

Le responsive design

Table des matières

I. Contexte	3
II. Les moyens principaux	3
III. Exercice : Appliquez la notion	8
IV. Les media queries	8
V. Exercice : Appliquez la notion	11
VI. Les breakpoints	12
VII. Exercice : Appliquez la notion	15
VIII. Mobile-first	17
IX. Auto-évaluation	19
A. Exercice final	19
B. Exercice : Défi.....	21
Solutions des exercices	22

I. Contexte

Durée : 1 h

Environnement de travail : Visual Studio Code

Pré-requis : Bases de CSS

Contexte

L'évolution des usages professionnels et du Web en général tend vers une diversification de plus en plus importante des appareils utilisés, et vers une démocratisation de la mobilité.

Cela a un impact fort sur les applications web que nous développons, car il est devenu nécessaire de pouvoir adapter leurs rendus à toute situation. Les moyens mis en place pour atteindre cet objectif constituent ce que l'on appelle le **responsive design**.

Nous allons ici décrire les moyens permettant de répondre aux cas d'utilisation les plus fréquents auxquels les applications sont soumises.

Comment adapter dynamiquement une interface aux changements d'affichage d'un appareil ? Peut-on définir un graphisme différent lors d'une impression ? Quels sont les impacts sur les normes et les habitudes du développement front ?

II. Les moyens principaux

Objectifs

- Découvrir les solutions permettant de réaliser une application web responsive
- Utiliser des unités relatives en CSS

Mise en situation

La mise en place d'un design évoluant dynamiquement nécessite l'utilisation de techniques et d'instructions apparues pour la plupart d'entre elles avec CSS3. En effet, toutes les problématiques de dynamisme des applications web vont être résolues directement en CSS : soit en appelant une feuille de style spécifique à une situation, soit directement dans la feuille de style initiale. Pour cela, nous allons voir que deux possibilités – éventuellement complémentaires – s'offrent à nous : l'utilisation d'unités de mesure de longueur relatives (que nous allons développer ici) et l'utilisation de **media queries** que nous verrons dans un second temps.

Les unités de longueur relatives en CSS3

En CSS, de nombreuses propriétés acceptent une valeur de longueur. Celle-ci peut être définie en unités de longueur absolues, ce qui ne permet pas d'adaptabilité aux évolutions d'affichage d'un appareil ; ou en unités de longueur relatives, qui, elles, s'adaptent à la taille d'un élément ou sur celle d'un parent.

Cela signifie qu'une longueur exprimée en unité relative permet une évolution dynamique de la propriété concernée en fonction de l'évolution de l'affichage demandé par l'utilisateur.

Il existe de nombreuses unités, mais nous n'évoquerons ici que les unités les plus utilisées, dont la liste est présentée ci-dessous.

Méthode Unités basées sur la taille de la police de caractères

Les principales unités relatives basées sur la taille de la police de caractères sont `em` et `rem` (*root em*) :

- L'unité `em` correspond à la taille de la police de caractères de l'élément sur lequel la propriété est rattachée : c'est-à-dire que, si la taille de la police est de 16 px, alors 1 `em` vaut 16 px, tandis que 2 `em` vaudra 32 px.
- L'unité `rem` correspond à la taille de la police de caractères de l'élément racine auquel la propriété est rattachée : c'est-à-dire que, si la taille de la police du premier élément dans le DOM est de 100% (16px par défaut sur tous les navigateurs), alors 1 `rem` vaut 16 px, tandis que 2 `rem` vaudra 32 px.

Exemple Feuille de style illustrant l'utilisation de `em`

```
1 .element {
2   font-size: 16px;
3 }
4
5 .big {
6   font-size: 2em;
7 } /* La taille de la police des éléments HTML portant la classe big va varier en fonction de
   la taille de police de leur parent direct, ici les éléments de la classe element.*/
```

The screenshot shows a web browser window with two lines of text. The first line is "je fais un 1em" in a standard font size. The second line is "Je fais 2 em" in a larger font size. The browser's developer tools are open, showing the HTML structure and the CSS styles applied to the elements.

The HTML structure is as follows:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <div class="element">
      je fais un 1em
    </div>
    <div class="big">Je fais 2 em</div>
  </body>
</html>
```

The CSS styles are as follows:

```
.element {
  font-size: 16px;
}

.big {
  font-size: 2em;
}
```

The browser's developer tools show the following information:

- The HTML structure is displayed on the left, with the `<div class="big">Je fais 2 em</div>` element selected.
- The CSS styles are displayed on the right, showing the `font-size` property for the `.big` class, which is set to `2em`.

Feuille de style illustrant l'utilisation de rem :

