Programmation Web

Le CSS - Mise en page

Pierre Blarre - Florian Rodriguez

Flux

- Le cours normal est la manière dont le navigateur présente les pages HTML par défaut quand vous ne faites rien pour contrôler la mise en page.
- Notez que le HTML est affiché dans l'ordre exact où il est dans le code source : les éléments s'empilent les uns sur les autres — le premier paragraphe, puis la liste non ordonnée suivie par le second paragraphe.
- Les éléments disposés en empilement sont désignés comme étant des éléments *block*, par opposition aux éléments *inline* qui apparaissent l'un après l'autre telle la succession de mots distincts d'un paragraphe.
- Lorsque vous utilisez les CSS pour faire une mise en page, vous déplacez les éléments de leur cours normal ; toutefois, pour la plupart des éléments de la page, ce cours normal crée exactement la mise en page dont vous avez besoin. C'est pourquoi il est si important de commencer avec **un document HTML bien structuré**, car vous pouvez alors travailler la façon dont les choses seront disposées par défaut au lieu de lutter contre cette disposition.

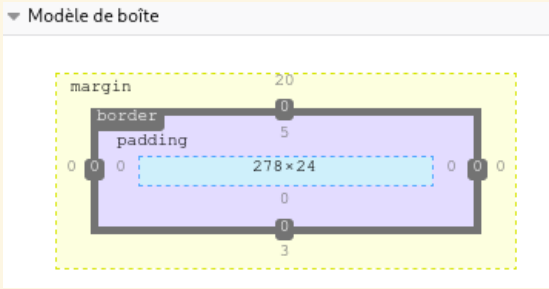
Placement des éléments

Les méthodes des CSS pouvant changer le placement des éléments sont les suivantes :

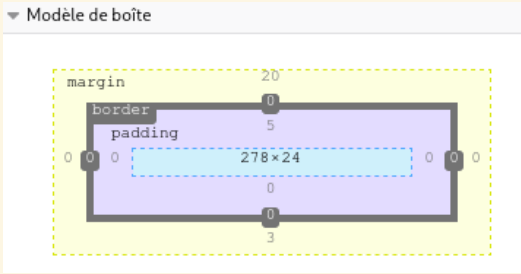
- **La propriété display**: les valeurs standards comme `block`, `inline` ou `inline-block` peuvent changer la manière dont l'élément se comporte dans le cours normal.
- **La propriété position**: vous permet de contrôler avec précision le placement de boîtes dans d'autres boîtes. `static` est le placement par défaut dans le cours, mais vous pouvez manipuler les éléments pour qu'ils se comportent différemment à l'aide d'autres valeurs, par exemple en les fixant en haut à gauche de la fenêtre d'affichage du navigateur.
- **La propriété float**: une valeur comme left peut créer une juxtaposition d'un élément bloc à côté d'un autre, tout comme les images « baignent » dans le texte dans les mises en page de magazines.
- **Mise en page sur plusieurs colonnes** Les propriétés Multi-column peuvent faire qu'un contenu bloc soit disposé en colonnes, comme dans un journal. (`column-count`, `column-width`)

Boîtes

- En CSS, tout élément est inclus dans une boîte ("box" en anglais)
- Comprendre le fonctionnement de ces boîtes est essentiel pour maîtriser la mise en page CSS ainsi que le positionement des éléments d'une page HTML.



Composant du modèle de boîte



- **Le contenu** : Il s'agit de la zone où sont affichés les éléments contenus par notre boîte, qui peut être dimensionnée en utilisant les propriétés CSS width et height.
- **Le padding** (marge intérieure) : Le padding (ou remplissage en français) est une zone vierge qui se présente comme un espacement encadrant le contenu; sa taille peut être contrôlée sur chaque côté en utilisant la propriété padding et ses autres propriétés connexes.
- **La bordure** : La bordure englobe le contenu et le padding pour former une bordure. Sa taille et son style sont paramétrés par la propriété border et ses propriétés sous-jacentes.
- **La marge** : La marge est la couche la plus à l'extérieur, englobant le contenu, le padding et la bordure. Comme le padding , il s'agit d'une zone vierge d'espacement mais qui est cette fois située à l'extérieur de l'élément, séparant l'élément des autres éléments de la page. sa taille peut être contrôlée sur chaque côté en utilisant la propriété margin et ses autres propriétés connexes.

Boîtes

- En CSS, il existe deux type de boîtes *Block* et *Inline*
- Le type de boîte appliqué à un élément est défini par la valeur de la propriété display tel que block ou inline, et se réfère à la valeur extérieure de positionnement

Block

- La boîte **s'étend en largeur** pour remplir totalement l'espace offert par son conteneur. Dans la plupart des cas, la boîte devient alors aussi large que son conteneur, occupant 100% de l'espace disponible.
- La boîte **occupe sa propre nouvelle ligne** et crée un retour à la ligne, faisant ainsi passer les éléments suivants à la ligne d'après.
- Les propriétés de largeur (width) et de hauteur (height) sont toujours respectées.
- Les propriétés padding, margin et border auront pour effet de repousser les autres éléments.

À moins que l'on ne décide de changer le type de positionnement de la boîte en "en ligne", certains éléments tels que les titres (<h1>, <h2>, etc.) et les paragraphes (<p>) utilisent le mode "bloc" comme propriété de positionnement extérieur par défaut.

Inline

- La boîte ne crée **pas de retour à la ligne**, les autres éléments se suivent donc en ligne.
- Les propriétés de largeur (width) et de hauteur (height) ne s'appliquent pas.
- Le padding , les marges et les bordures verticales (en haut et en bas) seront appliquées mais ne provoqueront pas de déplacement des éléments alentours.

- Le padding , les marges et les bordures horizontales (à gauche et à droite) seront appliquées et provoqueront le déplacement des éléments alentours.

  Les éléments `<a>`, utilisés pour les liens, ou encore `<span>`, `<em>` et `<strong>` sont tous des éléments qui s'affichent "en ligne" par défaut.

