possibles (**400, 600, 800...**) dépend de la police (voir plus tard, « Concepts avancés »).

Eviter « **bold** », « **light** », « **bolder** », etc., comme pour les couleurs !

BASIQUES

MISE EN FORME DE TEXTE

- Si on récapitule :

```
<div class="my-block">Ceci est un test</div>
```

```
.my-block {  
    color: #333;  
    font-size: 20px;  
    font-family: Arial, serif;  
    font-style: italic;  
    font-weight: 900;  
}
```

Ceci est un test

BASIQUES

MISE EN FORME DE TEXTE

- Quelques petites choses un peu plus avancées :

```
<div class="my-block">Ceci est un test</div>
```

```
text-transform: capitalize;
```

Ceci Est Un Test

```
text-transform: uppercase;
```

CECI EST UN TEST

```
text-transform: lowercase;
```

ceci est un test

→ Cette propriété permet de bien séparer le fond (HTML) et la forme (CSS).
Si les majuscules ne présentent qu'un intérêt graphique, on les met en CSS !

→ Lorsque le texte est issu d'une base de données, le serveur doit le transmettre sans la moindre mise en forme (HTML/majuscules/minuscules incluses).

BASIQUES

MISE EN FORME DE TEXTE

- Quelques petites choses un peu plus avancées :

```
<div class="my-block">Ceci est un test</div>
```

```
text-decoration: underline;
```

Ceci est un test

```
letter-spacing: 2px;
```

C e c i e s t u n t e s t

```
text-shadow: -2px 2px 4px #a40000;
```

PREVIEW

{écart en partant de la gauche} {écart en partant du haut} {distance (blur)} {couleur}
→ Il existe de nombreux générateurs (Google : « text-shadow generator »)

BASIQUES

MISE EN FORME DE TEXTE - MÉMO

```
<div class="my-block">Ceci est un test</div>
```

```
.my-block {  
    color: #333;  
    font-size: 20px;  
    font-family: Arial, serif;  
    font-style: italic;  
    font-weight: 900;  
    text-transform: uppercase;  
    text-decoration: underline;  
    letter-spacing: 2px;  
    text-shadow: -2px 2px 4px #a40000;  
}
```

CECI EST UN TEST



BASIQUES

MISE EN FORME D'ÉLÉMENTS

- Maintenant que nous savons mettre en forme le contenu d'un élément, il nous faut mettre en forme ledit élément.
- Nous allons donc voir :
 - Les arrière-plans (**background**)
 - Les bordures (**border**)
 - Les bordures externes (**outline**)
 - Les bords arrondis (**border-radius**)
 - Les ombres (**box-shadow**)



BASIQUES

MISE EN FORME D'ÉLÉMENTS

- La propriété `background` est une **propriété raccourcie**.
- Cela signifie qu'elle peut servir à définir plusieurs propriétés à la fois.

```
.my-block {  
    background: #ea7d45 url(..../images/myimage.png) no-repeat center center;  
}
```

équivaut à :

```
.my-block {  
    background-color: #ea7d45;  
    background-image: url(..../images/myimage.png);  
    background-repeat: no-repeat;  
    background-position: center center;  
}
```

transparent
none
repeat
top left

- Dans une définition raccourcie, toute valeur non exprimée vaut sa valeur par défaut.

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- La propriété `background-position` permet de positionner l'arrière-plan.
- Exemple :

CSS Demo: background-position

Reset

```
background-position: top;  
background-position: left;  
background-position: center;  
background-position: 25% 75%;  
background-position: bottom 50px right 100px;    
background-position: right 35% bottom 45%;
```



BASIQUES

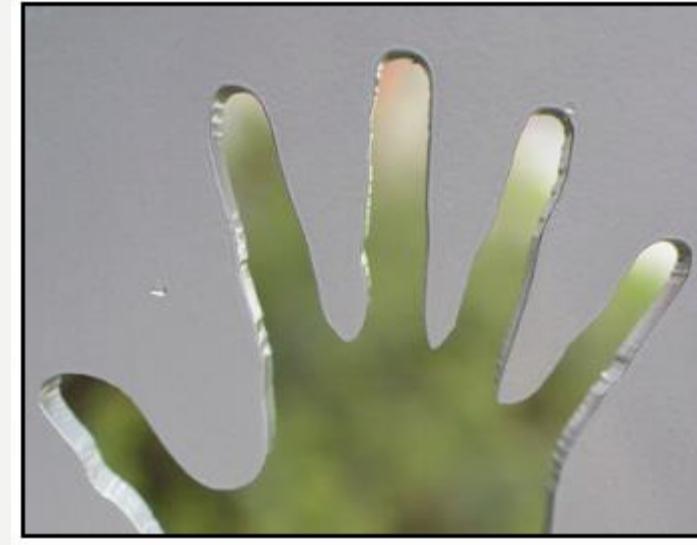
MISE EN FORME D'ÉLÉMENTS

- La propriété `background-size` permet d'indiquer la taille de l'arrière-plan.
- Les valeurs `cover` et `contain` sont particulièrement pratiques !



`contain`

L'image est contenue entièrement dans l'élément, quitte à laisser de l'espace pour ne pas la déformer



`cover`

L'image couvre entièrement l'élément, quitte à être rognée pour occuper tout l'espace sans être déformée

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- **Attention** : utiliser les images (`img`) et les `background-image` à bon escient !

The screenshot shows a website for 'Together Day by Sopra Steria'. The header includes the logo 'Together Day by Sopra Steria', navigation links for 'Together Day France', 'La soirée', 'Tournoi eSport', 'Inscription', and social media icons for Facebook, Twitter, and LinkedIn. The main content area has a red background with white text. It features a graphic of interlocking geometric shapes (cubes) in various colors (red, blue, yellow, green) with the words 'Innover', 'Partager', and 'Construire' on them. Below this is the event title 'TOGETHER DAY - ROUEN', the date 'Mercredi 17 octobre 2018', the time 'à partir de 18h', the location 'à Vue sur Seine', and the address 'Hangar 10, Quai Ferdinand de Lesseps 76000 Rouen'. To the right is a large image of several League of Legends characters.

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- **Attention** : utiliser les images (`img`) et les `background-image` à bon escient !



Le plus souvent, on peut utiliser des `div` avec `background-image` plutôt qu'une `img`

Avantages :

- **Mise en cache** (les `background-image` sont mises en cache, pas les `img`)
- **Responsive design facilité** (possibilité de redimensionner les blocs sans déformer les images)

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- **Attention** : utiliser les images (`img`) et les `background-image` à bon escient !
- Certains sites utilisent même des « sprites »



Une seule image pour tous les boutons, icônes, etc.

→ Performances ++

→ Position avec `background-position`



BASIQUES

MISE EN FORME D'ÉLÉMENTS

- On peut avoir plusieurs `background-image` superposés (séparés par des virgules).



- **BG 1** : aiguille
→ `Image (base 64)`
- **BG 2** : fond rouge semi-transparent
→ `linear-gradient(...)`
- **BG 3** : post-it
→ `image (fichier)`

```
.bloc {  
    background-image:  
        url(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAQAAAEACAQAAAD2e2DtAAAACXBIW...zMQH2T5/Zj  
        linear-gradient(to bottom, rgba(209, 83, 51, 0.65) 0%, rgba(209, 83, 51, 0.65) 100%),  
        url(epikure.jpg);  
}
```

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- La propriété `border` est une **propriété raccourcie**.
- Cela signifie qu'elle peut servir à définir plusieurs propriétés à la fois.

```
.my-block {  
    border: 3px solid #ffff00;  
}
```

équivaut à :

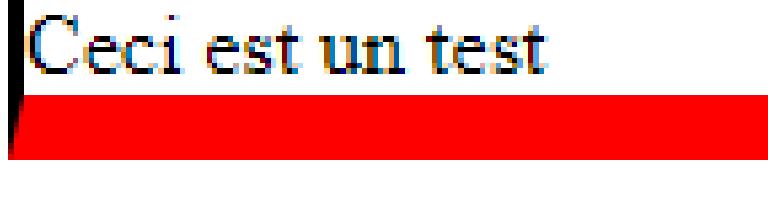
```
.my-block {  
    border-width: 3px;  
    border-style: solid;  
    border-color: #ffff00;  
}
```

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- `border-[width|style|color]` sont elles-mêmes des propriétés raccourcies : elles englobent
 - `border-top-[width|style|color]`
 - `border-left-[width|style|color]`
 - `border-right-[width|style|color]`
 - `border-bottom-[width|style|color]`
- On les distingue lorsqu'on souhaite mettre des bordures différentes sur certains côtés.

```
.my-block {  
    border-color: red;  
    ...  
    border-left: 3px solid #000;  
    ...  
    border-bottom-width: 12px;  
    border-bottom-style: solid;  
}
```



BASIQUES

MISE EN FORME D'ÉLÉMENTS

- La bordure d'un élément impacte l'agencement des autres éléments.
- Exemple :

```
<p>
    Ceci est un paragraphe avec une jolie image
     dedans.
</p>
```

Ceci est un paragraphe avec une jolie image  dedans.

```
p img {
    border: 10px solid #000;
```



Ceci est un paragraphe avec une jolie image  dedans.

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- La bordure extérieure (**outline**) d'un élément, elle, n'impacte pas l'agencement des autres éléments.
- Exemple :

```
<p>
    Ceci est un paragraphe avec une jolie image
     dedans.
</p>
```

Ceci est un paragraphe avec une jolie image  dedans.

```
p img {
    outline: 10px solid #000;
}
```

Ceci est un paragraphe avec une jolie image  dedans.

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- `outline` peut servir, par exemple, pour mettre en évidence un champ de formulaire...

Full Name *

First Name Last Name

- D'ailleurs, les navigateurs s'en servent nativement pour le focus !

Email:

Password:

Remember me

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- Border et outline fonctionnent de la même façon en ce qui concerne les coins.



```
HTML
1 <div></div>

CSS
1 div {
2   width: 100px;
3   height: 100px;
4   border-top: 10px solid black;
5   border-left: 10px solid red;
6 }
```

On peut exploiter ce « biseautage » pour la mise en forme (ex. flèches).
Mais ceci appartient aux concepts avancés ;)

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- La propriété `border-radius` permet de créer des coins arrondis.

```
.my-block {  
    border: 2px solid #000;  
    border-radius: 15px;  
}
```



- On peut spécifier chaque angle séparément avec `border-top-left-radius`, `border-bottom-right-radius`, etc.
- On peut créer des ronds/ovales avec une valeur supérieure à la largeur/la hauteur.

```
border-radius: 100%;
```



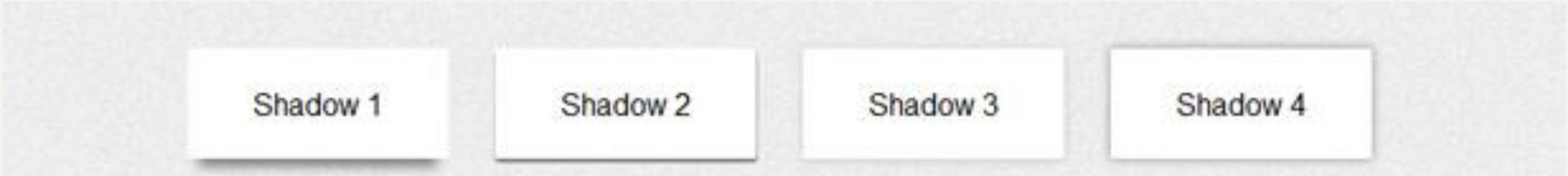
BASIQUES

MISE EN FORME D'ÉLÉMENTS

- La propriété **box-shadow** permet d'ajouter des ombres à un élément.

```
box-shadow: 2px -2px 15px #ccc;
```

- Comme **text-shadow** :
{écart en partant de la gauche} {écart en partant du haut} {distance (blur)} {couleur}
- Plutôt que vous en expliquer le fonctionnement, que vous oublieriez aussitôt, quelques exemples :



Shadow 1

Shadow 2

Shadow 3

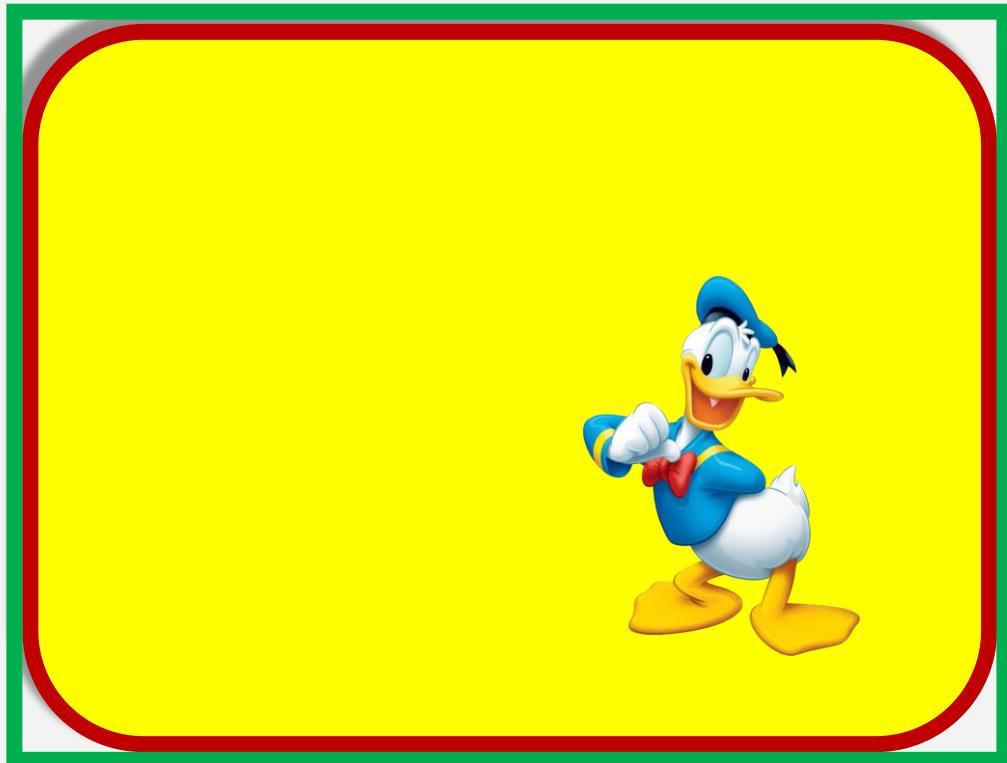
Shadow 4

- De nombreux générateurs existent : Google « box-shadow generator »

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- En résumé :



```
.my-block {  
    background: yellow  
    url(donaldduck.png)  
    no-repeat  
    right 10% bottom 10%;  
    border: 5px solid red;  
    border-radius: 20px;  
    outline: 5px solid green;  
    box-shadow: -5px -5px 10px gray;  
}
```

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- Les propriétés `padding` et `margin` permettent de définir les écarts à l'intérieur et autour d'un élément.

```
 .my-block {  
    padding-left: 25px; /* Ecart intérieur à gauche */  
    margin-top: 12px;   /* Ecart extérieur en haut */  
}
```

BASIQUES

MISE EN FORME D'ÉLÉMENTS

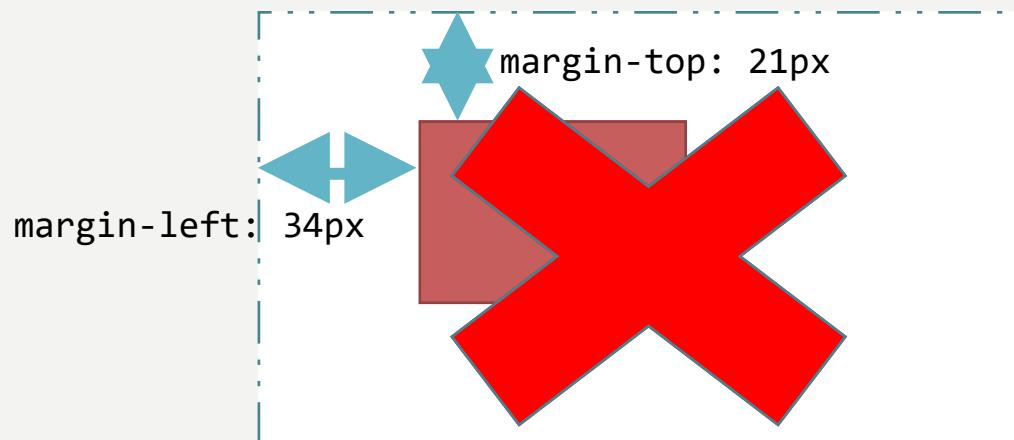
- Les propriétés `padding` et `margin` permettent de définir les écarts à l'intérieur et autour d'un élément.



BASIQUES

MISE EN FORME D'ÉLÉMENTS

- Les propriétés `padding` et `margin` permettent de définir les écarts à l'intérieur et autour d'un élément.
- `padding` sert à ajouter de l'espace à l'intérieur de l'élément
- `margin` sert à espacer les éléments (ajouter de l'espace entre eux)
 - ➔ Il ne sert pas à positionner un élément



BASIQUES

MISE EN FORME D'ÉLÉMENTS

- Les propriétés `padding` et `margin` permettent de définir les écarts à l'intérieur et autour d'un élément.
- On paramètre le `padding` et le `margin` avec une propriété par côté (`margin-top`, `margin-bottom`, `margin-left`, `margin-right`).
- Cependant, il existe des propriétés raccourcies `margin` et `padding`.

```
.my-block {  
    margin: 10px;  
    padding: 12px 15px 0 2px;  
}
```

Applique une marge de 10 pixels de tous les côtés

Applique un écart de :

- 12 pixels en haut
- 15 pixels à droite
- 0 pixel en bas
- 2 pixels à gauche

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- Les propriétés `padding` et `margin` permettent de définir les écarts à l'intérieur et autour d'un élément.

