Nom : BIDOUNDA

Prénoms : Béni Valérien

Mail : benivalerien@gmail.com

Promo 9

Exercice Développement Web et web mobile

Série 1 : Les bases du HTML

1. *Qu'est-ce que le HTML, et à quoi sert-il ?*

Le HTML (HyperText Markup Language) est un langage de balisage créer en **1991** par **Tim Berners-Lee**. Son rôle est de structurer une page web en mettant des titres, des paragraphes, des liens, des images.

2. *Quelle est la structure de base d'un document HTML ?*

La structure de base d'un document HTML comprend les éléments essentiels pour créer une page web. Voici un exemple de cette structure :

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Titre de la page</title>
</head>
<body>
    <h1>Bienvenue sur ma page web</h1>
    <p>Ceci est un paragraphe de texte.</p>
</body>
</html>
```

- **<!DOCTYPE html>** : Indique au navigateur que le document est en HTML.
- **<html lang="fr">** : La balise racine de l'ensemble du document HTML, avec l'attribut lang spécifiant la langue du document.
- **<head>** : Contient des méta-informations sur le document, comme le jeu de caractères, les titres, les liens vers les styles CSS, etc.
- **<meta charset="UTF-8">** : c'est un attribut qui détermine la façon dont le texte est transmis et stocké..

- **<meta name="viewport" content="width=device-width, initial-scale=1.0">** : Assure que la page s'affiche correctement sur tous les types d'appareils, y compris les mobiles.
- **<title>** : Définie le titre de la page, affiché dans l'onglet du navigateur.
- **<body>** : Contient le contenu visible de la page, tel que les titres, les paragraphes, les images, les liens, etc.

3. *Que signifie la balise <h1>, et comment est-elle utilisée ?*

Le langage HTML comprend six balises de titres différentes de **<h1>** à **<h6>.** La balise **<h1>** représente un titre de premier niveau. Elle est utilisée pour définir le titre le plus important sur une page web. Il est possible que la balise **<h1>** soit similaire à la balise de **<title>,** car Les moteurs de recherche accordent une importance particulière aux titres **<h1>** pour comprendre le contenu principal de la page. Utiliser un titre **<h1>** pertinent peut aider à améliorer le classement et l'accessibilité de votre page dans les résultats de recherche.

Voici un exemple d'utilisation de la balise **<h1>** :

```
<body>
    <h1>Bienvenue sur mon site web</h1>
    <p>Ceci est un paragraphe d'introduction.</p>
</body>
```

4. *Qu'est-ce qu'une balise <a>et comment crée-t-on un lien hypertexte ?*

La balise **<a>** (pour **ancre** ou **anchor** en anglais), avec son attribut **href,** est utilisée pour créer des liens hypertexte en HTML. Elle permet de lier une page web à une autre, de télécharger des fichiers ou même de créer des liens vers des sections spécifiques d'une même page. Le contenu de chaque élément **<a>** doit indiquer la destination du lien.

Structure de base

```
<a href="URL">Texte du lien</a>
```

Explication des attributs

href : Cet attribut contient l'URL de la page vers laquelle vous souhaitez créer un lien. Il peut s'agir d'une page web, d'un fichier téléchargeable, ou d'une ancre sur la même page.

Texte du lien : Le texte visible sur lequel les utilisateurs cliqueront pour suivre le lien.

5. *Comment insérer une image en HTML ?*

La balise qui permet d'insérer une image est une **balise orpheline** : **<img>.** Pour fonctionner correctement elle doit être accompagnée de deux attributs :

- **Src (source)** : il permet d'indiquer la source de l'image pour montrer au navigateur comment récupérer l'image que l'on veut insérer. Il existe deux type de chemin, le chemin absolu (qui consiste à indiquer la source d'une image lorsque celle-ci est en ligne, sur un site, ou qu'elle est hébergée sur le web) et le chemin relatif (qui consiste à utilise le nom et l'arborescence du fichier de l'image en local pour indiquer sa source
- **Alt (texte alternatif) :** il permet de donner à l'image une description. C'est-à-dire si la page web n'affiche pas l'image au moment du chargement le texte de la description sera inscrit à la place.

Syntaxe de base :

