ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

HONORIS UNITED UNIVERSITIES

Développement Web:

JavaScript

Professeur:

Nouhaila MOUSSAMMI

n.moussammi@emsi.ma

2024/2025

Plan

01 **Introduction à JavaScript**
Définition, Historique et importance dans le développement web

02 **Comparaison avec un langage compilé**
Langage interprété, langage compilé

03 **Architecture Client/serveur**
Rôle de JavaScript côté client et serveur

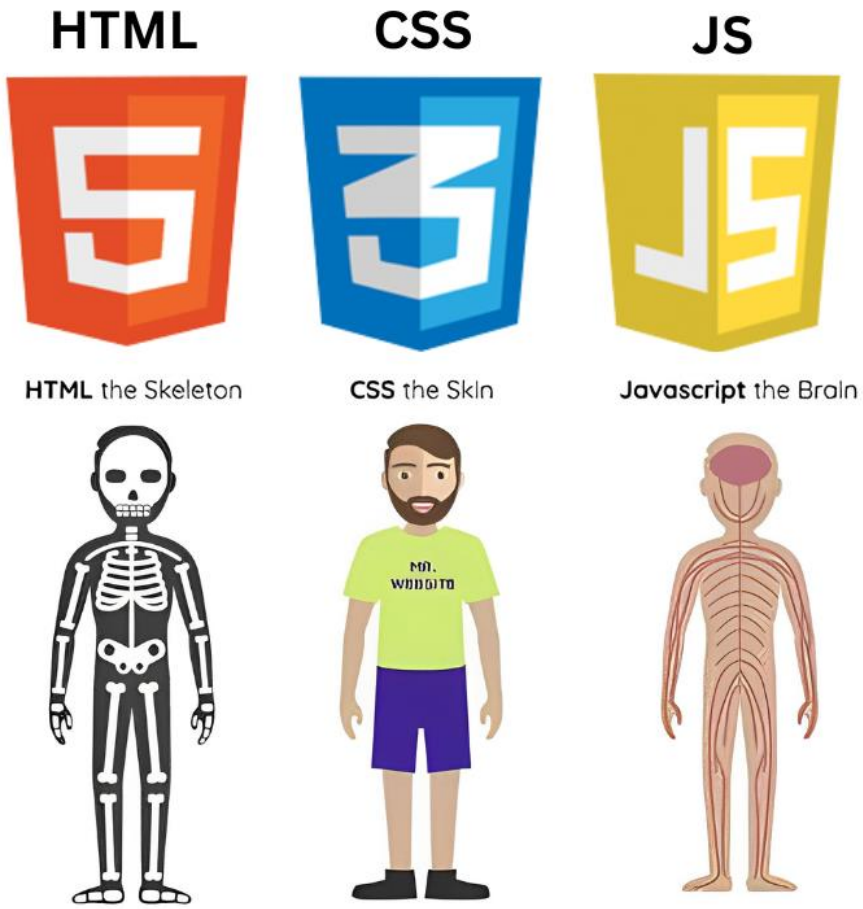
04 **Écosystème de Développement**
Présentation des outils et bibliothèques populaires

05 **Syntaxe de Base en JavaScript**
Variables, types de données, opérateurs.

Pourquoi apprendre HTML, CSS et JavaScript ?

- ✓ **HTML :** L'épine dorsale de chaque page web, elle définit la structure et le contenu.
- ✓ **CSS :** Les pages web avec des styles, des mises en page et des conceptions réactives.
- ✓ **JavaScript :** Ajoute de l'interactivité, améliore l'expérience utilisateur et permet du contenu dynamique.

Notez que : HTML et CSS ne sont pas considérés comme des langages de programmation car ils n'impliquent pas d'opérations logiques, telles que des conditionnels, des boucles ou des fonctions.



- JavaScript
  - Langage de script orienté objet, principalement pour des pages web interactives.
  - Permet de manipuler le **Document Object Model (DOM)**, c'est-à-dire la structure d'une page web.
  - Réagit aux actions de l'utilisateur (clics, saisie de texte, survols de souris).
- Historique
  - Créé en 1995 par Netscape pour ajouter de l'interactivité aux pages HTML.
  - Premier langage de script largement adopté pour le web.
  - Utilisé par les navigateurs actuels et fait partie des standards du web.
- Comparaison avec HTML
  - HTML structure le contenu d'une page.
  - JavaScript apporte la logique et l'interactivité.
- Qualités
  - Gratuit et disponible sur tous les navigateurs modernes.
  - Simple à apprendre pour des débutants en programmation.
- Défauts
  - Langage interprété, donc parfois lent pour des calculs complexes
  - Peut être difficile à déboguer (parfois, les erreurs ne s'affichent pas clairement)

- Un **langage compilé** est un langage dont le code source est traduit directement en code machine par un compilateur avant d'être exécuté (exemple : C, C++).

- Un **langage interprété**, comme JavaScript, est traduit et exécuté à la volée par un interpréteur (exemple : javaScript, Python, etc).

- **Différence :** Un compilateur convertit le code en code machine (crée un exe) avant l'exécution du programme. L'interpréteur convertit le code en code machine, ligne par ligne, au moment d'exécution du programme.

