Développement d’applications web

Vincent Séguin

Qui suis je?

Ingénieur logiciel

Diplômé de l'Université Laval

Vincent Séguin

Senior Software Engineer

Rakuten Ready (Curbside)

Pourquoi ce cours?

Passion du web

Aucun cours de web à l'université

(Maintenant oui!)

Web omniprésent

Sites web, applications web, applications desktop, mobiles, cloud

Navigateurs de plus en plus puissants

Pourquoi ce cours?

But poursuivis

Approfondir les notions reliées au Web, de la base jusqu'aux tendances 'on the edge'

Travailler sur un projet concret, vous faire expérimenter par vous-mêmes!

Faire de vous des développeurs polyvalents et prêts à affronter le marché du travail.

“ATTENTION!

Ce cours explique les notions de base du développement web, mais demandera tout de même un peu de **travail!**

I Am Devloper
@iamdevloper

Following

Things to try when fixing a bug:

1. Google
2. Stack Overflow
3. Documentation

...

8277. Disturb your co-worker who has headphones in

RETWEETS 3,952 LIKES 1,566

1:36 AM - 1 Feb 2014

4K 1.6K

Format du cours

~3h de cours par semaine

2 heures de labo

(Les labos comptent!)

Labo = Appliquer la théorie

Pondération

2 Examens

20% chaque

Projet de session

45%

Laboratoires

10%

10 labs

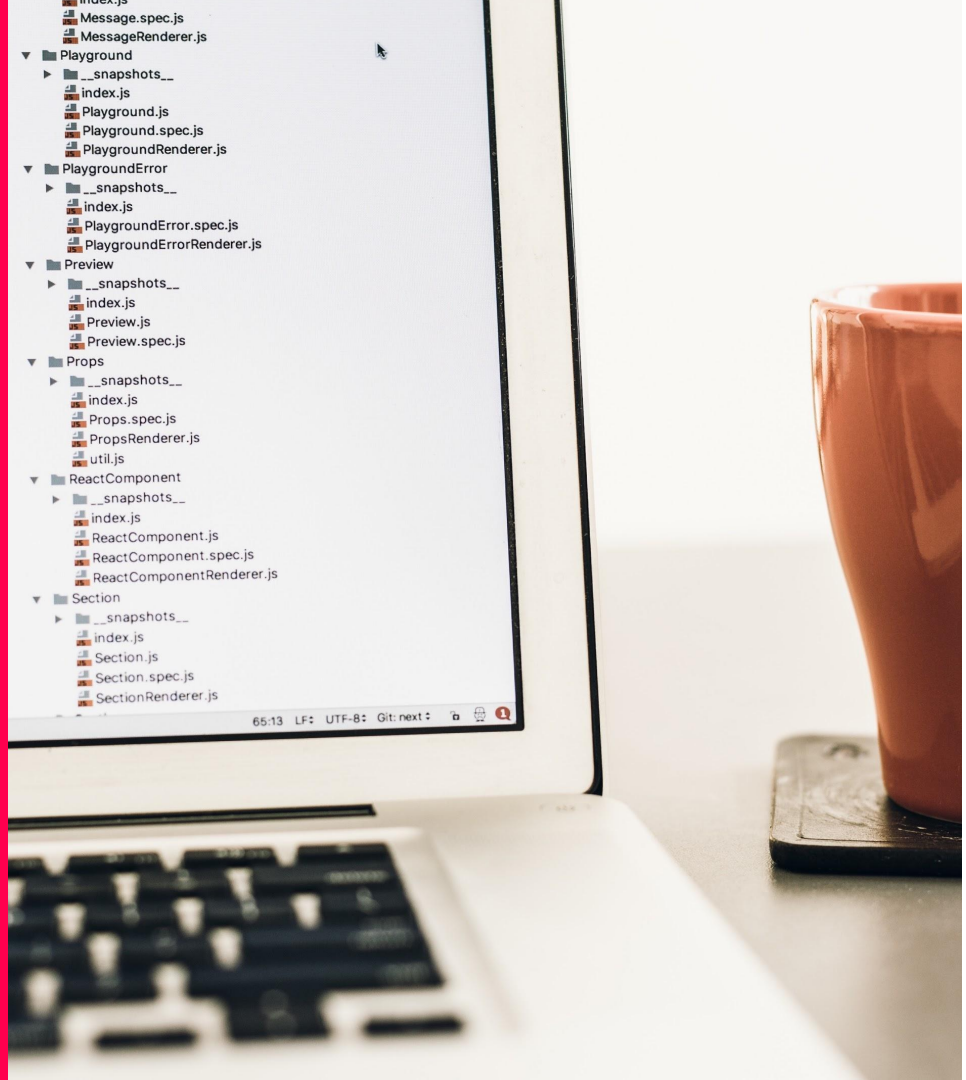
1% chaque

Évaluation des pairs

5%

Possibilité de perdre jusqu'à **4 cotes**

Projet de session



Outils

Pratique pour développer

- WebStorm (obtenez la version gratuite en tant qu'étudiant)
- Visual Studio Code

Pratique pour déboguer

- Console Chrome Dev Tools (voir la documentation pour plusieurs astuces)
- JsFiddle
- Mozilla Developer Network (contient toute la documentation sur JS, CSS, HTML)
- Postman/Insomnia
- VueJS DevTools

Chapitre 1

Les rudiments

HTML et CSS

