

1. HTML to App

Construction d'une application web

The MAIF logo is a red, three-dimensional triangular sign with the word "MAIF" in white, bold, sans-serif capital letters. It is mounted on a building's exterior. The background of the slide is a blurred image of a modern building with a glass facade and a grid of structural elements.

MAIF

Structure minimale pour créer une page **html** valide

- **DOCTYPE:** définit le type de document comme étant un document HTML
- **html:** définit un document HTML
- **head:** définit un ensemble d'informations sur le document
- **body:** définit le corps du document

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Titre de la page</title>
  </head>
  <body>
    Contenu de la page
  </body>
</html>
```

Il est possible de valider une page HTML en utilisant le service en ligne validator.w3.org

Exercice 1

Création d'une page HTML statique

- Créer un fichier `index.html`
- Ajouter la structure minimale
- Ajouter un titre



Hello world 🤖

Support de cours : <https://but-sd.github.io/guide-html>

Exercice 2

Enrichissement de la page HTML statique

- Ajouter un paragraphe
- Ajouter des liens hypertextes
- Ajouter une image grâce à l'url <https://picsum.photos/200>



Problématique

Bien que la page développée soit relativement simple, on rencontre déjà quelques problématiques :

- Difficulté à différencier dans la page les différents éléments (balises, attributs, texte)
- Pour voir une modification, il faut recharger la page
- Vérifier la validité de la page HTML n'est pas évident

IDE

Un **IDE** (Integrated **D**evelopment **E**nvironment) est un environnement de développement intégré qui regroupe un ensemble d'outils pour faciliter le développement de logiciels.

Par exemple, un IDE peut proposer :

- Coloration syntaxique
- Auto-complétion
- Vérification de la validité du code

Visual Studio Code

Visual Studio Code est un IDE gratuit et open-source développé par Microsoft. Il est disponible sur Windows, Linux et macOS. Il fonctionne avec un système d'extension qui permettent d'ajouter des fonctionnalités à l'IDE

<https://code.visualstudio.com/>



Visual Studio Code - extensions

- **Live Server** : permet de lancer un serveur local pour visualiser le rendu de la page web
- **W3C Web Validator** : permet de valider une page HTML

CSS

Le **Cascading Style Sheets** (CSS) est un langage de style utilisé pour décrire la présentation d'un document écrit en HTML.

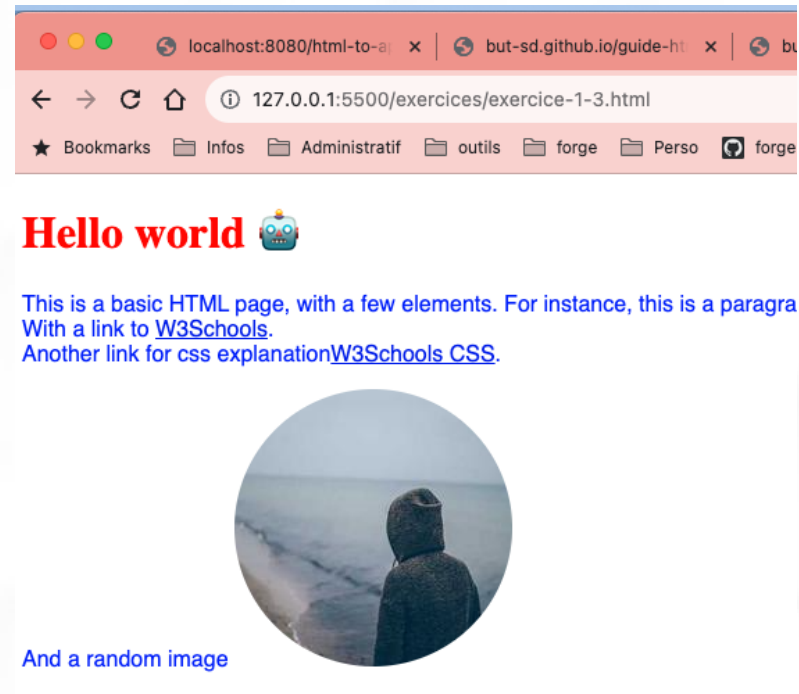
Il permet de séparer le contenu de la présentation.

Support de cours : <https://but-sd.github.io/guide-html/css/>

Exercice 3

Appliquer du style à la page HTML statique

- Utiliser un fichier CSS externe (style.css)
- Passer le titre en rouge
- Passer le paragraphe en bleu avec la police Lucida Sans ou par défaut sans-serif
- Arrondir les coins de l'image



CSS - Bonnes pratiques

- **Sélectionner les éléments à styliser** : utiliser les sélecteurs CSS
 - **classes** : permet de réutiliser un style
 - **identifiants** : permet de cibler un élément unique
- **Utiliser le bon type de style** : inline, interne ou externe
 - **inline ou interne** : pour des styles spécifiques à une page
 - **externe** : pour des styles communs à plusieurs pages