Langage compilé

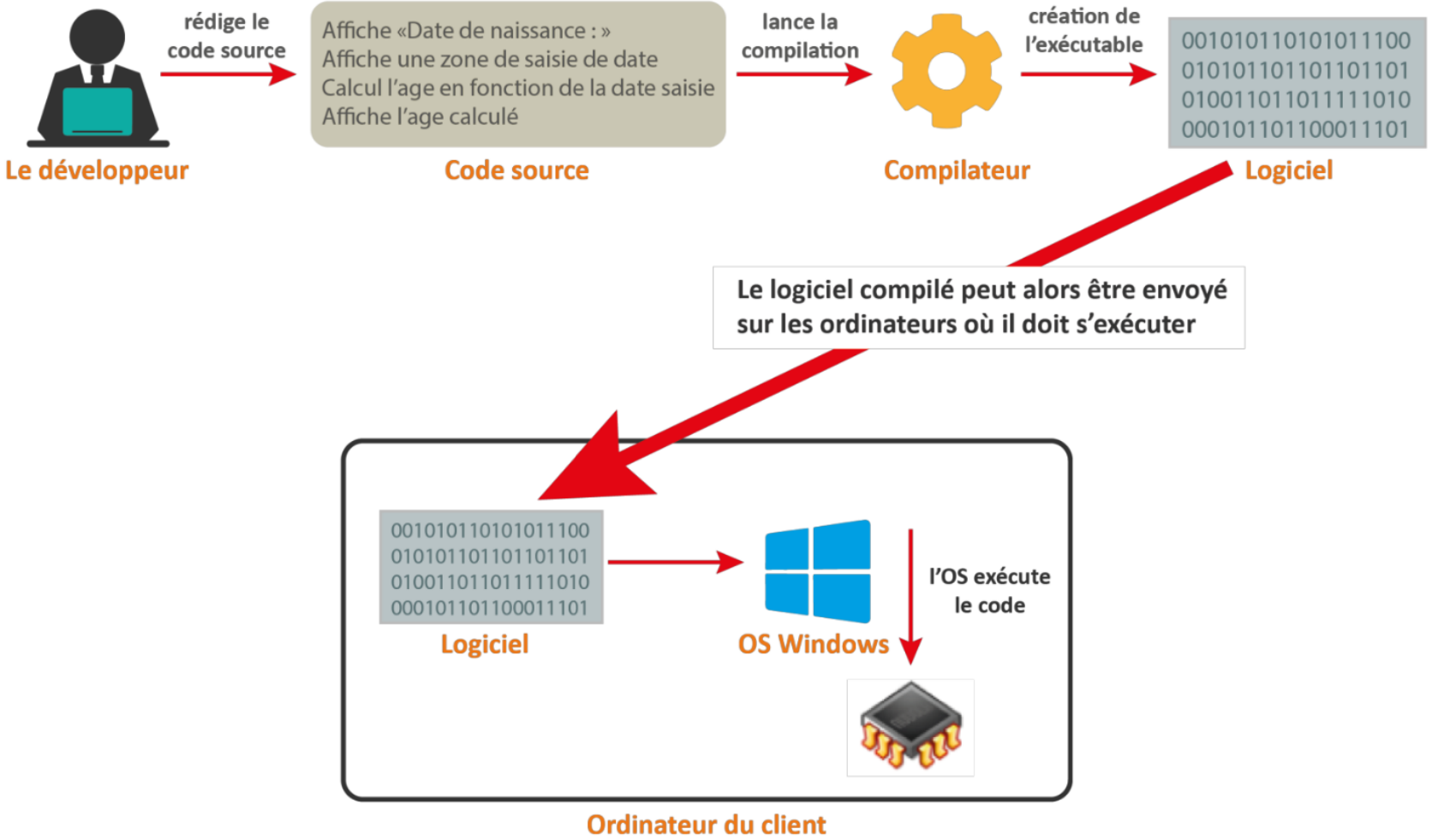


Figure 1 : Fonctionnement d'un langage compilé

Langage interprété

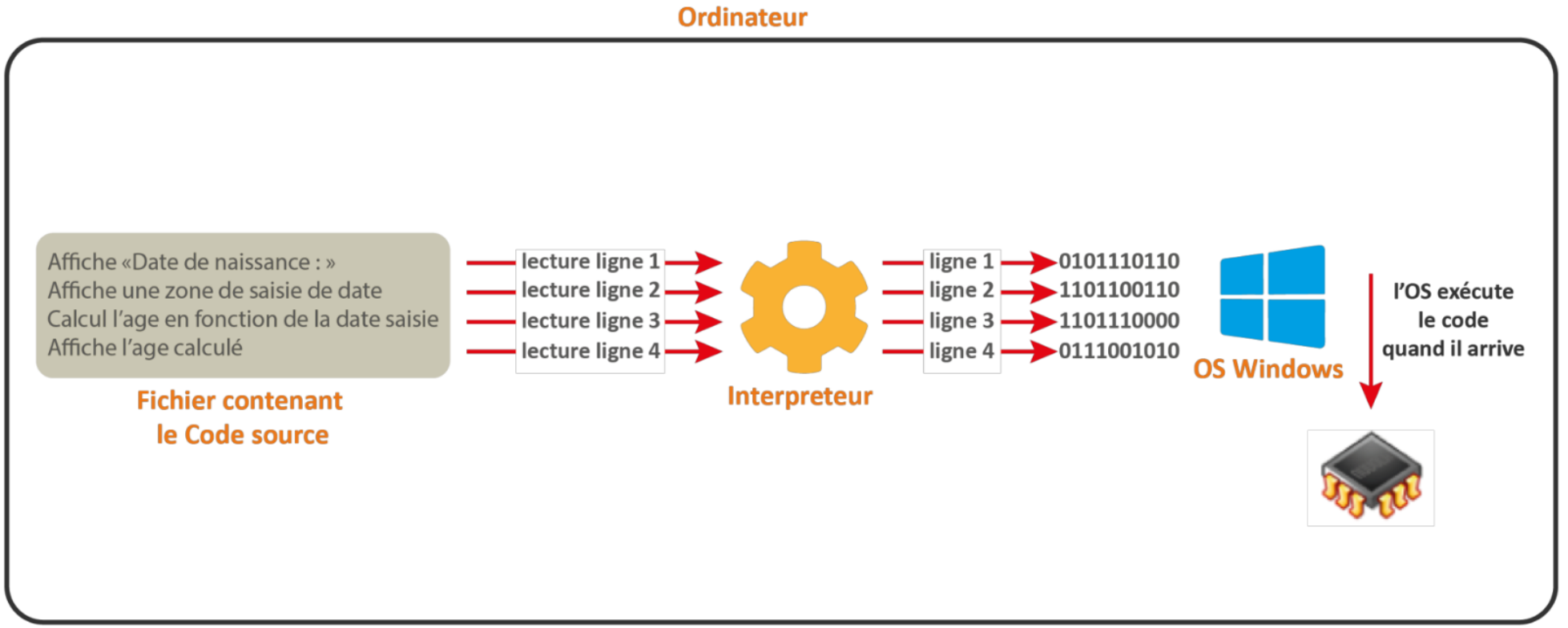


Figure 2 : Fonctionnement d'un langage interprété

Le rôle de l'interpréteur

- Le fonctionnement des langages de script est assuré par l'interpréteur. Son rôle réside dans la traduction des instructions du programme source en code machine.
- S'il y a une erreur dans l'instruction courante, l'interpréteur termine son processus de traduction à cette instruction et affiche un message d'erreur. L'interprète ne passe à la ligne d'exécution suivante qu'après avoir éliminé l'erreur.
- Un interpréteur exécute directement les instructions écrites dans un langage de script sans les convertir préalablement en code objet ou en code machine.

