Cours HTML & CSS

Table des matières

1 Définition et utilisation du HTML et du CSS....4
1.1 HTML : le langage de structure....4
1.2 CSS : le langage de styles....4
1.3 Histoire et évolution de l'informatique et du web....5
1.4 Les environnements : local, préproduction et production....5
1.5 Choisir et installer un éditeur de texte....6
1.6 Préparer la structure d'un site local....6
2 Présentation du HTML....7
2.1 Éléments, balises et attributs HTML....7
2.2 Structure minimale d'une page HTML....8
2.3 Indentation et lisibilité du code....8
2.4 Commentaires en HTML....8
2.5 Enregistrer et afficher une page HTML....9
2.6 Créer des liens hypertextes....9
2.6.1 Comprendre les chemins relatifs et absolus....10
2.6.2 Lien externe et interne....10
2.6.3 Liens d'ancrage (ancres)....10
2.6.4 Liens spéciaux : email, téléphone, téléchargement....11
2.6.5 Attribut target....11
2.7 Ajouter des images à une page....11
2.8 Les titres en HTML....12
2.9 Paragraphes et retours à la ligne....12
2.10 Séparer visuellement le contenu....12
2.11 Espaces et caractères spéciaux....13
2.12 Mise en forme du texte....13
2.13 Texte préformaté et citations....14
2.14 Les listes en HTML....14
2.15 Les listes de définitions....15
3 Introduction au CSS....15
3.1 Les trois méthodes d'intégration du CSS....16
3.1.1 Feuille de style interne....16
3.1.2 Style en ligne....16
3.1.3 Feuille de style externe....16
3.2 Les sélecteurs CSS....16
3.2.1 Les sélecteurs simples....16
3.2.2 Exercices....17
3.2.3 Les sélecteurs descendants et enfants....18
3.2.4 Les sélecteurs de proximité (frères)....18
3.2.5 Exercice....18
3.3 Les Classes et les Identifiants....19
3.3.1 Syntaxe CSS....19
3.3.2 Utilisation combinée....19
3.3.3 Hiérarchie et spécificité....20
3.3.4 Bonnes pratiques....20
3.3.5 Limites et erreurs fréquentes....20
3.3.6 Mini exercice pratique....20
3.3.7 Exercice sur les selecteur....21
3.4 La Cascade CSS....21

3.4.1 Le mot clé !important....21
3.4.2 La précision du sélecteur....21
3.4.3 L’ordre d’écriture....22
3.4.4 Priorité selon la localisation....22
3.4.5 Mini exercice pratique....22
3.5 L’Héritage CSS....22
3.5.1 Exemple de propriété héritée....22
3.5.2 Propriétés non héritées....22
3.5.3 Modifier les règles d’héritage....23
3.5.4 Mini exercice pratique....23
3.6 Les éléments HTML et 23
3.6.1 Introduction....23
3.6.2 Le rôle du HTML et la place de et 24
3.6.3 L’élément : le conteneur de blocs....24
3.6.4 L’élément : le conteneur en ligne....25
3.6.5 et avec class et id....25
3.7 Mini exercice pratique....26
Consignes....26
3.8 Les niveaux ou types d’éléments block et inline....27
3.8.1 Comprendre la notion de type d’affichage....27
3.8.2 La propriété display....27
3.8.3 Différences entre block et inline....28
Le modèle des boîtes (aperçu)....28
3.8.4 Autres valeurs courantes de display....29
3.9 Les notations complètes (longhand) et raccourcies (shorthand) en CSS....29
3.9.1 Comprendre le principe....29
3.9.2 L’ordre des valeurs dans une shorthand....30
3.9.3 Limites des notations raccourcies....31
3.9.4 Les notations shorthand les plus courantes....31
3.9.5 Exercice pratique....32
Objectif....32
Instructions....32
Résultat attendu....32
3.10 Exercice Recapitulatif des chapitres précédents....33
3.11 Projet fil rouge V1....33

1 Définition et utilisation du HTML et du CSS

Le HTML (HyperText Markup Language) et le CSS (Cascading Style Sheets) sont les deux piliers de toute page web. Le premier définit la structure et le contenu d'un document, tandis que le second en détermine l'apparence visuelle.

1.1 HTML : le langage de structure

Le HTML est un langage de balisage créé en 1991. Il repose sur des éléments appelés balises, qui permettent d'indiquer au navigateur le rôle de chaque contenu : un titre, un paragraphe, une image, une liste, etc. Lorsqu'un utilisateur accède à une page web, son navigateur reçoit un fichier HTML depuis un serveur et l'interprète pour afficher le contenu à l'écran.

Bon à savoir
Un serveur est simplement un ordinateur connecté en permanence à Internet, chargé de stocker les fichiers du site et de les « servir » aux navigateurs lorsque ceux-ci en font la demande.

Voici un exemple minimaliste d'une page HTML complète et valide :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ma première page</title>
  </head>
  <body>
    <h1>Bienvenue sur mon site</h1>
    <p>Ceci est un paragraphe.</p>
  </body>
</html>
```

Les balises <h1> et <p> indiquent respectivement un titre principal et un paragraphe. Le navigateur sait alors comment interpréter et afficher ces contenus.

1.2 CSS : le langage de styles

Le CSS a été créé en 1996 pour compléter le HTML. Là où le HTML structure la page, le CSS permet d'en personnaliser le rendu visuel : couleurs, tailles, marges, alignement, etc. Chaque règle CSS cible un ou plusieurs éléments HTML pour leur appliquer des propriétés de style.

