Exercices

Une infrastructure de développement HTML

Éditeur de texte

Pour écrire du code HTML, vous aurez besoin d'un éditeur de texte. Il existe de nombreux éditeurs spécialement conçus pour le développement web, tels que :

- **Visual Studio Code** : Gratuit, extensible et disponible sur toutes les principales plateformes.
- **Sublime Text** : Payant, mais très rapide et également extensible.
- **Atom** : Gratuit, créé par GitHub, et hautement personnalisable.
- **Notepad++** : Une option solide pour les utilisateurs de Windows.

Navigateur Web

Vous aurez également besoin d'un navigateur web pour afficher vos pages HTML. Les navigateurs les plus couramment utilisés sont :

- **Google Chrome**
- **Mozilla Firefox**
- **Safari**
- **Microsoft Edge**

Ces navigateurs ont des outils de développement intégrés qui vous permettent d'inspecter le HTML, le CSS et le JavaScript, ce qui est très utile pour le débogage.

Site de validation

Il est bon de valider votre HTML pour vous assurer qu'il est bien formé et qu'il respecte les dernières normes. Le W3C offre un service de validation en ligne gratuit :

- Validateur HTML du W3C

Hébergement

Si vous souhaitez partager vos pages HTML avec le monde, vous aurez besoin d'un service d'hébergement. GitHub Pages est une excellente option pour cela, surtout si vous êtes nouveau dans le domaine :

1. **Créez un dépôt GitHub** : Vous pouvez utiliser ce dépôt pour stocker votre code HTML, CSS et JavaScript.
2. **Activez GitHub Pages** : Allez dans les paramètres du dépôt et activez GitHub Pages. Vous pouvez choisir de publier le code de la branche `main` ou d'une branche spécifique.
3. **Accès au site** : Une fois que GitHub Pages est activé, votre site sera accessible au public. GitHub fournit une URL où vous pouvez voir votre site en direct.

Workflow basique avec GitHub Pages

1. **Écrivez votre code HTML** dans votre éditeur de texte préféré.
2. **Testez localement** en ouvrant le fichier HTML avec un navigateur web.
3. **Poussez le code** vers votre dépôt GitHub.
4. **Attendez la publication** : GitHub Pages mettra à jour votre site web en fonction des nouveaux fichiers poussés.
5. **Validez votre code** en utilisant le validateur du W3C pour vous assurer qu'il est conforme aux normes.

En combinant ces outils et services, vous aurez une infrastructure solide pour développer et héberger des sites web en HTML.

Installation de Git

Sur Windows

1. **Téléchargez l'installateur** : Allez sur le site officiel de Git (https://git-scm.com/) et téléchargez l'installateur pour Windows.
2. **Exécutez l'installateur** : Ouvrez le fichier `.exe` téléchargé et suivez les instructions à l'écran. Vous pouvez laisser la plupart des options par défaut.
3. **Vérifiez l'installation** : Ouvrez une invite de commande et tapez `git --version`. Si l'installation a réussi, cela devrait afficher la version de Git.

Sur macOS

1. **Utilisez Homebrew** : Si vous avez Homebrew installé, vous pouvez simplement taper `brew install git` dans le terminal.
2. **Téléchargez l'installateur** : Sinon, vous pouvez télécharger l'installateur macOS depuis le site officiel de Git (https://git-scm.com/).
3. **Vérifiez l'installation** : Ouvrez un terminal et tapez `git --version` pour vérifier que Git est bien installé.

Sur iPad

Git n'est pas nativement supporté sur iPad, mais il existe des applications tierces qui vous permettent d'utiliser Git, comme Juno ou Working Copy. Ces applications ont des interfaces graphiques pour gérer vos dépôts Git.

Utilisation de base de Git pour le développement HTML

1. **Initialisation du dépôt** : Ouvrez un terminal, naviguez vers votre dossier de projet HTML et tapez `git init` pour initialiser un nouveau dépôt Git.
2. **Ajout de fichiers** : Utilisez `git add .` pour ajouter tous les fichiers du dossier au suivi de version. Vous pouvez aussi ajouter des fichiers spécifiques avec `git add <nom-du-fichier>`.

3. **Commit** : Après avoir ajouté les fichiers, tapez `git commit -m "Votre message de commit"` pour enregistrer les modifications.
4. **Ajout d'un dépôt distant** : Si vous utilisez GitHub, créez un nouveau dépôt et ajoutez-le comme dépôt distant en utilisant `git remote add origin <URL-du-dépôt>`.
5. **Push** : Utilisez `git push -u origin main` pour envoyer vos commits au dépôt distant. Si vous utilisez une branche autre que `main`, remplacez `main` par le nom de votre branche.

