Le Javascript

Table des matières

1. Introduction

Le Javascript est un langage de programmation qui est exécuté côté **CLIENT**. C'est le navigateur qui exécute le code.

Il est, comme le **CSS**, basé sur la manipulation des éléments du **DOM** à l'aide de ce que l'on appelle des **SELECTEURS**.

Par exemple, si on a une DIV avec l'attribut ID valant « maDiv », alors on pourra la sélectionner comme ceci :

```
<div id="maDiv">
    Hello World
</div>
```

```
site_simple > js > JS homepage.js
1    document.getElementById("maDiv").innerHTML = "Hello Javascript !";
```

Et on se sert de l'attribut innerHTML pour écrire une nouvelle valeur dedans.

2. Les évènements

Les évènements sont des actions que Javascript est capable de **détecter**, comme par exemple les clics, le scroll, le survol, l'affichage d'un élément…

Pour cela on va utiliser ce que l'on appelle un **eventListener**, littéralement un « écouteur d'évènement », qui va être constamment attentif à vérifier si l'évènement est **déclenché** ou non. Exemple :

```
document.getElementById("maDiv").addEventListener('click', function(){
    alert('clicked')
});
```

Ici, on passe à la fonction addEventListener deux paramètres :

- Le premier est la **nature** de l'évènement à écouter, ici en l'occurrence le click.
- Le deuxième est la fonction à déclencher lorsque l'évènement est déclenché, ici une pop-up.

Liste non-exhaustive des évènements que l'on peut écouter :

Évènements souris	click	mouseenter	mouseleave	mouseover	scroll
Évènements clavier	keypress				
Actions	submit	close	load		

Exercice :

Afficher une balise « button » et lorsque l'on clique dessus, afficher une pop-up disant Bonjour.

Réponse :

```
<body>
    <button id="yo">Bonjour</button>
    <script>
        document.getElementById('yo').addEventListener('click', function() {
            alert('Salut');
        });
    </script>
</body>
```

3. Les variables

Tout comme en PHP, on peut tout à fait déclarer des variables en Javascript. Pour cela on va utiliser la syntaxe **var**. Il en existe d'autres comme **let** et **const** qui sont moins utilisées et qui dépendent du besoin que l'on a. **const** sert à déclarer des **constantes**, c'est-à-dire des valeurs qui ne changeront pas, et **let** et une version plus récente de **var** mais qui ne fonctionne pas sur tous les navigateurs.

Nous allons donc nous concentrer uniquement sur **var** pour l'instant.

Ces variables peuvent être testées avec la syntaxe **if** / **else** comme en **PHP**

Exemple

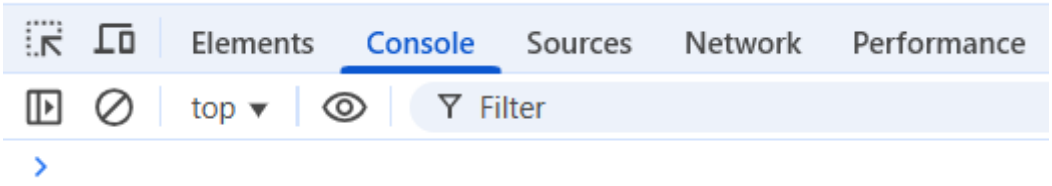
```
<script>
    var nombre = 69;
    if(nombre == 69) {
        document.write('Nice ;)');
    } else {
        document.write('Ton nombre est : ' + nombre);
    }
</script>
```

4. Débuguer efficacement en Javascript

Le code étant exécuté côté client, nous n'avons pas de gestion d'erreur native dépendant de la config du serveur. Pour cela nous avons besoin de pouvoir débuguer efficacement.

Pour cela nous avons plusieurs moyens de s'assurer des contenus de nos variables. Le premier et le plus important d'entre eux va être **console.log()**

Cette fonction va nous permettre d'afficher le contenu d'une variable dans la **console**. Qu'est-ce que la console ? Il s'agit de l'onglet « **Console** » auquel vous avez accès en ouvrant l'inspecteur d'élément de votre navigateur.