Nous ferons un survol rapide du **HTML** et **CSS** car nous considérons ces concepts comme étant simples à apprendre de façon autonome.

Utilisez la documentation sur MDN pour obtenir tous les détails sur ceux-ci.

HTML

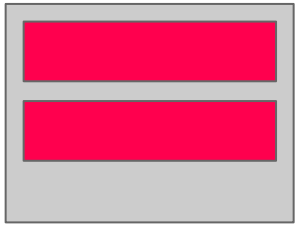
À savoir

Bien connaître les balises d'organisation

** **

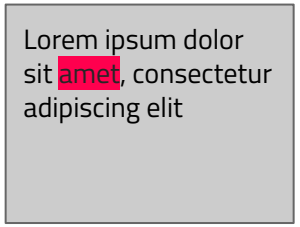
Les div sont les éléments les plus courants d'une page web. Ils représentent un simple conteneur.

Un div est display:block (implique un saut de ligne) par défaut.



****

Les span ressemblent aux div, mais sont en display:inline (pas de saut de ligne).

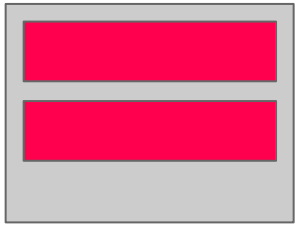


HTML

À savoir

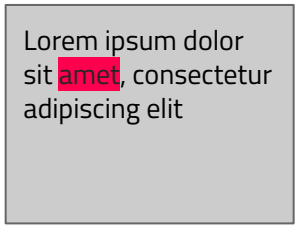
Éléments “block”

```
<p> <div> <form> <header> <nav> <ul> <li> <h1>
...
```



Éléments “inline”

```
<a> <span> <b> <em> <i> <cite> <mark> <code>
...
```



HTML

À savoir

Bien connaître les balises d'organisation

<p/>

Les **p** sont des balises très semblables aux div, exceptées qu'elles sont faites pour contenir du texte.

HTML valide :

```
<div>
   <p>Hel<span>lo</span></p>
</div>
```

HTML

À savoir

Bien connaître les balises d'organisation

 et

Balises qui devraient exclusivement servir pour des listes.
Par défaut, une liste HTML possède des puces.

<table/>, <td/> et <tr/>

Balises qui devraient exclusivement servir pour des tableaux.

Les balises **spécialisées** ont des utilisations **spécialisées!**.
Les tableaux/listes sont plus difficiles à utiliser, donc ne pas en abuser...

HTML

À savoir

Bien connaître les balises de mise en forme

, <b/>, <i/>

Balises de mise en forme de texte. Peut également se faire grâce au CSS.

<address/>, <date/>, <phone/>

Balises très spécifiques de mise en forme possédant déjà du style (et des interactions).