```
<img src="URL de l'image" alt="Description de l'image">
```

- **title** : cet attribut permet d'ajouter une infobulle Afin d'afficher une bulle d'aide sur vos images (cet attribut est facultatif, à ne pas confondre avec la balise **<title>** qui permet d'indiquer au navigateur le titre d'une page web)

Série 2 : Les Fondamentaux du CSS

1. *Qu'est-ce que le CSS, et comment s'intègre-t-il au HTML ?*

Le CSS (Cascading Style Sheets) est un langage utilisé pour décrire la présentation d'un document écrit en HTML ou XML. Tandis que le HTML structure le contenu de la page web, le CSS est utilisé pour définir l'apparence et le style de ce contenu, y compris les couleurs, les polices, les marges, l'espacement, et plus encore. Il existe trois façons principales d'intégrer le CSS dans un document HTML :

- **CSS en ligne (inline CSS)** : Utilisé pour appliquer des styles directement à un élément HTML individuel.

```
<p style="color: blue; font-size: 20px;">Ceci est un paragraphe styl
```

- **CSS intégré (internal CSS)** : Ajouté dans la section <head> d'un document HTML à l'aide de la balise <style>.

```html
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale
    <title>Exemple de CSS intégré</title>
    <style>
        body {
            background-color: #f0f0f0;
        }
        h1 {
            color: green;
        }
        p {
            font-size: 18px;
        }
    </style>
</head>
```

- **CSS externe (external CSS)** : Utilisé pour appliquer des styles à plusieurs pages web à partir d'un seul fichier CSS externe. Il faut créer un fichier CSS séparé (par exemple, styles.css) et le lier au document HTML avec la balise <link>.

```css
body {
    background-color: #f0f0f0;
}
h1 {
    color: green;
}
p {
    font-size: 18px;
}
```

Ces méthodes permettent de séparer le contenu (HTML) de la présentation (CSS), facilitant ainsi la gestion et la maintenance du code.

2. *Expliquez la différence entre les sélecteurs classes et ID en CSS.*

Les sélecteurs classes et ID en CSS permettent de cibler des éléments spécifiques pour appliquer des styles, mais ils ont des usages et des caractéristiques distinctes. Une classe est un sélecteur réutilisable qui peut être appliqué à plusieurs éléments. Dans une syntaxe CSS, les classes sont précédées d'un point (.) et en HTML elles utilisent l'attribut **class**. Contrairement à un ID qui est un sélecteur unique qui ne peut être appliqué qu'à un seul élément sur une page. Les IDs sont précédés d'un dièse **(#)** dans une syntaxe CSS et utilisent l'attribut **id** dans l'élément HTML.

Syntaxe classe en HTML :

```html
<p class="highlight">Ceci est un paragraphe.</p>
<p class="highlight">Ceci est un autre paragraphe.</p>
```

Syntaxe classe en CSS :

```css
.highlight {
    color: blue;
    font-weight: bold;
}
```

Syntaxe ID en HTML

```html
<p id="intro">Ceci est un paragraphe d'introduction.</p>
```

Syntaxe ID en CSS :

```css
#intro {
    color: green;
    font-style: italic;
}
```

En ésumé, on utilise les classes pour les styles réutilisables sur plusieurs éléments et les IDs pour cibler des éléments uniques et spécifique.

3. *Qu'est-ce qu'un "margin" et un "padding" en CSS ?*

En CSS, les concepts de "margin" et de "padding" sont utilisés pour créer de l'espace autour des éléments HTML

- **Margin (marge) :** est l'espace extérieur entourant un élément. Elle crée de la distance entre l'élément et les autres éléments autour de lui. Elle peut être définie individuellement pour chaque côté (haut, droite, bas, gauche) avec **margin-top**, **margin-right**, **margin-bottom**, et **margin-left**, ou en une seule propriété avec **margin**.

```css
p {
    margin: 20px; /* Ajoute une marge de 20 pixels autour de tous les c
}
```

- **Padding (remplissage) :** est l'espace intérieur entre le contenu de l'élément et ses bordures. Il crée de la distance entre le contenu de l'élément (comme du texte) et les bordures de cet élément. Il peut également être définis individuellement pour chaque côté avec **padding-top**, **padding-right**, **padding-bottom**, et **padding-left**, ou en une seule propriété avec **padding**.

