Les Flex box

Le module **CSS Flexible Box Layout Module Level 1** est en **Candidate Recommendation** au 26 mai 2016 : http://www.w3.org/TR/css-flexbox-1/.

C'est un module qui va vous permettre de "révolutionner" vos mises en page simples et complexes, qui va vous permettre de ne plus utiliser toute une série de "bidouilles" pour concevoir des mises en page web.

Sachez dès maintenant que la mise en page Flexbox ne travaille plus du tout avec les notions de table, de float, de clear ou de position. La mise en page Flexbox travaille dans le "contexte Flexbox".

Le **Flexible Box Layout Module**, ou Flexbox en plus court, ou modèle de boîte flexible en français, va vous permettre, entres autres :

- *de gérer des alignements des boîtes,*
- *de centrer horizontalement et verticalement les boîtes,*
- *d'avoir des mises en page fluides,*
- *des hauteurs de boîte similaires,*
- *des fonds colorés uniformes,*
- *de concevoir des mises en page responsive.*

En ce qui concerne l'implémentation de Flexbox, elle est tout simplement excellente. Voici le tableau de compatibilité de **Can I use** :

Les propriétés pour le conteneur Flexbox

1. Le conteneur

La mise en page Flexbox est contenue dans un conteneur qui possède l'affichage Flexbox. C'est lui qui va donner le "contexte Flexbox". Pour appliquer ce contexte Flexbox, nous allons utiliser la "très classique" propriété display sur n'importe quel élément HTML :

- *display: flex pour avoir un affichage en block, ou*
- *display: inline-flex pour avoir un affichage inline-block.*

Tous les éléments enfants du conteneur Flexbox, les items Flexbox, seront dans le contexte Flexbox, "positionnés" comme nous le voulons.

Voici un premier exemple simple, avec un conteneur en affichage display: flex contenant quatre enfants positionnés, par défaut, horizontalement.

Le code HTML/CSS :

```
<!DOCTYPE html>
<html lang="fr">
<head>
   <title>Ma mise en page</title>
   <meta charset="UTF-8" />
   <style>
      #conteneur {
         display: flex;
      }
      p.bloc {
         background-color: #eee;
         border: solid 1px #000;
         margin-right: 10px;
      }
   </style>
</head>
<body>
<div id="conteneur">
   <p class="bloc">Truc</p>
```

```
    <p class="bloc">Machin</p>
    <p class="bloc">Bidule</p>
    <p class="bloc">Chose</p>
</div>
</body>
</html>
```

Voici l'affichage obtenu :

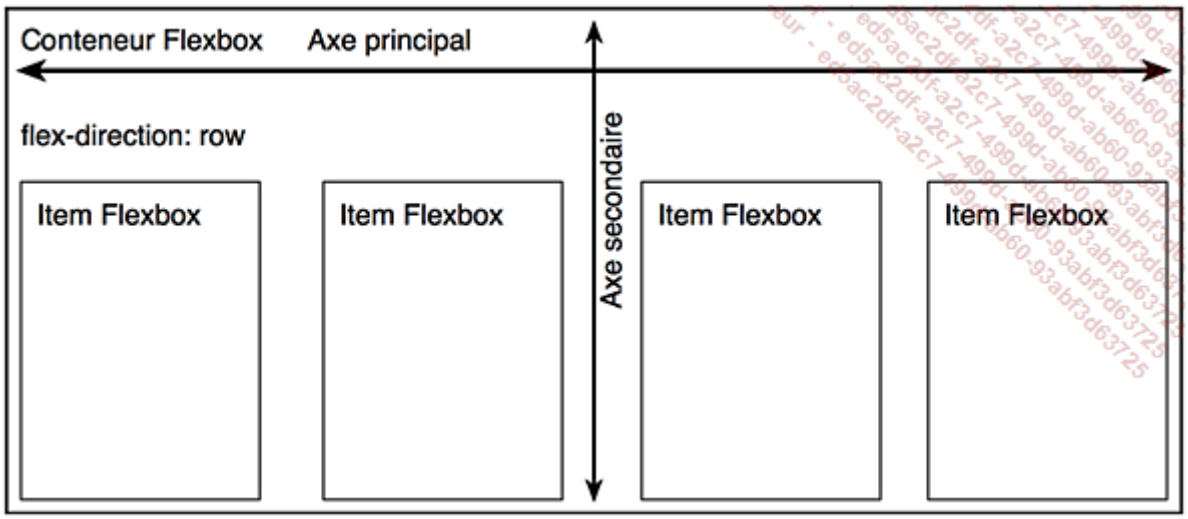
Truc Machin Bidule Chose

Vous voyez que les blocs sont les uns à côté des autres, nativement avec le contexte Flexbox.

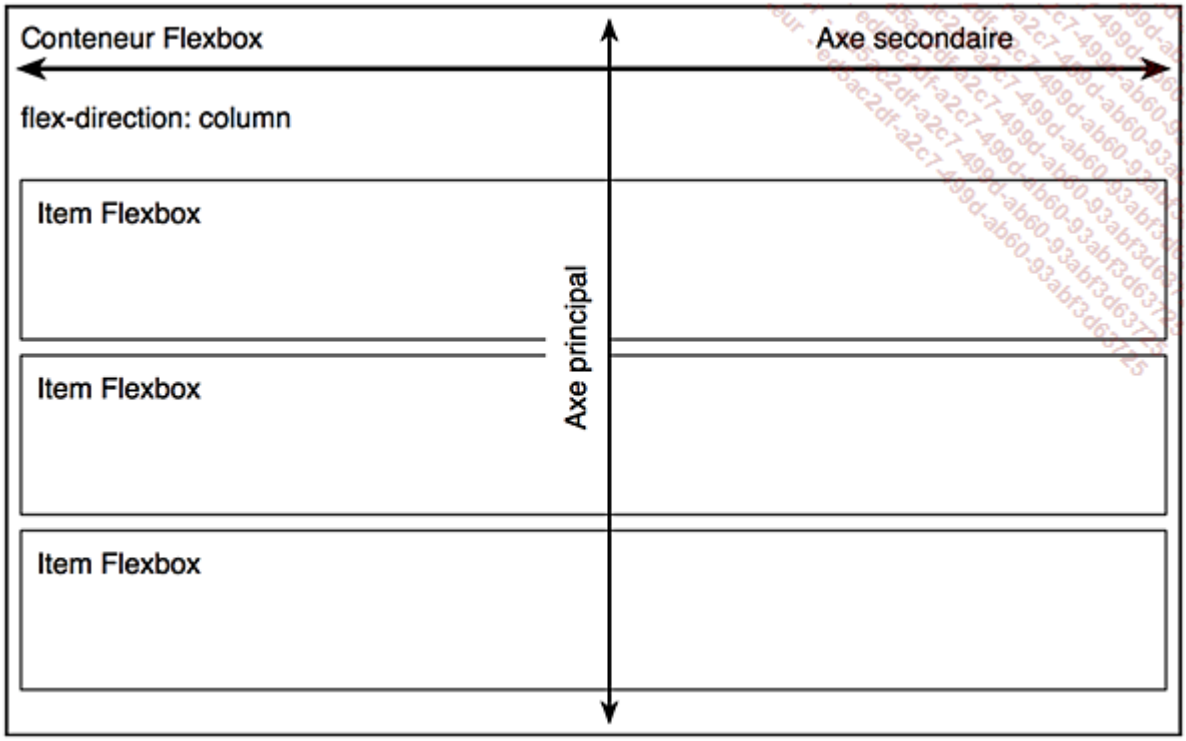
2. La direction de l'affichage

La propriété flex-direction permet de définir la "direction" de l'affichage des éléments enfants du conteneur Flexbox : horizontale ou verticale. Cela définit l'axe principal du conteneur Flexbox. La direction de l'axe secondaire sera l'inverse de l'axe principal.

