

# Rapport de TP : Responsive Design

## Sommaire :

I. Introduction	1
II. Le travail à réaliser	1
III. Conclusion	10

## I. Introduction

Dans le cadre de mon BTS SIO, j'ai réalisé un TP portant sur le **Responsive Design**, une notion essentielle dans le développement web moderne. L'objectif de ce travail était de comprendre comment adapter une page web à différents types d'écrans, en utilisant principalement les propriétés CSS telles que les largeurs flexibles, les valeurs minimales et maximales, ainsi que le modèle Flexbox. Au fil des exercices, j'ai progressivement appris à ajuster la mise en page, structurer les éléments selon l'espace disponible, manipuler les ratios et assurer un affichage cohérent sur ordinateur, tablette ou smartphone. Ce rapport présente donc l'ensemble des manipulations effectuées, les codes utilisés ainsi que les résultats obtenus tout au long du TP.

## II. Le travail à réaliser

### Exercice 1 :

Pour effectuer cet exercice, nous allons utiliser la page HTML de Chuck Norris réalisé précédemment :

1) Nous allons changer la valeur de la largeur width pour la mettre à 100% puis nous allons rajouter l'attribut auto à margin en utilisant ce code :

```
body{
background-color:#838892;
width:100%;
line-height:150%;
text-indent:5px;
text-align: center;
margin: auto;
-}
```

2) Puis nous allons modifier le <aside> afin de lui retirer la marge de gauche  
de 10% et lui changer sa largeur de 30% à 33% et le <article> pour lui  
changer sa largeur De 60% à 67% :

```
article {
width:67%;
background-color:#CCC;
}
aside {
width:33%;
background-color:#CCC;
}
```

## Exercice 2 :

1) Maintenant, nous allons ajouter une largeur maximum de 500px à  
<article> et de 250px à <aside> en utilisant la propriété CSS max-width :

```
article {
width:67%;
background-color:#CCC;
max-width:500px;
-}
aside {
width:33%;
background-color:#CCC;
max-width:250px;
-}
```


2) Puis nous allons faire la même chose mais avec une limite minimum  
(min-width) de 250px pour <article> et 150px pour <aside> :

```

article {
  width:67%;
  background-color:#CCC;
  max-width:500px;
  min-width:200px;
}
aside {
  width:33%;
  background-color:#CCC;
  max-width:250px;
  min-width:150px;
}

```

Résultat :

Chuck Norris						
<i>Biographie (From Wikipedia)</i>  <i>L'enfance</i>   <p>Carlos Ray Norris, plus connu en tant que Chuck Norris, est un acteur américain né le 10 mars 1940 à Ryan (Oklahoma).</p>	Acteurs	Karaté	Taekwondo	Judo	Chun Kuk Do	Tangsudo Ju-jitsu
	Chuck Norris	5	5	5	5	5
	Steven Seagal	3	5	3	2	3
	Bruce Lee	5	3	3	4	3
	Jean-Claude Van Damme	3	5	3	3	4
	Bolo Yeung	2	4	2	4	2

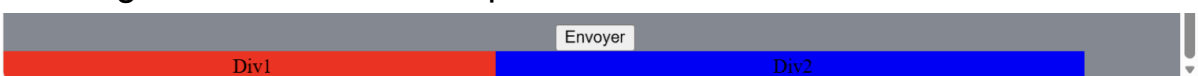
### Exercice 3 :

Nous allons maintenant ce que fait le mélange des propriétés min-width et max-width avec les tailles en pourcentages :

1) Lorsqu'on met la page à 800px ou moins div1 et div2 prennent toute la largeur :



2) Cependant, lorsque la page dépasse cette taille div1 va rester bloquer à sa largeur maximum de 400px :



Voici le code que j'ai utilisé :

```
<div style="display: flex;">
  <div style="width: 50%; max-width: 400px; background-color: red;">Div1</div>
  <div style="width: 50%; background-color: blue;">Div2</div>
</div>
```

#### Exercice 4 :

1) Je vais utiliser ce site pour m'aider :

[https://www.w3schools.com/css/css3\\_flexbox\\_items.asp](https://www.w3schools.com/css/css3_flexbox_items.asp)

2) Nous allons maintenant utiliser la propriété CSS flex-grow pour résoudre le problème rencontré à l'exercice 3 :

```
<div style="display: flex;">
  <div style="width: 50%; max-width: 400px; background-color: red;">Div1</div>
  <div style="width: 50%; background-color: blue; flex-grow:1;">Div2</div>
</div>
```

Résultat :




flex-grow:1 prend le dessus sur width:50% car il permet d'occuper tout l'espace restant de l'espace vide sur la largeur de la page.

