

HTML

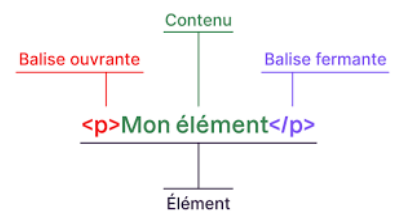
HyperText Markup Language

Langage de balisage hypertexte

Le **HTML** est un langage de balisage utilisé pour créer et structurer le contenu des pages web.

L'**hypertexte** est un système de gestion de l'information qui permet de créer des liens (ou "hyperliens") entre différentes parties de contenu, qu'il s'agisse de texte, d'images, de vidéos ou d'autres types de médias. Ces liens permettent à l'utilisateur de naviguer d'une ressource à une autre de manière non linéaire, en cliquant simplement sur des mots ou des éléments spécifiques.

Le **balisage** est un système de notation utilisé pour structurer, organiser et formater du contenu, notamment des pages web. Il consiste en l'utilisation de **balises**, qui sont des éléments de texte entourés de symboles spécifiques (des chevrons < > en HTML) pour indiquer la fonction ou la signification du contenu qu'elles encadrent. Le balisage ne concerne pas uniquement HTML, il est également utilisé dans des langages comme **XML**, Markdown ou LaTeX pour structurer des données ou des documents.



1 World Wide Web Consortium

Le W3C est une organisation internationale fondée en 1994 par Sir Tim Berners-Lee, l'inventeur du Web. Son but principal est de développer des standards ouverts pour garantir l'évolution et l'accessibilité du Web à tous.

1.1 But du W3C

- Standardisation : Établir des normes ouvertes (standards) pour le Web afin de garantir que les technologies soient interopérables entre les différents navigateurs, appareils et systèmes.
- Accessibilité : Promouvoir un Web accessible à tous, y compris aux personnes en situation de handicap, grâce à des directives comme les WCAG (Web Content Accessibility Guidelines).
- Innovation : Encourager l'innovation en développant de nouvelles technologies pour améliorer les performances, la sécurité et les fonctionnalités du Web.
- Universalité : Maintenir l'universalité du Web en veillant à ce que les contenus et services restent accessibles dans le monde entier, indépendamment de la langue, de la culture ou de l'infrastructure technique.

1.2 Rôle du W3C

- Élaboration de standards : Créer des normes techniques pour les langages comme HTML, CSS, XML, SVG, etc. Développer des protocoles comme HTTP ou des API utilisées pour le développement Web.
- Encourager l'interopérabilité : Garantir que les sites et services Web fonctionnent correctement sur différents navigateurs, systèmes d'exploitation et appareils.
- Promotion de l'accessibilité : Publier des recommandations pour aider les développeurs à rendre les sites Web accessibles à tous les utilisateurs.
- Collaboration avec les parties prenantes : Travailler avec des entreprises, des gouvernements, des organisations à but non lucratif et des développeurs individuels pour définir des standards qui répondent aux besoins de tous.
- Anticipation des évolutions : Étudier les nouvelles tendances et les besoins futurs pour préparer les standards du Web de demain, comme l'intelligence artificielle, le Web sémantique ou les technologies immersives (AR/VR).

2 HTML

2.1 Historique

Version	Année	Principales innovations
HTML	1991-1993	Hypertexte, balises de base (<p>, <a>, <h1>).
HTML 2.0	1995	Standardisation, support des formulaires et tableaux.
HTML 3.2	1997	Styles en ligne, applets Java.
HTML 4.01	1999	Séparation contenu/présentation, internationalisation.
HTML5	2008-2014	Audio/vidéo natifs, nouvelles balises sémantiques, API modernes.
HTML Living Standard	Depuis 2014	Évolution continue, API modernes, compatibilité PWA.

HTML continue d'évoluer pour s'adapter aux besoins des développeurs et des utilisateurs, tout en garantissant la compatibilité avec les versions précédentes.

2.2 Structure d'un page HTML

<!DOCTYPE html>	Indique que le document est écrit en HTML5
<html>	La balise d'ouverture de l'élément HTML
→ ² <head> </head>	Contient les métadonnées et les liens externes comme les feuilles de style ou les scripts.
→ ² <body > </body >	Contient tout le contenu visible de la page web.
</html>	La balise de fermeture de l'élément HTML

²L'**indentation** en programmation et en structuration de documents (comme en HTML ou en LaTeX) est l'ajout d'espaces ou de tabulations au début des lignes de code pour les décaler vers la droite. L'indentation est principalement utilisée pour améliorer la **lisibilité** et l'**organisation** du code, même si elle peut parfois être obligatoire dans certains langages comme Python.

L'ordre des fermetures est important : une balise ouverte en dernier **doit être fermée** en premier.

3 HEAD

L'élément **<head>** en HTML est une section d'un document qui contient des informations "métadonnées" ou des données utiles à la page web, mais qui ne sont pas directement visibles pour l'utilisateur lorsqu'il consulte la page. Cette section se trouve au début du document, juste après la balise d'ouverture <html>, et avant la balise <body>.

3.1 Titre

<title>Titre</title> : Définit le titre de la page qui apparaît dans l'onglet du navigateur.

