

Philippe GENOUD (LIG-SocSimK)
Philippe.Genoud@imag.fr

M2CCI – M2 GEOMAS 2026-2027
cours Technologies du Web (TW)

CSS (2ème partie)

positionnement des éléments

dernière modification : 30/09/2026 12:24

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Affichage des éléments HTML
flux du document

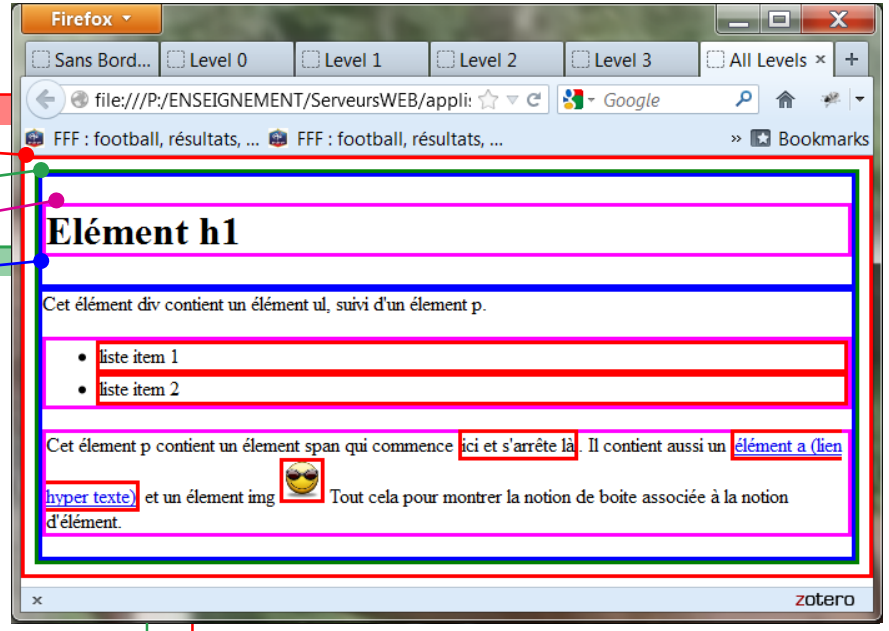
- chaque élément HTML considéré comme une **boîte rectangulaire**.
- **flux du document** : ordre dans lequel le navigateur affiche ces boîtes.

  - **flux normal** (par défaut)
    - un élément père est un conteneur
    - un élément fils s'affiche dans son conteneur
    - élément bloc
      - s'affiche en dessous de son frère précédent.
      - occupe toute la largeur disponible dans son conteneur.
    - élément en ligne
      - s'affiche à côté de son frère précédent.
      - retour à la ligne quand il n'y a plus de place dans le conteneur.
  - **flux personnalisé**
    - certaines propriétés CSS permettent de sortir des éléments du flux normal

Positionnement en flux

- exemple de flux normal

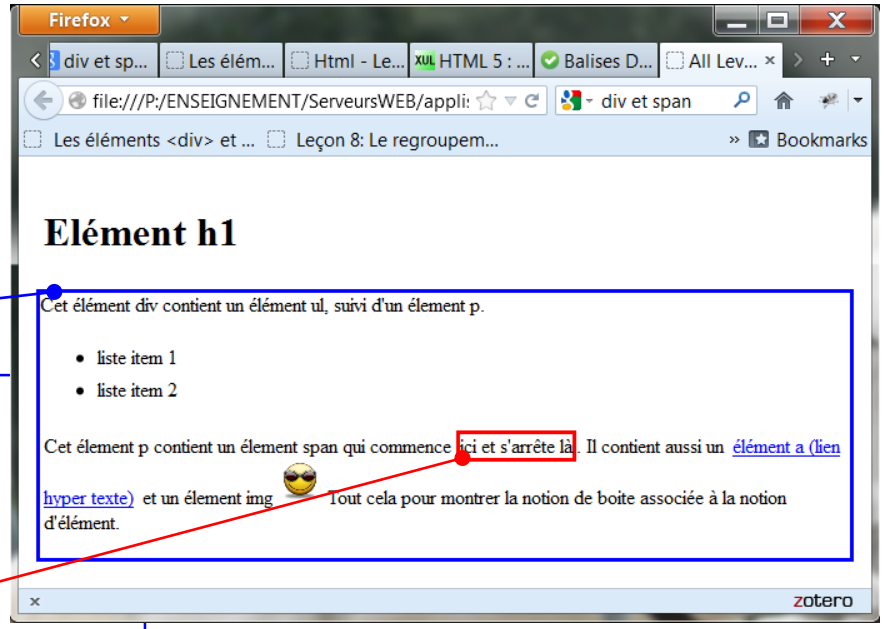
```
<html lang="fr">
  <head>
    <title>Exemple boites</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <header>
          <h1>Elément h1</h1>
    </header>
    <div>
    Cet élément div contient un élément ul, suivi d'un élement p.
      <ul id="menuAccesRapide">
        <li>liste item 1</li>
        <li>liste item 2</a></li>
      </ul>
      <p>
        Cet élement p contient un élement span qui
        commence <span> ici et s'arrête là</span>.
        Il contient aussi un <a href="#">élément a (lien hyper texte)</a>
        et un élement img <img src="smiley.png" height="30px" width="30px"/>
        Tout cela pour montrer la notion de boite associée
        à la notion d'élément.
      </p>
    </div>
  </body>
</html>
```



Positionnement en flux

- ** ** et **** éléments génériques sans information structurante prédéfinie
  - utilisés en association avec des feuilles de style CSS ou du Javascript via les attributs id, class ou style

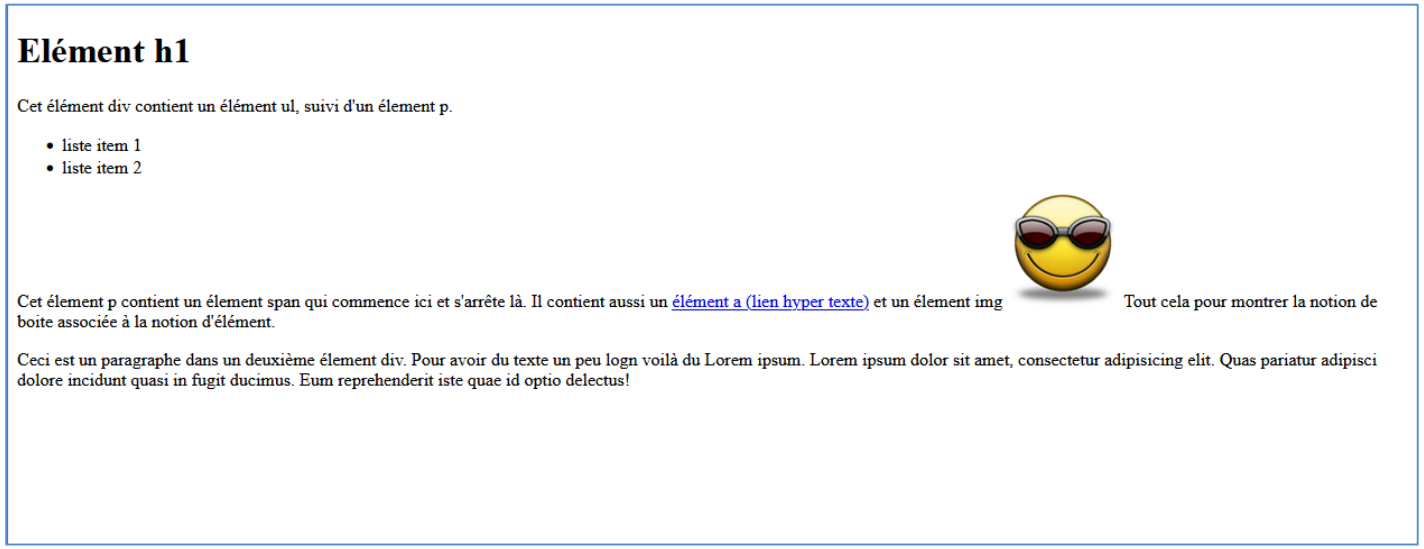
```
<html lang="fr">
  <head>
    <title>Exemple boites</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <header>
          <h1>Elément h1</h1>
    </header>
    <div>
    Cet élément div contient un élément ul, suivi d'un élement p.
      <ul id="menuAccesRapide">
        <li>liste item 1</li>
        <li>liste item 2</a></li>
      </ul>
      <p>
        Cet élement p contient un élement span qui
        commence <span> ici et s'arrête là</span>.
        Il contient aussi un <a href="#">élément a (lien hyper texte)</a>
        et un élement img <img src="smiley.png" height="30px" width="30px"/>
        Tout cela pour montrer la notion de boite associée
        à la notion d'élément.
      </p>
    </div>
  </body>
</html>
```

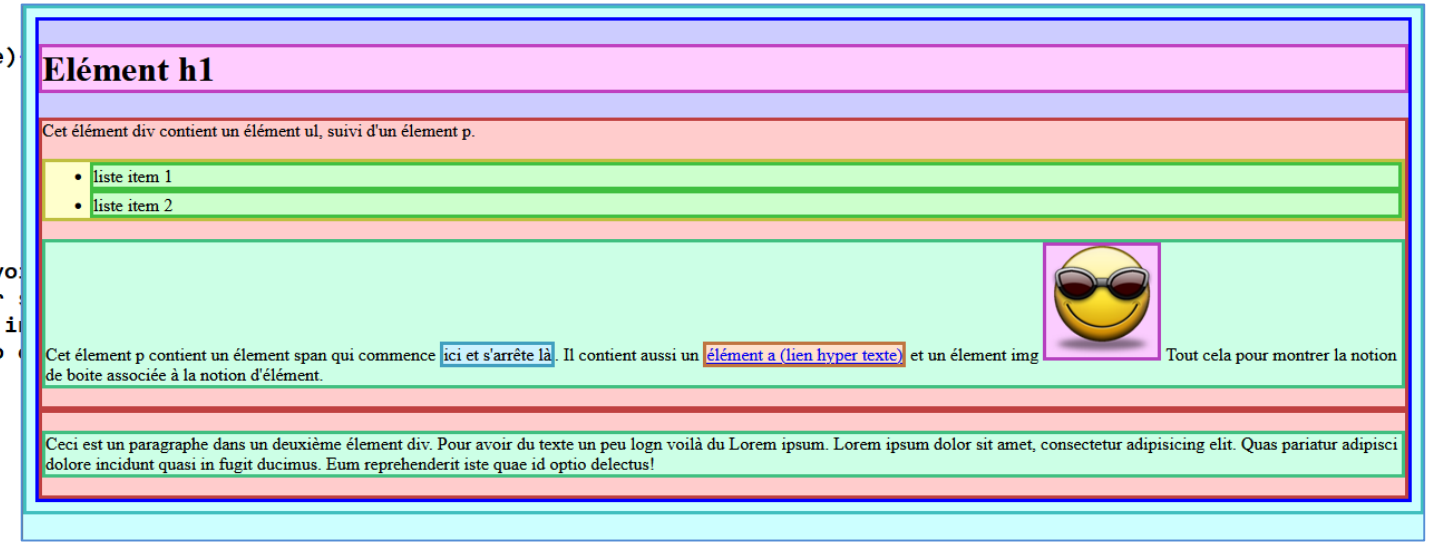


div : balise de type bloc
→zone rectangulaire qui ne peut être répartie sur plusieurs lignes
→ peut contenir tous les autres éléments de type bloc ou en ligne

span : balise de type en ligne (*inline*)
→ s'inscrit dans le flux du contenu, peut être répartie sur plusieurs lignes
→ peut contenir tous les autres éléments en ligne

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <title>All Levels</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/styles1.css">
</head>
<body>
    <header>
        <h1>Elément h1</h1>
    </header>
    <div>
        Cet élément div contient un élément ul, suivi d'un élement p.
        <ul>
            <li>liste item 1</li>
            <li>liste item 2</li>
        </ul>
        <p>
            Cet élement p contient un élement span qui
            commence <span> ici et s'arrête là</span>.
            Il contient aussi un <a href="#">élément a (lien hyper texte)
            et un élement img <img src="smiley.png" alt="un smiley">
            Tout cela pour montrer la notion de boite associée
            à la notion d'élément.
        </p>
    </div>
    <div>
        <p>
            Ceci est un paragraphe dans un deuxième élement div. Pour avo
            du texte un peu logn voilà du Lorem ipsum. Lorem ipsum dolor
            consectetur adipisicing elit. Quas pariatur adipisci dolore i
            quasi in fugit ducimus. Eum reprehenderit iste quae id optio
        </p>
    </div>
</body>
</html>
```

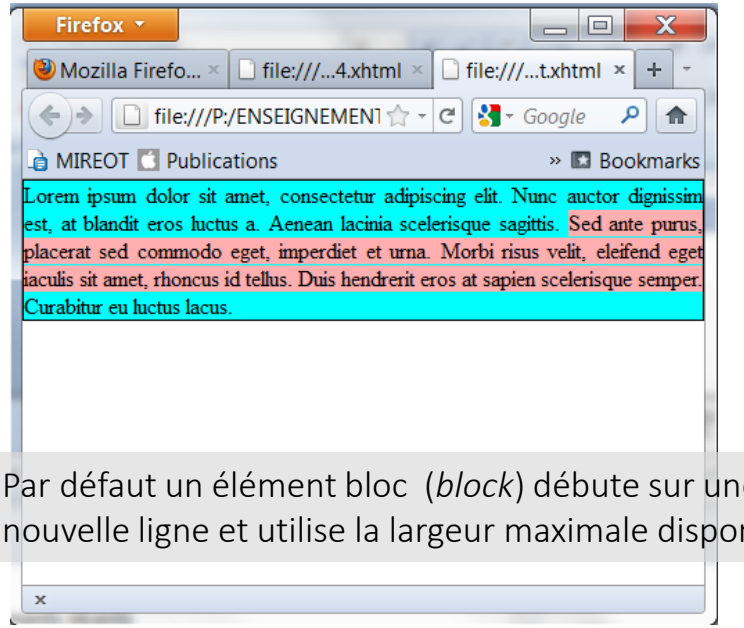




Positionnement en flux

- positionnement dans le flux en supprimant les marges par défaut

```
<div id ="div1">
    <p id="p1">
        Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc auctor dignissim est, at blandit eros luctus a.
        Aenean lacinia scelerisque sagittis. <span id="span1">Sed ante purus, placerat sed commodo eget,
        imperdiet et urna. Morbi risus velit, eleifend eget iaculis sit amet, rhoncus id tellus.
        Duis hendrerit eros at sapien scelerisque semper.</span> Curabitur eu luctus lacus.
    </p>
</div>
```




```
body {
    text-align: justify;
    padding: 0px;
    margin: 0px;
}
p {
    border-style: solid;
    border-width: thin;
    padding: 0px;
    margin: 0px;
}
```

```
#div1 {
    background: yellow;
}
#p1 {
    background: cyan;
}
#span1 {
    background: #ffafaf;
}
```

Efface les marges par défaut

Par défaut un élément bloc (*block*) débute sur une nouvelle ligne et utilise la largeur maximale disponible

Par défaut un élément en-ligne (*in line*) ne débute pas sur une nouvelle ligne et utilise la largeur nécessaire

Ce comportement est défini par la propriété **`display`** (valeurs: **`inline`**, **`block`**, **`none`**, **`inline-block`**….)

```
li {
    display: inline;
}
```

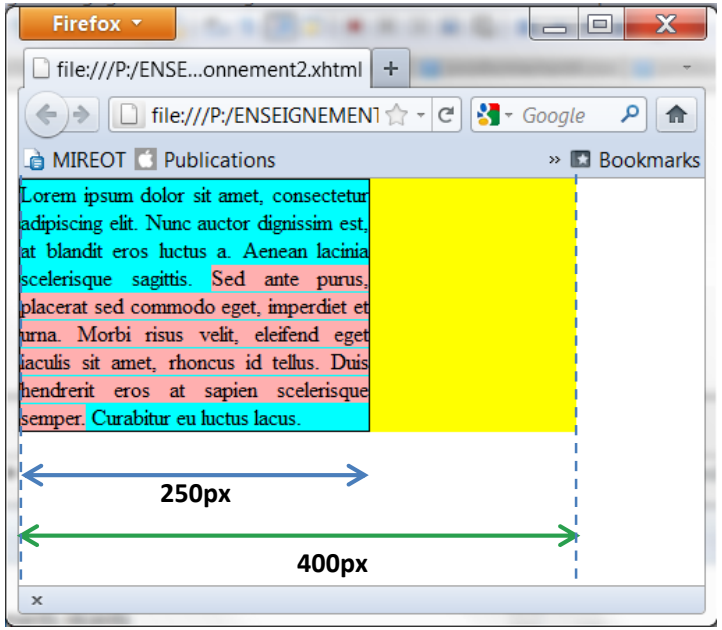
Try it yourself »

Dimensionnent et marge interne

attribution de dimensions aux éléments **div1** et **p1**

```
#div1 {
  background: yellow;
  width:400px;
}

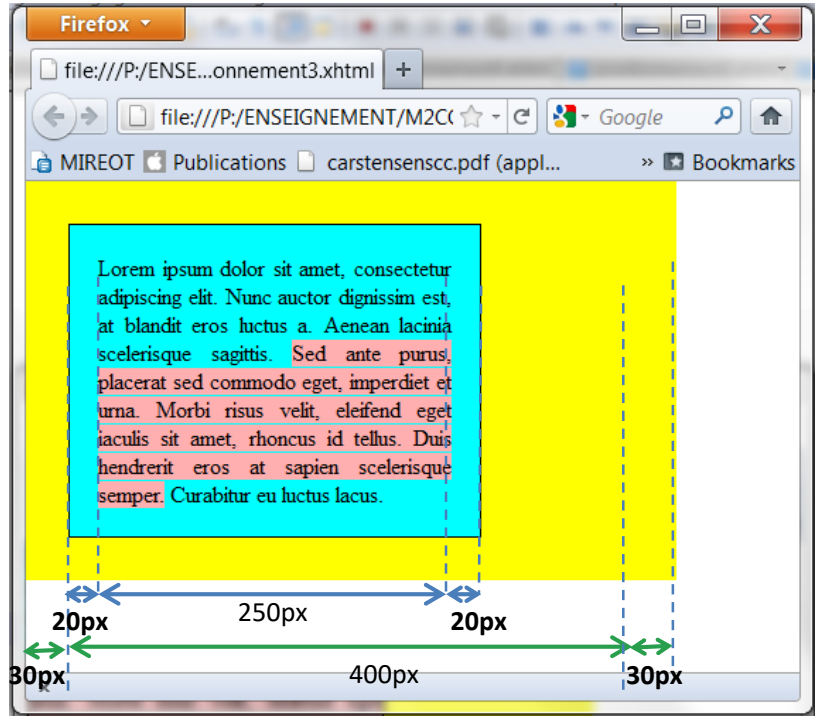
#p1 {
  background: cyan;
  width:250px;
}
```




```
#div1 {
  background: yellow;
  width:400px;
  padding: 30px;
}

#p1 {
  padding:20px;
  background: cyan;
  width:250px;
}
```

ajout de marges internes (*padding*) aux éléments **div1** et **p1**



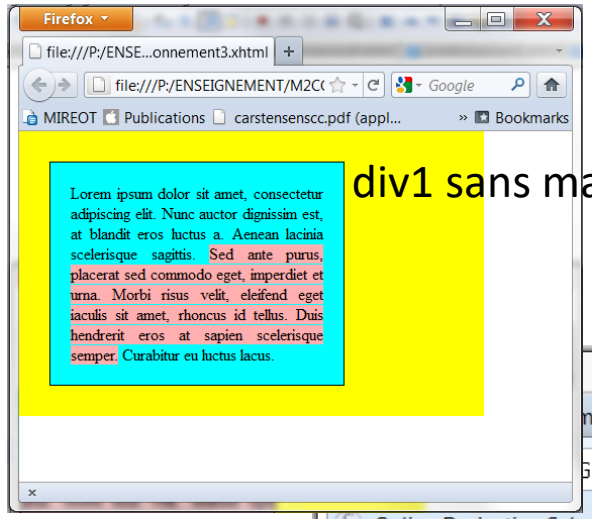
Définition d'une marge externe autour d'un bloc

ajout d'une marge externe (*margin*) a l'élément **div1**

```
#div1 {
  background-color: yellow;
  width:400px;
  padding:30px;
  margin:20px;
}

#p1{
  padding:20px;
  background: cyan;
  width:250px;

}
```



div1 sans marge

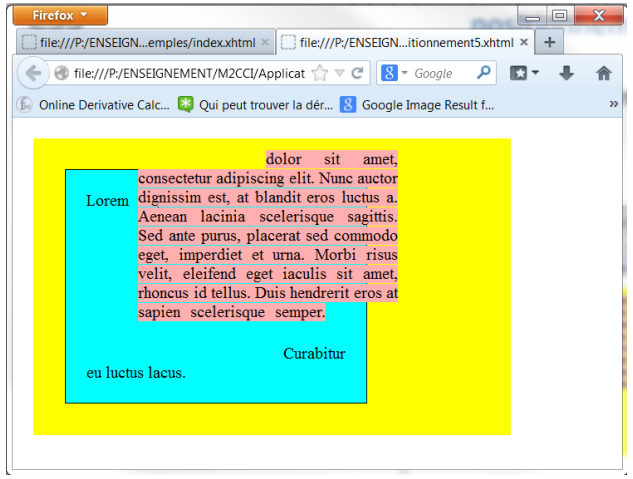


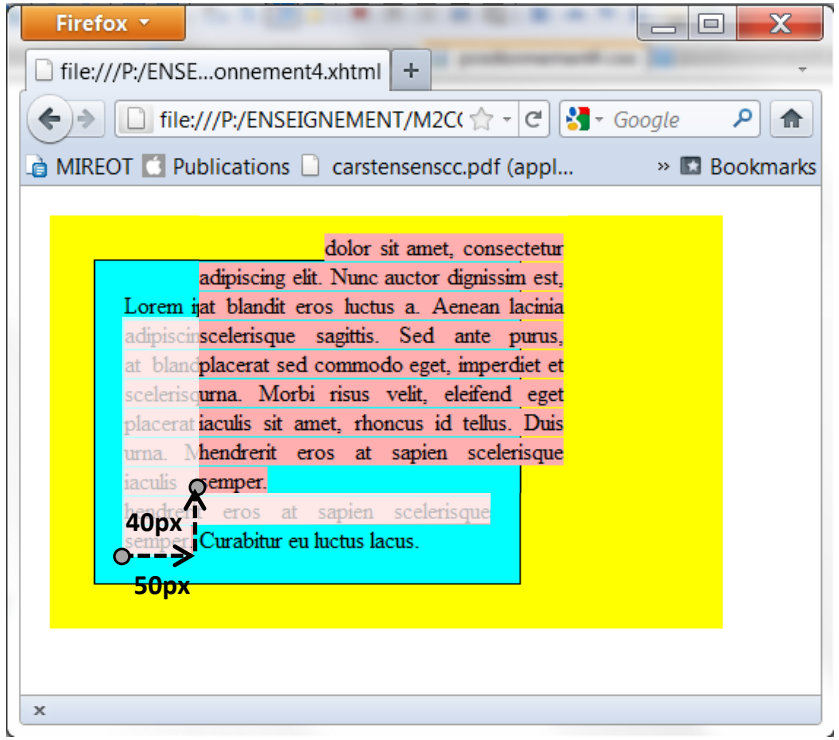
div1 avec marge

Positionnement relatif d'un élément

- positionnement relatif
  - l'élément est décalé à l'aide des propriétés **`top`**, **`right`**, **`left`**, **`bottom`** par rapport à sa position normale dans le flux courant
    ➔ l'élément peut prendre place au dessus de ses éléments frères.
  - n'affecte pas les boites qui l'entourent

