DEVELOPPEMENT D'APPLICATION WEB

WEB – Introduction à JavaScript

OBJECTIFS

A travers cet exercice, vous allez :

- Découvrir le langage de programmation JavaScript,
- Manipuler le Document Object Model (DOM),
- Manipuler un formulaire simple.

PREMIERS PAS AVEC JAVASCRIPT

EXERCICE

- Créez un dossier pour votre projet et ouvrez-le avec Visual Studio Code
- Créez un fichier **`index.html`** contenant une structure HTML de base.

> **Astuce** : lorsque vous tapez « **`html`** » dans un fichier HTML vide, Visual Studio Code vous propose le snippet (modèle de code) **`html:5`**. En le sélectionnant, vous obtenez une structure HTML de base.

- Créez un fichier **`script.js`**
- Ajoutez la balise **`<script>`** suivante dans le **`body`** de votre page HTML :

```
<script src="script.js"></script>
```

CONSOLE

Au même titre qu'un programme en C++ ou en Java peut utiliser le terminal pour afficher des informations avec **`std::cout`** ou **`System.out`**, un programme JavaScript dispose d'une console dans laquelle il va pouvoir écrire du texte.

Bien sûr, dans un navigateur Web, les informations dédiées à l'utilisateur se trouve au niveau de la page Web. Mais, la console reste très utile pour fournir des informations complémentaires aux développeurs en cas d'erreurs ou pour du débogage.

EXERCICE

- Dans le fichier **`script.js`** ajoutez la ligne ci-dessous :

```
console.log("Polytech Dijon !");
```

> **Note** : en JavaScript les ; sont facultatifs. A vous de voir si vous souhaitez les mettre ou non. Le tout est de ne pas perdre l'habitude de les placer dans les autres langages (Java, C, C++, PHP, ...).

- Ouvrez votre page HTML dans un navigateur
- Ouvrez les outils de développement et rendez-vous sur l'onglet « Console »
- Vous devriez y trouver le message « Polytech Dijon ! »

VARIABLES

En JavaScript, le type d'une variable n'est pas défini au moment de sa déclaration mais à celui de l'affectation. Ainsi une variable peut être initialisée avec un entier, puis se voir affecter une chaine de caractères et enfin le perfecto de Monsieur Chassel. Pour JavaScript, cela ne pose aucun problème (pour Monsieur Chassel, un peu plus, surtout avec le temps Dijonnais).

De fait, seul deux mots clés sont nécessaires :

- **`let`** pour déclarer une variable (dont la valeur peut varier)
- **`const`** pour déclarer une constante (dont la valeur ne peut pas varier)

EXERCICE

- Dans le fichier **`script.js`**, Déclarez une variable **`area`** initialisée avec la valeur **`"VIDE"`**.
- Déclarez deux constantes **`width`** et **`height`** initialisées avec les valeurs **`1920`** et **`1080`**.
- Affectez le produit de **`width`** et **`height`** à la variable area.
- Affichez dans la console la valeur de la variable **`area`**.
- Affectez la valeur **`1200`** à la constante **`height`**.
- Affichez dans la console la valeur de la constante **`height`**.
- Actualisez la page de votre navigateur et observez la console.

Le code JavaScript est interprété au fur et à mesure de l'exécution de son programme, là où le C++ est compilé avant son exécution. Ainsi, le compilateur C++ détecte les erreurs de programmation comme la modification de la valeur d'une constante et stoppe directement la production de l'exécutable. JavaScript lui déroule le code tant qu'il n'y a pas de problème et s'arrête dans le cas contraire.

Ici, la modification de la constante lève une exception de type **`TypeError`**.

Bonnes pratiques : bien que JavaScript permette de faire varier le type d'une variable au cours de sa durée de vie, il est fortement déconseillé de le faire. Vous n'aimeriez pas trouver une grenouille dans votre pot de Nutella, n'est-ce pas ? Et bien nous n'aimons pas trouver une chaine de caractères dans ce qui était censé contenir un nombre.

Prenez également l'habitude de nommer une variable / constante en fonction de ce qu'elle contient. Fini les a, b, c, ... qui n'ont aucun sens... à part peut-être en math.

Astuce : actualiser la page de votre navigateur après chaque changement peut vitre être lassant. L'extension « Live Server » disponible sur VS Code permet de recharger automatiquement la page de votre navigateur sitôt que vous enregistrez l'un des fichiers de votre site Web (HTML, CSS, JS, ...). Une fois l'extension installée, ouvrez votre fichier HTML sous VS Code et cliquez sur le lien « Go live » situé en bas à droite de votre écran.

TRY / CATCH

Il est possible de capturer les exceptions d'un programme avec des blocs **`try/catch`** :

```
try {
    // Portion de code susceptible de lever une exception
}
catch(exception) {
    // Traitement de l'exception passée en paramètre
}
```

EXERCICE

- Capturez l'exception produite lors de la tentative de modification de la constante **`height`**.
- Affichez un message d'erreur de votre choix puis le contenu du paramètre passé à **`catch`**.

