Architecture et langages Web
(NFA040)

CSS : flex (tutoriel)

Remarque. Ce tutoriel est extrait de la section de MDN intitulée : Flexbox. Les sources des exemples accompagnent ce sujet.

Flexbox est une méthode de mise en page selon un axe principal, permettant de disposer des éléments en ligne ou en colonne. Les éléments se dilatent ou se rétractent pour occuper l'espace disponible. Cet article en explique tous les fondamentaux.

1 Pourquoi Flexbox ?

Pendant longtemps, les seuls outils de mise en page CSS fiables et compatibles avec les navigateurs, étaient les propriétés concernant les flotteurs (boîtes flottantes) et le positionnement. Ces outils sont bien et fonctionnent, mais restent à certains égards plutôt limitatifs et frustrants.

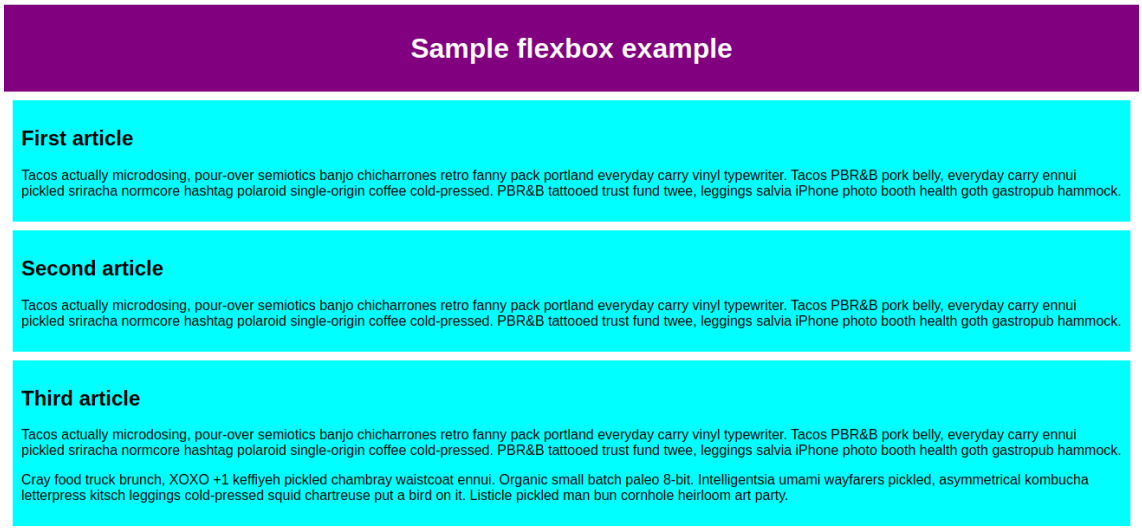
Les simples exigences de mise en page suivantes sont difficiles sinon impossibles à réaliser de manière pratique et souple avec ces outils :

- Centrer verticalement un bloc de contenu dans son parent ;
- Faire que tous les enfants d'un conteneur occupent tous une même quantité de hauteur/largeur disponible selon l'espace offert ;
- Faire que toutes les colonnes dans une disposition multi-colonnes aient la même hauteur même si leur quantité de contenu diffère.

Comme vous le verrez dans les paragraphes suivants, Flexbox facilite considérablement les tâches de mise en page. Approfondissons un peu !

2 Voici un exemple simple

Dans cet article, nous allons commencer une série d'exercices pour vous faciliter la compréhension du fonctionnement de Flexbox. Pour commencer, faites une copie locale du premier fichier de démarrage — flexbox0.html de notre dépôt GitHub. Chargez-le dans un navigateur moderne (comme Firefox ou Chrome) et regardez le code dans votre éditeur. Vous pouvez le voir en direct ici aussi.