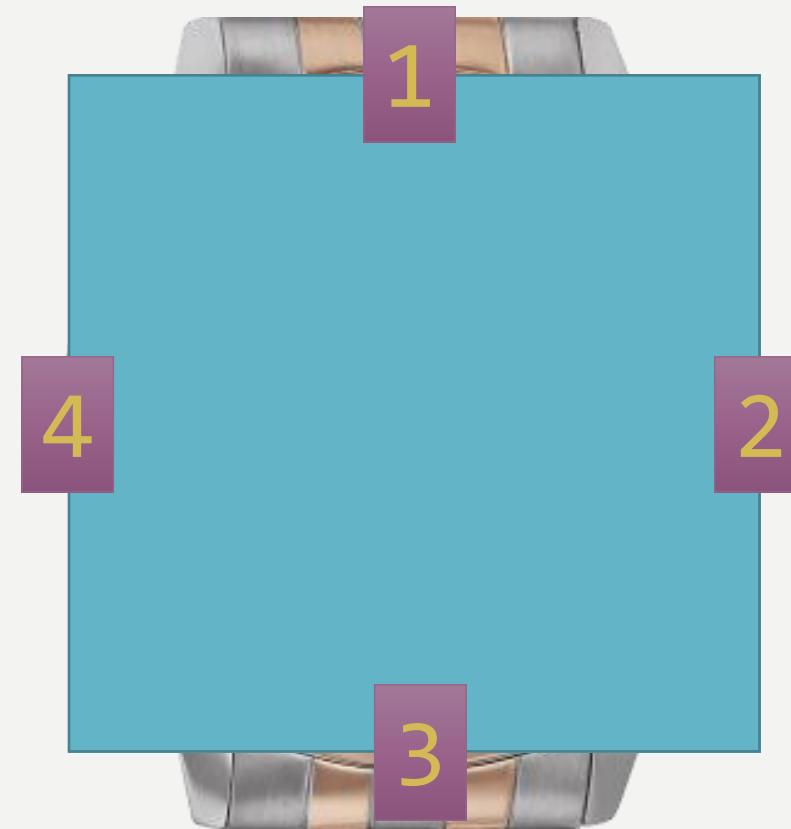
```
.my-block {  
    margin: 10px;  
    padding: 12px 15px 0 2px;  
}
```

1 2 3 4

Parfois, il n'y en a que deux ou trois...

```
.my-block {  
    margin: 10px 2px 5px;  
    padding: 3px 0;  
}
```

Les valeurs manquantes sont alors déduites des valeurs en face
(bottom manquant = top, left manquant = right)



BASIQUES

MISE EN FORME D'ÉLÉMENTS

- Les propriétés `padding` et `margin` permettent de définir les écarts à l'intérieur et autour d'un élément.
- Le `padding` négatif n'existe pas : valeur interdite.
- Le `margin` négatif existe.
Cependant, il est très rarement nécessaire !
En général, le `margin` négatif est une bidouille de dernier recours.
- Exemple :

Ceci est une div contenant du texte.

Ceci est un paragraphe.

```
element.style {  
⚠ padding: -2px;  
}
```

J'ai un écart de ouf entre la div et le p !!

Je vais donc mettre un ~~margin-bottom~~ négatif à la div pour remédier à ce qui vient de près.

➔ Je cherche pourquoi j'ai un écart (`margin-top` par défaut sur `p`) et je le surcharge (`margin-top: 0`)

BASIQUES

MISE EN FORME D'ÉLÉMENTS

- Certains éléments ont des valeurs de padding et de margin par défaut (relativement uniformes selon les navigateurs) :

<p>

```
margin-top: 1em;  
margin-bottom: 1em;
```

Ceci est un paragraphe.

,

```
margin-top: 1em;  
margin-bottom: 1em;  
padding-left: 40px;
```

- 1er item
- 2ème item
- 3ème item

<button>

```
padding: 1px 6px;  
border: 2px outset buttonface;  
text-align: center;  
color: buttontext;
```



Bouton de test

- Il existe des outils appelés « **CSS reset** » qui mettent toutes ces valeurs par défaut. Des dizaines d'implémentations existent : Google « CSS reset ».

BASIQUES

MISE EN FORME D'ÉLÉMENTS - MÉMO

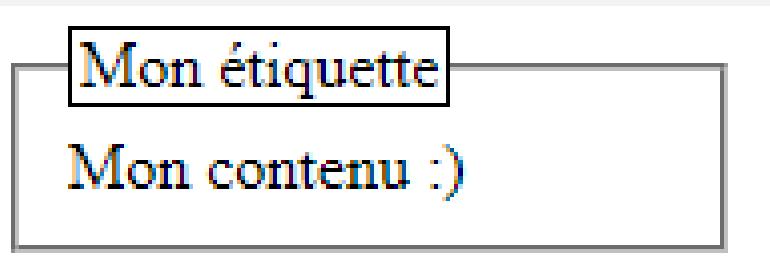
- A retenir :
 - **background** : préférer les **background-image** aux **img**
 - Les valeurs **cover** et **contain** de **background-size**
 - **border** : comptabilisé dans la largeur (**width**) et impacte les voisins
 - **outline** : non-comptabilisé et sans impact sur les voisins
 - Il existe des générateurs en ligne pour **box-shadow**
 - **margin** : extérieur / **padding** : intérieur
 - Les valeurs raccourcies tournent selon le sens des aiguilles d'une montre
 - **margin négatif à proscrire**
 - Bonne pratique : **0** sans unité



BASIQUES

MISE EN FORME D'ÉLÉMENTS – KNOW YOUR HTML!

- Problématique : je souhaite créer un bloc avec un titre sous forme d'étiquette de la façon suivante.



div

```
border: 1px solid #000;  
margin-bottom: -50%;
```

div

```
padding: 10px;
```

- Option 1 :

- Faire une **div** contenant « Mon étiquette » avec une bordure et un **margin-bottom** négatif
 - Faire une **div** contenant « Mon contenu :) » avec une bordure

- Option 2 :

- Utiliser un élément **<fieldset>**

Please know your HTML!

De nombreux éléments moins connus que **div** permettent de structurer l'information... et simplifier la vie du développeur CSS.

BASIQUES

MISE EN FORME D'ÉLÉMENTS – KNOW YOUR HTML!

- Exemples d'éléments utiles et moins connus :

HTML	Définition	Aperçu natif						
<pre><fieldset> <legend>Titre</legend> Mon contenu </fieldset></pre>	Un bloc contenant des informations ou des champs d'un même registre	<div style="border: 1px solid black; padding: 5px; width: fit-content;">Mon étiquette Mon contenu :)</div>						
<pre><table> <caption>Légende du tableau</caption> <thead>Lignes d'en-tête</thead> <tbody>Lignes du corps</tbody> <tfoot>Lignes de pied</tfoot> </table></pre>	Un tableau avec : <ul style="list-style-type: none">Une légendeUn en-têteUn corpsUn pied	<div style="border: 1px solid black; padding: 5px; width: fit-content;"><p>Ma légende</p><table border="1" style="border-collapse: collapse; text-align: center;"><tr><td>Header1</td><td>Header2</td></tr><tr><td>Cell1</td><td>Cell2</td></tr><tr><td>Foot1</td><td>Foot2</td></tr></table></div>	Header1	Header2	Cell1	Cell2	Foot1	Foot2
Header1	Header2							
Cell1	Cell2							
Foot1	Foot2							
<code><sup>, <sub></code>	Exposant, indice	Ceci est un formule chimique ¹ avec du CO ₂ .						
<code><i> / </code>	i distingue une portion de texte em met l'accent sur une partie du texte	Ceci est un texte <i>franchement</i> bien.						
<code> / </code>	b attire l'attention strong désigne un contenu important	Ceci est un texte franchement bien.						
<code><header>, <footer>, <article>...</code>	Structure type d'un site web	-						

BASIQUES

MISE EN FORME D'ÉLÉMENTS – KNOW YOUR HTML!

- Avant toute chose, si possible, vérifiez que vos données sont structurées correctement.
- Le HTML vient du XML, un langage conçu uniquement pour structurer l'information.
→ Ce n'est pas qu'un moyen de pouvoir faire du CSS.
- Il existe des dizaines de balises HTML méconnues.
- Si on n'a pas la main sur le HTML... eh bien on fait avec, et on use des **sélecteurs** (voir début).
→ **Cela ne veut pas dire** qu'il faut adapter le HTML en fonction du CSS,
mais qu'on peut devoir **corriger** la structure de l'information,
car ses défauts d'apparence anodins empêchent de réaliser des développements efficaces.

HTML = 102 balises !

Des questions ?



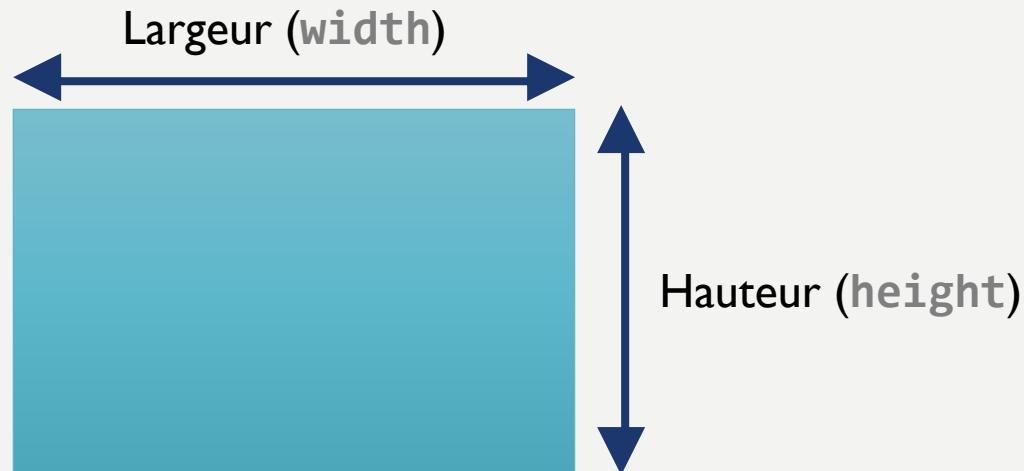


NOTIONS INTERMÉDIAIRES

AGENCEMENT DES ÉLÉMENTS
(LAYOUT)

NOTIONS INTERMÉDIAIRES

- Maintenant que nous avons vu comment mettre en forme le contenu d'un élément et l'élément lui-même, nous allons voir comment agencer les éléments ensemble.
- Un élément est défini par :
 - Ses dimensions (largeur/hauteur)
 - Sa propriété `display`
 - Son positionnement



`display` : comment l'élément se comporte par rapport aux autres éléments
`position` : comment l'élément se comporte dans le **flux**

NOTIONS INTERMÉDIAIRES

DIMENSIONS

- Les propriétés `width` et `height` permettent de spécifier une largeur et une hauteur pour l'élément.
- Une image (`img`) dont on ne spécifie que la largeur ou que la hauteur déduit son autre dimension de façon à ne pas être déformée.
- Un élément vide (ex. `<p></p>`) aura une hauteur de `0` par défaut, même s'il a un `padding`. Son `margin` est par ailleurs ignoré.
- Largeur et hauteur peuvent être spécifiées avec n'importe quelle unité.

```
.my-block {  
    width: 45%;  
    height: 1200px;  
}
```

NOTIONS INTERMÉDIAIRES

DIMENSIONS

- Les unités possibles sont notamment :

Unité	Signification	Exemple
px	Pixel	<code>width: 125px;</code>
%	Pourcentage de la largeur/la hauteur du parent	<code>width: 45%;</code>
em	Pourcentage de la taille de la police dans l'élément	<code>height: 1.2em;</code>
vh	Viewport Height : pourcentage de la hauteur de l'écran	<code>height: 80vh;</code>
vw	Viewport Width : pourcentage de la largeur de l'écran	<code>width: 100vw;</code>

NOTIONS INTERMÉDIAIRES

DIMENSIONS

- Les propriétés `min-width` et `min-height` permettent de spécifier des largeur et hauteur minimum.
- Par exemple, on peut dire à un élément que sa largeur est égale à la moitié de celle de son parent, tout en souhaitant qu'elle ne fasse pas moins de 300 pixels :

```
.my-block {  
    width: 50%;  
    min-width: 300px;  
}
```

- De même, on peut dire à un élément que sa hauteur est égale à 200 pixels, mais qu'elle ne doit pas descendre en-dessous de la moitié de la hauteur de l'écran :

```
.my-block {  
    height: 200px;  
    min-height: 50vh;  
}
```

NOTIONS INTERMÉDIAIRES

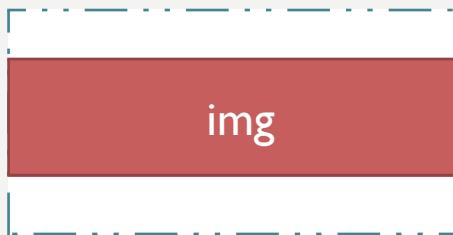
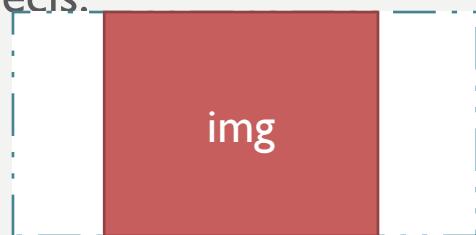
DIMENSIONS

- Les propriétés `max-width` et `max-height` permettent de spécifier des largeur et hauteur maximum.

```
.my-block {  
    width: 50%;  
    max-width: 300px;  
}
```

```
.my-block {  
    height: 200px;  
    max-height: 50vh;  
}
```

- Si on indique un `max-width` et un `max-height` à une image, on assure qu'elle rentre dans un cadre précis.



- → Mais bien sûr, on préférera si possible un `background-size: contain ;`

NOTIONS INTERMÉDIAIRES

DIMENSIONS

- CSS contient quelques **fonctions**, qui ne seront pas abordées dans cette formation.
- Nous en évoquerons cependant une : **calc()**.

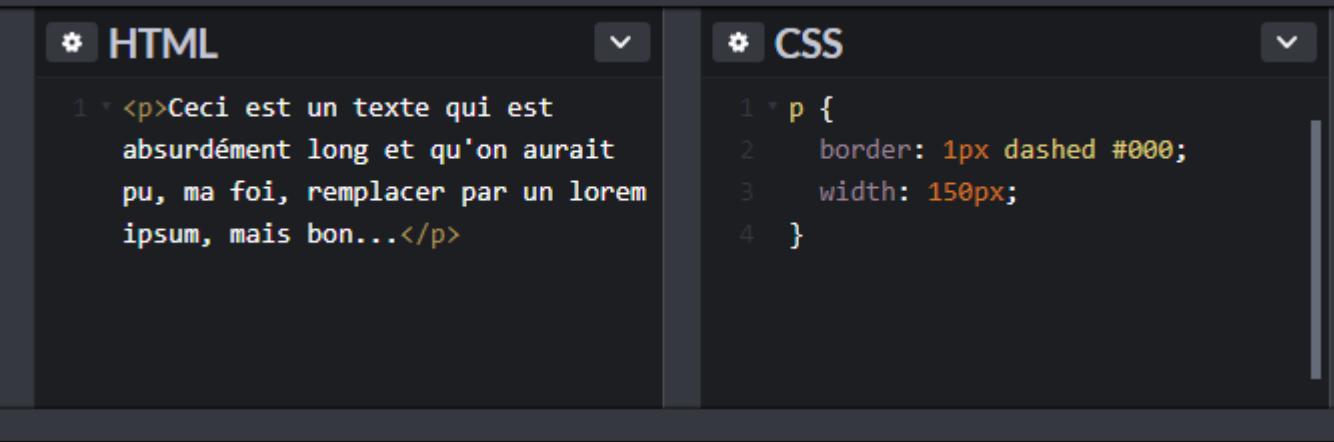
```
.my-block {  
    width: calc(50% - 12px);  
}
```

- La fonction **calc()** permet d'effectuer un calcul entre plusieurs opérandes qui n'ont pas forcément la même unité.
- Il est rare qu'elle soit nécessaire : c'est le plus souvent de la bidouille qui suggère de revoir la conception.

NOTIONS INTERMÉDIAIRES

DÉPASSEMENT/CONTENU

- `width` et `height` nous permettent de spécifier des dimensions... mais qu'en est-il du contenu ?



The screenshot shows a code editor interface with two tabs: "HTML" and "CSS".

HTML Tab:

```
<p>Ceci est un texte qui est  
absurdément long et qu'on aurait  
pu, ma foi, remplacer par un lorem  
ipsum, mais bon...</p>
```

CSS Tab:

```
p {  
    border: 1px dashed #000;  
    width: 150px;  
}
```

Below the code editor, a preview window displays the resulting HTML output. A dashed red box highlights the text "Ceci est un texte qui est absurdément long et qu'on aurait pu, ma foi, remplacer par un lorem ipsum, mais bon...", which has wrapped onto two lines due to the limited width specified in the CSS.

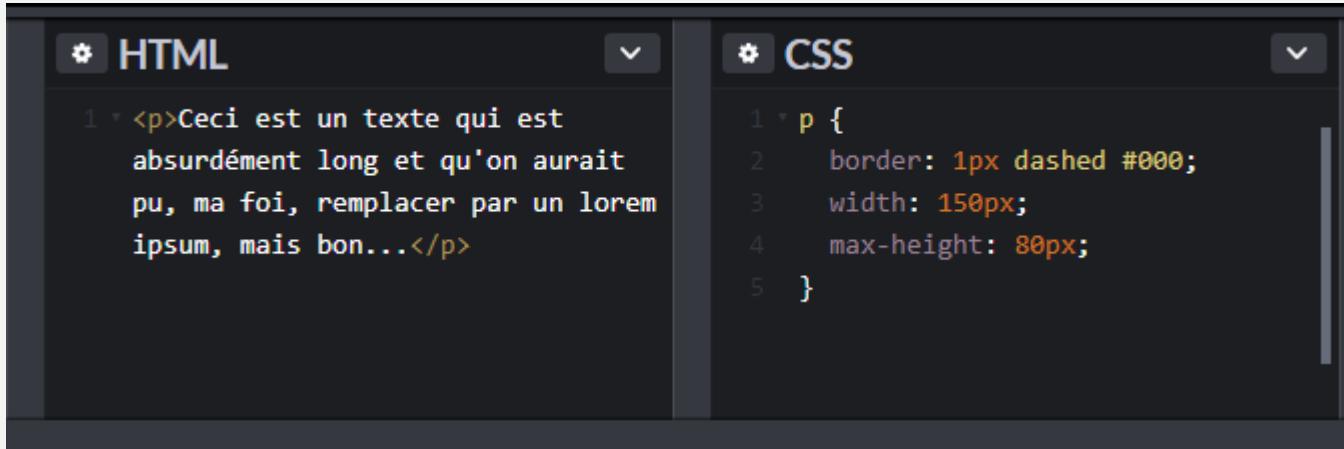
Le texte passe à la ligne, ce qui est logique et commode.