```css
p {
    padding: 10px; /* Ajoute un remplissage de 10 pixels à l'intérieur
}
```

4. *Comment changer la couleur de fond d'un élément ?*

Pour changer la couleur de fond d'un élément on utilise le langage CSS avec sa propriété qui joue ce rôle intitulé **background-color.** Il existe plusieurs notations pour définir la couleur d'un font tel que :

- **"background-color : "** suivi du nom de couleurs **(red, blue, green, yellow, etc.)**
- **"background-color : "** suivi de la valeurs hexadécimales **(#ffffff (blanc), #000000 (noir), etc.)**
- **"background-color : "** suivi de la valeurs RGB **(rgb (255, 0, 0) (rouge), rgb (0, 255, 0) (vert), etc.)**
- **"background-color : "** suivi de la valeurs RGBA **(rgba (255, 0, 0, 0.5) (rouge avec une transparence de 50%).)**

```
<style>
    body {
        background-color: #f0f0f0; /* Couleur de fond du corps de l
    }
    .conteneur {
        background-color: lightblue; /* Couleur de fond d'un div sp
        padding: 20px;
    }
    p {
        background-color: yellow; /* Couleur de fond d'un paragraph
    }
</style>
```

5. *Comment appliquer un style CSS uniquement sur un paragraphe spécifique ?*

Pour appliquer un style CSS uniquement à un paragraphe spécifique, vous pouvez utiliser des sélecteurs de classe (class) ou d'ID (id). En utilisant l'un de ses sélecteurs, vous pouvez cibler et styliser des paragraphes spécifiques sans affecter les autres éléments de la page.

Série 3 : Le JavaScript et le DOM

1. *Qu'est-ce que le JavaScript, et pourquoi est-il utilisé ?*

Le JavaScript est un langage de programmation polyvalent et dynamique utilisé principalement pour rendre les pages web interactives et réactives. Il s'exécute principalement côté client dans le navigateur web, mais peut également être utilisé côté serveur avec des environnements comme Node.js. Il est utilisé pour plusieurs raison tel que :

- **L'interactivité :** il permet d'ajouter des fonctionnalités interactives aux sites web, telles que les formulaires dynamiques, les animations, et les mises à jour de contenu sans rechargement de page.

- **La manipulation du DOM** : Il permet de manipuler le Document Object Model (DOM) pour changer la structure, le style et le contenu des pages web de manière dynamique.
- Validation des formulaires : JavaScript peut vérifier les données saisies par l'utilisateur avant qu'elles ne soient envoyées au serveur, améliorant ainsi l'expérience utilisateur.
-
- **Communication asynchrone :** Avec AJAX (Asynchronous JavaScript and XML), JavaScript peut effectuer des requêtes HTTP en arrière-plan pour charger des données supplémentaires sans recharger la page.
-
- **Compatibilité multiplateforme :** JavaScript fonctionne sur tous les navigateurs web modernes, ce qui en fait un choix universel pour le développement web.
-
- **Extensions et frameworks :** Il existe de nombreux frameworks et bibliothèques JavaScript, comme React, Angular, et Vue.js, qui simplifient le développement et ajoutent des fonctionnalités puissantes.

2. *Comment inclure-on un fichier JavaScript dans une page HTML ?*

Pour inclure un fichier JavaScript dans une page HTML, vous utilisez la balise **<script>** avec l'attribut **src** pour spécifier le chemin vers le fichier JavaScript.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale
    <title>Exemple d'inclusion de JavaScript</title>
    <!-- Inclusion du fichier JavaScript -->
    <script src="script.js"></script>
</head>
<body>
    <button onclick="afficherMessage()">Cliquez-moi</button>
</body>
</html>
```

