La mise en page avec CSS

Table des matières

I. Contexte

Durée : 1 h

Environnement de travail : Outil en ligne tierce

Pré-requis : Connaître les bases de CSS

Contexte

Nous allons découvrir les techniques de mise en page et de positionnement des éléments HTML grâce à différentes propriétés CSS. Il sera alors possible de sélectionner des éléments contenus dans une page web et de contrôler leur emplacement, que ce soit par rapport aux autres éléments de la page, par rapport à leur parent ou encore à la fenêtre du navigateur. Les techniques de mise en page abordées ici seront les propriétés flexbox, position, z-index et float.

II. Position

Objectif

- Contrôler le positionnement des éléments HTML avec la propriété `position`

Mise en situation

La propriété `position` définit le mode de positionnement d'un élément dans une page. Les éléments se positionnent par défaut en fonction de leur ordre d'écriture dans le fichier HTML, mais, grâce à cette propriété, nous allons pouvoir positionner un élément par rapport à un endroit donné de la page, en utilisant les propriétés `top`, `left`, `bottom` et `right`.

Les valeurs possibles pour la propriété `position` sont :

- static
- relative
- fixed
- absolute

static

C'est la valeur par défaut : les éléments se positionnent les uns à la suite des autres en suivant leur ordre d'écriture dans le document HTML.

```
.static {
    position: static;
}
```

relative

En ajoutant des propriétés, notamment les propriétés `top, left, bottom` ou `right`, cela permet d'ajuster la position de l'élément par rapport à sa position initiale, dictée par les autres éléments autour de lui.

```html
<div class="blocRelativeGreen">
    .blocRelativeGreen
</div>
<div class="blocRelativePink">
    .blocRelativePink
</div>
```

```css
.blocRelativeGreen {
    position: relative;
    height: 50px;
    background: #00BFBF;
    color: white;
    padding: 5px;
}
.blocRelativePink {
    position: relative;
    top: -25px;
    left: 25px;
    width: 400px;
    height: 50px;
    background: #FFAEC9;
    color: white;
    padding: 5px;
}
```

Résultat :

fixed

Un élément positionné en `fixed` est placé par rapport au référentiel de la fenêtre du navigateur. Comme avec la valeur relative, nous pouvons utiliser les propriétés `top, left, bottom` et `right`.

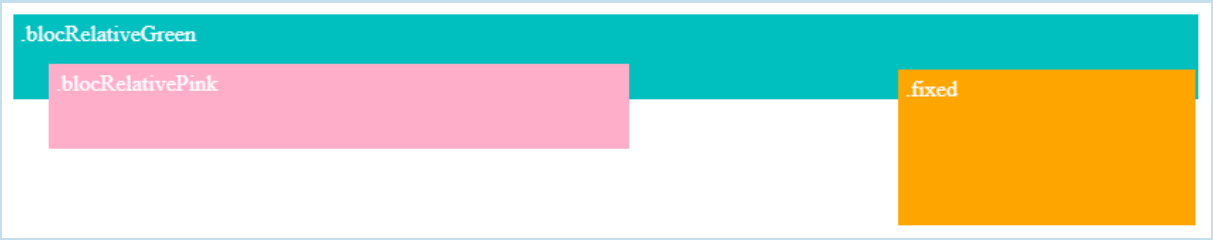
```html
<div class="blocRelativeGreen">
    .blocRelativeGreen
</div>
<div class="blocRelativePink">
    .blocRelativePink
</div>
<div class="blocFixed">
    .fixed
</div>

```

```css
.blocFixed {
    position: fixed;
    bottom: 10px;
    right: 10px;
    width: 200px;
    height: 100px;
    background-color: orange;
    color: white;
    padding: 5px;
}

```

Résultat :

Remarque

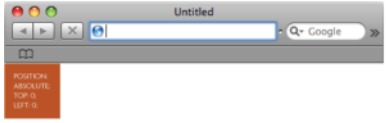
Un élément qui est positionné en `fixed` reste toujours à la même position, et ce même si la page défile ou si la fenêtre du navigateur est redimensionnée.



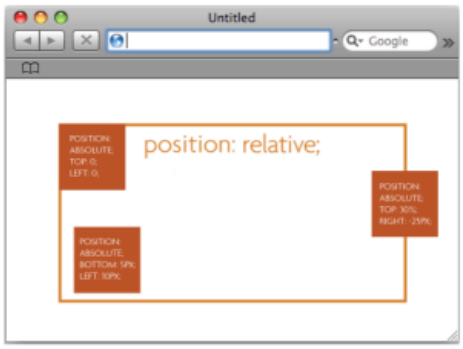
absolute

Un élément positionné avec position: absolute va être positionné par défaut par rapport à l'élément racine, la balise <body>.

Le point de référence pour les propriétés top, left, bottom et right va ainsi être le coin en haut à gauche de la page de votre navigateur.



Si l'élément positionné en position : `absolute` se trouve dans un bloc parent ayant une position différente de *static*, alors le bloc parent devient l'élément de référence.



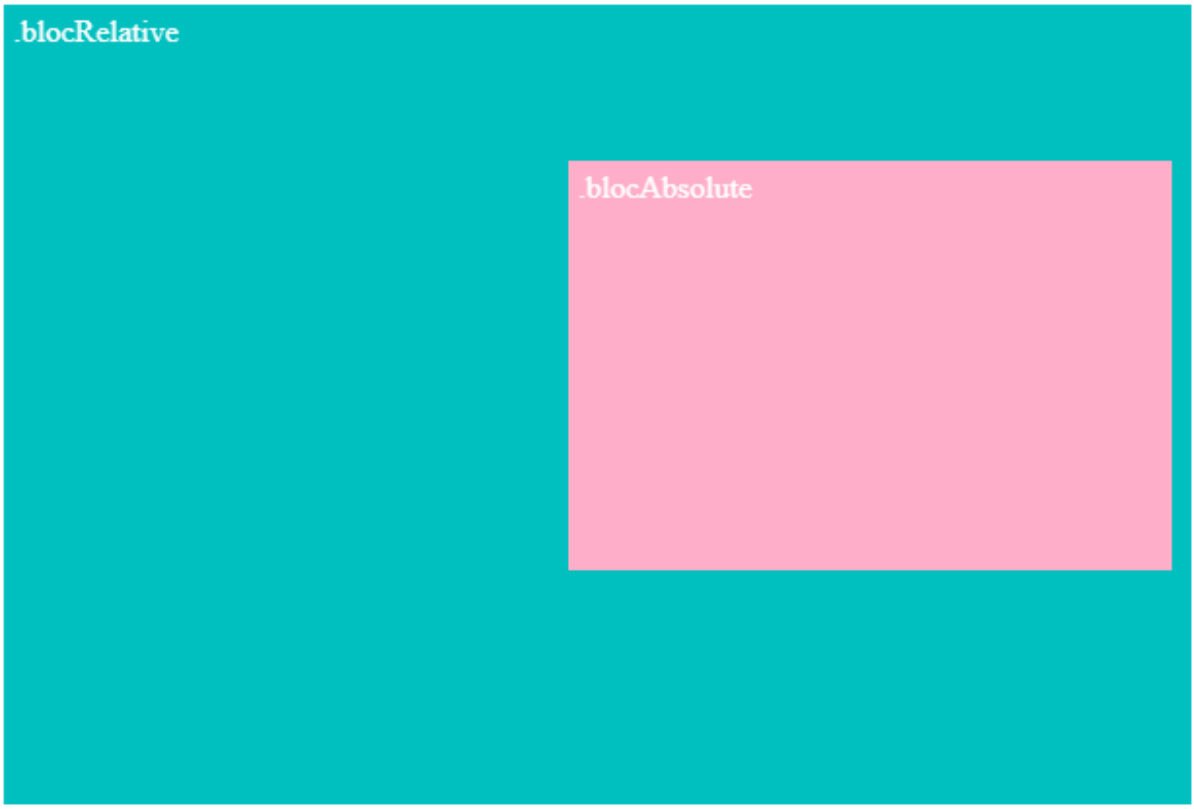
Un élément positionné avec position : `absolute` va être retiré du flux normal de la page. C'est comme si il n'était plus présent. Son espace initial sera alors occupé par les éléments suivants.Un élément positionné avec position : `absolute` va ainsi pouvoir se placer par-dessus d'autres éléments.

