

DÉVELOPPEMENT WEB

JAVASCRIPT

BTS DAKHLA

DAHAR RACHID

JAVASCRIPT

Est-il vraiment nécessaire de présenter JavaScript ?

- C'est l'un des langages de programmation les plus populaires de la planète ! Vous avez déjà visité un site Web qui vous a fait penser... "Hé, ce site est vraiment cool et interactif" ? Eh bien, c'est probablement grâce à JavaScript.

CRÉER VOTRE PREMIER CODE JAVASCRIPT

- Commençons par les bases et ajoutons un peu de JavaScript à une page Web.
- Sur le Web, le code JavaScript vit à l'intérieur du document HTML et doit être entouré de `<script>` et `</script>`

```
<script>
  ...
</script>
```

CRÉER VOTRE PREMIER CODE JAVASCRIPT

- On peut insérer du code JavaScript en faisant appel à un module externe se trouvant à n'importe quelle adresse (URL).
- Les deux balises de Javascript doivent être placés entre les Tags `<body>` et `</body>` dans le cas d'une exécution directe ou entre les Tags `<head>` et `</head>` de la page HTML pour une exécution différée.

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

CRÉER VOTRE PREMIER CODE JAVASCRIPT

- Utilisons JavaScript pour imprimer "Hello World" dans le navigateur. Voici à quoi cela ressemblerait.
- Il est temps d'introduire la fonction `document.write()`. C'est ce que nous devons utiliser pour écrire du texte dans notre document HTML.

```
<script>
  ...
  document.write("Hello World");
</script>
```

CRÉER VOTRE PREMIER CODE JAVASCRIPT

→ Vous pouvez également utiliser le langage de balisage HTML standard pour personnaliser l'apparence du texte dans la sortie

```
<script>
  document.write("<h1>Hello World!</h1>");
</script>
```

Attention !

`document.write()` ne doit être utilisé que pour les tests. Nous allons bientôt aborder d'autres mécanismes de sortie.

ÉCRIRE DANS LA CONSOLE DU NAVIGATEUR

- Pour cela, nous aurons besoin de la fidèle fonction `console.log()`.
- La console fait partie du navigateur Web et vous permet d'enregistrer des messages, d'exécuter du code JavaScript et de voir les erreurs et les avertissements.

```
<script>
  console.log('Hello!');
</script>
```

Attention !
Les développeurs utilisent principalement la console pour tester leur code JavaScript.

APPLICATION

Écrivez un programme pour imprimer dans la console "JS is fun".

LES VARIABLES

- Les variables sont des conteneurs permettant de stocker des valeurs de données. La valeur d'une variable peut changer tout au long du programme.
- Déclarer une variable est aussi simple que d'utiliser le mot clé **var**.

```
<script>  
    var x = 10;  
    document.write(x);  
    x = "Hello World!";  
    document.write(x);  
</script>
```

Attention!
JavaScript est sensible à la casse. Ainsi, des variables comme **lastName** et **lastname** ne sont pas identiques.

CONTRAINTE CONCERNANT LES NOMS DES VARIABLES

- Le premier caractère d'un nom de variable doit être une lettre, un trait de soulignement (_) ou un signe de dollar (\$)
- Le premier caractère d'un nom de variable ne peut pas être un chiffre.
- Les noms de variables ne peuvent pas inclure un opérateur mathématique ou logique dans leur nom. Par exemple, 2*quelque chose ou ceci+cela ;
- Les noms de variables ne peuvent pas contenir d'espaces.
- Vous n'êtes pas autorisé à utiliser des symboles spéciaux, comme my#num, num%, etc.
- Les mots réservés JavaScript ne peuvent être utilisés comme noms de variable

TYPE DE DONNÉES

- Le terme " type de données " fait référence aux types de valeurs avec lesquelles un programme peut travailler. Les variables JavaScript n'ont pas de limite, car elles peuvent contenir un grand nombre de types de données différents (nombres, chaînes de caractères, tableaux, etc.).
- Javascript vous permet de travailler avec trois types de données **primitifs**:
 - **Numéros:** par exemple 15 , 5 , 3.14
 - **Chaine de caractères** par exemple "My name is John"
 - **Booléean**, par exemple true ou false

NUMBERS

```
<script type="text/javascript">  
    console.log(1000000);  
    console.log(Number("1000000"));  
    console.log(1_000_000); // syntactic sugar  
    console.log(1e6); // e  
    console.log(10**6) // **  
    console.log(1000000.0);  
    console.log(Number.MAX_SAFE_INTEGER); // 9007199254740991  
    console.log(Number.MAX_VALUE); // 1.7976931348623157e+308  
    console.log(Number.MAX_VALUE + 26000); // 1.7976931348623157e+308  
</script>
```

NUMBERS METHODES

```
<script type="text/javascript">  
    console.log((100).toString()); // convert number to string  
    console.log(100..toString());  
  
    console.log(15.6666.toFixed(2)); // return string 15.67  
  
    console.log(Number("150.50 DSI")); //NAN  
    console.log(+ "150.50 DSI"); //NAN  
    console.log(parseInt("150 DSI")); // 150  
    console.log(parseInt("DSI 150.50 DSI")); // NAN  
    console.log(parseInt("150.50 DSI")); // 150  
    console.log(parseFloat("150.50 DSI")); // 150.5  
  
    console.log(Number.isInteger("100")); // false  
    console.log(Number.isInteger(100.500)); //false  
    console.log(Number.isInteger(100)); //true  
</script>
```

