

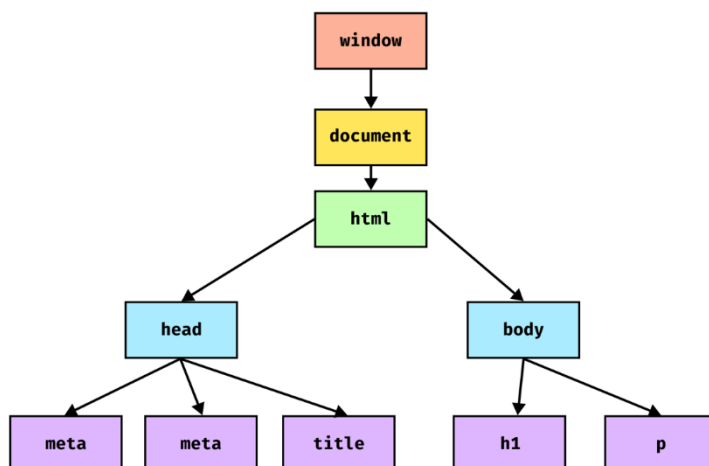
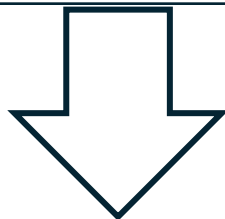
## I. Introduction :

JavaScript est un langage de programmation essentiel pour le développement web. Il est principalement utilisé pour ajouter de l'interactivité aux sites web, comme les animations, la validation de formulaires, ou les mises à jour dynamiques sans recharger la page.

Avec JavaScript, on peut manipuler le DOM (Document Object Model) pour modifier dynamiquement le contenu et le style d'une page et créer des applications web interactives.

Exemple arbre DOM :

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
<h1>Hello World!</h1>
<p>This is a paragraph element</p>
</body>
</html>
```



Abidi />  
É E MONJI  
M SBIBA

## II. Intégration de JavaScript dans le HTML :

Il existe plusieurs façons d'intégrer JavaScript dans un fichier HTML. Voici les méthodes principales :

### 1. intégration interne

Vous pouvez écrire du code JavaScript directement dans une balise `<script>` à l'intérieur du fichier HTML, généralement placée dans la section `<head>` ou avant la fermeture de `<body>`.

```
<!DOCTYPE html>
<html>
<head>
  <title>Exemple JavaScript</title>
  <script>
    // Exemple de code JavaScript
    alert("Bonjour depuis JavaScript !");
  </script>
</head>
<body>
  <h1>Intégration interne de JavaScript</h1>
</body>
</html>
```

### 2. intégration externe

Vous pouvez écrire le code JavaScript dans un fichier séparé (par exemple, `script.js`) et le lier au fichier HTML avec une balise `<script>` contenant l'attribut `src`.

*Fichier HTML*

```
<!DOCTYPE html>
<html>
<head>
  <title>Exemple JavaScript Externe</title>
  <script src="script.js"></script>
</head>
<body>
  <h1>Intégration externe de JavaScript</h1>
</body>
</html>
```

*Fichier `script.js`*

```
alert("Bonjour depuis un fichier externe !");
```

### 3. événements

Vous pouvez également écrire du JavaScript directement dans certains attributs HTML, comme onclick, onmouseover, etc. Cependant, cette méthode est moins recommandée, car elle mélange le code HTML et JavaScript.

```
<!DOCTYPE html>
<html>
<head>
  <title>Exemple JavaScript Inline</title>
</head>
<body>
  <h1>Exemple avec un événement JavaScript</h1>
  <button onclick="alert('Bonjour !')">Cliquez-moi</button>
</body>
</html>
```

### III. Affichage de données :

JavaScript offre plusieurs méthodes pour afficher du contenu, chacune adaptée à un usage spécifique. Voici les trois principales : alert, document.write, et console.log.

#### 1. alert

La méthode alert affiche un message dans une fenêtre contextuelle (popup) au navigateur. Cette fenêtre est modale, ce qui signifie qu'elle bloque l'interaction avec la page jusqu'à ce que l'utilisateur clique sur "OK".

#### Exemples :

