# R102 – TP5: Mise en page et positionnement

> Le but de ce TP est d'introduire le placement d'éléments par la méthode des boites flexibles (aka *flexbox*).

*Flexbox* est une méthode de mise en page selon un axe principal et permettant de disposer les éléments enfants d'un parent selon cet axe (horizontalement et de gauche à droite par défaut). Les éléments à positionner les uns par rapport aux autres peuvent se dilater ou se contracter pour occuper ou non l'espace disponible.

**Vocabulaire** : Le parent est appelé le *conteneur*. Les enfants sont des *items*.

# Ordre du jour



---

# 1. Découvrons les possibilités Flex par le jeu

*Faites au moins l'un des jeux proposés ici*

On trouve sur Internet quelques petites activités ludiques en accès libre qui permettent de prendre en main la méthode de placement flexible. Amusez-vous mais rappelez-vous que cela fait partie de l'apprentissage.

- **Pour les débutants qui ont des difficultés avec l'anglais** : https://flexboxfroggy.com
- **Pour les débutants** : http://www.flexboxdefense.com/ (baisser le son si nécessaire)
- **Pour ceux qui connaissent déjà un peu Flexbox** : Faire les niveaux moyens ou difficiles de ce jeu : https://codingfantasy.com/games/flexboxadventure/play

Les exercices ludiques vous dirige de manière assez précise et vous font tester les différentes propriétés une par une. C'est suffisant pour comprendre les effets de ces propriétés, mais il faut

aussi savoir les choisir sans être aidé.

Si les activités ludiques n'ont pas suffi à vous faire comprendre quoi utiliser et quand, cette page peut vous aider : https://developer.mozilla.org/fr/docs/Learn/CSS/CSS_layout/Flexbox.

# 2. Préparatif

*Comme d'habitude, il faut (faites-le) :*

- Créer les dossiers associés à ce TP :

```
prompt$ mkdir -p ~/public_html/r102/tp05/{css,img}
```

- Récupérez sur Moodle les ressources de ce tp et placez les correctement dans les dossiers de ce TP.

  Le fichier `positionnement.html` contient des blocks semblables contenant chacun 3 éléments enfants : 2 liens (balise `<a>`) sous forme d'image et un titre (balise `<h1>`). Une balise `<style>` a été utilisée dans l'en-tête de page (balise `<head>`) pour créer un design plus *agréable* sur ces blocks principaux. Vous n'avez donc rien à faire sur le design visuel, nous allons nous concentrer sur le placement des éléments.

Dans la suite, vous allez devoir cibler chacun des 4 blocks principaux pour placer du mieux possible leurs éléments enfants en utilisant à chaque fois une méthode différente.

Sauf peut-être pour la bordure rouge des logos, le résultat final devra ressembler à ceci :


*Figure 1. Résultat final des méthodes de positionnement classiques*

Cette bordure rouge est présente uniquement pour vous permettre de mieux visualiser certains effets de vos règles css.

Pour obtenir ce visuel, réalisez les étapes qui suivent.

## 2.1. Comparaison `inline` versus `inline-block`

**Question préliminaire** : Parmi les éléments enfants des blocks `header`, quel est l'élément qui a un comportement de type block ?

*Faisons un petit test*

1. Utiliser la propriété `display: inline;` sur ces blocks afin de changer leur comportement en block *en ligne*.
2. Changez la valeur `inline` en `inline-block` : Quelle différence constatez-vous ?
3. Laquelle de ces 2 valeurs vous semble la plus indiquée pour notre exercice ?