MATH OBJECT

```
<script type="text/javascript">
    console.log(Math.round(99.2)); // 99
    console.log(Math.round(99.5)); // 100

    console.log(Math.ceil(99.2)); // 100
    console.log(Math.floor(99.9)); // 99

    console.log(Math.min(10,20,100,-100,90)); // -100
    console.log(Math.max(10,20,100,-100,90)); // 100

    console.log(Math.pow(2,4)); // 16

    console.log(Math.random()); // random number between 0 and 1

    console.log(Math.trunc(99.5)); // 99
</script>
```

NUMBERS CHALENGE

```
<script type="text/javascript">

let a = 100;
let b = 2_00.5;
let c = 1e2;
let d = 2.4;

// Find Smallest Number In All Variables And Return Integer
console.log();

// Use Variables a + d One Time To Get The Needed Output
console.log(); // 10000

// Get Integer "2" From d Variable With 4 Methods
console.log();
console.log();
console.log();
console.log();

// Use Variables b + d To Get This Value
console.log(); // 66.67 => String
console.log(); // 67 => Number

</script>
```

STRING METHODES

```
<script type="text/javascript">
    var name = " Rachid ";
    //access with index
    console.log(name[3]); // R
    console.log(name.charAt(3)); // R

    //length
    console.log(name.length); // 10

    //trim
    console.log(name.trim()); //Rachid
    console.log(name.trim().charAt(3)); //h

    console.log(name.toLowerCase()); // rachid
    console.log(name.toUpperCase()); // RACHID
</script>
```

STRING METHODES

```
<script type="text/javascript">

// indexOf(value [Mand], Start [Opt] 0)
var text = "Bts Dsi Dakhla";
console.log(text.indexOf("Dsi")); // 4
console.log(text.indexOf("Dsi", 6)); // -1
console.log(text.indexOf("s")); // 2

// lastIndexOf(value [Mand], Start [Opt] Length)
console.log(text.lastIndexOf("s")); // 5

// slice(Start [Mand], End [Opt] Not include End)
console.log(text.slice(0)); // Bts Dsi Dakhla
console.log(text.slice(4, 7)); // Dsi
console.log(text.slice(-6)); // Dakhla
console.log(text.slice(-6, -4)); // Da

// repeat(Times) [ES6]
console.log(text.repeat(2)) // Bts Dsi Dakhla Bts Dsi Dakhla

// split(Separator [Opt], Limit [Opt])
console.log(text.split()); // [ 'Bts Dsi Dakhla' ]
console.log(text.split("")); // [ 'B', 't', 's', ' ', 'D', 's', 'i', ' ', 'D', 'a', 'k', 'h', 'l', 'a' ]
console.log(text.split(" ")); // [ 'Bts', 'Dsi', 'Dakhla' ]
console.log(text.split(" ", 2)); // [ 'Bts', 'Dsi' ]

</script>
```

```
<script type="text/javascript">
let a = "Elzero Web School";
console.log(a.length); // 17
/*
substring(Start [Mand], End [Opt] Not Including End)
--- Start > End Will Swap
--- Start < 0 It Start From 0
--- Use Length To Get Last Character
*/
console.log(a.substring(2, 6)); // zero
console.log(a.substring(6, 2)); // zero
console.log(a.substring(-10, 6)); // Elzero
console.log(a.substring(a.length - 5, a.length - 3)); // ch
/*
- substr(Start [Mand], Characters To Extract)
--- Start >= Length = ""
--- Negative Start From End
*/
console.log(a.substr(0, 6)); // Elzero
console.log(a.substr(17)); //
console.log(a.substr(-3)); // ool
console.log(a.substr(-5, 2)); // ch
// includes(Value [Mand], Start [Opt] Default 0) [ES6]
console.log(a.includes("Web")); // true
console.log(a.includes("Web", 8)); // false
// startsWith(Value [Mand], Start [Opt] Default 0) [ES6]
console.log(a.startsWith("E")); // true
console.log(a.startsWith("E", 2)); // false
console.log(a.startsWith("zero", 2)); // true
// endsWith(Value [Mand], Length [Opt] Default Full Length) [ES6]
console.log(a.endsWith("l")); // true
console.log(a.endsWith("ro", 6)); // true
</script>
```

STRING CHALLENGE

```
<script type="text/javascript">

/*
  String Challenge
  All Solutions Must Be In One Chain
  You Can Use Concatenate
*/

let a = "Elzero Web School";

// Include This Method In Your Solution [slice, charAt]
console.log(); // Zero

// 8 H
console.log(); // HHHHHHHH

// Return Array
console.log(); // ["Elzero"]

// Use Only "substr" Method + Template Literals In Your Solution
console.log(); // Elzero School

// Solution Must Be Dynamic Because String May Changes
console.log(); // eLZERO WEB SCHOOL
</script>
```

TYPE DE DONNÉES

```
<script>

var num = 42;
document.write(num + "<br>");

var price = 55.55
document.write(price + "<br>");

var name = 'John';
document.write(name + "<br>");

var text = "My name is John";
document.write(text + "<br>");
text = "My name is 'John' ";
document.write(text + "<br>");

text = "My name is \"John\" ";
document.write(text + "<br>");

</script>
```

le caractère d'échappement
backslash (\).