La ligne **<script src="script.js"></script>** inclut le fichier script.js dans le document HTML. Le chemin vers le fichier JavaScript peut être relatif ou absolu. La balise **<script>** peut être placée dans la section **<head>** ou à la fin de la section **<body>.** Placer le script à la fin du **<body>** peut améliorer le chargement de la page, car le HTML sera chargé avant le script.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <title>Exemple d'inclusion de JavaScript</title>
</head>
<body>
    <button onclick="afficherMessage()">Cliquez-moi</button>
    <!-- Inclusion du fichier JavaScript à la fin du body -->
    <script src="script.js"></script>
</body>
</html>
```

3. *Qu'est-ce que le DOM (Document Object Model) en JavaScript ?*

Le DOM (Document Object Model) est une interface de programmation qui permet aux scripts de mettre à jour le contenu, la structure et le style d'un document web. Le DOM représente le document comme une hiérarchie d'objets, ce qui permet de créer des pages web dynamiques et interactives en manipulant directement la structure et le contenu des documents HTML via JavaScript

Structure du DOM

- **Document** : L'objet racine qui représente la page entière.
- **Éléments** : Chaque balise HTML est un nœud dans l'arborescence DOM, comme ** , <p>, <a>,** etc.
- **Attributs** : Les propriétés des éléments HTML, comme **class, id, src**, etc., sont des nœuds d'attributs.
- **Texte** : Les contenus textuels des éléments sont des nœuds de texte.

4. *Expliquez ce que fait la méthode getElementById.*

La méthode getElementById en JavaScript est utilisée pour accéder à un élément du DOM en utilisant l'attribut id de cet élément. C'est l'une des méthodes les plus couramment utilisées pour cibler des éléments spécifiques dans une page web.

Pour qu'il fonctionne on doit insérer la syntaxe suivante :
document.getElementById("id") d'où le paramètre "id" est une chaîne de caractères qui correspond à l'attribut id de l'élément que vous souhaitez sélectionner. Cette méthode retourne une référence à l'élément HTML correspondant, ou **null** si aucun élément avec cet id n'est trouvé.

Exemple d'utilisation

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Exemple getElementById</title>
</head>
<body>
    <p id="intro">Ceci est un paragraphe d'introduction.</p>
    <button onclick="changerTexte()">Changer le texte</button>

    <script>
        function changerTexte() {
            // Utilisation de getElementById pour accéder à l'élément a
            var paragraphe = document.getElementById("intro");
            // Modification du texte de l'élément
            paragraphe.textContent = "Le texte a été changé !";
        }
    </script>
</body>
</html>
```

Explication du code

- **HTML :** Nous avons un paragraphe avec id="intro" et un bouton. Lorsque le bouton est cliqué, la fonction **changerTexte** est appelée.

- **JavaScript :**
  - **document.getElementById("intro")** : Cette ligne récupère l'élément **<p>** avec l'ID "intro".
  - **Modification** : Ensuite, la propriété **textContent** de cet élément est modifiée pour afficher **"Le texte a été changé !".**

5. *Comment afficher un message dans la console JavaScript ?*

Pour afficher un message dans la console JavaScript, vous pouvez utiliser la méthode **console.log().** Cette méthode est couramment utilisée pour déboguer et surveiller les valeurs des variables, ainsi que pour afficher des messages d'information.

```
console.log("Bonjour, monde !");
```

Explication

console.log() : La méthode **console.log()** affiche le message ou la valeur passée en argument dans la console du navigateur.

Pour voir le message dans la console, ouvrez les outils de développement de votre navigateur (généralement en appuyant sur F12 ou en cliquant avec le bouton droit de la souris et en sélectionnant "Inspecter"), puis accédez à l'onglet "Console".

Série 4 : Introduction aux Outils de Développement Web

1. *À quoi sert un éditeur de code comme Visual Studio Code ?*

Visual Studio Code **(VS Code)** est un éditeur de code source très populaire, développé par Microsoft en 2015. Il est largement utilisé par les développeurs pour diverses raisons tel que :

- Connu pour sa performance et sa rapidité, VS Code est relativement léger comparé à des IDE complets, tout en offrant des fonctionnalités avancées.

- Il prend en charge de nombreux langages de programmation (**JavaScript, Python, Java, C++, HTML, CSS, etc**…) et propose une coloration syntaxique et une auto complétion intelligente pour faciliter l'écriture du code.

- Il intègre **Git** et des outils de débogage puissants qui permettent de fixer des points d'arrêt, de surveiller les variables et de suivre l'exécution du code en temps réel.

- Il possède une vaste bibliothèque d'extensions pour ajouter des fonctionnalités supplémentaires, comme des linters, des thèmes, des

gestionnaires de paquets, etc et permet de personnaliser l'éditeur selon les besoins spécifiques grâce à des réglages et des extensions.

