

Les styles et l'approche responsive

Technologies web



Le style avec HTML

Les styles et l'approche responsive



Le CSS en quelques mots...

- Le **CSS** (pour **Cascading Style Sheets**) est un **langage** utilisé pour gérer le **style** et le format d'une **page web**
- Il permet de contrôler les **couleurs**, les **polices**, la **taille du texte**, l'**espacement entre les éléments**, la **position des éléments**, leur **façon de s'afficher** en fonction de la **taille de l'écran**, etc.
- Le **CSS** se présente de 3 façons :
 - **En ligne** : en utilisant l'**attribut style** dans un **élément HTML**
 - **En interne** : au sein de **balises <style></style>** situées dans la **section head** du **document**
 - **En externe** : avec une **balise <link>** située dans la **section head** du **document** et pointant vers un **fichier CSS**

En ligne - L'attribut **style**

- Comme son nom l'indique l'**attribut style** permet d'appliquer un **style** à un **élément HTML**
- Il est possible de changer sa **couleur**, sa **police**, sa **taille**, etc.

```
<p>Je suis un paragraphe normal</p>
<p style="color: red;">Je suis un paragraphe écrit en rouge</p>
<p style="font-size:50px;">Je suis un paragraphe écrit avec une police importante</p>
```

- En réalité, l'**attribut style** est un morceau de **CSS** imbriqué dans un **document HTML** - il faut donc respecter sa syntaxe (ne pas oublier le **point-virgule** à la fin de chaque **règle CSS**)
- Le **style** dans un **document HTML** va toujours respecter cette **syntaxe** :

```
<balise style="propriété: valeur;">
```

Quelques propriétés de style

- `background-color: red` : Définit une **couleur d'arrière plan** (accepte une des **140 couleurs prédéfinies** en HTML, des valeurs RGB ou hexadécimales - Cf. diapo suivante)
- `color: rgb(0, 0, 255)` : Définit la **couleur du texte**
- `font-family: Arial` : Définit la **police** d'un élément
- `font-size: 12px` : Définit la **taille de la police** d'un élément (en `%` ou en `px`)
- `text-align: center` : Définit l'**alignement horizontal du texte** d'un élément (`left`, `center`, `right`, `justify`, etc.)
- `width: 90%` : Définit la **largeur** d'un **élément** (en `%` ou en `px`)

Les couleurs

- HTML supporte plusieurs formats pour définir la couleur (**rgb**, **hexadécimal**, etc.)
- Il existe également **140 couleurs prédéfinies** (`tomato`, `orange`, `slateblue`, `gray`, etc.) - La liste complète est disponible à cette [adresse](#)
- Le **format RGB** (pour **Red Green Blue**) utilise des valeurs allant **de 0 à 255** pour chaque **canal de couleur** (**Exemple** : `rgb(60, 179, 113)`)
- Le **format HEX** attribue une valeur **hexadécimale** de la forme `#rrggbb` (**Exemple** : `#5dc647`)

Le CSS en externe (1/2)

- Il s'agit de la pratique recommandée pour écrire le code CSS afin de ne pas surcharger les **documents HTML**
- Il faut penser à lier le **document HTML** à un ou plusieurs **fichiers CSS** - Commençons par le lier à un seul à l'aide de la **balise link** :

```
1  <!DOCTYPE html>
2  <html lang="fr-FR">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <link rel="stylesheet" href=".//styles/style.css">
9      <title>Mon document HTML stylisé</title>
10 </head>
11
12 <body>
13     <div class="container">
14         <p id="mon-paragraphe">Mon super paragraphe</p>
15         <a>Mon super lien</a>
16     </div>
17 </body>
18
19 </html>
```



Le CSS en externe (2/2)

- Voici ce que l'on pourrait écrire dans le fichier `styles/style.css` pour obtenir le résultat suivant :

