



# JavaScript

---



---

# Introduction



---

## **Définition :**

**JavaScript** est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs.

C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés.

En outre, les fonctions sont des objets de première classe.

Ces scripts vont être gérés et exécutés par le navigateur lui-même sans devoir faire appel aux ressources du serveur. Langage non compilé mais interprété.



---

## Historique :

Brendan Eich : langage script côté serveur **LiveScript** pour Mosaic Communications Corporation.

Une version orientée client de **LiveScript**.

Changement le nom de **LiveScript** pour **JavaScript**.

Décembre 1995 :

Sun et Netscape annoncent la sortie de **JavaScript**.

En mars 1996 :

Netscape : moteur **JavaScript** Netscape Navigator 2.0.

## JavaScript n'est pas Java :

Il importe de savoir que **JavaScript** est totalement différent de **Java**.

Bien que les deux soient utilisés pour créer des pages Web évoluées, bien que les deux reprennent le terme Java, nous avons là deux outils informatiques bien différents.

### **Javascript**

Code intégré dans la page Html  
Code interprété par le browser au moment de l'exécution  
Codes de programmation simples mais pour des applications limitées  
Permet d'accéder aux objets du navigateur  
Confidentialité des codes nulle (code source visible)

### **Java**

Module (applet) distinct de la page Html  
Code source compilé avant son exécution  
Langage de programmation beaucoup plus complexe mais plus performant  
N'accède pas aux objets du navigateur  
Sécurité (code source compilé)



---

# Variable

## **Variable :**

Variable non typée : l'affectation type implicitement.

Déclaration par le mot clé *var*.

Variable locale dans les fonctions.

Variable globale hors des fonctions.

Variable déclarée mais non affectée : *undefined*

*Null* pour noter l'absence de valeur.

`var myVariable = 2;`

```
//Déclaration de i, de j et de k.  
var i, j, k;  
  
//Affectation de i.  
i = 1;  
  
//Déclaration et affectation de prix.  
var prix = 0;
```

## Variable :

En **JavaScript**, il n'existe pas de portée lexicale de type bloc comme en **Java** (ou dans de nombreux autres langages).

La portée des variables est au niveau de la fonction où elles sont déclarées, c'est-à-dire que toute variable déclarée dans le corps d'une fonction y est utilisable, peu importe où se situe sa déclaration.

```
function test()
{
    i = 1; // cette affectation affecte la
           // variable locale i (qui est déclarée plus loin)
    {
        for(var j = i; j > 0; j--)
        {
            // déclaration de la variable locale j
            var k = j*2; // déclaration de la variable locale k
            l = 42;      // aucune variable locale ne s'appelant l,
                        // c'est la propriété l du Global Object qui va être changée
        }
        var i; // déclaration de la variable locale i
    }

    return [i, j, k]; // toutes les variables déclarées dans la
                     // fonction sont utilisables ici (même en dehors de leur bloc)
}
```



## Variable :

La portée lexicale d'une variable locale à une fonction A s'étend aux déclarations de fonctions faites à l'intérieur de A, et ainsi de suite. L'inverse n'est pas vrai.

```
function A()
{
    var p = 2;           // déclaration d'une variable p locale à A

    function B(p)
    {
        // le paramètre p se comporte en portée comme une
        // variable locale à B, qui masque la variable p de A.
        var j = p;       // déclaration de la variable j locale à B
        return j - i;     // B a accès à la variable i déclarée dans A
    }

    var i = 5;           // déclaration (et affectation) de i

    alert( B(p*3) );     // appelle B(2*3) car p fait ici référence à la variable
                        // locale à A ; affichera 1 (2*3 - 5)
    return j;            // cette ligne provoquera une ReferenceError: A n'a pas accès à la variable j locale à B
}
```



---

# Fonction

## **Fonctions de base :**

### JavaScript Global Properties

Property	Description
<u>Infinity</u>	A numeric value that represents positive/negative infinity
<u>NaN</u>	"Not-a-Number" value
<u>undefined</u>	Indicates that a variable has not been assigned a value