Il vient à la rescousse lorsque vous devez mettre des guillemets dans des chaînes de caractères

LES TABLEAUX

L'objet Array vous permet de stocker plusieurs valeurs dans une même variable.

```
<script type="text/javascript">
    var etudiants = ["Rachid", "Omar", "Hakim"];
</script>
```

ACCÉDER AUX ÉLÉMENTS D'UNE LISTE

Vous faites référence à un élément de liste en vous référant au **numéro d'index** débuter sur zéro

```
<script type="text/javascript">
  var etudiants = ["Rachid", "Omar", "Hakim"];
  console.log(etudiants[0]); // Renvoie Rachid
  etudiants[0] = "Mohamed";
  console.log(etudiants[0]); // Renvoie Mohamed
</script>
```

```
<script type="text/javascript">
  /*
  Arrays
  - Create Arrays [Two Methods] new Array() + []
  - Access Arrays Elements
  - Nested Arrays
  - Change Arrays Elements
  - Check For Array Array.isArray(arr);
*/
let myFriends = ["Ahmed", "Mohamed", "Sayed", ["Marwan", "Ali"]];

console.log(`Hello ${myFriends[0]}`); // Hello Ahmed
console.log(`Hello ${myFriends[2]}`); // Hello Sayed
console.log(`${myFriends[1][2]}`); // h
console.log(`Hello ${myFriends[3][1]}`); // Hello Ali
console.log(`${myFriends[3][1][2]}`); // i

console.log(myFriends); // [ 'Ahmed', 'Mohamed', 'Sayed', [ 'Marwan', 'Ali' ] ]
myFriends[1] = "Gamal";
console.log(myFriends); // [ 'Ahmed', 'Gamal', 'Sayed', [ 'Marwan', 'Ali' ] ]
myFriends[3] = ["Sameh", "Ameer"];
console.log(myFriends); // [ 'Ahmed', 'Gamal', 'Sayed', [ 'Sameh', 'Ameer' ] ]

console.log(Array.isArray(myFriends)); // true
</script>
```

ARRAY METHODES

```
// Index Start From 0 [ 0, 1, 2, 3 ... ]  
  
let myFriends = ["Ahmed", "Mohamed", "Sayed", "Alaa"];  
  
myFriends.length = 2;  
  
console.log(myFriends); // [ 'Ahmed', 'Mohamed' ]  
  
myFriends[myFriends.length] = "Rachid";  
  
console.log(myFriends); // [ 'Ahmed', 'Mohamed', 'Rachid' ]  
  
myFriends[6] = "Gamal";  
  
console.log(myFriends.length) // 7
```

ARRAY METHODES

```
// Index Start From 0 [ 0, 1, 2, 3 ... ]  
  
let myFriends = ["Ahmed", "Mohamed", "Sayed", "Alaa"];  
  
myFriends.length = 2;  
  
console.log(myFriends); // [ 'Ahmed', 'Mohamed' ]  
  
myFriends[myFriends.length] = "Rachid";  
  
console.log(myFriends); // [ 'Ahmed', 'Mohamed', 'Rachid' ]  
  
myFriends[6] = "Gamal";  
  
console.log(myFriends.length) // 7
```

ADD AND REMOVE FROM ARRAY

```
<script type="text/javascript">
/*
  Arrays Methods [ Adding And Removing]
  - unshift("", "") Add Element To The First
  - push("", "") Add Element To The End
  - shift() Remove First Element From Array
  - pop() Remove Last Element From Array
*/
let myFriends = ["Ahmed", "Mohamed", "Sayed", "Alaa"];

console.log(myFriends); // [ 'Ahmed', 'Mohamed', 'Sayed', 'Alaa' ]

myFriends.unshift("Osama", "Nabil");

console.log(myFriends); // [ 'Osama', 'Nabil', 'Ahmed', 'Mohamed', 'Sayed', 'Alaa' ]

myFriends.push("Samah", "Eman");

console.log(myFriends); // [ 'Osama', 'Nabil', 'Ahmed', 'Mohamed', 'Sayed', 'Alaa', 'Samah', 'Eman' ]

let first = myFriends.shift();

console.log(myFriends); // [ 'Nabil', 'Ahmed', 'Mohamed', 'Sayed', 'Alaa', 'Samah', 'Eman' ]

console.log(`First Name Is ${first}`); // First Name Is Osama

let last = myFriends.pop();

console.log(myFriends); // [ 'Nabil', 'Ahmed', 'Mohamed', 'Sayed', 'Alaa', 'Samah' ]

console.log(`Last Name Is ${last}`); // Last Name Is Eman
</script>
```