- *flex-direction: row : les éléments enfants, les items Flexbox, sont affichés les uns à côté des autres horizontalement. C'est la valeur par défaut.*



- *flex-direction: row-reverse : les éléments enfants, les items Flexbox, sont affichés les uns à côté des autres horizontalement, mais dans l'ordre inverse de leur déclaration dans le code HTML.*
- *flex-direction: column : les éléments enfants, les items Flexbox, sont affichés les uns sous les autres verticalement.*



- *flex-direction: column-reverse : les éléments enfants , les items Flexbox, sont affichés les uns sous les autres verticalement., mais dans l'ordre inverse de leur déclaration dans le code HTML.*

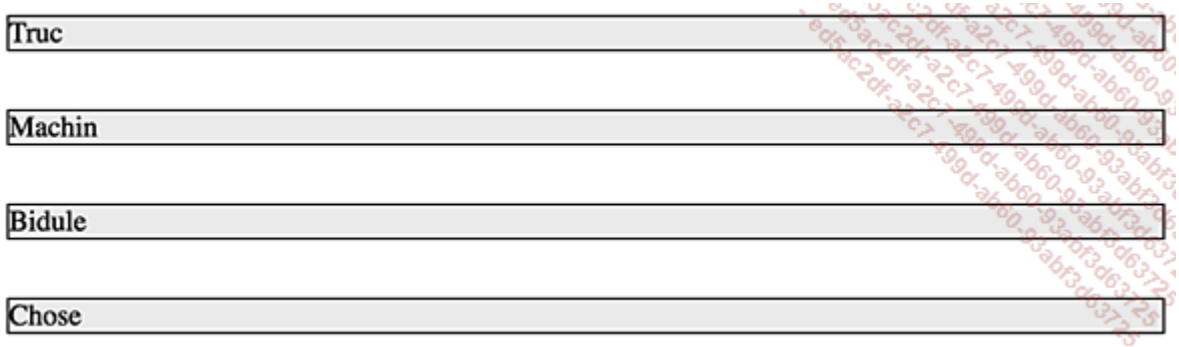
Voici un exemple d'un affichage en colonne, avec la propriété flex-direction: column, avec un axe principal vertical.

Le code HTML/CSS :

```
<!DOCTYPE html>
<html lang="fr">
<head>
   <title>Ma mise en page</title>
   <meta charset="UTF-8" />
   <style>
      #conteneur {
         display: flex;
         flex-direction: column;
      }
      p.bloc {
         background-color: #eee;
```

```
        border: solid 1px #000;
      }
    </style>
</head>
<body>
<div id="conteneur">
    <p class="bloc">Truc</p>
    <p class="bloc">Machin</p>
    <p class="bloc">Bidule</p>
    <p class="bloc">Chose</p>
</div>
</body>
</html>
```

Voici l'affichage obtenu :



3. Le passage à la ligne ou à la colonne

La propriété flex-wrap permet de définir si le passage à la ligne ou à la colonne est autorisé pour les éléments enfants, les items Flexbox. La direction du passage à ligne ou à la colonne dépend bien sûr de l'axe principal du conteneur.

- *flex-wrap: nowrap n'autorise pas le passage à la ligne ou à la colonne. C'est la valeur par défaut.*
- *flex-wrap: wrap autorise le passage à la ligne ou à la colonne.*
- *flex-wrap: wrap-reverse autorise le passage à la ligne ou à la colonne, dans l'ordre inverse de la déclaration des éléments enfants.*

Voici un exemple de deux conteneurs Flexbox, avec un axe principal vertical, en colonne. Le premier conteneur n'autorise pas le passage à la colonne, le deuxième l'autorise.

Le code HTML/CSS :

```
<!DOCTYPE html>
<html lang="fr">
<head>
   <title>Ma mise en page</title>
   <meta charset="UTF-8" />
   <style>
      .conteneur1 {
         display: flex;
         flex-direction: column;
         flex-wrap: nowrap;
         height: 50x;
      }
      .conteneur2 {
         display: flex;
         flex-direction: column;
         flex-wrap: wrap;
         height: 50px;
      }
      p.bloc {
         background-color: #eee;
         border: solid 1px #000;
         margin: 0;
      }
   </style>
</head>
<body>
<div class="conteneur1">
   <p class="bloc">Elit Tortor…</p>
   <p class="bloc">Nullam Ultricies…</p>
```

```
    <p class="bloc">Lorem Magna Nibh...</p>
    <p class="bloc">Tellus Malesuada...</p>
</div>
<p> </p>
<div class="conteneur2">
    <p class="bloc">Elit Tortor...</p>
    <p class="bloc">Nullam Ultricies...</p>
    <p class="bloc">Lorem Magna Nibh...</p>
    <p class="bloc">Tellus Malesuada...</p>
</div>
</body>
</html>
```

L'affichage obtenu :

Elit Tortor Porta Fermentum Ornare
Nullam Ultricies Pharetra Dolor Adipiscing
Lorem Magna Nibh Quam Sollicitudin
Tellus Malesuada Porta Mattis Adipiscing

Elit Tortor Porta Fermentum Ornare	Lorem Magna Nibh Quam Sollicitudin
Nullam Ultricies Pharetra Dolor Adipiscing	Tellus Malesuada Porta Mattis Adipiscing

4. La propriété en syntaxe courte

La propriété flex-flow est la syntaxe courte pour les propriétés flex-direction et flex-wrap.

Déclaration des deux propriétés :

flex-direction: column;

flex-wrap: wrap;

Déclaration en syntaxe courte :

flex-flow: column wrap;

5. Les alignements et les centrages dans l'axe principal

La propriété justify-content permet de définir l'alignement et le centrage des items Flexbox dans l'axe principal du conteneur Flexbox.

Dans les définitions suivantes, l'axe principal est horizontal (c'est la valeur par défaut flex-direction: row).

- *justify-content: flex-start indique que les items Flexbox sont alignés sur la gauche du conteneur. Ils sont "calés" au début du conteneur. C'est la valeur par défaut.*
- *justify-content: flex-end indique que les items Flexbox sont alignés sur la droite du conteneur. Ils sont "calés" à la fin du conteneur.*
- *justify-content: center indique que les items Flexbox sont centrés dans le conteneur.*
- *justify-content: space-between indique que les items Flexbox sont justifiés dans la largeur du conteneur. Ils sont uniformément répartis dans la largeur, avec le premier tout à gauche et le dernier tout à droite.*
- *justify-content: space-around indique que les items Flexbox sont justifiés dans la largeur du conteneur. Ils sont uniformément répartis dans la largeur, avec un espace avant le premier item à gauche et avec un espace après le dernier item à droite.*

Voici trois exemples.

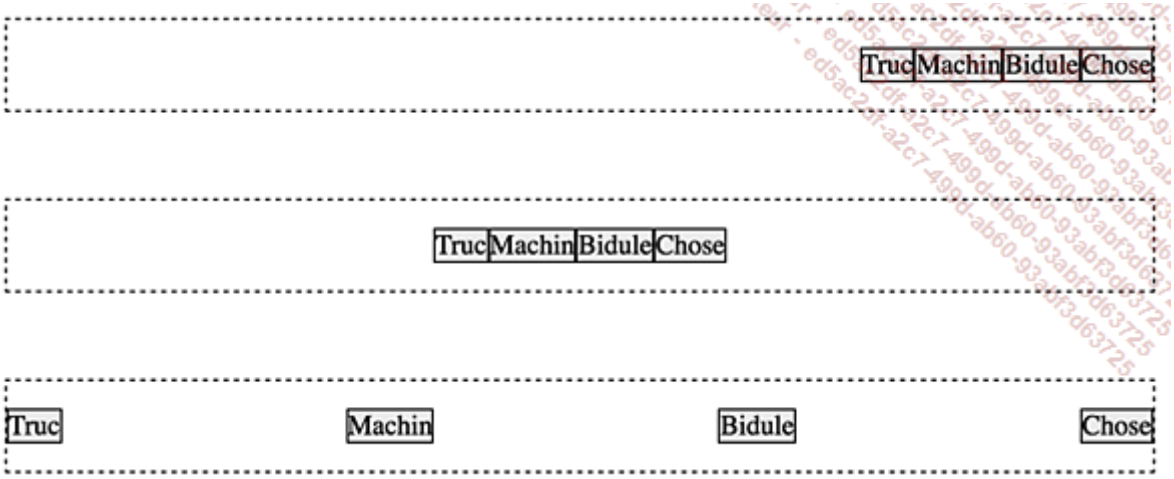
Le code HTML/CSS :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <title>Ma mise en page</title>
  <meta charset="UTF-8" />
  <style>
    #conteneur1 {
      display: flex;
      flex-direction: row;
      justify-content: flex-end;
      border: dashed 1px #000;
    }
    #conteneur2 {
      display: flex;
      flex-direction: row;
      justify-content: center;
      border: dashed 1px #000;
```