```

1 html {
2   font-size: 100%;
3 } /* La hauteur des éléments HTML définie en rem va varier en fonction de la taille de police
   saisie dans l'élément racine. Ici l'élément racine indique une police de 16px par défaut pour
   tous les navigateurs */
4
5 .element {
6   height: 1rem;
7   font-size: 16px;
8   border: solid;
9 }
10
11 .big {
12   font-size: 32px;
13   height: 2rem;
14   border: solid;
15 } /* La hauteur des éléments HTML portant la classe big ne variera pas en fonction de la
   taille de police définie pour son parent direct. */

```

**Définition Le viewport**

Historiquement, le viewport signifie "*tout ce qui est situé au dessus de la "pliure" (fold)*", en référence à la pliure des journaux papier. Ainsi, il s'agissait de tout ce qui peut être lu sans avoir à ouvrir le journal en entier.

En informatique, le **viewport** correspond à la taille d'affichage disponible d'un navigateur. Celle-ci varie d'un navigateur à l'autre, particulièrement sur les terminaux mobiles !

Méthode Unités basées sur le viewport

L'unité v_w est une unité relative à la taille de la largeur du viewport de l'appareil utilisé : ainsi, $1v_w$ correspond à 1 % de la largeur totale du viewport.

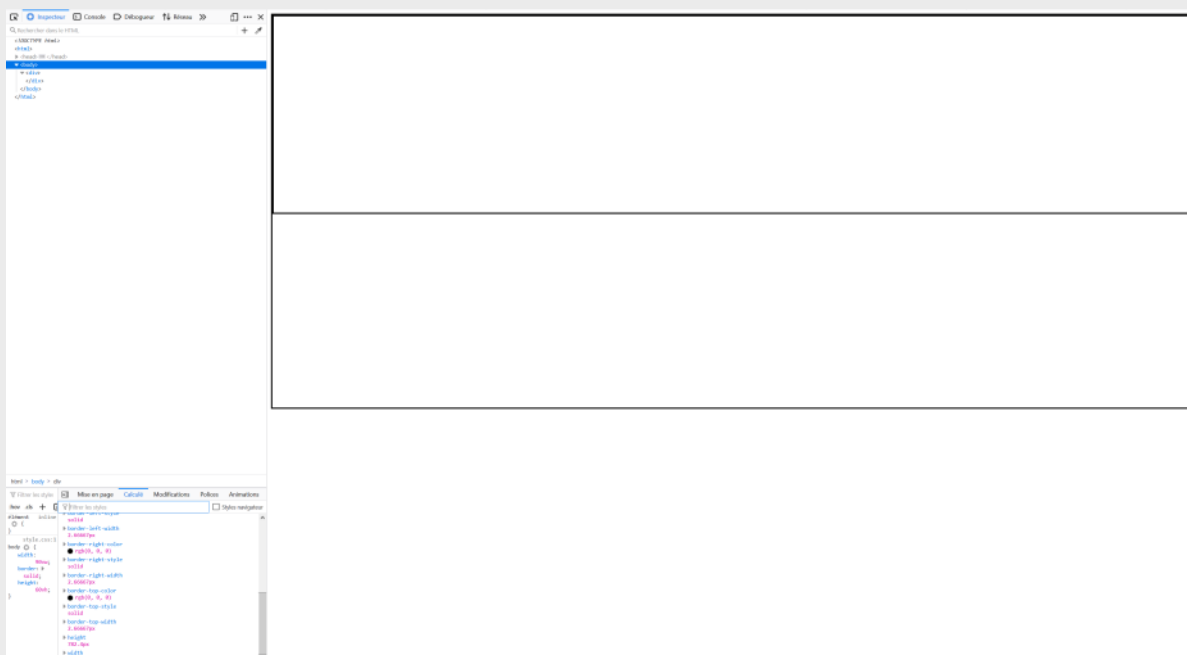
L'unité v_h est une unité relative à la taille de la hauteur du viewport de l'appareil utilisé : ainsi, $1v_h$ correspond à 1 % de la hauteur totale du viewport.

Exemple Feuille de style nommée style.css illustrant l'utilisation de vw et de vh

```
1 body {
2   /*La taille du body correspond a 90% de la largeur du viewport.*/
3   width: 90vw;
4   border: solid;
5   /*La hauteur du body correspond a 60% de la hauteur du viewport.*/
6   height: 60vh;
7 }
8
9 div {
10  /*La hauteur des div correspond a 30% de la hauteur du viewport.*/
11  height: 30vh;
12  border: solid;
13 }
```

Exemple de page HTML à associer à la feuille CSS précédente :

```
1 <html>
2   <head>
3     <meta charset="utf-8">
4     <link href="style.css" rel="stylesheet">
5     <title>Exemple</title>
6   </head>
7   <body>
8     <div>
9     </div>
10  </body>
11 </html>
```



Méthode Unités en pourcentages

De nombreuses valeurs de propriétés CSS peuvent être exprimées en pourcentages de l'élément parent.

Cela signifie que leur valeur sera toujours relative à celle de leur parent : un élément dont la largeur est définie comme faisant 80 % aura une largeur effective correspondant à 80 % de la largeur de son parent, et ce quelle que soit sa valeur à un moment donné.

Par conséquent, des modifications dynamiques sur l'affichage de l'application impacteront automatiquement tous les éléments, leur permettant ainsi de conserver leur proportionnalité les uns envers les autres.

Exemple Feuille de style nommée style.css illustrant l'utilisation des pourcentages

```

1 body {
2   border-style: solid;
3 }
4
5 p {
6   /*La largeur de tous les éléments <p> correspond à 80% de la longueur de leur parent.*/
7   width: 80%;
8   border-style: solid;
9 }
10
11 div {
12   /*La largeur de tous les éléments <div> correspond à 80% de la longueur de leur parent.*/
13   width: 80%;
14   border-style: solid;
15 }

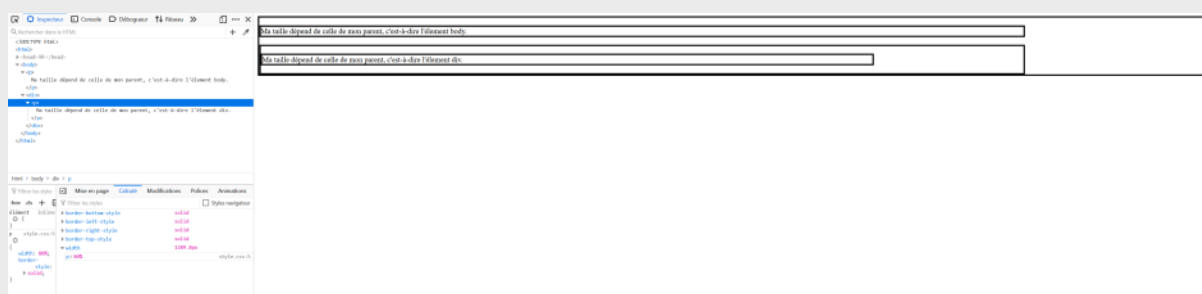
```

Exemple de page HTML à associer à la feuille CSS précédente :