TABLEAUX

Voici la syntaxe pour créer un tableau en JavaScript :

```
const empty_array = [];                 // Création d'un tableau vide
const filled_array = [1, 2, 3, 4];      // Création d'un tableau pré-rempli.
```

Les tableaux JavaScript ont une taille dynamique, c'est à dire que vous pouvez ajouter et retirer des éléments à loisir. Par ailleurs, un tableau peut contenir des éléments de types différents :

```
const array = [87, "étudiants"];
```

EXERCICE

- Créez un tableau vide nommé **`array`**.
- Ajoutez un entier dans le tableau (**`array.push(valeur_a_ajouter)`**).
- Ajoutez une chaine de caractères au tableau.
- Affichez dans la console le contenue de votre tableau (**`console.log(array)`**).

Si vous avez respectez la syntaxe donnée précédemment, vous avez déclarez un tableau avec le mot clé **`const`**. Pourtant votre tableau a pu être modifié puisque vous avez ajoutez des valeurs à ce dernier.

- Comment expliquez-vous cela ?

TESTS

Puisqu'il semble que vous ayez des difficultés avec les syntaxes révolutionnaires des derniers langages étudiés, voici celle de ce bon vieil if en JavaScript :

```
if(condition)
{
    // todo
}
else if(condition_suivante)
{
    // todo
}
else
{
    // todo
}
```

EXERCICE

- Créez une variable **notTrue** initialisée à **false**.
- Créez une variable **zero** initialisée à **0**.
- Si **noTrue == zero** alors affichez dans la console **"Elles sont égales"**, sinon affichez dans la console **"Elles ne sont pas égales"**.
- Testez

En JavaScript, l'opérateur **==** est un opérateur d'égalité non stricte. C'est à dire qu'il compare les valeurs des opérandes après avoir harmonisé leur type. Ainsi, lorsque vous comparez un entier et un booléen, l'opérateur **==** va commencer par transformer l'entier en booléen (**0** devient **false**) puis comparer les valeurs (**false == false** => **true**).

Pour réaliser des comparaisons strictes, vous devez utiliser l'opérateur **===** qui commence par vérifier que les types des opérandes sont identiques et seulement dans ce cas compare les valeurs.

EXERCICE

- Modifiez le code précédent pour réaliser le test avec l'opérateur **===**.
- Testez.

Bonne pratique : lorsque vous réalisez des tests d'égalité en JavaScript, prenez l'habitude d'utiliser l'opérateur **===**. L'opérateur **==** a ses avantages dans certains cas mais est problématique la plupart du temps.

BOUCLES

Il existe différentes syntaxes pour effectuer des boucles en JavaScript :

FOR

La version la plus connue :

```
for(let i = 0 ; i < condition_de_continuité ; ++i)
{
    // todo
}
```

Parcourir les éléments d'un tableau :

```
const array = [1, 2, 3, 4] ;

for(const integer of array)
{
    // todo
}
```

WHILE

```
while(condition_de_continuité)
{
    // todo
}
```

DO / WHILE

```
do
{
    // todo
}while(condition_de_continuité)
```

EXERCICE

- Créez un tableau vide.
- Ajoutez 30 valeurs aléatoires dans le tableau (**`Math.random()`**).
- Affichez le contenu du tableau dans la console.

FONCTIONS

Vous allez enfin pouvoir structurer quelque peu votre code avec des fonctions !

```
function nom_de_la_fonction(paramètre1, paramètre2, ...)
{
    // todo
}
```

Vous pouvez constater que la valeur de retour n'est pas précisée en JavaScript.

EXERCICE

- Réalisez une fonction qui prend en paramètre un tableau de nombres et retourne la moyenne de ce tableau (**`return`**).
- Affichez dans la console la valeur renvoyée par votre fonction.

DOCUMENT OBJECT MODEL

INTRODUCTION

Lorsque votre navigateur interprète le code HTML de votre page, il produit une structure arborescente d'objets en mémoire, correspondant à l'arborescence des balises HTML qui se trouvent dans votre document.

Cette arborescence est le Document Object Model ou DOM. Il est possible d'interagir avec le DOM via JavaScript.

EXERCICE

- Créez un nouveau projet avec un fichier HTML et un script JavaScript vide.
- Créez une fonction **`helloWorld`** qui contiendra le code suivant :

```
document.body.innerHTML = "<h1>Hello World !</h1>";
```

- Appelez la fonction **`helloWorld`**
- Observez la page dans votre navigateur.

Notes : JavaScript est un langage orienté objet. Le contexte du navigateur offre plusieurs classes dont certaines sont instanciées de manière globale.