```
alert("Bienvenue sur mon site !");
// Affiche une fenêtre avec le message "Bienvenue sur mon site !"
```

#### Points importants :

- ❖ Principalement utilisée pour des notifications ou des messages d'information simples.
- ❖ Elle peut être intrusive et gênante si elle est utilisée fréquemment.
- ❖ Ne permet pas de personnaliser la fenêtre.

#### 2. document.write

La méthode document.write écrit directement dans le document HTML. Elle peut être utilisée pour ajouter du contenu pendant le chargement de la page.

#### Exemples :

```
document.write("Bonjour, monde !");
// Affiche "Bonjour, monde !" directement dans la page.
```

**Points importants :**

- ❖ Si utilisée après le chargement initial de la page, elle remplace tout le contenu existant de la page.
- ❖ Rarement utilisée aujourd'hui, sauf pour des démonstrations ou des scripts très simples.

**IV. Déclaration et lecture des variables :****1. let**

**let** est un mot-clé qui permet de déclarer une **variable à portée de bloc**.

- ❖ **Portée** : bloc { }
- ❖ **Réassignable** : oui, on peut changer sa valeur
- ❖ **Non redéclarable** dans le même bloc
- ❖ **undefined**: Signifie qu'une variable n'a pas été initialisée
- ❖ **null** : Valeur vide.

**Exemple d'utilisation :**

```
let age = 25;
alert(age); // 25
age = 30; // Modification autorisée
alert(age); // 30
```

**2. const (pour les constantes)**

En JavaScript, **const** est un mot-clé qui permet de déclarer une **variable constante**, c'est-à-dire **une variable dont la valeur ne peut pas être réassignée** après sa déclaration.

- ❖ **Portée** : bloc { } (comme let)
- ❖ **Réassignation** : interdite
- ❖ **Mutabilité** : les objets et tableaux déclarés avec **const** **peuvent être modifiés**, mais pas réassignés.

**Exemple d'utilisation :**

```
const PI = 3.14;
alert(PI); // 3.14
// PI = 3.14159; // Erreur : une constante ne peut pas être modifiée
```

**3. Lecture des variables avec prompt**

La méthode **prompt** permet d'afficher une boîte de dialogue qui demande à l'utilisateur de saisir une valeur. Cette méthode est utilisée pour lire des entrées utilisateur directement depuis le navigateur.

**Syntaxe de prompt**

```
let variable = prompt(message, valeurParDefaut);
```

- ❖ **message** : Le message affiché dans la boîte de dialogue pour demander une saisie à l'utilisateur.
- ❖ **valeurParDefaut** (*facultatif*) : Une valeur initiale affichée dans le champ de saisie. Si elle n'est pas définie, le champ sera vide par défaut.

## V. typage dans JavaScript

### 1. Number (Nombres)

En JavaScript, les nombres sont représentés par le type number, qu'il s'agisse d'entiers ou de nombres à virgule flottante.

#### Exemples de déclaration :

