COURS COMPLET HTML & CSS – D'APRÈS LE PLAN VALENTIN DUPAS

==========

0. INTRO

Le Web est né comme un système très simple pour partager des idées sous forme de texte.

Au fil du temps, on est passé :

- d'un Web très simple (pages HTML statiques),
- à un Web stylisé (CSS),
- puis à un Web structuré et réactif (Flexbox, Grid, responsive),
- jusqu'au CSS moderne (variables, nesting, media queries avancés…).

Ce cours suit la progression :

1. Le web des années 90
2. Le web des années 2000
3. Le web des années 2010

4.x La fin du début (CSS moderne)

5.x Le CSS post-moderne

==========

1. LE WEB DES ANNÉES 90 – HTML, LIENS, IMAGES, CHEMINS

==========

1.1 HTML – Le langage du Web

HTML (HyperText Markup Language) est le langage qui décrit la structure d'une page Web.

Structure minimale d'un document HTML :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Titre de la page</title>
  </head>
  <body>
    Contenu visible ici.
  </body>
```

</html>

- <!DOCTYPE html> : indique au navigateur qu’on utilise HTML5.
- <html> : enveloppe tout le document.
- <head> : contient les métadonnées (titre, encodage, liens CSS…).
- <body> : contient tout ce qui est visible dans la fenêtre du navigateur.

Éléments de base :
- Titres : <h1> à <h6>
- Paragraphe : <p>
- Listes ordonnées : <ol> +
- Listes non ordonnées : <ul> +

1.2 Les hyperliens

Les liens sont créés avec la balise <a> (anchor).

Exemples :

```
<a href="https://example.com">Visiter un site externe</a>
<a href="page.html">Aller vers une autre page du même site</a>
<a href="page.html" target="_blank">Ouvrir dans un nouvel onglet</a>
```

- href : destination du lien.
- target="_blank" : ouverture dans un nouvel onglet.

1.3 Les images

Une image est insérée avec <img> :

```
<img src="images/chat.png" alt="Un chat noir sur un coussin">
```

- src : chemin vers le fichier image.
- alt : description textuelle (important pour l’accessibilité).

On peut rendre une image cliquable en la mettant dans un lien :

```
<a href="profil.html">
  <img src="images/avatar.png" alt="Photo de profil">
</a>
```

1.4 Les chemins : absolus et relatifs

Les chemins sont utilisés dans href et src pour dire où se trouve un fichier.

Schéma de projet :

site/

■■ index.html

■■ about.html

■■ images/

    ■■ logo.png

Depuis index.html :

- Chemin relatif vers le logo :
  images/logo.png

- Chemin absolu à partir de la racine du site :
  /images/logo.png

- Chemin absolu sur Internet :
  https://monsite.com/images/logo.png

1.4.1 Chemin absolu

Un chemin absolu ne dépend pas du fichier courant :

- /images/logo.png (absolu dans le site)
- https://monsite.com/images/logo.png (URL complète)

1.4.2 Chemins relatifs

Un chemin relatif dépend du fichier où on écrit.

Exemples (toujours depuis index.html) :

- images/logo.png → va dans le dossier images à côté d'index.html
- ../shared/menu.html → remonte d'un dossier, puis entre dans shared

1.4.3 Relatif à la racine

Un chemin qui commence par / est interprété à partir de la racine du site :

/images/logo.png

1.4.4 Relatif au fichier courant

Un chemin sans / au début est interprété depuis le dossier du fichier courant :

images/chat.png

docs/chapitre1.html

1.4.5 Pourquoi les débutants sont confus ?

Car parfois :

- /images/logo.png
- images/logo.png

semblent fonctionner pareil selon le serveur et la configuration, alors qu'ils n'ont pas la même logique.

1.4.6 Évitez les liens absolus (quand vous pouvez)

Dans un projet, il est souvent préférable d'utiliser des chemins relatifs :

- plus faciles à déplacer d'une machine à l'autre,
- évitent de coder en dur le domaine (ex : https://monsite.com/...).

1.5 index.html

---

index.html est le nom « spécial » d'une page par défaut.

- / → charge /index.html
- /blog/ → charge /blog/index.html

==========

2. LE WEB DES ANNÉES 2000 – CSS, BOX MODEL, UNITÉS

==========

2.1 Un peu de CSS

CSS (Cascading Style Sheets) sert à mettre en forme le HTML.

2.2 Pourquoi le tag <html> ?

La balise <html> sert de racine au document, et permet de définir des choses globales, comme la langue :

<html lang="fr">

et d’hériter de styles (par exemple font-size de base).

2.3 CSS local, inline, externe

- CSS inline : dans l’attribut style (à éviter) :

