CSS3 Flexbox Layout module

Vous connaissez certainement le modèle de boîte classique en CSS et ses dispositions de type “block” ou “inline”, sachez que Flexbox CSS3 a été conçu pour étendre ce périmètre en introduisant un nouveau modèle de boîte distinct, que l’on appellera **“le Modèle de boîte flexible”**.

Au sein de ce schéma, on ne raisonne plus en “block” ou “inline”, ni même en `float` ou autres types de boîtes “classiques” CSS, mais en “Modèle de boîte flexible”, dont **les quatre possibilités principales sont** :

1. **Distribution** des éléments horizontale ou verticale, avec passage à la ligne autorisé ou non,
2. **Alignements** et centrages horizontaux et verticaux, justifiés, répartis,
3. **Réorganisation** des éléments indépendamment de l’ordre du flux (DOM),
4. **Gestion des espaces disponibles** (fluidité).

Note : ce tutoriel a été initialement rédigé en octobre 2010. Il a subi une refonte intégrale en décembre 2014 pour se mettre à jour.

En action !

Flexbox (le modèle de boîte flexible) se fonde schématiquement sur une architecture de ce type :

- Un "flex-container" permettant de créer un contexte général d’affichage,
- Un ou plusieurs "flex-item" qui ne sont rien d’autre que les enfants directs du conteneur, quels qu’ils soient.

Le "flex-container", qui définit le contexte global de modèle de boîte flexible, est tout simplement n’importe quel élément HTML doté de la déclaration `display: flex;` ou `display: inline-flex;`.
Ses enfants deviennent alors automatiquement (inutile de leur déclarer quoi que ce soit) des éléments de type "flex-item" :

```
.container {
    display: flex;
}
```

Un élément "flex-item" n’est plus considéré comme un “bloc” ou un “inline” classique (d’ailleurs les valeurs de `display` autre que `none`, et même certaines propriétés telles que `float` n’ont plus d’effet sur lui).

Démonstration display: flex » /xmedia/tuto/exemples/flexbox/flextest01.html

Compatibilité

N'ayons pas peur des mots : le module Flexbox est **plutôt très bien reconnu** par les navigateurs, même certains glorieux anciens et même sur les mobiles en général, comme en témoigne l'excellente ressource CaniUse.com » http://caniuse.com/#search=flex .

Le seul retardataire est Internet Explorer qui ne supporte pas cette spécification pour ses anciennes versions (inférieures à IE10). Cela est éventuellement bloquant pour certains projets destinés à des ordinateurs de bureau.

Tableau des compatibilités

Navigateurs	Versions	Détails
	**Internet Explorer 10+** **IE mobile 10+**	- Ancienne spécification pour IE10 (2011). Avec préfixe `-ms-` - Spécification finale sans préfixe pour IE11
	**Firefox 2+**	- Ancienne spécification (2009). Avec préfixe `-moz-` - Spécification finale sans préfixe depuis Firefox 22
	**Chrome 4+** **Chrome Mobile (Android 4+)**	- Ancienne spécification (2009) depuis Chrome 4. Avec préfixe `-webkit-` - Spécification finale depuis Chrome 21. Avec préfixe `-webkit-` - Spécification finale sans préfixe depuis Chrome 29. - Ancienne spécification pour Chrome sur Android. Avec préfixe `-webkit-`
	**Opera 12.1+** **Opera Mobile 12.1+**	- Spécification finale sans préfixe entre les versions 12 et 15 - Spécification finale depuis Opera 16. Avec préfixe `-webkit-`
	**Safari 3.1+** **Safari Mobile (iOS 3.2+)**	- Ancienne spécification (2009). Avec préfixe `-webkit-` - Spécification finale depuis Safari 7. Avec préfixe `-webkit-`
	**Android Browser 2.1+**	- Ancienne spécification (2009). Avec préfixe `-webkit-`

Standardisation

L'état de l'art de la standardisation de "Flexible Box Layout Module" est pour le moins pittoresque tant il a connu de rebondissements.