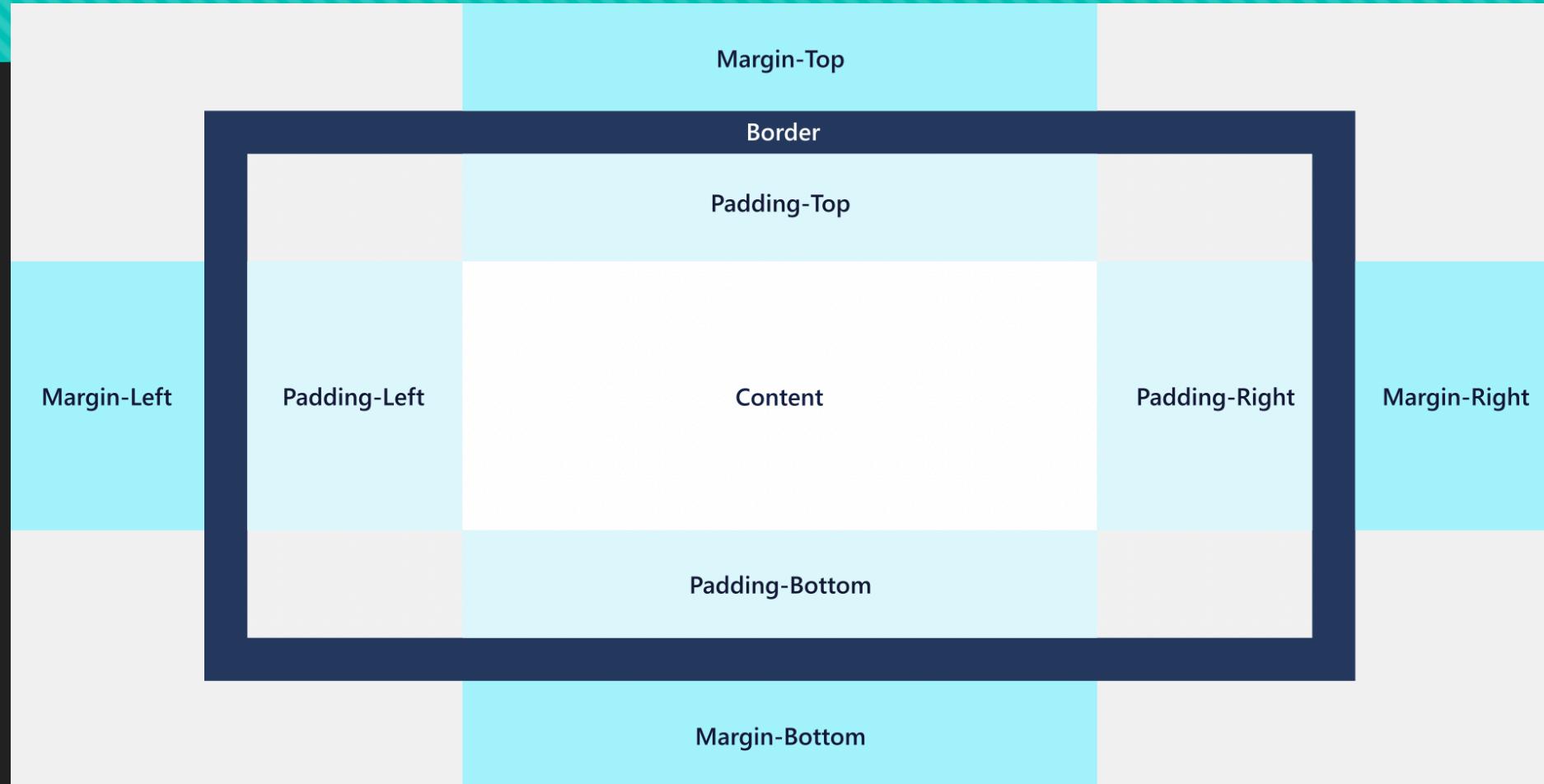
```
1 .container {  
2     background-color: #dodgerblue;  
3     color: #seagreen;  
4 }  
5  
6 #mon-paragraphe {  
7     font-weight: bold;  
8 }  
9  
10 a {  
11     color: #orange;  
12 }
```