```
h1 {
  color: orange;
  font-size: 24px;
}

p {
  color: blue;
  font-size: 16px;
}
```

Grâce à ce code, le titre principal s’affiche en orange avec une taille de 24 pixels, et les paragraphes en bleu avec une taille de 16 pixels.

Erreur fréquente
Ne jamais utiliser le HTML pour mettre en forme le contenu. Le HTML sert à décrire le sens et la structure des informations, tandis que le CSS gère l’aspect visuel. Confondre les deux dégrade la qualité du code et le référencement du site.

1.3 Histoire et évolution de l’informatique et du web

L’informatique est un domaine en constante évolution. Les progrès du matériel et des connexions Internet ont transformé le web en un environnement interactif et multimédia. Il y a 20 ans, les sites étaient simples, souvent composés uniquement de texte et d’images légères. Aujourd’hui, les pages peuvent intégrer de la vidéo, de l’audio, des animations et des interactions complexes.
L’évolution du web est guidée par plusieurs organisations internationales qui définissent les standards et les bonnes pratiques des langages informatiques.

- W3C : World Wide Web Consortium – supervise HTML et CSS
- WHATWG : Web Hypertext Application Technology Working Group – promeut le HTML Living Standard
- ECMA : organisme responsable du langage JavaScript
- PHP Group et Python Software Foundation – maintiennent PHP et Python

Le W3C classe ses documents selon trois niveaux de maturité : ‘Travail en cours’, ‘Candidat à la recommandation’ et ‘Recommandation officielle’. Seules les recommandations officielles sont considérées comme des standards pleinement reconnus.

Bon à savoir
Le WHATWG préconise une évolution continue du HTML appelée ‘Living Standard’. Cela signifie que le langage s’améliore progressivement sans attendre une nouvelle version majeure.

Les versions HTML et CSS évoluent désormais par modules indépendants : chaque fonctionnalité (gestion des couleurs, typographie, disposition des éléments, etc.) possède sa propre spécification. Cette approche modulaire permet d’intégrer plus rapidement les nouveautés et d’éviter les retards liés à la validation d’une version complète du langage.

Erreur fréquente
Certaines pratiques anciennes (balises obsolètes, attributs de mise en forme, anciens doctypes) ne sont plus reconnues par les navigateurs modernes. Utiliser des éléments non standards peut provoquer des incohérences d’affichage.

Les développeurs doivent donc rester à jour sur les évolutions des langages, vérifier la compatibilité des nouvelles fonctionnalités avec les navigateurs, et se tenir informés des recommandations du W3C.

1.4 Les environnements : local, préproduction et production

Dans le développement web, on distingue trois environnements de travail :

- Local : votre ordinateur personnel, où vous créez et testez vos fichiers HTML/CSS avant toute mise en ligne.
- Préproduction : un espace intermédiaire en ligne qui simule les conditions réelles du site, utile pour les tests collaboratifs.
- Production : le serveur public accessible aux internautes.

Bon à savoir
Travailler localement évite de modifier accidentellement le site en ligne et permet de tester rapidement les changements.

Les développeurs utilisent souvent un serveur local comme XAMPP, WampServer ou MAMP, ou plus récemment DOCKER qui permettent d'exécuter localement des sites dynamiques avec PHP et MySQL.
Une fois le site validé en local, il est transféré sur le serveur de préproduction, puis finalement sur le serveur de production via des outils comme FTP,SSH sur des plateformes d'hébergement (ex. OVH, Infomaniak, etc.).

1.5 Choisir et installer un éditeur de texte

Le choix de l'éditeur de texte influence directement votre confort de travail. Un bon éditeur propose la coloration syntaxique, l'autocomplétion, la gestion des projets et la prévisualisation du code.

- Quelques éditeurs populaires :
- Visual Studio Code – léger, rapide, extensible via des extensions.
- Sublime Text – fluide et personnalisable, adapté aux projets de taille moyenne.
- Atom – interface agréable et intégration Git native.
- Brackets – pensé pour le web avec aperçu en direct.
- PHPSTORM / WEBSTORM – logiciels pro et payants mais vous avez un accès gratuitement grâce à votre adresse mail IFAPME

Bon à savoir
Visual Studio Code est aujourd'hui le plus utilisé pour le développement web grâce à sa communauté active et ses extensions variées.

Erreur fréquente
Évitez les traitements de texte comme Word ou LibreOffice : ils ajoutent des caractères cachés et corrompent le code HTML.

1.6 Préparer la structure d'un site local

Avant de commencer à coder, il est essentiel d'organiser les fichiers de votre site. Une bonne structure facilite la maintenance et le travail collaboratif.
Structure recommandée :
- Un dossier principal nommé selon votre projet (ex. mon-site/).
- Un sous-dossier images/ pour toutes les illustrations.

- Un sous-dossier css/ pour les feuilles de style.
- Un sous-dossier js/ pour les scripts JavaScript.
- Un fichier index.html à la racine du projet.

Illustration textuelle
Schéma typique d’arborescence :

```
mon-site/
├── index.html
├── css/
│   └── style.css
├── images/
│   └── logo.png
└── js/
    └── script.js
```

2 Présentation du HTML

2.1 Éléments, balises et attributs HTML

Le html est composé d’éléments, chacun défini par une balise ouvrante et une balise fermante. Une balise est entourée de chevrons `< >` et indique la fonction du contenu qu’elle encadre.

```
<p>Ceci est un paragraphe.</p>
```

Ici, `<p>` ouvre un paragraphe et `</p>` le ferme. L’ensemble constitue un élément HTML.

Bon à savoir
Certaines balises, dites autofermantes, ne nécessitent pas de balise de fermeture. Par exemple : `<img src="photo.jpg" alt="Description">`.

Les attributs ajoutent des informations supplémentaires à une balise. Ils se placent à l’intérieur de la balise ouvrante et suivent la syntaxe `nom="valeur"`.

```
<a href="https://www.exemple.com">Visiter le site</a>
```

Dans cet exemple, l’attribut `href` indique la destination du lien.

Erreur fréquente
Les attributs ne doivent jamais contenir d’espaces autour du signe égal et les valeurs doivent être entre guillemets.

2.2 Structure minimale d’une page HTML

Une page HTML doit respecter une structure précise pour être valide. Voici le squelette de base :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Titre du document</title>
  </head>
  <body>
    <h1>Titre principal</h1>
    <p>Mon premier paragraphe.</p>
  </body>
</html>
```

Le doctype `<!DOCTYPE html>` indique au navigateur que le document utilise la norme HTML5. L’élément `<head>` contient les métadonnées, et le `<body>` regroupe tout le contenu visible.

Bon à savoir
Les navigateurs modernes sont tolérants, mais un code mal structuré peut provoquer des incohérences d’affichage.

2.3 Indentation et lisibilité du code

L’indentation consiste à décaler le code vers la droite selon la hiérarchie des balises. Cela ne change pas le rendu visuel de la page, mais facilite énormément la lecture du code.

```
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>
```

Chaque niveau de balise doit être indenté de deux ou quatre espaces. Les éditeurs de texte modernes effectuent cette mise en forme automatiquement.

Erreur fréquente
Une indentation incohérente rend le code difficile à maintenir et augmente les risques d’oublier une balise fermante.

2.4 Commentaires en HTML

Les commentaires permettent d’ajouter des notes invisibles pour l’utilisateur. Ils sont très utiles pour documenter le code ou séparer les sections.

```
<!-- Ceci est un commentaire →
```

Les commentaires ne s’affichent pas dans le navigateur mais restent visibles dans le code source.

Bon à savoir
Commentez régulièrement vos fichiers, surtout si plusieurs développeurs travaillent sur le même projet.

2.5 Enregistrer et afficher une page HTML

Pour créer votre première page web, il suffit d’un éditeur de texte et d’un navigateur. Après avoir écrit le code HTML, vous devez enregistrer votre fichier avec l’extension `.html`.

Bon à savoir
Le nom du fichier ne doit pas contenir d’espaces ni de caractères spéciaux. Utilisez uniquement des lettres, chiffres et tirets. Exemple : `ma-page.html`.