Inline-block

- `inline-block` une combinaison d'inline et de block
- Utile dans les situations où l'on désire utiliser les propriétés width et height, et éviter les superpositions, tout en conservant la disposition dans une même ligne (i.e. sans créer de nouvelle ligne, comme le ferait une disposition en bloc).
- La hauteur (height) et la largeur (width) seront appliqués sur l'élément (et non ignorés).
- Les propriétés padding, margin et border repousseront bien les éléments alentours.
- Conserve un positionnement sur la même ligne, sans retour à la ligne, ni affichage sur sa propre nouvelle ligne.

Positionnements intérieurs et extérieurs

- Les boîtes en CSS possèdent un type de positionnement extérieur qui détermine si la boîte est "en ligne" ou bien "en bloc".
- Les boîtes ont aussi un type de positionnement intérieur, qui décrit le comportement de mise en page des éléments contenus dans la boîte. Par défaut, les éléments contenus dans la boîte sont affichés dans la disposition normale, ce qui signifie qu'ils se comportent exactement comme n'importe quel autre élément "en bloc" ou "en ligne" (comme décrit auparavant).
- Ce type de positionnement intérieur peut naturellement être modifié en utilisant la propriété display et des valeurs comme *flex* ou *grid* (voir plus loin. D'autres existent mais ce sont les 2 principales)

Exemple

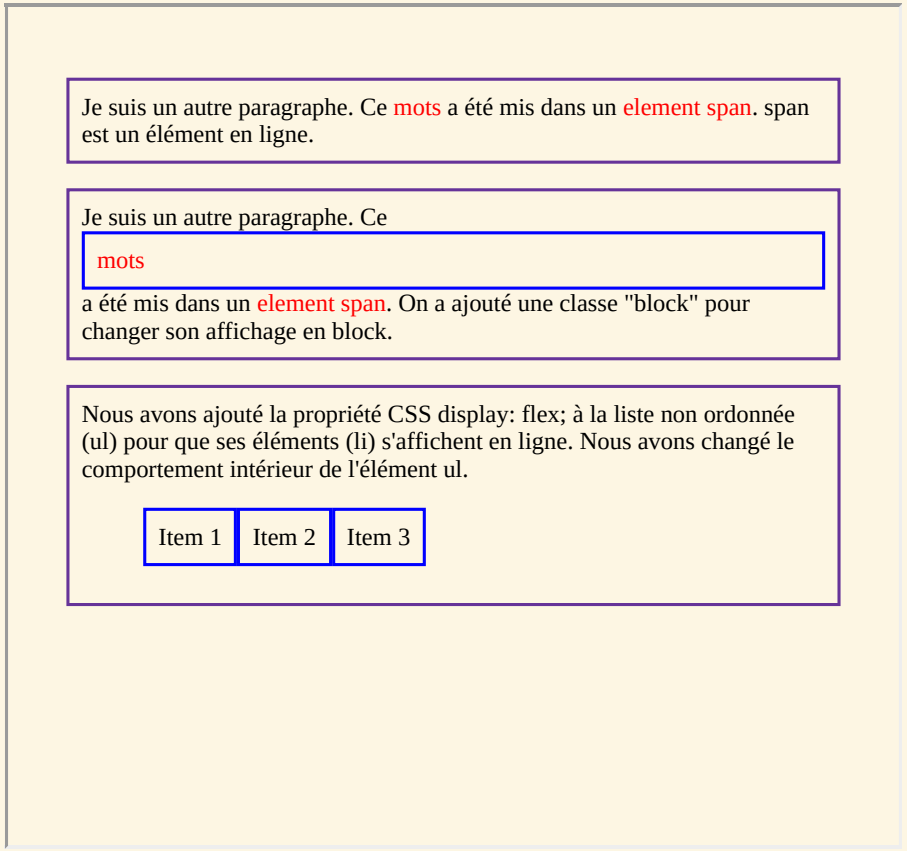
Source

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, i
    <title>Exemple box</title>
    <style>
     p, div{
       border: 2px solid rebeccapurple;
       padding: .5em;
     }
     span {
       color:red;
     }
     .block,
     li {
       border: 2px solid blue;
       padding: .5em;
     }

     ul {
       display: flex;
       list-style: none;
     }

      block {
```

Résultat



Ouvrir l'exemple

Float

Source

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>Boîte flottante</title>
    <style>
     body {
       width: 90%;
       max-width: 900px;
       margin: 0 auto;
       font:
         0.9em/1.2 Arial,
         Helvetica,
         sans-serif;
     }

     .box {
       float: left;
       margin-right: 15px;
       width: 150px;
       height: 100px;
       border-radius: 5px;
       background-color: rgb(207, 232, 220);
       padding: 1em;
     }
```

Résultat

Ouvrir l'exemple

Boîte standard vs alternative

- Dans le **modèle standard** les propriétés de largeur (width) et de hauteur (height), définissent la taille du contenu.
- La taille du padding et de la bordure (s'ils existent) s'ajoutent à la largeur et à la hauteur définies dans width et height pour obtenir les dimensions totales occupées par la boîte.
- La marge étant extérieure, elle ne rentre pas dans le compte.

- Exemple

```
.box {
    width: 350px;
    height: 150px;
    margin: 10px;
    padding: 25px;
    border: 5px solid black;
}
```

- Taille de la boîte:
  - largeur = 410px (350 + 25 + 25 + 5 + 5), et
  - hauteur = 210px (150 + 25 + 25 + 5 + 5)

Boîte standard vs alternative

- Dans le **modèle alternatif** les paramètres width et height définissent la largeur et la hauteur totale de la boîte.
- Pour obtenir la taille du contenu, il faut retirer aux dimensions la taille du padding et de la bordure.
- Le navigateur utilise le modèle standard par défaut.
- Pour utiliser le modèle alternatif, il faut définir la propriété `box-sizing: border-box;` sur notre boîte.

