- ✓ Le JavaScript fait partie des langages web avec le HTML et le CSS
- ✓ JavaScript a été initialement créé pour rendre les pages web interactives.
- ✓ Pour coder en JavaScript, nous n'allons avoir besoin que d'un éditeur de texte
- ✓ Les programmes écrits dans ce langage sont appelés **scripts**. Ils peuvent être rédigés :
  - **Directement dans une page HTML**, puis exécutés automatiquement lors du chargement de la page à l'aide des balises **<script>...</script>**
  - **Dans un fichier externe portant l'extension .js**, qui sera ensuite appelé depuis la page Web au moyen de l'élément **<script>** et de son attribut **src**, lequel contient l'URL du fichier **.js**
- ✓ Les instructions doivent être séparées par un point-virgule (ou retour à la ligne)

1. Les variables et les constantes

- ✓ Le nom d'une variable doit obligatoirement commencer par une lettre ou un **underscore** (_)
- ✓ Le nom d'une variable peut comporter des lettres, des chiffres et les caractères _ et **$** (les espaces ne sont pas autorisés!)
- ✓ Les noms de variables sont sensibles à la casse.

Avec le mot-clé let

- ✓ Le mot-clé **let** permet de déclarer une nouvelle variable

```
let Nom_Variable [= valeur ];
```

Pour initialiser une variable en JS

```
let test = "Hello world !";
```

```
let test;
test = "Hello world !";
```

- ✓ Si une variable est déclarée avec le mot-clé **let**, il est impossible de la redéclarer. Dans le cas contraire, une erreur de syntaxe se produit.

```
let test = 'Hello world !';
let test = 'Goodbye world !';
```

→ **Identifier 'test' has already been declared**

Avec le mot-clé const

✓ Le mot-clé **const** permet de déclarer une constante : `const Nom_Constante = valeur ;`

✓ Les variables déclarées à l'aide de const ne peuvent pas être réassignées:

```
const x = 20 ;
x=30 ;
```

→ **erreur, on ne peut pas réaffecter la constante !**

✓ Une constante déclarée avec **const** à une portée de **bloc**

Remarques

❖ Il existe une ancienne méthode pour déclarer une variable en utilisant le mot-clé **var**. Cette pratique est à éviter dans les nouveaux projets, car elle peut entraîner des comportements imprévisibles.

❖ Portabilité :

Contexte de déclaration	Let et const
Dans un bloc {} (ex. if, for, { ... })	Accessible **uniquement dans le bloc**
Dans une fonction	Accessible **uniquement dans la fonction**
hors fonction et bloc	Accessible dans tout le fichier

❖ une variable déclarée **sans let, const ou var** devient automatiquement **globale**, même si elle est créée à l'intérieur d'une fonction.

2. Les types de données

a) Le type nombre (Number)

✓ Une variable javascript est de type **Number** si sa valeur est une valeur numérique (entier ou réel)

✓

Opérateurs arithmétiques	
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Reste de la division euclidienne

✓

Méthodes de l'objet Math (Math)	
abs(x)	Retourne la valeur absolue.
sqrt(x)	Retourne la racine carrée.
round(x)	Retourne l'entier le plus proche.
trunc(x)	Retourne la troncature entière d'un nombre en retirant sa partie décimale.
random()	Retourne un réel aléatoire dans [0, 1[.

Remarques

Pour générer un entier aléatoire compris dans l'intervalle **[a,b]** :

- Formule 1 ➔ **Math.trunc(Math.random() * (b - a + 1)) + a**
- Formule 2 ➔ **Math.round(Math.random() * (b - a)) + a;**

b) Le type booléen (boolean)

✓ Une variable javascript de type Booléen peut prendre la valeur vrai (true) ou faux (false)
✓

Opérateurs de comparaison	
= =	Egal
!=	Différent
>	Supérieur à
>	Supérieur ou égal à
<	Inférieur
<=	Inférieur ou égal à

✓

Opérateurs logiques	
&&	ET
||	OU
!	NON

c) Le type chaine de caractères (String)

✓ Pour déclarer une chaîne de caractères, on utiliser les guillemets (") ou apostrophe (').

