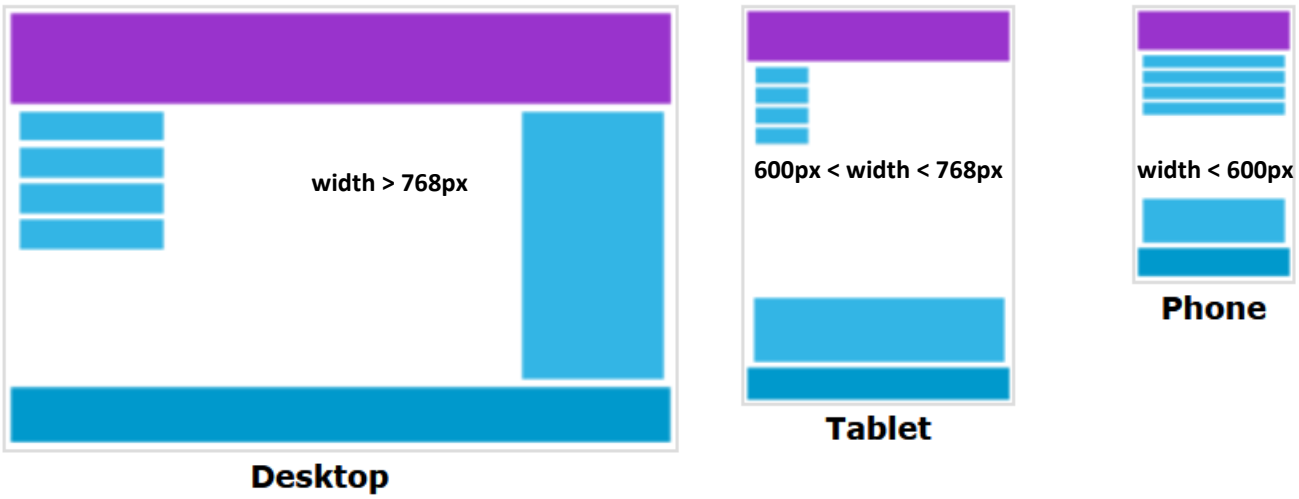
HTML & CSS

Sommaire

I. Introduction

1.1 Présentation

Le **Web responsive Design** permet de mettre en forme de façon convenable une page Web quel que soit le support ou la taille de l'écran :



On utilise la CSS pour **redimensionner, cacher, rétrécir, agrandir ou déplacer le contenu d'une page HTML** afin de le faire paraître correctement sur n'importe quel type d'écran.

1.2 Exemple

w3schools.com : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_col-s

II. Le Viewport : fenêtre d'affichage

2.1 Qu'est-ce que le Viewport ?

Le **Viewport** est la **zone visible d'une page web** vue par un utilisateur (voir image ci-dessus).
On utilise la CSS pour **redimensionner, cacher, rétrécir, agrandir ou déplacer le contenu d'une page HTML** afin de le faire paraître correctement sur n'importe quel type d'écran.
Avant l'apparition des les tablettes et des téléphones mobiles en 2011 (voir « histoire du Smartphone » – Futurascience [1]), les pages web étaient conçus uniquement pour les écrans d'ordinateur de bureau. Il était donc courant pour les pages web pour avoir une conception statique et une taille fixe.
Lorsque les utilisateurs ont commencés à surfer sur internet sur leurs Smartphones et tablettes, les pages web de taille fixes se sont retrouvées trop grande pour la taille des écrans. Les navigateurs de ces appareils ont donc rétréci la taille des pages pour qu'elles s'adaptent à la petite taille des écrans : une solution rapide à mettre en œuvre mais loin d'être idéale dans la mesure où certains éléments peuvent devenir illisibles.

2.2 Réglage du Viewport

En HTML5, la balise <meta> a été introduite pour que les designers web puissent contrôler le Viewport. Ceci est rendu possible en introduisant un élément "viewport" dans la balise <meta>, de la manière suivante :