HTML

À savoir

Bien connaître les balises de formulaire

<form>

Définit une zone de formulaire. Un form possède le comportement par défaut d'envoyer une requête **POST**…

<input type="text" name="fname">
<input type="date" name="fname">
<input type="submit" value="Submit">

Définit les types de champ d'un formulaire, ainsi que le type d'action possible. Nous y reviendrons…

HTML

À savoir

Autres balises

Balises multimédia : <img/>, <audio/>, <video/>

Balises de titre : <h1/>, <h2/> … <h6/>

Bouton : <button/>

Hyperlien : <a href=””/>

etc… à vous de les découvrir!

helloworld.html

Document de base

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <title>My awesome page</title>
        <meta name="description" content="My awesome page">
        <meta name="viewport" content="width=device-width">

        <link rel="stylesheet" href="css/mycss.css">
        <script src="js/myscript.js"></script>
    </head>
    <body>
        <p>Hello world!</p>
    </body>
</html>
```

CSS

À savoir

Le CSS est principalement basé sur des sélecteurs, qui permettent de styliser les balises HTML.

se référencera par le sélecteur .myDiv en CSS

se référencera par le sélecteur #myDiv en CSS

CSS

À savoir

Le CSS est principalement basé sur des sélecteurs, qui permettent de styliser les balises HTML.

Une class devrait être utilisée sur des éléments qui vont se répéter dans une page.

Un id devrait être placé sur une balise unique.

Sélecteurs

Petit guide des sélecteurs

Stylise tous les éléments ayant la classe *myClass*

```
.foo {
    ...
}
```

Stylise tous les div!

```
div {
    ...
}
```

Stylise tous les éléments ayant la classe *bar* dont un parent a la classe *foo.*

```
.foo .bar {
    ...
}
```

Sélecteurs

Petit guide des sélecteurs

Stylise tous les éléments ayant la classe *foo* **ET** la classe *bar*

(notez l'absence d'espace entre les 2 classes)

```
.foo.bar {
    ...
}
```

Stylise tous les éléments ayant la classe *bar* dont le parent direct possède la classe *foo*

```
.foo > .bar {
    ...
}
```

Variations possibles avec classes, id et éléments directs!

```
.foo a
.foo #bar
etc...
```

Sélecteurs

Petit guide des pseudo-sélecteurs

Stylise les éléments avec la classe *myClass* lorsque la souris passe dessus

```
.myClass:hover {
    ...
}
```

Stylise les liens avec la classe *myClass*, lorsque visités

```
.myClass:visited {
    ...
}
```

Stylise le PREMIER élément avec la classe *myClass*.

```
.myClass:first-child(){
    ...
}
```

Sélecteurs

MDN web docs
moz://a

Voir : https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started/Selectors

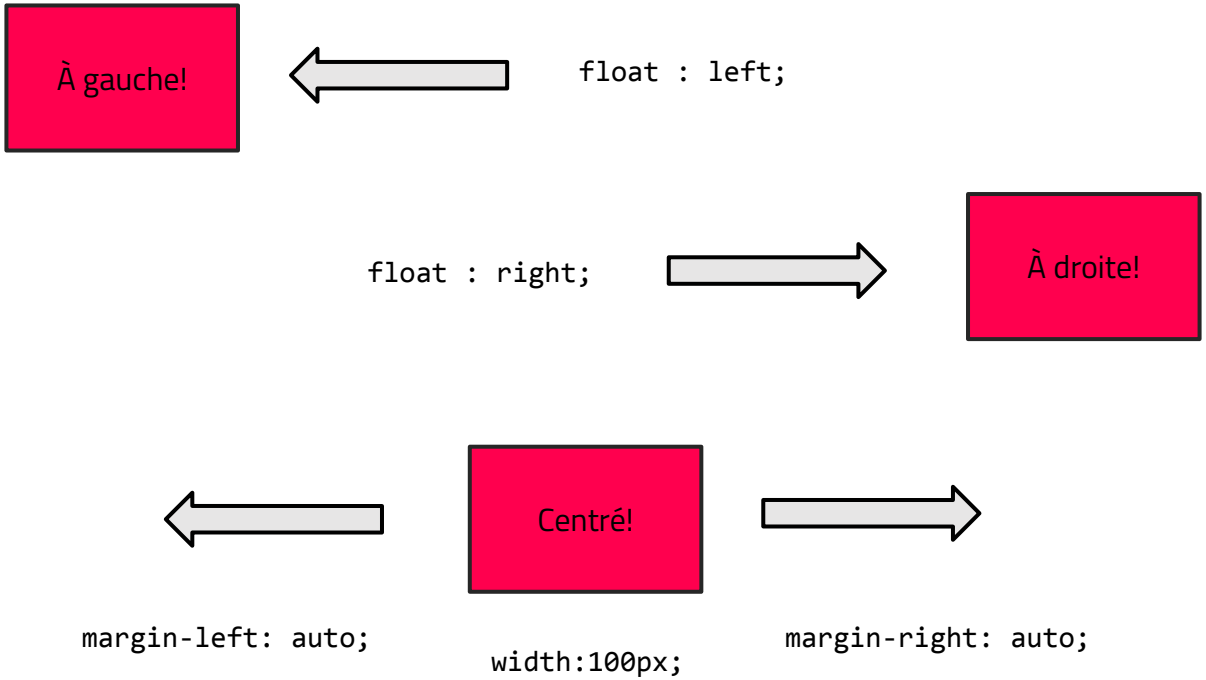
Qu’est-ce qu’on met dans un sélecteur

Tout ce qui est du STYLE!