- Exemple

```
.box {
    box-sizing: border-box;
    width: 350px;
    height: 150px;
    margin: 10px;
    padding: 25px;
    border: 5px solid black;
}
```

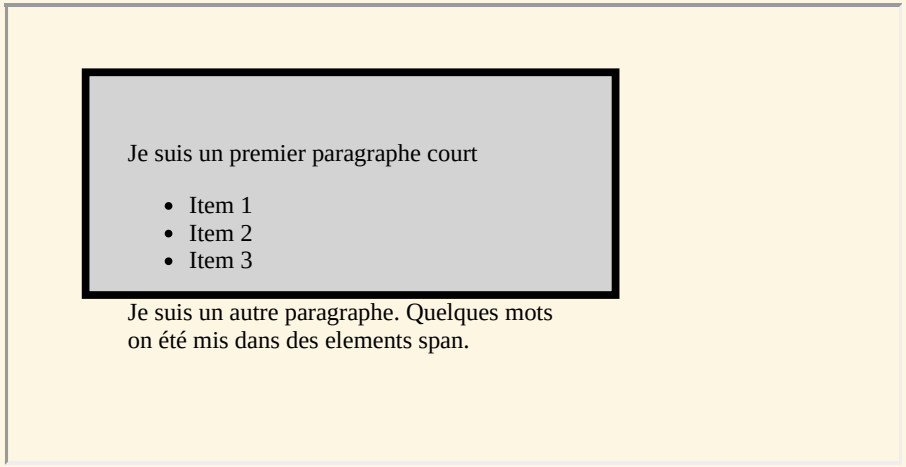
- Taille de la boîte:
  - largeur = 350px
  - hauteur = 150px

Exemple

Source

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, i
    <title>Exemple box</title>
    <style>
     .box {
       box-sizing:border-box;
       background-color:lightgrey;
       width: 350px;
       height: 150px;
       margin: 10px;
       padding: 25px;
       border: 5px solid black;
     }
    </style>
</head>
<body>
  <div class="box">
    <p>Je suis un premier paragraphe court</p>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ul>
```

Résultat



Ouvrir l'exemple

Positionnement (position)

- Il y a différents types de positionnement que vous pouvez appliquer à des éléments HTML.
- Pour utiliser un type particulier de positionnement sur un élément, nous utilisons la propriété position.
- Les différents types de positionnement:
  - **static**
  - **relative**
  - **absolute**
  - **fixed**
  - **sticky**

static

- Comportement normal (par défaut).
- L'élément est alors positionné dans le flux avec sa position.
- Les propriétés top, right, bottom, left et z-index ne s'appliquent pas.

relative

- L'élément est positionné dans le flux normal du document puis décalé, par rapport à lui-même, selon les valeurs fournies par `top`, `right`, `bottom` et `left`.
- L'élément est positionné par rapport à sa position normale.
- Cette valeur crée un nouveau contexte d'empilement lorsque z-index ne vaut pas auto

relative exemple

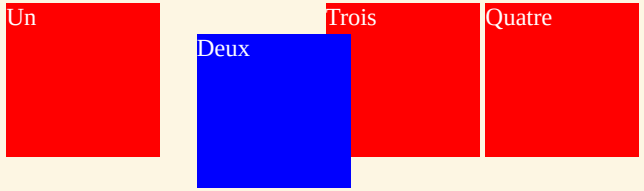
Source

```
<!DOCTYPE html>
<meta http-equiv="content-type" content="text/html; char
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
<title></title>
<style>
* {
  box-sizing: border-box;
}

.box {
  display: inline-block;
  width: 100px;
  height: 100px;
  background: red;
  color: white;
}

#deux {
  position: relative;
  top: 20px;
  left: 20px;
  background: blue;
}
</style>
<body>
```

Résultat



Ouvrir l'exemple

absolute

- L'élément est retiré du flux normal et aucun espace n'est créé pour l'élément sur la page.
- L'élément est positionné par rapport à son premier parent positionné (`position:relative`).
- Si aucun parent n'est positionné, l'élément est positionné par rapport au corps de la page (`body`).
- Les propriétés top, right, bottom, left et z-index s'appliquent.

absolute exemple

Source

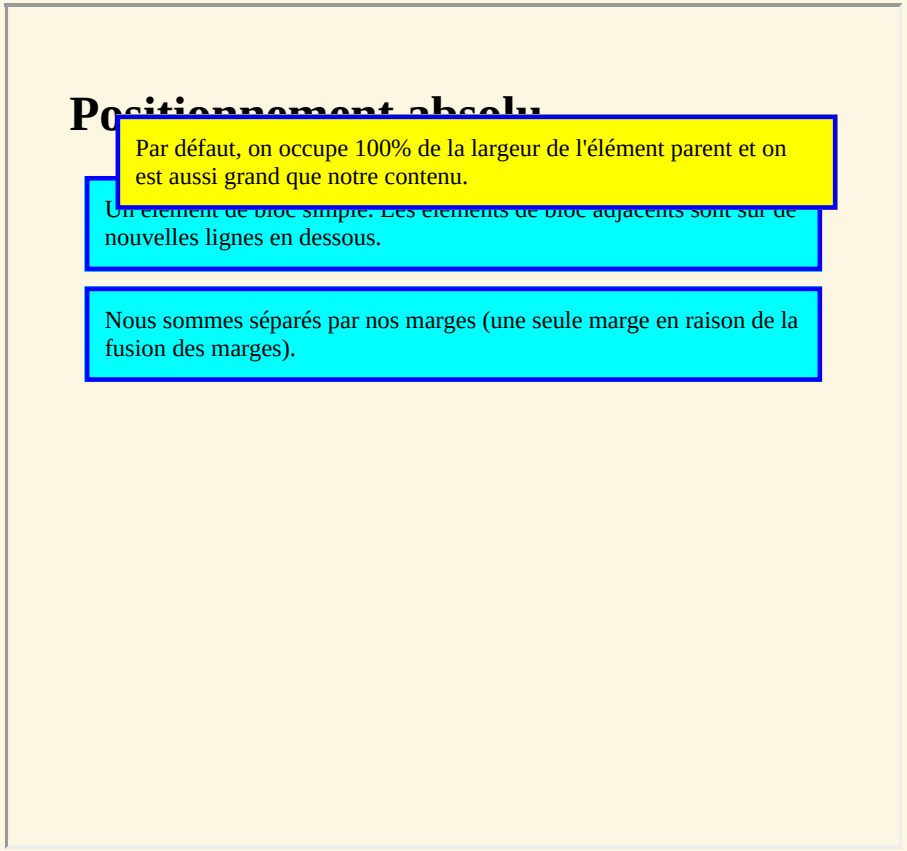
```
<!DOCTYPE html>
<meta http-equiv="content-type" content="text/html; char
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
<title></title>
<style>
 * {
   box-sizing: border-box;
 }