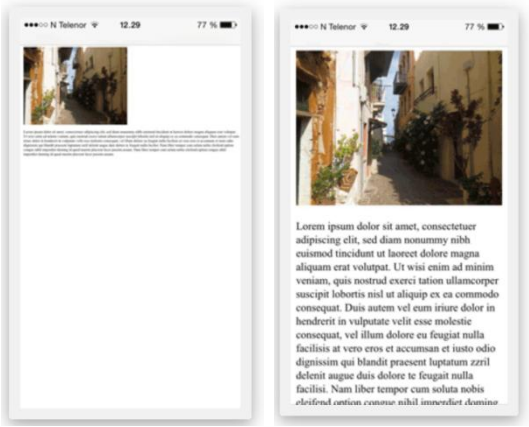
```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Description des éléments :

- L'élément "viewport" de la balise <meta> permet de donner des instructions au navigateur sur la manière de contrôler la dimension d'une page et sa mise à l'échelle.
- L'élément "width=device-width" permet de faire en sorte que la largeur de la page web soit exactement celle de l'écran utilisé, quel que soit l'appareil utilisé.
- L'élément "initial-scale=1.0" permet de régler le zoom initial de la page lorsque celle-ci est initialement chargée. Ici le zoom initial est de 100%.

[1] https://www.futura-sciences.com/tech/dossiers/telecoms-smartphones-guerre-systemes-exploitation-mobiles-1487/page/2/

Voici un exemple d'une page Web *sans* l'élément "viewport" dans la balise <meta>, et la même page web *avec* l'élément "viewport" dans la balise <meta> :



2.3 Adapter la taille du contenu à celle du Viewport

Les utilisateurs web sont habitués à faire défiler verticalement les sites sur les ordinateurs de bureau, les Smartphones ou les tablettes, mais ils le sont moins à les faire défiler horizontalement. Ainsi, si l'utilisateur est obligé de faire défiler horizontalement, ou d'effectuer un zoom arrière pour voir toute la page web, cela rend l'expérience de l'utilisateur très difficile.

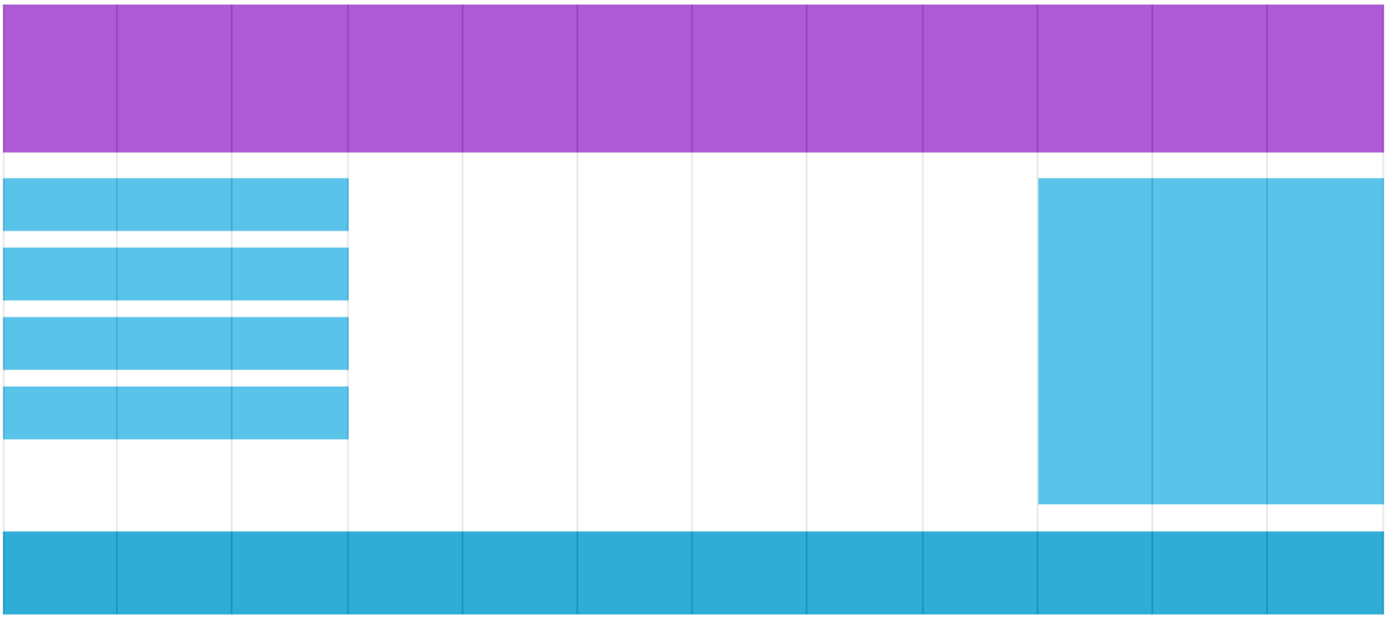
Un certain nombre de règle sont à respecter pour que l'expérience de l'utilisateur soit optimale :

1. **Ne pas utiliser un élément de grande largeur fixe.** Par exemple, si une image est affichée avec une taille supérieure à celle du Viewport, cela génère habituellement une barre de scrolling horizontale sur cette dernière. Il est donc nécessaire de toujours ajuster la taille des éléments à celle du Viewport.
2. **Ne pas fixer la taille d'un élément sur la taille d'un Viewport particulier.** En effet, comme la taille des écrans varie d'un appareil à l'autre il est peu probable que la taille de l'élément en question s'adapte automatiquement à la taille de l'écran utilisé.
3. **Utiliser les « media queries » de la CSS pour gérer l'affichage et la mise en forme des pages web sur des écrans de petites ou grandes tailles.** Lorsque la largeur et les dimensions des éléments d'une page sont fixées en nombre de pixels de façon absolue dans la CSS, cela peut provoquer un dépassement de l'élément de l'écran (viewport) si celui-ci est trop petit. Afin de remédier à ce problème, il est conseillé d'utiliser une largeur relative : « width : 100% ». Attention aussi à l'utilisation de positionnements fixes de grandes valeurs sur un élément. En effet, cela peut provoquer l'affichage de l'élément en question en dehors de l'écran utilisé si celui-ci est de trop petite taille.

III. La grille d'affichage : Grid View

3.1 Qu'est-ce que la grille d'affichage (Grid View) ?

Un grand nombre de pages web responsives sont basées sur une grille d'affichage. Cela signifie qu'elles sont divisées en plusieurs colonnes (voir image ci-dessous). L'utilisation d'une grille d'affichage est très utile pour le design d'une page web car elle permet de placer plus facilement les éléments dans la page.



Une grille d'affichage comporte souvent 12 colonnes pour une largeur totale d'affichage de 100%. La taille de la grille rétrécie ou s'agrandit lorsque la fenêtre du navigateur est redimensionnée.

Exemple : https://www.w3schools.com/css/tryresponsive_grid.htm

3.2 Construction d'une grille d'affichage adaptative (Responsive Grid-View)

En premier lieu, il faut s'assurer que tous les éléments HTML comportent la propriété « **box-sizing** » ayant la valeur « **border-box** ». Cela permet de s'assurer que le « **padding** » et le « **border** » sont inclus dans la largeur et hauteur totale de l'élément.

Le code CSS est le suivant :