```
#span1 {
    background: #ffafaf;
    position:relative;
    left:50px;
    bottom:40px;
}
```

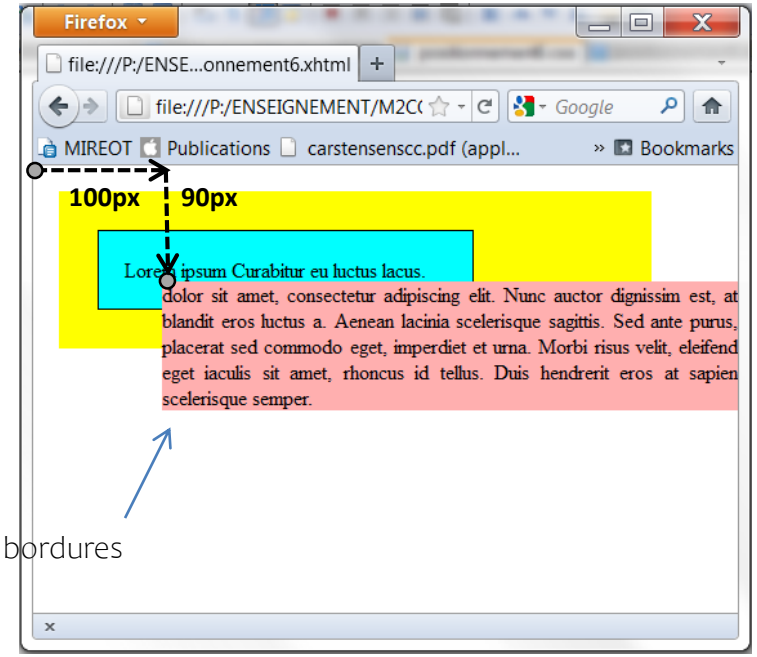




Positionnement absolu

- positionnement absolu
  - sort un élément du flux
  - l'élément ne participe plus à la position de ses frères
  - positionnement à l'aide des propriétés **`top`**, **`right`**, **`left`**, **`bottom`**
    - expriment des décalages non plus par rapport à position théorique (positionnement relatif) mais par rapport à la position d'un bloc conteneur de référence
    - boite conteneur de référence : le premier élément ancêtre positionné (relatif, absolu ou fixed)
      (élément <body> toujours considéré comme positionné)

```
#span1 {
    background: #ffafaf;
    position:absolute;
    left:100px;
    top:90px;
}
```

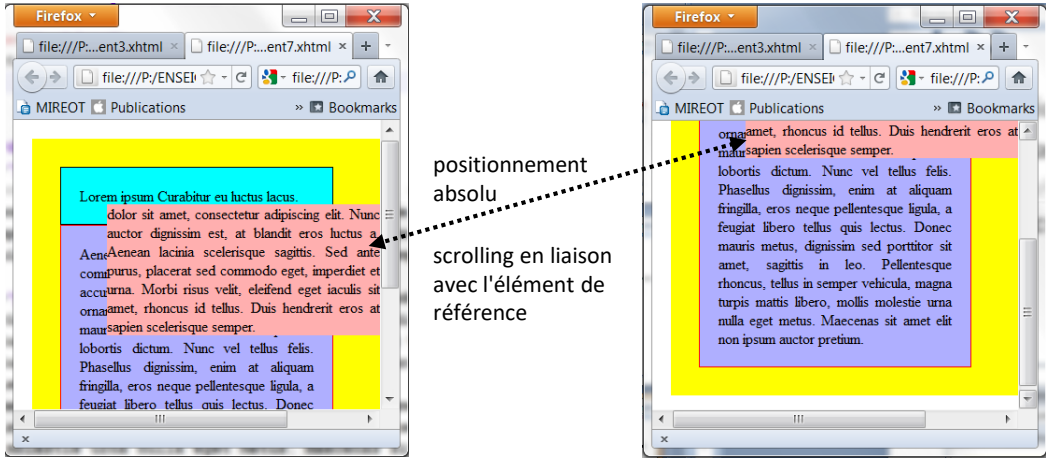


tout élément positionné en absolu est considéré de type bloc
=> éléments en ligne peuvent ainsi recevoir des dimensions et bordures

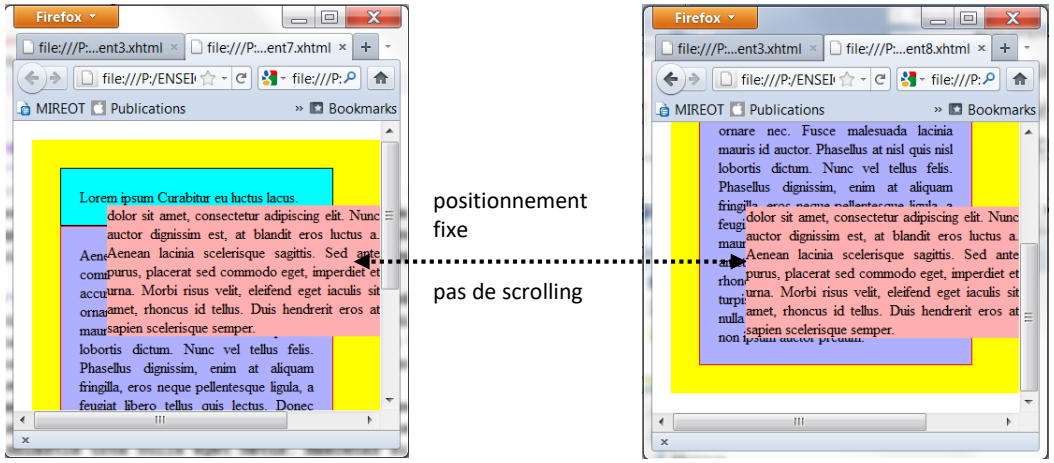
Positionnement fixe

- positionnement fixe
  - cas particulier du positionnement absolu
  - l'élément reste fixe dans la page par rapport à la zone de visualisation (pas de scroll)

```
#span1 {
    background: #ffafaf;
    position:absolute;
    left:100px;
    top:90px;
}
```




```
#span1 {
    background: #ffafaf;
    position:fixed;
    left:100px;
    top:90px;
}
```

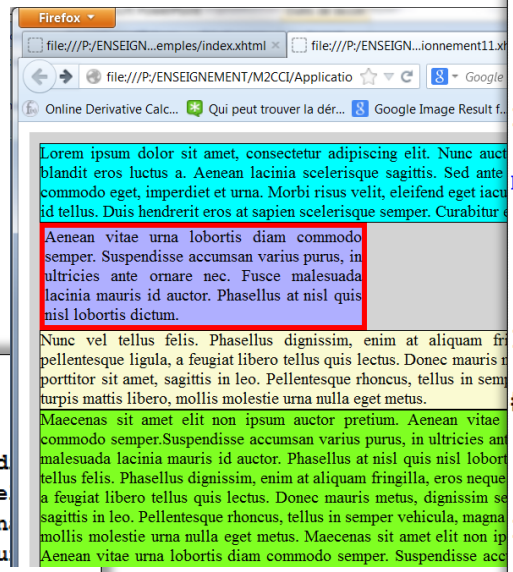


Positionnement flottant

- positionnement flottant
  1. sort l'élément du flux
  2. l'élément est "poussé" à gauche (float: left) ou à droite (float: right) de son conteneur.
  3. les éléments **qui le suivent** dans le conteneur prennent place autour de lui.

```
<body>
    <div id ="div1">
        <p id="p1">
            Lorem ipsum <span id="span1">dolor sit amet, consectetur ad
            dignissim est, at blandit eros luctus a. Aenean lacinia sce
            Sed ante purus, placerat sed commodo eget, imperdiet et urn
            Duis hendrerit eros at sapien scelerisque semper.</span> Cu
        </p>
        <p id="p2">
            Aenean vitae urna lobortis diam comm
            in ultricies ante ornare nec. Fusce
            Phasellus at nisl quis nisl lobortis
        </p>
        <p id="p3">
            Nunc vel tellus felis.  Phasellus d
            neque pellentesque ligula, a feugiat
            sagittis in leo. Pellentesque rhoncu
            mollis molestie urna nulla eget metu
        </p>
        <p id="p4">
            Maecenas sit amet elit non ipsum auc
            in ultricies ante ornare nec. Fusce
            ....
            mollis molestie urna nulla eget metu
        </p>
    </div>
</body>
```

positionnement normal (flux)



positionnement flottant (right)




```
body {
    text-align: justify;
    padding: 0px;
    margin: 0px;
}

p {
    border-style: solid;
    border-width: thin;
    padding:0px;
    margin: 0px;
}

#div1 {
    background-color: lightgrey;
    padding:10px;
    margin:10px;
}

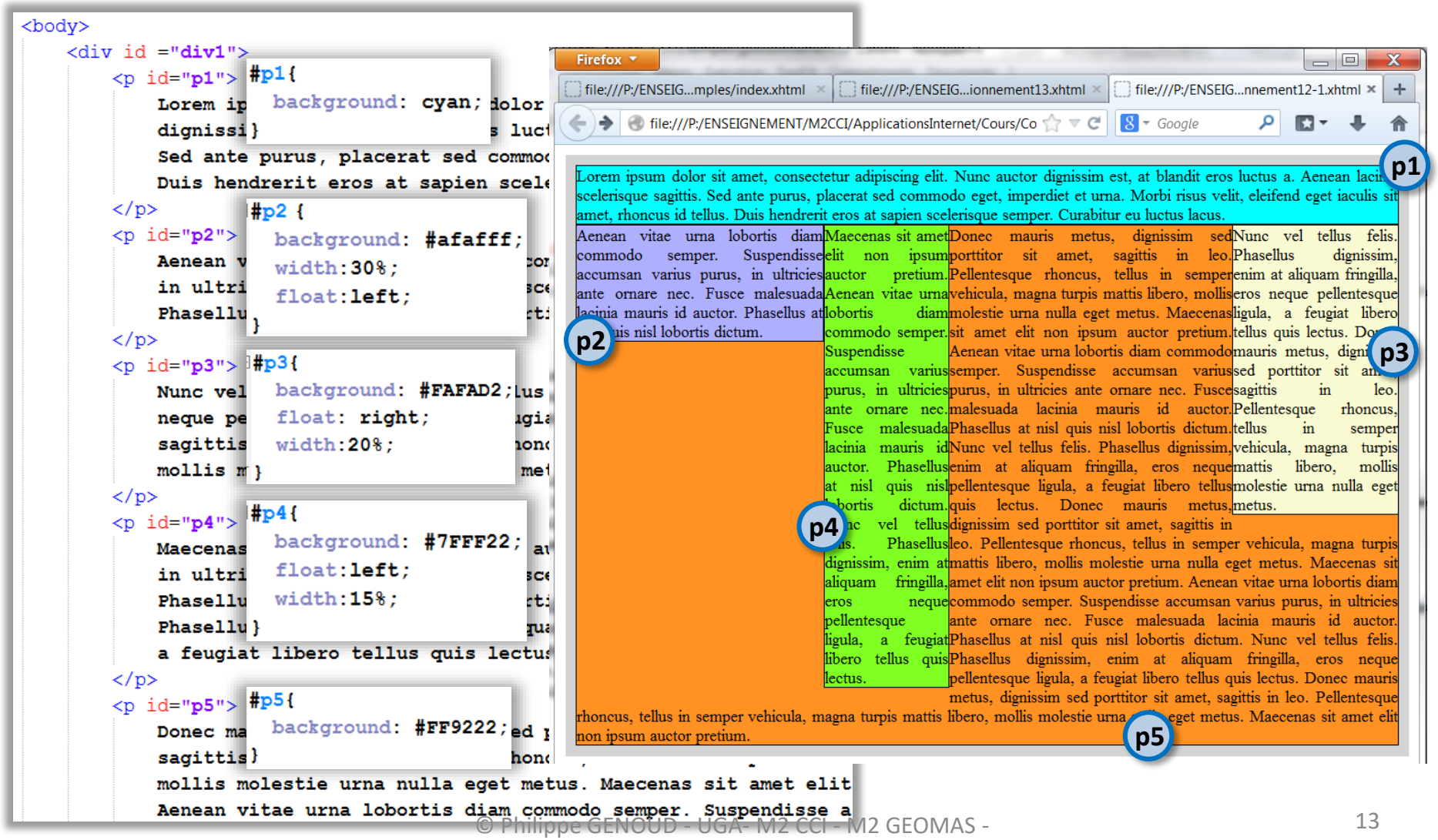
#p1{
    background: cyan;
}

#p2 {
    background: #afafff;
    border-color: red;
    border-width:5px;
    width:300px;
    float:right;
}

#p3{
    background: #FAFAD2;
}
#p4{
    background: #7FFF22;
}
```

Positionnement flottant

- possibilité de placer des blocs flottants côte à côte



Positionnement flottant

- **clear** permet d'interdire le voisinage avec un élément flottant (à droite, à gauche ou des deux côtés).

```
<body>
    <div id ="div1">
        <p id="p1">
            Lorem ip
            dignissi
            Sed ante purus, placerat sed commo
            Duis hendrerit eros at sapien scele
        </p>
        <p id="p2">
            Aenean v
            in ultri
            Phasellu
        </p>
        <p id="p3">
            Nunc vel
            neque pe
            sagittis
            mollis m
        </p>
        <p id="p4">
            Maecenas
            in ultri
            Phasellu
            Phasellu
            a feugiat libero tellus quis lectus
        </p>
        <p id="p5">
            Donec ma
            sagittis
            mollis molest
            Aenean vitae urna lobortis diam com
```

```
#p1{
    background: cyan;
}
```

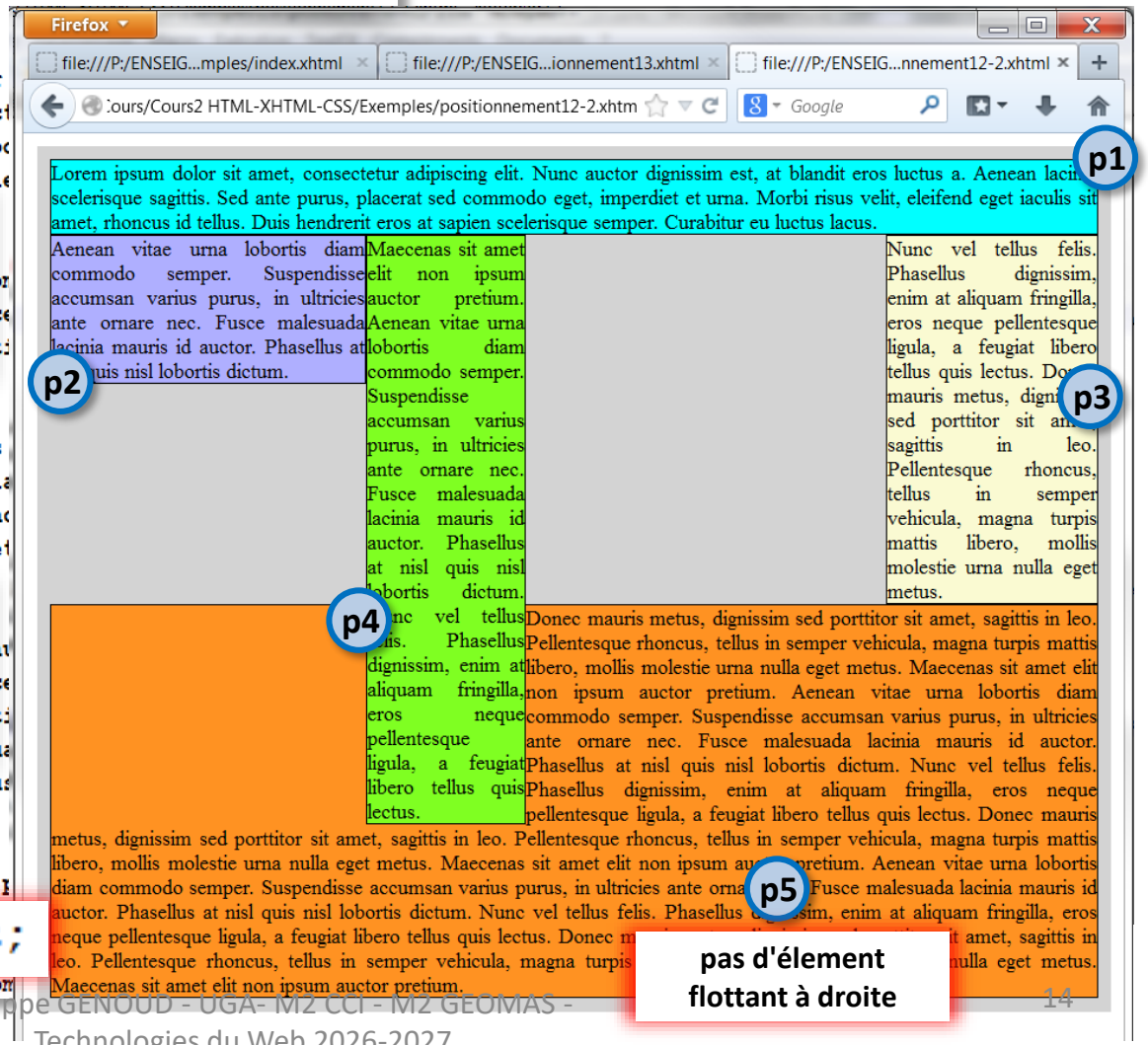
```
#p2 {
    background: #afafff;
    width:30%;
    float:left;
}
```

```
#p3{
    background: #FAFAD2;
    float: right;
    width:20%;
}
```

```
#p4{
    background: #7FFF22;
    float:left;
    width:15%;
}
```

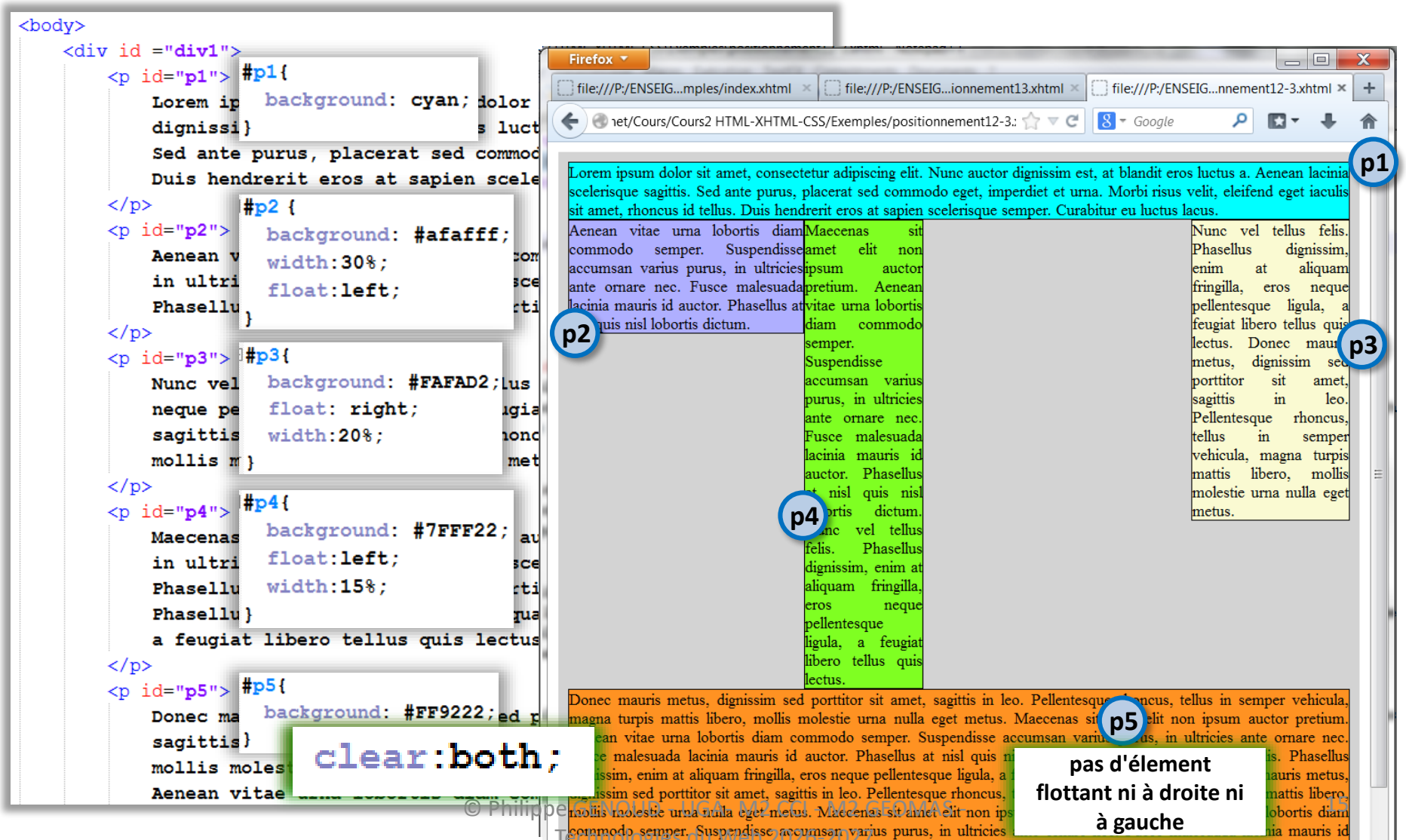
```
#p5{
    background: #FF9222;
}
```

