

- **Historique**
- **Éléments du langage JS**
- **HTML et JS**
- **Les objets JS et les objets du navigateur**
- **Prototypage**
- **Les événements**
- **DOM et JS**

Historique et versions

- Netscape donne naissance à LiveScript
 - 1995 : **LiveScript** devient JavaScript
 - Apparaît d'abord dans la version 2 du navigateur Netscape
 - Encadré par la norme **ECMAScript** (ES)
 - permet de rendre dynamique un site internet développé en HTML
 - Un langage interprété
 - Ecrit directement dans les pages WEB
 - **Événementiel**
 - A base d'objets
 - Faiblement typé
 - Ce n'est pas JAVA!



Historique et versions

- A été inventé par Brendan Eich en 1995
 - Est devenu un standard ECMA en 1997
 - ECMA-262 est le nom officiel du standard. ECMAScript est le nom officiel du langage



Anée	Nom	Description
1997	ECMAScript 1	Première Edition.
1998	ECMAScript 2	
1999	ECMAScript 3	Expressions Régulières. try/catch.
	ECMAScript 4	N'est jamais sortie
2009	ECMAScript 5	"strict mode". Support JSON.
2011	ECMAScript 5.1	
2015	ECMAScript 6	Classes et modules.
2016	ECMAScript 7	Opérateur exponentiel (**). Array.prototype.includes.

HTML et JAVASCRIPT

- Contrairement à un applet Java qui est un programme compilé, les scripts écrits en Javascript sont **interprétés**
 - Le Java, représenté par un ou plusieurs fichiers autonomes dont l'extension sera *.class ou *.jar, est invoqué par une balise HTML spécifique
 - Le code JavaScript s'intègre de deux manières avec le code HTML
 - **Insertion directe** dans le code HTML
 - Le code JavaScript s'insère le plus souvent dans la page HTML elle-même.
 - C'est la méthode la plus simple et la plus fréquemment utilisée par les développeurs de sites Internet.

```
<script language="JavaScript">  
.....  
</script>
```

- **Insertion comme un module externe**

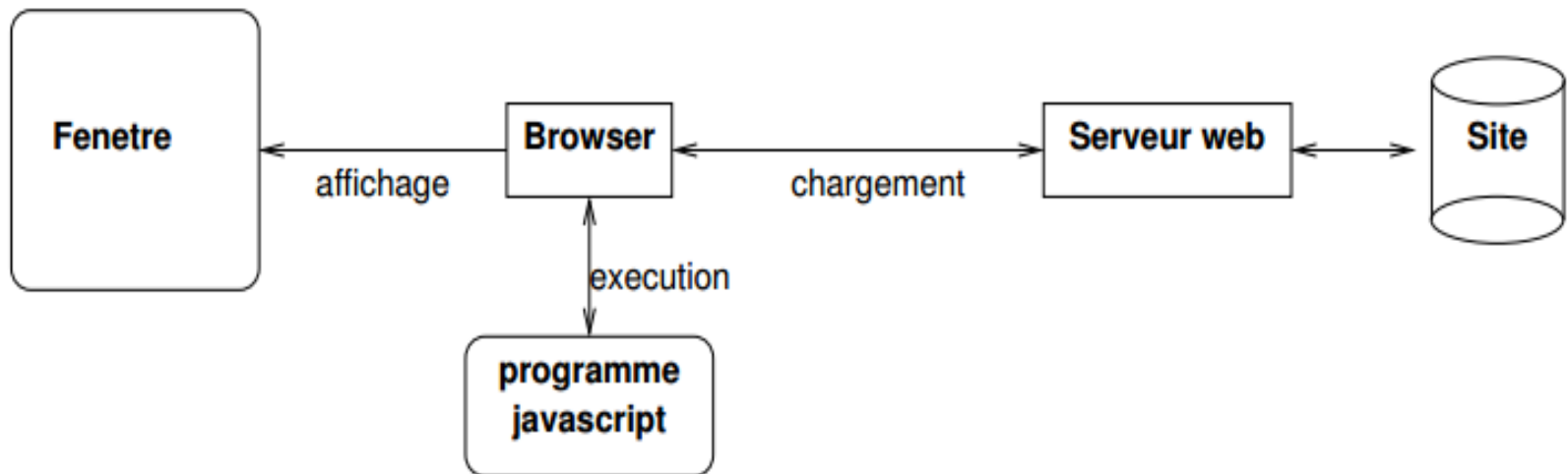
```
<script src="URL du module externe">  
.....  
</script>
```

HTML et JAVASCRIPT

- Pourquoi HTML + CSS au lieu d'un langage d'interface graphique ?
 - Java, XForms/XPath et d'autres langages qui seraient beaucoup mieux adaptés pour construire des interfaces homme-machine graphiques ne sont jamais vraiment arrivés à s'imposer sur le Web.
 - W3C n'a jamais réussi à créer ou standardiser un langage d'interface utilisateur (XAL, XAML, XUL, Konfabulator, JavaFX. . .). HTML est peut-être le pire des langages possibles pour une GUI, mais au moins c'est un langage neutre.
 - Avec JavaScript et autre technos les choses vont mieux

Navigateurs et JAVASCRIPT

- Le navigateur (Firefox) charge une page Web depuis le serveur (Apache). En plus du HTML, la page contient des programmes en JavaScript que le navigateur exécute.
- JavaScript peut être utilisé à l'extérieur d'un navigateur, notamment node.js (voir nodejs.org)



Insertion directe



The screenshot shows the JS.do Online JavaScript Editor interface. At the top, there's a browser-like address bar with a back arrow, a shield icon, and the text 'js.do'. Below this, the title 'JS.do Online JavaScript Editor' is displayed in orange and grey, followed by the tagline 'Edit your code online. Simple, light and fast!'. There are two input fields: 'Code address: http://js.do/code/' and 'Description:'. Below these are three buttons: 'Run code' (with a play icon), 'Save' (with a checkmark icon), and 'Add framework' (with a dropdown arrow). The main area is split into two panes. The left pane is a code editor with a light blue background and a red border around the script section. It contains the following code:

```
1 <html>
2 <head><title>First JavaScript Page</title></head>
3 <body>
4 <h1>First JavaScript Page</h1>
5 <script type="text/javascript">
6   document.write("<hr>");
7   document.write("Hello World Wide Web");
8   document.write("<hr>");
9 </script>
10 </body>
11 </html>
12
13
```

The right pane is a preview window with a white background. It displays the rendered HTML: 'First JavaScript Page' in a large, bold, black serif font, followed by a horizontal line, then 'Hello World Wide Web' in a smaller, black serif font, followed by another horizontal line.