- **Organiser le code** : pour faciliter la maintenance
- **Utiliser des commentaires** : pour expliquer le code

Exercice 4

Structurer son code

- Ajouter une page
- Pouvoir naviguer entre les 2 pages
- Tester, valider que l'application des styles est correcte
- En appliquant la structure suivante :



Title with inline style

Title with class style

[Go back to index](#)

JavaScript

Le **JavaScript** est un langage de programmation qui permet de rendre les pages web interactives. Il est souvent utilisé pour ajouter des fonctionnalités à une page web.

Support de cours : <https://but-sd.github.io/guide-html/js/>

Exercice 5

Manipuler le DOM en JavaScript

- Afficher *Hello world from script.js* dans la console
- Ajouter à la fin de l'élément h1 le texte " from JS" et changer la couleur en JavaScript
- Ajouter un élément h2 *Welcome to the DOM* sous l'élément h1



Chrome DevTools

Les **Chrome DevTools** sont un ensemble d'outils de développement intégrés à Google Chrome. Ils permettent de déboguer, de profiler et d'analyser les performances des applications web.

Accessible via le menu **Plus d'outils > Outils de développement** ou en appuyant sur **F12** ou **Ctrl+Shift+I** (Windows/Linux) ou **Cmd+Opt+I** (Mac)

Exercice 5 - Suite

Manipuler le DOM en JavaScript

- Ajouter un élément h3 affichant l'heure actuelle sous l'élément h2
- Faire en sorte que l'heure s'affiche toutes les secondes, la fonction **setInterval** sera utile



Exercice 5 - Suite

Manipuler le DOM en JavaScript

- Ajouter une liste de 2 éléments en html
- Supprimer le premier élément de la liste en JavaScript
- Ajouter un élément à la fin de la liste en JavaScript



Exercice 6

Gestions des événements

- Ajouter un événement lors du survol de l'élément caption

Hello world 🤖 from JS

Welcome to the DOM

09:04:47

This is a basic HTML page, with a few elements. For instance, this is a paragraph.
With a link to [W3Schools](#).
Another link for css explanation [W3Schools CSS](#).



And a random image

A list of items:

- Item 2
- Item 3

A table with some data:

People

Name	User name	Email
Leanne Graham	Bret	leanne.graham@google.com
Ervin Howell	Antonette	ervin.howell@google.com

[Go to page 2](#)

Exercice 6 - Suite

Gestions des événements

- Ajouter des boutons (en javascript) pour faire bouger l'image et le tableau

Hello world 🤖 from JS

Welcome to the DOM

09:31:24

This is a basic HTML page, with a few elements. For instance, this is a paragraph.
With a link to [W3Schools](#).
Another link for css explanation [W3Schools CSS](#).



And a random image

Shake the image Stop shaking the image

A list of items:

- Item 2
- Item 3

A table with some data:

People

Name	User name	Email
Leanne Graham	Bret	jeanne.graham@google.com
Ervin Howell	Antonette	ervin.howell@google.com

Shake the table Stop shaking the table [Go to page 2](#)

Exercice 7

Récupérer des données depuis une API

- Récupérer des données depuis une API (<https://jsonplaceholder.typicode.com/users>)
- Afficher les données dans le tableau

Hello world 🌍 from JS

Welcome to the DOM

10:14:15

This is a basic HTML page, with a few elements. For instance, this is a paragraph.
With a link to [W3Schools](#).
Another link for css explanation [W3Schools CSS](#).