```
* { box-sizing: border-box; }
```

Pour plus d'information sur box-sizing, se référer au chapitre « CSS Box Sizing » : https://www.w3schools.com/css/css3_box-sizing.asp.

Exemple 1 : Une page web avec 2 colonnes prenant respectivement 25% et 75% de la largeur de la page.

25%	75%

Code : Dans le code HTML, créer une classe « menu » et une classe « main » et les mettre en forma avec la CSS suivante

```
.menu { width: 25%; float: left; }
.main { width: 75%;  float: left; }
```

Pour le code complet, se référer au lien suivant : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_webpage.

Exemple 2 : Afin d'avoir un meilleurs contrôle sur l'architecture de la page il est plus intéressant de construire une grille d'affichage adaptative (responsive grid-view) comportant 12 colonnes.

Chacune de ces colonnes aura une largeur en pourcentage multiple de $(1/12)^{ième}$ de la largeur totale de la page. En d'autre terme width = n*(1/12)*100% = n*8.33%, avec $n \in \{1, \dots, 12\}$. Ceci permet de moduler de façon plus efficace la largeur d'une « div » dans la page.

Protocole de construction :

1. Calculer la largeur en pourcentage de chaque colonne : 100% / 12 = 8,33%.
2. Créer une classe pour chacune des 12 colonnes : class = " col-i ", pour $i \in \{1, \dots, 12\}$. Cela donne le code CSS suivant :

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_cols.

Code complet :

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
      * {  box-sizing: border-box;}
      .header {  border: 1px solid red;  padding: 15px;}
      .row::after {  content: "";  clear: both;  display: table;}
      [class *="col-"] {  float: left;  padding: 15px;  border: 1px solid red;}
      .col-1 {width: 8.33%;} .col-2 {width: 16.66%;} .col-3 {width: 25%;} .col-4 {width: 33.33%;} .col-5 {width: 41.66%;} .col-6 {width: 50%;} .col-7
      {width: 58.33%;} .col-8 {width: 66.66%;} .col-9 {width: 75%;} .col-10 {width: 83.33%;} .col-11 {width: 91.66%;} .col-12 {width: 100%;}
    </style>
</head>
<body>
    <div class="header">
      <h1>Chania</h1>
    </div>
    <div class="row">
    <div class="col-3">
     <ul>
      <li>The Flight</li>
      <li>The City</li>
      <li>The Island</li>
      <li>The Food</li>
     </ul>
    </div>
    <div class="col-9">
     <h1>The City</h1>
     <p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern
    city.</p>
     <p>Resize the browser window to see how the content respond to the resizing.</p>
    </div>
    </div>
    </body>
</html>
```