- Avec l'extension Live Share, les développeurs peuvent collaborer en temps réel sur le même code, partager des sessions de débogage et éditer simultanément.

2. *Comment exécuter une page web en local sur votre ordinateur ?*

La manière la plus simple d'exécuter une page web en local est d'ouvrir le fichier HTML directement dans un navigateur web. Faire un clic droit sur le fichier **HTML** et sélectionnez "Ouvrir avec" puis choisissez votre navigateur préféré (Chrome, Firefox, Edge, etc.). Pour des projets plus complexes, vous pouvez utiliser un serveur local. C'est-à-dire ouvrir Visual Studio Code, se rendre dans le gestionnaire d'extensions (icône des carrés), et installez l'extension **"Live Server".** Ensuite ouvrir le dossier contenant vos fichiers HTML, CSS, et JavaScript pour enfin cliquer avec le bouton droit sur votre fichier index.html et sélectionnez **"Open with Live Server**". Votre page s'ouvrira dans le navigateur et se mettra à jour automatiquement à chaque modification.

3. *Qu'est-ce que l'inspecteur d'éléments dans les navigateurs web ?*

L'inspecteur d'éléments est un outil intégré dans les navigateurs web (comme Chrome, Firefox, Safari, etc.) qui permet aux utilisateurs d'examiner et de modifier temporairement le code HTML, CSS et JavaScript d'une page web. Il est très utile pour les développeurs, les designers et même les curieux souhaitant comprendre comment une page est construite.

Les fonctions principales de l'inspecteur d'éléments sont :

- Examiner le code HTML et CSS : Il permet de voir la structure HTML de la page et les styles CSS appliqués aux éléments, ce qui aide à comprendre comment la page est mise en page et stylisée.

- Modifier en temps réel : Les utilisateurs peuvent modifier le code HTML ou CSS directement dans l'inspecteur pour voir immédiatement le résultat dans le navigateur sans modifier le site réel. Ces changements sont temporaires et n'affectent pas le site d'origine.

- Débogage JavaScript : L'inspecteur fournit des outils de console pour exécuter et tester du code JavaScript, analyser des erreurs ou des messages de débogage, et observer le comportement interactif de la page.

- Analyse des performances : Certains inspecteurs d'éléments, comme celui de Chrome, offrent des outils pour analyser les performances de la page,

mesurer les temps de chargement, identifier les ressources lourdes et optimiser les performances.

- Responsivité et tests de mise en page : On peut simuler différents appareils (smartphones, tablettes, etc.) pour tester comment la page s'affiche sur différentes tailles d'écran

4. *2. Comment ouvrir la console JavaScript sur votre navigateur ?*

Pour ouvrir la console JavaScript dans votre navigateur, vous pouvez utiliser différentes méthodes selon le navigateur que vous utilisez. La console JavaScript est un outil intégré dans la plupart des navigateurs modernes, utilisé pour tester et déboguer le code JavaScript. Pour l'ouvrir dans les navigateurs les plus courants comme **Google Chrome, Mozilla Firefox et Microsoft edge**, il faut appuyer sur la touches F12 ou faire un clic droit sur la page, sélectionner Inspecter l'élément, puis allez dans l'onglet Console. Maintenant sur **Safari (Mac)** il faut aller **dans Safari > Préférences > Avancées**, et cochez l'option **Afficher le menu Développement** dans la barre de menus si ce n'est pas déjà fait, ensuite dans **Développement >** Afficher la console ou appuyez sur **Cmd + Option + C** pour ouvrir directement la console JavaScript.

Une fois la console ouverte, vous pouvez y saisir et exécuter des commandes JavaScript, voir les erreurs, afficher des messages de débogage, et bien plus encore.

5. *Pourquoi utiliser Git dans un projet de développement ?*

Il est utile d'utiliser **Git** dans un projet de développement parce que :

- Git enregistre tous les changements effectués dans le code, ce qui permet de revenir à une version antérieure en cas de problème ou d'erreur. Cela facilite également la compréhension de l'évolution du projet.

