Secteur Tertiaire Informatique

Filière « Etude et développement »

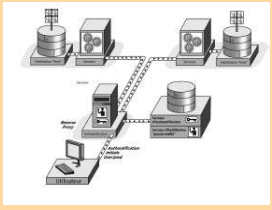
CSS

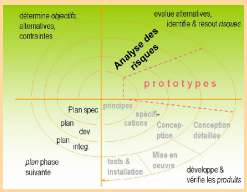
# Flexbox

**Apprentissage**

Mise en pratique

Evaluation






| Version | Date | Auteur(s) | Action(s) |
|---|---|---|---|
| 1.0 | 25/10/19 | Alexis Rouyer | Création du document |

# TABLE DES MATIERES

## Objectifs

Découverte des Flexbox en CSS

## Pré requis

## Outils de développement

## Méthodologie

## Mode d'emploi

Symboles utilisés :

 Renvoie à des supports de cours, des livres ou à la documentation en ligne constructeur.

 Propose des exercices ou des mises en situation pratiques.

 Point important qui mérite d'être souligné!

## Ressources

## Lectures conseillées

# 1. PREAMBULE

Dans un contexte où il devient primordial d'accéder aux informations sur différents types de supports, un enjeu majeur est de rendre les pages web accessibles sur l'ensemble des outils à notre disposition.

La duplication des sites pour l'adaptation aux différents modèles n'a plus cours. S'il était répandu d'avoir plusieurs versions des pages en fonction des résolutions d'écran, l'apparition des modèles portables (Smartphones, tablettes) a chamboulé le visage de la mise en page.

Les ressources déjà à disposition étaient trop limitées (block / inline / float / dimensions exprimées en pixel ou pourcentage etc.). Les Flexbox ont alors vu le jour. Avec un panel varié que nous allons aborder, elles permettent une grande flexibilité de présentation. Attention cependant, si les problèmes de compatibilité de navigateurs ont tendance à se résorber, nous ne sommes encore pas à l'abri de comportements parfois spectaculaires ou inexistants.

## 2. ORIENTATION

La notion d'horizontalité et de verticalité perd son sens avec les Flexbox. Il vous appartient de définir votre axe.

Si par défaut votre axe est horizontal, de gauche à droite, vous pouvez le définir à la verticale, de droite à gauche, de bas en haut. On parlera d'axe principal et d'axe secondaire.

L'orientation donnée dans le container vaut pour l'ensemble de ses enfants. S'il sera possible d'orienter des enfants par rapport à d'autres, ils ne pourront pas aller à l'encontre du parent. En revanche, vous pouvez imbriquer différents enfants pour des résultats dont les limites seront votre créativité. En effet, chaque enfant d'un container peut lui aussi devenir container de ses propres enfants.

Il est très important de ne jamais perdre de vue votre orientation. Il n'est pas forcément évident de travailler sur un axe différent de notre référentiel horizontal gauche-droite et il est facile de se perdre sans méthode et rigueur. Notez qu'un seul changement dans les instructions peut déstructurer votre page.

# 3. PROPRIETES PARENTES

## 3.1 DISPOSITION

Pour utiliser les propriétés Flexbox, commencez par indiquer un display: flex dans le style de votre container.

Par défaut, cette simple écriture mettra en ligne horizontale l'ensemble des éléments dans votre container. Pratique pour aligner 2  par exemple.

Un ensemble de propriétés peuvent être utilisées pour aligner et orienter vos éléments :

- flex-direction
- flex-wrap
- flex-flow

### 3.1.1 Flex-direction

Les différentes valeurs de flex-direction vont vous permettre de définir vos axes.

Par défaut, flex-direction : row ; signifie que vos éléments vont s'aligner de gauche à droite (il n'y a pas de notion de bas en haut, par défaut, c'est une « ligne », notez pour plus tard

flex-wrap : nowrap ;).