```
.myClass {
    color : #666666; // Couleur du texte
    display: inline-block; // Saut de ligne ou
non
    float: left; // Gauche, droite
    background: url(‘mypicture.jpg’)
no-repeat; // Arrière-plan
    font-size: 14px; // Grosseur du texte
    ...
}
```

CSS

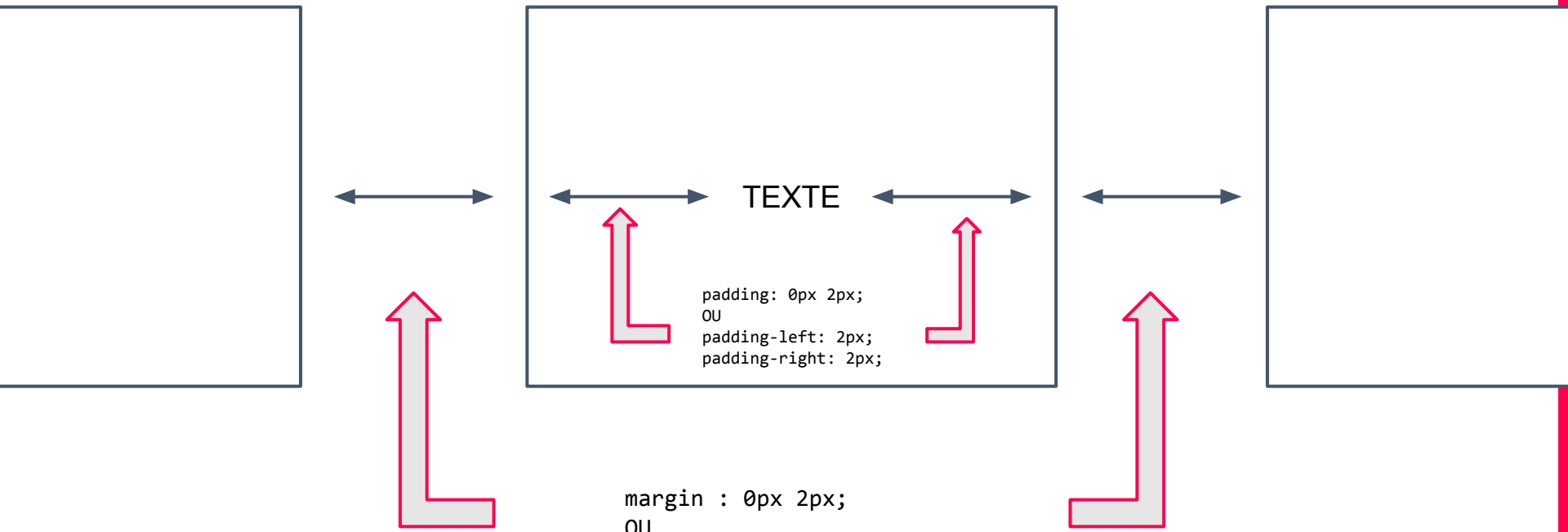
À savoir - positionnement



CSS