Exemple : <title>Ma page web</title>



3.2 Métadonnées

Les métadonnées fournissent des informations supplémentaires sur le contenu de la page, mais ne sont généralement pas visibles pour l'utilisateur final. Elles sont souvent utilisées par les moteurs de recherche, les navigateurs et les autres systèmes pour comprendre et traiter correctement la page web.

Exemple : `<meta name="description" content="Un exemple de page HTML.">`

Métadonnée	Description	Exemple
Charset (charset)	Définit le jeu de caractères utilisé dans le document.	<code><meta charset="UTF-8"></code>
Description (description)	Fournit une brève description du contenu de la page, utilisée par les moteurs de recherche.	<code><meta name="description" content="Un tutoriel sur la création d'une page HTML avec des exemples."></code>
Mots-clés (keywords)	Spécifie les mots-clés pertinents pour la page.	<code><meta name="keywords" content="HTML, tutoriel, programmation, balisage"></code>
Auteur (author)	Indique l'auteur de la page.	<code><meta name="author" content="Pierre Cendret"></code>
Viewport	Définit la mise en page sur les appareils mobiles (responsive design).	<code><meta name="viewport" content="width=device-width, initial-scale=1.0"></code>
Robots (robots)	Indique aux moteurs de recherche comment indexer la page et suivre les liens.	<code><meta name="robots" content="index, follow"></code>

3.3 Ressources

<link> : Permet d'inclure des ressources externes, comme des feuilles de style CSS ou des icônes.

Type de lien	Attribut rel	Description	Exemple
Feuille de style	stylesheet	Lier une feuille de style CSS	<code><link rel="stylesheet" href="styles.css"></code>
Icône de favicon	icon	Lier un favicon	<code><link rel="icon" href="favicon.ico" type="image/x-icon"></code>
Polices web	stylesheet	Charger des polices externes	<code><link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto"></code>
Sitemap	sitemap	Lier un fichier sitemap XML	<code><link rel="sitemap" href="/sitemap.xml"></code>
Flux RSS ou Atom	alternate	Lier un flux RSS ou Atom	<code><link rel="alternate" type="application/rss+xml" href="feed.rss"></code>

3.4 JavaScript

La balise `<script>` en HTML est utilisée pour inclure du code JavaScript dans un document web. JavaScript permet d'ajouter de l'interactivité, de la dynamique, et des fonctionnalités avancées à une page web. Il existe plusieurs manières d'utiliser la balise `<script>` pour intégrer du JavaScript, que ce soit directement dans le fichier HTML ou via des fichiers externes. La balise `<script>` peut être placée soit dans le `<head>`, soit dans le `<body>` du document HTML.

3.4.1 Script interne (inline) :

Le code JavaScript peut être directement écrit entre les balises `<script>...</script>` dans le fichier HTML. Ce type de script est dit "interne" ou "inline".

Exemple :

```
<script>
```

```
  console.log("Ceci est un script interne !");
```

```
  alert("Bienvenue sur la page !");
```

```
</script>
```

3.4.2 Script externe :

Vous pouvez lier un fichier JavaScript externe à la page web. Le fichier externe contient le code JavaScript séparé du HTML, ce qui rend le code plus propre et plus facile à maintenir.

Exemple : `<script src="script.js"></script>`

4 BODY

4.1 Principales balises

Balise	Description	Exemple
<code><h1></code> à <code><h6></code>	Définit des titres, du plus important (<code><h1></code>) au moins important (<code><h6></code>).	<code><h1>Titre principal</h1></code>
<code><p></code>	Définit un paragraphe de texte.	<code><p>Ceci est un paragraphe.</p></code>
<code><a></code>	Crée un lien hypertexte vers une autre page ou ressource.	<code>Visiter Exemple</code>
<code></code>	Insère une image dans la page.	<code></code>
<code></code> , <code></code> , <code></code>	Crée des listes non ordonnées (<code></code>) ou ordonnées (<code></code>), avec des éléments de liste (<code></code>).	<code>Élément 1Élément 2</code>
<code><nav></code>	Représente une section de navigation contenant des liens vers d'autres pages ou sections du site.	<code><nav>AccueilÀ propos</nav></code>
<code><div></code>	Crée un conteneur pour structurer et regrouper des éléments.	<code><div class="conteneur">Contenu ici</div></code>
<code></code>	Conteneur en ligne pour appliquer du style ou des scripts à une partie de texte.	<code>Texte en rouge</code>
<code><form></code>	Crée un formulaire pour la saisie de données.	<code><form action="/submit" method="post"><input type="text" name="nom"><button>Envoyer</button></form></code>
<code><input></code>	Définit un champ de saisie dans un formulaire (texte, bouton radio, case à cocher, etc.).	<code><input type="text" name="nom"></code>
<code><button></code>	Crée un bouton cliquable pour soumettre des formulaires ou exécuter des scripts.	<code><button type="submit">Envoyer</button></code>
<code><table></code> , <code><tr></code> , <code><td></code> , <code><th></code>	Crée des tableaux pour organiser des données en lignes (<code><tr></code>) et colonnes (<code><td></code> , <code><th></code> pour l'en-tête).	<code><table><tr><th>Titre</th></tr><tr><td>Données</td></tr></table></code>
<code><section></code>	Définit une section thématique du document pour structurer le contenu.	<code><section><h2>Titre de la section</h2><p>Contenu</p></section></code>
<code><article></code>	Représente un contenu autonome, comme un article de blog ou un post.	<code><article><h2>Titre de l'article</h2><p>Contenu de l'article</p></article></code>
<code><header></code>	Définit l'en-tête d'une page ou d'une section, souvent utilisé pour les titres et les barres de navigation.	<code><header><h1>Mon site web</h1></header></code>
<code><footer></code>	Définit le pied de page d'une section ou d'une page.	<code><footer><p>Copyright 2024</p></footer></code>
<code><audio></code>	Insère un fichier audio avec des contrôles de lecture (play, pause, etc.).	<code><audio controls><source src="/dossier/audio.mp3" type="audio/mp3"></audio></code>
<code><video></code>	Intègre un fichier vidéo avec des contrôles.	<code><video controls><source src="/dossier/video.mp4" type="video/mp4"></video></code>