 body {
   width: 500px;
   margin: 0 auto;
 }

 p {
   background: aqua;
   border: 3px solid blue;
   padding: 10px;
   margin: 10px;
 }

 span {
   background: red;
   border: 1px solid black;
 }
```

Résultat



Ouvrir l'exemple

fixed

- L'élément est retiré du flux normal et aucun espace n'est créé pour l'élément sur la page.
- L'élément est positionné par rapport à la fenêtre du navigateur.
- Les propriétés top, right, bottom, left et z-index s'appliquent.

fixed exemple

Source

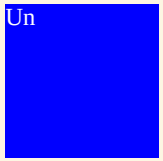
```
<!DOCTYPE html>
<meta http-equiv="content-type" content="text/html; char
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
<title></title>
<style>
 * {
   box-sizing: border-box;
 }

 .box {
   width: 100px;
   height: 100px;
   background: red;
   color: white;
 }

 #un {
   position: fixed;
   top: 80px;
   left: 10px;
   background: blue;
 }

 .outer {
   width: 500px;
```

Résultat



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam congue tortor eget pulvinar lobortis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nam ac dolor augue. Pellentesque mi mi, laoreet et dolor sit amet, ultrices varius risus. Nam vitae iaculis elit. Aliquam mollis interdum libero. Sed sodales placerat egestas. Vestibulum ut arcu aliquam purus viverra dictum vel sit amet mi. Duis nisl mauris, aliquam sit amet luctus eget, dapibus in enim. Sed velit augue, pretium a sem aliquam, congue porttitor tortor. Sed tempor nisl a lorem consequat, id maximus erat aliquet. Sed sagittis porta libero sed condimentum. Aliquam finibus lectus nec ante congue rutrum. Curabitur quam quam, accumsan id ultrices ultrices, tempor et tellus.

Ouvrir l'exemple

sticky

- L'élément est traité comme relatif jusqu'à ce qu'il atteigne un certain seuil, puis il devient fixe.
- L'élément est positionné par rapport à son premier parent positionné.
- Les propriétés top, right, bottom, left et z-index s'appliquent.
- Un tel élément « adhère » à l'ancêtre le plus proche qui dispose d'un mécanisme de défilement (c'est-à-dire quand overflow vaut scroll, auto ou overlay)
- Cette valeur ne fonctionne pas dans un élément pour lequel la propriété vaut overflow: hidden ou clip

sticky exemple

Source

```
<!DOCTYPE html>
<meta http-equiv="content-type" content="text/html; char
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
<title></title>
<style>
 * {
   box-sizing: border-box;
 }

 dl {
   height:300px;
   overflow:scroll;
 }
 dl > div {
   background: #fff;
   padding: 24px 0 0 0;
 }