Commentaire du code HTML :

Chaque ligne et chaque colonne doivent être encapsulées dans une . Par exemple :

```
<div class="row">
  <div class="col-3">...</div> <!-- 25% -->
  <div class="col-9">...</div> <!-- 75% -->
</div>
```

Afin de prendre en compte la largeur maximale de la page, il est nécessaire que la somme des largeurs occupées par chaque colonne fasse 100%. Pour cela, il suffit que la somme des indices de chaque colonne « col-*i* » soit égale à 12. L'exemple précédent comporte deux colonnes ("col-3" et "col-9") dans une ligne. On remarque que la somme des indice fait bien 12 (=3+9).

Commentaire du code CSS :

- Chaque colonne doit flotter à gauche et doit avoir un « padding » de 15px :

  ```
  [class *="col-"] {  float: left;  padding: 15px;  border: 1px solid red;}
  ```

  Dans la ligne précédente, un sélecteur d'attribut de CSS « [] » (crochets) est utilisé pour sélectionner un élément avec un attribut spécifique. Ici, la sélection s'effectue sur tous les attributs « class » dont les noms commencent par « col- » et sur lesquelles on applique les propriétés float: left; padding: 15px; border: 1px solid red; .

- La ligne « .row::after { content: ""; clear: both; display: table; } » permet de mettre en forme la classe « row ». Il utilise le Pseudo-element[2] « after[3] », via l'opérateur « :: ». Ce Pseudo-élément permet d'insérer un contenu particulier après le contenu de l'élément sur lequel il s'applique :
  - Ici le contenu inséré est vide : « content: ""; ».
  - La propriété « clear[4] » spécifie de quel coté d'un élément (droite ou gauche) un élément fils n'a pas le droit d'être flottant. Ici la valeur « both » signifie qu'un élément n'a le droit d'être flottant ni à droite ni à gauche.
  - La propriété « display[5] » permet de spécifier le rendu graphique d'un élément (généralement « block » ou « inline »). Ici la valeur de display est « table[6] », ce qui signifie que l'élément en question se comporte comme l'élément d'une table.

La page précédente peut être mise en forme de façon lus élégante en rajoutant des couleurs de fond (background-color), unes effet d'ombrage (box-shadow) et des changements de style lorsque certains élément sont survolés par la sourie (li:hover). On obtient le code CSS suivant :

```
html {
  font-family: "Lucida Sans", sans-serif;
}
.header {
  background-color: #9933cc;
  color: #ffffff;
  padding: 15px;
}
.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
.menu li {
  padding: 8px;
  margin-bottom: 7px;
  background-color :#33b5e5;
  color: #ffffff;
  box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}
.menu li:hover {
  background-color: #0099cc;
}
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_styles.

[2] https://www.w3schools.com/css/css_pseudo_elements.asp
[3] https://www.w3schools.com/css/tryit.asp?filename=trycss_after
[4] https://www.w3schools.com/CSSref/pr_class_clear.asp
[5] https://www.w3schools.com/CSSref/pr_class_display.asp
[6] https://www.w3schools.com/CSSref/playit.asp?filename=playcss_display&preval=table

IV. Mise en œuvre d'une page « responsive » : Media Queries

4.1 Qu'est-ce qu'un « Media Queriy » ?

Un « Media Query [7]» est une requête associé à un média particulier (un écran, une imprimante, une TV, une tablette, un Smartphone, etc.). Elle utilise la règle **@media** qui ne s'applique pas au type d'appareil utilisé, mais permet de tester les caractéristiques des appareils tels que :

- Largeur et hauteur du « Viewport »
- Largeur et hauteur de l'appareil
- Orientation de l'appareil : mode portrait ou paysage sur une tablette ou un Smartphone
- Résolution de l'appareil

La règle **@media** permet ainsi d'adapter la feuille de style à chaque appareil utilisé (ordinateur de bureau ou portable, tablette, Smartphone). Elle permet par exemple d'appliquer un block d'instruction CSS si et seulement si une condition est vrai.

Par exemple, si la sortie média est uniquement un écran et si la largeur de la fenêtre est inférieure ou égale à 600px , la couleur du fond devient bleu clair :

```
@media only screen and (max-width: 600px) {
 body {
   background-color: lightblue;
 }
}
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_mediaquery .