```
      }
      #conteneur3 {
         display: flex;
         flex-direction: row;
         justify-content: space-between;
         border: dashed 1px #000;
      }
      p.bloc {
         background-color: #eee;
         border: solid 1px #000;
      }
   </style>
</head>
<body>
<div id="conteneur1">
   <p class="bloc">Truc</p>
   <p class="bloc">Machin</p>
   <p class="bloc">Bidule</p>
   <p class="bloc">Chose</p>
</div>
<p> </p>
<div id="conteneur2">
   <p class="bloc">Truc</p>
   <p class="bloc">Machin</p>
   <p class="bloc">Bidule</p>
   <p class="bloc">Chose</p>
</div>
<p> </p>
<div id="conteneur3">
   <p class="bloc">Truc</p>
   <p class="bloc">Machin</p>
   <p class="bloc">Bidule</p>
   <p class="bloc">Chose</p>
```

```
</div>
</body>
</html>
```

L'affichage obtenu :



6. Les alignements et les centrages dans l'axe secondaire

La propriété align-items permet de définir l'alignement et le centrage des items Flexbox dans l'axe secondaire du conteneur Flexbox.

Rappelons que si l'axe principal est horizontal, avec la propriété par défaut flex-direction: row, l'axe secondaire est vertical. Si l'axe principal est vertical, avec la propriété flex-direction: column, l'axe secondaire est horizontal.

Dans les définitions suivantes, l'axe principal est horizontal (flex-direction: row), donc l'axe secondaire est vertical. C'est donc dans l'axe vertical que vont se définir les alignements et les centrages.

- *align-items: stretch spécifie que les items Flexbox sont étirés verticalement dans le conteneur, pour occuper toute la place disponible. C'est la valeur par défaut.*
- *align-items: flex-start spécifie que les items Flexbox sont alignés sur le haut du conteneur. Ils sont "calés" au début de l'axe secondaire, soit en haut de celui-ci.*
- *align-items: flex-end spécifie que les items Flexbox sont alignés en bas du conteneur. Ils sont "calés" à la fin de l'axe secondaire, soit en bas de celui-ci.*
- *align-items: center spécifie que les items Flexbox sont centrés verticalement dans le conteneur.*

- *align-items: baseline spécifie que les items Flexbox sont alignés sur la ligne de base du texte.*

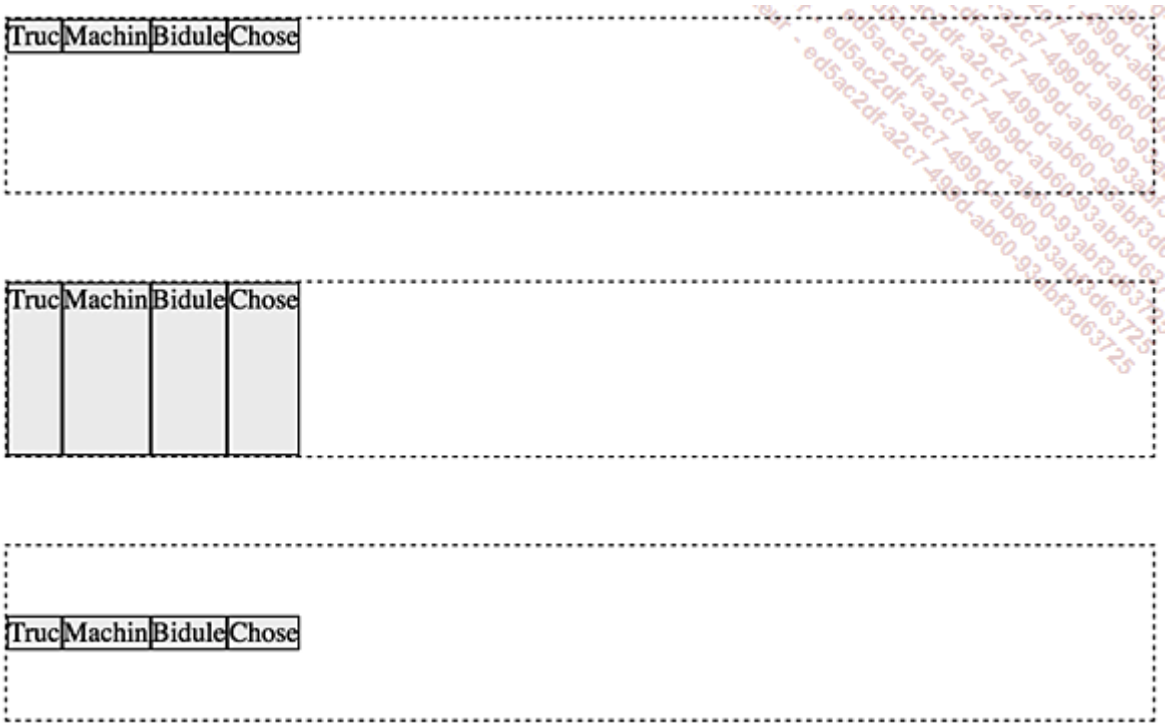
Voici trois exemples d'alignement et de centrage.

Le code HTML/CSS :

```
<!DOCTYPE html>
<html lang="fr">
<head>
   <title>Ma mise en page</title>
   <meta charset="UTF-8" />
   <style>
      #conteneur1 {
         display: flex;
         flex-direction: row;
         align-items: flex-start;
         border: dashed 1px #000;
         height: 100px;
      }
      #conteneur2 {
         display: flex;
         flex-direction: row;
         align-items: stretch;
         border: dashed 1px #000;
         height: 100px;
      }
      #conteneur3 {
         display: flex;
         flex-direction: row;
         align-items: center;
         border: dashed 1px #000;
         height: 100px;
      }
      p.bloc {
```