Mon super paragraphe
[Mon super lien](#)

margin, border et padding

- La **propriété margin** représente l'**espace vide** laissé **autour** d'un **élément HTML**
- La **propriété border** représente les **bordures** d'un **élément HTML**
- La **propriété padding** représente l'**espace entre** un **élément HTML** et sa **bordure**

margin, border et padding en image



margin, border et padding - Exemple

- Ces **propriété** peuvent être ajustées en leur donnant des **valeurs numériques** en **px**, **%** ou **em** ou encore avec une valeur **auto** pour la **propriété margin**
- Les 4 cotés d'un **élément HTML** peuvent être réglés indépendamment ou en même temps dans une seule **propriété CSS**
- La **propriété padding: 25px** définit le **padding** des 4 côtés de l'**élément HTML** à **25px**
- La **propriété margin: 0 10px 0 10px** définit les **marges** du haut et du bas de l'**élément HTML** à **0** tandis que les **marges** gauche et droite sont définie à **10px**
- Les **propriétés** sont lues dans le sens des aiguilles d'une montre (haut, droite, bas, gauche)



En externe

- Il s'agit de la méthode la plus utilisée puisqu'elle permet de ne modifier qu'un seul **document CSS** pour changer toute l'apparence d'un site plutôt que d'appliquer les changements sur chaque page séparément
- Pour utiliser une **feuille de style CSS en externe** il faut indiquer un **lien** pointant vers ce **fichier** dans la **section head** du **document HTML**

```
3  <head>
4  |   <link rel="stylesheet" href="styles.css">
5  </head>
```

- L'**attribut** **rel** permet d'indiquer le **type de relation**, ici il précise que nous allons charger une **feuille de style** ce qui permet au **navigateur** de savoir quoi faire avec ce **lien**



L'attribut `class` en quelques mots...

- L'**attribut `class`** sert, le plus souvent, à pointer un **nom de classe** dans une **feuille de style CSS** pour attribuer à l'**élément HTML** certaines **propriétés**
- Il peut également être utilisé en **JavaScript** pour manipuler des **éléments**
- Il est possible d'utiliser la même **classe** sur plusieurs **éléments HTML** ainsi que d'attribuer plusieurs **classes** à un même **élément**
- En **CSS**, une référence à une **classe** est indiquée par un **point `.` avant son nom**



L'attribut **class** en image...

```
1  <!DOCTYPE html>
2  <html lang="fr-FR">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  |   <link rel="stylesheet" href="./styles/style.css">
7  |   <title>Mon document HTML stylisé</title>
8  </head>
9  <body>
10 |   <h1 class="ma-classe autre-classe">Mon titre</h1>
11 </body>
12 </html>
```

```
1  body {
2  |   background-color: #gray;
3  }
4
5  .ma-classe {
6  |   color: #lightgray;
7  }
8
9  .autre-class {
10 |   font-size: 18px;
11 }
```

Mon titre



L'attribut `id` en quelques mots...

- Contrairement à une **classe** qui peut être attribuée à plusieurs **éléments HTML**, l'attribut `id` pointe vers un **élément unique** avec un **identifiant** qui lui est propre
- En **CSS**, une référence à un `id` est indiquée par un **dièse # avant son nom**



L'attribut **id** en image...

```
1  <!DOCTYPE html>
2  <html lang="fr-FR">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  |   <link rel="stylesheet" href="./styles/style.css">
7  |   <title>Mon document HTML stylisé</title>
8  </head>
9  <body>
10 |   <h1 class="ma-classe" id="mon-id">Mon titre</h1>
11 </body>
12 </html>
```

```
1  body {
2  |   background-color: #gray;
3  }
4
5  .ma-classe {
6  |   color: #lightgray;
7  }
8
9  #mon-id {
10 |   font-size: 18px;
11 }
```