Exemple d'un bloc en position : `absolute` placé dans un bloc parent ayant une position différente de static. Le bloc parent devient le bloc de référence.

```
<div class="blocRelative">
    .blocRelative
    <div class="blocAbsolute">
        .blocAbsolute
    </div>
</div>

```

```
.blocRelative {
    position: relative;
    width: 600px;
    height: 400px;
    background: #00BFBF;
    color: white;
    padding: 5px;

}
.blocAbsolute {
    position: absolute;
    bottom: 120px;
    right: 10px;
    width: 300px;
    height:200px;
    background: #FFAEC9;
    color: white;
    padding: 5px;
}

```

Résultat :



Syntaxe **À retenir**

- Le positionnement `absolute` permet de placer un élément où l'on souhaite sur la page en tenant compte de son parent.
- Le positionnement `fixed` est équivalent au positionnement absolu, mais l'élément concerné restera toujours visible au même endroit, même si l'on descend plus bas dans la page.
- Le positionnement `relative` permet de déplacer un élément par rapport à sa position initiale.

Complément

Documentation MDN sur position[1]

III. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[2]

1 https://developer.mozilla.org/fr/docs/Web/CSS/position
2 https://repl.it/

Question

[solution n°1 p.37]

Nous allons définir une série de boîtes `<div>` sur la page et les positionner de la manière suivante :

- Les boîtes A, B et C doivent être l'une au-dessus de l'autre dans le coin supérieur gauche de la fenêtre
- La boîte C doit mordre sur la partie inférieure de la boîte B sur une surface de 40 pixels
- La boîte D doit être placée dans le coin inférieur droit de la fenêtre. Cette boîte doit rester toujours visible en bas à droite lorsqu'on défile dans la page
- La boîte E doit être placée dans le coin inférieur gauche de la fenêtre. Si on défile dans la page, elle doit finir par disparaître (comme la boîte A et B)

Chaque `<div>` devra avoir une taille de 150 px par 150 px, et avoir une couleur de fond distinctive :

- a : paleturquoise
- b : lightgreen
- c : lightpink
- d : palegoldenrod
- e : lightseagreen


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Document</title>
</head>
<body>
  <div class="a">A</div>
  <div class="b">B</div>
```

```
  <div class="c">C</div>
  <div class="d">D</div>
  <div class="e">E</div>
</body>
</html>
```

Indice :

Pour tester le bon fonctionnement du positionnement des boîtes D et E, vous pouvez utiliser le code suivant à placer tout en haut de vos styles :

```
html, body {
  width: 100%;
  height: 3000px;
  margin: 0;
}
```

IV. Z-index

Objectif

- Positionner/empiler des éléments grâce à la propriété `z-index`

Mise en situation

En positionnant nos différents blocs, il nous arrivera que certains d'entre eux se chevauchent. Afin de gérer l'ordre d'affichage de ceux-ci, nous avons accès à une nouvelle propriété : `z-index`.

La propriété `z-index` est utilisée pour gérer la superposition d'éléments. Plus la valeur est élevée, plus l'élément sera positionné au-dessus des autres sur l'axe Z.

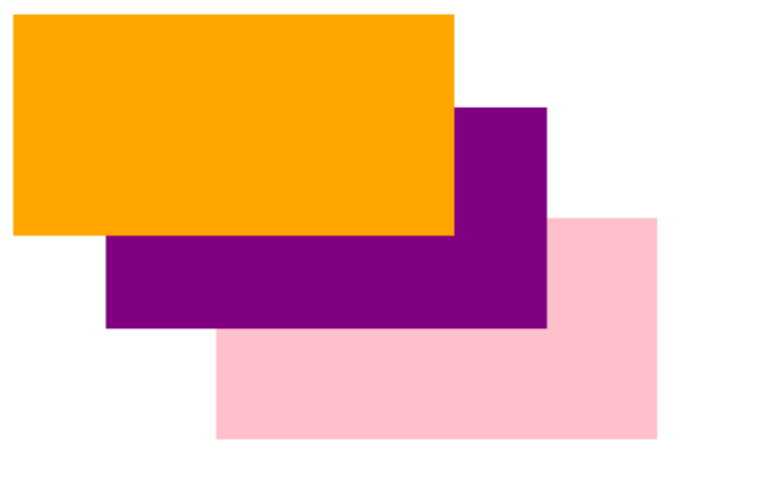
Exemple

```
<div class="index-1"></div>
<div class="index-2"></div>
<div class="index-3"></div>

```

```
.index-1 {
    z-index: 3;
    position : absolute;
    width: 200px;
    height:100px;
    background: orange;
}

.index-2 {
    z-index: 2;
    position : absolute;
    width: 200px;
    height:100px;
    background: purple;
    top: 50px;
    left:50px;
}
.index-3 {
    z-index: 1;
    position : absolute;
```

```
    width: 200px;
    height:100px;
    background: pink;
    top: 100px;
    left:100px;
}

```

Résultat :



Syntaxe **À retenir**

- La propriété `z-index` permet de gérer l'ordre d'affichage des éléments. Plus la valeur est élevée, plus l'élément apparaîtra au premier plan.

Attention

Il convient d'utiliser la propriété `z-index` avec parcimonie. Son utilisation trop intensive pourrait causer des bugs graphiques dont il serait difficile de trouver la cause. En cas de doute, vérifiez sur la documentation[1] pour bien comprendre le *stacking context* ou l'ordre d'affichage des éléments en fonction de leur état.

Complément

Documentation MDN sur z-index[2]

V. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[3]

1 Documentation MDN z-index
2 https://developer.mozilla.org/fr/docs/Web/CSS/z-index
3 https://repl.it/

Question

[solution n°2 p.37]

Nous avons repris le code précédent, un bouton **Remonter** a été ajouté, situé tout en bas à droite de la fenêtre. Une barre grise de pied de page a également été ajoutée : vous pouvez voir que le pied de page masque notre bouton.

Nous allons faire en sorte que le bouton soit visible et passe au-dessus du pied de page. Il y a deux solutions à ce problème.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Document</title>
</head>
<body>
  <div class="header">
    <div class="logo-wrapper">
      <img
        src="deep-space-inc.svg"
        alt="Deep Space inc. logo"
        class="logo"
      />
      <span class="company-name">Deep Space inc.</span>
    </div>
    <div class="menu">
      <div class="menu-item">A propos</div>
      <div class="menu-item">Contact</div>
    </div>
  </div>
  <div class="content">
    <img src="https://picsum.photos/id/966/300/300.jpg" />
    <img src="https://picsum.photos/id/966/300/300.jpg" />
    <img src="https://picsum.photos/id/966/300/300.jpg" />
    <img src="https://picsum.photos/id/966/300/300.jpg" />
    <img src="https://picsum.photos/id/966/300/300.jpg" />
    <img src="https://picsum.photos/id/966/300/300.jpg" />
    <img src="https://picsum.photos/id/966/300/300.jpg" />
    <img src="https://picsum.photos/id/966/300/300.jpg" />
    <img src="https://picsum.photos/id/966/300/300.jpg" />
    <img src="https://picsum.photos/id/966/300/300.jpg" />
  </div>

  <div class="back-to-top">remonter</div>
  <div class="footer"></div>
</body>
</html>
```

```css
body {
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
}

.header {
  display: flex;
  justify-content: space-between;
}

.logo-wrapper,
```

```css
.menu {
  display: flex;
  align-items: center;
}

.logo {
  height: 5rem;
  width: auto;
  float: left;
}

.company-name {
  font-size: 2rem;
  font-weight: 700;
}

.menu-item {
  font-size: 1.4rem;
  font-weight: 700;
  margin-left: 10px;
  margin-right: 10px;
}

.content {
  display: flex;
  flex-wrap: wrap;
  width: 1200px;
  min-height: 600px;
  padding: 30px;
  margin: auto;
  background:#ffeeee;
}

.footer {
  position: fixed;
  bottom: 0;
  left: 0;
  width: 100%;
  height: 150px;
  background: hsl(0, 0%, 10%);
}

.back-to-top {
  position: absolute;
  bottom: 10px;
  right: 20px;
  background: white;
  font-weight: 600;
  padding: 10px 20px;
  border-radius: 6px;
}
```

VI. Float

Objectifs

- Positionner des éléments grâce à la propriété `float`
- Contrôler le comportement des éléments autour d'un flottant avec la propriété `clear`

Mise en situation

La propriété CSS float permet de sortir un élément du flux normal de la page et de le faire “flotter” contre un bord de son élément parent conteneur ou contre un autre élément flottant.

Une utilisation bien connue de la propriété float est de s’en servir pour faire flotter une image à droite ou à gauche d’un texte et ainsi l’entourer avec du texte.

La propriété float est donc une autre propriété qui va impacter la disposition dans la page et qu’il convient de manier avec précaution pour ne pas obtenir de comportement indésirable.

