

# Exercices - Git

Pour ces exercices, en plus de taper dans le terminal vos commandes, veuillez me rendre le répertoire de votre dépôt zipé.

Vous m'enverrez via WhatsApp vos exercices.

## Exercice n°1 - Creation / indexation / validation

Le nom du fichier zip aura la structure suivante **prenom\_ex1.zip**. Par exemples: **johnny\_ex1.zip** ou encore **bruno\_ex1.zip**.

### 1.1 - Création d'un dépôt

1. Créez un répertoire nommé PremierDepot
2. Faites-en un dépôt à l'aide d'une commande git bien entendu.
3. Quelle commande utiliser pour vérifier que le répertoire est bien devenu un dépôt ?

### 1.2 - Modification (Ajouter de fichiers)

Pour éviter des alertes de sécurité de Chrome, j'ai nommé les fichiers avec l'extension .txt. Nous les renommerons plus tard avec la bonne extension.

1. Dans le répertoire que vous venez de créer, copiez les 3 fichiers suivants:

- [index.txt](#)
- [genius.txt](#)
- [display.txt](#)

2. Faites un git status. (Que vous dit/signale git ?)

### 1.3 - Indexation

1. Indexez le fichier index.txt avec git.
2. Indexez les autres fichiers.
3. Vérifiez avec la bonne commande git si les 2 indexations se sont bien passées. (Comment le voyez-vous ?)

### 1.4 - Validation/Commit

1. Validez les fichiers présents dans la zone d'index avec le message suivant: "Commit initial" (C'est souvent le premier message que l'on donne pour le tout premier commit)
2. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
3. Tapez la commande git tag v1 (Je vous expliquerai plus tard dans le cours l'utilité de git tag. Donc, comme un robot, exécutez les "ordres" et tapez git tag v1 :-))

## 1.5 - Affichage de l'historique des commits

Affichez l'historique des commits à l'aide de la commande: git log

Vous verrez l'ensemble des commits effectués avec la date, l'heure, l'auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

### 1.6 - Modifications (Renommer des fichiers)

Ici les modifications vont porter sur le fait que l'on va renommer nos fichiers.

Pour rappel, voilà comment renommer un fichier en ligne de commandes:

1. Windows: ren fichier1 fichier2 (va renommer le fichier1 avec le nom fichier2)
2. Mac Os/Linux: mv fichier1 fichier2 (va renommer le fichier1 avec le nom fichier2)

Renommez les fichiers de la manière suivante:

1. index.txt devient index.html
2. genius.txt devient genius.js
3. display.txt devient display.js
4. Évidemment vérifiez avec dir (windows) ou ls (mac/linux) s'ils sont bien renommés.

### 1.7 - Indexation

1. Faites un git status
2. Indexez les fichiers modifiés.

3. Refaites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)

## 1.8 - Validation/Commit

1. Faites un git status
2. Validez/Commitez vos fichiers en ajoutant comme message "Renommage d'index, genius et display"
3. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
4. Tapez la commande git tag v2

## 1.9 - Affichage de l'historique des commits

Affichez l'historique des commits à l'aide de la commande: git log

Vous verrez l'ensemble des commits effectués avec la date, l'heure, l'auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

### 1.10 - Modification (de code)

Explication des différents fichiers:

- index.html: il sert à afficher une page qui affiche le résultat de calculs.
- genius.js: c'est le fichier qui contient nos fonctions javascript de mathématique: add et subtract.
- display.js: il contient une fonction display qui permet d'afficher le résultat d'un calcul à l'écran. Vous ne devez pas y toucher !

L'exercice:

Vous devrez ajouter une fonction qui multiplie deux nombres dans genius.js et l'utiliser dans index.html. Le code devra être fonctionnel. C'est à dire qu'à l'affichage dans index.html, votre fonction calcule bien la multiplication de deux nombres et affiche à l'écran le résultat. Donc index.html permettra de vérifier à l'affichage de la page web si votre code est bon.

1. Editez le fichier genius.js
2. Ajoutez la fonction multiply qui retourne (return) la multiplication des deux nombres donnés en paramètre dans la signature de la fonction multiply.
3. Editez le fichier index.html
4. Appelez la fonction display pour qu'elle affiche la multiplication de a par b ( $a * b$ ).
5. Pour l'utilisation de display dans index.html, inspirez-vous des deux lignes précédentes qui font la soustraction et l'addition de a et b.
6. Testez votre programme en lançant index.html et voyez le résultat.
7. Si votre programme fonctionne allez au point suivant sinon cherchez votre erreur.
8. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)

### 1.11 - Indexation

1. Faites un git status
2. Indexez les fichiers modifiés.
3. Refaites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)

### 1.12 - Validation

1. Faites un git status
2. Validez/Commitez vos fichiers en ajoutant comme message "Ajout de la fonction multiply et adaptation du fichier index.html"
3. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
4. Tapez la commande git tag v3

## 1.13 - Affichage de l'historique des commits

Affichez l'historique des commits à l'aide de la commande: git log

Vous verrez l'ensemble des commits effectués avec la date, l'heure, l'auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

### 1.14 - Modification (de code)

Vous devrez ajouter une fonction qui divise deux nombres dans genius.js et l'utiliser dans index.html. Le code devra être fonctionnel. C'est à dire qu'à l'affichage dans index.html, votre fonction calcule bien la division de deux nombres et affiche à l'écran le résultat. Donc index.html permettra de vérifier à l'affichage de la page web si votre code est bon.

1. Editez le fichier genius.js
2. Ajoutez la fonction divide qui retourne (return) la division des deux nombres donnés en paramètre dans la signature

de la fonction `divide`. N'oubliez pas que l'on ne peut diviser par 0 en mathématiques. Donc il faudra tester que le diviseur est bien différent de 0 avant de retourner le quotient.

3. Editez le fichier `index.html`
4. Appelez la fonction `display` pour qu'elle affiche la division de `a` par `b` (`a / b`).
5. Pour l'utilisation de `display` dans `index.html`, inspirez-vous des trois lignes précédentes qui font la soustraction, l'addition et la multiplication de `a` et `b`.
6. Testez votre programme en lançant `index.html` et voyez le résultat.
7. Si votre programme fonctionne allez au point suivant sinon cherchez votre erreur.
8. Faites un `git status` (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)

## 1.15 - Indexation

1. Faites un `git status`
2. Indexez les fichiers modifiés.
3. Refaites un `git status` (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)

## 1.16 - Validation

1. Faites un `git status`
2. Validez/Commitez vos fichiers en ajoutant comme message "Ajout de la fonction `divide` et adaptation du fichier `index.html`"
3. Faites un `git status` (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
4. Tapez la commande `git tag v4`

## 1.17 - Affichage de l'historique des commits

Affichez l'historique des commits à l'aide de la commande: `git log`

Vous verrez l'ensemble des commits effectués avec la date, l'heure, l'auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

## Exercice n°2 - Creation / indexation / validation

### 2.1 - Création d'un dépôt

1. Créez un répertoire nommé `Recettes`
2. Faites-en un dépôt à l'aide d'une commande `git` bien entendu.
3. Quelle commande utiliser pour vérifier que le répertoire est bien devenu un dépôt ?

### 2.2 - Création d'une recette

1. Allez dans le répertoire `Recettes`
2. Copiez le fichier [omelette.txt](#)

30g de gruyère râpé