 dt {
   background: #b8c1c8;
   border-bottom: 1px solid #989ea4;
   border-top: 1px solid #717d85;
   color: #fff;
   font:
     bold 18px/21px Helvetica
```

Résultat

A
Andrew W.K.
Apparat
Arcade Fire
At The Drive-In
Aziz Ansari

Ouvrir l'exemple

Flexbox / Grid

Les boîtes flexibles et la disposition en grille (MDN)

- Les modèles de mise en page Flexbox et Grid sont des alternatives modernes aux anciens modèles de mise en page CSS basés sur les boîtes flottantes (float) et le positionnement (position).
- Flexbox et Grid sont conçus pour être plus efficaces, plus intuitifs et plus flexibles que les anciens modèles.
- Flexbox est idéal pour les mises en page à une dimension (une seule ligne ou une seule colonne), tandis que Grid est plus adapté aux mises en page à deux dimensions (lignes et colonnes).
- Bien que Flexbox et Grid soient des modèles de mise en page puissants, il est important de noter qu'ils ne remplacent pas complètement les anciens modèles basés sur les boîtes flottantes et le positionnement.
- Dans certains cas, les deux méthodes peuvent fonctionner sans problème. En les utilisant plus fréquemment, vous pourrez voir qu'elles répondent à des besoins différents et vous utiliserez sans doute ces deux méthodes dans votre CSS. Comme souvent, il n'y a pas de solution miracle et de « bonne » ou de « mauvaise » réponse.

Flexbox

A guide to flexbox

- Flexbox est une méthode de mise en page selon un axe principal, permettant de disposer des éléments en ligne ou en colonne. Les éléments se dilatent ou se rétractent pour occuper l'espace disponible.
- Pendant longtemps, les seuls outils de mise en page CSS fiables et compatibles avec les navigateurs, étaient les propriétés concernant les flotteurs (boîtes flottantes) et le positionnement.
- Les simples exigences de mise en page suivantes sont difficiles sinon impossibles à réaliser de manière pratique et souple avec ces outils :
  - Centrer verticalement un bloc de contenu dans son parent ;
  - Faire que tous les enfants d'un conteneur occupent tous une même quantité de hauteur/largeur disponible selon l'espace offert ;
  - Faire que toutes les colonnes dans une disposition multi-colonnes aient la même hauteur même si leur quantité de contenu diffère.
- Flexbox facilite considérablement les tâches de mise en page.

Flexbox - exemple

Source

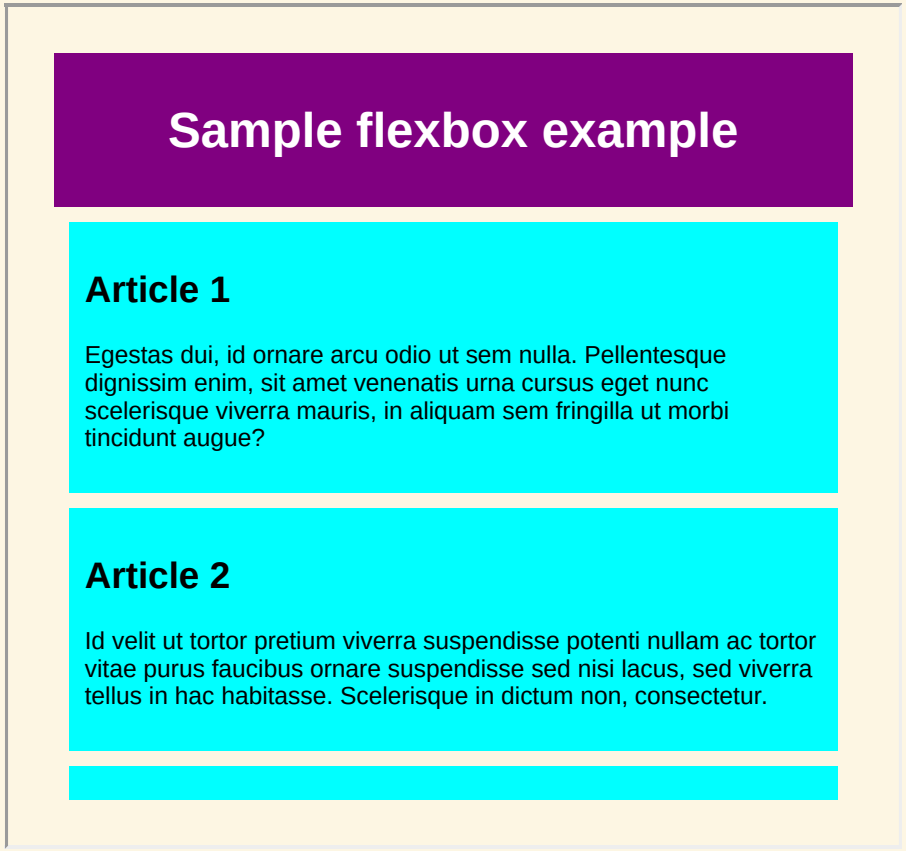
```
<!DOCTYPE html>
<meta http-equiv="content-type" content="text/html; char
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
<title>Flexbox 0 — starting code</title>
<style>
 html {
   font-family: sans-serif;
 }

 body {
   margin: 0;
 }

 header {
   background: purple;
   height: 100px;
 }

 h1 {
   text-align: center;
   color: white;
   line-height: 100px;
   margin: 0;
 }
```

Résultat



Ouvrir l'exemple

Flexbox - exemple

Source

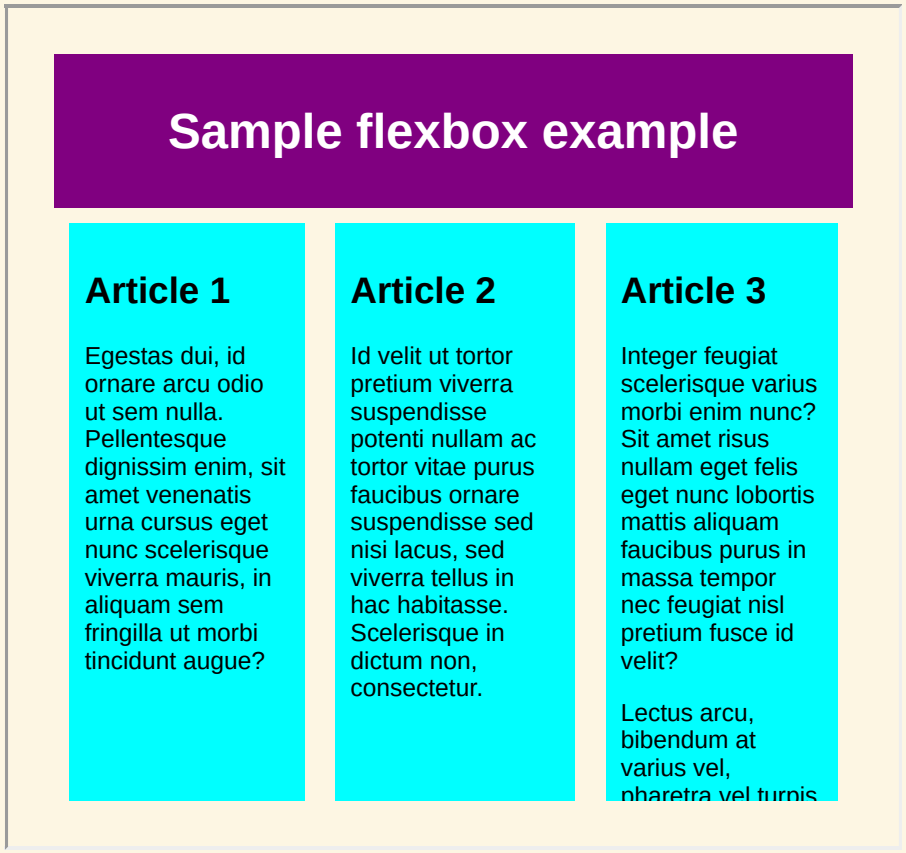
```
<!DOCTYPE html>
<head>
  <meta http-equiv="content-type" content="text/html; ch
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>Flexbox 0 – starting code</title>
  <style>
   html {
     font-family: sans-serif;
   }