Mais si j'indique une hauteur maximum au paragraphe ?

NOTIONS INTERMÉDIAIRES

DÉPASSEMENT/CONTENU

- `width` et `height` nous permettent de spécifier des dimensions... mais qu'en est-il du contenu ?



The screenshot shows a code editor with two tabs: "HTML" and "CSS". The "HTML" tab contains the following code:

```
<p>Ceci est un texte qui est  
absurdément long et qu'on aurait  
pu, ma foi, remplacer par un lorem  
ipsum, mais bon...</p>
```

The "CSS" tab contains the following code:

```
p {  
    border: 1px dashed #000;  
    width: 150px;  
    max-height: 80px;  
}
```

Below the code editor, there is a preview area. Inside the preview area, there is a dashed border box containing the first few lines of the HTML text. To the right of the preview area, there is some explanatory text.

Ah, là, ça déborde...

... et Dieu inventa la propriété « overflow »

NOTIONS INTERMÉDIAIRES

DÉPASSEMENT/CONTENU

- `width` et `height` nous permettent de spécifier des dimensions... mais qu'en est-il du contenu ?

```
 CSS
1 * p {
2   border: 1px dashed #000;
3   width: 300px;
4   max-height: 80px;
5   overflow: scroll;
6 }
```

Ceci est un texte qui est absurdément long et qu'on aurait pu, ma foi, remplacer par un lorem ipsum, mais bon... Oh et puis je continue, car dans cet exemple, j'avais

```
 CSS
1 * p {
2   border: 1px dashed #000;
3   width: 300px;
4   max-height: 80px;
5   overflow: auto;
6 }
```

Ceci est un texte qui est absurdément long et qu'on aurait pu, ma foi, remplacer par un lorem ipsum, mais bon... Oh et puis je continue, car dans cet exemple, j'avais

```
 CSS
1 * p {
2   border: 1px dashed #000;
3   width: 300px;
4   max-height: 80px;
5   overflow: hidden;
6 }
```

Ceci est un texte qui est absurdément long et qu'on aurait pu, ma foi, remplacer par un lorem ipsum, mais bon... Oh et puis je continue, car dans cet exemple, j'avais

- `overflow` est une propriété raccourcie pour `overflow-x` et `overflow-y`.

NOTIONS INTERMÉDIAIRES

DÉPASSEMENT/CONTENU

- De manière générale, `overflow` indique le comportement que doit avoir l'élément lorsque son contenu « déborde » (par défaut :`visible`).
C'est elle qui détermine la présence ou non de barres de défilement.
- On associe parfois `overflow` avec `white-space`, qui indique si le texte doit aller à la ligne une fois atteint le bord, ou continuer.

```
 CSS
1 *p {
2   border: 1px dashed #000;
3   width: 300px;
4 }
```

Ceci est un texte qui est absurdément long et qu'on aurait pu, ma foi, remplacer par un lorem ipsum, mais bon... Oh et puis je continue, car dans cet exemple, j'avais besoin d'un paragraphe un poil plus grand.

```
 CSS
1 *p {
2   border: 1px dashed #000;
3   width: 300px;
4   white-space: nowrap;
5 }
```

Ceci est un texte qui est absurdément long et qu'on aurait pu, ma foi, remplacer pa

```
 CSS
1 *p {
2   border: 1px dashed #000;
3   width: 300px;
4   white-space: nowrap;
5   overflow: hidden;
6 }
```

Ceci est un texte qui est absurdément long et q

NOTIONS INTERMÉDIAIRES

DISPLAY

- Pourquoi les paragraphes (`p`) ne se comportent pas comme les images (`img`) ?
Pourquoi les tableaux (`table`) ne se comportent pas comme les cellules (`td`) ?
- Parce qu'ils ont des valeurs de propriété `display` différentes !
- Il existe de nombreuses valeurs de `display` possibles, mais les plus fréquentes sont :

inline	inline-block	block
Du texte	Un élément inséré dans le texte	Un élément qui contient tout ça

NOTIONS INTERMÉDIAIRES

DISPLAY

- Pourquoi les paragraphes (`p`) ne se comportent pas comme les images (`img`) ?
Pourquoi les tableaux (`table`) ne se comportent pas comme les cellules (`td`) ?

```
/* CSS */  
p {  
    border: 1px dashed #ccc;  
    width: 300px;  
}  
p a {  
    border: 1px solid red;  
}
```

Ceci est un texte qui est absurdément long et qu'on aurait pu, ma foi, remplacer par un lorem ipsum, mais bon... Oh et puis je continue, car dans cet exemple, j'avais besoin d'un paragraphe un poil plus grand.

inline

L'élément peut se séparer pour passer à la ligne, il suit le texte (ex. `a`)

```
/* CSS */  
p {  
    border: 1px dashed #ccc;  
    width: 300px;  
}  
p a {  
    border: 1px solid red;  
}
```

Ceci est un texte qui est absurdément long et qu'on aurait pu, ma foi, remplacer par un lorem ipsum, mais bon... G Oh et puis je continue, car dans cet exemple, j'avais besoin d'un paragraphe un poil plus grand.

inline-block

L'élément ne peut pas se séparer mais il suit le texte (ex. `img`)

```
/* CSS */  
p {  
    border: 1px dashed #ccc;  
    width: 300px;  
}
```

Ceci est un texte qui est absurdément long et qu'on aurait pu, ma foi, remplacer par un lorem ipsum, mais bon...
Oh et puis je continue, car dans cet exemple, j'avais besoin d'un paragraphe un poil plus grand.

block

L'élément va directement à la ligne (ex. `p`)

NOTIONS INTERMÉDIAIRES

DISPLAY

- Concrètement :

- Un élément **inline** représente un flot de contenu dans sa forme la plus basique.
On ne force jamais un élément à avoir un **display: inline**... parce qu'on n'en voit pas trop l'intérêt.
Exemples : a, strong, em...
- Un élément **inline-block** est un bloc insécable qui vient s'insérer dans le contenu.
Un **inline-block** a pour largeur celle de son contenu (0 si vide).
Plusieurs **inline-block** peuvent se suivre horizontalement.

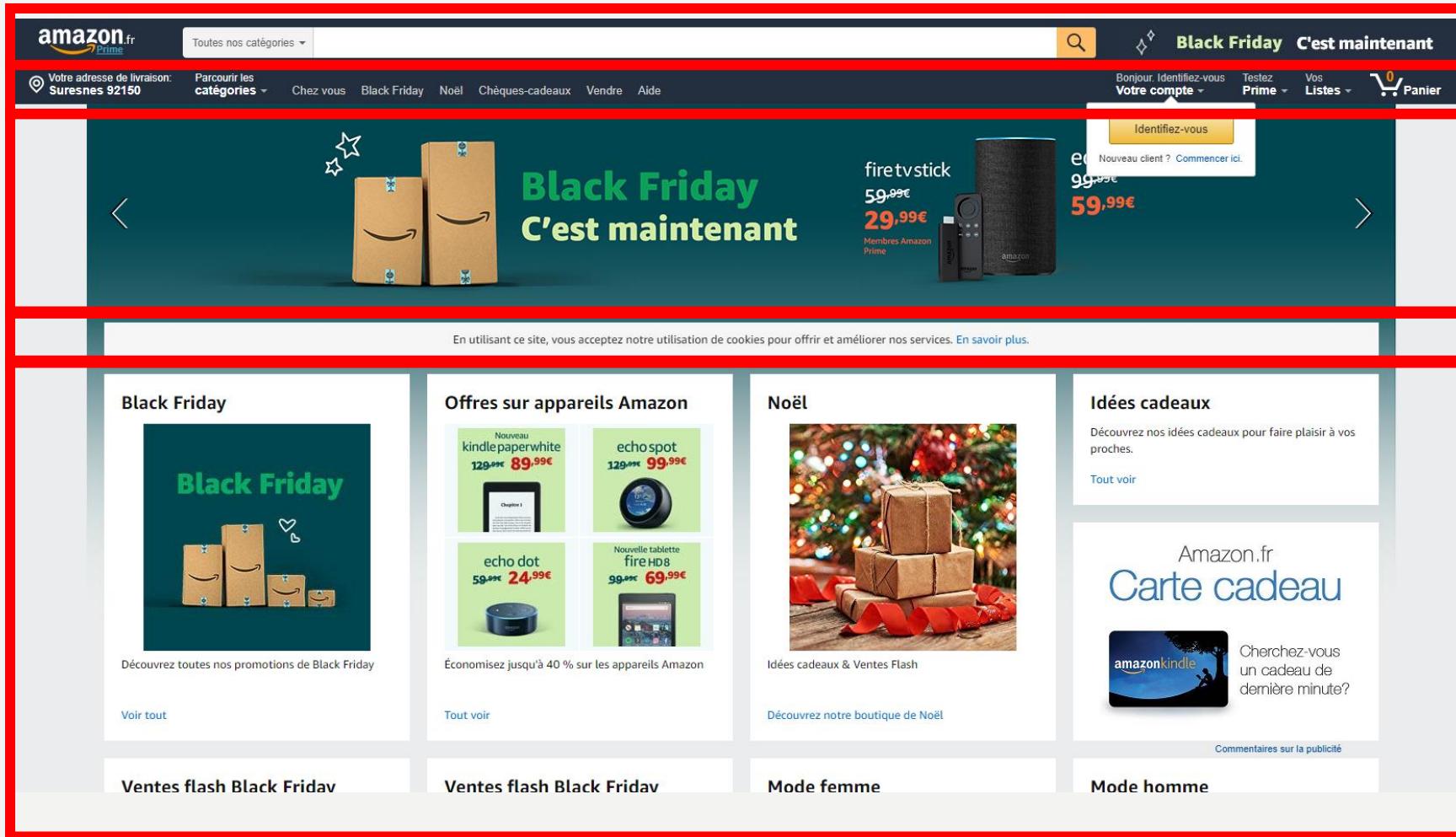
Ceci est un texte avec une image  , puis une autre  , et ainsi de suite, pour constater qu'elles se suivent bien .

Exemples : img...

- Un élément **block** est un bloc qui structure le contenu.
Un **block** occupe par défaut toute la largeur de son parent.
Qu'il ait une largeur de 100% ou pas, ce qui vient avant et après est à la ligne.
Exemples : div, table...

NOTIONS INTERMÉDIAIRES DISPLAY

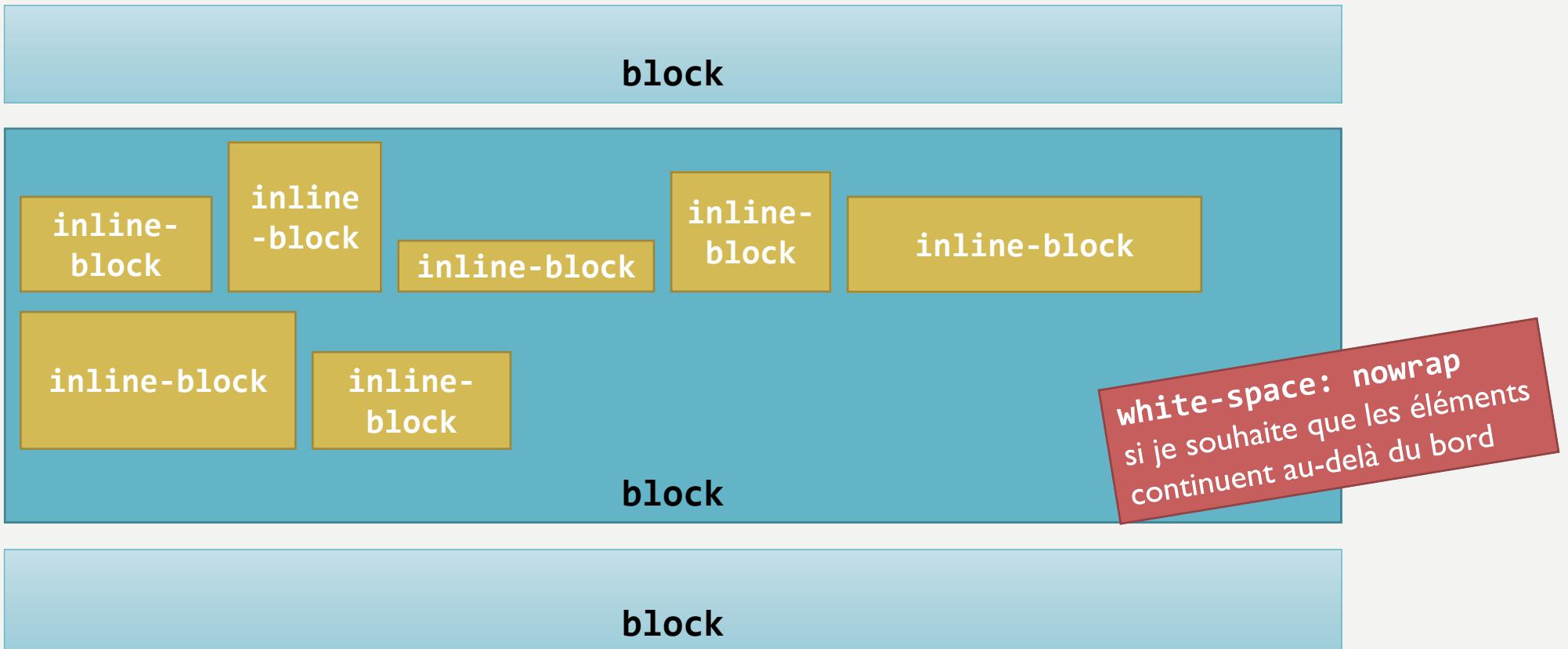
- Il faut imaginer une page web comme un enchaînement de blocks



NOTIONS INTERMÉDIAIRES

DISPLAY

- Un dernier petit schéma pour bien comprendre la différence entre **block** et **inline-block** :



NOTIONS INTERMÉDIAIRES

DISPLAY

- Les deux éléments les plus courants, qu'on utilise à tort pour leur valeur de `display` par défaut, sont :
 - `block` : `div`
 - `inline` : `span`

w3schools.com

The `<div>` tag defines a division or a section in an HTML document.

The `` tag is used to group inline-elements in a document.

- Afin d'avoir un HTML correct, on ne met jamais une `div` dans un `span`.
On ne met pas non plus de `span` dans des `span`.
→ Erreur de structure de l'information !

NOTIONS INTERMÉDIAIRES

DISPLAY

- Exemples d'usage de changement de valeur `display`

ENFIN, ICI, CE QUI VA NOUS INTÉRESSER, C'EST LE TRAVAIL SPÉCIAL DE MICHAEL GIACCHINO. Fidèle de Pixar (*Les Indestructibles*, *Ratatouille*, *Vice Versa*) et de J.J. Abrams (*Mission Impossible 3*, *Star Trek 1 & 2*), oscarisé pour la superbe musique de *Là-Haut*, le compositeur américain devait pour les besoins de *Jurassic World* broder avec l'exigence des fans, pour qui la franchise Jurassic, c'est avant tout la musique cuivrée du grand, de l'unique John Williams.



Et contre toute attente, Giacchino a réalisé un superbe travail en reprenant tous les thèmes célèbres, distillés aux instants clés de l'histoire pour faire tout le lien avec la saga originale, tout en ajoutant ses propres refrains, identité d'une nouvelle franchise. En imitant le style de John Williams (des vents, et en particulier des trompettes, entonnant une phrase simple à un volume qui monopolise l'attention), Giacchino est parvenu à mêler l'esprit Park au nouvel

```
HTML
1 <p>Mon texte</p>
2 
3 <p>La suite de mon texte</p>
```

```
CSS
1 img {
2   display: block;
3   max-width: 100%;
4 }
```

→ Eviter les div superflues !

NOTIONS INTERMÉDIAIRES DISPLAY

- Exemples d'usage de changement de valeur display

The screenshot displays a section of a financial news website. At the top, there is a table of historical stock quotes:

Date	Open	High	Low	Close	Change	Volume
21/11/2018	84,100	+1,39%	34 049	83,400	84,400	82,650
20/11/2018	82,950	-2,98%	53 374	85,000	85,200	81,950
19/11/2018	85,500	+0,94%	47 499	84,750	87,350	84,650
16/11/2018	84,700	-0,70%	55 785	85,050	86,000	83,050

Below this is a two-column box:

- HISTORIQUE DES COTATIONS**
 - 1S 1M 3M 6M 1AN 2ANS 5ANS
 - Cours: 84,700
 - Variation: +0,71%
 - Plus haut: 87,350
 - Date plus haut: 19/11/2018
 - Plus bas: 81,950
 - Date plus bas: 20/11/2018
 - Volumes moyens: 16 611
- AUTRES CHIFFRES**
 - Clôture précédente: 85,150
 - Variation depuis le 01/01/2018: -45,36%
 - Cours au 01/01/2018: 155,849
 - Plus haut 1an: 187,900
 - Plus bas 1an: 81,950
 - Capitaux échangés ce jour: -

At the bottom, there is a section titled **FICHE SOCIÉTÉ SOPRA STERIA GROUP** with the following text:

Sopra Steria Group figure parmi les principales sociétés françaises de services informatiques. L'activité s'organise essentiellement autour de 3 pôles , prestations de conseil : prestations de conseil stratégique, de conseil dans la mise en oeuvre de projets de restructuration et d'évolution vers ...

The screenshot shows the CSS panel of a browser's developer tools. It contains the following code:

```
* div > div {  
    display: inline-block;  
    max-width: 50%;  
}
```

NOTIONS INTERMÉDIAIRES

DISPLAY

- Autres valeurs de display
 - `table`, `table-row`, `table-cell`, `flex`, `grid`, `inline-table`...
 - Au total 23 valeurs possibles, spécifiques à tous les comportements natifs
 - On ne joue en général qu'avec `inline-block`, `block`... et `none`
- Un élément en `display: none` disparaît totalement de la page.
Les éléments voisins font comme s'il n'existeit plus.
- C'est ce qu'on utilise pour l'affichage conditionnel (`.hide()` chez JQuery...)