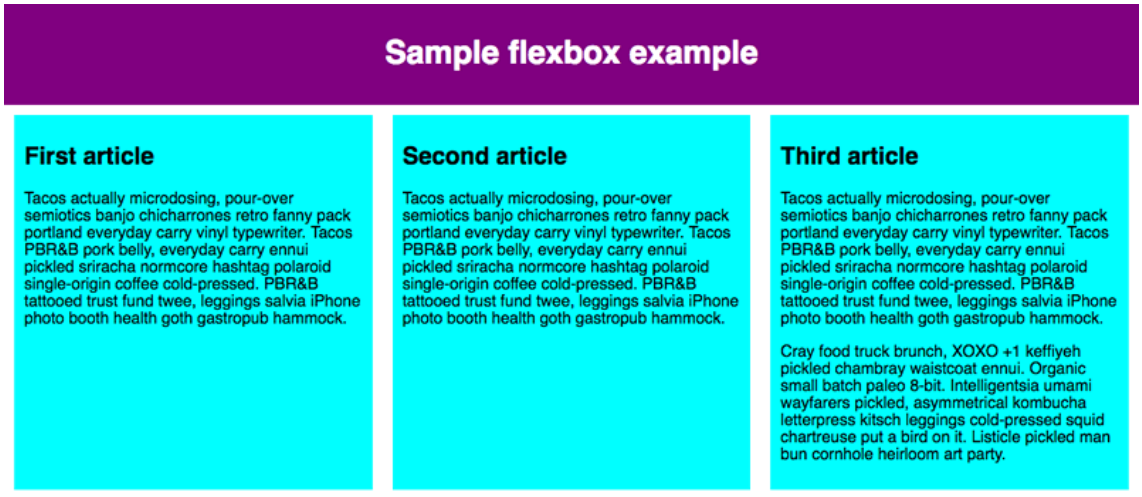
Qu'avons-nous ? Un élément `<header>` avec un en-tête de haut niveau à l'intérieur, et un élément `<section>` contenant trois éléments `<article>`. Nous allons les utiliser pour créer une disposition vraiment classique sur trois colonnes.

Détermination des éléments à disposer en boîtes flexibles

Pour commencer, sélectionnons les éléments devant être présentés sous forme de boîtes flexibles. Pour ce faire, donnons une valeur spéciale à la propriété `display` du parent de ces éléments à disposer. Dans ce cas, comme cela concerne les éléments `<article>`, nous affectons la valeur `flex` à l'élément `<section>` (qui devient un conteneur flex) :

```
section {
  display: flex;
}
```

Voici le résultat :

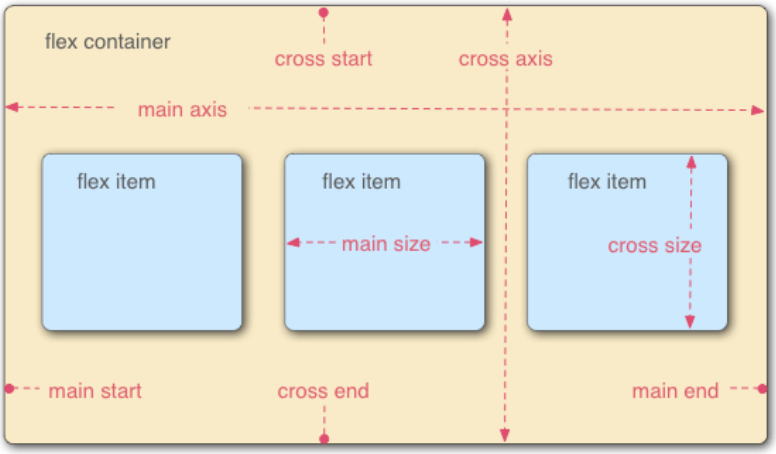


Cette unique déclaration donne tout ce dont nous avons besoin — incroyable, non ? Nous avons ainsi notre disposition en plusieurs colonnes de largeur égale et toutes de même hauteur. Ceci parce que les valeurs par défaut données aux éléments flex (les enfants du conteneur flex) sont configurés pour résoudre des problèmes courants tels celui-ci.

Récapitulons ce qui se passe ici : l'élément auquel nous avons affecté une valeur `display` de `flex` se comporte d'une manière d'un élément de bloc par rapport à sa façon d'interagir avec le reste de la page, tandis que les enfants se disposent comme des éléments flexibles. On en reparlera plus tard. À noter également que vous pouvez définir une valeur `inline-flex` pour `display` si vous voulez disposer des éléments en ligne sous forme de boîtes modulables.

