1 Git

1.1 Faire une copie locale des démos associées au cours de JS & Node

Veuillez faire une copie locale du projet associé au cours.
Le repository se trouve à cette URL : https://github.com/e-vinci/js-demos.git

1.2 Créer un repository local et distant pour gérer vos exercices

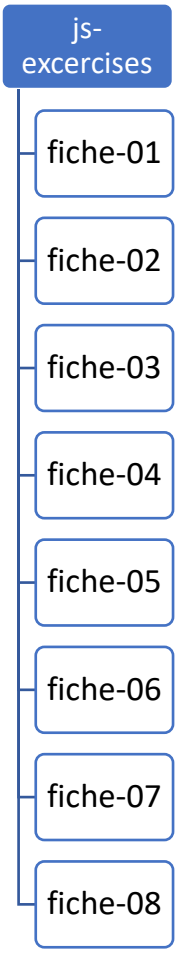
Veuillez créer un repository local qui reprendra tous les exercices que vous allez réaliser.

Ce repository doit reprendre la structure de répertoires que vous allez mettre en place pour couvrir les 8 fiches d'instructions associées aux exercices.

Voici comment nous souhaitons que vous organisiez cette structure de répertoires :

- Nom de votre répertoire de projet reprenant tous les exercices :
  js-excercises
  NB : c'est le répertoire qui va contenir votre repository.

- Structure de dossiers



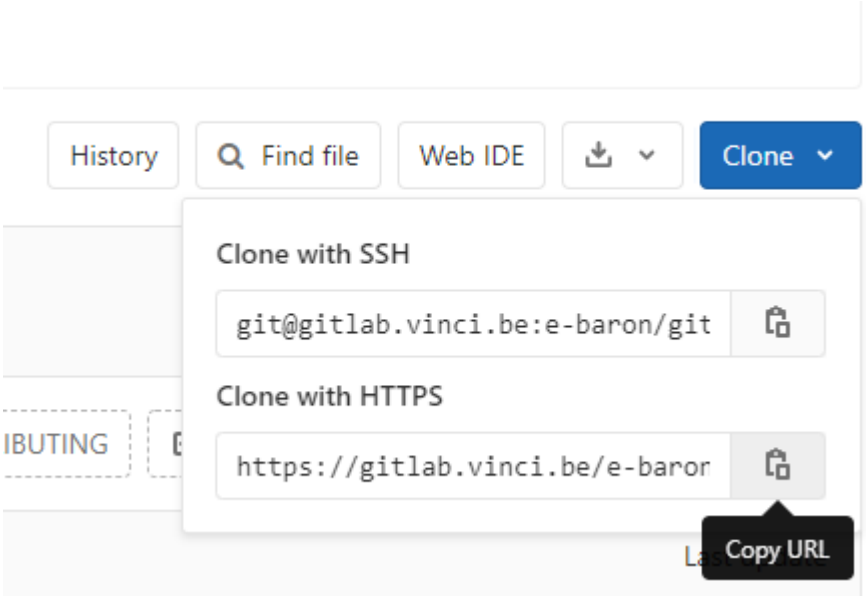
Veuillez créer un web repo « private » sur GitLab portant le nom **js-excercises**.
Pour ce faire, pour les étudiants de Vinci, veuillez :

- créer votre Gitlab repository dans le Gitlab offert par Vinci. Pour vous logger sur la plateforme Gitlab de Vinci, accédez à https://gitlab.vinci.be/ , puis cliquez sur **Sign in with Azure AD**
- créer un projet « **private** » sur GitLab portant le nom **js-excercises**.
- ajouter comme membre les deux enseignants associés au cours (rôle de « Developer ») :
    - Sébastien Strebelle
    - e-Baron (Raphael Baroni)

  NB : pour ce faire, dans l'interface web de GitLab : **Settings**, **Members**, **Invite member**, **Choose a role permission** : **Developer**

Pour les étudiants de Vinci :

- veuillez soumettre le lien vers votre web repository (qqch du genre https://gitlab.vinci.be/votre_username/JS-Node-Exercices.git) d'exercices via le devoir associé sous Moodle.
  Pour trouver le lien vers votre repo, il vous suffit de cliquer sur **Clone** dans votre projet sous Gitlab, et de prendre l'info associée à **Clone with HTTPS**



NB : Lorsque vous allez ajouter une **remote** pour vous synchroniser avec votre web repository, nous vous conseillons d'utiliser l'URL associée à « Clone with HTTPS ». Si vous souhaitez malgré tout vous connecter en SSH, vous pouvez consulter cet article : https://dev.to/sndrx/how-to-set-up-an-ssh-key-and-use-it-in-gitlab--42p1

2 DOM & la gestion d'événements

2.1 1er script externe & appel de fonction

Via un script JS externe, veuillez créer une application web affichant, au chargement d'une page HTML, un pop-up dont le message reprend la date et l'heure au moment de l'affichage de ce pop-up.

Ce même message doit être affiché dans la console.