```
clear:right;
```



Positionnement flottant

- **clear** permet d'interdire le voisinage avec un élément flottant (à droite, à gauche ou des deux côtés).



Positionnement flottant

- En utilisant les propriétés `float` et `clear` possibilité de définir des grilles de boites remplissant automatiquement la largeur de leur conteneur

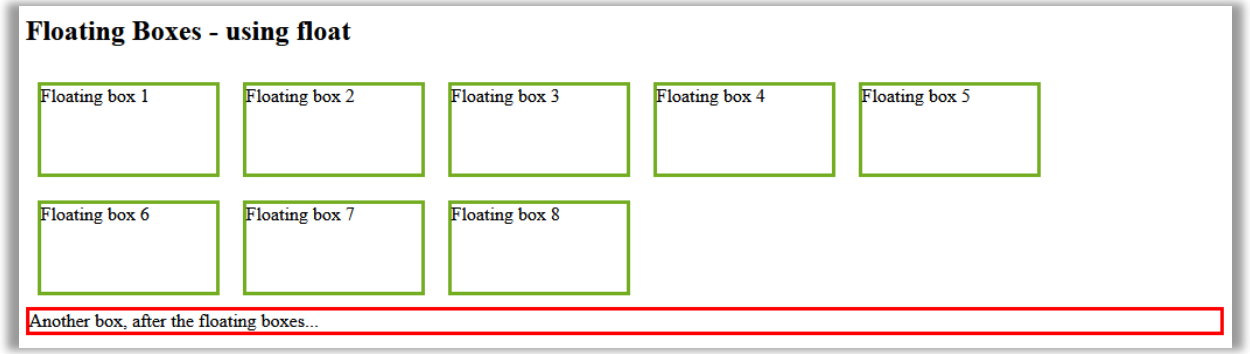
```
<h2>Floating Boxes - using float</h2>

<div class="floating-box">Floating box 1</div>
<div class="floating-box">Floating box 2</div>
<div class="floating-box">Floating box 3</div>
<div class="floating-box">Floating box 4</div>
<div class="floating-box">Floating box 5</div>
<div class="floating-box">Floating box 6</div>
<div class="floating-box">Floating box 7</div>
<div class="floating-box">Floating box 8</div>

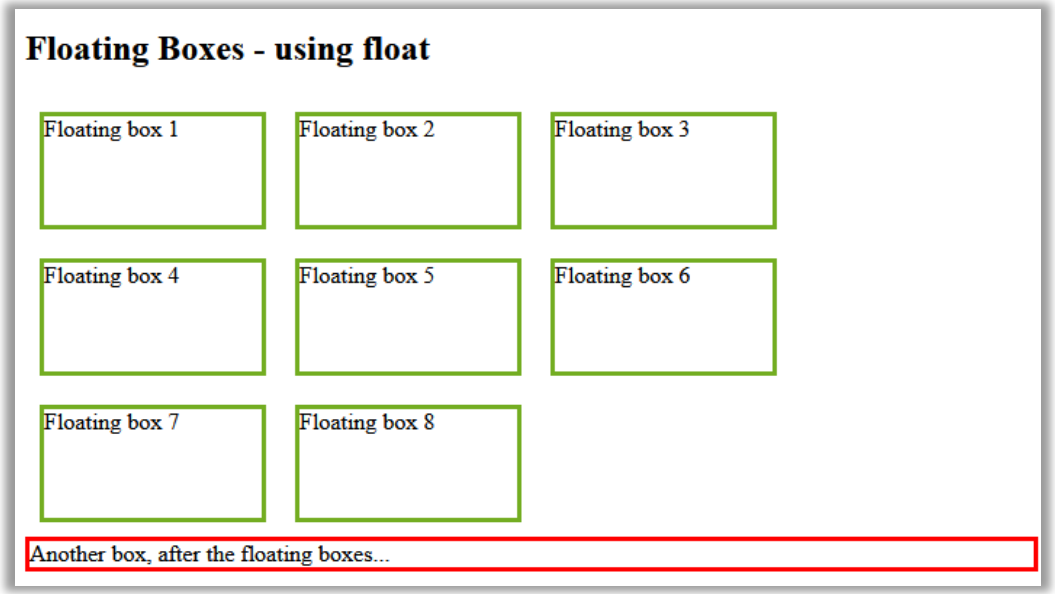
<div class="after-box">Another box, after the floating boxes...</div>
```

```
.floating-box {
  float: left;
  width: 150px;
  height: 75px;
  margin: 10px;
  border: 3px solid #73AD21;
}

.after-box {
  clear: left;
  border: 3px solid red;
}
```



https://www.w3schools.com/css/tryit.asp?filename=trycss_inline-block_old



Positionnement "inline-block"

- Possible d'obtenir facilement le même effet avec la propriété **`display:inline-block`**.
- Les éléments **`inline-block`** se comportent comme des éléments en ligne mais peuvent avoir une largeur et hauteur.

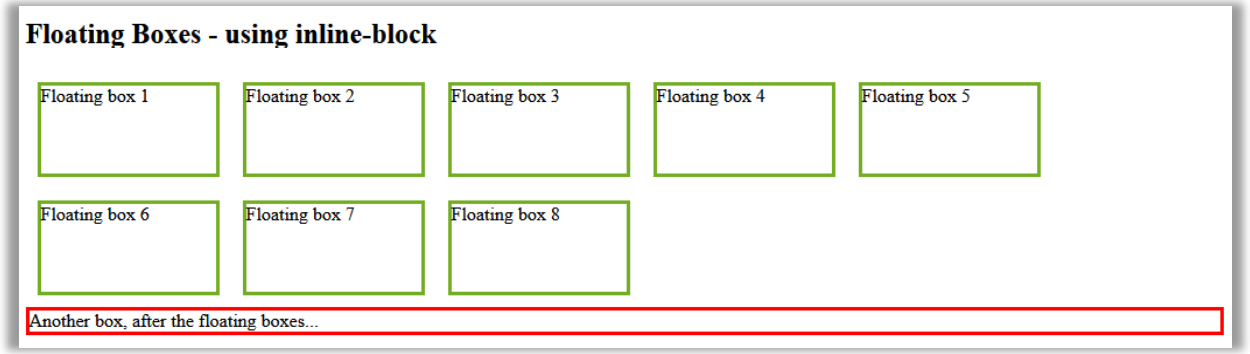
```
<h2>Floating Boxes - using float</h2>

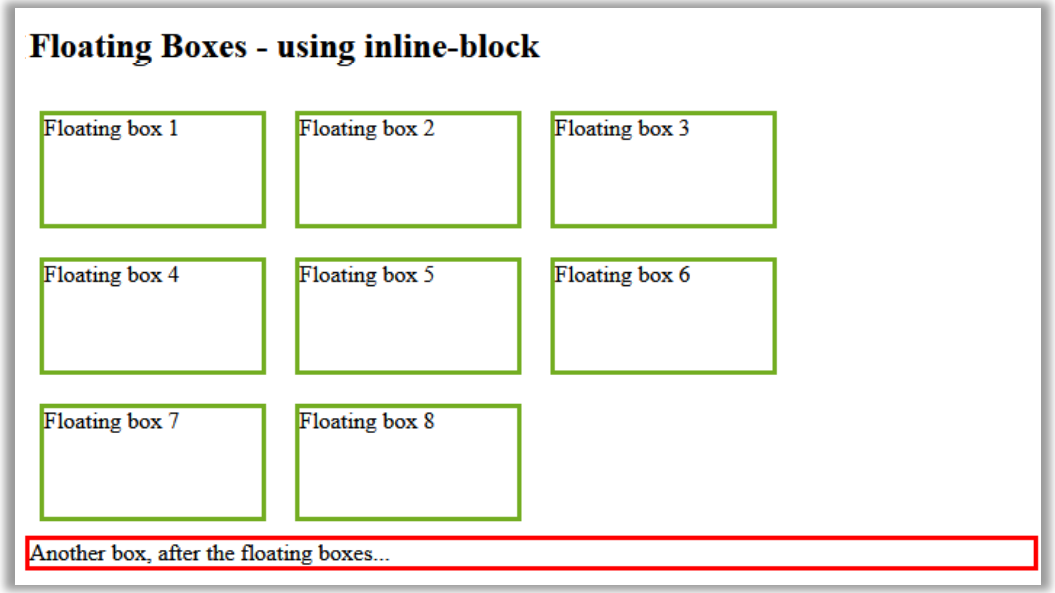
<div class="floating-box">Floating box 1</div>
<div class="floating-box">Floating box 2</div>
<div class="floating-box">Floating box 3</div>
<div class="floating-box">Floating box 4</div>
<div class="floating-box">Floating box 5</div>
<div class="floating-box">Floating box 6</div>
<div class="floating-box">Floating box 7</div>
<div class="floating-box">Floating box 8</div>

<div class="after-box">Another box, after the floating boxes...</div>
```

```
.floating-box {
  float: left;
  display: inline-block;
  width: 150px;
  height: 75px;
  margin: 10px;
  border: 3px solid #73AD21;
}

.after-box {
  clear: left;
  border: 3px solid red;
}
```





https://www.w3schools.com/css/tryit.asp?filename=trycss_inline-block

Positionnement "inline-block"

```css
body {
    font-family:Arial, Helvetica, sans-serif;
}

article {
    background-color: rgb(236, 206, 169);
    padding-left: 10px;
    padding-right: 10px;
    text-align: justify;

}

p {
    text-align: justify;
}

figure {
    text-align: center;
}

img {
    max-width: 100%;
    height: auto;
}
```

```html
<body>
    <div class="container">
        <h1>Tempore excepturi qui molestias.</h1>
        <p>Lorem ipsum ..</p>
        <article>
            <h2>Les Ours</h2>
            <p>
                Lorem ipsum ... magni?
            </p>
            <figure>
                <img src="images/ours.png">
                <figcaption>Un ours en Alaska</figcaption>
            </figure>
        </article>
        <article>
            <h2>Les Orques</h2>
            <p>
                Lorem ...
            </p>
            <figure>
                <img src="images/orque.png">
                <figcaption>Un orque à Haida Gwaii</figcaption>
            </figure>
        </article>
        <article>
            <h2>Les Caribous</h2>
            ...
        </article>
        ...
        <p>
            Lorem ipsum ...
        </p>
    </div>
</body>
```

positionnement en flux

Tempore excepturi qui molestias.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Recusandae deleniti odit est suscipit esse, eos molestias, harum inventore corrupti non laborum temporibus quis fugit rerum minima, ea quia ut hic incidunt? Optio qui facilis a! Rerum pariatur, laudantium magnam esse molestias ex!

Les Ours

Lorem ipsum dolor sit amet consectetur adipisicing elit. Ut facere quae eos fugiat sunt? Hic, magni?



Un ours en Alaska

Les Orques

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Quaerat est sunt voluptas necessitatibus. Aliquam enim architecto quisquam accusantium commodi vitae!



Positionnement "inline-block"

```css
body {
    font-family:Arial, Helvetica, sans-serif;
}

p {
    text-align:justify;
}

figure {
    text-align: center;
}

article {
    display: inline-block;
    background-color: rgb(236, 206, 169);
    padding-left: 10px;
    padding-right: 10px;
    width: 200px;
}

img {
    max-width: 100%;
    height: auto;
}
```

Tempore excepturi qui molestias.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Recusandae deleniti odit est suscipit esse, eos molestias, harum inventore corrupti non laborum temporibus quis fugit rerum minima, ea quia ut hic incidunt? Optio qui facilis a! Rerum pariatur, laudantium magnam esse molestias ex!

220px

20px padding
+ 200px width

Les Ours

Lorem ipsum dolor sit amet consectetur adipisicing elit. Ut facere quae eos fugiat sunt? Hic, magni?



Un ours en Alaska

Les Orques

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Quaerat est sunt voluptas necessitatibus. Aliquam enim architecto quisquam accusantium commodi vitae!



Un orque à Haida Gwaii

Les Caribous

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quibusdam in obcaecati provident placeat numquam eum rerum alias magni, voluptates quam minima explicabo qui eaque vitae aspernatur, fugit deleniti. Ex, nemo non. Tenetur vel hic non deleniti atque adipisci velit dolorem magni cumque tempora.



Des caribous au Yukon

Les lions de mer

Lorem ipsum obcaecati provident placeat magni, voluptates quam minima qui eaque vitae aspernatur, fugit deleniti. Ex, nemo non. Tenetur vel hic non deleniti atque adipisci velit dolorem magni cumque tempora.



Lions de mer à Rose Harbour

Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime similique modi eum velit sed rerum voluptatem debitis hic at nobis temporibus esse illum culpa consequuntur animi explicabo delectus numquam veniam consequatur vero necessitatibus laboriosam, adipisci harum eius. Ex ut in nam aliquam minima id autem.

Positionnement "inline-block"

```css
body {
    font-family:Arial, Helvetica, sans-serif
}

p {
    text-align:justify;
}

figure {
    text-align: center;
}

article {
    display: inline-block;
    background-color: rgb(236, 206, 169);
    padding-left: 10px;
    padding-right: 10px;
    width: 200px;
    vertical-align: top;
}

img {
    max-width: 100%;
    height: auto;
}
```

Tempore excepturi qui molestias.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Recusandae deleniti odit est suscipit esse, eos molestias, harum inventore corrupti non laborum temporibus quis fugit rerum minima, ea quia ut hic incidunt? Optio qui facilis a! Rerum pariatur, laudantium magnam esse molestias ex!

Les Ours

Lorem ipsum dolor sit amet consectetur adipisicing elit. Ut facere quae eos fugiat sunt? Hic, magni?



Un ours en Alaska

Les Orques

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Quaerat est sunt voluptas necessitatibus. Aliquam enim architecto quisquam accusantium commodi vitae!



Un orque à Haida Gwaii

Les Caribous

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quibusdam in obcaecati provident placeat numquam eum rerum alias magni, voluptates quam minima explicabo qui eaque vitae aspernatur, fugit deleniti. Ex, nemo non. Tenetur vel hic non deleniti atque adipisci velit dolorem magni cumque tempora.



Des caribous au Yukon

Les lions de mer

Lorem ipsum obcaecati provident placeat magni, voluptates quam minima qui eaque vitae aspernatur, fugit deleniti. Ex, nemo non. Tenetur vel hic non deleniti atque adipisci velit dolorem magni cumque tempora.



Lions de mer à Rose Harbour

Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime similique modi eum velit sed rerum voluptatem debitis hic at nobis temporibus esse illum culpa consequuntur animi explicabo delectus numquam veniam consequatur vero necessitatibus laboriosam, adipisci harum eius. Ex ut in nam aliquam minima id autem.

Positionnement "inline-block"

- problème des espaces

```css
body {
    font-family:Arial, Helvetica, sans-serif;
}

p {
    text-align:justify;
}

figure {
    text-align: center;
}

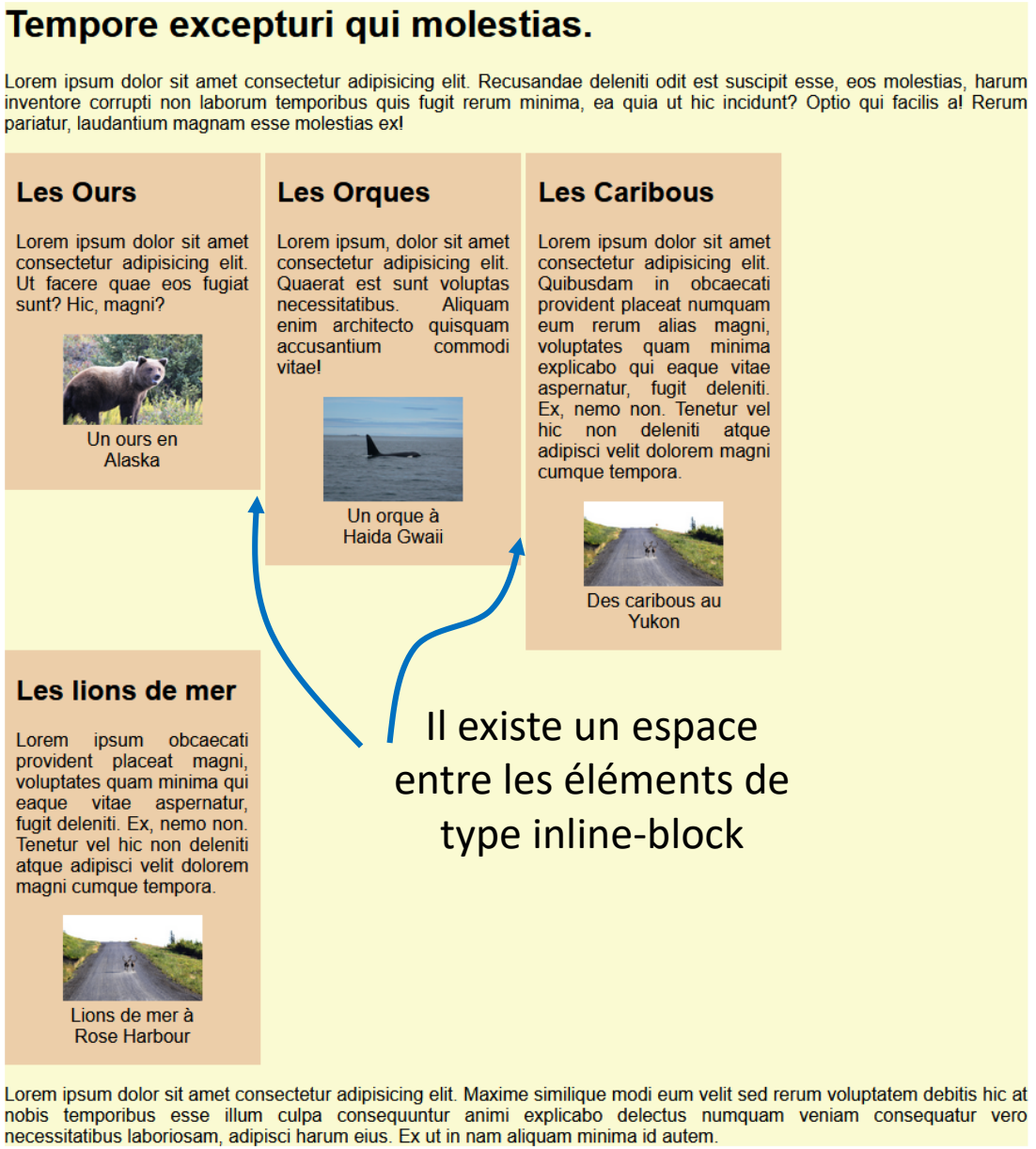
article {
    display: inline-block;
    background-color: rgb(236, 206, 169);
    padding-left: 10px;
    padding-right: 10px;
    width: 200px;
    vertical-align: top;
}

img {
    max-width: 100%;
    height: auto;
}
```

```css
.container {
    width:880px;
    margin: auto;
    background-color: lightgoldenrodyellow;
}
```

solution , voir article :

http://www.alsacreations.com/astuce/lire/1432-display-inline-block-espacesindesirables.html



Positionnement des éléments
positionnement flottant

- possibilité de faire flotter des éléments en ligne.

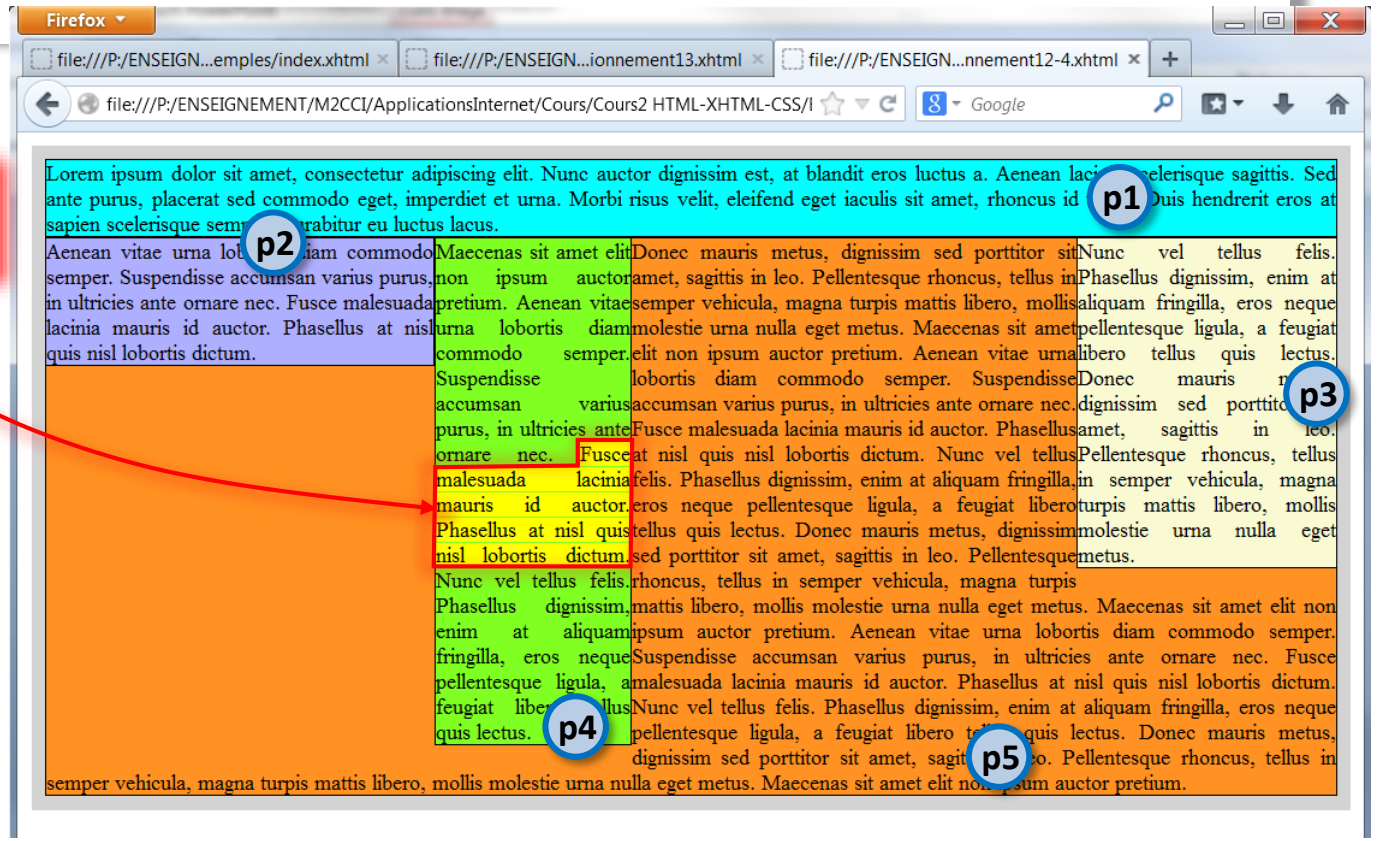
```
<p id="p4">
    Maecenas sit amet elit non ipsum auctor pretium. Aenean vitae urna lobortis diam commodo semper. Sus
    in ultricies ante ornare nec. <span id="span2">Fusce malesuada lacinia mauris id auctor.
    Phasellus at nisl quis nisl lobortis dictum.</span> Nunc vel tellus felis.
    Phasellus dignissim, enim at aliquam fringilla, eros neque pellentesque ligula,
    a feugiat libero tellus quis lectus.
</p>
```