## **Fonctions de base :**

### JavaScript Global Functions

Function	Description
<u><a href="#">decodeURI()</a></u>	Decodes a URI
<u><a href="#">decodeURIComponent()</a></u>	Decodes a URI component
<u><a href="#">encodeURI()</a></u>	Encodes a URI
<u><a href="#">encodeURIComponent()</a></u>	Encodes a URI component
<u><a href="#">escape()</a></u>	Deprecated in version 1.5. Use <u><a href="#">encodeURI()</a></u> or <u><a href="#">encodeURIComponent()</a></u> instead

## **Fonctions de base :**

### JavaScript Global Functions

<u>eval()</u>	Evaluates a string and executes it as if it was script code
<u>isFinite()</u>	Determines whether a value is a finite, legal number
<u>isNaN()</u>	Determines whether a value is an illegal number
<u>Number()</u>	Converts an object's value to a number
<u>parseFloat()</u>	Parses a string and returns a floating point number
<u>parseInt()</u>	Parses a string and returns an integer
<u>String()</u>	Converts an object's value to a string
<u>unescape()</u>	Deprecated in version 1.5. Use <u>decodeURI()</u> or <u>decodeURIComponent()</u> instead



---

## **Déclaration sans paramètre :**

```
function maFonction()  
{ }
```



---

## **Declaration avec paramètres d'entrée :**

```
function maFonction(parametre1, parametre2)  
{  
}
```

```
function maFonction(parametre1)  
{  
}
```



---

## **Déclaration avec paramètres de sortie :**

```
function direBonjour()  
{  
  return 'Bonjour !';  
}
```





---

## **Fonction anonyme :**

```
function()  
{  
  return 'Bonjour !';  
}
```

```
var direBonjour = function()  
{  
  return 'Bonjour !';  
};
```

## Fonction en paramètre :

```
function carre(x)
{
    return x*x;
}

function map (a,f)
{
    for (var i=0 ; i<a.length ; i++)
    {
        a[i]=f(a[i]);
    }

    return a.toString();
}

map([1,2,3],carre);
```



---

# Objets



---

## **Prototypes :**

Les variables contiennent des objets, qui peuvent être des nombres, des chaînes de caractères ou des booléens.

Mais le **JavaScript** n'est pas un langage orienté objet (**C++**, **C#** ou **Java**), mais un langage orienté objet par prototype.

Les objets contiennent trois choses :

- Un constructeur
- Des propriétés
- Des méthodes



---

# Prototype booléen

## **Propriétés booléen :**

### Boolean Properties

Property	Description
<u>constructor</u>	Returns the function that created JavaScript's Boolean prototype
<u>prototype</u>	Allows you to add properties and methods to the Boolean prototype

### Boolean Methods

Method	Description
<u>toString()</u>	Converts a boolean value to a string, and returns the result
<u>valueOf()</u>	Returns the primitive value of a boolean



---

# Prototype entier

## **Prototype entier :**

### Number Properties

Property	Description
<u>constructor</u>	Returns the function that created JavaScript's Number prototype
<u>MAX_VALUE</u>	Returns the largest number possible in JavaScript
<u>MIN_VALUE</u>	Returns the smallest number possible in JavaScript
<u>NEGATIVE_INFINITY</u>	Represents negative infinity (returned on overflow)
<u>NaN</u>	Represents a "Not-a-Number" value
<u>POSITIVE_INFINITY</u>	Represents infinity (returned on overflow)
<u>prototype</u>	Allows you to add properties and methods to an object



## **Prototype entier :**

### Number Methods

Method	Description
<u>toExponential(x)</u>	Converts a number into an exponential notation
<u>toFixed(x)</u>	Formats a number with x numbers of digits after the decimal point
<u>toPrecision(x)</u>	Formats a number to x length
<u>toString()</u>	Converts a number to a string
<u>valueOf()</u>	Returns the primitive value of a number



---

# Prototype chaîne de caractères

---

## **Prototype chaîne de caractères :**

### String Properties

Property	Description
<u>constructor</u>	Returns the string's constructor function
<u>length</u>	Returns the length of a string
<u>prototype</u>	Allows you to add properties and methods to an object

