Apprenez à programmer

Avec JavaScript

Ce cours est conçu pour vous enseigner les bases du **langage de programmation JavaScript** et pour vous apporter beaucoup de pratique au passage ! Nous nous intéresserons à :

- Utiliser les données et les types de données dans JavaScript.
- Gérer la logique d'un programme en JavaScript (conditions, boucles et erreurs).
- Écrire du code propre et facile à maintenir à l'aide de méthodes en JavaScript.

Prérequis :

- Connaissances générales de la terminologie de la programmation orientée objet (classes, objets, fonctions, méthodes, etc.),

1) Introduction au JavaScript

Le JavaScript est un langage de programmation créé en 1995. Le JavaScript est aujourd'hui l'un des langages de programmation les plus populaires et il fait partie des langages web dits « standards » avec le HTML et le CSS.

Pour définir ce qu'est le JavaScript et le situer par rapport aux autres langages, et donc pour comprendre les intérêts et usages du JavaScript il faut savoir que :

- Le JavaScript est un langage dynamique ;
- Le JavaScript est un langage (principalement) côté client ;
- Le JavaScript est un langage interprété ;
- Le JavaScript est un langage orienté objet.

2) Où écrire le code JavaScript ?

On va pouvoir placer du code JavaScript à trois endroits différents :

- Directement dans la balise ouvrante d'un élément HTML ;
- Dans un élément **script**, au sein d'une page HTML ;
- Dans un fichier séparé contenant exclusivement du JavaScript et portant l'extension **.js.**

3) L'environnement de travail JavaScript

Pour coder en JavaScript, nous n'allons avoir besoin que d'un éditeur de texte. Il existe de nombreux éditeurs de texte sur le web et la majorité d'entre eux sont gratuits :

- Komodo Edit
- Atom
- NotePad++
- Brackets

Les éditeurs en ligne : Certains sites comme codepen.io ou jsbin.com permettent d'écrire du code HTML, CSS ou JavaScript et de voir le résultat immédiatement.

4) Les Commentaires en JavaScript

Les commentaires sont des lignes de texte (des indications) placées au milieu d'un script et servant à documenter le code, c'est-à-dire à expliquer ce que fait tel ou tel bout de script et éventuellement comment le manipuler.

Pour écrire un commentaire multilignes, il faudra entourer le texte de notre commentaire avec la syntaxe suivante **/* */**.

Pour écrire un commentaire monoligne, on utilisera un double slash **//** qui sera suivi du texte de notre commentaire (ou éventuellement la syntaxe multilignes).

5) Un premier point sur la syntaxe de base du JavaScript

« Toute instruction en JavaScript doit être terminée explicitement avec un point-virgule »

Le point-virgule est généralement utilisé en informatique pour indiquer la fin d'une instruction, c'est-à-dire pour séparer deux instructions l'une de l'autre et cela va également être le cas en JavaScript.

6) Déclarez des variables et modifiez leurs valeurs

Votre programme utilise des **variables** pour enregistrer et manipuler des données. Plus précisément, une variable est un **contenant** utilisé pour enregistrer une donnée spécifique dont votre programme a besoin pour travailler. Une donnée placée dans une variable s'appelle une **valeur**.

Une variable admet un **nom**. Voici quelques règles générales pour la création de noms :

En JavaScript, la déclaration d'une variable se limite au mot clé ***var*** ou ***let***, suivi du nom de variable choisi :

```
let numberOfCats = 2;
var numberOfDogs = 4;
```

La façon la plus simple de modifier la valeur d'une variable est simplement de la réaffecter :

```
let numberOfCats = 3;
numberOfCats = 4;
```

7) Les opérateurs arithmétiques et d'affectation JavaScript

Il existe différents types d'opérateurs qui vont nous servir à réaliser des opérations de types différents. Les plus fréquemment utilisés sont :

- Les opérateurs arithmétiques

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

Exemple :

```
let x = 2;
let y = 3;
let f = 5 % 3; //f stocke le reste de la division euclidienne de 5 par 3
let g = x ** 3; //g stocke 2^3 = 2 * 2 * 2 = 8
/*On affiche les résultats dans une boite d'alerte en utilisant l'opérateur de concaténation "+". On retourne à la ligne dans l'affichage avec "\n"*/
alert('f contient : ' + f + '\ng contient : ' + g);
```

Priorité des calculs et utilisation des parenthèses

Concernant les règles de calcul, c'est-à-dire l'ordre de priorité des opérations, celui-ci va être le même qu'en mathématiques : l'élévation à la puissance va être prioritaire sur les autres opérations, tandis que la multiplication, la division et le modulo vont avoir le même ordre de priorité et être prioritaires sur l'addition et la soustraction qui ont également le même niveau de priorité.

