Introduction à javascript

I) Présentation

Le javascript est un langage de programmation qui a été crée en 1995. Il fait partie des langages standards du web au même titre que le HTML et le CSS.

Le javascript est un langage dynamique qui s'exécute(principalement côté client).

Un langage dynamique est un langage qui permet de générer du contenu dynamique pour les pages web. Un contenu dynamique et un contenu qui va se mettre à jour sans que l'on ait besoin de changer le code manuellement mais en fonction de certains facteurs extérieurs.

Le javascript (dans la suite je le noterai js) peut s'écrire à trois endroits différents :

- Directement dans la balise ouvrante d'un élément HTML.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <title>Simple page</title>
6
7  </head>
8  <body>
9
10
11     <button onclick=alert("bonjour")> Cliquez Moi</button>
12
13     <p>Ceci est une simple page HTML.</p>
14
15     <p>Elle est aussi basique que possible.</p>
16
17
18 </body>
19 </html>
```

- Dans le head de la page HTML

  On ajoutera dans le head la balise HTML

  <script>
  alert("Bonjour")

  </script>
  On obtiendra une boite de dialogue qui vous souhaite bonjour par contre la page HTML ne s'affiche plus, il faut cliquer sur OK pour la voir apparaitre.

  Si on déplace le script dans le body les éléments placés avant le script apparaissent mais pas ceux situés en dessous. Que se passe t-il ?

  En fait les navigateurs exécute le js dès qu'il le rencontre et bloque l'exécution du reste de la page. Lorsque l'on clique sur OK on arrête l'exécution du js. Donc pendant longtemps on a placé le script en tout fin de page.

- Dans un fichier js externe. Pour cela il faut créer un nouveau fichier et l'enregistrer avec une extension js. Dans l'élément head du HTML on va ajouter un élément script mais cette fois ci on va simplement indiquer à l'aide l'attribut src le chemin vers le fichier js.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Intro à javascript</title>
<script src="ex1.js" defer></script>
</head>
<body>
<h1> Exemple 1</h1>
<p id="hello"></p>
<p id="date"></p>
<button onclick='affiche()' type="button">Présentation</button>
</br>
<button onclick='pres()' type="button">Présentation</button>
<p id="coucou"></p>



</body>
</html>
```

le mot defer placé dans la balise ouvrante <script> permet de différer l'exécution du js jusqu'à ce que l'intégralité d la page web soit affichée.

II) Les commentaires en Javascript

Les commentaires sont des lignes de texte qui servent à documenter, c'est à dire à expliquer ce que fait tel ou tel bout de script.

Les commentaires ne sont pas lues par le navigateur et seront donc invisibles par l'utilisateur de la page mais ils sont très utiles si une autre personne reprend le script ou même pour la personne qui à écrit le code au bout de quelques semaines on ne se rappelle plus forcément pourquoi on avait codé un bout de programme de la façon où on la fait.

Il y a deux types de commentaires en js :

- Les commentaires monolignes. Comme leur nom l'indique ils s'écrivent sur une seule ligne. Ils sont introduits par un double slash //.

  // Ceci est un commentaire monoligne en js.

- Pour les commentaires plus long on utilisera les commentaires multilignes. Ils commencent par /* et finissent par */.

  /* Ceci est un
  commentaire multiligne*/

III) L'indentation et la syntaxe

L'indentation n'est pas obligatoire en js par contre c'est une bonne pratique pour rendre son code plus simple à comprendre et plus lisible.

En ce qui concerne la syntaxe, une instruction en js se termine par un point virgule (;). Vous trouverez sur le net des cours de js dans lequel il est dit que les points virgules en fin d'instructions ne sont pas obligatoires. Ils ont raison mais ils ne s'adressent pas au même public, javascript est censé "savoir" ou se termine les instructions donc il rajoute des points virgules la où cela lui semble cohérent. Dans la mesure ou vous débutez et que vous ne savez pas comment les points virgules sont ajoutés vous risquez de vous retrouvez avec un code qui ne fonctionnera pas si vous oubliez les points virgules. Donc pour nous les poins virgules à la fin des instructions seront obligatoires.

IV) Les variables et les constantes

Une variable est un espace de stockage temporaire. En js les variables doivent être déclarées pour cela on va utiliser soit le mot clé var soit le mot clé let.