3) Maintenant que nous savons cela nous allons changer la largeur de <article> et <aside> pour les mettre respectivement à 300px et 200px. Puis nous allons rajouter les propriétés CSS flex-grow et flex-shrink qui seront de valeur 0 dans le code :

```
article {
  width:300px;
  background-color:#CCC;
  max-width:500px;
  min-width:200px;
  display:flex;
  flex-grow:0;
  flex-shrink:0;
}
aside {
  width:200px;
  background-color:#CCC;
  max-width:250px;
  min-width:150px;
  display:flex;
  flex-grow:0;
  flex-shrink:0;
}
```

Voici le résultat :

Biographie From Wikipédia		Acteurs	Karaté	Taekwondo	Judo
L'enfance					
 <p>Carlos Ray Norris, que Chuck Norris américain né le 10 (Oklahoma)</p>		Chuck Norris	5	5	5
Spécialiste en arts martiaux. il s'est essentiellement					

En bougeant la page on s'aperçoit que rien ne change, car la valeur de flex-grow et flex-shrink sont à 0.

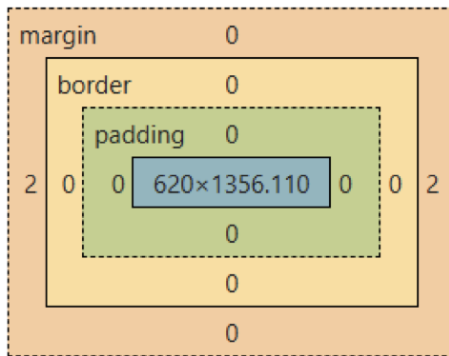
4) Pour que <article> soit 3 fois plus grand que <aside> il faut que la valeur de son flex-grow soit 3 fois plus grande c'est-à-dire qu'il faut changer la valeur de <aside> à 1 et changer la valeur de <article> à 3 :

```
<div style="display: flex; width: 95%; margin: 20px auto; height: 100px; border: 2px solid black;">
  <div style="flex: 3 0 0; background-color: #5BBDBF; text-align: center; line-height: 100px;">
    DIV 1 (Article - flex-grow: 3)
  </div>
  <div style="flex: 1 0 0; background-color: #FF5850; text-align: center; line-height: 100px;">
    DIV 2 (Aside - flex-grow: 1)
  </div>
</div>
```

Voici le résultat :

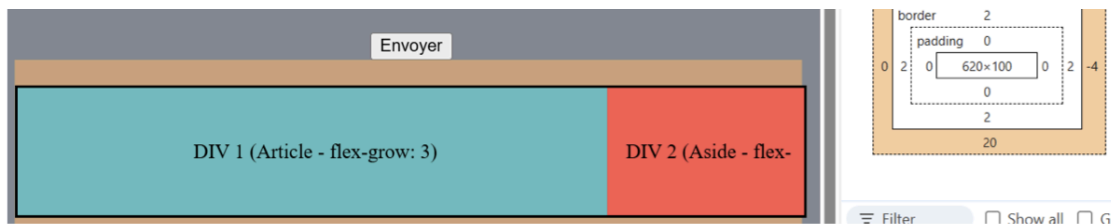
DIV 1 (Article - flex-grow: 3)	DIV 2 (Aside - flex-grow: 1)
--------------------------------	------------------------------

5) Pour faire cette question nous allons changer la taille du body dans l'inspecteur d'élément (étant donné qu'il n'y a pas de CSS dans la page HTML où j'ai créé ces deux div, ils ne remplissent pas toute la largeur mais le concept du ratio 3 : 1 est toujours présent) :



```
<div style="display: flex; width: 95%; margin: 20px auto; height: 100px; border: 2px solid black;">
  <div style="flex: 3 0 0; background-color: #5BBDBF; line-height: 100px;">
    DIV 1 (Article - flex-grow: 3)
  </div>
  <div style="flex: 1 0 0; background-color: #FF5850; line-height: 100px;">
    DIV 2 (Aside - flex-grow: 1)
  </div>
</div>
```

Nous allons donc simplement changer la largeur de l'ensemble des deux div :