```

1 <html>
2   <head>
3     <meta charset="utf-8">
4     <link href="style.css" rel="stylesheet">
5     <title>Exemple</title>
6   </head>
7   <body>
8     <p>Ma taille dépend de celle de mon parent, c'est-à-dire l'élément body.</p>
9     <div>
10      <p>Ma taille dépend de celle de mon parent, c'est-à-dire l'élément div.</p>
11    </div>
12  </body>
13 </html>

```



Fondamental

L'utilisation d'unités de longueur relatives signifie que les valeurs effectives des propriétés CSS auxquelles elles sont rattachées seront calculées à la volée, en fonction de la taille affichée de leur parent ou de leur objet racine dans le DOM. Cela a donc pour conséquence d'assurer une uniformité graphique selon les appareils ou les actions modifiant l'affichage de l'utilisateur. Si vous souhaitez dimensionner les éléments de manière cohérente sur l'ensemble de votre page, la mesure rem est la plus adaptée. Par contre si vous souhaitez dimensionner des éléments par rapport à d'autres éléments, l'unité em conviendra mieux.

III. Exercice : Appliquez la notion

Question

[solution n°1 p.23]

Pour mettre en application ce que nous venons de voir, vous allez créer une page HTML sur votre environnement de travail VSCode et écrire la feuille CSS associée.

Pour cela, vous réutiliserez le code HTML proposé ci-dessous et devrez écrire les classes CSS suivantes :

- `change-width` : elle devra afficher la bordure de l'élément et s'assurer que sa largeur soit toujours de 40 % de la largeur du parent.
- `change-height` : elle devra afficher la bordure de l'élément et s'assurer que sa hauteur soit toujours le double de la police de l'élément parent (`div`).
- `bottom-right` : elle devra placer l'élément à droite de la page, s'assurer que sa largeur soit toujours de la moitié de la largeur de la page, le colorer en rouge et faire en sorte que la police corresponde à 2 fois la taille spécifiée pour l'élément HTML.
- `change-viewport` : elle devra définir une taille de police correspondant à 7 % de la hauteur du viewport.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail VSCode.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Les unités relatives</title>
6     <link href="style.css" rel="stylesheet" type="text/css" />
7   </head>
8   <body>
9     <p class="change-width">Je change de taille si la taille de l'écran change</p>
10    <div>
11      <p class="change-height">Ma hauteur est égale a deux fois celle de la police de mon
12      parent</p>
13    </div>
14    <p class="bottom-right">Je suis en rouge et je reste toujours collé à droite</p>
15    <p class="change-viewport">La taille de ma police change avec le viewport</p>
16  </body>
17 </html>
```

Voici le résultat attendu :



La taille de ma police change avec le viewport

Indice :

Le positionnement à droite peut se faire par l'utilisation de `float : right;`.

IV. Les media queries

Objectifs

- Découvrir le fonctionnement des media queries
- Découvrir la syntaxe des media queries

Mise en situation

L'utilisation de **media queries** est une composante majeure du **responsive design**, car il s'agit de la meilleure solution pour mettre en place un style évoluant dynamiquement en fonction des contraintes des appareils et des actions de l'utilisateur. Nous allons voir pourquoi, en établissant tout d'abord une définition de ce dont il s'agit, puis en développant leur fonctionnement.

Un peu d'histoire

La compréhension des principes de fonctionnement liés aux media queries nécessite un petit retour en arrière sur l'histoire de l'adaptabilité du design.

En effet, il est possible, depuis HTML4 et CSS2, d'utiliser l'attribut `media` afin de cibler un style particulier en fonction de l'utilisation qui est faite d'une application. Par exemple, le `media screen` permet de cibler les appareils avec écran, tandis que le `media print` permettra de spécifier une feuille de style précise pour l'affichage avant impression et pour l'impression. Il existait alors 9 media auxquels une application web pouvait s'adapter.

L'évolution des technologies et la diversification des appareils utilisateurs de nos applications a nécessité la création d'une solution technique offrant une meilleure souplesse à ces applications : les **media queries**.

En effet, celles-ci ne se basent plus uniquement sur le media utilisé, mais permettent l'ajout de conditions spécifiques à l'appareil utilisé, afin de permettre une granularité plus fine de la gestion des styles.

Définition Media query

Une media query, telle que définie par la RFC W3C de CSS3, se caractérise par l'utilisation des mots-clefs `@media` ou `@import` et se traduit par un ensemble de conditions booléennes sur le media utilisé et/ou sur un ensemble de critères propres à l'appareil utilisé, appelés **media features**.

```
1 @media screen and (min-width : 1280px) {} /* Cette media query cible les écrans dont la
   largeur minimale est de 1280px */
```

Méthode La syntaxe

La construction d'une media query se décompose de la façon suivante :

- Spécification du media ciblé (optionnel)
- Utilisation d'un opérateur logique
- Définition de la media feature souhaitée

Il est ensuite possible d'ajouter autant de conditions que voulues, si on les chaîne avec un opérateur logique.

Les opérateurs logiques sont les suivants :

- - `and` : permet de combiner plusieurs conditions de media features en une seule. Pour que la règle soit appliquée, il faut que chaque condition séparée par `and` soit vérifiée.
- - `,` : permet de combiner plusieurs conditions en une seule. Pour que la règle soit appliquée, il faut qu'au moins une des conditions séparées par `,` soit vérifiée.
- - `not` : ce mot-clef s'applique à la totalité d'une media query et ne se vérifie que si tous les critères concernés par `not` ne sont pas vrais.
- - `only` : ce mot-clef permet de ne vérifier la requête que si la totalité de la media query est vérifiée. Il est utilisé par exemple pour forcer certains anciens navigateurs à vérifier l'entièreté d'une media query, plutôt que sa composante media uniquement.

Attention

Le mot-clef `not`, même s'il peut être placé avant le media, ne s'applique qu'aux media features.

Syntaxe

Les media queries peuvent s'utiliser en HTML, sous la forme suivante :

```
1 <link rel="stylesheet" media="screen and (max-width: 700px)" href="phone.css" type="text/css" />
```

Mais également en CSS, avec la syntaxe :

```
1 @media screen and (max-width: 700px) {
2   /* Écriture du code CSS souhaité pour les appareils avec écran et dont la largeur maximale est
   de 700px */
3 }
```

Exemple

```
1 /* La media query suivante importe la feuille de style exemple.css pour les écrans dont la
   largeur minimale est de 800px */
2 @import url("exemple.css") screen and (min-width: 800px);

1 /* La media query suivante s'applique pour tous les appareils dont la largeur minimale est de
   800px et au format paysage */
2 @media (min-width: 800px) and (orientation: landscape) { ... }

1 /*La media query suivante s'applique pour tous les appareils dont la hauteur minimale est de
   680px ou pour les écrans au format paysage */
2 @media (min-height: 680px), screen and (orientation: landscape) { ... }
```

Remarque

En CSS2, l'équivalent des media queries actuelles pouvait s'écrire de la façon suivante :

- Directement dans un fichier CSS

```
1 @media print {
2   .element{
3     width:80%;
4     font-size:15pt;
5   }
6 }
```

- En associant une feuille de style spécifique dans la page HTML

```
1 <link rel="stylesheet" media="print" href="print.css" type="text/css" />
```

On peut donc constater que les media queries actuelles ont été définies comme une amélioration de ce qui existait dans la version précédente de CSS, en ajoutant une plus grande flexibilité grâce à l'utilisation de conditions sur des media features.

Fondamental

Une media query est un ensemble de conditions basées éventuellement sur un type de media et/ou une ou plusieurs conditions sur des caractéristiques de celui-ci.