Le nom des variables doit respecter certaines règles :

- le nom de la variable doit commencer par une lettre ou un underscore (tiret du 8).
- Le nom de la variable ne doit contenir que des lettres, des chiffres (sauf en première position) et des underscores.
- Le nom d'une variable ne doit pas contenir d'espaces.

Le js est sensible à la casse cela signifie qu'il fait la différence entre majuscules et minuscules donc en js mavariable et Mavariable seront considérées comme deux variables différentes.

Une variable peut être initialisée soit au moment de sa création :

let cours="NSI"

ou après se création

let effectif

effectif=14

Les constantes sont définies avec le mot clé const une fois la constante définie on ne peut plus changer sa valeur.

const pi=3.1415

V) Les types de données

En javascript il existe 7 types de données.

Les trois plus courants pour nous seront :

- String, chaine de caractère ;
- Number, Nombre ;
- Boolean,booléen ;

Une chaine de caractères est une séquence de caractères, ou ce que l'on appelle un texte.

Remarque toute valeur stockée dans une variable en utilisant des guillemets "" ou des apostrophes sera considérée comme une chaine de caractères.

exemple si on écrit let age="29"

La variable âge ne contient pas le nombre 29 mais le texte 29.

VI) Les opérateurs

Il existe différents types d'opérateurs qui vont nous servir à réaliser des opérations . Ceux que l'on utilisera Les plus fréquemment sont :

- Les opérateurs arithmétiques :

Opérateur	Nom de l'opération associée
+	Addition
–	Soustraction

Opérateur	Nom de l'opération associée
*	Multiplication
/	Division
%	Modulo (reste d'une division euclidienne)
**	Exponentielle (élévation à la puissance d'un nombre par un autre)

- Les opérateurs d'affection :

Opérateur	Définition
+=	Additionne puis affecte le résultat
-=	Soustrait puis affecte le résultat
*=	Multiplie puis affecte le résultat
/=	Divise puis affecte le résultat
%=	Calcule le modulo puis affecte le résultat

- Les opérateurs de comparaison :

Opérateur	Définition
==	Permet de tester l'égalité sur les valeurs
===	Permet de tester l'égalité en termes de valeurs et de types
!=	Permet de tester la différence en valeurs
<>	Permet également de tester la différence en valeurs
!==	Permet de tester la différence en valeurs ou en types
<	Permet de tester si une valeur est strictement inférieure à une autre
>	Permet de tester si une valeur est strictement supérieure à une autre
<=	Permet de tester si une valeur est inférieure ou égale à une autre
>=	Permet de tester si une valeur est supérieure ou égale à une autre

- Les opérateurs de logique :

Opérateur (nom)	Opérateur (symbole)	Description
AND (ET)	&&	Lorsqu'il est utilisé avec des valeurs booléennes, renvoie true si toutes les comparaisons sont évaluées à true ou false sinon
OR (OU)	||	Lorsqu'il est utilisé avec des valeurs booléennes, renvoie true si au moins l'une des comparaisons est évaluée à true ou false sinon
NO (NON)	!	Renvoie false si une comparaison est évaluée à true ou renvoie true dans le cas contraire

- La concaténation
  Pour concaténer autrement dit pour mettre bout à bout deux éléments on utilise le symbole + si les deux éléments sont des nombres le + représente l'addition classique si l'un des deux éléments n'est pas un nombre il seront mis bout à bout.

ex
let a=3 ;
let b=4 ;
let res=a+b
console.log(res)
affichera 7 dans la console

let a=" N" ;
let b="SI" ;
let res=a+b
console.log(res)

affichera NSI dans la console

let a=" 3" ;
let b="4" ;
let res=a+b
console.log(res)

affichera 34 dans la console

Il y a un autre moyen de concaténer le backtype
let a=3 ;
let b=4 ;
let res=a+b
console.log(`a`)

VII) Les structures de controle

- **Structure conditionnelle**

  Pour faire une conditionnelle en js on va utiliser la structure :

```
if(condition à vérifié){
        instructions
        }
else{
    instructions
    }
```

  Il existe aussi ce que l'on appelle un opérateur ternaire (comme en Python avec le elif) la structure est la suivante :

```
1 if(condition1){
2     instructions
3 }
4 else if(condition 2){
5     instructions
6 }
7 else{
8     instructions
9 }
```

- **Les boucles en javascript**

  Comme dans les autres langages de programmation une boucle en js est composéee de trois éléments.