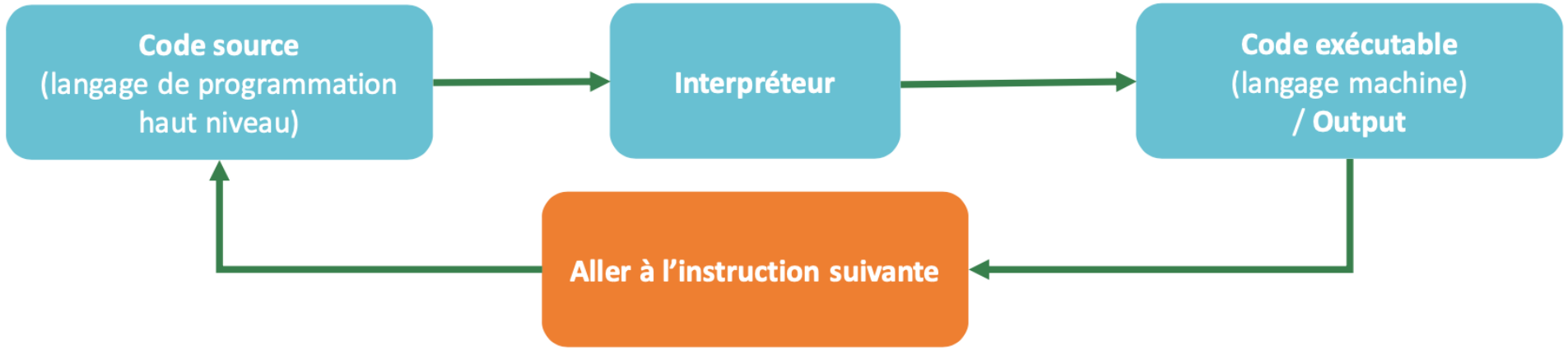


Figure 3: Fonctionnement d'un langage de script [1]

- Un **langage de script** est conçu pour **automatiser des tâches spécifiques** ou pour exécuter des scripts (instructions) dans un environnement particulier (comme un navigateur web,…, etc.).
- Un langage de script (également appelé script) est une série de commandes qui peuvent être exécutées sans compilation.
- Tous les langages de script sont des langages de programmation, mais tous les langages de programmation ne sont pas des langages de script.
- Les langages de script utilisent un programme appelé interpréteur pour traduire les commandes.
- Les langages de script sont souvent utilisés pour des tâches répétitives ou des petites actions, comme manipuler des éléments d'une page web.
- **Exemples** : JavaScript (pour le web), Python (utilisé pour des scripts système), et PHP (pour les serveurs web).

Définition de l'architecture Client / Serveur

- L'architecture client-serveur correspond à l'architecture d'un réseau informatique dans lequel de nombreux clients (processeurs distants) demandent et reçoivent des services d'un serveur centralisé (Serveur).
- Les **clients** sont souvent situés sur des postes de travail ou des ordinateurs personnels, tandis que les **serveurs** sont situés ailleurs sur le réseau, généralement sur des machines plus puissantes.

