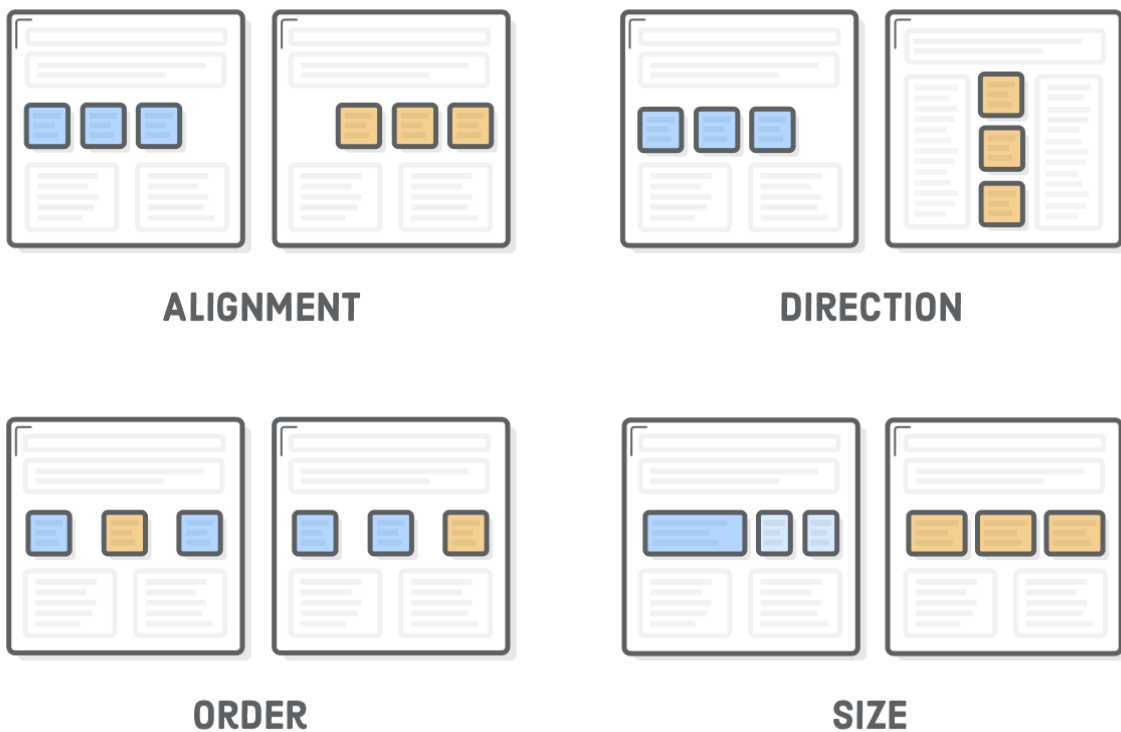


Flexbox

La "Boîte Flexible" ou le mode de mise en page "Flexbox" offre une alternative à float pour définir l'apparence générale d'une page Web. Alors que les float ne nous laissent positionner horizontalement nos boîtes, FlexBox nous donne le contrôle *total* sur l'alignement, la direction, l'ordre et la taille de nos boîtes.



Le web connaît actuellement une transition majeure, alors une petite discussion autour de l'état de l'industrie est justifiée. Dans la dernière décennie, les float étaient la seule option pour une mise en page web complexe. En conséquence, toutes ces règles sont bien prises en charge dans les navigateurs existants, et les développeurs les ont utilisées pour construire des millions de pages Web. Cela signifie que vous rencontrerez inévitablement des float au cours de votre carrière de développement web.

Malgré ce que nous avons vu jusqu'à présent, les types de configurations que vous pouvez créer avec des float sont en fait assez limitées. Même une

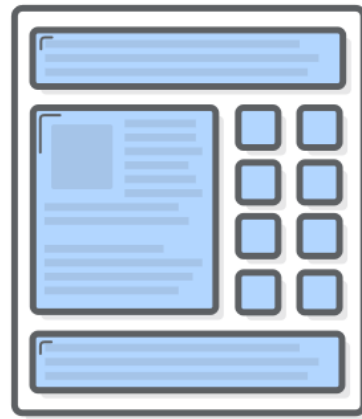
mise en page en deux colonnes simple est, techniquement parlant, un peu “un hack”. Flexbox a été inventé pour sortir de ces limites.

Nous sommes enfin à un point où le support de flexbox par les navigateurs a atteint une taille critique et les développeurs peuvent commencer à construire des sites Web complets avec FlexBox. La recommandation est d'utiliser FlexBox pour vos pages web, autant que possible, en réservant les float lorsque vous avez besoin pour par exemple faire couler du texte autour d'une boîte (ie, une mise en page de type magazine) ou lorsque vous devez prendre en charge les navigateurs web ne supportant pas flexbox.



FLOATS

(MAGAZINE-STYLE LAYOUTS)



FLEXBOX

(OVERALL PAGE STRUCTURE)

Dans ce chapitre, nous allons explorer l'ensemble du modèle de mise en page FlexBox étape par étape.

Installer

L'exemple de ce chapitre est relativement simple, mais il démontre clairement toutes les propriétés de Flexbox. Nous nous retrouvons avec quelque chose qui ressemble à ceci:



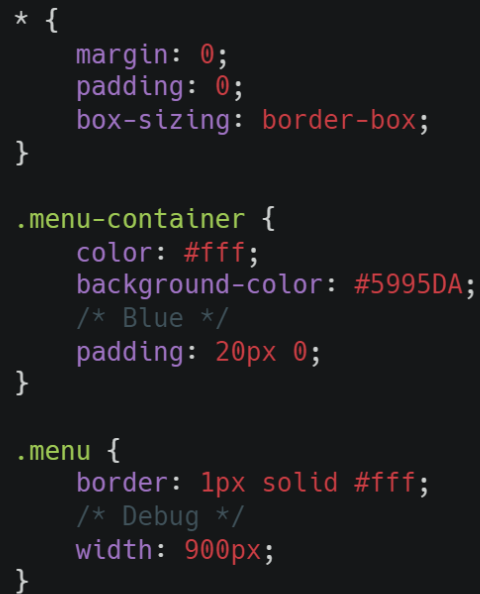
Faire un nouveau dossier appelé *flexbox* dans lequel on mettra tous les fichiers d'exemple pour ce chapitre.

Ensuite, créer un fichier *flexbox.html* et y ajoutez le balisage suivant:

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Flexbox</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div class="menu-container">
      <nav class="menu">
        <div class="date">20 mars 2017</div>
        <div class="signup">S'inscrire</div>
        <div class="login">Se connecter</div>
      </nav>
    </div>
```

Ensuite, nous devons créer la feuille de style correspondante *styles.css*. Cela ne ressemblera à pas grand chose: juste une barre de menu bleue de pleine largeur avec à l'intérieur une boîte à bordure blanche.

Notez que nous allons utiliser FlexBox au lieu de notre technique traditionnelle de “*margin: auto*” pour centrer le menu.



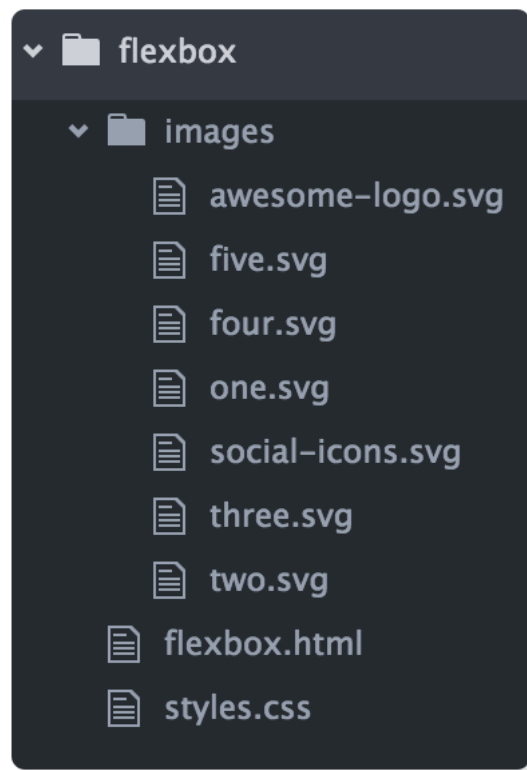
```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

.menu-container {
  color: #fff;
  background-color: #5995DA;
  /* Blue */
  padding: 20px 0;
}

