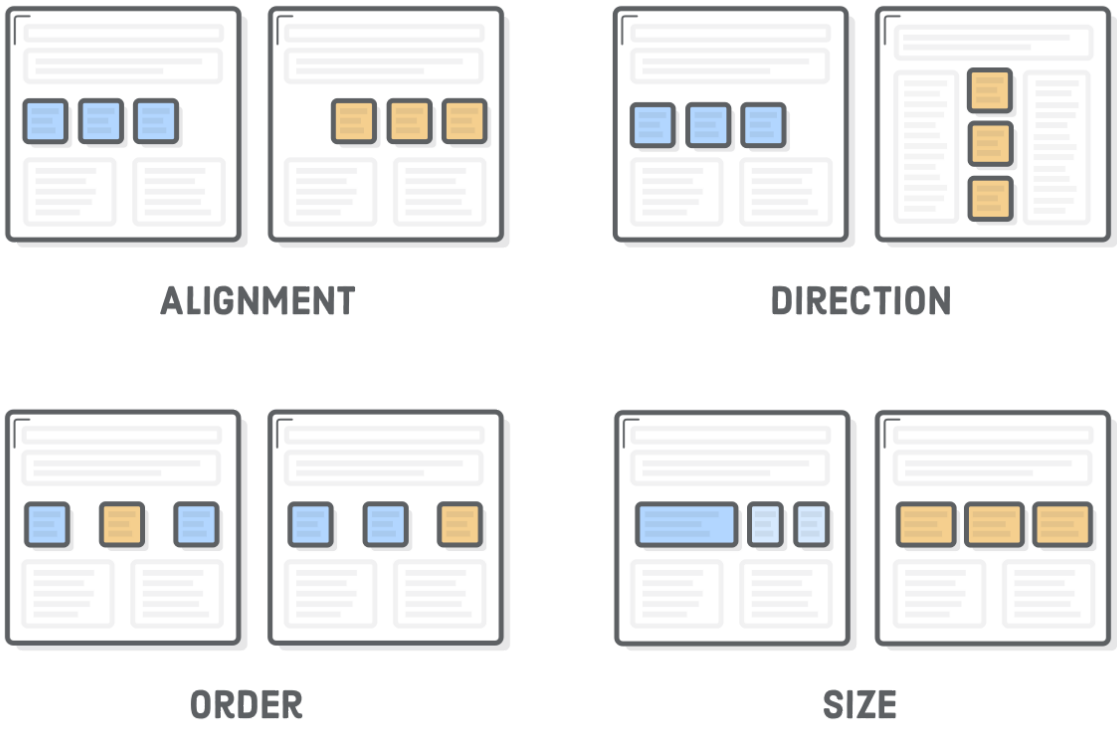
flexbox

La "Box 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.

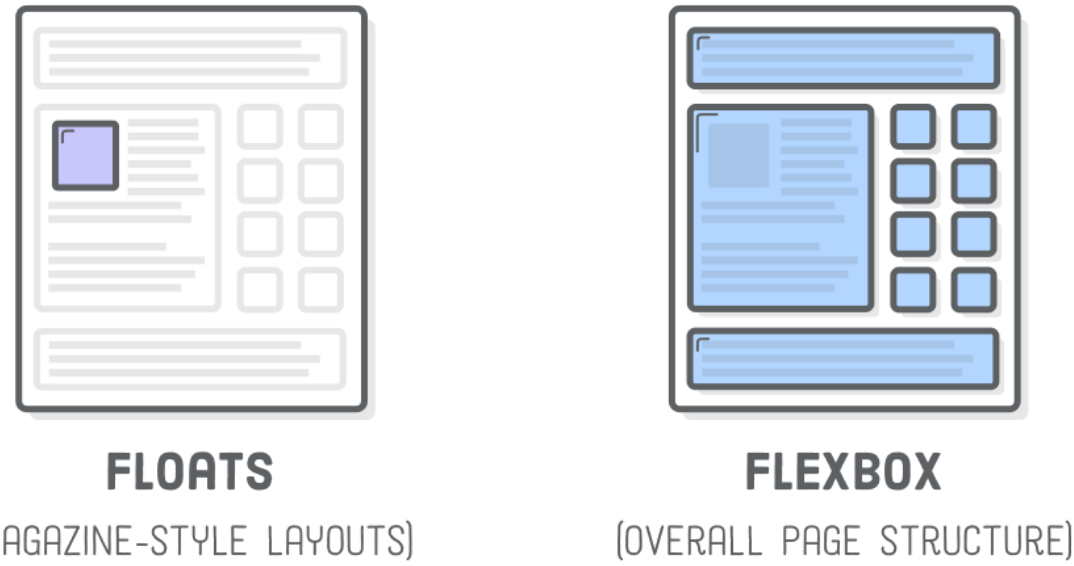


Le web connaît actuellement une transition majeure, alors une petite discussion autour de l'état de l'industrie est justifiée. Dans la dernière décennie, les float étaient la seule option pour une mis page web complexe. En conséquence, toutes ces règles sont bien pris en charge dans les navigateurs existants, et les développeurs les ont utilisés 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 limitée. 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.



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:

Pour commencer, nous avons besoin d'un document HTML vide qui ne contient rien d'autre qu'une une barre de menu. Faire un nouveau projet Atom 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:

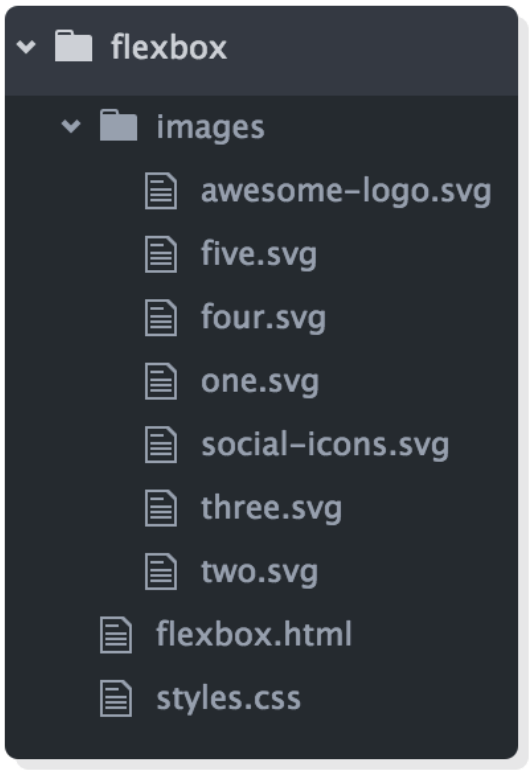
```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <title>Flexbox</title>
  <link rel="stylesheet" href="styles.css">
 </head>
 <body>
  <div class="menu-container">
   <div class="menu">
    <div class="date">20 mars 2017</div>
    <div class="signup">S'inscrire</div>
    <div class="login">Se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

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 marge 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 tas d'éléments flexibles ensemble et de définir la façon dont ils sont positionnés.

Chaque élément HTML qui est un enfant direct d'un conteneur flexible "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’emboitement. 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.

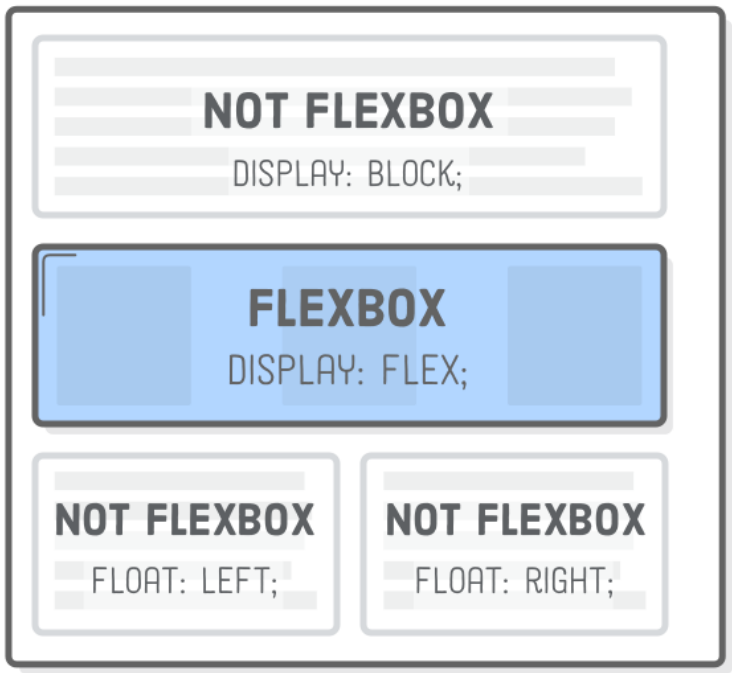
flex container

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 le 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:

