Principaux langages de programmation
Maîtriser les Langages Web : HTML, CSS, JS

---

1. HTML et Intégration des contenus

1.1 Les notions génériques

1.1.1 Les bases techniques

- Local vs distant :
    - Local : fichiers sur votre ordinateur
    - Distant : fichiers sur un serveur web
- Les protocoles :
    - HTTP (Hypertext Transfer Protocol) : non sécurisé
    - HTTPS (HTTP Secure) : version sécurisée d'HTTP
    - FTP (File Transfer Protocol) : pour le transfert de fichiers
- Les extensions :
    - .html : fichiers HTML
    - .css : feuilles de style CSS
    - .js : scripts JavaScript
- Les navigateurs et l'inspecteur de code :
    - Navigateurs populaires : Chrome, Firefox, Safari, Edge
    - Utilisation de l'inspecteur : F12 ou clic droit > Inspecter

1.1.2 Les langages web : Le web statique et web dynamique

- Web statique :
    - HTML : structure du contenu
    - CSS : mise en forme
    - JavaScript : interactivité côté client
- Web dynamique :
    - PHP : génération de contenu côté serveur
    - MySQL : gestion de bases de données
- Flux de données :
    a. Back-office : interface d'administration
    b. Bases de données : stockage des données
    c. Front-office : interface utilisateur
- Notion de gabarit :
    - Modèle de page réutilisable
    - Facilite la création de pages dynamiques avec PHP et MySQL

1.2 Les bases du HTML

1.2.1 Les balises HTML

- Syntaxe des balises :

```
<balise>Contenu</balise>
<balise attribut="valeur">Contenu</balise>
```

- Balises fondamentales :

```html
<!DOCTYPE html>
<html>
  <head>
    <!-- Métadonnées -->
  </head>
  <body>
    <!-- Contenu visible -->
  </body>
</html>
```

- Arborescence du HTML (DOM) :

```
html
├── head
│   ├── title
│   └── meta
└── body
    ├── header
    ├── main
    └── footer
```

- Attributs courants :
    - src : source pour les images, vidéos, scripts
    - href : liens hypertextes
    - alt : texte alternatif pour les images
- Balises orphelines :

```html
<img src="image.jpg" alt="Description">
<br>
<input type="text">
```

- Balises du head :

```
<head>
  <title>Titre de la page</title>
  <meta charset="UTF-8">
  <meta name="description" content="Description de la page">
  <link rel="stylesheet" href="styles.css">
  <script src="script.js"></script>
</head>
```

1.2.2 La mise en forme des contenus

- Balises de titre :

```
<h1>Titre principal</h1>
<h2>Sous-titre</h2>
<h3>Sous-sous-titre</h3>
<!-- ... jusqu'à h6 -->
```

- Mise en forme du texte :

```
<p>Paragraphe de texte</p>
<strong>Texte important</strong>
<em>Texte mis en emphase</em>
<a href="https://example.com">Lien hypertexte</a>
<blockquote>Citation</blockquote>
```

- Listes :

```
<ul>
  <li>Élément de liste non ordonnée</li>
</ul>
<ol>
  <li>Élément de liste ordonnée</li>
</ol>
```

- Tableaux :

```html
<table>
  <tr>
    <th>En-tête 1</th>
    <th>En-tête 2</th>
  </tr>
  <tr>
    <td>Cellule 1</td>
    <td>Cellule 2</td>
  </tr>
</table>
```

- Formulaires :

```html
<form action="/submit" method="post">
  <label for="name">Nom :</label>
  <input type="text" id="name" name="name" required>
  <input type="submit" value="Envoyer">
</form>
```

- Médias :

```html
<img src="image.jpg" alt="Description de l'image">
<video src="video.mp4" controls></video>
<audio src="audio.mp3" controls></audio>
```

- Éléments block vs inline :
  - Block : occupent toute la largeur disponible (ex: `<div>`, `<p>`, `<h1>`)
  - Inline : occupent seulement l'espace nécessaire (ex: `<span>`, `<a>`, `<strong>`)

1.2.3 Les balises neutres et sémantiques

- Balises neutres :

```html
<div>Conteneur générique de type block</div>
<span>Conteneur générique de type inline</span>
```

- Balises sémantiques :

```
<header>En-tête de page ou de section</header>
<nav>Navigation principale</nav>
<main>Contenu principal</main>
<article>Article indépendant</article>
<section>Section de contenu</section>
<aside>Contenu annexe</aside>
<footer>Pied de page</footer>
```

