TP N°2

Développement d'applications web

HTML ET CSS

Préparée par : RGUIBI Farah

Objectif :

Ce travail pratique vise à maîtriser les fondamentaux de la mise en page moderne avec **CSS Flexbox**. À travers trois exercices complémentaires, nous explorons la création de layouts complexes et de composants de navigation flexibles et responsives. L'objectif est de développer une compréhension approfondie du système Flexbox pour créer des interfaces web structurées, adaptatives et professionnelles, en appliquant les standards modernes du CSS.

FIGURE 1 – Html et css

1 Les exercices :

1.1 Exercice 1 :

1.2 partie 1 : Layout avec 3 Sections :

1.2.1 Idée de l'exercice

L'idée principale de cet exercice est de comprendre comment créer une mise en page à plusieurs niveaux en utilisant Flexbox. Cela permet de structurer une page web en sections verticales et horizontales, avec une répartition proportionnelle de l'espace. Cette technique est fondamentale pour créer des layouts modernes sans recourir aux anciennes méthodes de positionnement.

1.2.2 Description

Dans cet exercice, j'ai créé un layout HTML de 3 sections principales avec Flexbox :

- **Section supérieure (Bleu)** : Une barre qui occupe 25% de la hauteur totale avec `height: 25%`
- **Section centrale (Row)** : Une zone divisée en 2 colonnes occupant 50% de la hauteur :
  - Colonne gauche **Orange** : 30% de la largeur
  - Colonne droite **Blanche** : 70% de la largeur
  - Disposition horizontale avec `display: flex`
- **Section inférieure (Vert)** : Une barre qui occupe 25% de la hauteur

Le container principal utilise `display: flex` sur le body pour centrer le contenu verticalement et horizontalement avec `justify-content: center` et `align-items: center`.

1.2.3 Résultat

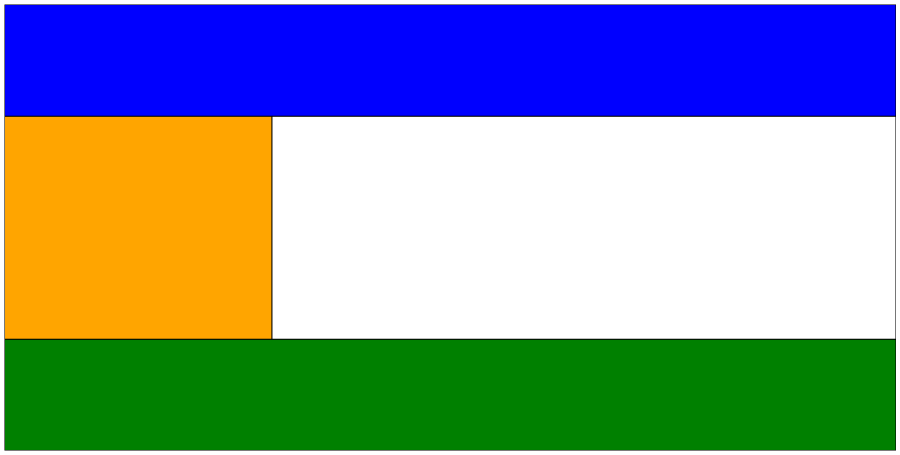

FIGURE 2 – Layout avec 3 Sections

1.3 partie 2 : Layout avec 4 Sections :

1.3.1 Idée de l'exercice

L'objectif est d'approfondir la maîtrise de Flexbox en créant un layout plus complexe avec une section centrale divisée en **trois colonnes** au lieu de deux.

1.3.2 Description

J'ai développé un layout similaire à l'exercice 1, mais avec une complexité supplémentaire dans la section centrale :

- **Section supérieure (Bleu)** : 25% de hauteur
- **Section centrale (Row)** : 50% de hauteur divisée en **3 colonnes** :
  - Colonne gauche **Orange** : 30% de largeur
  - Colonne centrale **Blanche** : 40% de largeur
  - Colonne droite **Orange** : 30% de largeur
- **Section inférieure (Vert)** : 25% de hauteur

La disposition utilise le même principe Flexbox avec `display: flex` sur la classe `.row` pour aligner les trois colonnes horizontalement, créant ainsi une symétrie parfaite.