Insertion comme un module

```
1 <html>
2 <head><title>First JavaScript Page</title></head>
3 <body>
4 <h1>First JavaScript Page</h1>
5 <script type="text/javascript" src= "js/hello.js">
6 </script>
7 </body>
8
9 </html>
10
```

```
1 document.write("<hr>");
2 document.write("Hello World Wide Web");
3 document.write("<hr>");
4
```

First JavaScript Page

Hello World Wide Web

La structure d'un script en JavaScript

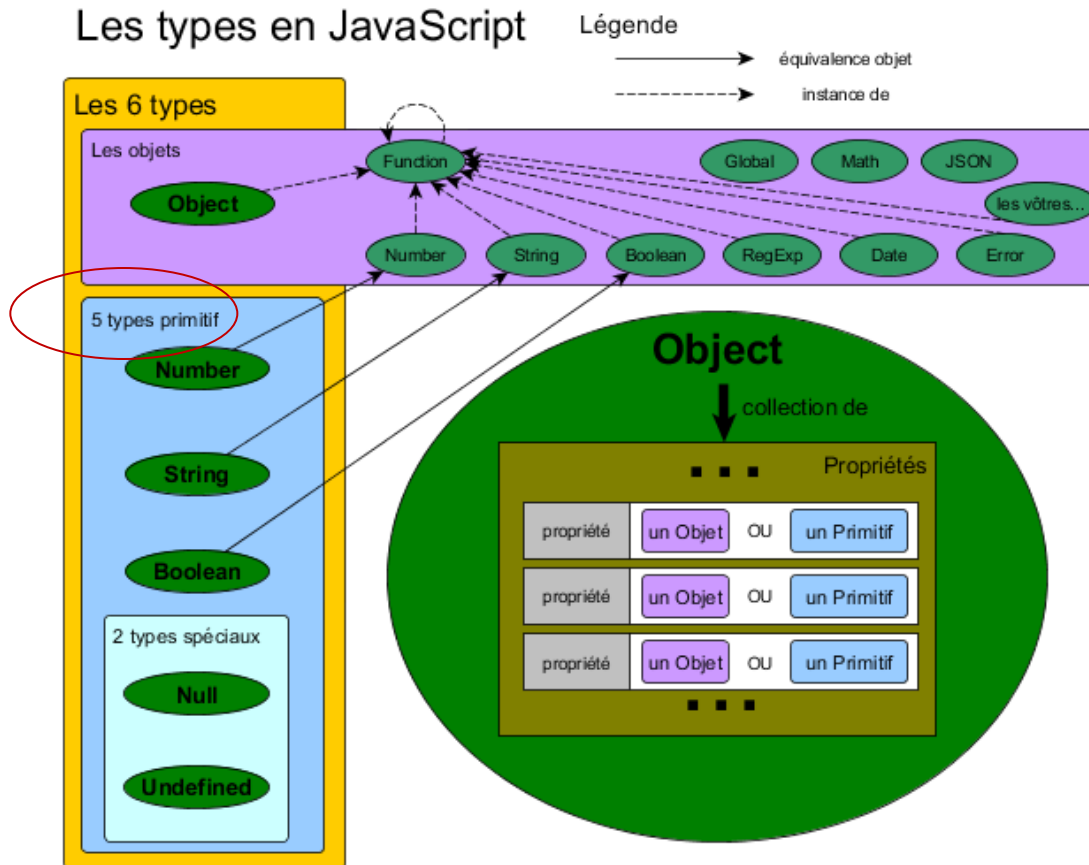
- La syntaxe du langage **JavaScript** s'appuie sur le modèle de Java et C
- **Règles générales**
 - L'insertion des espaces peut s'effectuer n'importe où dans le script
 - Chaque commande doit être terminée par un point-virgule (;).
 - Un nombre à virgule est séparé par un point (.) et non par une virgule
 - Sensible à la casse
 - Le langage **JavaScript** y est **sensible à la casse**
 - Il existe deux méthodes permettant d'intégrer des commentaires à vos scripts.
 - Placer un double slash (//) devant le texte
 - Encadrer le texte par un slash suivi d'une étoile (/*) et la même séquence inversée (*//)

Les mots réservés

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

Les types de base

Le JavaScript n'a que 6 types : **Object**, **Number**, **String**, **Boolean**, **Null** et **Undefined**.



Les types de base

- Le JavaScript n'a que 6 types : **Object, Number, String, Boolean, Null et Undefined.**
 - À part le type **Object** : les 5 autres types sont dit des types primitifs.
 - Les types **Null** et **Undefined** sont des types spéciaux.
 - La **Function** n'est qu'un type **Object** qui peut être exécuté et instancié avec « new ».
 - **Array, Date** et **RegExp** sont des types Object instanciables (Function).
 - **Math** et **JSON** sont simplement un type Object (ne s'instancie pas avec « new »).
 - Bien que **Number, String** et **Boolean** soient des types primitifs, il existe un équivalent de type Object instanciable (**Function**) pour chacun d'eux (à ne pas confondre).


Témoignage

JavaScript: Types

Posted on [August 27, 2012](#) by [Dmitry Baranovskiy](#)

I like JavaScript. It is a language that is both powerful and flexible, but only if you know how to use it. Once you have mastered the JavaScript language you can build almost anything, and you can do so quickly and interactively.