```
<p style="color:red;">Texte rouge</p>
```

- CSS local : dans une balise <style> :

```
<style>
  p { color: red; }
</style>
```

- CSS externe : fichier séparé (méthode recommandée) :

```
<link rel="stylesheet" href="style.css">
```

2.4 Unités et formats, explication en couleurs

Formats de couleur :

- mots-clés : red, blue…
- hexadécimal : #ff0000
- RGB : rgb(255, 0, 0)
- HSL : hsl(0, 100%, 50%)

HSL (Hue, Saturation, Lightness) est souvent plus intuitif pour créer des variantes de couleur.

2.5 Le C de CSS veut dire Cascade

La cascade = l’ordre dans lequel les styles sont appliqués :

- Spécificité (id > classe > balise)
- Ordre d’apparition (le plus bas écrase le plus haut)
- Styles par défaut du navigateur

2.6 Le flux : Bloc et Inline

- block : prend toute la largeur, revient à la ligne.
  Ex : , <p>, <h1>.

- inline : prend seulement la place du contenu, ne revient pas à la ligne.
  Ex : <a>, .

2.7 Polices : la forme et la font

Changer la police :

```
body {
  font-family: Arial, sans-serif;
}
```

Importer une police :

```
@font-face {
  font-family: "MaPolice";
  src: url("fonts/mapolice.woff2") format("woff2");
}
```

2.8 Le box model et les dimensions

Chaque élément est une « boîte » :

- content : le contenu (texte, image…)
- padding : espace interne
- border : bordure
- margin : espace externe

width/height s’appliquent au content (par défaut).

2.9 Unités em, rem, ch, %

- em : relatif à la font-size du parent.
- rem : relatif à la font-size du html (root).
- ch : largeur approximative du caractère « 0 ».
- % : relatif à la taille du parent (ex : largeur de son conteneur).

2.10 Classes et IDs

- Classe (.nom) : peut être utilisée plusieurs fois.

```
.important { color:red; }
```

- ID (#nom) : doit être unique dans la page.

```
#header { background:black; }
```

==========

3. LE WEB DES ANNÉES 2010 – FLEXBOX, GRID, MEDIA QUERIES

==========

3.1 Flexbox – Mise en page flexible

Activer Flexbox sur un conteneur :

```
.container {
  display: flex;
}
```

Propriétés pour le parent :

- flex-direction: row | column
- justify-content: flex-start | center | space-between | space-around
- align-items: flex-start | center | flex-end
- flex-wrap: nowrap | wrap

Propriétés pour les enfants :

- flex-grow : comment l'élément grandit
- flex-shrink : comment il rétrécit
- flex-basis : taille de base
- order : ordre d'affichage

3.2 Éléments wrappers : <article>, <section>, , etc.

- <article> : contenu autonome (article de blog, carte…)
- <section> : section logique de la page
- : conteneur générique (sans sens particulier)
- <header>, <footer>, <nav>, <main>, <aside> : éléments sémantiques.

3.3 vw et vh

- 1vw = 1% de la largeur de la fenêtre.

- 1vh = 1% de la hauteur de la fenêtre.

Pratique pour des sections pleine largeur/hauteur.

3.4 Grid – Mise en page en grille

Activer grid sur un parent :

```
.grid {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 10px;
}
```

Tailles implicites :

- Les lignes/colonnes peuvent être créées automatiquement selon le contenu.

Alignements :

- justify-items, align-items, justify-content, align-content.

Areas (zones nommées) :

```
.grid {
  grid-template-areas:
    "header header"
    "menu   contenu";
}
```

Propriétés enfants :

- grid-column, grid-row, grid-area.

fit-content :

- Permet de limiter la taille d’une colonne/ligne :

```
grid-template-columns: fit-content(300px) 1fr;
```

3.5 Media queries

Permet de changer le layout selon la taille d’écran.