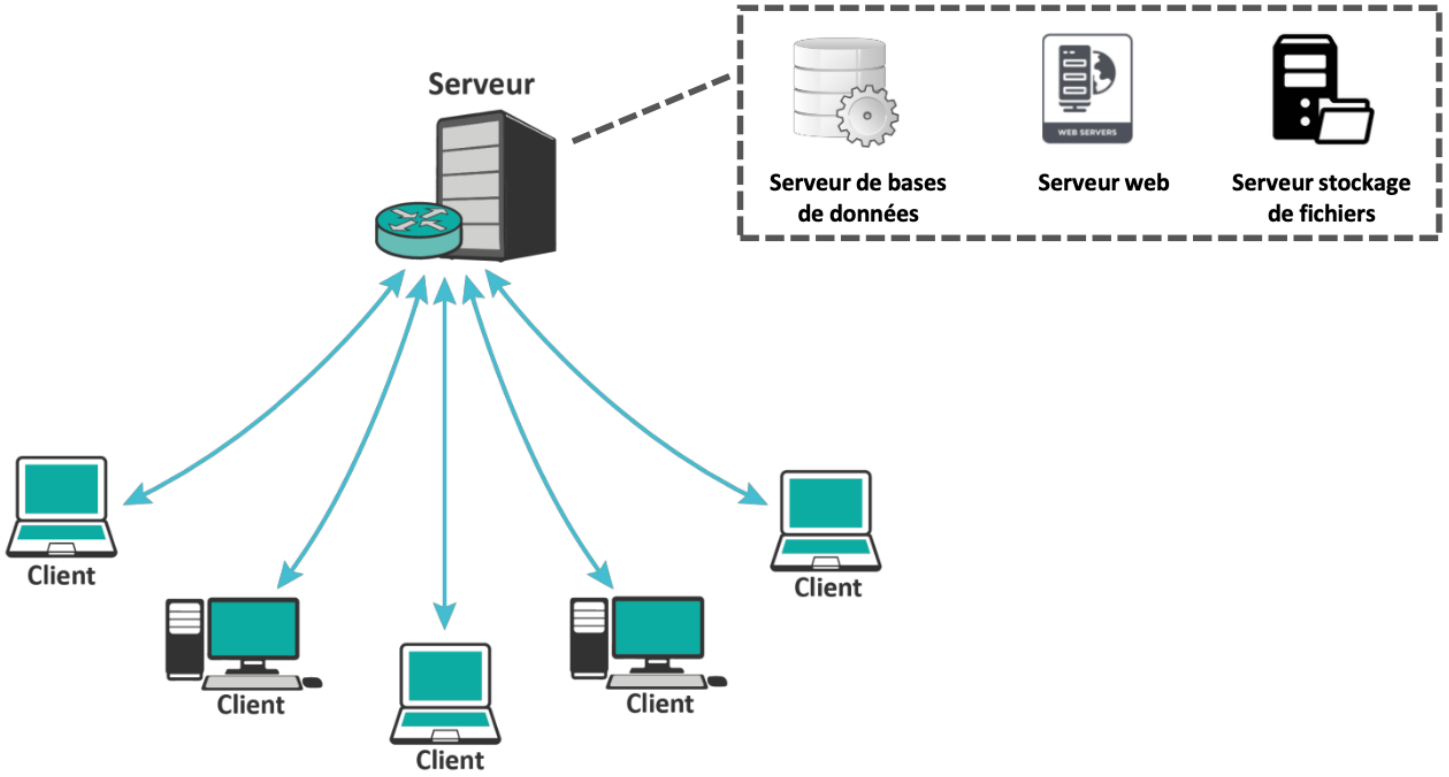
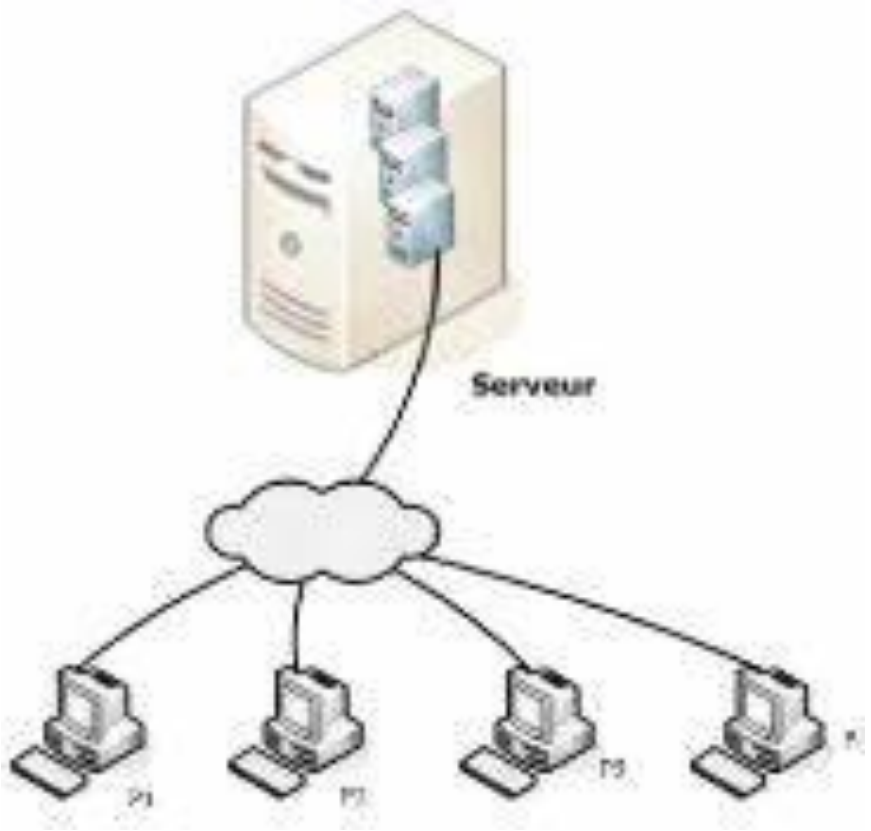


Figure 4 : Architecture Client-serveur [2]

Serveur

Un ordinateur de bureau peut jouer le rôle d'un serveur pour les autres machines :

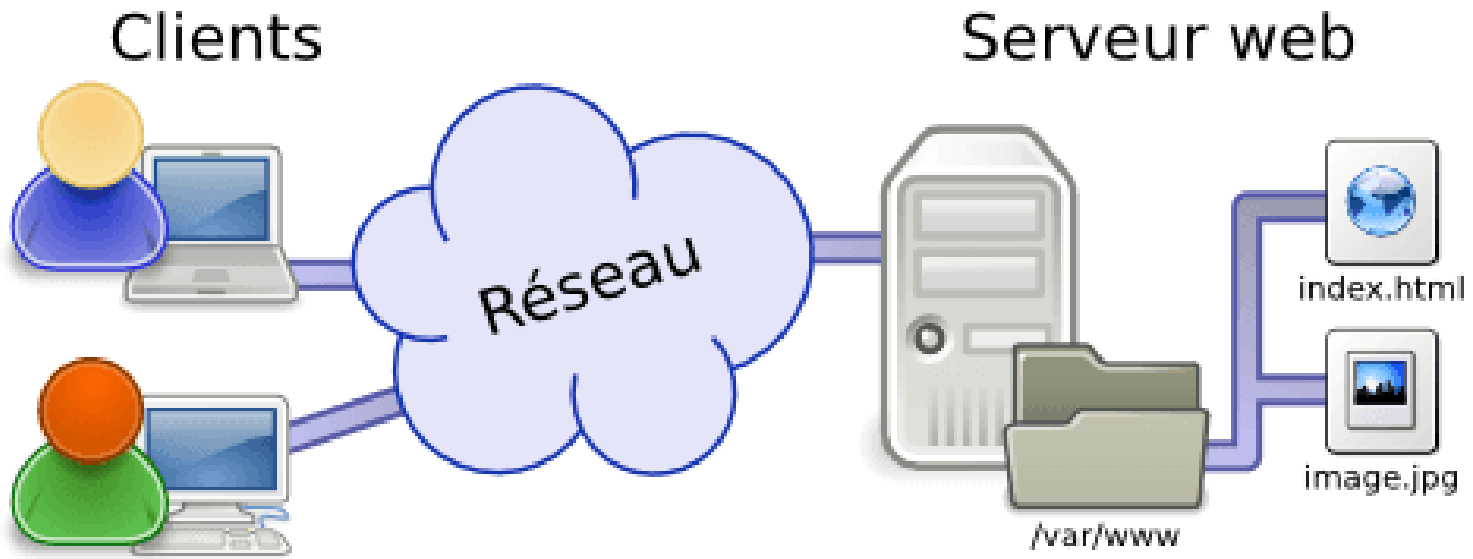
- Serveurs de fichiers
- Serveurs de web