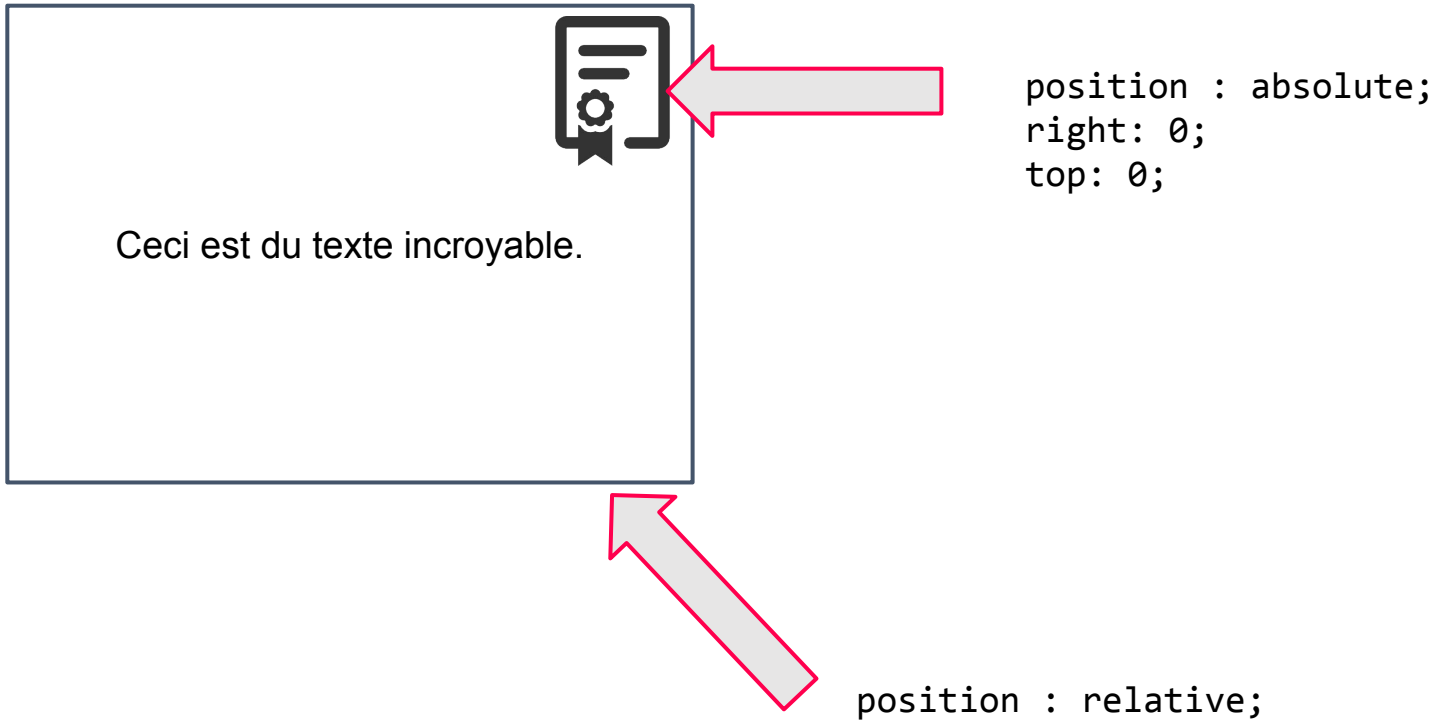
À savoir - espacement

(padding et margin)



CSS

À savoir - positionnement



CSS

Héritage et surcharge

Le CSS appliqué sur une seule balise peut être le résultat de plusieurs règles, appliquées en ordre de définition et de précision.

Ainsi, l'élément suivant : pourrait hériter des règles suivantes en ordre :

