JavaScript ()

Pourquoi le JavaScript ?

Imaginez un site web comme un corps humain :

- Le HTML, c'est le squelette (la structure).
- Le CSS, c'est la peau, les vêtements, l'apparence (le style).
- Le JavaScript, c'est le cerveau et les muscles : il permet au site de réagir, de bouger, et d'interagir avec vous !

Le JavaScript est un langage incontournable en programmation web car il permet de rendre votre page web « intelligente ». C'est un langage qui permet notamment de :

Rendre une page dynamique : Réagir aux actions de l'utilisateur (clics, saisie, défilement),
Modifier le contenu : Changer du texte, des images, ou des styles CSS sans recharger la page,
Communiquer avec des serveurs : Récupérer des données en temps réel (ex : météo, actualités).
Créer des animations : Effets visuels, jeux, transitions fluides.

Lier la page web au code JS

Le code JS peut s'écrire directement dans le document .html mais il est préférable de l'écrire dans un document séparé, un document.js.

Afin de relier notre document .js à la page web, il est nécessaire de placer la balise suivante à la fin de votre document HTML, avant la balise `</html>` :

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

Déclaration de variable

Variable : Espace mémoire qui stocke une valeur modifiable, associée à un nom, permettant de manipuler des données dynamiquement dans un programme.

Constante : Valeur fixe qui ne peut être modifiée pendant l'exécution normale du programme.

Mot-clés	Re-déclaration	Mis à jour	Explications
var	**OK**	**OK**	Une variable déclarée avec le mot-clés « var » peut être redéclaré et mis à jour à n'importe quel moment dans le code.
let	**KO**	**OK**	Une variable déclarée avec le mot-clés « let » ne peut pas être redéclaré mais peut être mis à jour à n'importe quel moment dans le code.
const	**KO**	**KO**	Une variable déclarée avec le mot-clés « const » est utilisé pour déclarer une constante. Le variable ne peut pas être redéclaré ni mis à jour plus tard dans le code.

Exemple :

```
var variable1 = 0 ; // OK
var variable1 = 2 ; // OK
variable1 = 3 ; // OK
```

```
let variable2 = 0 ; // OK
let variable2 = 2 ; // KO
variable2 = 3 ; // OK
```

```
const variable3 = 0; // OK
const variable3 = 2 ; // KO
variable3 = 3 ; // KO
```

NB : Une variable peut stocker une valeur numérique comme du texte, un objet ou une suite de nombre ou de texte.

Les Array

Une array est comme une liste ordonnée où vous pouvez stocker plusieurs valeurs dans une seule variable.
Imaginez une boîte à compartiments :

- Chaque compartiment a un numéro (appelé index, en commençant par 0).
- Chaque compartiment peut contenir une valeur (un nombre, du texte, etc.).

Exemple :

```
// Une array qui stocke des fruits
let fruits = ["pomme", "banane", "orange"];
```

On a donc dans la variable `fruits`, stocker trois textes : « pomme » à l'index 0, « banane » à l'index 1 et « orange » à l'index 3.

Opérations de base

1. Créer une array

```
let notes = [12, 15, 18, 9]; // Tableau de notes
let couleurs = ["rouge", "vert", "bleu"];
```

2. Accéder à un élément

```
console.log(notes[0]); // Affiche 12 (premier élément)
console.log(couleurs[2]); // Affiche "bleu"
```

3. Modifier un élément

```
notes[1] = 16; // Remplace 15 par 16
console.log(notes); // [12, 16, 18, 9]
```

4. Ajouter un élément

```
couleurs.push("violet"); // Ajoute "violet" à la fin
console.log(couleurs); // ["rouge", "vert", "bleu", "violet"]
```

5. Supprimer un élément

```
couleurs.splice(1, 1); // Supprime 1 élément à partir de l'index 1 ("vert")
console.log(couleurs); // ["rouge", "bleu", "violet"]
```

6. Connaître la taille du tableau

```
console.log(couleurs.length); // Affiche 3 (nombre d'éléments)
```

Déclaration de fonction

Fonction : Bloc de code réutilisable qui effectue une tâche spécifique. Elle peut recevoir des données (paramètres), les traiter, et renvoyer un résultat. Une fois définie, elle peut être appelée plusieurs fois depuis différents endroits du programme.