```
.menu-container {
  /* ... */
  display: flex;
}
```

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 float).



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.

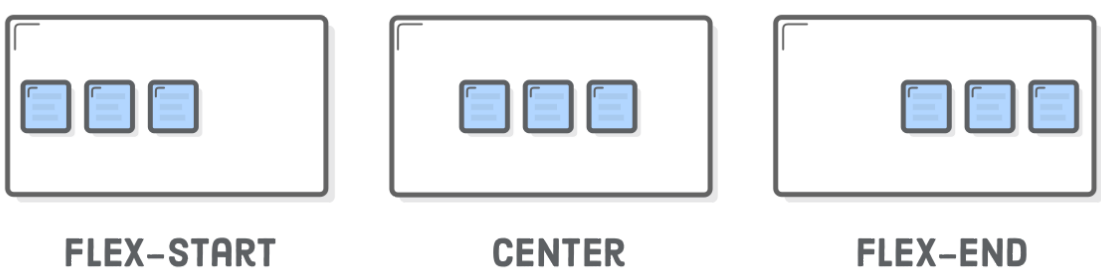
Notez que les propriétés columns- du module multi-colonnes n'ont pas d'effet sur un container flex ainsi que float, clear et vertical-align n'ont pas d'effet sur un élément flex.

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

```
.menu-container {
  /* ... */
  display: flex;
  justify-content: center;
}
```

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 une divergence par rapport à la façon dont nous avons fait cela dans avec le positionnement des boîtes jusqu'à présent.



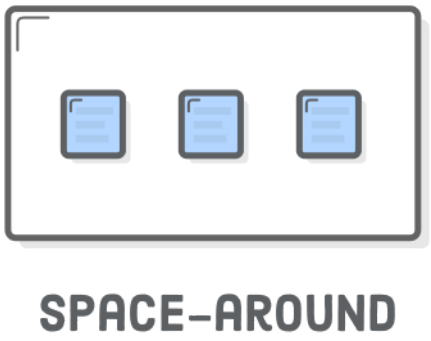
Les autres valeurs pour justify-content sont :

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

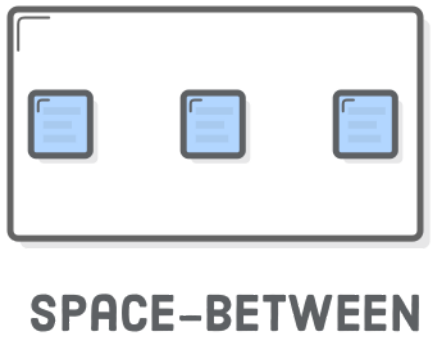
Essayez de changer justify-content par flex-start et flex-end. Cela devrait aligner le menu respectivement sur le côté gauche et à droite de la fenêtre du navigateur. Assurez-vous de changer de nouveau à center. Les deux dernières options ne sont utiles que lorsque vous avez plusieurs éléments flexibles 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:

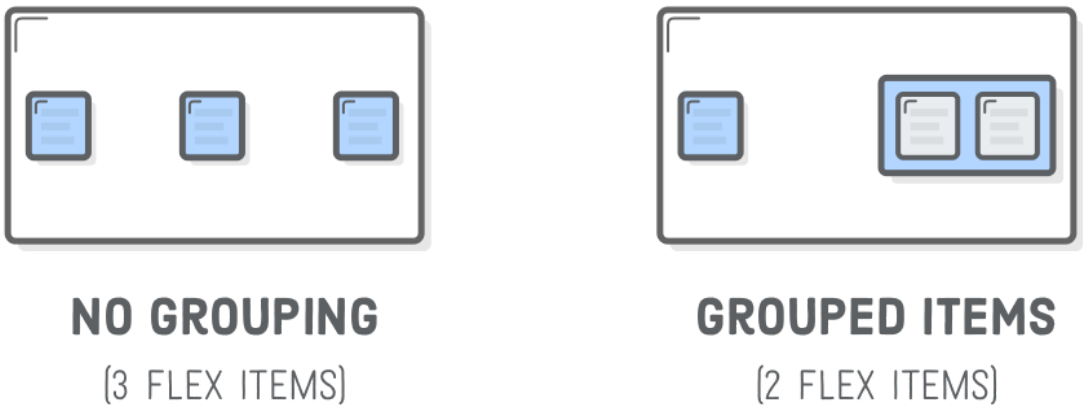
```
.menu {
...
  justify-content: space-between;
}
```

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 enfants). Ils ne se soucient pas de ce qui est à l'intérieur de leurs 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 amènera à un résultat totalement différent dans notre page Web.



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 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>
</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:

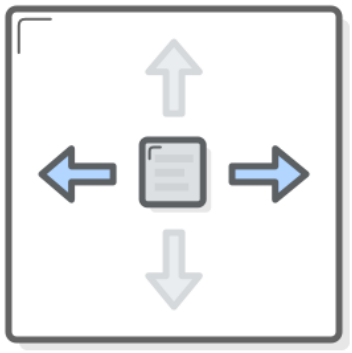
```
.links {
  border: 1px solid #fff;  /* For debugging */
  display: flex;
  justify-content: flex-end;
}

.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).

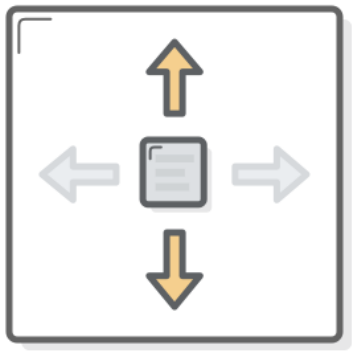
Nous n’aurons plus besoin de ces bordures blanches, de sorte que vous pouvez aller de l'avant et les supprimer si vous le souhaitez.

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 flexbox.html après l'élément .menu-container:

```
<div class='header-container'>
  <div class='header'>
    <div class='subscribe'>Souscrire &#9662;</div>
    <div class='logo'><img src='images/awesome-logo.svg'/></div>
    <div class='social'><img src='images/social-icons.svg'/></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;
}
```

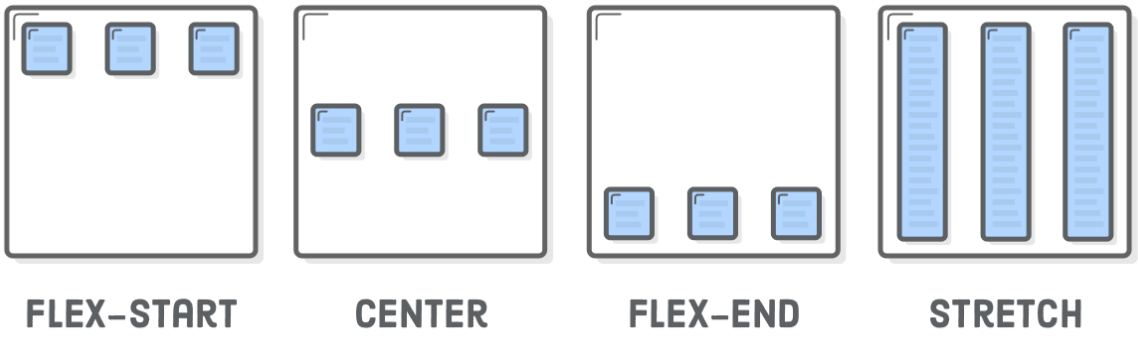
Tout cela devrait être familier. Cependant, le scénario est un peu différent de celui de notre menu. Puisque .header a une hauteur explicite, les éléments peuvent être positionnés verticalement à l'intérieur de celui-ci. La spécification officielle appelle cet alignement "axe transversal" -cross-axis- (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 {
  /* ... */
  align-items: center;  /* Nouveau */
}
```

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

- center
- flex-start (haut)
- flex-end (bas)
- stretch
- baseline



La plupart d'entre elles sont assez simples. L'option stretch est une valeur avec laquelle il faut prendre un peu de temps et jouer avec, car il vous permet de jouer sur l'arrière-plan de chaque élément. Jetons un bref coup d'oeil en ajoutant ce qui suit à styles.css:

