Programmation Web

Le language HTML

Pierre Blarre - Florian Rodriguez

Sujets abordés pendant ce cours

- Introduction au HTML
- Structure et balises du HTML
- Création d'un projet HTML dans VSCode
- DevTools
- DOM et code source
- Validateur W3C
- Exercices supplémentaire

Le HTML

- Le **HyperText Markup Language**, généralement abrégé **HTML** ou, dans sa dernière version, **HTML5**, est le langage de balisage conçu pour représenter les pages web.
- Ce langage permet:
  - d'écrire de l'hypertexte (d'où son nom),
  - de structurer sémantiquement une page web,
  - de mettre en forme du contenu,
  - de créer des formulaires de saisie ou encore d'inclure des ressources multimédias dont des images, des vidéos, et des programmes informatiques.
- https://fr.wikipedia.org/wiki/Hypertext_Markup_Language

Le HTML

- Le HTML se présente sous forme de **balises**, majoritairement avec une balise ouvrante et une balise fermante
- Les balises HTML sont composées :
  - des symboles inférieur `<` et supérieur `>`
  - du **nom** de la balise
  - du symbole `/` pour signifier la fin de la balise
  - **d'attributs**, qui varient en fonction des balises et des besoins, sous la forme `nom="valeur"`
- Exemples :

```
<p>Je suis un paragraphe</p>
<a href="https://google.com">Je suis un lien hypertexte</a>
<img src="https://placehold.co/600x400" alt="Texte alternatif si l'image ne peut être affichée" />
```

Copy

Le HTML

- Le HTML est utilisé conjointement avec le langage de programmation JavaScript et des feuilles de style en cascade (CSS)
- Le HTML permet de définir la sémantique du contenu d'une page web
  - Titres, paragraphes, section, articles
- Le HTML permet de diffuser des contenus multimédias et de créer des liens vers d'autres documents (HTML ou autre)
  - Images, vidéos, audio, documents, liens
- Le HTML utilise majoritairement l'UTF-8 d'Unicode, mais a aussi défini des entités pour représenter certains caractères non ASCII:
  - Par exemple, on peut utiliser *Á* pour afficher Á (A avec accent aigu)

Le HTML - Structure d'un document HTML

index.html | styles.css | scripts.js

EXPLORER

HTML (WORKSPACE)

- html
  - index.html
  - scripts.js
  - styles.css

html > index.html > html > body

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Titre du document (affiché dans l'onglet du navigateur)</title>

    <!-- Je suis un commentaire -->
    <link rel="stylesheet" href="./styles.css"> <!-- J'inclus un fichier CSS -->
</head>
<body>

    <h1>Je suis un titre principal</h1>

    <p>Je suis un paragraphe</p>

    <img src="https://placehold.co/600x400" alt="Je suis une image" />

    <br><!--·Je·suis·un·saut·de·ligne·-->

    <!-- J'inclus un fichier Javascript -->
    <script src="./scripts.js"></script>

</body>
</html>
```

Source HTML et structure du projet

Résultat dans le navigateur

Voici un exemple simple d'un document HTML.

Stocké sur un serveur, il sera reçu et interprété par le client (un navigateur)

Imbrication d'élément

On peut placer des éléments à l'intérieur d'autres éléments : c'est ce qu'on appelle **l'imbrication**.

```
<p>Mon chat est <strong>très</strong> grincheux.</p>
```

Copy

!! Les éléments doivent être correctement imbriqués. !!

Le code suivant est incorrect:

```
<p>Mon chat est <strong>très grincheux.</p></strong>
```

Copy

Éléments vides

Certains éléments n'ont pas de contenu, on les qualifie d'éléments vides.

```
<img src="images/firefox-icon.png" alt="Mon image de test" />
```

Copy

Cet élément contient deux attributs (src et alt) mais n'a pas de contenu et il n'y a pas de balise fermante. En effet, un élément d'image n'encadre pas du contenu pour avoir un effet sur celui-ci. Son but est d'intégrer une image dans un document HTML à l'endroit où il est placé.

Anatomie d'un document html

Comment sont assemblés les éléments pour former une page HTML complète?

Copy

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Titre du document (affiché dans l'onglet du navigateur)</title>
    <!-- Je suis un commentaire -->
    <link href="style.css" rel="stylesheet"> <!-- J'inclus un fichier CSS -->
  </head>
  <body>

    <h1>Je suis le titre principal</h1>

    <p>Je suis un paragraphe</p>

    <img alt="Je suis une image" src="https://placehold.co/600x400">

    <br><!-- Je suis un saut de ligne -->

    <!-- J'inclus un fichier Javascript -->

    <script src="scripts.js"></script>
  </body>
</html>
```