Nous avons le choix entre plusieurs valeurs pour la propriété float :

- **none :** Valeur par défaut, l'élément n'est pas un élément flottant
- **left :** L'élément va venir se placer à la limite gauche de son élément parent ou va être déplacé sur la gauche jusqu'à atteindre un autre élément ayant float: left
- **right :** L'élément va venir se placer à la limite droite de son élément parent ou va être déplacé sur la droite jusqu'à atteindre un autre élément ayant float: right

```
1 <div class="parent">
2   <img src="https://picsum.photos/150/150.jpg" alt="" class="float" />
3       <p>Lucas ipsum dolor sit amet ben wookiee hutt calamari jango mon jinn wampa coruscant
  wicket. Yoda wedge dantooine boba kessel hutt mustafar antilles. Binks coruscant hutt leia
  yavin. Darth darth jinn skywalker. Calrissian aayla mon darth kenobi naboo solo darth. Endor
  c-3p0 owen dooku darth mace kessel hutt droid. Jabba darth gamorrean skywalker. </p>
4     </div>
```

```css
.parent{
    width: 300px;
    border: 1px solid red;
    padding:25px;
}
.float {
    float:left;
}
```

Et si on ne souhaite pas qu'un élément se place à côté d'un élément « flottant » ?

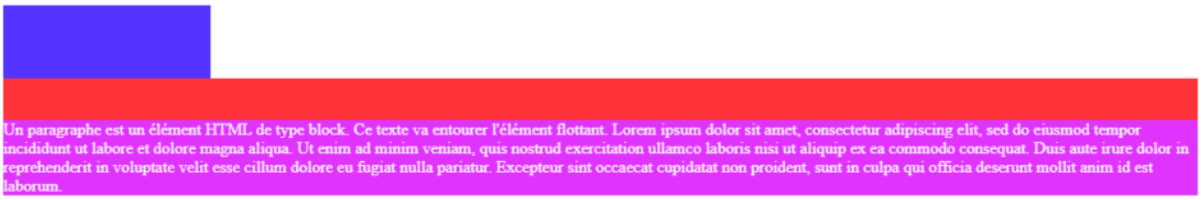
La propriété CSS `clear` va nous permettre de résoudre ce problème :

- `clear: none` : C'est la valeur par défaut. Pas de restriction, les éléments se positionneront à côté d'éléments flottants s'il y en a.
- `clear: left` : Empêche un élément de se placer à côté d'éléments placés en `float: left`
- `clear: right` : Empêche un élément de se placer à côté d'éléments placés en `float: right`
- `clear: both` : Empêche un élément de se placer à côté d'éléments placés en `float: left` ou `float: right`

```html
<div class="floatBlue"></div>
<div></div>

```

```css
.floatBlue {
    width: 200px;
    height: 70px;
    float: left;
    background-color: #5233FF;
}

div {
height: 40px;
background-color: #FF3339;
clear: left;
}
```

Résultat :



L'élément `div` ayant la propriété `clear: left` n'accepte pas d'être placé à côté d'un flottant, donc du bloc `floatBlue`, et passe donc en dessous.

Syntaxe **À retenir**

Pour placer un bloc par rapport au contenu principal, on utilise :

- `float: left` pour le placer à gauche
- `float: right` pour le placer à droite

Pour empêcher un élément d'être placé après un élément flottant, on utilise :

- `clear: left` pour empêcher de le placer à côté d'un élément en `float: left`
- `clear: right` pour empêcher de le placer à côté d'un élément en `float: right`
- `clear: both` pour les deux

Remarque **Flexbox vs float**

Nous venons de voir deux manières de gérer le *layout* de la page pour afficher des éléments sur une même ligne. `Flexbox` est plus récent et permet de positionner plus finement les éléments sur la page, il est donc à préconiser par rapport à `float`, sauf dans certains scénarios particuliers où float est adapté, comme venir faire s'enrouler un texte autour d'une boîte. `Flexbox` est un peu plus complexe à prendre en main, mais c'est un outil très puissant qui permet de couvrir presque tous les besoins de *layout* d'une page web.

Complément

Documentation MDN sur float[1]

Documentation MDN sur clear[2]

VII. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[3]

Question

[solution n°3 p.38]

Nous allons créer l'en-tête d'un site vitrine. Pour cela, il faudra construire un premier conteneur qui aura :

- Un logo placé à gauche (vous pouvez utiliser l'image de votre choix)
- Le nom de la société placé à gauche après le logo
- Deux éléments de menu "À propos" et "Contact" placés à droite

Puis un second conteneur, placé en dessous de l'en-tête qui portera notre contenu.

Dans cet exercice, nous utiliserons que la propriété **`float`**.

1 https://developer.mozilla.org/fr/docs/Web/CSS/float
2 https://developer.mozilla.org/fr/docs/Web/CSS/clear
3 https://repl.it/


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Document</title>
</head>
<body>
  <div class="header">
    <img
      src="deep-space-inc.svg"
      alt="Deep Space inc. logo"
      class="logo"
    />
    <span class="company-name">Deep Space inc.</span>
    <div class="menu-item">A propos</div>
    <div class="menu-item">Contact</div>
  </div>
  <div class="content">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam vitae vulputate nisl. Cras ornare nunc ac scelerisque
    lacinia. Suspendisse potenti. Donec luctus nulla vel velit finibus, sed cursus arcu cursus. Etiam sed nisi condimentum,
    imperdiet sem eu, lobortis enim. Nulla ut sapien lacus. Ut tincidunt egestas semper. Nulla facilisi.

    Sed laoreet lorem sed lobortis bibendum. Vivamus convallis tortor vestibulum leo condimentum vestibulum. In hac
    habitasse platea dictumst. Maecenas quis pharetra lacus. Aliquam vel tortor nec enim aliquet efficitur vitae vel tortor.
    Nunc non dolor pretium, ultrices nunc at, condimentum diam. Quisque eget enim at erat pharetra malesuada. Quisque
    consequat leo nec justo pretium blandit vitae vitae lacus. Nam faucibus vehicula ex nec vehicula.

    Praesent est enim, fringilla eu molestie eget, feugiat aliquam purus. Nunc sagittis rutrum sapien sit amet scelerisque.
    In hac habitasse platea dictumst. Suspendisse sem orci, accumsan nec elit nec, imperdiet ultricies nulla. Sed porta odio
    eget lacus porttitor, dapibus pellentesque elit congue. Donec euismod malesuada massa nec ultrices. Aliquam ut sem vitae
    tellus fringilla interdum. Vivamus vulputate purus eros, a tempus diam bibendum ut. Suspendisse fringilla leo sem,
```

```
    vehicula consequat odio scelerisque nec. Donec non magna quis augue porttitor elementum sit amet sed arcu. Praesent in
    dictum mauris. Sed ac metus odio. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.
    Sed gravida eleifend sapien non molestie. In nec ante porttitor, consectetur nisi vitae, vehicula ex. Ut bibendum
    sagittis ex, quis malesuada urna vehicula et.
  </div>
</body>
</html>
```

Indice :

Pour que les polices d'écriture soient un peu plus jolies, vous pouvez placer ce code tout en haut de votre CSS.

```
body {
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
}
```

Si vous ne trouvez pas d'image pour le logo, en voici en *open source* que vous pouvez utiliser : source[1]

[cf. deep-space-inc.svg]

VIII. Flexbox

Objectif

- Positionner des éléments grâce à `flexbox`

Mise en situation

Pour faciliter le placement de blocs HTML, de nouvelles possibilités sont apparues au fil des années. Parmi celles-ci, `flexbox` nous permet de très facilement organiser notre contenu.

`Flexbox`, ou bien « modèle de boîte flexible », est un système de positionnement qui vise à faciliter et optimiser la disposition, l'alignement et la répartition entre les éléments.

Lorsqu'on utilise `flexbox`, on ne parlera plus des notions `inline, block` ou `float`.

Propriétés du conteneur flexible

Pour définir un élément conteneur en tant que conteneur flexible, il suffit de définir la propriété `display` à « *flex* ».