Mon titre

Le HTML au service du responsive

Les styles et l'approche responsive

Design responsive

- Le **design responsive** vise à créer des **pages adaptatives** au **format de l'écran** depuis lequel elles sont consultées
- Pour cela, des **règles HTML et CSS** sont utilisées pour agrandir, réduire ou cacher des éléments afin que l'intégralité de la page soit accessible et lisible depuis un écran de n'importe quel dimensions (ordinateur, tablette, téléphone, etc.)
- Un **élément meta** permet de régler la **taille de la partie visible d'une page web** dans un navigateur
`<meta name="viewport" content="width=device-width, initial-scale=1.0">`
- Cette simple **balise** déclare que la **largeur d'affichage maximale** correspond à celle de l'appareil lisant la page et que le **zoom initial** est de **1.0** (soit **aucun zoom**)
- Elle permet ainsi de s'assurer du **bon affichage** de la page **peu importe l'appareil utilisé**

Les balises <picture></picture>

- L'élément **picture** offre plus de souplesse que **img**
 - Il permet de spécifier **plusieurs images** à utiliser selon le cas (par exemple, en fonction de la **largeur de la fenêtre** ou en fonction du support de certains formats)
 - On peut assigner **plusieurs sources** et le navigateur choisira la première remplissant les conditions pour l'afficher
- ```
8 <picture>
 9 <source media="(min-width: 650px)" srcset="./images/logo1.png">
10 <source media="(min-width: 400px)" srcset="./images/logo2.png">
11
12 </picture>
```
- L'utilisation de l'**attribut srcset** dans la **balise <source>** remplace l'**attribut src** de **img** et permet de spécifier plusieurs sources pour une même image en fonction de la situation (peut éviter de charger une image de **1280px** de large sur un écran qui n'en fait que **480px**, par exemple)

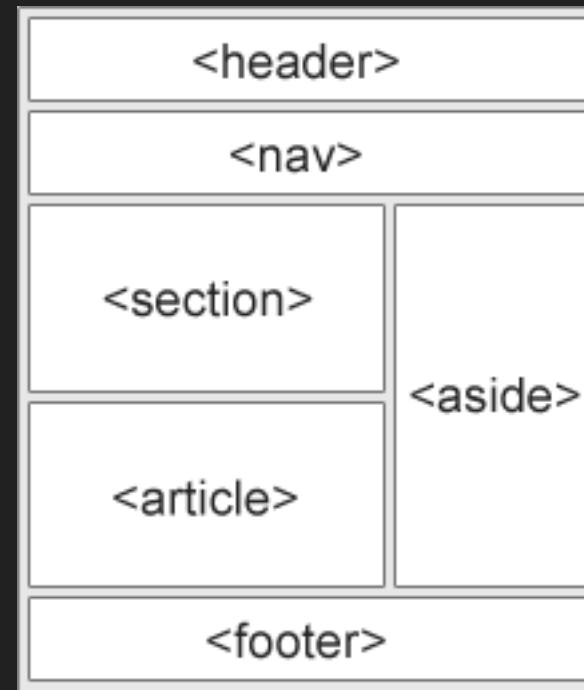
# Les balises <iframe></iframe>

- L'**élément** `iframe` permet d'incorporer une page web dans une autre  
`<iframe src="url" name="nom" title="description"></iframe>`
- L'**attribut** `src` définit l'**adresse web** à incorporer dans le `iframe`
- L'**attribut** `name` permet d'identifier le `iframe`
- L'**attribut** `title` permet de donner un titre au `iframe` (utile pour les options d'accessibilité pour les malvoyants, par exemple)

# Agencer une page HTML (1/2)

- Il existe des **balises** permettant de décrire la structure d'une **page HTML**, il s'agit des **balises sémantiques** :
  - `<header></header>` pour **l'entête**
  - `<nav></nav>` pour la **navigation**
  - `<section></section>` pour la **partie centrale** du document
  - `<article></article>` pour une **partie indépendante**
  - `<aside></aside>` pour du **contenu placé sur le côté**
  - `<footer></footer>` pour le pied de page
  - `<details></details>` pour du contenu que l'utilisateur **peut ouvrir et fermer**
  - `<summary></summary>` pour **filtrer l'élément** *details*