```
let entier = 42;           // Entier
let decimal = 3.14;       // Nombre décimal
let negatif = -10;        // Négatif
let nan = NaN;            // "Not a Number"
```

#### a. Opérations arithmétiques :

Opérateur	Description	Exemple	Résultat
+	Addition	5 + 3	8
-	Soustraction	5 - 3	2
*	Multiplication	5 * 3	15
/	Division	15 / 3	5
%	Modulo (reste de la division)	5 % 3	2
++	Incrémentation (ajoute 1)	let x = 5; x++;	x = 6
--	Décrémentation (retire 1)	let x = 5; x--;	x = 4

#### b. Objet Math :

Méthode	Description	Exemple	Résultat
Math.abs(x)	Valeur absolue de x	Math.abs(-5)	5
Math.sqrt(x)	Racine carrée de x	Math.sqrt(16)	4
Math.round(x)	Arrondi à l'entier le plus proche	Math.round(4.6)	5
Math.trunc(x)	Partie entière de x (supprime les décimales)	Math.trunc(4.9)	4
Math.random()	Renvoie un nombre aléatoire compris entre 0 (inclus) et 1 (exclus)	Math.random()*5	nombre entre 0 et 5 (exclus)

## c. Conversion vers un nombre :

Méthode	Description	Exemple	Résultat
<b>Number(value)</b>	Convertit une valeur en nombre (entier ou flottant).	Number("42")	42
		Number("42.5")	42.5
		Number("abc")	NaN
<b>parseInt(string)</b>	Convertit une chaîne en <b>entier</b> (base 10 par défaut).	parseInt("42")	42
		parseInt("42.9")	42
<b>parseInt(string, radix)</b>	Convertit une chaîne en <b>entier</b> selon la base spécifiée (radix).	parseInt("1010", 2)	10
		parseInt("FF", 16)	255
		parseInt("77", 8)	63
<b>parseFloat(string)</b>	Convertit une chaîne en <b>nombre flottant</b> .	parseFloat("42.5")	42.5
		parseFloat("abc 42.5")	NaN

## 2. String (Chaînes de caractères)

Les chaînes de caractères sont des séquences de caractères, délimitées par des guillemets simples ('), doubles (") ou des backticks (` `). Elles sont immuables : toute modification génère une nouvelle chaîne.

## Exemple de déclaration :

```
let chaine = "Bonjour, JavaScript!";
```

## Opération sur les chaînes :

Méthode / Propriété	Description	Exemple	Résultat
+	Opérateur de concaténation de chaînes	"Hello " + "World"	"Hello World"
ch.length	Retourne la longueur de la chaîne ch	"Hello".length	5
ch.indexOf(ch1 [, p])	Position de la <b>1ère occurrence</b> de ch1 dans ch, recherche à partir de p (optionnel)	"Hello".indexOf("l")	2
ch.lastIndexOf(ch1 [, p])	Position de la <b>dernière occurrence</b> de ch1 dans ch, recherche à partir de p (optionnel)	"Hello".lastIndexOf("l")	3

Méthode / Propriété	Description	Exemple	Résultat
ch.substring(d, f)	Retourne une <b>sous-chaîne</b> de ch de la position d à f (non incluse)	"Hello".substring(1,4)	"ell"
ch.replace(ch1, ch2)	Remplace la <b>1ère occurrence</b> de ch1 par ch2 dans ch	"Hello".replace("l","L")	"HeLlo"
ch.toLowerCase()	Convertit tous les caractères de ch en <b>minuscules</b>	"HELLO".toLowerCase()	"hello"
ch.toUpperCase()	Convertit tous les caractères de ch en <b>majuscules</b>	"hello".toUpperCase()	"HELLO"
ch.trim()	Supprime les <b>espaces</b> au début et à la fin de ch	" Hello ".trim()	"Hello"
String.fromCharCode(num m, ..., numN)	Crée une chaîne à partir des <b>codes ASCII</b> passés en paramètre	String.fromCharCode(72,101,108,108,111)	"Hello"

### 3. Les booléens :

Un booléen est un type de donnée en JavaScript qui ne peut avoir que deux valeurs :

- ❖ true (vrai)
- ❖ false (faux)

Les booléens sont souvent le résultat d'une opération de comparaison ou logique. `false`

#### a. Opérateurs de comparaison

Les opérateurs de comparaison permettent de comparer des valeurs. Ils renvoient toujours une valeur booléenne (true ou false).

Opérateur	Description	Exemple	Résultat
==	Égalité (valeur seulement)	5 == "5"	true
!=	Différent (valeur seulement)	5 != "5"	false
>	Supérieur	10 > 5	true
<	Inférieur	5 < 10	true
>=	Supérieur ou égal	10 >= 10	true
<=	Inférieur ou égal	5 <= 10	true

**b. Opérateurs logiques :**

Opérateur	Nom	Description
&&	ET logique	Renvoie true si toutes les conditions sont vraies
	OU logique	Renvoie true si au moins une condition est vraie
!	NON logique (négation)	Renverse la valeur : true devient false et vice-versa

**4. Objet Date :**

L'objet **Date** en JavaScript permet de **travailler avec les dates et les heures**. On peut créer un objet date, récupérer ses composants (jour, mois, année) et afficher la date sous différentes formes.

Méthode / Syntaxe	Description	Exemple	Résultat
new Date()	Crée un objet Date représentant <b>la date et l'heure courante</b>	<code>let d = new Date();</code>	d contient la date et l'heure système
new Date(ch)	Crée un objet Date à partir d'une <b>chaîne de caractères</b> représentant une date	<code>let d= new Date("2025-08-19")</code>	"Tue Aug 19 2025 00:00:00 GMT+0100"
d.getDate()	Retourne le <b>jour du mois</b> (1 à 31)	<code>d.getDate()</code>	19
d.getMonth()	Retourne le <b>numéro du mois</b> (0 = janvier, 11 = décembre)	<code>d.getMonth()</code>	7
d.getFullYear()	Retourne l' <b>année sur 4 chiffres</b>	<code>d.getFullYear()</code>	2025
d.toString()	Retourne la <b>date complète sous forme de chaîne</b>	<code>d.toString()</code>	"Mon Aug 19 2025 09:00:00 GMT+0100"

**5. Array (Tableaux)**

En JavaScript, un tableau (ou Array) est une collection ordonnée de valeurs. Ces valeurs peuvent être de n'importe quel type (numériques, chaînes de caractères, objets, autres tableaux, etc.).

**a. Créer un tableau vide :**

