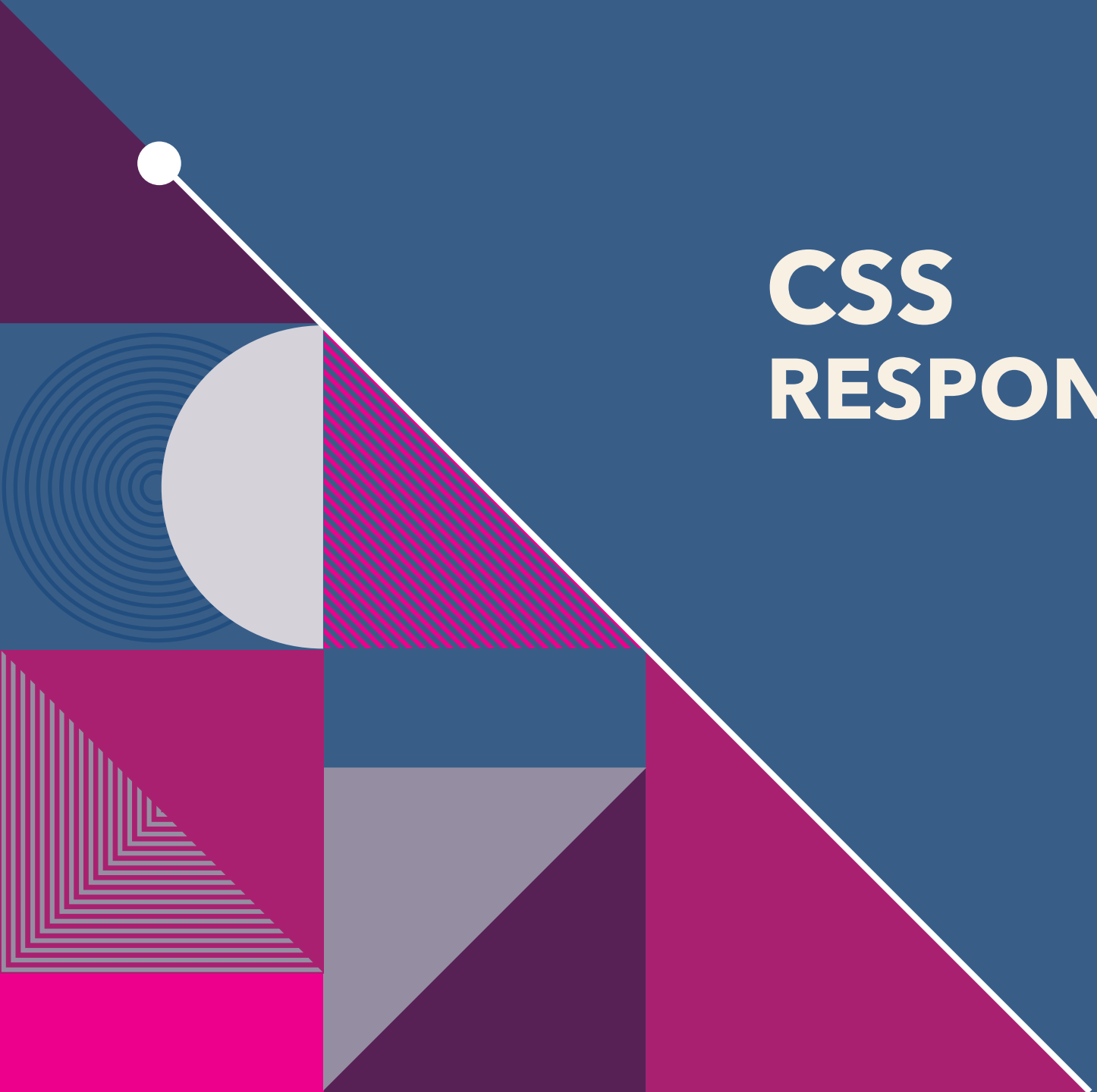


CSS
RESPONSIVE DESIGN

CSS : RESPONSIVE DESIGN INTRODUCTION

Jusqu'aux années 2010, les personnes surfaient sur la toile principalement à l'aide d'ordinateurs personnels.