And a random image

[Shake the image](#) | [Stop shaking the image](#)

A list of items:

- Item 2
- Item 3

A table with some data:

People

Name	User name	Email
Leanne Graham	Bret	Sincere@april.biz
Ervin Howell	Antonette	Shanna@melissa.tv
Clementine Bauch	Samantha	Nathan@yesenia.net
Patricia Lebsack	Karianne	Julianne.OConner@kory.org

Analyse de ce que nous avons fait

- Outillage pour le développement web
 - IDE et extensions
 - Chrome DevTools
- Création d'une page HTML statique
- Enrichissement de la page avec du CSS
- Utilisation de JavaScript pour manipuler le DOM et gérer des événements
- Récupération de données depuis une API

Analyse de ce que nous avons fait - Suite

- Structuration du code
 - Séparation du HTML, CSS et JavaScript
 - Utilisation de classes et d'identifiants
 - Utilisation de fichiers externes pour le CSS et le JavaScript
- Respect des bonnes pratiques
 - Commentaires
 - Organisation du code
 - Utilisation des sélecteurs CSS

Analyse de ce qu'il reste à faire

- Factoriser le code
 - Réutiliser les styles communs
 - Réutiliser les fonctions JavaScript
- Gérer les erreurs
 - Vérifier que les données récupérées sont bien celles attendues
 - Gérer les cas où les données ne sont pas disponibles
- Tester le code sur différents navigateurs
 - Chrome, Firefox, Edge, Safari

Analyse de ce qu'il reste à faire - Suite

Avant de refactoriser le code, il est important d'utiliser un système de contrôle de version comme Git pour pouvoir revenir en arrière en cas de problème.

Nous allons donc commencer par versionner le code actuel en suivant le guide suivant :

<https://but-sd.github.io/prez/guide-git.html>

Refactorisation du code

Utiliser git pour versionner le code et apporter des modifications en toute sécurité.

- Refactoriser le code JavaScript pour utiliser des fonctions et rendre le code plus lisible et maintenable
- Commiter régulièrement les modifications par fonctionnalité ou par tâche
 - Ajouter un message de commit explicite pour expliquer les modifications apportées

Exercice 8 - Refactorisation du code - git

Refactorisation du code - Fonctionnalité **shake**

- Ajouter des ids aux éléments à faire bouger (image et tableau)
- Utiliser des boutons dans la page html plutôt que des boutons dans le code JavaScript
- Factoriser les fonctions **shake** et **unshake** en une seule permettant de shake ou unshake un élément en fonction de son id et de son état actuel

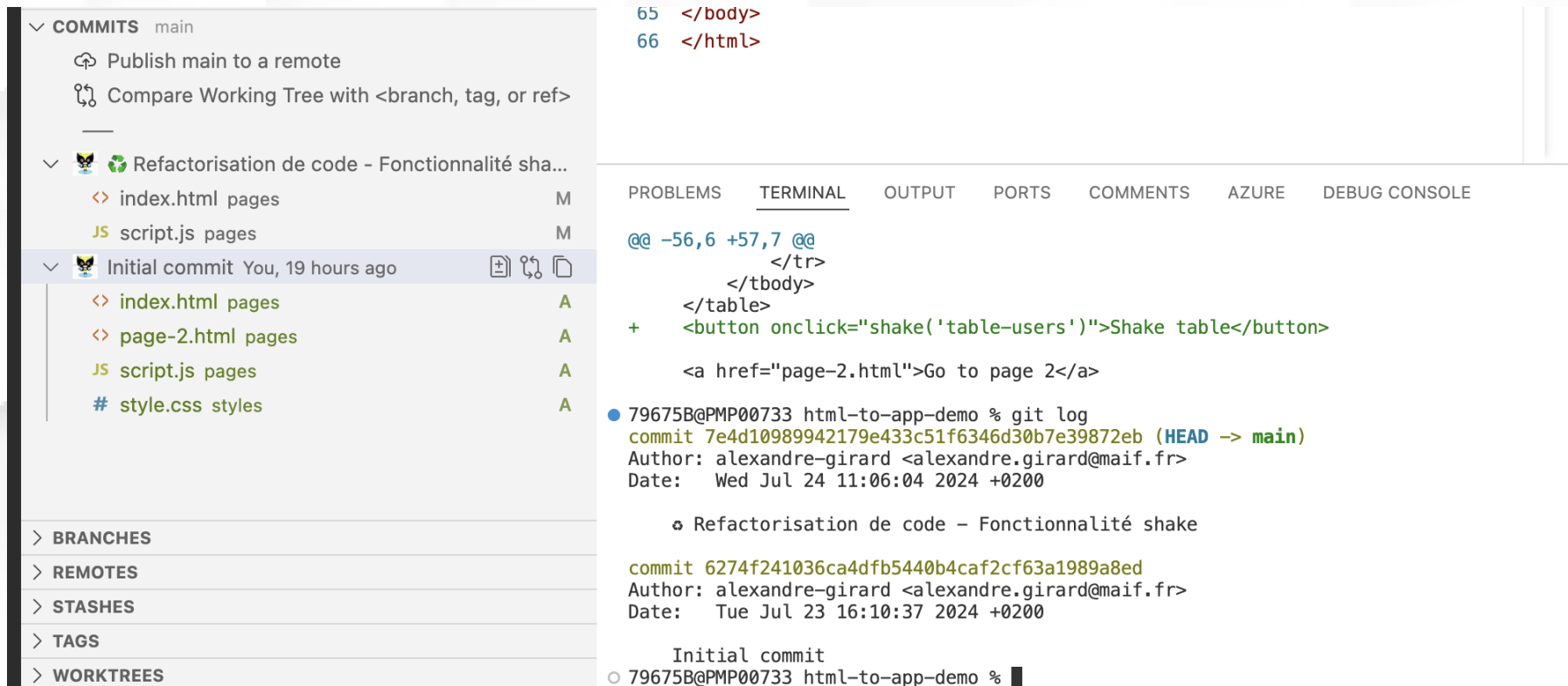
Exercice 8 - Refactorisation du code - git (suite)

Refactorisation du code - Fonctionnalité **shake**

- Préparer un commit pour l'ensemble des modifications
 - Il serait possible de ne pas tout commiter en une seule fois, cependant les 3 modifications sont liées, il est donc préférable de les regrouper dans un seul commit
 - Analyser les modifications apportées pour s'assurer qu'elles sont cohérentes et fonctionnent correctement, on ne commit pas du code instable
- Ajouter un message de commit explicite
 - Exemple : "Refactorisation de code - Fonctionnalité shake"

Exercice 8 - Refactorisation du code - git (suite)

Refactorisation du code - Fonctionnalité **shake**



65 `</body>`
66 `</html>`

PROBLEMS TERMINAL OUTPUT PORTS COMMENTS AZURE DEBUG CONSOLE