The screenshot shows a navigation bar with links for Home, HTML, CSS, JavaScript, SQL, PHP, More, and Reference. The 'CSS' link is selected. A sidebar on the left lists various CSS properties, with 'display' highlighted in green. The main content area is titled 'CSS display Property' and contains a red-bordered section labeled 'Example'. Below it, a code snippet shows four examples of different display values:

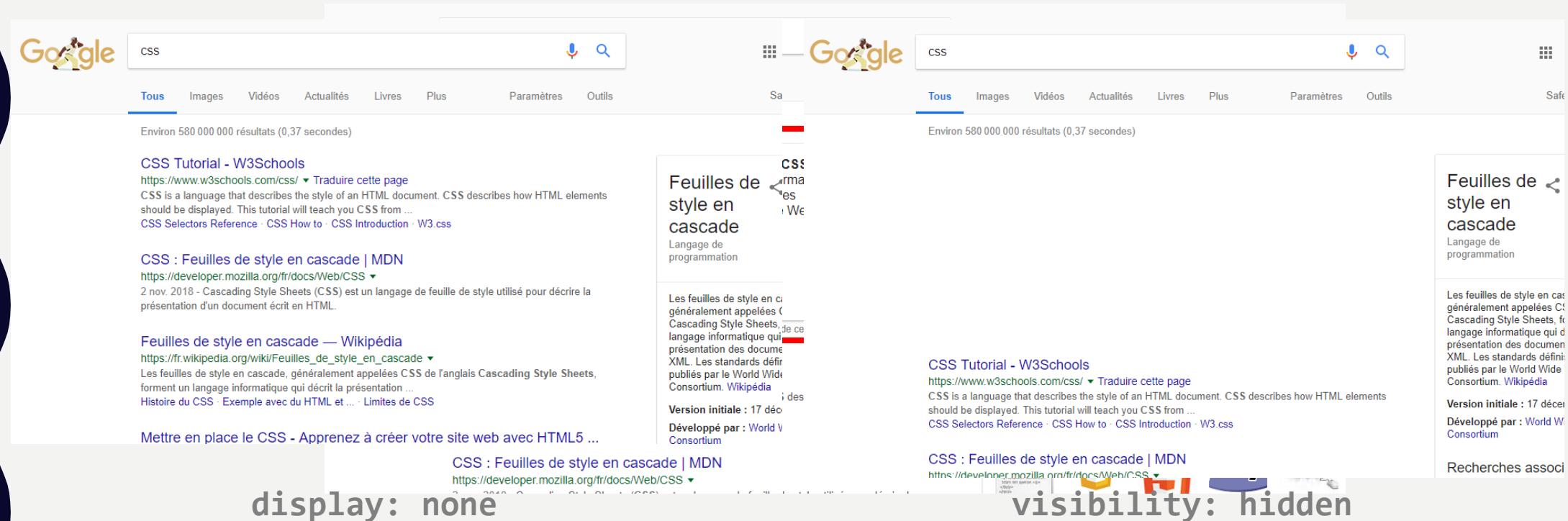
```
p.ex1 {display: none;}  
p.ex2 {display: inline;}  
p.ex3 {display: block;}  
p.ex4 {display: inline-block;}
```

There is also a 'Try it Yourself' button and a note about more examples below.

NOTIONS INTERMÉDIAIRES

DISPLAY

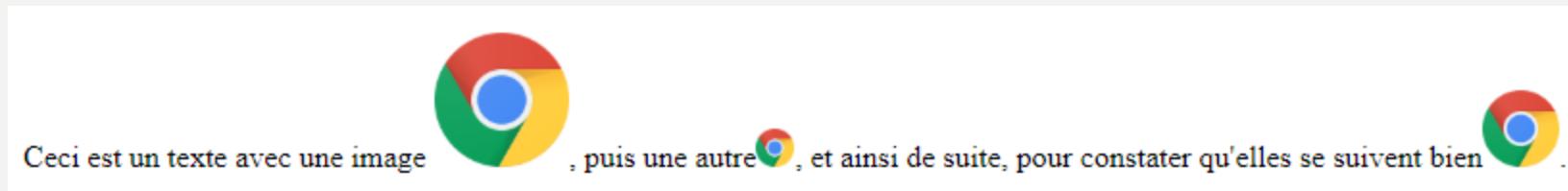
- Différence entre **display: none** et **visibility: hidden**
 - display: none** permet d'effacer entièrement l'élément de l'affichage ;
 - visibility: hidden** masque l'élément, mais ses voisins continuent de le considérer.



NOTIONS INTERMÉDIAIRES

DISPLAY

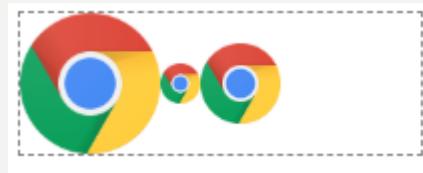
- Alignement des `inline-block`
 - Par défaut, les `inline-block` s'alignent en bas.



- On peut modifier ce comportement à l'aide de `vertical-align` sur l'`inline-block`.



Défaut



`vertical-align: middle`



`vertical-align: top`

NOTIONS INTERMÉDIAIRES

DISPLAY - MÉMO

- A retenir :
 - Chaque élément HTML a une valeur de **display** qui détermine son comportement natif
 - **div = block / span = inline**
 - Un **block** occupe sa ligne seul et a une largeur de 100% par défaut
 - Un **inline-block** est un élément intégré dans le texte
 - **display: none** fait disparaître l'élément
 - **vertical-align** détermine l'alignement des **inline** et **inline-block**
 - Eviter les div superflues



NOTIONS INTERMÉDIAIRES

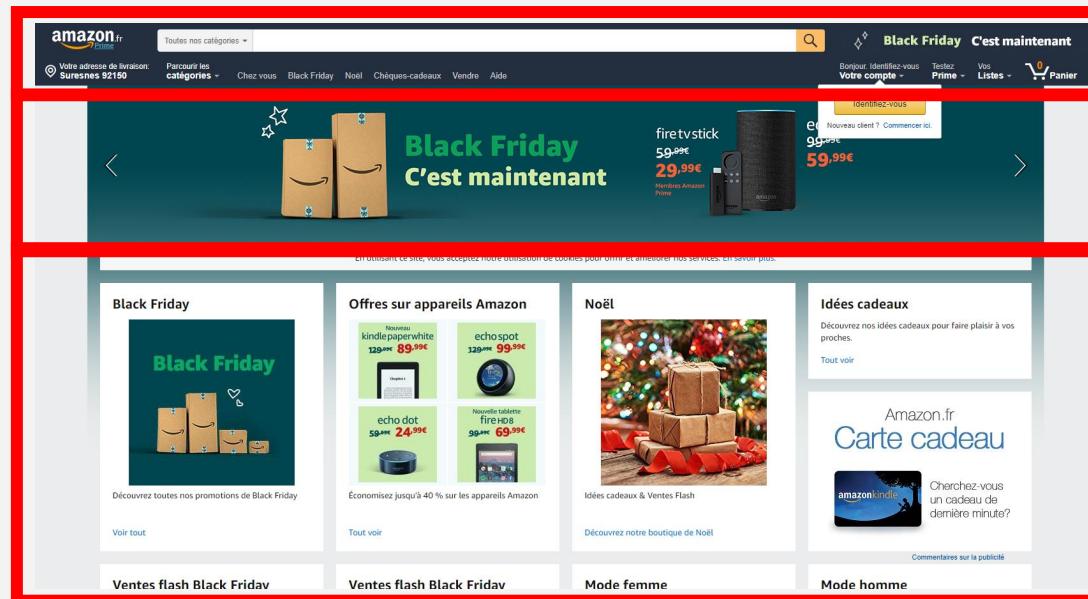
POSITIONNEMENT

Le flux

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

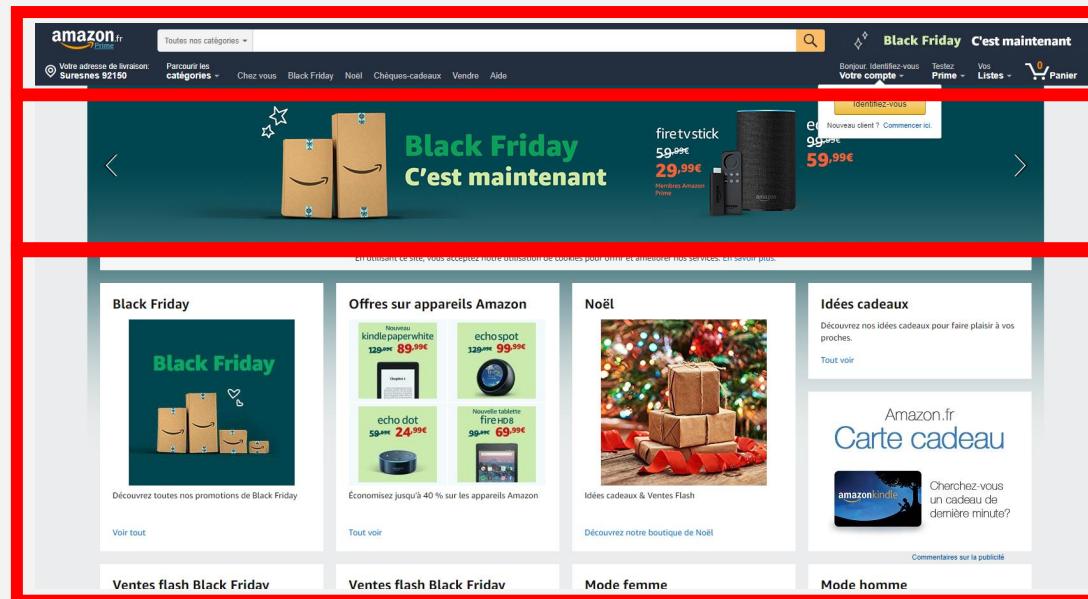
- **Le flux** correspond à la façon dont les éléments de bloc et les éléments en ligne sont disposés avant tout changement apporté à leur disposition.
Lorsqu'un élément est retiré du flux, il est traité indépendamment.



NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

- En clair, c'est bien beau, les blocs les uns en-dessous des autres, mais c'est un peu limité...
- On va donc vite chercher à **sortir les éléments du flux** pour mettre un peu de fun.



NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

- Pour ce faire, on utilise la propriété **position**, qui peut prendre les valeurs :
 - static
 - relative
 - absolute
 - fixed
 - sticky (/IE)
- La propriété **position** va souvent de pair avec les propriétés :
 - top
 - bottom
 - left
 - right
- Il n'existe pas de propriété raccourcie, on doit les préciser individuellement.
- Elles ont un effet différent selon la valeur de **position**.

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

static

- La valeur par défaut de tous les éléments HTML.

relative

- Un élément **static** est dit « non-positionné ».

Toutes les autres valeurs de **position** rendent les éléments « positionnés ».

absolute

- Les propriétés **top**, **bottom**, **left** et **right** n'ont aucun impact et sont ignorées.

fixed



NOTIONS INTERMÉDIAIRES

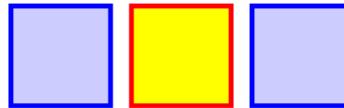
POSITIONNEMENT

static

- Permet de conserver un élément dans le flux tout en le décalant depuis sa position d'origine.

relative

In this demo you can control the position property for the yellow box.



absolute

In this demo you can control the position property for the yellow box.



fixed

- Les éléments voisins prennent toujours en compte l'élément cible (resté dans le flux), mais l'élément est décalé de sa position d'origine.

101

sticky



```
6 .my-block {  
7   position: relative;  
8   top: 40px; left: 40px;  
9 }  
10  
11
```

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

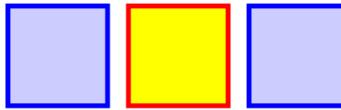
static

- Sort l'élément du flux et le positionne en fonction de son premier parent positionné.



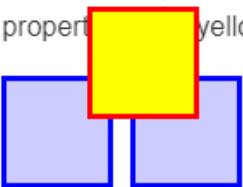
relative

In this demo you can control the position property for the yellow box.



absolute

In this demo you can control the position property for the yellow box.



```
6 .my-block {  
7   position: absolute;  
8   top: 40px; left: 40px;  
9 }
```

A screenshot of a CSS editor interface. It shows a code block with a rule for ".my-block" that includes "position: absolute;" and "top: 40px; left: 40px;" properties.

fixed

- Les éléments voisins ne prennent plus en compte l'élément cible (sorti du flux).
- Les propriétés `top`, `bottom`, `left`, `right` positionnent l'élément par rapport à son parent (le plus proche ayant une position non static).

sticky

The IE logo, which is a stylized lowercase 'e' inside a circle with a diagonal line through it, followed by the word "sticky".

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

static

- Avec **absolute**, on peut dire à un élément de prendre toute la largeur/toute la hauteur de son premier parent positionné sans utiliser **width/height**.

relative

- On indique **left: 0** et **right: 0** pour qu'il se colle aux bords gauche et droit

absolute

- On indique **top: 0** et **bottom: 0** pour qu'il se colle aux bords haut et bas

fixed

- **width** et **height** prendront toujours le dessus sur cette astuce ; il faut donc ne pas les préciser pour en bénéficier.



NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

static

- On indique `left: 0` et `right: 0` pour qu'il se colle aux bords gauche et droit
- On indique `top: 0` et `bottom: 0` pour qu'il se colle aux bords haut et bas

relative

```
position: absolute;  
top: 0;  
left: 0;
```

absolute

```
width: 150px;  
height: 150px;
```

fixed



NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

static

- On indique `left: 0` et `right: 0` pour qu'il se colle aux bords gauche et droit
- On indique `top: 0` et `bottom: 0` pour qu'il se colle aux bords haut et bas

relative

absolute

fixed

sticky

```
position: absolute;  
top: 0;  
left: 0;
```

```
position: relative;  
width: 150px;  
height: 150px;
```

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

static

- On indique `left: 0` et `right: 0` pour qu'il se colle aux bords gauche et droit
- On indique `top: 0` et `bottom: 0` pour qu'il se colle aux bords haut et bas

relative

absolute

```
position: absolute;  
top: 0;  
left: 0;  
right: 0;
```

fixed

```
position: relative;  
width: 150px;  
height: 150px;
```



NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

static

- On indique `left: 0` et `right: 0` pour qu'il se colle aux bords gauche et droit
- On indique `top: 0` et `bottom: 0` pour qu'il se colle aux bords haut et bas

relative

absolute

```
position: absolute;  
top: 0;  
left: 0;  
right: 0;  
bottom: 0;
```

fixed



sticky

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

static

- On indique `left: 0` et `right: 0` pour qu'il se colle aux bords gauche et droit
- On indique `top: 0` et `bottom: 0` pour qu'il se colle aux bords haut et bas

relative

absolute

fixed

sticky

```
position: absolute;  
top: 0;  
left: 0;  
right: 0;  
bottom: 0;  
width: 75px;  
height: 75px;
```

```
position: relative;  
width: 150px;  
height: 150px;
```



NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

static

- On pourrait placer tous les éléments d'une page avec une position **absolute** et des **top/left**.
- On aurait ainsi un agencement au pixel près impeccable.

relative

- Par contre, on serait à l'origine du projet le moins maintenable de tous les temps, et si vous faites ça,
je vous chercherai, je vous trouverai et je vous tuerai.

absolute



fixed

➔ N'utiliser **absolute** qu'en cas d'extrême nécessité, car c'est la valeur la plus difficile à maintenir (sorti du flux, mais agencé en fonction du parent...).



NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

static

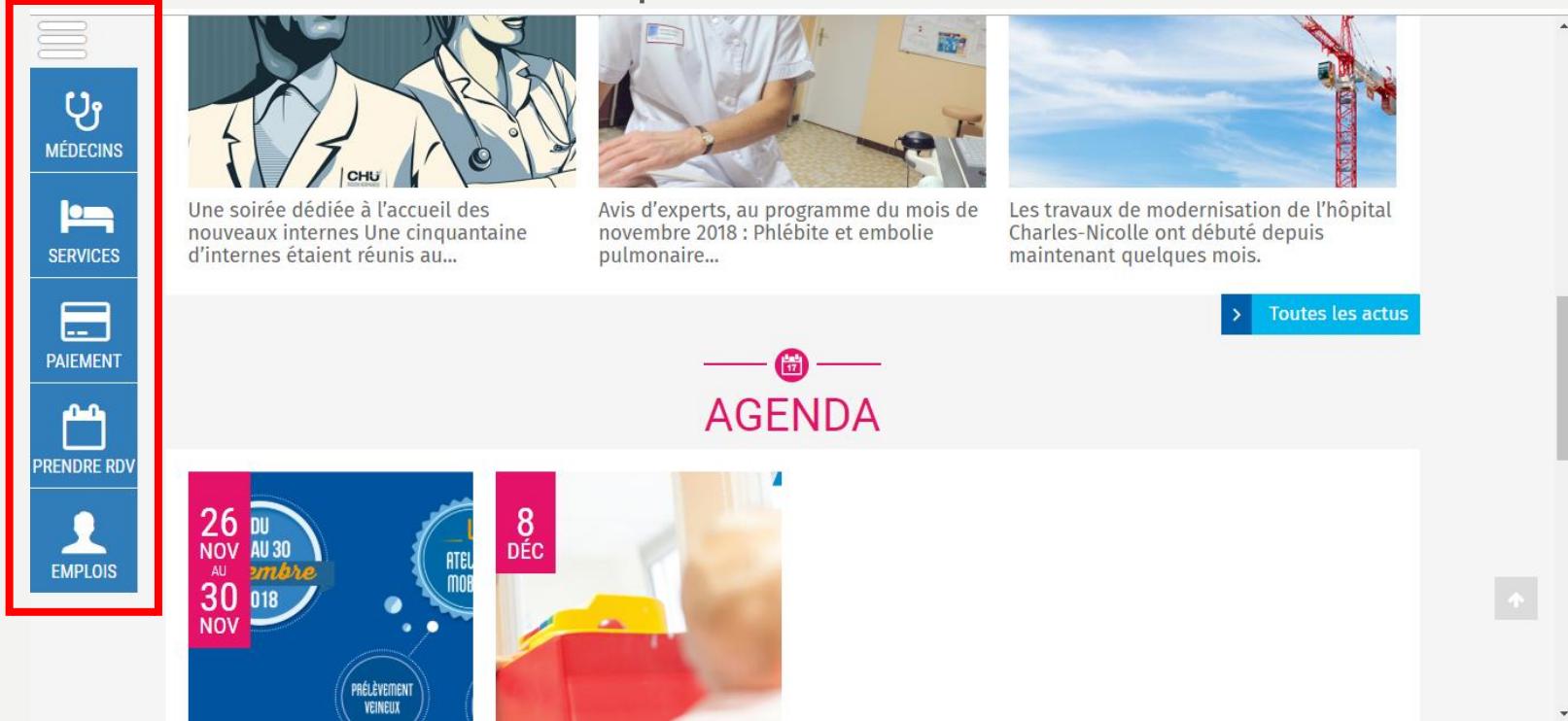
relative

absolute

fixed

sticky

- Sort l'élément du flux et le positionne en fonction de la fenêtre.