✓ Une chaîne de caractères est indexée (le premier caractère est à **l'indice 0**)

✓ On peut accéder à chaque caractère en utilisant la notation avec crochets **[ ]** ou la méthode **charAt()**

✓ Les chaînes de caractères sont **immuables** (on peut lire des caractères via leurs indices, mais pas les changer)

✓ L'opérateur "+" permet de concaténer des chaines des caractères .

Les chaînes de caractères	
+	Opérateur de concaténation.
ch.**length**	Propriété qui retourne la longueur de ch.
ch.**indexOf**(ch [,p])	Retourne la position de la 1ère occurrence de ch1dans ch, en effectuant la recherche à partir de la position p (sinon -1).
ch.**lastindexOf**(ch1 [,p])	Retourne la position de la dernière occurrence de ch1dans ch à partir de la position p.
ch.**substring**(d, f)	Retourne une sous-chaîne de ch de la position d à la position f (non incluse).
ch.**replace**(ch1, ch2)	Retourne une chaîne dans laquelle la 1ère occurrence de ch1 dans ch est remplacée par ch2.
ch.**toLowerCase**()	Convertit tous les caractères de ch en minuscules.
ch.**toUpperCase**()	Convertit tous les caractères de ch en majuscules.
ch.**trim**()	Supprime tous les espaces existants au début et à la fin de ch.
String.fromCharCode(num, ..., numN)	Retourne une chaîne formée par la concaténation des résultats de conversion des codes ASCII passés en paramètres.

d) Le type tableau (array)

Pour déclarer un tableau, il faut utiliser l'instruction : `Nom_Tableau = new Array( ) ;`

✓ La méthode **.length** retourne la longueur d'un tableau

e) L'objet Date

✓ L'objet **Date** est un objet intégré en JavaScript qui stocke la **date et l'heure**.

✓ L'instruction qui permet de récupérer la date et heure actuelle est : **new Date( )**

Exemple : let maintenant = new Date() ;

⇨ maintenant= **Wed Aug 27 2025 11:53:19 GMT+0100 (UTC+01:00)**

✓ Pour convertir une chaine contenant une date valide en un objet Date : **new Date(chaine)**

Exemple : let date1 = new Date("2025/12/31") ;

⇨ Date1 = **Wed Dec 31 2025 01:00:00 GMT+0100 (UTC+01:00)**

Methode	Description
.getFullYear()	Permet de retourner l'année dans un objet Date
.getMonth()	Permet de retourner le numéro de mois dans un objet Date (0: janvier, 1: février ...)
.getDate()	Permet de retourner le numéro de jour dans un objet Date (1 à 31)
.toString()	Convertir un objet Date en chaine

3. Fonctions prédéfinis en javascript

Fonction	Description
isNaN(ch)	Retourne : ▪ true si ch ne contient pas un nombre ▪ false si ch contient un nombre **isNaN('123'); ➔ false** **isNaN('Hello'); ➔ true** **isNaN('Année 2025');➔ true**
Number(ch)	▪ Convertir une chaine en nombre ▪ Ignore les espaces en début et fin, mais pas à l'intérieur. ▪ Les virgules et lettres invalides → NaN. ▪ La notation scientifique (**E**) et hexadécimale (commence par **0x**) est reconnue. ▪ true → 1, false → 0, null → 0, undefined → NaN. **Number("10"); ➔ 10** **Number(" 10"); ➔ 10** **Number("10 "); ➔10** **Number(" 10 "); ➔10** **Number("10.33"); ➔10.33** **Number("10,33"); ➔NaN** **Number("10 33"); ➔NaN** **Number("Année 2025"); ➔NaN** **Number("10E3"); ➔ 10000** **Number("0xF"); ➔ 15**

