

# Institut Supérieur Numérique

## Cours Langage Web

### JavaScript

# JavaScript

- ❖ JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives.
- ❖ JavaScript est le langage de programmation le plus populaire au monde.
- ❖ JavaScript est le langage de programmation du Web.
- ❖ JavaScript est facile à apprendre.

# JavaScript

- ❖ JavaScript est l'un des 3 langages que tous les développeurs web doivent apprendre :
  - ✓ HTML pour définir le contenu des pages web.
  - ✓ CSS pour spécifier la mise en page des pages Web.
  - ✓ JavaScript pour programmer le comportement des pages Web

# JavaScript peut modifier le contenu HTML

- ❖ L'une des nombreuses méthodes HTML JavaScript est `getElementById()`
- ❖ L'exemple ci-dessous trouve un élément HTML (avec `id="demo"`), et change le contenu de l'élément (`innerHTML`) en « Hello JavaScript » :

```
<html><body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button"
  onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!'">
  Click Me!
</button>

</body></html>
```

- ❖ JavaScript accepte les guillemets doubles et simples

# JavaScript peut masquer des éléments HTML

- ❖ Le masquage des éléments HTML peut être effectué en modifiant le style `display`

```
<html><body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can hide HTML elements.</p>

<button type="button"
  onclick="document.getElementById('demo').style.display='none'">
  Click Me!
</button>

</body></html>
```

# JavaScript peut afficher des éléments HTML

- ❖ L'affichage des éléments HTML masqués peut également être effectué en modifiant le style `display`

```
<html><body>

<h2>What Can JavaScript Do?</h2>

<p>JavaScript can show hidden HTML elements.</p>

<p id="demo" style="display:none">Hello JavaScript!</p>

<button type="button"
  onclick="document.getElementById('demo').style.display='block'">Click Me!
</button>
</body></html>
```

# Balise <script>

- ❖ En HTML, le code JavaScript est inséré entre et les balises <script></script>

```
<html><body>

<h2>JavaScript in Body</h2>

<p id="demo"></p>

<script>
    document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
</body></html>
```

# Fonctions et événements JavaScript

- ❖ Une fonction JavaScript est un bloc de code JavaScript, qui peut être exécuté lorsqu'il est « appelé »
- ❖ Par exemple, une fonction peut être appelée lorsqu'un événement se produit, par exemple lorsque l'utilisateur clique sur un bouton.

```
<html><body>
<script>
  function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph changed.";
  }
</script>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>
</body></html>
```

# JavaScript externe

- ❖ Les scripts peuvent être placés dans le <body> ou le <head>.
- ❖ Les scripts peuvent également être placés dans des fichiers externes
- ❖ Fichier externe : myScript.js

```
function myFunction() {  
    document.getElementById("demo").innerHTML = "Paragraph changed.";}  

```

- ❖ Les scripts externes sont pratiques lorsque le même code est utilisé dans de nombreuses pages Web différentes.
- ❖ Pour utiliser un script externe, placez le nom du fichier de script dans l'attribut (source) **src** d'une balise <script>
- ❖ Les scripts externes ne peuvent pas contenir de balises <script>
- ❖ Exemple :

```
<script src="myScript.js"></script>
```

# Avantages de JavaScript externe

- ❖ Placer des scripts dans des fichiers externes présente certains avantages :
  - Il sépare HTML et code
  - Il rend HTML et JavaScript plus faciles à lire et à maintenir
  - Les fichiers JavaScript mis en cache peuvent accélérer le chargement des pages
- ❖ Pour ajouter plusieurs fichiers de script à une page, utilisez plusieurs balises de script :

```
<script src="myScript1.js"></script>  
<script src="myScript2.js"></script>
```
- ❖ Références externes : Un script externe peut être référencé de 3 manières différentes :
  - Avec une URL complète (une adresse Web complète)
  - Avec un chemin de fichier (comme /js/)
  - Sans aucun chemin

# Sortie JavaScript

- ❖ JavaScript peut afficher les données de différentes manières :
  - Écriture dans un élément HTML, à l'aide de `innerHTML`
  - Écriture dans la sortie HTML à l'aide de `document.write()`
  - Écriture dans une zone d'alerte, à l'aide de `alert()`
  - Écriture dans la console du navigateur, à l'aide de `console.log()`

# Utilisation de innerHTML

- ❖ Pour accéder à un élément HTML, JavaScript peut utiliser la méthode.