Voici ce qu'on y trouve =>

Anatomie d'un document html

- `<!DOCTYPE html>`

  Le « doctype ». Il s'agit d'un préambule obligatoire. Aux débuts de HTML (en 1991/1992), les doctypes servaient de liens vers des ensembles de règles qu'une page HTML devait suivre pour être considérée valides (avec des fonctionnalités de vérifications d'erreur et autres). Aujourd'hui, ils ne sont plus utilisés ainsi et ce marqueur sert uniquement au bon comportement du document. C'est tout ce qu'il y a à savoir là-dessus pour le moment.
- `<html></html>`

  L'élément <html> est celui qui contient tout le reste de la page.
  On l'appelle aussi l'**élément racine**.
- `<head></head>`

  L'élément <head> sert de conteneur pour tout ce qu'on veut inclure dans une page HTML qui n'est pas du contenu à afficher à l'écran. Cela inclut:
  les **mots-clés** et une **description** de la page destinée aux moteurs de recherches,
  les **liens vers le CSS** qui mettra en forme notre contenu, les déclarations pour les **jeux de caractères utilisés**, etc.

Anatomie d'un document html

- `<meta charset="utf-8">`

  Cet élément définit le jeu de caractères utilisé pour le document, ici UTF-8 (qui inclut la plupart des caractères pour un grand nombre de langues écrites). Ce jeu de caractères permettra de gérer n'importe quel contenu textuel placé dans le document. *(L'UTF-8 est utilisé par de 95,2 % des sites web en octobre 2020)*
- `<meta name="viewport" content="width=device-width">`

  Cet élément viewport permet que la page soit affichée avec la bonne largeur par rapport à la zone d'affichage, empêchant les navigateurs sur mobile d'afficher la page plus largement que la zone d'affichage avant de la réduire. (Doc mdn)

Anatomie d'un document html

- `<title></title>`

  L'élément <title> définit le titre de la page, qui apparaîtra dans l'onglet sur lequel la page est chargée. Il sert aussi à décrire la page lorsqu'on l'ajoute aux favoris ou qu'elle est listée dans les résultats d'un moteur de recherche.
- `<body></body>`

  L'élément <body> contient tout le contenu qu'on veut afficher aux utilisatrices et utilisateurs web lorsqu'ils visitent la page, que ce soit du texte, des vidéos, des jeux, des pistes audio, etc.

Baliser du texte

- Titres

```
<!-- 4 niveaux de titres : -->
<h1>Mon titre principal</h1>
<h2>Mon titre de deuxième niveau</h2>
<h3>Mon sous-titre</h3>
<h4>Mon sous-sous-titre</h4>
```

Copy

- Paragraphes

```
<p>Voici un paragraphe simple</p>
```

Copy

- Listes

```
<ul>
  <li>Pommes</li>
  <li>Poires</li>
  <li>Pêches</li>
</ul>
```

Copy

`<ul>` pour «unordered list».

Pour une liste ordonnée on utiliserait `<ol>`

Baliser du texte

- Liens

```
<a href="https://fr.wikipedia.org">Liens vers Wikipédia</a>
<a href="https://fr.wikipedia.org" target="_blank">Liens vers Wikipédia</a>
```

Copy

`target="_blank"` pour ouvrir dans un nouvel onglet

Exercice 1 - Création d'un projet HTML basique

- **Créer** un dossier `html` sur votre machine
- **Ouvrir VSCode** (ou autre éditeur de votre choix) et ajouter le dossier au Workspace, puis sauvegarder votre Workspace
- **Créer 3 fichiers** : `index.html`, `styles.css`, `scripts.js`
- **Recréer le code précédent** (en utilisant emmet avec le snippet html:5)
- **Lancer l'exécution** de la page avec Go Live ou Live Preview (pour VSCode)
- **Explorer** la référence des éléments HTML et ajouter les balises suivantes à votre document: *h1, h2, h3, p, a, b, strong, ul, ol, button, form, table, img, picture, video, audio, div, section, iframe*
- **Créer un nouveau fichier** html dans votre projet, et le lier avec une balise *a*

Déboguer le HTML

- Écrire du code HTML, c'est bien, **mais si quelque chose se passe mal**, que faire pour trouver où est l'erreur dans le code ?
- Dans les **langages de programmation traditionnels**, le compilateur ou l'interpréteur vont afficher et donner des informations sur les erreurs du code source
- Le problème avec le **HTML**, c'est que **l'interpréteur est très permissif** et va exécuter le code HTML, même s'il contient des erreurs !
- C'est problématique, car si parfois l'affichage de la page nous permet de détecter des problèmes, ce n'est pas toujours le cas

Déboguer le HTML

- Il existe 3 solutions pour écrire du HTML correctement :
  - Être attentif et utiliser un éditeur de code comme VSCode, des extensions HTML comme emmet, mais aussi en utilisant le formatage du code, qui permet de visualiser si le code est correctement structuré
  - Utiliser le DevTools des navigateurs
  - Utiliser le validateur du W3C pour vérifier la structure du code