Aparté sur le modèle flex

Lorsque les éléments sont disposés en boîtes flexibles, ils sont disposés le long de deux axes :



- L'**axe principal (*main axis*)** est l'axe de la direction dans laquelle sont disposés les éléments flex (par exemple, horizontalement sur la page, ou verticalement de haut en bas de la page). Le début et la fin de cet axe sont appelés l'**origine principale (*main start*)** et la **fin principale (*main end*)**.

- L'**axe croisé (*cross axis*)** est l'axe perpendiculaire à l'axe principal, c'est-à-dire à la direction dans laquelle sont disposés les éléments flex. Le début et la fin de cet axe sont appelés le **début (*cross start*)** et la **fin (*cross end*)** de l'axe croisé.

- L'élément parent dont la propriété est `display: flex` (`<section>` dans notre exemple) est appelé le **conteneur flex (*flex container*)**.

- Les éléments disposés en tant que boîtes flexibles à l'intérieur du conteneur flex sont appelés **éléments flex (*flex items*)** (les éléments `<article>` dans notre exemple).

Gardez cette terminologie en tête en lisant les paragraphes suivants. Vous pouvez toujours vous y référer si vous avez un doute sur la signification des termes utilisés.

3 Colonnes ou lignes ?

Flexbox dispose de la propriété `flex-direction` pour indiquer la direction de l'axe principal (direction dans laquelle les enfants flexibles sont disposés). Cette propriété est égale par défaut à `row` : ils sont donc disposés en ligne, dans le sens de lecture de la langue par défaut du navigateur (de gauche à droite, dans le cas d'un navigateur français).

Ajoutez la déclaration suivante dans la règle CSS pour l'élément `<section>` :