`document.getElementById(id).innerHTML`

```
<html>
<body>
<h1>My First Web Page</h1>
  <p>My First Paragraph</p>

  <p id="demo"> </p>
  <script>document.getElementById("demo").innerHTML=5+6;</script>
</body>
</html>
```

# Utilisation de innerHTML

- ❖ il est pratique d'utiliser `document.write()`

```
<html>
<body>
  <h1>My First Web Page</h1>
  <p>My first paragraph.</p>

  <script> document.write(5 + 6); </script>
</body>
</html>
```

# Utilisation de alert()

- ❖ Vous pouvez utiliser une zone d'alerte pour afficher les données :

```
<html>
<body>
  <h1>My First Web Page</h1>
  <p>My first paragraph.</p>

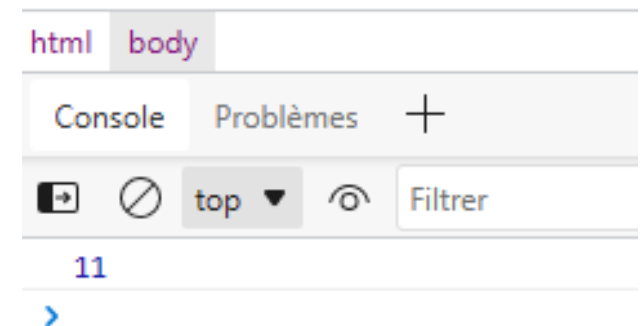
  <script>alert(5+6);</script>
</body>
</html>
```

# Utilisation de la console.log()

- ❖ À des fins de débogage, vous pouvez appeler la méthode dans le navigateur pour afficher les données `console.log()`

```
<html>
<body>
  <h1>My First Web Page</h1>
  <p>My first paragraph.</p>

  <script>console.log(5 + 6);</script>
</body>
</html>
```



# Impression JavaScript

- ❖ JavaScript n'a pas d'objet d'impression ou des méthodes d'impression.
- ❖ Vous ne pouvez pas accéder aux périphériques de sortie à partir de JavaScript.
- ❖ La seule exception est que vous pouvez appeler la méthode `window.print()` dans le navigateur pour imprimer le contenu de la fenêtre actuelle.

```
<html>
<body>
  <h1>My First Web Page</h1>
  <p>My first paragraph.</p>

  <button onclick="window.print()">Print this page</button>
</body>
</html>
```

# Mots-clés JavaScript

- ❖ Les instructions JavaScript commencent souvent par un mot-clé pour identifier l'action JavaScript à effectuer.
- ❖ Voici une liste de certains des mots-clés

| Mot-clé  | Description                                                   |
|----------|---------------------------------------------------------------|
| var      | Déclare une variable                                          |
| let      | Déclare une variable de bloc                                  |
| const    | Déclare une constante de bloc                                 |
| if       | Marque un bloc d'instructions à exécuter sur une condition    |
| switch   | Marque un bloc d'instructions à exécuter dans différents cas  |
| for      | Marque un bloc d'instructions à exécuter dans une boucle      |
| function | Déclare une fonction                                          |
| return   | Quitte une fonction                                           |
| try      | Implémente la gestion des erreurs dans un bloc d'instructions |

# Syntaxe JavaScript

- ❖ Dans un langage de programmation, les variables sont utilisées pour stocker des valeurs de données.
- ❖ JavaScript utilise les mots-clés et pour déclarer des variables : `var`, `let` et `const` pour les constantes.
- ❖ Le code après des barres obliques doubles `//` ou entre `/*` et `*/` est traité comme un commentaire.
- ❖ JavaScript est sensible à la casse : Tous les identificateurs JavaScript sont sensibles à la casse.  
Les variables `lastName` et `lastname` sont deux variables différentes.
- ❖ JavaScript n'interprète pas `LET` ou `Let` comme le mot-clé `let`.
- ❖ Les chaînes sont écrites entre guillemets doubles ou simples. Les chiffres sont écrits sans guillemets.