.menu {
  border: 1px solid #fff;
  /* Debug */
  width: 900px;
}
```

Enfin, [télécharger des images](#) pour une utilisation dans notre page web. Décompressez-les dans le projet *flexbox*, en gardant le répertoire parent *images*.

Votre projet devrait ressembler à ceci:



Vue d'ensemble de flexbox

Flexbox utilise deux types de boîtes que nous avons jamais vu auparavant. à savoir les « *flex container* » et « *flex items* ».

L'utilité d'un conteneur flexible est de regrouper un ensemble d'éléments flexibles ensemble et de définir la façon dont ils sont positionnés.



“FLEX CONTAINER”



“FLEX ITEMS”

Chaque élément HTML qui est un enfant direct d'un conteneur flexible "flex container" est un élément, un "item". Les éléments Flex peuvent être manipulés individuellement, mais c'est le container qui va déterminer leur mise en page.

Comme avec les configurations à base de *float*, définir des pages web complexes avec FlexBox est un problème d'emboîtement. Vous alignez un tas d'éléments flexibles dans un conteneur, et, à son tour, ces éléments peuvent servir de conteneurs flexibles pour leurs propres éléments. Rappelez-vous que l'objectif de faire une mise en page n'a pas changé: *nous en sommes toujours à déplacer un tas de boîtes imbriquées les unes dans les autres.*

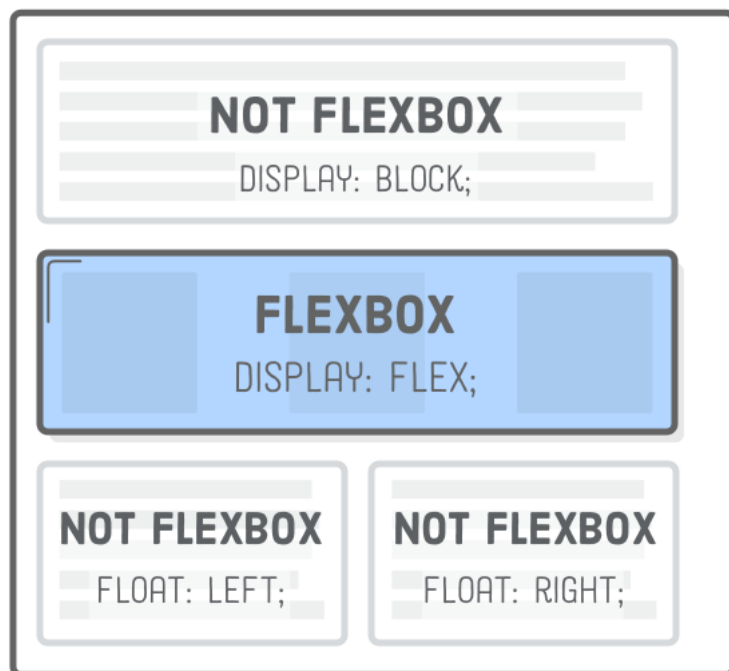
Flex container

La première étape de l'utilisation de FlexBox consiste à transformer l'un de ses éléments HTML dans un conteneur flexible. Nous le faisons avec la propriété *display*, qui devrait vous être familière. En lui donnant une valeur de *flex*, nous disons au navigateur que tout dans la boîte doit être affiché selon FlexBox au lieu du modèle de boîte par défaut.

Ajoutez la ligne suivante à notre règle *.menu-container* pour le transformer en un conteneur flexible:



Cela permet le mode de mise en page flexbox, sans cela, le navigateur ignorera toutes les propriétés flexbox que nous sommes sur le point d'introduire. La définition explicite des conteneurs flex permet de mélanger et de combiner flexbox avec d'autres modèles de mise en page (par exemple, des floats).



Nous avons un conteneur flexible avec un élément flexible en lui même. Cependant, notre page sera exactement comme avant parce que l'on n'a pas dit au conteneur comment afficher son élément.

Il existe aussi la propriété *inline-flex* qui génère un container flex, de niveau inline, à l'intérieur de l'élément.

Notez que les propriétés `columns` du module [multi-colonnes](#) n'ont pas d'effet sur un container flex ainsi que `float`, `clear` et `vertical-align` n'ont pas d'effet sur un élément flex.

Alignement d'un objet flex

Une fois que vous avez un conteneur flexible, votre prochaine étape est de définir l'alignement horizontal de ses éléments. Voilà ce pourquoi la propriété *justify-content* est faite.

Nous pouvons l'utiliser pour centrer notre *.menu*, comme ceci:

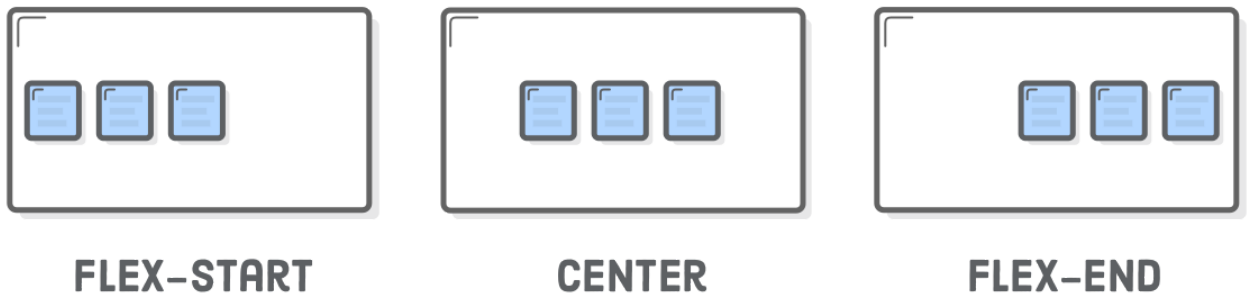


```
.menu-container {  
  color: #fff;  
  background-color: #5995DA;  
  /* Blue */  
  padding: 20px 0;  
  /* Nouveau */  
  justify-content: center;  
  display: flex;  
}
```

Cela a le même effet que l'ajout d'une déclaration *margin: 0 auto* à l'élément *.menu*. Mais, remarquez comment nous avons fait cela en ajoutant une

propriété au *parent* (le conteneur flex) au lieu de l'appliquer directement à l'élément que nous voulons centrer (l'élément flex).

Manipuler des l'élément depuis leurs conteneurs est très commun dans flexbox, et c'est un peu en opposition avec la façon dont nous avons fait cela jusqu'à présent avec le positionnement de boîtes.



Les autres valeurs pour *justify-content* sont :

- *center*
- *flex-start*
- *flex-end*
- *space-around*
- *space-between*

Essayez de changer *justify-content* par *flex-start* et *flex-end*. Cela devrait aligner le menu respectivement sur le côté gauche et à droite de la fenêtre du navigateur.

Assurez-vous de changer de nouveau à *center*. Les deux dernières options ne sont utiles que lorsque vous avez plusieurs éléments flexibles dans un conteneur.

Distribution de plusieurs éléments flex

«Et alors ?» pourriez dire... nous pouvons déjà le faire par l'alignement gauche/droite avec des float et le centrage automatiques des marges. Vrai. Flexbox ne montre pas toutes ses possibilités jusqu'à ce que nous ayons

plus d'un élément dans un conteneur. La propriété *justify-content* vous permet également de distribuer des éléments à l'intérieur d'un conteneur.



SPACE-AROUND



SPACE-BETWEEN

Changer notre règle *.menu* pour correspondre à ce qui suit:

```
.menu {  
  border: 1px solid #fff;  
  width: 900px;  
  display: flex;  
  justify-content: space-around;  
}
```

Cela transforme notre *.menu* en un conteneur flexible imbriqués, et la valeur *space-around* fait que ses éléments se propagent sur toute sa largeur. Vous devriez voir quelque chose comme ceci:

Le conteneur flexible distribue automatiquement l'espace horizontal supplémentaire de chaque côté de chaque élément.

La valeur *space-between* est similaire, mais il n'ajoute que l'espace supplémentaire *entre* les éléments. Mettons à jour la ligne *justify-content*:

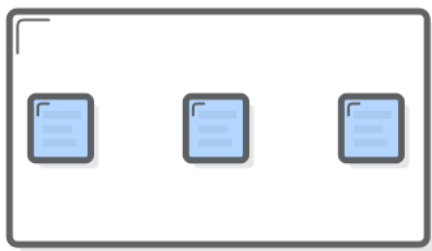


Bien sûr, vous pouvez également utiliser ici *center*, *flex-start*, *flex-end* si vous voulez pousser tous les éléments d'un côté ou un autre, mais laissons comme valeur *space-between*.

Regroupement des éléments flex

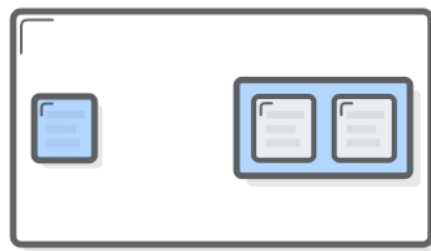
Les conteneurs flex ne savent que positionner des éléments de niveau 1 (c'est-à-dire, leurs éléments directement enfants). Ils ne se soucient pas de ce qui est à l'intérieur de chacun de ses éléments flex. Cela signifie que le regroupement des éléments flex est une autre arme dans votre arsenal de création de mise en page.

Enveloppons un tas d'éléments dans une `<div>` amènera à un résultat totalement différent dans notre page Web.



NO GROUPING

(3 FLEX ITEMS)



GROUPED ITEMS

(2 FLEX ITEMS)

Par exemple, disons que vous voulez à la fois les liens de connexion et d'inscription calés sur le côté droit de la page, comme dans la capture d'écran ci-dessus. Tout ce que nous devons faire est de les mettre dans une autre `<div>`:

```
<div class="menu-container">
  <nav class="menu">
    <div class="date">20 mars 2017</div>
    <div class="links">
      <div class="signup">S'inscrire</div>
      <div class="login">Se connecter</div>
    </div>
  </nav>
</div>
```

Au lieu d'avoir trois éléments, notre `.menu` conteneur de flex a maintenant seulement deux éléments (`.date` et `.links`). Sous l'actuel comportement *space-between*, ils se calent sur le côté gauche et à droite de la page.

Mais, maintenant , nous devons mettre en forme l'élément `.links` parce qu'il utilise le mode de mise en page de bloc par défaut. La solution: des conteneurs flexibles imbriqués !

Ajouter une nouvelle règle à notre fichier `styles.css` qui transforme l'élément `.links` en un conteneur flexible:

```
.links {  
  border: 1px solid #fff;  
  /* Pour debug */  
  display: flex;  
  justify-content: flex-end;  
}  
  
.login {  
  margin-left: 20px;  
}
```

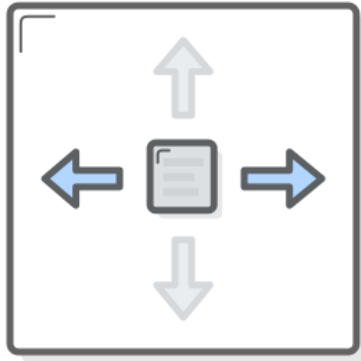
Cela mettra nos liens là où nous voulons. Notez que les marges continuent de fonctionner comme dans le modèle de boîte CSS. Et, comme avec le modèle de boîte normal, les marges auto ont une signification particulière dans FlexBox (nous allons laisser cela pour la fin du chapitre cependant).

Nous n'aurons plus besoin de ces bordures blanches, de sorte que vous pouvez aller de l'avant et les supprimer si vous le souhaitez.

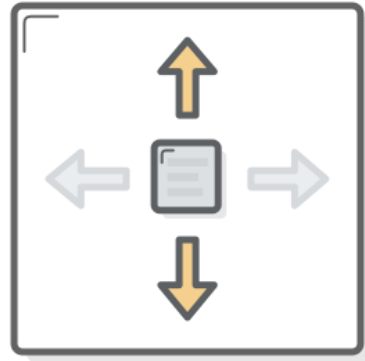
Cross-axis l'alignement vertical

Jusqu'à présent, nous avons manipulons alignement horizontal, mais les conteneurs Flex peut également définir l'alignement vertical de leurs

éléments. Ceci est quelque chose qui est tout simplement pas possible avec des float.



JUSTIFY-CONTENT



ALIGN-ITEMS

Pour montrer cela, nous avons besoin d'ajouter un en-tête sous notre menu. Ajouter le balisage suivant dans le fichier *flexbox.html* après l'élément *.menu-container*:

```
<div class="header-container">
  <div class="header">
    <div class="subscribe">Souscrire &#9662;</div>
    <div class="logo"></div>
    <div class="social"></div>
  </div>
</div>
```

Ensuite, ajoutez quelques styles de base pour l'obtenir aligné avec notre élément *.menu* :

```
.header-container {  
  color: #5995DA;  
  background-color: #D6E9FE;  
  display: flex;  
  justify-content: center;  
}  
  
.header {  
  width: 900px;  
  height: 300px;  
  display: flex;  
  justify-content: space-between;  
}
```

Tout cela devrait être familier. Cependant, le scénario est un peu différent de celui de notre menu. Puisque `.header` a une hauteur explicite, les éléments peuvent être positionnés verticalement à l'intérieur de celui-ci.

La spécification officielle appelle cet alignement "axe transversal" *-cross-axis-* (nous verrons pourquoi dans un instant), mais pour nos fins, il pourrait aussi bien être appelé alignement "vertical".



L'alignement vertical est défini par l'ajout d'une propriété *align-items* à un conteneur flexible. Pour que notre page d'exemple corresponde à la capture d'écran ci-dessus ajoutons la ligne suivante:

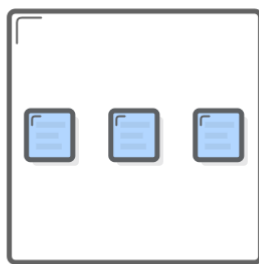

```
.header {  
  width: 900px;  
  height: 300px;  
  display: flex;  
  justify-content: space-between;  
  align-items: center; /* Nouveau */  
}
```

Les options disponibles pour *align-items* sont similaire à *justify-content*:

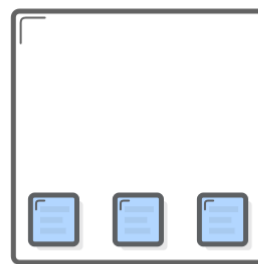
- *center*
- *flex-start*
- *flex-end*
- *stretch*



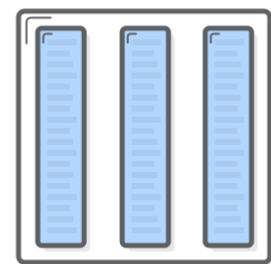
FLEX-START



CENTER



FLEX-END



STRETCH

La plupart d'entre elles sont assez simples. L'option *stretch*, par défaut, tasse tous les éléments flex de manière à emplir le conteneur parent dans le sens de l'axe en croix. Si le parent n'a pas de largeur définie dans la direction de l'axe en croix, alors tous les éléments flex auront la longueur du plus long des éléments. Jetons un bref coup d'oeil en ajoutant ce qui suit à *styles.css*:



La boîte pour chaque article s'étend sur toute la hauteur du conteneur flexible, quelle que soit la quantité de contenu qu'il contient. Un cas d'utilisation courant pour ce comportement est de créer des colonnes de même hauteur avec une quantité variable de contenu dans chaque, ce qui est quelque chose de très difficile à faire avec des *float*.

Assurez-vous de supprimer les changements ci-dessus et centrer verticalement votre contenu dans *.header* avant de passer à la suite.

Ecoulement des éléments flex

Flexbox est une alternative plus puissante pour faire des grilles. Non seulement il peut afficher les éléments sous forme de grille mais il peut changer leur alignement, la direction, l'ordre et la taille, aussi. Pour créer une grille, nous avons besoin de la propriété *flex-wrap*.



NO WRAPPING

FLEX-WRAP: NOWRAP;



WITH WRAPPING

FLEX-WRAP: WRAP;

Ajouter une liste d'images à notre fichier *flexbox.html* de sorte à ce que nous ayons un peu de contenu à travailler.

Il sera positionné à l'intérieur de `<body>`, sous l'élément `.header-container`:

```
<div class="photo-grid-container">
  <div class="photo-grid">
    <div class="photo-grid-item first-item">
      