4.2 Attributs

Dans les balises HTML utilisées à l'intérieur de l'élément `<body>`, de nombreux attributs peuvent être appliqués pour ajouter des fonctionnalités, des styles, ou des interactions avec JavaScript.

4.2.1 Attributs généraux

Ces attributs peuvent être utilisés avec la plupart des balises HTML.

Attribut	Description	Exemple
id	Un identifiant unique pour un élément, souvent utilisé pour le cibler avec CSS ou JavaScript.	<code><div id="contenu">Texte ici</div></code>
class	Définit une ou plusieurs classes CSS pour appliquer des styles communs. Le terme class est utilisé pour regrouper des éléments qui partagent des caractéristiques similaires.	<code><p class="texte-important">Texte important</p></code>
style	Permet d'ajouter des styles CSS en ligne directement à l'élément.	<code><h1 style="color: red;">Titre en rouge</h1></code>
title	Fournit une information supplémentaire affichée comme infobulle au survol.	<code></code>
hidden	Masque un élément visuellement et le retire du flux de la page sans le supprimer.	<code><p hidden>Ce paragraphe est caché.</p></code>
tabindex	Détermine l'ordre de tabulation des éléments lorsqu'un utilisateur navigue avec la touche Tab.	<code><button tabindex="1">Bouton 1</button><button tabindex="2">Bouton 2</button></code>

4.2.2 Attributs pour les images (``)

Attribut	Description	Exemple
src	Définit l'URL de la source de l'image.	<code></code>
alt	Texte alternatif pour les images, affiché si l'image ne se charge pas.	<code></code>
width	Spécifie la largeur de l'image en pixels ou en pourcentage.	<code></code>
height	Spécifie la hauteur de l'image en pixels ou en pourcentage.	<code></code>
loading	Définit si l'image doit être chargée immédiatement ou différée (lazy loading).	<code></code>

4.2.3 Attributs pour les liens (`<a>`)

Attribut	Description	Exemple
href	Définit l'URL ou le chemin vers lequel le lien pointe.	<code>Visiter Exemple</code>
target	Définit où le lien doit être ouvert : <code>_self</code> (par défaut) ou <code>_blank</code> pour une nouvelle fenêtre/onglet.	<code>Ouvrir dans un nouvel onglet</code>
rel	Spécifie la relation entre la page actuelle et la page liée (souvent utilisé pour les liens externes avec <code>rel="noopener noreferrer"</code>).	<code>Externe</code>
download	Indique que le lien pointe vers un fichier téléchargeable.	<code>Télécharger le PDF</code>

4.2.4 Attributs pour les formulaires (<form>, <input>, etc.)

Attribut	Description	Exemple
action	URL où les données du formulaire seront envoyées lors de la soumission.	<form action="/submit" method="post">
method	Méthode de soumission du formulaire : GET ou POST.	<form action="/submit" method="post">
name	Nom de l'élément de formulaire pour identifier les données envoyées.	<input type="text" name="username">
type	Spécifie le type de l'élément <input> (texte, mot de passe, email, etc.).	<input type="email" name="email">
placeholder	Texte indicatif qui s'affiche dans le champ avant la saisie de l'utilisateur.	<input type="text" placeholder="Entrez votre nom">
required	Indique que le champ est obligatoire pour valider le formulaire.	<input type="text" name="username" required>
value	Définit une valeur initiale pour un champ <input>.	<input type="text" value="Valeur par défaut">
checked	Définit si une case à cocher ou un bouton radio est coché par défaut.	<input type="checkbox" checked>

4.2.5 Attributs pour les vidéos et audios (<video>, <audio>)

Attribut	Description	Exemple
src	Définit la source de la vidéo ou de l'audio.	<video src="video.mp4" controls></video>
controls	Ajoute les contrôles de lecture (play, pause, volume) pour l'utilisateur.	<audio src="audio.mp3" controls></audio>
autoplay	Joue automatiquement la vidéo ou l'audio dès le chargement de la page (souvent désactivé par défaut pour les vidéos).	<video src="video.mp4" autoplay></video>
loop	Joue en boucle (recommence automatiquement après la fin).	<audio src="audio.mp3" loop></audio>
muted	Démarre la lecture avec le son coupé.	<video src="video.mp4" muted></video>
poster	Spécifie une image à afficher avant le début de la lecture de la vidéo.	<video src="video.mp4" poster="poster.jpg"></video>