- top, bottom, left et right sont en fonction des bords de la fenêtre.

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

static

- **Nouveau !**
- Se « colle » aux bords du premier parent positionné.
- Implémentation en cours...

relative

absolute

fixed

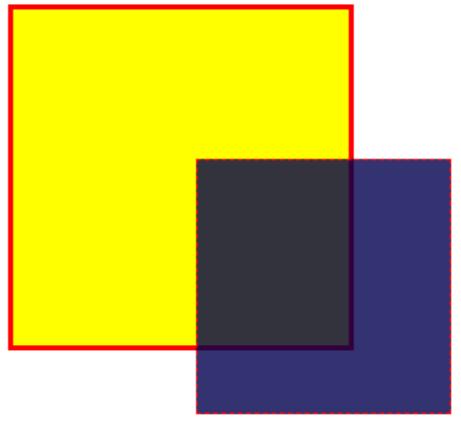


- Exemple :
 - <https://developer.mozilla.org/fr/docs/Web/CSS/position>

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

- On combine souvent **position relative** avec **overflow** pour indiquer si le contenu déplacé doit déborder ou pas.



CSS

```
1 * div {  
2   background-color: yellow;  
3   border: 3px solid red;  
4   width: 200px;  
5   height: 200px;    + overflow: hidden  
6 }  
7  
8 * div > div {  
9   background-color: rgba(0, 0, 77, .8);  
10  border: 1px dashed red;  
11  width: 150px;  
12  height: 150px;  
13  position: relative;  
14  top: 100px;  
15  left: 120px;  
16 }
```



NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

- La propriété `z-index` indique le niveau de l'élément par rapport à ses voisins.
- `z-index` prend un nombre entier quelconque.
- Il ne s'applique qu'aux éléments positionnés.



NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

- La propriété `z-index` indique le niveau de l'élément par rapport à ses voisins.

```
display: inline-block;  
width: 150px;  
height: 150px;
```

```
display: inline-block;  
width: 150px;  
height: 150px;
```

```
width: 150px;  
height: 100px;  
position: absolute;  
top: 120px;  
left: -70px;
```

HTML

```
1 <div id="yellow"></div>  
2 <div id="blue">  
3   <div id="green"></div>  
4 </div>
```

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

- La propriété `z-index` indique le niveau de l'élément par rapport à ses voisins.

```
display: inline-block;  
width: 150px;  
height: 150px;  
z-index: 2;
```

```
display: inline-block;  
width: 150px;  
height: 150px;
```

```
width: 150px;  
height: 100px;  
position: absolute;  
top: 120px;  
left: -70px;  
z-index: 1;
```

HTML

```
1 <div id="yellow"></div>  
2 <div id="blue">  
3   <div id="green"></div>  
4 </div>
```

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

- La propriété `z-index` indique le niveau de l'élément par rapport à ses voisins.

```
display: inline-block;  
width: 150px;  
height: 150px;  
z-index: 2;  
position: relative;
```

```
display: inline-block;  
width: 150px;  
height: 150px;
```

```
width: 150px;  
height: 100px;  
position: absolute;  
top: 120px;  
left: -70px;  
z-index: 1;
```

HTML

```
1 <div id="yellow"></div>  
2 <div id="blue">  
3   <div id="green"></div>  
4 </div>
```

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

- La propriété `z-index` indique le niveau de l'élément par rapport à ses voisins.

```
display: inline-block;  
width: 150px;  
height: 150px;  
z-index: 2;  
position: relative;
```

```
display: inline-block;  
width: 150px;  
height: 150px;  
position: relative;  
z-index: 3;
```

```
width: 150px;  
height: 100px;  
position: absolute;  
top: 120px;  
left: -70px;  
z-index: 1;
```

HTML

```
1 <div id="yellow"></div>  
2 <div id="blue">  
3   <div id="green"></div>  
4 </div>
```

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

- La propriété **float** permet de sortir un élément du flux, tout en faisant en sorte que les voisins s'adaptent à lui où qu'il soit.
- On l'a vu :
 - **position: relative** déplace l'élément, mais les voisins s'adaptent à son emplacement d'origine
 - **position: absolute** déplace l'élément, mais les voisins font comme si il n'existe plus
- **float** permet, elle, de déplacer l'élément et adapter les voisins.
- Un exemple idéal est la gestion des images sur Wikipédia.

Imprimer / exporter
Créer un livre
Télécharger comme PDF
Version imprimable

Dans d'autres projets
Wikimedia Commons
Meta-Wiki
Wikiquote

Dans d'autres langues
Alemannisch
Brezhoneg
Corsu
Deutsch
English
Español
Italiano
Occitan
Português

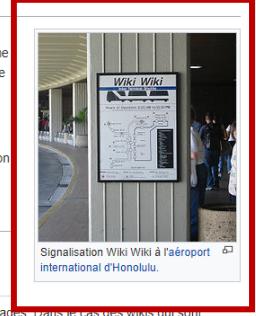
119 de plus
Modifier les liens

5 Voir aussi
5.1 Articles connexes
5.2 Bibliographie
5.3 Liens externes

Définition du terme « wiki »
Le mot « wiki » signifie, en hawaïen, *rapide*², *vite*³ ou *informel*⁴. Il a été choisi par Ward Cunningham lorsqu'il crée le premier wiki, qu'il appela WikiWikiWeb. Il utilise l'expression « wiki wiki », un *redoublement* qui signifie « très rapide », « très vite »⁵, car c'est le premier terme hawaïen qu'il apprit lorsqu'il dut prendre un bus à la sortie de l'aéroport, et qu'à la création de son site, il voulait un terme amusant pour dire rapide. Dans l'URL du site, apparaît uniquement le terme « wiki », ce qui a probablement poussé les visiteurs à l'appeler ainsi⁶. Pour l'OQLF le terme « wiki » est donc un nom commun qui s'accorde au pluriel⁶.
Le journal *The Economist* fait remarquer que le mot *wiki* peut être lu comme l'*acronyme* de « *What I Know Is* » (littéralement : « ce que je sais est » ou « voici ce que je sais »)⁷. Le concours de création littéraire et artistique *Dis-moi dix mots* a sélectionné le mot « *wiki* » pour son édition de 2014-2015⁸ et en donne une définition.

Fonctionnement technique
Un wiki fonctionne grâce à un moteur de wiki : c'est un logiciel installé sur le système hôte du site web.

Identification des visiteurs
Un wiki n'est pas forcément modifiable par tout le monde ; il peut exiger que les visiteurs s'inscrivent avant d'être autorisés à modifier les pages. Dans le cas des wikis qui sont complètement ouverts au public, diverses procédures techniques et sociales sont mises en œuvre pour limiter et annuler les modifications indésirables.
Lorsqu'un wiki autorise des visiteurs anonymes à modifier les pages, c'est l'adresse IP de ces derniers qui les identifie ; les utilisateurs inscrits peuvent quant à eux se connecter sous leur nom d'utilisateur.



NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

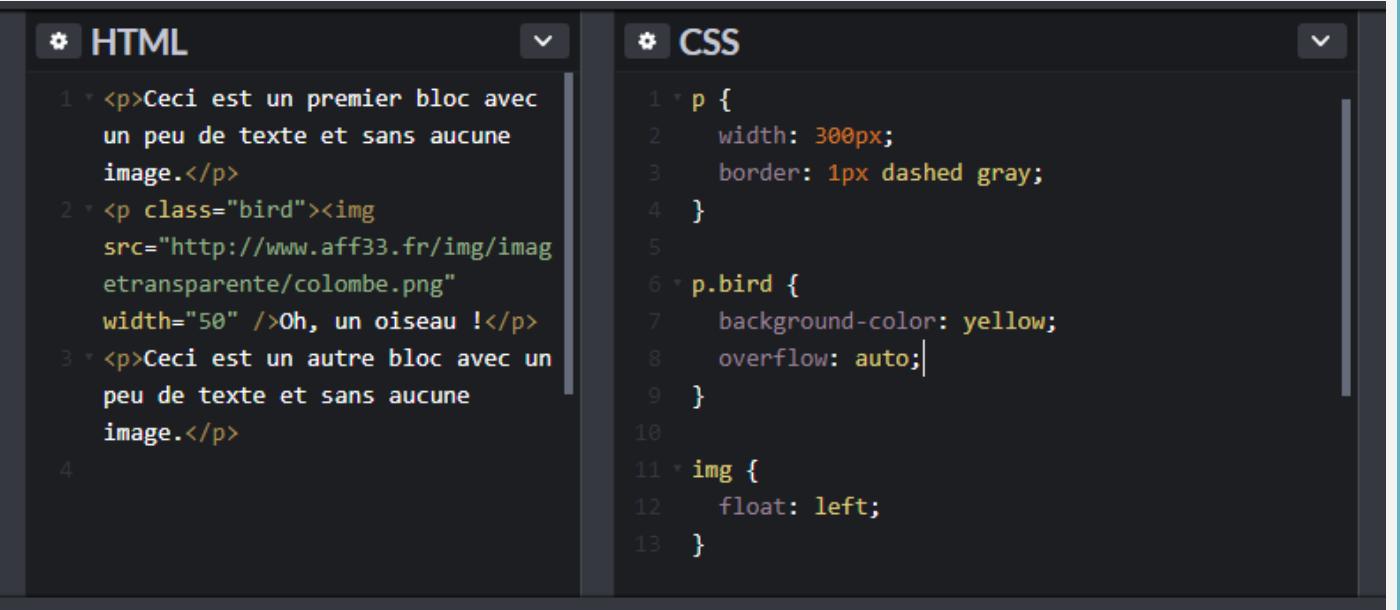


- Un élément peut être flottant à gauche ou flottant à droite.
 - Il s'aligne verticalement en fonction de son emplacement dans le HTML.
 - Pour le déplacer, on joue sur les margin.
-
- Les **float** sont usés avec abus par de nombreux frameworks comme **Bootstrap (< 4)** car ils facilitent le développement *responsive* (les éléments flottants se mettent automatiquement les uns sous les autres quand l'espace manque).
 - Cependant, il ne faut, ici non plus, surtout pas en abuser, car ils sont difficiles à maintenir.

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT

- Un élément flottant sort du flux.
- Cela signifie qu'un élément flottant plus haut que son parent va « déborder ».
- Si on souhaite que le parent s'adapte à ses enfants flottants, on utilise `overflow: auto`.



The screenshot shows a code editor with two panes: HTML on the left and CSS on the right.

HTML:

```
1 <p>Ceci est un premier bloc avec  
2   un peu de texte et sans aucune  
3   image.</p>  
4 <p class="bird">Oh, un oiseau !</p>  
8 <p>Ceci est un autre bloc avec un  
9   peu de texte et sans aucune  
10  image.</p>
```

CSS:

```
1 p {  
2   width: 300px;  
3   border: 1px dashed gray;  
4 }  
5  
6 p.bird {  
7   background-color: yellow;  
8   overflow: auto;  
9 }  
10  
11 img {  
12   float: left;  
13 }
```

The visual representation below the code editor shows three blocks of text. The first block is a standard paragraph. The second block contains an image of a dove (a white bird) and the text "Oh, un oiseau !". The third block is another standard paragraph. The dove image is positioned to the left of the text "Oh, un oiseau !", demonstrating its effect as a floating element.

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT - MÉMO

- A retenir :
 - Les valeurs de position :
 - **static** (par défaut) : non-positionné
 - **relative** :
 - Pour déplacer un élément depuis sa position d'origine
 - Pour « positionner » un élément en vue d'agencement ses enfants
 - **absolute** (si vraiment nécessaire) :
 - Pour déplacement un élément en fonction des bords de son parent
 - **fixed** :
 - Pour déplacer un élément en fonction des bords de l'écran
 - **z-index** s'applique aux éléments positionnés
 - **float** sort un élément du flux en faisant en sorte que ses voisins le prennent en compte
 - Exemple : vignettes images Wikipédia



NOTIONS INTERMÉDIAIRES

POSITIONNEMENT – CENTRER HORIZONTALEMENT

- Centrer un élément horizontalement dans son parent dépend de sa valeur de display et de position.

The screenshot shows a code editor with two tabs: 'HTML' and 'CSS'. The 'HTML' tab contains the following code:

```
1 <div id="blue">
2   <span id="green">abc</span>
3 </div>
```

The 'CSS' tab contains the following code:

```
1 #blue {
2   border: 1px solid gray;
3   width: 450px;
4   height: 150px;
5   padding: 2px;
6   text-align: center;
7 }
8
9 #green {
10   border: 1px solid red;
11 }
```

Below the code editor, a preview window shows a large blue square with a red border. Inside the blue square, the word 'abc' is written in black text and is centered horizontally within the blue area. The letter 'a' is highlighted with a red rectangular box.

L'élément est un **inline** (ex. **span**) ou **inline-block**

Les **inline** et les **inline-block** s'intègrent dans le texte.

Pour les centrer horizontalement, on utilise donc la propriété **text-align** du parent.

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT – CENTRER HORIZONTALEMENT

- Centrer un élément horizontalement dans son parent dépend de sa valeur de display et de position.

The screenshot shows a code editor with two panes. The left pane is labeled "HTML" and contains the following code:

```
1 <div id="blue">
2   <table id="green">
3     <tr>
4       <td>Cell11</td>
5       <td>Cell12</td>
6     </tr>
7     <tr>
8       <td>Cell21</td>
9       <td>Cell22</td>
10    </tr>
11   </table>
12 </div>
```

The right pane is labeled "CSS" and contains the following code:

```
1 #blue {
2   border: 1px solid gray;
3   width: 450px;
4   height: 150px;
5   padding: 2px;
6   text-align: center;
7 }
8
9 #green {
10   border: 1px solid red;
11   margin: 0 auto;
12 }
```

Below the code editor, a preview window shows a blue square containing a green table. The table has two rows, each with two cells. The cells are centered horizontally within their respective table columns. The first row's cells are labeled "Cell11" and "Cell12", and the second row's cells are labeled "Cell21" and "Cell22". The entire table is centered within the blue square.

L'élément est un **block** (ex. `table`)

Ce n'est donc pas du texte :
`text-align` sur le parent n'aura
pas l'effet escompté.

On utilise ici
`margin-left` et `margin-right`.

Ou sa version raccourcie
(à condition de spécifier haut et bas).

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT – CENTRER HORIZONTALEMENT

- Centrer un élément horizontalement dans son parent dépend de sa valeur de display et de position.



The screenshot shows a code editor with two tabs: 'HTML' and 'CSS'. The 'HTML' tab contains the following code:

```
1 <div id="blue">
2   <div id="green"></div>
3 </div>
```

The 'CSS' tab contains the following code:

```
8 #green {
9   border: 1px solid red;
10  position: absolute;
11  width: 100px;
12  height: 100px;
13  top: 20px;
14  left: calc(50% - 50px);
15 }
```

Below the code editor is a diagram illustrating the horizontal centering. It shows a large white square representing the parent element. Inside it, a smaller red square represents the child element (#green). A blue double-headed arrow at the bottom indicates a width of 50%. A purple star is positioned at the center of the red square, with the text '50px' written above it, indicating the distance from the left edge of the red square to its center.

L'élément est en position **absolute** ou **fixed**

Puisqu'il n'est plus dans le flux,
les propriétés de son parent
ne l'affectent pas.

On joue donc avec
left et la fonction **calc()**.

Attention : cela nécessite que l'élément ait une largeur fixe en pixels.
En général, un élément **absolute** n'a pas à être centré horizontalement
(bad feng-shui).

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT – CENTRER VERTICALEMENT

Best way to center a <div> on a page vertically and horizontally?

Asked 11 years, 5 months ago Active 11 months ago Viewed 911k times

527
Closed last year.

This question already has answers here:

[How to center an element horizontally and vertically](#) (24 answers)

238

Best way to center a <div> element on a page both vertically and horizontally?

⌚

I know that `margin-left: auto; margin-right: auto;` will center on the horizontal, but what is the best way to do it vertically, too?

html css alignment vertical-alignment centering

share improve this question follow

edited May 1 '18 at 10:41

 John Slegers
34.5k 15 168 143

asked Dec 10 '08 at 17:11

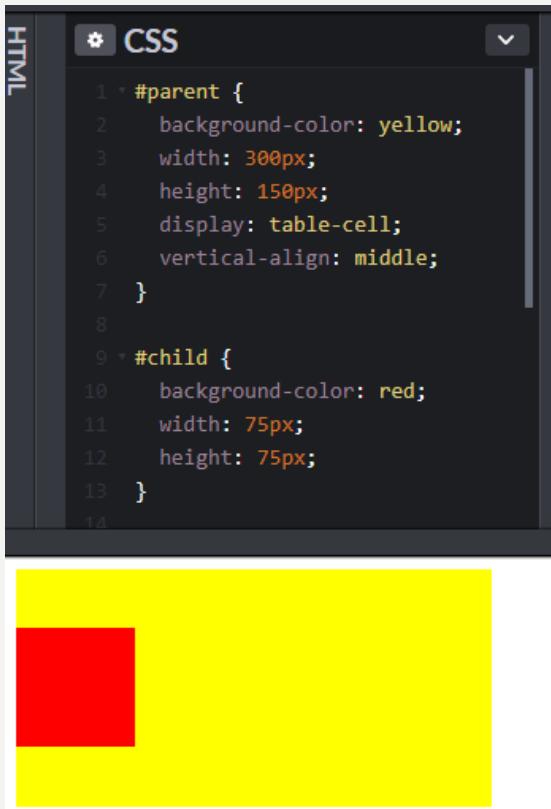
 J-Dog

- C'est une question épineuse car on a l'habitude des éléments dont la hauteur est définie en fonction de leur contenu, et pas d'éléments qui s'adapteraient à la hauteur de leur parent.