Depuis, les « devices » (terminaux) utilisés pour naviguer sur le web se sont multipliés (tablettes, mobiles) et il est devenu primordial de rendre compatible et agréable la navigation sur les sites peu importe le device que l'on utilise.

A partir de là, plusieurs possibilités se sont révélées :

- Créer une version « mobile » des sites : c'est ce qui était massivement fait au début.
- Créer une appli mobile : Très bénéfique d'un point de vue commercial mais également pour l'optimisation et les performances. Toutefois, peut nécessiter des compétences techniques qui ne sont pas présentes dans l'équipe. De plus, certains utilisateurs peuvent ne pas souhaiter installer d'app.

CSS : RESPONSIVE DESIGN INTRODUCTION

Une autre solution est de proposer un site qui s'adapte à l'écran utilisé pour le parcourir.

C'est le « Responsive Design ».

- Mobile First vs Responsive Design :
  - Lorsqu'on fait du Responsive Design, on va développer en ayant à l'esprit l'usage sur un écran de PC en premier, puis on ajoute des adaptations pour les autres tailles d'écrans pouvant être utilisées.
  - En Mobile First, on procède à l'inverse. On développe en 1er notre interface pour le mobile, puis on ajoute le responsive pour s'adapter aux tailles d'écrans plus grands. On utilisera le Mobile First lorsque la majorité de nos utilisateurs consulteront le site avec un smartphone ou une tablette. Il peut aussi être plus simple « d'agrandir » un design plutôt que le « rétrécir ».

CSS : RESPONSIVE DESIGN MEDIA QUERIES

Les Media Queries vont nous permettre d'appliquer certaines règles CSS de manière conditionnelle. Par exemple, on va pouvoir définir une largeur pour un élément pour certaines tailles d'écrans et une largeur différente pour le même élément pour d'autres tailles d'écran.

Avec les Media Queries, nous allons pouvoir appliquer des règles CSS totalement différentes selon les tailles d'écrans. Notez par ailleurs qu'on va tout à fait pouvoir utiliser le flexbox, etc. dans nos Media Queries.

CSS : RESPONSIVE DESIGN MEDIA QUERIES : EXEMPLE

Nous allons créer un conteneur flex contenant 3 éléments flex.

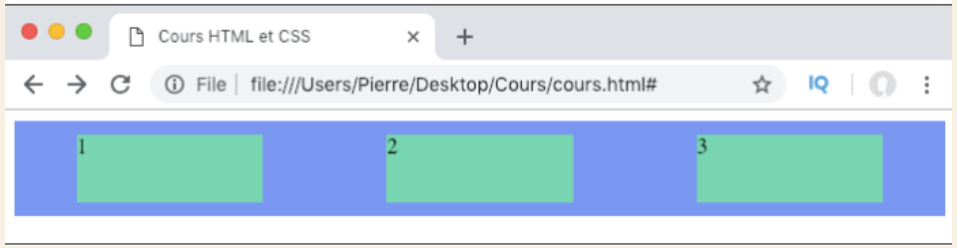
Puis, dans le CSS nous allons définir le fonctionnement général du conteneur et des éléments (axe et direction, alignement).

On définit ensuite une media query (@media). Dans l'exemple, la règle s'appliquera aux appareils disposant d'un écran (screen) de taille inférieure ou égale à 780px (max-width: 780px).