1.3.3 Résultat

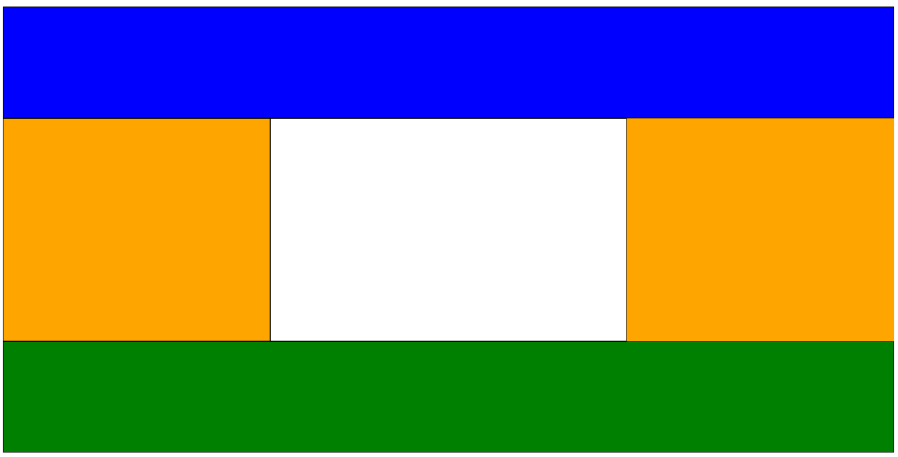

FIGURE 3 – Layout avec 4 Sections

1.4 Exercice 2 :

1.5 partie 1 : Menu de Navigation Horizontal :

1.5.1 Idée de l'exercice

Cet exercice vise à créer un menu de navigation moderne et responsive en utilisant Flexbox. L'objectif est de comprendre comment aligner des éléments horizontalement, les répartir équitablement sur toute la largeur disponible, et ajouter des effets interactifs au survol. Cette technique est essentielle pour créer des barres de navigation professionnelles sans recourir aux anciennes méthodes de floats ou de display inline-block.

1.5.2 Description

Dans cet exercice, j'ai créé un menu de navigation horizontal avec 5 éléments : **Accueil**, **Produits**, **Services**, **Equipe** et **Contact**. Le menu utilise les propriétés Flexbox suivantes :

- `display: flex` sur l'élément `<ul>` pour activer Flexbox
- `flex: 1` sur chaque `<li>` pour une distribution équitable de l'espace
- `justify-content: center` et `align-items: center` pour centrer le texte dans chaque élément
- Suppression du style de liste par défaut avec `list-style-type: none`

1.5.3 Résultat

Le menu s'affiche horizontalement avec 5 éléments répartis équitablement sur toute la largeur. Chaque élément a un fond vert, des bordures noires, et le texte est centré verticalement et horizontalement. Les éléments ont la même largeur grâce à `flex: 1`.

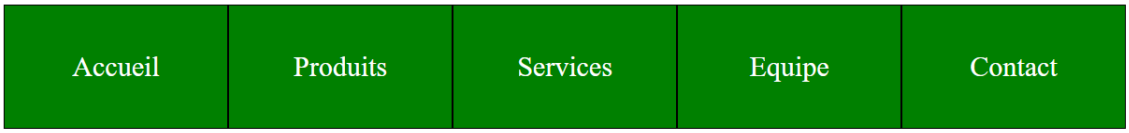

FIGURE 4 – Menu de navigation horizontal

1.6 partie 2 : Menu de Navigation Vertical :

1.6.1 Idée de l'exercice

Cet exercice vise à créer un menu de navigation vertical, idéal pour les sidebars ou les menus latéraux. L'objectif est de comprendre comment modifier l'orientation d'un layout Flexbox en changeant simplement la propriété `flex-direction`. Cette technique démontre la flexibilité et la puissance de Flexbox pour créer différentes mises en page avec un minimum de modifications CSS.

1.6.2 Description

J'ai développé un menu vertical avec les mêmes 5 éléments que l'exercice 3. La principale différence réside dans l'utilisation de :

- `flex-direction: column` pour empiler les éléments verticalement au lieu de horizontalement
- `width: 300px` pour une largeur fixe adaptée à un menu latéral
- Positionnement naturel des éléments les uns au-dessus des autres

Cette configuration est couramment utilisée pour créer des menus latéraux dans les applications web modernes, les dashboards administratifs, ou les interfaces avec navigation principale sur le côté.

1.6.3 Résultat

Le menu s'affiche verticalement avec les 5 éléments empilés les uns au-dessus des autres. Chaque élément occupe la même hauteur grâce à `flex: 1`, et le menu a une largeur fixe de 300px, parfaite pour une sidebar.