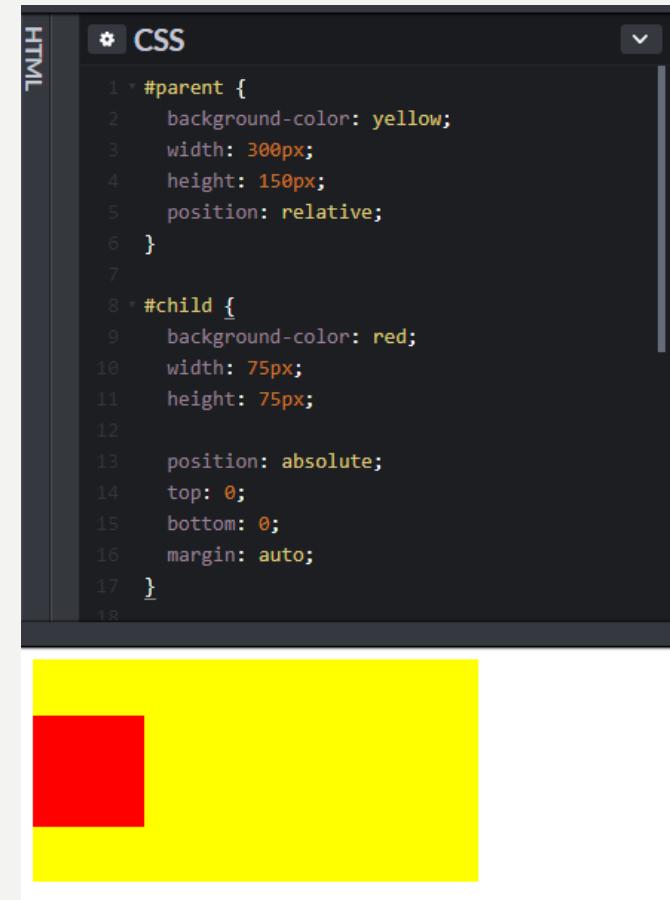
NOTIONS INTERMÉDIAIRES

POSITIONNEMENT – CENTRER VERTICALEMENT

- Il y a plusieurs méthodes, toutes pas hyper feng-shui :



```
HTML CSS
1 * #parent {
2   background-color: yellow;
3   width: 300px;
4   height: 150px;
5   display: table-cell;
6   vertical-align: middle;
7 }
8
9 * #child {
10  background-color: red;
11  width: 75px;
12  height: 75px;
13 }
```

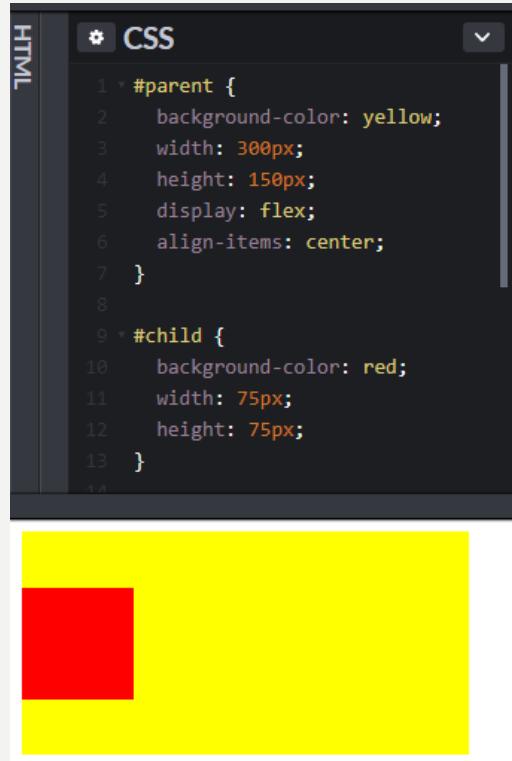


```
HTML CSS
1 * #parent {
2   background-color: yellow;
3   width: 300px;
4   height: 150px;
5   position: relative;
6 }
7
8 * #child {
9  background-color: red;
10 width: 75px;
11 height: 75px;
12
13 position: absolute;
14 top: 0;
15 bottom: 0;
16 margin: auto;
17 }
18 }
```

NOTIONS INTERMÉDIAIRES

POSITIONNEMENT – CENTRER VERTICALEMENT

- Heureusement, le W3C dans sa grande bonté a inventé les flexbox...



- ... mais on verra ça après ;)

Des questions ?





CONCEPTS AVANCÉS

PSEUDO-MACHINS, POLICES,
FLEXBOXES, RESPONSIVE DESIGN

CONCEPTS AVANCÉS

INTRODUCTION

- Maintenant que vous êtes un peu plus confirmés, voyons quelques outils sympas qui vous distingueront du reste du troupeau.

1 Pseudo-classes

:hover
:checked
:disabled, :read-only, :required
:focus
:active, :visited
:first-child, :last-child...
:not

2 Pseudo-éléments

::first-letter / ::first-line
::before / ::after

3 Polices

- Intégration
- Icônes

4 Layout

- Flexbox
- Grid

5 Responsive

- Media query
- Mobile-first

CONCEPTS AVANCÉS

PSEUDO-CLASSES

- « Une pseudo-classe est un mot-clé qui peut être ajouté à un sélecteur afin d'indiquer l'état spécifique dans lequel l'élément doit être pour être ciblé par la déclaration. »
- Liste non-exhaustive :
 - `:hover` appliqué au survol de l'élément
 - `:checked` appliqué sur les checkbox et boutons radio cochés
 - `:disabled` appliqué sur les champs désactivés (= [disabled])
 - `:read-only` appliqué sur les champs en lecture seule (= [read-only])
 - `:required` appliqué sur les champs obligatoires (= [required]) (cf. « **Know your HTML!** »)
 - `:focus` appliqué sur le champ qui a le focus
 - `:active` appliqué sur le lien en cours de clic
 - `:visited` appliqué sur le lien précédemment visité



CONCEPTS AVANCÉS

PSEUDO-CLASSES

- Exercice :

```
a:hover {  
    background-color: yellow;  
    color: red;  
}  
  
.my-form :required {  
    color: red;  
}  
  
.my-form :checked + label {  
    background-color: green;  
}  
  
.item-menu:hover .sub-menu {  
    display: block;  
}
```

Au survol d'un lien,
celui-ci passe en rouge sur fond jaune.

Les champs obligatoires présents
dans l'élément de classe « `my-form` »
sont écrits en rouge.

Les labels qui suivent des cases cochées
présentes dans l'élément de classe « `my-form` »
ont un fond vert.

Au survol d'un élément de classe « `item-menu` »,
ses descendants de classe « `sub-menu` » s'affichent.

CONCEPTS AVANCÉS

PSEUDO-CLASSES

- Exemples d'usage :

The screenshot shows a code editor with two tabs: 'HTML' and 'CSS'. The 'HTML' tab contains the following code:

```
1 <div class="form-row">
2   <input type="checkbox" id="agree" />
3   <label for="agree">J'ai compris les conditions</label>
4 </div>
5
6 <div class="form-row">
7   <input type="checkbox" id="agree2" checked />
8   <label for="agree2">J'ai compris les conditions</label>
9 </div>
```

The 'CSS' tab contains the following code:

```
1 .form-row { position: relative; width: 300px; margin-bottom: 10px; }
2
3 .form-row label {
4   display: block;
5   border: 1px solid gray;
6   border-radius: 10px;
7 }
8
9 .form-row input[type=checkbox] {
10   float: left;
11 }
12
13 .form-row :checked + label {
14   background-color: #39b45a;
15 }
```

Below the code editor, there are two form rows. The first row has an unchecked checkbox and a label 'J'ai compris les conditions'. The second row has a checked checkbox (indicated by a green checkmark) and a label 'J'ai compris les conditions', which is styled with a green background color.

CONCEPTS AVANCÉS

PSEUDO-CLASSES

- Exemples d'usage :



HTML

```
1 <menu>
2   <ul>
3     <li>Menu 1</li>
4     <li>Menu 2
5       <ul>
6         <li>Sous-menu 1</li>
7         <li>Sous-menu 2</li>
8       </ul>
9     </li>
10    <li>Menu 3</li>
11  </menu>
```

CSS

```
1 * menu > ul > li {
2   display: inline-block;
3   background-color: darkred;
4   color: white;
5   position: relative;
6   padding: 10px 20px;
7 }
8
9 * menu > ul > li ul {
10   display: none;
11   position: absolute;
12   top: 100%;
13   left: 0;
14   background-color: red;
15   white-space: nowrap;
16   padding: 10px;
17 }
18
19 * menu > ul > li:hover ul {
20   display: block;
21 }
```

Menu 1 Menu 2 Menu 3

Menu 1 Menu 2 Menu 3

Sous-menu 1
Sous-menu 2

CONCEPTS AVANCÉS

PSEUDO-CLASSES

- Il existe des pseudo-classes pour sélectionner un élément en fonction de ses voisins.
- La pseudo-classe `:first-child`/`:last-child` permet de sélectionner l'élément s'il est le premier enfant/le dernier enfant de son parent.

The screenshot shows a browser developer tools window with the following components:

- Toolbar:** Includes icons for Home, Refresh, Stop, and Run, followed by a green "Run" button.
- Code Editor:** Displays the following CSS and HTML code:

```
<!DOCTYPE html>
<html>
<head>
<style>
p:first-child {
    background-color: yellow;
}
</style>
</head>
<body>

<p>This paragraph is the first child of its parent (body).</p>
<h1>Welcome to My Homepage</h1>
<p>This paragraph is not the first child of its parent.</p>

<div>
    <p>This paragraph is the first child of its parent (div).</p>
    <p>This paragraph is not the first child of its parent.</p>
</div>

<p><b>Note:</b> For :first-child to work in IE8 and earlier, a DOCTYPE must be declared.</p>

</body>
</html>
```
- Result Area:** Shows the rendered HTML with styling applied:
 - The first paragraph inside the body is yellow because it is a child of the body element.
 - The first paragraph inside the div is yellow because it is a child of the div element.
 - The other paragraphs are white because they are not the first children of their respective parents.
- Text at the bottom:** "Result Size: 668 x 477"

Note: For `:first-child` to work in IE8 and earlier, a DOCTYPE must be declared.

Text on the right: La pseudo-classe `:only-child` est un raccourci pour `:first-child:last-child`

CONCEPTS AVANCÉS

PSEUDO-CLASSES

- Il existe des pseudo-classes pour sélectionner un élément en fonction de ses voisins.
- La pseudo-classe `:nth-child():nth-last-child()` permet de sélectionner le n-ième enfant/le n-ième dernier enfant de son parent.

```
HTML
1 <ol>
2   <li>Item</li>
3   <li>Item</li>
4   <li>Item</li>
5   <li>Item</li>
6   <li>Item</li>
7   <li>Item</li>
8   <li>Item</li>
9   <li>Item</li>
10 </ol>
```

```
CSS
1 li:nth-child(4) {
2   background-color: yellow;
3 }
4
5 li:nth-last-child(2) {
6   background-color: cyan;
7 }
```

1. Item
2. Item
3. Item
4. Item
5. Item
6. Item
7. Item
8. Item

On note que le compteur commence à 1 et pas à 0.

CONCEPTS AVANCÉS

PSEUDO-CLASSES

- Il existe des pseudo-classes pour sélectionner un élément en fonction de ses voisins.
- La pseudo-classe `:nth-child():nth-last-child()` peut prendre en argument « n », ce qui permet de sélectionner une fréquence d'éléments (suite arithmétique).



`:nth-child(n+6)`



`:nth-child(-n+5)`



`:nth-child(4n+1)`



`:nth-child(odd)`



`:nth-child(even)`

CONCEPTS AVANCÉS

PSEUDO-CLASSES

- Il existe des pseudo-classes pour sélectionner un élément en fonction de ses voisins.
- La pseudo-classe `:nth-child():nth-last-child()` peut prendre en argument « n », ce qui permet de sélectionner une fréquence d'éléments (suite arithmétique).

On peut ainsi colorer une ligne sur deux dans un tableau standard sans utiliser ni JavaScript ni classes « ligne-paire »/« ligne impaire » !

```
tr:nth-child(even) td {  
    background-color: green;  
}  
  
tr:nth-child(odd) td {  
    background-color: blue;  
}
```

pseudo	score	durée	équipe
Toto	1643	3	villetaneuse
Marc	734	1	villetaneuse
Tom	75	6	epinay
Titi	12	4	villetaneuse
Joe	2	2	epinay
Ben	0	20	epinay

CONCEPTS AVANCÉS

PSEUDO-CLASSES

- La pseudo-classe `:not()` permet de sélectionner un élément qui ne correspond pas au sélecteur passé en argument.
- Exemples :

```
a:not([href^=http]) {  
    color: blue;  
}  
  
.menu-item:not(:first-child) {  
    font-size: 2em;  
}  
  
article:not(.sticky) h2 {  
    text-decoration: underline;  
}
```

Les liens dont la cible ne commence pas par « http »

Les éléments de classe « menu-item » qui ne sont pas premier élément de leur parent

Les titres de niveau 2 qui sont dans des articles qui n'ont pas la classe « sticky »

CONCEPTS AVANCÉS

PSEUDO-ÉLÉMENTS

- En CSS, un sélecteur de pseudo-élément applique des styles à des parties du contenu de votre document dans le cas où il n'y a pas d'élément HTML à cibler.
- Liste non exhaustive :
 - **::first-letter** la première lettre de l'élément
 - **::first-line** la première ligne de l'élément
 - **::before** l'espace au début de l'élément
 - **::after** l'espace à la fin de l'élément
- Les **pseudo-éléments** permettent d'ajouter des éléments de mise en forme sans polluer le HTML.
➔ Structure de l'information VS mise en forme



Les pseudo-éléments, au contraire des pseudo-classes, s'écrivent avec deux « deux-points ».

CONCEPTS AVANCÉS

PSEUDO-ÉLÉMENTS – FIRST-LETTER/FIRST-LINE

- Le sélecteur de pseudo-élément `::first-letter` sélectionne la première lettre de l'élément.
Cela permet notamment de faire des lettrines.

The screenshot shows a code editor interface with two panels: HTML on the left and CSS on the right. The HTML panel contains the following code:

```
1 <p>Once upon a time in a  
2 wonderful IT company were  
3 ten magnificent engineers  
4 who wanted to learn some  
5 CSS tricks.</p>
```

The CSS panel contains the following code:

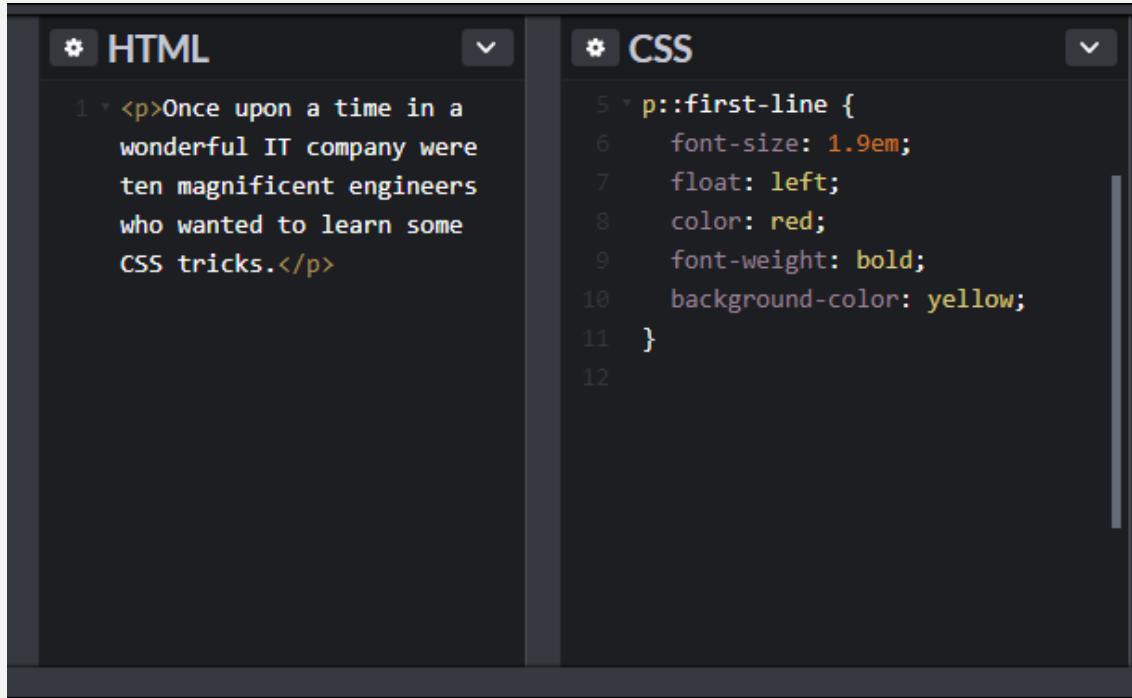
```
5 p::first-letter {  
6   font-size: 1.9em;  
7   float: left;  
8   color: red;  
9   font-weight: bold;  
10  background-color: yellow;  
11 }  
12
```

Once upon a time in a wonderful IT company were ten magnificent engineers who wanted to learn some CSS tricks.

CONCEPTS AVANCÉS

PSEUDO-ÉLÉMENTS – FIRST-LETTER/FIRST-LINE

- Le sélecteur de pseudo-élément `::first-line` sélectionne la première ligne de l'élément.
Cela permet notamment de faire des... euh... jamais utilisé, en fait, mais vous voyez le principe.



The screenshot shows a code editor with two panes. The left pane is labeled "HTML" and contains the following code:

```
<p>Once upon a time in a  
wonderful IT company were  
ten magnificent engineers  
who wanted to learn some  
CSS tricks.</p>
```

The right pane is labeled "CSS" and contains the following code:

```
p::first-line {  
    font-size: 1.9em;  
    float: left;  
    color: red;  
    font-weight: bold;  
    background-color: yellow;  
}
```

Once upon a time in a
wonderful IT company were ten magnificent
engineers who wanted to learn some CSS
tricks.