# Variables JavaScript

## ❖ 4 façons de déclarer une variable JavaScript :

- Utilisant var
- Utilisant let
- Utilisant const
- Ne rien utiliser

## ❖ Quand utiliser JavaScript var ?

- ✓ Déclarez toujours les variables JavaScript avec **var**, **let** ou **const**
- ✓ Le mot-clé **var** est utilisé dans tout le code JavaScript de 1995 à 2015.
- ✓ Les mots-clés **let** et **const** ont été ajoutés à JavaScript en 2015.
- ✓ Si vous souhaitez que votre code s'exécute dans un navigateur plus ancien, vous devez utiliser **var**

# Opérateurs arithmétiques JavaScript

- ❖ Les opérateurs arithmétiques sont utilisés pour effectuer de l'arithmétique sur des nombres :

| Opérateur | Description                               |
|-----------|-------------------------------------------|
| +         | Addition                                  |
| -         | Subtraction                               |
| *         | Multiplication                            |
| **        | Exponentiation ( <a href="#">ES2016</a> ) |
| /         | Division                                  |
| %         | Modulus (Division Remainder)              |
| ++        | Incrémente                                |
| --        | Décrémente                                |

- ❖ L'opérateur **+** peut également être utilisé pour ajouter (concaténer) des chaînes.

# Opérateurs arithmétiques JavaScript

## ❖ Opérateurs de comparaison JavaScript :

| Opérateur | Description                       |
|-----------|-----------------------------------|
| ==        | equal to                          |
| ===       | equal value and equal type        |
| !=        | not equal                         |
| !==       | not equal value or not equal type |
| >         | greater than                      |
| <         | less than                         |
| >=        | greater than or equal to          |
| <=        | less than or equal to             |
| ?         | ternary operator                  |

# Opérateurs arithmétiques JavaScript

## ❖ Opérateurs logiques JavaScript :

| Opérateur | Description |
|-----------|-------------|
| &&        | and         |
|           | or          |
| !         | not         |

## ❖ Opérateurs de type JavaScript

| Opérateur  | Description                                                |
|------------|------------------------------------------------------------|
| typeof     | Returns the type of a variable                             |
| instanceof | Returns true if an object is an instance of an object type |

# Types de données JavaScript

- ❖ Les variables JavaScript peuvent contenir différents types de données : nombres, chaînes, objets, etc :

```
let length = 16;           // Number
let lastName = "Johnson"; // String
let x = {firstName:"John", lastName:"Doe"}; // Object
```

- ❖ Tableaux JavaScript : Les tableaux JavaScript sont écrits entre crochets.

```
const cars = ["Saab", "Volvo", "BMW"];
```

```
const cars = new Array("Saab", "Volvo", "BMW");
cars.push("Toyota");
```

# Objets JavaScript

- ❖ Vous définissez (et créez) un objet JavaScript :

```
const person = {  
  firstName : "John",  
  lastName : "Doe",  
  age : 50,  
  eyeColor : "blue"};
```

- ❖ Les paires name values dans les objets JavaScript sont appelées propriétés
- ❖ Les objets peuvent également avoir des méthodes.
- ❖ Vous pouvez accéder aux propriétés de l'objet de deux manières :

```
objectName.propertyName  
Ou  
objectName["propertyName"]
```

# JavaScript if, else, and else if

- ❖ Un booléen JavaScript représente l'une des deux valeurs suivantes : true ou false.
- ❖ Les instructions conditionnelles sont utilisées pour effectuer différentes actions en fonction de différentes conditions.

```
time = 5;
if (time <10) {
    greeting = "Good morning";
} else if(time <20){
    greeting = "Good day";
} else {
    greeting = "Good evening";
}
document.write(greeting);
```

# JavaScript Switch

- ❖ L'instruction **switch** est utilisée pour effectuer différentes actions en fonction de différentes conditions.

```
switch(expression) {  
  case x:  
    // code block  
    break;  
  case y:  
    // code block  
    break;  
  default:  
    // code block  
}
```

```
switch (new Date().getDay()) {  
  case 0:  
    day = "Sunday";  
    break;  
  case 1:  
    day = "Monday";  
    break;  
  case 2:  
    day = "Tuesday";  
    break;  
  case 3:  
    day = "Wednesday";  
    break;  
  case 4:  
    day = "Thursday";  
    break;  
  case 5:  
    day = "Friday";  
    break;  
  case 6:  
    day = "Saturday";  
}  
document.write(day);
```

# Boucle for

- ❖ La syntaxe de la boucle **for** est la suivante :

```
for (statement 1; statement 2; statement 3) {  
    // code block to be executed  
}
```

```
<p id="demo"></p>  
<script>  
    let text = "";  
    for( let i = 0; i < 5; i++) {  
        text += "The number is " + i + "<br>";  
    }  
    document.getElementById("demo").innerHTML=text;  
</script>
```