Serveur Web

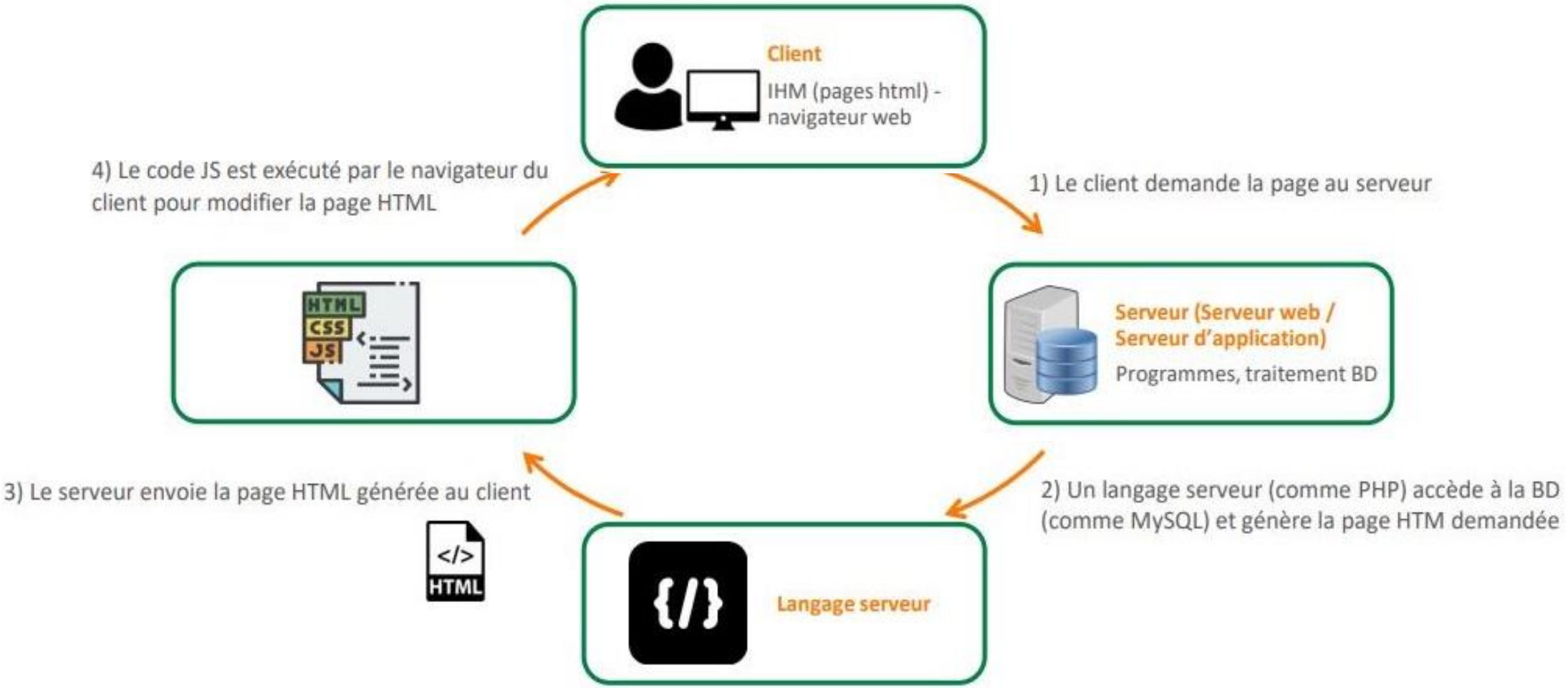
Définition : Serveur Web

- Un **serveur web** est un logiciel qui sert du contenu web aux utilisateurs en réponse aux requêtes de leurs navigateurs. Il peut délivrer des fichiers statiques (HTML, CSS, images) ou exécuter des scripts côté serveur pour générer dynamiquement du contenu (PHP, Node.js, etc.).



Fonctionnement

Le fonctionnement d'un système client/serveur peut être illustré par le schéma suivant :



Définition : HTTP (HyperText Transfer Protocol)

Le **protocole HTTP** est le protocole de communication utilisé sur le web.

- Il définit comment les messages sont formatés et transmis, et comment les serveurs et les navigateurs réagissent à différentes commandes.
- Lorsqu'un utilisateur entre une URL dans son navigateur, celui-ci envoie une requête HTTP au serveur web pour obtenir la ressource demandée.

Serveurs Web et HTTP

Les navigateurs web (clients) communiquent avec les serveurs web via le protocole HTTP (Hypertext Transfer Protocol). En tant que protocole de requête-réponse, ce protocole permet aux utilisateurs d'interagir avec des ressources Web telles que des fichiers HTML en transmettant des messages hypertextes entre les clients et les serveurs. Les clients HTTP utilisent généralement des connexions TCP (Transmission Control Protocol) pour communiquer avec les serveurs.

Une requête HTTP inclut :

- Une URL pointant sur la cible et la ressource (un fichier HTML, un document, …).
- Une méthode de requête spécifique afin d'effectuer diverses tâches (par exemple mise à jour des données, récupération d'un fichier, …).

Les différentes méthodes de requêtes et les actions associées sont présentées dans le tableau ci-dessous :

Méthode	Rôle
GET	Récupération d'une ressource spécifique (fichier html par exemple).
POST	Création d'une nouvelle ressource, enregistrement des données d'un formulaire d'inscription...
HEAD	Récupération des informations "metadata" d'une ressource spécifique sans le "body« .
PUT	Met à jour une ressource existante ou en crée une si elle n'existe pas.
DELETE	Suppression la ressource spécifiée.