# Agencer une page HTML (2/2)



# Les images responsives

- En définissant la **propriété CSS** `width` d'une **image** en `%`, sa **taille** s'ajustera aux **dimensions de l'élément parent**
- Il est également possible d'utiliser la **propriété CSS** `max-width` d'une **image** conjointement à une **taille fixe** (défini en `px`)
- De cette manière, la **taille** de l'**image** diminue en fonction des **dimensions de l'écran** sans jamais excéder la **taille** saisie
- **Rappel :** En utilisant l'**élément** `picture` il est possible d'assigner une image différente en fonction de la taille de l'écran

# Les media queries (1/3)

- Les **media queries** permettent de modifier l'apparence d'un **site** ou d'une **application web** en fonction des **dimensions de l'écran** sur lequel il est consulté
- Elles peuvent être utilisées dans le but d'appliquer un **style conditionnel** mais également pour connaître la **largeur** et la **hauteur** de la zone visible d'une **page web**, de l'appareil utilisé ou encore son **orientation (portrait ou paysage)**
- Une **media query** représente un type de média et peut contenir une ou plusieurs expressions

```
@media mediatype and (expressions) {
 /*CSS-Code;*/
}
```

# Les media queries (2/3)

- Si le **type de média** spécifié correspond à l'appareil utilisé pour consulter la page web et si toutes les expressions passées en argument sont vraies, alors la requête est vraie
- Lorsqu'une **media query** est vraie, le code **CSS** qu'elle contient est interprété par le **navigateur web**
- Le **type de support** est facultatif - Le type « **all** » est celui déclaré par défaut
- Des **feuilles de style CSS** peuvent également être déclarées en respectant une **media query**

```
<link rel="stylesheet" media="mediatype and (expressions)" href="style.css">
```

# Les media queries (3/3)

- Les **types de média** possibles :
  - `all` : Utilisé pour tous les types de média (option par défaut)
  - `print` : Utilisé pour les imprimantes
  - `screen` : Utilisé pour les écrans d'ordinateur, de tablette et de téléphone
  - `speech` : Utilisé pour les lecteurs d'écran (pour les malvoyants)
- Une bonne façon d'utiliser les **media queries** consiste à dédier une section de votre fichier **CSS**

# Les media queries en image...

- O L'exemple suivant change la **couleur de fond** du **document** en **vert clair** seulement si la fenêtre de visualisation a une largeur supérieure à **480px**

```
@media screen and (min-width: 480px) {
 body {
 background-color: lightgreen;
 }
}
```

# Les fichiers multimédia

**Les styles et l'approche responsive**

# Les balises <video></video> (1/2)

- L'élément HTML <video> ressemble à l'élément <picture> dans sa structure

```
<video width="320" height="240" controls>
 <source src="/video/mov_bbb.mp4" type="video/mp4">
 <source src="/video/mov_bbb.ogg" type="video/ogg">
 Votre navigateur ne supporte pas la video.
</video>
```

- L'attribut **controls** ajoute au lecteur des **contrôles** pour **lire**, mettre en **pause** ou encore modifier le **volume** audio de la vidéo lue
- L'élément **source** permet à votre navigateur de définir le format supporté
- Le texte enfant de **video** ne sera affiché que si le navigateur ne supporte aucune des sources données

# Les balises <video></video> (2/2)

- Notez que contrairement à **picture**, on utilise l'**attribut src** et non **srcset**
- Il est possible d'intégrer des **sous-titres** à la **vidéo** à l'aide de la **balise <track>**

```
<video width="320" height="240" controls>
 <source src="video.mp4" type="video/mp4">
 <source src="video.ogg" type="video/ogg">
 <track src="subtitles_en.vtt" kind="subtitles" srclang="en" label="English">
 <track src="subtitles_no.vtt" kind="subtitles" srclang="no" label="Norwegian">
</video>
```