## **Prototype chaîne de caractères :**

### String Methods

Method	Description
<u>charAt()</u>	Returns the character at the specified index (position)
<u>charCodeAt()</u>	Returns the Unicode of the character at the specified index
<u>concat()</u>	Joins two or more strings, and returns a new joined strings
<u>fromCharCode()</u>	Converts Unicode values to characters
<u>indexOf()</u>	Returns the position of the first found occurrence of a specified value in a string
<u>lastIndexOf()</u>	Returns the position of the last found occurrence of a specified value in a string
<u>localeCompare()</u>	Compares two strings in the current locale
<u>match()</u>	Searches a string for a match against a regular expression, and returns the matches
<u>replace()</u>	Searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced

## **Prototype chaîne de caractères :**

### String Methods

<u>search()</u>	Searches a string for a specified value, or regular expression, and returns the position of the match
<u>slice()</u>	Extracts a part of a string and returns a new string
<u>split()</u>	Splits a string into an array of substrings
<u>substr()</u>	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
<u>substring()</u>	Extracts the characters from a string, between two specified indices
<u>toLocaleLowerCase()</u>	Converts a string to lowercase letters, according to the host's locale
<u>toLocaleUpperCase()</u>	Converts a string to uppercase letters, according to the host's locale
<u>toLowerCase()</u>	Converts a string to lowercase letters
<u>toString()</u>	Returns the value of a String object
<u>toUpperCase()</u>	Converts a string to uppercase letters
<u>trim()</u>	Removes whitespace from both ends of a string
<u>valueOf()</u>	Returns the primitive value of a String object



---

# Prototype date



---

## **Prototype date :**

### Date Object Properties

Property	Description
<u>constructor</u>	Returns the function that created the Date object's prototype
<u>prototype</u>	Allows you to add properties and methods to an object

## **Prototype date :**

### Date Object Methods

Method	Description
<u>getDate()</u>	Returns the day of the month (from 1-31)
<u>getDay()</u>	Returns the day of the week (from 0-6)
<u>getFullYear()</u>	Returns the year (four digits)
<u>getHours()</u>	Returns the hour (from 0-23)
<u>getMilliseconds()</u>	Returns the milliseconds (from 0-999)
<u>getMinutes()</u>	Returns the minutes (from 0-59)
<u>getMonth()</u>	Returns the month (from 0-11)
<u>getSeconds()</u>	Returns the seconds (from 0-59)
<u>getTime()</u>	Returns the number of milliseconds since midnight Jan 1, 1970
<u>getTimezoneOffset()</u>	Returns the time difference between UTC time and local time, in minutes



## **Prototype date :**

### Date Object Methods

<u>getUTCDate()</u>	Returns the day of the month, according to universal time (from 1-31)
<u>getUTCDay()</u>	Returns the day of the week, according to universal time (from 0-6)
<u>getUTCFullYear()</u>	Returns the year, according to universal time (four digits)
<u>getUTCHours()</u>	Returns the hour, according to universal time (from 0-23)
<u>getUTCMilliseconds()</u>	Returns the milliseconds, according to universal time (from 0-999)
<u>getUTCMinutes()</u>	Returns the minutes, according to universal time (from 0-59)
<u>getUTCMonth()</u>	Returns the month, according to universal time (from 0-11)
<u>getUTCSeconds()</u>	Returns the seconds, according to universal time (from 0-59)
<u>getYear()</u>	<b>Deprecated.</b> Use the <u>getFullYear()</u> method instead
<u>parse()</u>	Parses a date string and returns the number of milliseconds since January 1, 1970
<u>setDate()</u>	Sets the day of the month of a date object

## **Prototype date :**

### **Date Object Methods**

<u>setFullYear()</u>	Sets the year (four digits) of a date object
<u>setHours()</u>	Sets the hour of a date object
<u>setMilliseconds()</u>	Sets the milliseconds of a date object
<u>setMinutes()</u>	Set the minutes of a date object
<u>setMonth()</u>	Sets the month of a date object
<u>setSeconds()</u>	Sets the seconds of a date object
<u>setTime()</u>	Sets a date to a specified number of milliseconds after/before January 1, 1970
<u>setUTCDate()</u>	Sets the day of the month of a date object, according to universal time
<u>setUTCFullYear()</u>	Sets the year of a date object, according to universal time (four digits)
<u>setUTCHours()</u>	Sets the hour of a date object, according to universal time
<u>setUTCMilliseconds()</u>	Sets the milliseconds of a date object, according to universal time
<u>setUTCMinutes()</u>	Set the minutes of a date object, according to universal time
<u>setUTCMonth()</u>	Sets the month of a date object, according to universal time