    </div>
    <div class="photo-grid-item">
      
    </div>
    <div class="photo-grid-item">
      
    </div>
  </div>
</div>
```

Encore une fois, le CSS correspondant doit être familier:



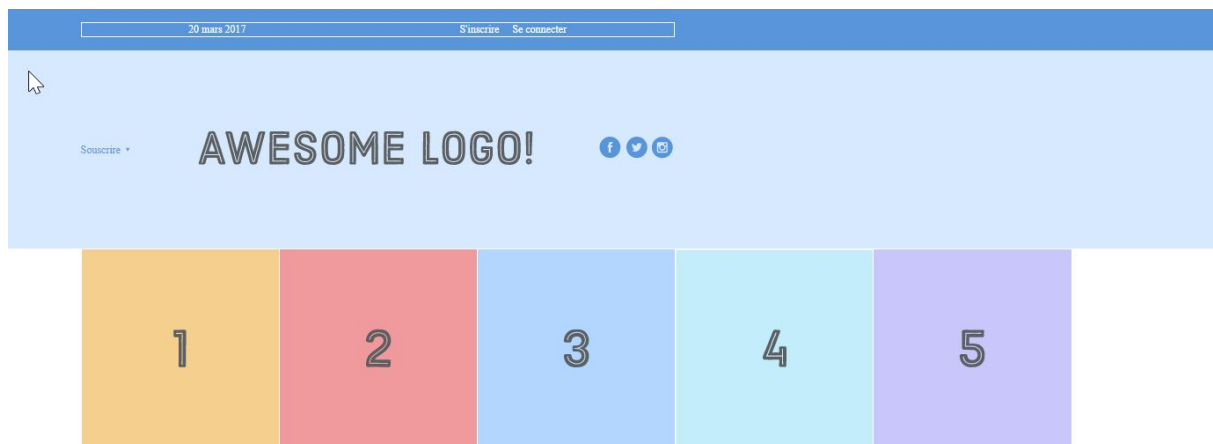
```
.photo-grid-container {  
  display: flex;  
  justify-content: center;  
}  
  
.photo-grid {  
  width: 900px;  
  display: flex;  
  justify-content: flex-start;  
}  
  
.photo-grid-item {  
  border: 1px solid #fff;  
  width: 300px;  
  height: 300px;  
}
```

Cela devrait fonctionner comme prévu, mais regarder ce qui se passe lorsque l'on ajoute plus d'éléments que ne peut contenir le conteneur flexible.

Insérez deux photos dans l'élément *.photo-grid* :

```
<div class="photo-grid-item">
  
</div>
<div class="photo-grid-item last-item">
  
</div>
```

Par défaut, ils écoulent sur le bord de la page:



Si vous essayez de construire un bandeau qui permet à l'utilisateur de faire défiler horizontalement des photos, cela pourrait être le comportement souhaité, mais ce n'est pas ce que nous voulons.

En ajoutant la propriété *flex-wrap* cela va forcer les éléments qui débordent à passer à la ligne suivante:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains CSS code for a class named .photo-grid.

```
.photo-grid {  
  width: 900px;  
  display: flex;  
  justify-content: flex-start;  
  flex-wrap: wrap;  
}
```

Maintenant, nos éléments flex se comportent un peu comme des boîtes *float*, sauf que FlexBox nous donne plus de contrôle sur la façon dont les éléments "en trop" seront positionnés sur la dernière ligne via la propriété *justify-content*.

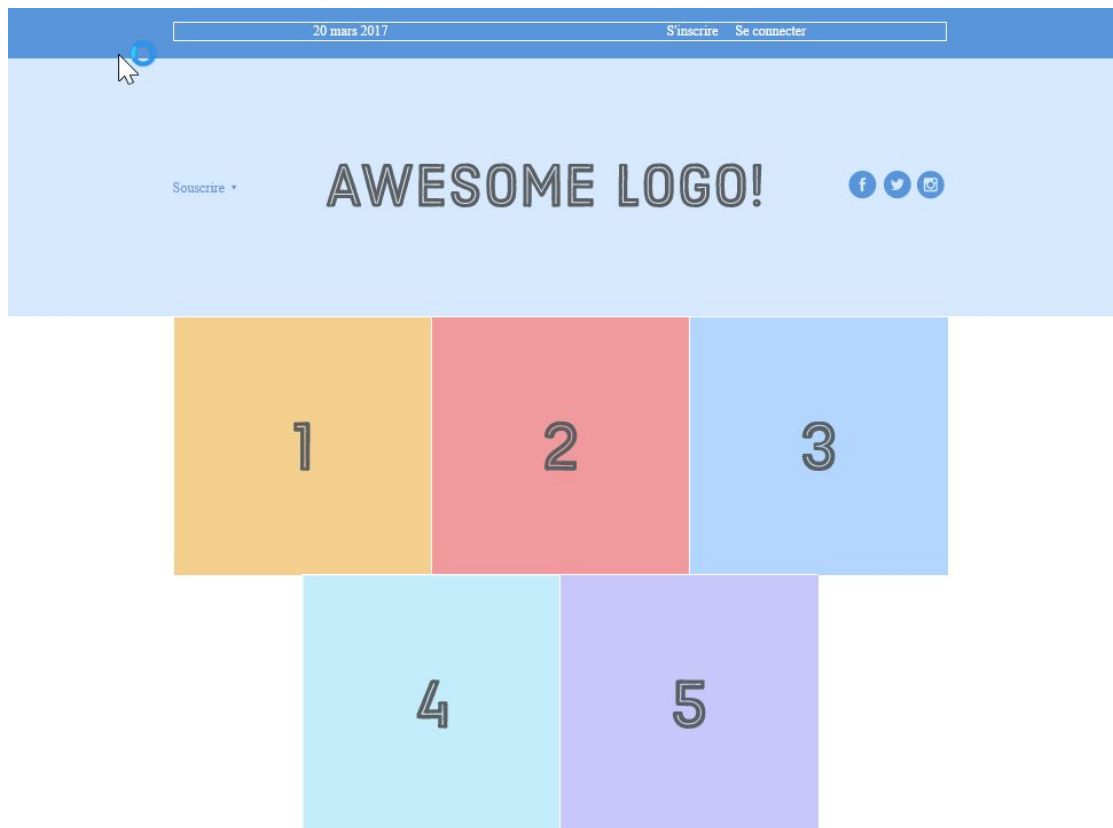
Par exemple, la dernière ligne est actuellement alignée à gauche.

Essayez de la centrer en modifiant notre règle *.photo-grid* comme suit:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains updated CSS code for the class .photo-grid, changing the justify-content property to center.

```
.photo-grid {  
  width: 900px;  
  display: flex;  
  justify-content: center;  
  /* Changé */  
  flex-wrap: wrap;  
}
```

La réalisation de cet mise en page avec une configuration à base de *float* est très compliquée.



Direction et container flex

La "direction" permet de savoir si un container affiche ses éléments horizontalement ou verticalement. Jusqu'à présent, tous les conteneurs que nous avons vu utilisent la direction horizontale par défaut, ce qui signifie que les éléments sont positionnés l'un après l'autre sur la même ligne avant de se décaler vers le bas à la colonne suivante quand ils arrivent hors de l'espace.



ROW

FLEX-DIRECTION: ROW;



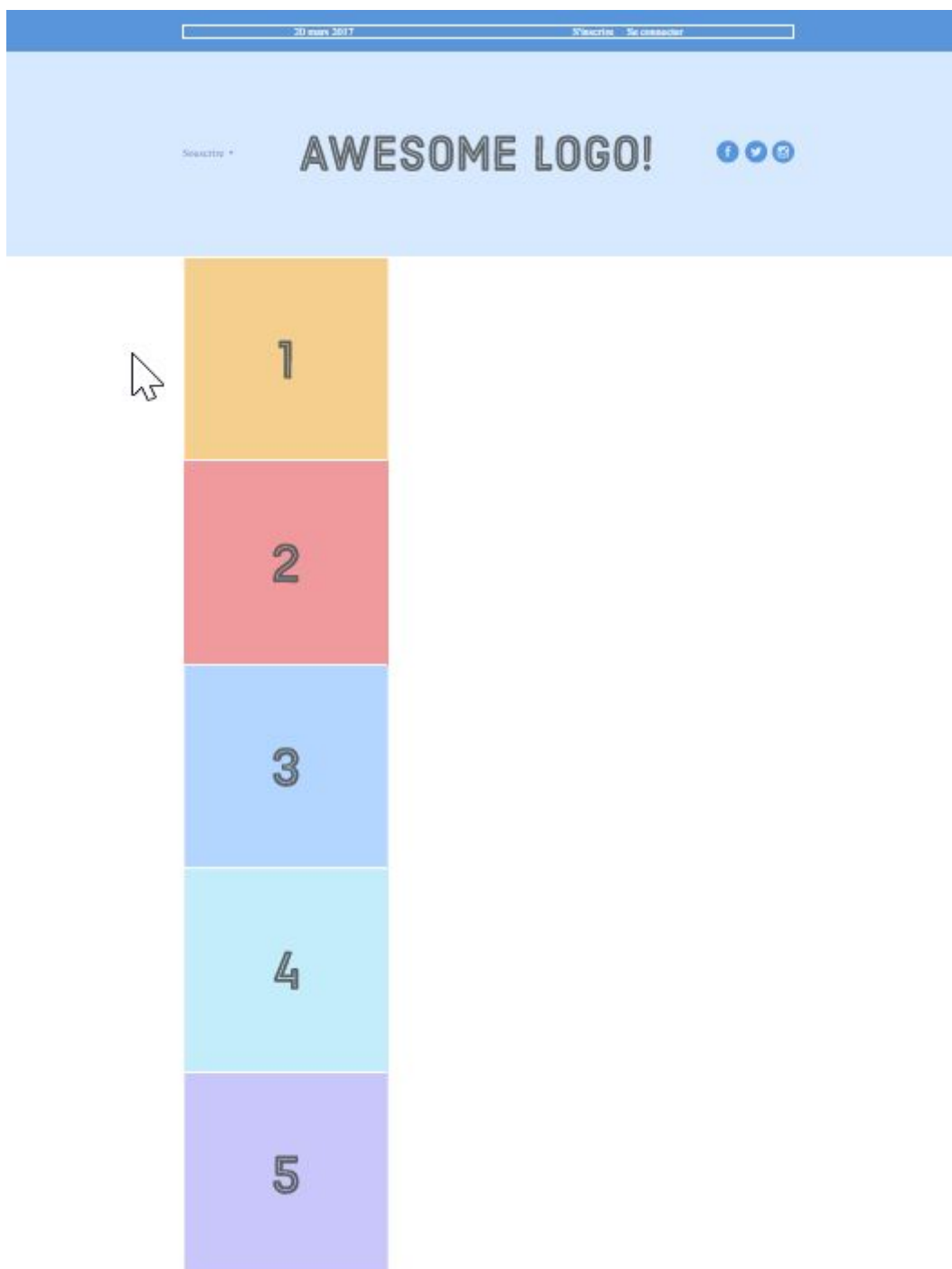
COLUMN

FLEX-DIRECTION: COLUMN;

L'une des choses les plus étonnantes au sujet de FlexBox est sa capacité à transformer les lignes en colonnes en utilisant seulement une seule ligne de CSS.

Ajouter la déclaration suivante *flex-direction* à notre règle *.photo-grid*:

```
.photo-grid {  
  width: 900px;  
  display: flex;  
  justify-content: center;  
  flex-wrap: wrap;  
  /* Nouveau */  
  flex-direction: column;  
}
```

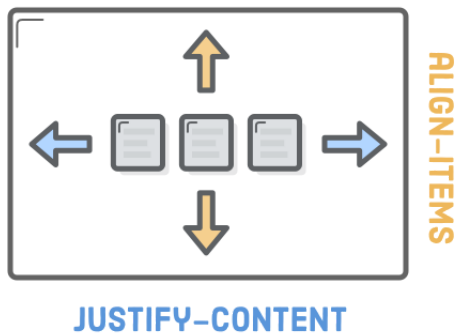
Cela change la direction du conteneur qui pour l'instant avait la valeur par défaut à savoir la valeur *row*. Au lieu d'une grille, notre page a maintenant une seule colonne verticale.

Un des éléments clé du “responsive design” est de présenter le même balisage HTML pour les utilisateurs quelque soit l'appareil utilisé -téléphone, tablette ou ordinateur de bureau. Cela pose un petit problème, car la plupart des configurations mobiles sont sur une seule colonne, tandis que la plupart des dispositions de bureau vont empiler des éléments horizontalement. Vous pouvez imaginer à quel point sera utile la propriété `flex-direction` une fois que nous commencerons à parler de site responsive.

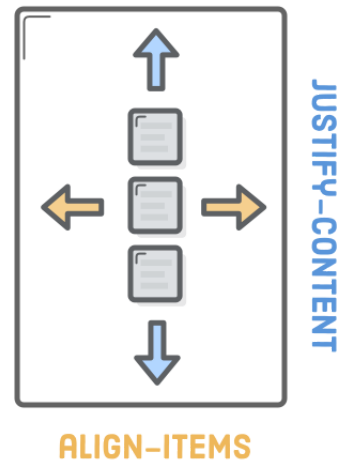
Considérations sur l'alignement

Notez que la colonne est contrainte sur le côté gauche de son conteneur flexible en dépit de notre déclaration `justify-content: center;`. Lorsque vous changez la direction d'un conteneur, vous tournez également le sens de la propriété `justify-content`. Il se réfère maintenant à un alignement vertical et pas à l'alignement horizontal du conteneur.

FLEX-DIRECTION: ROW;



FLEX-DIRECTION: COLUMN;



Pour centrer horizontalement notre colonne, nous avons besoin de définir une propriété `align-items` sur notre `.photo-grid`:



```
.photo-grid {  
  width: 900px;  
  display: flex;  
  justify-content: center;  
  flex-wrap: wrap;  
  flex-direction: column;  
  /* Nouveau */  
  align-items: center;  
}
```

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;

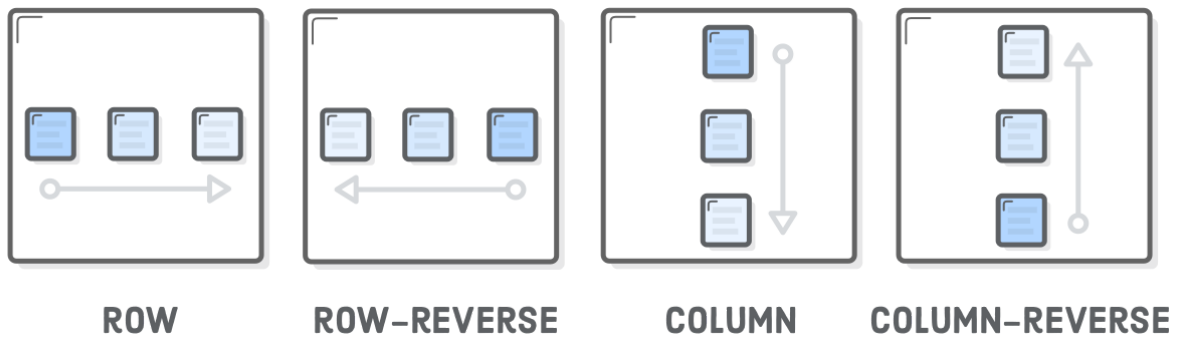
flex-wrap: wrap;

par

flex-flow: row wrap;

Ordre et flex container

Jusqu'à présent, il y a eu une corrélation étroite entre l'ordre de nos éléments HTML et la façon dont les boîtes sont rendus dans une page Web. Avec float ou les techniques flexbox que nous avons vu jusqu'à présent, la seule façon de faire pour qu'une boîte apparaisse avant ou après une autre est de déplacer le balisage HTML, modifier l'ordre des balises dans. Flexbox nous permet cela et sans toucher au html !



La propriété *flex-direction* propose également le contrôle de l'ordre dans lequel les éléments apparaissent via les valeurs *row-reverse* et *column-reverse*. Pour le voir en action, nous allons transformer notre colonne pour en faire une grille, mais cette fois-ci, nous allons inverser l'ordre avec *row-reverse* :