4.2 Insertion d'un « Breakpoint » : frontière délimitant deux styles

Les Media Queries permettent d'insérer des « Breakpoints » ou « points d'arrêts ». Un Breakpoint est une frontière autour de laquelle le design d'une page pourra se comporter de façon totalement différente. Par exemple une page vue à travers un écran d'ordinateur ou un Smartphone :

Pour réaliser le passage du design pour un écran d'ordinateur (**Desktop** sur l'image précédente) à celui pour l'écran d'un Smartphone (**Phone** sur l'image précédente), nous pouvons ajouter un **Breakpoint à 768px**.
Ainsi, lorsque la taille du Viewport (écran ou fenêtre du navigateur), chaque colonne prend une taille de 100% de la largeur :

```
/* For desktop: */
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

[7] https://www.w3schools.com/css/css3_mediaqueries.asp

```
@media only screen and (max-width: 768px) {
 /* For mobile phones: */
  [class*="col-"] {
    width: 100%;
  }
}
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_breakpoints .

4.3 Prise en compte initiale d'un design pour Smartphone

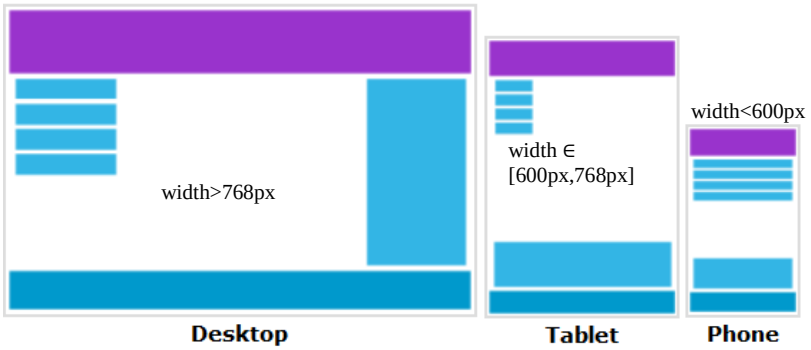
Le code CSS précédent est destiné à designer initialement les pages pour les écrans d'ordinateur et ensuite pour les Smartphones. Il est préférable de réaliser l'inverse pour que l'affichage sur les Smartphone soit plus rapide.

Ainsi, au lieu d'effectuer un changement de style lorsque la largeur du Viewport est *inférieure* à 768px (pour les Smartphones), il vaudra mieux effectuer le changement de style lorsque la largeur du Viewport est *supérieure* à 768px. De cette manière, la première mise en forme affichée sera celle dédiée aux Smartphones :

```
/* For mobile phones: */
[class*="col-"] {
  width: 100%;
}
@media only screen and (min-width: 768px) {
  /* For desktop: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}
```

4.4 Insertion de plusieurs « Breakpoints » : frontière délimitant plusieurs styles

Il est possible d'insérer un nombre quelconque de Breakpoints. Il est ainsi possible d'insérer des Breakpoints pour délimiter l'utilisation d'un écran d'ordinateur, d'une tablette et d'un Smartphone :



La première mise en page est par défaut celle destinée aux Smartphones (width<600px). La deuxième est destinée aux tablettes dont la largeur du Viewport est supérieure à 600px mais (implicitement) inférieure à 768px.). La troisième est destinée aux ordinateurs dont la largeur du Viewport est supérieure à 768.

Pour prendre en compte la taille des **tablettes**, il est nécessaire d'introduire un nouvel ensemble de classes nommées « **col-s-*i*** » pour $i \in [1, \ldots, 12]$.
Le code CSS est le suivant :

```
/* For mobile phones: */
[class*="col-"] {
  width: 100%;
}
@media only screen and (min-width: 600px) {
  /* For tablets: */
  .col-s-1 {width: 8.33%;}
  .col-s-2 {width: 16.66%;}
  .col-s-3 {width: 25%;}
  .col-s-4 {width: 33.33%;}
  .col-s-5 {width: 41.66%;}
  .col-s-6 {width: 50%;}
  .col-s-7 {width: 58.33%;}
  .col-s-8 {width: 66.66%;}
  .col-s-9 {width: 75%;}
  .col-s-10 {width: 83.33%;}
  .col-s-11 {width: 91.66%;}
  .col-s-12 {width: 100%;}
}
@media only screen and (min-width: 768px) {
  /* For desktop: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_col-s%20.

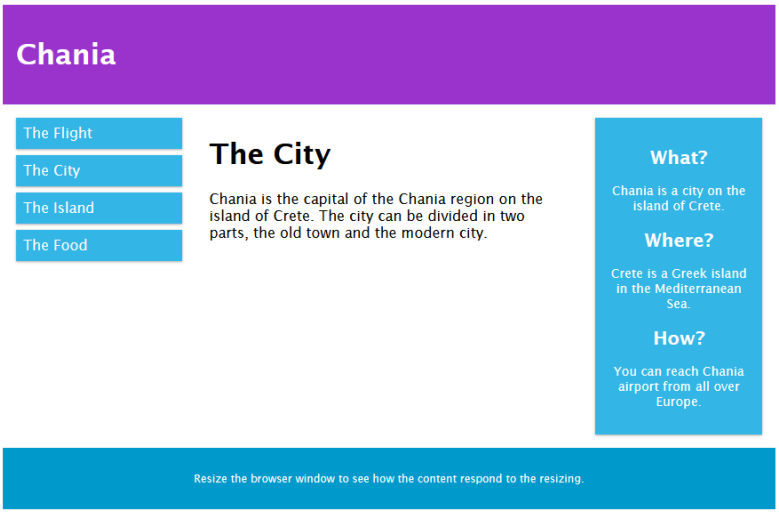
Remarque :
Cela peut paraître étrange de créer deux ensembles de classes strictement identiques dans le fichier HTML : col-*i* et col-s-*i*.
Le code HTML est le suivant :