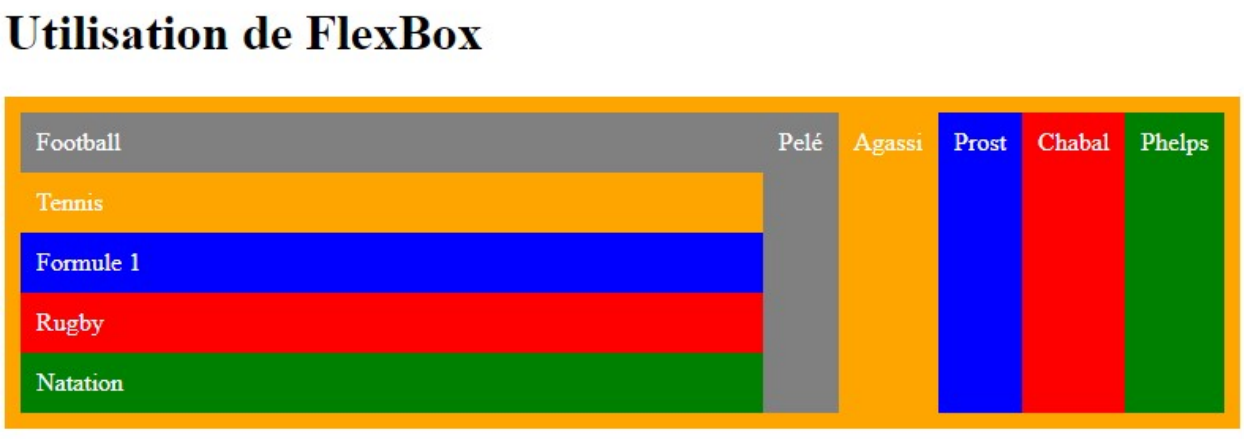
flex-direction : column ; signifie que vos éléments vont s'aligner de haut à bas. Là encore, il n'y a pas de notion de gauche à droite, chaque élément sera une ligne de la colonne.

flex-direction : row-reverse ; signifie que vos éléments vont s'aligner de droite à gauche.

flex-direction : column-reverse ; signifie que vos éléments vont s'aligner de bas en haut.

Exercice :

En utilisant Flexbox, avec des div ou des listes, ressortir le tableau suivant :



### 3.1.2 Flex-wrap

Les valeurs de flex-wrap permettent de gérer le comportement du contenu si la résolution est trop petite par rapport au contenu.

Par défaut, flex-wrap : nowrap ; signifie qu'il n'y a pas de modification d'affichage, l'écran fait apparaître des ascenseurs.

flex-wrap : wrap ; signifie que les éléments n'ayant plus la place se positionnent à la suite des informations visibles. Attention, la suite signifie à la suite de votre axe d'orientation !

flex-wrap : wrap-reverse ; signifie que les éléments n'ayant plus la place se positionnent avant les informations visibles. Attention, toujours en fonction de votre axe d'orientation !

Exercice :

Créer 15 div alignées horizontalement de 50x50px minimum et tester les 3 valeurs en changeant la taille de fenêtre de navigateur. Penser à numéroter chaque div pour mieux apprécier le résultat.

Question :

Est-ce pertinent un wrap sur une colonne ? Tester le.

### 3.1.3 Flex-flow

Si vous avez compris les 2 propriétés précédentes, flex-flow est celle à retenir : il s'agit d'un raccourci pour flex-direction et flex-wrap. Vous pouvez ainsi définir vos valeurs dans une seule ligne. Exemple pour une orientation horizontale en partant de la droite avec repositionnement en bas des éléments non visibles :

flex-flow : row-reverse wrap ;

## 3.2 L'ORGANISATION

### 3.2.1 Avant-propos

Comme on a pu le constater dans l'exercice de flex-direction, le contenu semble donner la taille de l'enfant dans le container. C'est le cas. Sans précision particulière, chaque élément prendra la place qui lui convient d'avoir pour être affiché entièrement. Dans tous les cas, si les informations à afficher ne prennent pas toute la place, les enfants se repartiront le reste de la place. Il est alors possible de mettre en forme le contenu tout comme leur disposition entre eux.

Les propriétés suivantes vous nous permettre de faire cette mise en forme :

- justify-content
- align-items
- align-content

### 3.2.2 Justify-content

Les différentes valeurs de justify-content vont permettre d'agencer les enfants du container en fonction de celui-ci mais également entre eux horizontalement.

justify-content : center ; centre les enfants dans le container

justify-content : flex-start ; les enfants sont au départ du container. Attention à l'axe !

justify-content : flex-end ; les enfants sont à la fin du container. Attention à l'axe !

justify-content : space-between ; les enfants se répartiront l'espace en laissant un espace égal entre chaque.

justify-content : space-around ; les enfants se répartiront l'espace en laissant un espace égal entre eux et les bords du container.