SEARCHING ARRAY

```
<script type="text/javascript">
  /*
  Arrays Methods [Search]
  - indexOf(Search Element, From Index [Opt])
  - lastIndexOf(Search Element, From Index [Opt])
  - includes(valueToFind, fromIndex [Opt]) [ES7]
  */
let myFriends = ["Ahmed", "Mohamed", "Sayed", "Alaa", "Ahmed"];

console.log(myFriends); // [ 'Ahmed', 'Mohamed', 'Sayed', 'Alaa', 'Ahmed' ]

console.log(myFriends.indexOf("Ahmed")); // 0
console.log(myFriends.indexOf("Ahmed", 2)); // 4

console.log(myFriends.lastIndexOf("Ahmed")); // 4
console.log(myFriends.lastIndexOf("Ahmed", -2)); // 0

console.log(myFriends.includes("Ahmed")); // true
console.log(myFriends.includes("Ahmed", 2)); // true

if (myFriends.lastIndexOf("Osama") === -1) {
  console.log("Not Found"); // Not Found
}
console.log(myFriends.indexOf("Osama")); // -1
console.log(myFriends.lastIndexOf("Osama")); // -1
</script>
```

SORTING ARRAY

```
<script type="text/javascript">  
/*  
 Arrays Methods [Sort]  
 - sort(Function [Opt])  
 - reverse  
 */  
  
let myFriends = [10, "Sayed", "Mohamed", "90", 9000, 100, 20, "10", -20, -10];  
console.log(myFriends); // [10, 'Sayed', 'Mohamed', '90', 9000, 100, 20, '10', -20, -10]  
console.log(myFriends.sort().reverse()); // ['Sayed', 'Mohamed', 9000, '90', 20, 100, '10', 10, -20, -10]  
  
</script>
```

SLICING ARRAY

```
<script type="text/javascript">
    /*
    *
    * Arrays Methods [Slicing]
    - slice(Start [Opt], End [Opt] Not Including End)
    --- slice() => All Array
    --- If Start Is Undefined => 0
    --- Negative Count From End
    --- If End Is Undefined || > Indexes => Slice To The End Array.length
    --- Return New Array
    - splice(Start [Mand], DeleteCount [Opt] [0 No Remove], The Items To Add [Opt])
    --- If Negative => Start From The End
    */

let myFriends = ["Ahmed", "Sayed", "Ali", "Osama", "Gamal", "Ameer"];
console.log(myFriends); // ["Ahmed", "Sayed", "Ali", "Osama", "Gamal", "Ameer"]
console.log(myFriends.slice()); // ['Ahmed', 'Sayed', 'Ali', 'Osama', 'Gamal', 'Ameer' ]
console.log(myFriends.slice(1)); // ['Sayed', 'Ali', 'Osama', 'Gamal', 'Ameer' ]
console.log(myFriends.slice(1, 3)); // ['Sayed', 'Ali' ]
console.log(myFriends.slice(-3)); // ['Osama', 'Gamal', 'Ameer' ]
console.log(myFriends.slice(1, -2)); // ['Sayed', 'Ali', 'Osama' ]
console.log(myFriends.slice(-4, -2)); // ['Ali', 'Osama' ]
console.log(myFriends); // ['Ahmed', 'Sayed', 'Ali', 'Osama', 'Gamal', 'Ameer' ]

myFriends.splice(1, 2, "Sameer", "Samara");

console.log(myFriends); // ['Ahmed', 'Sameer', 'Samara', 'Osama', 'Gamal', 'Ameer' ]

</script>
```

JOINING ARRAY

```
<script type="text/javascript">

/*
  Arrays Methods [Joining]
  - concat(array, array) => Return A New Array
  - join(Separator)
*/

let myFriends = ["Ahmed", "Sayed", "Ali", "Osama", "Gamal", "Ameer"];
let myNewFriends = ["Samar", "Sameh"];
let schoolFriends = ["Haytham", "Shady"];

let allFriends = myFriends.concat(myNewFriends);
console.log(allFriends); // ['Ahmed', 'Sayed', 'Ali', 'Osama', 'Gamal', 'Ameer', 'Samar', 'Sameh']

allFriends = myFriends.concat(myNewFriends, schoolFriends, "Gameel", [1, 2]);
console.log(allFriends); //['Ahmed', 'Sayed', 'Ali', 'Osama', 'Gamal', 'Ameer', 'Samar', 'Sameh', 'Haytham', 'Shady', 'Gameel', 1, 2]

console.log(allFriends.join()); //Ahmed,Sayed,Ali,Osama,Gamal,Ameer,Samar,Sameh,Haytham,Shady,Gameel,1,2
console.log(allFriends.join("")); //AhmedSayedAliOsamaGamalAmeerSamarSamehHaythamShadyGameel12
console.log(allFriends.join(" @ ")); // Ahmed @ Sayed @ Ali @ Osama @ Gamal @ Ameer @ Samar @ Sameh @ Haytham @ Shady @ Gameel @ 1 @ 2
console.log(allFriends.join("|")); //Ahmed|Sayed|Ali|Osama|Gamal|Ameer|Samar|Sameh|Haytham|Shady|Gameel|1|2
console.log(allFriends.join("|").toUpperCase()); //AHMED|SAYED|ALI|OSAMA|GAMAL|AMEER|SAMAR|SAMEH|HAYTHAM|SHADY|GAMEEL|1|2

</script>
```