Serveurs Web et HTTP

- La réponse HTTP **(HTTP Response)** est l'information fournie par le serveur suite à la demande du client. Elle sert à confirmer que l'action demandée a été exécutée avec succès. En cas d'erreur dans l'exécution de la demande du client, le serveur répond par un message d'erreur.
- Les réponses HTTP se présentent sous la forme d'un texte brut formaté au format JSON ou XML, tout comme les demandes HTTP. Le corps d'une réponse aboutie à une requête GET contiendrait la ressource demandée.

Exemples de code d'état HTTP (HTTP status codes) :

- **"200 OK" :** succès
- **"404 Not Found" :** ressource introuvable
- **"403 Forbidden" :** accès non autorisé

Choix de l'environnement de développement

Un Environnement de Développement Intégré (Integrated development environment– IDE an anglais) désigne un outil de développement dit tout en un qui permet aux programmeurs de consolider les différents aspects de l'écriture d'un programme informatique.

Les IDE assistent les programmeurs dans l'édition des logiciels en combinant les activités courantes de programmation en une seule application :

- Édition du code source
- Mise en évidence de la syntaxe (colorisation)
- Saisie automatique (Auto-complétion)
- Création d'exécutables
- Débogage

IDE pour le web :

(Liste complète et lien de téléchargement disponible sur https://www.guru99.com/web-development-ide.html)

- IntelliJ IDEA
- CodePen
- JSFiddle
- Visual Studio Code
- Bluefish
- SUBLIME TEXT 3

Front-end vs back-end

Front-End	Back-End
• Le terme "**front-end**" désigne l'interface utilisateur. • Le front-end est construit en utilisant une combinaison de technologies telles que le langage de balisage hypertexte (HTML), JavaScript et les feuilles de style en cascade (CSS).	• Le terme "**back-end**" désigne le serveur, l'application et la base de données qui travaillent en coulisses pour fournir des informations à l'utilisateur. • La programmation back-end est définie comme la logique informatique d'un site Web ou d'un logiciel, depuis le stockage et l'organisation des données jusqu'à la création des algorithmes et des séquences logiques complexes qui fonctionnent, d'une manière transparente, sur le front-end. • Les langages back-end les plus populaires pour sont Ruby, Python, Java, ASP .Net et PHP.

Les frameworks front-end

1. Angular JS
2. React.js
3. JQuery
4. Vue.js

Bibliothèques et Frameworks JavaScript

jQuery	React	Express.js
• Simplifie la manipulation du DOM, • La gestion des événements, les animations	• Permet de créer des composants réutilisables • Offre une performance optimale grâce à son DOM virtuel.	• Framework minimaliste pour Node.js, • Facilite la gestion des routes et des requêtes HTTP dans une application serveur.

Activité à faire

Exercice 1 :

1. Qu'est-ce que JavaScript ?
2. Quelle est la différence entre Java et JavaScript ?
3. Quels sont les avantages de JavaScript ?
4. Quelle entreprise a développé JavaScript ?
5. Quelles sont les compétences techniques nécessaires pour devenir développeur front-end ?
6. Dans quelle balise HTML peut-on placer le code JavaScript ?
7. Le fichier externe de JavaScript doit-il contenir la balise <script> ?
8. Quel est le bon endroit pour insérer un code JavaScript ?
9. Est-ce que Javascript peut accéder au système de fichiers ?

Solution (Exercice 1) :

1. **Qu'est-ce que JavaScript ?**
   - JavaScript est un langage de script côté client qui peut être inséré dans des pages HTML et être interprété par les navigateurs Web.

2. **Quelle est la différence entre Java et JavaScript ?**

Java	JavaScript
Java est un langage de programmation compilé	JavaScript est un langage de programmation interprété
Les applications Java peuvent s'exécuter sur n'importe quelle machine virtuelle (JVM) ou navigateur	Le code JavaScript est exécuté sur le navigateur uniquement
Les objets de Java sont basés sur les classes, même si nous ne pouvons créer aucun programme en Java sans créer une classe	Les objets JavaScript sont basés sur des prototypes
Java est un langage autonome	JavaScript est inclut dans une page Web et s'intègre à son contenu HTML
Un programme Java utilise plus de mémoire	JavaScript nécessite moins de mémoire, il est donc utilisé dans des pages Web

Solution (Exercice 1) :

3. **Quels sont les avantages de JavaScript ?**
   - JS est un langage côté client.
   - JS est relativement rapide pour l'utilisateur final.
   - JS ajoute des fonctionnalités aux pages Web.
   - JS ne nécessite aucune compilation nécessaire.
   - JS est facile à déboguer et à tester.
   - JS est indépendant de la plateforme.

4. **Quelle entreprise a développé JavaScript ?**
   - Javascript a été développé par la société NetScape.

5. **Quelles sont les compétences techniques nécessaires pour un développeur front-end ?**
   - Maitriser les langages HTML, CSS, Javascript, Jquery, NodeJS, et les frameworks tels que Angular, React et viewJS.

4. **Dans quelle balise HTML plaçons-nous le code JavaScript ?**
   - La balise <script>.

5. **Le fichier externe de JavaScript doit contenir la balise <script> ?**
   - Non.

7. **Est-ce que Javascript peut accéder au système de fichiers ?**
   - Non.

Intégration de JavaScript dans HTML

JavaScript peut être intégré n'importe où dans le document HTML. Il est aussi possible de l'utiliser plusieurs fois dans le même document HTML.

- **Méthode 1 :** Code JavaScript intégré au document html

```
<script language="JavaScript">

/* ou // code js*/

</script>
```

Intégration de JavaScript dans HTML

- **Méthode 2 :** Code JavaScript externe

```
<script language="javascript" src="monScript.js"> </script>
```

Identifiants JavaScript :

- Un identifiant en JS correspond au nom d'une variable. Ce nom doit être unique.
- Ces identifiants peuvent être des noms courts (comme x et y) ou bien des noms plus descriptifs (comme note, total, NomComplet, etc.).