4.2.6 Attributs pour les tableaux (<table>, <tr>, <td>, etc.)

Attribut	Description	Exemple
colspan	Définit combien de colonnes une cellule <td> ou <th> doit occuper.	<td colspan="2">Cellule fusionnée</td>
rowspan	Définit combien de lignes une cellule <td> ou <th> doit occuper.	<td rowspan="2">Cellule fusionnée sur deux lignes</td>
scope	Spécifie si une cellule d'en-tête (<th>) est pour une ligne ou une colonne.	<th scope="col">En-tête de colonne</th>

4.2.7 Attributs pour l'accessibilité (ARIA)

Les attributs ARIA (**Accessible Rich Internet Applications**) sont utilisés pour améliorer l'accessibilité des pages web, en particulier pour les utilisateurs utilisant des technologies d'assistance.

Attribut ARIA	Description	Exemple
aria-label	Fournit un label accessible pour les éléments interactifs (boutons, liens, etc.).	<code><button aria-label="Fermer le menu">X</button></code>
aria-hidden	Indique qu'un élément est caché aux lecteurs d'écran.	<code><div aria-hidden="true">Contenu caché pour les lecteurs d'écran</div></code>
role	Définit explicitement le rôle d'un élément, comme button, navigation, etc.	<code><div role="navigation">Menu de navigation</div></code>

5 CSS

Le **CSS** (Cascading Style Sheets, ou feuilles de style en cascade) est un langage utilisé pour définir l'apparence et la mise en page des éléments HTML sur une page web. Il permet de séparer la structure (HTML) de la présentation (CSS), offrant ainsi une flexibilité et une simplicité accrues dans la conception des sites web.

5.1 Structure de base d'une règle CSS

Une règle CSS se compose de deux parties principales : **le sélecteur** et **les déclarations**. Les déclarations elles-mêmes sont constituées de propriétés et de valeurs.

```
Sélecteur {
  propriété: valeur;
  propriété: valeur;
}
```

- **Sélecteur** : Il spécifie quel élément HTML la règle va cibler (par exemple, `p`, `.class`, `#id`, etc.).
- **Propriété** : Ce que vous voulez modifier, comme la couleur, la taille de la police, la marge, etc.
- **Valeur** : La valeur que vous souhaitez attribuer à cette propriété (par exemple, `red`, `16px`, etc.).

5.2 Comment intégrer du CSS

1. **CSS en ligne (Inline CSS)** : Les styles sont appliqués directement dans l'élément HTML via l'attribut `style`. Ce n'est pas recommandé pour les grands projets, car cela peut rendre le code difficile à maintenir.
2. **CSS dans le <head> du document (Interne)** : Le CSS est défini dans une balise `<style>` au sein de la section `<head>` du document HTML.
3. **Fichier CSS externe** : Le CSS est placé dans un fichier séparé (généralement avec l'extension `.css`), et lié à la page HTML à l'aide de la balise `<link>` dans le `<head>`.

Html	Exemple de contenu styles.css
<pre><head> <link rel="stylesheet" href="./dossier/styles.css"> </head></pre>	<pre>p { color: blue; }</pre>

5.3 Sélecteurs CSS

Les **sélecteurs** sont des éléments clés du CSS, car ils définissent les éléments HTML sur lesquels les styles doivent être appliqués.

- **Sélecteurs de type** : Ils ciblent des éléments HTML par leur nom.
`H1 { color: green; }`
- **Sélecteurs de classe (.)** : Ils ciblent des éléments ayant une classe spécifique, définie par l'attribut `class` dans HTML.
`.contenu { background-color: yellow; }`
- **Sélecteurs d'identifiant (#)** : Ils ciblent des éléments avec un identifiant unique défini par l'attribut `id`.
`#header { font-size: 24px; }`
- **Sélecteurs universels (*)** : Ils ciblent tous les éléments de la page
`* { margin: 0; padding: 0; }`
- **Sélecteurs d'attributs** : Ils ciblent des éléments en fonction de leurs attributs.
`input[type="text"] { border: 1px solid black; }`

5.4 Propriétés CSS communes

5.4.1 Couleur et typographie :

- **color** : Définit la couleur du texte.
`h1 { color: green; }`
- **font-size** : Définit la taille de la police.
- **font-family** : Définit la police utilisée.
- **text-align** : Définit l'alignement du texte (gauche, droite, centré).
 - **text-align: right** (Alignement à droite)
 - **text-align: left** (Alignement à gauche)
 - **text-align: center** (Alignement centré)
 - **text-align: justify** (Justification)

5.4.2 Background (fond) :

- **background-color** : Définit la couleur d'arrière-plan d'un élément.
`body { background-color: lightblue; }`
- **background-image** : Définit une image d'arrière-plan.
- **background-repeat** : Contrôle si l'image de fond doit se répéter (par défaut, l'image se répète dans les deux directions si elle est plus petite que l'élément).
 - **repeat** (valeur par défaut) : L'image se répète dans les deux directions (horizontalement et verticalement).
 - **no-repeat** : L'image ne se répète pas.
 - **repeat-x** : L'image se répète seulement horizontalement.
 - **repeat-y** : L'image se répète seulement verticalement.
- **background-position** : Définit la position de l'image de fond dans l'élément.
Valeurs courantes : `left`, `right`, `top`, `bottom`, `center`, ou des coordonnées spécifiques en pixels ou en pourcentages.