## **Prototype date :**

### Date Object Methods

<u>setFullYear()</u>	Sets the year (four digits) of a date object
<u>setHours()</u>	Sets the hour of a date object
<u>setMilliseconds()</u>	Sets the milliseconds of a date object
<u>setMinutes()</u>	Set the minutes of a date object
<u>setMonth()</u>	Sets the month of a date object
<u>setSeconds()</u>	Sets the seconds of a date object
<u>setTime()</u>	Sets a date to a specified number of milliseconds after/before January 1, 1970
<u>setUTCDate()</u>	Sets the day of the month of a date object, according to universal time
<u>setUTCFullYear()</u>	Sets the year of a date object, according to universal time (four digits)
<u>setUTCHours()</u>	Sets the hour of a date object, according to universal time
<u>setUTCMilliseconds()</u>	Sets the milliseconds of a date object, according to universal time
<u>setUTCMinutes()</u>	Set the minutes of a date object, according to universal time
<u>setUTCMonth()</u>	Sets the month of a date object, according to universal time



---

# Prototype mathématique

## **Prototype mathématique :**

### Math Object Properties

Property	Description
<u>E</u>	Returns Euler's number (approx. 2.718)
<u>LN2</u>	Returns the natural logarithm of 2 (approx. 0.693)
<u>LN10</u>	Returns the natural logarithm of 10 (approx. 2.302)
<u>LOG2E</u>	Returns the base-2 logarithm of E (approx. 1.442)
<u>LOG10E</u>	Returns the base-10 logarithm of E (approx. 0.434)
<u>PI</u>	Returns PI (approx. 3.14)
<u>SQRT1_2</u>	Returns the square root of 1/2 (approx. 0.707)
<u>SQRT2</u>	Returns the square root of 2 (approx. 1.414)

## **Prototype mathématique :**

### Math Object Methods

Method	Description
<u><a href="#">abs(x)</a></u>	Returns the absolute value of x
<u><a href="#">acos(x)</a></u>	Returns the arccosine of x, in radians
<u><a href="#">asin(x)</a></u>	Returns the arcsine of x, in radians
<u><a href="#">atan(x)</a></u>	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
<u><a href="#">atan2(y,x)</a></u>	Returns the arctangent of the quotient of its arguments
<u><a href="#">ceil(x)</a></u>	Returns x, rounded upwards to the nearest integer
<u><a href="#">cos(x)</a></u>	Returns the cosine of x (x is in radians)
<u><a href="#">exp(x)</a></u>	Returns the value of $E^x$
<u><a href="#">floor(x)</a></u>	Returns x, rounded downwards to the nearest integer
<u><a href="#">log(x)</a></u>	Returns the natural logarithm (base E) of x

## **Prototype mathématique :**

### Math Object Methods

<u>max(x,y,z,...,n)</u>	Returns the number with the highest value
<u>min(x,y,z,...,n)</u>	Returns the number with the lowest value
<u>pow(x,y)</u>	Returns the value of x to the power of y
<u>random()</u>	Returns a random number between 0 and 1
<u>round(x)</u>	Rounds x to the nearest integer
<u>sin(x)</u>	Returns the sine of x (x is in radians)
<u>sqrt(x)</u>	Returns the square root of x
<u>tan(x)</u>	Returns the tangent of an angle



---

# Prototype tableau





---

## **Prototype tableau :**

### Array Properties

Property	Description
<u>constructor</u>	Returns the function that created the Array object's prototype
<u>length</u>	Sets or returns the number of elements in an array
<u>prototype</u>	Allows you to add properties and methods to an Array object