Les règles à respecter lors du choix des noms de variables sont :

- Un identifiant peut contenir des lettres, des chiffres, des traits de soulignement (_) et des signes dollar ($).
- Un identifiant peut commencer par une lettre, un signe $ ou bien un signe _
- JS est sensible à la casse (y et Y sont considérés comme des variables différentes).
- Un identifiant ne peut pas être un mot réservé du langage.

Types de données JavaScript :

JavaScript est un langage faiblement typé : le type d'une variable est défini au moment de l'exécution. On peut déclarer une variable JS en utilisant les mots-clés suivants :

- var : utilisé pour déclarer une variable globale (function scope) ;
- let : utilisé pour déclarer une variable dont la portée est limitée à un bloc (block scope) ;
- const : permet de déclarer une variable qui doit avoir une valeur initiale et ne peut pas être réassignée.

Types de données JavaScript

- Déclaration explicite (en utilisant le mot clé var) :

  var nom_variable = new Type de la variable;

  var nom_variable;

- Déclaration implicite (sans utiliser var, on écrit le nom de la variable suivie du caractère d'affectation et de la valeur à affecter)

  Numéro = 1 ; Prénom = "xyz" ;

Types de données JavaScript (exemple)

Déclaration des variable booléennes :

- **var** test=new Boolean(true) ;
- test=new Boolean(1) ;
- **let** test = true.

Déclaration des chaînes de caractères :

- **var** chaine = "Bonjour";

Déclaration des nombres :

- var entier = 60; //un entier ;
- **let** pi = 3.14; //un nombre réel.

Déclaration de plusieurs variables :

- var variable3 = 2, variable4 = "mon texte d'initialisation";

Déclaration sans initialisation :

- Une variable déclarée sans valeur aura la valeur **undefined**.

Types de données JavaScript (exemple)

Const permet de créer des variables JavaScript qui ne peuvent être ni redéclarées ni réaffectées (constantes). Ces variables doivent être initialisées à la déclaration. Exemple :

```
const PI = 3.141592653589793;
PI = 3.14;      // Erreur
PI = PI + 10;   // Erreur
```

```
const PI ;
PI= 3.14159265359; // Incorrect
```

Portée des variables (variable scope)

La portée d'une variable détermine son accessibilité (visibilité). En JS, trois types de portées sont distinguées :

Portée du bloc : (Block scope)

- En utilisant le mot clé **let**, les variables déclarées à l'intérieur d'un bloc { } ne sont pas accessibles depuis l'extérieur du bloc :

```
{        let x = 2;          }
 // x n'est pas accessible ici
```

- En utilisant le mot clé **var**, les variables déclarées à l'intérieur d'un bloc { } sont accessibles depuis l'extérieur du bloc.

```
{        var x = 2;          }
 // x est accessible ici
```

Portée locale : (Function scope)

- Les variables déclarées dans une fonction JavaScript deviennent LOCALES à la fonction : Ils ne sont accessibles qu'à partir de la fonction.

```
function Test()
{
        var x = "test1";
        let y = "test2";
        const z = "test3";
}
//x, y et z ne sont pas
accessibles en dehors de la
fonction
```

Portée globale : (Global scope)

- Une variable déclarée en dehors d'une fonction, devient GLOBAL. Les variables globales sont accessibles de n'importe où dans un programme JavaScript..

Opérateurs arithmétiques JavaScript:

Les opérateurs arithmétiques sont utilisés pour effectuer des opérations arithmétiques sur des nombres :

Opérateur	Signification
+	Addition
-	Soustraction
*	Multiplication
**	Puissance
/	Division
%	Modulo (Reste de la division)
++	Incrémentation
--	Décrémentation

Opérateurs d'affectation JavaScript :

Les opérateurs d'affectation permettent d'assigner des valeurs aux variables JavaScript. Le tableau suivant regroupe ces opérateurs :

Opérateur	Exemple d'utilisation	Signification	Exemple complet	Résultat
=	x = y	x prend la valeur de y	let x = 5	x vaut 5
+=	x += y	x = x + y	let x = 10; x += 5;	x vaut 15
-=	x -= y	x = x – y	let x = 10; x -= 5;	x vaut 5
*=	x *= y	x = x * y	let x = 10; x *= 5;	x vaut 50
/=	x /= y	x = x / y	let x = 10; x /= 5;	x vaut 2
%=	x %= y	x = x % y	let x = 10; x %= 5;	x vaut 0
**=	x **= y	x = x ** y ⇔ x = x à la puissance y	let x = 3; x %= 2;	x vaut 9

Concaténation des chaînes de caractères en JavaScript

L' opérateur + , appliqué aux chaînes de caractères, permet de concaténer des chaînes.

Exemple 1 :

```
let texte1 = "OFPPT";
texte1 += " ";
let texte2 = "en force";
let texte3 = texte1 + texte2;

//Output : texte3 = "OFPPT en force"
```

Concaténation des chaînes de caractères en JavaScript

L'application de l'opérateur + pour concaténer les chaines de caractères et les nombres :

Exemple 2 :

```
let x = 1 + 1;
let y = "5" + 1;

//x=2
//y="51"
```

Opérateurs de comparaison en JavaScript

Les opérateurs de comparaison permettent de comparer des opérandes (qui peuvent être des valeurs numériques ou des chaînes de caractères) et renvoie une valeur logique : **true** (vrai) si la comparaison est vraie, **false** (faux) sinon.