```
#container {
    display: flex;
    background-color:blue;
}

```

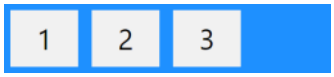
Les éléments enfants d'un conteneur flexible sont appelés « items », et seront alors par défaut alignés horizontalement.

Seuls les enfants directs du conteneur Flex deviennent des éléments Flex.

Les autres descendants, les enfants des enfants, ne seront pas Flex et ne seront pas influencés par le conteneur Flex.

1 logodust

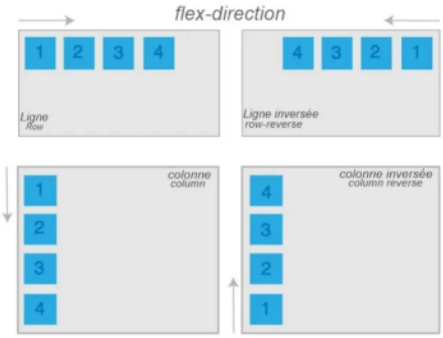
```html
<div id="container">
    <div class="item1"> 1</div>
    <div class="item2"> 2</div>
    <div class="item3"> 3</div>
</div>
```



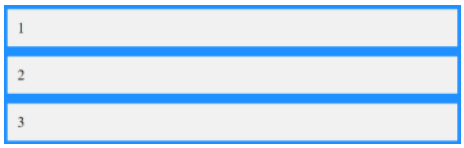
Flex-direction

Selon l'axe principal défini sur le conteneur grâce à la propriété `flex-direction`, les items seront affichés horizontalement (de gauche vers la droite ou vice versa), ou verticalement (de haut vers le bas ou vice versa) en fonction de sa valeur.

- `row` : de gauche vers la droite
- `row-reverse` : droite vers la gauche
- `column` : du haut vers le bas
- `column-reverse` : du bas vers le haut




```css
#container {
  display:flex;
  flex-direction: column;
}

```



La valeur par défaut de `flex-direction` est "*row*" : le comportement d'un conteneur `flex` est d'afficher tous ses enfants côte à côte sur une même ligne.

Justify-content

En suivant l'axe principal défini par flex-direction, la propriété justify-content permet de gérer l'alignement des items sur le conteneur flexible sur cet axe principal.

```
.container {
        justify-content: flex-start | flex-end | center | space-between | space-around |
space-evenly;
    }
```

- `flex-start` : les items seront alignés par rapport au début de l'axe principal (à gauche si l'axe est horizontal, en haut s'il est vertical, à droite si l'axe est horizontal reverse et en bas s'il est vertical *reverse*)
- `flex-end` : les items seront alignés par rapport à la fin de l'axe principal (contraire de `flex-start`)
- center : ils seront alignés au centre sur l'axe principal
- `space-between` : ils sont répartis sur l'axe principal : le premier item sera placé au début de la ligne et le dernier à la fin
- `space-around` : les items sont répartis le long de l'axe principal avec un espace égal autour d'eux (mais, attention, il n'y aura pas le même espace avec les deux bords extrêmes)
- `space-evenly` : les items sont répartis le long de l'axe principal avec un espace égal entre eux (même espace avec les deux bords extrêmes)

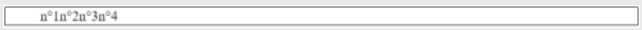
À noter que `flex` utilise les notions de "début" et de "fin" d'un axe : on ne parle pas de "gauche" ou "droite". Certains langages sont lus de droite à gauche comme l'arabe, et de plus en plus de dispositions sont prises par les langages web et les navigateurs pour écrire du code facilement compatible avec les langages LTR (*Left To Right*) et RTL (*Right To Left*).

Exemple

```
<ul class="container">
      <li class="item">n°1</li>
      <li class="item">n°2</li>
      <li class="item">n°3</li>
      <li class="item">n°4</li>
  </ul>
```

```
.container {
      list-style: none;
      display: flex;
      justify-content: flex-start;
  }
```

Résultat : `flex-start`



Résultat : `flex-end`

Résultat : `center`

Résultat : `space-between`

Résultat : `space-around`

Résultat : `space-evenly`

Flex-wrap

Par défaut, les items resteront sur une même ligne quitte à rétrécir s'ils n'ont pas la place de tous s'afficher. La propriété `flex-wrap` permet de modifier ce comportement.

```
#container {
    flex-wrap: nowrap | wrap | wrap-reverse;
}

```

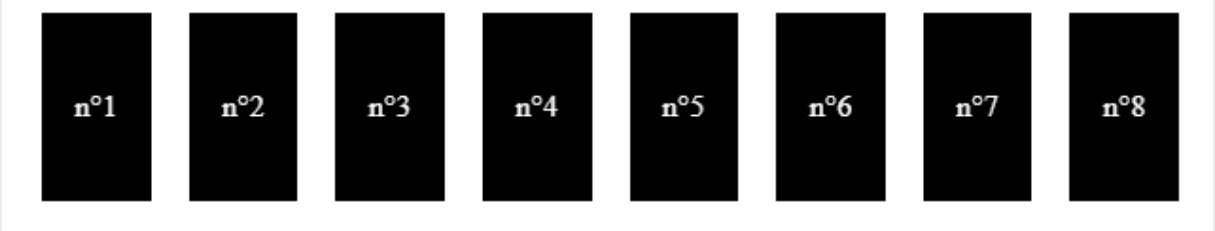
- `nowrap` : valeur par défaut, les items seront positionnés sur une même ligne même si cela implique de les rétrécir.
- `wrap` : si les éléments n'ont pas la place de s'afficher en largeur sans être rétrécis, les items qui débordent s'afficheront sur la ligne du dessous.
- `wrap-reverse` : même comportement qu'avec `wrap`, mais les items s'afficheront cette fois du bas vers le haut.

Exemple

```
<ul class="container">
    <li class="item">n°1</li>
    <li class="item">n°2</li>
    <li class="item">n°3</li>
    <li class="item">n°4</li>
    <li class="item">n°5</li>
    <li class="item">n°6</li>
    <li class="item">n°7</li>
    <li class="item">n°8</li>
</ul>

```

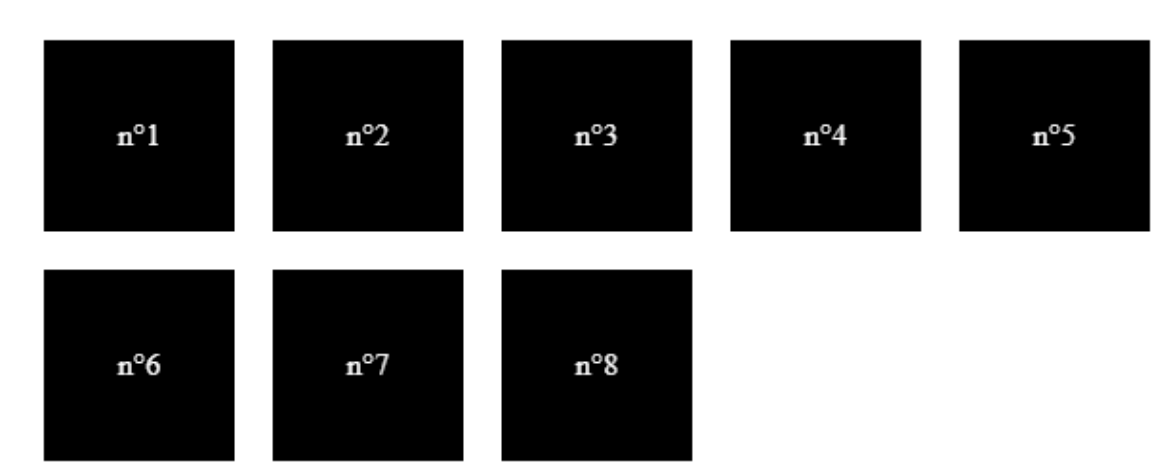
```
.container {
    list-style: none;
    display: flex;
    flex-wrap: nowrap;
}


.item {
    background: black;
    width: 100px;
    height: 100px;
    margin: 10px;
    color: white;
    text-align: center;
    line-height: 100px;

```

Résultat : `nowrap`



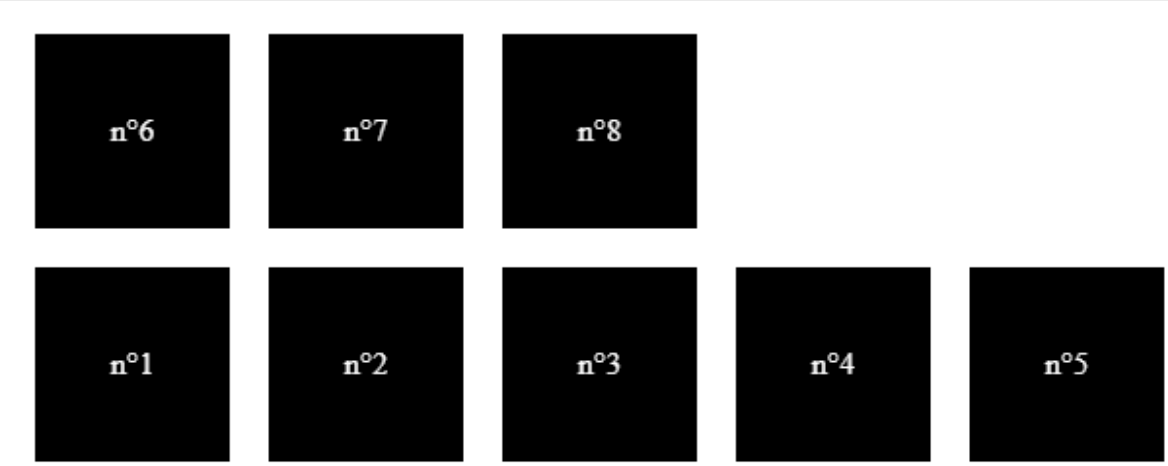
On peut voir que les éléments sont carrés (100 px par 100 px) dans leur définition, mais la fenêtre est trop petite pour permettre d'afficher les 8 enfants à 100 px de large : `flex` va automatiquement les redimensionner pour les afficher sur la même ligne.

Résultat : `wrap`



Avec `wrap`, les éléments retrouvent leur taille comme nous l'avons définie, et les trois éléments qui ne peuvent pas s'afficher sur la première ligne sont repoussés en dessous.

Résultat : `wrap-reverse`



Remarque

Il existe une propriété combinant les deux propriétés vues ci-dessus (`flex-direction` et `flex-wrap`) : `flex-flow`.