```
<div class="row">
  <div class="col-3  col-s-3">...</div>
  <div class="col-6  col-s-9">...</div>
  <div class="col-3  col-s-12">...</div>
</div>
```

Il est donc possible d'attribuer une < div > plusieurs class qui seront choisies en fonction des tests réalisés dans la règle @media définie plus haut.

Dans l'exemple précédent :

- Pour les écrans d'ordinateur de bureau : la 1ère et 3ième section s'étendent sur 3 colonnes et la 2nde sur 9 colonnes.



- Pour les tablettes : 1ère section s'étend sur 3 colonnes, la 2nde sur 9 colonnes, la 3ième section est affichée en dessous des deux premières avec une largeur de 12 colonnes.

4.5 Les « Breakpoints » typiques de chaque appareil

Il existe une pléthore d'écran et d'appareils comportant des hauteurs et largeurs différentes. Il est donc très difficile de créer un Breakpoint exact associé à chaque appareil. Pour simplifier la classification, il est possible de regrouper les différentes cibles en 5 groupes résumés dans l'exemple suivant :

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) {...}
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_mediaquery_breakpoints.

4.6 Orientation en Portrait ou Paysage

Les Media Queries peuvent également être utilisés pour modifier la mise en page d'une page en fonction de l'orientation du navigateur. Il est possible de faire en sorte qu'un ensemble de propriété CSS soit appliqué lorsque la largeur de la fenêtre du navigateur est plus grande que sa hauteur, c'est-à-dire en mode « paysage ».

Exemple : Lorsque la page est en mode paysage (widht>height), la couleur du fond (background-color) devient bleue ciel, sinon le fond est vert. Le code CSS est le suivant :

```
@media only screen and (orientation: landscape) {
 body {
  background-color: lightblue;
 }
}
```

Le code complet, se trouve sur le lien suivant :
https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_mediaquery_orientation .

4.7 Cacher des éléments avec les Media Queries

Une autre utilisation courante des Media Queries consiste à cacher certains éléments d'une page en fonction de la taille du Viewport ou de l'écran.
Exemple : Cacher une div « exemple » ayant un fond jaune lorsque l'écran a une largeur inférieure à 600px.
Le code CSS est le suivant :

```
/* If the screen size is 600px wide or less, hide the element */
@media only screen and (max-width: 600px) {
 div.example {
  display: none;
 }
}
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=trycss_mediaqueries_hide .

4.8 Modifier la taille de la police avec les Media Queries

Il est possible de modifier la taille de la police (Font Size) avec les Media Queries.

Exemple : Fixer la taille de la police de caractère dans une div « exemple » à 80px lorsque l'écran a une largeur supérieure à 601px, et modifier la taille à 30px lorsque l'écran a une largeur inférieure à 600px.
Le code CSS est le suivant :

```
/* If the screen size is 601px or more, set the font-size of <div> to 80px */
@media only screen and (min-width: 601px) {
 div.example {
  font-size: 80px;
 }
}
/* If the screen size is 600px or less, set the font-size of <div> to 30px */
@media only screen and (max-width: 600px) {
 div.example {
  font-size: 30px;
 }
}
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=trycss_mediaqueries_fontsize .

4.9 Bilan : toutes les règles CSS des Media Queries (@media)

De manière générale, la syntaxe de la règle **@media** est la suivante :