Opérateur	Signification	Exemple
==	Egal à (comparaison des valeurs)	let x = 5; let y = "5"; let z=(x==y); //z=true
===	Egal à (comparaison de la valeur et du type)	let x = 5; let y = "5"; let z=(x===y); //z=false
!=	Différent de (n'est pas égal à)	let x = 5; let y = 5; let z=(x!=y); //z=false
!==	Type ou valeur différente	let x = 5; let y = "5"; let z=(x!==y); //z=true
>	Supérieur à	let x = 5; let y = 5; let z=(x>y); //z=false
<	Inférieur à	let x = 5; let y = 5; let z=(x<y); //z=false
>=	Supérieur ou égal à	let x = 5; let y = 5; let z=(x>=y); //z=true
<=	Inférieur ou égal à	let x = 5; let y = 5; let z=(x<=y); //z=true

Opérateurs logiques en JavaScript

Les opérateurs logiques, aussi appelés opérateurs booléens, sont utilisés pour combiner des valeurs booléennes (des conditions qui peuvent avoir les valeurs true ou false) et produire une nouvelle valeur booléenne.

Opérateur	Signification
&&	ET logique
||	OU logique
!	NON logique

Opérateurs de type en JavaScript

Opérateur	Signification	Exemple
typeof	Retourne le type de la variable	typeof(5) retourne "number" typeof("5") retourne "string"
instanceof	Retourne true si l'objet est une instance de la classe donnée en paramètre	console.log("JavaScript" instanceof String);

Opérateurs bit-à-bit en JavaScript

Les opérateurs bit à bit sont des opérateurs qui permettent d'effectuer des transformations sur des nombres entiers de 32 bits comme des chiffres binaires.

Opérateur	Signification	Exemple	Équivalent à	Résultat binaire	Résultat décimal
&	ET binaire : Renvoie 1 pour chaque position de bits pour laquelle les bits correspondants des deux opérandes sont 1	5 & 1	0101 & 0001	0001	1
|	OU binaire : Renvoie 1 pour chaque position de bits pour laquelle le bit correspondant d'au moins un des deux opérandes est 1	5 | 1	0101 | 0001	0101	5
~	NON binaire : inverse les bits de l'opérande	~ 5	~0101	1010	10
^	XOR binaire : Renvoie 1 pour chaque position de bit pour laquelle le bit correspondant d'un seul des deux opérandes est 1	5 ^ 1	0101 ^ 0001	0100	4
<<	Décalage de bits à gauche : a<<b décale "a" en représentation binaire de "b" bits vers la gauche, en introduisant des zéros par la droite	5 << 1	0101 << 1	1010	10
>>	Décalage de bits à droite: a>>b décale "a" en représentation binaire de "b" bits vers la droite, en rejetant les bits à droite	5 >> 1	0101 >> 1	0010	2

Exercice 1 :

Quel est le résultat de l'exécution de chacune de ces lignes dans la console ? Pourquoi ?

1. var a; typeof a;
2. var s = '1s'; s++;
3. !!"false"
4. !!undefined
5. typeof -Infinity
6. 10 % "0"
7. undefined == null
8. false === ""
9. typeof "2E+2"
10. a = 3e+3; a++;

Solution (Exercice 1) :

1. undefined
2. NaN
3. true
4. false
5. number
6. NaN
7. true
8. false
9. string
10. 3001

Exercice 2 :

Donner le résultat logique de chacune des déclarations suivantes :

1. `4 > 3 && 10 < 12`
2. `4 > 3 && 10 > 12`
3. `4 > 3 || 10 < 12`
4. `4 > 3 || 10 > 12`
5. `!(4 > 3)`
6. `!(4 < 3)`
7. `!(false)`
8. `!(4 > 3 && 10 < 12)`
9. `!(4 > 3 && 10 > 12)`
10. `!(4 === '4')`

Solution (Exercice 2) :

1. true
2. false
3. true
4. true
5. false
6. true
7. true
8. false
9. true
10. true

Exercice 3 :

1. Créer l'interface HTML suivante puis créer une fonction qui permet de calculer la somme des valeurs saisies dans les deux champs et de mettre le résultat dans un élément <label>

2. Créer l'interface suivante puis le code javascript qui permet de permuter les valeurs des deux champs de saisie

3. Créer l'interface suivante, puis une fonction qui permet de séparer le nom et le prénom et les mettre dans des éléments <label>. On considère que le nom et le prénom sont constitués d'un seul mot (Utiliser les fonctions indexOf et slice)

Solution (Exercice 3) :

1.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Exercice 3-Q1</title>
    <script src="test.js"></script>
</head>
    <body>
        <input id="n1" type="text">
        +<input id="n2" type="text">
        =<label id="resultat"></label>
        <button onclick="calculer()">Calculer</button>
    </body>
</html>
```

```
function calculer()
{
    n1 = parseFloat(document.getElementById("n1").value);
    n2 = parseFloat(document.getElementById("n2").value);
    document.getElementById("resultat").innerHTML = n1+n2;
}
```

Solution (Exercice 3) :

2.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Exercice 3-Q2</title>
    <script src="test.js"></script>
</head>
    <body>
        <input id="n1" type="text">
        <input id="n2" type="text">
        <button onclick="permuter()">Permuter</button>
    </body>
</html>
```

```
function permuter()
{
    const temp = document.getElementById("n1").value;
    document.getElementById("n1").value=document.getElementById("n2").value;
    document.getElementById("n2").value=temp;
}
```

Solution (Exercice 3) :

3.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Exercice 3-Q3</title>
    <script src="test.js"></script>
</head>
    <body>
        <input id="nom_complet" type="text" placeholder="Nom et prénom">
        <button onclick="separer()">Séparer</button><br>
        Nom:<label id="nom"></label><br>
        Prénom:<label id="prenom"></label>
    </body>
</html>
```

```
function separer()
{
    let nom_complet = document.getElementById("nom_complet").value
    let position_espace=nom_complet.indexOf(" ")
    document.getElementById("nom").innerHTML = nom_complet.slice(0,position_espace)
    document.getElementById("prenom").innerHTML = nom_complet.slice(position_espace+1)
}
```

- **Les commentaires**
  - Pour mettre en commentaires toute une ligne, on utilise le double slash:

```
// Tous les caractères derrière le // sont ignorés
```

  - Pour mettre en commentaire une partie du texte (éventuellement sur plusieurs lignes) on utilise le **/*** et le ***/**:

```
/* Toutes les lignes comprises entre ces repères sont ignorées par l'interpréteur de code */
```