Fonction	Description
pareseFloat(ch)	▪ Convertir une chaîne en nombre à virgule flottante (nombre décimal). ▪ Ignore les espaces au début et à la fin. ▪ Lit uniquement jusqu'au premier caractère non valide. ▪ Prend en charge la notation scientifique (E). ▪ La virgule n'est pas reconnue comme séparateur décimal **parseFloat("100.50") ; ➔ 100.5** **parseFloat("100,50") ; ➔ 100** **parseFloat("100") ; ➔ 100** **parseFloat("Année2025") ; ➔ NaN** **parseFloat("2025Année") ; ➔ 2025**
parseInt(ch, [b])	▪ Convertir une chaîne en nombre entier (en tronquant la partie décimale si nécessaire). ▪ Le paramètre optionnel b permet de définir la base de conversion. Par défaut, la base est 10. ▪ Ignore les espaces au début et à la fin. ▪ Tronque la partie décimale si présente. ▪ Lit jusqu'au premier caractère non valide, retourne le nombre lu. **parseInt ("100.5") ; ➔ 100** **parseInt ("aa2025") ; ➔ NaN** **parseInt ("FF") ; ➔ NaN** **parseInt ("FF",16) ; ➔ 255** **parseInt ("350", 2) ; ➔ NaN** **parseInt ("150", 2) ; ➔ 1** **parseInt ("111", 2) ; ➔ 7**
String (x)	Convertir le nombre x en chaine **String(2025) ➔ "2025";**

4. Les actions élémentaires simples

a) Actions de sortie

- **alert()** : affiche un texte dans une boîte simple (composée d'une fenêtre et d'un bouton OK)
- **document.write() :** affiche un texte dans le document, à la position de l'appel du script.

b) Action d'entrée

- **prompt() :** permet d'ouvrir une petite boîte de dialogue dans le navigateur pour demander une saisie à l'utilisateur, elle retourne la valeur saisie sous forme de chaîne de caractères (string).

5. Les structures conditionnelles :

a) Forme réduite :

```
if ( condition ) une_seule_instruction ;
Ou
if ( condition )
{
     instruction_1 ;
                .
                .
          instruction_N ;
}
```

b) Forme généralisée :

```
if ( condition_1 )
    { Bloc 1 ;}
else if ( condition_2 )
    { bloc 2 ;}
} else if ( condition_3 )
    { bloc 3 ;}
.
.
 else
    {  bloc N ;}
```

c) La structure switch :

```
switch ( expression ) {
   case valeur_1 :
                instruction(s);
                break;
   case valeur_2 :
                instruction(s);
                break;
.
.

   default :
                instruction(s);
}
```

Remarques :

- La condition doit être toujours placée entre parenthèses
- **break** permet de sortir de switch sort du switch (sinon le code continue dans le case suivant).
- La partie **default** dans la structure switch n'est pas obligatoire

6. Les structures itératives

a) Boucle for

```
for ( initialisation ; condition ; incrémentation )
{
bloc_instructions;
}
```

- **Initialisation** : Initialiser la valeur du compteur, exécutée une seule fois au début
- **Condition** : condition a évalué avant chaque itération. Tant que c'est **true,** la boucle continue
- **Incrémentation** : modifier la valeur de compteur , exécutée à la fin de chaque tour

b) Boucle while

```
do {
        bloc_instructions
} while ( condition(s) )
```

do : exécute le bloc au moins une fois. Ensuite, la condition(s) est testée :

- Si elle est vraie , la boucle recommence.
- Si elle est fausse , la boucle s'arrête.

c) Boucle do while

```
while ( condition )
{
bloc_instructions;
}
```

- La condition est testée avant chaque iteration.
- Si la condition est vraie , le bloc s'exécute.
- Si la condition est fausse , la boucle s'arrête immédiatement.
- Contrairement à **do...while**, il est possible que le bloc ne s'exécute jamais si la condition est fausse dès le départ.

7. Les fonctions :

Une définition de fonction est construite avec le mot-clé **function,** suivi par :

- Le nom de la fonction.
- Une liste d'arguments à passer à la fonction, entre parenthèses et séparés par des virgules.
- Les instructions JavaScript définissant la fonction, entre accolades, { }.
- Le mot-clé **return** sert à Renvoyer une valeur à l'endroit où la fonction est appelée et **arrêter l'exécution** de la fonction immédiatement.