```
@media not|only mediatype and (mediafeature and|or|not mediafeature) {
 CSS-Code;
}
```

Les opérateurs logiques[8] « and », « or », « not » et « only » peuvent être utilisés afin de construire une requête média complexe. Il est aussi possible de combiner plusieurs requêtes média en les séparant par des virgules :

- « and » : L'opérateur and permet de combiner plusieurs requêtes média en une seule. Pour que la requête résultante soit vraie, il faut que chacune des sous-requêtes soit vraie. Cet opérateur est également utilisé afin de relier des caractéristiques média avec des types de média.

- « not » : L'opérateur not est utilisé afin d'obtenir le résultat opposé d'une requête média (il renvoie true si l'opérande renvoie false). S'il est utilisé dans une liste de requêtes séparées par des virgules, il ne nie que la requête sur laquelle il est appliqué. Si l'opérateur not est utilisé, la requête doit nécessairement contenir un type de média.

- « only » : L'opérateur only est utilisé afin d'appliquer un style uniquement si l'intégralité de la requête est vérifiée. Il permet d'empêcher les anciens navigateurs d'appliquer les styles concernés. Si on n'utilise pas only, un ancien navigateur interprètera screen and (max-width: 500px) comme screen uniquement (appliquant ainsi le style à tous les écrans). Si l'opérateur only est utilisé, la requête doit nécessairement contenir un type de média.

- , (virgule) : Les virgules permettent de combiner plusieurs requêtes en une. Chaque requête est traitée séparément. Autrement dit, si une des requêtes de la liste renvoie true, toute la requête combinée renverra true. En ce sens, l'opérateur « , » agit comme un opérateur booléen or.

Il est possible d'appeler différentes feuilles de style pour différents media de cette manière :

```
<link rel="stylesheet" media="screen and (min-width: 900px)" href="widescreen.css">
<link rel="stylesheet" media="screen and (max-width: 600px)" href="smallscreen.css">
....
```

[8] https://developer.mozilla.org/fr/docs/Web/CSS/Requ%C3%AAtes_m%C3%A9dia/Utiliser_les_Media_queries

Les différents types de média pris en compte par la règle **@media** sont les suivants :

- all : Valeur par défaut. Utilisé pour tous les types d'appareils.
- print : Utilisé pour les imprimantes
- screen : Utilisé pour les écrans d'ordinateur, les tablettes, Smartphones, etc.
- speech : Utilisé pour les lecteurs d'écran permettant de lire une page « à voix haute »

Il existe 35 paramètres qu'il est possible de contrôler grâce à la règle **@media** :

any-hover,	max-color	min-width
any-pointer	max-color-index	monochrome
aspect-ratio	max-height	orientation
color	max-monochrome	overflow-block
color-gamut	max-resolution	overflow-inline
color-index	max-width	pointer
grid	min-aspect-ratio	resolution
height	min-color	scan
hover	min-color-index	scripting
inverted-colors	min-height	update
light-level	min-monochrome	width
max-aspect-ratio	min-resolution	

La description de ces paramètres est accessible sur le lien suivant : https://www.w3schools.com/cssref/css3_pr_mediaquery.asp .

V. Gestion « responsive » des images

5.1 Utilisation de la propriété « width »

Lorsque la propriété relative à la largeur « width » est réglée sur un certain pourcentage et que celle relative à la hauteur « height » est réglée sur « auto », l'image aura une taille adaptative. Sa taille augmente ou diminue en fonction de la taille du Viewport.
Le code CSS associé est le suivant :

```
img {
  width: 100%;
  height: auto;
}
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image2 .

Remarque 1 : Ici, il n'est pas nécessaire d'utiliser la règle **@media** pour rendre la taille de l'image adaptative.

Remarque 2 : Dans l'exemple précédent, l'image peur être agrandie à une taille supérieure à sa taille originale. Pour éviter les effets de pixellisation lié à une taille trop importante, il convient d'utiliser la propriété « max-width » à la place de « width ».