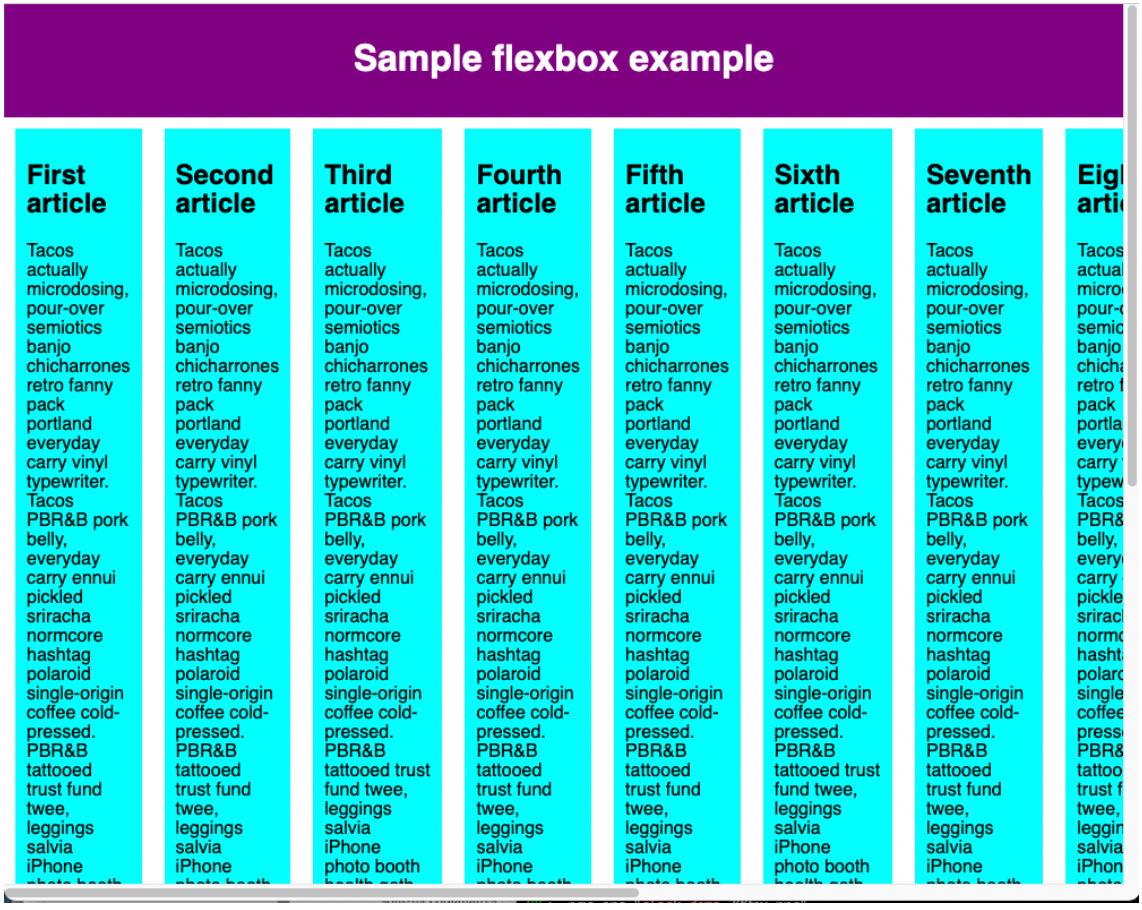
```
flex-direction: column;
```

Cela dispose de nouveau les éléments en colonnes, comme c'était le cas avant l'ajout de la CSS. Avant de poursuivre, enlevez cette déclaration de l'exemple.

Note : Vous pouvez aussi disposer les éléments flex dans la direction inverse avec les valeurs `row-reverse` et `column-reverse`. Expérimentez ces valeurs aussi !

4 Enveloppement

Problème : quand votre structure est de largeur ou hauteur fixe, il arrive que les éléments flex débordent du conteneur et brisent cette structure. Voyez l'exemple flexbox-wrap0.html et essayez-le directement (faites une copie locale de ce fichier maintenant si vous voulez suivre cet exemple) :



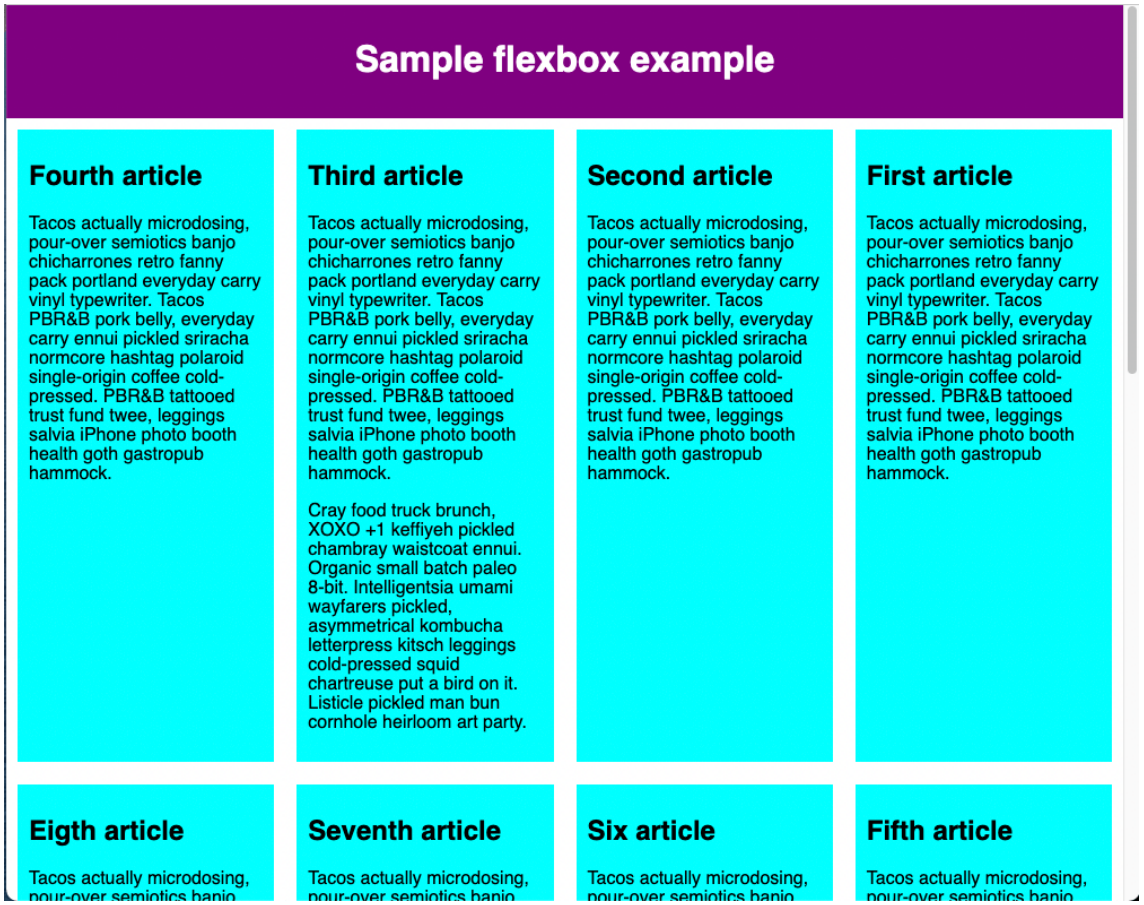
Ici, nous voyons que les enfants débordent du conteneur. Une façon d'y remédier est d'ajouter la déclaration suivante à votre règle pour `<section>` :

```
flex-wrap: wrap;
```

