HTML 5 et CSS

Le langage HTML

HyperText Markup Language

= Langage de structuration d'un document à base de balise

Structure d’une page

le modèle des boîtes

Quelques balises à connaître

... : une division

... : un paragraphe

... : une partie d'un texte

 : un retour à la ligne

 : une image

<a ...>... : un lien

<h1>...</h1> : un titre principal

<h2>...</h2> : un titre secondaire (etc...)

Composition d'une page

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>ceci est le titre de ma page</title>
  </head>
  <body>
    <p>ceci est ma première page</p>
    <img src="uneImage.jpg" alt="texte alternatif">
  </body>
</html>
```

Balisage des textes : les titres

Un titre est défini avec une balise de type <hn>...</hn>

- h1 : titre principal du texte
- h2 : titre de section
- h3 : sous-titre de section
- etc... (jusqu'à h6)

Un titre est automatiquement précédé et suivi par un retour à la ligne

Balisage des textes : les paragraphes

Un paragraphe est défini avec une balise ...

 Un paragraphe est automatiquement précédé et suivi par un retour à la ligne

Balisage des textes : les listes

Les listes sont définies par les balises <ul>... ou <ol>...

- ul : liste non-ordonnée ⇒ puce devant chaque élément
- ol : liste ordonnée ⇒ numéro ou lettre devant chaque élément

Les éléments d'une liste sont identifiés par la balise <li>...

Balisage des textes : les liens

Un lien est défini par la balise ...

Il permet :

- de naviguer sur internet en suivant le *lien* indiqué
- de se rendre à un emplacement spécifique de la page
- de créer un email ou d'appeler un numéro de téléphone *

** en fonction des capacité de l'appareil*

Exemples de liens

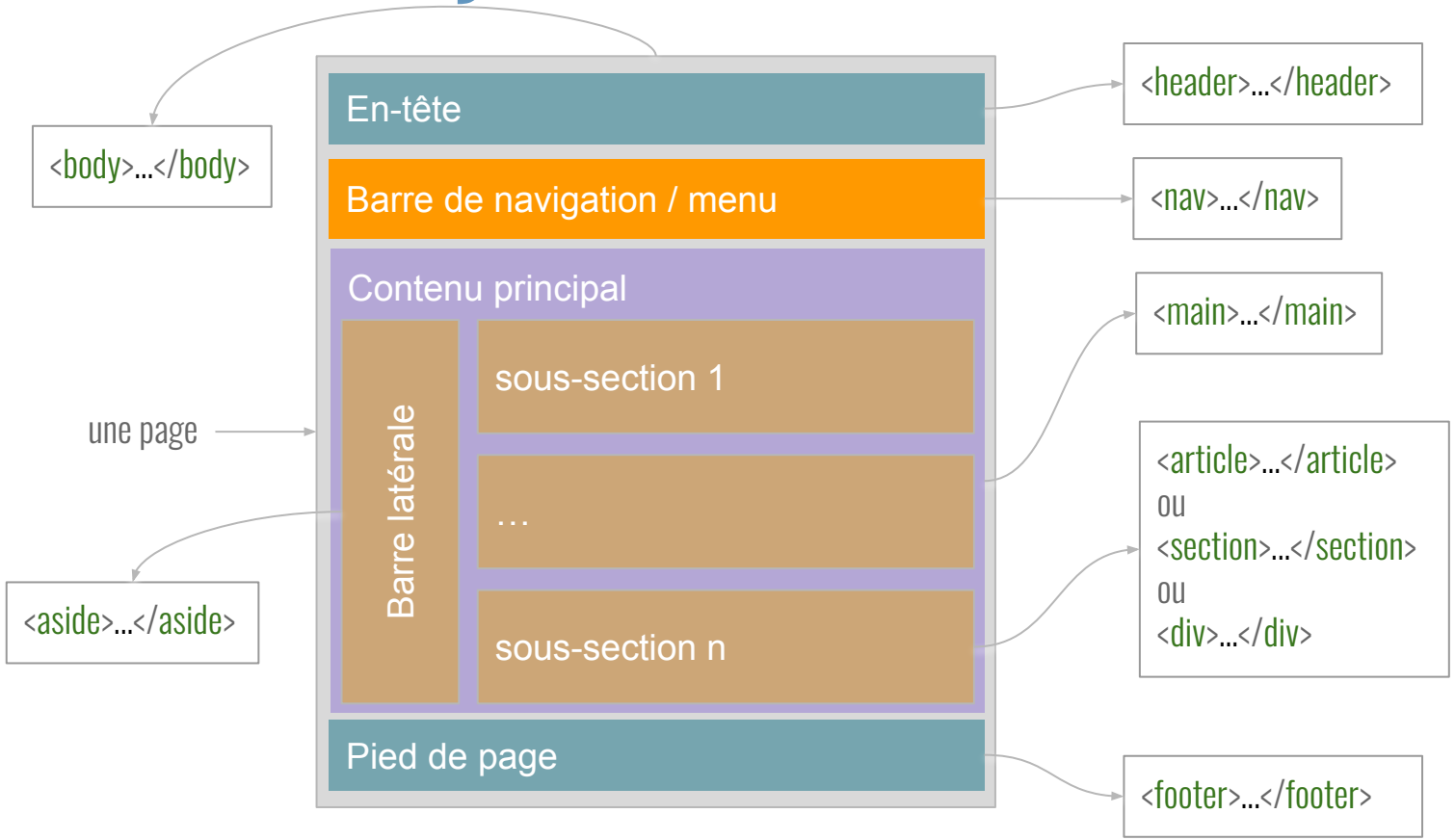
vers la page d'accueil de Google

vers un élément avec l'id coucou

créer un mail pour toto

appeler (?) le 12345

Retour sur le système de boîtes



De l’audio ? De la vidéo ?