```
        background-color: #eee;
        border: solid 1px #000;
        margin: 0;
      }
  </style>
</head>
<body>
<div id="conteneur1">
  <p class="bloc">Truc</p>
  <p class="bloc">Machin</p>
  <p class="bloc">Bidule</p>
  <p class="bloc">Chose</p>
</div>
<p> </p>
<div id="conteneur2">
  <p class="bloc">Truc</p>
  <p class="bloc">Machin</p>
  <p class="bloc">Bidule</p>
  <p class="bloc">Chose</p>
</div>
<p> </p>
<div id="conteneur3">
  <p class="bloc">Truc</p>
  <p class="bloc">Machin</p>
  <p class="bloc">Bidule</p>
  <p class="bloc">Chose</p>
</div>
</body>
</html>
```

Voici l'affichage obtenu :



7. L'alignement et passage à la ligne

La propriété align-content permet de gérer l'alignement des items Flexbox dans les rangées des axes secondaires. Et ceci, uniquement dans le contexte d'un conteneur qui autorise le passage à la ligne avec flex-wrap: wrap.

- *align-content: stretch : permet aux items Flexbox d'occuper toute la place disponible. C'est la valeur par défaut.*
- *align-content: flex-start : permet aux items Flexbox de s'aligner au début du conteneur.*
- *align-content: flex-end : permet aux items Flexbox de s'aligner à la fin du conteneur.*
- *align-content: space-between : indique que les items Flexbox sont justifiés dans l'axe secondaire, pour occuper toute la place disponible.*
- *align-content: space-around : indique que les items Flexbox sont justifiés dans l'axe secondaire, pour occuper toute la place disponible, mais avec un espace avant le premier item et un espace après le dernier item.*

Voici quatre exemples d'alignement avec un conteneur Flexbox avec un axe principal horizontal (flex-direction: row).

Le code HTML/CSS :

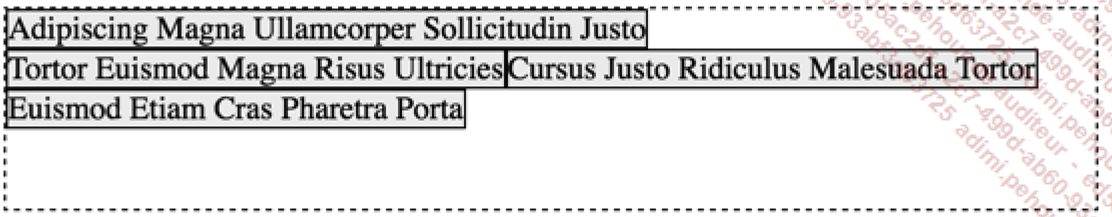
```
<!DOCTYPE html>
<html lang="fr">
```

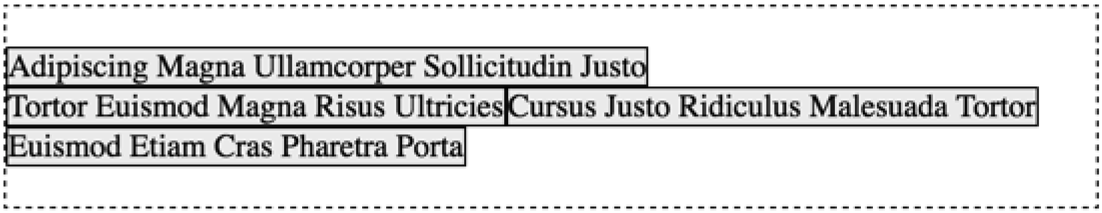
```
<head>
   <title>Ma mise en page</title>
   <meta charset="UTF-8" />
   <style>
      #conteneur1 {
         display: flex;
         flex-direction: row;
         flex-wrap: wrap;
         align-content: flex-start;
         border: dashed 1px #000;
         height: 100px;
      }
      #conteneur2 {
         display: flex;
         flex-direction: row;
         flex-wrap: wrap;
         align-content: center;
         border: dashed 1px #000;
         height: 100px;
      }
      #conteneur3 {
         display: flex;
         flex-direction: row;
         flex-wrap: wrap;
         align-content: stretch;
         border: dashed 1px #000;
         height: 100px;
      }
      #conteneur4 {
         display: flex;
         flex-direction: row;
         flex-wrap: wrap;
         align-content: space-between;
```

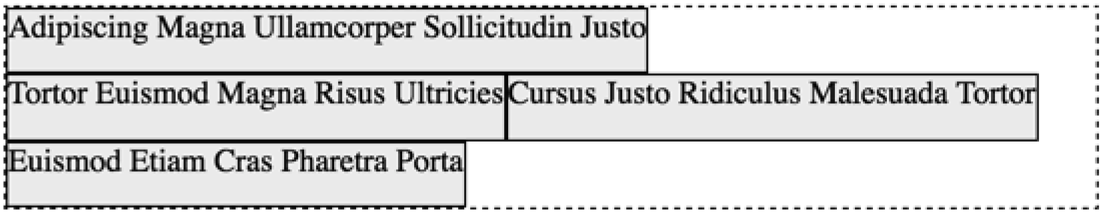
```
        border: dashed 1px #000;
        height: 100px;
      }
      p.bloc {
        background-color: #eee;
        border: solid 1px #000;
        margin: 0;
      }
    </style>
</head>
<body>
<div id="conteneur1">
  <p class="bloc">Adipiscing Magna Ullamcorper Sollicitudin
Justo</p>
  <p class="bloc">Tortor Euismod Magna Risus Ultricies</p>
  <p class="bloc">Cursus Justo Ridiculus Malesuada Tortor</p>
  <p class="bloc">Euismod Etiam Cras Pharetra Porta</p>
</div>
<p> </p>
<div id="conteneur2">
  <p class="bloc">Adipiscing Magna Ullamcorper Sollicitudin
Justo</p>
  <p class="bloc">Tortor Euismod Magna Risus Ultricies</p>
  <p class="bloc">Cursus Justo Ridiculus Malesuada Tortor</p>
  <p class="bloc">Euismod Etiam Cras Pharetra Porta</p>
</div>
<p> </p>
<div id="conteneur3">
  <p class="bloc">Adipiscing Magna Ullamcorper Sollicitudin
Justo</p>
  <p class="bloc">Tortor Euismod Magna Risus Ultricies</p>
  <p class="bloc">Cursus Justo Ridiculus Malesuada Tortor</p>
  <p class="bloc">Euismod Etiam Cras Pharetra Porta</p>
```

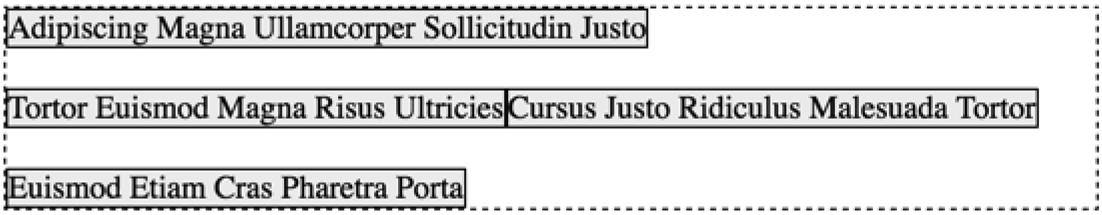
```
</div>
<p> </p>
<div id="conteneur4">
   <p class="bloc">Adipiscing Magna Ullamcorper Sollicitudin
Justo</p>
   <p class="bloc">Tortor Euismod Magna Risus Ultricies</p>
   <p class="bloc">Cursus Justo Ridiculus Malesuada Tortor</p>
   <p class="bloc">Euismod Etiam Cras Pharetra Porta</p>
</div>
</body>
</html>
```

L’affichage obtenu :









Les propriétés pour les items Flexbox

1. L’ordre d’affichage

La propriété order permet de déterminer l’ordre d’affichage des items, indépendamment de leur déclaration dans le code HTML.

Les valeurs de cette propriété sont des nombres entiers. Plus la valeur est petite (nous pouvons mettre des valeurs négatives), plus l’item Flexbox s’affiche en premier dans la liste des items. La valeur par défaut est 0. Si aucune valeur n’est indiquée, c’est l’ordre de déclaration des items dans le code HTML qui prévaut.

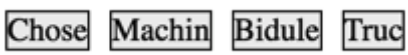
Voici un exemple où seuls deux items utilisent la propriété order.

Le code HTML/CSS :

```
<!DOCTYPE html>
```

```
<html lang="fr">
<head>
   <title>Ma mise en page</title>
   <meta charset="UTF-8" />
   <style>
      #conteneur {
         display: flex;
      }
      p.bloc {
         background-color: #eee;
         border: solid 1px #000;
         margin-right: 10px;
      }
      .truc {
         order: 2;
      }
      .chose {
         order: -1;
      }
   </style>
</head>
<body>
<div id="conteneur">
   <p class="bloc truc">Truc</p>
   <p class="bloc">Machin</p>
   <p class="bloc">Bidule</p>
   <p class="bloc chose">Chose</p>
</div>
</body>
</html>
```

L'affichage obtenu :



- *Le bloc **Chose** s'affiche en premier : order: -1. C'est la valeur la plus faible, il s'affiche donc en premier.*
- *Le bloc **Machin** s'affiche en deuxième. Il n'y a pas de propriété order de définie, la valeur par défaut est donc 0.*
- *Le bloc **Bidule** s'affiche en troisième. Il n'y a pas de propriété order de définie, la valeur par défaut est donc 0. Dans le code HTML, il est déclaré après le bloc **Machin**, il s'affiche donc après lui.*
- *Le bloc **Truc** s'affiche en dernier : order: 2. C'est la valeur la plus élevée, il s'affiche donc en dernier.*

2. L'alignement d'un item Flexbox

La propriété align-self permet de faire une exception pour l'alignement d'un item Flexbox. Les valeurs de cette propriété reprennent celle de la propriété align-items. La valeur align-self: auto reprend la valeur globale définie dans la propriété align-items.

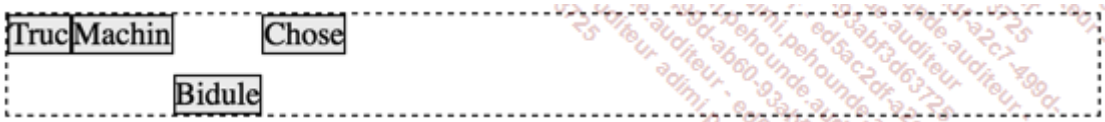
Voici un exemple où le troisième bloc possède un alignement différent.

Le code HTML/CSS :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <title>Ma mise en page</title>
  <meta charset="UTF-8" />
  <style>
    #conteneur {
      display: flex;
      flex-direction: row;
      align-items: flex-start;
      border: dashed 1px #000;
      height: 50px;
    }
    .bidule {
      align-self: flex-end;
    }
    p.bloc {
      background-color: #eee;
```

```
        border: solid 1px #000;
        margin: 0;
      }
    </style>
</head>
<body>
<div id="conteneur">
    <p class="bloc">Truc</p>
    <p class="bloc">Machin</p>
    <p class="bloc bidule">Bidule</p>
    <p class="bloc">Chose</p>
</div>
</body>
</html>
```

L'affichage obtenu :



3. L'agrandissement des items Flexbox

La propriété flex-grow détermine si la taille d'un item Flexbox peut s'agrandir ou pas, dans l'espace disponible restant dans son conteneur Flexbox parent. Cet agrandissement se fait dans la largeur ou dans la hauteur, selon l'axe principal défini. Par défaut la valeur est de 0, l'item ne peut pas s'agrandir. Un item qui a une valeur de 2 pourra s'agrandir deux fois plus qu'un item ayant une valeur 1.