CONCEPTS AVANCÉS

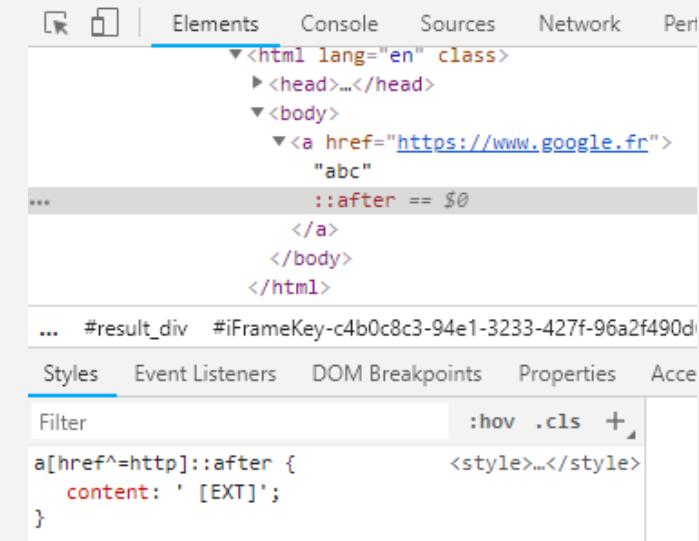
PSEUDO-ÉLÉMENTS – BEFORE/AFTER

- Les sélecteurs de pseudo-élément ::before et ::after sélectionnent l'espace avant et après le contenu de l'élément.
- Ces sélecteurs ne fonctionnent que si l'on précise la propriété « **content** ».
- Ils permettent d'ajouter des éléments de mise en forme qui n'ont rien à faire dans le HTML.

- Exemple :

```
a[href^=http]::after {  
    content: ' [EXT]';  
}
```

[abc \[EXT\]](#)



CONCEPTS AVANCÉS

PSEUDO-ÉLÉMENTS – BEFORE/AFTER

- before/after peuvent avoir trois usages, selon la valeur que l'on donne à content.

L'ajout de texte
avant/après l'élément

```
li::before {  
    content: "> ";  
    font-weight: bold;  
}
```

- > Item 1
- > Item 2

L'ajout d'une image
avant/après l'élément

```
a[href^=http]::after {  
    content: url(..../images/external-link.png);  
}
```

Liens externes

- (en) [WikiMatrix.org](#) : Outil de comparaison des moteurs wiki ↗ [archive]
- (en) [wikipatterns.com](#) ↗ [archive] /les modèles de comportements s

A noter : content n'accepte que du texte brut
(pas de HTML, pas d'entités &xxx;, mais caractères hexadécimaux possibles (\xxxx))

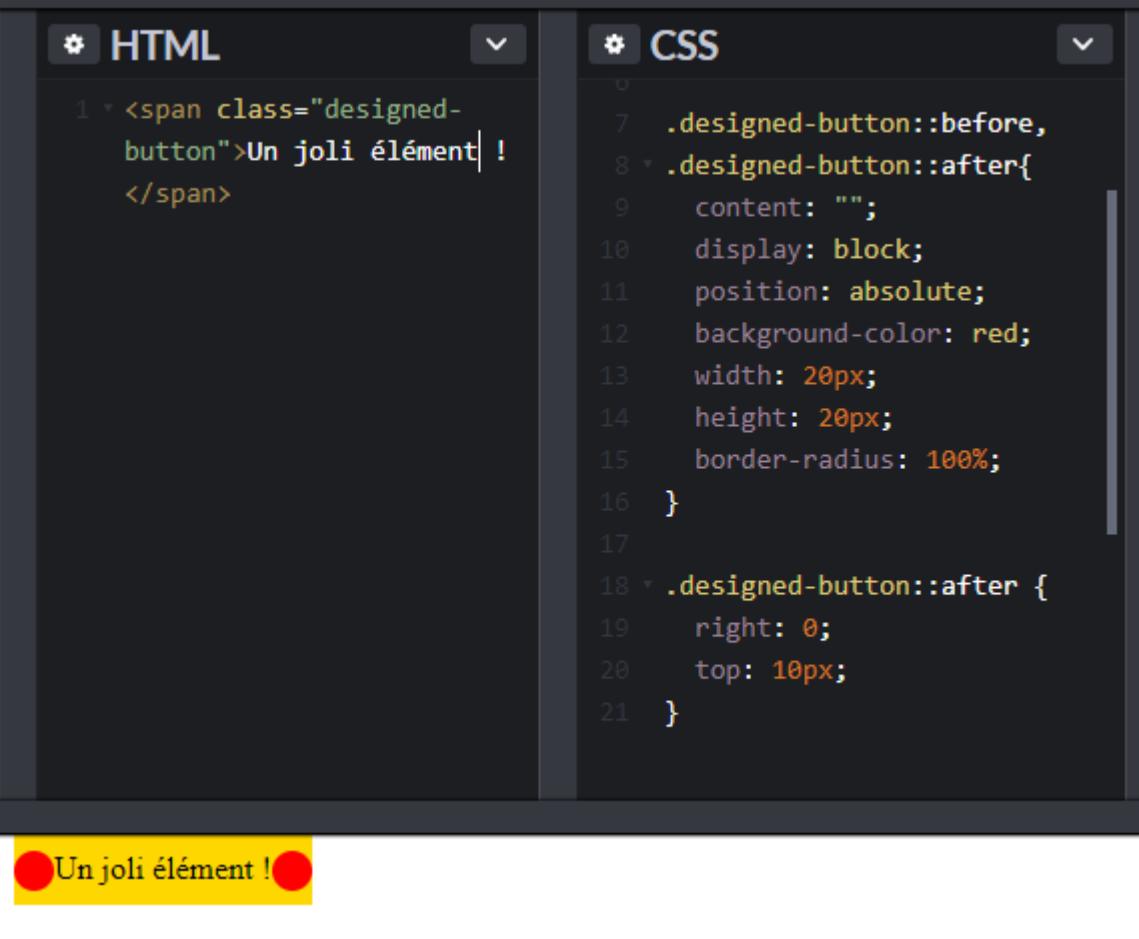
CONCEPTS AVANCÉS

PSEUDO-ÉLÉMENTS – BEFORE/AFTER

- before/after peuvent avoir trois usages, selon la valeur que l'on donne à content.

**L'ajout d'un bloc
entièvement personnalisable !**

On met une chaîne vide
en tant que content
et on peut manipuler le pseudo-élément
comme n'importe quel élément HTML



The screenshot shows a code editor interface with two panels: HTML on the left and CSS on the right. In the HTML panel, there is a single line of code: `Un joli élément !`. In the CSS panel, there are two style definitions for the `.designed-button` class. The first definition, starting at line 7, applies to the `::before` pseudo-element and sets its content to an empty string, displays it as a block, positions it absolutely, gives it a red background color, a width and height of 20px, and a border-radius of 100%. The second definition, starting at line 18, applies to the `::after` pseudo-element and sets its right position to 0 and its top position to 10px. Below the code editor, a yellow rectangular box contains the text "Un joli élément !" with red circular markers on either side, demonstrating the visual result of the CSS styling.

```
HTML
1 <span class="designed-button">Un joli élément !</span>

CSS
7 .designed-button::before,
8 .designed-button::after{
9   content: "";
10  display: block;
11  position: absolute;
12  background-color: red;
13  width: 20px;
14  height: 20px;
15  border-radius: 100%;
16 }
17
18 .designed-button::after {
19   right: 0;
20   top: 10px;
21 }
```

CONCEPTS AVANCÉS

PSEUDO-ÉLÉMENTS – BEFORE/AFTER

- before/after peuvent avoir trois usages, selon la valeur que l'on donne à content.
- Exemple :

```
<span class="keyword">CSS</span>
```

```
.keyword {  
    position: relative;  
    background-color: brown;  
    color: white;  
    padding: 5px;  
    box-shadow: -1px 1px 5px gray;  
    font-family: sans-serif;  
    font-weight: bold;  
    font-size: 2em;  
}
```

```
.keyword::before {  
    content: "";  
    position: absolute;  
    right: -15px;  
    top: calc(50% - 5px);  
    background-color: white;  
    width: 10px;  
    height: 10px;  
    border-radius: 10px;  
    z-index: 1;  
    box-shadow: -1px 1px 2px gray inset;  
}
```

```
.keyword::after {  
    content: "";  
    position: absolute;  
    right: -30px;  
    top: 0;  
    border-left: 30px solid brown;  
    border-top: 23px solid white;  
    border-bottom: 23px solid white;  
}
```



CONCEPTS AVANCÉS

PSEUDO-ÉLÉMENTS – BEFORE/AFTER

- before/after peuvent avoir trois usages, selon la valeur que l'on donne à content.
- Exemple :

```
<span class="keyword">CSS</span>
```

```
.keyword {  
    position: relative;  
    background-color: brown;  
    color: white;  
    padding: 5px;  
    box-shadow: -1px 1px 5px gray;  
    font-family: sans-serif;  
    font-weight: bold;  
    font-size: 2em;  
}
```

Car on va avoir des enfants **absolute** à positionner

box-shadow (rappel) :

- 1px de décalage horizontal (donc vers la gauche)
- 1px de décalage vertical (donc vers le bas)



CONCEPTS AVANCÉS

PSEUDO-ÉLÉMENTS – BEFORE/AFTER

- before/after peuvent avoir trois usages, selon la valeur que l'on donne à content.
- Exemple :

```
<span class="keyword">CSS</span>
```

La bulle blanche

Positionnée en **absolute**
par rapport à son parent

Décalage à droite,
centrage vertical

```
.keyword::before {  
    content: "";  
    position: absolute;  
    right: -15px;  
    top: calc(50% - 5px);  
    background-color: white;  
    width: 10px;  
    height: 10px;  
    border-radius: 10px;  
    z-index: 1;  
    box-shadow: -1px 1px 2px gray inset;  
}
```

Hauteur = largeur = radius
→ un rond !

z-index car c'est le before,
mais on veut qu'il passe
devant l'after

inset signifie « ombre à l'intérieur »



CONCEPTS AVANCÉS

PSEUDO-ÉLÉMENTS – BEFORE/AFTER

- before/after peuvent avoir trois usages, selon la valeur que l'on donne à content.
- Exemple :

```
<span class="keyword">CSS</span>
```

Le triangle rouge

Positionnée en absolute par rapport à son parent

Décalage à droite, commence en haut de son parent

Largeur à 0, hauteur à 0 : seule la bordure est visible

Bordure gauche rouge, bordure haut et bas blanche

```
.keyword::after {  
    content: "";  
    position: absolute;  
    right: -30px;  
    top: 0;  
    border-left: 30px solid brown;  
    border-top: 23px solid white;  
    border-bottom: 23px solid white;  
}
```



Des questions ?



CONCEPTS AVANCÉS

POLICES

- On l'a vu : `font-family` permet d'utiliser toute police présente sur le poste de l'utilisateur.

```
.my-block {  
    font-family: Economica, Georgia, serif;  
}
```

- Sauf qu'Economica, personne ne l'a, et on sait que l'affichage sera dégradé sur 99% des postes.
- **Solution 1** : n'utiliser que des polices génériques que tout le monde a.
Mais un site avec de l'Arial partout, c'est un peu triste...
- **Solution 2** : intégrer des polices via CSS !

CONCEPTS AVANCÉS

POLICES - INTÉGRATION

- Pour intégrer une nouvelle police, on utilise @font-face.

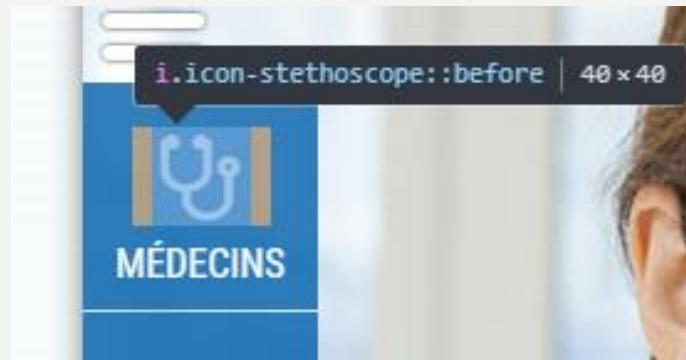
```
@font-face {  
    font-family: "Bitstream Vera Serif Bold";  
    src: url("/static/styles/libs/font-awesome/fonts/fontawesome-webfont.fdf491ce5ff5.woff");  
}  
  
body {  
    font-family: "Bitstream Vera Serif Bold", serif;  
}
```

- Rappel bonne pratique :
N'utiliser les guillemets que s'il y a des espaces dans le nom.

CONCEPTS AVANCÉS

POLICES - ICÔNES

- Les polices servent à améliorer l'expérience utilisateur... mais aussi à afficher des icônes !
- Plusieurs polices contenant des centaines d'icônes existent :
 - Fontello
 - FontAwesome
 - ...
- Elles fonctionnent à peu près de la même façon :



```
▼<a href="https://www.chu-rouen.fr/patients-public/nos-medecins/">
  ▼<i class="icon-stethoscope">
    ::before
  </i>
  <span>Médecins</span>
</a>
```

```
.fontello-zone .icon-stethoscope::before { font-family: "fontchurouen"; font-style: normal; font-weight: normal; speak: none; display: inline-block; text-decoration: inherit; content: '\e808'; }
.fontello-zone [class^="icon-"]::before, [class*=" icon-"]::before { font-family: "fontchurouen"; font-style: normal; font-weight: normal; speak: none; display: inline-block; text-decoration: inherit; }
```

CONCEPTS AVANCÉS

POLICES - ICÔNES

Image

VS

Police

16M Couleur

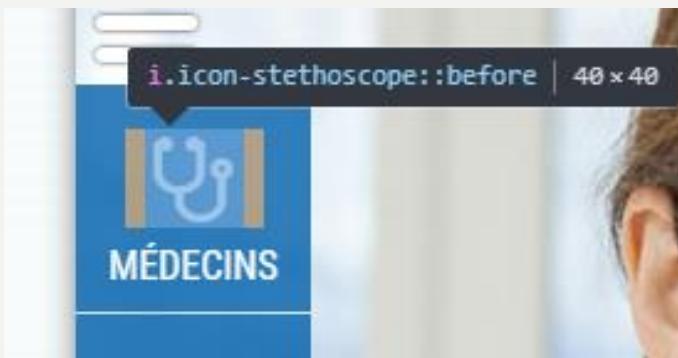
Couleur unique
choisie avec « color »

Taille fixe
déformable

Vectorielle
choisie avec « font-size »

Mise en cache si en background

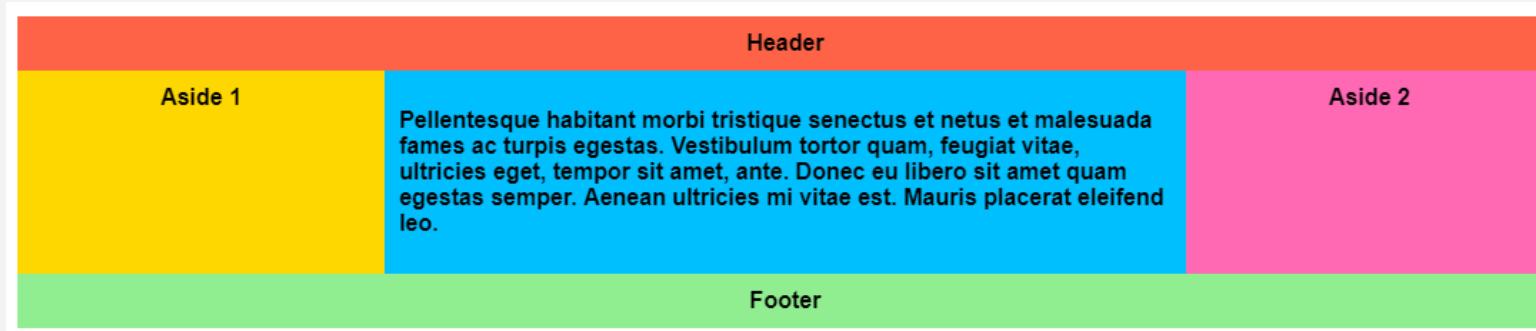
Mise en cache



CONCEPTS AVANCÉS

LAYOUT

- Constat : toutes les pages web se ressemblent...



- Avec CSS3 (oui, on en n'a pas parlé jusqu'ici, mais on en est à CSS3), de nouvelles techniques de layout sont donc apparues pour faciliter l'agencement des éléments.

CONCEPTS AVANCÉS

LAYOUT

- Parmi elles :
 - Les **flexbox** : <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
 - De plus en plus utilisées, car très pratiques et désormais supportées par tous les navigateurs
 - Malheureusement pas toujours maîtrisées, car fonctionnement un peu complexe
 - Les **grid** : <https://css-tricks.com/snippets/css/complete-guide-grid/>
 - Très peu utilisées pour l'instant
 - Fonctionnement très intuitif et supporté par tous les navigateurs

CONCEPTS AVANCÉS

RESPONSIVE DESIGN

- Le CSS a évidemment la responsabilité du responsive design.



CONCEPTS AVANCÉS

RESPONSIVE DESIGN

- L'idéal, pour s'adapter à toutes les tailles d'écrans, est d'utiliser des **largeurs en %** chaque fois que c'est possible.
- Pour faire du design spécifique à une taille d'écran, on utilise des *media query*.
 - > Les *media query* peuvent aussi servir à faire du design spécifique à l'impression

```
@media screen and (max-width: 640px) {  
    .block {  
        height: 120px;  
        font-size: 1.2em;  
    }  
}
```

```
@media print {  
    .block {  
        height: 120px;  
        font-size: 1.2em;  
    }  
}
```

CONCEPTS AVANCÉS

RESPONSIVE DESIGN – MOBILE FIRST

- Pour le développement de sites web « responsive », on choisit parfois la méthode du mobile-first.

Méthode
dégueu

style.css

```
.my-block {  
    width: 50%;  
}  
  
@media screen and (max-width: 320px) {  
    .my-block {  
        width: 100%;  
    }  
}
```

Méthode
naïve

style.css

```
.my-block {  
    width: 50%;  
}  
  
@media screen and (max-width: 320px) {  
    .my-block {  
        width: 100%;  
    }  
}
```

Mobile-first !

style.css

```
.my-block {  
    width: 100%;  
}  
  
@media screen and (min-width: 320px) {  
    .my-block {  
        width: 50%;  
    }  
}
```

CONCEPTS AVANCÉS

RESPONSIVE DESIGN – MOBILE FIRST

- Pour le développement de sites web « responsive »,
on choisit parfois la méthode du mobile-first.