```
.photo-grid {  
  width: 900px;  
  display: flex;  
  justify-content: center;  
  flex-wrap: wrap;  
  align-items: center;  
  /* Nouveau et intéressant ! */  
  flex-direction: row-reverse;  
}
```

Les deux lignes sont maintenant affichées de droite à gauche au lieu de gauche à droite. Mais, remarquez comment cela ne permute l'ordre sur une base par ligne: la première ligne ne commence pas à 5, elle commence à 3. Ce comportement est utile pour un grand nombre de modèles de conception courants (*column-reverse* en particulier ouvre un grand nombre de possibilités pour les agencements mobiles).



Réorganiser les éléments à l'intérieur d'une feuille de style est une grosse affaire. Avant FlexBox, les développeurs web devaient recourir à des hacks JavaScript pour accomplir ce genre de chose. Cependant, il ne faut pas abuser de ces capacités nouvelles, *de grands pouvoirs entraînent de grandes responsabilités !*

Comme nous l'avons déjà évoqué, vous devez toujours séparer le contenu de la présentation. Une modification de l'ordre comme celui-ci est purement fait pour la présentation, votre HTML doit encore faire sens même si ces styles ne sont pas appliqués.

Ordre des éléments flex

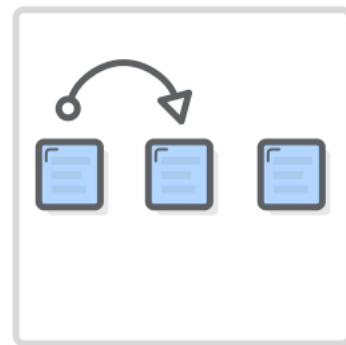
Jusqu'à présent nous avons parlé principalement du positionnement des éléments flexibles dans leurs conteneurs parents mais il est également possible de manipuler les éléments contenu dans le parent.

Le reste de ce chapitre va moins porter sur les conteneurs flexibles que sur les éléments qu'ils contiennent.



FLEX-DIRECTION

(WHOLE CONTAINER)



ORDER

(INDIVIDUAL ITEMS)

L'ajout d'une propriété *order* à un élément flex définit l'ordre dans le conteneur sans affecter les éléments environnants.

Voyons comment cela fonctionne dans le détail :

- par défaut, tous les éléments *flex* ont une valeur *order* égale à 0.
- les éléments *flex* avec des valeurs de *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 du 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 premier.

Vous pouvez donner des valeurs négatives à `order` pour faire en sorte que ces éléments soient affichés avant les éléments d'ordre 0.

Sa valeur par défaut est 0, et en augmentant ou en diminuant la valeur, on va faire que l'élément se déplace respectivement à droite ou à gauche.

Cela peut être utilisé, par exemple, pour échanger l'ordre des éléments `.first-item` et les `.last-item` de notre grille. Nous devons également changer la valeur `row-reverse` de la section précédente à `row` parce que ça va rendre nos modifications un peu plus facile à voir:




```
.photo-grid {  
  width: 900px;  
  display: flex;  
  justify-content: center;  
  flex-wrap: wrap;  
  align-items: center;  
  flex-direction: row; /* Mettre à jour */  
}  
  
.first-item {  
  order: 1;  
}  
  
.last-item {  
  order: -1;  
}
```

Contrairement à la définition `row-reverse` et `column-reverse` sur un conteneur flexible, `order` fonctionne à travers les lignes et les colonnes. L'extrait ci-dessus inversera nos premiers et derniers éléments, même s'ils apparaissent sur des lignes différentes.

Alignement d'un élément flex

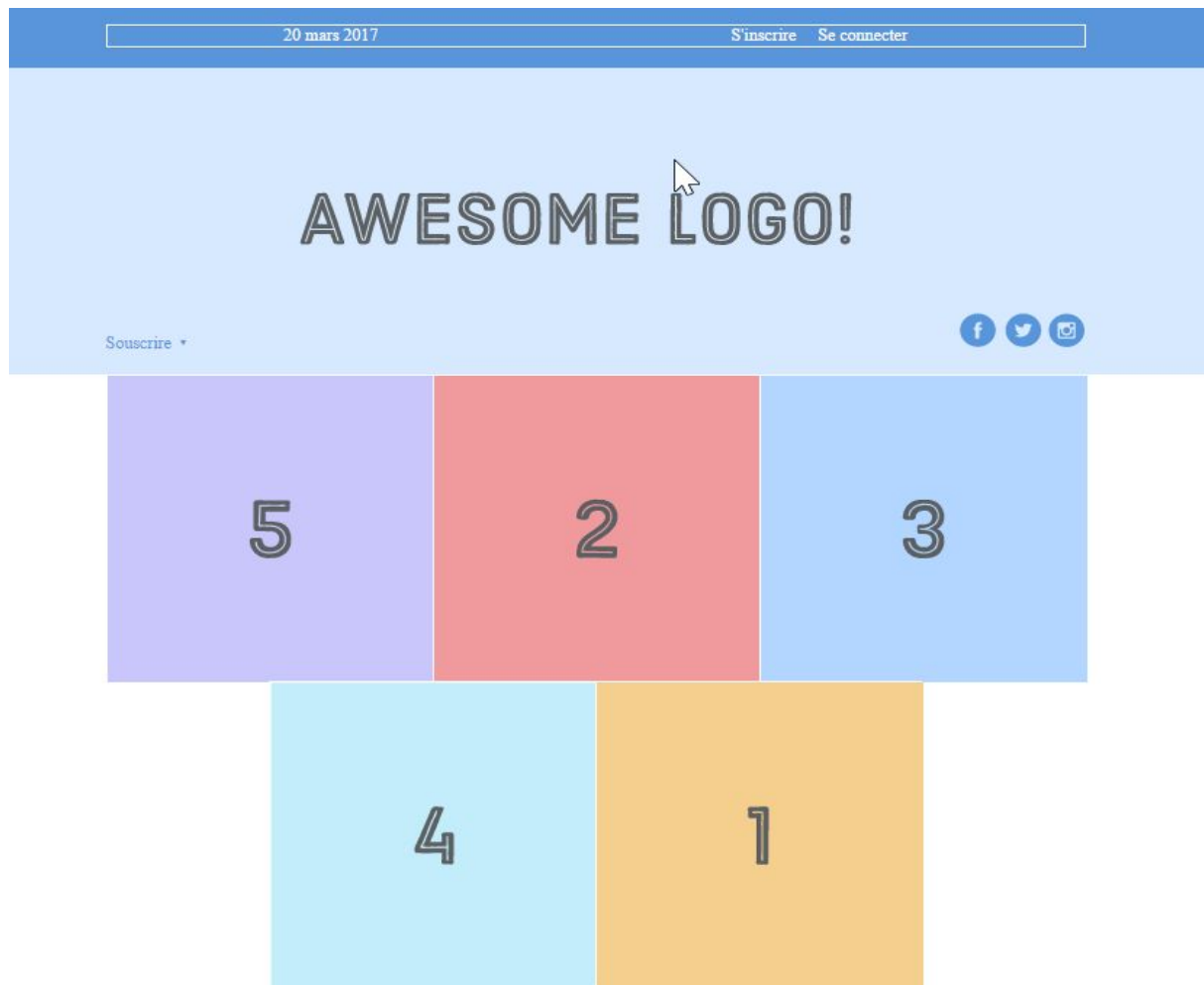
Nous pouvons faire la même chose avec un alignement vertical. Que faire si nous voulons que le lien “*S’inscrire*” et les icônes des réseaux sociaux se positionne en bas de l'en-tête au lieu du centre ?

Alignez-les individuellement ! C'est là que la propriété *align-self* entre en jeu. L'ajout de ceci à un élément flex surcharge la valeur *align-items* de son conteneur:



```
.social,  
.subscribe {  
  align-self: flex-end;  
  margin-bottom: 20px;  
}
```

Cela devrait les envoyer en bas du *.header*. Notez que les marges (*padding*, *margin*) fonctionnent exactement comme on pouvait s'y attendre.



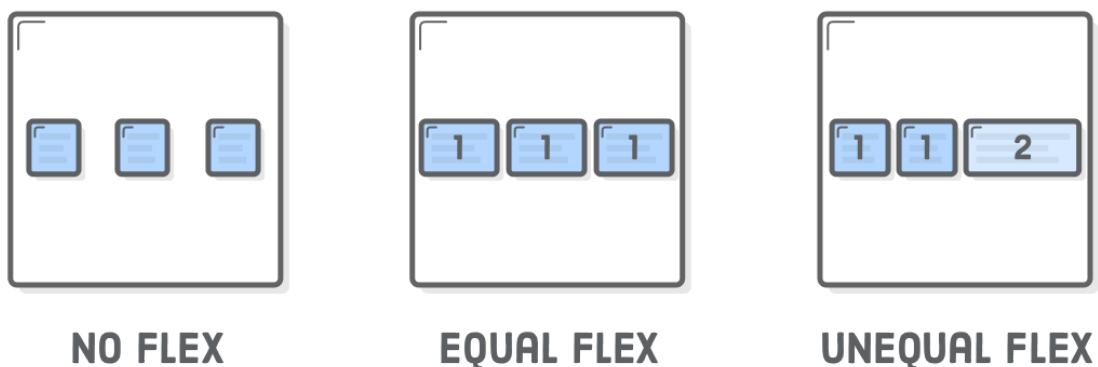
Vous pouvez aligner les éléments de différentes façons en utilisant les mêmes valeurs que la propriété *align-items*, énumérés ci-dessous pour plus de commodité.

- *center*
- *flex-start*
- *flex-end*
- *stretch*

Eléments flexibles

Tous nos exemples ont tourné autour d'éléments de largeurs fixes ou définies par le contenu. Cela nous a permis de nous concentrer sur les aspects de positionnement de flexbox, mais nous avons pour le moment ignoré sa nature éponyme de "boîte flexible". Les éléments flex sont *flexibles* cela veut dire qu'ils peuvent rétrécir et s'étirer pour correspondre à la largeur de leurs conteneurs.


La propriété *flex* définit la largeur des éléments individuels dans un conteneur flexible. Ou, plus exactement, il leur permet d'avoir des largeurs flexibles. Il fonctionne comme un poids qui indique au conteneur flexible comment distribuer l'espace supplémentaire pour chaque élément. Par exemple, un élément avec une valeur *flex* de 2 va croître deux fois plus vite que les éléments avec la valeur par défaut de 1.



Tout d'abord, nous avons besoin d'un pied de page pour expérimenter. Collez ceci après l'élément `.photo-grid-container`:

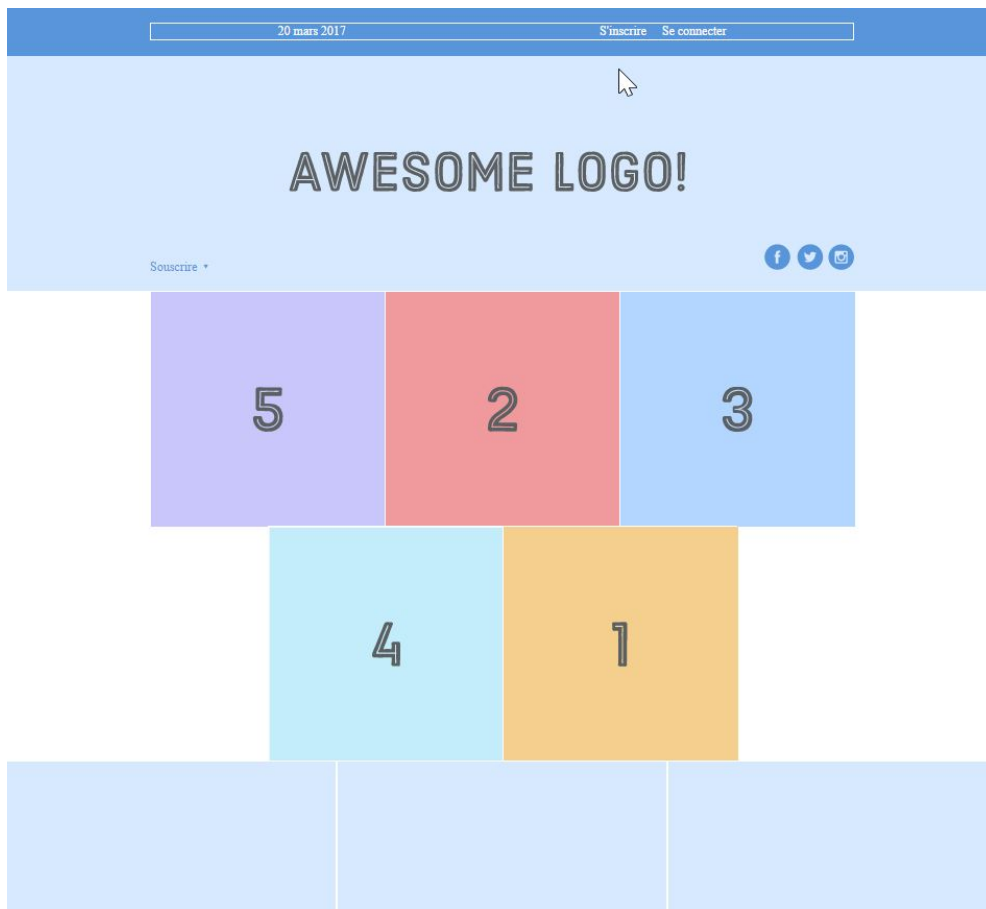
```
<div class="footer">
  <div class="footer-item footer-one"></div>
  <div class="footer-item footer-two"></div>
  <div class="footer-item footer-three"></div>
</div>
```

Puis, un peu de CSS:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains CSS code for a footer and its items.

```
.footer {  
  display: flex;  
  justify-content: space-between;  
}  
  
.footer-item {  
  border: 1px solid #fff;  
  background-color: #D6E9FE;  
  height: 200px;  
  flex: 1;  
}
```

Ce *flex: 1;* ligne indique que les éléments vont s'étirer pour correspondre à la largeur du *.footer*. Comme ils ont tous le même poids, ils vont étirer de façon égale:



Augmenter le poids de l'un des éléments pour le faire plus large que les autres. Par exemple, nous pouvons faire en sorte que le troisième élément prenne une largeur deux fois plus grande, facteur d'agrandissement de 2, que les deux autres avec la règle suivante:



Comparez cela à la propriété *justify-content*, qui distribue l'espace supplémentaire *entre* les éléments. Ceci est similaire, mais maintenant nous allons distribuer cet espace dans les éléments eux-mêmes. Le

résultat est un contrôle complet sur la façon dont les éléments flexibles s'intègrent dans leurs conteneurs

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



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/réactivité — si vous redimensionnez la fenêtre du navigateur ou ajoutez un autre élément, la mise en page continue de fonctionner correctement.

flex : forme abrégée vs longue

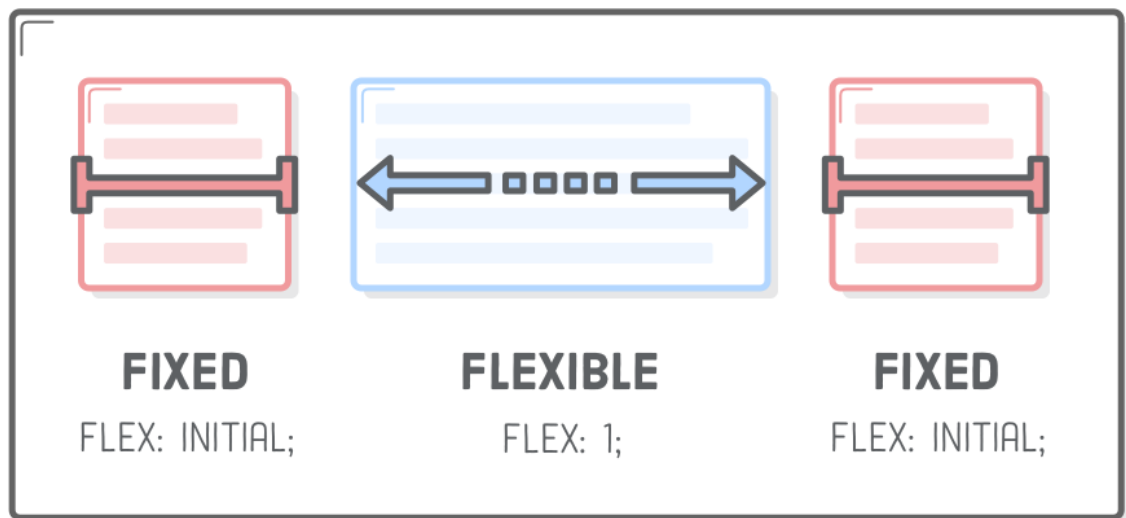
flex est une forme abrégée de propriété qui peut servir à définir trois valeurs différentes :

- la valeur de proportion sans unité vue ci-dessus. Elle peut être précisée seule avec la forme longue de la propriété *flex-grow*.
- une deuxième valeur de proportion sans unité — *flex-shrink* — intervenant quand les éléments *flex* débordent du conteneur. Elle indique la quantité de débordement à retirer de la taille de chacun des éléments flex pour les empêcher de déborder du conteneur. Il s'agit d'une fonctionnalité avancée de flexbox, et nous n'en parlerons plus dans cet article.
- la valeur de taille minimale vue ci-dessus. Elle peut être précisée seule avec la forme longue de la propriété *flex-basis*.

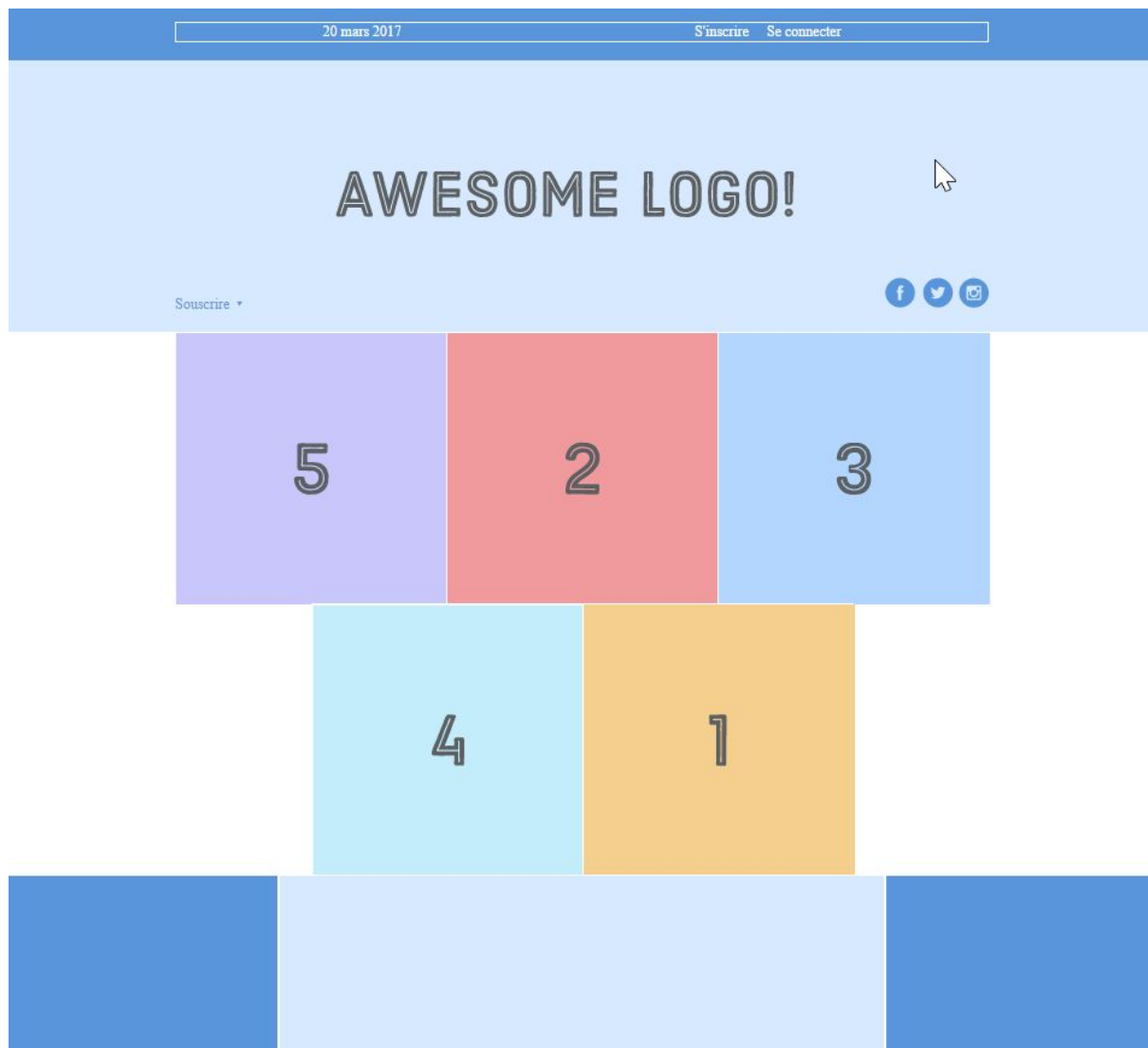
Nous vous déconseillons d'utiliser les propriétés flex sous leur forme longue sauf impossibilité alternative (par exemple, pour annuler quelque chose déjà défini). Elles amènent l'écriture de code supplémentaire et peuvent être quelque peu déroutantes.

Largeurs des éléments fixes

Nous pouvons même mélanger et assortir boîtes flexibles avec celles de largeur fixe. *flex: initial* permet à l'élément de prendre la valeur explicite de la propriété *width*. Cela nous permet de combiner des boîtes fixes et flexibles de façon complexe.



Nous allons faire en sorte que notre pied de page se comporte comme le schéma ci-dessus. L'élément central est flexible, mais celles de chaque côté sont toujours les mêmes dimensions. Tout ce que nous devons faire est d'ajouter la règle suivante à notre feuille de style :