- Une valeur de départ qui va nous servir à initialiser notre boucle et nous servir de compteur ;
- Un test ou une condition de sortie qui précise le critère de sortie de la boucle ;
- Un itérateur qui va modifier la valeur de départ de la boucle à chaque nouveau passage jusqu'au moment où la condition de sortie est vérifiée. Bien souvent, on incrémentera la valeur de départ.

La boucle while

Elle permet de répéter une série d'instructions tant que la condition de sortie n'est pas vérifiée.

```
while(condition){
    instructions
}
```

ex :

Code HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Simple page</title>
    <script src="wh2.js" defer></script>
</head>
<body>
    <h1>Exemple Boucle While</h1>

    <p id="para"></p> <!--On définit un paragraphe identifié à l'aide du mot

</body>
</html>
```

Code javascript

```
let x=0; // On définit une variable x initialisée à zéro
let S=parseInt(prompt("Entrez un nombre")); //On demande la saisie d'un nomb

while(x<=S){/* tant que x est plus petit que S*/
    document.getElementById('para').innerHTML+=`x stocke la valeur ${x} lors
    n°${x+1} dans la boucle<br>`;/* On récupère l'élément appellé para dans
    et on le remplace par une phrase*/
    x++ //On incrémente x d'une unité
}
```

Affichage pour x=5

Exemple Boucle While

x stocke la valeur 0 lors du passage n°1 dans la boucle
x stocke la valeur 1 lors du passage n°2 dans la boucle
x stocke la valeur 2 lors du passage n°3 dans la boucle
x stocke la valeur 3 lors du passage n°4 dans la boucle
x stocke la valeur 4 lors du passage n°5 dans la boucle
x stocke la valeur 5 lors du passage n°6 dans la boucle

La boucle do while

Elle est relativement similaire à la boucle while,la grande différence c'est que la boucle sera exécutée au moins une fois même si la condition de sortie est vérifiée dès le départ.

```
do{
    instructions
}

while(condition)
```

La boucle for

On l'utilise lorsque l'on souhaite que la boucle s'exécute un nombre déterminé de fois.

```
for (let i=0; i<=10; i++){
    instructions
}
```

La boucle for in

On l'utilise pour boucler sur les éléments d'un objet itérable. Un tableau par exemple.

```
let A=[1,2,3,4,0]

for (x in A){
    document.getElementById('para').innerHTML+=`A stocke la valeur ${x} <br>
}
```

III) Les fonctions

En javascript une fonction se définit à partir du mot clé function

La syntaxe est la suivante :

```
let a=2,b=3, result; //on définit des variables dont deux initialisées

function /*mot clé pour définir un fonction*/ Somme(){
    result=a+b;
    console.log(`la somme vaut ${result}`);

}

Somme(); /*appel de la fonction*/
```

En javascript les fonction sont des variables on aurait pu utliser la syntaxe suivante :

```
let a=2,b=3, result; //on définit des variables dont deux initialisées

var Somme=function(){
    result=a+b;
    console.log(`la somme vaut ${result}`);
    alert(`la somme vaut ${result}`);

}

Somme(); /*appel de la fonction*/
```

La fonction que l'on a définie au dessus ne fait qu'une seule chose elle additionne 2 et 3 et affiche le résultat. Si l'on souhaite faire une fonction qui additionne des nombres que l'on peut choisir il va falloir passer ces nombres en paramètres c'est à cela que servent les parenthèses après le nom de la fonction.

```
function Somme(a,b){
    result=a+b;
    console.log(`la somme vaut ${result}`);
    alert(`la somme vaut ${result}`);

}

Somme(2,6); /*appel de la fonction*/
Somme(3,7);
```

Ici le résultat est juste affiché on ne peut rien en faire. Si on souhaite utiliser le résultat il va falloir demander à la fonction de nous le retourner à l'aide du mot clé return.

```
function Somme(a,b){
    var result;
    result=a+b;
    return result;
}
var S=Somme(3,9);
alert(S)
```

IX) Les évènements

En JavaScript, un évènement est une action qui se produit et à laquelle on va réagir.

Par exemple un clic sur un élément de la page,le survol d'un élément etc.