Déboguer le HTML - VSCode

- VSCode, l'éditeur le plus populaire actuellement, supporte nativement le HTML et inclut emmet, qui permet de facilement générer du code HTML
- emmet est un outil simple et très puissant qu'il faut apprendre à maîtriser

Déboguer le HTML - DevTools

- Tous les navigateurs incluent un outil pour aider au développement, le DevTools
- Le DevTools permet de déboguer le HTML, mais aussi le CSS et le Javascript
- Nous allons le survoler rapidement
- Pour aller plus loin, il faudra consulter la documentation des navigateurs concernés :
  - Chrome (Idem pour Chromium)
  - Firefox
  - Safari
  - Opera
  - Brave - Brave étant basé sur Firefox, vous pouvez utiliser celle de Firefox

Déboguer le HTML - DevTools - Chrome

- Les exemples suivant sont sur Chrome, mais le DevTools est similaire sur les autres navigateurs
- Pour accéder au DevTools sur Chrome :
  - Pour utiliser le DOM ou CSS, effectuez un *clic droit* sur un élément de la page et sélectionnez *Inspecter* pour accéder au panneau Éléments. Vous pouvez également appuyer sur *Commande+Option+C* (Mac) ou sur *Ctrl+Maj+C (Windows, Linux, ChromeOS)*.
  - Pour afficher les messages enregistrés ou exécuter JavaScript, appuyez sur *Cmd+Option+J (Mac)* ou sur *Ctrl+Maj+J (Windows, Linux, ChromeOS)* pour accéder directement au panneau de la Console.

Déboguer le HTML - DevTools - Chrome

- On peut voir ici que le DevTools nous présente des informations sur le DOM
- On y voit la structure de la page HTML (*attention ce n'est pas le code source ! C'est le DOM*)
- On y voit aussi des informations sur le CSS, ainsi que d'autres onglets, nous reviendrons sur cette partie par la suite
- En passant la souris sur un élément du DOM, il devient surligné sur la page et on peut tout de suite avoir un aperçu des marges et de la taille du bloc

Le HTML - DOM vs Code Source

- Le Document Object Model ou DOM (pour modèle objet de document) est une interface de programmation pour les documents HTML, XML et SVG
- https://developer.mozilla.org/fr/docs/Web/API/Document_Object_Model
- Il fournit une représentation structurée du document sous forme d'un arbre et définit la façon dont la structure peut être manipulée par les programmes, en termes de style et de contenu
- *Attention, le DOM n'est pas la même chose que le code source d'une page HTML !*
    - Le code HTML se situe sur le serveur
    - Il est récupéré et interprété par le navigateur
    - Le résultat de cette interprétation est le DOM
    - On peut manipuler le DOM avec Javascript

Exercice 2 - Corriger un fichier HTML

- Récupérer ce fichier HTML et l'ajouter dans votre projet
- Vérifier le fichier avec le validateur W3C
- Identifier et corriger les problèmes en utilisant le DevTools et les informations du validateur W3C
- Valider à nouveau le fichier corrigé avec le validateur W3C

Exercice 3 - Faire une lettre

- Créer un nouveau dossier `html-lettre` sur votre machine
- Ajouter ce dossier à votre Workspace VSCode
- Réaliser l'exercice Faire une lettre de la documentation Mozilla, en mettant les fichier dans votre dossier `html-lettre`

Exercice 4 - Structurer une page de contenu

- Pour les plus rapides, en attendant que tout le monde ait terminé l'exercice 3, vous pouvez commencer l'exercice Structurer une page de contenu
- Créer un nouveau dossier et ajouter le à votre workspace VSCode

Une note sur les interfaces logicielles

- Les langages de programmation classiques (Java, QT, Python, etc.) disposent de librairies pour réaliser des interfaces graphiques
  - La majorité des logiciels (dit clients lourds) utilisent ces librairies
- Les interfaces réalisées en HTML, CSS et Javascript ont pendant longtemps été cantonnées aux sites web
- Mais depuis l'avènement de projets comme Electron, ce n'est plus le cas !
- De nombreuses applications utilisent aujourd'hui la stack HTML/CSS/Js pour réaliser des clients lourds
  - Exemples : VSCode, Discord, Asana, Figma, Notion, Skype, Slack, Trello, Twitch, etc.
- L'avantage principal est d'avoir des applications de bureau qui sont accessibles directement dans le navigateur

Conclusion

- Le HTML est un langage de balisage du contenu
- Chaque balise a une fonction propre
- Les balises *form* et *a* permettent de rediriger le visiteur vers d'autres pages web
- Le code source HTML est interprété par les navigateurs
- Le code source interprété est le DOM
- On peut explorer et manipuler le DOM depuis le DevTools et avec Javascript
- On peut utiliser VSCode, le DevTools et le validateur W3C pour corriger le HTML
- On peut se référer à la documentation Mozilla et W3Schools, et il existe de nombreuses autres ressources sur internet