ARRAY CHALLENGE

```
<script type="text/javascript">
/*
| Array Challenge
*/
let zero = 0;

let counter = 3;

let my = ["Ahmed", "Mazero", "Elham", "Osama", "Gamal", "Ameer"];

// Write Code Here

console.log(my); // ["Osama", "Elham", "Mazero", "Ahmed"];

// don't use number use only variables
console.log(my.slice("????")); // ["Elham", "Mazero"]

console.log(); // "Elzero"

console.log(); // "ro"

</script>
```

DATE OBJECT

- l'objet date nous permet de travailler avec la date.
- Une date se compose d'une année, d'un mois, d'un jour, d'une heure, d'une minute, d'une seconde et de millisecondes.
- à l'aide de new Date (), créez un nouvel objet date avec la date et l'heure actuelles.

```
<script>
  var d = new Date()
  console.log(d)
</script>
```

DATE OBJECT

- Les autres méthodes d'initialisation des dates permettent de créer de nouveaux objets date à partir de la date et de l'heure spécifiées.
- Les dates JavaScript sont calculées en millisecondes à partir du 01 janvier 1970 00:00:00 Temps universel (UTC). Un jour contient 86 400 000 millisecondes.

```
/*
new Date()
new Date(year, month, day, hours, minutes, seconds, milliseconds)
new Date(milliseconds)
new Date(date string)
*/
var d = new Date(86400000)
console.log(d); //1970-01-02T00:00:00.000Z

// months start from 0 to 11
d = new Date(2018, 15, 24, 10, 33, 30, 0);
console.log(d); //2019-04-24T10:33:30.000Z

d = new Date("February 25, 2022 09:00:00")
console.log(d); //2022-02-25T09:00:00.000Z
```

DATE OBJECT

- Lorsqu'un objet Date est créé, un certain nombre de méthodes permettent d'effectuer des opérations sur celui-ci.

Method	Description
getFullYear()	gets the year
getMonth()	gets the month
getDate()	gets the day of the month
getDay()	gets the day of the week
getHours()	gets the hour
getMinutes()	gets the minutes
getSeconds()	gets the seconds
getMilliseconds()	gets the milliseconds

DATE OBJECT

- Nous avons déclaré une fonction `printTime()`, qui récupère l'heure actuelle dans l'objet `date` et l'imprime à l'écran. Nous avons ensuite appelé la fonction une fois par seconde, à l'aide de la méthode `setInterval`.

```
<script>

    function printTime() {
        var d = new Date();
        var hours = d.getHours();
        var mins = d.getMinutes();
        var secs = d.getSeconds();
        document.body.innerHTML = hours+":"+mins+":"+secs;
    }
    setInterval(printTime, 1000);
</script>
```

LES OPÉRATEURS NUMÉRIQUES

Les opérateurs numériques effectuent des opérations sur les variables et les valeurs.

opérateur	nom	exemple
+	Addition	$x + y$
-	Soustraction	$x - y$
*	Multiplication	$x * y$
/	Division	x / y
%	Reste de division	$x \% y$

LES OPÉRATEURS D'AFFECTATION

affectation	pareil que	Description
$x += y$	$x = x + y$	Addition
$x -= y$	$x = x - y$	Soustraction
$x *= y$	$x = x * y$	Multiplication
$x /= y$	$x = x / y$	Division
$x %= y$	$x = x \% y$	Reste de division
$x += y$ (pour les chaînes de caractère)	$x = x + y$	concatène à la suite de la chaîne existante
$x--$	$x -= 1$	Diminuer de 1
$x++$	$x += 1$	Augmenter de 1

LES OPÉRATEURS DE COMPARAISON

Opérateur	Exemple	Résultat
<code>==</code>	<code>x == y</code>	renvoie vrai si <code>x</code> est égal à <code>y</code>
<code>==></code>	<code>x ==> y</code>	renvoie vrai si <code>x</code> est égal à <code>y</code> , et ils sont de même type
<code>!=</code>	<code>x != y</code>	renvoie vrai si <code>x</code> n'est pas égal à <code>y</code>
<code>!==</code>	<code>x !== y</code>	renvoie vrai si <code>x</code> n'est pas égal à <code>y</code> , ou s'ils ne sont pas du même type
<code>></code>	<code>x > y</code>	Renvoi vrai si <code>x</code> est supérieur strictement à <code>y</code>
<code>>=</code>	<code>x >= y</code>	Renvoi vrai si <code>x</code> est supérieur ou égal à <code>y</code>
<code><</code>	<code>x < y</code>	Renvoi vrai si <code>x</code> est inférieur strictement à <code>y</code>
<code><=</code>	<code>x <= y</code>	Renvoi vrai si <code>x</code> est inférieur ou égal à <code>y</code>

LES OPÉRATEURS LOGIQUES

opérateur	exemple	résultat
<code>&&</code>	<code>x && y</code>	Renvoie vrai si x et y sont vrais
<code> </code>	<code>x y</code>	Renvoie vrai si x ou y est vrai
<code>!</code>	<code>!x</code>	Renvoie vrai si x n'est pas vrai

BOÎTE DE DIALOGUE

Javascript prend en charge trois types de boîtes de dialogues importants, ces boîtes de dialogue peuvent être utilisées pour déclencher une alerte, ou pour obtenir une confirmation sur n'importe quelle opération ou pour lire des entrées de la part des utilisateurs .