```
.header {
  /* ... */
  align-items: stretch;
}

.social, .logo, .subscribe {
  border: 1px solid #5995DA;
}
```

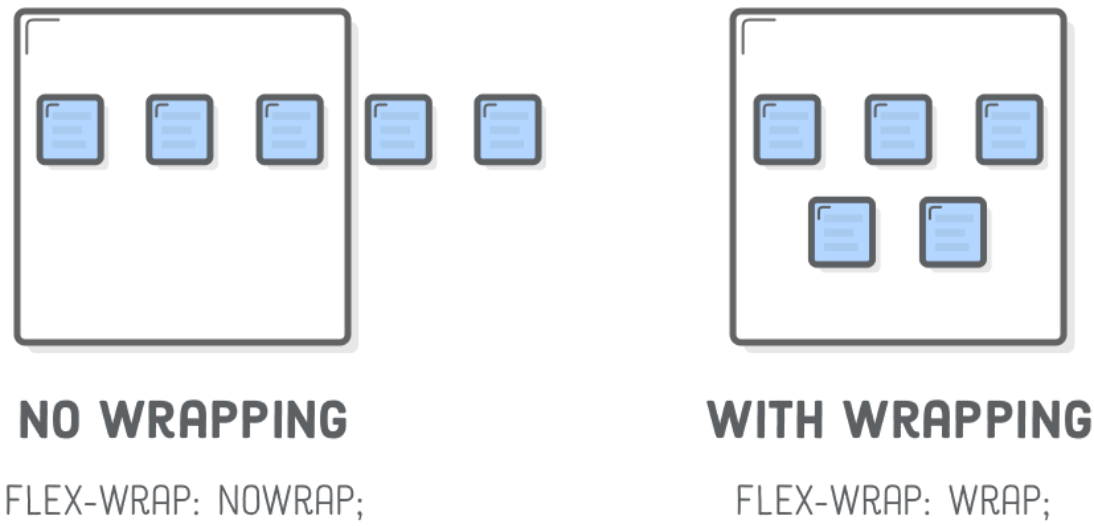
La boîte pour chaque article s'étend sur toute la hauteur du conteneur flexible, quelle que soit la quantité de contenu qu'il contient. Un cas d'utilisation courant pour ce comportement est de créer des colonnes de même hauteur avec une quantité variable de contenu dans chaque, ce qui est quelque chose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



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'>
      <img src='images/one.svg'/>
    </div>
    <div class='photo-grid-item'>
      <img src='images/two.svg'/>
    </div>
    <div class='photo-grid-item'>
      <img src='images/three.svg'/>
    </div>
  </div>
</div>
```

Encore une fois, le CSS correspondant doit être familière:

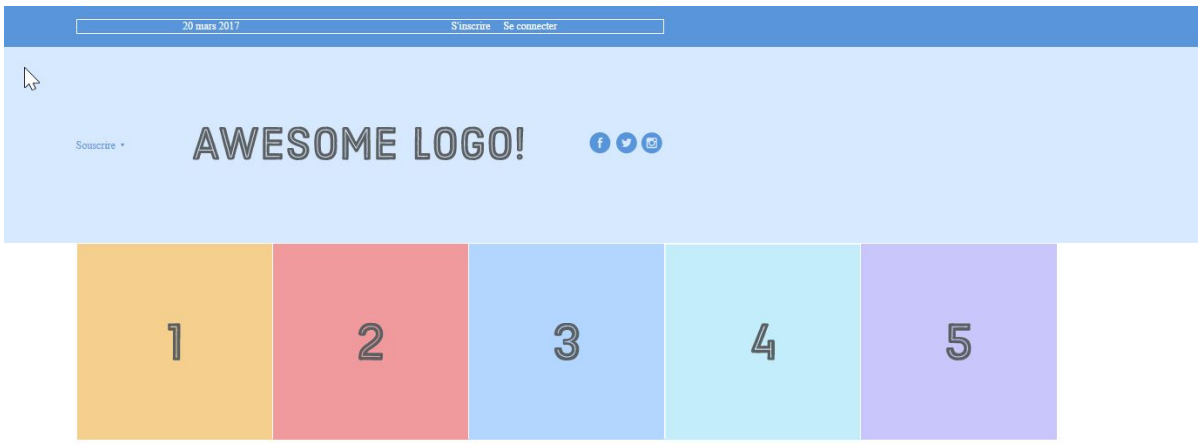
```
.photo-grid-container {
  display: flex;
  justify-content: center;
}

.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'>
  <img src='images/four.svg'/>
</div>
<div class='photo-grid-item last-item'>
  <img src='images/five.svg'/>
</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:

```
.photo-grid {
  /* ... */
  flex-wrap: wrap;
}
```

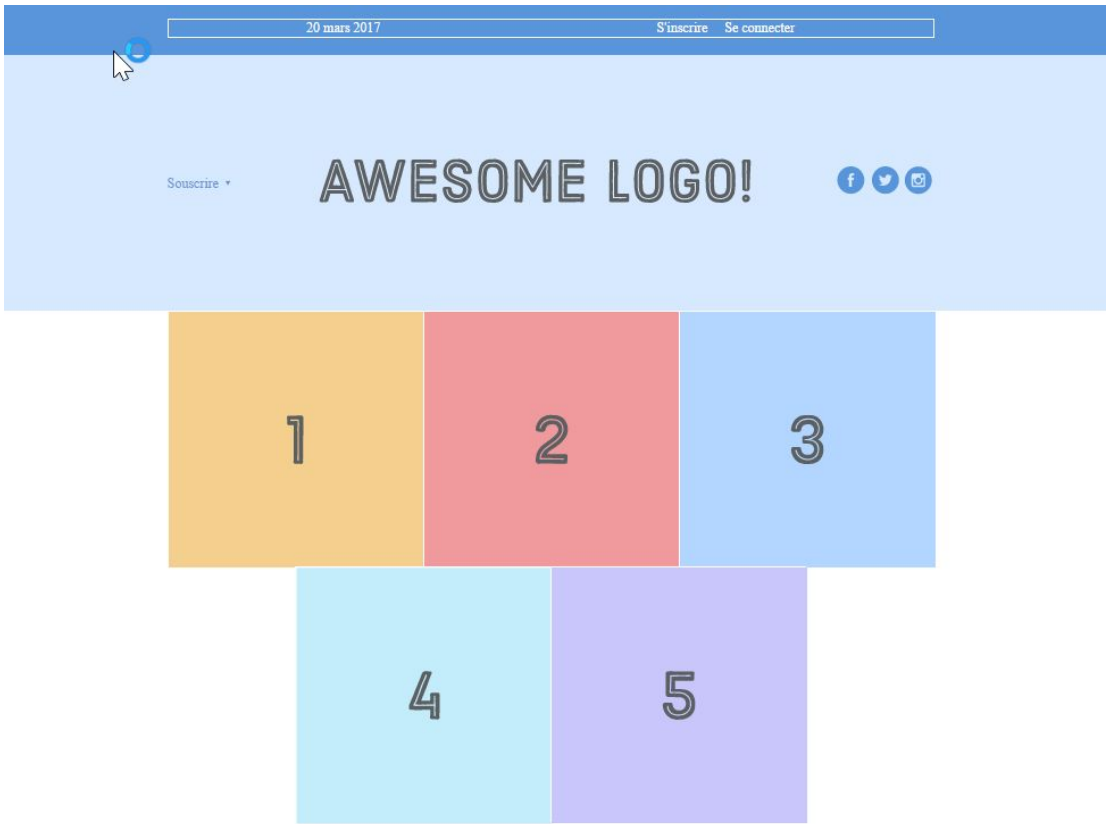
Maintenant, nos éléments flex se comportent un peu comme des boîtes “float”, sauf que FlexBox nous donne plus de contrôle sur la façon dont les éléments "en trops" seront positionnés sur la dernière ligne via la propriété justify-content.

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

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

```
.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 compliqué.



direction et container flex

"Direction" se réfère à 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;