V. Exercice : Appliquez la notion

Question

[solution n°2 p.23]

Pour mettre en application ce que nous venons de voir, vous allez créer une page HTML sur votre environnement de travail VSCode et écrire la feuille CSS associée.

Pour cela, vous réutiliserez le code HTML ainsi que la feuille CSS proposés ci-dessous, et devrez écrire les media queries répondant aux questions suivantes :

Je fais les trois quarts de la taille de mon parent lors d'une impression

Je deviens rouge sur les grands écrans ou lorsque la hauteur de l'écran est inférieure à 800px

L'élément HTML avec la classe `first-block` se positionne en bas de la page si la largeur de l'écran est inférieure à 760 px.

Je passe en bas de la page sur les petits écrans

Je passe en bas de la page sur les petits écrans

Je fais les trois quarts de la taille de mon parent lors d'une impression

Je deviens rouge sur les grands écrans ou lorsque la hauteur de l'écran est inférieure à 800px

L'élément HTML avec la classe `second-block` ne doit faire que 75 % de la taille de son parent lors d'une impression.

Je passe en bas de la page sur les petits écrans

Je fais les trois quarts de la taille de mon parent lors d'une impression

Je deviens rouge sur les grands écrans ou lorsque la hauteur de l'écran est inférieure à 800px

L'élément HTML avec la classe `third-block` devient rouge lorsque la largeur de l'écran fait plus de 1024 px ou lorsque sa hauteur est inférieure à 800 px.

Il est recommandé d'agir sur les différents paramètres d'affichage de la fenêtre du navigateur afin de voir les modifications dynamiques du style grâce aux outils de développement des navigateurs.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail VSCode.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Application media queries</title>
7     <link href="style.css" rel="stylesheet" type="text/css" />
8   </head>
9   <body>
```

```

10 <p class="first-block">Je passe en bas de la page sur les petits écrans</p>
11 <div>
12 <p class="second-block">Je fais les trois quarts de la taille de mon parent lors d'une
impression </p>
13 </div>
14 <p class="third-block">Je deviens rouge sur les grands écrans ou lorsque la hauteur de
l'écran est inférieure à 800px</p>
15 </body>
16 </html>

```

```

1 .first-block {
2   border-style: solid;
3   width: 40%;
4 }
5
6 .second-block {
7   border-style: solid;
8   height: 2em;
9 }
10
11 .third-block {
12   float: right;
13   width: 50%;
14 }

```

Indice :

Le positionnement en bas de la page peut se faire par l'utilisation de `position: absolute;` et `bottom: 0;`.

VI. Les breakpoints

Objectifs

- Découvrir comment prévoir les media queries nécessaires
- Découvrir un nouveau paradigme de développement front

Mise en situation

Dans la pratique, c'est généralement au développeur qu'incombe la tâche de définition des media queries à mettre en place. Cette responsabilité peut être une grande source de confusion ou d'erreurs tout au long du projet.

C'est pourquoi, dans cette partie, nous allons nous attacher à décrire les méthodes pratiques à notre disposition pour déterminer les différentes media queries à mettre en place afin d'assurer un fonctionnement homogène de notre application web.

Au vu du nombre de media existants et des media features qui les composent, nous allons concentrer nos exemples et notre démonstration sur les éléments des media queries que nous rencontrons dans la très grande majorité des cas d'utilisation lors des développements d'applications web.

D'une manière générale, les méthodologies présentées ici restent applicables quels que soient le media ou la media feature ciblés.

Définition Breakpoint

On appelle **breakpoint** (ou « point de rupture ») la valeur d'une media feature pour laquelle l'affichage de l'application web change.

Méthode Définir ses breakpoints

Lors du développement d'une application web, nous allons généralement vouloir que celle-ci s'adapte à tous les supports. Nous allons donc devoir définir les conditions pour lesquelles tel ou tel style sera appliqué. Chacune de ces conditions constitue ce que l'on appelle un **breakpoint**.

Plusieurs approches sont possibles pour cela :

- La première consiste à écrire des media queries pour chaque appareil que l'on souhaite cibler : cette méthode nécessite de connaître la largeur et la hauteur en pixels de l'écran pour chacun d'entre eux, et ne permet pas de s'adapter à l'évolution technologique des appareils.
- La deuxième se base sur l'utilisation de quelques tailles "repères" : 320 px (mobiles), 768 px (tablettes/ordinateurs portable), 1200 px (ordinateurs de bureau), par exemple. Même si cette méthode peut sembler efficace, elle ne permet pas de s'assurer du comportement de notre application dans toutes les situations.
- La troisième consiste à se détacher des notions d'appareils pour se concentrer principalement sur le design de notre application en fonction de son comportement avec différentes résolutions. Dans la pratique, cela consiste à mettre en place le design souhaité, puis à observer son comportement lors du changement de taille de l'écran et à l'adapter à chaque fois qu'il se détériore. Cette méthode présente l'avantage de définir des styles qui resteront valables quelles que soient les évolutions technologiques des appareils. Cette approche est vitale pour l'approche de développement mobile first, que nous aborderons un peu plus loin.

Dans la majorité des situations, les breakpoints sont définis en fonction de la largeur de l'écran, mais les trois méthodes de détermination présentées sont applicables quelle que soit la caractéristique ciblée.

Une fois les différents breakpoints souhaités identifiés, il ne reste plus qu'à redéfinir les éléments de style nécessaires pour que le contenu de notre application reste consultable.

Exemple

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <link href="style.css" rel="stylesheet" type="text/css" />
6   </head>
7   <body>
8     <p class="block">Je suis un bloc de contenu</p>
9   </body>
10 </html>

1 .block {
2   border-style: solid;
3   height: 20px;
4 }
```

Lorsque je fais varier la largeur de l'écran, je constate une incohérence graphique à partir d'une largeur de 189 px. Je peux donc définir un breakpoint pour cette valeur.