```
#span2 {
    background: yellow;
}
```

+

```
font-size: 50%;
float: left;
width: 20%;
margin: 5px;
padding: 5px;
```

?



Positionnement des éléments
positionnement flottant

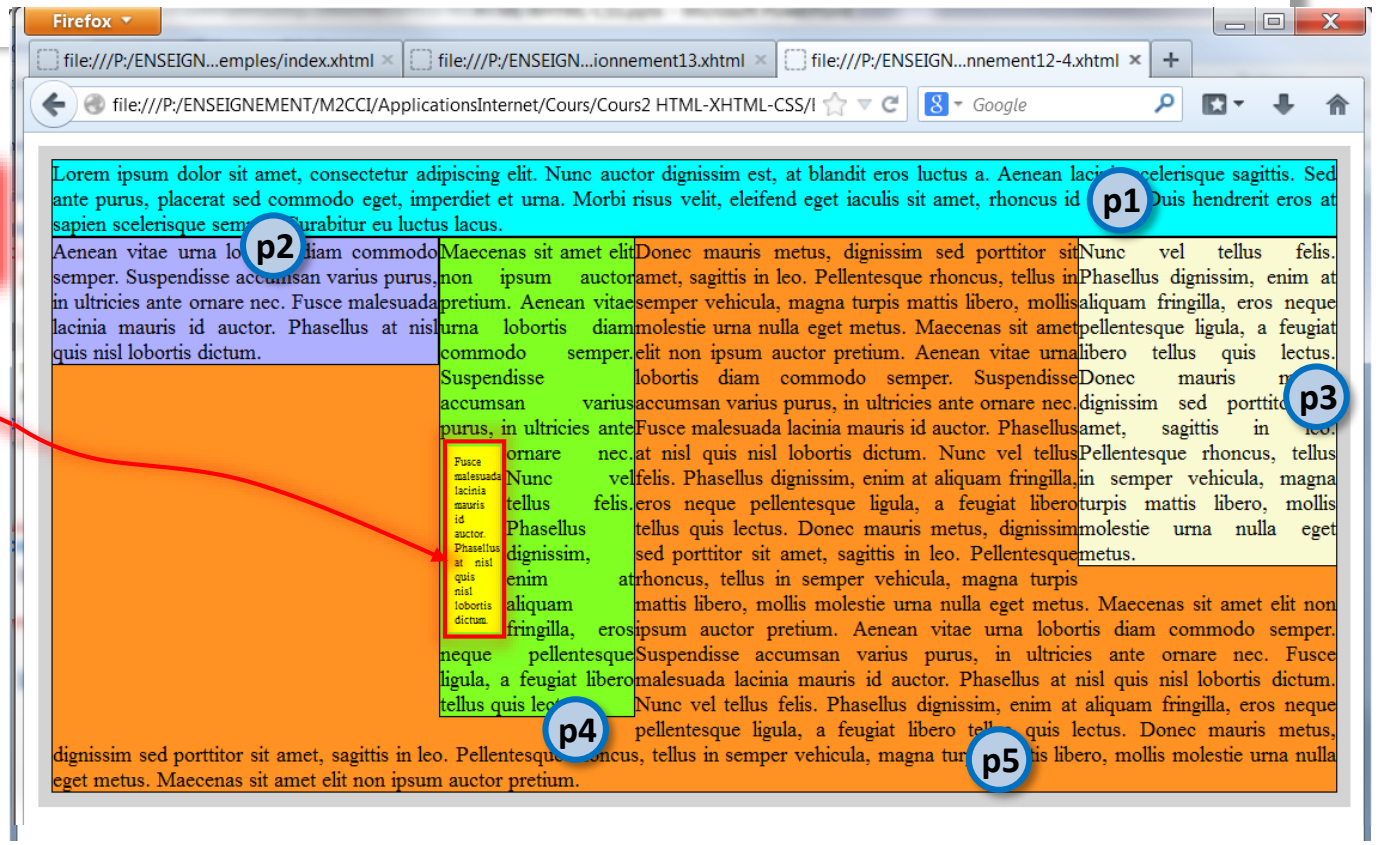
- possibilité de faire flotter des éléments en ligne.

```
<p id="p4">
    Maecenas sit amet elit non ipsum auctor pretium. Aenean vitae urna lobortis diam commodo semper. Sus
    in ultricies ante ornare nec. <span id="span2">Fusce malesuada lacinia mauris id auctor.
    Phasellus at nisl quis nisl lobortis dictum.</span> Nunc vel tellus felis.
    Phasellus dignissim, enim at aliquam fringilla, eros neque pellentesque ligula,
    a feugiat libero tellus quis lectus.
</p>
```

```
#span2 {
    background: yellow;
}
```

+

```
font-size: 50%;
float: left;
width: 20%;
margin: 5px;
padding: 5px;
```



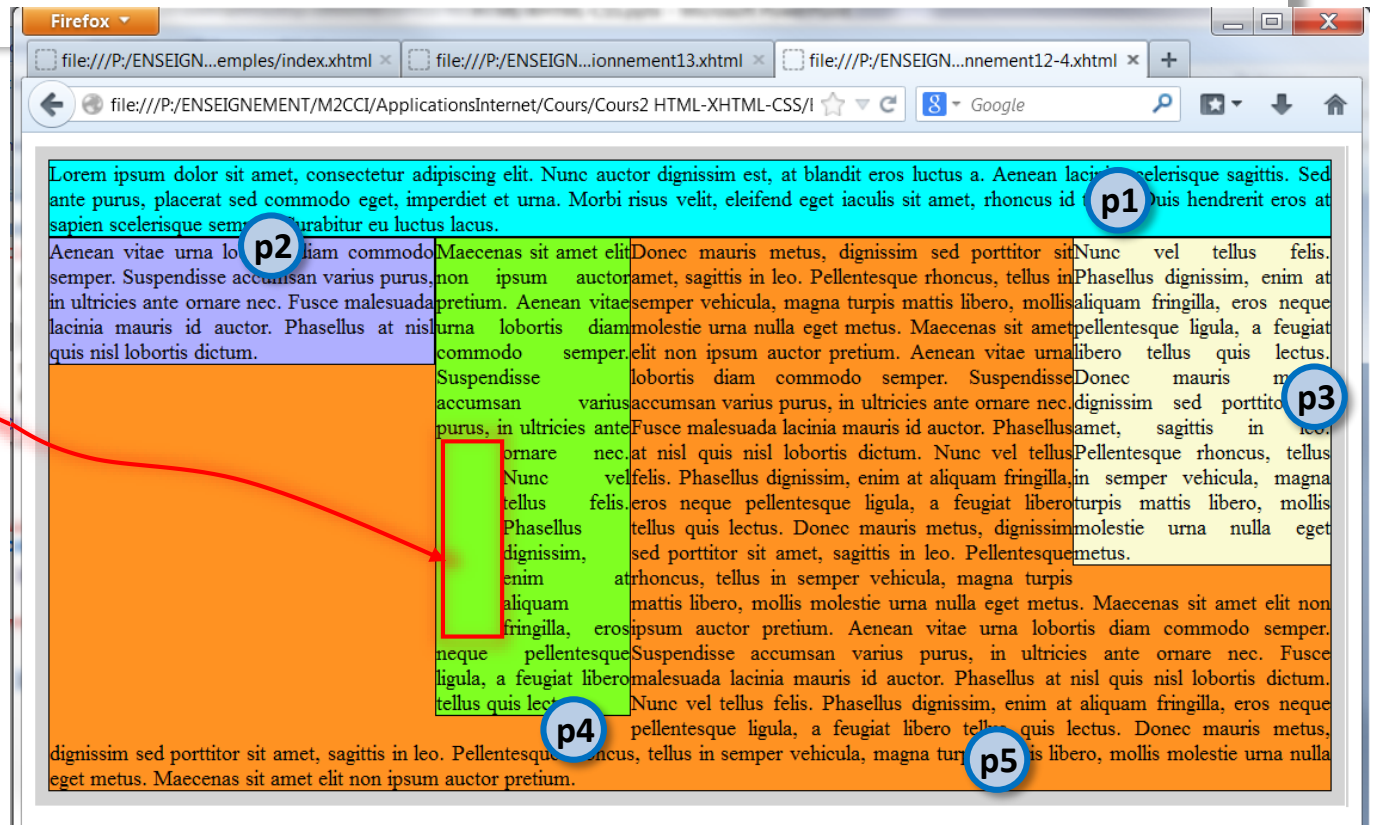
• tout élément flottant est considéré de type bloc
• les éléments en ligne peuvent alors recevoir bordures et dimensions

Positionnement des éléments
Afficher et masquer des éléments

- possibilité de rendre invisible des éléments

```
<p id="p4">
    Maecenas sit amet elit non ipsum auctor pretium. Aenean vitae urna lobortis diam commodo semper. Sus
    in ultricies ante ornare nec. <span id="span2">Fusce malesuada lacinia mauris id auctor.
    Phasellus at nisl quis nisl lobortis dictum.</span> Nunc vel tellus felis.
    Phasellus dignissim, enim at aliquam fringilla, eros neque pellentesque ligula,
    a feugiat libero tellus quis lectus.
</p>
```

```
#span2 {
    background: yellow;
    font-size: 50%;
    float: left;
    width: 20%;
    margin: 5px;
    padding: 5px;
    visibility:hidden;
}
```



• **visibility:hidden** masque l'élément mais réserve sa position et ses dimensions. L'élément occupe de l'espace sur la page.

• autres valeurs : **visible**, **collapse** (pour tables), **inherit**

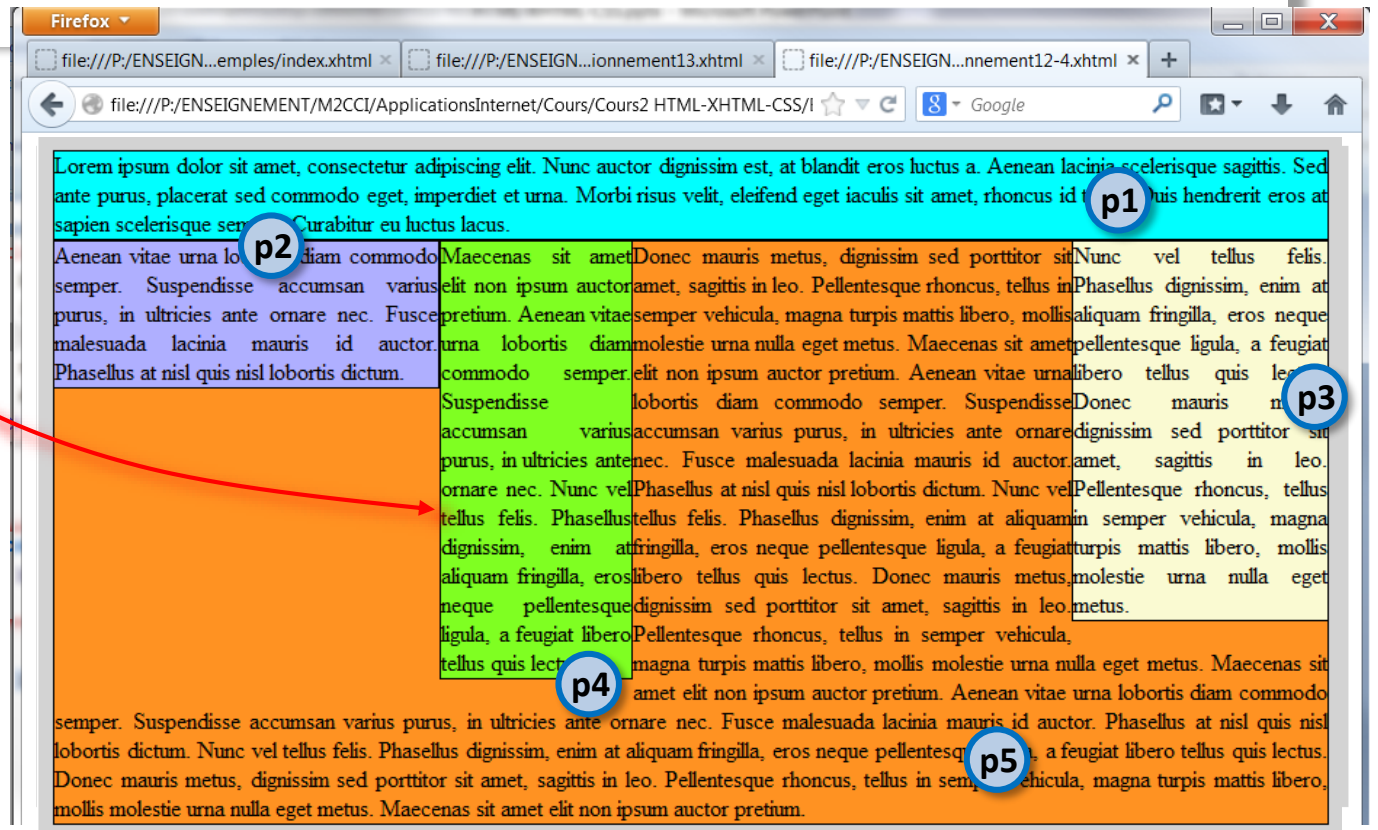
Positionnement des éléments
Afficher et masquer des éléments

- possibilité de retirer un élément de l'affichage

```
<p id="p4">
    Maecenas sit amet elit non ipsum auctor pretium. Aenean vitae urna lobortis diam commodo semper. Sus
    in ultricies ante ornare nec. <span id="span2">Fusce malesuada lacinia mauris id auctor.
    Phasellus at nisl quis nisl lobortis dictum.</span> Nunc vel tellus felis.
    Phasellus dignissim, enim at aliquam fringilla, eros neque pellentesque ligula,
    a feugiat libero tellus quis lectus.
</p>
```

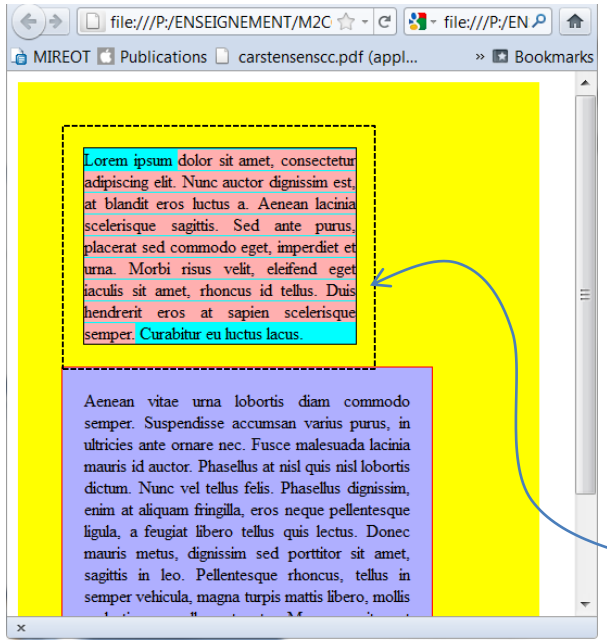
```
#span2 {
    background: yellow;
    font-size: 50%;
    float: left;
    width: 20%;
    margin: 5px;
    padding: 5px;
    display: none;
}
```

• **display:none** l'élément n'est plus du tout affiché. Tout se passe comme si il n'existait pas



Positionnement des éléments
positionnement flottant

- positionnement flottant
  - autre exemple.



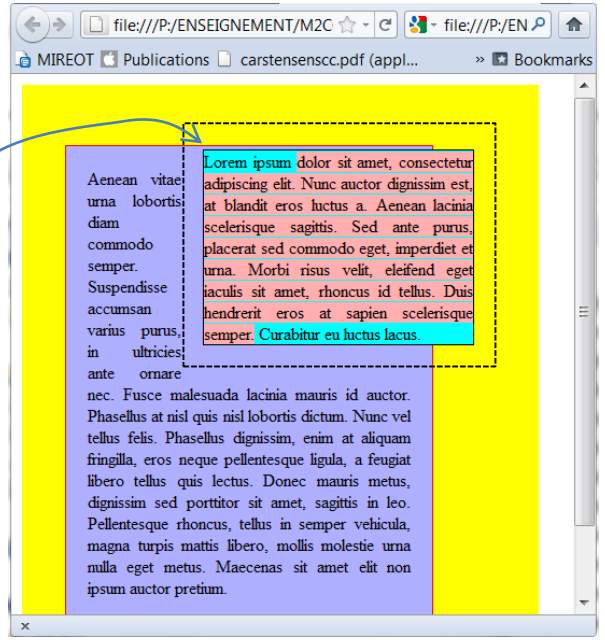
positionnement en flux

```
p {
  border-style: solid;
  border-width: thin;
  margin: 0px;
}

#div1 {
  background-color: yellow;
  width:400px;
  padding:40px;
  margin:10px;
}

#p1{
  background: cyan;
  width:250px;
  margin: 20px;
}
  float: right;