1.2.4 Les attributs ARIA

- Rôles et utilisation :

```
<button role="button" aria-label="Fermer">X</button>
<div role="alert" aria-live="assertive">Message
important</div>
<input type="text" aria-describedby="password-requirements">
<div id="password-requirements" aria-hidden="true">
  Le mot de passe doit contenir au moins 8 caractères
</div>
```

1.2.5 Importance du respect de la syntaxe

- Validité du Code : utiliser des validateurs HTML (ex: W3C Markup Validation Service)
- Accessibilité : faciliter l'accès aux utilisateurs de technologies d'assistance
- SEO : améliorer le référencement naturel
- Maintenabilité : code plus facile à lire et à maintenir
- Indentation et commentaires :

```
<!-- En-tête de la page -->
<header>
  <!-- Navigation principale -->
  <nav>
    <ul>
      <li><a href="#home">Accueil</a></li>
      <li><a href="#about">À propos</a></li>
    </ul>
  </nav>
</header>
```

2. CSS et Stylisation des pages

2.1 Les bases du CSS

2.1.1 Savoir cibler un élément

- Méthodes d'inclusion du CSS :
    a. Inline :

    ```
    <p style="color: blue;">Texte bleu</p>
    ```

    b. Internal :

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

    c. External :

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

- Écriture du CSS :

```
sélecteur {
  propriété: valeur;
}
```

- Sélecteurs CSS :

```
/* Sélecteur de type */
p { color: blue; }

/* Sélecteur de classe */
.highlight { background-color: yellow; }

/* Sélecteur d'identifiant */
#header { font-size: 24px; }

/* Sélecteur descendant */
nav ul li { display: inline-block; }

/* Sélecteur d'enfant direct */
main > p { margin-bottom: 10px; }
```

- Utilisation de l'inspecteur de code :
    - Ouvrir l'inspecteur (F12)
    - Sélectionner un élément
    - Modifier les styles en temps réel

2.1.2 Comprendre les bases du CSS

- Notion de flux :
    - Éléments affichés dans l'ordre du code HTML
    - Peut être modifié avec des propriétés comme `position` ou `order`
- Positionnement :

```
.static { position: static; } /* Par défaut */
.relative { position: relative; top: 10px; }
.absolute { position: absolute; top: 0; left: 0; }
.fixed { position: fixed; bottom: 20px; right: 20px; }
```

- La cascade :
    - Ordre d'application des styles (importance, spécificité, ordre dans le code)

2.1.3 Les couleurs, tailles et polices

- Gestion des styles typographiques :

```
body {
  font-family: Arial, sans-serif;
  font-size: 16px;
  line-height: 1.5;
}
```

- Couleurs :

```
.text-color { color: #ff0000; } /* Rouge */
.bg-color { background-color: rgba(0, 255, 0, 0.5); } /* Vert semi-transparent */
```

- Gestion des visuels et des backgrounds :

```
.bg-image {
  background-image: url('image.jpg');
  background-size: cover;
  background-position: center;
}
```

2.1.4 Les techniques plus avancées

- Pseudo-classes :

```
a:hover { color: red; }
input:focus { border-color: blue; }
```

- Pseudo-éléments :

```
div::before { content: "«"; margin-left: 5px; }
div::after { content: "»"; margin-right: 5px; }
```

- Sélecteurs d'attribut :

```
input[type="text"] { border: 1px solid #ccc; }
```

- Sélecteur universel :

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

- Le hack "!important" :

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

2.2 Les principes de mise en page

2.2.1 Le Box Model

- Compréhension du modèle de boîte :

```css
.box {
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 2px solid black;
  margin: 10px;
}
```

- La propriété `box-sizing` :

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

2.2.2 Savoir gérer son layout

Flexbox (Flexible Box Layout) est un modèle de mise en page introduit dans CSS3, conçu pour offrir une meilleure manière de distribuer l'espace entre des éléments dans un conteneur, même si leur taille est inconnue ou dynamique.

Avantages :

- Alignement simplifié des éléments (horizontalement et verticalement).
- Gestion efficace des espaces entre les éléments (marges, paddings).
- Adaptabilité aux différents écrans sans recours à des calculs complexes (responsive).

Flexbox repose sur deux éléments principaux :

- **Le conteneur flex** (parent) : l'élément qui contient les éléments à aligner.
- **Les items flex** (enfants) : les éléments à l'intérieur du conteneur flex, qui seront disposés en fonction des propriétés Flexbox.