- **background-size** : Contrôle la taille de l'image de fond.
 - **auto** (valeur par défaut) : Garde la taille d'origine de l'image.
 - **cover** : Redimensionne l'image pour couvrir tout l'élément, en maintenant les proportions de l'image (l'image peut être recadrée pour s'adapter).
 - **contain** : Redimensionne l'image pour s'adapter à l'intérieur de l'élément tout en conservant les proportions (l'image peut ne pas remplir tout l'élément).
 - **Valeurs spécifiques** : Vous pouvez aussi définir des dimensions en pixels (px) ou en pourcentages (%).
- **background-attachment** : Contrôle si l'image de fond se déplace avec la page lors du défilement (scroll).
 - **scroll** (valeur par défaut) : L'image se déplace avec le reste du contenu lorsque l'utilisateur fait défiler la page.
 - **fixed** : L'image reste fixe, ne défilant pas avec le contenu.
 - **local** : L'image de fond défile avec le contenu de l'élément auquel elle est appliquée (utile pour les éléments avec overflow).

Exemple :

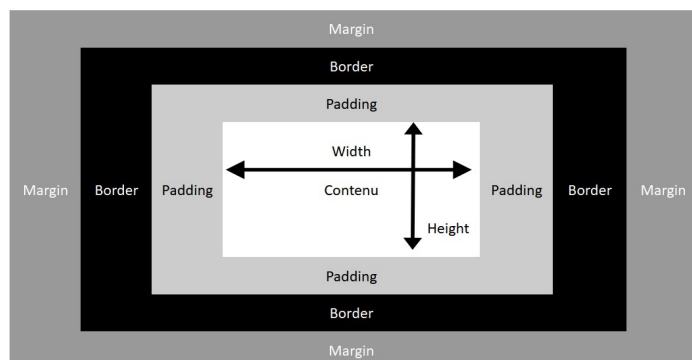
```
div {
  width: 300px;
  height: 200px;
  background-image: url('https://example.com/image.jpg');
  background-repeat: no-repeat;
  background-position: center center;
  background-size: cover;
  background-attachment: fixed;
  border: 1px solid black;
}
```

5.4.3 Mise en page et dimensions :

- **width** et **height** : Définit la largeur et la hauteur d'un élément.
- **margin** : Définit la marge externe autour de l'élément.
- **padding** : Définit l'espace entre le contenu de l'élément et ses bordures.
- **border** : Définit la bordure autour de l'élément.

Exemple :

```
div {
  width: 200px;
  padding: 10px;
  margin: 10px 12px 13px 14px;
  border: 1px solid black;
}
```



Lorsque vous utilisez une propriété comme

margin avec quatre valeurs (comme dans

`margin: 10px 12px 13px 14px;`), chaque valeur représente l'espace extérieur appliqué à un côté spécifique de l'élément HTML. L'ordre des valeurs suit une logique dans le sens des aiguilles d'une montre :

- **10px** : La marge en haut de l'élément.
- **12px** : La marge à droite de l'élément.
- **13px** : La marge en bas de l'élément.
- **14px** : La marge à gauche de l'élément.

5.4.4 Affichage et positionnement :

- **display** : Définit la manière dont un élément doit être affiché (`block`, `inline`, `flex`, etc.).

Type de display	Description	Comportement
block	Élément de bloc.	Occupe toute la largeur, commence sur une nouvelle ligne.
inline	Élément en ligne.	Ne commence pas sur une nouvelle ligne, occupe juste la place du contenu.
inline-block	Bloc en ligne.	Se comporte comme un élément en ligne, mais accepte la largeur et la hauteur.
none	Retire l'élément du flux du document.	L'élément n'apparaît pas et ne prend pas de place.
flex	Active le modèle de boîte flex pour des dispositions flexibles.	Dispose les éléments enfants de manière flexible (ligne ou colonne).
grid	Active le modèle de mise en page par grille pour des dispositions en lignes et colonnes.	Dispose les éléments enfants dans une grille bidimensionnelle.
inline-flex	Comme flex, mais l'élément parent reste en ligne.	Dispose les éléments enfants de manière flexible, mais l'élément lui-même reste en ligne.

- **Justify-content** : Elle contrôle la distribution de l'espace libre entre les éléments enfants. Elle fonctionne uniquement si le conteneur a une propriété `display: flex` ou `display: grid`.

