Programmation Web

Introduction : Java Script

Le JS est un langage de programmation évènementiel. Il permet de faire des applications web, ce qui permet d'attendre par exemple le clic d'un utilisateur pour une action. Le code ne va donc pas s'exécuter de manière séquentielle (d'une ligne à l'autre). Il y a une interaction entre **Browser (front) et Web Serveur (back)**. Avec des requêtes dans le protocole http, https.

Push : lorsqu'on veut envoyer des informations sans que ça soit demandé par le Browser. (Exemple de site de news). Cependant le protocole *http* ne permet pas de le faire. Donc il faut faire une requête qui demande s'il y a des news, le problème d'une telle solution est qu'un temps de latence se créer pour rien si aucunes news n'existent dans le back. Donc changement de protocole pour opter pour WS.

➔La méthode par défaut est une requête GET, ce qui peut être changé grâce à JS (grâce à un FETCH API).

Il est possible de dialoguer avec le serveur ainsi que modifier le DOM (Dialog Object Model).

➔Le browser est capable d'exécuter le JS.

Base du langage JS

Algorithme : lire, traiter et afficher de l'information.

Rappel :

Mémoire morte = ne s'efface pas, utile pour les bases de données par exemple.

Mémoire vive = s'efface, les variables par exemple.

Variable :

- let avec = (comme affectation).
- 4 types de base.

Formats :

1. Séquence.
2. If (condition testée à vraie).
3. Itération (répéter une action x nombre de fois).
4. Fonction. But aussi de documenter grâce au nom de la fonction.

Fonction :

Anonyme ➔ pour répondre une fois à un évènement. Utilisée qu'une seule fois dans le code.

Constante informatique : change entre les exécutions. N'a pas la même signification qu'une constante mathématique.

Il est possible de vouloir afficher une fonction sans avoir rien rempli en paramètre alors que cette dernière en demande normalement deux. Le terminal le permet, mais mauvais réflex. Permet de créer des fonctions avec des nombres de paramètres variables.

En informatique il n'y a pas de calcul précis. Adition de deux réels est donc faux, car il ajoute des chiffres après la virgule. C'est pour ça qu'il est utile de savoir arrondir ou afficher que les deux premiers caractères d'un calcul.

Plusieurs manières d'écrire des fonctions :

const add3 = (a, b) => a + b ; //fonction fléché. Le return est implicite.

Seules les libraires commencent par une majuscule. (Exemple libraire Math.random).

- Les fonctions sont un sous-ensemble des objets.

Penser à bien documenter et nommer correctement les variables, car l'éditeur de code permet (exemple exercice random) de donner le nom des paramètres quand on appelle une fonction,

Boucle for

Construction d'une boucle en JS.

```
for (begin; condition; step) {
// ... loop body ...
}
```

Exemple :

```
for(let i = 0; i < 3; i++){

}
```

Objet

Tout comme en Java, il est possible de faire des objets. (Même signification quand Java, l'entité). Ce qui n'est pas égal à la classe, qui elle est le constructeur.

Exemple :

```
let objet = {Min : 1 , Foo : function(){}};
```

Tableau

Le tableau est une constante. Donc utilisation de const arr = [] ;

- Le tableau est un élément dynamique.

Tableau de nombre random

Si on veut faire un tableau de nombre random, il faut définir la taille, et ensuite ce avec quoi on veut remplir les cases de notre tableau.

Donc c'est dans cette deuxième partie qu'on va pouvoir faire appel à notre fonction préalablement faite pour remplir les cases.

➔Délégation de fonctions. Différence entre le fait d'appeler une fonction ou alors de déléguer.

Possibilité de trouver une valeur grâce à .find., ou findIndex Normalement besoin de trois paramètres mais comme c'est du JS, il est possible d'en donner une seule.

Principe de déconstruction

Pour retourner deux éléments, ou plus d'un tableau, il est possible de retourner un objet qu'on va par la suite pouvoir déstructurer.

Exemple :

```
return{tails, heads : nbTry-tails};
const {tails, heads} = headsTails(100);
```

Curryfication

Transforme une fonction à plusieurs arguments en une fonction à un argument. Cette dernière retourne une fonction sur le reste des arguments. Cela permet d'utiliser une fonction antérieure comme argument.

Méthode : Object.freeze()

Permet de rendre « intouchable un objet ». On ne peut pas ajouter de nouvelles propriétés à l'objet, ni supprimer, ni éditer des propriétés existantes. La méthode renvoie l'objet gelé.