« J'aime le JavaScript. C'est un langage alliant puissance et flexibilité, mais à condition de bien savoir l'utiliser. Une fois que vous maîtrisez le langage JavaScript, vous pouvez construire pratiquement n'importe quoi, et cela vraiment rapidement et de manière interactive. »

 About the author



I am Sydney based web developer, interested in HTML, CSS, JavaScript, XSLT and SVG. Currently I am working at [Adobe](#) as Senior Computer Scientist.

Éléments du langage et règles d'écriture

- **Déclaration et affectation**

- Le mot-clé **var** permet de déclarer une ou plusieurs variables.
- Après la déclaration de la variable, il est possible de lui affecter une valeur par l'intermédiaire du signe d'égalité (=).
- Si une valeur est affectée à une variable sans que cette dernière ne soit déclarée, alors Javascript la déclare automatiquement.

```
//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;
```

```
//Déclaration et affectation de  
caractere
```

```
var caractere = ["a", "b", "c"];
```

Éléments du langage et règles d'écriture

- **Un objet est défini comme une fonction**

- Un **objet** JavaScript est défini sous la forme syntaxique d'une fonction, qui correspond à une première instance et que l'on clone pour créer d'autres instances.
 - En outre cette structure est dynamique, on peut lui ajouter durant l'exécution des **méthodes** (en fait des fonctions internes) et des attributs.
 - L'objet se crée en définissant un **constructeur**, on peut l'assigner à un identificateur pour avoir une instance. L'opérateur **this** définit une variable comme attribut et une fonction comme méthode.

Éléments du langage et règles d'écriture

- Création d'un objet littéral

```
1 <script>
2 monfruit = {
3   couleur:"vert",
4   prix:4,
5   pomme: {
6       couleur:"vert",
7       prix:12
8   },
9   orange :{
10      citron : {
11          couleur:"jaune",
12          prix: 13
13      },
14      couleur:"orange",
15      prix :14
16  }
17 };
18 document.write(monfruit.orange.citron.prix);
19 document.write('<hr>');
20 document.write(monfruit.orange.prix);
21 document.write('<hr>' +typeof monfruit);
22 </script>
```

JavaScript

13

14

object

Éléments du langage et règles d'écriture

- Un objet est défini comme une fonction

▶ Run code

✓ Save update

🔄 Save as a new js

Add framework

```
1 <script>
2 function Fruit(prix,couleur) {
3     this.couleur=couleur;
4     this.prix=prix;
5 }
6
7 monfruit = new Fruit(12, 'jaune');
8 document.write(monfruit.couleur);
9 document.write('<hr>');
10 document.write(monfruit.prix);
11
12 tonfruit={
13     orange:monfruit,
14     pomme: {
15         couleur: 'vert',
16         prix:13
17     }
18 }
19 document.write('<hr>');
20 document.write(tonfruit.orange.prix);
21 document.write('<hr>' +typeof monfruit);
22 </script>
```

JavaScript

jaune

12

12

object

function fruit(v) {
 var couleur = "rouge";
 this.prix = v;
}

Éléments du langage et règles d'écriture

- Un objet peut être redéfini dynamiquement avec la propriété *prototype*

▶ Run code

✓ Save update

📄 Save as a new js

Add framework ▼

```
1 <script>
2 function Fruit(prix, couleur) {
3     this.prix=prix;
4     this.couleur=couleur;
5 }
6
7 monfruit = new Fruit(10, 'orange');
8 monfruit.genre='Orange';
9 document.write('<hr>'+monfruit.genre+'<hr>');
10 Fruit.prototype.origine= 'Europe';
11 tonfruit= new Fruit(20, 'vert');
12 tonfruit.origine='Afrique';
13 document.write(tonfruit.origine);
14 document.write('<hr>'+tonfruit.genre+'<hr>');
15 </script>
```

JavaScript

Orange

Afrique

undefined

Éléments du langage et règles d'écriture

- Un objet peut contenir d'autres objets assignés comme valeurs d'attributs

```
1 <script>
2 function Fruit(prix, couleur) {
3   this.prix=prix;
4   this.couleur=couleur;
5 }
6 function orange(p)
7 {
8   this.prix = p;
9 }
10 monfruit = new Fruit(100, 'inconnu');
11 x = new orange(25);
12 monfruit.orange = x;
13 document.write("prix orange: ");
14 document.write(monfruit.orange.prix);
15 </script>
```

JavaScript

prix orange: 25

```
monfruit = {
  prix:100,
  couleur: 'inconnue',
  orange : {
    prix : 25
  }
};
```

Éléments du langage et règles d'écriture

- On ajoute des méthodes à un objet de façon statique ou dynamique

```
▶ Run code  ✓ Save update  in  Save as a new js  Add framework  JavaScript 110
1 <script>
2 function outer()
3 {
4   x = 10;
5   this.inner = function (y)
6   {
7     return y * 2 + x;
8   }
9 }
10 var o = new outer();
11 document.write(o.inner(50));
12 </script>
13
```

```
var a = {
  x : 10,
  inner : function(y) {
    return y * 2 + this.x;
  }
}
```

```
JavaScript 110
1 <script>
2 function outer()
3 {
4   x = 10;
5   this.inner = function (y)
6   {
7     return y * 2 + x;
8   }
9 }
10 var o = new outer();
11 document.write(o.inner(50));
12
13 outer.x2 = 10;
14 outer.inner2 = function(y)
15 {
16   return y * 2 + x + this.x2;
17 }
18 document.write('<hr>' + outer.inner2(50));
19
20 </script>
```

Éléments du langage et règles d'écriture

- JavaScript est un langage orienté objet à base d'instances et non à base de classes (POO orientée prototype)
 - Une instance peut être construite à partir d'une fonction constructeur
 - **this** est initialisé par le constructeur pour faire référence à l'instance courante
 - L'héritage dans la programmation objet orientée prototype se fait à base de clonage
 - L'instance est composée de slots de données (les fonctions sont des données !)