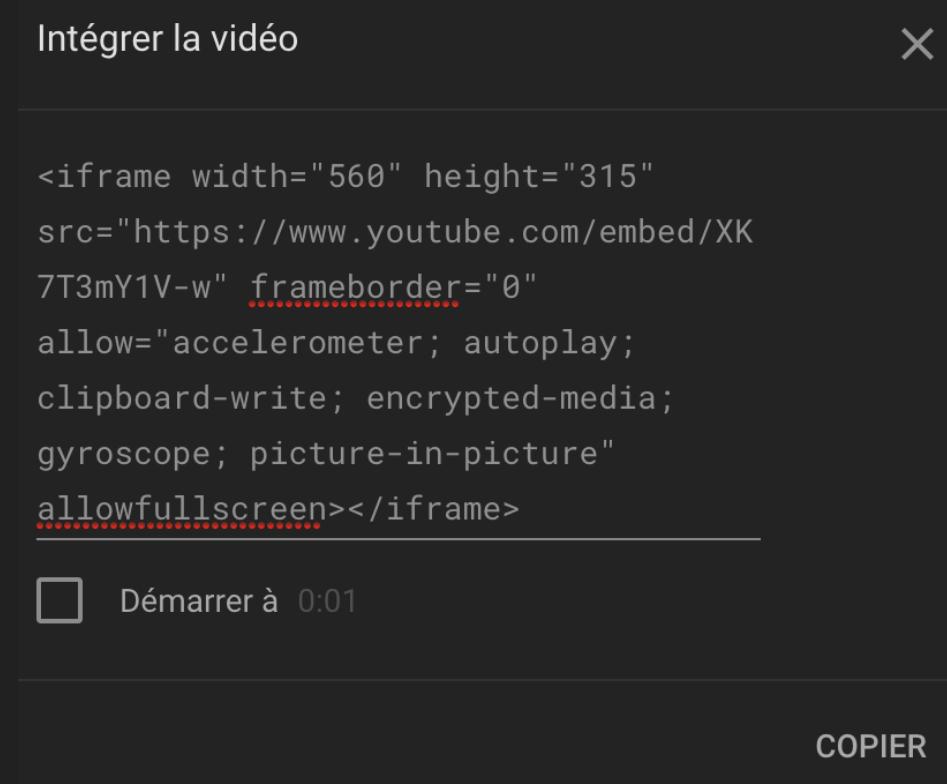
# Les balises `<audio></audio>`

- O L'**audio** s'intègre à un **document HTML** exactement de la même manière qu'une **video**

```
<audio controls>
 <source src="audio.ogg" type="audio/ogg">
 <source src="audio.mp3" type="audio/mpeg">
 Votre navigateur ne supporte pas l'audio.
</audio>
```

# Intégrer une vidéo Youtube (1/2)

- Les vidéos **Youtube** s'intègrent en utilisant un **élément `iframe`** fourni directement par le site (**Partager > Intégrer > Copier**)



# Intégrer une vidéo Youtube (2/2)

- Il est possible d'ajouter des options à la suite de l'**URL** de l'**attribut** **src** pour préciser comment la **vidéo** doit être jouée

```
<iframe width="560" height="315" src="https://www.youtube.com/embed/XK7T3mY1V-w?controls=0&autoplay=1&mute=1"></iframe>
```

- Ici, elle sera lancée automatiquement sans le **son** ni les **contrôles**
- Depuis **2018, Google Chrome** n'autorise plus les **vidéos** en lecture automatique si elles ne sont pas **mises en sourdine**

# TP1 - Construire son CV (Rappels)

**Les styles et l'approche responsive**

# TP1 - Construire son CV (Rappels - 1/2)

-  Il n'est pas demandé de remplir le CV avec vos propres informations mais ça pourrait être intéressant pour vous de le faire pour ensuite le mettre en ligne et le présenter aux entreprises que vous démarcherez dans le futur
- Si vous souhaitez avoir une idée d'une proposition de résultat final, vous pouvez consulter ma version en ligne de l'exercice => <https://chris-chevalier.github.io/tp-cv/>
- Dans un nouveau répertoire **cv**, créez un fichier **cv.html**
- Une **photo** doit apparaître sur cette unique page HTML
- La **description** de votre profil, votre **âge** et votre **numéro de téléphone** doivent être mis en forme à l'aide d'un **tableau** stylisé comportant vos **nom** et **prénom sous le même titre Noms**