## 2.2. Alignement horizontal

Pour chacun des 3 premiers blocs principaux (balises `<header>`), créez les règles css en suivant les consignes de placement indiquées ci-dessous pour chacun des 3 premiers blocs. Cela va vous faire utiliser une méthode de placement horizontale différente pour chaque bloc.

| | |
|---|---|
| **Bloc 1** | *Positionnement classique* : Le titre est centré horizontalement et les logos sont étalées au moyen des marges droite et gauche. |
| **BLoc 2** | *Positionnement relatif* : Le titre est centré horizontalement, les logos sont déplacés au moyen des propriétés `position` conjointement avec `left` ou `right`. |
| **Bloc 3** | *Positionnement flottant* : Le titre est centré horizontalement et les logos utilisent la propriété `float` et la marge droite ou gauche (selon la cible) pour espacer les logos du bord de la boite englobante, et la marge haute pour les centrer verticalement. |
| **Remarque** | Pour le bloc 4, il n'y a rien à faire pour le moment, nous le modifierons un peu plus loin. |

## 2.3. Alignement vertical

Nous allons maintenant centrer verticalement les éléments de nos bandeaux d'en-tête.

- Afin de mieux visualiser si vos propriétés permettent effectivement le centrage vertical ou pas, vous allez agrandir la hauteur des en-têtes en écrivant le code suivant dans votre css :

```
header { height: 150px; }
```

*Indications*

Dans certains cas, il va falloir utiliser avec la propriété `vertical-align` vu (normalement) dans un TP précédent. Quand cette propriété n'est pas utilisable, il faudra alors utiliser les mêmes astuces que pour le positionnement horizontal vu plus haut : la hauteur (height), les marges extérieures (`margin`) et intérieures (`padding` *qui évite la fusion des marges entre un élément parent et ses enfants*) ou

encore la propriété `position`. Mais bien sûr, n'oubliez pas que cela revient souvent à effectuer un alignement approximatif.

**Bloc 1** Positionnement classique : Le centrage vertical pourra s'effectuer au moyen de propriété `vertical-align` sur les éléments enfants, conjointement avec les propriétés `height` et `padding-block` (ou padding-top et -bottom) sur le parent afin de régler au mieux l'emplacement vertical des enfants tout en conservant une hauteur de 150 px pour le parent.

**BLoc 2** Positionnement relatif : La propriété `position` étant déjà utilisée, le centrage va se faire plus simplement avec la propriété `top` ou `bottom` qui permettra déplacement vertical des éléments ciblés.

**Bloc 3** Positionnement flottant : Le positionnement ne pourra pas se faire avec `vertical-algin` car les images sont ici des éléments flottants, donc ils sont hors du flot standard. Une solution sera d'utiliser `margin-top` pour déplacer ces éléments et utiliser au choix la méthode du bloc 1 ou 2 pour centrer la balise `<h1>`.

**Bloc 4** Voir la section suivante.

# 3. Positionnement flexible

Vous avez vu sur les jeux et dans un TP précédent que la propriété `display` permet de définir le comportement d'une boîte dans son environnement, mais la méthode de placement de ses enfants.

C'est la valeur `flex` ou `inline-flex` qui va nous intéresser ici.

## 3.1. Dernier bloc d'en-tête

Utilisez la méthode *flexbox* sur le dernier bloc d'en-tête (classe `quatre`) pour réaliser l'étalement horizontal uniforme et l'alignement verticale du contenu de ce bloc.

*Aucune difficulté ici :*

Un seul sélecteur contenant seulement trois propriétés sera suffisant.

## 3.2. Allons un peu plus loin avec Flex

Récupérer la page html `langProg.html` et créer la feuille css manquante.

- Utilisez la méthode *flexible* pour centrer verticalement et étaler horizontalement les éléments de l'en-tête de cette page.

Nous allons maintenant nous concentrer sur le contenu principal de la page (balise `<main>`). Celle-ci est composée d'une série de sections contenant chacune un logo avec une bordure et flottant sur le coté gauche de sa section, et au moins deux paragraphes.

- Remplissez votre feuille de styles en utilisant la méthode des boîtes flexibles pour étaler les sections de manière à obtenir un affichage comme sur l'ensemble des captures ci-dessous. Ces

captures sont réalisées avec plusieurs largeurs différentes afin de vous montrer l'effet sur l'alignement des sections ainsi que l'affichage lors du passage de la souris sur une section.