Éléments du langage et règles d'écriture

```
<script type='text/javascript'>
function Animal(nb_pattes) {
  this.pattes = nb_pattes ;
} ;
var garfield = new Animal(4) ;
garfield.aime = 'les lasagnes' ;
var odie = new Animal(4) ;
odie.aime = 'les os' ;
window.alert("Garfield a " + garfield.pattes + " pattes et aime " + garfield.aime) ;
window.alert("Odie a " + odie.pattes + " pattes et aime " + odie.aime) ;
</script>
```

Garfield a 4 pattes et aime les lasagnes

Odie a 4 pattes et aime les os

Éléments du langage et règles d'écriture

```
<script type='text/javascript'>
function Animal(nb_pattes) {
  this.pattes = nb_pattes ;
  this.dormir = function() { return "ZZZzzzz..." ; }
} ;
Chat.prototype = new Animal(4) ;
function Chat() { } ;

Chien.prototype = new Animal(4) ;
function Chien() { } ;

var garfield = new Chat() ;
var odie = new Chien() ;

window.alert("Garfield a " + garfield.pattes) ;

window.alert("Garfield, au lit ! " + garfield.dormir()) ;

window.alert("Odie a " + odie.pattes) ;
</script>
```

Garfield a 4 pattes

ZZZzzzz...

Odie a 4 pattes

Éléments du langage et règles d'écriture

```
<script type='text/javascript'>
function Animal(nb_pattes) {
  this.pattes = nb_pattes ;
  this.dormir = function() { return "ZZZzzz..." ; }
} ;
Chat.prototype = new Animal(4) ;
function Chat() { } ;

Chien.prototype = new Animal(4) ;
function Chien() {
  this.dormir = function() { return "... " ; }
} ;

var garfield = new Chat() ;
var odie = new Chien() ;
window.alert("Garfield, au lit ! " + garfield.dormir()) ;

window.alert("Odie, au lit ! " + odie.dormir()) ;
</script>
```

ZZZzzz...

...

Éléments du langage et règles d'écriture

```
<script type='text/javascript'>
function Animal(param) {
    this.pattes = param.pattes || 4 ;
    this.aime   = param.aime   || '?' ;
} ;
Chat.prototype = new Animal({pattes : 4,
                             aime   : 'le poisson' }) ;

function Chat() { } ;
Chien.prototype = new Animal({pattes : 4,
                              aime   : 'la viande' }) ;

function Chien() { } ;
var garfield = new Chat() ;
var odie     = new Chien() ;

window.alert("Garfield a " + garfield.pattes + " pattes") ;
window.alert("Garfield aime " + garfield.aime) ;
window.alert("Odie a " + odie.pattes + " pattes") ;
window.alert("Odie aime " + odie.aime) ;
</script>
```

Garfield a 4 pattes

Garfield aime le poisson

Odie a 4 pattes

Odie aime la viande

Éléments du langage et règles d'écriture

- Objet Array: Déclaration

```
var tab1 = new Array(taille) ;  
var tab2 = new Array(1, "a", 9, ...) ;  
                index → 0  1  2  ...
```

- Utilisation

```
window.alert(tab2[0]) ; // 1  
tab2[2] = 6 // 6 remplace 9
```

- Accroissement automatique de la taille

```
var tab1 = new Array(2) ;  
tab1[200] = 5 ;
```

- Parcours

```
for (i in tab2)  
    window.alert("tab2[" + i + "] = " + tab2[i]) ;
```

Éléments du langage et règles d'écriture

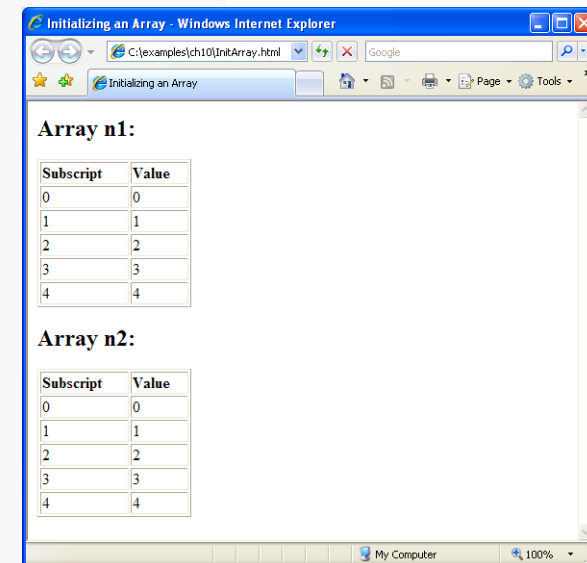
Objet Array

```
14    <script type = "text/javascript">
15    <!--
16    // create (declare) two new arrays
17    var n1 = new Array( 5 ); // allocate five-element Array
18    var n2 = new Array(); // allocate empty Array
19
20    // assign values to each element of Array n1
21    for ( var i = 0; i < n1.length; ++i )
22        n1[ i ] = i;
23
24    // create and initialize five elements in Array n2
25    for ( i = 0; i < 5; ++i )
26        n2[ i ] = i;
27
28    outputArray( "Array n1:", n1 );
29    outputArray( "Array n2:", n2 );
```

Éléments du langage et règles d'écriture

Objet Array

```
31 // output the heading followed by a two-column table
32 // containing subscripts and elements of "theArray"
33 function outputArray( heading, theArray )
34 {
35     document.writeln( "<h2>" + heading + "</h2>" );
36     document.writeln( "<table border = \"1\">" );
37     document.writeln( "<thead><th>Subscript</th>" +
38         "<th>value</th></thead><tbody>" );
39
40     // output the subscript and value of each array element
41     for ( var i = 0; i < theArray.length; i++ )
42         document.writeln( "<tr><td>" + i + "</td><td>" +
43             theArray[ i ] + "</td></tr>" );
44
45     document.writeln( "</tbody></table>" );
46 } // end function outputArray
47 // -->
48 </script>
49 </head><body></body>
50 </html>
```



```
18 var colors = new Array( "cyan", "magenta", "yellow", "black" );
19 var integers1 = [ 2, 4, 6, 8 ];
20 var integers2 = [ 2, , 8 ];
```