Ajoutez aussi la déclaration suivante à votre règle pour `<article>` :

```
flex: 200px;
```

Essayons ; la disposition est bien meilleure avec ces ajouts :



Nous avons maintenant plusieurs lignes — un nombre sensé d'enfants flexibles est placé sur chaque ligne, et le débordement est déplacé vers le bas sur une ligne supplémentaire. La déclaration `flex: 200px` pour les éléments `article` signifie que chacun aura au moins 200px de large ; nous discuterons de cette propriété plus en détail plus tard. Vous noterez aussi que chacun des enfants de la dernière rangée est plus large, de façon à ce que toute la rangée reste remplie.

Mais nous pouvons faire plus ici. Tout d'abord, essayez de changer la valeur de la propriété `flex-direction` en `row-reverse` — maintenant vous avez toujours la disposition sur plusieurs lignes, mais elles commencent dans l'angle opposé de la fenêtre du navigateur et se disposent à l'envers.

Forme abrégée « flex-flow »

Notez maintenant qu'il y a une forme abrégée pour `flex-direction` et `flex-wrap` — `flex-flow`. Ainsi, par exemple, vous pouvez remplacer :

```
flex-direction: row;
flex-wrap: wrap;
```

par :

```
flex-flow: row wrap;
```

5 Taille modulable des éléments flex

Revenons maintenant au premier exemple, et examinons comment nous pouvons contrôler la proportion d'éléments flexibles dans l'espace. Lancez votre copie locale de flexbox0.html ou prenez une copie de flexbox1.html comme nouveau point de départ (voir en direct).

Ajoutez d'abord la règle ci-dessous en fin de la CSS :

```
article {
  flex: 1;
}
```

Il s'agit d'une valeur de proportion, sans unité, définissant la quantité d'espace disponible que chaque élément flex prendra le long de l'axe principal. Dans ce cas, nous donnons à chaque élément `<article>` une valeur de 1, ce qui signifie qu'ils prendront tous une portion égale de l'espace libre après le calcul du remplissage et de la marge. Cette valeur représente une proportion, c'est-à-dire que le fait de donner une valeur de 400 000 simultanément à tous les éléments flex aurait exactement le même effet.

Maintenant ajoutons cette règle en dessous de la précédente :

```
article:nth-of-type(3) {
  flex: 2;
}
```

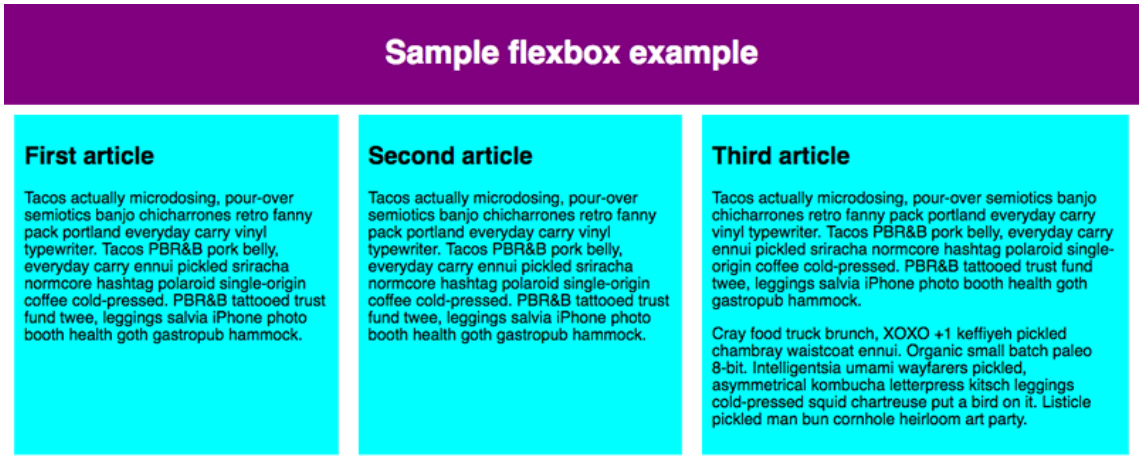
Maintenant, lorsque vous actualisez, vous voyez que le troisième `<article>` occupe deux fois plus de largeur disponible que chacun des deux autres — il y a maintenant quatre unités de division disponibles au total (puisque 1 + 1 + 2 = 4). Les deux premiers éléments flexibles en occupent chacun un, soit 1/4 de l'espace disponible pour chacun. Le troisième remplit deux unités, soit 2/4 (la moitié) de l'espace disponible.

