

Documentation

Notre formation est basée sur la pédagogie active dans laquelle on a assisté à des work shop et des veilles dans lesquels on a retenu un bagage sur plusieurs langages tel que java Script, Html, Css. Au bout de cette documentation, je vais citer les bases de java Script.

**Java Script**

**Définition :**

JavaScript est un langage de programmation principalement utilisé pour créer des pages web interactives.

**Déclaration des variables :**

Définition :

Une variable est un espace de stockage pour une valeur qui peut être manipulée et modifiée au cours de l'exécution du programme.

La déclaration :

* Var : Le plus ancien mot-clé, mais à éviter dans la plupart  
  des cas en raison de sa portée fonctionnelle.
* Let : il a une portée de bloc (à privilégier).
* Const : Déclare une constante, une variable dont la valeur

ne peut pas être réassignée après sa déclaration.

Exemple :

// Utilisation de let : pour des variables dont la valeur peut changer

*Let* nom = "Alice";

console.log("Nom :", nom);       // Alice

// Modification de variables déclarées avec let

nom = "Bob";

console.log("Nouveau nom :", nom); // Bob

// Utilisation de const : pour des constantes (valeurs immuables)

*const* pi = 3.14;

console.log("Valeur de pi :", pi);

// Tentative de modifier une constante (provoquera une erreur)

// Utilisation de var (ancien mot-clé, peu recommandé)

*var* couleur = "rouge";

console.log("Couleur :", couleur);

// Re-déclaration avec var (possible mais à éviter)

*var* couleur = "bleu";

console.log("Nouvelle couleur :", couleur); // bleu

**Les types de données :**

**1. Types de données primitifs**

* **Number** : Représente les nombres (entiers et décimaux).
  + Exemple : 42, 3.14, -7.
* **BigInt** : Représente des nombres entiers très grands.
  + Exemple : 1234567890123456789012345678901234567890n.
* **String** : Représente une séquence de caractères.
  + Exemple : "Bonjour", 'JavaScript'.
* **Boolean** : Représente une valeur logique : true ou false.
  + Exemple :

*let* isActive = true;

* **Undefined** : Une variable déclarée mais non initialisée a la valeur undefined.
  + Exemple :

*let* x;

console.log(x); // undefined

* **Null** : Représente une valeur intentionnellement vide ou inexistante.
  + Exemple :

*let* y = null;

* **Symbol** : Représente une valeur unique et immuable utilisée comme identifiant.
  + Exemple

*let* sym = Symbol('id');

console.log(sym); // Symbol(id)

**2. Types objets**

* **Object** : Structure de base qui peut contenir des paires clé-valeur.
  + Exemple :

*let* person = { name: 'Ali', age: 25 };

* **Array** : Un objet spécialisé pour stocker des listes de valeurs.
  + Exemple :

*let* fruits = ['apple', 'banana', 'cherry'];

* **Function** : Les fonctions sont des objets en JavaScript.
  + Exemple :

*function* greet() {

    return 'Hello!';

  }

* **Date** : Représente des dates et heures.
  + Exemple :

*let* today = new *Date*();

### ****3. Opérations arithmétiques****

#### **a. Opérateurs de base**

* **Addition (+)** : Additionne deux nombres.  
  Exemple : 5 + 3 // Résultat : 8
* **Soustraction (-)** : Soustrait un nombre d’un autre.  
  Exemple : 5 - 3 // Résultat : 2
* **Multiplication (\*)** : Multiplie deux nombres.  
  Exemple : 5 \* 3 // Résultat : 15
* **Division (/)** : Divise un nombre par un autre.  
  Exemple : 6 / 3 // Résultat : 2
* **Modulo (%)** : Renvoie le reste de la division entière.  
  Exemple : 7 % 3 // Résultat : 1
* **Exponentiation (\*\*)** : Élève un nombre à une puissance.  
  Exemple : 2 \*\* 3 // Résultat : 8

#### **b. Opérateurs d’incrémentation et de décrémentation**

* **Incrémentation (++)** : Augmente une valeur de 1.  
  Exemple :

*let* x = 5;

x++; // Résultat : 6

* **Décrémentation (--)** : Diminue une valeur de 1.  
  Exemple :

*let* x = 5;

x--; // Résultat : 4

### ****4. Opérations logiques****

#### **a. Opérateurs logiques**

* **ET logique (&&)** : Retourne true si **toutes** les conditions sont vraies.  
  Exemple :

*let* x = true && false; // Résultat : false

* **OU logique (||)** : Retourne true si **au moins une** des conditions est vraie.  
  Exemple :

*let* x = true || false; // Résultat : true

* **NON logique (!)** : Inverse la valeur booléenne.  
  Exemple :

*let* x = !true; // Résultat : false

#### **b. Opérateurs de comparaison**

Ces opérateurs comparent des valeurs et retournent des résultats booléens (true ou false).