Éléments du langage et règles d'écriture

Objet Array

```
7 <html>
8   <head>
9     <title>Linear Search of an Array</title>
10    <script type = "text/javascript">
11      <!--
12      var a = new Array( 100 ); // create an Array
13
14      // fill Array with even integer values from 0 to 198
15      for ( var i = 0; i < a.length; ++i )
16        a[ i ] = 2 * i;
17
18      // function called when "Search" button is pressed
19      function buttonPressed()
20      {
21        // get the input text field
22        var inputVal = document.getElementById( "inputVal" );
23
24        // get the result text field
25        var result = document.getElementById( "result" );
26
27        // get the search key from the input text field
28        var searchKey = inputVal.value;
```

Éléments du langage et règles d'écriture

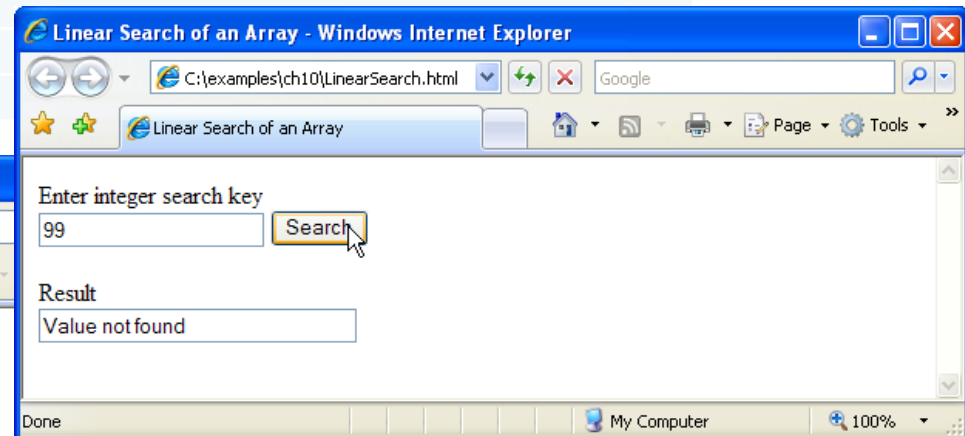
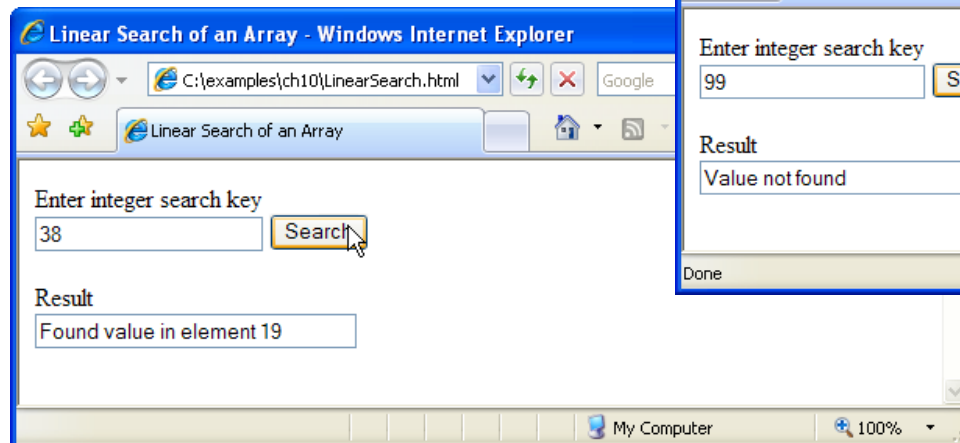
```
33  var element = linearSearch( a, parseInt( searchKey ) );
34
35  if ( element != -1 )
36      result.value = "Found value in element " + element;
37  else
38      result.value = "Value not found";
39  } // end function buttonPressed
40
41  // Search "theArray" for the specified "key" value
42  function linearSearch( theArray, key )
43  {
44      // iterates through each element of the array in order
45      for ( var n = 0; n < theArray.length; ++n )
46          if ( theArray[ n ] == key )
47              return n;
48
49      return -1;
50  } // end function linearSearch
51  // -->
52  </script>
53  </head>
```

Objet Array

Éléments du langage et règles d'écriture

Objet Array

```
55 <body>
56   <form action = "">
57     <p>Enter integer search key<br />
58     <input id = "inputVal" type = "text" />
59     <input type = "button" value = "Search" onclick= "buttonPressed()" /><br />
60
61     <p>Result<br />
62     <input id = "result" type = "text" size = "30" />
63   </form>
64 </body>
65 </html>
```