```
<audio src="chemin vers un son" controls>...</audio>
<video src="chemin vers une vidéo" controls>...</video>
```

...

Correction

la table

```
<table>
    <thead>                                   l'entête
        <tr>
            <td rowspan="2"></td>
            <th colspan="3" scope="col">T1</th> […]
        </tr>
        <tr>
            <th scope="col">Jan</th> […]
        </tr>
    </thead>
    <tbody>                                   le corps
        <tr>
            <th scope="row">USA</th>
            <td>1.2</td> […]
        </tr> […]
    </tbody>
    <tfoot>                                   le pied
        <tr>
            <th scope="row">Total</th>
            <td>1.7</td> […]
        </tr>
    </foot>
</table>
```

la 1ère cellule a une hauteur de 2 lignes

la cellule a une largeur de 3 colonnes

entête de colonne

entête de ligne

On sait désormais
structurer une page web.

Et si on s'occupait un peu
de l’esthétisme ?

Les CSS

Qu’est-ce que CSS ?

Cascading **S**tyle **S**heets

= Feuilles de style en cascade

Qu’est-ce que CSS ?

- un ensemble de règles de présentation
- chaque règle est composée :
  - d’un sélecteur (qui est impacté par la règle ?)
  - d’une liste de propriétés auxquelles est affectée une valeur
- si plusieurs règles correspondent à un élément, elles sont additionnées

Un exemple simple

sélecteur : “tous les titres de niveau 1”

```
h1 {
    color: red;
    font-size: 5em;
}
```

couleur d’écriture

taille d’écriture

sélecteur : “tous les paragraphes”

```
p {
    color: black;
}
```

Comment et où intégrer les CSS ?

- dans un où des fichiers *nomfichier*.css puis en liant le fichier à votre page

```
<head>
      [...]
      <link rel="stylesheet" href="joli.css" type="text/.css">
      [...]
</head>
```

- dans le code de votre page

```
<style>
      h1{
            color :blue;
      }
</style>
```

- sur un élément en particulier

```
<p style=”color:red”>
```

Les sélecteurs

- une balise

```
CSS : h1{...}
```

- une classe

```
HTML : <p class="p1 blue">
```

```
CSS : .p1{...}
```

- un identifiant

```
HTML : <p id="p1">
```

```
CSS :  #p1{...}
```

- une position relative

```
HTML : <p> [...]
           <span>[...]</span>
           [...]
       </p>
```