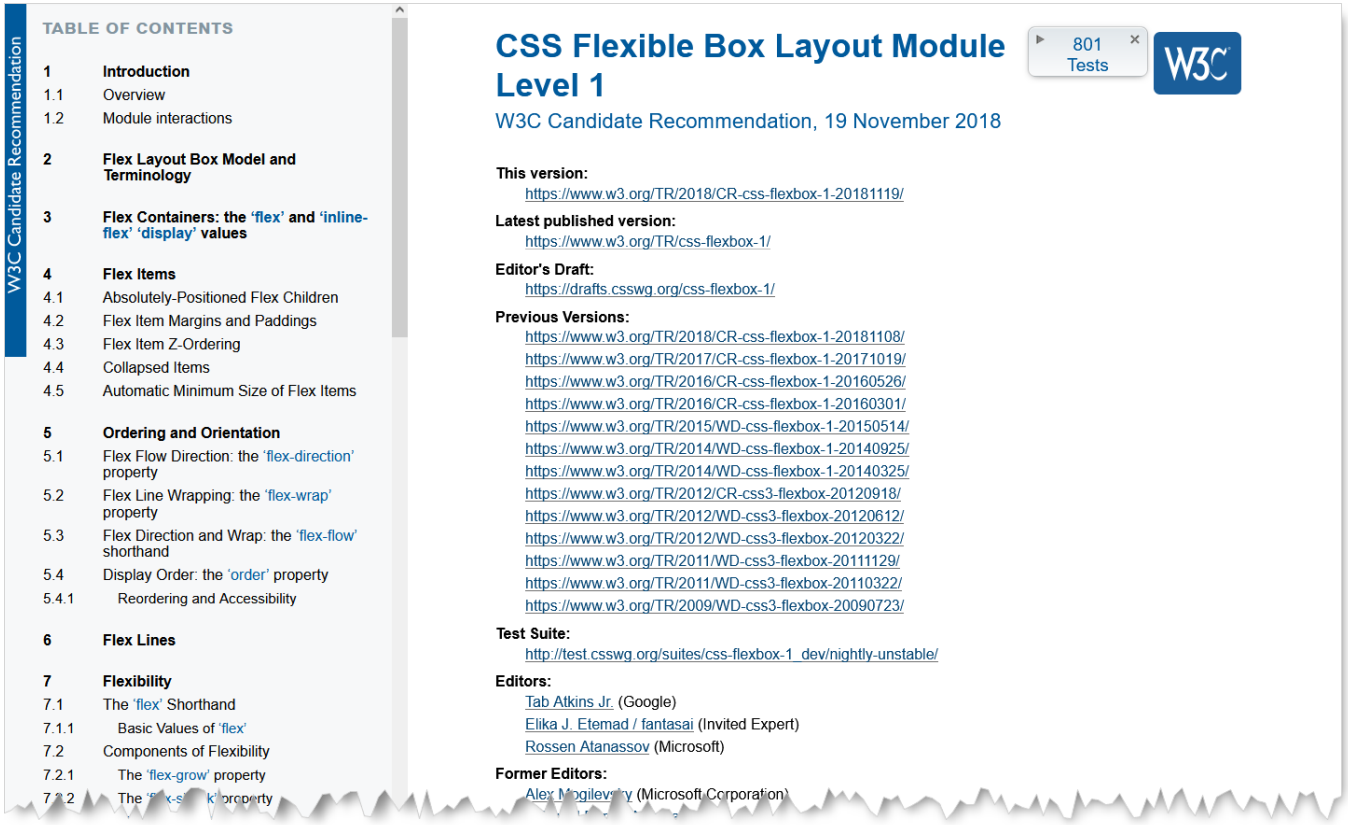
#p2 {
  background: #afafff;
  border-color: red;
  padding:20px;
  width:300px;
}
```



positionnement en flottant

Positionnement 'Flexbox'

- positionnement flottant complexe à maîtriser
- Boites flexibles (flex boxes ou flexs) un nouveau mode de positionnement (layout) des éléments introduits avec CSS3.
  - amélioration par rapport au modèle de bloc
  - comportement plus prévisible pour prise en compte de tailles différentes d'affichage.



W3C Candidate Recommendation

TABLE OF CONTENTS

1 Introduction
1.1 Overview
1.2 Module interactions

2 Flex Layout Box Model and Terminology

3 Flex Containers: the 'flex' and 'inline-flex' 'display' values

4 Flex Items
4.1 Absolutely-Positioned Flex Children
4.2 Flex Item Margins and Paddings
4.3 Flex Item Z-Ordering
4.4 Collapsed Items
4.5 Automatic Minimum Size of Flex Items

5 Ordering and Orientation
5.1 Flex Flow Direction: the 'flex-direction' property
5.2 Flex Line Wrapping: the 'flex-wrap' property
5.3 Flex Direction and Wrap: the 'flex-flow' shorthand
5.4 Display Order: the 'order' property
5.4.1 Reordering and Accessibility

6 Flex Lines

7 Flexibility
7.1 The 'flex' Shorthand
7.1.1 Basic Values of 'flex'
7.2 Components of Flexibility
7.2.1 The 'flex-grow' property
7.2.2 The 'flex-shrink' property

CSS Flexible Box Layout Module Level 1

801 Tests

W3C Candidate Recommendation, 19 November 2018

This version:
https://www.w3.org/TR/2018/CR-css-flexbox-1-20181119/

Latest published version:
https://www.w3.org/TR/css-flexbox-1/

Editor's Draft:
https://drafts.csswg.org/css-flexbox-1/

Previous Versions:
https://www.w3.org/TR/2018/CR-css-flexbox-1-20181108/
https://www.w3.org/TR/2017/CR-css-flexbox-1-20171019/
https://www.w3.org/TR/2016/CR-css-flexbox-1-20160526/
https://www.w3.org/TR/2016/CR-css-flexbox-1-20160301/
https://www.w3.org/TR/2015/WD-css-flexbox-1-20150514/
https://www.w3.org/TR/2014/WD-css-flexbox-1-20140925/
https://www.w3.org/TR/2014/WD-css-flexbox-1-20140325/
https://www.w3.org/TR/2012/CR-css3-flexbox-20120918/
https://www.w3.org/TR/2012/WD-css3-flexbox-20120612/
https://www.w3.org/TR/2012/WD-css3-flexbox-20120322/
https://www.w3.org/TR/2011/WD-css3-flexbox-20111129/
https://www.w3.org/TR/2011/WD-css3-flexbox-20110322/
https://www.w3.org/TR/2009/WD-css3-flexbox-20090723/

Test Suite:
http://test.csswg.org/suites/css-flexbox-1_dev/nightly-unstable/

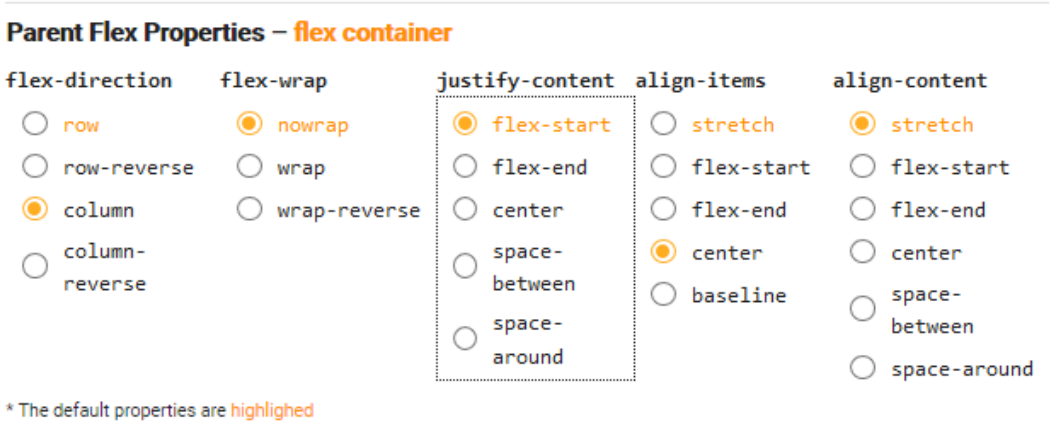
Editors:
Tab Atkins Jr. (Google)
Elika J. Etemad / fantasai (Invited Expert)
Rossen Atanassov (Microsoft)

Former Editors:
Alex Mogilevsky (Microsoft Corporation)

https://www.w3.org/TR/css-flexbox-1/
une norme récente mais largement anticipée par les navigateurs modernes

Positionnement 'Flexbox'

- Modèle de boîte flexible – flex box
- 4 caractéristiques principales
  - **Distribution** des éléments horizontale ou verticale, avec passage à la ligne autorisé ou non,
  - **Alignements** et centrages horizontaux et verticaux, justifiés, répartis,
  - **Réorganisation** des éléments indépendamment de l'ordre du flux (DOM),
  - **Gestion des espaces disponibles** (fluidité).

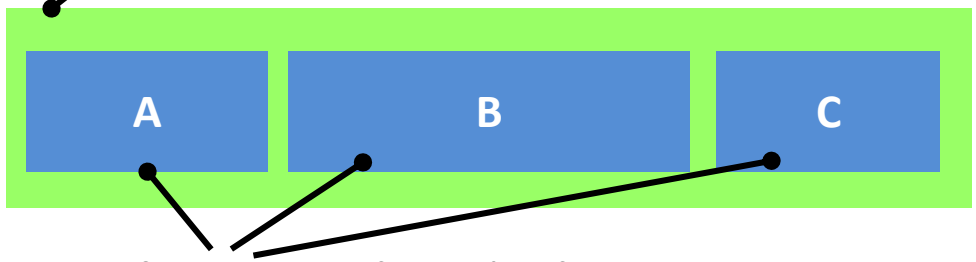




https://codepen.io/justd/full/yydezN/

Positionnement 'Flexbox'

flex container : n'importe quel élément HTML doté de la déclaration `display: flex;` ou `display: inline-flex;`



flex items: enfants d'un flex container.

le type "flex-item" est implicite, inutile de déclarer quoi que ce soit

Eléments "flex-item" ne sont plus considérés comme des éléments classiques ("bloc" ou "inline") mais sont par défaut distribués horizontalement dans leur conteneur flex.

➔ les propriétés de positionnement "classiques" (`display`*, `float` …) **sont** sans effet sur les éléments flex-items.

*mis à part `diplay:none;`

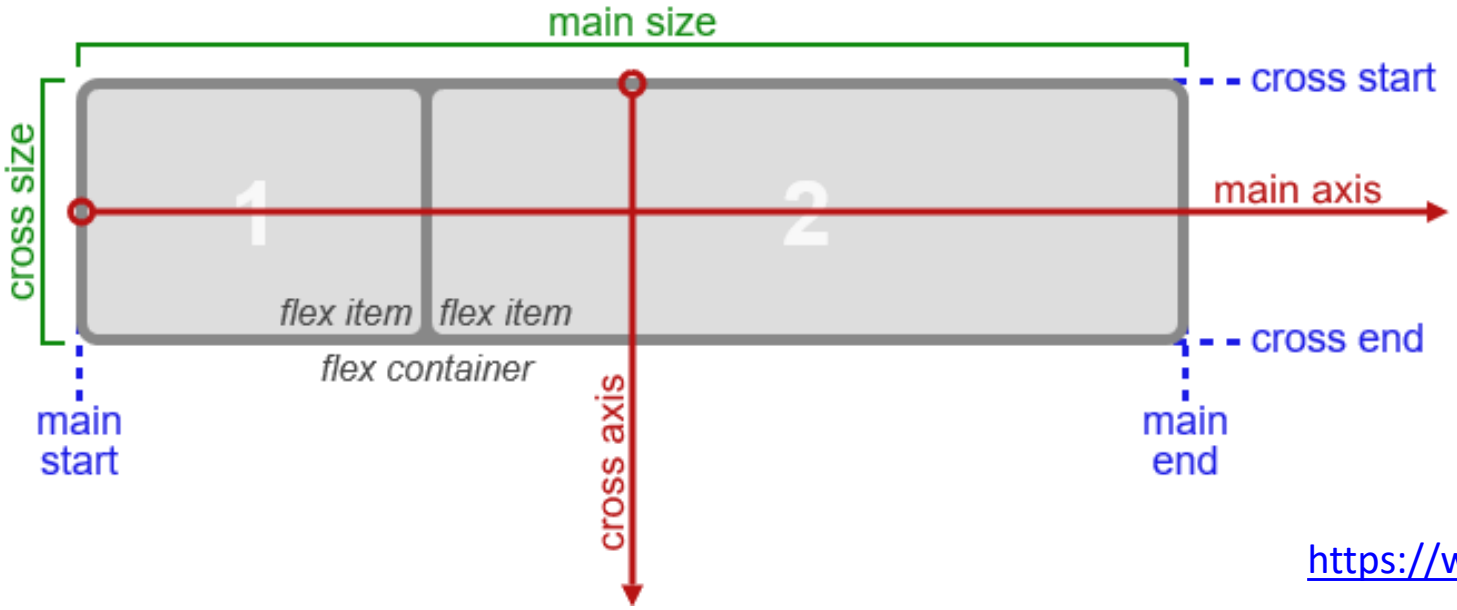
HTML

```html
<div  class="flex-container">
    <div id="div-1">A</div>
    <div id="div-2">B</div>
    <div id="div-3">C</div>
</div>
```

CSS

```css
.flex-container {
    display: flex;
    background-color: rgb(153,255,102);
}
.flex-container > div {
    background-color: rgb(85,142,213);
    box-sizing:border-box;
    border: solid red 2px;
    padding: 15px;
    margin: 10px;
    text-align: center;
    color: white;
    font-size: 30px;
}
#div-1, #div-3 {
    width: 25%;
}
#div-2 {
    width: 50%;
}
```

Positionnement 'Flexbox'



https://www.w3.org/TR/css-flexbox-1/

main axis
main dimension
The ***main axis*** of a flex container is the primary axis along which flex items are laid out. It extends in the ***main dimension***.

main-start
main-end
The flex items are placed within the container starting on the ***main-start*** side and going toward the ***main-end*** side.

main size
main size property
The width or height of a flex container or flex item, whichever is in the main dimension, is that box's ***main size***. Its ***main size property*** is thus either its 'width' or 'height' property, whichever is in the main dimension. Similarly, its ***min*** and ***max main size properties*** are its 'min-width'/'max-width' or 'min-height'/'max-height' properties, whichever is in the main dimension, and determine its ***min***/***max main size***.

cross axis
cross dimension
The axis perpendicular to the main axis is called the ***cross axis***. It extends in the ***cross dimension***.

cross-start
cross-end
Flex lines are filled with items and placed into the container starting on the ***cross-start*** side of the flex container and going toward the ***cross-end*** side.

cross size
cross size property
The width or height of a flex container or flex item, whichever is in the cross dimension, is that box's ***cross size***. Its ***cross size property*** is thus either its 'width' or 'height' property, whichever is in the cross dimension. Similarly, its ***min*** and ***max cross size properties*** are its 'min-width'/'max-width' or 'min-height'/'max-height' properties, whichever is in the cross dimension, and determine its ***min***/***max cross size***.

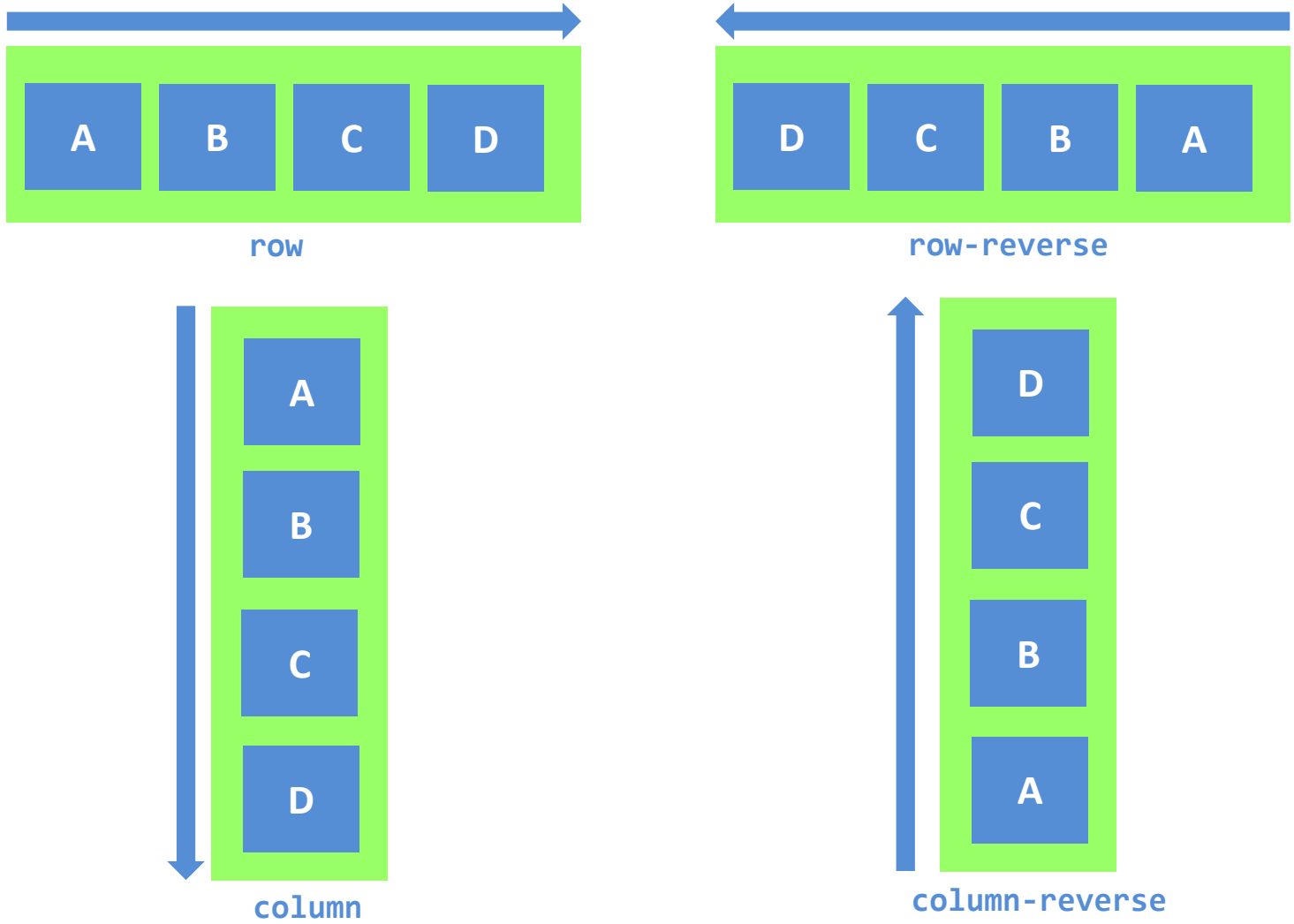
Positionnement flexbox

- Les propriétés d'un conteneur flex sont :
  - `flex-direction` : définit la direction de l'axe principal (main-axis) selon laquelle les flex-items sont distribués
  - `flex-wrap` :
  - `flex-flow` : raccourci pour spécifier simultanément les propriétés `flex-direction` et `flex-wrap`
  - `justify-content` : définit l'alignement selon l'axe principal
  - `align-items` : définit l'alignement selon l'axe secondaire (perpendiculaire à l'axe principal)
  - `align-content:`

Positionnement 'Flexbox'

`flex-direction:` (propriété de flex-container)

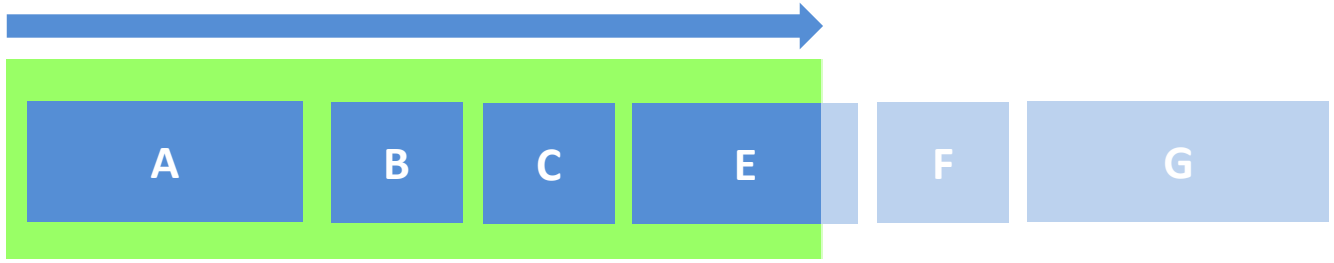
définit la direction (axe principal ou main-axis) selon laquelle les flex-items sont distribués



Positionnement 'Flexbox'

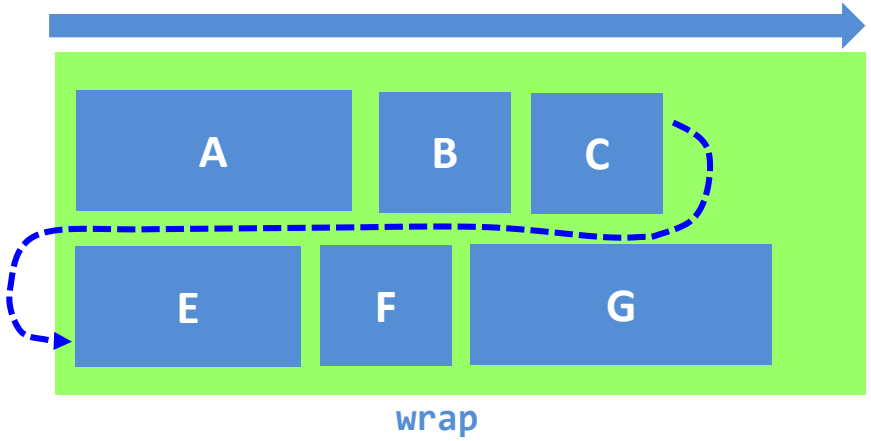
`flex-wrap` (propriété de flex-container)

définit si le contenu sera distribué sur une seule ligne (ou colonne selon l'axe principal) ou sur plusieurs lignes



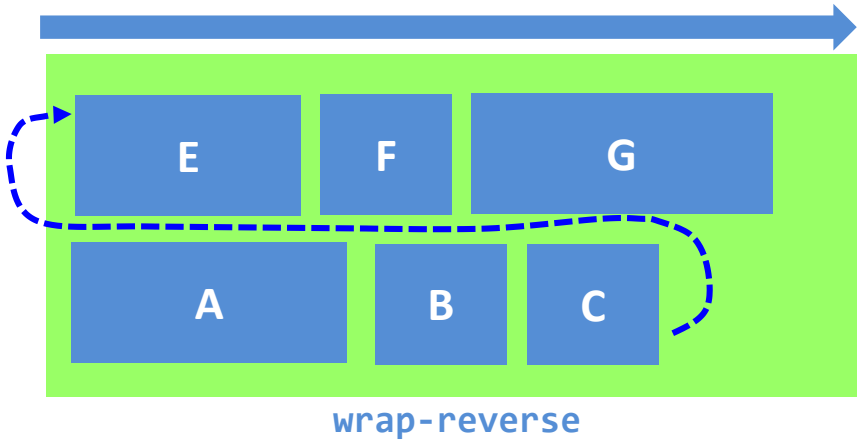
`nowrap`

valeur par défaut : les éléments ne passent pas à la ligne.



`wrap`

les éléments passent à la ligne dans le sens de lecture.



`wrap-reverse`

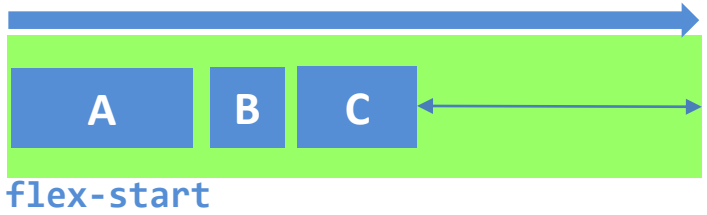
les éléments passent à la ligne dans le sens de lecture inverse.

Positionnement 'Flexbox'

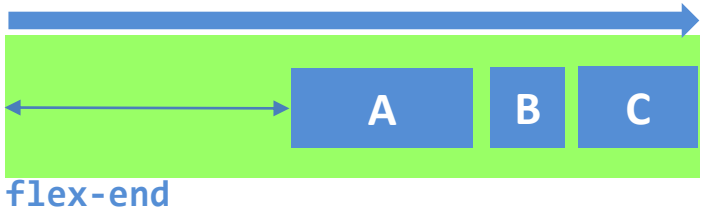
`justify-content` (propriété de flex-container)

définit l'alignement selon l'axe principal

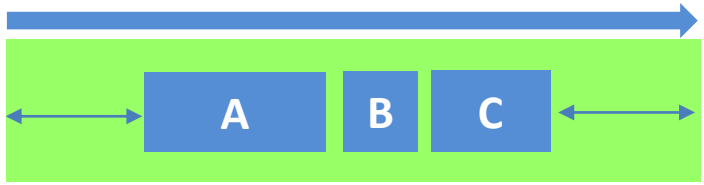
permet de distribuer l'espace restant libre supplémentaire lorsque les éléments d'une ligne sont soit rigides ou si ils sont flexibles ont atteint leur taille maximale. Permet d'exercer également un certain contrôle sur l'alignement des éléments lorsqu'ils débordent la ligne.



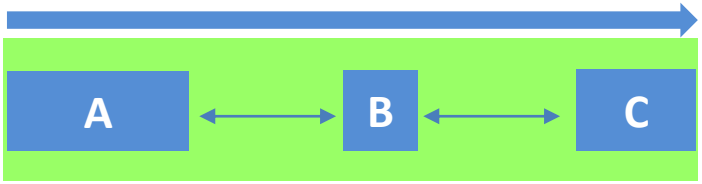
`flex-start`



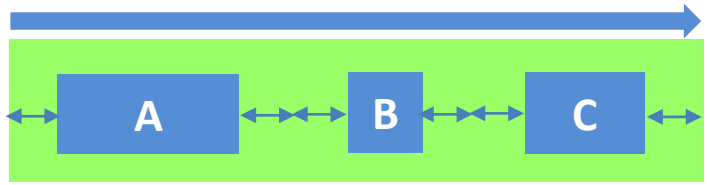
`flex-end`



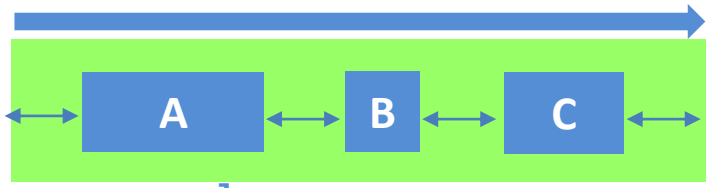
`center`



`space-between`



`space-around`

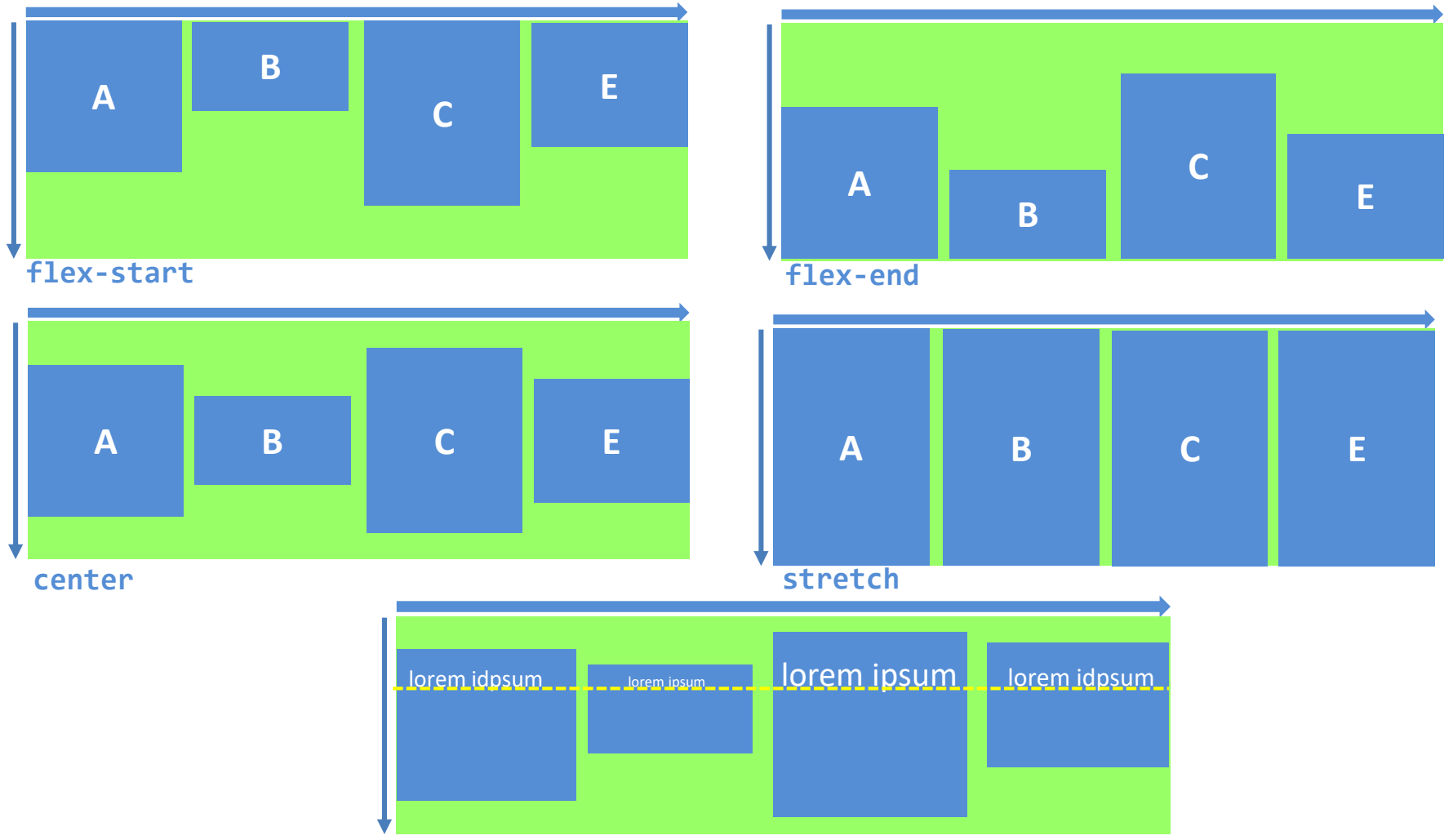


`space-evenly`

Positionnement 'Flexbox'

`align-items` (propriété de flex-container)

définit l'alignement selon l'axe secondaire (perpendiculaire à l'axe principal)

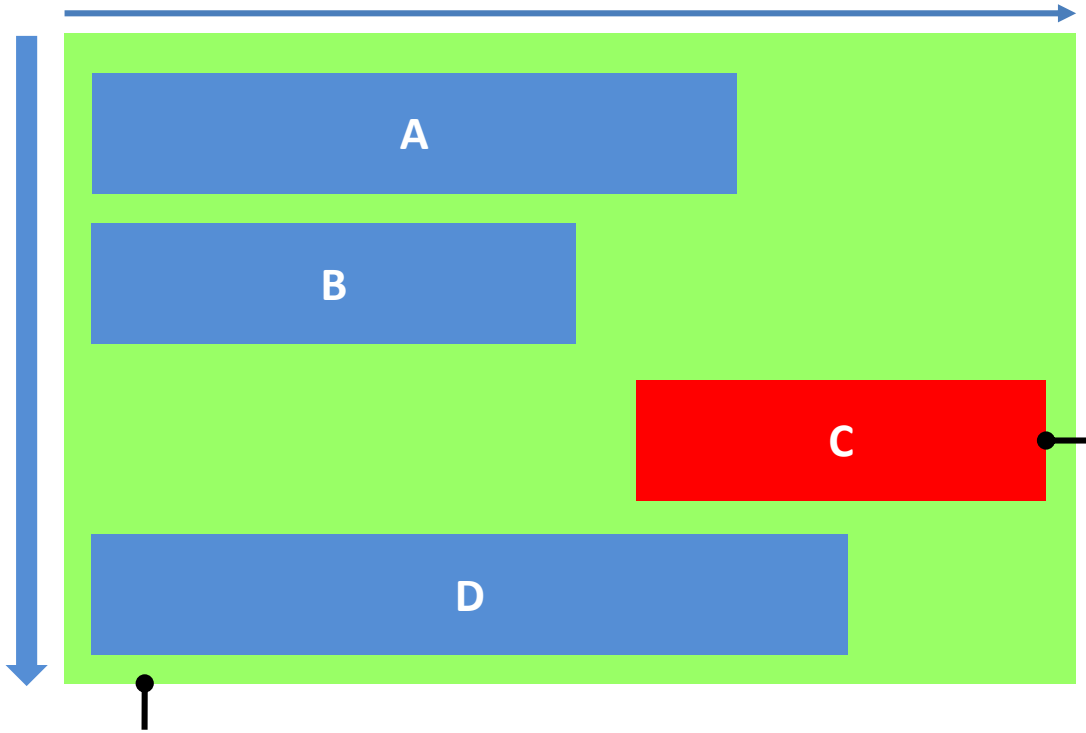


Positionnement 'Flexbox'

`align-self` (propriété de flex-item)

permet de définir un alignement spécifique pour un item particulier

valeurs de cette propriété identiques à celles de **`align-items`**



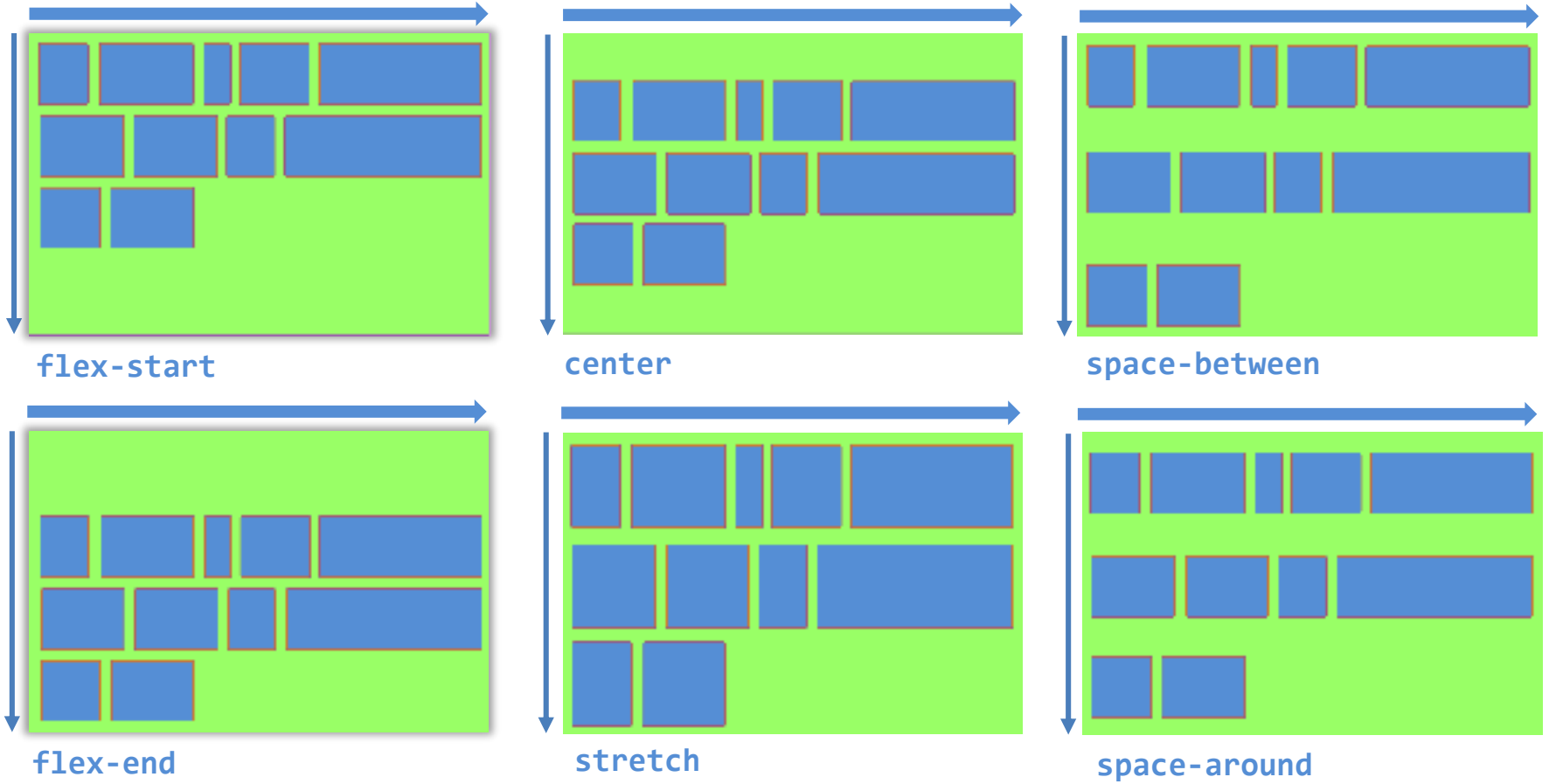