```
let tableauVide = [];
```

**b. Créer un tableau avec des valeurs initiales :**

```
let fruits = ["pomme", "banane", "cerise"];
```



**c. accéder à un élément du tableau :**

```
let fruits = ["pomme", "banane", "cerise"];
fruits[2]= "orange" ;
alert(fruits[2]) // orange
```

**d. taille d'un tableau (length)**

```
let fruits = ["pomme", "banane", "cerise"];
alert(fruits.length) // 3
```

**VI. Les structures de contrôle conditionnelles :**

Les structures conditionnelles permettent d'exécuter des instructions spécifiques en fonction de la validité d'une ou plusieurs conditions.

Structure	Description	Exemple simple
if	Exécute un bloc de code si la condition est vraie	if(x > 0) { alert("Positif"); }
if...else	Exécute un bloc si vrai, un autre bloc si faux	if(x > 0) { alert("Positif"); } else { alert("Non positif"); }
if...else if...else	Permet tester plusieurs conditions successives	if(x > 0){ alert("Positif"); } else if(x < 0){ alert("Négatif"); } else{ alert("Zéro"); }
switch	Compare une expression à plusieurs valeurs possibles	<pre>switch(month) {   case 9:case 10 :case 11: case 12 :     alert("1er trimestre");     break;   case 1:case 2:case 3:     alert("2ème trimestre");     break;   case 4:case 5:case 6:     alert("3ème trimestre");     break;   default:     alert("Mois invalide"); }</pre>

**VII. Structures itératives**

Les **structures itératives** permettent d'exécuter un bloc de code **plusieurs fois**, tant qu'une condition est vraie ou selon un compteur.

Structure	Description	Exemple simple
for	Exécute un bloc de code un nombre déterminé de fois	<code>for (let i=1; i&lt;=3; i++) {   alert(i); } </code>
while	Exécute un bloc <b>tant que la condition est vraie</b>	<code>let i=1; while (i&lt;=3) {   alert(i); i++; } </code>
do...while	Exécute un bloc <b>au moins une fois</b> , puis répète tant que la condition est vraie	<code>let i=1; do{ alert(i); i++; } while (i&lt;=3); </code>

## VIII. Les fonctions

Une **fonction** en JavaScript est un **bloc de code réutilisable** qui peut être appelé plusieurs fois. Elle peut recevoir des **paramètres** et retourner une **valeur**.

### Syntaxe d'une fonction :