BOÎTE DE DIALOGUE D'ALERTE

Une boîte de dialogue alerte est principalement utilisée pour donner un message d'avertissement aux utilisateurs. La boîte d'alerte donne seulement un button « OK »

```
<script type="text/javascript">  
    alert("Ceci est un message d'avertissement!");  
</script>
```



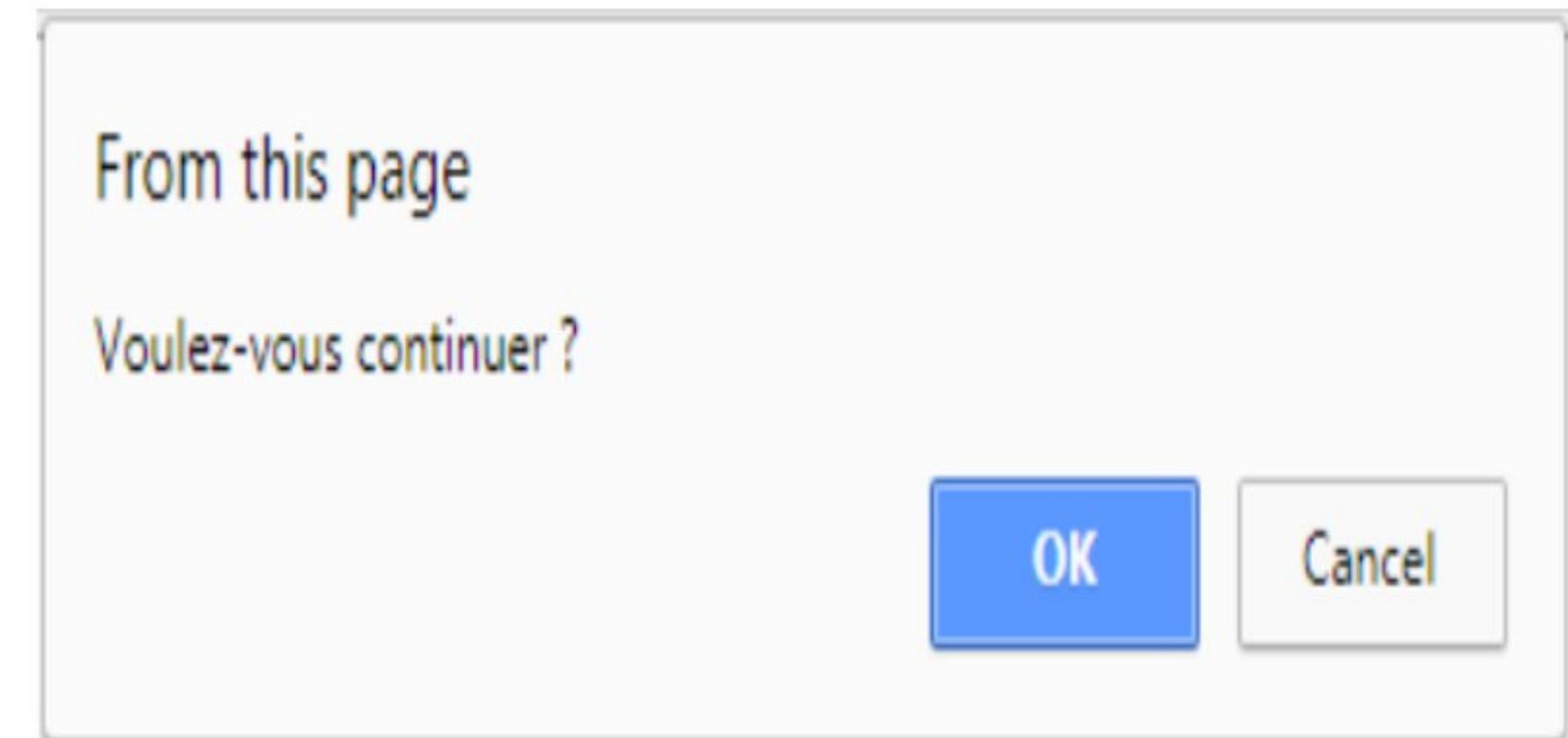
BOÎTE DE DIALOGUE DE CONFIRMATION

Utilisée pour obtenir le consentement de l'utilisateur sur n'importe quelle option. Il affiche une boîte de dialogue avec deux buttons : « OK » et « Annuler »

si l'utilisateur clique « OK », la méthode `confirm()` retournera `true`, sinon s'il clique sur « Annuler » renvoie `false`

BOÎTE DE DIALOGUE DE CONFIRMATION

```
<script type="text/javascript">  
  
var retVal = confirm("Voulez-vous continuer ?");  
  
if (retVal == true) {  
    document.write("L'utilisateur veut continuer!");  
}  
  
else {  
    document.write("L'utilisateur ne veut pas continuer!");  
}  
  
</script>
```



BOÎTE DE DIALOGUE DE SAISIE

Est très utile lorsque vous voulez faire apparaître une zone de texte pour obtenir une entrée utilisateur.