```
@@ -56,6 +57,7 @@
    </tr>
  </tbody>
</table>
+ <button onclick="shake('table-users')">Shake table</button>

<a href="page-2.html">Go to page 2</a>

79675B@PMP00733 html-to-app-demo % git log
commit 7e4d10989942179e433c51f6346d30b7e39872eb (HEAD -> main)
Author: alexandre-girard <alexandre.girard@maif.fr>
Date: Wed Jul 24 11:06:04 2024 +0200

    ✨ Refactorisation de code - Fonctionnalité shake

commit 6274f241036ca4dfb5440b4caf2cf63a1989a8ed
Author: alexandre-girard <alexandre.girard@maif.fr>
Date: Tue Jul 23 16:10:37 2024 +0200

    Initial commit

79675B@PMP00733 html-to-app-demo %
```


Exercice 9 - Amélioration de la récupération des données - Gestion des erreurs

- Gérer les cas où les données ne sont pas disponibles (erreur 404 ou autre)
 - Afficher un message d'erreur à l'utilisateur pour l'informer que les données ne sont pas disponibles
- Gérer le cas où il n'y a pas de données à afficher
 - Afficher un message à l'utilisateur pour l'informer qu'il n'y a pas de données à afficher
- Préparer et effectuer un commit pour l'ensemble des modifications
 - Analyser les modifications
 - Ajouter un message de commit explicite

Exercice 9 - Amélioration de la récupération des données - Gestion des erreurs (suite)

The screenshot displays the VS Code interface. On the left, the 'Git Explorer' sidebar is open, showing the 'COMMIT' view. It lists several commits, including 'Initial commit' and 'Refactorisation de code - Fonctionnalité shake'. The main editor area shows a code diff for a file named 'users.html'. The diff highlights changes in the 'users' array, specifically adding a new user object. The 'Terminal' panel at the bottom shows the output of a 'git rebase' command, indicating a successful rebase with 31 insertions and 21 deletions.

Exercice - Fin

- Poussez vos modifications sur votre dépôt distant pour les sauvegarder.
- Nous avons maintenant une application web basique mais fonctionnelle qui récupère des données depuis une API et les affiche dans une page web.
- Le code est versionné et peut être partagé avec d'autres personnes. Les modifications apportées sont enregistrées et peuvent être consultées à tout moment.
- Il serait possible de continuer à améliorer l'application en ajoutant de nouvelles fonctionnalités ou en améliorant les fonctionnalités existantes.

Pour aller plus loin

On atteint ici les limites de la programmation front-end. Pour aller plus loin, il est possible de se tourner vers des frameworks ou des bibliothèques JavaScript comme React, Angular ou Vue.js qui permettent de développer des applications web plus complexes.

- **Responsive Design** : adapter la page web à différents supports (mobile, tablette, desktop)
- **Performance** : optimiser le temps de chargement de la page
- **Tests** : écrire des tests unitaires et d'intégration pour garantir le bon fonctionnement de l'application
- **Accessibilité** : rendre la page accessible à tous