**Flexbox et grid**

Abordons deux notions des plus importante que vous utiliserez certainement toute votre vie de développeur. Le flexbox et le grid. Commençons par le flexbox. Le module CSS Flexbox Layout a été introduit pour faciliter la conception d'une mise en page réactive et flexible sans utiliser de flotteur ou de positionnement.

Pour faire fonctionner la méthode CSS flex, entourez la balise de votre choix mais prenons en exemple une <div> avec un autre <div> et donne le statut de conteneur flexible.

Les div se positionne l’une en dessous de l’autre cela est du leur type block, nous utiliserons flexbox afin de pouvoir par exemple les aligner.

.mycontainer {  
  display: flex;  
}  
.mycontainer > div {  
  width:33%;  
}

Nous aurons nos containers donc les div l’une a cote de l’autre. Nous aurons acces a de nouvelles propriété tels que justify-content et align-items qui disposent eux aussi de valeurs diff »rentes voici une liste des plus couramment utilisés :

* Flex-start : Valeur par défaut. Les éléments sont positionnés au début du conteneur
* Flex-end : Les éléments sont positionnés à la fin du conteneur
* Center : Les éléments sont positionnés au centre du conteneur
* Space-between : Les éléments auront un espace entre eux
* Space-around : Les éléments auront un espace avant, entre et après eux
* Space-evenly : Les éléments auront un espace égal autour d’eux

div {  
  display: flex;  
  justify-content: flex-start;  
}

Passons au **grid** désormais, le module CSS Grid Layout offre un système de mise en page basé sur une grille, avec des lignes et des colonnes, ce qui facilite la conception de pages Web sans avoir à utiliser de flotteurs ni de positionnement.

Cela ressemble presque à flex, mais avec la capacité de définir plus d'une ligne et de positionner chaque ligne individuellement.

La méthode de grille CSS nécessite d'entourer l’élément <div> avec un autre <div> et donner le statut de conteneur de grille et vous devez spécifier la largeur de chaque colonne.

.grid-container {  
  display: grid;  
  grid-template-columns: 33% 33% 33%;  
}

Faites un test des deux et voyez la différence.

Voyons désormais l’attribut **class** afin e cibler un élément pour lui apporter un style bien précis ou agir plus précisément sur une balise. L’attribut class est souvent utilisé pour pointer à un nom de classe dans une feuille de style. Il peut également être utilisé par un JavaScript pour accéder et manipuler des éléments avec le nom de classe spécifique.

<div class="city">  
  <h2>Paris</h2>  
  <p>Paris is the capital of France.</p>  
</div>

Nous avons donc créé la classe city nous allons désormais la rappelé en Css afin d’appliquer des propriétés exclusivement a la balise qui contient cette classe, une même classe peut être appliqué a plusieurs éléments.

.city {  
  background-color: tomato;  
  color: white;  
  border: 2px solid black;  
  margin: 20px;  
  padding: 20px;  
}

Abordons l’attribut **id,** l'attribut id spécifie un identifiant unique pour un élément HTML. La valeur de l’attribut id doit être unique dans le document HTML.

L'attribut id est utilisé pour pointer à une déclaration de style spécifique dans une feuille de style.

<h1 id="myTitle">Mon titre</h1>

On le rappel en Css en utilisant le caractère # afin de pointer dessus.

#myTitle {  
  background-color: lightblue;  
  color: black;  
  padding: 40px;  
  text-align: center;  
}

**Les Formulaires**

Un formulaire HTML est utilisé pour collecter les entrées des utilisateurs. L'entrée de l'utilisateur est le plus souvent envoyé à un serveur pour traitement. Pour initialiser un formulaire nous utiliserons la balise <form></form>.

L’élément <form> est un conteneur pour différents types d'éléments d'entrée, tels que : champs de texte, cases à cocher, radio boutons, boutons de soumission, etc.

* <input type="text"> Affiche un champ de saisie de texte sur une ligne
* <input type="radio"> Affiche un bouton (choix multiple)
* <input type="checkbox"> Affiche une case à cocher (choix unique)
* <input type="submit"> Affiche un bouton de soumission de formulaire
* <input type="button"> Affiche un bouton cliquable(utilisé avec Javascript)

L’élément <label> définit une étiquette pour plusieurs éléments de forme.

L'élément <label> est utile pour les utilisateurs de lecteurs d'écran, car le lecteur d'écran lira à haute voix l'étiquette lorsque l'utilisateur se concentre sur l'élément d'entrée.

L'élément <label> aide également les utilisateurs qui ont difficulté à cliquer sur de très petites régions (comme des boutons radio ou des cases à cocher) - parce que lorsque l'utilisateur clique sur le texte dans le <label>, il bascule le bouton radio/la case à cocher.

L’attribut for du <label> devrait être égal à l’ ttribut id du <input> pour les lier ensemble.

<input type="radio" id="html" name="fav\_language" value="HTML">  
  <label for="html">HTML</label><br>

Le <input type="submit">définit un bouton pour soumettre les données du formulaire à un gestionnaire de formulaire.

Le gestionnaire de formulaire est généralement un fichier sur le serveur avec un script pour le traitement des données d'entrée.

Le gestionnaire de formulaire est spécifié dans le fichier du formulaire par l’attribut action, et la method.

<form action="url back " method= “get ou post”>

L'attribut method spécifie la méthode HTTP à utiliser lors de la soumission des données du formulaire.

Les données du formulaire peuvent être envoyées sous forme de variables URL (avec method="get") ou sous forme de post-transaction HTTP (avec method="post").

La méthode HTTP par défaut lors de la soumission des données du formulaire est GET.

Le HTML <form>L'élément peut contenir un ou plusieurs des éléments de formulaire suivants :

* <input>
* <label>
* <select>
* <textarea>
* <button>
* <fieldset>
* <legend>
* <datalist>
* <output>
* <option>

Dans les input il y a également une liste de types différents les voici :

* <input type="button">
* <input type="checkbox">
* <input type="color">
* <input type="date">
* <input type="datetime-local">
* <input type="email">
* <input type="file">
* <input type="hidden">
* <input type="image">
* <input type="month">
* <input type="number">
* <input type="password">
* <input type="radio">
* <input type="range">
* <input type="reset">
* <input type="search">
* <input type="submit">
* <input type="tel">
* <input type="text">
* <input type="time">
* <input type="url">
* <input type="week">

Ils disposent également d’une liste d’attributs conséquentes que l’on peut consulter sur la documentation officielle selon nos besoins.

Il ne reste plus qu’a aborder le dernier point et pas des moindres la partie responsive ce qui permet de lire notre site web sur différent support tels que l’ordinateur(desktop), tablette ou mobile on utilise pour cela les média query. Il y a une balise obligatoire a ajouter a votre **head** pour que les media query soit pris en compte voici cette ligne.

<meta name="viewport" content="width=device-width, initial-scale=1.0">

Cela définira la fenêtre d'affichage de votre page, qui donnera au navigateur des instructions sur la façon dont pour contrôler les dimensions et la mise à l'échelle de la page.

Ensuite rendons nous sur notre fichier Css et créons les media query afin de rendre responsive notre site web.

@media screen and (max-width: 800px) {  
  .left, .main, .right {  
    width: 100%; /\* La width sera 100%, quand l’écran sera de 800px ou moins \*/  
  }

Vous voilà désormais fin prêt a créer votre propre site web qui pourra être lu sur tous types de support. Passons au premier gros TP.