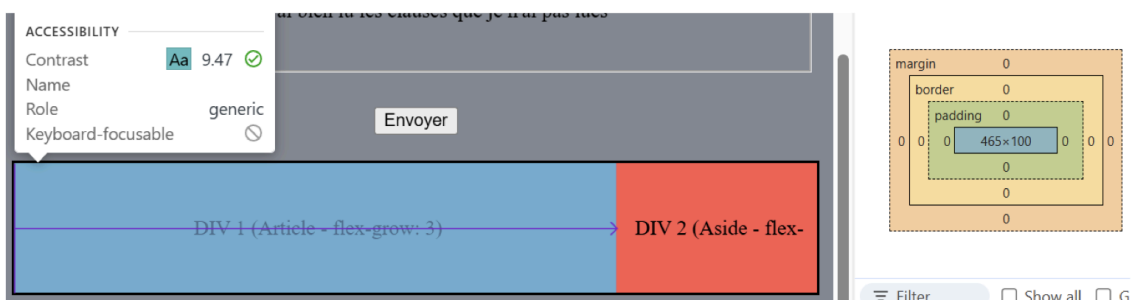


DIV 1 occupe 3/4 et DIV 2 1/4 . Donc DIV 1 doit faire  $\frac{3}{4} * 620$  et DIV 2  $\frac{1}{4} * 620$  :

DIV 1 doit donc faire : 465px

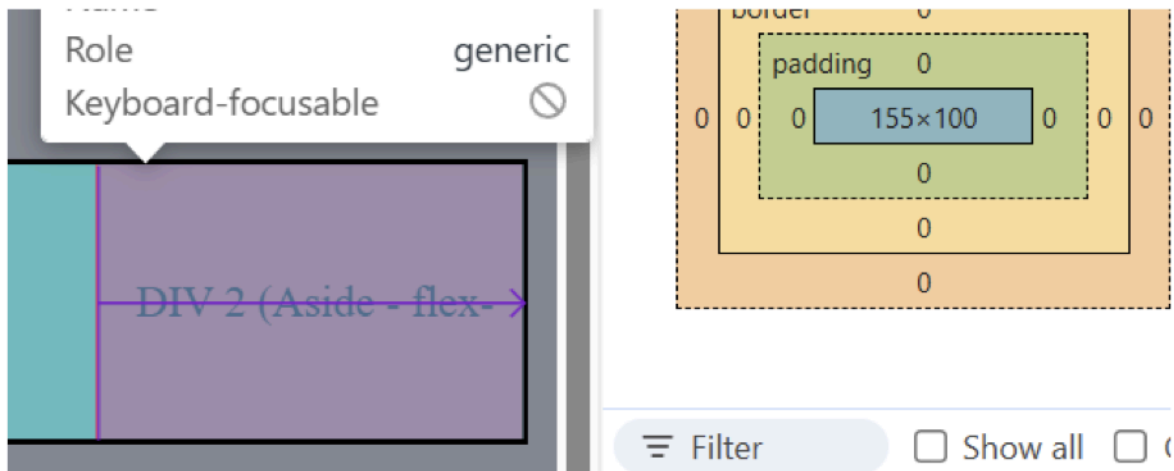
DIV 2 doit donc faire : 155px

Résultat pour DIV 1 :



C'est correct.

Résultat pour DIV 2 :



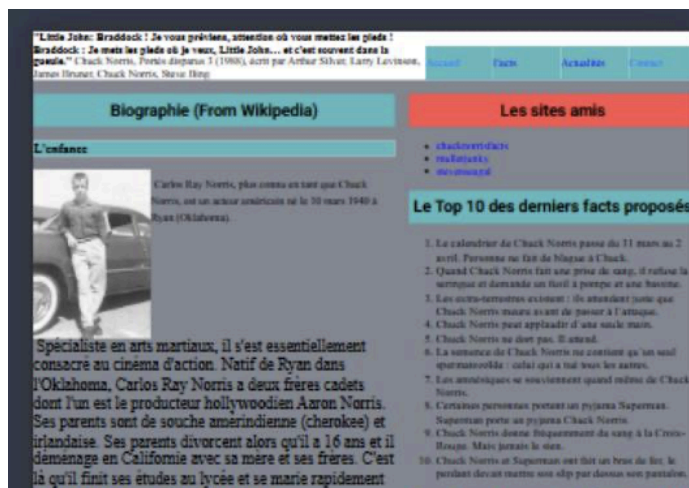
C'est également correct.