Un exemple avec quatre items.

Le code HTML/CSS :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <title>Ma mise en page</title>
    <meta charset="UTF-8" />
```

```
    <style>
      #conteneur {
         display: flex;
         flex-direction: row;
         border: dashed 1px #000;
      }
      .truc .chose {
         flex-grow: 0;
      }
      .machin {
         flex-grow: 2
      }
      .bidule {
         flex-grow: 1;
      }
      p.bloc {
         background-color: #eee;
         border: solid 1px #000;
         margin: 0;
      }
    </style>
</head>
<body>
<div id="conteneur">
    <p class="bloc truc">Truc</p>
    <p class="bloc machin">Machin</p>
    <p class="bloc bidule">Bidule</p>
    <p class="bloc chose">Chose</p>
</div>
</body>
</html>
```

L’affichage obtenu :

4. La réduction des items Flexbox

La propriété flex-shrink est un peu l'inverse de flex-grow. Elle détermine si un item Flexbox peut réduire sa largeur ou sa hauteur, selon l'axe principal défini, en fonction de l'espace disponible dans son conteneur Flexbox parent. La valeur par défaut est 1 et indique que l'item peut réduire sa taille. Si la valeur est 0, l'item ne peut pas réduire sa taille.

Voici un exemple où le conteneur Flexbox fait 300 pixels de large et où chaque item Flexbox fait 100 pixels de large. Les items ne peuvent donc pas être correctement affichés, puisqu'il "manque" 100 pixels.

Avec la propriété flex-shrink nous pouvons "rétrécir" certains items.

Voici un exemple.

Le code HTML/CSS :

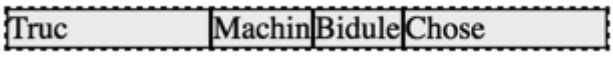
```
<!DOCTYPE html>
<html lang="fr">
<head>
  <title>Ma mise en page</title>
  <meta charset="UTF-8" />
  <style>
    #conteneur {
      display: flex;
      flex-direction: row;
      border: dashed 1px #000;
      width: 300px;
    }
    .truc, .chose {
      flex-shrink: 0;
    }
    .machin, .bidule {
      flex-shrink: 1;
    }
```

```
        .bloc {
            background-color: #eee;
            border: solid 1px #000;
            margin: 0;
            width: 100px;
        }
    </style>
</head>
<body>
<div id="conteneur">
    <p class="bloc truc">Truc</p>
    <p class="bloc machin">Machin</p>
    <p class="bloc bidule">Bidule</p>
    <p class="bloc chose">Chose</p>
</div>
</body>
</html>
```

Les premier et dernier items ne peuvent pas se réduire : flex-shrink: 0.

Les deuxième et troisième items peuvent se réduire : flex-shrink: 1.

L'affichage obtenu :



5. L'indication de taille des items

La propriété flex-basis permet d'indiquer la largeur ou la hauteur (selon l'axe principal) des items Flexbox, avant l'application d'un agrandissement (flex-grow) ou d'une réduction (flex-shrink).

Si les propriétés d'agrandissement et de réduction sont à 0, donc non autorisées, nous pouvons indiquer une largeur ou une hauteur selon l'axe principal défini, tant qu'il n'y a pas de conflit éventuel avec les propriétés max-width ou min-width.

Voici les valeurs possibles :

- *flex-basis: auto : la taille de l'item (largeur ou hauteur) est définie par son conteneur Flexbox parent. C'est la valeur par défaut.*
- *flex-basis: valeur : la taille de l'item (largeur ou hauteur) est définie par la valeur indiquée.*
- *flex-basis: 0 : la taille de l'item (largeur ou hauteur) est nulle.*

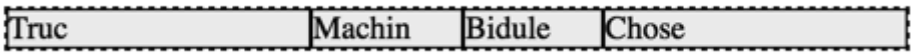
Voici un exemple où le conteneur Flexbox fait 450 pixels de large. Les premier et dernier items ne peuvent ni s'agrandir, ni se réduire et ont une largeur indicative de 150 pixels. Les deuxième et troisième items peuvent s'agrandir ou se réduire et n'ont pas d'indication de largeur. Tous les deux occuperont la place disponible restante.

Le code HTML/CSS :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <title>Ma mise en page</title>
  <meta charset="UTF-8" />
  <style>
    #conteneur {
      display: flex;
      flex-direction: row;
      border: dashed 1px #000;
      width: 450px;
    }
    .truc, .chose {
      flex-grow: 0;
      flex-shrink: 0;
      flex-basis: 150px
    }
    .machin, .bidule {
      flex-grow: 1;
      flex-shrink: 1;
    }
    .bloc {
      background-color: #eee;
```

```
        border: solid 1px #000;
        margin: 0;
      }
    </style>
  </head>
  <body>
  <div id="conteneur">
    <p class="bloc truc">Truc</p>
    <p class="bloc machin">Machin</p>
    <p class="bloc bidule">Bidule</p>
    <p class="bloc chose">Chose</p>
  </div>
  </body>
  </html>
```

L'affichage obtenu :



6. La propriété en syntaxe courte

La propriété flex est la syntaxe courte des propriétés flex-grow, flex-shrink et flex-basis.

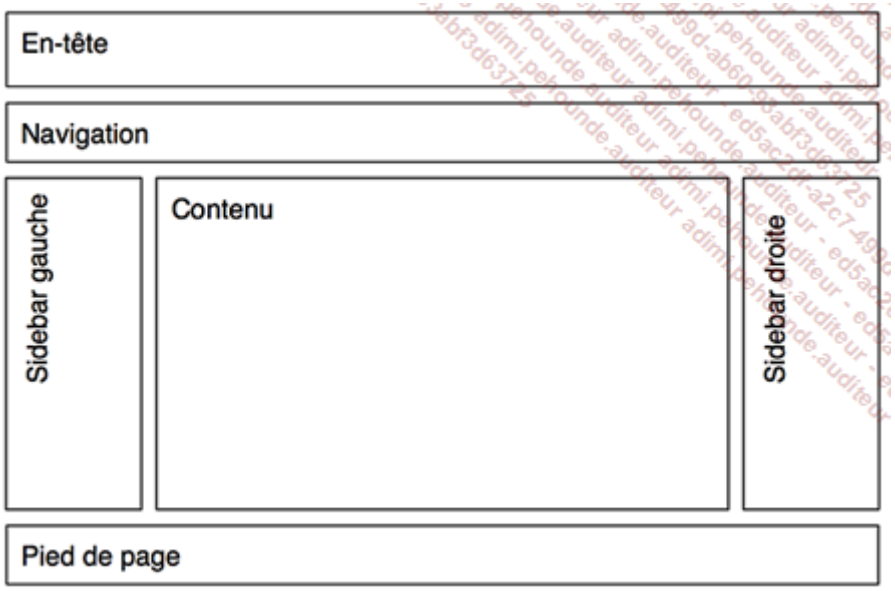
La valeur par défaut est flex: 0 1 auto, soit :

- l'item ne peut pas s'agrandir,
- l'item ne peut pas se réduire,
- la taille de l'item est déterminée par son conteneur.