```
div {
    …
}
```

```
.class1 {
    …
}
```

```
.class1 {
    …
}
```

```
.class1:first-child {
    …
}
```

Le style **inline** est toujours appliqué en dernier.
Possible de forcer l'ordre en utilisant le tag `!important`, mais c'est souvent un mauvais signe…

Voir : https://developer.mozilla.org/en/docs/Web/CSS/Specificity

Sites web adaptatifs (Responsive)

Techniques

Le style responsive est de plus en plus important : il s'agit de supporter plusieurs résolutions d'écran avec une même feuille de style!

Tailles en %, rm, rem, vh, vw au lieu de pixels

Les width, padding, height, etc. peuvent s'exprimer en % au lieu de pixels fixes.

Max-width/min-width

Permet de définir des règles de CSS plus complètes qu'une simple largeur fixe.

```
.container {
  width: 50%;
  max-width: 1200px;
}
```

Overflow

hidden, auto, scroll

```
.container {
  overflow: hidden;
}
```

Media queries

Permet de surcharger le CSS existant pour des résolutions d'écran données.

Media Queries

Permet de surcharger le CSS existant pour des résolutions d'écran données.

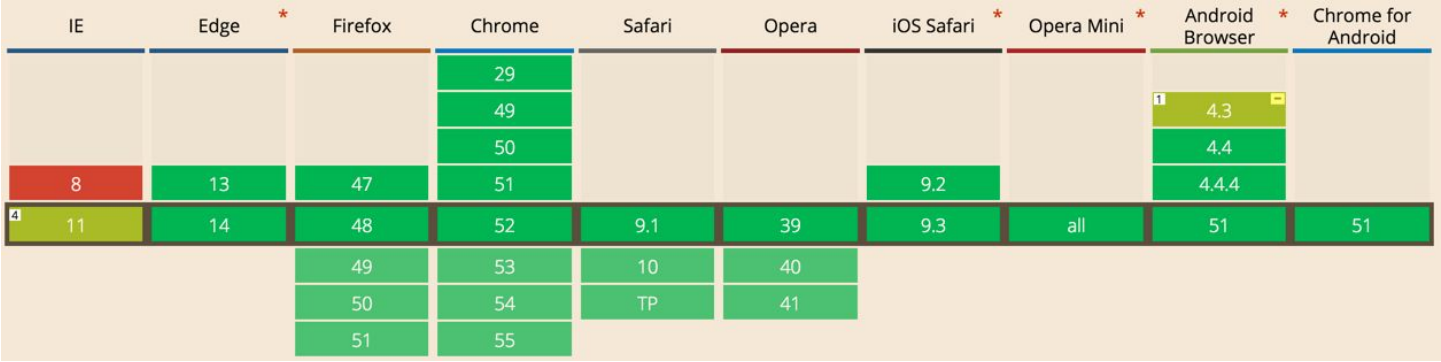
```
@media screen and (max-width: 640px) {
        .container {
                width: 600px;
        }
}
```

Lorsque l'écran aura une largeur en bas de 640px, les éléments avec la classe container auront une largeur différente!

CSS - Flexbox / Flex

Flex est maintenant bien supporté par la majorité des navigateurs modernes.

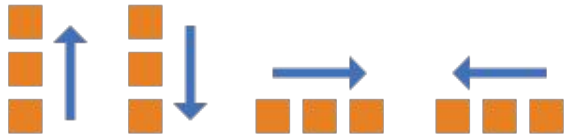
Flex permet de simplifier plusieurs affichages autrement **complexe**.



caniuse.com/#feat=flexbox

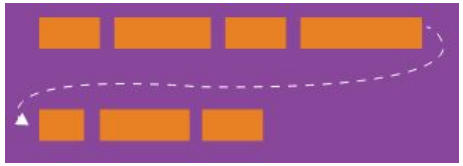
CSS - Élément parent

```
.container {
  display: flex; /* ou inline-flex */
}
```



```
.container {
  flex-direction: row | row-reverse | column |
                  column-reverse;
}
```

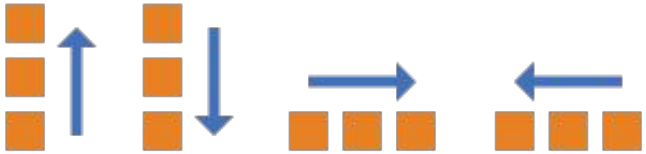
```
.container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