Bon à savoir
Le nom du fichier n’est pas toujours visible sur Windows (et mac?) Il faut absolument afficher les extensions dans la configuration des dossiers

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ma première page</title>
  </head>
  <body>
    <h1>Bonjour tout le monde !</h1>
  </body>
</html>
```

Une fois le fichier enregistré, double-cliquez dessus ou glissez-le dans un navigateur pour l’afficher. Vous verrez apparaître votre titre et le contenu du corps de la page.

2.6 Créer des liens hypertextes

Les liens hypertextes permettent de naviguer entre différentes pages ou vers des sites externes. Ils sont définis par la balise `<a>` (pour ‘anchor’).

```
<a href="https://www.example.com">Visitez mon site</a>
```

L’attribut `href` précise la destination du lien. Lorsqu’on clique sur le texte entre les balises `<a>` et `</a>`, le navigateur charge la page correspondante.

Bon à savoir
Pour créer un lien vers une autre page du même site, indiquez un chemin relatif : `<a href="contact.html">Page de contact</a>`.

Erreur fréquente
Ne mettez pas d’espaces dans les noms de fichiers liés. Préférez `page-contact.html` à `page contact.html`.

2.6.1 Comprendre les chemins relatifs et absolus

Un chemin absolu indique la position complète d’un fichier sur Internet, tandis qu’un chemin relatif indique sa position par rapport au fichier HTML courant.

```
<!-- Chemin absolu -->
<img src="https://www.monsite.com/images/logo.png" alt="Logo">

<!-- Chemin relatif -->
<img src="images/logo.png" alt="Logo">
```

L’usage des chemins relatifs est recommandé pour la portabilité : vos liens fonctionneront même si vous déplacez le site sur un autre serveur.

2.6.2 Lien externe et interne

Exemple de lien externe (adresse absolue) :

```
<a href="https://www.openai.com">Visiter OpenAI</a>
```

Exemple de lien interne (adresse relative) :

```
<a href="contact.html">Contactez-nous</a>
```

2.6.3 Liens d’ancrage (ancres)

Les ancres permettent de pointer vers une section spécifique d’une page en utilisant un identifiant unique.

```
<a href="#apropos">Aller à la section "À propos"</a>
<h2 id="apropos">À propos</h2>
<p>Bienvenue dans notre section à propos…</p>
```

2.6.4 Liens spéciaux : email, téléphone, téléchargement

```
<a href="mailto:info@exemple.com">Contactez-nous</a>
<a href="tel:+33123456789">Appeler le support</a>
<a href="image.png" download>Télécharger</a>
<a href="image.png" download="nouvelle-image.png">Télécharger sous un autre
nom</a>
```

2.6.5 Attribut target

L'attribut target détermine où le lien s'ouvre (_self, _blank, _parent, _top). Exemple :

```
<a href="https://developer.mozilla.org" target="_blank" rel="noopener
noreferrer">Documentation MDN</a>
```

Valeurs possibles	Description
_blank	Ouverture de la page cible dans une nouvelle fenêtre
_parent	Ouverture de la page cible dans le cadre parent
_self	Ouverture de la page cible dans le cadre hôte
_top	Ouverture de la page cible dans la fenêtre hôte
Nom du cadre	Ouverture de la page cible dans le cadre portant le nom cité

2.7 Ajouter des images à une page

La balise `<img>` permet d'afficher une image dans une page web. Elle utilise au minimum l'attribut `src` pour indiquer le chemin du fichier image, et l'attribut `alt` pour fournir un texte alternatif.

```
<img src="images/photo.jpg" alt="Photo de paysage">
```

L'attribut `alt` est essentiel pour l'accessibilité : il permet aux lecteurs d'écran de décrire l'image aux personnes malvoyantes.

Bon à savoir
Le texte alternatif (`alt`) s'affiche aussi si l'image ne peut pas être chargée. Utilisez un texte concis et descriptif.

Erreur fréquente
Évitez d'utiliser des images trop lourdes. Optimisez-les pour le web afin d'accélérer le chargement de vos pages.

2.8 Les titres en HTML

Les titres servent à organiser le contenu d'une page web. Ils vont de `<h1>` à `<h6>`, où `<h1>` représente le niveau le plus important et `<h6>` le moins important.

```
<h1>Titre principal</h1>
<h2>Sous-titre</h2>
<h3>Sous-section</h3>
```

Chaque page ne doit contenir qu'un seul `<h1>`, généralement utilisé pour le titre principal. Les autres niveaux structurent les sous-parties.

Bon à savoir
Les moteurs de recherche comme Google utilisent la hiérarchie des titres pour comprendre la structure de la page. Une bonne utilisation améliore le référencement naturel (SEO).

2.9 Paragraphes et retours à la ligne

Le texte d'une page est principalement contenu dans des paragraphes, définis par la balise `<p>`. Chaque paragraphe doit être clairement séparé des autres.

```
<p>Ceci est un premier paragraphe.</p>
<p>Ceci est un second paragraphe.</p>
```

Pour forcer un simple retour à la ligne sans créer de nouveau paragraphe, on utilise la balise `<br>`, qui est auto-fermante.

```
<p>Ligne 1<br>Ligne 2<br>Ligne 3</p>
```

Erreur fréquente
N'utilisez pas plusieurs balises `<br>` à la suite pour créer des espaces. Utilisez plutôt le CSS pour gérer les marges entre les éléments.

2.10 Séparer visuellement le contenu

La balise `<hr>` crée une ligne horizontale pour séparer les sections d'une page. Elle est utile pour aérer visuellement le contenu.

```
<h2>Introduction</h2>
<p>Texte d'introduction...</p>
<hr>
<h2>Conclusion</h2>
<p>Texte de conclusion...</p>
```

Bon à savoir
La balise `<hr>` est purement visuelle. Pour structurer le contenu, utilisez toujours les titres (`<h1>` à `<h6>`) et les balises sémantiques appropriées.

2.11 Espaces et caractères spéciaux

En HTML, plusieurs espaces consécutifs sont interprétés comme un seul. Pour insérer un espace insécable (qui empêche le retour à la ligne), on utilise le code ` `.

```
<p>Prix : 10 €</p>
```

De même, certains symboles nécessitent un code spécial, car ils sont réservés par le HTML.

```
&lt; pour <
&gt; pour >
&amp; pour &
```

Bon à savoir
Les entités HTML assurent que le navigateur affiche correctement les symboles réservés. Elles commencent toujours par `&` et se terminent par `;`.

2.12 Mise en forme du texte

Le HTML permet de mettre en valeur certaines portions de texte à l'aide de balises spécifiques. Cependant, ces balises ne doivent pas être confondues avec la mise en forme visuelle, qui relève du CSS.