```
#elementC {
    align-self: flex-end;
}
```

```
#flexContainer {
    display: flex;
    flex-direction: column;
    align-items: flex-start;
}
```

Positionnement 'Flexbox'

`align-contents:` (propriété de flex-container)

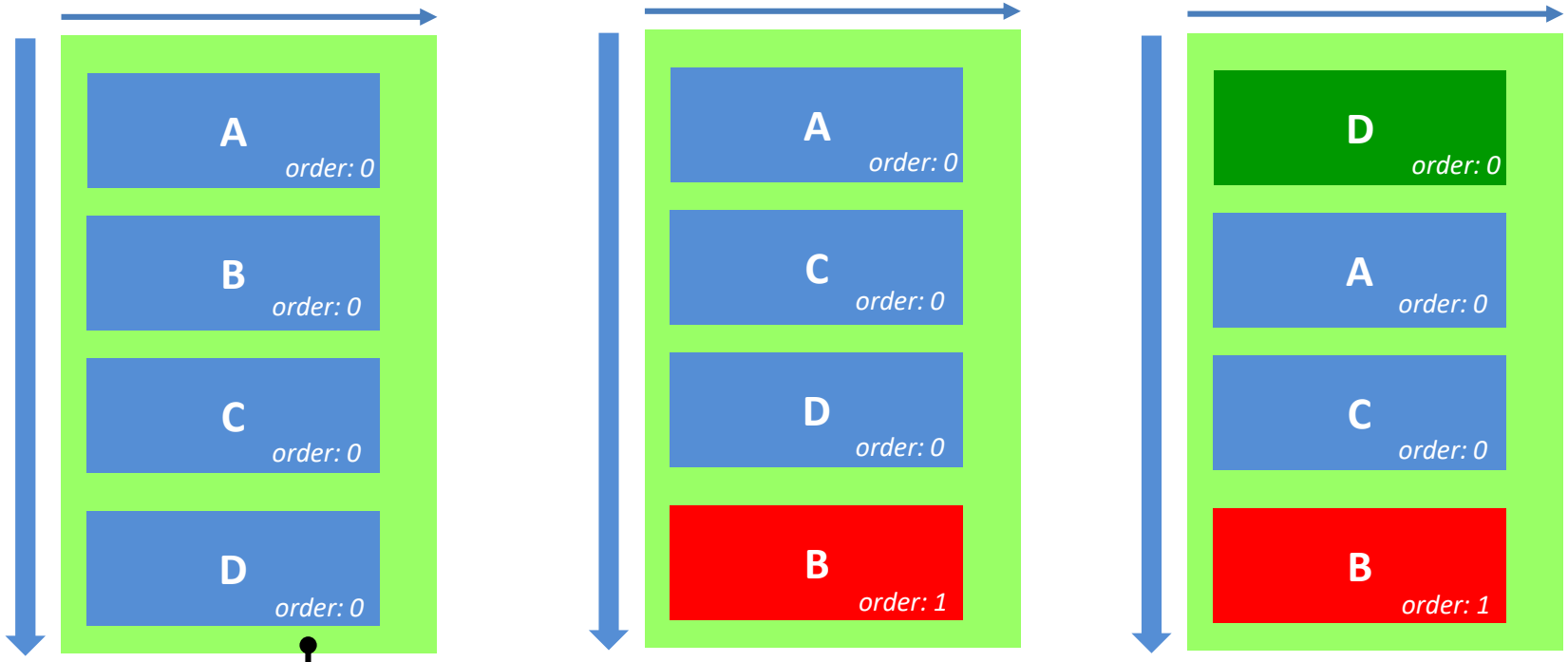
permet d'aligner les lignes d'un conteneur flex quand il reste de la place disponible selon l'axe secondaire (similaire à **`justify-content`** pour l'axe principal)



Positionnement 'Flexbox'

`order:` (propriété d'un flex item)

- Par défaut, les éléments flex items sont affichés selon leur ordre d'apparition dans le code source de la page. La propriété **`order`** offre la possibilité de contrôler cet ordre élément par élément.
- La valeur de order (0 par défaut) indique l'ordre dans lequel les éléments sont affichés (de plus petit au plus grand)




```
#flexContainer {
    display: flex;
    flex-direction: column;
    align-items: stretch;
}
```

```
#elementB {
    order: 1;
}
```

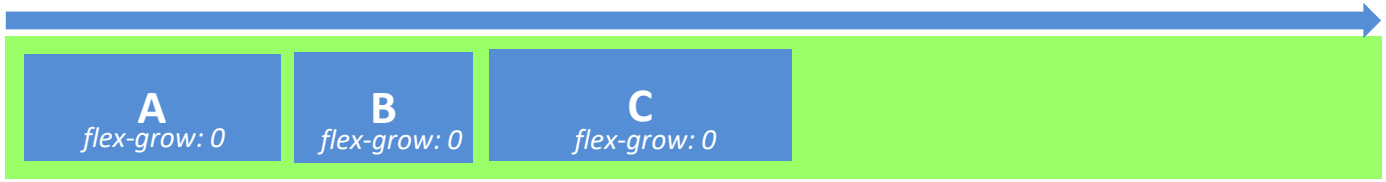
```
#elementB {
    order: 1;
}
#elementD {
    order: 1;
}
```

Positionnement 'Flexbox'

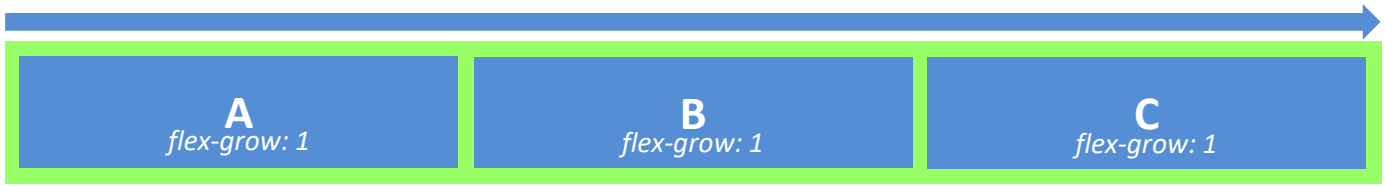
`flex-grow:` (propriété d'un flex item)

- définit la possibilité pour un flex-item de s'étirer ou non dans l'espace restant du conteneur flex.
- par défaut 0, pas d'étirement
- si valeur > 0, étirement (la proportion de l'espace restant attribué aux éléments étirables sera proportionnelle à la valeur de leur attribut flex-grow)

```
#flexContainer {
    display: flex;
}
```

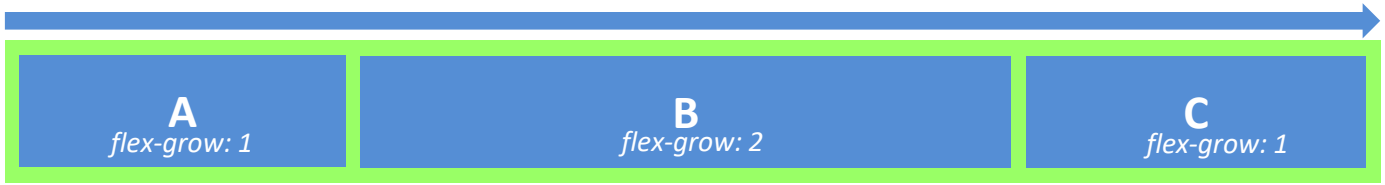



```
#flexContainer {
    display: flex;
}
#A, #B, #C {
    flex-grow: 1;
}
```



tous les éléments flex-item ont un flex-grow égal à 1, l'espace restant est réparti de manière égale entre tous les items