Pour résumer, après trois refontes complètes depuis sa création en 2009, Flexbox *était* quasiment finalisé en 2012 (la spécification était au stade de "Candidate Recommandation"), puis, en septembre 2014 a été rétrogradé au stade de "Last Call Working Draft" (brouillon en dernier appel).

Pourquoi ce retour en arrière ?

Parce que les éditeurs de la spécification semblent vraiment souhaiter que ce module "ait de la gueule", qu'il devienne une réelle solution à l'ensemble des contraintes actuelles de mise en forme CSS. Contraintes qui datent parfois de plus de 15 ans, comme le centrage vertical par exemple.

Est-ce grave ou bloquant ? Certainement pas, à condition de prendre quelques précautions sous la forme d'un outil génial qu'est **Autoprefixer. » http://css-tricks.com/autoprefixer/**

Distribution et axe principal

La distribution, c'est à dire le sens d'affichage horizontal ou vertical des éléments "flex-items" est définie par la propriété **`flex-direction`** dont les valeurs peuvent être :

- `row` (distribution horizontale, valeur par défaut)
- `row-reverse` (distribution horizontale inversée)
- `column` (distribution verticale)
- `column-reverse` (distribution verticale inversée)

Cette propriété s'applique au "flex-container" et **détermine l'axe principal** du modèle de boîte flexible.

```
.container {
    display: flex;
    flex-direction: column;
}
```

Démonstration flex-direction: column » /xmedia/tuto/exemples/flexbox/flextest02.html

La propriété `flex-wrap` définit si le contenu sera distribué sur une seule ligne (ou colonne selon l'axe principal) ou sur plusieurs lignes. En clair, si les "flex-items" **ont le droit de passer à la ligne ou non**.

Les valeurs de `flex-wrap` sont :

- `nowrap` (les éléments ne passent pas à la ligne, valeur par défaut)
- `wrap` (les éléments passent à la ligne dans le sens de lecture)
- `wrap-reverse` (les éléments passent à la ligne dans le sens inverse)

```
.container {
    display: flex;
    flex-wrap: wrap-reverse;
}
```

Démonstration flex-wrap: wrap-reverse » /xmedia/tuto/exemples/flexbox/flextest03b.html

À noter qu'il existe une propriété raccourcie `flex-flow` qui regroupe `flex-direction` et `flex-wrap` .

```
/* affichage en ligne et passage à la ligne autorisé */
.container {
    flex-flow: row wrap;
}
```

Alignements

Flexbox propose de gérer très finement les alignements et centrages, en différenciant les deux axes d'affichage de cette manière :

-L'alignement dans l'axe principal est traité via la propriété **`justify-content`**
- L'alignement dans l'axe secondaire est géré avec **`align-items`**

Ces deux propriétés s'appliquent au "flex-container".

Axe principal : justify-content

Les alignements dans l'axe de lecture principal sont définis à l'aide de la propriété `justify-content`, dont les valeurs possibles sont :

- `flex-start` (éléments positionnés au début du sens de lecture, valeur par défaut)
- `flex-end` (éléments positionnés à la fin)
- `center` (position centrale)
- `space-between` (répartition "justifiée")
- `space-around` (variante de répartition "justifiée")

```
/* éléments positionnés en bas du conteneur */
.container {
    flex-direction: column;
    justify-content: flex-end;
}
```

Démonstration justify-content » /xmedia/tuto/exemples/flexbox/flextest04.html


```
.parent {
   display: flex;
   flex-direction: column;
   justify-content: flex-end;
}
```


```
.parent {
   display: flex;
   flex-direction: column;
   justify-content: space-
   between;
}
```

Axe secondaire: align-items

Dans l'axe secondaire, les alignements sont régis via la propriété `align-items`, dont les valeurs sont :

- `flex-start` (au début)
- `flex-end` (à la fin)
- `center` (au centre)
- `baseline` (généralement identique à `flex-start`)
- `stretch` (étirés dans l'espace disponible, valeur par défaut)

```
/* éléments étirés (valeur par défaut) */
.container {
    flex-direction: column;
    align-items: stretch;
}
```