```
<p><strong>Texte important</strong></p>
<p><em>Texte en italique</em></p>
<p><mark>Texte surligné</mark></p>
<p><small>Texte plus petit</small></p>
```

Les balises `<strong>` et `<em>` ont une valeur sémantique : elles indiquent une importance ou une emphase. Le navigateur les affiche souvent en gras et en italique par défaut, mais leur signification dépasse le simple style.

Bon à savoir
Les lecteurs d'écran utilisent la sémantique du HTML pour adapter la lecture du contenu. L'usage correct des balises améliore donc l'accessibilité de votre site.

2.13 Texte préformaté et citations

Le HTML propose des balises pour afficher du texte tel qu'il est écrit dans le code, ainsi que pour insérer des citations.

```
<pre>
Ligne 1
  Ligne 2 (indentée)
Ligne 3
</pre>
```

La balise `<pre>` conserve les espaces et retours à la ligne du texte source, utile pour afficher du code ou des exemples de texte formaté.

```
<blockquote>
  Ceci est une citation plus longue extraite d'un texte.
</blockquote>
```

```
<p>Comme disait <q>Einstein</q> : tout est relatif.</p>
```

La balise `<blockquote>` crée un bloc de citation, tandis que `<q>` insère une citation courte en ligne.

> Erreur fréquente
> Ne combinez pas des balises de citation avec des balises de mise en forme pour simuler des styles visuels. Le CSS doit gérer l'apparence.

2.14 Les listes en HTML

Les listes sont des éléments essentiels pour structurer l'information. Il existe deux types principaux : les listes non ordonnées et les listes ordonnées.

```
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>

<ol>
  <li>Étape 1</li>
  <li>Étape 2</li>
  <li>Étape 3</li>
</ol>
```

La balise `<ul>` crée une liste à puces, tandis que `<ol>` crée une liste numérotée. Chaque élément de liste est défini par la balise `<li>`.

Bon à savoir
Les listes peuvent être imbriquées pour représenter des hiérarchies d’informations. Veillez à conserver une indentation claire pour la lisibilité.

2.15 Les listes de définitions

Les listes de définitions permettent d’associer des termes à leurs descriptions. Elles utilisent les balises `<dl>`, `<dt>` et `<dd>`.

```
<dl>
  <dt>HTML</dt>
  <dd>Langage de structure des pages web</dd>
  <dt>CSS</dt>
  <dd>Langage de mise en forme</dd>
</dl>
```

Ces listes sont souvent utilisées dans les glossaires, les FAQ ou les présentations de caractéristiques techniques.

Erreur fréquente
N’utilisez pas de listes de définitions pour faire de simples listes à puces. Elles ont une sémantique spécifique et doivent être réservées aux associations terme/définition

3 Introduction au CSS

Le CSS (Cascading Style Sheets) permet de contrôler l’apparence d’une page HTML. Il agit sur la couleur, la taille, les marges, la position et d’autres propriétés visuelles des éléments HTML.

```
h1 {
  color: darkblue;
  font-size: 32px;
}
```

Dans cet exemple, la règle CSS s’applique à tous les titres `<h1>` de la page, leur donnant une couleur bleue foncée et une taille de police de 32 pixels.

Bon à savoir
Le terme *cascading* (cascade) fait référence à la manière dont les styles sont appliqués : si plusieurs règles ciblent le même élément, la priorité dépend de leur ordre et de leur spécificité.

3.1 Les trois méthodes d’intégration du CSS

Il existe trois façons principales d’ajouter du CSS à une page HTML : interne, externe et en ligne.

3.1.1 Feuille de style interne

Le style interne s’intègre directement dans le fichier HTML, à l’intérieur de la balise `<head>`. Il est utile pour tester ou styliser rapidement une seule page. Ce n’est pas une bonne méthode en général !

3.1.2 Style en ligne

```
<p style="color:red; font-size:16px;">Texte rouge</p>
```

Le style en ligne applique des règles CSS directement sur un élément HTML grâce à l’attribut `style`. Cette méthode est déconseillée pour les sites complets, car elle rend le code difficile à maintenir.

Erreur fréquente
Évitez de mélanger les styles en ligne, internes et externes dans une même page. Cela complique le débogage et provoque souvent des conflits de priorités.

3.1.3 Feuille de style externe

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

La feuille de style externe est le moyen le plus recommandé. Elle permet de séparer totalement la structure (HTML) et la présentation (CSS). Les modifications de style s’appliquent à tout le site.

```
<style>
p {
  color: green;
  font-size: 18px;
}
</style>
```

3.2 Les sélecteurs CSS

3.2.1 Les sélecteurs simples

Un sélecteur simple cible directement un type d’élément HTML, par exemple tous les paragraphes ou tous les titres.

```
p {
  color: blue;
}
```
→ Ici, tous les éléments `<p>` du document seront affichés en bleu.

```
h1, h2 {
  font-family: Arial, sans-serif;
}
```
→ Ce sélecteur multiple applique la même police à tous les titres `<h1>` et `<h2>`.

3.2.2 Exercices

Reproduisez en html :

Ex1 :

Dans le premier mot, les tailles des caractères sont 4em, 5em, 6em et 7em tandis que dans le second mot, les couleurs sont à prendre dans #ff0000, #123456, #0000ff, #37f8d3, #ff00ff, #000000 et #324890.
la balise à utiliser est ... . Donc votre code ressemblera à :

Concve
span>

Utilisez un style en ligne !

Concave

Couleur

Ex2 :

La font est a 50px L'espace entre les mots est de 15 pixels tandis que celui entre les lettres est de 2 pixels (sauf le dernier mot où la valeur est 15 pixels).

Le texte en **caractères gras**
Le texte en caractès soulignés
Le texte en caractères *italiques*
Le texte en caractères ~~rayés~~
Le texte en caractères plus gros
Le texte en caractères plus petits
Le texte en caractères exposant
Le texte en caractères indice
1 Le texte en caractères e s p a c é s

3.2.3 Les sélecteurs descendants et enfants

Les sélecteurs descendants ciblent les éléments contenus dans d'autres éléments.
Cela permet de styliser un lien, une image ou un paragraphe en fonction de son contexte HTML.

```
p a {
  color: red;
}
```

→ Ici, seuls les liens situés à l'intérieur d'un paragraphe `<p>` seront rouges.

```
body > a {
  color: green;
}
```

→ Le signe `>` indique qu'on cible uniquement les éléments `<a>` qui sont des enfants directs de `<body>` (pas les liens imbriqués plus profondément).

```
* {
  color: green;
}
```

→ Le signe `*` indique « toutes les balises »

3.2.4 Les sélecteurs de proximité (frères)

Les sélecteurs de proximité permettent de cibler un élément qui suit ou partage le même parent qu'un autre.

```
p + a {
  color: orange;
}
```

→ Sélectionne le lien `<a>` qui suit directement un paragraphe `<p>` (le premier frère).

