JavaScript

INTRODUCTION

Le Javascript est un langage de programmation de script. Il permet de manipuler et d'implémenter des fonctionnalités complexes, pour rendre vos pages web plus interactives et dynamiques.

En savoir plus

Ce langage a été inventé par Brendan Eich en 1995, soit un an après le langage CSS. Il a fallu dix jours pour le créer. Depuis juin 2020, nous sommes actuellement à la 11ème version de ce langage.

LIER SON FICHIER JAVASCRIPT

Comme pour le fichier HTML et le CSS, le JavaScript se créé depuis un éditeur de texte en l'enregistrant en **.js**

Pour associer un fichier JavaScript à son fichier HTML il faut placer une balise `<script>` et lui donner l'attribut pour lui indiquer de quel type de ressource il s'agit `type="text/javascript"`. Ainsi que son « chemin » pour lui dire où elle se trouve par rapport à notre fichier HTML `src="javascript.js"`.

```
<script type="text/javascript" src="javascript.js"></script>
```

On peut également mettre du JavaScript directement dans le fichier HTML à l'intérieur d'une balise `<script>`.

```
<script>
  /* Ici je peux mettre du JavaScript */
</script>
```

À noter : Les appels aux fichiers JavaScript, se font en fin de fichier, avant la fermeture de la balise `</body>` et non pas dans la balise `<head>` de notre fichier.

Un commentaire JavaScript s'écrit comme en CSS, sous la forme :

```
/* Ceci est un commentaire*/
```

Ou sur une seule ligne :

```
// Un commentaire sur une ligne seulement.
```

LES VARIABLES

Une variable est un identifiant qui permet de stocker une valeur, qui comme son nom l'indique peut varier. On fait appel à une variable pour pouvoir faire référence à cette valeur de manière dynamique ou pour pouvoir la modifier.

Déclarer des variables

Pour déclarer une variable on écrit :

```
var numero ;
```

On peut directement attribuer une valeur initiale à notre variable lorsqu'on la déclare :

```
var numero = 2;
```

Une Constante est un identifiant qui permet de stocker une valeur, mais à l'inverse d'une variable sa valeur va rester constante. Pour déclarer une constante on écrit :

```
const numero = 2;
```

On peut déclarer plusieurs variables ou constantes directement avec une virgule :

```
var variable1 = 1,
    variable2 = 2;
```

Les noms des variables et des constantes ne peuvent pas contenir d'espace ni de tiret. Pour donner un nom composé on va simplement mettre en majuscule le mot suivant. Par exemple :

```
var monNomDeVariable ;
```

Modifier des variables

Pour modifier une variable on va l'appeler et lui indiquer sa nouvelle valeur :

```
numero = 3;
```

On peut aussi utiliser plusieurs méthodes pour additionner, soustraire, etc. des valeurs ou des variables. Pour cela on utilise des **opérateurs** :

- `=` Assigne une valeur.
- `+` Permet d'ajouter une valeur.
- `-` Permet de soustraire une valeur.
- `*` Permet de multiplier une valeur.
- `/` Permet de diviseur une valeur.
- `+=` Ajoute et assigne la valeur.
- `-=` Soustrait et assigne la valeur.
- `*=` Multiplie et assigne la valeur.
- `/=` Divise et assigne la valeur.
- `++` Permet d'incrémenter de 1.
- `--` Permet de décrémenter de 1.

Exemple :

```
var nombre1 = 2,
    nombre2 = 6,
    total = nombre1 + nombre2 ;  //  La variable total est égale à 8.
```

Autre exemple :

```
var   nombre = 2 ;
nombre += 8 ; //  Le résultat sera 10.
```

Autre exemple :

```
var nombre = 11 ;
nombre ++ ; //  Le résultat sera 12.
```

LES TYPES DE DONNÉES

Les trois principaux types de données primitives en JavaScript sont :

- **Les nombres.**

```
var numero = 2 ;
```

- **Les chaînes de caractères** (du texte).

```
var texte = "Ceci est une chaîne de caractères";
```

- **Les valeurs logiques, appelées booléens** (vrai ou faux).

```
var vrai = true,
    faux = false;
```

Il est possible d'appeler différents types de données. Par exemple :

```
var paragraphe = 'paragraphe numéro :' + numero ;
```

On peut vérifier nos valeurs dans la « console » de notre navigateur (grâce à l'inspecteur d'élément) en utilisant la fonction « `console.log();` »
Par exemple :

```
console.log(maVariable);
```

Dans la console on verra alors la valeur de notre variable.

LE DOM

DOM signifie Document Object Model (Modèle d'Objet du Document).
C'est la représentation de l'interface HTML. On va donc pouvoir grâce au JavaScript manipuler notre HTML et notre CSS.