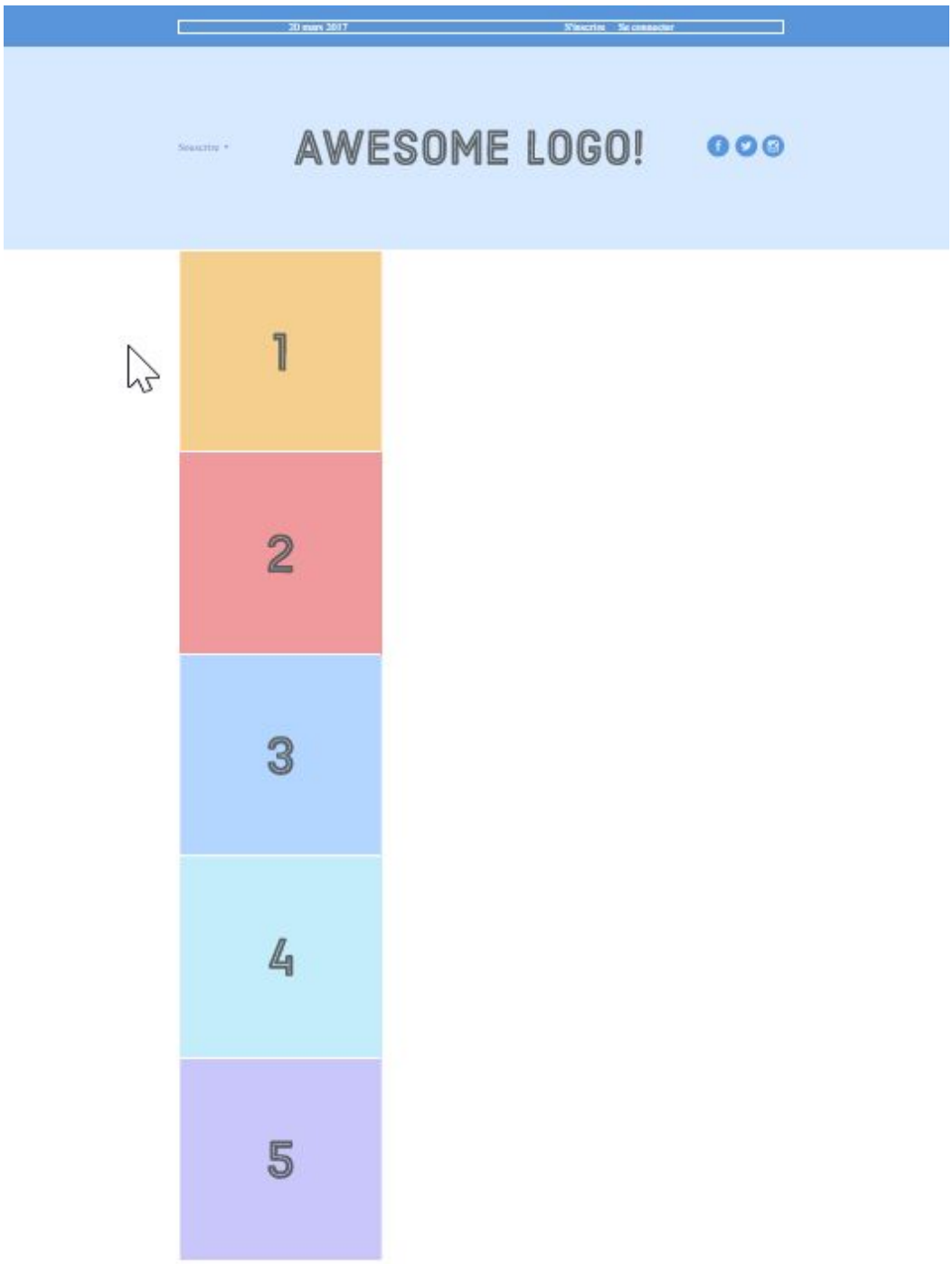
COLUMN

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.

Essayez d'ajouter la déclaration suivante `flex-direction` à notre règle `.photo-grid`:

```
.photo-grid {
  /* ... */
  flex-direction: column;
}
```

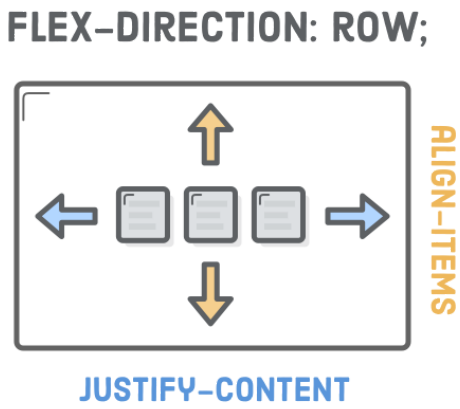


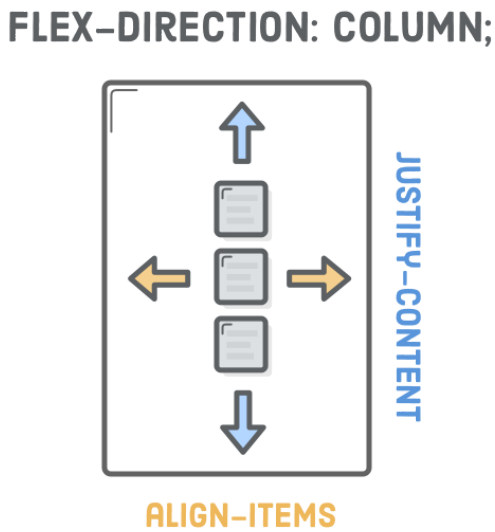
Cela change la direction du récipient de la valeur par défaut la valeur row. Au lieu d'une grille, notre page a maintenant une seule colonne verticale.

Un des éléments clé du “responsive design” est de présenter le même balisage HTML pour les utilisateurs mobiles et 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 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.



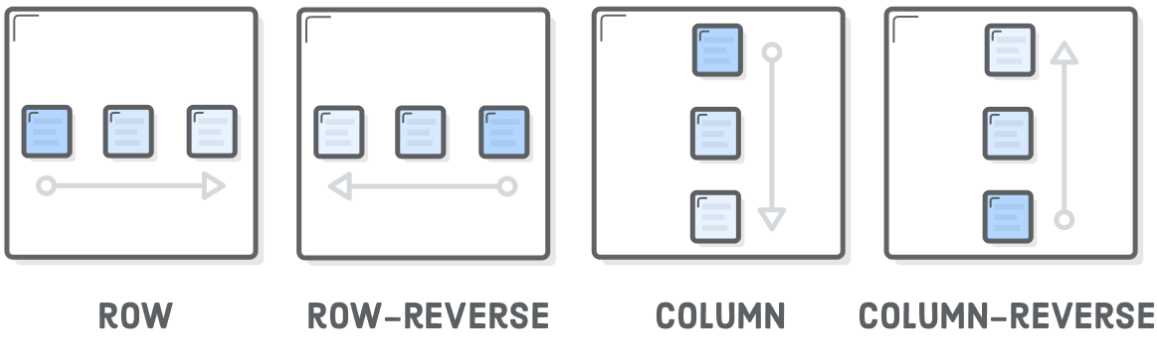


Pour centrer horizontalement notre colonne, nous avons besoin de définir une propriété `align-items` sur notre `.photo-grid`:

```
.photo-grid {
  /* ... */
  align-items: center;      /* Nouveau */
}
```