```
p ~ a {
  color: purple;
}
```

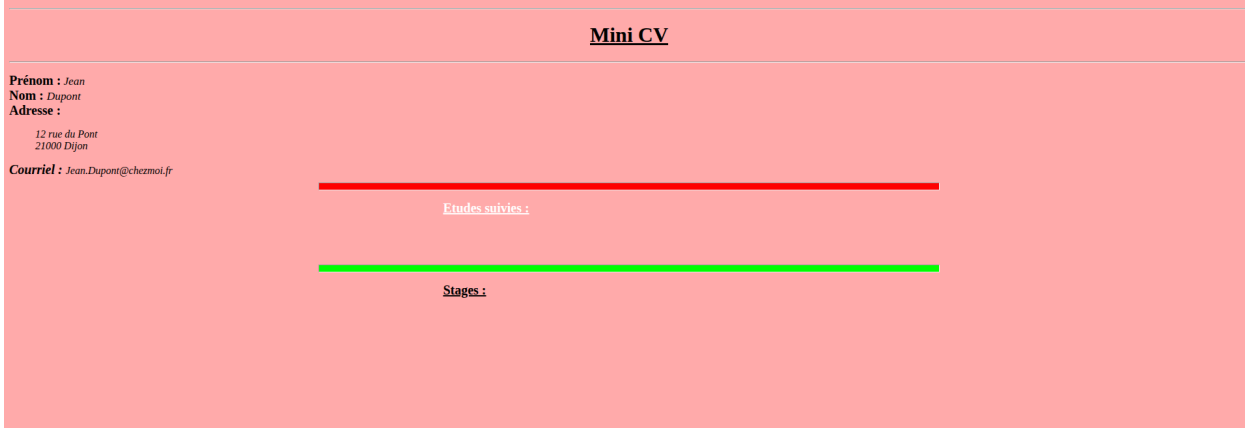
→ Sélectionne tous les liens `<a>` qui suivent un paragraphe `<p>` ayant le même parent.

Exercices :

3.2.5 Exercice

Réaliser ceci en html/css

Le fond est rose. Il existe una balise permettant de spécifier une adresse (postale ou de courriel)
rappel: hr permet de séparer des parties distinctes
la balise blockquote est utilisée...



FIN G1

3.3 Les Classes et les Identifiants

Les attributs 'class' et 'id' permettent d'identifier des éléments HTML afin de leur appliquer des styles CSS précis. Ils sont essentiels pour structurer une page et rendre le code lisible et réutilisable.

```
<p class="important">Texte important</p>
<p id="intro">Introduction</p>
```

3.3.1 Syntaxe CSS

En CSS, on utilise les symboles suivants pour cibler ces attributs : le point (.) pour les classes et le dièse (#) pour les identifiants.

```
.important { color: red; }
#intro { font-weight: bold; }
```

Bon à savoir :
Les classes sont réutilisables sur plusieurs éléments, tandis que l'identifiant doit rester unique.

3.3.2 Utilisation combinée

Il est possible d'attribuer plusieurs classes à un même élément HTML.

```
<p class="important grand">Texte mis en avant</p>

.important { color: red; }
.grand { font-size: 20px; }
```

→ Le texte sera rouge et en taille 20px grâce à la combinaison des deux classes.

Bon à savoir :
Les classes peuvent être combinées dans les sélecteurs CSS pour des cas spécifiques (ex. p.important.grand).

3.3.3 Hiérarchie et spécificité

La spécificité CSS détermine quelle règle s'applique lorsqu'un même élément est ciblé par plusieurs sélecteurs.

```
p { color: blue; }
.important { color: orange; }
#intro { color: red; }
```

→ Si un paragraphe possède la classe 'important' et l'id 'intro', la couleur rouge l'emportera car l'identifiant a une spécificité plus forte.

Attention :
Les identifiants (#) ont toujours une spécificité supérieure aux classes (.) et aux éléments simples.

3.3.4 Bonnes pratiques

Voici quelques règles simples pour bien utiliser les classes et identifiants :
- Utiliser les classes pour le style visuel.
- Réserver les identifiants pour les ancres ou les interactions JavaScript.
- Employer des noms explicites et cohérents (ex. .menu-principal, #header).
- Ne pas combiner class et id pour le même effet visuel.

Bon à savoir :
Les frameworks modernes comme Bootstrap ou Tailwind reposent uniquement sur des classes, jamais sur des identifiants, pour plus de flexibilité.

3.3.5 Limites et erreurs fréquentes

Quelques points d'attention à garder en tête lors de l'utilisation des classes et identifiants :

Attention :
Un identifiant doit toujours être unique sur une page. Son utilisation répétée provoque des conflits CSS et JavaScript.

```
<p class="1texte">Texte</p> <!-- Erreur -->
<p class="texte-1">Texte</p> <!-- Correct -->
```

→ Les noms ne doivent pas commencer par un chiffre ni contenir d'espaces. Utilisez plutôt des tirets ou underscores.

3.3.6 Mini exercice pratique

1. Crée une page avec trois paragraphes :
- Un avec class='important'

- Un avec class='note'
- Un avec id='intro'

2. Applique ces styles CSS :

```
.important { color: red; }
.note { font-style: italic; }
#intro { font-weight: bold; }
```

3. Observe la priorité des styles si tu ajoutes !important à certaines règles.

4. Combine plusieurs classes sur un même paragraphe et analyse le résultat.

3.3.7 Exercice sur les selecteur

https://flukeout.github.io/ Jusqu'au 14 compris !

3.4 La Cascade CSS

La cascade CSS détermine quels styles doivent être appliqués à un élément lorsque plusieurs règles coexistent. Trois critères principaux déterminent la priorité : la présence du mot-clé !important, la précision du sélecteur et l'ordre d'écriture.

3.4.1 Le mot clé !important

Ce mot clé force l'application d'une règle CSS, la rendant prioritaire sur toutes les autres.

```
p {
  color: blue !important;
}
```

Bon à savoir :
Le mot clé !important est utile pour forcer un style ponctuel, mais son usage excessif complique la maintenance.

Attention :
Évitez d'utiliser !important dans les feuilles de style globales. Préférez la bonne hiérarchie de sélecteurs.

3.4.2 La précision du sélecteur

La précision dépend du type de sélecteur utilisé. Plus un sélecteur cible un élément spécifique, plus il est prioritaire.

```
#id > .class p { color: red; }
```

Ordre de précision décroissant : Style en ligne > #id > .class > élément.

3.4.3 L’ordre d’écriture

Quand deux règles ont la même précision, la dernière lue par le navigateur l’emporte.