# Boucle for in

- ❖ L'instruction JavaScript **for in** parcourt en boucle les propriétés d'un objet
- ❖ Chaque itération renvoie une clé (x)
- ❖ La clé est utilisée pour accéder à la valeur
- ❖ Syntaxe :

```
for (key in object) {  
    // code block to be executed  
}
```

- ❖ Exemple :

```
<p id="demo"></p>  
<script>  
    const person = {fname:"John", lname:"Doe", age:25};  
    let text = "";  
    for (let x in person) {  
        text += person[x];  
    }  
    document.getElementById("demo").innerHTML=text;  
</script>
```

John Doe 25

# Boucle for in

- ❖ L'instruction JavaScript **for in** peut également effectuer une boucle sur les propriétés d'un tableau
- ❖ Syntaxe :

```
for (variable in array) {  
    code }  
}
```

- ❖ Exemple :

```
const numbers = [45, 4, 9, 16, 25];  
let txt = "";  
for (let x in numbers) {  
    txt += numbers[x] + "<br>";  
}  
document.getElementById("demo").innerHTML = txt;
```

45  
4  
9  
16  
25

# Boucle for of

- ❖ L'instruction JavaScript **for of** parcourt en boucle les valeurs d'un objet itérable.
- ❖ Pour chaque itération, la valeur de la propriété suivante est affectée à la variable.
- ❖ Syntaxe :

```
for (variable of iterable) {  
    // code block to be executed  
}
```

# Boucle for of

## ❖ Boucle for of sur un tableau

```
<script>
  const cars = ["BMW", "Volvo", "Mini"];
  let text = "";
  for (let x of cars) {
    text += x + "<br>";
  }
  document.getElementById("demo").innerHTML = text;
</script>
```



BMW  
Volvo  
Mini

# Boucle for of

## ❖ Boucle for of sur une chaîne

```
<p id="demo"></p>
<script>
  let language = "JavaScript";
  let text = "";
  for (let x of language) {
    text += x + "<br>";
  }
  document.getElementById("demo").innerHTML = text;
</script>
```

J  
a  
v  
a  
S  
c  
r  
i  
p  
t

# Boucle while

- ❖ La boucle **while** traverse un bloc de code tant qu'une condition spécifiée est vraie.
- ❖ Syntaxe :

```
while (condition) {  
    // code block to be executed  
}
```

```
<p id="demo"></p>  
<script>  
    let text = "";  
    let i = 0;  
    while (i < 10) {  
        text += "<br>The number is " + i;  
        i++;  
    }  
    document.getElementById("demo").innerHTML = text;  
</script>
```

The number is 0  
The number is 1  
The number is 2  
The number is 3  
The number is 4  
The number is 5  
The number is 6  
The number is 7  
The number is 8  
The number is 9

# La boucle do while

## ❖ Syntaxe :

```
do {  
    // code block to be executed  
}  
while (condition);
```

```
<p id="demo"></p>  
<script>  
    let text = "";  
    let i = 0;  
    do {  
        text += "<br>The number is " + i;  
        i++;  
    }while (i < 10);  
  
    document.getElementById("demo").innerHTML = text;  
</script>
```

The number is 0  
The number is 1  
The number is 2  
The number is 3  
The number is 4  
The number is 5  
The number is 6  
The number is 7  
The number is 8  
The number is 9

# JavaScript Break and Continue

```
<p id="demo"></p>
<script>
  let text = "";
  for(let i = 0; i < 10; i++) {
    if (i === 3) {
      break;
    }
    text += "The number is " + i + "<br>";
  }
  document.getElementById("demo").innerHTML = text;
</script>
```

The number is 0  
The number is 1  
The number is 2

# JavaScript Break and Continue

```
<p id="demo"></p>
<script>
  let text = "";
  for(let i = 0; i < 10; i++) {
    if (i === 3) {
      continue;
    }
    text += "The number is " + i + "<br>";
  }
  document.getElementById("demo").innerHTML = text;
</script>
```

The number is 0  
The number is 1  
The number is 2  
The number is 4  
The number is 5  
The number is 6  
The number is 7  
The number is 8  
The number is 9

# JavaScript typeof

❖ Vous pouvez utiliser **typeof** pour rechercher le type de données d'une variable JavaScript.