* **Égalité stricte (===)** : Vérifie si les valeurs et types sont égaux.  
  Exemple : 5 === 5 // Résultat : true  
  Exemple : 5 === "5" // Résultat : false
* **Différence stricte (!==)** : Vérifie si les valeurs ou types sont différents.  
  Exemple : 5 !== "5" // Résultat : true
* **Inférieur (<)** et **supérieur (>)** : Vérifient les relations numériques.  
  Exemple :

3 < 5;  // Résultat : true

10 > 8; // Résultat : true

* **Inférieur ou égal (<=)** et **supérieur ou égal (>=)**.

**Les boucles :**

**Définition :**

Une boucle permet d’exécuter une série d’instructions plusieurs fois de manière répétitive.

**Objectif :**

Réduire la répétition de code et automatiser des tâches répétitives.

**Types de boucles :**

**Boucle for :** la plus utilisée pour des itérations avec un nombre défini de répétitions.

***Structure de base*** : for (initialisation ;condition ;incrémentation){

…}

**Exemple :**

for (*let* i = 0; i < 5; i++) {

    console.log("Itération numéro", i);

}

**Boucle while :** utilisée quand le nombre des répétitions n’est pas connu d’avance

***Structure de base*** : while (condition) {…}

**Exemple :**

*let* compteur = 0;

while (compteur < 5) {

    console.log("Itération numéro",compteur);

    compteur++;

}

**Boucle do…while :** similaire au while ,sauf qu’elle exécute le bloc au moins une fois , meme si la condition est fausse.

**Exemple :**

*let* compteur = 0;

do {

    console.log("Itération numéro",compteur);

    compteur++;

} while (compteur < 5);

Astuces : Pour arrêter l’exécution des boucles infinies on clique sur  ctr+c

**Les conditions :**

**Définition :**

Les conditions en JavaScript permettent de prendre des décisions en fonction de certaines valeurs ou expressions.

**Types de conditions :**

**La condition if :** La structure if est utilisée pour exécuter un bloc de code si une expression est vraie.

**Exemple :**

*let* age = 18;

if (age >= 18) {

    console.log("Vous êtes majeur."); // Affiche "Vous êtes majeur."

}

**La condition if …else :** La structure if...else permet d’exécuter un bloc de code si une condition est vraie, et un autre bloc si elle est fausse.

Exemple:

*let* age = 16;

if (age >= 18) {

    console.log("Vous êtes majeur.");

} else {

    console.log("Vous êtes mineur."); // Affiche "Vous êtes mineur."

}

**La condition if …else if …else :** La condition else if permet d’ajouter des vérifications supplémentaires lorsque la première condition if n’est pas vraie.

Exemple:

*let* note = 85;

if (note >= 90) {

    console.log("Excellent !");

} else if (note >= 70) {

    console.log("Bien !"); // Affiche "Bien !"

} else {

    console.log("Besoin d'amélioration.");

}

**La condition switch :** La structure switch est utile lorsqu'il y a plusieurs valeurs possibles pour une variable. Elle permet de simplifier le code par rapport à plusieurs if...else if.

Exemple :

*let* jour = "mardi";

switch (jour) {

    case "lundi":

        console.log("Début de la semaine !");

        break;

    case "mardi":

        console.log("Deuxième jour !"); // Affiche "Deuxième jour !"

        break;

    default:

        console.log("Autre jour.");

}

**L’opérateur ternaire** :L’opérateur ternaire est une syntaxe courte pour une condition if...else. Elle est utilisée pour des conditions simples.

Exemple :

*let* age = 20;

*let* statut = (age >= 18) ? "majeur" : "mineur";

console.log(statut); // Affiche "majeur"

**Les fonctions :**

Définition :

Les **fonctions** sont des blocs de code réutilisables qui effectuent une tâche spécifique ou renvoient une valeur. Elles permettent de rendre le code plus modulaire et maintenable.