Cela se traduit par la media query suivante :

```
1 @media screen and (max-width: 189px) {
2   .block {
3     height: 40px;
4   }
5 }
```

Exemple

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <link href="style.css" rel="stylesheet" type="text/css" />
6   </head>
7   <body>
8     <div class="bloc">Je suis un bloc coloré</div>
9   </body>
10 </html>

1 /* bloc de couleur bleue sur mobile et + */
2 .bloc {
3   width: 50px;
4   height: 50px;
5   background-color: blue;
6 }
7 /* bloc de couleur rouge à partir de 640px */
8 @media screen and (min-width: 640px) {
9   .bloc {
10    background-color: red;
11  }
12 }
13 /* bloc de couleur verte à partir de 800px */
14 @media screen and (min-width: 800px) {
15   .bloc {
16    background-color: green;
17  }
18 }

```

Fondamental

L'utilisation des media queries nécessite la définition des différents breakpoints de notre application web, grâce à plusieurs méthodes. Celle présentant la plus grande souplesse consiste à analyser les valeurs d'une media feature, le plus souvent la taille de l'écran, pour lesquelles le design n'est plus valide.

Complément **Portrait / Paysage**

Précédemment, nous avons vu comment gérer les différences de tailles d'écran sans nous préoccuper de l'orientation portrait/paysage de l'appareil. En effet, le passage de portrait à paysage modifie la largeur de l'écran et va par conséquent être pris en compte par les breakpoints déjà définis.

Il est cependant possible que des comportements spécifiques soient nécessaires en fonction de l'orientation. Pour cela, nous avons besoin de l'utilisation de la media feature `orientation`.

Cette media feature n'est pas toujours prise en compte dynamiquement par les navigateurs : cela signifie que, lors d'un changement d'orientation pendant l'utilisation de l'application, l'exécution du style souhaité via une media query ne sera pas toujours effective.

Pour palier ce problème, il est possible d'utiliser la méthode `window.matchMedia()` disponible en JavaScript.

Celle-ci requiert une media query en paramètre, et permet d'exécuter le code souhaité lorsqu'elle est vérifiée. On peut ainsi s'assurer du bon chargement d'une feuille CSS spécifique si nécessaire.

```

1 if (window.matchMedia("(orientation: landscape)").matches) {
2   /* Gestion de l'orientation en paysage */
3 } else {
4   /* Gestion de l'orientation en portrait */
5 }

```

Il est toutefois possible d'utiliser le css uniquement pour définir des style différents suivant l'orientation du viewport.

Exemple

```

1 <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="utf-8">
5       <link href="style.css" rel="stylesheet" type="text/css" />
6     </head>
7     <body>
8       <p class="paragraph">Je suis un bloc de texte</p>
9     </body>
10    </html>
11
12 html {
13   font-size : 100%;
14 }
15
16 .paragraph {
17   font-size: 1rem;
18 }
19
20 /* bloc de texte rouge en mode paysage */
21 @media (orientation: landscape) {
22   .paragraph {
23     color: red;
24   }
25 }
26
27 /* bloc de texte verte en mode portrait */
28 @media (orientation: portrait) {
29   .paragraph {
30     color: green;
31   }
32 }

```

VII. Exercice : Appliquez la notion

Question

[solution n°3 p.24]

Pour mettre en application ce que nous venons de voir, vous allez créer une page HTML sur votre environnement de travail VSCode et écrire la feuille CSS associée.

Pour cela, vous réutiliserez le code HTML et la feuille CSS proposés ci-dessous et devrez définir les breakpoints et les media queries permettant de répondre aux demandes suivantes.



Lorsque le contenu de l'élément HTML avec la classe `second-block` chevauche la bordure, sa hauteur change pour correspondre à 100 % de celle de son parent.

Résultat attendu :

Je suis un premier bloc de contenu Je suis un bloc de contenu et je me comporte mal lorsque la largeur de l'écran diminue Je suis un troisième bloc de contenu

Je suis un premier bloc de contenu Je suis un bloc de contenu et je me comporte mal lorsque la largeur de l'écran diminue Je suis un troisième bloc de contenu

Lorsque la hauteur des éléments HTML avec les classes `first-block` et `third-block` dépasse celle de l'élément HTML avec la classe `second-block`, celui-ci devient rouge.

Résultat attendu :

Je suis un premier bloc de contenu Je suis un bloc de contenu et je me comporte mal lorsque la largeur de l'écran diminue Je suis un troisième bloc de contenu

Je suis un premier bloc de contenu Je suis un bloc de contenu et je me comporte mal lorsque la largeur de l'écran diminue Je suis un troisième bloc de contenu

Lorsque la hauteur de l'élément HTML avec la classe `first-block` dépasse celle de l'élément HTML avec la classe `second-block`, celui-ci disparaît de la page.

Résultat attendu :

Je suis un premier bloc de contenu Je suis un troisième bloc de contenu

Il est recommandé d'agir sur les différents paramètres d'affichage de la fenêtre du navigateur afin de voir les modifications dynamiques du style grâce aux outils de développement des navigateurs.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail VSCode.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta name="viewport" content="width=device-width, initial-scale=1.0">
5     <meta charset="utf-8">
6     <title>Les breakpoints</title>
7     <link href="style.css" rel="stylesheet" type="text/css" />
8   </head>
9   <body>
10    <div class="main">
11      <p class="first-block">Je suis un premier bloc de contenu</p>
12      <p class="second-block">Je suis un bloc de contenu et je me comporte mal lorsque la
largeur de l'écran diminue </p>
13      <p class="third-block">Je suis un troisième bloc de contenu</p>
14    </div>
15  </body>
16 </html>

```

```

1 .first-block {
2   border-style: solid;
3   width: 40%;
4 }
5
6 .second-block {
7   border-style: solid;
8   width: 75%;
9 }
10
11 .third-block {
12   float: right;
13 }
14
15 .main {
16   display: flex;
17 }

```

VIII. Mobile-first

Objectifs

- Découvrir et comprendre le principe du mobile-first
- Découvrir comment mettre en pratique le mobile-first

Mise en situation

Le principe du mobile-first consiste à penser sa page web et son code en partant de l'écran le plus petit vers l'écran le plus grand. En effet, aujourd'hui les usages ont changé et avoir un site accessible sur mobile est indispensable.

Internet sur mobile étant plus lent de manière générale (même avec la fibre), c'est pourquoi le fait de concevoir sa page web en pensant mobile-first dès le début du développement permet d'optimiser le poids et le nombre d'éléments à charger.

Dans cette partie nous allons donc voir comment penser mobile-first et comment le mettre en pratique sur vos futurs projets.

Définition mobile-first

Le terme **mobile-first** (ou « mobile d'abord ») est une manière de développer son projet en faisant en sorte que l'affichage soit optimisé pour les mobiles et qu'il s'adapte progressivement aux écrans plus grands (tablette, ordinateur...).