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <div class="conteneur-flex">
            <div class="element-flex">1</div>
            <div class="element-flex">2</div>
            <div class="element-flex">3</div>
        </div>
    </body>
</html>
```

```css
.conteneur-flex{
    display: flex;
    flex-flow : row wrap;
    justify-content: space-around;
    background-color: RGBa(120,150,240,1);
}

.element-flex{
    flex: 0 1 20%;
    background-color: RGBa(120,240,150,0.7);
    height: 50px;
    margin: 10px;
}

@media screen and (max-width: 780px){
    .conteneur-flex{
        flex-flow: column wrap;
    }
}
```

CSS : RESPONSIVE DESIGN MEDIA QUERIES : EXEMPLE



HTML CSS

```
.conteneur-flex{
    display: flex;
    flex-flow : row wrap;
    justify-content: space-around;
    background-color: RGBa(120,150,240,1);
}

.element-flex{
    flex: 0 1 20%;
    background-color:
RGBa(120,240,150,0.7);
    height: 50px;
    margin: 10px;
}

@media screen and (max-width: 780px){
    .conteneur-flex{
        flex-flow: column wrap;
    }
```

Result

EDIT ON CODEPEN

1
2
3

Resources 1× 0.5× 0.25× Rerun

CSS : RESPONSIVE DESIGN <META NAME="VIEWPORT">

Avant d'expliquer le rôle de cette balise, reparlons un peu de la balise HTML « <meta> »

Cette balise est utilisée pour définir les métadonnées d'un document HTML.

Une métadonnée n'est pas affichée sur la page mais est lue par les robots qui vont parcourir et interpréter notre page.

Les attributs les plus courants de l'élément meta vont être les attributs « charset », « name » et « content ».

- L'attribut « charset » : sert à définir le jeu de caractères utilisé sur le document. Cela permet aux navigateurs d'afficher correctement les caractères spéciaux ou accentués du document.
- L'attribut « name » permet d'indiquer le type de métadonnée que l'on va passer. Il va de pair avec l'attribut « content » qui va permettre d'indiquer la valeur de la métadonnée. Par exemple (liste non exhaustive) :
  - author : Nom de l'auteur du document, ce qui donnerait <meta name="author" content="Guillaume">
  - description : Description/résumé du contenu de la page. Utilisé par les moteurs de recherche.
  - viewport : La valeur passée à « content » permet d'indiquer comment le navigateur doit afficher la page sur différents appareils.

CSS : RESPONSIVE DESIGN <META NAME="VIEWPORT">

Le viewport représente la partie visible d'une page web ou la fenêtre active.

La taille du viewport va donc varier en fonction de la taille de l'écran utilisé.

Le problème sur smarphone est que la taille du viewport est souvent différente de la taille réelle de l'écran, pour des raisons de praticité (éviter de devoir dézoomer un site qui ne serait pas optimisé pour mobile).

Le problème qui en découle est que les pages apparaitront plus ou moins dézoomées selon les appareils utilisés.

La balise <meta name="viewport"> va nous permettre de reprendre le contrôle du viewport et afficher correctement le site sur tous les devices.

CSS : RESPONSIVE DESIGN <META NAME="VIEWPORT">

Nous allons pouvoir passer plusieurs propriétés à l'attribut « content » pour donner des instructions d'affichage concernant la taille et l'échelle du **viewport**.

- Les propriétés **width** et **height** vont nous permettre de contrôler la taille du **viewport** dans lequel notre page doit s'afficher. On peut leur passer un nombre ou le mot clef « **device-width** » qui correspond à la taille de l'écran en pixels CSS à l'échelle 100%.

- Les **pixels CSS** correspondent à la surface utilisable de l'écran. Ce sont des pixels virtuels que l'appareil « pense » avoir. L'idée importante ici est qu'un pixel CSS n'est pas toujours égal à un pixel physique.
  Les pixels physiques correspondent aux pixels réels qui composent un écran. C'est également ce qu'on appelle la définition d'un écran. Les écrans retina et haute définition possèdent généralement 4 fois plus de pixels réels que de pixels CSS.

CSS : RESPONSIVE DESIGN <META NAME="VIEWPORT">

Nous allons pouvoir passer plusieurs propriétés à l'attribut « content » pour donner des instructions d'affichage concernant la taille et l'échelle du **viewport**.

- La propriété **user-scalable** permet à l'utilisateur de zoomer dans la page (avec la valeur yes) ou, au contraire, lui interdit de la faire (avec la valeur no).
- Cette propriété est souvent utilisée avec les propriétés **minimum-scale** et **maximum-scale** auxquelles on va pouvoir passer un nombre entre 0 et 10 et qui va représenter le niveau de dézoom ou de zoom que l'utilisateur est autorisé à faire.
- Finalement, la propriété **initial-scale** permet de définir de niveau de zoom initial du viewport, c'est-à-dire son échelle. Nous allons également pouvoir lui passer un nombre entre 0 et 10.

CSS : RESPONSIVE DESIGN <META NAME="VIEWPORT">

En résumé, il va être indispensable de définir le viewport par le biais d'une balise meta afin que les navigateurs mobiles ne définissent pas eux-mêmes leurs propres valeurs de zoom ou de viewport.

Généralement, nous définirons une largeur de viewport égale à la largeur de l'appareil dans le viewport ainsi qu'un niveau de zoom initial égal à 1 et pouvons interdire les utilisateurs de zoomer ou de dézoomer. Toutefois, cela n'est pas recommandé par le W3C.

Bonne nouvelle, à l'exception de la possibilité de zoomer pour les users qu'il faudra potentiellement réactiver, la commande « emmet » html:5 permet de générer cette balise correctement :