Cette boîte est affichée par la méthode `prompt()` qui prend 2 paramètres :

1. Une étiquette que vous voulez afficher dans la zone de texte
2. Une chaîne par défaut à afficher dans la zone texte

Cette boîte a 2 buttons « OK » et « Annuler »

Si l'utilisateur clique « OK » `prompt()` renvoie la valeur entrée par dans la zone de texte

Si l'utilisateur clique « Annuler » renvoie null

BOÎTE DE DIALOGUE DE SAISIE

```
<script type="text/javascript">  
    var retVal = prompt("Entrez votre nom : ", "votre nom ici");  
    document.write("Vous avez entré:" + retVal);  
</script>
```

Entrez votre nom :

votre nom ici

OK

Annuler

BOÎTE DE DIALOGUE DE SAISIE

```
<script type="text/javascript">  
var retVal = prompt("Entrez un nombre : ", "");  
var somme = parseInt(retVal) + 6;  
document.write("Résultat :" + somme);  
</script>
```

```
<script type="text/javascript">  
var retVal = +prompt("Entrez un nombre : ", "");  
var somme = retVal + 6;  
document.write("Résultat :" + somme);  
</script>
```

Attention !

La méthode `prompt()` renvoie du texte . Ainsi , si vous saisie un nombre à l'entrée. Vous devez convertir le texte en nombre en utilisant `parseInt()`

EXERCICE D'APPLICATION

Réaliser un programme qui permet de lire deux variables entières A et B et d'afficher leur somme .

EXERCICE D'APPLICATION

Réaliser un programme qui permet d'afficher dans le document HTML le carré et le cube d'un nombre saisi par l'utilisateur

- 3
- $3^2 = 6$
- $3^3 = 27$

EXERCICE D'APPLICATION

Réaliser un programme qui permet de lire les données suivantes :

- Un nom d'un produit informatique
- La quantité demandée
- Le prix unitaire

Et de calculer et afficher un tableau contenant les données suivantes :

- Le Montant Hors Taxe
- Le Montant de la TVA sachant que la TVA est de 20%
- Le Montant TTC (Toutes Taxes Comprises)

La table résultante devrait ressembler à ceci :

Le nom du produit	GTX 1080
Le montant Hors Taxe	13190 DH
Le montant TVA	2638 DH
Le montant à payer (TTC)	15828 DH

STRUCTURE CONDITIONNELLE

- Souvent, lorsque nous écrivons du code, nous voulons effectuer différentes actions en fonction de différentes conditions.
- Et c'est là que les instructions conditionnelles entrent en jeu.
- Il y a un tas de conditionnels différents à couvrir, mais nous allons commencer par l'un des plus utiles : « `if` »
- Nous utilisons « `if` » pour spécifier un bloc de code que nous voulons exécuter si une condition spécifiée est vraie.

STRUCTURE CONDITIONNELLE

```
if (condition) {  
    //instruction(s) à exécuter si l'expression est vraie  
}
```

```
<script type="text/javascript">  
    var age = 20;  
    if (age > 18) {  
        document.write("<b>Tu peux conduire!</b>");  
    }  
</script>
```

STRUCTURE CONDITIONNELLE

Nous pouvons utiliser l'instruction « `else` » pour spécifier un bloc de code qui sera exécuté si la condition est fausse. Comme ceci :

```
if (expression) {  
    // executed if condition is true  
} else {  
    // executed if condition is false  
}
```

STRUCTURE CONDITIONNELLE

```
<script type="text/javascript">  
    var age = 15;  
    if (age > 18) {  
        document.write("<b>Tu peux conduire!</b>");  
    }else {  
        document.write("<b>Tu es trop jeune!</b>");  
    }  
</script>
```

STRUCTURE CONDITIONNELLE

- Nous avons vu else, nous avons vu if, il est temps de rencontrer else if.
- L'instruction else if est utile car elle nous permet de spécifier une nouvelle condition si la première est fausse.

```
if (expression 1) {  
    // Instruction(s) à exécuter si l'expression 1 est vraie  
} else if (expression 2) {  
    // Instruction(s) à exécuter si l'expression 2 est vraie  
} else if (expression 3) {  
    // Instruction(s) à exécuter si l'expression 3 est vraie  
} else {  
    // Instructions(s) à exécuter si aucune expression n'est vraie  
}
```

STRUCTURE CONDITIONNELLE

```
<script type="text/javascript">  
    var course = 5;  
    if (course == 1) {  
        document.write("<h1>HTML Tutorial</h1>");  
    } else if (course == 2) {  
        document.write("<h1>CSS Tutorial</h1>");  
    } else if (course == 3) {  
        document.write("<h1>PHP Tutorial</h1>");  
    } else {  
        document.write("<h1>JavaScript Tutorial</h1>");  
    }  
</script>
```