Valeur	Description	Comportement
flex-start	Aligne les éléments au début de l'axe principal.	Tous les éléments sont regroupés au début du conteneur, sans espace supplémentaire.
flex-end	Aligne les éléments à la fin de l'axe principal.	Tous les éléments sont regroupés à la fin du conteneur, sans espace supplémentaire.
center	Centre les éléments le long de l'axe principal.	Les éléments sont centrés avec un espace égal de chaque côté, si l'espace est disponible.
space-between	Distribution avec espace entre les éléments.	Le premier élément est collé au début, le dernier est collé à la fin, et un espace égal est distribué entre les éléments restants.
space-around	Distribution avec espace autour de chaque élément.	Chaque élément a un espace égal autour de lui . L'espace entre les éléments est deux fois plus grand que l'espace entre un élément et le bord du conteneur.
space-evenly	Distribution avec espace uniforme.	Un espace égal est appliqué entre chaque élément et les bords du conteneur.
start	Aligne au début de la ligne (dans la direction écrite).	Semblable à flex-start, mais respecte la direction d'écriture (ltr ou rtl).
end	Aligne à la fin de la ligne (dans la direction écrite).	Semblable à flex-end, mais respecte la direction d'écriture.
left	Aligne les éléments à gauche du conteneur.	Fonctionne dans un conteneur où l'axe principal est horizontal (indépendamment de la direction écrite).
right	Aligne les éléments à droite du conteneur.	Fonctionne dans un conteneur où l'axe principal est horizontal (indépendamment de la direction écrite).
stretch	Étire les éléments pour occuper tout l'espace disponible.	Les éléments s'étendent pour remplir l'espace disponible si leurs tailles ne sont pas définies (valeur par défaut dans certains contextes de grid).

- **Align-items** : Elle agit sur l'alignement vertical lorsque l'axe principal est horizontal (par défaut avec flex-direction: row). Elle fonctionne uniquement si le conteneur a une propriété display: flex ou display: grid.

Valeur	Description	Comportement
stretch	Étire les éléments pour occuper tout l'espace disponible (par défaut).	Les éléments s'étirent pour remplir la hauteur (ou largeur si l'axe principal est vertical), sauf si une taille explicite est définie.
flex-start	Aligne les éléments au début de l'axe transversal.	Les éléments sont alignés au bord supérieur (ou gauche si l'axe principal est vertical) du conteneur.
flex-end	Aligne les éléments à la fin de l'axe transversal.	Les éléments sont alignés au bord inférieur (ou droit si l'axe principal est vertical) du conteneur.
center	Centre les éléments sur l'axe transversal.	Les éléments sont alignés verticalement (ou horizontalement selon l'axe transversal), avec un espace égal de chaque côté.
baseline	Aligne les éléments sur leur ligne de base de texte.	Les éléments sont alignés de façon à ce que leurs lignes de base de texte soient au même niveau.
start	Aligne les éléments au début de l'axe d'écriture.	Similaire à flex-start, mais respecte la direction d'écriture (ltr ou rtl).
end	Aligne les éléments à la fin de l'axe d'écriture.	Similaire à flex-end, mais respecte la direction d'écriture (ltr ou rtl).
self-start	Aligne chaque élément au début de sa propre boîte parent.	Les éléments suivent leur propre alignement selon leur conteneur, respectant les marges et dimensions.
self-end	Aligne chaque élément à la fin de sa propre boîte parent.	Les éléments suivent leur propre alignement selon leur conteneur, respectant les marges et dimensions.

Exemple :

```
div.container {
  display: flex; /* Active Flexbox */
  justify-content: space-around; /* Distribution des éléments */
  align-items: center; /* Aligne verticalement si besoin */
}
```

- **position** : Contrôle la position de l'élément (relative, absolute, fixed).

Valeur	Description	Caractéristiques
static	Position par défaut dans le flux normal du document.	- Ne répond pas aux propriétés top, left, right, bottom. - Les éléments s'affichent selon l'ordre normal du document.
relative	Positionné par rapport à sa position normale dans le document.	- Peut être déplacé à l'aide de top, left, right, ou bottom. - L'espace initial de l'élément reste réservé dans le document.
absolute	Retiré du flux normal et positionné par rapport à son ancêtre positionné le plus proche, ou la fenêtre.	- Se place par rapport à son conteneur parent le plus proche ayant une position autre que static. - Les autres éléments se comportent comme s'il n'existait pas.
fixed	Retiré du flux normal et fixé par rapport à la fenêtre du navigateur, même lorsque la page défile.	- L'élément reste à une position fixe même en faisant défiler la page. - Utile pour des éléments comme des barres de navigation fixes ou des boutons de retour en haut de page.
sticky	Mélange de relative et fixed. L'élément est relatif jusqu'à atteindre une certaine position, puis se fixe.	- Se comporte comme relative jusqu'à ce que l'élément atteigne la position spécifiée, puis devient fixe (collant). - Utile pour des éléments qui doivent rester visibles lors du défilement.

- **z-index** : Définit l'ordre de superposition des éléments.

Exemple :

```
.box {
  display: flex;
  position: absolute;
  top: 50px;
  left: 100px;
}
```

5.4.5 hover

Le pseudo-classe `:hover` en CSS permet de définir un style spécifique qui s'applique à un élément lorsqu'un utilisateur place son curseur de souris dessus (le survole). C'est une manière simple d'ajouter de l'interactivité et d'améliorer l'expérience utilisateur.