Mobile-first !

style.css

```
.my-block {  
    width: 100%;  
}
```

style-large.css

```
@media screen and (min-width: 320px) {  
    .my-block {  
        width: 50%;  
    }  
}
```

Ainsi, le moteur CSS :

- Calcule d'abord le style mobile
- Calcule le style PC si nécessaire

CONCEPTS AVANCÉS

POLICES, LAYOUT, RESPONSIVE - MÉMO

- A retenir :
 - On installe les polices avec `@font-face`.
 - Pour un jeu d'icônes, on préfère utiliser une police et la balise `<i>`.
 - Le développement CSS repose beaucoup sur des structures CSS3 comme les **flexbox**, ou plus sporadiquement les **grid**.
 - Le **mobile-first** permet d'économiser les ressources des appareils mobiles.





MASTERING CSS

POUR SAUVER LE MONDE
AVEC CLASSE ET PANACHE

MASTERING CSS

INTRODUCTION

- Quelques petites choses utiles au quotidien :

- Points de suspension

Ceci est un texte beaucoup trop long pour être affiché dans une fenêtre.

- Tableau scrollable

Column 1	Column 2	Column 3	Column 4
Column 1 Content	Column 2 Content	Column 3 Content	Column 4 Content
Column 1 Content	Column 2 Content	Column 3 Content	Column 4 Content
Column 1 Content	Column 2 Content	Column 3 Content	Column 4 Content
Column 1 Content	Column 2 Content	Column 3 Content	Column 4 Content
Column 1 Content	Column 2 Content	Column 3 Content	Column 4 Content
Column 1 Content	Column 2 Content	Column 3 Content	Column 4 Content

MASTERING CSS

POINTS DE SUSPENSION

- Points de suspension

The screenshot shows a code editor interface with two panels: HTML and CSS.

HTML Panel:

```
1 <span class="keyword">Ceci est un texte beaucoup trop long  
pour une étiquette</span>
```

CSS Panel:

```
36 .keyword {  
37   max-width: 500px;  
38 }  
39  
40  
41
```

A large red arrow points from the left towards the text "Ceci est un texte beaucoup trop long pour une étiquette". Below the arrow, the text "895 pixels" is displayed in yellow, indicating the width of the text content.

- Malgré le `max-width`, l'étiquette prend la largeur de son contenu car `span` est un `inline` par défaut.

MASTERING CSS

POINTS DE SUSPENSION

- Points de suspension

```
HTML
1 <span class="keyword">Ceci est un texte beaucoup trop long
    pour une étiquette</span>
```

```
CSS
36 .keyword {
37     display: block;
38     max-width: 500px;
39 }
```

Ceci est un texte beaucoup trop long pour une étiquette

500 pixels

- Avec la largeur fixe, le texte va irrémédiablement à la ligne.
On utilise donc, comme vu précédemment, white-space: nowrap.

MASTERING CSS

POINTS DE SUSPENSION

- Points de suspension



```
1 <span class="keyword">Ceci est un texte beaucoup trop long  
pour une étiquette</span>
```

```
36 .keyword {  
37   display: block;  
38   max-width: 500px;  
39   white-space: nowrap;  
40 }
```

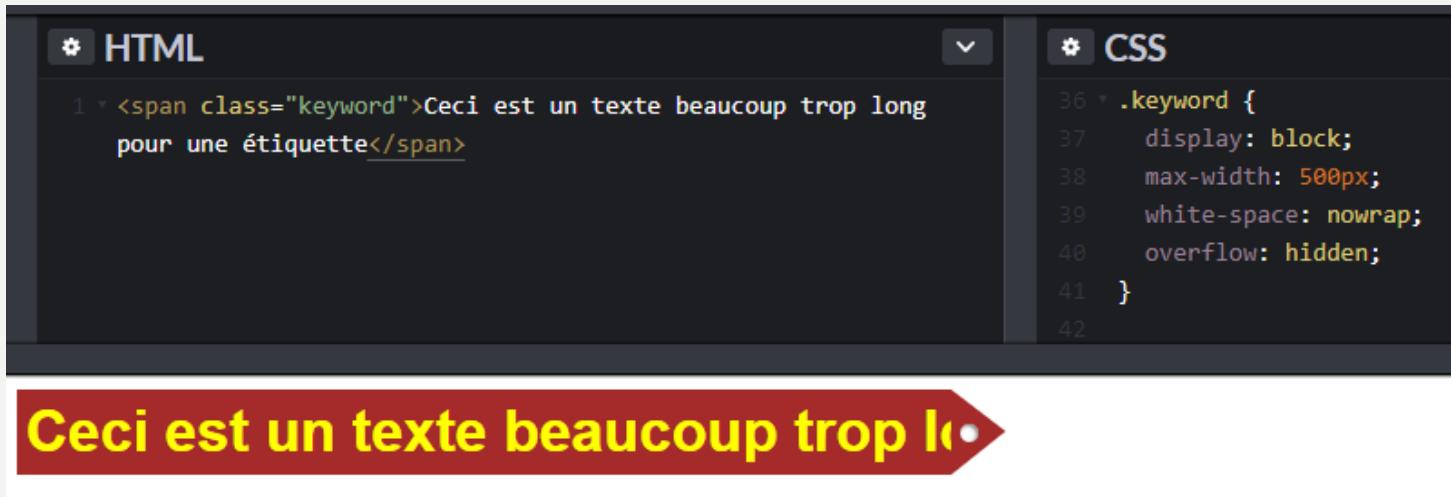
Ceci est un texte beaucoup trop long pour une étiquette

- Ok, on est bons en largeur et en hauteur... mais ça déborde.
On utilise donc overflow: hidden pour cacher ce qui dépasse.

MASTERING CSS

POINTS DE SUSPENSION

- Points de suspension



The screenshot shows a code editor interface with two panels: HTML on the left and CSS on the right.

HTML Panel:

```
1 <span class="keyword">Ceci est un texte beaucoup trop long  
pour une étiquette</span>
```

CSS Panel:

```
36 .keyword {  
37   display: block;  
38   max-width: 500px;  
39   white-space: nowrap;  
40   overflow: hidden;  
41 }  
42
```

Below the code editor, there is a red arrow pointing to the right containing the text: "Ceci est un texte beaucoup trop lo".

- Et voilà, manque plus que les points de suspension.
Pour ce faire, on utilise la propriété `text-overflow` avec la valeur `ellipsis`.

MASTERING CSS

POINTS DE SUSPENSION

- Points de suspension

The screenshot shows a code editor interface with two panes. The left pane is labeled "HTML" and contains the following code:

```
1 <span class="keyword">Ceci est un texte beaucoup trop long  
pour une étiquette</span>
```

The right pane is labeled "CSS" and contains the following code:

```
36 .keyword {  
37   display: block;  
38   max-width: 500px;  
39   white-space: nowrap;  
40   overflow: hidden;  
41   text-overflow: ellipsis;  
42 }
```

Below the code editor, a red arrow points to the right, containing the truncated text "Ceci est un texte beaucoup tro...".

- La triade à retenir pour faire des points de suspension :
 - `white-space: nowrap;`
 - `overflow: hidden;`
 - `text-overflow: ellipsis;`



MASTERING CSS TABLEAUX SCROLLABLES



MASTERING CSS TABLEAUX SCROLLABLES

→ Une méthode consiste à faire de l'en-tête et du corps deux blocs distincts.

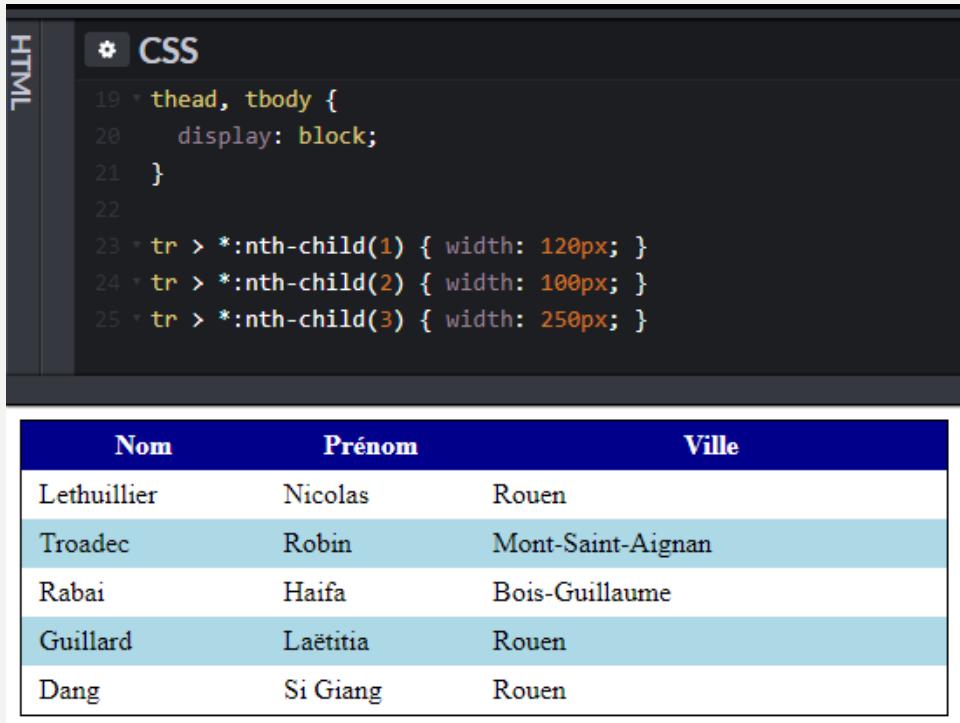
```
thead {  
    display: block;  
}  
  
tbody {  
    display: block;  
}
```

```
<table>  
  <thead>  
    <tr>  
      <th>Nom</th>  
      <th>Prénom</th>  
      <th>Ville</th>  
    </tr>  
  </thead>  
  <tbody>  
    <tr>  
      <td>Lethuillier</td>  
      <td>Nicolas</td>  
      <td>Rouen</td>  
    </tr>  
    <tr>  
      <td>Troadec</td>  
      <td>Robin</td>  
      <td>Mont-Saint-Aignan</td>  
    </tr>  
    <tr>  
      <td>Rabai</td>  
      <td>Haifa</td>  
    </tr>  
  </tbody>  
</table>
```

Nom	Prénom	Ville
Lethuillier	Nicolas	Rouen
Troadec	Robin	Mont-Saint-Aignan
Rabai	Haifa	Bois-Guillaume
Guillard	Laëtitia	Rouen
Dang	Si Giang	Rouen

- De l'intérêt de bien structurer ses données

MASTERING CSS TABLEAUX SCROLLABLES



The screenshot shows a code editor with two tabs: 'HTML' and 'CSS'. The 'CSS' tab is active, displaying the following code:

```
19 * thead, tbody {  
20   display: block;  
21 }  
22  
23 tr > *:nth-child(1) { width: 120px; }  
24 tr > *:nth-child(2) { width: 100px; }  
25 tr > *:nth-child(3) { width: 250px; }
```

The 'HTML' tab shows the generated HTML code for a table:

Nom	Prénom	Ville
Lethuillier	Nicolas	Rouen
Troadec	Robin	Mont-Saint-Aignan
Rabai	Haifa	Bois-Guillaume
Guillard	Laëtitia	Rouen
Dang	Si Giang	Rouen

Et là, on commence à tout péter...

On doit donc fixer une largeur à chaque colonne.

MASTERING CSS TABLEAUX SCROLLABLES

The screenshot shows a code editor interface with two tabs: 'HTML' and 'CSS'. The 'CSS' tab is active, displaying the following code:

```
19 *:nth-child(1) { width: 120px; }
20 *:nth-child(2) { width: 100px; }
21 *:nth-child(3) { width: 250px; }
22
23 thead, tbody {
24   display: block;
25 }
26
27 tbody {
28   max-height: 100px;
29   overflow-y: auto;
30 }
```

The 'HTML' tab shows the generated HTML code for a table:

Nom	Prénom	Ville
Lethuillier	Nicolas	Rouen
Troadec	Robin	Mont-Saint-Aignan
Rabai	Haifa	Bois-Guillaume
Guillard	Laëtitia	Rouen

Maintenant, mettons en place le système de scroll, en commençant par fixer une hauteur maximum.

Et enfin la mécanique de scroll à proprement parler.

Cette méthode a toutefois de nombreux défauts. D'autres méthodes, plus tordues, existent, et on espère que le W3C y remédiera vite.

Gardez tout de même en tête que du scroll dans une page scrollable, c'est une aberration en matière de web design...





POUR ALLER PLUS LOIN

COMPATIBILITÉ DES NAVIGATEURS,
PRÉPROCESSEURS, CONVENTIONS

POUR ALLER PLUS LOIN

COMPATIBILITÉ DES NAVIGATEURS

- Certains logiciels, que leurs éditeurs appellent à tort navigateurs, ont un peu de mal à suivre les évolutions de CSS.



Safari is the new IE

POUR ALLER PLUS LOIN COMPATIBILITÉ DES NAVIGATEURS

- Ils ont alors créé des préfixes « vendeurs » (vendor prefixes)

CSS préfixes ↗

En général, les principaux navigateurs utilisent ces préfixes :



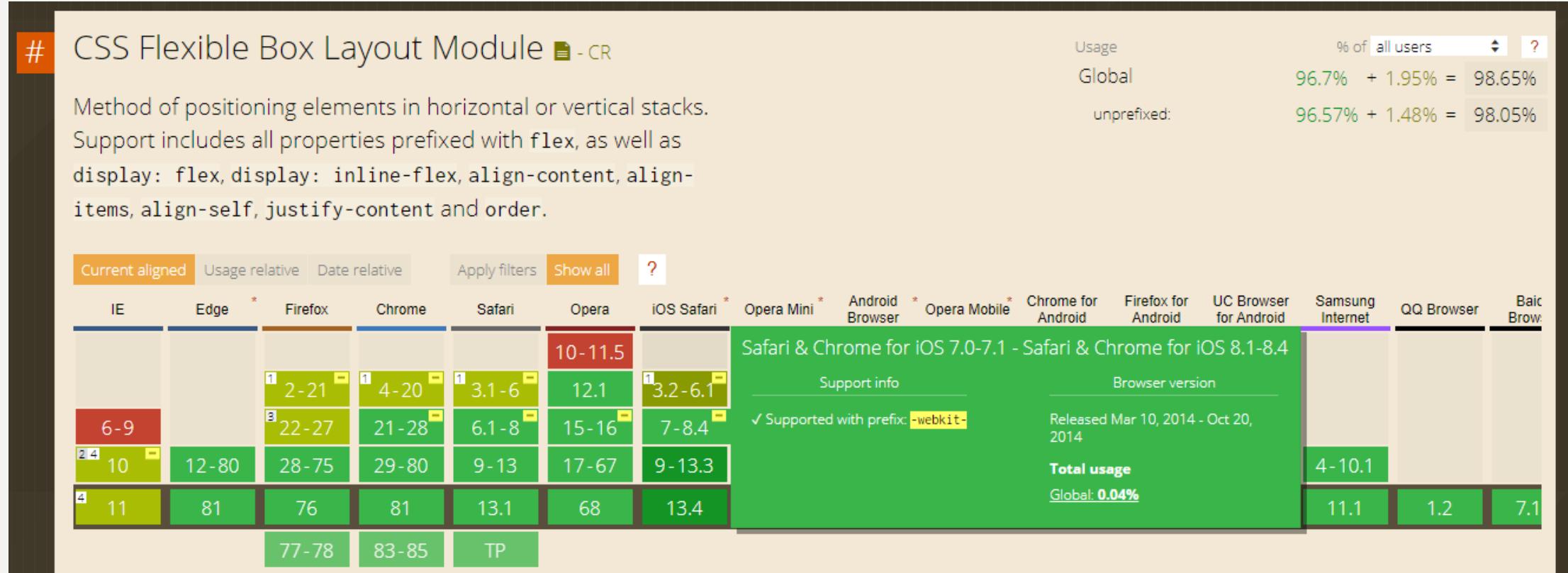
- `-webkit-` (Chrome, nouvelles versions d'Opera.)
- `-moz-` (Firefox)
- `-o-` (Anciennes versions d'Opera)
- `-ms-` (Internet Explorer et Edge)

- Attention à ne pas trop en mettre :
 - Depuis IE9, les `-ms-` sont rares.
 - Seul Safari mobile continuait de ramer un peu et requérait des `-webkit-`, notamment pour les flexboxes.

POUR ALLER PLUS LOIN

COMPATIBILITÉ DES NAVIGATEURS

- Dans le doute, consulter la Bible du développeur CSS : [Can I use...](#)



POUR ALLER PLUS LOIN

PRÉPROCESSEURS

- Pour simplifier le code CSS, on a inventé des préprocesseurs, qui ajoutent à CSS des syntaxes simplifiantes, et qui sont compilés pour générer des fichiers CSS.

- Par exemple :

Sass

```
#header {  
    display: block;  
    color: #ae67f6;  
    background-color: #121212;  
}  
  
#header > div {  
    width: calc(50% + 12px);  
    background-color: #ae67f6;  
    display: flex;  
    display: -webkit-flex;  
}
```



```
$primary-color: #ae67f6;  
$secondary-color: #121212;  
  
#header {  
    display: block;  
    color: $primary-color;  
    background-color: $secondary-color;  
  
    > div {  
        width: 50% + 12px;  
        background-color: $primary-color;  
        display: flex;  
    }  
}
```

POUR ALLER PLUS LOIN

CONVENTIONS

- Rappel slide 8 : « *On peut arriver à ses fins en faisant n'importe quoi* ».

→ Des conventions (pour les noms de classes) commencent à poindre, mais sont encore peu utilisées.

Exemples :

- OOCSS
- BEM

- `block-name`
- `block-name_modifier_name`
- `block-name__element-name`
- `block-name__element-name_modifier_name`

- A vous de voir si vous êtes convaincus, moi, j'attends de voir.
→ Donc pour l'instant, 0 convention.



RÉFÉRENCES

QUELQUES URL INTÉRESSANTES

RÉFÉRENCES

- WordPress best practices : <https://make.wordpress.org/core/handbook/best-practices/coding-standards/css/>
- Guide to flexbox : <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- Guide to grid : <https://css-tricks.com/snippets/css/complete-guide-grid/>
- Can I use... : <https://caniuse.com/>
- OOCSS & BEM : <https://www.alsacreations.com/article/lire/1641-Bonnes-pratiques-en-CSS--BEM-et-OOCSS.html>
- CSS-tricks : <https://css-tricks.com/>