```
<meta name="viewport"
      content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
```

CSS : RESPONSIVE DESIGN MEDIA QUERIES : SYNTAXE

Les medias queries permettent de conditionner nos règles CSS.

Pour cela, on va utiliser la règle CSS « @media ».

Dans une règle « @media » nous allons pouvoir utiliser 2 types de conditions : une condition sur le media utilisé (device) et une condition sur les caractéristiques du media.

Les différents types de media sont :

- all : Valeur par défaut. Nos règles vont s'appliquer à tous les appareils
- screen : Nos règles ne vont s'appliquer qu'aux appareils dotés d'un écran
- printer : Nos règles ne s'appliqueront que pour les imprimantes
- speech : Nos règles ne s'appliqueront qu'aux liseurs d'écran qui sont capable de rendre le contenu d'une page de manière sonore

On peut également inverser la valeur logique d'un test avec le mot clé « not » (exemple : not(screen)).

CSS : RESPONSIVE DESIGN MEDIA QUERIES : SYNTAXE

Conditions sur les caractéristiques du media :

- Entourer chaque condition par des parenthèses
- Possibilité d'utiliser des opérateurs logiques (« and », « or », « not »)

Il existe de nombreuses caractéristiques sur lesquelles on peut conditionner nos règles CSS

Mais la plupart du temps, nous utiliserons des conditions de taille. Les propriétés que nous utiliserons le plus seront ainsi : « width », « min-width », « max-width ».

Pour avoir un aperçu des autres possibilités pour conditionner nos styles CSS, se référer à la doc officielle :
https://developer.mozilla.org/fr/docs/Web/CSS/CSS_media_queries/Using_media_queries

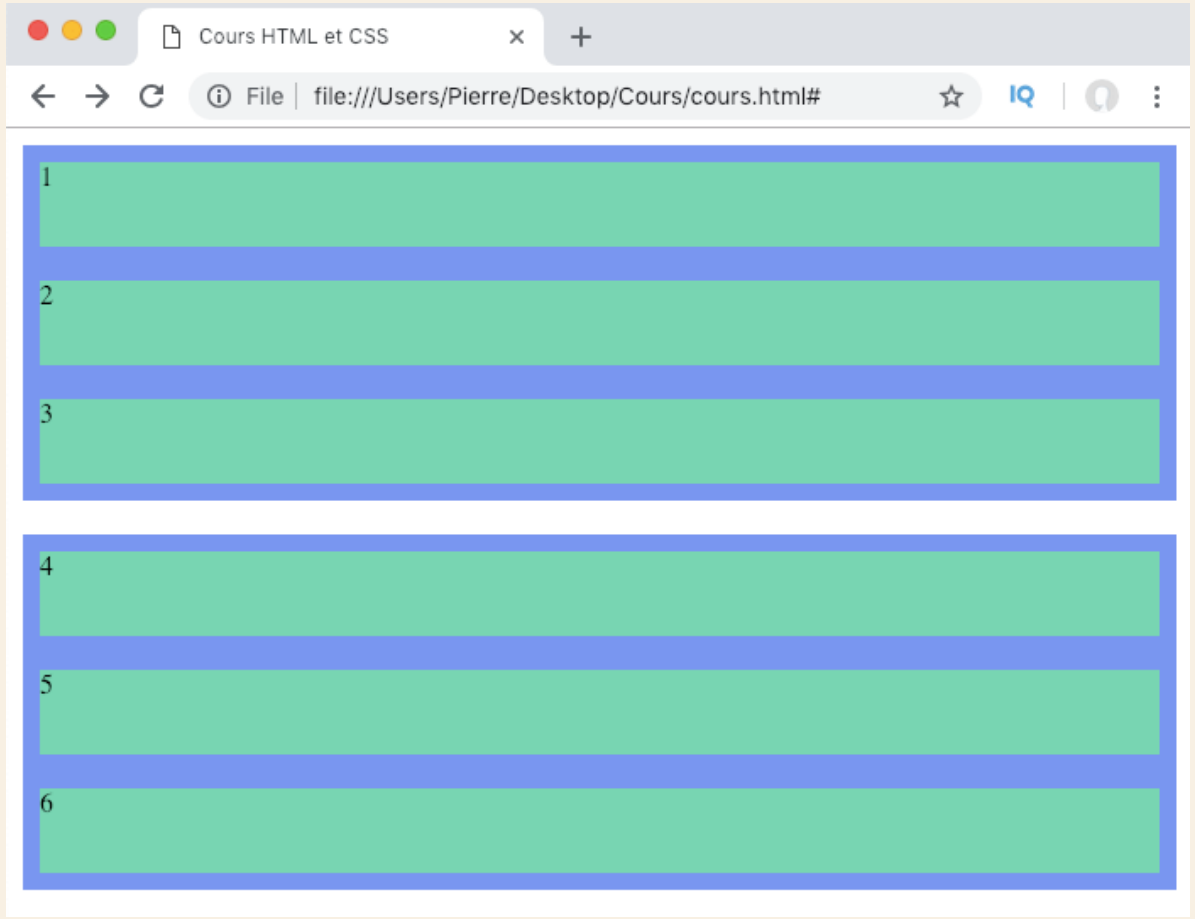
CSS : RESPONSIVE DESIGN EXEMPLE EN MOBILE FIRST

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset="utf-8">
        <meta name="viewport"
              content="width=device-width, initial-scale=1.0, user-scalable=no">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <div class="conteneur-flex">
            <div class="sous-conteneur-flex">
                <div class="element-flex">1</div>
                <div class="element-flex">2</div>
                <div class="element-flex">3</div>
            </div>
            <div class="sous-conteneur-flex">
                <div class="element-flex">4</div>
                <div class="element-flex">5</div>
                <div class="element-flex">6</div>
            </div>
        </div>
    </body>
</html>
```