Exemple :

```
let x = 1 - 2 - 3; //Calcule (1 - 2) - 3 = _& - 3 = - 4
let y = 1 - (2 - 3); //Calcule 1 - (2 - 3) = 1 - (- 1) = 1 + 1 = 2
let z = 2 ** 3 ** 2; //Calcule 3 ** 2 = 3 * 3 = 9 puis 2 ** 9 = 512
```

- Les opérateurs d'affectation / d'assignation

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

Exemple :

```
let x = 2; //x stocke 2
let y = 10; //y stocke 10
/*On ajoute 3 à la valeur stockée précédemment par x (2) puis on affecte le résultat à x. x stocke désormais 2 + 3 = 5*/
x += 3;
/*On mutiplie la valeur de y (10) par celle de z (5) puis on affecte le résultat à y. y stocke désormais 10 * 5 = 50*/
y *= x;
alert('x stocke : ' + x + 'y stocke : ' + y);
```

- L'opérateur de concaténation ;

Concaténer signifie littéralement « mettre bout à bout ». La concaténation est un mot généralement utilisé pour désigner le fait de rassembler deux chaines de caractères pour en former une nouvelle.

En JavaScript, l'opérateur de concaténation est le signe +. Faites bien attention ici : lorsque le signe + est utilisé avec deux nombres, il sert à les additionner. Lorsqu'il est utilisé avec autre chose que deux nombres, il sert d'opérateur de concaténation.

Exemple :

```
let x = 28 + 1; //Le signe "+" est ici un opérateur arithmétique
let y = 'Bonjour';
let z = x + ' ans'; //Le signe "+" est ici un opérateur de concaténation
alert(y + ', je m\'appelle Pierre, j\'ai ' + z);
```

On peut remplacer l'opérateur de concaténation par les expressions entre **${** et **}**.

Exemple :

```
let x = 5;
let y = 50;
//Code affiché dans des boites d'alerte
alert('x contient ' + x +
      '\ny contient ' + y +
      '\nLeur somme vaut ' + (x + y));
//remplacer les signes de concaténation
alert(`x contient ${x}
      y contient ${y}
      Leur somme vaut ${x + y}`);
```

- Les opérateurs d'incrémentation et décrémentation ; ++ , -- , + = et - =

(opérateur + variable)	**Résultat**
++x	Pré-incrémentation : incrémente la valeur contenue dans la variable x, puis retourne la valeur incrémentée
x++	Post-incrémentation : retourne la valeur contenue dans x avant incrémentation, puis incrémente la valeur de $x
--x	Pré-décrémentation : décrémente la valeur contenue dans la variable x, puis retourne la valeur décrémentée
x--	Post-décrémentation : retourne la valeur contenue dans x avant décrémentation, puis décrémente la valeur de $x

Exemple 1:

```
//On déclare et initialise nos variables sur la même ligne
let a = 10, b = 10, c = 20, d = 20;
/*On incrémente / décrémente et affecte le résultat dans un paragraphe.
 *Attention : le premier "+" est un opérateur de concaténation */
document.getElementById('p1').innerHTML = 'a stocke la valeur ' + a++;
document.getElementById('p2').innerHTML = 'b stocke la valeur ' + ++b;
document.getElementById('p3').innerHTML = 'c stocke la valeur ' + c--;
document.getElementById('p4').innerHTML = 'd stocke la valeur ' + --d;
```

```
//On affiche ensuite à nouveau le contenu de nos variables
document.getElementById('p5').innerHTML =
'a = ' + a + ', b = ' + b + ', c = ' + c + ', d = ' + d;
```

Exemple 2:

```
let cookiesInJar = 10;
let numberOfLikes = 10;
 /* manger deux cookies */
cookiesInJar -= 2;  //il reste 8 cookies
 /* cuisson d'un nouveau lot de cookies */
cookiesInJar += 12; // il y a maintenant 20 cookies dans la boîte
numberOfLikes++;  // cela fait 11
numberOfLikes--; // et on revient à 10...qui n'a pas aimé mon article ?
```

- Les opérateurs logiques :

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

- 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

- L'opérateur ternaire :

Les structures conditionnelles ternaires (souvent simplement abrégées "ternaires") correspondent à une autre façon d'écrire nos conditions en utilisant une syntaxe basée sur l'opérateur ternaire ? : qui est un opérateur de comparaison.