```
@media (max-width: 600px) {
  body { font-size: 14px; }
```

}

- screen : type de media principal (écran).
- min-width / max-width : largeur minimale/maximale ciblée.

==========

4.0 LA FIN DU DÉBUT – CSS MODERNE

==========

4.1 Pseudo-classes

- :hover → quand la souris survole
- :visited → lien déjà visité
- :nth-child(n) → n-ième enfant

Exemples :

```
a:hover { text-decoration: underline; }
li:nth-child(2) { color: red; }
```

4.2 Sélecteurs combinés

- , : sélection multiple

```
h1, h2 { margin-bottom: 1rem; }
```

- espace : descendant

```
.card p { color: gray; }
```

- > : enfant direct

```
.menu > li { list-style: none; }
```

4.3 Spécificité

Chaque type de sélecteur a un poids :

- ID (#id) : fort
- Classe, pseudo-classe (.classe, :hover) : moyen
- Balise (p, div…) : faible

Les styles par défaut du navigateur (ex : liens bleus <a>) sont simplement des règles appliquées avant les nôtres. Dès qu'on ajoute un style plus spécifique, on les remplace.

!important :

- Ajouté à une propriété, il passe devant tout le reste.
- À utiliser très rarement.

4.4 Variables CSS

Déclaration :

```
:root {
  --couleur-principale: #3498db;
}
```

Utilisation :

```
button {
  background: var(--couleur-principale);
}
```

4.5 Transitions

Permettent d’animer en douceur les changements de propriétés.

Propriétés :

- transition-property
- transition-duration
- transition-delay
- transition-timing-function

Shorthand :

```
transition: all 0.3s ease-in-out;
```

4.6 Animations

Définir une animation :

```
@keyframes rebond {
  0%   { transform: translateY(0); }
  100% { transform: translateY(-20px); }
}
```

Appliquer :

```
.balle {
  animation-name: rebond;
  animation-duration: 0.5s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
  animation-timing-function: ease-out;
  animation-fill-mode: both;
}
```

4.7 Gradients

Exemples :

```
background: linear-gradient(to right, red, blue);
background: radial-gradient(circle, red, blue);
background: conic-gradient(red, yellow, blue);
```

On peut superposer plusieurs backgrounds :

```
background:
  linear-gradient(to bottom, rgba(0,0,0,.5), rgba(0,0,0,0)),
  url('image.jpg');
```

4.8 Small / Large / Dynamic (viewport)

Nouvelles unités pour gérer mieux les barres d’outils mobiles :

- lvh, svh, dvh…

4.9 Position : absolute, relative, fixed, sticky

- position: relative → crée une référence pour les éléments en absolute.
- position: absolute → se place par rapport au premier parent positionné.
- position: fixed → collé à la fenêtre (header fixe).
- position: sticky → devient « collant » après un certain scroll.

Sticky avec grid ou flex : permet de faire des sidebars qui restent visibles pendant le scroll.

==========

5.0 LE CSS POST-MODERNE

==========

5.1 Nesting (&, implicite, pseudos)

Le nesting permet d’imbriquer des règles CSS pour plus de lisibilité.

Exemple :

```
.card {
  padding: 1rem;

  & h2 {
    margin-bottom: .5rem;
  }

  &:hover {
    box-shadow: 0 0 10px rgba(0,0,0,.2);
  }

  & a:hover {
    text-decoration: underline;
  }
}
```

5.2 Media queries 2 : le retour

Syntaxes plus modernes possibles :

```
@media (width < 600px) { ... }
```

ou des combinaisons :

```
@media (600px <= width <= 900px) { ... }
```

Ces nouvelles formes rendent les conditions plus lisibles.

==========

FIN DU COURS

==========

Ce document reprend l’ensemble des points listés dans ton plan de cours, avec une explication structurée et des exemples concrets pour chaque notion.