**EXPOSÉ FLEX BOX**

**SLIDE 1 : Debut**

**SLIDE 2 : Historically**

Flexbox Layout officiellement reconnu comme CSS Flexible Box Layout Module est un nouveau modèle de mise en page en CSS3.

La mise en page Flexbox est une mise en page à une seule dimension, ce qui signifie qu’elle dépose des éléments dans une dimension à la fois, soit en tant que ligne ou colonne, mais pas les deux ensembles. Cela peut être opposé au modèle à deux dimensions - CSS Grid Layout, qui place les éléments en deux dimensions simultanément (lignes et colonnes ensemble).

Flexbox a été introduit comme alternative à CSS Floats pour définir l’apparence générale d’une page Web. Flexbox a beaucoup d’avantages. En particulier, on peut contrôler l’alignement, la direction, l’ordre et la taille des objets.

Les objets dans le conteneur Flex:

* peuvent être organisés dans toutes les directions (à gauche, à droit, en bas ou même en haut) (flex-direction).
* peuvent avoir un ordre direct (de gauche à droite) ou inversé (de droite à gauche).
* peuvent avoir une largeur et une hauteur “flex” pour remplir l’espace disponible (flex-item).
* peuvent être définis pour réduire ou agrandir dynamiquement le long de l’axe principal, tout en préservant la taille de l’axe transversal secondaire.
* peuvent être organisés linéairement le long de l’axe principal ou enroulés en plusieurs lignes le long ou en travers de l’axe transversal.

Flexbox est relativement nouvelle, mais elle bénéficie d'une excellente prise en charge du navigateur aujourd'hui (près de 83% des navigateurs le prennent en charge).

(En savoir [plus](https://caniuse.com/#feat=flexbox)sur la prise en charge et la compatibilité du navigateur).

## Les Bases de Flexbox

Lorsque vous travaillez avec Flexbox, vous devez penser en deux axes — l’axe principal et l’axe transversal, et différencier les conteneurs flex et les éléments flex.

**SLIDE 2 : Introduction**

The Flexbox Layout (Flexible Box aims at providing a more efficient way to lay out, align and distribute space among items in a container, even when their size is unknown and/or dynamic.

The main idea behind the flex layout is to give the container the ability to alter its items’ width, height and order to best fill the available space (mostly to accommodate to all kind of display devices and screen sizes). A flex container expands items to fill available free space or shrinks them to prevent overflow.

Most importantly, the flexbox layout is direction-agnostic as opposed to the regular layouts (block which is vertically-based and inline which is horizontally-based). While those work well for pages, they lack flexibility to support large or complex applications (especially when it comes to orientation changing, resizing, stretching, shrinking, etc.).

Flexbox layout is most appropriate to the components of an application, and small-scale layouts, while the [Grid](https://css-tricks.com/snippets/css/complete-guide-grid/) layout is intended for larger scale layouts.

**SLIDE 3 : Basic**

Since flexbox is a whole module and not a single property, it involves a lot of things including its whole set of properties. Some of them are meant to be set on the container (parent element, known as “flex container”) whereas the others are meant to be set on the children (said “flex items”).

If “regular” layout is based on both block and inline flow directions, the flex layout is based on “flex-flow directions”.

Items will be laid out following either the main axis (from main-start to main-end) or the cross axis (from cross-start to cross-end).

Main axis – The main axis of a flex container is the primary axis along which flex items are laid out. Beware, it is not necessarily horizontal; it depends on the flex-direction property

**SLIDE 4 : Flex properties**

Presntation

Differents flex properties

**SLIDE 5 : display**

Pour transformer un [élément HTML](https://fr.w3docs.com/apprendre-html/les-elements-html.html) en conteneur Flex, vous devez utiliser la propriété [display](https://fr.w3docs.com/apprendre-css/display.html) avec les valeurs "flex" (zone de conteneur flexible au niveau des blocs) ou "inline-flex" (définissant la zone de conteneur flexible au niveau intégré). Sinon, le navigateur ignore toutes les propriétés Flexbox que vous avez utilisées.

This defines a flex container; inline or block depending on the given value. It enables a flex context for all its direct children.

**SLIDE 6 : align-items**

* stretch (default): stretch to fill the container (still respect min-width/max-width)
* flex-start / start / self-start: items are placed at the start of the cross axis. The difference between these is subtle, and is about respecting the flex-direction rules or the writing-mode rules.
* flex-end / end / self-end: items are placed at the end of the cross axis. The difference again is subtle and is about respecting flex-direction rules vs. writing-mode rules.
* center: items are centered in the cross-axis
* baseline: items are aligned such as their baselines align

**SLIDE 7 : Flex-direction**

This establishes the main-axis, thus defining the direction flex items are placed in the flex container. Flexbox is a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.

* row (default): left to right
* row-reverse: right to left
* column top to bottom
* column-reverse: bottom to top

**SLIDE 8 : justify-content**

This defines the alignment along the main axis. It helps distribute extra free space leftover when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

* flex-start (default): items are packed toward the start of the flex-direction.
* flex-end: items are packed toward the end of the flex-direction.
* center: items are centered along the line
* space-between: items are evenly distributed in the line; first item is on the start line, last item on the end line
* space-around: items are evenly distributed in the line with equal space around them. Note that visually the spaces aren’t equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies.
* space-evenly: items are distributed so that the spacing between any two items (and the space to the edges) is equal.

**SLIDE 8 : order**

**SLIDE 9 : Let’s Pratice**

Tp flexblox

**SLIDE ++ : En plus**

#### flex-flow

This is a shorthand for the flex-direction and flex-wrap properties, which together define the flex container’s main and cross axes. The default value is row nowrap.