Exemple :

```
let x = 15;
let y = -20;
```

```
document.getElementById('p1').innerHTML =
x >= 10 ? 'x supérieur à 10' : 'x stric. inférieur à 10';
document.getElementById('p2').innerHTML =
y >= 10 ? 'y supérieur à 10' : 'y stric. inférieur à 10';
```

8) Découvrez les constantes

Dans de nombreux programmes, certaines données ne seront pas modifiées pendant l'exécution du programme. Pour s'assurer de ne pas réaffecter par inadvertance de nouvelles valeurs à ces données, vous allez utiliser des **constantes**.
Pour créer ou déclarer une constante en JavaScript, nous allons utiliser le mot clef **const**.
On va pouvoir déclarer une constante exactement de la même façon qu'une variable à la différence qu'on va utiliser **const** à la place de **let** ou de **va**r.

9) Les Structures de contrôle

9.1) L'instruction if

Nous avons accès aux structures conditionnelles suivantes en JavaScript :

- La condition if (si) ;
- La condition if… else (si… sinon) ;
- La condition if… elseif… else (si… sinon si… sinon).

Exemple :

```
let x = 4; //On stocke le chiffre 4 dans x
/*Les comparaisons sont effectuées avant l'affectation. Le JavaScript va donc commencer par comparer et renvoyer true ou false et nous allons stocker ce résultat dans nos variables test*/
let test1 = x == 4;
let test2 = x === 4;
let test3 = x == '4';
let test4 = x === '4';
let test5 = x != '4';
let test6 = x !== '4';
let test7 = x > 4;
let test8 = x >= 4;
let test9 = x < 4;
alert('Valeur dans x égale à 4 (en valeur) ? : ' + test1 +
      '\nValeur dans x égale à 4 (valeur & type) ? : ' + test2 +
      '\nValeur dans x égale à "4" (en valeur) ? : ' + test3 +
      '\nValeur dans x égale à "4" (valeur & type) ? : ' + test4 +
      '\nValeur dans x différente de "4" (en valeur) ? : ' + test5 +
      '\nValeur dans x différente de "4" (valeur & type) ? : ' + test6 +
      '\nValeur dans x strictement supérieure à 4 ? : ' + test7 +
```

```
      '\nValeur dans x supérieure ou égale à 4 ? : ' + test8 +
      '\nValeur dans x strictement inférieure à 4 ? : ' + test9);
```

9.2) L'instruction switch

L'instruction **switch** peut être utiliser dans certaines situations précises à la place d'une condition if…else if…else.

Exemple : l'exemple suivant affichera « x stocke la valeur 15 »

```
let x = 15;
switch(x){
    case 2:
        document.getElementById('p1').innerHTML = 'x stocke la valeur 2';
        break;
    case 5:
        document.getElementById('p1').innerHTML = 'x stocke la valeur 5';
        break;
    case 10:
        document.getElementById('p1').innerHTML = 'x stocke la valeur 10';
        break;
    case 15:
        document.getElementById('p1').innerHTML = 'x stocke la valeur 15';
        break;
    case 20:
        document.getElementById('p1').innerHTML = 'x stocke la valeur 20';
        break;
    default:
        document.getElementById('p1').innerHTML =
        'x ne stocke ni 2, ni 5, ni 10, ni 15 ni 20';
}
```

10) Les Structures itératives

10.1) Le boucle while

```
//Tant que...
while(condition)
{
    //...exécute ce code
    Traitement ;
    //...incrémentation compteur
}
```

10.2) La boucle do… while

```
//Faire .. Tant que...
Do
{
    //...exécute ce code
    Traitement ;
    //...incrémentation compteur
}
while(condition) ;
```

10.3) la boucle for

```
//Faire .. Tant que...
For (intialisation ;condition ;progression)
{
    //...exécute ce code
    Traitement ;
}
```

a) Utiliser une instruction continue pour passer directement à l'itération suivante d'une boucle