Syntaxe :

*function* nomDeLaFonction(*param1*, *param2*) {

    // Corps de la fonction

    return resultat; // Facultatif

}

Exemple :

*function* addition(*a*, *b*) {

    return *a* + *b*;

}

console.log(addition(5, 3)); // Affiche : 8

**La comparaison entre let et var :**

| **Caractéristique** | **var** | **let** |
| --- | --- | --- |
| **Portée** | Globale | Bloc- |
| **Redéclaration** | Autorisée | Non autorisée |
| **Boucles** | Peut causer des problèmes | Comportement attendu |

**Les fonctions prédifinies :**

**1. Fonctions liées aux chaînes de caractères (Strings)**

* **toUpperCase()** et **toLowerCase()** : Convertissent en majuscules ou minuscules.

console.log("hello".toUpperCase()); // "HELLO"

* **split(separator)** : Divise une chaîne en un tableau selon un séparateur.

console.log("a,b,c".split(",")); // ["a", "b", "c"]

**2. Fonctions liées aux nombres (Math)**

* Math.max(a, b, ...) et Math.min(a, b, ...) : Renvoient le maximum ou le minimum.

console.log(Math.max(1, 2, 3)); // 3

**3. Fonctions liées aux tableaux (Arrays)**

* **pop()** : Supprime le dernier élément.

arr.pop();

console.log(arr); // [1, 2]

* **map(callback)** : Applique une fonction à chaque élément et renvoie un nouveau tableau.

*let* doubled = [1, 2, 3].map(*x* *=>* *x* \* 2);

console.log(doubled); // [2, 4, 6]

* **filter(callback)** : Filtre les éléments selon une condition.

*let* even = [1, 2, 3, 4].filter(*x* *=>* *x* % 2 === 0);

console.log(even); // [2, 4]

* **reduce(callback, initialValue)** : Réduit un tableau à une seule valeur.

*let* sum = [1, 2, 3].reduce((*acc*, *val*) *=>* *acc* + *val*, 0);

console.log(sum); // 6

**4. Fonctions liées aux objets (Objects)**

* **Object.keys(obj)** : Renvoie un tableau des clés de l'objet.

*let* obj = { a: 1, b: 2 };

console.log(*Object*.keys(obj)); // ["a", "b"]

* **Object.values(obj)** : Renvoie un tableau des valeurs de l'objet.

console.log(*Object*.values(obj)); // [1, 2]

**Arrays & object :**

* 1. **Arrays (Tableaux) :**

Définition :

Un tableau est une collection **ordonnée** d'éléments, accessibles par leur **index** (numéroté à partir de 0).

Création d'un tableau :

// Tableau vide

*let* tableauVide = [];

// Tableau avec des éléments

*let* fruits = ["Pomme", "Banane", "Orange"];

Accès aux éléments :

console.log(fruits[0]); // "Pomme"

console.log(fruits[2]); // "Orange"

* 1. **Objects (Objets) :**

Définition :

Un objet est une collection **non ordonnée** de paires **clé/valeur**. Chaque clé doit être unique, et la valeur peut être de n’importe quel type.

Création d'un objet :

*let* personne = {

    nom: "Ali",

    age: 25,

    profession: "Développeur"

};

Accès aux valeurs :

**Notation par point**

console.log(personne.nom); // "Ali"

**Notation par crochets**

console.log(personne["age"]); // 25

**Différences principales entre Arrays et Objects :**

| **Caractéristique** | **Tableau (Array)** | **Objet (Object)** |
| --- | --- | --- |
| **Structure** | Ordonnée (indexées) | Non ordonnée (clé/valeur) |
| **Accès aux éléments** | Via un index (numérique) | Via une clé (nommée) |
| **Utilisation courante** | Listes d'éléments | Représenter des entités complexes |
| **Méthodes courantes** | push, pop, map, filter | Accès direct avec des clés, manipulation des paires clé/valeur |

**Les boucles for…in et for…of:**

| **Caractéristique** | **for...in** | **for...of** |
| --- | --- | --- |
| **Itère sur** | Les **clés** d’un objet ou les **indices** d’un tableau | Les **valeurs** des objets itérables |
| **Type de données supporté** | Objets, tableaux | Tableaux, chaînes, Maps, Sets, etc. |
| **Valeurs retournées** | Clés ou indices (comme chaînes) | Valeurs des éléments |
| **Usage typique** | Parcourir les propriétés d’un objet | Parcourir les éléments d’un tableau ou d’une structure itérable |

**Réalisée par :** SALMA BAKKOU