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

.conteneur-flex{
    display: flex;
    flex-flow: column wrap;
    margin: 10px;
}
.sous-conteneur-flex{
    flex: 1 1 auto;
    display: flex;
    flex-flow: column wrap;
    justify-content: space-around;
    background-color: RGBa(120,150,240,1);
    margin: 0px 0px 20px 0px;
}

.element-flex{
    flex: 1 1 auto;
    background-color: RGBa(120,240,150,0.7);
    height: 50px;
    margin: 10px;
}

@media screen and (min-width: 780px) and (max-width: 979px){
    .conteneur-flex{
        flex-flow: row wrap;
    }
    .sous-conteneur-flex{
        margin: 0px 10px;
    }
}
@media screen and (min-width: 980px){
    .conteneur-flex{
        flex-flow: row wrap;
    }
    .sous-conteneur-flex{
        flex-flow: row wrap;
        margin: 0px 10px;
    }
}
```

CSS : RESPONSIVE DESIGN EXEMPLE EN MOBILE FIRST

Résultat sur mobile

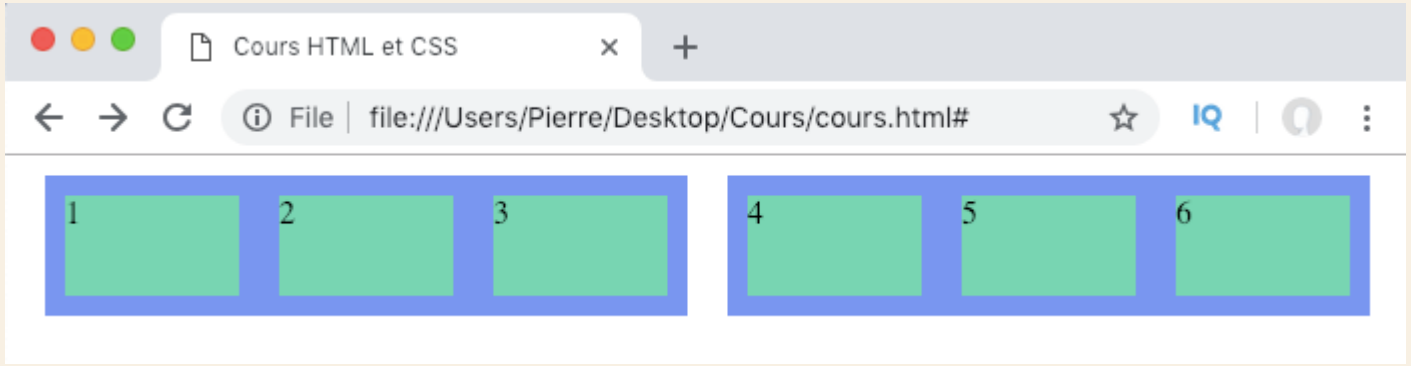


CSS : RESPONSIVE DESIGN EXEMPLE EN MOBILE FIRST

Résultat sur tablette

CSS : RESPONSIVE DESIGN EXEMPLE EN MOBILE FIRST

Résultat sur PC



CSS : RESPONSIVE DESIGN IMAGES RESPONSIVES

Nous savons maintenant comment rendre responsive nos éléments HTML mais il sera aussi nécessaire de rendre les images utilisées par notre site responsives.

Nous pouvons placer l'image dans un conteneur et rendre l'image responsive notamment avec flexbox mais :

- En n'utilisant qu'une seule image, il faudra utiliser une image de la meilleure résolution possible ce qui va alourdir et ralentir notre page web, ce qui pose des problèmes d'ergonomie ainsi que pour le référencement SEO.
- L'image risque de ne pas rendre très bien sur mobile.

CSS : RESPONSIVE DESIGN
PIXELS PHYSIQUE, CSS ET ÉCRANS RETINA

Les écrans retina ont généralement une résolution deux fois plus importante que les écrans standards. Cela signifie que chaque « pixel retina » est l'équivalent de 4 « pixels standards » (2 en largeur, 2 en hauteur).