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 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;
  flex-direction: row-reverse;  /* Nouveau et intéressant ! */
  align-items: center;
}
```

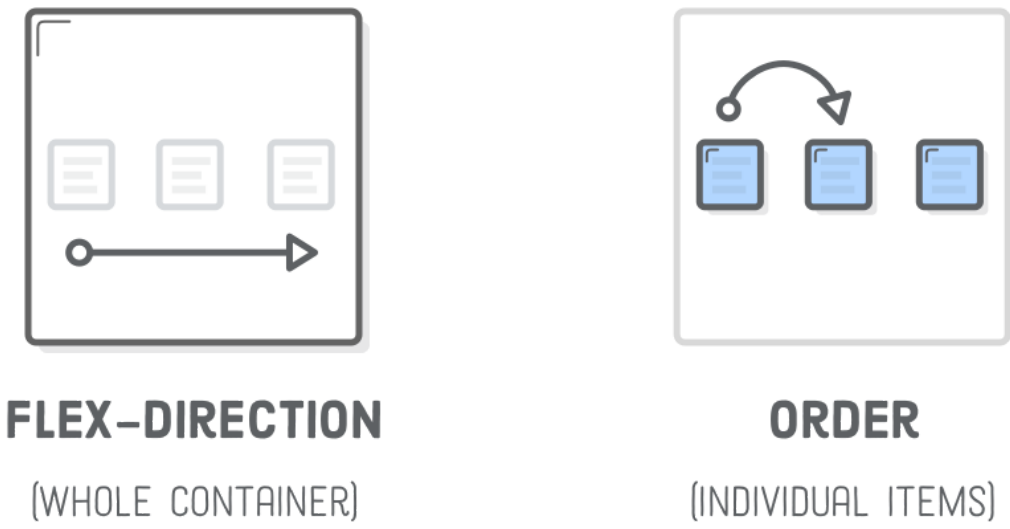
Les deux lignes sont maintenant affichées de droite à gauche au lieu de gauche à droite. Mais, remarquez comment cela ne permute l'ordre sur une

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

Ce chapitre entier a été sur le 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.



L’ajout d'une propriété order à un élément flex définit l'ordre dans le conteneur sans affecter les éléments environnants. 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 {
  /* ... */
  flex-direction: row;  /* Mettre à jour */
  align-items: center;
}

.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 si ils apparaissent sur des lignes différentes.

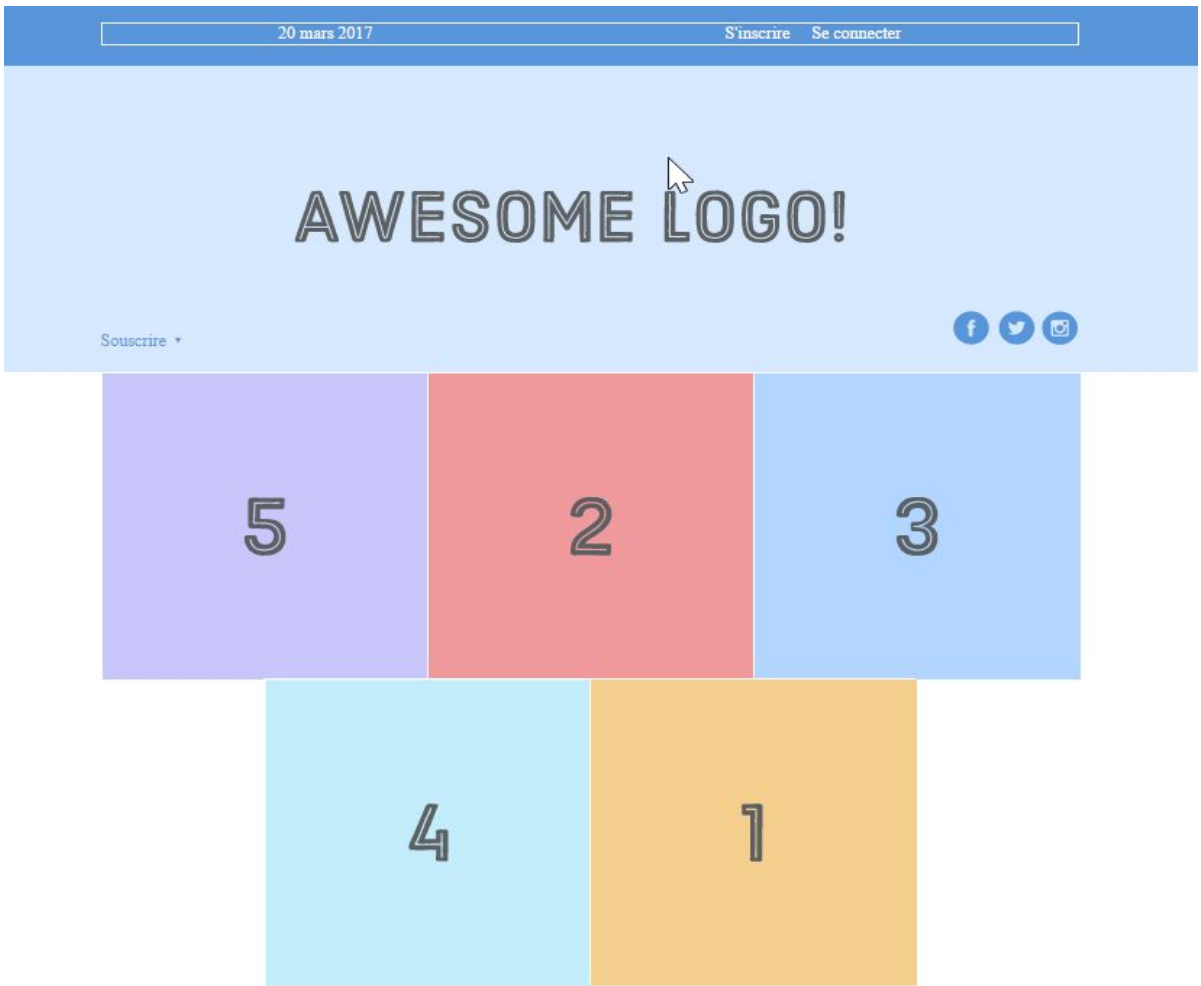
alignement d’un élément flex

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.



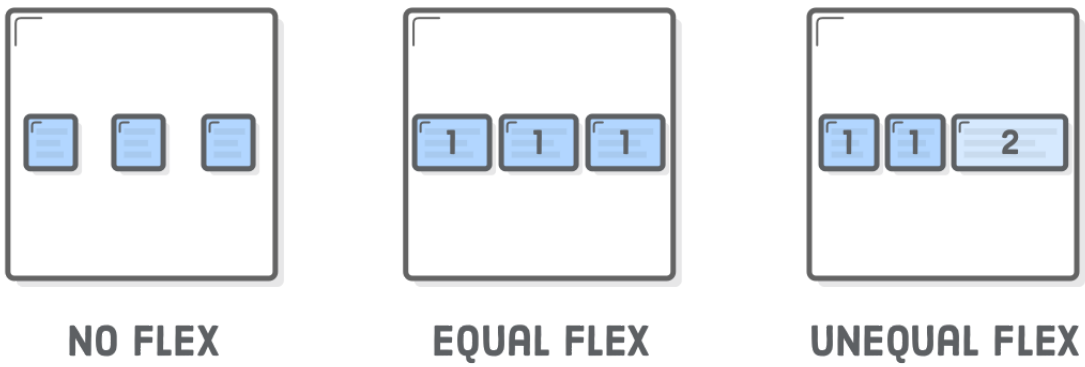
Vous pouvez aligner les éléments de différente façons en utilisant les mêmes valeurs que la propriété `align-items`, énumérés ci-dessous pour plus de commodité.

- `center`
- `flex-start` (haut)
- `flex-end` (bas)
- `stretch`
- `baseline`

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 ignorer 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`.



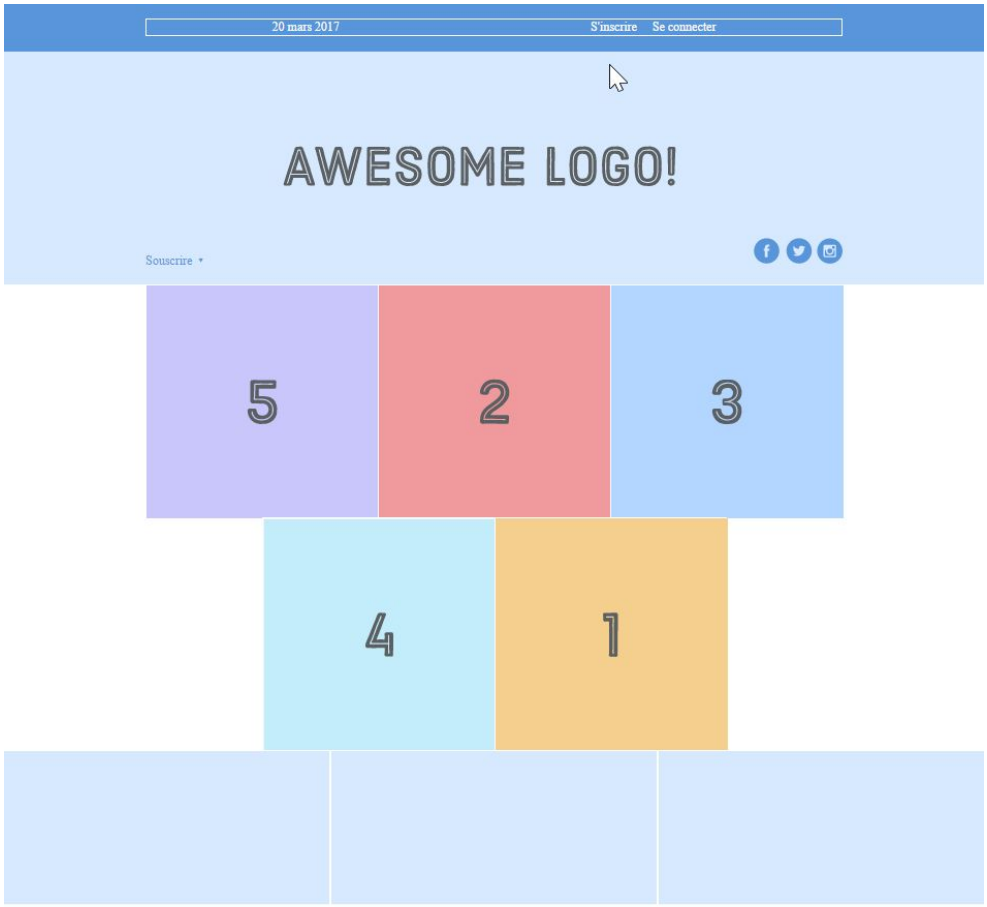
Tout d'abord, nous avons besoin d'un pied de page pour expérimenter. Collez ceci après l'élément .photo-grid-container:

```
<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:

```
.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:



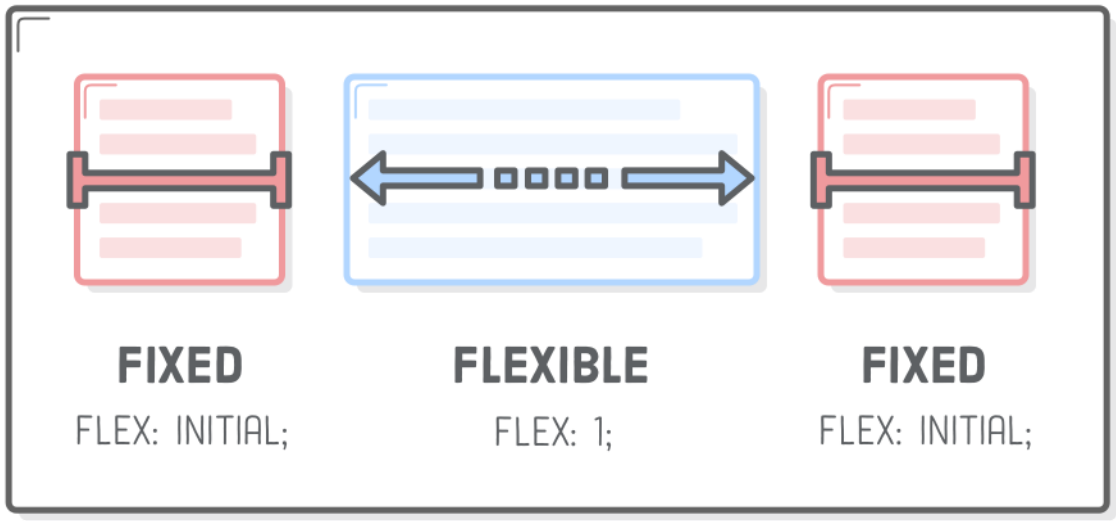
Augmenter le poids de l'un des éléments pour le faire plus large que les autres. Par exemple, nous pouvons faire en sorte que le troisième élément prenne une largeur deux fois plus grande, facteur d'agrandissement de 2, que les deux autres avec la règle suivante:

```
.footer-three {
  flex: 2;
}
```

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.

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.

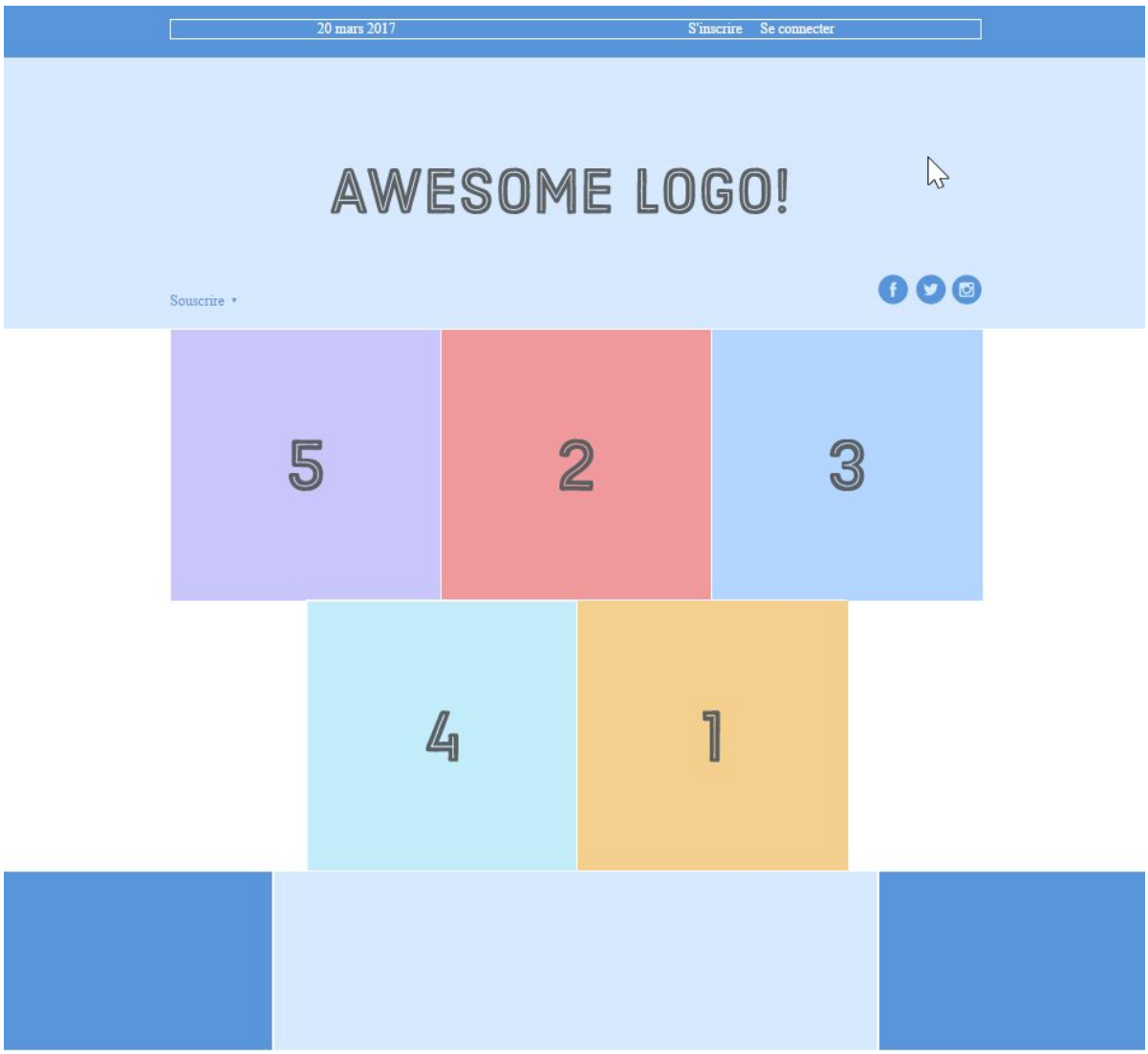


Nous allons faire en sorte que notre pied de page se comporte comme le schéma ci-dessus. L'élément central est flexible, mais celles de chaque côté sont toujours les mêmes dimensions. Tout ce que nous devons faire est d'ajouter la règle suivante à notre feuille de style:

```
.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 les propriétés `width` vont 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.

Flex = flex-grow + flex-shrink + flex-basis

Flex est en fait un raccourci pour 3 propriétés que nous allons détailler maintenant.

La propriété flex-grow indique à flexbox la façon dont il doit augmenter la taille de l'item pour occuper plus d'espace si nécessaire, sa valeur par défaut à 0. La propriété flex-shrink indique à flexbox comment réduire cette taille si nécessaire, sa valeur par défaut à 1. La propriété flex-basis a la responsabilité de déterminer l'espace de départ dont dispose un item avant que l'espace résiduel soit réparti, sa valeur par défaut est à "auto".

