



# Développement Front-End Partie 1 (suite)

**Aous Karoui**

[aous.karoui@iut2.univ-grenoble-alpes.fr](mailto:aous.karoui@iut2.univ-grenoble-alpes.fr)





# Partie 1

## Programmation statique (suite)



### Règles de positionnement

- 1. Positionnement manuel
- 2. Profondeur
- 3. Flexbox Layout

# Les display types



## Trois types de display

- ***block*** (h, p, header)
  - s'affichent en colonne
  - possèdent des dimensions (largeur / hauteur)
- ***inline*** (a, em, strong)
  - s'affichent en ligne
  - Prennent les dimensions du conteneur
- ***inline-block*** (input)
  - s'affichent en ligne
  - peuvent avoir des dimensions propres
    - dimensions fixées par le contenu
    - dimension fixées manuellement

# 1. Positionnement manuel



## Les positions en css

- *static*
  - Par défaut, tous les éléments sont *static*
  - Apparaissent dans l'ordre du flux HTML
- *relative*
  - Permet de décaler des éléments à partir de leur position d'origine (dans le flux)
  - Peut servir de référentiel pour les enfants
- *absolute*
  - positionner par rapport au cadre de la page
- *fixed*
  - Permet une visibilité permanente, même avec scroll (menus)
- *sticky*
  - Se comporte comme fixe, mais tant que son parent est visible
  - N'est plus visible si le parent n'est plus visible

# 1. Positionnement manuel



## Positionnement statique

```
div {  
  width: 600px ;  
  padding-top: 20px ;  
  border: solid 1px black;  
}  
p {  
  margin-left: 20px ;  
  margin-bottom: 20px ;  
  width: 300px;  
  border: solid 1px black;  
}
```

```
<div>  
  <p>paragraphe</p>  
  <p>paragraphe</p>  
</div>
```

paragraphe

paragraphe

# 1. Positionnement manuel



## Positionnement relatif

```
<div>
<p id="premier">paragraphe</p>
<p id="second">paragraphe</p>
</div>
```

```
...
div {
  width : 600px ;
  padding-top : 20px ;
  border: solid 1px black;
}
p#premier {
  margin-left : 20px ;
  width: 300px;
  border: solid 1px black;
}
p#second {
  margin-left : 20px ;
  width: 300px;
  border: solid 1px black;
  position: relative;
  left: 4px;
  bottom: 22px;
}
```

paragraphe  
paragraphe

# 1. Positionnement manuel



## Positionnement relatif

Un autre exemple

Autre exemple : le décalage est relatif à la position normale de l'élément dans le bloc parent

Un paragraphe avec un élément décalé du reste du texte.

```
.decale {  
  position: relative;  
  bottom: 5px;  
  border: solid 1px black;  
}  
...  
<p>Un paragraphe avec  
<span class="decale">un  
&eacute;l&eacute;ment  
d&eacute;cal&eacute;.</span>  
du reste du texte.</p>  
...
```

# 1. Positionnement manuel



- La position de l'élément est déterminée de manière **absolue** dans son conteneur parent **positionné** le plus proche, ou à défaut, dans la fenêtre du navigateur
- On utilise la propriété **position**, avec la valeur **absolute**, pour positionner un élément de manière absolue.
- Les propriétés **top**, **right**, **left**, **bottom**, permettent alors de fixer la position.

## Positionnement absolu

```
#boitel {  
  position: relative;  
  width: 300px;  
  border: solid 1px black;  
}  
#boite2 {  
  position: absolute;  
  top: 10px;  
  right: 30px;  
  border: solid 1px black;  
}  
...  
<div id="boitel">  
  <p>Boite 1</p>  
  <div id="boite2">Boite 2</div>  
</div>
```

Boite 1 avec son contenu son contenu  
son contenu

Boite 2



# 1. Positionnement manuel



## Positionnement fixe

Le **positionnement fixé** est très comparable au positionnement absolu, sauf que l'élément fixé **reste à sa place sur l'écran** même lorsque l'utilisateur fait défiler le contenu.

Un élément fixé est comme « ancré » à sa place.

On utilise la propriété **position**, avec la valeur **fixed**.

Les propriétés **top**, **right**, **left**, **bottom**, permettent alors de définir la position.

# 1. Positionnement manuel



## Positionnement flottant

- Créer des éléments flottants les uns par rapport aux autres
- Par défaut tous les éléments sont *float : none*; (s'empilent dans le flux)
  - [Démonstration très simple](#)
- Change le type de *display* éléments
  - Exemple: si élément “inline”, il devient “block” (attention aux problèmes de dimensions)

# 1. Positionnement manuel



## Positionnement flottant

Risque d'avoir des problèmes de dimensions

- Sortie du flux → hauteur 0
- Débordement à cause de dimension trop grande

Lorem ipsum dolor sit amet consectetur adipisicing elit. Tempora temporibus nisi tenetur saepe optio quos, nostrum accusantium maiores eligendi dolore eos, provident inventore delectus harum error impedit rem omnis rerum.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Tempora temporibus nisi tenetur saepe optio quos, nostrum accusantium maiores eligendi dolore eos, provident inventore delectus harum error impedit rem omnis rerum.