## **Prototype tableau :**

### Array Methods

Method	Description
<u>concat()</u>	Joins two or more arrays, and returns a copy of the joined arrays
<u>indexOf()</u>	Search the array for an element and returns its position
<u>join()</u>	Joins all elements of an array into a string
<u>lastIndexOf()</u>	Search the array for an element, starting at the end, and returns its position
<u>pop()</u>	Removes the last element of an array, and returns that element
<u>push()</u>	Adds new elements to the end of an array, and returns the new length
<u>reverse()</u>	Reverses the order of the elements in an array
<u>shift()</u>	Removes the first element of an array, and returns that element
<u>slice()</u>	Selects a part of an array, and returns the new array
<u>sort()</u>	Sorts the elements of an array
<u>splice()</u>	Adds/Removes elements from an array
<u>toString()</u>	Converts an array to a string, and returns the result
<u>unshift()</u>	Adds new elements to the beginning of an array, and returns the new length
<u>valueOf()</u>	Returns the primitive value of an array



---

# Opérateurs



---

## Opérateurs classiques :

Tous les opérateurs arithmétiques, de comparaison, conditionnels ou logiques ont la même syntaxe qu'en **C** ou en **Java**.

## Opérateurs spéciaux :

*instanceof* retourne vrai si l'objet spécifié est une instance de l'objet spécifié.

```
var cars = ["Saab", "Volvo", "BMW"];  
cars instanceof Array;
```

*void* calcule une expression et retourne *undefined*.

*in* retourne vrai si la propriété spécifiée est dans l'objet spécifié  
sinon faux.

*typeof* retourne le type d'une variable, object, fonction ou expression.

```
var i = 1; typeof i;
```

*delete* supprime une propriété d'un objet.



---

# DOM



---

## **Définition :**

Le **D**ocument **O**bject **M**odel est un standard du W3C qui décrit une interface indépendante de tout langage de programmation et de toute plate-forme, permettant à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu, la structure ou le style de documents HTML3 et XML4.

Le document peut ensuite être traité et les résultats de ces traitements peuvent être réincorporés dans le document tel qu'il sera présenté.