```
1 flex-flow: row nowrap
```

Align-items

Nous venons de voir l'alignement des items le long de l'axe principal, il va être intéressant de voir à présent l'alignement le long d'une ligne sur le second axe (axe transversal).

Pour faire cela, nous allons utiliser la propriété `align-items`.

```
.container {
    align-items: stretch | flex-start | flex-end | center | baseline
}

```

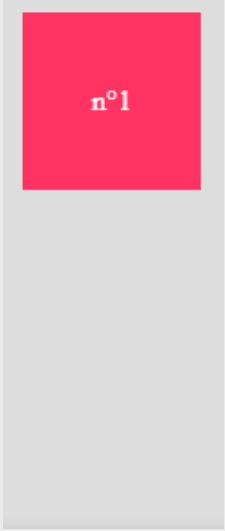
- `stretch` : les items seront étirés pour prendre toute la place du conteneur flexible.
- `flex-start` : les items seront placés au début de l'axe transversal.
- `flex-end` : les items seront placés à la fin de l'axe transversal.
- `center` : les items seront alignés au centre sur l'axe transversal.
- `baseline` : les items seront alignés en fonction de leur ligne de base.

Exemple

```
<ul class="container">
    <li class="item">n°1</li>
</ul>

```

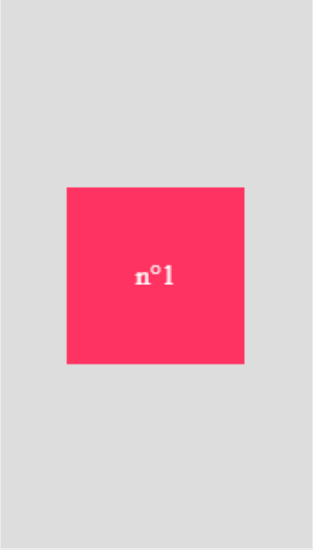
```
body{
    background: #ddd;
    height:200px;
}
.container {
    list-style: none;
    display: flex;
    justify-content: center;
    margin:0;
    padding:0;
    height:300px;
    align-items: baseline;
    flex-wrap: wrap;
}

.item {
    background: #FF3361;
    width: 100px;
    height: 100px;
    color: white;
    text-align: center;
    line-height: 100px;

}
.item2 {
    font-size:2em;
    background: #FF3361;
}

```

Résultat : `flex-start`



Résultat : `flex-end`

Résultat : `center`



Résultat : `baseline`

```html
<ul class="container">
    <li class="item">n°1</li>
    <li class="item item2">n°2</li>
</ul>

```

Résultat : `stretch`

```html
<ul class="container">
    <li class="item">1<br>2</li>
    <li class="item">3</li>
    <li class="item">4<br>5</li>
    <li class="item">6</li>
</ul>

```

```css
.container {
    padding: 0;
    margin: 0;
    list-style: none;
    display: flex;
}

li {
    background: #FF3361;
}

.item {
    padding: 5px;
    width: 100px;
    margin: 5px;
    line-height: 50px;
    color: white;
    font-weight: bold;
    font-size: 2em;
    text-align: center;
}

```

Align-content

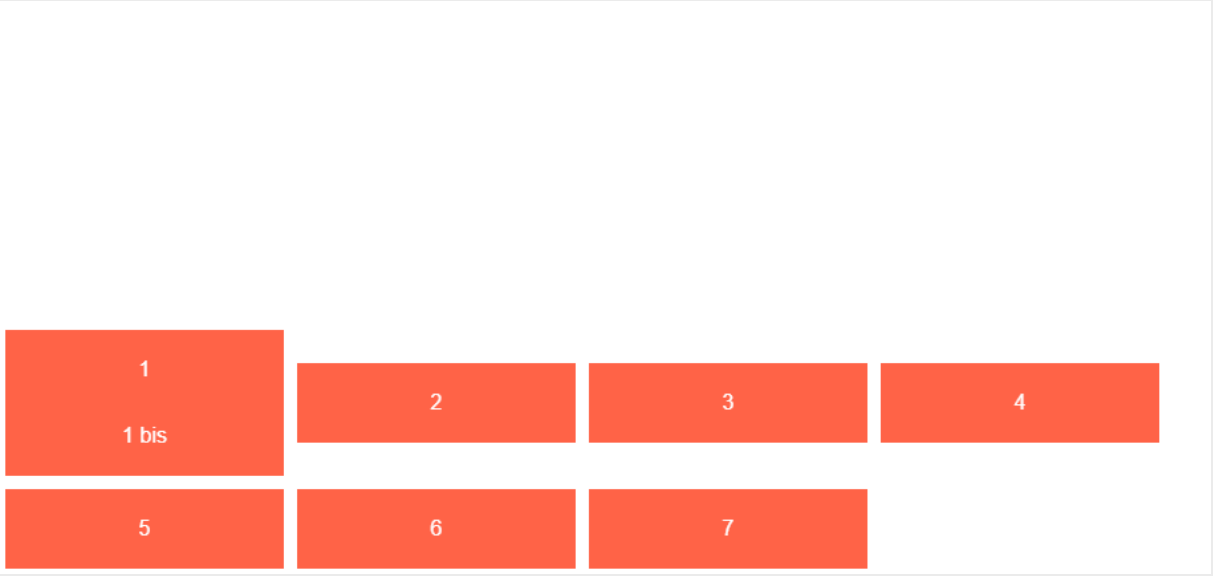
Comme nous venons de le voir, la propriété `align-items` permet d'aligner les items d'une ligne sur l'axe transversal. Il existe également une propriété permettant de gérer l'alignement des lignes le long de l'axe transversal. Les valeurs possibles sont les mêmes que celles de la propriété `justify-content`.

Exemple

```html
<ul class="container">
    <li class="item">1<br>1 bis</li>
    <li class="item">2</li>
    <li class="item">3</li>
    <li class="item">4</li>
    <li class="item">5</li>
    <li class="item">6</li>
    <li class="item">7</li>
</ul>
```

```css
.container {
    padding: 0;
    margin: 0;
    list-style: none;
    display: flex;
    flex-wrap:wrap;
    align-items: center;
    align-content: flex-end;
    height:100vh;
}

.item {
    background: orange;
    padding: 5px;
    width: 200px;
    margin: 5px;
    line-height: 50px;
    color: white;
    text-align: center;
}
```

Résultat :



Propriétés des items

Nous avons vu comment agir sur le contenant, mais nous pouvons aussi agir sur le contenu grâce à une autre propriété.

Définition **Order**

Au sein d'un conteneur flexible, les items sont affichés par défaut dans l'ordre d'écriture dans le fichier HTML.

Il existe un moyen de modifier ce comportement en CSS grâce à la propriété **`order`**.

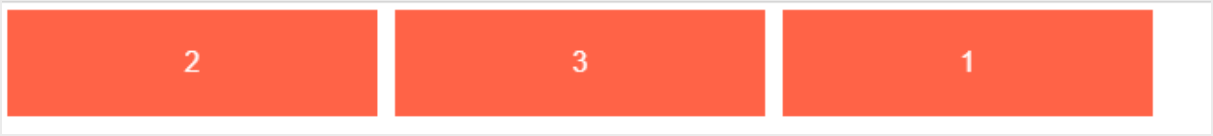
Exemple

```html
<ul class="container">
    <li class="item one">1</li>
    <li class="item two">2</li>
    <li class="item three">3</li>
</ul>
```

```css
.container {
    padding: 0;
    margin: 0;
    list-style: none;
    display: flex;
    flex-wrap:wrap;
    align-items: center;
    align-content: flex-start;
    height:100vh;
}