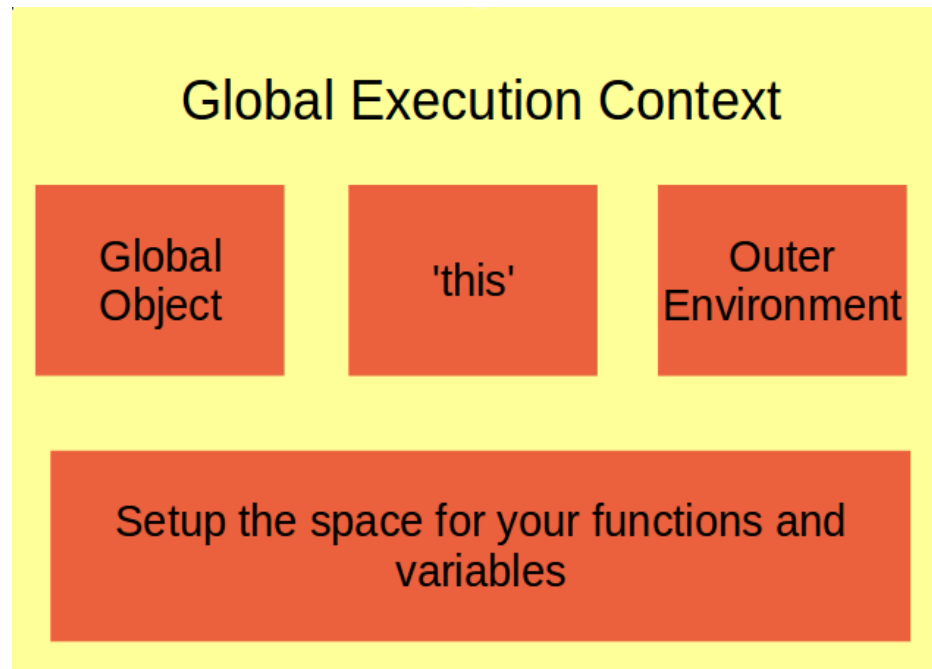


Éléments du langage et règles d'écriture

- Objet Array
 - Propriété :
 - *length* : retourne le nombre d'éléments du tableau;
 - Méthodes :
 - *concat()* : permet de concaténer 2 tableaux;
 - *join()* : converti un tableau en chaîne de caractères;
 - *reverse()* : inverse le classement des éléments du tableau;
 - *slice()* : retourne une section du tableau;
 - *sort()* : permet le classement des éléments du tableau

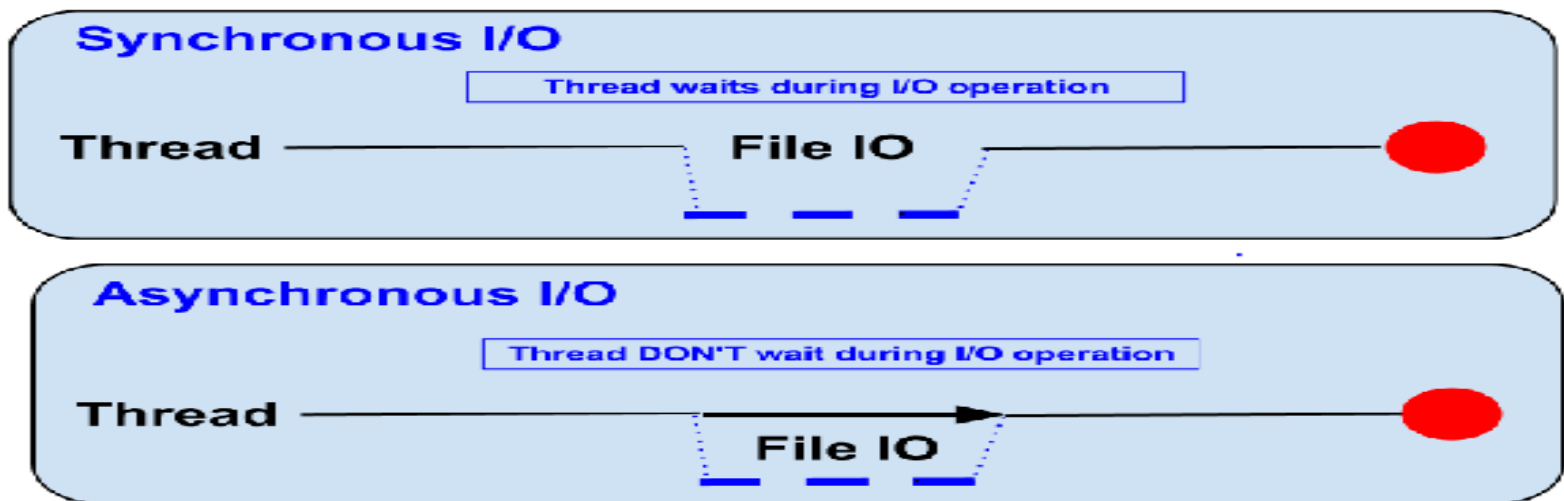
Éléments du langage et règles d'écriture

- Contexte d'exécution



Éléments du langage et règles d'écriture

- Historiquement et par conception, JavaScript n'est exécuté que dans un seul thread (un seul « fil d'exécution » sur le processeur). Le modèle d'exécution est synchrone:
 - Si tu appelles une ressource (requête réseau, système de fichier, etc.), le thread JavaScript est suspendu le temps que la transaction se termine, par exemple le temps qu'un fichier soit complètement uploadé.



Éléments du langage et règles d'écriture

- L'opérateur `typeof`
 - Retourne le type de son opérande
 - Number, Boolean, Function, Object, Undefined, Null

```
var i = 1;  
typeof i; //retourne number  
var titre="Les raisins de la colère";  
typeof titre; //retourne string  
var jour = new Date();  
typeof jour; //retourne object  
var choix = true;      typeof choix; //retourne boolean  
var cas = null;        typeof cas; //retourne object  
typeof parseFloat; //retourne function  
typeof Math; //retourne object (IE 5.*, NS 6.*, NS 4.78, Opera 6.*, Opera 5.*  
typeof Math; //retourne function NS 3.*, Opera 3.*
```

Éléments du langage et règles d'écriture

- Le type d'une variable dépend de la valeur stockée dans cette variable. **Pas de déclaration de type.**
 - Exemple

```
var maVariable = ' toto ' ;  
maVariable =10;
```
- trois principaux types de valeurs
 - String
 - Number : $10^{-308} > \text{nombre} < 10^{308}$
 - Les nombres entiers
 - les nombres décimaux en virgule flottant
 - 3 valeurs spéciales :
 - » Positive Infinity ou +Infinity (valeur infini positive)
 - » Negative Infinity ou -Infinity (valeur infinie négative)
 - » Nan (Not a Number) habituellement générée comme résultat d'une opération mathématique incohérente
 - Boolean
 - deux valeurs littérales : true (vrai) et false (faux).