```
function nomDeLaFonction(param1, param2, ...){
  // Instructions à exécuter
  return valeur; // (optionnel)
}
```

### Exemple simple :

```
function addition(a, b){
  return a + b;
}
alert(addition(4, 6)); // Résultat : 10
```

## IX. Manipulations sur les éléments HTML

### 1. Ciblage des éléments :

Pour manipuler un élément HTML dans une page, on doit d'abord le **cibler**. Il existe plusieurs façons de le faire, les plus simples étant via **id** et **name**.

#### a. Par identifiant :

Le ciblage par **id** se fait avec `document.getElementById("id")`. Cette méthode retourne un seul élément car l'attribut **id** doit être unique dans la page. On peut ensuite accéder ou modifier son contenu.

### Exemple :

```
<h1 id="titre">Bienvenue</h1>
<h1 id="titre2"> Contact </h1>
<script>
  let t = document.getElementById("titre");
  alert(t.innerHTML); // Affiche : Bienvenue
</script>
```

**b. Par nom :**

Le ciblage par **name** se fait avec `document.getElementsByName("nom")`. Cette méthode retourne un tableau car plusieurs balises peuvent partager le même attribut name. Pour accéder à un élément précis, il faut utiliser un indice comme 0,1,... , `tableau.length-1`.

```
<input type="radio" name="genre" value="M">Malle
<input type="radio" name="genre" value="F">Femelle

<script>
  let champ = document.getElementsByName("email")[1];
  alert(champ.value); // Affiche : F
</script>
```

**2. Manipulation de contenu d'un élément innerHTML :**

La propriété **innerHTML** permet de récupérer ou de modifier le contenu HTML d'un élément ciblé. On l'utilise après avoir sélectionné l'élément (par id, name, etc.).

La propriété **innerHTML** permet de récupérer ou de modifier le contenu HTML d'un élément ciblé. On l'utilise après avoir sélectionné l'élément (par id, name, etc.).

```
<p id="paragraphe">Texte original</p>
<button onclick="changer()">Changer le texte</button>
<script>
  function changer() {
    let p = document.getElementById("paragraphe");
    p.innerHTML = "Texte modifié avec JavaScript";
  }
</script>
```

Texte original

Changer le texte



Texte modifié avec JavaScript

Changer le texte

si l'utilisateur clic sur le bouton

**3. Modification des attributs :**

En JavaScript, on peut directement modifier la valeur d'un attribut d'un élément HTML en utilisant la syntaxe :

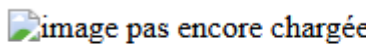
```
element.attribut = "valeur";
```

**Exemple d'utilisation :**

```


<button onclick="changer()">Changer l'image</button>
<script>
  function changer() {
    let img = document.getElementById("photo");
    img.src = "image2.jpg"; // changement d'attribut src
  }
</script>

```



**4. Modification du style**

En JavaScript, on peut accéder directement au style d'un élément HTML et changer ses propriétés CSS avec la syntaxe :

```
element.style.propriété = "valeur";
```

Remarque : en JavaScript, les propriétés CSS qui contiennent un tiret (background-color, font-size) doivent être écrites en **camelCase** :

- ✓ backgroundColor
- ✓ fontSize

**Exemple d'utilisation :**

```

<p id="texte">Bonjour tout le monde !</p>
<button onclick="changerCouleur()">Changer couleur</button>
<script>
  function changerCouleur() {
    let p = document.getElementById("texte");
    p.style.color = "red";           // couleur du texte
    p.style.fontSize = "20px";      // taille de la police
    p.style.backgroundColor = "yellow"; // couleur de fond
  }
</script>

```

Bonjour tout le monde !



Bonjour tout le monde !

## 5. Méthodes principales pour contrôler une vidéo/audio

En JavaScript, on peut cibler un élément <video> ou <audio> et utiliser ses méthodes intégrées :

Syntaxe	Description
element.play()	Lance la lecture du média.
element.pause()	Met en pause la lecture du média.

### Exemple d'utilisation :