Ainsi, pour qu'une image s'affiche correctement sur un écran retina, il faudra qu'elle soit deux fois plus grande que l'écran sur lequel elle doit s'afficher (1960x 980px par exemple pour un affichage sur un écran retina de 980x 490px).

Certains écrans retina possèdent une résolution 3 fois plus importante qu'un écran standard, auquel cas il faudra une image 3 fois plus grande et etc.

Nous allons aborder les différentes possibilités :

CSS : RESPONSIVE DESIGN
IMAGES RESPONSIVES

Utiliser des Images SVG (scalable vector graphics)

Ces images sont vectorielles, ce qui signifie qu'on va pouvoir les agrandir ou les rapetisser à l'infini sans perte de qualité.

Cependant, cela ne résout qu'une partie du problème puisqu'en n'utilisant qu'une seule image pour toutes les versions de notre site, cette image risque d'apparaitre comme trop imposante pour la version bureau ou trop dézoomée pour la version mobile.

De plus, souvent, vous serez obligés d'intégrer des photos ou images d'un format différent comme jpeg ou png.

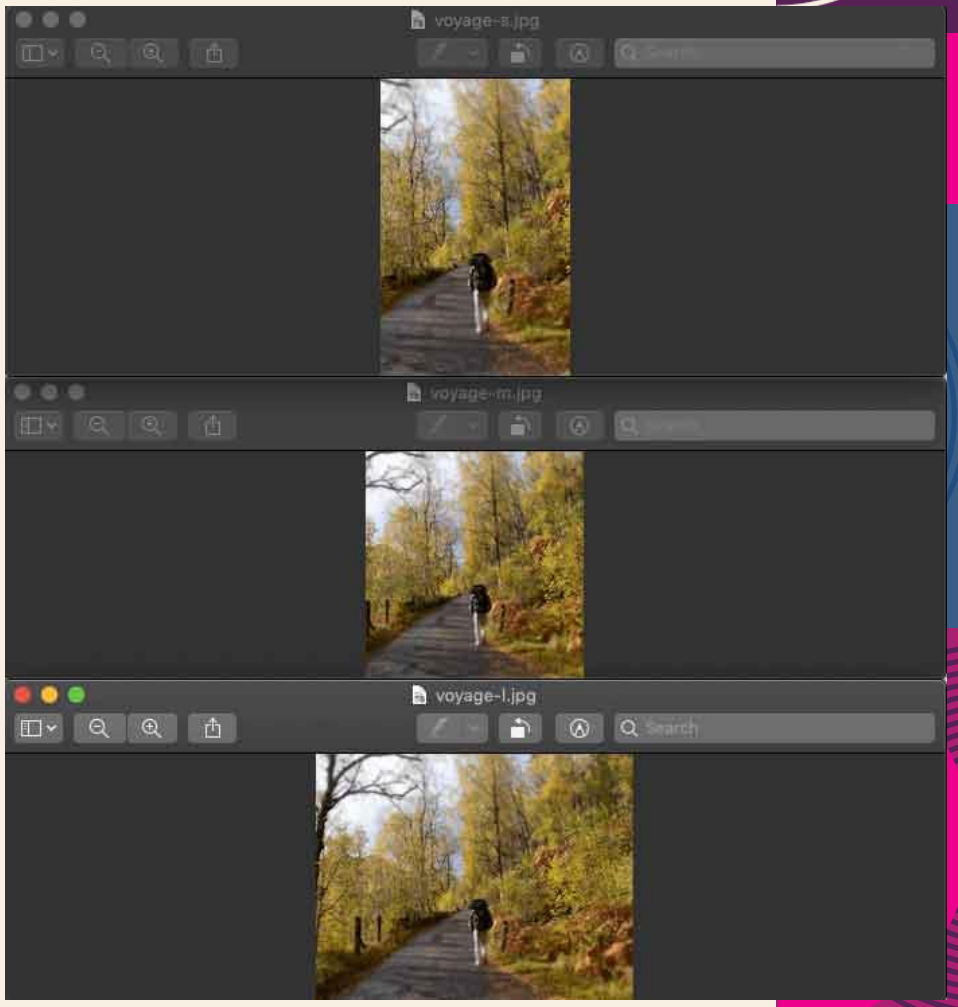
CSS : RESPONSIVE DESIGN
IMAGES RESPONSIVES

Proposer plusieurs versions d'une image (attributs srcset et sizes)

Nous allons pouvoir ajouter un attribut srcset dans notre élément img qui va nous permettre de fournir plusieurs sources d'images (c'est-à-dire concrètement de fournir plusieurs images différentes) au navigateur parmi lesquelles choisir.

L'idée ensuite est de proposer des images différentes en fonction du device utilisé. Par exemple :

- Une version « L » complète de l'image destinée aux grands écrans
- Une version « M » de l'image, rognée et centrée sur le sujet pour les écrans de taille moyenne.
- Une version « S » encore plus centrée sur le sujet pour les petits écrans



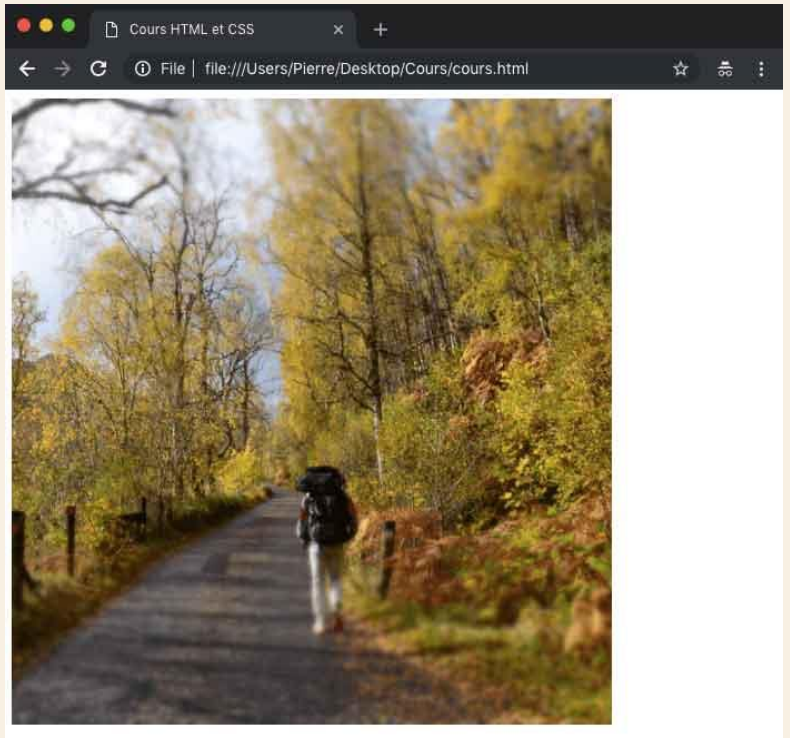
Nous allons étudier un exemple. Vous pouvez télécharger les fichiers sur le site de Pierre Giraud : https://www.pierre-giraud.com/wp-content/uploads/2019/07/voyage.zip