Pour accéder à un élément de notre structure HTML on va devoir utiliser un sélecteur :

- **`getElementsByTagName`** : sélectionne une balise.
- **`getElementById`** : sélectionne un attribut **`id`**.
- **`getElementsByClassName`** : sélectionne un attribut **`class`**.
- **`querySelector`** : permet de sélectionner un sélecteur complexe (avec la même syntaxe qu'en CSS).
- **`querySelectorAll`** : permet de sélectionner plusieurs éléments.

Par exemple, on écrit :

```
document.getElementById("demo");
```

« `document` » *permet la recherche de l'élément que l'on souhaite sélectionner dans la page.*

Autre exemple :

```
document.querySelector("#demo > div");
```

MODIFIER LE DOM

Créer, ajouter et supprimer des éléments

Pour créer un élément on va utiliser la fonction « **`createElement`** » :

```
var nouvelleDiv = document.createElement("div");
```

Maintenant que notre élément est créé on va pouvoir l'ajouter à notre structure. Pour cela on peut appeler la variable qui sera son élément parent et utiliser la fonction « **`appendChild`** » :

```
var demo = document.getElementById("demo"),
    nouvelleDiv = document.createElement("div");

demo.appendChild(nouvelleDiv);
```

Si on souhaite supprimer un élément, on peut utiliser la fonction « **`removeChild`** » :

```
demo.removeChild(ancien) ;
```

Insérer ou modifier du contenu

Pour insérer ou modifier le contenu d'un élément on va utiliser la fonction « **`innerHTML`** ». La valeur sera alors une chaîne de caractères (pouvant contenir du code HTML) ou une variable.

Par exemple :

```
var  demo = document.getElementById("demo") ;

demo.innerHTML = "<p>Mon contenu HTML</p>" ;
```

Si par exemple, notre élément contient un attribut, il y aura alors une confusion entre les fermetures et les ouvertures des guillemets dans la chaîne de caractère. Il faut donc changer le type de guillemet. Par exemple :

```
demo.innerHTML = "<img src='mon-image.png'>";
```

OU

```
demo.innerHTML = '<img src="mon-image.png">';
```

Créer et modifier des attributs

Pour ajouter ou modifier un attribut et sa valeur, on va utiliser la fonction « `setAttribut` ». Il va donc falloir lui donner deux paramètres :

```
var demo = document.getElementById("demo");

demo.setAttribute("name", "nom");
```

Si on souhaite supprimer un attribut on va utiliser la fonction « `removeAttribut` ». On aura alors besoin de lui donner uniquement le nom de l'attribut à retirer.

```
demo.removeAttribute("name");
```

Ajouter, supprimer des classes

Pour accéder à la liste des classes d'une balise, on va utiliser la fonction « `classiList` ».

On peut ajouter une classe :

```
demo.classList.add("nouvelleClasse");
```

On peut aussi supprimer une classe :

```
demo.classList.remove("ancienneClasse");
```

Changer le style

Pour changer le style CSS d'un élément, on va utiliser la fonction « `style` ». Il faut donc cibler la variable dont on souhaite changer le style puis, faire appel à la fonction, nommer la propriété CSS que l'on veut modifier. Enfin on va lui passer le nouveau paramètre, par exemple :

```
demo.style.color = "red";
```

Pour rappel, en JavaScript, les noms de variables, de fonction, etc. ne peuvent pas avoir d'espace ni de tiret. Une propriété CSS composée aura donc la forme :

```
demo.style.backgroundColor = "red";
```

ÉCOUTER DES ÉVÉNEMENTS

Un événement est une réaction, liée à une action faite par l'utilisateur. On l'utilise grâce à la fonction « **`addEventListenner`** ». Par exemple si on souhaite changer le style d'un élément au clique sur un élément :

```
var demo = document.getElementById("#demo");

demo.addEventListener("click", function(){
  demo.style.display = "none";
});
```

Il existe de nombreuses actions que l'utilisateur peut faire et sur lesquelles on peut déclencher des événements.

LES FONCTIONS

Une fonction est un groupement d'instructions qui exécutent des tâches. On les utilise pour des mécaniques et des fonctionnalités qui vont être réutilisées, ou pour les appeler à un moment précis.

Pour donner des instructions à une fonction on va devoir la nommer, on écrit alors :

```
function maFonction(){
  //  Ici on place les instructions que la fonction va effectuer.
} ;
```

Ensuite, pour déclencher la fonction on écrit simplement :

```
maFonction();
```

On peut aussi déclencher par exemple une fonction sur un écouteur d'événement :

```
demo.addEventListener("click", maFonction);
```

Pour vérifier si notre fonction se déclenche bien, on peut utiliser une alerte, qui nous ouvrira un message dans notre navigateur :

```
function maFonction(){
  alert("Ma fonction se déclenche !");
} ;
```

À noter : En JavaScript, les variables créées peuvent être vues ou utilisées qu'à l'intérieur du bloc de code dans lequel elles sont déclarées. Par exemple, si elles sont déclarées dans une fonction, elles n'existeront pas en dehors de celle-ci.