   body {
     margin: 0;
   }

   header {
     background: purple;
     height: 100px;
   }

   h1 {
     text-align: center;
     color: white;
     line-height: 100px;
     margin: 0;
   }
```

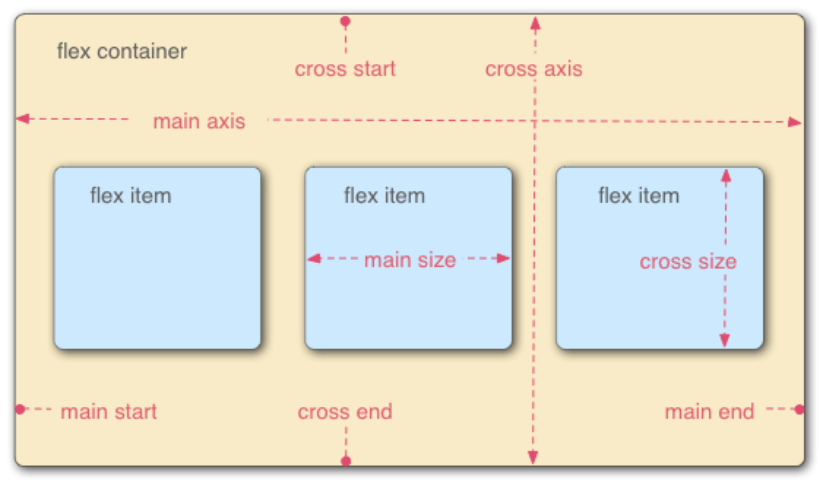
Résultat



Ouvrir l'exemple

Flexbox

- L'**axe principal (main axis)** est l'axe de la direction dans laquelle sont disposés les éléments flex (par exemple, horizontalement sur la page, ou verticalement de haut en bas de la page). Le début et la fin de cet axe sont appelés l'**origine principale (main start)** et la **fin principale (main end)**.
- L'**axe croisé (cross axis)** est l'axe perpendiculaire à l'axe principal, c'est-à-dire à la direction dans laquelle sont disposés les éléments flex. Le début et la fin de cet axe sont appelés le **début (cross start)** et la **fin (cross end)** de l'axe croisé.
- L'élément parent dont la propriété est `display: flex` (`<section>` dans notre exemple) est appelé le **conteneur flex (flex container)**.
- Les éléments disposés en tant que boîtes flexibles à l'intérieur du conteneur flex sont appelés **éléments flex (flex items)** (les éléments `<article>` dans notre exemple).



Source MDN

Flexbox propriétés

- **flex-direction** column/row; (reverse)
- **flex-wrap** wrap; (no-wrap) // Les éléments peuvent passer à la ligne
- **flex-flow** mix de flex-direction et flex-wrap
- **flex** (mix des propriétés suivantes) taille des éléments flex (portion sans unité et/ou taille min avec une unité)
  - **flex-grow** croissance
  - **flex-shrink** rétrécissement (en cas de manque de place)
  - **flex-basis** taille initiale / minimum
- **justify-content** alignement sur l'axe principale
- **align-item** alignement sur l'axe opposé
- **order**

Flexbox - exemple

Source

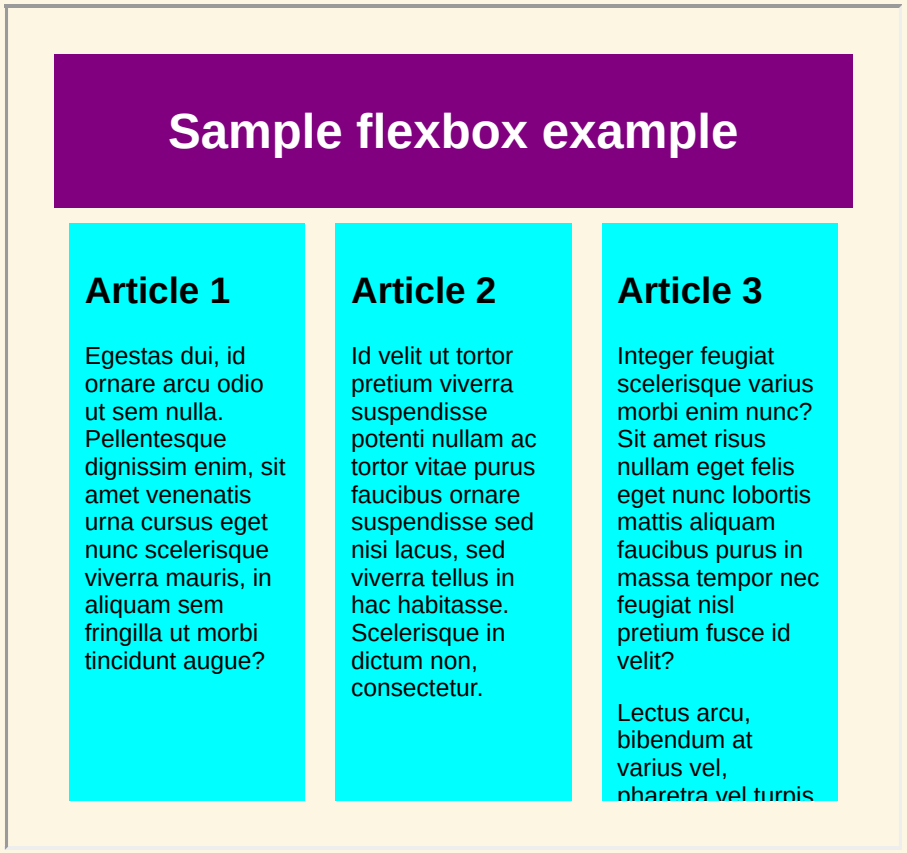
```
<!DOCTYPE html>
<head>
  <meta http-equiv="content-type" content="text/html; ch
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>Flexbox 0 – starting code</title>
  <style>
   html {
     font-family: sans-serif;
   }

   body {
     margin: 0;
   }

   header {
     background: purple;
     height: 100px;
   }

   h1 {
     text-align: center;
     color: white;
     line-height: 100px;
     margin: 0;
   }
```

Résultat



Ouvrir l'exemple

Grid

Complete guide grid

- Système de mise en page bidimensionnel. Il vous permet de disposer les contenus en lignes et en colonnes.
- Une mise en page grid est constituée généralement de rangées (colonnes ou lignes). L'espace entre chaque ligne ou colonne est communément appelé gutter (goutière).