Si nous voulons que nos colonnes se comportent de la même façon lorsqu'elles s'agrandissent ou se rapetissent, donnons à nos deux propriétés la valeur 1.

Ajoutons le HTML suivant:

```
<section class="info">
  <div class="column-item column-one"></div>
  <div class="column-item column-two"></div>
  <div class="column-item column-three"></div>
</section>
```

Et le css…

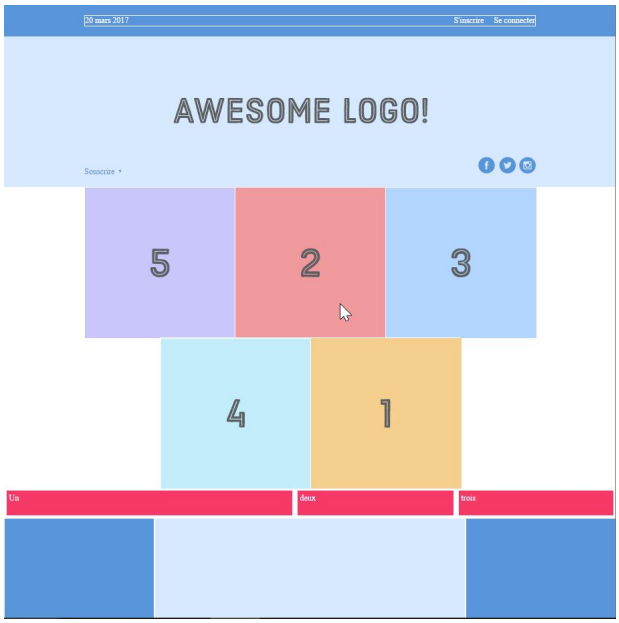
```
.info{
  display: flex;
}

.column-item{
  margin: 10px;
  height: 50px;
  color: #fff;
  flex-grow: 1;
  flex-shrink: 1;
  background-color: #f73967;
}
```

Notre container flexbox occupe toujours trois colonnes. Les valeurs de flex-grow et de flex-shrink sont proportionnelles, ce qui signifie qu'elles

changent en fonction des valeurs des autres éléments. Flexbox fait la somme des valeurs données à chaque élément, puis divise chaque valeur par cette somme. Chaque colonne occupe donc 1 ÷ (1 + 1 + 1) soit 1/3 de l'espace total. Que se passe-t-il si l'une des colonnes a une valeur différente ?

```
.column-one{
/* … */
 flex-grow: 2;
 flex-shrink: 2;
}
```

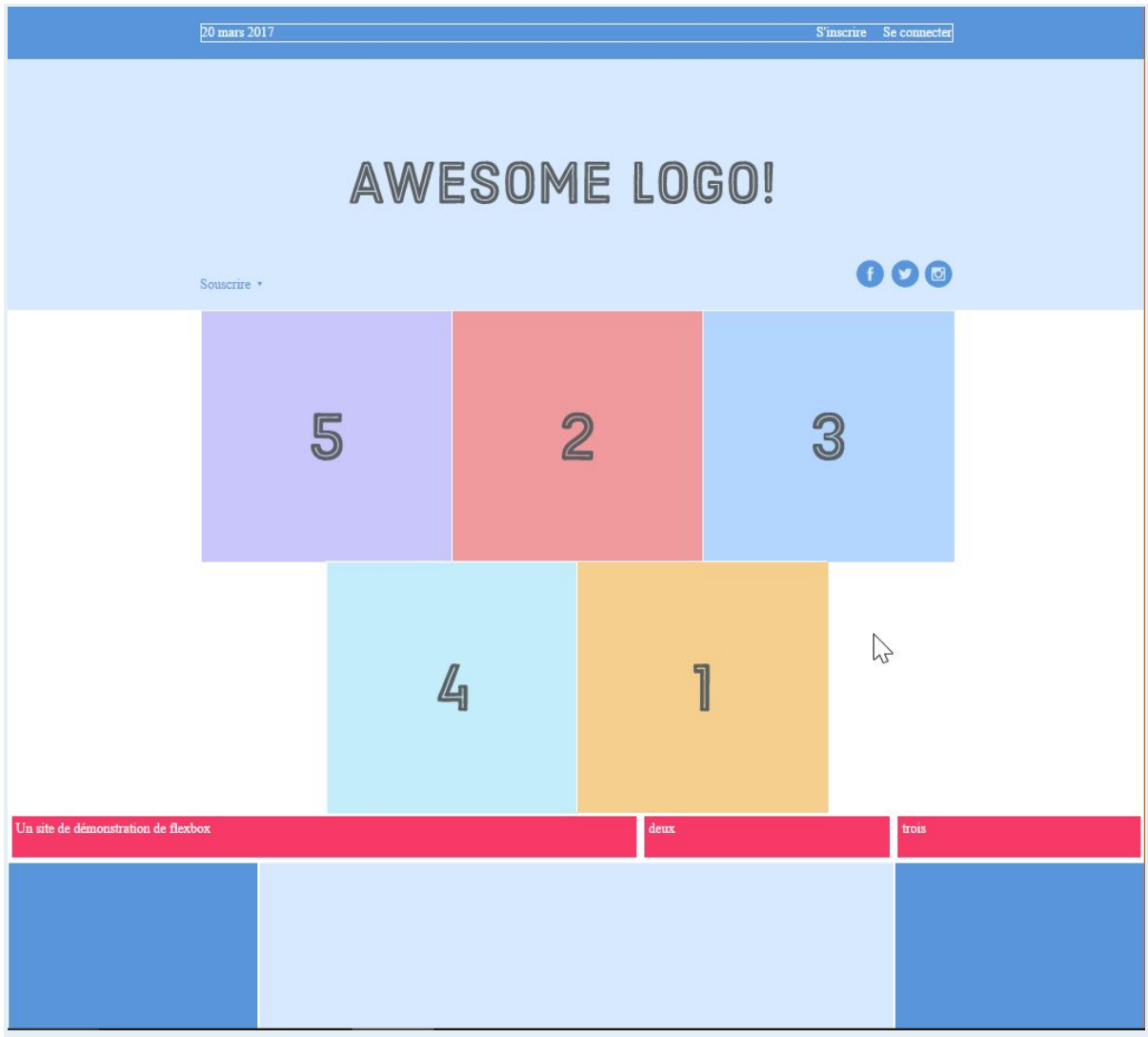
La première colonne prend la même quantité d'espace que les deux autres réunies. La raison en est que la somme des valeurs est égale à 4, donc la première colonne occupe 2 ÷ (2 + 1 + 1), c'est à dire 1/2, et les deux autres occupent 1 ÷ (2 + 1 + 1), c'est à dire 1/4 de l'espace.



C'est une question de basis

Si vous regardez attentivement le dernier exemple, vous verrez que la première colonne n'occupe pas tout à fait la moitié du container. Si nous ajoutons plus de contenu à la première colonne, nous allons voir clairement le problème.

```
<section class="info">
    <div class="column-item column-one">Un site de démonstration de flexbox</div>
    <div class="column-item column-two">deux</div>
    <div class="column-item column-three">trois</div>
</section>
```