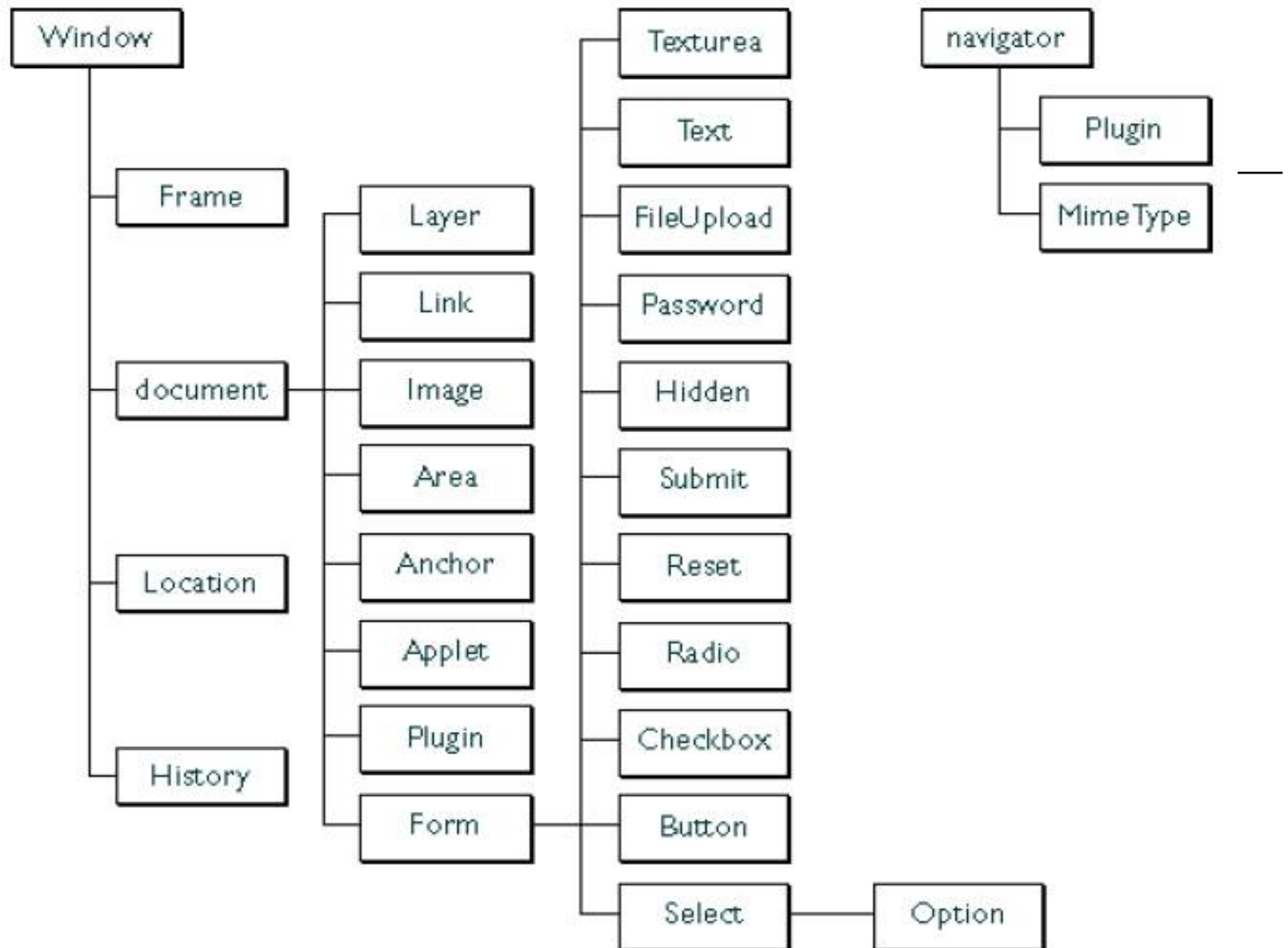
---

## **Arborescence :**

**DOM** permet de construire une arborescence de la structure d'un document et de ses éléments.

**DOM** est utilisé pour pouvoir modifier facilement des documents **XML** ou accéder au contenu des pages web.





## Accès aux éléments :

Par identifiant :

*document.getElementById();*

```
var ml = document.getElementById("maListe");  
ml.setAttribute("style","color:"+color);
```

Par nom :

*document.getElementsByTagName();*

Par collection :

*document.links[0];*

```
alert(document.title);  
  
for(var i = 0; i < document.links.length; ++i)  
{  
    alert(document.links[i]);  
}
```




---

## **Événements :**

La capture d'un événement consiste à exécuter une action (par exemple un programme en **JavaScript**) lorsque l'événement surveillé se produit dans le document.

## Plusieurs exemples d'événements :

<code>click</code>	Cliquer (appuyer puis relâcher) sur l'élément
<code>dblclick</code>	Double-cliquer sur l'élément
<code>mouseover</code>	Faire entrer le curseur sur l'élément
<code>mouseout</code>	Faire sortir le curseur de l'élément
<code>mousedown</code>	Appuyer (sans relâcher) sur le bouton gauche de la souris sur l'élément
<code>mouseup</code>	Relâcher le bouton gauche de la souris sur l'élément
<code>mousemove</code>	Faire déplacer le curseur sur l'élément
<code>keydown</code>	Appuyer (sans relâcher) sur une touche de clavier sur l'élément
<code>keyup</code>	Relâcher une touche de clavier sur l'élément
<code>keypress</code>	Frapper (appuyer puis relâcher) une touche de clavier sur l'élément
<code>focus</code>	« Cibler » l'élément
<code>blur</code>	Annuler le « ciblage » de l'élément
<code>change</code>	Changer la valeur d'un élément spécifique aux formulaires ( <code>input</code> , <code>checkbox</code> , etc.)
<code>select</code>	Sélectionner le contenu d'un champ de texte ( <code>input</code> , <code>textarea</code> , etc.)



---

# Intégration dans HTML

---

## Dans le corps du document et en dur :

```
<html>
  <body>
    <script type="text/javascript">
      alert('bonjour');
    </script>
  </body>
</html>
```

## Dans l'en tête du document et en dur :

```
<html>

  <head>
    <script type="text/javascript">
      function f () { alert('Au revoir'); }
    </script>
  </head>

  <body onUnload="f()">
  </body>
</html>
```

## Dans l'en-tête du document et externalisé :

```
<html>
  <head>
    <script type="text/javascript" src="fichier.js"></script>
  </head>

  <body>
    <input type="button" onclick="popup()" value="Clic">
  </body>
</html>
```