Pour ce faire, vous allez créer, puis appeler, une fonction **addDateTime(message)** qui renvoie une string concaténant :

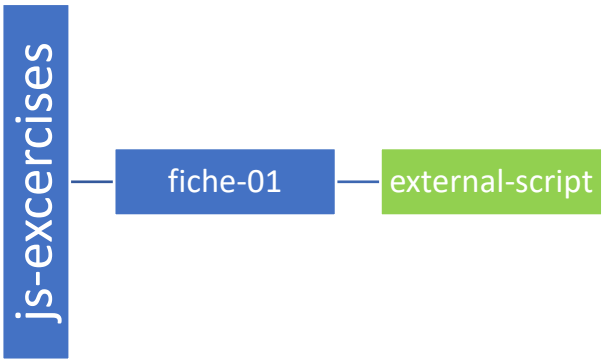
- La date et l'heure à l'appel de la fonction
- Le message passé en argument de la fonction

Vous pouvez vous aider du code donné ci-dessous :

```
const dateTimeNow = new Date();
console.log(dateTimeNow.toLocaleDateString()); // 17/08/2020
console.log(dateTimeNow.toLocaleTimeString()); // 13:26:15
```

Autre référence pouvant vous aider : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

Le code de votre application web doit se retrouver dans ce dossier (en vert) de votre repository local et de votre web repository (**js-excercises**).



2.2 Gestion d'un compteur de clics

Veuillez réaliser une application web permettant d'afficher un compteur de clic sur votre page web.

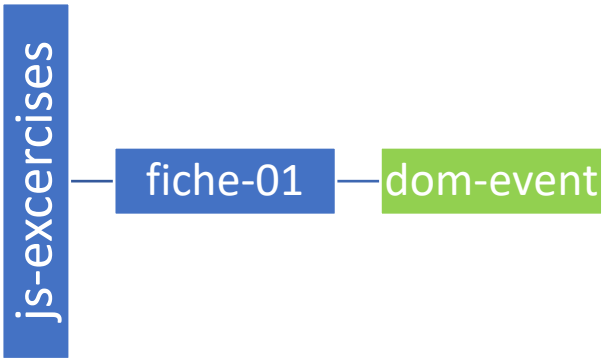
Affichez le message suivant sur votre page web :

- Au 5ème clic : « Bravo, bel échauffement ! »
- Au 10ème clic : « Vous êtes passé maître en l'art du clic ! »

N'oubliez pas, bien sûr, d'afficher le compteur de clic sur votre page.

- *Vous pouvez utiliser la propriété **.innerHtml** ou **textContent** d'un paragraphe **<p>** ou d'un **** pour afficher votre compteur.*
- *Idem pour votre message.*
- *Vous pouvez ajouter un gestionnaire d'événement sur l'objet **window** de votre navigateur.*

Le code de votre application web doit se retrouver dans ce dossier (en vert) de votre repository local et de votre web repository (**js-excercises**).



2.3 Gestion d'événements différés dans le temps : l'horloge

Veuillez réaliser une application web permettant d'afficher une horloge digitale sur votre page web.
A chaque seconde, veuillez mettre à jour l'heure actuelle.

Lors du clic sur l'horloge, veuillez stopper ou réactiver son rafraîchissement.

En résumé :

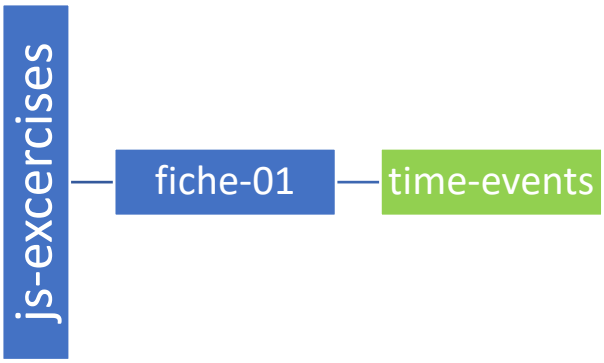
- pas de clic sur l'horloge, elle se met à jour chaque seconde.
- 1er clic, 3ème clic, … : elle se stoppe sur l'heure en cours ;
- 2ème clic, 4ème clic,… : le rafraîchissement de l'horloge se fait chaque seconde ;

- *Vous pouvez vous aider du code donné ci-dessous :*

```
const dateTimeNow = new Date();
console.log(dateTimeNow.toLocaleTimeString()); // 13:26:15
```

- *Vous pouvez utiliser la propriété **.innerHtml** ou **textContent** d'un paragraphe **<p>** ,ou d'un **** , ou d'une div, ou…, pour afficher votre horloge.*
- *Nous vous recommandons la dernière slide de [05-DOM-Events] (**setInterval** et **clearInterval**) pour lancer et stopper des appels récurrents de méthodes.*

Le code de votre application web doit se retrouver dans ce dossier (en vert) de votre repository local et de votre web repository (**js-excercises**).



2.4 Exercice optionnel

N'hésitez pas à libérer votre créativité et approfondir la gestion des événements.
Vous pourriez compléter l'exercice fait au §2.2 en gérant d'autres événements que le clic.
Voici quelques idées :

- Affichez des informations cachées de votre page lorsque l'utilisateur passe la souris sur une partie bien précise de la page : par exemple, apparition d'une image quand la souris se trouve sur sa zone d'affichage, disparition lorsqu'elle en sort (ou ne s'y trouve pas).
- Réalisez une action spécifique lors d'un clic droit sur votre page…
- C'est votre application web, qu'est ce qui serait chouette à ajouter ? ;)
- …