```
CSS :  p span{...}
```

- un état

```
HTML : <a> [...]</a>
```

```
CSS :  a:link{...}
       a:hover{...}
       a:visited{...}
```

La combinaison de sélecteurs

`p span.bleu {...}` → tous les spans dans un paragraphe avec la classe “bleu”

`p#p1 span.bleu {...}` → tous les spans avec la classe “bleu” dans le paragraphe “p1”

`h1,h2 {...}` → tous les titres de niveau 1 et tous les titres de niveau 2

Il existe de nombreux opérateurs permettant d’affiner ces règles

`*,+,-,~,>,:first-child,:last-child,:n-child(),…`

Quelle règle sera appliquée ?

Si plusieurs règles sont applicables,
elles sont combinées propriété par propriété

⚠️ il y a une priorité dans les règles calculée sur 4 positions

Calcul de la priorité (a,b,c,d)

a = propriété définie dans l'attribut style de l'élément

b = sélection via un identifiant

c = sélection via une classe

d = sélection par une balise

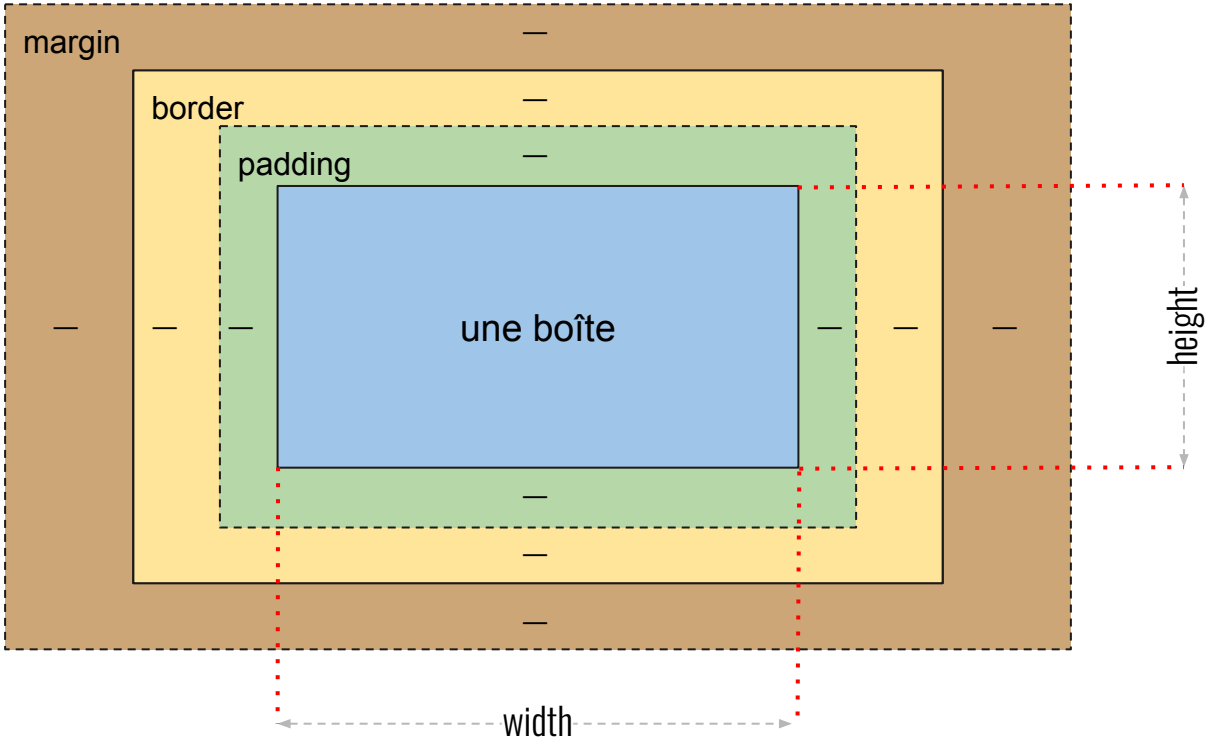
La propriété ayant le plus gros score gagne !

En cas d'égalité l'ordre de définition des règles est pris en compte :
"le dernier qui parle a raison"

Sélecteur	Priorité
`*{}`	0,0,0,0
`li{}`	0,0,0,1
`li:first-line {}`	0,0,0,2
`ul li{}`	0,0,0,2
`ul ol+li{}`	0,0,0,3
`ul li.liste`	0,0,1,2
`li.red.level`	0,0,2,1
`#exemple`	0,1,0,0
`style=""`	1,0,0,0

Retour sur le modèle de boîtes (oui, encore...)

Retour sur le modèle de boîtes (encore)



Le positionnement extérieur

Chaque élément à un positionnement extérieur par défaut
mais on peut le modifier en CSS !

- display : block → positionnement en bloc
- display : inline → positionnement en ligne
- display : inline-block → positionnement en bloc (sans retour à la ligne)

Quelles différences ?

La boîte en bloc

- occupe toute la largeur de son conteneur
- commence sur une nouvelle ligne et finit par un retour à la ligne
- respecte toujours les valeurs *width* et *height*
- les *padding*, *margin* et *border* repoussent les autres éléments

Exemples : h1, h2, p, ...

La boîte en ligne

- pas de retour à la ligne ⇒ les éléments se suivent en ligne
- *width* et *height* ne s'appliquent pas
- les *padding*, *margin* et *border* verticales s'appliquent mais ne repoussent pas les autres éléments
- les *padding*, *margin* et *border* horizontales s'appliquent et repoussent les autres éléments

Exemples : a, span,...

Arrière-plans (`background`)

- une couleur → `background-color : #45A924;`
- une image → `background-image : url("a.png");`
  - positionner (`background-position`)
  - répéter (`background-repeat`)
  - dimensionner (`background-size`)

Bordures (`border`)

4 propriétés :

- épaisseur → `border-width : 2px;`
- style → `border-style : solid;`
- couleur → `border-color : red;`
- coin arrondi → `border-radius : 20px;`

Version courte → `border : 2px solid red;`

ou

Ciblage d'un côté en particulier → `border-top-*`

Exercice

Utilisation des CSS

Reproduisez cette cible
à l'aide de div et de CSS

Le positionnement relatif des boîtes

On connaît désormais :

- le fonctionnement des boîtes
- comment changer la couleur, la taille, etc... de différents éléments
- la sélection d'éléments

Mais comment se place une boîte par rapport aux autres ?

Disposition des éléments par défaut

C'est ce que nous avons vu pour le moment !

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer ligula purus, facilisis ac tellus sit amet, commodo convallis nulla. Praesent quis enim efficitur metus elementum feugiat. </p>
<p>Vestibulum at luctus dolor. <span>Mauris sit amet </span> porta felis, nec suscipit justo. Ut luctus pretium erat, id hendrerit lectus dictum id. </p>
```