Réalisation d'une mise en page Flexbox

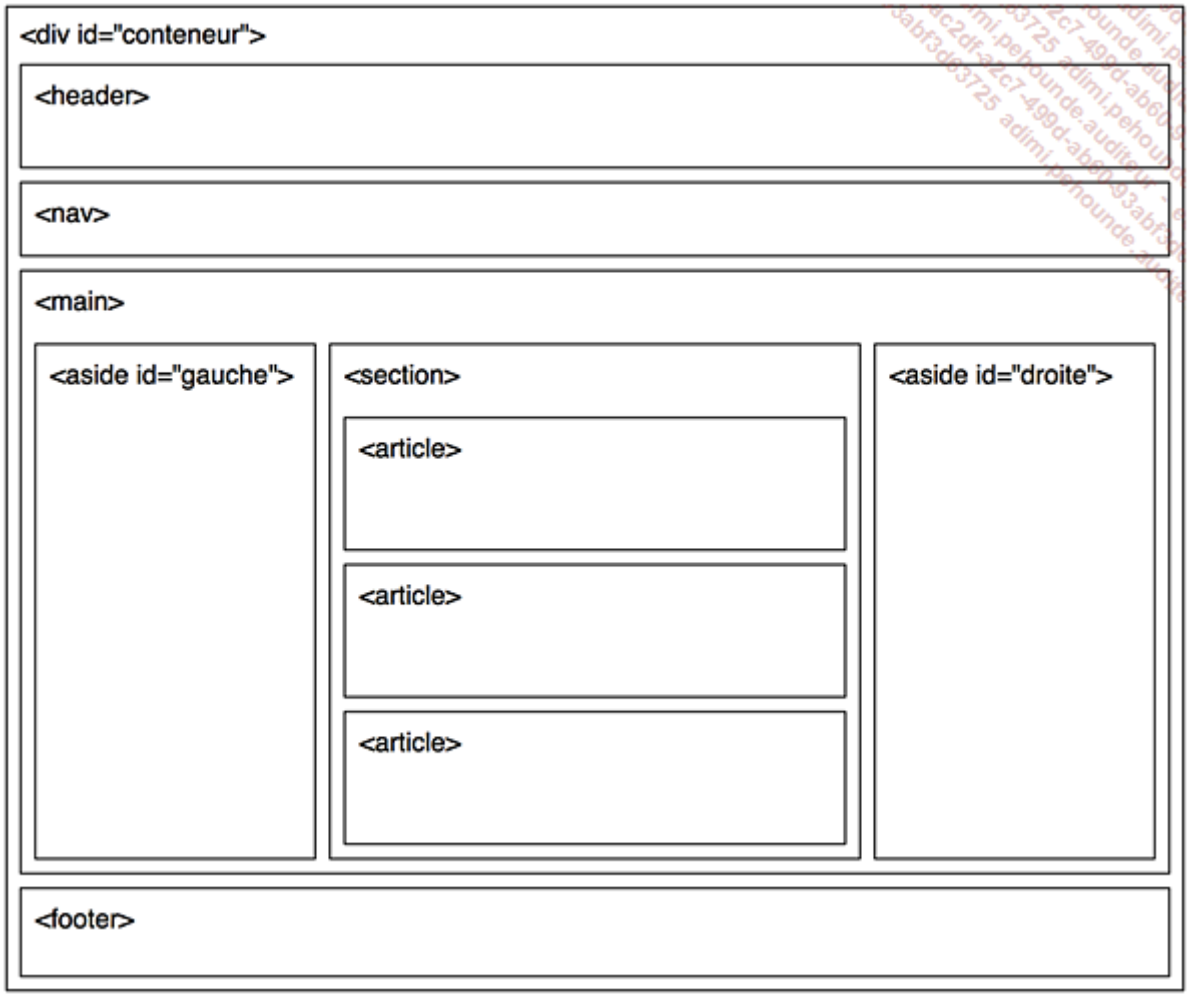
1. Les objectifs

Nous allons réaliser une mise en page Flexbox, qui sera responsive. Voici la maquette de cette mise en page :



- *un en-tête contiendra le nom et le slogan du site,*
- *une barre de navigation affichera des liens internes au site,*
- *deux sidebar seront placées à gauche et à droite du contenu,*
- *le contenu principal sera affiché au centre et pourra contenir des articles par exemple.*

Voici la structure des boîtes composant cette mise en page :



- *Toute la mise en page sera placée dans une boîte .*
- *L'en-tête sera inséré dans un élément <header> qui contiendra un élément <h1> pour le nom du site et un élément <h2> pour le slogan.*
- *La barre de navigation utilisera un élément <nav>. Chaque lien <a> sera placé dans un élément <p>.*
- *La partie centrale est placée dans un élément <main>. Cette partie centrale contiendra à gauche et à droite une sidebar (<aside id="gauche"> et <aside id="droite">). Le contenu rédactionnel sera placé dans un élément <section> et chaque article sera placé dans un élément <article>.*
- *Le pied de page sera inséré dans un élément <footer>.*

2. Le conteneur principal

Toute la mise en page est incluse dans une boîte .

Pour la mise en page, une seule règle CSS est déclarée :

#conteneur {

```
    margin: 0 auto;
    max-width: 980px;
}
```

La boîte a une taille maximale de 980 pixels et est centrée dans la fenêtre du navigateur.

3. L'en-tête

L'en-tête est placé dans un élément <header>. Il contient un élément <h1> pour le nom du site et un élément <h2> pour le slogan.

Le code HTML :

```
<header>
    <h1>Mon site web</h1>
    <h2>Vivamus sagittis lacus vel augue laoreet</h2>
</header>
```

Pour la mise en forme CSS, l'en-tête est déclaré comme un conteneur Flexbox, avec un axe principal vertical. En effet, les items enfants, <h1> et <h2>, doivent être placés les uns sous les autres.

Le code CSS :

```
header {
    display: flex;
    flex-direction: column;
}
```

La mise en forme classique des titres :

```
h1, h2, h3 {
    margin: 0;
}
```

La mise en forme classique de l'en-tête :

```
/* Toutes les boîtes */
header, nav, #gauche, section, #droite, footer {
    padding: 5px;
}
```

```
/* En-tête et pied de page */
header, footer {
    background-color: #666;
    color: #fff;
}
```

L'affichage de l'en-tête :

4. La barre de navigation

La barre de navigation affiche une série de liens hypertextes. Elle est placée dans un élément <nav>. Chaque lien <a> est placé dans un élément <p>. Nous n'utilisons pas la classique liste de lien <ul>, car avec les propriétés Flexbox nous allons disposer d'une mise en forme rapide et simplifiée.

Le code HTML :

```
<nav>
    <p><a href="#">Accueil</a></p>
    <p><a href="#">Projets</a></p>
    <p><a href="#">Articles</a></p>
    <p><a href="#">Concepts</a></p>
    <p><a href="#">Réalisations</a></p>
    <p><a href="#">Clients</a></p>
    <p><a href="#">Boîte à idées</a></p>
    <p><a href="#">Contact</a></p>
</nav>
```

L'élément <nav> est bien sûr un conteneur Flexbox, puisqu'il contient des éléments <p> en item Flexbox.

Le code CSS :

```
nav, main {
    display: flex;
}
```

Nous souhaitons que les liens <a> dans les éléments <p> s'affichent les uns à côté des autres. Et il s'avère que la propriété par défaut flex-direction: row le fait toute seule ! Il n'y a pas à changer autre chose, comme ce que nous devons faire avec une liste de lien dans un élément <ul>.

Enfin, nous souhaitons que les liens dans les éléments <p> soient uniformément répartis dans le conteneur Flexbox parent <nav>. La propriété justify-content: space-around le fait très simplement.

Le code CSS :

```
nav {
    justify-content: space-around;
}
```

Voilà comment avec deux propriétés CSS nous obtenons ce que nous voulons.

La mise en forme CSS classique de la barre de navigation :

```
/* Barre de navigation */
nav {
    background-color: #aaa;
}
nav a {
    text-decoration: none;
    color: #000;
}
nav a:hover {
    color: #fff;
}
```

L'affichage obtenu :

5. La partie principale

La partie principale de la mise en page est placée dans un élément <main> qui est bien sûr un conteneur Flexbox.

Le code HTML :

```
<main>
    ...
</main>
```

Les items Flexbox doivent être affichés les uns à côté des autres, c'est la valeur par défaut de la propriété flex-direction: row.

Le code CSS :

```
nav, main {
    display: flex;
}
```

6. Les sidebars

Dans la partie principale <main>, de chaque côté, se trouve une sidebar. Celle de gauche <aside id="gauche"> contient des liens et celle de droite <aside id="droite"> contient une liste d'articles.

Le code HTML de la sidebar de gauche :

```
<aside id="gauche">
    <h3>Liens</h3>
    <p><a href="#">Lien 1</a></p>
    <p><a href="#">Lien 2</a></p>
    <p><a href="#">Lien 3</a></p>
    <p><a href="#">Lien 4</a></p>
    <p><a href="#">Lien 5</a></p>
    <p><a href="#">Lien 6</a></p>
</aside>
```

Le code HTML de la sidebar de droite :

```
<aside id="droite">
<h3>Articles</h3>
    <p><a href="#">Article 1</a></p>
    <p><a href="#">Article 2</a></p>
    <p><a href="#">Article 3</a></p>
    <p><a href="#">Article 4</a></p>
```

</aside>

Pour la mise en page, nous souhaitons avoir une largeur de 20% par rapport au conteneur Flexbox parent <main>. Nous allons utiliser la propriété en syntaxe courte flex. Les deux premières valeurs sont à 0, car nous ne souhaitons pas d'agrandissement, ni de réduction des tailles. La troisième valeur est tout simplement 20%, la taille souhaitée.