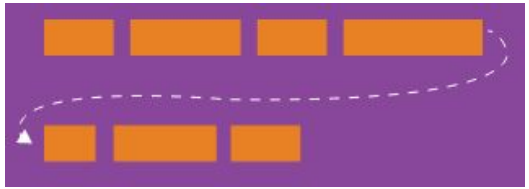


Source : https://css-tricks.com/snippets/css/a-guide-to-flexbox/

CSS - Élément parent

```
// Combinaison de flex-direction et flex-wrap
.container {
    flex-flow: <‘flex-direction’> || <‘flex-wrap’>
}
```

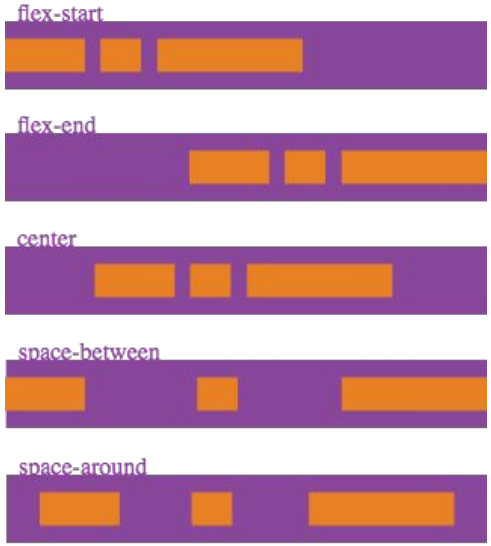




Source : https://css-tricks.com/snippets/css/a-guide-to-flexbox/

CSS - Alignement horizontal

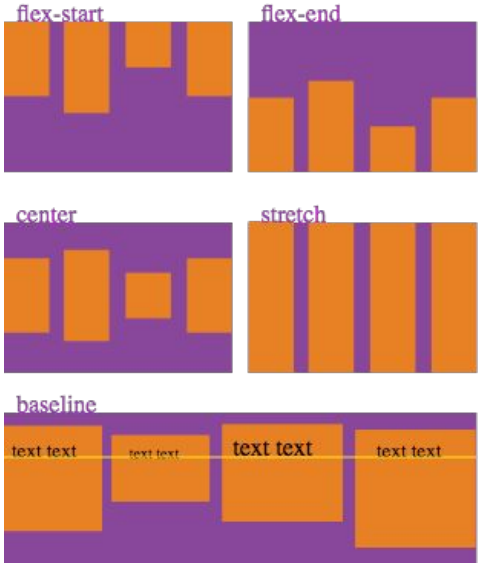
```
.container {
  justify-content: flex-start | flex-end |
                   center | space-between |
                   space-around;
}
```



Source : https://css-tricks.com/snippets/css/a-guide-to-flexbox/

CSS - Alignement vertical

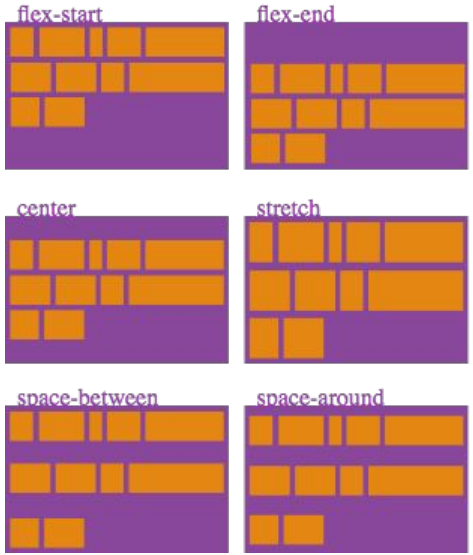
```
.container {
  align-items: flex-start | flex-end |
               center | baseline | stretch;
}
```



Source : https://css-tricks.com/snippets/css/a-guide-to-flexbox/

CSS - Alignement des enfants

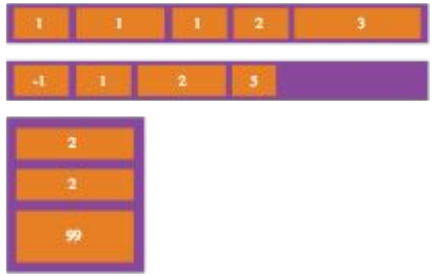
```
.container {
  align-content: flex-start | flex-end |
                 center | space-between |
                 space-around | stretch;
}
```