Exemple : Changer la couleur d'un lien lors du survol

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

Lorsqu'un utilisateur survole un lien (`<a>`), sa couleur devient rouge.

Exemple d'effet visuel sur une image

Html	Exemple de contenu styles.css
<pre><div class="interactive-img"> </div></pre>	<pre>.interactive-img { position: relative; display: inline-block; cursor: pointer; } .interactive-img img { max-width: 100%; transition: transform 0.3s ease, opacity 0.3s ease; } .interactive-img img:hover { transform: scale(1.1); opacity: 0.8; }</pre>

`cursor: pointer;`

Change le curseur de la souris en une main cliquable lorsque l'utilisateur survole l'élément. Cela indique que l'élément est interactif (par exemple, un lien ou un bouton cliquable).

6 Introduction aux Formulaires en HTML

Un formulaire HTML est un moyen d'interagir avec les utilisateurs en leur permettant de saisir des données et de les envoyer à un serveur pour traitement. Les formulaires sont très courants dans les sites web, que ce soit pour s'inscrire, se connecter, rechercher, etc.

6.1 Structure de base d'un formulaire

Un formulaire HTML est défini à l'aide de la balise `<form>`. Cette balise regroupe plusieurs types de champs et boutons.

```
<form action="url_de_traitement" method="POST">
  <!-- Différents champs du formulaire -->
</form>
```

- **action** : L'URL du serveur où les données du formulaire seront envoyées.
- **method** : Détermine comment les données sont envoyées (généralement GET ou POST).

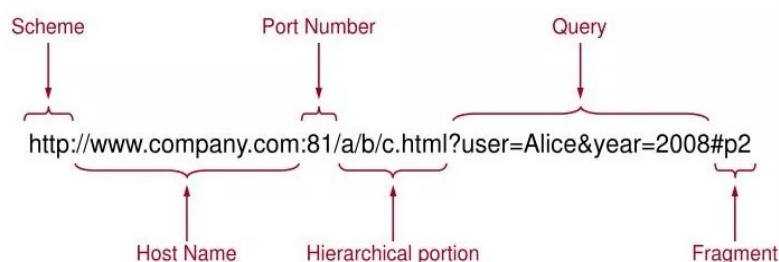
6.2 Champs de saisie courants

Type de champ	Description	Exemple HTML
Champ texte (text)	Permet la saisie de texte simple (comme un nom ou un prénom).	<code><input type="text" name="nom"></code>
Champ mot de passe (password)	Permet la saisie d'un mot de passe, le texte est masqué par des astérisques ou des points.	<code><input type="password" name="motdepasse"></code>
Champ email (email)	Champ dédié à la saisie d'adresses email, valide le format email (exemple@domaine.com).	<code><input type="email" name="email"></code>
Boutons radio (radio)	Permet de sélectionner une seule option parmi plusieurs (exclusif).	<code><input type="radio" name="sexe" value="Homme"> Homme
<input type="radio" name="sexe" value="Femme"> Femme</code>
Cases à cocher (checkbox)	Permet de sélectionner plusieurs options (non exclusif).	<code><input type="checkbox" name="langue" value="Français"> Français
<input type="checkbox" name="langue" value="Anglais"> Anglais</code>
Liste déroulante (select)	Permet de choisir une option parmi plusieurs dans une liste déroulante.	<code><select name="pays"><option value="France">France</option><option value="Canada">Canada</option></select></code>
Zone de texte (textarea)	Permet de saisir du texte long (comme un commentaire ou un message).	<code><textarea name="commentaire" rows="4" cols="50"></textarea></code>
Bouton de soumission (submit)	Envoie le formulaire lorsque l'utilisateur clique dessus.	<code><input type="submit" value="Envoyer"></code>
Champ date (date)	Permet de sélectionner une date (avec un calendrier déroulant).	<code><input type="date" name="date_naissance"></code>
Champ nombre (number)	Permet de saisir des nombres avec la possibilité de définir des bornes minimales et maximales.	<code><input type="number" name="age" min="1" max="100"></code>
Champ couleur (color)	Permet de choisir une couleur via un sélecteur de couleurs.	<code><input type="color" name="couleur_favorie"></code>
Champ fichier (file)	Permet à l'utilisateur de sélectionner un fichier sur son appareil.	<code><input type="file" name="fichier"></code>

7 URL (Uniform Resource Locator)

Une URL (Uniform Resource Locator), ou « localisateur uniforme de ressource » en français, est une adresse web qui permet d'accéder à une ressource spécifique sur Internet, telle qu'une page web, une image ou un document. Elle indique l'emplacement de la ressource et le protocole à utiliser pour la récupérer.