Pour sauter une itération de boucle et passer directement à la suivante, on peut utiliser une instruction continue. Cette instruction va nous permettre de sauter l'itération actuelle et de passer directement à l'itération suivante.

```
for(let i = 0; i < 10; i++)
{
    //Si i / 2 possède un reste, alors i est impair
    if(i % 2 != 0)
     {
        continue;
     }
    document.getElementById('p1').innerHTML +=
    'i stocke la valeur ' + i + ' lors du passage n°'
    + (i + 1) + ' dans la boucle<br/>';
```

b) Utiliser une instruction break pour sortir prématurément d'une boucle

On va également pouvoir complètement stopper l'exécution d'une boucle et sortir à un moment donné en utilisant une instruction **break** au sein de la boucle.

```
for(let i = 0; i < 1000; i++){
    //On sort de la boucle dès que la valeur de i atteint 13
    if(i == 13){
```

```
        break;
    }
    document.getElementById('p1').innerHTML +=
    'i stocke la valeur ' + i + ' lors du passage n°'
    + (i + 1) + ' dans la boucle<br>';
}
```

11) Présentation des fonctions JavaScript

Une fonction correspond à un bloc de code nommé et réutilisable et dont le but est d'effectuer une tâche précise.
Deux types de fonctions en JavaScript : les fonctions prédéfinies et les fonctions personnalisées.

11.1) Les fonctions personnalisées :

```
function name(parameter1, parameter2, parameter3)
{
  // code to be executed
}
```

Les fonctions utilisent également une instruction **return**. Cette instruction va permettre aux fonctions de retourner une valeur qu'on va ensuite pouvoir manipuler.
Pour exécuter le code d'une fonction, il faut l'appeler. Pour cela, il suffit d'écrire son nom suivi d'un couple de parenthèses en passant éventuellement des arguments dans les parenthèses.

```
/*On définit deux fonctions personnalisées.
 *La fonction aleatoire() se sert de la fonction (méthode) random().
 *La fonction multiplication() multiplie deux nombres entre eux.
 function aleatoire(){
    return Math.random() * 100;
}
function multiplication(nombre1, nombre2){
    //Attention : les "+" sont utilisés pour la concaténation !
    return nombre1 + ' * ' + nombre2 + ' = ' + (nombre1 * nombre2);
}
/*On appelle ou "invoque" ou encore "exécute" nos fonctions et on place
les résultats retournés dans les paragraphes p id='p1' et p id='p2' de
notre fichier HTML.
 *On fournit ici deux arguments à multiplication() pour que la fonction
s'exécute normalement. Ces arguments vont prendre la place des
paramètres*/
document.getElementById('p1').innerHTML = aleatoire();
document.getElementById('p2').innerHTML = multiplication(5, 10);
```

11.2) Les fonctions prédéfinies :

Le langage JavaScript dispose de nombreuses fonctions que nous pouvons utiliser pour effectuer différentes tâches. Les fonctions définies dans le langage sont appelées fonctions prédéfinies ou fonctions prêtes à l'emploi car il nous suffit de les appeler pour nous en servir.
Par exemple, le JavaScript dispose d'une fonction nommée **random()** (qui appartient à **l'objet Math** que nous étudierons plus tard) et qui va générer un nombre décimal aléatoire entre 0 et 1 ou encore d'une fonction **replace()** (qui appartient cette fois-ci à **l'objet String**) qui va nous permettre de chercher et de remplacer une expression par une autres dans une chaine de caractères.

```
let prez = 'Bonjour, je suis Pierre';
/*Math.random() génère un nombre décimal aléatoire entre 0 et 1 qu'on
 *place ici au sein de notre paragraphe  p id='p1'*/
document.getElementById('p1').innerHTML = Math.random();

/*String.replace() chercher une expression dans une chaine de caractères
 *et la remplace par une autre. Ici, on cherche "Pierre" dans let prez et
 *on remplace par "Mathilde" avant d'afficher le résultat dans p id='p2'*/
let prez2 = prez.replace('Pierre', 'Mathilde');
document.getElementById('p2').innerHTML = prez2;
```

WebBiographie :

https://openclassrooms.com/fr/courses/6175841-apprenez-a-programmer-avec-javascript/6278392-declarez-des-variables-et-modifiez-leurs-valeurs

https://waytolearnx.com/2019/07/comment-recuperer-la-valeur-dun-input-texte-en-javascript.html

https://www.pierre-giraud.com/javascript-apprendre-coder-cours/boucle-while-do-for-in/