*Indice :*

- Il faudra utiliser des propriétés de la méthode flexible spécifique aux enfants (et pas seulement globale)
- Seul le premier paragraphe doit être visible.
- Les paragraphes suivants seront visibles lorsque la souris se trouve sur la section
- Au passage de la souris sur une section :
  1. La couleur de la bordure de la section et du logo devient noire,
  2. Bonus : Effectuer une rotation complète du logo
  3. Bonus ++ : Remplacer la rotation complète précédente par une rotation d'un demi-tour avec un rétrécissement de 50 % du logo, puis retour à l'état initial. L'enseignant pourra vous montrer cette animation.

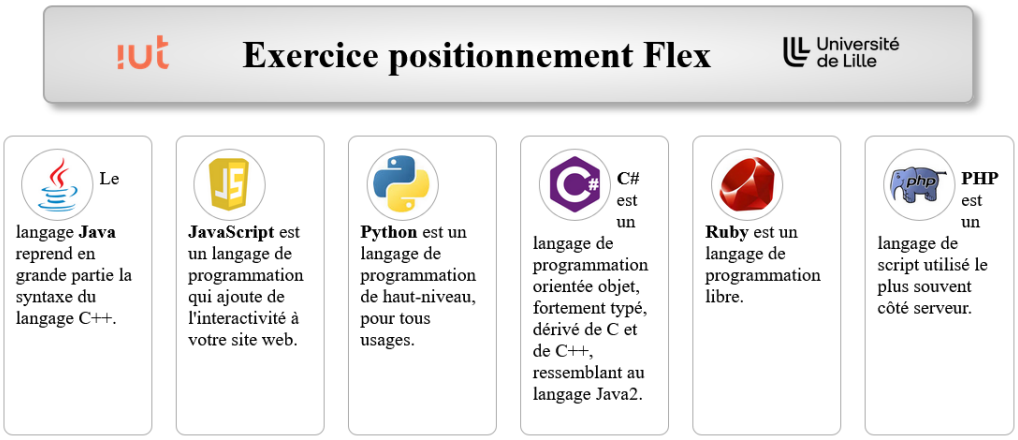


*Figure 2. Largeur fenêtre >= 900 pixels*

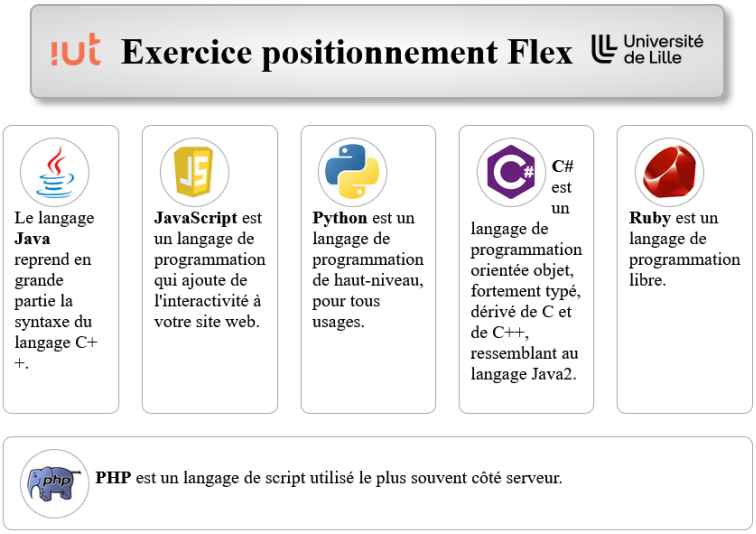


*Figure 3. Largeur fenêtre = 700 pixels*

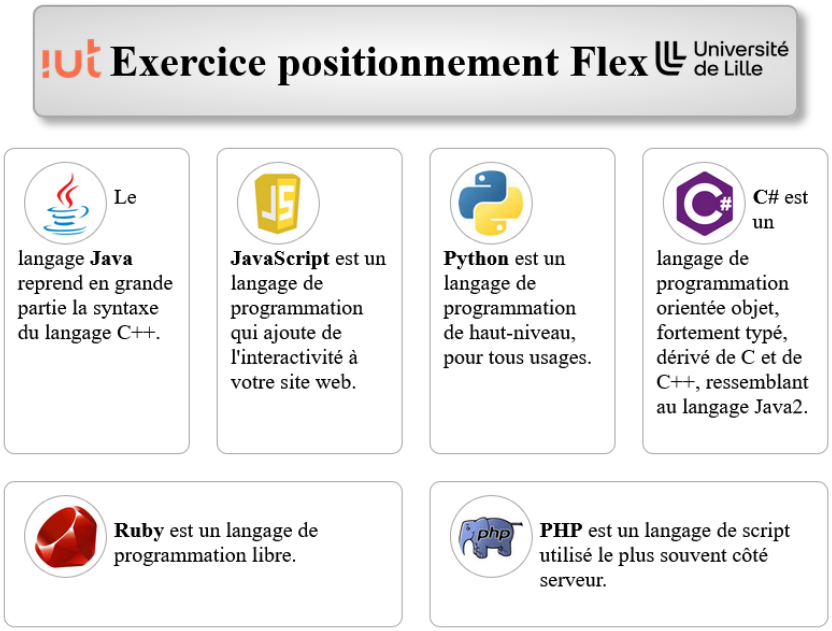


*Figure 4. Largeur fenêtre = 650 pixels*


*Figure 5. Largeur fenêtre <= 400 pixels*