C'est comme une machine à jus d'orange ! Tu mets des oranges (les paramètres) dedans, la machine fait son travail toujours de la même façon, et te donne un jus d'orange (le résultat). Tu peux l'utiliser autant de fois que tu veux avec différentes oranges.

Construction d'une fonction :

```
[mot clé] [nom de la fonction]() {
      [code]
}
```

Exemple :

```
function main() {
      console.log("Hello World!");
}
```

Modification du HTML et CSS

Récupération de l’élément HTML

Comme indiqué plus haut, le JavaScript peut être utilisé pour modifier le code HTML lui même ou de modifier les propriétés CSS d’un élément HTML. Pour ça la 1ère étape est de venir récupérer l’élément HTML qui nous intéresse grâce à l’une des fonctions suivantes :

Nom de fonction	Description	Exemple
`document.getElementById`	Permet de récupérer un élément HTML à partir de son ID.	`const btnPause = document.getElementById("btnPause");`
`document.getElementsByClassName`	Permet de récupérer les éléments HTML associé à une même Class.	`const testElements = document.getElementsByClassName("test");`
`document.getElementsByName`	Permet de récupérer les éléments HTML ayant un même attribut Name.	`const testElements = document.getElementsByName("test");`
`document.getElementsByTagName`	Permet de récupérer les éléments HTML de même type.	`const pElements = document.getElementsByTagName("p"); //on récupère tous les <p>`
`document.querySelector`	Retourne le premier élément du document HTML qui correspond au sélecteur CSS défini, ou à un groupe de sélecteurs CSS. Si aucune correspondance n'est trouvée, null est retourné.	`const el1 = document.querySelector(".maclasse"); const el2 = document.querySelector("#myId");`
`document.querySelectorAll`	Retourne les éléments du document HTML qui correspondent au sélecteur CSS défini, ou à un groupe de sélecteurs CSS. Si aucune correspondance n'est trouvée, null est retourné.	`const el1 = document.querySelector("div");`

Modification des propriétés CSS

Une fois l'élément HTML récupéré on peut modifier ses propriétés CSS grâce au format de fonction suivant :
`[elementHTML].style.[propriété] = "[valeur]"`

Exemple :

```
const btnPlay = document.getElementById("btnPlay");
btnPlay.style.visibility = "hidden";
```

NB : les noms des propriétés sont les même que ceux utilisés dans le CSS.

Modification du code HTML

Il est possible de modifier le code HTML à partir du JavaScript grâce aux fonction suivantes :

On considèrera pour le tableau ci-dessous, la balise HTML suivante :
`<button id="btnPause">Pause</button>`
Et la variable js suivante :
`let btnPause= document.getElementById("btnPause");`

Nom de fonction	Description	Exemple
Modifier le texte d'une balise		
`Element.textContent`	Permet de **modifier** le texte d'une balise	`btnPause.textContent = "Play";` `// la balise a été modifiée en <button id="btnPause">Play</button>`
`Element.insertAdjacentText`	Permet d'**ajouter** du texte à une balise	`btnPause.insertAdjacentText("afterbegin","Play");` `// la balise a été modifiée en <button`

		`id="btnPause">PlayPause</button>`
Modifier l'intérieur d'une balise		
`Element.innerHTML`	**Remplace** l'actuel contenue de la balise par un code HTML libre	`btnPause.innerHTML = "<img src='img.png'>"` `// la balise a été modifiée en <button id="btnPause"><img src='img.png'></button>`
`Element.insertAdjacentHTML`	**Ajoute** un code HTML dans la balise. L'emplacement de l'ajout est spécifié par les mots-clés suivants : `afterbegin, beforeend`	`btnPause.insertAdjacentHTML("beforeend", "<img src='img.png'>");` `// la balise a été modifiée en <button id="btnPause">Pause<img src='img.png'></button>`
Modifier la balise en elle-même		
`Element.outerHTML`	**Remplace** l'actuelle balise	`btnPause.outerHTML= "<img src='img.png'>"` `// la balise a été modifiée en <img src='img.png'>`
Changer l'extérieur de la balise		
`insertAdjacentHTML`	**Ajoute** un code HTML avant ou après la balise. L'emplacement de l'ajout est spécifié par les mots-clés suivants : `beforebegin, afterend`	`btnPause.insertAdjacentHTML("beforebegin", "<img src='img.png'>");` `// la balise a été modifiée en <img src='img.png'><button id="btnPause">Pause</button>`