```
.footer-one,  
.footer-three {  
  background-color: #5995DA;  
  flex: initial;  
  width: 300px;  
}
```


Sans cette ligne *flex: initial;*, la déclaration *flex: 1;* sera héritée de la règle *.footer-item*, ce qui fait que la propriété *width* va être ignorées. La valeur *initial* corrige cela, et nous obtenons une mise en page flexible qui contient également des éléments de largeur fixe. Lorsque vous redimensionner la fenêtre du navigateur, vous verrez que seule la case du milieu dans le pied se redimensionner.

Ceci est une disposition assez commune, et pas seulement pour les pieds de page. Par exemple, de nombreux sites ont une largeur fixe pour la barre latérale (ou plusieurs barres latérales) et un bloc de contenu flexible contenant le texte principal de la page. Ceci est essentiellement une version plus grande du pied de page que nous venons de créer.

Elément flex et marges “auto”

Les marges “*auto*” dans FlexBox sont spéciales. Elles peuvent être utilisées comme une alternative à un `<div>` supplémentaire pour permettre d'aligner un groupe d'éléments vers la gauche ou la droite d'un conteneur. Pensez marges “*auto*” comme étant un “diviseur” pour les éléments flexibles dans le même conteneur.

Jetons un coup d'oeil en “aplatissant” les éléments de notre *.menu* de sorte à ce qu'il corresponde à ce qui suit. Nous allons supprimer l'élément *.link*.



```
<div class="menu-container">
  <nav class="menu">Menu</nav>
  <div class="date">20 mars 2017</div>
  <div class="signup">S'inscrire</div>
  <div class="login">Se connecter</div>
</div>
```

Recharger la page devrait faire que les éléments sont répartis également grâce à notre `.menu`, tout comme au début. Nous pouvons reproduire la disposition souhaitée en collant une marge gauche à auto entre les éléments que nous voulons séparer, comme ceci:



Les marges auto “mangent” tout l'espace supplémentaire dans un conteneur flexible, donc au lieu de distribuer de façon égale tous les éléments, cela déplace `.signup` et tous les éléments suivants (`.login`) sur le côté droit du conteneur. Cela vous donnera exactement la même disposition que nous avons avant, mais sans ce `<div class="link">` supplémentaire imbriqué pour les regrouper.

Il est toujours bon de garder son HTML le plus simple et concis possible.

Résumé

Flexbox nous donne une tonne d'incroyables nouveaux outils pour la mise en forme d'une page web. Comparez ces techniques à ce que nous avons pu faire avec des float, et il devrait être assez clair que FlexBox est une option plus propre pour la pose des sites web modernes:

- Utilisez `display: flex;` pour créer un conteneur flexible.
- Utilisez `justify-content` pour définir l'alignement horizontal des éléments.
- Utilisez `align-items` pour définir l'alignement vertical des éléments.
- Utilisez `flex-direction` si vous avez besoin des colonnes au lieu de lignes.

- Utilisez les valeurs *row-reverse* ou *column-reverse* pour retourner l'élément.
- Utilisez *order* pour personnaliser l'ordre des éléments individuels.
- Utilisez *align-self* pour aligner verticalement des éléments individuels.
- Utilisez *flex* pour créer des boîtes flexibles qui peuvent étirer et rétrécir.

Rappelez-vous que ces propriétés de flexbox sont juste une langue qui vous permet de dire aux navigateurs comment organiser un tas d'éléments HTML. La partie difficile est pas réellement d'écrire le code HTML et CSS, il est de déterminer, sur le plan conceptuel (sur un morceau de papier), le comportement de toutes les cases nécessaires pour créer une mise en page donnée.

Quand un concepteur vous remet une maquette à mettre en œuvre, votre première tâche est d'en déduire un tas de boîtes sur elle et de déterminer comment elles sont censées s'empiler, s'étirer et rétrécir pour réaliser la conception souhaitée. Une fois que vous avez fait cela, il devrait être assez facile à coder en utilisant ces nouvelles techniques de flexbox.

Le mode de mise en page de FlexBox doit être utilisé pour la plupart de vos pages Web, mais il y a certaines choses pour lequel il n'est pas si bon, comme par exemple peaufiner les positions des éléments.

Pour expérimenter, réviser, tout apprendre sur toutes ces propriétés flexbox consultez cette revue du web:

- <http://yoksel.github.io/flex-cheatsheet/>
- <http://bennettfeely.com/flexplorer/>
- <http://codepen.io/enxaneta/details/adLPwv>
- <http://jackintheflexbox.tumblr.com/>
- <https://flexboxfroggy.com/>

<https://medium.freecodecamp.com/an-animated-guide-to-flexbox-d280cf6afc35#.gnmc07rbs>