.HTML

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer ligula purus, facilisis ac tellus sit amet, commodo convallis nulla. Praesent quis enim efficitur metus elementum feugiat.

Vestibulum at luctus dolor. Mauris sit amet porta felis, nec suscipit justo. Ut luctus pretium erat, id hendrerit lectus dictum id.

```
p{ border : 2px solid olive}

span{
        color : white;
        background-color  : indianred;
}
```

.CSS

Les éléments s'affichent dans l'ordre de déclaration

→ les "block" prennent toute la largeur

→ les "inline" s'intègrent sur la même ligne que ces voisins "inline" ou texte

→ les "inline-block" fait un peu les deux

\+ prise en compte des marges

La disposition par positionnement

Permet de moduler la position d'un élément par rapport :

- à un autre
- à la page
- à la fenêtre d'affichage

Positionnement “static”

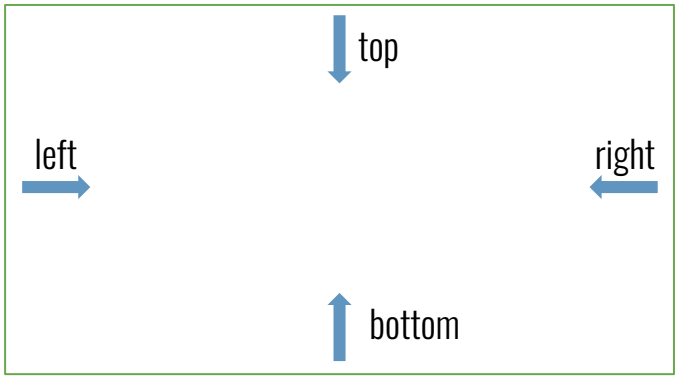
C’est le positionnement par défaut

```
position: static;
```

Positionnement “relative”

Il décale la position par rapport la “normale”

```
position: relative;
top: 30px;
bottom: 30px;
left: 30px;
right: 30px;
```



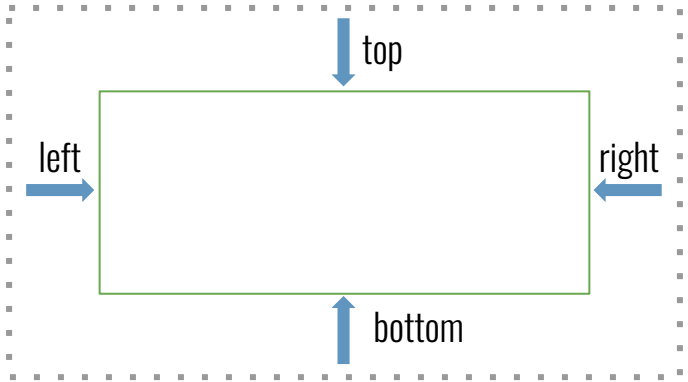
Notes :