STRUCTURE CONDITIONNELLE

- Mais qu'en est-il si vous devez tester plusieurs conditions ? Dans ce cas, écrire des instructions `if` `else` pour chaque condition n'est peut-être pas la meilleure solution.
- Au lieu de cela, nous pouvons utiliser l'instruction `switch` pour effectuer différentes actions en fonction de différentes conditions.

```
switch (expression) {  
    case n1:  
        // instructions  
        break;  
    case n2:  
        // instructions  
        break;  
    default:  
        // instructions  
}
```

STRUCTURE CONDITIONNELLE

```
<script type="text/javascript">
    var day = 2;
    switch (day) {
        case 1:
            document.write("Monday");
            break;
        case 2:
            document.write("Tuesday");
            break;
        case 3:
            document.write("Wednesday");
            break;
        default:
            document.write("Another day");
    }
</script>
```

STRUCTURE RÉPÉTITIVES

- Les boucles peuvent exécuter un bloc de code un certain nombre de fois. Elles sont pratiques lorsque vous souhaitez exécuter le même code à plusieurs reprises, en ajoutant une valeur différente à chaque fois.
- JavaScript propose trois types de boucles : `for`, `while` et `do while`.
- Nous allons commencer par la boucle « `for` » classique.

LA BOUCLE FOR

Le boucle for est utilisée lorsque vous savez à l'avance combien de fois le script doit être exécuté.

```
for (init; test; increment) {  
    //code à exécuter;  
}
```

Paramètres:

init: est exécutée avant le début de la boucle (le bloc de code).

test: définit la condition d'exécution de la boucle (le bloc de code).

increment: est exécutée chaque fois après l'exécution de la boucle (le bloc de code).

STRUCTURE RÉPÉTITIVE

```
<script type="text/javascript">  
  
    for (var i = 0; i < 5; i++) {  
  
        document.write(i + "<br />");  
  
    }  
  
</script>
```

1
2
3
4
5

LA BOUCLE WHILE

La boucle while exécute un bloc de code tant que la condition spécifiée est vraie

```
while (condition) {  
    // code à exécuter  
}
```

Si la condition ne devient jamais fausse, l'instruction continuera à s'exécuter indéfiniment.

STRUCTURE RÉPÉTITIVE

```
<script type="text/javascript">  
  
    var i = 0;  
  
    while (i <= 10) {  
  
        document.write(i + "<br />");  
  
        i++;  
    }  
  
</script>
```

0
1
2
3
4
5
6
7
8
9
10

LA BOUCLE DO WHILE

La boucle do ... while exécute toujours le bloc de code une fois, vérifie la condition et répète la boucle tant que la condition spécifiée est vraie.

```
do{  
    //code à exécuter  
}while (condition);
```

Que la condition soit vraie ou fausse, le code sera exécuté au moins une fois, ce qui peut être nécessaire dans certaines situations.

STRUCTURE RÉPÉTITIVE

```
<script type="text/javascript">  
  
    var i = 20;  
  
    do{  
  
        document.write(i + "<br />");  
  
        i++;  
  
    } while (i<=25);  
  
</script>
```

20
21
22
23
24
25

Contrôle des boucles

- Javascript fournit un contrôle complet pour gérer les boucles :
 - Situation où vous devez sortir du boucle sans atteindre la fin
 - Situation où vous souhaitez ignorer un partie de votre bloc de code et lancer l'itération suivante de la boucle
- Pour gérer toutes ces situations , javascript fournit les instructions break et continue . Ces instructions sont utilisées pour sortir immédiatement de n 'importe quelle boucle ou pour commencer l'itération suivante de n 'importe quelle boucle respectivement

Contrôle des boucles

```
<script type="text/javascript">  
  
    for (var index = 1; index < 10; index++) {  
  
        if (index == 5) {  
  
            break; // casse complètement la boucle  
        }  
  
        document.write(index);  
    }  
  
</script>
```

```
<script type="text/javascript">  
  
    for (var index = 1; index < 10; index++) {  
  
        if (index == 5) {  
  
            continue; // sauter le reste du corps de  
        }  
  
        document.write(index)  
    }  
  
</script>
```

LOOP CHALLENGE

```
<script type="text/javascript">  
/*  
 | Loop Challenge  
 */  
  
let myAdmins = ["Ahmed", "Osama", "Sayed", "Stop", "Samera"];  
let myEmployees = ["Amgad", "Samah", "Ameer", "Omar", "Othman", "Amany", "Samia", "Anwar"];  
  
document.write(`<div>We Have X Admins</div>`);  
  
</script>
```

We Have 3 Admins

The Admin For Team 1 Is Ahmed

Team Members:

- 1 Amgad
- 2 Ameer
- 3 Amany

The Admin For Team 2 Is Osama

Team Members:

- 1 Omar
- 2 Othman

The Admin For Team 3 Is Sayed

Team Members:

- 1 Samah
- 2 Samia

<div>We Have 3 Admins</div>

<hr>

... ▶<div> == \$0

"The Admin For Team 1 Is Ahmed"

<h3>Team Members: </h3>

<p>- 1 Amgad</p>

<p>- 2 Ameer</p>

<p>- 3 Amany</p>

</div>

<hr>

▶<div>...</div>

<hr>

▶<div>...</div>