```
.container {
    display: grid;
}
```

Grid propriétés

- **grid-template-columns**, **grid-template-rows** : définir le nombre de colonnes et de lignes
- **grid-template-areas**, **grid-area** : nommer les zones
- Les fonctions:
  - repeat() permet de répéter un fragment d'une liste de pistes. Autrement dit, lorsqu'on a une grille avec de nombreuses lignes/colonnes, cela permet de réutiliser un même motif sur la grille. On a alors des règles plus concises.
  - minmax() définit un intervalle de taille supérieure ou égale à min et inférieure ou égale à max.

Grid placement

Source

```
<!DOCTYPE html>
<html lang="en-US"><head>
<meta http-equiv="content-type" content="text/html; char
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>CSS Grid - line-based placement starting poin
    <style>
      body {
        width: 90%;
        max-width: 900px;
        margin: 2em auto;
        font: 0.9em/1.2 Arial, Helvetica, sans-serif;
      }

      .container {
        display: grid;
        grid-template-columns: 1fr 3fr;
        gap: 20px;
      }

      header,
      footer {
        border-radius: 5px;
        padding: 10px;
        background-color: rgb(207, 232, 220);
        border: 2px solid rgb(79, 185, 227);
```

Résultat

Ouvrir l'exemple

Grid placement

Source

```
<!DOCTYPE html>
<html lang="en-US"><head>
<meta http-equiv="content-type" content="text/html; char
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>CSS Grid - line-based placement starting poin
    <style>
     body {
       width: 90%;
       max-width: 900px;
       margin: 2em auto;
       font: 0.9em/1.2 Arial, Helvetica, sans-serif;
     }

     .container {
       display: grid;
       grid-template-columns: 1fr 3fr;
       gap: 20px;
     }

     header,
     footer {
       border-radius: 5px;
       padding: 10px;
       background-color: rgb(207, 232, 220);
       border: 2px solid rgb(79, 185, 227);
```

Résultat

Ouvrir l'exemple

Grid placement avec grid-template-areas

Source

```
<!DOCTYPE html>
<html lang="en-US"><head>
<meta http-equiv="content-type" content="text/html; char
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>CSS Grid - line-based placement starting poin
    <style>
     body {
       width: 90%;
       max-width: 900px;
       margin: 2em auto;
       font: 0.9em/1.2 Arial, Helvetica, sans-serif;
     }

     header,
     footer {
       border-radius: 5px;
       padding: 10px;
       background-color: rgb(207, 232, 220);
       border: 2px solid rgb(79, 185, 227);
     }

     aside {
       border-right: 1px solid #999;
     }
      .container {
```

Résultat

Ouvrir l'exemple

Exemple repeat / minmax

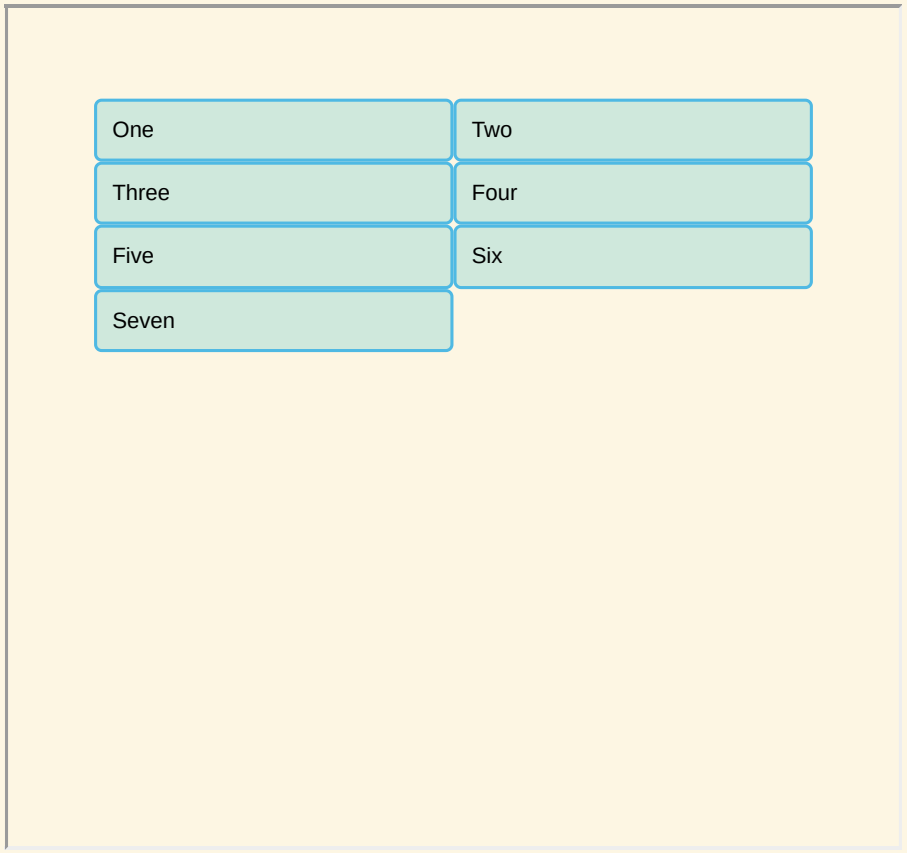
Source