- **`document`** est une instance de la classe **`Document`** qui est la racine du DOM.
- **`body`** est un attribut de **`Document`** qui donne accès à l'objet représentant la balise **`<body>`** de votre document HTML.
- **`innerHTML`** est un attribut de **`body`** qui représente, sous la forme d'une chaine de caractères, la structure HTML contenue dans la balise **`<body>`**

SELECTEURS CSS

La classe **`Document`** possède une méthode **`querySelector`** qui retourne le premier élément du DOM correspondant au sélecteur CSS (vu au dernier TD de web) passé en paramètre :

```
//Récupérer le premier lien de la page
const link = document.querySelector("a");

//Récupérer le premier paragraphe contenu dans l'élément ayant l'id 'content'
const paragraph = document.querySelector("#content p");
```

Il est également possible de récupérer tous les éléments du DOM correspondant au sélecteur CSS fourni avec la méthode **querySelectorAll** :

```
//Récupérer un tableau contenant tous les titres de niveau 1 de la page
const titles = document.querySelectorAll("h1") ;
```

EXERCICE :

- Ajoutez le code HTML suivant, dans le body de votre page HTML, après la balise **<script>** (et non dans la balise) :

```
<h1>Titre</h1>
<p>Un paragraphe avec <strong>un texte important</strong></p>
<p>Un autre paragraphe avec <a href="#">un lien</a></p>
```

- Ecrivez une fonction JavaScript qui :
  - Modifie l'intitulé du titre **h1** en « Polytech Dijon »
  - Modifie la couleur du texte important en #4691ff (attribut **style.color**)
  - Modifie l'attribut **href** du lien pour que ce dernier pointe vers : **https://esirem.u-bourgogne.fr**
- N'oubliez pas d'appeler la fonction.
- Testez les changements au niveau du navigateur.
- Vérifiez la console.

Le navigateur charge les éléments de la page au fur et à mesure. Le script étant placé avant les balises **<h1>** et **<p>**, il est chargé en premier. Or un script est exécuté dès son chargement. Les balises qui suivent ne sont donc pas encore dans le DOM au moment où votre script essaie de les modifier.

PREMIERE SOLUTION

- Déplacez la balise **<script>** pour la placer à la fin de la balise **<body>**.
- Testez à nouveau.

DEUXIEME SOLUTION

- Replacez la balise **<script>** au début de la balise **<body>**.
- Ajoutez l'attribut **defer** à la balise script
- Testez à nouveau

Notes : L'attribut **defer** indique au navigateur de différer l'exécution du script à la fin du chargement du document.

Bonne pratique : de manière générale, il reste recommandé de placer les balises **<script>** à la fin de la balise **<body>** pour permettre au navigateur d'afficher le contenu de la page avant de charger les scripts qui peuvent être lourds.

FORMULAIRE SIMPLE ET EVENEMENT

EXERCICE

- Ajoutez à votre document HTML un champ texte et un bouton :

```
<input type="text">
<button>Tester</button>
```

- Créez une fonction qui **testValue** la valeur du champ texte :
  - Si la valeur saisie n'est pas un nombre (fonction **isNaN**), l'arrière-plan du champ texte devient rouge.
  - Si la valeur saisie est un nombre pair, l'arrière-plan du champ texte devient jaune.
  - Sinon, l'arrière-plan du champ texte devient bleu.

Bien sûr, il ne faut pas appeler cette fonction directement dans le script, sans quoi, elle sera exécutée au chargement de la page, à un moment où l'utilisateur n'aura pas encore eu le temps de saisir quoi que ce soit.

Il est possible de déclencher l'appel d'une fonction lorsqu'un événement se produit sur un élément de la page :

```
const txtMail = document.querySelector("#txt-mail");
input.addEventListener("input", function_to_call);
```

Attention : notez bien l'absence de parenthèse après le nom de la fonction qui signifie que la fonction **function_to_call** n'est pas appelée ici, mais simplement donnée en paramètre à la fonction **addEventListener**. C'est cette dernière qui se chargera d'appeler **function_to_call** lorsqu'un événement de type **input** sera déclenché.

EXERCICE

- Créez une nouvelle fonction **connectClickEvent**.
- Dans cette fonction, récupérer une référence sur le bouton avec **querySelector** vue précédemment.
- Appelez la méthode **addEventListener** du bouton pour appeler la fonction **testValue** lorsqu'un événement **click** est déclenché.
- Appelez la méthode **connectClientEvent** dans votre script.
- Testez le bon fonctionnement.

BONUS DU DEVELOPPEUR

RETABLIR LA VERITE

Un mensonge a été formulé précédemment : il existe un troisième mot clé pour déclarer une variable : **var**

EXERCICE

- Demandez à ChatGPT la différence entre **let** et **var** et mettez en évidence cette différence avec un petit programme.

SELECT

La balise **<select>** permet de créer des listes déroulantes. Chaque élément de la liste est représenté par une sous-balise **<option>** :

```
<select>
    <option>Valeur 1</option>
    <option>Valeur 2</option>
    <option>Valeur 3</option>
</select>
```

EXERCICE

- Créez une application possédant un champ texte, une liste déroulante, un bouton « Ajouter » et un bouton « Supprimer ».
  - Cliquez sur le bouton « Ajouter » ajoute la valeur saisie dans le champ texte dans la liste déroulante.
  - Cliquez sur le bouton « Supprimer » supprime l'élément sélectionné dans la liste déroulante.