Vous pouvez également définir une valeur minimale de taille dans la valeur `flex`. Modifiez comme suit vos règles `article` existantes :

```
article {
  flex: 1 200px;
}

article:nth-of-type(3) {
  flex: 2 200px;
}
```

En gros, cela dit : « Chaque élément flex reçoit d'abord 200px de l'espace disponible. Ensuite, le reste de l'espace disponible est réparti selon les unités de proportion ». Actualisez et vous devriez voir une différence dans la façon dont l'espace est réparti.

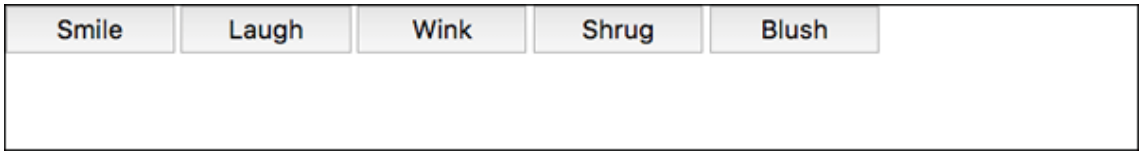


Le véritable intérêt de Flexbox apparaît dans sa souplesse et sa réactivité — si vous redimensionnez la fenêtre du navigateur ou ajoutez un autre élément `<article>`, la mise en page continue de fonctionner correctement.

6 Alignement horizontal et vertical

Vous pouvez également utiliser les fonctionnalités de Flexbox pour aligner les éléments flex le long de l'axe principal ou croisé. Voyons cela avec un nouvel exemple — flex-align0.html (voir

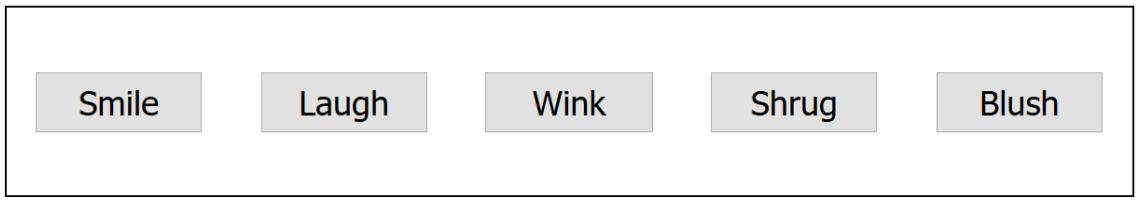
aussi en direct). Nous allons le transformer facilement en une barre souple de boutons. Actuellement, nous avons une barre de menu horizontale avec quelques boutons tassés dans l'angle supérieur gauche.



D'abord, faites une copie locale de cet exemple.