- Avec Git, plusieurs développeurs peuvent travailler simultanément sur un même projet sans risquer de perdre leur travail ou de se gêner. Chaque développeur peut travailler dans sa propre branche, ce qui permet de travailler en parallèle et d'éviter les conflits.

- Git permet de créer des branches, c'est-à-dire des versions alternatives du projet. Les branches sont utiles pour développer de nouvelles fonctionnalités, corriger des bugs ou expérimenter des idées sans affecter la version principale (branche main ou master).

- Les modifications apportées dans les différentes branches peuvent être fusionnées (ou intégrées) dans la branche principale. Git aide à gérer les conflits éventuels entre les changements, ce qui rend l'intégration continue plus facile.

- Avec des plateformes comme **GitHub, GitLab** ou **Bitbucket**, Git facilite les revues de code et les discussions autour des changements. Les développeurs peuvent faire des **pull requests** pour proposer leurs modifications, permettant aux autres membres de l'équipe de les examiner et de donner leur avis avant la fusion.

- Git étant un système décentralisé, chaque copie locale d'un dépôt est une sauvegarde complète du projet. Cela rend le projet plus résilient en cas de problème de serveur ou de perte de données.

- Git s'intègre facilement avec des outils de **CI/CD** (comme **Jenkins, GitHub Actions, GitLab CI**), permettant de tester, de valider et de déployer automatiquement le code lorsqu'il est modifié.

Donc pour conclure **Git** offre aux développeurs un moyen structuré, sûr et efficace de gérer les modifications du code, de travailler en équipe, et de maintenir un projet stable tout en permettant des développements en parallèle. C'est un outil essentiel pour la gestion des versions et le travail collaboratif dans le développement logiciel.

Série 5 : Premiers Pas avec Git et GitHub

1. *Qu'est-ce que Git, et à quoi sert-il ?*

Git est un système de gestion de version distribué qui permet de suivre les modifications apportées au code source d'un projet au fil du temps. Créé par **Linus Torvalds** en **2005**, Git est aujourd'hui l'un des outils de versionnement les plus utilisés dans le développement logiciel.

Git est donc essentiel pour :

- Gérer l'historique des versions de manière efficace,
- Faciliter la collaboration et la communication au sein des équipes de développement,
- Expérimenter sans risque grâce aux branches,
- Améliorer la qualité du code par des revues et des tests automatisés,
- Garantir la sécurité et la sauvegarde du code grâce à la décentralisation.

2. *Quelle est la différence entre git add et git commit ?*

git add et **git commit** sont deux commandes essentielles dans le workflow de Git, mais elles remplissent des rôles différents dans le processus de gestion des versions. Voici un aperçu de leurs fonctions et de leurs différences :

- La commande **git add** ajoute les modifications de fichiers dans l'index (également appelé "**staging area**"). Cela signifie que vous préparez ces modifications pour le prochain commit. Il utilise la syntaxe **git add <nom_du_fichier>** pour ajouter un fichier spécifique. Donc **git add .** lui il ajoute tous les fichiers modifiés dans le répertoire courant et ses sous-répertoires. L'objectif de git add est de marquer les changements dans les fichiers que vous voulez inclure dans votre prochain commit. Notez que cette commande ne crée pas encore de commit, elle se contente de préparer les fichiers.

La commande git commit crée un commit avec toutes les modifications qui ont été ajoutées à l'index par git add. Un commit enregistre un instantané de l'état actuel de l'index. Il utilise la syntaxe **git commit -m "Message de commit"** pour Créer un commit avec le message spécifié. Son objectif est d'enregistré les modifications dans l'historique du dépôt. Chaque commit crée une nouvelle version du projet.

Exemple

Voici un exemple concret pour mieux comprendre :

Modification d'un fichier : Vous modifiez **index.html** et **style.css**.

Ajout des modifications :

```
git add index.html
git add style.css
```

Création d'un commit :

```
git commit -m "Mise à jour de la page d'accueil et du styl
```

En résumé, git add prépare les modifications pour un commit, tandis que git commit crée effectivement un commit avec ces modifications. Utilisés ensemble, ils permettent de suivre efficacement l'historique des versions de votre projet.