Exemple :

```
const obj = {prop : 42}; //déclaration de l'objet avec un paramètre
Object.freeze(obj); //utilisation de la méthode pour freezer l'objet
console.log(obj.prop);
```

Méthode : Map()

Elle créer un nouveau tableau avec les résultats de l'appel d'une fonction fournie sur chaque élément du tableau appelant.

```
1 const array1 = [1, 4, 9, 16];
2
3 // pass a function to map
4 const map1 = array1.map(x => x * 2);
5
6 console.log(map1);
7 // expected output: Array [2, 8, 18, 32]
8
```

Méthode : filter()

Créer et retourne un nouveau tableau qui contiens tous les éléments du tableau d'origine qui remplissent une condition déterminée par la fonction *callback*.

Explication : la méthode appelle la fonction *callback* fournie pour tous les éléments.

Méthode : slice()

S'utilise directement dans le console.log, il suffit d'indiquer l'intervalle du tableau de base qu'on souhaite. Cette méthode renvoie un objet tableau qui contient une copie superficielle de la portion choisie du tableau d'origine. (Indice de début et indice de fin). Le tableau original n'est pas modifié puisque c'est seulement une copie spéciale.

Méthode : reduce()

Applique une fonction sur un accumulateur et chaque valeur du tableau typé (de la gauche à la droite) afin de réduire le tableau en une seule valeur.

```
1 const uint8 = new Uint8Array([0, 1, 2, 3]);
2
3 function sum(previousValue, currentValue) {
4   return previousValue + currentValue;
5 }
6
7 console.log(uint8.reduce(sum));
8 // expected output: 6
9
```

Méthode : push()

Cette méthode ajoute un ou plusieurs éléments à la fin d'un tableau et retourne la nouvelle taille du tableau.

```
const animals = ['pigs', 'goats', 'sheep'];

const count = animals.push('cows');
console.log(count);
// expected output: 4
console.log(animals);
// expected output: Array ["pigs", "goats", "sheep", "cows"]
```

Méthode : unshift()

Cette méthode ajoute un ou plusieurs éléments au début du tableau et renvoie la nouvelle longueur du tableau.

S'utilise de la même manière que la méthode push().

Méthode : join()

Cette méthode permet de créer et de renvoyer **une chaîne de caractères** en concaténant tous les éléments d'un **tableau**. Pour pouvoir faire la concaténation il faut un séparateur qu'on fournit en argument (dans les paramètres de la méthode) le caractère (virgule, chaîne ou autre).

Méthode : toLowerCase()

Cette méthode retourne une chaîne de caractère en minuscule.

Méthode : split()

Divise une **chaîne de caractères** en une liste ordonnée de sous-chaînes, ces sous-chaînes sont ensuite placées dans un tableau qui est retourné. La division de ces sous-chaînes est effectuée en recherchant un motif (ce motif est fourni comme premier paramètre dans l'appel de la méthode).

```
const str = 'The quick brown fox jumps over the lazy dog.';

const words = str.split(' ');
console.log(words[3]);
// expected output: "fox"

const chars = str.split('');
console.log(chars[8]);
// expected output: "k"

const strCopy = str.split();
console.log(strCopy);
// expected output: Array ["The quick brown fox jumps over the lazy dog."]
```

Dans les deux dernières lignes de l'exemple, la string est placé dans un tableau qui contient une case avec le texte.

Méthode : reverse()

Cette méthode permet de présenter un tableau dans un autre contexte. Donc ici, les éléments du tableau sont présentés dans un autre ordre. Le premier élément devient le dernier et le dernier devient le premier.

Extension Live Server :

Permet d'ouvrir un fichier html dans un browser. C'est une simulation d'un serveur puisque le fichier s'ouvre en local (repérable dans l'adresse).

Le script est construit progressivement. (Exemple du null quand le script est dans le head mais s'affiche quand il est à la fin du document).

Donc possible de créer un fichier externe au code html pour pouvoir dire qu'il faut attendre que le dom soit prêt pour exécuter le code qui se trouve dans le fichier externe.

- Pour pouvoir atteindre un élément (par exemple un titre h1) il faut utiliser le CSS :

const titre document.querySelector('h1')//qui permet de sélectionner un élément.

Méthode : querySelector

La méthode de l'interface Document retourne le premier élément dans le document qui correspond au sélecteur (ou groupe de sélécteurs) ou null si aucune correspondance n'est trouvée.

Interface Document = n'importe quelle page Web chargée dans le navigateur et sert de point d'entrée dans le contenu de la page Web. Cette interface hérite aussi des interfaves Node et EvenTarget.

⇨ Possible d'accéder par couche pour spécifier un élément précis dans la page html.

Méthode : getElementById()

Possible d'utiliser cette méthode pour trouver par identifiant.

Méthode : querySelector()

Trouver par attributs.

Méthode : EventTarget.addEventListener()

La méthode attache une fonction à appeler à chaque fois que l'évènement spécifié est envoyé à la cible.