Ensuite, ajoutez ce qui suit à la fin de la CSS de l'exemple :

```
div {
  display: flex;
  align-items: center;
  justify-content: space-around;
}
```



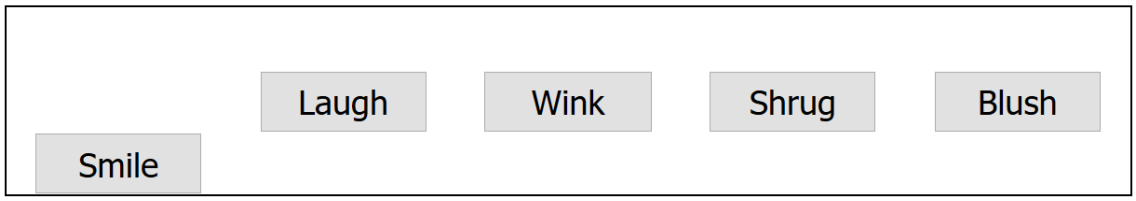
Actualisez la page et vous verrez que les boutons sont maintenant bien centrés, horizontalement et verticalement. Cette transformation a été opérée grâce à deux nouvelles propriétés.

`align-items` fixe là où les éléments flex sont placés sur l'axe perpendiculaire, dit aussi croisé (*cross axis*).

- Par défaut, la valeur est `stretch`, qui étire tous les éléments flex de manière à emplir le conteneur parent le long de l'axe croisé. Si le parent ne possède pas de dimension définie dans la direction de l'axe croisé, alors tous les éléments flex auront la dimension du plus étiré des éléments. C'est pour cette raison que, dans notre premier exemple, les colonnes ont toutes la même hauteur par défaut.
- Avec la valeur `center` utilisée dans le code ci-dessus, les éléments gardent leur dimension intrinsèque, tout en étant centrés sur l'axe croisé. C'est la raison pour laquelle, dans l'exemple actuel, les boutons sont centrés verticalement.
- Il y a également des valeurs comme `flex-start` et `flex-end` qui alignent respectivement tous les éléments au début ou à la fin de l'axe croisé. Voyez `align-items` pour tous les détails.

Vous pouvez prendre le pas sur le comportement de `align-items` pour un élément flex donné en lui appliquant la propriété `align-self`. Par exemple, ajoutez ce qui suit aux CSS :

```
button:first-child {
  align-self: flex-end;
}
```



Voyez l'effet obtenu, puis supprimez ensuite la règle.

`justify-content` fixe où les éléments flex sont placés sur l'axe principal.

- La valeur par défaut est `flex-start` : tous les éléments sont disposés vers l'origine de l'axe principal.
- Vous utiliserez `flex-end` pour les disposer vers la fin.
- `center` est aussi une valeur possible pour `justify-content`. Avec elle, les éléments flex sont placés vers le centre de l'axe principal.
- La valeur `space-around`, utilisée plus haut, est pratique — elle distribue régulièrement tous les éléments le long de l'axe principal, en laissant autant d'espace à chaque extrémité qu'entre chacun.
- Une autre valeur, `space-between`, est semblable à `space-around`, mais elle ne laisse pas d'espace aux extrémités.

N'hésitez pas à jouer avec ces valeurs pour visualiser leur fonctionnement avant de poursuivre.

7 Ordonner les éléments flex

Flexbox dispose aussi d'une fonctionnalité pour modifier l'ordre d'affichage des éléments flex, sans en modifier l'ordre dans la source. C'est une chose impossible à réaliser avec les méthodes classiques de mise en page.

Le code pour ce faire est simple — ajoutez la règle CSS suivante dans l'exemple de code de la barre de boutons :

```
button:first-child {
  order: 1;
}
```

Actualisez, et vous pouvez voir que le bouton « Smile » a été déplacé en fin de l'axe principal. Voyons en détail comment cela fonctionne :