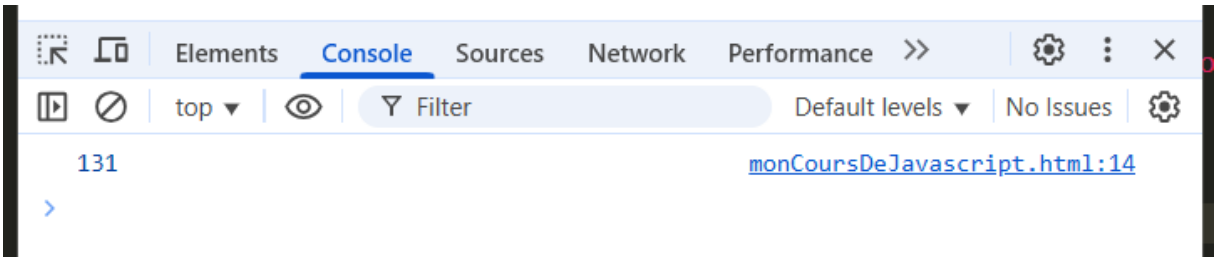
Si je veux vérifier que ma variable prend bien les bonnes valeurs, je peux donc l'afficher ici.

Exemple

Ici on additionne des nombres en les préfixant d'un **+** car le plus est également le caractère servant à la concaténation… Il faut donc bien lui préciser ici qu'on veut le résultat d'un calcul et non deux chaînes de caractères collées ensemble

```
<script>
    var nombre = +35 + +96;
    console.log(nombre);
</script>
```

Et mon nombre va s'afficher comme ceci en console :



Deux choses à noter : On a notre valeur à gauche, alors qu'elle n'apparaît nulle part dans le code. Elle est le résultat du calcul que je lui ai demandé de faire. Et enfin, à droite, on voit l'endroit dans lequel le **console.log()** ; a été interprété, en l'occurrence ici, à la ligne 14 de mon fichier « monCoursDeJavascript.html ».

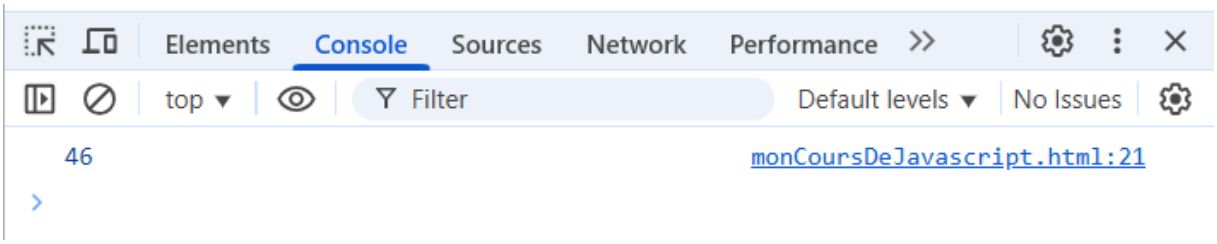
Grâce à cette information, on peut efficacement **TRACER** le chemin que va prendre notre variable. Dans le cas où on a plusieurs embranchements possibles, des conditions et autres joyeusetés, il peut être utile de simplement vérifier **Où** est-ce qu'on passe.

Pour mettre à l'épreuve cette logique, nous allons demander à l'utilisateur de saisir deux chiffres, on va les additionner, puis on va prévoir plusieurs chemins possibles pour afficher un message différent.

Exemple

```
12      <script>
13          var nombre1 = prompt('Saissez un nombre');
14          var nombre2 = prompt('Saissez un deuxième nombre');
15
16          var resultat = +nombre1 + +nombre2;
17
18          if(resultat == 69) {
19              console.log(resultat);
20          } else if(resultat % 2 == 0) {
21              console.log(resultat);
22          } else {
23              console.log(resultat);
24          }
25      </script>
```

Et en console on a :



On voit ici que seul le console.log ligne 21 s'est exécuté, donc ça veut dire qu'on passe dans la condition du milieu, ligne 20.