CSS : RESPONSIVE DESIGN
IMAGES RESPONSIVES

Proposer plusieurs versions d'une image (attributs srcset et sizes)

L'attribut srcset va être accompagné d'un attribut sizes qui va nous permettre de préciser un ensemble de conditions relatives au terminal utilisé par un utilisateur (généralement des conditions de tailles de fenêtre) et d'indiquer la largeur que doit occuper l'image dans chaque situation.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset= "utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <img srcset="voyage-s.jpg 400w,
                     voyage-m.jpg 600w,
                     voyage-l.jpg 800w"
             sizes="(max-width: 576px) 380px,
                    (max-width: 768px) 540px,
                    700px"
             src="voyage-l.jpg"
             alt="Voyage en Ecosse">
    </body>
</html>
```



CSS : RESPONSIVE DESIGN
MISE EN PRATIQUE

A partir du code suivant, créez les bonnes règles CSS pour obtenir le résultat.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset=”utf-8”>
  <title>Media Queries</title>
</head>
<body>
<div class="iphone"> Cette div apparait en rouge si la
  largeur maximale de l écran est de 480 pixels. </div>
<div class="inf600"> Cette div apparait en rouge si de la
  fenêtre est inférieure a 600 pixels. </div>
<div class="de600a1200"> Cette div apparait en rouge si la
  largeur de la fenêtre est compris entre 600 et 1200 pixels. </div>
<div class="sup1200"> Cette div apparait en rouge si la
  largeur de la fenêtre est supérieure a 1200 pixels. </div>
<div class="sup1600"> Cette div apparait en rouge si la
  largeur de la fenêtre est supérieure a 1600 pixels. </div>
</body>
</html>
```

CSS : RESPONSIVE DESIGN
ALLER PLUS LOIN

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_media_queries/Using_media_queries

https://www.pierre-giraud.com/html-css-apprendre-coder-cours/introduction-responsive-design/