.item {
    background: orange;
    padding: 5px;
    width: 200px;
```

```
    margin: 5px;
    line-height: 50px;
    color: white;
    text-align: center;
}

.one {
    order: 2;
}

.two {
    order: 0;
}

.three {
    order: 1;
}

```

Résultat :



Taille d'un enfant

La propriété **`flex`** sur un enfant d'un conteneur flexible permet de régler la taille de l'enfant de manière dynamique, un peu à la manière de l'unité **%**.

Cette propriété est le raccourci de trois propriétés qu'il est possible de définir unitairement, dans l'ordre :

- `flex-grow` : permet de définir la capacité de l'élément à s'élargir au besoin pour remplir le conteneur flexible. Accepte une valeur numérique positive.
- `flex-shrink` : permet de définir la capacité de l'élément à rétrécir au besoin pour laisser la place à d'autres éléments d'être affichés dans le conteneur flexible. Accepte une valeur numérique positive.
- `flex-basis` : permet d'indiquer une taille par défaut pour l'élément avant que l'espace restant dans le conteneur flexible ne soit redistribué, la valeur auto indiquera au navigateur de prendre la valeur de `width` par défaut. Accepte une taille exprimée dans n'importe quelle unité supportée par `width` ou `height` (px, pt, %, em, ...).

Soit un conteneur flexible de 1000 px de largeur : imaginons deux enfants, dont la taille souhaitée est de respectivement 1/3 et 2/3 du conteneur flexible.

```
.parent {
  display: flex;
  width: 1000px;
}

.enfant1 {
  flex: 1;
}

.enfant2 {
  flex: 2;
}
```

```
13
14 // enfant1 et enfant2 vont automatiquement s'élargir pour couvrir l'espace disponible dans le conteneur flexible ".parent"
15 // enfant2 prendra deux fois plus de place que enfant1, ce qui revient à leur donner 1/3 et 2/3 de l'espace disponible.
```

Pour répartir équitablement la place disponible, il suffit de mettre la même valeur pour les deux enfants.

```
1 .parent {
2   display: flex;
3   width: 1000px;
4 }
5
6 .enfant1 {
7   flex: 1;
8 }
9
10 .enfant2 {
11   flex: 1;
12 }
13
14 // Les deux enfants se partageront l'espace disponible
```

`Flex-shrink` permet de gérer la distribution de l'espace dans un conteneur flexible s'il n'y a pas assez de place pour afficher tous les éléments sans les déformer.

10 enfants de 200 px de large sont définis pour un conteneur de 1000 px de large, `flex-wrap` n'est pas activé, donc `flexbox` va conserver tous les éléments sur la même ligne. Les 10 enfants ont alternativement une valeur de `flex-shrink` de 1 et 2.

```
1 .enfant1 {
2   flex-shrink: 1;
3   background: #0984e3;
4 }
5
6 .enfant2 {
7   flex-shrink: 2;
8   background: #00b894;
9 }
10
11 .enfant3 {
12   flex-shrink: 1;
13   background: #a29bfe;
14 }
15
16 .enfant4 {
17   flex-shrink: 2;
18   background: #fdcb6e;
```

```
}

.enfant5 {
  flex-shrink: 1;
  background: #00cec9;
}

.enfant6 {
  flex-shrink: 2;
  background: #e17055;
}

.enfant7 {
  flex-shrink: 1;
  background: #e84393;
}

.enfant8 {
  flex-shrink: 2;
  background: #2d3436;
}

.enfant9 {
  flex-shrink: 1;
  background: #1e3799;
}

.enfant10 {
  flex-shrink: 2;
  background: #f6b93b;
}
```

Le résultat montre qu'un élément sur deux s'est plus déformé pour laisser la place aux éléments pour lesquels a été indiqué le `flex-shrink` le plus faible.

`flex-basis` spécifie la longueur initiale d'un élément flexible. Dans notre cas l'enfant 1 aura une taille initiale de 200px;

```
.parent {
    display: flex;
    width: 1000px;
  }

  .enfant1 {
    flex-basis: 200px;
  }

  .enfant2 {
    flex: 2;
  }

```

Syntaxe **À retenir**

Pour déclarer un contenu flexible, il faut ajouter la règle CSS `display: flex`.

Plusieurs options pour manipuler l'affichage du contenu sont disponibles :

- `flex-direction` : pour définir la façon dont les éléments sont placés
- `flex-wrap` : pour définir si les éléments peuvent être placés à la ligne ou non
- `justify-content` : pour définir l'espace entre les éléments
- `align-items` : pour définir l'alignement des éléments
- `align-content` : pour définir l'espace entre et autour des éléments
- `order` : pour définir l'ordre d'affichage des éléments

Complément

Documentation MDN sur les Flexbox[1]

IX. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[2]

Question

[solution n°4 p.39]

Reprenant notre exemple précédent, nous allons recréer le site vitrine en utilisant cette fois-ci `flexbox`. Comme précédemment, nous allons avoir besoin d'un conteneur d'en-tête qui contiendra les éléments suivants :

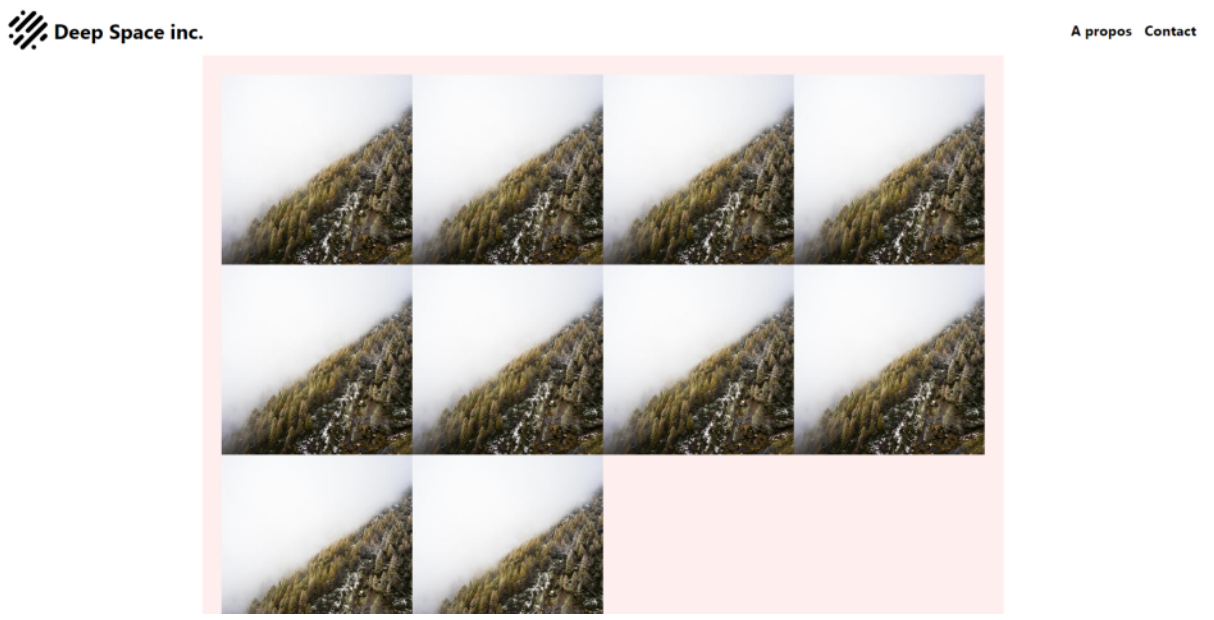
- Un logo placé à gauche
- Le nom de la société placé à la droite du logo
- Deux éléments de menu placés à droite

Tous les éléments dans le conteneur d'en-tête devront être centrés verticalement.

Un autre conteneur, affiché en dessous, permettra d'afficher une galerie d'images côte à côte. Ce conteneur devra être centré sur la page, et mesurer **1200 px** de largeur.

Dans ce conteneur, nous afficherons 10 images d'une dimension de **300 px** de haut par **300 px** de large. Il ne faut pas que ces images soient compressées.

1 https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Box_Alignment/Alignement_bo%C3%AEtes_disposition_Flexbox
2 https://repl.it/