Si le dépôt existe déjà sur une plateforme comme GitHub, GitLab ou Bitbucket, vous pouvez le cloner sur votre machine locale pour travailler dessus.
Voici comment faire :

Clonage d'un dépôt Git

1. **Obtenez l'URL du dépôt** : Rendez-vous sur la page du dépôt sur le site web où il est hébergé (GitHub, GitLab, Bitbucket, etc.). Vous devriez voir un bouton ou un lien pour "Cloner" ou "Clone". Cliquez dessus pour copier l'URL du dépôt.
2. **Ouvrez un terminal** : Ouvrez une fenêtre de terminal sur votre ordinateur.
3. **Naviguez vers le dossier où vous souhaitez cloner le dépôt** : Utilisez la commande `cd` pour naviguer vers le dossier où vous souhaitez que le dépôt soit cloné.

```
cd chemin/vers/le/dossier
```

4. **Clonez le dépôt** : Utilisez la commande `git clone` suivie de l'URL du dépôt pour le cloner.

```
git clone https://github.com/utilisateur/nom-du-depot.git
```

   Cela créera un nouveau dossier dans votre dossier actuel, contenant une copie du dépôt.
5. **Naviguez dans le dossier du dépôt** : Utilisez `cd` pour entrer dans le dossier du dépôt cloné.

```
cd nom-du-depot
```

6. **Vérifiez l'état du dépôt** : Vous pouvez utiliser la commande `git status` pour voir l'état actuel du dépôt.

```
git status
```

Maintenant, vous avez une copie locale du dépôt et vous pouvez commencer à travailler dessus. N'oubliez pas de faire régulièrement des `git pull` pour récupérer les dernières modifications du dépôt distant, surtout si d'autres personnes y travaillent également.

Après avoir effectué vos modifications, vous pouvez les envoyer (push) vers le dépôt distant en utilisant les commandes `git add`, `git commit` et `git push`, comme décrit dans mon message précédent.

Cela vous permet de garder votre version locale synchronisée avec la version en ligne, ce qui est particulièrement utile si vous travaillez en équipe ou si vous souhaitez accéder à votre code depuis différents ordinateurs.

GitHub Pages

Après avoir poussé votre code sur GitHub, vous pouvez activer GitHub Pages dans les paramètres du dépôt pour publier votre site web.

1. **Allez dans les paramètres du dépôt** sur GitHub.
2. **Trouvez la section GitHub Pages** et choisissez la branche que vous souhaitez publier (généralement `main` ou `gh-pages`).
3. **Cliquez sur "Save" ou "Enregistrer"**. Votre site sera accessible à l'URL indiquée.

En suivant ces étapes, vous aurez un environnement de développement HTML fonctionnel avec Git pour le suivi de version et GitHub Pages pour l'hébergement.

Exercices

Exercice 1: Structure de base

Créez une page HTML simple avec les éléments de base : `<!DOCTYPE>`, `<html>`, `<head>`, et `<body>`.

Exercice 2: Titres et Paragraphes

Ajoutez des titres de différents niveaux (`<h1>`, `<h2>`, etc.) et des paragraphes (`<p>`) à votre page.

Exercice 3: Listes

Créez une liste à puces (`<ul>`) et une liste numérotée (`<ol>`), chacune contenant au moins 3 éléments (`<li>`).

Exercice 4: Liens

Ajoutez des liens hypertexte (`<a>`) à votre page qui mènent vers d'autres sites web, ainsi qu'un lien qui envoie un e-mail (utilisez l'attribut `mailto:`).

Exercice 5: Images

Intégrez une image (`<img>`) dans votre page et utilisez l'attribut `alt` pour ajouter une description textuelle de l'image.

Exercice 6: Tableaux

Créez un tableau (`<table>`) avec des en-têtes (`<th>`) et au moins deux lignes (`<tr>`) et deux colonnes (`<td>`).

Exercice 7: Formulaires

Créez un formulaire simple (`<form>`) avec différents types de champs : texte (`<input type="text">`), mot de passe (`<input type="password">`), bouton radio (`<input type="radio">`), case à cocher (`<input type="checkbox">`), et bouton d'envoi (`<input type="submit">`).

Exercice 8: Utilisation de `div` et `span`

Utilisez les éléments `<div>` et `<span>` pour regrouper des blocs de contenu et appliquez des styles CSS simples à ces éléments.

Exercice 9: Structure sémantique

Refaites la structure de votre page en utilisant des éléments HTML5 sémantiques comme `<header>`, `<footer>`, `<article>`, `<section>`, `<nav>`, etc.