Éléments du langage et règles d'écriture

- Les opérations sur les chaînes
 - La concaténation
 - `Var chaine = « bonjour » + « FI3/FCD1 »;`
 - Déterminer la longueur d'une chaîne
 - `Var ch1 = « bonjour »;`
 - `Var longueur = ch1.length;`
 - Identifier le nième caractère d'une chaîne
 - `Var ch1 = « Rebonjour ! »;`
 - `Var carac = ch1.charAt(2);`
 - Extraction d'une partie de la chaîne
 - `Var dateDuJour = « 04/04/03 »`
 - `Var mois = dateDuJour.substring(3, 5);`
 - » 3: est l'indice du premier caractère de la sous-chaîne à extraire
 - » 5 : indice du dernier caractère à prendre en considération ; ce caractère ne fera pas partie de la sous-chaîne à extraire

Éléments du langage et règles d'écriture

- Les fonctions prédéfinies
 - **Number**
 - convertit l'objet spécifié en valeur numérique

```
var jour = new Date("December 17, 1995 03:24:00");//convertit la date en millisecondes  
alert (Number(jour));
```

819167040000

OK

- **String**
 - convertit l'objet spécifié en chaîne de caractères

```
var jour = new Date("December 17, 1995 03:24:00");//convertit la date en millisecondes  
alert (String(jour));
```

Sun Dec 17 1995 03:24:00 GMT+0100

OK

Éléments du langage et règles d'écriture

- **parseFloat**

- analyse une chaîne de caractères et retourne un nombre décimal.
- Si l'argument évalué n'est pas un nombre, renvoie *NaN* (Not a Number).

```
var numero="125";  
var nombre=parseFloat(numero); //retourne le nombre 125
```

- **parseInt**

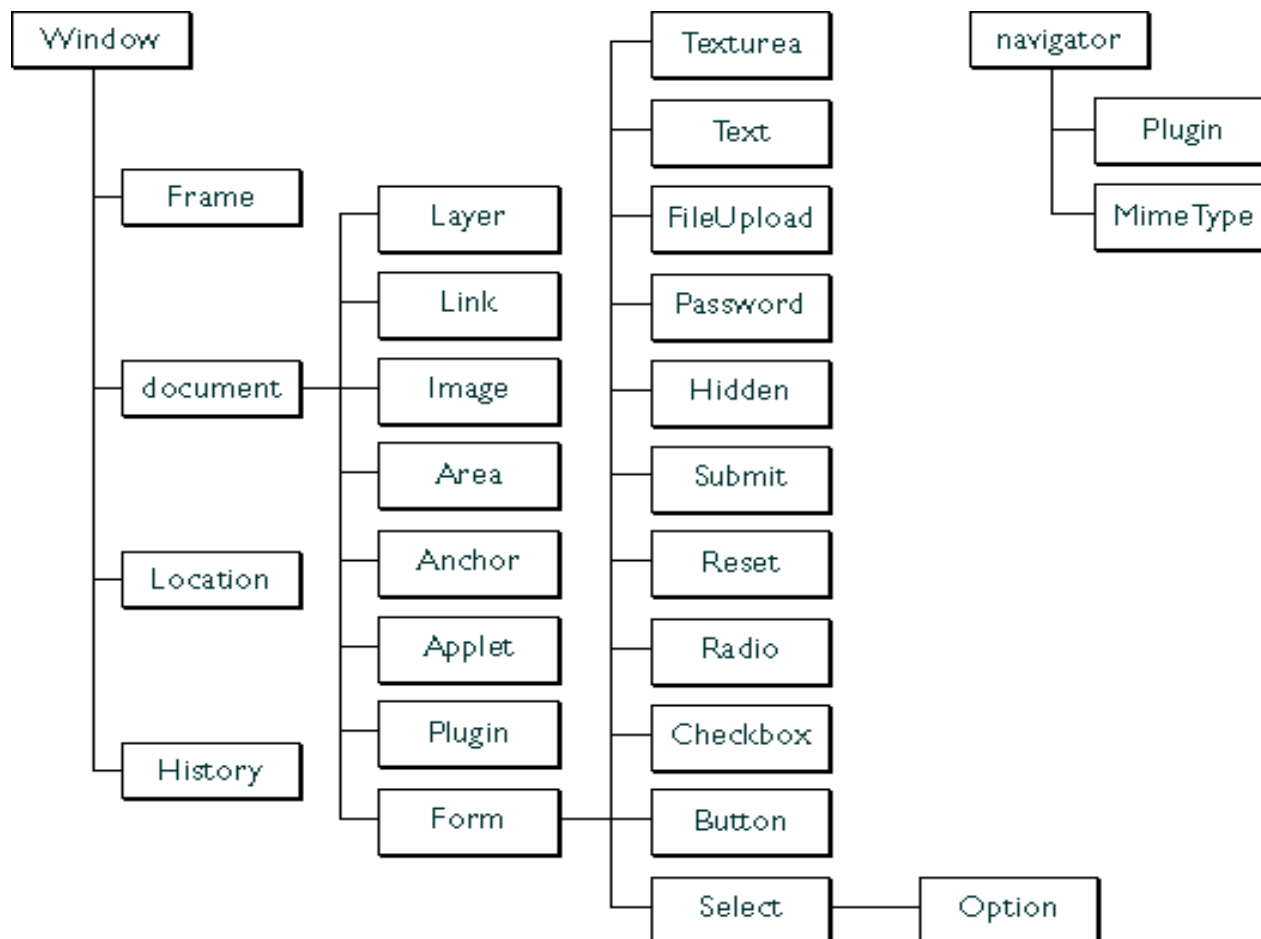
- analyse une chaîne de caractères et retourne un nombre entier de la base spécifiée.
- La base peut prendre les valeurs 16 (hexadécimal) 10 (décimal), 8 (octal), 2 (binaire).

```
var prix=30.75;  
var arrondi = parseInt(prix, 10); //retourne 30
```

Les objets du navigateur web

- L'objet le plus haut dans la hiérarchie est *window* qui correspond à la fenêtre même du navigateur.
- L'objet *document* fait référence au contenu de la fenêtre.
- *document* regroupe au sein de propriétés l'ensemble des éléments HTML présents sur la page. Pour atteindre ces différents éléments, nous utiliserons :
 - soit des méthodes propres à l'objet *document*, comme la méthode *getElementById()*, qui permet de trouver l'élément en fonction de son identifiant (ID);
 - soit des collections d'objets qui regroupent sous forme de tableaux Javascript tous les éléments de type déterminé.
 - Les objets javascript peuvent réagir à des "Evénements".