Démonstration align-items » /xmedia/tuto/exemples/flexbox/flextest05.html


```
.parent {
   display: flex;
   flex-direction: column;
   align-items: stretch;
}
```

Traiter les cas particuliers : align-self

La propriété `align-self` , permet de distinguer l'alignement d'un "flex-item" de ses frères. Les valeurs de cette propriété sont identiques à celles de `align-items`.

```
/* seul le paragraphe sera à droite */
.container {
    align-items: stretch;
}
p {
    align-self: flex-end;
}
```

Démonstration align-self » /xmedia/tuto/exemples/flexbox/flextest05b.html


```
.parent {
   display: flex;
   flex-direction: column;
   align-items: stretch;
}
.oignon {
   align-self: flex-end;
}
```

Propriété margin

La propriété `margin` lorsqu'elle est affectée à un "flex-item" ouvre de nouvelles perspectives, notamment dans l'axe vertical puisque Flexbox n'est plus lié à un sens de lecture en particulier.

En clair, il devient possible de positionner un élément en bas de son conteneur à l'aide d'un `margin-top: auto`, ou mieux : **centrer à la fois horizontalement et verticalement via un simple `margin: auto`.**

```
/* paragraphe centré horizontalement et verticalement */
.container {
    display: flex;
}
.container > p {
    margin: auto;
}
```

Démonstration margin » /xmedia/tuto/exemples/flexbox/flextest05c.html

Ordonnancement

L'une des fonctionnalités les plus avant-gardistes du modèle d'affichage Flexbox est de pouvoir réordonner à sa guise chacun des éléments indépendamment grâce à la propriété `order`.

Les valeurs de `order` agissent telles des pondérations : les éléments dont la valeur est la plus forte se trouveront en bas de la pile.

La propriété `order` s'applique aux "flex-items" et sa valeur initiale est `0`.

```
/* le premier de la liste s'affichera en bas de pile */
li:first-of-type {
    order: 1;
}
```

Démonstration order » /xmedia/tuto/exemples/flexbox/flextest07.html


```
.oignon {
   order: 1;
}
```


```
.picon-biere {
   order: -1;
}

.oignon {
   order: 1;
}
.salade {
   order: 2;
}
```

Flexibilité

Cela ne devrait étonner personne, la notion de flexibilité constitue le fondement du module de positionnement Flexbox, et c'est là qu'intervient l'indispensable propriété `flex`.

La propriété `flex` est un raccourci de trois propriétés, `flex-grow`, `flex-shrink` et `flex-basis`, qui s'appliquent aux "flex-items" et dont les fonctionnalités sont:

- `flex-grow` : capacité pour un élément à s'étirer dans l'espace restant,
- `flex-shrink` : capacité pour un élément à se contracter si nécessaire,
- `flex-basis` : taille initiale de l'élément avant que l'espace restant ne soit distribué.

Par défaut, les valeurs de ces propriétés sont : `flex-grow: 0`, `flex-shrink: 1` et `flex-basis: auto`.
En clair, les `flex-items` n'occupent initialement que la taille minimale de leur contenu.

Pour rendre un élément flexible, il suffit de lui attribuer une valeur de `flex-grow` (ou `flex` en raccourci) supérieure à zéro.
Cet élément occupera alors l'espace restant au sein de son conteneur :

```
/* .salade occupera l'espace restant */
.salade {
    flex: 1;
}
```

Plusieurs éléments peuvent être rendus flexibles et se répartir l'espace restant. L'espace disponible est alors tout simplement distribué entre les éléments flexibles.