3. *Qu'est-ce qu'un dépôt (repository) GitHub ?*

Un dépôt (repository) GitHub est un espace centralisé où vous pouvez stocker et gérer les fichiers de votre projet. C'est une pierre angulaire pour la collaboration sur le code, la gestion de versions et le suivi des modifications. Voici quelques points clés sur les dépôts GitHub :

- **Stockage des fichiers :** Un dépôt contient tous les fichiers de votre projet, y compris le code source, les images, les documents, etc.

- **Gestion des versions :** GitHub utilise Git, un système de contrôle de version distribué, pour suivre l'historique des modifications de vos fichiers. Vous pouvez revenir à des versions antérieures du projet si nécessaire.
-
- **Collaboration :** Les dépôts permettent à plusieurs développeurs de travailler ensemble sur le même projet. Ils peuvent cloner (copier) le dépôt, apporter des modifications et proposer leurs changements via des pull requests.

- **Branches :** Vous pouvez créer des branches dans un dépôt pour travailler sur différentes fonctionnalités ou correctifs de manière isolée. Cela permet de développer et de tester des modifications sans affecter la version principale (main ou master) du projet.

- **Issues et Projets :** GitHub propose des outils pour suivre les problèmes (issues), gérer les tâches et organiser le travail avec des tableaux de projet.

- **Documentation :** Vous pouvez ajouter des fichiers de documentation comme le README.md pour fournir des informations sur votre projet, ainsi que des wikis et des pages GitHub pour une documentation plus complète.

Exemple d'un dépôt GitHub

Vous pouvez créer un nouveau dépôt sur **GitHub** via l'interface web en cliquant sur "**New repository**", en donnant un nom et en choisissant si le dépôt doit être public ou privé. Pour copier un dépôt sur votre machine locale, utilisez la commande :

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

Ajoutez des fichiers à votre dépôt local et suivez les étapes **git add** et **git commit** pour les préparer et enregistrer les modifications. Et utilisez la commande **git push** pour envoyer vos modifications au dépôt distant sur GitHub.

4. *Comment cloner un dépôt GitHub en local ?*

Pour cloner un dépôt GitHub en local, il faut s'assurer que Git est installé sur votre ordinateur. Se rendre sur la page du dépôt GitHub que vous souhaitez cloner. Cliquer sur le bouton **"Code**" puis copiez l'URL du dépôt. L'URL peut être HTTP(S) ou SSH, selon votre préférence. Ouvrir le terminal (ou l'invite de commandes sur Windows) sur votre ordinateur et utiliser la commande **git clone** suivie de l'URL que vous avez copiée. Exemple :

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

Après l'exécution de la commande, un nouveau répertoire portant le nom du dépôt sera créé. Accédez à ce répertoire avec la commande cd :

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

C'est tout ! Vous avez maintenant cloné le dépôt GitHub en local et pouvez commencer à travailler dessus.

5. *Comment faire un "push" de vos modifications sur GitHub ?*

Pour envoyer vos modifications locales vers un dépôt distant sur GitHub (ce qu'on appelle faire un "push"), il faut s'assurer que le dépôt local est à jour avec les dernières modifications du dépôt distant. Vous pouvez faire cela en utilisant :

```
git pull origin main
```

(Remplacez **main** par le nom de la branche principale si elle diffère, comme **master**.). Utiliser la commande **git add** pour préparer les fichiers modifiés :

```
git add .
```

Ou pour un fichier spécifique :

```
git add nom-du-fichier
```

Et une fois que vos fichiers sont ajoutés à l'index, créez un **commit** avec un message descriptif :

```
git commit -m "Votre message de commit"
```

Pour enfin utilisez la commande **git push** pour envoyer vos modifications vers GitHub :

```
git push origin main
```

Exemple complet

Imaginons qu’on veut modifier un fichier **index.html** et qu’on souhaite pousser ces modifications :

- Ajouter le fichier modifié :

```
git add index.html
```

- Créer un commit :

```
git commit -m "Mise à jour du fichier index.html avec le n
```

- Faire un push vers GitHub :

```
git push origin main
```

Si c'est la première fois qu’on pousse des modifications vers GitHub, il se peut qu’on doit entrer nos informations de connexion ou configurer une clé SSH.

Si le dépôt distant a des modifications qui ne sont pas présentes dans notre dépôt local, on doit peut-être résoudre des conflits avant de pouvoir pousser nos modifications.