Méthode Pensez mobile-first

Même si concevoir dès le début votre page pour les mobiles semble plus compliqué, il est nécessaire de le faire en premier. Une fois toutes les contraintes de design mobiles réglées, la conception sur d'autres tailles de support sera plus facile. En effet, une dégradation progressive du design pour que tout le contenu s'adapte aux petits écrans réduira grandement l'expérience utilisateur. Comme nous l'avons vu dans le chapitre précédent, nous allons utiliser des points de rupture (breakpoints) afin de définir le style en fonction du support. Le code sera écrit en partant sur la base de l'affichage mobile puis personnalisé selon ces breakpoints afin de le paramétrer sur les écrans plus grands.

Ce qui est important de prendre en compte, c'est que même si vous n'avez qu'une maquette desktop first, il est nécessaire de la démarrer en mobile en priorisant certains points:

- Commencer par la typographie.
- Pour chaque point de rupture, placer les modifications qu'il apporte dans un media screen min-width.
- Utiliser max-width pour paramétrer certains points quand c'est nécessaire.

Ainsi, les écrans aux formats plus petits (et donc potentiellement moins puissants) ne chargeront que le contenu nécessaire pour eux.

Les autres écrans, en revanche, récupéreront en plus tout le style précisé dans les points de rupture qui les concernent.

Exemple

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <link href="style.css" rel="stylesheet" type="text/css" />
6   </head>
7   <body>
8     <p class="text">Je suis un paragraphe en sans-sérif</p>
9   </body>
10 </html>

1 html {
2   font-size:100%;
3 }
4 /* réglages mobile et + */
5 .text {
6   font-family: sans-serif;
7   font-size: 1rem;
8 }
9 @media screen and (min-width: 640px) {
10  .text {
11    /* réglages à partir de 640px et + : même police et taille un peu plus grande. */
12    font-size: 1.3rem;
13  }
14 }
```

Fondamental

Le fait de penser mobile-first permet d'optimiser les performances du site, de prioriser les contenus importants et faire en sorte que l'expérience utilisateur soit complète quelle que soit la taille du support. La fluidité étant la base de l'expérience utilisateur sur téléphone, il faut que tout soit fait pour faciliter la navigation mobile, avec par exemple la mise en place d'un burger menu, d'un formulaire simple et accessible ou encore des boutons d'appels à l'action clairement identifiés.

IX. Auto-évaluation**A. Exercice final****Exercice 1 : Quiz - Culture**

[solution n°4 p.25]

Exercice

La gestion des styles en fonction des media utilisés est apparue avec CSS3.

- ☐ Vrai
- ☐ Faux

Exercice

Dans la pratique, il est conseillé d'écrire une media query suivant les caractéristiques de chaque appareil susceptible d'utiliser une application web.

- ☐ Vrai
- ☐ Faux

Exercice

Je peux...

- ☐ Utiliser uniquement des unités relatives en CSS pour déplacer des éléments graphiques
- ☐ Utiliser uniquement des unités relatives en CSS pour conserver la proportionnalité de mon affichage quelle que soit la taille de l'écran
- ☐ Combiner des media queries et l'utilisation d'unités relatives en CSS pour adapter le rendu d'une application web, le plus précisément possible, aux modifications d'affichage

Exercice 5 : Quiz - Méthode

[solution n°5 p.25]

Exercice

Parmi les media présentés, cochez les plus utilisés dans les media queries.

- ☐ print
- ☐ screen
- ☐ speech
- ☐ all

Exercice

Quels sont les opérateurs utilisables dans une media query ?

- ☐ and
- ☐ ()
- ☐ not
- ☐ only
- ☐ &&
- ☐ ,
- ☐ ||
- ☐ {}

Exercice

Parmi ces unités, lesquelles sont relatives aux valeurs des propriétés du parent de l'élément sur lequel elles s'appliquent ?

- ☐ px
- ☐ em
- ☐ pt
- ☐ vw
- ☐ vh
- ☐ cm
- ☐ %

Exercice 9 : Quiz - Code

[solution n°6 p.26]

Exercice

Quelle media query se vérifie pour les écrans inférieurs à 1024 px ?

- ☐ @media screen and (min-width : 1024px) {}
- ☐ @media screen and (max-width : 1024px) {}
- ☐ @media screen, (max-width : 1024px) {}

Exercice

Quand est-ce que s'applique la media query suivante `@media screen and (min-width: 480px) and (orientation: landscape) { ... }` ?

- ☐ Si la largeur de l'écran est d'au moins 480 px et au format paysage
- ☐ Si la largeur de l'écran est d'au plus de 480 px et au format paysage
- ☐ Si la largeur de l'écran est d'au moins 480 px
- ☐ Si l'écran est au format paysage

Exercice

- ☐ Si l'écran a une hauteur de 380 px et au format paysage
- ☐ Si la hauteur de l'écran est de 360 px et qu'il est au format portrait
- ☐ Si l'écran a une hauteur d'au moins 680 px et qu'il est au format paysage
- ☐ Si la page imprimée fait 850 px de haut

- ☐ Si la page imprimée fait 380 px de large
- ☐ Si l'écran fait 799 px de large
- ☐ Si la page imprimée fait 799 px de large

[solution n°7 p.27]

Réalisez une liste HTML composée de 5 éléments textuels contenant un extrait du *lorem ipsum*.

Lorsque le texte descend trop bas, cachez-le et n'affichez qu'une seule ligne, suivie de...

Premier titre

Deuxième titre

Troisième titre

Quatrième titre

Cinquième titre

Lorsque ce texte devient illisible, réduisez la liste à un menu de 5 éléments, suivi d'un titre.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail VSCode.

Indice :

Le lorem ipsum est un texte dénué de sens utilisé en imprimerie pour permettre de visualiser les mises en page à l'avance.

Exemple de Lorem ipsum :

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Non sodales neque sodales ut etiam sit amet nisl. Elementum pulvinar etiam non quam lacus. Id leo in vitae turpis massa. Euismod quis viverra nibh cras pulvinar mattis nunc sed. Sed risus pretium quam vulputate dignissim suspendisse. Elementum nibh tellus molestie nunc. Habitant morbi tristique senectus et netus et malesuada. Sed pulvinar proin gravida hendrerit lectus. Dui ut ornare lectus sit amet est placerat in egestas. Commodore odio aenean sed adipiscing diam donec adipiscing tristique risus. Amet consectetur adipiscing elit ut aliquam purus sit. Egestas erat imperdiet sed euismod nisi porta. Hac habitasse platea dictumst vestibulum rhoncus est. Vitae ultricies leo integer malesuada nunc vel risus commodo.

Indice :

L'affichage d'une ligne coupée par ... peut se faire en CSS grâce à `text-overflow: ellipsis;`.

Solutions des exercices

p. 8 Solution n°1