```
<video id="maVideo" width="320" height="240" controls>
  <source src="exemple.mp4" type="video/mp4">
  Votre navigateur ne supporte pas la vidéo.
</video>
<br>
<button onclick="lire()">Lire</button>
<button onclick="pause()">Pause</button>
<script>
  let video = document.getElementById("maVideo");
  function lire() {
    video.play();    // jouer la vidéo
  }
  function pause() {
    video.pause();   // mettre en pause
  }
</script>
```



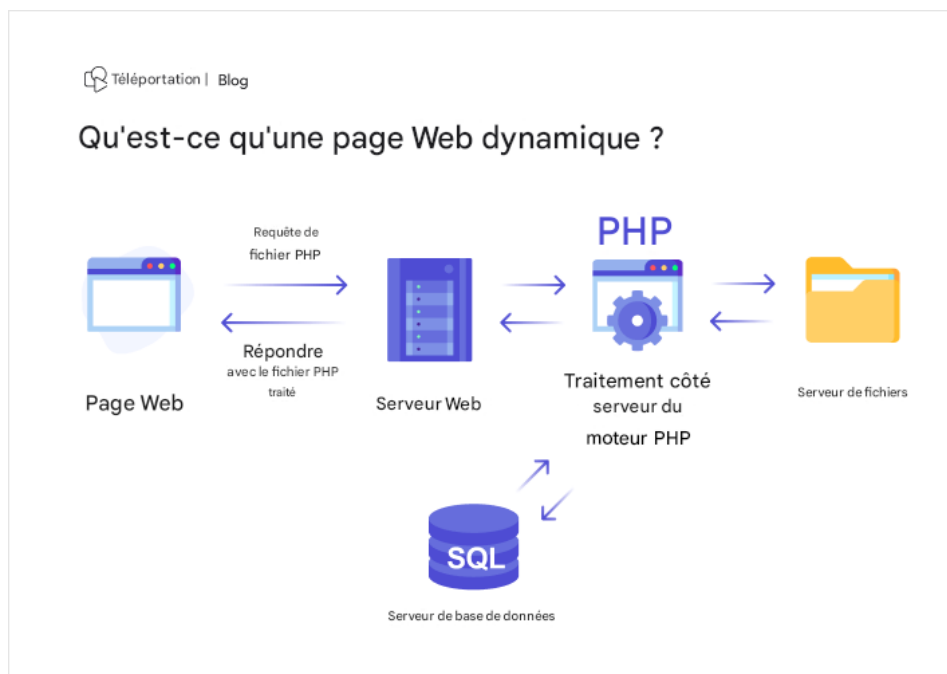
## I. Introduction

PHP (**Hypertext Preprocessor**) est un langage de programmation côté serveur, spécialement conçu pour le développement web. PHP est souvent utilisé en conjonction avec des technologies comme HTML, CSS, JavaScript, et des bases de données, formant un socle solide pour la majorité des applications web modernes.

Les principales utilisations de PHP incluent :

- ❖ **Création de sites dynamiques** : Génération de contenu en fonction des préférences des utilisateurs.
- ❖ **Gestion des formulaires** : Collecte, validation et traitement des données saisies par les utilisateurs.
- ❖ **Interaction avec des bases de données** : Gestion des données stockées, comme les utilisateurs, les produits, ou les articles.

## II. Fonctionnement d'un site web dynamique en PHP



PHP est un langage de programmation côté serveur, ce qui en fait un outil idéal pour créer des sites web dynamiques. Voici les étapes qui expliquent comment un site web dynamique fonctionne avec PHP :

### 1. Requête de l'utilisateur :

L'utilisateur envoie une requête HTTP (par exemple, en saisissant une URL ou en soumettant un formulaire). Cette requête est envoyée au serveur web (Apache dans notre cas).

### 2. Traitement par le serveur web :

- Le serveur web identifie le fichier PHP correspondant à la requête (par exemple, index.php).