```
p { color: blue; }
p { color: red; } /* Le texte sera rouge */
```

Bon à savoir :
Le navigateur lit les fichiers CSS de haut en bas. La dernière règle applicable est retenue.

3.4.4 Priorité selon la localisation

L’ordre de lecture des styles influe aussi selon leur emplacement : les styles internes (<style>) sont prioritaires sur les fichiers CSS externes.

```
<link rel='stylesheet' href='style.css'>
<style>
  p { color: red; }
</style>
```

3.4.5 Mini exercice pratique

Créez une page contenant deux paragraphes. Appliquez trois règles contradictoires : dans un fichier externe, dans une balise <style> et dans un attribut style. Observez laquelle s’applique.

3.5 L’Héritage CSS

L’héritage CSS désigne le fait que certains styles appliqués à un élément parent se propagent à ses enfants. Mais toutes les propriétés ne sont pas héritables.

3.5.1 Exemple de propriété héritée

```
html { font-family: Arial; }
h1 { font-family: 'Times New Roman'; }
```

→ Tous les textes utilisent Arial, sauf les titres h1 qui héritent d’une police différente.

3.5.2 Propriétés non héritées

```
ul { margin-left: 50px; }
```

Les éléments enfants (li) ne reprennent pas cette marge, car la propriété margin n’est pas héritée.

Bon à savoir :
Les propriétés liées au texte (font, color) sont souvent héritées. Celles liées à la mise en page (margin, padding, border) ne le sont pas.

3.5.3 Modifier les règles d’héritage

Quatre valeurs universelles peuvent modifier le comportement d’héritage : initial, inherit, unset et revert.

```
p {
  color: inherit;
  margin: initial;
}
```

Mot-clé	Effet principal	Héritage naturel ?	Revient à...
`initial`	Valeur CSS standard par défaut	ignoré	Valeur **de la spécification CSS**
`inherit`	Force la valeur du parent	forcé	Valeur **du parent**
`unset`	`inherit` si hérité, sinon `initial`	dépend	Mix des deux
`revert`	Annule les styles “auteur” et revient au navigateur	dépend	Valeur **du navigateur ou de l’utilisateur**

Attention :
L’usage combiné de ces valeurs peut provoquer des comportements inattendus si les styles sont déjà surchargés ailleurs.

3.5.4 Mini exercice pratique

Créez des paragraphes à l’intérieur d’un div. Appliquez une couleur sur le div et testez les valeurs inherit, initial, unset et revert sur le paragraphe pour observer les différences.

```
<div class="parent">
  <p>Texte normal</p>
  <p class="initial">color: initial</p>
  <p class="inherit">color: inherit</p>
  <p class="unset">color: unset</p>
  <p class="revert">color: revert</p>
</div>
```

3.6 Les éléments HTML `<div>` et `<span>`

3.6.1 Introduction

Les éléments `<div>` et `<span>` sont des **conteneurs génériques** utilisés pour structurer le contenu HTML **sans lui attribuer de signification particulière**.

Contrairement aux balises sémantiques comme `<p>` ou `<h1>`, ils n’ont **aucune valeur sémantique** et servent principalement à **l’organisation et au style via CSS**.

Bon à savoir :
Le mot *sémantique* signifie “qui a du sens”.
`<p>` représente un paragraphe, `<h1>` un titre… tandis que `<div>` et `<span>` ne décrivent pas le contenu, ils structurent seulement la page.

3.6.2 Le rôle du HTML et la place de `<div>` et `<span>`

Le HTML sert à **structurer le contenu** d’une page et à lui donner du sens.
Chaque balise a un rôle précis : paragraphe, liste, image, etc.

Les éléments `<div>` et `<span>` interviennent lorsque nous avons besoin de **regrouper ou cibler certaines zones sans signification spécifique**.
Ils facilitent l’organisation du code et l’application des styles CSS.

Attention :
Évitez d’utiliser trop de `<div>` et `<span>` sans raison : cela rend le code illisible et confus.
On parle alors de **“divitis”**.

3.6.3 L’élément `<div>` : le conteneur de blocs

L’élément `<div>` regroupe plusieurs éléments HTML à l’intérieur d’un même conteneur.
Il est souvent utilisé pour définir des **zones principales** de la page.

Exemple HTML :

```
<div id="conteneur">
  <h2>Titre</h2>
  <p>Texte dans un conteneur.</p>
</div>
```

Exemple CSS :

```
#conteneur {
  background-color: lightblue;
  padding: 10px;
  border-radius: 5px;
}
```

Bon à savoir :
`<div>` est un élément **de type block** : il occupe toute la largeur et commence toujours sur une nouvelle ligne.

Attention :
Ne remplacez pas des balises sémantiques (`<header>`, `<main>`, `<footer>`) par des `<div>`.
Les balises sémantiques sont meilleures pour le **référencement (SEO)** et l’accessibilité.

3.6.4 L’élément : le conteneur en ligne

`<span>` sert à **cibler une portion de texte** spécifique à l’intérieur d’un élément, sans modifier la structure du document.

Exemple HTML :

```
<p>Voici un <span class="accent">mot important</span> dans la phrase.</p>
```

Exemple CSS :

```
.accent {
  color: red;
  font-weight: bold;
}
```

Bon à savoir :
`<span>` est un élément **inline (en ligne)** : il ne crée pas de retour à la ligne et ne prend que la largeur de son contenu.

Attention :
N’insérez pas de grands blocs (comme `<div>` ou `<p>`) à l’intérieur d’un `<span>`.
Les éléments inline ne doivent contenir que du **texte** ou d’autres **inline**.

3.6.5 et avec `class` et `id`

Ces éléments sont souvent associés aux **attributs `class` et `id`** pour appliquer des styles précis.

Exemple HTML :

```
<div class="section-principale">
  <p id="intro">Bienvenue sur mon site</p>
</div>
```

Exemple CSS :

```
.section-principale {
  background-color: beige;
}

#intro {
  font-style: italic;
}
```

Bon à savoir :
Les attributs class et id sont universels : ils peuvent être utilisés sur n’importe quel élément HTML.
6. Différences entre et

Les deux balises servent à structurer le contenu, mais leur comportement diffère :

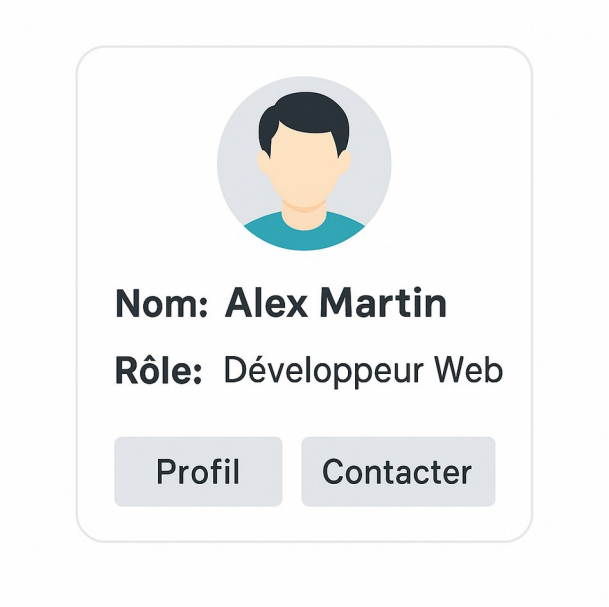
Élément	Type	Utilisation principale
	Block	Regroupe plusieurs éléments
	Inline	Cible une portion de texte