❖ Exemple :

```
var bool = true;  
  
console.log(typeof(bool));//boolean
```

# Fonctions JavaScript

- ❖ Une fonction JavaScript est un bloc de code conçu pour effectuer une tâche particulière.
- ❖ Une fonction JavaScript est définie avec le mot-clé, suivi d'un nom, suivi de parenthèses `()..function`

```
function name(parameter1, parameter2, parameter3) {  
    // code to be executed  
}
```

- ❖ Exemple :

```
function myFunction(p1, p2) {  
    return p1 * p2; // The function returns the product of p1 and p2  
}
```

# Fonctions JavaScript

❖ valeurs par défaut :

```
var result;  
  
function addNumbers(num1 = 2, num2 = 3){  
    result = num1 + num2;  
    console.log(result);  
}  
  
addNumbers();//5  
addNumbers(10, 5);//15
```

# Fonctions JavaScript

❖ IIFE Immediately Invoked Function Execution:

```
(function (num1, num2) {  
  var result = num1 + num2;  
  console.log(result);  
})(10, 12);
```

# Chaînes JavaScript

- ❖ Les chaînes JavaScript servent à stocker et à manipuler du texte.
- ❖ Une chaîne JavaScript est composée de zéro ou plusieurs caractères écrits entre guillemets.
- ❖ Longueur de la chaîne : Pour trouver la longueur d'une chaîne, utilisez la propriété : **length**

```
let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
let length = text.length;
```

# Événements JavaScript

- ❖ Lorsque JavaScript est utilisé dans des pages HTML, JavaScript peut réagir à ces événements (onclick, onchange ...).

```
<element event="some JavaScript">
```

```
<button onclick="alert('Hello')">Cliquer ici.</button>
```

# JavaScript JSON

- ❖ JSON est un format de stockage et de transport de données.
- ❖ JSON signifie JavaScript Object Notation
- ❖ JSON est un format d'échange de données léger
- ❖ JSON est indépendant du langage \*

```
{  
  "employees": [  
    {"firstName": "John", "lastName": "Doe"},  
    {"firstName": "Anna", "lastName": "Smith"},  
    {"firstName": "Peter", "lastName": "Jones"}  
  ]  
}
```

# The HTML DOM (Document Object Model)

- ❖ Avec le DOM HTML, JavaScript peut accéder et modifier tous les éléments d'un document HTML.
- ❖ La méthode `getElementById` : la façon la plus courante d'accéder à un élément HTML consiste à utiliser le **id** de l'élément.
- ❖ La propriété **innerHTML** : le moyen le plus simple d'obtenir le contenu d'un élément et consiste à utiliser la propriété **innerHTML**.
- ❖ `element.innerHTML` : Change the inner HTML of an element
- ❖ `element.style.property` : Change the style of an HTML element

# The HTML DOM (Document Object Model)

## ❖ Modification du style HTML :

```
<h1 id="id1">My Heading 1</h1>

<button type="button"
  onclick="document.getElementById('id1').style.color = 'red'">
  Click Me!
</button>
```

My Heading 1

Click Me!

My Heading 1

Click Me!

# The HTML DOM (Document Object Model)

❖ Ajout et suppression d'éléments :

| Method                                                     | Description                       |
|------------------------------------------------------------|-----------------------------------|
| <code>document.createElement(<i>element</i>)</code>        | Create an HTML element            |
| <code>document.removeChild(<i>element</i>)</code>          | Remove an HTML element            |
| <code>document.appendChild(<i>element</i>)</code>          | Add an HTML element               |
| <code>document.replaceChild(<i>new</i>, <i>old</i>)</code> | Replace an HTML element           |
| <code>document.write(<i>text</i>)</code>                   | Write into the HTML output stream |

# Affecter des événements à l'aide du DOM HTML

- ❖ Le DOM HTML vous permet d'affecter des événements à des éléments HTML en JavaScript :

```
<h2>JavaScript HTML Events</h2>
<p>Click "Try it" to execute the displayDate() function.</p>

<button id="myBtn">Try it</button>

<p id="demo"></p>

<script>
document.getElementById("myBtn").onclick = displayDate;

function displayDate() {
document.getElementById("demo").innerHTML = Date();
}
</script>
```

# JavaScript HTML DOM EventListener

- ❖ Ajoutez un listener d'événements qui se déclenche lorsqu'un utilisateur clique sur un bouton :

```
<h2>JavaScript addEventListener()</h2>
<p>This example uses the addEventListener() method to attach a click event to a button.</p>

<button id="myBtn">Try it</button>
<p id="demo"></p>
<script>
    document.getElementById("myBtn").addEventListener("click", displayDate);

function displayDate() {
    document.getElementById("demo").innerHTML = Date();
}
</script>
```