6) DIV 1 diminue des deux tiers (2/3) de la largeur à supprimer et DIV 2 diminue du reste (1/3), on doit utiliser la propriété CSS flex-shrink. Pour implémenter ce ratio de 2 : 1 sur la réduction :

```
<div style="display: flex; width: 95%; margin: 20px auto; height: 100px; border: 2px solid black;">
  <div style="flex: 3 2 0; background-color: #5BBDBF; line-height: 100px;">
    DIV 1 (Article - flex-grow: 3)
  </div>
  <div style="flex: 1 1 0; background-color: #FF5850; line-height: 100px;">
    DIV 2 (Aside - flex-grow: 1)
  </div>
</div>
```

## Exercice 5 :

1) Exemple de résultat sur téléphone :



2)



**Exercice 6 :** En ajoutant cette ligne de code :

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Le navigateur ne va plus chercher à essayer de garder la résolution de base, on demande au navigateur d'utiliser la vraie largeur de l'écran.

**Exercice 7 :**

LIEN INVALIDE

**Exercice 8 :**

1) Pour pouvoir faire cela nous devons utiliser un système de condition en utilisant and, on veut enlever le tableau donc on le fait disparaître avec none :

```
@media screen and (max-width: 768px) {  
  table {  
    display: none;  
  }  
}
```

**Exercice 9 :**

1) Nous allons créer un <div> « burger » :

```
<div class="burger">  
  
```

2) Maintenant dans le CSS nous allons cacher le menu par défaut avec display:none puis nous allons positionner l'image en haut à droite de la fenêtre :



```
.burger img {
  position: fixed;
  top: 10px;
  right: 10px;
  z-index: 1000;
  cursor: pointer;
}
```

3) Ensuite nous implémentons ce code avec d'autres choses en plus :

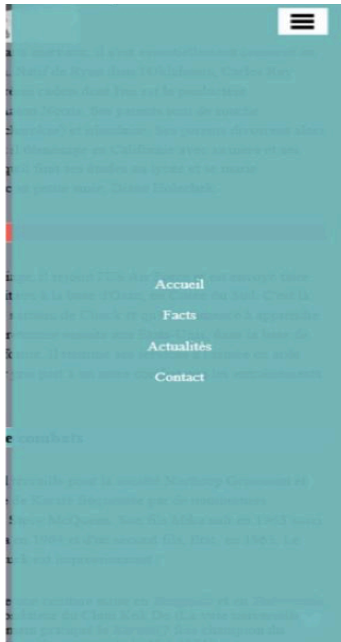
```
#menu2 {
  position: fixed;
  top: 0;
  right: 0;
  width: 80vw;
  height: 100vh;
  background-color: rgba(91, 189, 191, 0.95);
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  gap: 20px;
  z-index: 10;
  margin-right: -80vw;
}
```

```
<div class="burger">
  
  <div id="menu2">
    <div><a href="/index.html">Accueil</a></div>
    <div><a href="/facts.html">Facts</a></div>
    <div><a href="/news.html">Actualités</a></div>
    <div><a href="/contact.html">Contact</a></div>
  </div>
</div>
```

4) Maintenant nous allons faire en sorte de faire apparaître le menu lorsqu'on survole le menu :

```
.burger:hover #menu2 {
  display: flex;
}
```

Résultat :



### III. Conclusion

Ainsi, ce TP m'a permis de mieux comprendre les principes fondamentaux du **Responsive Design** et l'importance de concevoir des interfaces capables de s'adapter automatiquement à différentes résolutions. Grâce à des propriétés telles que *width*, *min-width*, *max-width*, *flex-grow* et *flex-shrink*, j'ai pu observer concrètement comment se comporte une page web lorsque la taille de la fenêtre change. L'utilisation du Flexbox a également facilité la mise en page dynamique et m'a permis de résoudre plusieurs contraintes rencontrées dans les exercices. Enfin, j'ai découvert la nécessité d'utiliser des media queries et des éléments d'interface comme le menu « burger » pour optimiser l'expérience utilisateur sur mobile. Ce TP constitue une base solide pour développer des sites modernes, ergonomiques et compatibles avec tous les types de supports. J'ai passé environ 4h car j'ai trouvé ce TP relativement compliqué.