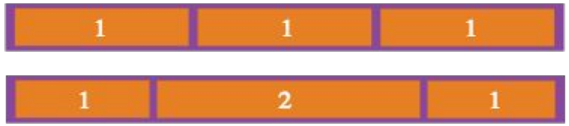
Source : https://css-tricks.com/snippets/css/a-guide-to-flexbox/

CSS - Éléments enfants

```
.item {
  order: <integer>;
}
```

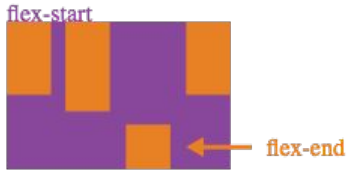



```
.item {
  flex-grow: <number>; /* default 0 */
}
```




```
.item {
  flex-shrink: <number>; /* default 1 */
}
```

```
.item {
  align-self: auto | flex-start | flex-end |
              center | baseline | stretch;
}
```



Source : https://css-tricks.com/snippets/css/a-guide-to-flexbox/

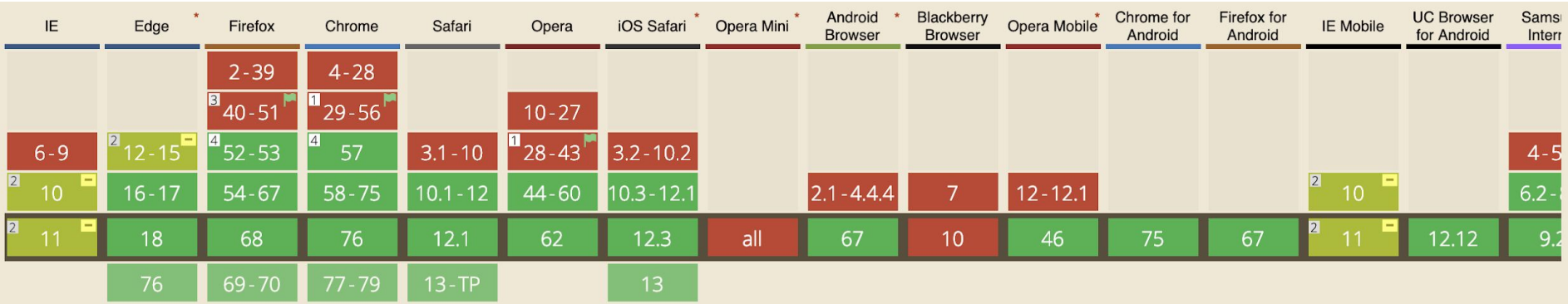
Ressources CSS Flex

- MDN (Flexible Box Layout)
- CSS Tricks (Complete guide to flexbox)
- Solved by Flexbox
    - Centrer verticalement
    - Footer “sticky”
- Flexbox froggy (Jeu interactif pour apprendre flex)

CSS - Grid

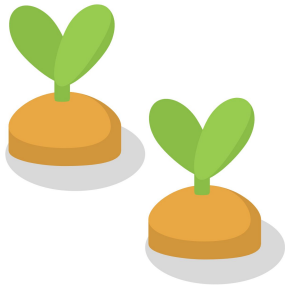
Le support de la syntaxe **Grid** permet de rapidement faire de l'affiche en **deux-dimensions** - versus flexbox qui focuse sur une seule dimension.

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout



Ressources CSS Grid

- MDN (Grid Layout)
- CSS Tricks (Complete guide to grid)
- Grid garden (Jeu interactif pour apprendre grid)



Autres fonctionnalités utiles...

- **Variables** (!)

```
--main-bg-color: blue;

background-color: var(--main-bg-color);
```

- **Supports** queries

```
@supports (display: flex) { ...
```

Frameworks

Bootstrap

https://getbootstrap.com/

Bulma

https://bulma.io/

Materialize

https://materializecss.com/

Semantic

https://semantic-ui.com/

À vous de choisir... lisez la documentation avant.

Focusez sur la simplicité!