```
1 .change-width {
2   border-style: solid;
3   width: 40%;
4 }
5
6 div {
7   font-size: 16px; /Cette valeur peut être changée pour modifier la hauteur des éléments ayant
   la classe change-height/
8 }
9
10 .change-height {
11   border-style: solid;
12   height: 2em;
13 }
14
15 .bottom-right {
16   font-size: 2rem;
17   float: right;
18   background: red;
19   width: 50%;
20 }
21
22 .change-viewport {
23   font-size: 7vh;
24 }
```

p. 11 Solution n°2

```
1 .first-block {
2   border-style: solid;
3   width: 40%;
4 }
5
6 .second-block {
7   border-style: solid;
8   height: 2em;
9 }
10
11 .third-block {
12   float: right;
13   width: 50%;
14 }
15
16 @media screen and (max-width: 760px) {
17   .first-block {
18     border-style: solid;
19     width: 40%;
20     position: absolute;
21     bottom: 0;
22   }
23 }
24
25 @media print {
```

```

26 .second-block {
27     border-style: solid;
28     height: 2em;
29     width: 75%;
30 }
31 }
32
33 @media screen and (min-width: 1024px), (max-height: 800px) {
34     .third-block {
35         float: right;
36         background: red;
37         width: 50%;
38     }
39 }
40

```

p. 15 Solution n°3