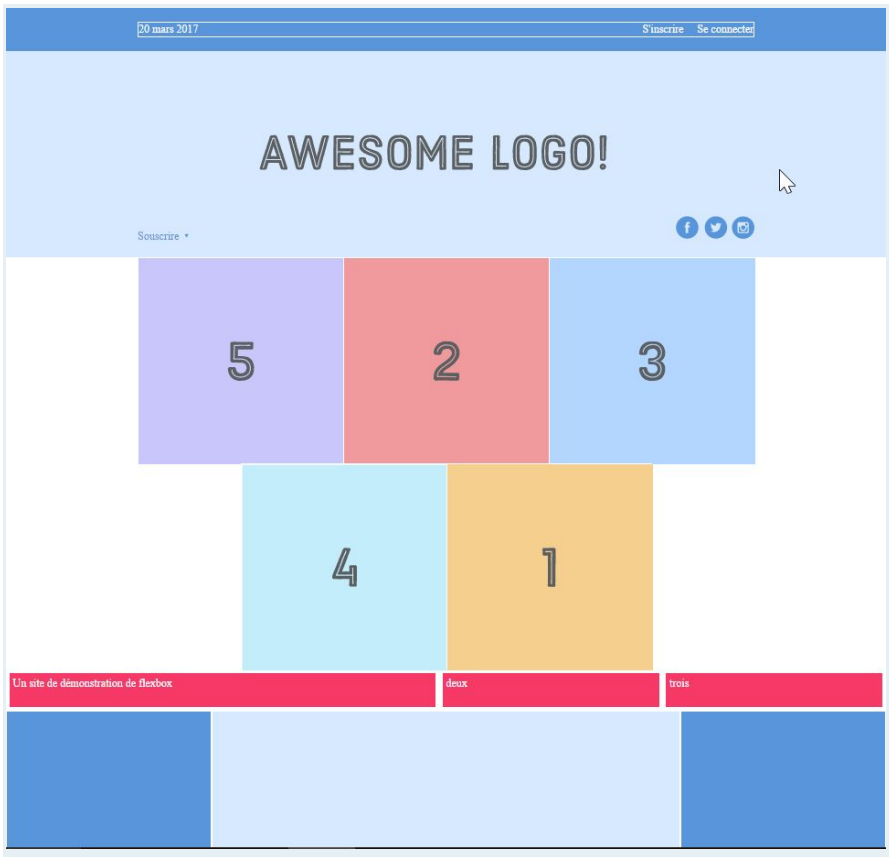
Pourquoi flexbox n'est-il pas flexible ?

Eh bien en fait, il se trouve que flexbox ne répartit pas l'espace de manière égale entre les colonnes. Il cherche d'abord à savoir de combien d'espace chaque colonne a besoin, puis répartit l'espace qui reste via les propriétés flex-grow et flex-shrink.

Cela semble peut-être un peu confus, et pour tout dire ça l'est. La façon dont ce truc est construit est sacrément compliquée, mais ne vous inquiétez pas, vous n'avez pas besoin de comprendre toutes les nuances pour utiliser flexbox.

Dans la mesure où nous ne nous soucions pas de l'espace d'origine, tout ce que nous avons à faire est de régler flex-basis sur la valeur 0 pour indiquer à flexbox qu'il doit ignorer la taille par défaut du container.

```
.column-item{
 margin: 10px;
 flex-grow: 1;
 flex-shrink: 1;
 flex-basis: 0;
}

.column-one{
 flex-grow: 2;
 flex-shrink: 2;
 flex-basis: 0;
}
```



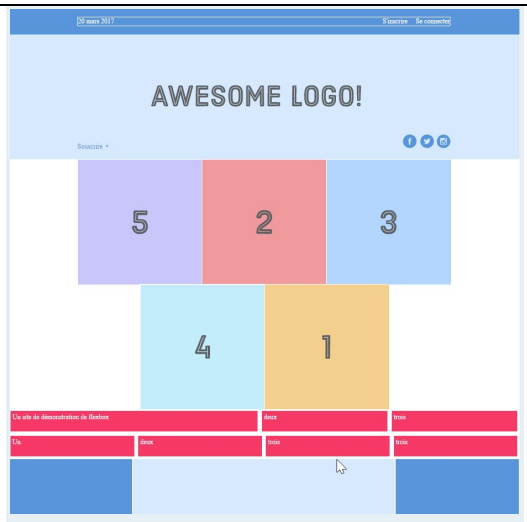
Ça marche ! Enfin… à peu près, il nous reste une dernière petite chose à régler.

Encore plus de flex-basis

Si nous ajoutons une autre section sous la première, nous voyons tout de suite le problème :

```
<section class="info2">
    <div class="column-item column-one">Un </div>
    <div class="column-item column-two">deux</div>
    <div class="column-item column-three">trois</div>
<div class="column-item column-four">quatre</div>
</section>
```

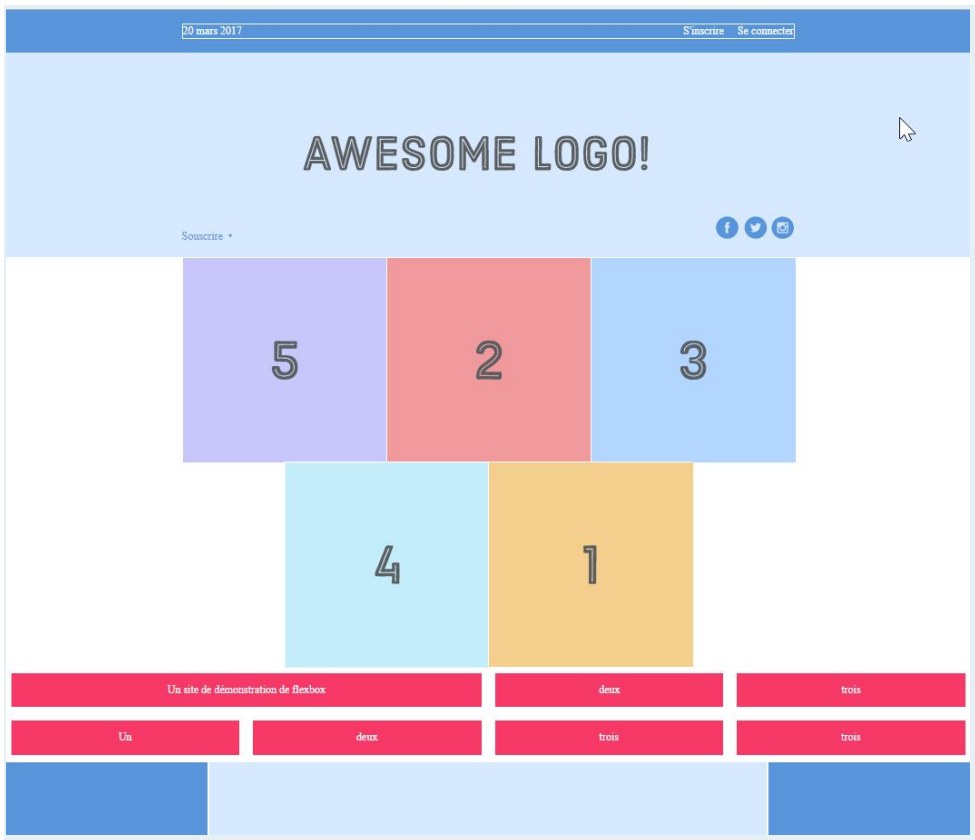
```
.info, .info-row-2{
  display: flex;
}


.info .column-one{
 flex-grow: 2;
 flex-shrink: 2;
 flex-basis: 0;
}
```



Vous avez remarqué que les colonnes ne s'alignent pas ? C'est parce que flexbox inclut les bordures et les marges dans le basis lorsqu'il calcule la taille que devrait faire l'item.

La première et la deuxième colonne de la deuxième rangée sont séparées par 22 pixels (20px pour la marge et 2px pour les bordures). Nous pouvons ajouter cet espace manquant dans la première colonne de la première rangée en fixant flex-basis à 22px.

```
.info .column-one{
  flex-grow: 2;
  flex-shrink: 2;
  flex-basis: 22px;
}
```



Ensemble, flex-grow, flex-shrink et flex-basis forment la pierre angulaire de la flexibilité de flexbox (ça fait beaucoup de flex). Comme ces propriétés sont très liées, flex propose une propriété raccourcie qui vous permet de régler les trois.

Vous pouvez l’utiliser ainsi :

```
flex: <flex-grow> <flex-shrink> <flex-basis>;
```

Nous pouvons réécrire notre CSS :

```
.column-item{
  flex: 1 1 0;
  margin: 0 auto;
}

.info .column-one{
  flex: 2 2 22px;
}
```

Aaah, c’est mieux, non ?

Elément flex et marges “auto”

Les marges “auto” dans FlexBox sont spéciales. Elles peuvent être utilisées comme une alternative à un 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">
  <div class="menu">
    <div class="date">20 mars 2017</div>
    <div class="signup">S'inscrire</div>
    <div class="login">Se connecter</div>
  </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:

```
.signup {
  margin-left: auto;
}
```

Les marges auto “mange” tout l'espace supplémentaire dans un conteneur flexible, donc au lieu de distribuer de façon égale tous les éléments, cela déplace les .signup et tous les éléments suivants (.login) sur le côté droit du conteneur. Cela vous donnera exactement la même disposition que nous avions avant, mais sans ce 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