```
#flexContainer {
    display: flex;
}
#A, #B {
    flex-grow: 1;
}
#B {
    flex-grow: 2;
}
```



l'élément B a un flex-grow de 2, l'espace restant qui lui sera attribué sera (si possible) 2 fois plus grand que celui attribué aux flex-items ayant un valeur de flex-grow égale à 1.

Positionnement 'Flexbox'

`flex-shrink:` (propriété d'un flex item)

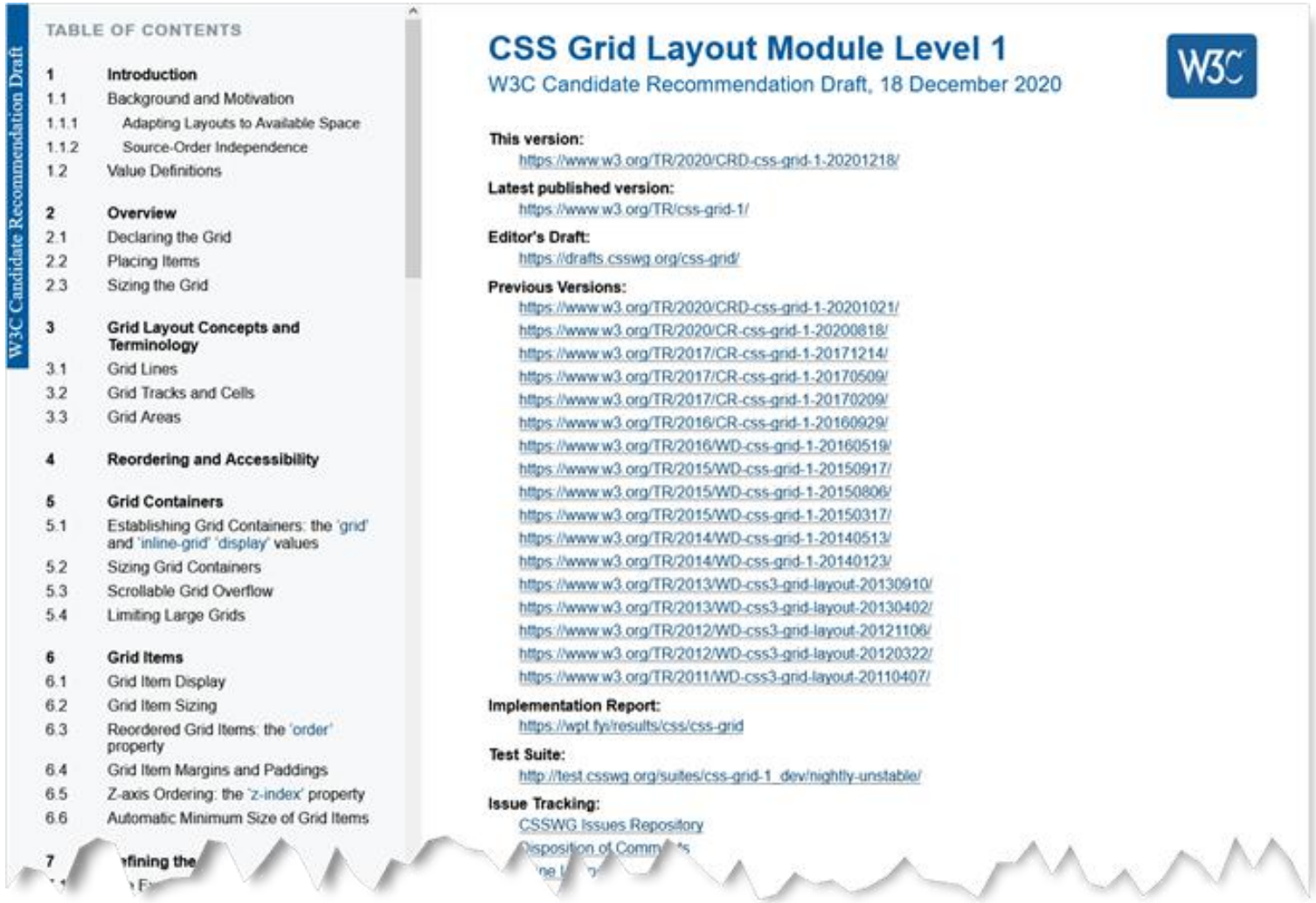
- définit la possibilité pour un flex-item de se contracter si nécessaire.
- par défaut 1, pas d'étirement

`flex-shrink:` (propriété d'un flex item)

- définit la possibilité pour un flex-item de se contracter si nécessaire.
- par défaut 1, pas d'étirement



CSS Grid



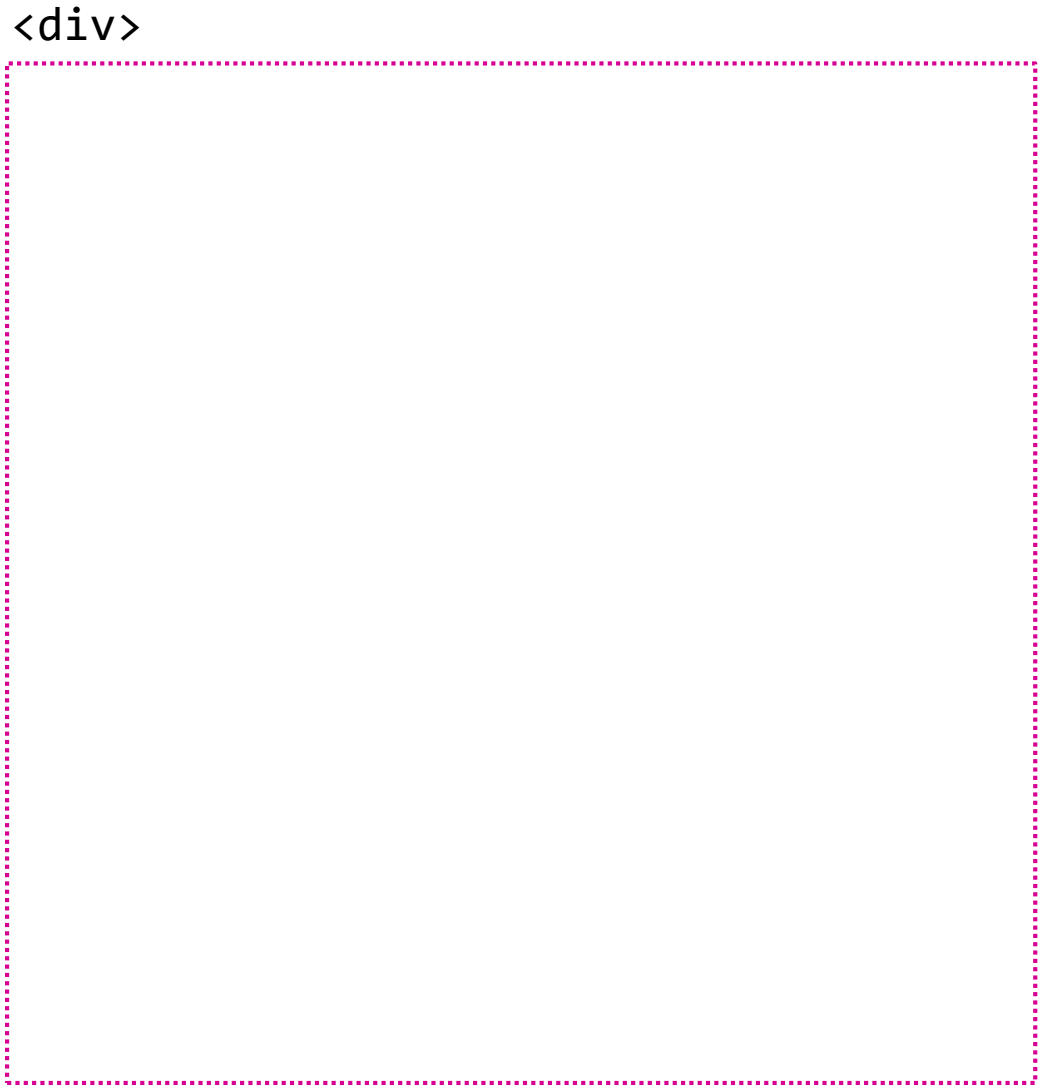
https://www.w3.org/TR/css-grid-1/

CSS Grid

- Terminologie

  Grid container

  - Elément qui accueille la grille
  - Peut être n'importe quel type d'élément :
    - body, div, section, footer, header ….



CSS Grid

- Terminologie

Grid Lines

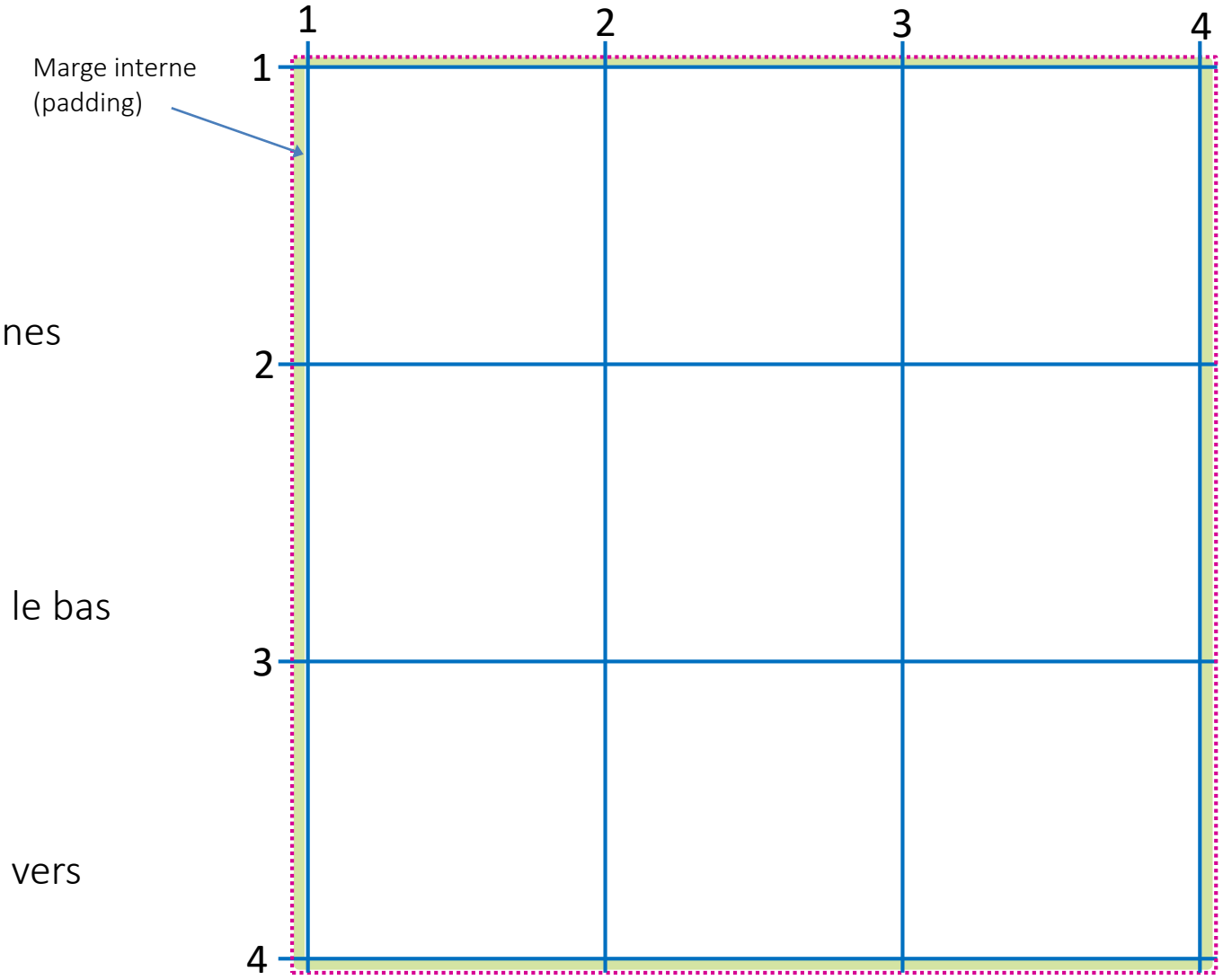
- Divisent le grid container pour créer une structure de grille régulière en lignes et colonnes

row Lines

- Lignes horizontales
- Numérotées de 1 à n_h en partant du haut vers le bas

column Lines

- Lignes verticales
- Numérotées de 1 à n_v en partant de la gauche vers la droite



La grille peut être quelconque (carrée ou rectangulaire) : $n_h >= 2$ et $n_v >= 2$

CSS Grid

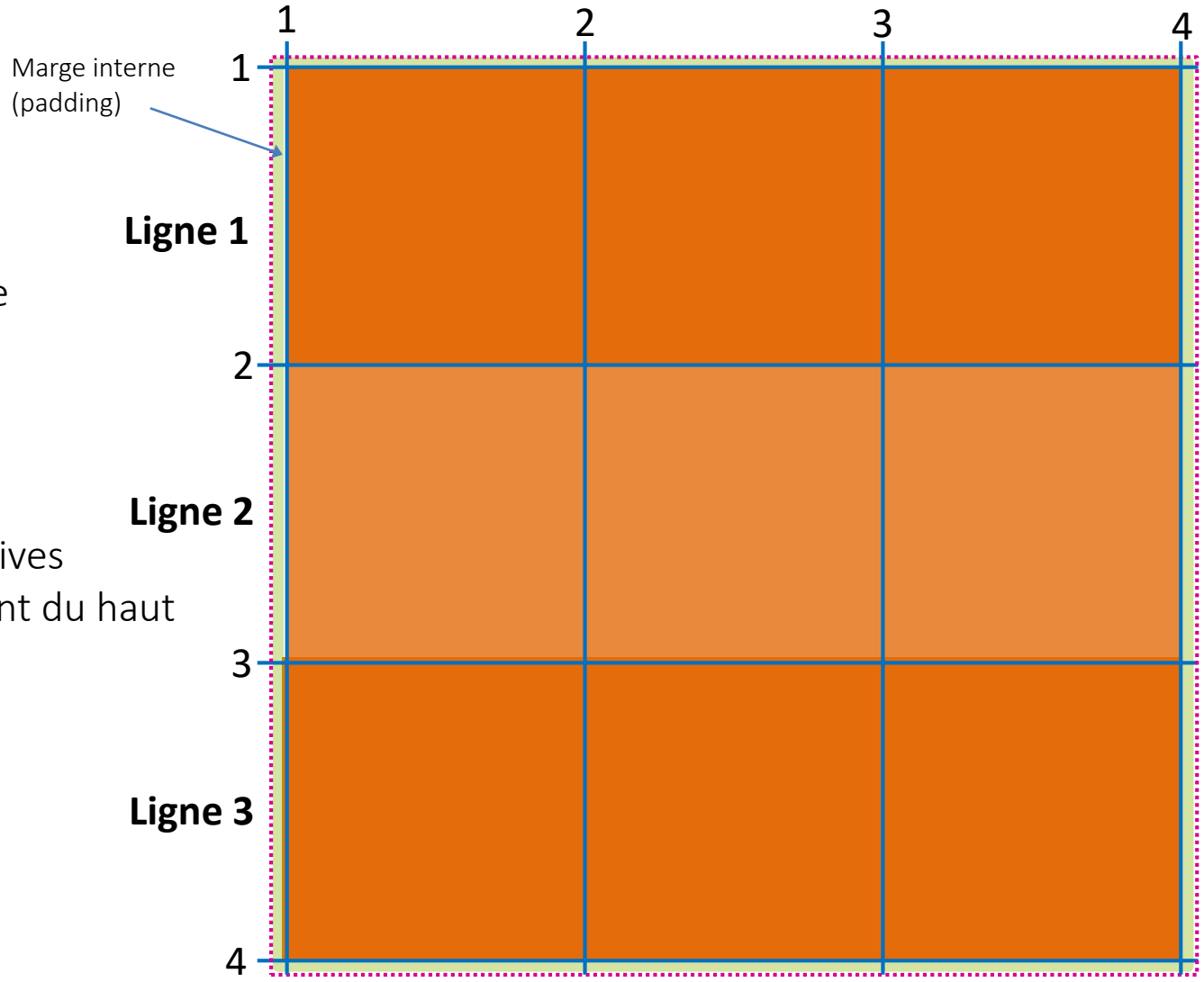
- Terminologie

Grid Tracks

- Région entre deux lignes adjacentes de la grille

Rows (lignes)

- Région entre deux lignes horizontales consécutives
- Les lignes sont numérotées de 1 à n_{h-1} en partant du haut



CSS Grid

- Terminologie

Grid Tracks

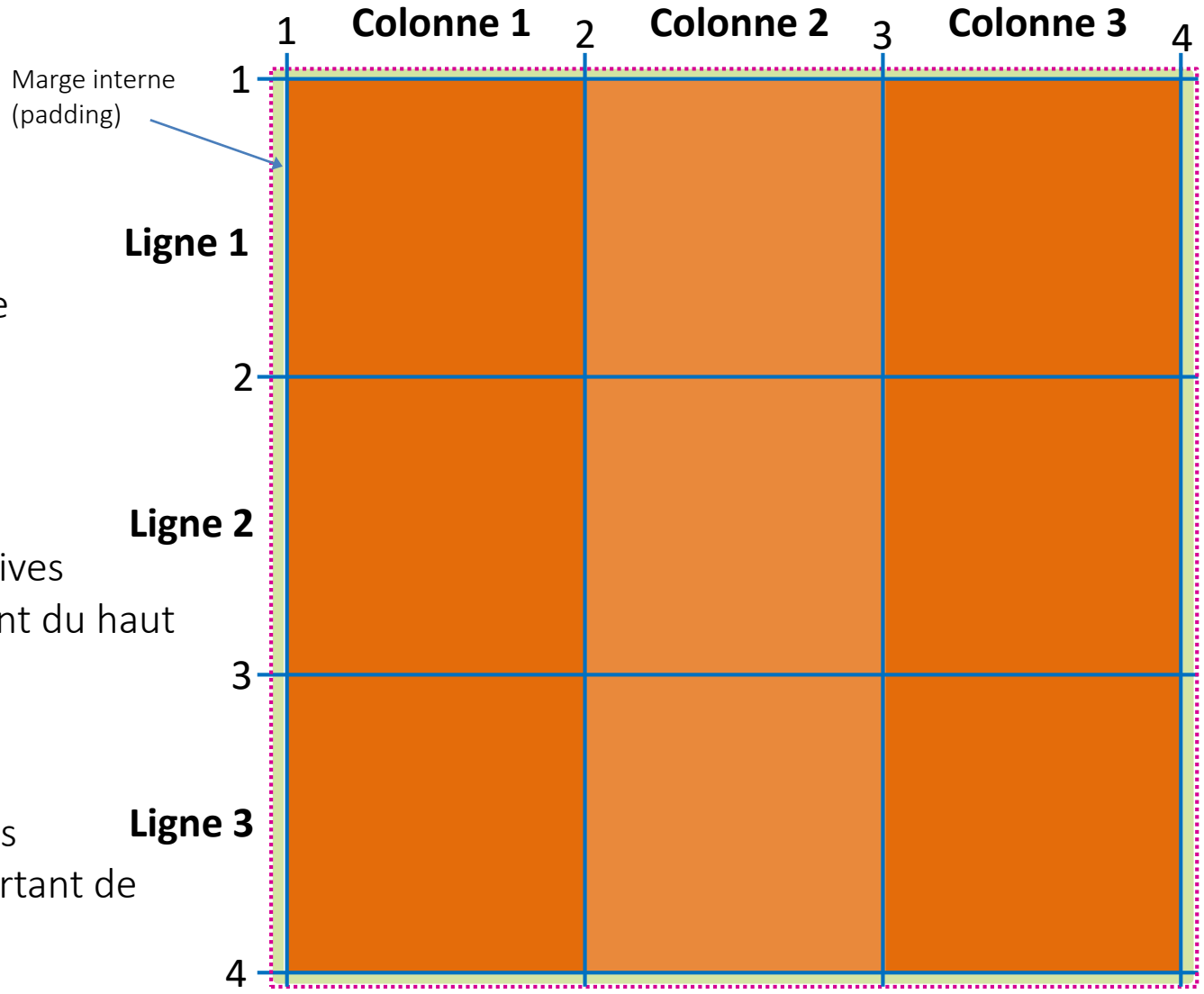
- Région entre deux lignes adjacentes de la grille

Rows (lignes)

- Région entre deux lignes horizontales consécutives
- Les lignes sont numérotées de 1 à n_{h-1} en partant du haut

Columns (colonnes)

- Région entre deux lignes verticales consécutives
- Les colonnes sont numérotées de 1 à n_{v-1} en partant de la gauche

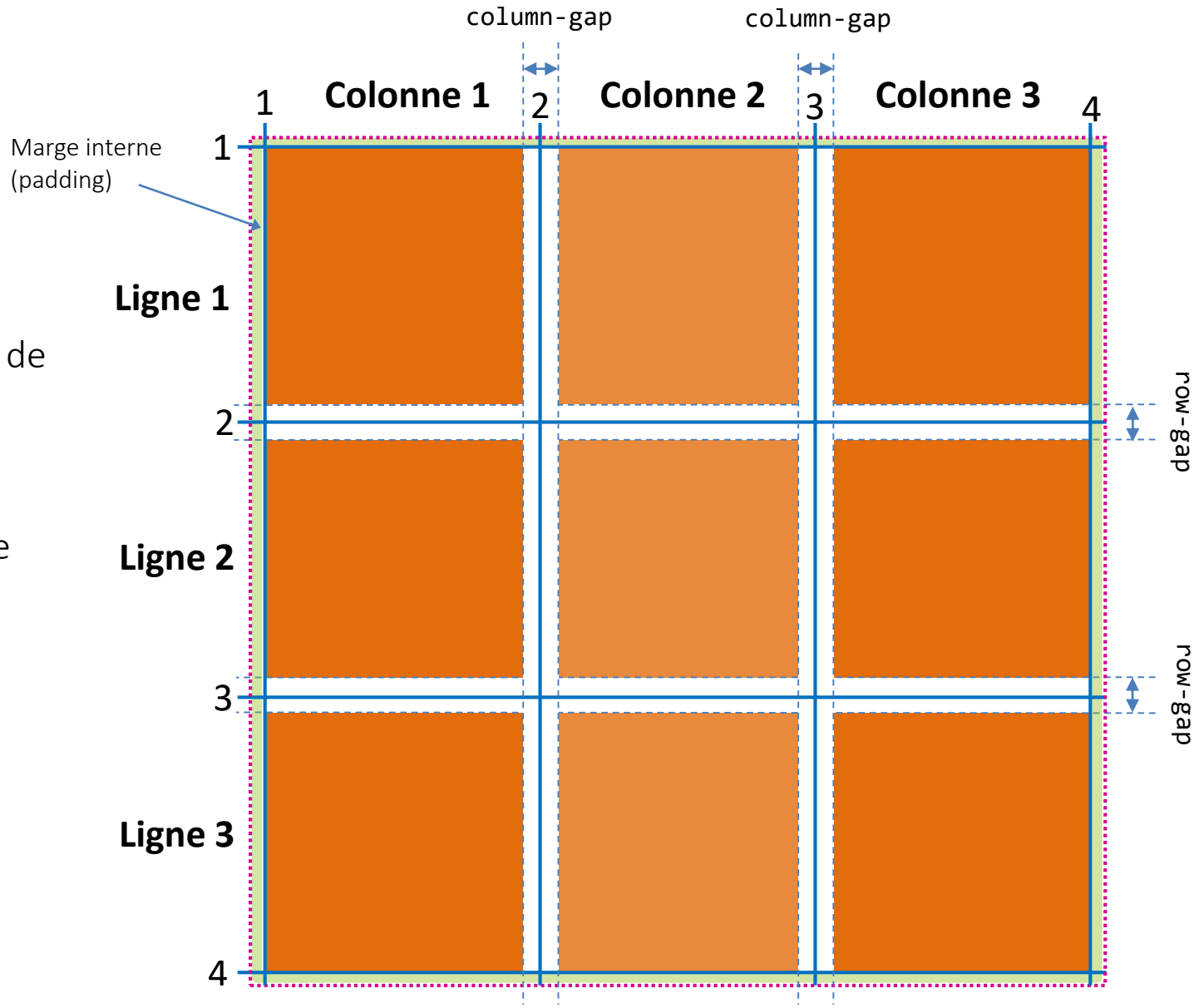


CSS Grid

- Terminologie

Grid Gaps

- Espace entre les lignes et colonnes adjacentes de la grille
- `column-gap` : espace entre deux colonnes (identique pour toutes les colonnes)
- `row-gap` : espace entre deux lignes (identique pour toutes les lignes)
- `column-gap` et `row-gap` peuvent avoir des valeurs différentes

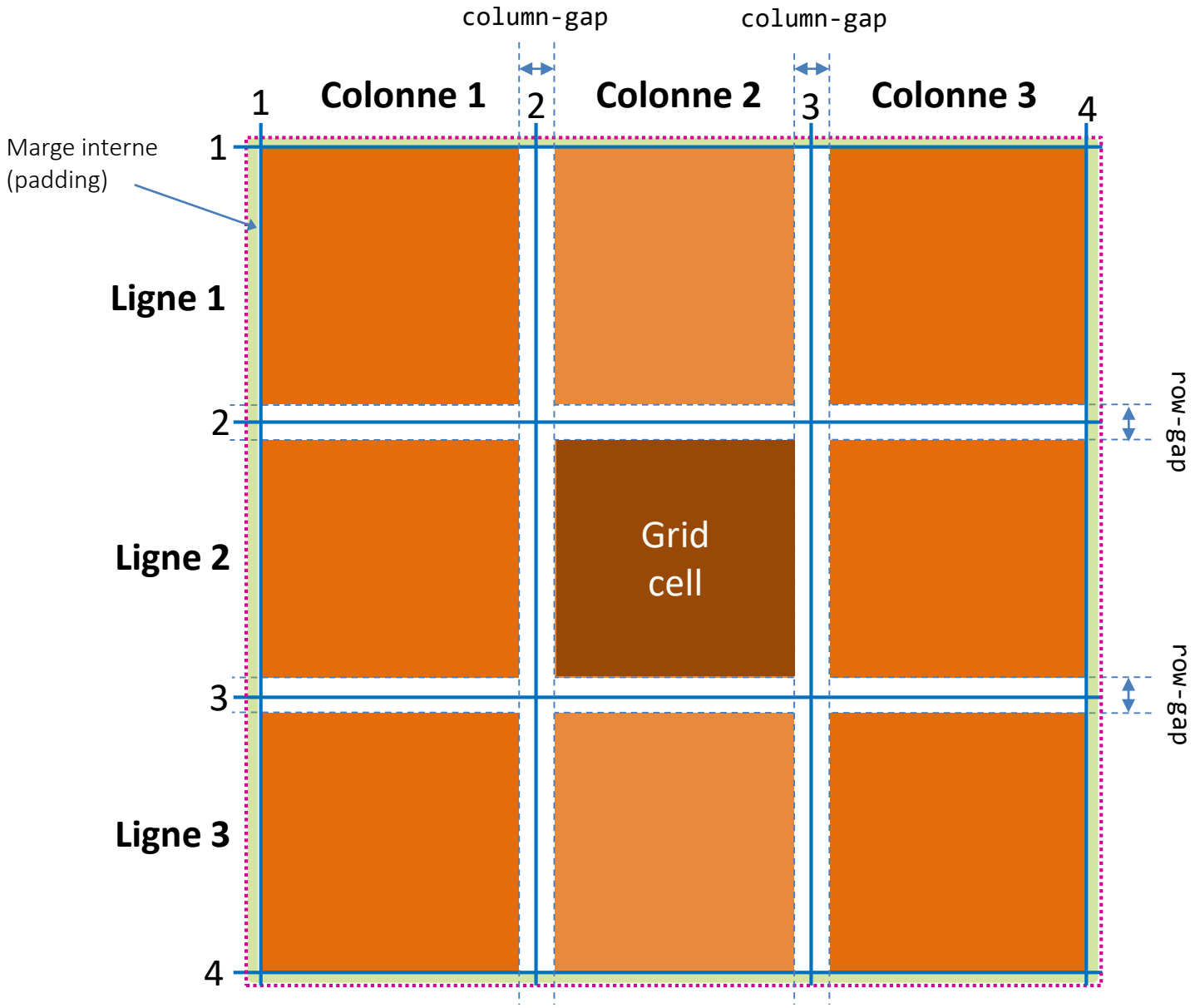


CSS Grid

- Terminologie

Grid Cells

- Intersection d'un ligne et d'une colonne
- L'unité de base de la grille
- Le nombre de cellules est $n_{h-1} * n_{v-1}$



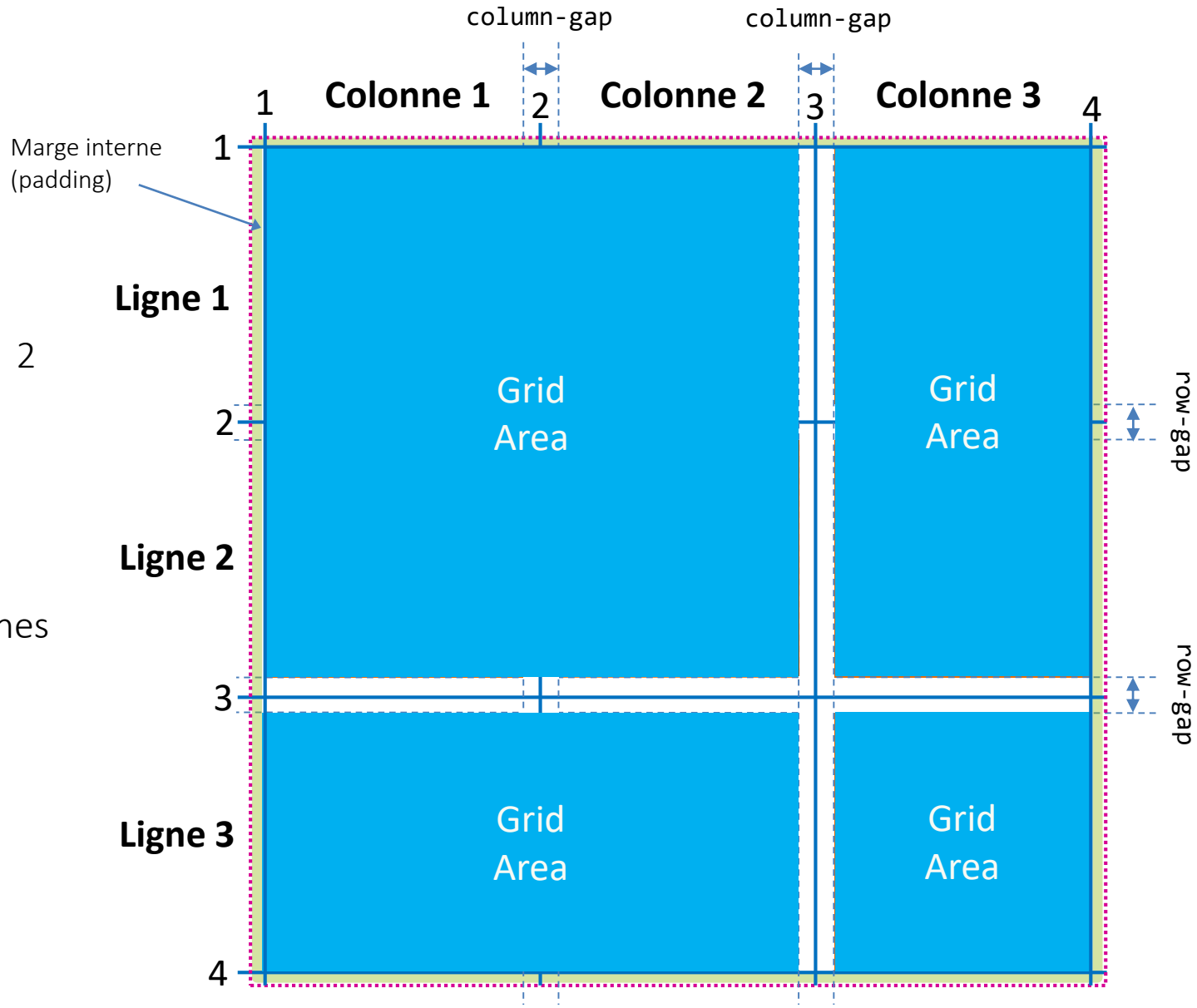
CSS Grid

- Terminologie

Grid Area

- La zone définie par 4 grid lines (2 rows lines et 2 column lines)
- C'est **toujours** une zone rectangulaire

- Identifiée par les numéros des lignes et colonnes qui la délimitent



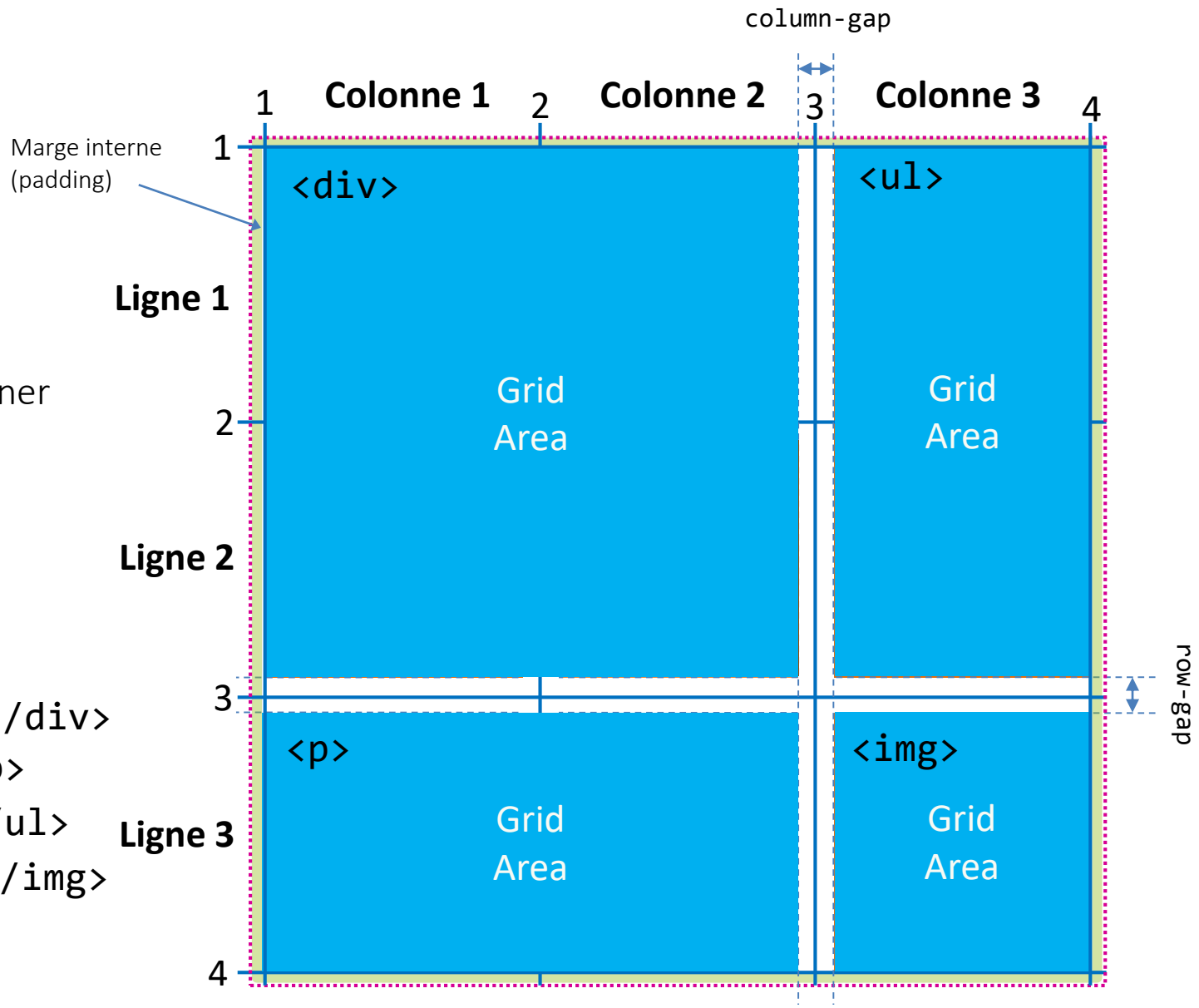
CSS Grid

- Terminologie

Grid items

- Eléments **descendants directs** du grid container
- Associés à une zone de la grille (grid area)
- Peuvent être de n'importe quel type :
  - p, ul, div, a, img …