```

1 .first-block {
2     border-style: solid;
3     width: 40%;
4 }
5
6 .second-block {
7     border-style: solid;
8     height: 2em;
9 }
10
11 .third-block {
12     float: right;
13 }
14
15 .main {
16     display: flex;
17 }
18
19 @media screen and (max-width: 1341px) {
20     .second-block {
21         border-style: solid;
22         height: 100%;
23     }
24 }
25
26 @media screen and (max-width: 576px) {
27     .second-block {
28         border-style: solid;
29         height: 100%;
30         background: red;
31     }
32 }
33
34 @media screen and (max-width: 464px) {
35     .second-block {
36         display: none;
37     }
38 }


```


Exercice p. 19 Solution n°4**Exercice**

La gestion des styles en fonction des media utilisés est apparue avec CSS3.

☐ Vrai

☒ Faux


 La spécification d'une feuille de style particulière en fonction des media existe depuis CSS2, avec l'utilisation de l'attribut media.

Exercice

Dans la pratique, il est conseillé d'écrire une media query suivant les caractéristiques de chaque appareil susceptible d'utiliser une application web.

☐ Vrai

☒ Faux

 Même si cette méthode est valide, elle impliquerait d'écrire autant de styles qu'il y a d'appareils potentiels, et de devoir suivre les évolutions de ces derniers pour que l'application reste utilisable sur la durée.

Exercice

Je peux...

☐ Utiliser uniquement des unités relatives en CSS pour déplacer des éléments graphiques

☒ Utiliser uniquement des unités relatives en CSS pour conserver la proportionnalité de mon affichage quelle que soit la taille de l'écran

☒ Combiner des media queries et l'utilisation d'unités relatives en CSS pour adapter le rendu d'une application web, le plus précisément possible, aux modifications d'affichage

Exercice p. 19 Solution n°5**Exercice**

Parmi les media présentés, cochez les plus utilisés dans les media queries.

☒ print

☒ screen

☐ speech

☒ all

Il s'agit en effet d'un type de media qui correspond à n'importe quel media existant. C'est aussi la valeur par défaut si aucun media n'est spécifié dans une media query.

Exercice

Quels sont les opérateurs utilisables dans une media query ?

- ☒ and
- ☐ ()
- ☒ not
- ☒ only
- ☐ &&
- ☒ ,
- ☐ ||
- ☐ {}

Exercice

Parmi ces unités, lesquelles sont relatives aux valeurs des propriétés du parent de l'élément sur lequel elles s'appliquent ?

- ☐ px
- ☒ em
- ☐ pt
- ☐ vw
- ☐ vh
- ☐ cm
- ☒ %

Exercice p. 20 Solution n°6

Exercice

Quelle media query se vérifie pour les écrans inférieurs à 1024 px ?

- ☐ @media screen and (min-width : 1024px) {}
- ☒ @media screen and (max-width : 1024px) {}
- ☐ @media screen, (max-width : 1024px) {}

Exercice

Quand est-ce que s'applique la media query suivante @media screen and (min-width: 480px) and (orientation: landscape) { ... } ?

- ☒ Si la largeur de l'écran est d'au moins 480 px et au format paysage
- ☐ Si la largeur de l'écran est d'au plus de 480 px et au format paysage
- ☐ Si la largeur de l'écran est d'au moins 480 px
- ☐ Si l'écran est au format paysage

Exercice

Quand est-ce que s'applique la media query suivante `@media (min-height: 680px), screen and (orientation: portrait) { ... }`?

- ☐ Si l'écran a une hauteur de 380 px et au format paysage
- ☒ Si la hauteur de l'écran est de 360 px et qu'il est au format portrait
La media query s'applique, car la seconde condition `screen and (orientation: portrait)` est vérifiée.
- ☒ Si l'écran a une hauteur d'au moins 680 px et qu'il est au format paysage
La media query s'applique, car la première condition `(min-height: 680px)` est vérifiée.
- ☒ Si la page imprimée fait 850 px de haut
La première condition n'ayant pas de media, elle s'applique pour tous les media.

Exercice

Quand est-ce que s'applique la media query suivante `@media not screen and (max-width: 800px), print and (max-width: 400px) { ... }`?

- ☒ Si la page imprimée fait 380 px de large
- ☐ Si l'écran fait 799 px de large
- ☒ Si la page imprimée fait 799 px de large

p. 21 Solution n°7**Proposition de fichier HTML**

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Défi</title>
6     <link href="style.css" rel="stylesheet" type="text/css" />
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   </head>
9   <body>
10    <script src="script.js"></script>
11    <ul class="list">
12      <li class="list-element first">
13        <div class="menu">
14          <p class="title">Premier titre</p>
15        </div>
16        <p class="text-element">
17          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
18          incididunt ut labore et dolore magna aliqua. Mi bibendum neque egestas congue quisque
19          egestas. Velit egestas dui id ornare arcu odio ut. Tincidunt id aliquet risus feugiat in
20          ante. Purus in mollis nunc sed id. Enim facilisis gravida neque convallis a. Mauris
21          pellentesque pulvinar pellentesque habitant. Commodo sed egestas egestas fringilla. Eros
          donec ac odio tempor orci dapibus ultrices in. Porta nibh venenatis cras sed. Convallis
          posuere morbi leo urna. Lacinia at quis risus sed vulputate odio ut. Faciliis morbi tempus
          iaculis urna id volutpat lacus laoreet. Semper risus in hendrerit gravida rutrum quisque non
          tellus orci. Risus at ultrices mi tempus imperdiet nulla. Adipiscing at in tellus integer.
          Sit amet est placerat in egestas erat imperdiet sed. Phasellus egestas tellus rutrum tellus
          pellentesque. Scelerisque mauris pellentesque pulvinar pellentesque habitant.
18        </p>
19      </li>
20      <li class="list-element second">
21        <div class="menu">
```

```

22     <span class="glyphicon glyphicon-plus"></span>
23     <p class="title">Deuxième titre</p>
24 </div>
25     <p class="text-element">
26         Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
        incididunt ut labore et dolore magna aliqua. Mi bibendum neque egestas congue quisque
        egestas. Velit egestas dui id ornare arcu odio ut. Tincidunt id aliquet risus feugiat in
        ante. Purus in mollis nunc sed id. Enim facilisis gravida neque convallis a. Mauris
        pellentesque pulvinar pellentesque habitant. Commodo sed egestas egestas fringilla. Eros
        donec ac odio tempor orci dapibus ultrices in. Porta nibh venenatis cras sed. Convallis
        posuere morbi leo urna. Lacinia at quis risus sed vulputate odio ut. Facilisi morbi tempus
        iaculis urna id volutpat lacus laoreet. Semper risus in hendrerit gravida rutrum quisque non
        tellus orci. Risus at ultrices mi tempus imperdiet nulla. Adipiscing at in tellus integer.
        Sit amet est placerat in egestas erat imperdiet sed. Phasellus egestas tellus rutrum tellus
        pellentesque. Scelerisque mauris pellentesque pulvinar pellentesque habitant.
27     </p>
28 </li>
29     <li class="list-element third">
30         <div class="menu">
31             <p class="title">Troisième titre</p>
32         </div>
33         <p class="text-element">
34             Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
            incididunt ut labore et dolore magna aliqua. Mi bibendum neque egestas congue quisque
            egestas. Velit egestas dui id ornare arcu odio ut. Tincidunt id aliquet risus feugiat in
            ante. Purus in mollis nunc sed id. Enim facilisis gravida neque convallis a. Mauris
            pellentesque pulvinar pellentesque habitant. Commodo sed egestas egestas fringilla. Eros
            donec ac odio tempor orci dapibus ultrices in. Porta nibh venenatis cras sed. Convallis
            posuere morbi leo urna. Lacinia at quis risus sed vulputate odio ut. Facilisi morbi tempus
            iaculis urna id volutpat lacus laoreet. Semper risus in hendrerit gravida rutrum quisque non
            tellus orci. Risus at ultrices mi tempus imperdiet nulla. Adipiscing at in tellus integer.
            Sit amet est placerat in egestas erat imperdiet sed. Phasellus egestas tellus rutrum tellus
            pellentesque. Scelerisque mauris pellentesque pulvinar pellentesque habitant.
35         </p>
36     </li>
37     <li class="list-element fourth">
38         <div class="menu">
39             <p class="title">Quatrième titre</p>
40         </div>
41         <p class="text-element">
42             Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
            incididunt ut labore et dolore magna aliqua. Mi bibendum neque egestas congue quisque
            egestas. Velit egestas dui id ornare arcu odio ut. Tincidunt id aliquet risus feugiat in
            ante. Purus in mollis nunc sed id. Enim facilisis gravida neque convallis a. Mauris
            pellentesque pulvinar pellentesque habitant. Commodo sed egestas egestas fringilla. Eros
            donec ac odio tempor orci dapibus ultrices in. Porta nibh venenatis cras sed. Convallis
            posuere morbi leo urna. Lacinia at quis risus sed vulputate odio ut. Facilisi morbi tempus
            iaculis urna id volutpat lacus laoreet. Semper risus in hendrerit gravida rutrum quisque non
            tellus orci. Risus at ultrices mi tempus imperdiet nulla. Adipiscing at in tellus integer.
            Sit amet est placerat in egestas erat imperdiet sed. Phasellus egestas tellus rutrum tellus
            pellentesque. Scelerisque mauris pellentesque pulvinar pellentesque habitant.
43         </p>
44     </li>
45     <li class="list-element fifth">
46         <div class="menu">
47             <p class="title">Cinquième titre</p>
48         </div>
49         <p class="text-element">
50             Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
            incididunt ut labore et dolore magna aliqua. Mi bibendum neque egestas congue quisque
            egestas. Velit egestas dui id ornare arcu odio ut. Tincidunt id aliquet risus feugiat in
            ante. Purus in mollis nunc sed id. Enim facilisis gravida neque convallis a. Mauris
            pellentesque pulvinar pellentesque habitant. Commodo sed egestas egestas fringilla. Eros
            donec ac odio tempor orci dapibus ultrices in. Porta nibh venenatis cras sed. Convallis
            posuere morbi leo urna. Lacinia at quis risus sed vulputate odio ut. Facilisi morbi tempus
            iaculis urna id volutpat lacus laoreet. Semper risus in hendrerit gravida rutrum quisque non
            tellus orci. Risus at ultrices mi tempus imperdiet nulla. Adipiscing at in tellus integer.
            Sit amet est placerat in egestas erat imperdiet sed. Phasellus egestas tellus rutrum tellus
            pellentesque. Scelerisque mauris pellentesque pulvinar pellentesque habitant.
51         </p>
52     </li>
53 </ul>
54 </body>

```

```
55 </html>
```

Proposition de feuille CSS associée :

```
1  li {
2    width: 90%;
3    list-style-type: none;
4    margin: 15px 0;
5  }
6
7  li:hover {
8    cursor: pointer;
9  }
10 .text-element {
11   display: none;
12   padding: 10px 0;
13   width: 100%;
14   text-overflow: ellipsis;
15   overflow: hidden;
16   white-space: nowrap;
17 }
18
19 .title{
20   padding: 0 10px;
21 }
22
23 @media screen and (min-width: 741px) {
24   .menu{
25     display: none;
26   }
27
28   .text-element {
29     display: block;
30   }
31 }
32
33 @media screen and (min-width: 1024px) {
34   li {
35     list-style-type: initial;
36   }
37
38   .text-element {
39     white-space: normal;
40   }
41 }
```