- L’emplacement “normal” de l’élément est toujours réservé
- les éléments suivants ne se décalent pas

Positionnement “absolute”

Il fixe la position par rapport à l’élément le contenant

```
position: absolute;
top: 30px;
bottom: 30px;
left: 30px;
right: 30px;
```



Notes :

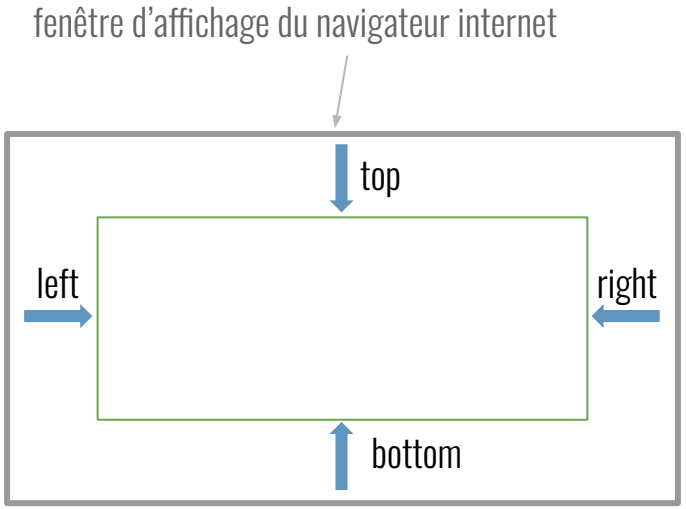
- l’emplacement “normal” n'apparaît plus
- les éléments suivants se décalent
- l’élément contenant est le plus proche parent avec une position “relative”
  par défaut l’élément contenant est la page

. . .

Positionnement “fixed”

Identique à “absolute”

```
position: fixed;
top: 30px;
bottom: 30px;
left: 30px;
right: 30px;
```

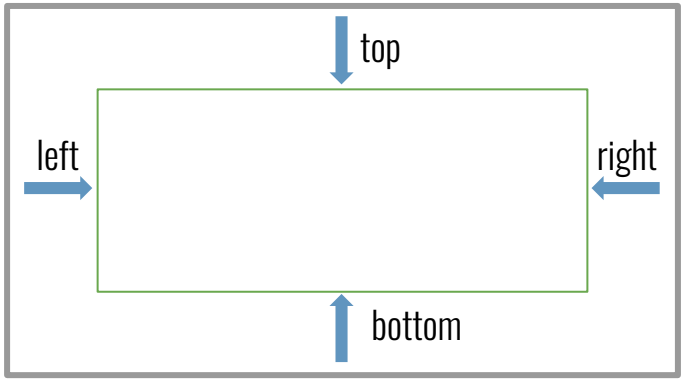


Sauf que… le contenant est la fenêtre du navigateur

Positionnement “sticky”

C’est un hybride : il est “relative” mais devient “fixed” lorsqu’on dépasse un seuil

```
position: sticky;
top: 30px;
left: 30px;
```



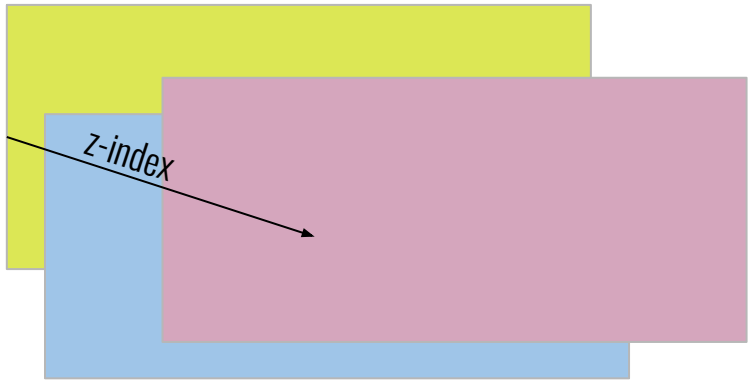
Exemple d’utilisation : un menu qui bouge avec la page mais qui reste apparent si on scrolle trop

Le z-index

Il est possible de créer des chevauchements d'éléments.