200 x 400

### Spécialités

Lorem ipsum dolor sit amet consectetur adipisicing elit. Tempora temporibus nisi tenetur saepe optio quos, nostrum accusantium maiores eligendi dolore eos, provident inventore delectus harum error impedit rem omnis rerum.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Tempora temporibus nisi tenetur saepe optio quos, nostrum accusantium maiores eligendi dolore eos, provident inventore delectus harum error impedit rem omnis rerum.

# 1. Positionnement manuel



## Positionnement flottant

Pour éviter les problèmes de dimensions

- **Dégager les éléments du flottement**
  - **clear**: (**right**, **left** ou **both**) sur le 1er élément qui suit
  - Utiliser un élément supplémentaire (peut être vide)
- **Limiter au conteneur le formatage en bloc** des éléments flottants
  - **overflow** : **hidden**
  - à utiliser sur le conteneur

Lorem ipsum dolor sit amet consectetur adipisicing elit. Tempora temporibus nisi tenetur saepe optio quos, nostrum accusantium maiores eligendi dolore eos, provident inventore delectus harum error impedit rem omnis rerum.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Tempora temporibus nisi tenetur saepe optio quos, nostrum accusantium maiores eligendi dolore eos, provident inventore delectus harum error impedit rem omnis rerum.

200 x 400

## 2. Profondeur



Si on a des éléments qui se chevauchent

- Le navigateur place au premier plan l'élément le plus récent dans le flux HTML

Un élément en position *static* est par défaut au dessous des autres éléments positionnés

S'il passe en position relative il devient au dessus

- Le Z-index permet de changer cette logique
- L'élément au **z-index** le **+ élevé**, apparaît au **1er plan**



# 3. Flexbox Layout

Il est nécessaire de le spécifier dans le paramètre "display" du conteneur

```
.flex-container {  
  display: flex;  
}
```

Par conséquent, les éléments fils deviennent flexibles à leur tour

# 3. Flexbox Layout

## flex-direction

Permet d'agencer les éléments dans le sens voulu. C'est-à-dire les positionner verticalement ou les inverser.

Cette propriété peut prendre les valeurs suivantes :

- `row` : organisés sur une ligne (par défaut)
- `column` : organisés sur une colonne
- `row-reverse` : organisés sur une ligne, mais en ordre inversé
- `column-reverse` : organisés sur une colonne, mais en ordre inversé



# 3. Flexbox Layout

## flex-wrap

Permet un retour à la ligne automatique des blocs

Cette propriété peut prendre les valeurs suivantes :

- nowrap : pas de retour à la ligne (par défaut)
- wrap : les éléments vont à la ligne lorsqu'il n'y a plus la place
- wrap-reverse : les éléments vont à la ligne lorsqu'il n'y a plus la place en sens inverse

# 3. Flexbox Layout

## justify-content

Pour aligner horizontalement les blocs

Cette propriété peut prendre les valeurs suivantes :

- `flex-start` : alignés au début (par défaut)
- `flex-end` : alignés à la fin
- `center` : alignés au centre
- `space-between` : les éléments sont étirés sur tout l'axe (il y a de l'espace entre eux)
- `space-around` : idem, les éléments sont étirés sur tout l'axe, mais ils laissent aussi de l'espace sur les extrémités

# 3. Flexbox Layout

## align-items

Pour aligner verticalement les blocs

Cette propriété peut prendre les valeurs suivantes :

- `stretch` : les éléments sont étirés sur tout l'axe (valeur par défaut)
- `flex-start` : alignés au début
- `flex-end` : alignés à la fin
- `center` : alignés au centre
- `baseline` : alignés sur la ligne de base (semblable à `flex-start`)

# 3. Flexbox Layout

## align-content

Pour gérer plusieurs lignes de blocs

Cette propriété peut prendre les valeurs suivantes :

- `flex-start` : les éléments sont placés au début
- `flex-end` : les éléments sont placés à la fin
- `center` : les éléments sont placés au centre
- `space-between` : les éléments sont séparés avec de l'espace entre eux
- `space-around` : idem, mais il y a aussi de l'espace au début et à la fin
- `stretch` (par défaut) : les éléments s'étirent pour occuper tout l'espace

# 3. Flexbox Layout

Il existe également des propriétés pour gérer la mise en page des éléments fils du conteneur

Quelques exemples ci-après

- order
- flex-grow
- flex-shrink
- flex-basis
- flex (réunit les 3 ci-dessus à la fois)
- align-self

Pour plus de détails consulter:

[https://www.w3schools.com/cSS/css3\\_flexbox.asp](https://www.w3schools.com/cSS/css3_flexbox.asp)

## 4. TP

- *Maintenant, assez de théorie, vous méritez une pause !*
- *Voici donc un petit jeu pour consolider vos connaissances avant de reprendre votre TP2*

<https://flexboxfroggy.com/#fr>