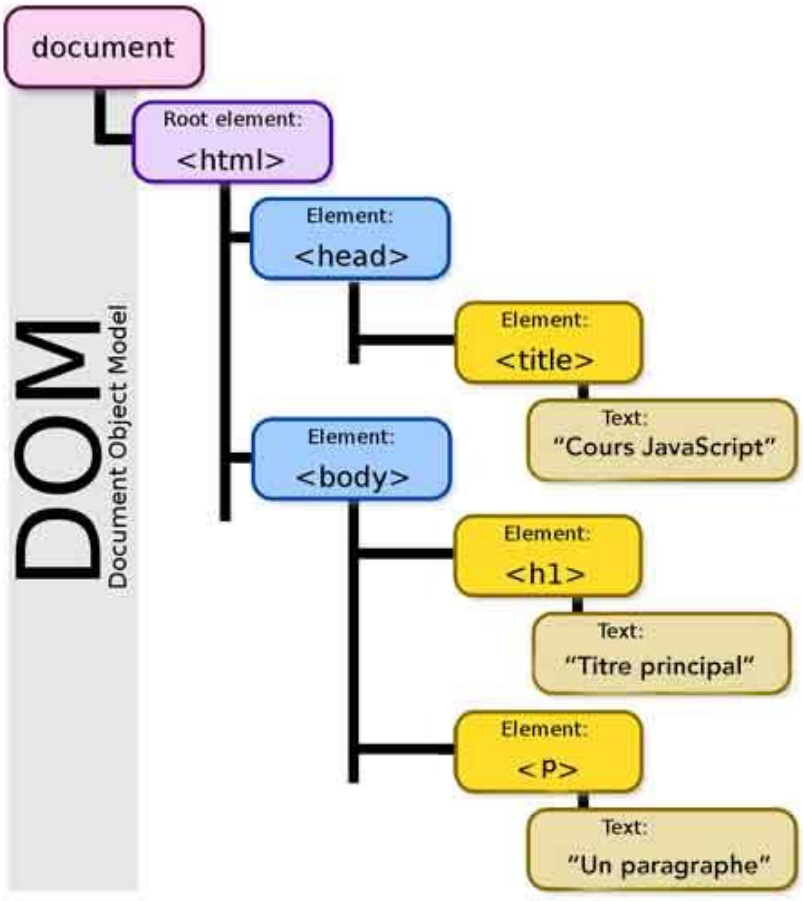
8. Document Object Model :

Le **DOM** (Document Object Model) est la représentation de la page HTML en mémoire, où chaque élément (titre, paragraphe, bouton…) devient un objet que JavaScript peut lire, modifier ou supprimer.

Soit code HTML suivant :

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Cours JavaScript</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Titre principal</h1>
        <p>Un paragraphe</p>
    </body>
</html>
```

Voici le DOM de cette page :



Grâce au DOM, le JavaScript peut :

- modifier le contenu et le style des éléments,
- créer ou supprimer des éléments,
- réagir aux événements de l'utilisateur (clic, saisie clavier, survol…).

9. Accéder aux éléments dans une page web

a) Accès par id

```
document.getElementById("Id_Elément");
```

b) Accès par name

```
document.getElementByName("Name_Elément");
```

- Cette méthode retourne une liste d'éléments, même s'il n'y a qu'un seul élément avec ce name.
- Pour accéder à un élément précis, on utilise l'indice [0], [1], etc.

10. Modifier le contenu d'un élément dans une page web

```
Elément. innerHTML=  Nouveau Contenu ;
```

11. Interagir avec les éléments d'un formulaire

a) Lire et modifier un champ

- Lire la valeur d'un champ

```
let variable = document.getElementById("Id_Champ").value;
```

- Modifier la valeur d'un champ

```
document.getElementById("Id_Champ").value = valeur;
```

- Rendre un champ en lecture seulement

```
document.getElementById("Id_Champ").readonly= true;
```

Remarque : readonly fonctionne **seulement sur certains éléments**, comme <input> (types texte, email, etc.)et <textarea>.

b) Travailler avec les cases à cocher et boutons radio

- Vérifier si la case est cocher