Uniform Resource Locators (URLs)



7.1 Structure

La structure d'une URL se compose généralement des éléments suivants :

- Le schéma (Scheme ou protocole) : Il spécifie le protocole de communication à utiliser, tel que HTTP ou HTTPS pour les pages web, FTP pour le transfert de fichiers, ou mailto pour les adresses e-mail. Par exemple, `https://` indique que le protocole HTTPS sera utilisé.
- Le sous-domaine : Par exemple, `www`, `snt` ou d'autres comme `blog` ou `support`.
- Le nom de domaine : Le nom unique identifiant le site, comme `hexalys.net`.
- Le port (facultatif) : Spécifie le port de communication, souvent omis si le port par défaut est utilisé (80 pour HTTP, 443 pour HTTPS).
- Le chemin d'accès : Il indique le répertoire ou le fichier spécifique sur le serveur, par exemple `index.html`.
- Les paramètres de requête (facultatifs) : Introduits par un `?`, ils fournissent des informations supplémentaires sous forme de paires clé-valeur séparées par le symbole égal (`=`), par exemple `?id=123`. Lorsqu'il y a plusieurs paramètres, ils sont séparés par une esperluette `&` (et).
- L'ancre (facultative) : Introduite par un `#`, elle pointe vers une section spécifique de la page, par exemple `#section2`.

Par exemple, dans l'URL `https://www.example.com:8080/dossier/page.html?id=123#section2` :

- `https` est le schéma.
- `www` est le sous-domaine.
- `example.com` est le nom de domaine.
- `8080` est le port.
- `/dossier/page.html` est le chemin d'accès.
- `id=123` est le paramètre de requête.
- `section2` est l'ancre.

8 HyperText Transfer Protocol

Le HyperText Transfer Protocol (HTTP) est un protocole de communication utilisé pour le transfert d'informations entre les clients (navigateurs Web ou autres applications) et les serveurs sur le World Wide Web.

8.1 HTTPS

HTTPS est la version sécurisée de HTTP :

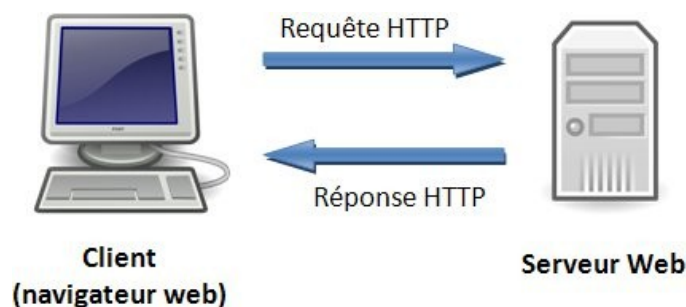
- Utilise le protocole **SSL/TLS** pour chiffrer les données échangées.
- Protège contre les interceptions et les attaques.

8.2 Protocole sans état :

HTTP est un protocole sans état (stateless), ce qui signifie qu'il ne conserve pas d'informations entre deux requêtes. Chaque requête est indépendante.

8.3 Architecture client-serveur :

- Un client (souvent un navigateur) envoie une requête HTTP à un serveur.
- Le serveur répond avec des données ou des messages d'erreur.
- Les requêtes et réponses HTTP sont sous forme de texte, faciles à lire et comprendre.



8.4 Requête HTTP :

Une requête HTTP contient plusieurs éléments :

- Méthode : Action demandée (GET, POST, etc.).
- URL : Ressource demandée.
- Entêtes : Métadonnées (type de contenu, cookies, etc.).
- Corps (facultatif) : Données supplémentaires (par exemple pour un formulaire).

Exemple de requête :

<pre>POST /api/login HTTP/1.1 Host: www.example.com Content-Type: application/json Content-Length: 49 { "username": "Pierre", "password": "123456" }</pre>	<p>POST : Méthode utilisée pour envoyer des données. /api/login : Chemin de la ressource sur le serveur. HTTP/1.1 : Version du protocole HTTP.</p> <p>Host : Indique le nom de domaine de la ressource. Content-Type : Spécifie le type des données dans le corps (ici, du JSON). Content-Length : Longueur des données envoyées (en octets).</p> <p>Contient les données envoyées au serveur, ici en format JSON</p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

8.5 Réponse HTTP :

Une réponse HTTP contient :

- Code d'état : Indique le statut (200, 404, 500, etc.).
- Entêtes : Métadonnées (type de contenu, longueur, etc.).
- Corps : Les données demandées (par exemple, une page HTML).

Exemple de réponse :

<pre>HTTP/1.1 200 OK Content-Type: text/html Content-Length: 1234 <html> <body>Page Web...</body> </html></pre>	<p>HTTP/1.1 : Indique la version du protocole HTTP utilisée. 200 : Le code d'état HTTP, signifiant que la requête a été traitée avec succès. OK : Message descriptif pour le code d'état.</p> <p>Indique le type MIME¹ du contenu renvoyé. Ici, text/html signifie que le corps de la réponse contient du HTML.</p> <p>Indique la taille du corps de la réponse en octets. Ici, le serveur annonce que le contenu (le HTML) fait 1234 octets.</p> <p>Le corps contient les données que le serveur envoie au client, ici une page HTML</p>
---------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1 : MIME signifie Multipurpose Internet Mail Extensions. C'est une norme Internet qui décrit le type de contenu d'un fichier ou d'une ressource, utilisée principalement pour indiquer comment les données doivent être interprétées par les navigateurs Web, les clients de messagerie ou d'autres applications.