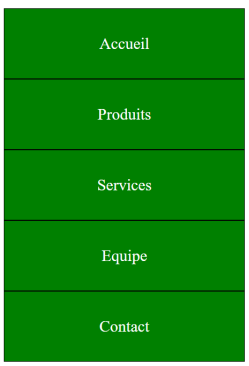

FIGURE 5 – Menu de Navigation Vertical

1.7 Exercice 3 :

1.8 partie 1 : z-index et Empilement de Carrés :

1.8.1 Idée de l'exercice

L'idée principale de cet exercice est de comprendre la propriété CSS `z-index` qui contrôle l'ordre d'empilement des éléments positionnés sur une page web. Cette propriété est essentielle pour gérer la superposition des éléments et créer des effets visuels en couches. Elle nécessite que les éléments aient une propriété `position` autre que `static` (comme `absolute`, `relative` ou `fixed`).

1.8.2 Description

Dans cet exercice, j'ai créé quatre carrés colorés qui se chevauchent avec des ordres d'empilement différents :

- **Structure HTML** : Un container relatif contenant 4 divs avec la classe `.square`
- **Positionnement absolu** : Tous les carrés utilisent `position: absolute` pour pouvoir se superposer
- **Dimensions** : Chaque carré a une taille de 280px × 280px avec une bordure de 25px
- **Ordre d'empilement avec z-index** :
  - Carré **Rouge** : `z-index: 4` (en haut) - Position : `top: 50px; left: 30px`
  - Carré **Bleu** : `z-index: 3` - Position : `top: 110px; left: 90px`
  - Carré **Vert** : `z-index: 2` - Position : `top: 50px; left: 90px`
  - Carré **Jaune/Orange** : `z-index: 1` (en bas) - Position : `top: 110px; left: 30px`

Le container principal a une taille fixe de 400px × 400px avec `position: relative` pour servir de référence au positionnement absolu des carrés enfants. Les valeurs de `z-index` déterminent quel carré apparaît au-dessus des autres lors du chevauchement.

1.8.3 Résultat

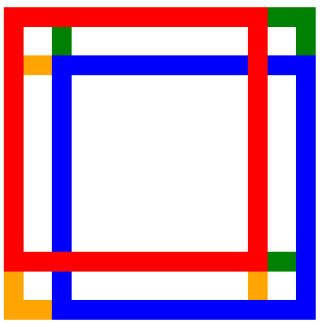

FIGURE 6 – Empilement de carrés avec z-index

1.9 partie 2 : Galerie d'Équipe avec Flexbox :

1.9.1 Idée de l'exercice

L'objectif de cet exercice est de créer une galerie responsive d'une équipe de direction en utilisant Flexbox. Cette technique permet d'afficher plusieurs cartes de profil qui s'adaptent automatiquement à la largeur de l'écran grâce à flex-wrap. C'est une application pratique couramment utilisée dans les pages "À propos" ou "Notre équipe" des sites web professionnels.

1.9.2 Description

J'ai développé une galerie de 8 membres de l'équipe avec les caractéristiques suivantes :

- **Container principal** : Utilise display : flex avec flex-wrap : wrap pour permettre aux éléments de passer à la ligne suivante automatiquement
- **Distribution** : justify-content : space-evenly pour espacer uniformément les cartes
- **Cartes de profil** : Chaque élément occupe 20% de la largeur du container avec width : 20%
- **Images** :
  - Hauteur fixe de 300px pour uniformiser toutes les photos
  - object-fit : cover pour conserver les proportions sans déformation
  - border-radius : 10px pour des coins arrondis
- **Informations** :
  - Nom en gras dans un div avec padding : 15px
  - Poste en petit texte (font-size : 12px) en couleur grise (color : #666)

Cette mise en page est responsive : les cartes s'ajustent automatiquement grâce à Flexbox, permettant d'afficher 4 cartes par ligne sur grand écran et moins sur petits écrans.

1.9.3 Résultat

La galerie affiche 8 membres de l'équipe de direction avec leurs photos, noms et postes. Les cartes sont espacées uniformément et ont toutes la même hauteur d'image (300px), créant une présentation visuelle cohérente et professionnelle.

FIGURE 7 – Galerie d'équipe avec Flexbox

2 Conclusion

Ce travail pratique m'a permis de maîtriser les concepts fondamentaux de **CSS Flexbox** à travers quatre exercices progressifs.

2.1 Compétences Acquises

Layouts complexes : Création de mises en page à plusieurs niveaux avec sections horizontales et verticales

Navigation moderne : Développement de menus responsives horizontaux et verticaux

Distribution d'espace : Maîtrise de `flex: 1` pour répartir équitablement l'espace

Flexibilité : Compréhension de `flex-direction` pour changer l'orientation facilement

Centrage : Utilisation de `justify-content` et `align-items` pour aligner les éléments