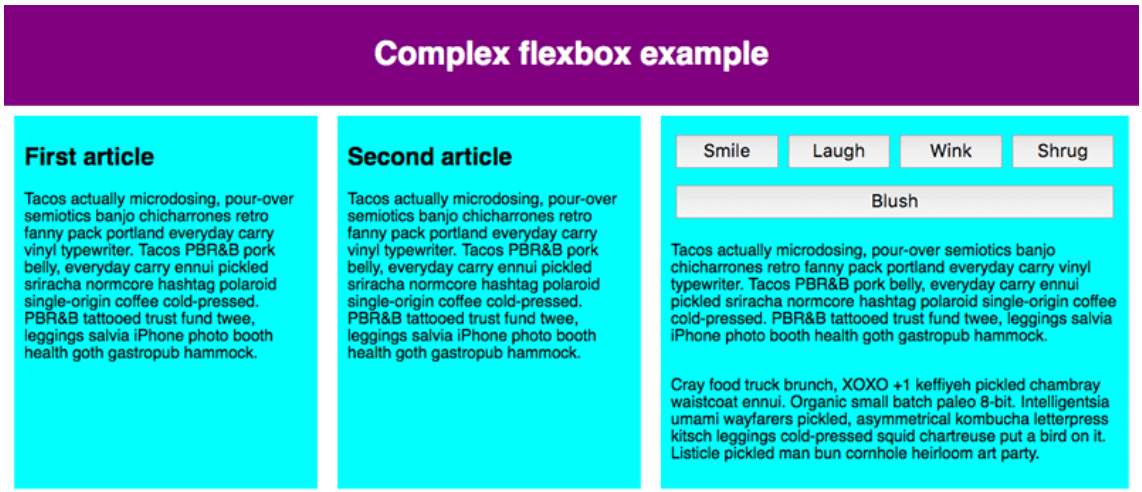
- Par défaut, tous les éléments flex possèdent une valeur `order` égale à 0 ;
- Les éléments flex avec des valeurs `order` plus élevées apparaîtront plus tard dans l'ordre d'affichage que ceux avec des valeurs plus faibles ;
- Les éléments flex avec les mêmes valeurs pour `order` sont affichés dans l'ordre de la source. Ainsi, si vous avez 4 éléments avec des valeurs `order` de 2, 1, 1 et 0, leur ordre d'affichage sera 4e, 2e, 3e et 1er ;
- Le troisième élément suit le deuxième, car il a la même valeur pour `order` et qu'il est placé après dans le code source.

Vous pouvez donner des valeurs négatives à `order` pour faire en sorte que ces éléments soient affichés avant les éléments d'ordre 0. Par exemple, vous pouvez faire apparaître le bouton « Blush » en tête de l'axe principal avec la règle suivante :

```
button:last-child {
  order: -1;
}
```

8 Boîtes flex imbriquées

Il est possible de créer des mises en page joliment complexes avec Flexbox. Il est parfaitement loisible de déclarer un élément flex en tant que conteneur flex, de sorte que ses enfants sont également disposés en tant que boîtes modulables. Regardez complex-flexbox.html (à voir en direct également).



Le HTML pour cela est vraiment simple. Voici un élément `<section>` contenant trois éléments `<article>`. Le troisième élément `<article>` contient trois éléments `<div>`, et le premier élément `<div>` contient cinq éléments `<button>` :

```
section - article
          article
          article - div - button
                    div   button
                    div   button
                          button
                          button
```

Regardez le code utilisé pour cette disposition.

Primo, nous déterminons que les enfants de l'élément `<section>` seront des boîtes flexibles.

```
section {
  display: flex;
}
```

Secundo, nous définissons des valeurs flex pour les éléments `<article>` eux-mêmes. Remarquez en particulier ici la deuxième règle — nous paramétrons le troisième élément `<article>` pour que ses enfants soient eux-mêmes disposés en tant qu'éléments flex, mais cette fois-ci en colonne.

```
article {
  flex: 1 200px;
}

article:nth-of-type(3) {
  flex: 3 200px;
  display: flex;
  flex-flow: column;
}
```

Tertio, nous sélectionnons le premier élément `<div>` et lui assignons la valeur `flex: 1 100px;` pour qu'il ait effectivement une hauteur minimale de 100px. Ensuite, nous indiquons que ses enfants (les éléments `<button>`) doivent être disposés en tant qu'éléments flex dans une ligne enveloppante, centrés dans l'espace disponible comme dans l'exemple des boutons vu plus haut.

```
article:nth-of-type(3) div:first-child {
  flex: 1 100px;
  display: flex;
  flex-flow: row wrap;
  align-items: center;
  justify-content: space-around;
}
```

Enfin, nous définissons un dimensionnement des boutons, et plus précisément nous leur donnons une valeur flex de `1 auto`. L'effet obtenu est très intéressant ; vous l'observerez en modifiant la largeur de la fenêtre du navigateur. Les boutons prennent autant d'espace qu'il leur est permis et sont, si possible, disposés sur la même ligne ; sinon ils « descendent » pour créer de nouvelles lignes.

```
button {
  flex: 1 auto;
  margin: 5px;
  font-size: 18px;
  line-height: 1.5;
}
```