

## **Atelier 7 TD 5**

Exercice 1	2
Exercice 2	3
Exercice 3	3
Exercice 4	4
Exercice 5	4
Exercice 6	5
Exercice 7	5
Exercice 8	5
Exercice 9	7
Conclusion	8

# 1)

J'ai initié le design en convertissant la largeur fixe du <body> en width: 100% et margin: auto. Parallèlement, j'ai réajusté les largeurs des colonnes article et aside en pourcentages (67%/33%) pour qu'elles occupent la totalité de l'espace parent.

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

aside {
    width: 33%;
    margin-left: 0;
    background-color: #CCC;
}
```

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

## 2)

J'ai introduit des contraintes dimensionnelles pour garantir la lisibilité. En appliquant max-width et min-width à article et aside, j'ai empêché le contenu de devenir illisible (lignes trop longues ou blocs trop étroits) sur les écrans aux tailles extrêmes.

```
article {  
  width: 67%;  
  background-color: #CCC;  
  max-width: 500px;  
  min-width: 200px;  
}  
  
aside {  
  width: 33%;  
  margin-left: 0;  
  background-color: #CCC;  
  max-width: 250px;  
  min-width: 150px;  
}
```

## 3)

J'ai observé le problème de "sur-contrainte". L'application d'un max-width en pixels à un élément dimensionné en pourcentage brise la mise en page fluide sur grand écran, créant un espace vide indésirable.

## 4)

J'ai implémenté Flexbox comme solution. J'ai défini des largeurs de base (width) en pixels et utilisé flex-grow (ratio 3:1) pour distribuer l'espace excédentaire. J'ai aussi utilisé flex-shrink (ratio 2:1) pour gérer la réduction proportionnelle des blocs en cas de manque d'espace.

```
article {  
    background-color: #CCC;  
    width: 300px;  
    flex-grow: 3;  
    flex-shrink: 2;  
}  
  
aside {  
    margin-left: 0;  
    background-color: #CCC;  
    width: 200px;  
    flex-grow: 1;  
    flex-shrink: 1;  
}
```

## 5)

J'ai inséré la balise <meta name="viewport"> dans le <head> HTML. Cette instruction force le navigateur mobile à utiliser la largeur réelle de l'appareil (width=device-width) et à appliquer une échelle de 1:1 (initial-scale=1), empêchant ainsi le "zoom arrière" par défaut.

```
<head>  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <link rel="stylesheet" type="text/css" href="css/styles.css">
```

## 6)

J'ai constaté l'effet du viewport. Le navigateur a cessé de dézoomer, affichant le site à 100% de sa taille. Cela a révélé un débordement horizontal, prouvant que le site n'était pas encore adapté à cette largeur.

## 7)

J'ai analysé un site professionnel (Bootstrap). J'ai observé l'utilisation d'un point de rupture (breakpoint) à 768px pour masquer la navigation principale et la remplacer par une icône "burger", une pratique standard du design responsive.

## 8)

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

@media (max-width: 480px) {
  main {
    flex-direction: column;
  }
  article {
    width: 100%;
    flex-grow: 0;
    flex-shrink: 1;
  }
  aside {
    width: 100%;
    flex-grow: 0;
    flex-shrink: 1;
  }
}

@media (max-width: 168px) {
  article {
    display: none;
  }
  header {
    display: none;
  }
  footer {
    display: none;
  }
}
```

J'ai mis des @media queries pour définir des points de rupture. À max-width: 768px, j'ai masqué le tableau (display: none). À max-width: 480px, j'ai redéfini le flex-direction du main en column pour empiler les blocs, et j'ai forcé leur largeur à 100%.

## L'enfance



**Carlos Ray Norris**, plus connu en tant que **Chuck Norris**, est un acteur américain né le 10 mars 1940 à Ryan (Oklahoma).

>Spécialiste en arts martiaux, il s'est essentiellement consacré au cinéma d'action. Natif de Ryan dans l'Oklahoma, Carlos Ray Norris a deux frères cadets dont l'un est le producteur hollywoodien Aaron Norris. Ses parents sont de souche amérindienne (cherokee) et irlandaise. Ses parents divorcent alors qu'il a 16 ans et il déménage en Californie avec sa mère et ses frères. C'est là qu'il finit ses études au lycée et se marie rapidement avec sa petite amie, Diane Holechek.

## L'armée

Après son mariage, il rejoint l'US Air Force et est envoyé faire son service militaire à la base d'Osan, en Corée du Sud. C'est là qu'il acquiert le surnom de Chuck et qu'il commence à apprendre le *Tangsudo*. Il retourne ensuite aux



## 9)

J'ai finalisé l'interface mobile en créant le menu "burger". J'ai ajouté le HTML de l'icône et du panneau de menu. En CSS, via la media query à 480px, j'ai masqué la nav, affiché l'icône, et utilisé la pseudo-classe :hover pour déclencher l'affichage (display: block) du menu (.menu2) positionné en fixed.