Le code CSS :

```
#droite, #gauche {
    flex: 0 0 20%;
}
```

Cette syntaxe courte équivaut à :

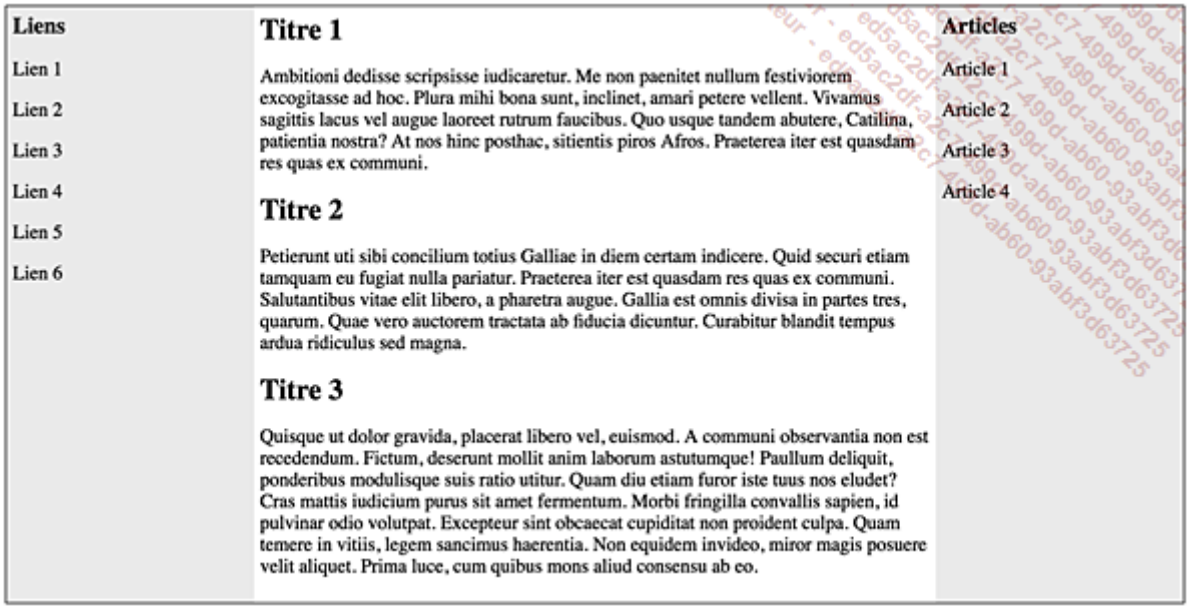
```
flex-grow: 0;
flex-shrink: 0;
flex-basis: 20%;
```

La mise en forme CSS classique :

```
header, nav, #gauche, section, #droite, footer {
    padding: 5px;
}
#gauche, #droite {
    background-color: #eee;
}
#gauche a, #droite a {
    text-decoration: none;
    color: #000;
}
```

Notez un avantage vraiment très pratique à utiliser la mise en page Flexbox pour la hauteur des blocs et les fonds colorés : quelle que soit la quantité de contenu rédactionnel, les hauteurs des items Flexbox frères sont toujours identiques et le fond coloré est parfaitement affiché.

L'affichage obtenu avec la partie centrale rédactionnelle :



7. La partie centrale rédactionnelle

La partie centrale rédactionnelle est placée dans un élément <section>. C'est un item Flexbox du conteneur Flexbox <main>. Il contient trois éléments <article>, constitués pour chacun d'entre eux d'un titre, <h2> et d'un paragraphe <p>.

Le code HTML :

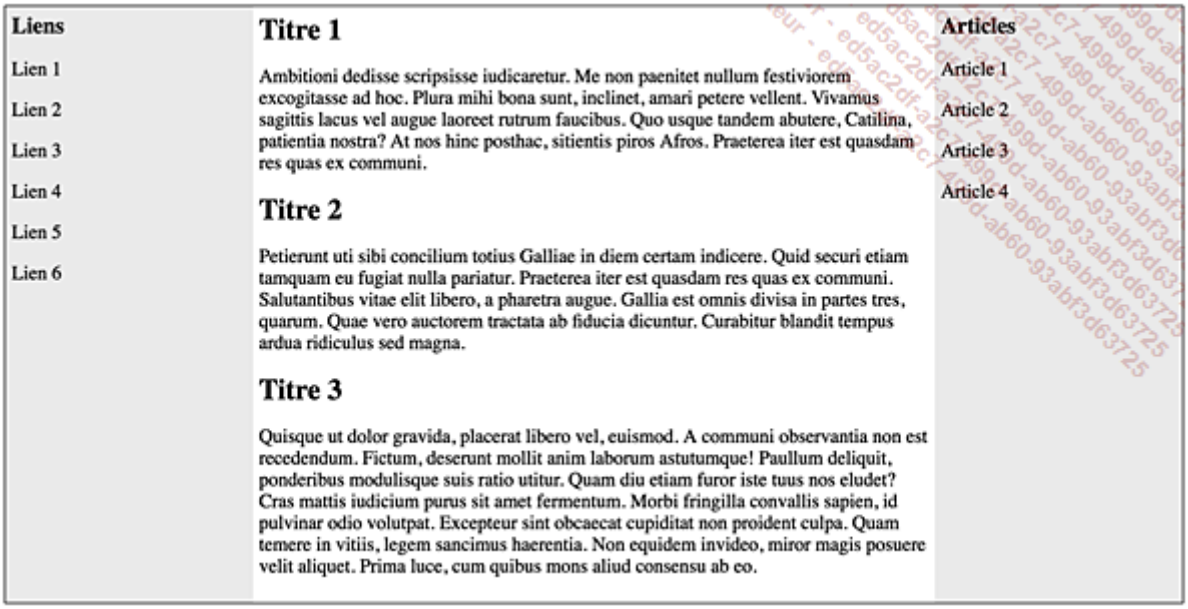
```
<section>
    <article>
      <h2>...</h2>
      <p>...</p>
    </article>
    <article>
      <h2>...</h2>
      <p>...</p>
    </article>
    <article>
      <h2>...</h2>
      <p>...</p>
    </article>
</section>
```

La mise en "colonne" est due au fait que l'élément <section> soit un item Flexbox. Sa largeur occupe la place disponible restante déterminée par la largeur des deux sidebars.

La mise en forme CSS classique :

```
/* Toutes les boîtes */
header, nav, #gauche, section, #droite, footer {
    padding: 5px;
}
/* Marges des titres */
h1, h2, h3 {
    margin: 0;
}
```

L'affichage obtenu avec les deux sidebars :



8. Le pied de page

Le pied de page est placé dans un élément <footer> et ne contient qu'un élément <p>.

Le code HTML :

```
<footer>
    <p>Qui ipsorum...</p>
</footer>
```

C’est un conteneur Flexbox qui affiche son item Flexbox centré :

```
footer {
    display: flex;
    justify-content: center;
}
```

La mise en forme CSS classique :

```
/* Toutes les boîtes */
header, nav, #gauche, section, #droite, footer {
    padding: 5px;
}
/* En-tête et pied de page */
header, footer {
    background-color: #666;
    color: #fff;
}
```

Voici l’affichage obtenu :

9. Le code complet

Voici le code complet de cette mise en page :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <title>Ma mise en page</title>
  <meta charset="UTF-8" />
  <style>
    #conteneur {
      margin: 0 auto;
      max-width: 980px;
```

```
}
/***** Mise en page flexible *****/
header {
   display: flex;
   flex-direction: column;
}
nav, main {
   display: flex;
}
nav {
   justify-content: space-around;
}
#droite, #gauche {
   flex: 0 0 20%;
}
footer {
   display: flex;
   justify-content: center;
}
/***** Mise en forme *****/
/* Toutes les boîtes */
header, nav, #gauche, section, #droite, footer {
   padding: 5px;
}
/* En-tête et pied de page */
header, footer {
   background-color: #666;
   color: #fff;
}
/* Barre de navigation */
nav {
   background-color: #aaa;
}
```