5. Les fonctions

Ce qu'il y a de bien avec les fonctions, c'est que vous les connaissez déjà : on les utilise depuis le début de ce cours.

*Une fonction est un bout de programme détaché du reste qui ne s'exécutera que lorsque qu'elle sera **appelée**.*

Elle s'écrit avec la syntaxe ***function,*** puis son **nom**, qui doit être unique, et enfin entre parenthèses ses **paramètres**.

Attention : des fonctions qui ne prennent pas de paramètres peuvent exister.

Exemple

```
afficherBonjourPrenom('Abdelazize');
afficherBonjourPrenom('Nadège');
afficherBonjourPrenom('Manu');
afficherBonjourPrenom('Hugo');
afficherBonjourPrenom('Dylan');
afficherBonjourPrenom('Carole');
afficherBonjourPrenom('Thomas');
afficherBonjourPrenom('Nicolas');
function afficherBonjourPrenom(prenom) {
    document.write('Bonjour ' + prenom + '<br>');
}
```

Exemple

```
<script>
    var monNombre = prompt('Saisissez un nombre');
    // Appel de ma fonction
    afficherNombre(monNombre);
      afficherNombre(44);


    // Définition de ma fonction
    function afficherNombre(nombreAAfficher) {
        console.log(nombreAAfficher);
    }
</script>
```

Exercice

Définir une fonction qui demande son prénom à l'utilisateur, puis qui l'affiche dans la console. La fonction sera appelée lorsque l'on clique sur un bouton.

```
<button id="trigger_bonjour">Bonjour</button>
<script>
    // Évènement qui va déclencher l'appel de ma fonction
    document.getElementById('trigger_bonjour').addEventListener('click',
function() {
        askName();
    });


    // Définition de ma fonction
    function askName() {
        var name = prompt('Saisissez votre nom');
        console.log(name);
    }
</script>
```

Exercice

Prendre deux valeurs dans des inputs et prévoir deux boutons : un va multiplier les deux valeurs, l'autre va les diviser. On va utiliser un eventListener sur chaque bouton pour faire deux actions différentes, puis on va faire évoluer cette façon de faire pour n'utiliser qu'une seule fonction en prenant un paramètre.

```
<label for="nombre1">Nombre 1</label>
<input type="number" name="nombre1" id="nombre1"/>
<label for="nombre2">Nombre 2</label>
<input type="number" name="nombre2" id="nombre2"/>
<br><br><br>
<button id="multiply">
    Multiplier
</button>
<button id="divide">
    Diviser
</button>
<br><br><br>
<div id="resultat">

</div>
<script>
    document.getElementById('multiply').addEventListener('click', function() {
        var nombre1 = document.getElementById('nombre1').value;
        var nombre2 = document.getElementById('nombre2').value;

        var resultat = nombre1 * nombre2;

        document.getElementById('resultat').innerHTML = resultat;
    });

    document.getElementById('divide').addEventListener('click', function() {
        var nombre1 = document.getElementById('nombre1').value;
        var nombre2 = document.getElementById('nombre2').value;

        var resultat = nombre1 / nombre2;

        document.getElementById('resultat').innerHTML = resultat;
    });

</script>
```

Le code optimisé avec une seule fonction et différents appels :

```
    <script>

        function calcul(operation) {
            var nombre1 = document.getElementById('nombre1').value;
            var nombre2 = document.getElementById('nombre2').value;

            switch(operation) {
                case "multiplier":
                    var resultat = nombre1 * nombre2;
                    break;
                case "diviser":
                    var resultat = nombre1 / nombre2;
                    break;
                case "soustraire":
                    var resultat = nombre1 - nombre2;
                    break;
                case "additionner":
                    var resultat = +nombre1 + +nombre2;
                    break;
                default:
                    var resultat = "Petit malin tu n'as pas envoyé d'opération
!";
                    break;
            }

            document.getElementById('resultat').innerHTML = resultat;
        }


    </script>
```

