Activité d'apprentissage S1 : Conventions de dénomination

Aperçu

Lorsque vous travaillez sur le web, divers facteurs tels que le navigateur, les protocoles utilisés, le système d'exploitation, les langues, etc., peuvent influencer le fonctionnement des fichiers. Bien que nombre de ces éléments soient hors de votre contrôle, vous pouvez garantir la cohérence et la facilité d'organisation de votre travail en adoptant des conventions de nommage standardisées pour les fichiers et les dossiers.

Préparer

- Les conventions de nommage des fichiers et des dossiers présentées dans cette leçon constituent les bonnes pratiques recommandées pour ce cours. Ces règles doivent être appliquées à tous les fichiers et dossiers créés dans le cadre de vos travaux.
  Il convient de noter que la plupart des organisations possèdent leurs propres règles et bonnes pratiques de gestion des fichiers, notamment des conventions pour nommer les fichiers, les dossiers et autres flux de travail. Ces conventions peuvent varier d'une organisation à l'autre.

Conventions de dénomination

Vous devrez respecter ces conventions de dénomination tout au long du cours.

- Utilisez la syntaxe **en minuscules** , comme par exemple :**`products.html`**
  Les plateformes et les systèmes gèrent différemment la sensibilité à la casse. Il est important de comprendre ce concept pour bien gérer les fichiers et les dossiers.
- N’utilisez **PAS d’espaces** dans les noms de fichiers et de dossiers. Utilisez plutôt des tirets, comme ceci :**`design-document.html`**
  Les navigateurs gèrent les espaces de manière incohérente ; il est donc déconseillé de les utiliser. Le protocole HTTP ignore les espaces, sauf dans les noms de fichiers. Dans ce cas, il remplace un espace

par la syntaxe suivante : «**`%20`** », ce qui peut prêter à confusion. Évitez d'utiliser des espaces et, si vous souhaitez créer un espace visuel, utilisez plutôt des tirets.

- N'utilisez **PAS de caractères spéciaux** , par exemple ,**`<,>, \, /, #, ?, !`**
  Les caractères spéciaux ont souvent une signification spécifique pour les ordinateurs, il ne faut donc pas les utiliser dans le nommage des fichiers et des dossiers.
- Les noms de fichiers et de dossiers doivent être aussi courts et significatifs (**sémantiques**) que possible, par exemple **`winter-scene-sm.png`**
  :**`image13-v123523brokenbranchlifeimagery w200x200.png`**
  Des noms courts vous évitent, ainsi qu'aux autres développeurs et aux visiteurs du site, d'avoir à mémoriser des noms longs et complexes pour les fichiers et les dossiers. De plus, des noms pertinents peuvent aider à prédire la fonction ou la nature du contenu d'un fichier ou d'un dossier.
- Dans ce cours, les **noms de dossiers standard** pour nos sites/sous-dossiers sont :
  - **styles** – Les dossiers portant ce nom contiennent des fichiers CSS.
  - **images** – Les dossiers portant ce nom contiennent des images.
  - **scripts** – Les dossiers portant ce nom contiennent des fichiers JavaScript.

Page 1 – Les bases du CSS avec exemples concrets

Le CSS (Cascading Style Sheets) est le langage qui permet de styliser une page HTML. Le HTML donne la structure (titres, paragraphes, sections), tandis que le CSS définit l'apparence visuelle. Sans CSS, une page web fonctionne, mais elle est très basique et peu agréable à lire. Avec le CSS, on peut contrôler les couleurs, les tailles, les positions et l'organisation des éléments.

Exemple : Reset CSS

Voici un exemple très simple de reset CSS :

* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

Explication :

- `*` signifie « tous les éléments HTML »
- `margin: 0` enlève les marges automatiques
- `padding: 0` enlève les espacements internes
- `box-sizing: border-box` simplifie le calcul des tailles

Sans ce reset, chaque navigateur peut afficher la page différemment.

Exemple : Mise en page générale avec Grid

body {

font-family: Arial, sans-serif;

min-height: 100vh;

display: grid;

grid-template-rows: auto 1fr auto;

background-color: #f4f4f4;

}

Explication :

- `font-family` définit la police
- `min-height: 100vh` force la page à occuper tout l'écran
- `display: grid` active CSS Grid
- `grid-template-rows` crée 3 zones : header, main, footer

Cet exemple montre comment structurer toute la page avec très peu de code.

Page 2 – Header et navigation avec Flexbox (exemples)

Le header contient souvent le titre du site et le menu de navigation. Pour bien les aligner, Flexbox est l'outil idéal.

Exemple : Header avec titre à gauche et menu à droite

header {

display: flex;

justify-content: space-between;

align-items: center;

background-color: #1f2933;

color: white;

padding: 1rem;

}

Explication :

- `display: flex` met les éléments sur une ligne
- `justify-content: space-between` pousse les éléments aux extrémités
- `align-items: center` centre verticalement

Exemple : Menu horizontal

nav ul {

list-style: none;

display: flex;

gap: 1.5rem;

}

Explication :

- `list-style: none` enlève les puces
- `display: flex` aligne les liens horizontalement
- `gap` crée un espace entre les liens

Exemple : Liens du menu

nav a {

color: white;

text-decoration: none;

font-weight: bold;

}

nav a:hover {

text-decoration: underline;

}

Cet exemple montre comment améliorer l'expérience utilisateur avec un effet au survol.

Page 3 – Main, cartes et différenciation des sections (exemples)

La zone `main` contient le contenu principal. Dans cet exemple, chaque section et l'aside sont affichés sous forme de cartes.

Exemple : Grille de cartes

main {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 1rem;

padding: 1rem;

}

Explication :

- `repeat(3, 1fr)` crée trois colonnes de même taille
- `gap` définit l'espace entre les cartes

Exemple : Style commun des cartes

section,

aside {

padding: 1rem;

border-radius: 8px;

}

Cela évite de répéter le même code plusieurs fois.

Exemple : Différencier les sections

section:nth-of-type(1) {

background-color: white;

}

section:nth-of-type(2) {

background-color: #fecaca;

}

aside {

background-color: #e5e7eb;

}

Grâce à ces exemples, les étudiants voient clairement comment le CSS permet de distinguer visuellement chaque partie de la page.

Enfin, le footer est stylisé simplement pour rester cohérent avec le header :

footer {

background-color: #1f2933;

color: white;

text-align: center;

padding: 1rem;

}

Ces exemples complets montrent comment CSS Grid et Flexbox travaillent ensemble pour créer une mise en page moderne et facile à comprendre pour les débutants.