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

<div class="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>
```

```
.burger {
  display: none;
}

.menu2 {
  display: none;

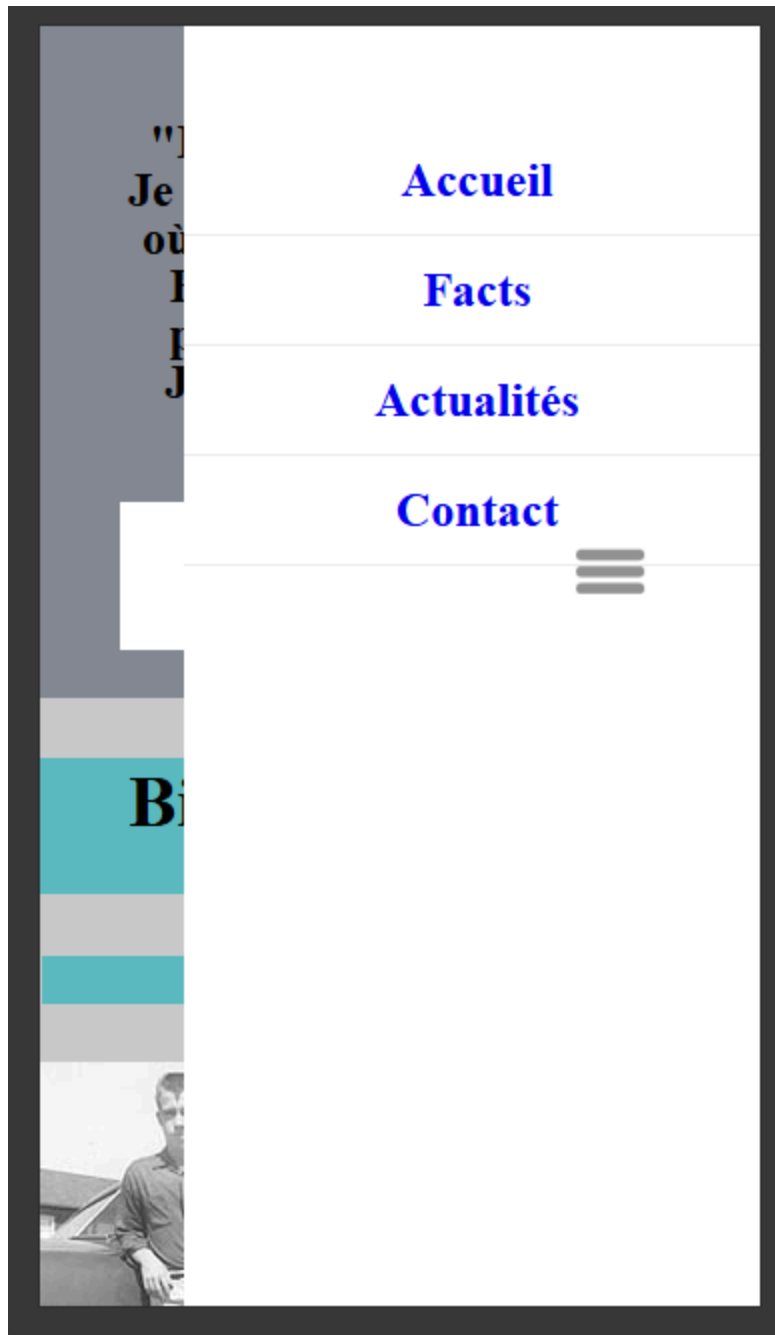
  position: fixed;
  top: 0;
  right: 0;
  width: 80%;
  height: 100vh;
  background-color: white;
  z-index: 100;
  padding-top: 50px;
}

.menu2 > div {
  padding: 15px;
  border-bottom: 1px solid #eee;
}

@media (max-width: 480px) {
  header nav {
    display: none;
  }

  .burger {
    display: block;
    text-align: right;
    padding: 10px;
    cursor: pointer;
    position: relative;
    z-index: 101;
  }

  .burger:hover + .menu2 {
    display: block;
  }
}
```



## **Conclusion**

Ce TD a transformé un site fixe en site responsive. Nous avons utilisé Flexbox pour une mise en page fluide, la balise meta viewport pour les mobiles, et les media queries pour les points de rupture. Le site réorganise ses colonnes et affiche un menu "burger" sur petits écrans, s'adaptant à tous les appareils.