L'instruction return

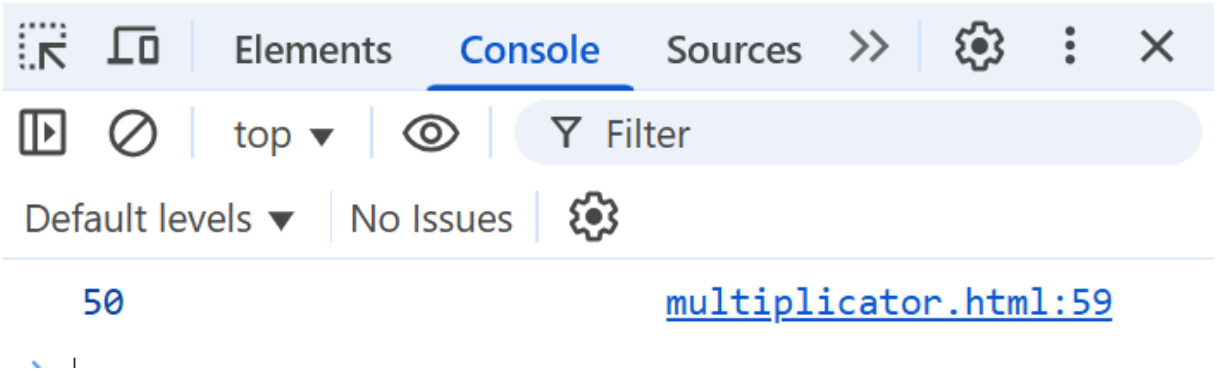
Il existe des fonctions qui en plus d'effectuer des **traitements**, permettent de **renvoyer une valeur**. Cette valeur peut être récupérée dans une variable, et ce qui est écrit après l'instruction **return** n'est **pas** interprété.

Exemple

```
function getResultatMultiplication(nombre1, nombre2) {
    var resultat = nombre1 * nombre2;
    return resultat;
}
```

```
var monResultat = getResultatMultiplication(5, 10);
console.log(monResultat);
```

Ce code va nous afficher le résultat du calcul en console :



Exercices

1. Écrire une fonction qui affiche une alert disant bonjour, puis appeler la fonction
2. Écrire une fonction qui affiche un prompt demandant le prénom, et souhaiter la bienvenue à la personne avec une alert
3. Écrire une fonction qui affiche « Click ! » dans le document, et déclencher l'appel de la fonction lorsque l'on clique sur un bouton HTML (**LÀ** vous pouvez utiliser la gestion d'évènements ;))
4. Écrire une fonction qui demande un nombre à l'utilisateur (utiliser la fonction **prompt**) et qui écrit ce nombre en console
5. Écrire une fonction qui demande son prénom à l'utilisateur, récupérer ce nom dans une variable et afficher le prénom dans la console

Exercices complémentaires : https://www.codingame.com/playgrounds/83722/exercice-javascript

Exercices complémentaires : https://www.w3schools.com/js/js_exercises.asp

6. Les fonctions de chaînes de caractères

Les chaînes de caractères, (type string), peuvent être manipulées à l'aide de fonctions permettant d'en extraire des informations. On peut par exemple en extraire la longueur, la couper en deux, l'éclater en plusieurs parties, en concaténer plusieurs…

length

La fonction length s'appelle directement sur la chaîne de caractère avec un point •

```
let text = "Hello World!";
let longueur = text.length;

console.log(longueur);
```

Ici, cet exemple nous affiche le chiffre 12 en console.

slice

Slice permet de découper une chaîne de caractère afin d'en extraire une **sous-chaîne**. La fonction admet un ou deux paramètres, le deuxième étant facultatif. Le premier paramètre correspond à la position de départ de la chaîne, le deuxième paramètre correspond à la position de fin (position calculée depuis le début). Si le deuxième paramètre est absent, le reste de la chaîne sera envoyé en entier.