Démonstration flex n°1 » /xmedia/tuto/exemples/flexbox/flextest06.html

Gabarit flexible » http://codepen.io/raphaelgoetter/full/EajYme/


```
.oignon {
   flex: 1;
}
```


```
.tomate {
   flex: 2;
}
```

2/3

```
.oignon {
   flex: 1;
}
```

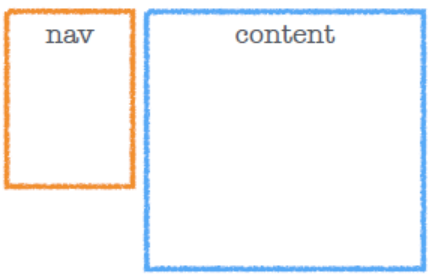
1/3

Bonus : BFC

En guise de bonus, sachez que le modèle de boîte flexible dispose également de sérieux atouts lorsqu'il s'agit de dompter ces fichus flottants qui débordent tout le temps.

Les éléments "flex-container" et "flex-item" construisent un *Block Formatting Context* (BFC), ou plus exactement un *Flex Formatting Context* dont les avantages sont similaires et décrits dans cet article. » http://www.alsacreations.com/astuce/lire/1543-le-contexte-de-formatage-block-en-css.html :

- un flottant ne déborde pas d'un "flex-container" ni d'un "flex-item",
- un "flex-container" ou un "flex-item" ne s'écoule pas autour d'un flottant (ils demeurent *à côté*),
- il n'y a pas de fusion de marges » http://www.alsacreations.com/article/lire/629-fusion-des-marges.html .




```
nav {
   float: left;
}
.content {
   display: flex;
}
```

Conclusion

Flexbox est une spécification vaste et complexe, nous n'en n'avons démontré que les parties les plus utiles au quotidien mais d'autres recoins restent à explorer et seront dévoilés au fur et à mesure de leur support avec les navigateurs.
Quoi qu'il en soit, le positionnement Flexible Layout regroupe de nombreuses bonnes pratiques et attentes des webdesigners et

intégrateurs depuis de longues années.

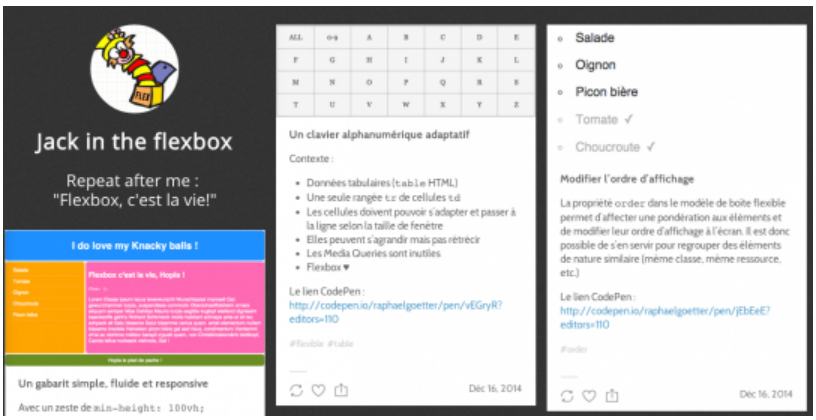
Dès aujourd'hui (ou dans un avenir proche selon votre cible), il constitue sans aucun doute la méthode de positionnement la plus pratique et polyvalente qui n'ait jamais existé en CSS.

Ressources

Voici quelques ressources externes pour en savoir plus sur ce positionnement très polyvalent :

Articles et tutoriels connexes

- W3C : Spécification Flexbox Module » http://www.w3.org/TR/css3-flexbox/
- Jack in the flexbox » http://jackintheflexbox.tumblr.com/ : une galerie de démonstrations dédiée à Flexbox
- Solved by Flexbox » http://philipwalton.github.io/solved-by-flexbox/ : des problèmes CSS classiques réglés grâce à Flexbox
- Flexyboxes » http://the-echoplex.net/flexyboxes/ : un bac à sable pour tester Flexbox
- Flexbox Adventures » http://chriswrightdesign.com/experiments/flexbox-adventures/ : un article anglais très détaillé sur les subtilités de Flexbox
- Flexbugs » https://github.com/philipwalton/flexbugs : les bugs connus de flexbox et comment les contourner



» http://jackintheflexbox.tumblr.com/