z-index change l'ordre d'affichage

```
z-index: 10;
```



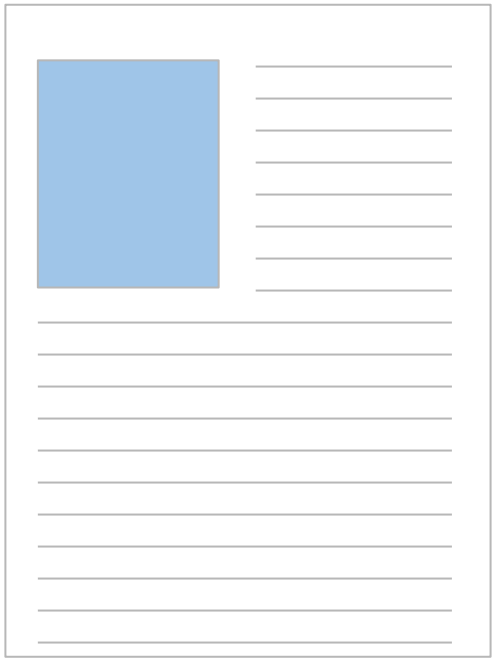
Note :

- plus z-index est grand, plus l'élément est mis en avant
- Les éléments en "static" ne sont pas impactés

A vous de jouer !

Créez une page contenant plusieurs éléments et observez leur comportement en modifiant “position” et “z-index”

Les boîtes flottantes



- Créées initialement pour insérer des images dans du texte
- Colle une boîte à gauche ou à droite

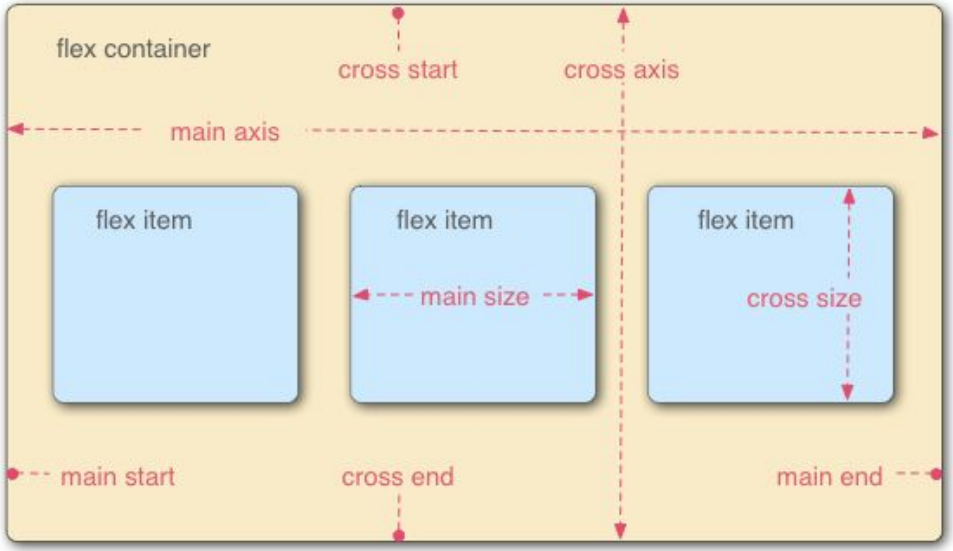
```
float : left|right;
```

- La boîte est retirée de l'affichage normal
- Les autres boîtes l'entourent

Note : Avec l'émergence de flexbox, on revient à l'usage initial

Flexbox

```
<section style="display:flex">
    <article>flex item</article>
    <article>flex item</article>
    <article>flex item</article>
</section>
```



Les options de la flexbox

Au niveau de la flexbox :

`flex-direction : row | column;` → détermine le sens de l'axe principal (avec les variantes *`-reverse`*)

`flex-wrap : wrap;` → empêche le débordement des éléments flex (taille contrainte du conteneur)

`flex-flow : row wrap;` → combine `flex-direction` et `flex-wrap`

`align-items : stretch|center|flex-start|flex-end;` → positionne les éléments sur l'axe croisé

`justify-content : space-around|space-between|center|flex-start|flex-end;`
→ positionne les éléments sur l'axe principal

Les options de la flexbox

Au niveau des éléments flex :

`flex : 150px;` → donne une taille minimale aux éléments flex (ici 150px)

`flex : 2;` → indique “le nombre de places” occupé par un élément (ici l’élément prendra 2x plus de place)

`flex : 2 150px;` → l’élément fera au moins 150px et l’espace restant sera réparti avec une importance 2 pour lui

`order : 10;` → donne une position de l’élément (ici tous les éléments avec un `order` <10 seront placés avant)

`align-self : flex-start;` → positionne l’élément sur l’axe croisé (ici au niveau de flex-start)