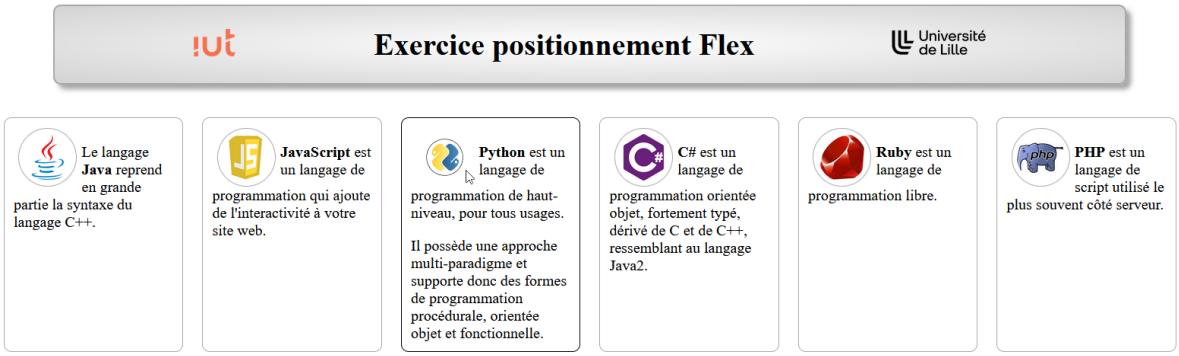


*Figure 6. Passage souris sur la section* **Python**

# 4. Conclusion

Comme vous avez pu le constater, la méthode de positionnement flexible est très simple d'utilisation et permet d'aligner des éléments enfants (appelés *items*) au sein d'un conteneur (= le parent) le long d'un axe principal, horizontalement et ordonnés de gauche à droite par défaut. La première étape est de définir le conteneur, c'est à dire le parent des éléments à aligner, avec la propriété `display`

Une fois ceci fait, il est possible de déterminer un comportement global directement dans le conteneur et si nécessaire, de définir sur un ou plusieurs items un comportement spécifique.

Néanmoins, ce n'est pas la méthode d'alignement ultime. Il est parfois utile d'utiliser des méthodes de placements plus ancienne, par exemple dans le cas d'habillage d'image par du texte. Il existe également d'autres méthode de placement