```
<div class="grid-container">
     <div class="grid-item1"> ... </div>
     <p class="grid-item2"> ... </p>
     <ul class="grid-item3"> ... </ul>
     <img class="grid-item4"> ... </img>
</div>
```

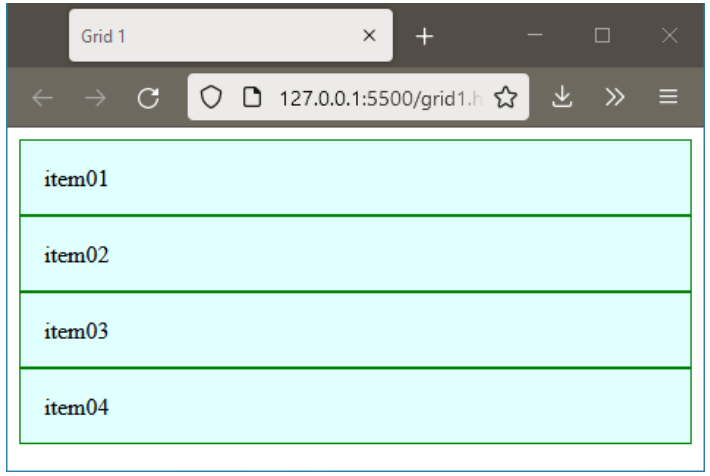


CSS Grid

- Créer un conteneur grid
  - Possibilité d'utiliser n'importe quel type d'élément HTML (**body**, **div**, **p** , **ul**…) en lui associant la propriété `display` avec l'une des deux valeurs `grid` ou `inline-grid`

```css
#my-grid {
    display:grid;
}

/* style pour démo : pas nécessaire pour les grilles */
.item {
    border: solid 1px green;
    background-color: lightcyan;
    padding: 1rem;
}
```



Par défaut le conteneur grille se comporte comme un flow layout normal (ici **div** les un en dessous des autres)

```html
<div id="my-grid">
    <div class="item">item01</div>
    <div class="item">item02</div>
    <div class="item">item03</div>
    <div class="item">item04</div>
</div>
```

Tout type d'élément ajouté à la grille est considéré comme un *grid item*

Pour les positionner il faut définir les lignes et colonnes de la grille explicitement

Nb: le nom de classe **item** est arbitraire (il peut être ce que vous voulez et n'est pas imposé par le layout en grille)

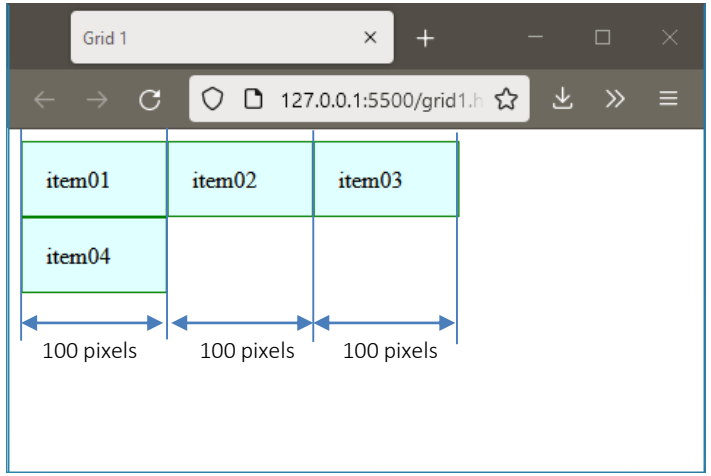
CSS Grid

- Définir les lignes et les colonnes
  - Effectué à l'aide des propriétés `grid-template-rows` et `grid-template-columns` qui permettent de spécifier le nombre de lignes (colonnes) avec éventuellement leur hauteur (largeur)

```css
#my-grid {
    display: grid;
    grid-template-columns: 100px 100px 100px;
}

/* style pour démo : pas nécessaire pour les grilles */
.item {
    border: solid 1px green;
    background-color: lightcyan;
    padding: 1rem;
}
```

```html
<div id="my-grid">
    <div class="item">item01</div>
    <div class="item">item02</div>
    <div class="item">item03</div>
    <div class="item">item04</div>
</div>
```



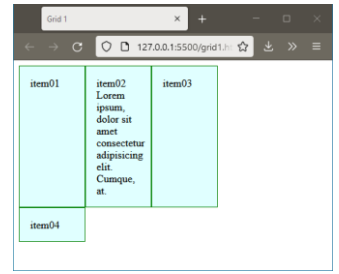
Grille avec 3 colonnes d'une largeur de 100 pixels

Les 3 premiers items sont placés successivement dans chacune des colonnes

Pour le 4ème item la grille utilise un algorithme d'auto placement qui le positionne automatiquement dans une nouvelle ligne

Ici les élément d'une ligne de la grille ont une hauteur calculée automatiquement (qui dépend du contenu)

```html
<div id="my-grid">
    <div class="item">item01</div>
    <div class="item">item02<br> Lorem ipsum, dolor sit amet
        consectetur adipisicing elit. Cumque, at.</div>
    <div class="item">item03</div>
    <div class="item">item04</div>
</div>
```

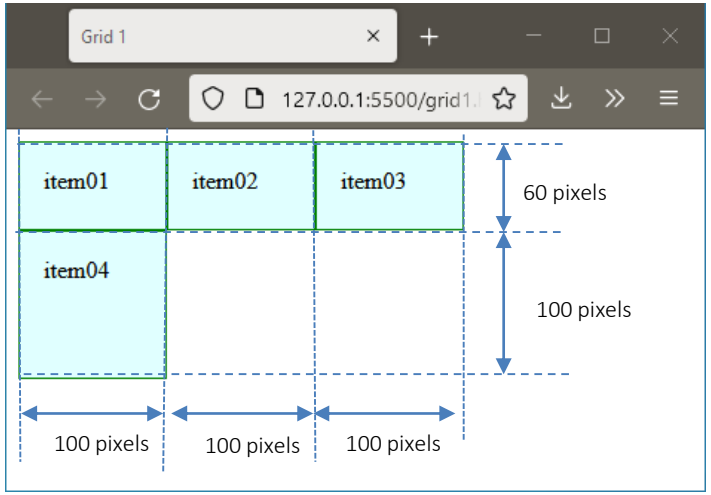


CSS Grid

- Définir les lignes et les colonnes
  - Effectué à l'aide des propriétés css `grid-template-rows` et `grid-template-columns` qui permettent de spécifier le nombre de lignes (colonnes) avec éventuellement leur hauteur (largeur)

```css
#my-grid {
    display: grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows : 60px 100px
}
/* style pour démo : pas nécessaire pour les grilles */
.item {
    border: solid 1px green;
    background-color: lightcyan;
    padding: 1rem;
}
```

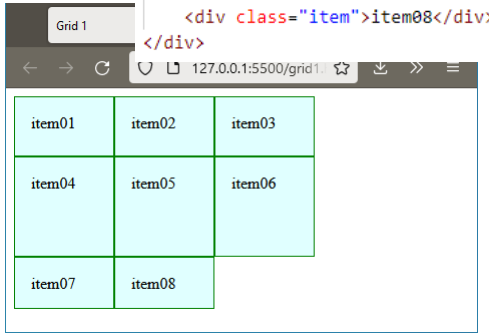
```html
<div id="my-grid">
    <div class="item">item01</div>
    <div class="item">item02</div>
    <div class="item">item03</div>
    <div class="item">item04</div>
</div>
```



Grille avec 3 colonnes d'une largeur de 100 pixels et deux lignes de hauteur respective 60 et 100 pixels

Si le container contient plus d'items que la grille définit de cellules l'algorithme d'auto placement les positionne automatiquement dans de nouvelle ligne dont la hauteur sera calculée automatiquement en fonction du contenu des items qui la constituent

```html
<div id="my-grid">
    <div class="item">item01</div>
    <div class="item">item02</div>
    <div class="item">item03</div>
    <div class="item">item04</div>
    <div class="item">item05</div>
    <div class="item">item06</div>
    <div class="item">item07</div>
    <div class="item">item08</div>
</div>
```



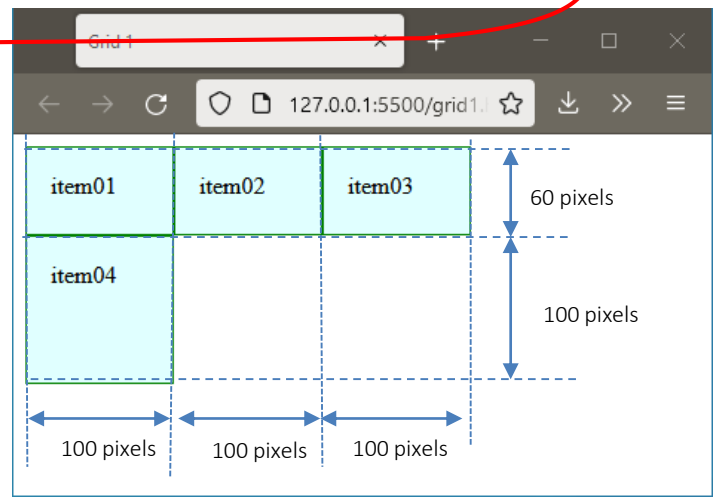
CSS Grid

- Définir les lignes et les colonnes
  - `grid-template : rows / columns` raccourci des propriétés `grid-template-rows` et `grid-template-columns`

```css
#my-grid {
    display: grid;
    grid-template-columns : 100px 100px 100px;
    grid-template-rows : 60px 100px;
}

/* style pour démo : pas nécessaire pour les grilles */
.item {
    border: solid 1px green;
    background-color: lightcyan;
    padding: 1rem;
}
```

```
grid-template : 60px 100px / 100px 100px 100px;
```

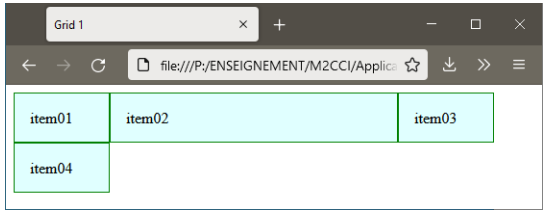



```html
<div id="my-grid">
    <div class="item">item01</div>
    <div class="item">item02</div>
    <div class="item">item03</div>
    <div class="item">item04</div>
</div>
```

CSS Grid

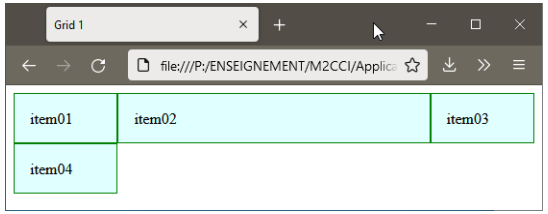
- Unités pour définir les largeurs de colonnes et hauteurs de lignes
  - Colonnes :
    - Taille fixe : `px`
    - Taille relative : `%` pourcentage de la largeur du container
    - Fraction : `fr` fraction de la largeur disponible
    - ...

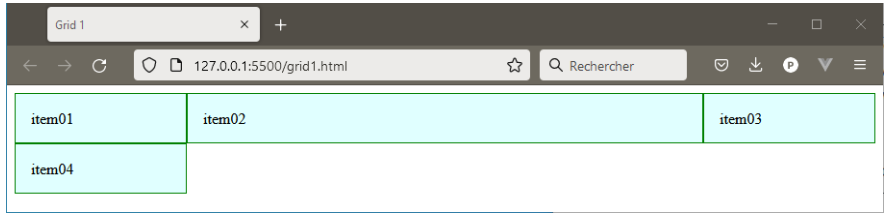
```
grid-template-columns : 100px 300px 100px;
```






```
grid-template-columns : 20% 60% 20%;
```





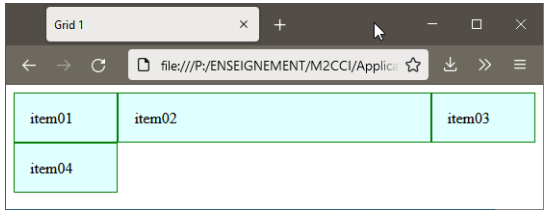

```
grid-template-columns : 2fr 6fr 2fr;
```

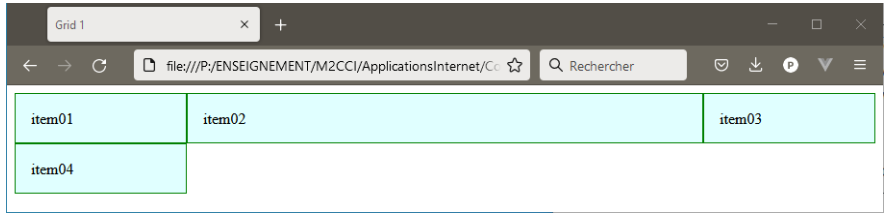
Le navigateur calcule le nombre total de fractions : 10

1ère colonne occupe 2 fractions sur 10

2ème colonne occupe 6 fractions sur 10

3ème colonne occupe 2 fractions sur 10





CSS Grid



Responsive Web Design : Media queries

- Nécessité d'adapter l'affichage des pages à différents modes de consultation



- Avec CSS2 et HTML4 il était déjà possible de spécifier un média de destination pour l'application d'une feuille de style

Attribut précisant le contexte dans lequel les styles doivent être appliqués

```
<!doctype html>
<head>
   ...
   <link rel="stylesheet" media="screen" href="screen.css" type="text/css" />
   <link rel="stylesheet" media="print" href="print.css" type="text/css" />
</head>
```

Media queries CSS3 permettent de définir un panel de critères plus précis en fonction de facteurs liés au dispositif d'affichage : Largeur, hauteur, orientation, résolution…

- **media query** : expression booléenne définissant un ensemble de conditions à réunir pour l'application de styles

```
@media screen and (min-width: 600px) and (max-width: 1024px) {
  .bloc {
    display:block;
    clear:both;
  }
  img {
     float left;
  }
}
```

styles appliqués si l'expression media est vraie

```
<link rel="stylesheet" media="screen and (max-width: 640px)"
                       href="smallscreen.css" type="text/css" />
```

Opérateurs logiques:
`and`, `or`, `not`, `only`

Critères possibles:
préfixés par `min-` ou `max-` lorsqu'ils acceptent des valeurs numériques

- `color` support de la couleur (bits/pixel)
- `color-index` périphérique utilisant une table de couleurs indexées
- `device-aspect-ratio` ratio du périphérique de sortie (par exemple 16/9)
- `aspect-ratio` ratio de la zone d'affichage
- `device-height` dimension en hauteur du périphérique
- `device-width` dimension en largeur du périphérique
- `grid` périphérique bitmap ou grille (ex : lcd)
- `height` dimension en hauteur de la zone d'affichage
- `monochrome` périphérique monochrome ou niveaux de gris (bits/pixel)
- `orientation` orientation du périphérique (`portait` ou `landscape`)
- `resolution` résolution du périphérique (en dpi, dppx, ou dpcm)
- `scan` type de balayage des téléviseurs (`progressive` ou `interlace`)
- `width` dimension en largeur de la zone d'affichage