Exemple HTML de base :

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

Exemple CSS pour créer un conteneur Flexbox :

```
.container {
  display: flex;
}
.item {
  background-color: #f0f0f0;
  padding: 10px;
  margin: 5px;
  border: 1px solid #ccc;
}
```

Les principales propriétés à utiliser sur le conteneur Flexbox sont :

- **`display: flex;`** : Active le mode Flexbox sur le conteneur.
- **`flex-direction`** : Définit la direction des éléments à l'intérieur du conteneur.
    - `row` (par défaut) : Alignement des éléments en ligne.
    - `column` : Alignement des éléments en colonne.

```
.container {
  flex-direction: row; /* ou column */
}
```

- **`justify-content`** : Contrôle l'alignement horizontal des items.
    - `flex-start` : Aligne les items au début du conteneur.
    - `center` : Centre les items.
    - `space-between` : Espacement égal entre les items.

```css
.container {
  justify-content: center;
}
```

- **`align-items`** : Contrôle l'alignement vertical des items.
  - `flex-start` : Aligne les items en haut du conteneur.
  - `center` : Aligne les items au centre.
  - `stretch` : Étire les items pour remplir le conteneur.

```css
.container {
  align-items: center;
}
```

Les éléments enfants peuvent être manipulés avec les propriétés suivantes :

- **`flex-grow`** : Permet à un item de grandir pour occuper l'espace disponible.

```css
.item {
  flex-grow: 1; /* L'item grandit pour remplir l'espace */
}
```

- **`flex-shrink`** : Permet à un item de rétrécir lorsque l'espace est insuffisant.

```css
.item {
  flex-shrink: 1; /* L'item rétrécit si nécessaire */
}
```

- **`flex-basis`** : Définit la taille de base d'un item avant que Flexbox ne distribue l'espace restant.

```css
.item {
  flex-basis: 200px; /* Largeur de base de 200px */
}
```

Voici un exemple pratique pour créer une mise en page avec trois colonnes qui s'ajustent en fonction de la taille de l'écran :

```
<div class="container">
  <div class="item">Colonne 1</div>
  <div class="item">Colonne 2</div>
  <div class="item">Colonne 3</div>
</div>
```

```
.container {
  display: flex;
  justify-content: space-between; /* Distribution égale des colonnes
*/
}
.item {
  flex-basis: 30%; /* Chaque colonne occupe 30% de la largeur */
  padding: 20px;
  background-color: #eaeaea;
  border: 1px solid #ccc;
}
```

2.2.3 Le responsive Design

- Flexbox facilite la création de layouts responsives sans avoir à utiliser des media queries complexes. Par exemple, pour ajuster automatiquement la disposition des éléments sur les petits écrans, on peut combiner Flexbox avec les propriétés de `flex-wrap` :

  ```
  .container {
    display: flex;
    flex-wrap: wrap; /* Les items vont se répartir sur plusieurs
  lignes si nécessaire */
  }
  .item {
    flex: 1 1 100%; /* Chaque item occupe 100% de la largeur sur
  petits écrans */
  }
  ```

- Utilisation des Media Queries :

```
@media screen and (max-width: 768px) {
  .container {
    flex-direction: column;
  }
}
```

3. Interactivité avec JavaScript

3.1 Les bases pour démarrer

3.1.1 Manipulation du DOM avec jQuery

- Sélection d'éléments :

```
$('.class-name') // Par classe
$('#id-name')    // Par ID
$('div')         // Par type de balise
```

- Modification d'éléments :

```
$('#element').text('Nouveau texte');
$('.element').html('<strong>Contenu HTML</strong>');
```

- Gestion des événements :

```
$('button').click(function() {
  alert('Bouton cliqué !');
});
```

3.1.2 Ajout d'un fichier JS externe

```
<!DOCTYPE html>
<html>
<head>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <!-- Contenu de la page -->
  <script src="mon-script.js"></script>
</body>
</html>
```

3.1.3 Ajouter un plugin JS et le paramétrer

Exemple avec le plugin Slick Carousel :

```
<head>
  <link rel="stylesheet" type="text/css" href="slick/slick.css"/>
</head>
<body>
  <div class="carousel">
    <div><img src="image1.jpg"></div>
    <div><img src="image2.jpg"></div>
    <div><img src="image3.jpg"></div>
  </div>

  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script type="text/javascript" src="slick/slick.min.js"></script>
  <script>
    $(document).ready(function(){
      $('.carousel').slick({
        dots: true,
        infinite: true,
        speed: 300,
        slidesToShow: 1
      });
```

```
    });
  </script>
</body>
```

3.1.4 Utilisation de l'IA pour coder

Exemple d'utilisation de l'IA pour générer une fonction JavaScript en utilisant jQuery.

Créer des accordéons interdépendants avec HTML, CSS et JS (jQuery). En utilisant, l'IA générative de votre choix (chatGPT, Claude...)

Notez qu'il est important de comprendre et de tester le code généré par l'IA avant de l'utiliser dans vos projets.