```
document.getElementById("Id_Case").checked
```

- cocher une case

```
document.getElementById("Id_Case").checked = true ;
```

- décocher une case

```
document.getElementById("Id_Case").checked = false ;
```

c) Travailler avec les listes déroulantes

- Récupérer l'indice de l'option sélectionnée dans une liste

```
document.getElementById("Id_liste").selectedIndex
```

- **selectedIndex** retourne un entier correspondant à l'indice de l'option sélectionnée.
- Le premier élément a l'indice 0.
- Si aucune option n'est sélectionnée, retourne -1.

- Récupérer la valeur de l'option sélectionnée dans une liste

```
document.getElementById("Id_liste").value
```

- **value** retourne la valeur de l'option actuellement sélectionnée.
- Si la valeur (value) n'est pas définie dans une <option>, **value** retourne le texte affiché de l'option.
- Si aucune option n'est sélectionnée, **value** retourne la valeur de la première option par défaut.

- Récupérer la valeur de l'option d'indice i dans une liste

```
let L= document.getElementById("Id_liste") ;
let valeur= L.options[i].value ;
```

d) Désactiver/Activer un élément dans le formulaire

- Désactiver un élément

```
document.getElementById("Id_Elément").disabled = true ;
```

- Activer un élément

```
document.getElementById("Id_Elément").disabled = false ;
```

12.Les événements

Les événements permettent de rendre les pages web interactives en répondant aux actions des utilisateurs.

a) onblur

- L'événement **onblur** se déclenche lorsqu'un élément **perd le focus**. Il est souvent utilisé dans les formulaires pour faire des vérifications dès que l'utilisateur quitte un champ.
- Utilisable sur input, textarea, select, button…

Exemple

```
<input type="text" id="nom" onblur="f1()" placeholder="Entrez votre nom">
```

⇨ Quand l'utilisateur tape quelque chose puis quitte le champ, la fonction f1() s'exécute automatiquement.

b) onfocus

- L'événement **onfocus** se déclenche lorsqu'un élément reçoit le focusse .
- Utilisable sur input, textarea, select, button…

Exemple

```
<input type="text" id="nom" onfocus="f1()" placeholder="Entrez votre nom">
```

⇨ Quand l'utilisateur clique dans la zone de texte , la fonction f1() s'exécute automatiquement.

c) onclick

- Se déclenche lorsqu'un utilisateur clique sur un élément
- Peut être appliqué sur presque tous les éléments HTML (bouton, image, paragraphe, etc.).

Exemple

```
< a id="lien1" onclick="f1()"  href="#"> Lien 1 </a>
```

⇨ Quand l'utilisateur clique sur ce lien , la fonction f1() s'exécute automatiquement.

Critère	onfocus	onclick
Quand déclenché ?	Quand l'élément reçoit le focus (clic **ou** navigation clavier avec Tab) ➔Détecte clavier + souris	Quand l'élément est cliqué (souris ou tap tactile) ➔Détecte uniquement **souris**
Éléments courants	Champs de formulaire (<input>, <textarea>, <select>)	Boutons, liens, paragraphe, image ….

d) oninput

- Se déclenche dès que l'utilisateur tape, colle, supprime ou modifie le contenu (il n'attend pas que le champ perde le focus)
- Il est utilisé principalement sur les champs de formulaire (<input>, <textarea>).

Exemple

```
<input type="text" id="nom" oninput="f1()" placeholder="Entrez votre nom">
```

⇨ À chaque caractère que l'utilisateur tape, la fonction f1() s'exécute automatiquement.

e) onchange

- Se déclenche lorsqu'un élément de formulaire change de valeur **ET** que l'utilisateur quitte cet élément (perd le focus).
- Il s'utilise avec des champs comme : <input> (text, number, checkbox, radio) , <textarea>, <select>

Exemple

```
<input type="text" id="nom" onchange="f1()" placeholder="Entrez votre nom">
```

⇨ f1() s'exécutera uniquement si la valeur du champ a changé ET que l'utilisateur quitte le champ

f) onload

- Se déclenche lorsque la page ou un élément est complètement chargé.

Exemple

```
<body onload="f1()">
```

⇨ La fonction f1() sera exécutée dès que la page HTML est entièrement chargée.