5.2 Utilisation de la propriété « max-width »

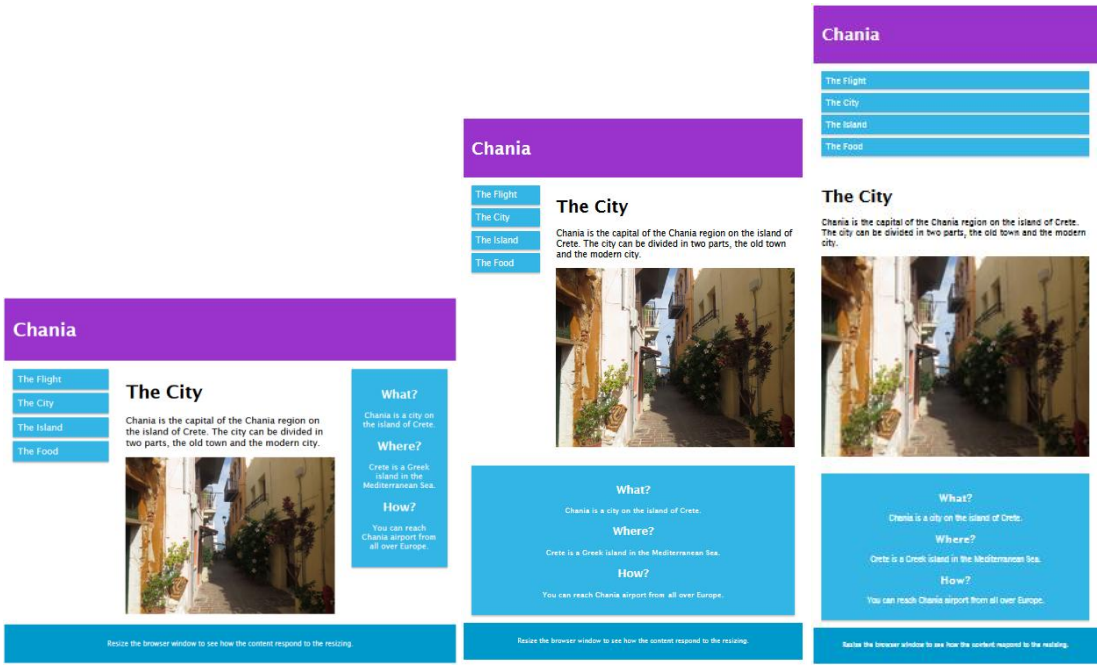
Lorsque la propriété « max-width » est réglée sur 100%, la taille de l'image pourra diminuer sans restriction, mais elle ne pourra pas augmenter au-delà de sa taille originale.
Le code CSS associé est le suivant :

```
img {
  max-width: 100%;
  height: auto;
}
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image .

5.3 Exemple d'insertion d'une image responsive dans un site

Dans l'exemple suivant, la taille et la position de l'image varie en fonction de la largeur du Viewport :



Le code CSS est le suivant :

```
img {
  width: 100%;
  height: auto;
}
```

L'image est insérée dans la div centrale relative au texte dont le titre 1 est « The City », à la suite d'un paragraphe <p> contenant le texte « Chania is the capital… » . Le code HTML est le suivant :

```
<div class= "col-6 col-s-9">
  <h1>The City </h1>
  <p> Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern city.</p>
  <img src = "img_chania.jpg" width = "460" height= "345" >
</div>
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image3 .

5.4 Gestion des images de fond (Background Images)

Les images de fond peuvent aussi être redimensionnées ou mise à l'échelle. Il existe 3 méthodes permettant la gestion des images de fond de façon « responsive » utilisant la propriété « background-size »:

a. L'image de fond garde sa taille originale quelle que soit la taille de la page

b. L'image de fond est étirée sur la largeur de la page quitte à être déformée

c. L'image de fond agrandie ou rétrécie pour s'adapter à la largeur de la page

Nous détaillons ces trois méthodes dans les sections suivantes.

a. Conservation de la taille originale

Lorsque la propriété « background-size » prend la valeur « contain », l'image de fond se redimensionne de sorte qu'elle s'adapte au contenu tout en conservant ses proportions initiales.

Le code CSS est le suivant :

```
div {
  width: 100%;
  height: 400px;
  background-image: url('img_flowers.jpg');
  background-repeat: no-repeat;
  background-size: contain;
  border: 1px solid red;
}
```

Le code complet, se trouve sur le lien suivant : https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image_background1

Le rendu est le suivant :

b. Conservation de la taille originale

Lorsque la propriété « background-size » prend la valeur "100% 100%", l'image de fond s'étire de sorte qu'elle recouvre l'ensemble de la page.

Le code CSS est le suivant :

VI. Liens de Référence

w3schools.com :

- https://www.w3schools.com/css/css_rwd_intro.asp
- https://www.w3schools.com/css/css3_mediaqueries.asp

MDN web docs moz://a :

- https://developer.mozilla.org/fr/docs/Web
- https://developer.mozilla.org/fr/docs/Web/CSS/@media
- https://developer.mozilla.org/fr/docs/Web/CSS/Requ%C3%AAtes_m%C3%A9dia/Utiliser_les_Media_queries
- https://developer.mozilla.org/fr/docs/Web/CSS/R%C3%A8gles_@#Les_r%C3%A8gles_de_groupe_conditionnelles