# TP1 - Construire son CV (Rappels - 2/2)

- Vos **compétences** doivent apparaître dans une **liste numérotée**
- Les **noms des villes** de vos **formations** doivent être en gras et d'une **couleur différente** du reste du contenu textuel de la page
- Les **tâches** réalisées en entreprise doivent apparaître dans une **liste non ordonnée**
- Des **liens** doivent permettre d'**atteindre les différentes parties** de votre CV
- ! Vos **styles CSS** doivent **TOUS** être définis dans un **fichier style.css** situé dans le répertoire **style** de votre projet (Cf. [diapos 7 et 8](#))

# TP1 - Construire son CV (contraintes)

- En plus de votre **photo** et de vos **informations**, l'**en-tête** de votre **page web** doit comporter des **liens ouvrant un nouvel onglet** (LinkedIn, site perso, etc.) représenté par une **image** ou une **icône**
- L'**en-tête** doit également comporter une **image de fond**
- Chaque **sous-partie** de votre **CV** doit présenter une **couleur de fond** différente
- **Stylisez** les **titres** des différentes **sections** ainsi que les **liens**
- N'hésitez pas à utiliser aussi souvent que nécessaire des **id** et/ou **classe** pour identifier vos **éléments HTML** depuis votre **fichier CSS** (Cf. diapos 13 à 16)

# TP2 - Agencez votre page HTML

**Les styles et l'approche responsive**

# TP2 - Agencer une page HTML (header)

- Plusieurs sections de ce TP ne sont volontairement pas explicitées dans le cours théorique - C'est l'occasion pour vous de réaliser vos propres recherches - Je vous invite à chercher vos réponses en priorité sur le très bon site [W3Schools](#) ✓
- Créez une **page HTML** avec un **titre** centré horizontalement dans la section **header** qui comporte un fond de couleur (Cf. [diapo 5](#))
- La **section header** doit prendre **toute la largeur** de l'écran, **sans marge** et proposer une **hauteur fixe**
- Les **couleurs de fond** de la **section header** doivent aller d'une couleur **foncée à gauche** vers une plus **claire à droite (dégradé** - à vous d'aller trouver « seul » ce point 😊 )
- Le **titre** doit être écrit dans une **couleur** différente de celles du fond - ⚡ Pensez à l'accessibilité en rendant le tout lisible !
- 💡 Pensez à ajouter la **balise meta viewport** à l'en-tête (`<head></head>`) de votre **document HTML** pour assurer un affichage cohérent peu importe l'appareil utilisé (Cf. [diapo 18](#))

# TP2 - Agencer une page HTML (**article**)

- Utilisez la **balise sémantique article** (Cf. diapo 21 et 22) pour ajouter une **iframe** permettant de visualiser votre CV (Cf. diapo 20)
- La **iframe** doit proposer une **largeur** et une **hauteur fixe** ainsi qu'un cadre aux **bordures épaisses, arrondies** et de **couleur**
- Ajoutez à votre **section article** un **lien** renvoyant vers votre CV
- Le **lien** vers votre CV doit s'ouvrir dans un **nouvel onglet** et apparaître dans un **bouton stylisé**
- En **passant la souris sur le bouton**, le **curseur** doit être un **pointeur**, la **couleur du fond** et de la **police de caractère** doivent être différentes (**mon-element:hover**)

# TP2 - Agencer une page HTML (**footer**)

- Dans la **section footer**, créez un **élément picture** affichant **3 images différentes** en **fonction de la taille de l'écran** (Cf. [diapo 19](#))
- La **section footer** doit être **fixée** en **bas de l'écran** - Découvrez comment mettre en place ce traitement en suivant ce [lien W3Schools](#)