```
<div>
  <p>Texte dans un bloc</p>
</div>

<p>Texte avec un <span>mot coloré</span>.</p>
```

Bon à savoir :
`<div>` et `<span>` sont souvent utilisés avec **CSS** ou **JavaScript** pour cibler des sections précises ou des parties de texte.

3.7 Mini exercice pratique



Consignes

1. Crée une structure HTML avec des `<div>` pour organiser les sections :
    - Un conteneur global (id `profil-card`)
    - Une zone image
    - Une zone d'informations (nom, rôle)
    - Une zone d'actions avec deux boutons.
2. Utilise des classes (`.photo`, `.infos`, `.actions`, etc.) pour appliquer des styles CSS. Utilise un ou deux `id` si nécessaire (par exemple `#profil-card` pour la carte principale).

3. Ajoute les styles CSS suivants :
    - Bordure arrondie et ombre sur la carte
    - Couleur de fond légère
    - Le nom en gras et plus grand
    - Le rôle en dessous
    - Une image ronde en haut (base carrée ou rectangle)
    - Deux boutons : “Profil” et “Contacter” côte à côte avec un effet au survol (`:hover`)

3.8 Les niveaux ou types d’éléments block et inline

3.8.1 Comprendre la notion de type d’affichage

En CSS, chaque élément HTML possède un **type d’affichage** (ou *display*) qui détermine la **façon dont il se place dans la page** et **interagit avec les autres éléments**.
C’est un concept essentiel : il influence la mise en page, l’espace occupé, et même les propriétés CSS applicables.

Par défaut, tout élément HTML est rendu avec un certain type d’affichage :

- **block** → affiche un bloc complet sur une nouvelle ligne,
- **inline** → s’affiche à la suite du texte,
- **none** → n’est pas affiché du tout.

Bon à savoir :
La propriété CSS `display` contrôle le comportement d’un élément.
Même si elle n’est pas précisée dans ton code, le navigateur applique une **valeur par défaut** définie dans sa feuille de styles interne.

3.8.2 La propriété `display`

Tu peux modifier le type d’affichage de n’importe quel élément grâce à la propriété `display` :

```
p {
  display: block;
}
span {
  display: inline;
}
.menu {
  display: flex;
}
```

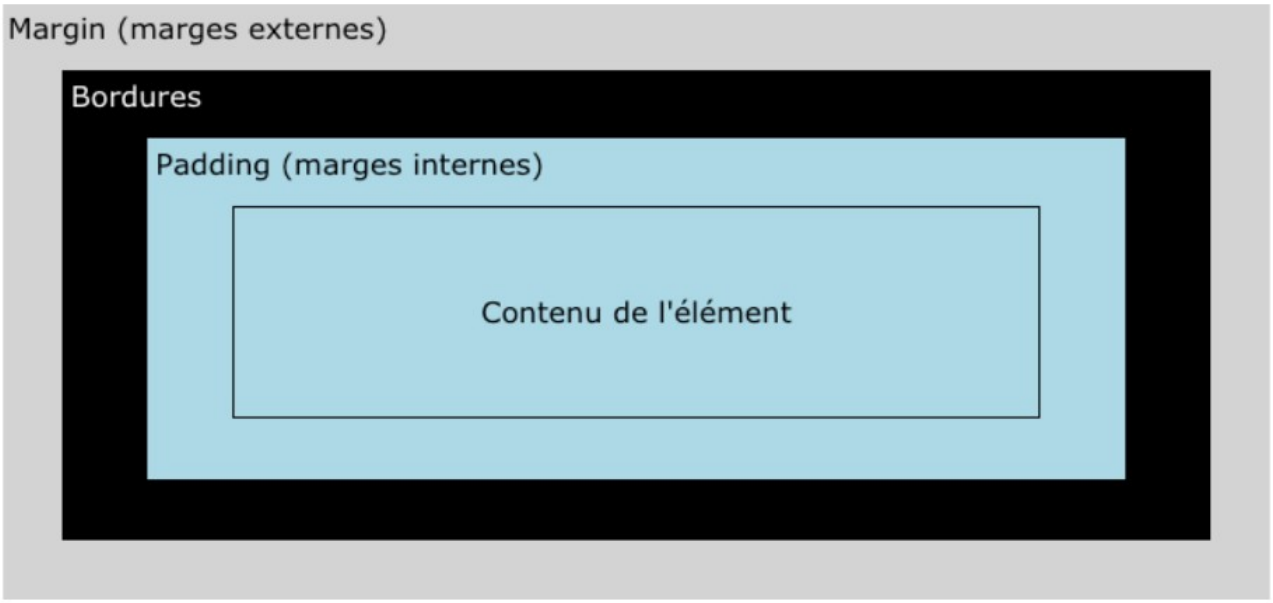
Bon à savoir :
Les navigateurs suivent en général les recommandations du **W3C**, mais tu peux toujours **surcharger ces valeurs** dans ton propre fichier CSS.

3.8.3 Différences entre block et inline

Type	Comportement	Exemple d'éléments
block	Occupe toute la largeur disponible. Commence sur une nouvelle ligne.	`<div>`, `<p>`, `<section>`, `<h1>`, `<article>`
inline	N'occupe que la largeur de son contenu. Peut apparaître à côté d'autres éléments.	`<span>`, `<a>`, `<em>`, `<strong>`, `<img>`

Bon à savoir :
Certains éléments (comme `<img>`) sont *inline* mais peuvent recevoir des dimensions (width/height) car ils sont considérés comme **éléments remplacés**.

Le modèle des boîtes (aperçu)



Tout élément HTML est représenté comme une **boîte rectangulaire invisible**.
Ce qu'on appelle le **modèle des boîtes** (*box model*).
Chaque boîte comprend :

- le **contenu** (textes, images...),
- le **remplissage** (*padding*),

- la **bordure** (*border*),
- la **marge externe** (*margin*).

Bon à savoir :
Les éléments *block* respectent entièrement le modèle des boîtes, tandis que les *inline* ne tiennent compte que du contenu (padding vertical et margin vertical sont souvent ignorés).

3.8.4 Autres valeurs courantes de `display`

Valeur	Description	Exemple d'utilisation
`inline-block`	Mélange les comportements inline et block.	Créer des boutons alignés.
`list-item`	Appliqué aux `<li>`. Crée un bloc avec un marqueur.	Listes.
`table`	Simule le comportement d'un tableau.	Structures complexes.
`none`	Cache complètement l'élément.	Masquage temporaire.

Bon à savoir :
Aujourd'hui, les nouvelles valeurs comme `flex` et `grid` permettent de gérer facilement des mises en page modernes — mais elles reposent toujours sur les **fondations block/inline**.

3.9 Les notations complètes (longhand) et raccourcies (shorthand) en CSS

3.9.1 Comprendre le principe

En CSS, une **notation raccourcie (shorthand)** permet de **définir plusieurs propriétés en une seule ligne**, plutôt que d'écrire chaque propriété séparément (notation longue ou *longhand*).

C'est un **gain de temps**, un **gain de clarté**, mais qui nécessite de **connaître l'ordre** et les **valeurs par défaut** des propriétés concernées.

Exemple typique : la bordure.

/* Notation longue */

border-width: 2px;

border-style: solid;

border-color: blue;

/* Notation raccourcie */

border: 2px solid blue;

Bon à savoir :

Les propriétés shorthand regroupent **plusieurs propriétés connexes** (par exemple, tout ce qui concerne la bordure, la marge, ou la police).
Elles ont été créées pour simplifier l'écriture du code et rendre les feuilles de style plus lisibles.

3.9.2 L'ordre des valeurs dans une shorthand

L'ordre **a toujours une importance** : il permet au navigateur de savoir quelle valeur s'applique à quelle propriété.

Prenons l'exemple de `padding`, qui est la version raccourcie des quatre marges internes :

padding-top

padding-right

padding-bottom

padding-left

Exemple d'utilisation :

padding: 10px 20px 15px 5px;

Cet ordre est **dans le sens des aiguilles d'une montre** :
→ **haut**, **droite**, **bas**, **gauche**.

Si tu n'écris pas les quatre valeurs, le navigateur applique des **règles par défaut** :

Déclaration	Interprétation
`padding: 10px;`	10px sur les 4 côtés
`padding: 10px 20px;`	10px haut/bas, 20px gauche/droite
`padding: 10px 20px 5px;`	haut=10px, droite/gauche=20px, bas=5px