```
<!DOCTYPE html>
<html lang="en-US"><head>
<meta http-equiv="content-type" content="text/html; char
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>CSS Grid - line-based placement starting poin
    <style>
     body {
       width: 90%;
       max-width: 900px;
       margin: 2em auto;
       font: 0.9em/1.2 Arial, Helvetica, sans-serif;
     }

     div {
       border-radius: 5px;
       padding: 10px;
       background-color: rgb(207, 232, 220);
       border: 2px solid rgb(79, 185, 227);
     }

     .box {
       display: grid;
       grid-template-columns: repeat(auto-fill, minmax(2
     }
    </style>
```

Résultat



Ouvrir l'exemple

Débordements de contenu (overflow)

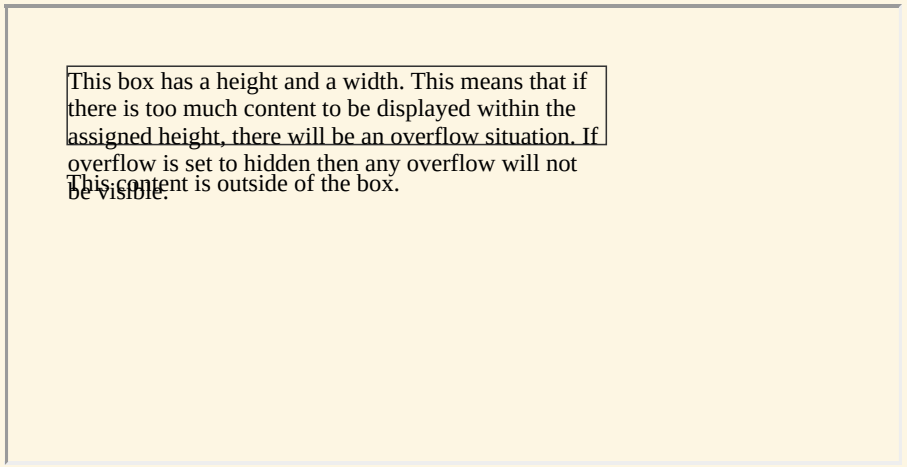
Source

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>Overflow</title>
    <style>
     .box {
       border: 1px solid #333333;
       width: 350px;
       height: 50px;
     }
    </style>
  </head>
  <body>
    <div class="box">This box has a height and a width.

    <p>This content is outside of the box.</p>

  </body>
</html>
```

Résultat



Ouvrir l'exemple

Débordements de contenu (overflow)

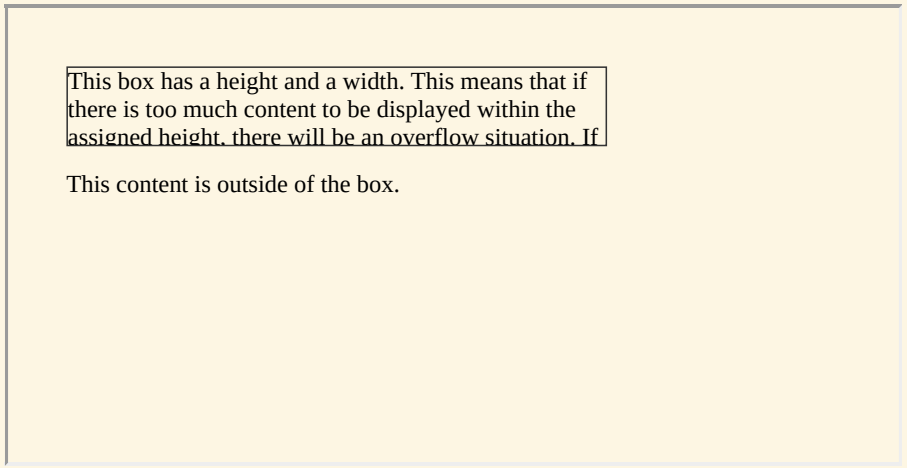
Source

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>Overflow</title>
    <style>
     .box {
       border: 1px solid #333333;
       width: 350px;
       height: 50px;
       overflow: hidden;
     }
    </style>
  </head>
  <body>
    <div class="box">This box has a height and a width.

    <p>This content is outside of the box.</p>

  </body>
</html>
```

Résultat



Ouvrir l'exemple

Débordements de contenu (overflow)

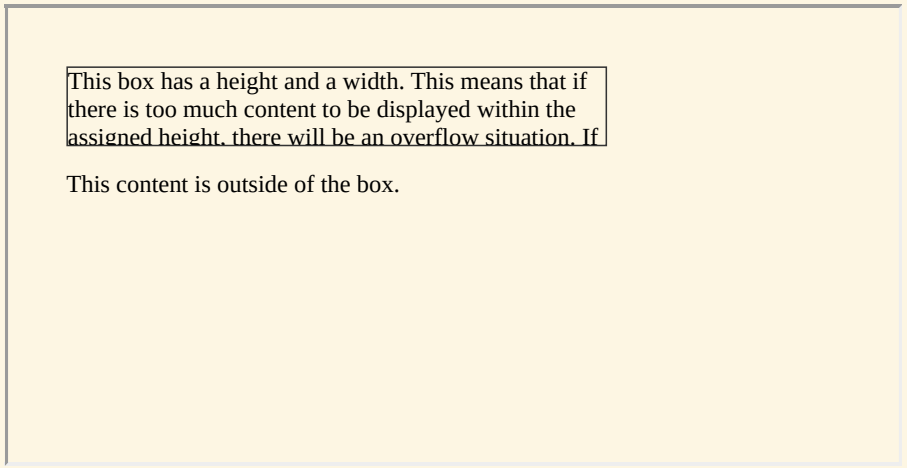
Source

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>Overflow</title>
    <style>
     .box {
       border: 1px solid #333333;
       width: 350px;
       height: 50px;
       overflow: auto;
     }
    </style>
  </head>
  <body>
    <div class="box">This box has a height and a width.

    <p>This content is outside of the box.</p>

  </body>
</html>
```

Résultat



Ouvrir l'exemple