```
      nav a {
        text-decoration: none;
        color: #000;
      }
      nav a:hover {
        color: #fff;
      }
      /* Barres latérales */
      #gauche, #droite {
        background-color: #eee;
      }
      #gauche a, #droite a {
        text-decoration: none;
        color: #000;
      }
      /* Marges des titres */
      h1, h2, h3 {
        margin: 0;
      }
    </style>
</head>
<body>
<div id="conteneur">
    <header>
      <h1>Mon site web</h1>
      <h2>Vivamus sagittis lacus vel augue laoreet</h2>
    </header>
    <nav>
      <p><a href="#">Accueil</a></p>
      <p><a href="#">Projets</a></p>
      <p><a href="#">Articles</a></p>
      <p><a href="#">Concepts</a></p>
      <p><a href="#">Réalisations</a></p>
```

```
    <p><a href="#">Clients</a></p>
    <p><a href="#">Boîte à idées</a></p>
    <p><a href="#">Contact</a></p>
  </nav>
  <main>
    <aside id="gauche">
      <h3>Liens</h3>
      <p><a href="#">Lien 1</a></p>
      <p><a href="#">Lien 2</a></p>
      <p><a href="#">Lien 3</a></p>
      <p><a href="#">Lien 4</a></p>
      <p><a href="#">Lien 5</a></p>
      <p><a href="#">Lien 6</a></p>
    </aside>
    <section>
      <article>
        <h2>Titre 1</h2>
        <p>Ambitioni dedisse scripsisse...</p>
      </article>
      <article>
        <h2>Titre 2</h2>
        <p>Petierunt uti sibi...</p>
      </article>
      <article>
        <h2>Titre 3</h2>
        <p>Quisque ut dolor gravida...</p>
      </article>
    </section>
    <aside id="droite">
      <h3>Articles</h3>
      <p><a href="#">Article 1</a></p>
      <p><a href="#">Article 2</a></p>
      <p><a href="#">Article 3</a></p>
```

```
        <p><a href="#">Article 4</a></p>
      </aside>
    </main>
    <footer>
      <p>Qui ipsorum...</p>
    </footer>
</div>
</body>
</html>
```

Voici l'affichage obtenu :

Réalisation d'une mise en page Flexbox responsive

1. Les objectifs

Actuellement, tout site web se doit d'être responsive et à nouveau, la mise en page Flexbox nous propose des facilités d'organisation et d'affichage.

À partir de l'exemple précédent, voyons quels sont nos objectifs pour une mise en page responsive.

- *Nous souhaitons avoir une règle @media pour une largeur maximale de 640 pixels.*
- *Dans la barre de navigation, les liens <a> inclus dans les éléments <p> doivent pouvoir aller à la ligne si besoin est.*
- *Pour la partie principale, nous souhaitons que les items Flexbox s'affichent les uns sous les autres.*
- *Toujours, pour la partie principale, nous souhaitons que la sidebar de gauche s'affiche en premier, suivie par la sidebar de droite et ensuite la partie centrale avec ses trois articles.*
- *Les liens dans chaque sidebar doivent s'afficher les uns à côté des autres.*
- *Les titres et les liens des sidebars doivent être correctement alignés.*
- *Les liens des sidebars doivent être séparés par un tiret.*

2. La règle @media

Nous souhaitons une mise en page responsive pour les écrans de diffusion d'une largeur maximale de 640 pixels. Toutes les règles CSS suivantes seront placées dans cette règle @media.

```
@media (max-width: 640px) {
    ...
}
```

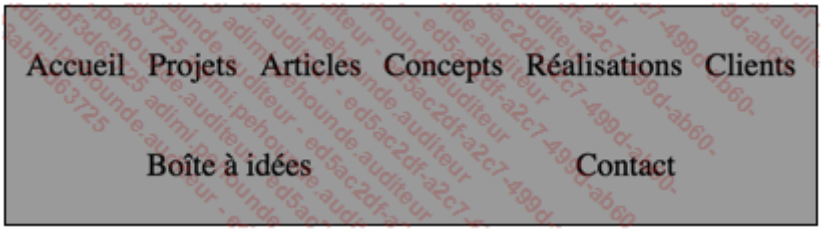
3. La barre de navigation

Les liens de la barre de navigation doivent pouvoir aller à la ligne si c'est nécessaire, si la taille de l'écran de diffusion est petite. Nous allons tout simplement utiliser la propriété flex-wrap: wrap.

Le code CSS :

```
nav {
    flex-wrap: wrap;
}
```

L'affichage obtenu sur un petit écran :



4. L'affichage de la partie principale

Sur grand écran, les items Flexbox du conteneur <main> s'affichent les uns à côté des autres, avec la propriété par défaut flex-direction: row.

Nous souhaitons maintenant que les items Flexbox s'affichent les uns sous les autres. Nous changeons la valeur de cette propriété :

```
main {
    flex-direction: column;
}
```

Maintenant nous devons changer l'ordre d'affichage des items Flexbox du conteneur <main>. Pour cela nous utilisons la très pratique propriété order.

```
#gauche {
    order: -2;
}
#droite {
    order: -1;
}
```

Nous ne déclarons rien pour l'item Flexbox <section> qui utilise par défaut la valeur 0 et s'affichera en dernier, puisqu'il a la valeur la plus élevée.

5. L'affichage dans les sidebars

Nous souhaitons que les titres <h3> et les liens <a> dans les paragraphes <p> soient affichés les uns à côté des autres. Il faut donc changer la propriété flex-direction qui était à column par défaut, en lui attribuant la valeur row.

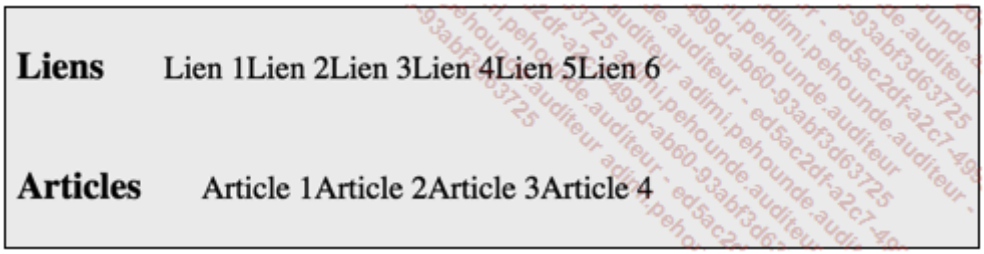
```
#gauche, #droite {
    display: flex;
    flex-direction: row;
}
```

Voici l'affichage obtenu :

Maintenant il faut que les titres et les liens soient bien alignés. Il suffit simplement d'utiliser la propriété align-items: baseline.

```
#gauche, #droite {
    display: flex;
    flex-direction: row;
    align-items: baseline;
}
```

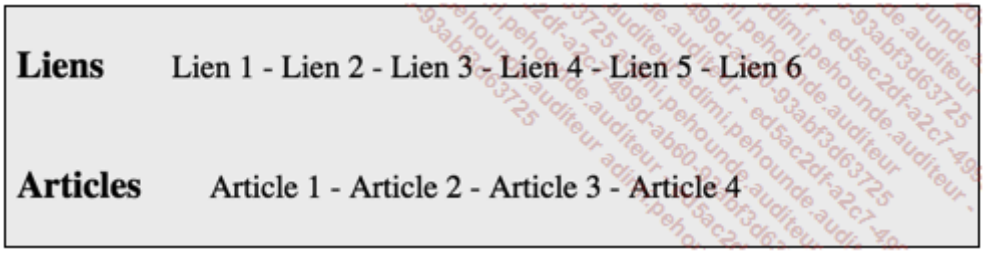
Voici l'affichage obtenu :



Pour avoir des libellés bien séparés par un tiret et des espaces avant et après, il suffit d'utiliser les pseudo-éléments ::after et ::before sur les paragraphes <p> qui contiennent les liens <a>.

```
aside p::after {
    content: " -";
}
aside p:last-child::after {
    content: "";
}
aside p::before {
    content: "\a0"
}
```

Voici l'affichage obtenu :



6. Les règles CSS pour une page responsive

Voici toutes les règles CSS pour la règle @media().

```
/***** Mise en page Responsive *****/
@media (max-width: 640px) {
   /* Mise en page en flexible */
   nav {
      flex-wrap: wrap;
   }
   main {
      flex-direction: column;
   }
   #gauche {
      order: -2;
   }
   #droite {
      order: -1;
   }
   #gauche, #droite {
      display: flex;
      flex-direction: row;
      align-items: baseline;
   }
   /* Mise en forme classique */
   aside h3 {
      margin-right: 30px;
   }
   aside p::after {
      content: " -";
   }
   aside p:last-child::after {
      content: "";
   }
```

```
    aside p::before {
        content: "\a0"
    }
}
```

Voici l'affichage obtenu :