```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Document</title>
</head>
<body>
  <div class="header">
    <div class="logo-wrapper">
      <img
        src="deep-space-inc.svg"
        alt="Deep Space inc. logo"
        class="logo"
      />
      <span class="company-name">Deep Space inc.</span>
    </div>
    <div class="menu">
      <div class="menu-item">A propos</div>
      <div class="menu-item">Contact</div>
    </div>
  </div>
  <div class="content">
    <img src="<!-- insérez le chemin de votre image 300 x 300 pixels ici -->" />
    <img src="<!-- insérez le chemin de votre image 300 x 300 pixels ici -->" />
    <img src="<!-- insérez le chemin de votre image 300 x 300 pixels ici -->" />
    <img src="<!-- insérez le chemin de votre image 300 x 300 pixels ici -->" />
    <img src="<!-- insérez le chemin de votre image 300 x 300 pixels ici -->" />
    <img src="<!-- insérez le chemin de votre image 300 x 300 pixels ici -->" />
    <img src="<!-- insérez le chemin de votre image 300 x 300 pixels ici -->" />
    <img src="<!-- insérez le chemin de votre image 300 x 300 pixels ici -->" />
    <img src="<!-- insérez le chemin de votre image 300 x 300 pixels ici -->" />
    <img src="<!-- insérez le chemin de votre image 300 x 300 pixels ici -->" />
  </div>
</body>
</html>
```

Indice :

- Vous pouvez utiliser vos propres images ou, pour plus de simplicité, utiliser le lien suivant https://picsum.photos/300/300 (les deux valeurs 300 dans l'URL correspondent à la hauteur et largeur souhaitées pour l'image).
- Si vous utilisez la propriété `flexbox` pour distribuer les éléments de l'en-tête sur la largeur, vous allez voir que le résultat n'est pas celui attendu. Gardez à l'esprit que cette méthode va distribuer sur la largeur seulement ses enfants directs.
- Pour le logo, vous pouvez utiliser votre propre image, ou le logo suivant :

[cf. deep-space-inc.svg]

X. Auto-évaluation

A. Exercice final

Exercice 1

[solution n°5 p.39]

Exercice

Quelles sont les valeurs autorisées de la propriété `position` ?

- ☐ justify
- ☐ relative
- ☐ fixed
- ☐ absolute
- ☐ static
- ☐ block

Exercice

Quel est le comportement de deux blocs A et B avec une position `relative` ?

- ☐ Ils sont affichés dans l'ordre dans lequel ils ont été écrits dans le HTML
- ☐ Ils sont superposés par défaut
- ☐ Il s'affichent l'un en dessous de l'autre
- ☐ Ils s'affichent l'un à côté de l'autre

Exercice

Un bloc positionné en absolu a pour référentiel...

- ○ Il n'a aucun référentiel.
- ○ La fenêtre du navigateur
- ○ La balise <body>.
- ○ Son parent le plus proche.

Exercice

La différence entre un bloc en `fixed` et un bloc en `absolute` est :

- Le bloc absolute suit le défilement de la page, ce n'est pas vrai pour le bloc fixed.
- Le bloc fixed suit le défilement de la page, ce n'est pas vrai pour le bloc absolute.
- Le bloc fixed s'affiche toujours au-dessus du bloc absolute s'ils sont à la même position.
- Il n'y a pas de différence, les valeurs sont interchangeables.

Exercice

Que permet de faire la propriété `float` ?

- Reproduire le comportement d'un bloc en position absolute.
- Afficher deux éléments sur la même ligne.
- Centrer verticalement deux éléments.

Exercice

Que faut-il écrire pour ne pas qu'un élément soit sujet au contexte `float` de l'élément qui le précède ?

- flow : normal ;
- float : none ;
- clear : both ;

Exercice

Quel élément aura la priorité d'affichage (affiché au-dessus des autres) ?

- Un élément en static ou relative.
- Un élément en absolute ou fixed.

Exercice

`Flexbox` met à disposition la propriété `justify-content` qui aligne les éléments...

- De haut en bas.
- De gauche à droite.
- Sur l'axe principal.
- Sur l'axe secondaire.

Exercice

Un conteneur `flex` va par défaut...

- [ ] Aligner les enfants de premier niveau sur une même ligne.
- [ ] Aligner les enfants de premier niveau sur une même colonne.
- [ ] Centrer les éléments sur la page.
- [ ] Compresser les enfants de premier niveau si leur largeur cumulée est supérieure à la capacité en largeur.
- [ ] Faire passer les enfants qui dépassent de la capacité en largeur sur une nouvelle ligne.

Exercice

Quelle est la propriété `flexbox` qui sert à modifier l'orientation des axes principaux et secondaires ?

- transform : rotate(90deg) ;
- flex-direction : column ;
- rotate : clockwise ;
- langage-mode : ttb ;

B. Exercice : Défi

Nous allons maintenant, avec l'aide de tous les éléments que nous avons vus, construire la structure d'une page de blog.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[1]

1 https://repl.it/

Question

[solution n°6 p.41]

En vous appuyant sur la propriété `position`, sur `float`, le contexte d'empilement sur l'axe Z et `flexbox`, reproduisez l'intégration suivante :

Vous pouvez vous baser sur le code CSS suivant pour le rendu visuel.

```css
html, body {
  margin: 0;
}

* {
  box-sizing: border-box;
}

body {
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
```

Indice :

- Pour les images, vous pouvez utiliser le service https://picsum.photos/300/300 comme dans les exercices précédents.
- Pour récupérer les icônes, vous pouvez utiliser Font Awesome[1] en suivant la documentation, ou bien mettre des images à la place.
- Les couleurs utilisées sont #000000 pour le noir, #ffffc00 pour le jaune et #ff4757 pour le rose, vous n'êtes bien sûr pas contraint d'utiliser les mêmes couleurs, l'important ici étant le positionnement des éléments.

Solutions des exercices

1 https://fontawesome.com/start

p. 8 **Solution n°1**

```
html, body {
width: 100%;
height: 3000px;
margin: 0;
}

.a,
.b,
.c,
.d,
.e {
height: 150px;
width: 150px;
}

.a {
background: paleturquoise;
}

.b {
background: lightgreen;
}

.c {
position: relative;
top: -40px;
background: lightpink;
}

.d {
position: fixed;
bottom: 0;
right: 0;
background: palegoldenrod;
}

.e {
position: absolute;
bottom: 0;
left: 0;
background: lightseagreen;
}

```

p. 11 **Solution n°2**

Première solution : z-index

Cette solution consiste à ajouter une propriété `z-index` à notre élément `.back-to-top` d'une valeur numérique plus importante que celle du pied de page.

```
.back-to-top {
  position: absolute;
  bottom: 10px;
  right: 20px;
  background: white;
  font-weight: 600;
  padding: 10px 20px;
  border-radius: 6px;
  z-index: 2;
}
```

Seconde solution : modifier l'ordre des éléments dans le HTML

Si on intervertit les éléments `footer` et `back-to-top`, ce dernier sera rendu au-dessus des autres éléments positionnés en `absolute` ou en `flex` de la page. Un élément `absolute` ou `fixed` est quant à lui toujours rendu au-dessus d'un élément en `static` ou `relative`.

```
<div class="footer"></div>
<div class="back-to-top">remonter</div>
```

p. 15 **Solution n°3**

```
body {
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
}

.logo {
  height: 5rem;
  width: auto;
  float: left;
}

.company-name {
  float: left;
  font-size: 2rem;
  font-weight: 700;
}

.menu-item {
  font-size: 1.4rem;
  font-weight: 700;
  margin-left: 10px;
  margin-right: 10px;
  float: right;
}

.content {
  clear: both;
  width: 1200px;
  min-height: 600px;
  padding: 30px;
  margin: auto;
```

```
  background:#ffeeee;
}
```

p. 30 **Solution n°4**

```
body {
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
}

.header {
  display: flex;
  justify-content: space-between;
}

.logo-wrapper,
.menu {
  display: flex;
  align-items: center;
}

.logo {
  height: 5rem;
  width: auto;
  float: left;
}

.company-name {
  font-size: 2rem;
  font-weight: 700;
}

.menu-item {
  font-size: 1.4rem;
  font-weight: 700;
  margin-left: 10px;
  margin-right: 10px;
}