# TP3 - Améliorer son CV

**Les styles et l'approche responsive**

# TP3 - Améliorer son CV (1/3)

- Reprenez **votre fichier** `cv/cv.html` et ajoutez-y les éléments suivants :
  - Une **image** dans un **élément picture** s'adaptant en **fonction des dimensions de l'écran** (Cf. [diapo 19](#))
  - Un **fichier audio** (Cf. [diapo 31](#))
  - Un **lien Spotify** intégré (À vous de chercher !)
  - Une **vidéo Youtube** intégrée (Cf. [diapos 32](#))
  - Une **image responsive** centrée prenant **50%** de largeur en **desktop**, **80%** sur une **tablette** et **90%** sur un **mobile** (Cf. [diapo 27](#))
  - Intégrez un **lien pointant vers une adresse email** dans la **section footer** de la page

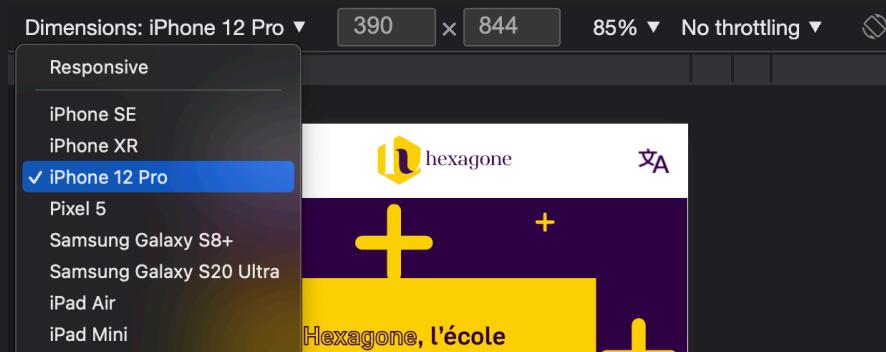
# TP3 - Améliorer son CV (2/3)

- L'intégralité de votre CV doit être **lisible** sur **ordinateur, tablette et smartphone**
- Sur un navigateur web comme **Google Chrome** ou **FireFox**, vous pouvez faire **clic droit > Inspecter** et sélectionner cette icône pour afficher votre site web des dimensions différentes de celles de votre écran (ne fonctionne pas sur **Safari**) :



```
<!DOCTYPE html>
<html lang="fr">
 <head>...</head>
```

- Il ne vous reste plus qu'à **sélectionner l'appareil** que vous souhaitez simuler :



# TP3 - Améliorer son CV (3/3)

- Certains contenus doivent **disparaître** sur un écran de **mobile** (`display: none`)
- Le **lien** présent dans le **footer** doit apparaître dans un **bouton** dont le **style change** au **passage de la souris** (**couleur du fond + curseur + ombres portées**)
- La **section footer** doit présenter une **couleur de fond différente** du reste et doit prendre **100% de la largeur** de la page

# ★ Bonus

Les styles et l'approche responsive

# ★ Bonus - Flexbox et Grid

- Deux philosophies modernes du langage **CSS** permettent de faciliter grandement le **placement** des **éléments HTML** - Il s'agit de Flexbox et Grid
- Pour apprendre à les utiliser, je vous invite à consulter (et **ajouter aux favoris** de votre navigateur) les guides suivants :
  - Flexbox
  - Grid
- Je vous invite également à réaliser les séries d'exercices proposés par Flexbox Froggy et Grid Garden

# Pour aller plus loin...

- Quelques exemples **W3Schools** de **media queries** : [https://www.w3schools.com/css/css3\\_mediaqueries\\_ex.asp](https://www.w3schools.com/css/css3_mediaqueries_ex.asp)
- Un **guide** complet sur l'utilisation des **Flexbox CSS** : <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- Un **guide** complet sur l'utilisation des **Grid CSS** : <https://css-tricks.com/snippets/css/a-guide-to-grid/>
-

# Des questions ?