```
p {
  font-style: italic;
  font-weight: bold;
  font-size: 16px;
  font-family: Arial;
}

p.special {
  font: 14px Verdana;
}
```

ésultat :

> Le paragraphe `.special` **n’est plus en italique ni en gras**,
> car `font-style` et `font-weight` sont revenus à leurs **valeurs par défaut** (`normal`).

Attention :
Les propriétés shorthand **écrasent toutes les propriétés longhand associées**.
Utilise-les donc **avec prudence**, surtout quand tu veux conserver certains styles existants.

3.9.3 Limites des notations raccourcies

Les notations shorthand sont pratiques, mais elles ont quelques contraintes :

1. Impossible d’utiliser les valeurs globales `inherit`, `initial` ou `unset` à l’intérieur d’une shorthand.
2. Pas d’héritage automatique des valeurs omises : celles-ci reviennent à leurs **valeurs initiales**.
3. Moins explicites pour les débutants : elles masquent la granularité du style.

Bon à savoir :
Pour les réglages fins ou dynamiques, il vaut mieux utiliser les **notations longues**.
Pour la mise en page générale, les **notations raccourcies** sont parfaites.

3.9.4 Les notations shorthand les plus courantes

Propriété Shorthand	Équivalent(s) Longhand
`font`	`font-style`, `font-variant`, `font-weight`, `font-size`, `line-height`, `font-family`
`border`	`border-width`, `border-style`, `border-color`
`margin`	`margin-top`, `margin-right`, `margin-bottom`, `margin-left`
`padding`	`padding-top`, `padding-right`, `padding-bottom`, `padding-left`
`background`	`background-image`, `background-position`, `background-size`, `background-repeat`, `background-origin`, `background-clip`, `background-attachment`, `background-color`
`transition`	`transition-property`, `transition-duration`, `transition-timing-function`, `transition-delay`
`animation`	`animation-name`, `animation-duration`, `animation-timing-function`, `animation-delay`, `animation-iteration-count`, `animation-direction`, `animation-fill-mode`, `animation-play-state`
`flex`	`flex-grow`, `flex-shrink`, `flex-basis`

Bon à savoir :

Toutes les propriétés n’ont **pas** d’équivalent raccourci.
Le CSS conserve certaines notations longues pour garantir la clarté et la compatibilité.

3.9.5 Exercice pratique

Objectif

Comparer les notations shorthand et longhand à travers un cas concret.

Instructions

Crée une page HTML avec une boîte de texte stylisée deux fois : une avec des notations longues, une avec une shorthand.

```
<div class="longhand">Notation longue</div>

<div class="shorthand">Notation courte</div>
```

```
.longhand {
  border-width: 3px;
  border-style: solid;
  border-color: blue;
  background-color: lightyellow;
  padding-top: 10px;
  padding-right: 20px;
  padding-bottom: 10px;
  padding-left: 20px;
}


.shorthand {
  border: 3px solid blue;
  background: lightyellow;
  padding: 10px 20px;
}
```

Résultat attendu

Les deux boîtes affichent **le même rendu visuel** :

- une bordure bleue,
- un fond jaune clair,

- un texte centré avec des marges internes.

Bon à savoir :
L'exercice montre que le CSS raccourci rend le code plus léger et lisible, mais qu'il **ne remplace pas toujours la précision** de la notation longue.

3.10 Exercice Recapitulatif des chapitres précédents

CSS Diner : un jeu interactif sur les sélecteurs CSS (élément, classe, id, combinateurs) — tu tapes un sélecteur pour cibler les bons éléments. flukeout.github.io

Pour ceux qui veulent, d'autres propositions

Selector Showdown : un jeu axé sur la spécificité CSS — tu dois choisir quel sélecteur "gagne". Codepip

CSSBattle : un défi où tu dois reproduire une image via CSS avec le code le plus court possible — parfait pour l'exercice sur les notations shorthand vs longhand. cssbattle.dev

CSS Speedrun : un quiz / challenge rapide sur les sélecteurs CSS. CSS Speedrun | Test your CSS Skills

3.11 Projet fil rouge V1

Création d'un site de présentation vous concernant, vous pourrez le mettre en ligne pour vous aider dans votre recherche de stage.

Inspiration https://preview.themeforest.net/item/careerster-cvresume-elementor-templates/full_screen_preview/25950036

Vous etes libre de partir d'un autre graphisme, mais il vous faut une base !

3 pages : présentation, portfolio et contact
Step by step. Vous ne faite QUE ce que l'on a déjà vu. Commencez par positionner les elements de la page d'accueil (vous piochez dans le layout les elements qui vous plaisent et sont utiles. Skils, experience,…