.content {
  display: flex;
  flex-wrap: wrap;
  width: 1200px;
  min-height: 600px;
  padding: 30px;
  margin: auto;
  background:#ffeeee;
}
```

Exercice p. 32 **Solution n°5**

Exercice

Quelles sont les valeurs autorisées de la propriété `position` ?

- ☐ justify
- ☑ relative
- ☑ fixed
- ☑ absolute
- ☑ static
- ☐ block
  `block` est une valeur acceptée pour la propriété `display`.

Exercice

Quel est le comportement de deux blocs A et B avec une position `relative` ?

- ☑ Ils sont affichés dans l'ordre dans lequel ils ont été écrits dans le HTML
- ☐ Ils sont superposés par défaut
- ☑ Il s'affichent l'un en dessous de l'autre
- ☐ Ils s'affichent l'un à côté de l'autre

Exercice

Un bloc positionné en absolu a pour référentiel...

- ○ Il n'a aucun référentiel.
- ○ La fenêtre du navigateur
- ◉ La balise <body>.
- ○ Son parent le plus proche.

Exercice

La différence entre un bloc en `fixed` et un bloc en `absolute` est :

- ○ Le bloc absolute suit le défilement de la page, ce n'est pas vrai pour le bloc fixed.
- ◉ Le bloc fixed suit le défilement de la page, ce n'est pas vrai pour le bloc absolute.
- ○ Le bloc fixed s'affiche toujours au-dessus du bloc absolute s'ils sont à la même position.
- ○ Il n'y a pas de différence, les valeurs sont interchangeables.

Exercice

Que permet de faire la propriété `float` ?

- ○ Reproduire le comportement d'un bloc en position absolute.
- ◉ Afficher deux éléments sur la même ligne.
- ○ Centrer verticalement deux éléments.

Exercice

Que faut-il écrire pour ne pas qu'un élément soit sujet au contexte `float` de l'élément qui le précède ?

- ○ flow : normal ;
- ○ float : none ;
- ◉ clear : both ;

Exercice

Quel élément aura la priorité d'affichage (affiché au-dessus des autres) ?

- ○ Un élément en static ou relative.
- ◉ Un élément en absolute ou fixed.

Exercice

`Flexbox` met à disposition la propriété `justify-content` qui aligne les éléments...

- ○ De haut en bas.
- ○ De gauche à droite.
- ◉ Sur l'axe principal.
  On parle d'axe principal plutôt que de gauche/droite pour prendre en compte les langages écrits de droite à gauche.
- ○ Sur l'axe secondaire.

Exercice

Un conteneur `flex` va par défaut...

- ☑ Aligner les enfants de premier niveau sur une même ligne.
- ☐ Aligner les enfants de premier niveau sur une même colonne.
- ☐ Centrer les éléments sur la page.
- ☑ Compresser les enfants de premier niveau si leur largeur cumulée est supérieure à la capacité en largeur.
- ☐ Faire passer les enfants qui dépassent de la capacité en largeur sur une nouvelle ligne.

Exercice

Quelle est la propriété `flexbox` qui sert à modifier l'orientation des axes principaux et secondaires ?

- ○ transform : rotate(90deg) ;
- ◉ flex-direction : column ;
- ○ rotate : clockwise ;
- ○ langage-mode : ttb ;

p. 35 **Solution n°6**

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="VOTRE URL DE KIT FONT AWESOME.JS" crossorigin="anonymous"></script>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="VOTRE URL DE KIT FONT AWESOME.CSS">
    <title>Blog</title>
</head>
<body>
  <header>
    <div class="logo-wrapper">
      <i class="fas fa-code"></i>
      <h1>Mon blog!</h1>
    </div>
    <ul>
      <li>Blog</li>
      <li>CV</li>
      <li>Contact</li>
      <li>Social</li>
    </ul>
  </header>
  <div class="content">
    <section>
      <img class="illustration floated-left" src="https://picsum.photos/400/300">
      <p>
        Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam mattis, nunc sed gravida varius, massa nunc pharetra nulla,
        ut finibus nibh mauris ac lorem. Fusce hendrerit convallis quam, ut consectetur felis placerat in.
      </p>
      <p>
        Vestibulum sed nunc porttitor, varius libero in, consequat enim. Maecenas aliquet, metus vehicula hendrerit pharetra,
        tellus orci
        ullamcorper lectus, vel ornare purus ipsum et eros. Maecenas blandit nec elit vitae malesuada. Nulla dui urna, venenatis
        id pellentesque vel, vehicula vitae neque.
      </p>
      <p>
        Sed iaculis, mi vitae ornare luctus, nibh justo ultricies ante, in accumsan
        libero mauris ac est. Vestibulum ullamcorper faucibus mauris at condimentum. Fusce dictum sapien sit amet lorem
        pellentesque laoreet. Suspendisse eros neque, condimentum eget mauris in, faucibus vestibulum quam. Integer porttitor
        interdum ligula, dictum porta eros iaculis posuere.
      </p>
      <p>
        Maecenas odio neque, ultrices sed cursus et, porta id dolor. Sed
        scelerisque, turpis et congue condimentum, justo urna semper enim, ac cursus nisi quam nec diam.
      </p>
      <img class="illustration floated-right" src="https://picsum.photos/400/300">
      <p>
        Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam mattis, nunc sed gravida varius, massa nunc pharetra
        nulla,
        ut finibus nibh mauris ac lorem. Fusce hendrerit convallis quam, ut consectetur felis placerat in.
      </p>
```

```
      <p>
        Vestibulum sed nunc porttitor, varius libero in, consequat enim. Maecenas aliquet, metus vehicula hendrerit pharetra,
        tellus orci
        ullamcorper lectus, vel ornare purus ipsum et eros. Maecenas blandit nec elit vitae malesuada. Nulla dui urna,
        venenatis
        id pellentesque vel, vehicula vitae neque.
      </p>
      <p>
        Sed iaculis, mi vitae ornare luctus, nibh justo ultricies ante, in accumsan
        libero mauris ac est. Vestibulum ullamcorper faucibus mauris at condimentum. Fusce dictum sapien sit amet lorem
        pellentesque laoreet. Suspendisse eros neque, condimentum eget mauris in, faucibus vestibulum quam. Integer porttitor
        interdum ligula, dictum porta eros iaculis posuere.
      </p>
      <p>
        Maecenas odio neque, ultrices sed cursus et, porta id dolor. Sed
        scelerisque, turpis et congue condimentum, justo urna semper enim, ac cursus nisi quam nec diam.
      </p>
    </section>
    <aside>
      <ul>
        <li>Articles de Mars 2020</li>
        <li>Articles de Février 2020</li>
        <li>Articles de Janvier 2020</li>
        <li>Articles de Décembre 2019</li>
        <li>Articles de Novembre 2019</li>
        <li>Articles de Octobre 2019</li>
        <li>Archives</li>
      </ul>
    </aside>
  </div>
  <footer>
    <div class="upper-footer">
      <div>
        <ul>
          <li><a href="Javascript:void(0);">Home</a></li>
          <li><a href="Javascript:void(0);">A propos</a></li>
          <li><a href="Javascript:void(0);">Sponsors</a></li>
          <li><a href="Javascript:void(0);">Mes autres projets</a></li>
        </ul>
      </div>
      <div class="socials">
        <i class="fab fa-facebook"></i>
        <i class="fab fa-twitter"></i>
        <i class="fab fa-instagram"></i>
      </div>
    </div>
    <div class="watermark">@2020</div>
  </footer>

  <div class="back-to-top">
    <i class="fas fa-rocket"></i>
  </div>
</body>
</html>
```

```css
html, body {
  margin: 0;
}

* {
  box-sizing: border-box;
}

body {
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

header {
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 20px;
  background: #fffc00;
}

header i {
  color: #ff4757;
}

header h1,
header i {
  margin: 0 5px;
  font-size: 6rem;
}

header ul {
  display: flex;
  font-weight: 700;
}

header li {
 margin: 0 10px;
 text-decoration: underline;
}

.logo-wrapper {
  display: flex;
  align-items: center;
}

.content {
  display: flex;
  background: #fffc00;
  padding: 5rem;
}

```

```
section,
aside {
  background: white;
  margin: 15px;
  padding: 20px;
}

section {
  flex: 2;
}

.illustration {
  margin: 20px;
}

.floated-left {
  float: left;
}

.floated-right {
  float: right;
}

aside {
  flex: 1;
}

aside ul {
  display: flex;
  flex-direction: column;
}

aside li {
  flex: 1;
  align-self: stretch;
  background: #ececec;
  margin: 5px;
  padding: 10px;
}

footer {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
  width: 100%;
  min-height: 500px;
  background: black;
  color: #fffc00;
  padding: 2rem 0;
}

footer a { color: #fffc00; font-weight: 700; }
footer a:active { color: #fffc00; font-weight: 700; }
footer a:hover { color: #fffc00; font-weight: 700; }
footer a:enabled { color: #fffc00; font-weight: 700; }

.upper-footer {
```

```
  display: flex;
  align-items: center;
  justify-content: space-evenly;
  flex: 1;
  align-self: stretch;
}

.socials {
  display: flex;
  flex-direction: column;
  font-size: 1.6rem;
}

.socials i {
  margin: 10px;
}

.back-to-top {
  position: fixed;
  bottom: 40px;
  right: 40px;
  width: 100px;
  height: 100px;
  border-radius: 50%;
  background: #ff4757;
  color: #fff;
  font-size: 2rem;
  display: flex;
  align-items: center;
  justify-content: center;
}
```