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

console.log(str.slice(31));
// Expected output: "the lazy dog."

console.log(str.slice(4, 19));
// Expected output: "quick brown fox"

console.log(str.slice(-4));
// Expected output: "dog."

console.log(str.slice(-9, -5));
// Expected output: "lazy"
```

split

La fonction split permet d'**éclater** une chaîne de caractère en un tableau, en se basant sur un caractère précis qu'on lui passe en paramètre :

```
const str = "The quick brown fox jumps over the lazy dog.";
const words = str.split(" ");

console.log(words);
```

```
▼ Array(9) i
    0: "The"
    1: "quick"
    2: "brown"
    3: "fox"
    4: "jumps"
    5: "over"
    6: "the"
    7: "lazy"
    8: "dog."
    length: 9
  ▶ [[Prototype]]: Array(0)
```

join

La fonction join permet de construire une chaîne de caractère nouvelle à partir d'un tableau de chaînes de caractères. On lui passe en paramètre le caractère avec lequel concaténer les autres chaînes.

```
var bits = ['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd'];
alert(bits.join(''));  // Hello World
alert(bits.join(' '));  // H e l l o   W o r l d
```

7. Les attributs de formulaire

Avec les formulaires HTML et Javascript, nous pouvons passer les valeurs saisies par l'utilisateur d'un langage à l'autre en utilisant des attributs.

L'attribut **value** par exemple nous permet d'obtenir la valeur saisie dans un champ de type saisie, qu'il soit de type text ou number ou autre :

```
<input type="text" name="prenom" id="prenom"/>
<script>
    var valeurPrenom = document.getElementById('prenom').value;
```

L'attribut **checked** nous permet de vérifier si un élément est sélectionné dans une liste de boutons radio ou de checkbox :

```
<input type="radio" name="sexe" id="genderMale" value="H"/>
<input type="radio" name="sexe" id="genderFemale" value="F"/>
<script>
    // Ici mes variables prendront true ou false comme valeur
    var isCheckedMale = document.getElementById('genderMale').checked;
    var isCheckedFemale = document.getElementById('genderFemale').checked;
```

Pour récupérer l'ensemble de checkboxes cochées, on peut utiliser une boucle qui, pour chaque checkbox existante sur notre page, va vérifier si elle est cochée ou non. Voici un bout de code qui récupère toutes les valeurs d'une liste de checkbox pour en faire une liste :

```
<input type="checkbox" name="departements[]" value="42"/>
<input type="checkbox" name="departements[]" value="03"/>
<input type="checkbox" name="departements[]" value="63"/>
<input type="checkbox" name="departements[]" value="69"/>
<input type="checkbox" name="departements[]" value="38"/>
<button onclick="displayCheckedBoxes()">Bouton</button>
<script>
    function displayCheckedBoxes() {
        var checkedBoxes = document.getElementsByName('departements[]');
        var liste = "";
        for (var i=0, n=checkedBoxes.length;i<n;i++)
        {
            if (checkedBoxes[i].checked)
            {
                liste += ", "+checkedBoxes[i].value;
            }
        }
        if (liste.length > 0) {
            liste = liste.substring(1);
        }
        console.log(liste);
    }
```

Exercices

On est chez Subway

- Créer un formulaire avec des boutons radio proposant une liste d'ingrédients à choisir. Lorsque l'on choisit un ingrédient, l'afficher dans une div id=recette
- Transformer notre liste d'ingrédients en une liste de checkboxes et afficher dans la div **#recette** la liste complète des ingrédients sélectionnés séparés par des «, ».
- Si aucune case n'est cochée, afficher « Votre liste est vide »
- Rajouter une liste de type **select** « Sauces » dans laquelle on va mettre ketchup, barbecue, mayonnaise… Lorsque l'on choisit une sauce, rajouter dans notre div **#recette**. Notre div recette doit maintenant contenir à l'heure actuelle un message du type « Vos ingrédients sont : salade, tomates, jalapenos, avec de la sauce andalouse. »
- Rajouter un input de type **text** pour une saisie libre. L'utilisateur pourra alors faire part d'une instruction supplémentaire du genre « Pain sans gluten » ou « viande pas trop cuite ». Faire apparaître la consigne avec la phrase « Votre demande : … »