### 3.2.3 Align-items

Les différentes valeurs d'align-items vont permettre d'agencer les enfants du container en fonction de celui-ci verticalement.

align-items : center ; centre les enfants dans le container.

align-items : flex-start ; place les enfants en haut du container.

align-items : flex-end ; place les enfants en bas du container.

align-items : stretch ; par défaut, les enfants prennent toute la place dont ils disposent.

align-items : baseline ; Les enfants se placent symétriquement par rapport à une ligne définie dans chaque enfant via la propriété line-height : xx px ;

### 3.2.4 Align-content

Les valeurs d'align-content permettent d'agencer les enfants dans le container si celui-ci change, notamment lors des wraps. C'est donc très proche de justify-content mais les effets sont différents.

align-content : center ; centre les enfants dans le container

align-content : flex-start ; les enfants sont au départ du container. Attention à l'axe !

align-content : flex-end ; les enfants sont à la fin du container. Attention à l'axe !

align-content : space-between ; les enfants se répartiront l'espace en laissant un espace égal entre chaque.

align-content : space-around ; les enfants se répartiront l'espace en laissant un espace égale entre eux et les bords du container.

Exercice :

Tester les différences entre justify-content, align-items et align-content

# 4. PROPRIETES ENFANTES

S'il est possible de gérer l'ensemble du contenu via les containers, il est souvent nécessaire de gérer les contenus individuellement de part leurs natures différentes. Il existe donc des propriétés inhérentes aux enfants du container :

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

## 4.1 DISPOSITION

### 4.1.1 Order

Les valeurs d'order permettent de changer l'ordre des enfants dans le container. La balise order : X ; où X désigne la position dans chaque enfant permet de gérer cet ordre.

Exemple :

Code :

```
<div style="display: flex">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```

Résultat :

4213

## 4.2 DIMENSIONS

### 4.2.1 Flex-grow

Les valeurs de flex-grow permettent de d'agrandir certains objets par rapport aux autres. Par défaut, ils sont tous à 0 (ils ne grandissent pas). Flex-grow ne prend que des entiers.

Exemple :

Code :

```
<div style="display: flex">
  <div style="flex-grow: 1; border: solid; background-color: lightblue;">1</div>
  <div style="flex-grow: 1; border: solid; background-color: lightblue;">2</div>
  <div style="flex-grow: 7; border: solid; background-color: lightblue;">3</div>
</div>
```

Résultat :


Le div 3 grandira 7 fois plus que les div 1 et 2.

### 4.2.2 Flex-shrink

Les valeurs de flex-shrink permettent de donner un maximum de rétrécissement. Par défaut, la valeur est 1 (le contenant se réduit tant qu'il peut afficher le contenu). A 0, le contenant ne se réduit pas. Mettre différentes valeurs au dessus de 1 va donner la priorité de réduction. Le résultat peut être lu de cette manière : les éléments se réduisent de la valeur de shrink la plus grande à la plus petite. Pratique pour laisser apparaître un élément plus important que d'autres si l'écran se réduit.

Exercice :

Tester flex-shrink sur 10 div numérotées en ligne. L'ordre de réduction doit être div 8, div 5, div 7, div 2. La div 3 ne doit jamais réduire.

### 4.2.3 Flex-basis

Les valeurs de flex-basis permettent de donner une longueur de départ à un élément. Celui-ci s'exprimer en pixels.

#### 4.2.4 Flex

La propriété flex permet de gérer les propriétés de dimensions en une seule ligne. L'ordre de flex est flex-grow, flex-shrink et flex-basis. Exemple pour un élément sans agrandissement, avec rétrécissement de 100 pixels :

```
<div style="flex: 0 1 100px">X</div>
```

### 4.3 ALIGNEMENT

#### 4.3.1 Align-self

La propriété align-items du container parent permet de gérer l'alignement de l'ensemble des éléments enfants. Align-self permet de gérer cet alignement individuellement et vient écraser pour l'élément la valeur d'align-items. On y retrouve les mêmes valeurs avec les mêmes effets : center, flex-end, flex-start, stretch, baseline.

## 5. POUR ALLER PLUS LOIN

- https://bennettfeely.com/flexplorer/
- https://la-cascade.io/flexbox-guide-complet/

## CREDITS


### ŒUVRE COLLECTIVE DE l'AFPA
### Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

### Equipe de conception (IF, formateur, mediatiseur)
Alexis Rouyer- Formateur
Ch. Perrachon– Ingénieure de formation



**Date de mise à jour** : 25/09/2019