Les objets du navigateur web



Les objets du navigateur web

- L'objet *window* :
 - Propriétés
 - *closed* : indique que la fenêtre a été fermée;
 - *defaultStatus* : indique le message par défaut dans la barre de status;
 - *document* : retourne l'objet *document* de la fenêtre;
 - *frames* : retourne la collection de cadres dans la fenêtre;
 - *history* : retourne l'historique de la session de navigation;
 - *Location* : retourne l'adresse actuellement visitée;
 - *name* : indique le nom de la fenêtre;
 - *navigator* : retourne le navigateur utilisé;
 - *opener* : retourne l'objet *window* qui a créé la fenêtre en cours;
 - *parent* : retourne l'objet *window* immédiatement supérieur dans la hiérarchie;
 - *self* : retourne l'objet *window* correspondant à la fenêtre en cours;
 - *status* : indique le message affiché dans la barre de status;
 - *top* : retourne l'objet *window* le plus haut dans la hiérarchie.

`window.history.forward();`

`window.history.back();`

`var numberOfEntries = window.history.length;`

Les objets du navigateur web

- L'objet *window* :
 - Propriétés
 - *navigator* : retourne le navigateur utilisé;
 - *opener* : retourne l'objet *window* qui a créé la fenêtre en cours;
 - *parent* : retourne l'objet *window* immédiatement supérieur dans la hiérarchie;
 - *self* : retourne l'objet *window* correspondant à la fenêtre en cours;
 - *status* : indique le message affiché dans la barre de status;
 - *top* : retourne l'objet *window* le plus haut dans la hiérarchie.

Les objets du navigateur web

- L'objet *window* :
 - Méthodes
 - *blur()* : enlève le focus de la fenêtre;
 - *close()* : ferme la fenêtre;
 - *focus()* : place le focus sur la fenêtre;
 - *moveBy()* : déplace d'une distance;
 - *moveTo()* : déplace la fenêtre vers un point spécifié;
 - *open()* : ouvre une nouvelle fenêtre;
 - *print()* : imprime le contenu de la fenêtre;
 - *resizeBy()* : redimensionne d'un certain rapport;
 - *resizeTo()* : redimensionne la fenêtre;
 - *setTimeout()* : évalue une chaîne de caractère après un certain laps de temps.

Les objets du navigateur web

- L'objet *window*

```
<!DOCTYPE html>  
<html>  
<body>
```

```
<p>Click the button to open a new browser window.</p>  
<button onclick="myFunction()">Try it</button>
```

```
<script>  
  function myFunction() {  
    window.open("https://www.w3schools.com");  
  }  
</script>
```

```
</body>  
</html>
```

Les objets du navigateur web

- L'objet *document* :
 - propriétés
 - *applets* : retourne la collection d'applets java présente dans le document;
 - *cookie* : permet de stocker un cookie;
 - *domain* : indique le nom de domaine du serveur ayant apporté le document;
 - *forms* : retourne la collection de formulaires présents dans le document;
 - *images* : retourne la collection d'images présentes dans le document;
 - *links* : retourne la collection de liens présents dans le document;
 - Méthodes :
 - *close()* : ferme le document en écriture;
 - *open()* : ouvre le document en écriture;
 - *write()* : écrit dans le document;
 - *writeln()* : écrit dans le document et effectue un retour à la ligne

Les objets du navigateur web

```
<HTML>
<BODY>
  <FORM NAME="truc"></FORM>
  <FORM NAME="machin"></FORM>
  <FORM NAME="chouette"></FORM>
</BODY>
</HTML>
```

- Pour accéder au formulaire "machin" :

```
<SCRIPT LANGUAGE="javascript">
// notation la plus courante
  document.forms["machin"];

// ou bien : accéder au formulaire par son numéro
// ici 0=truc ; 1=machin ; 2=chouette
// (dans l'ordre de déclaration en HTML, commence à 0)

document.forms[1];
// ou encore :
  document.machin;
</SCRIPT>
```

Il est ensuite possible d'accéder à des propriétés ou exécuter des méthodes pour ce formulaire.

Exemple :

```
document.forms["machin"].submit();
document.forms["machin"].action;
```

L'objet *document*

Éléments du langage et règles d'écriture

- Les évènements
 - Javascript est dépendant des événements
 - se produisent lors d'actions diverses sur les objets d'un document HTML.
 - onLoad;
 - onClick
 - onMouseover
 - onMouseout
 - ...
 - Il est possible de baser l'exécution de fonctions sur des événements

Éléments du langage et règles d'écriture

- **Événement onLoad**

- Se produit lorsque une page web est chargée dans la fenêtre du navigateur
- Toute la page (y compris les images qu'elle contient si leur chargement est prévu) doit avoir été chargée pour qu'il ait lieu
- Cet événement peut être associé à une image seulement ; auquel cas, il se produit une fois son chargement terminé

```
<HTML>  
  <BODY onLoad="alert('page chargée');">  
    Exemple de l'événement onLoad  
  </BODY>  
</HTML>
```

Éléments du langage et règles d'écriture

- **Événement onClick**

- Se produit lorsque l'utilisateur clique sur un élément spécifique dans une page, comme un lien hypertexte, une image, un bouton, du texte, etc.
- Ces éléments sont capables de répondre séparément à cet événement
- Il peut également être déclenché lorsque l'utilisateur clique n'importe où sur la page s'il a été associé non pas à un élément spécifique, mais à l'élément body tout entier

```
<HTML>
  <BODY>
    <INPUT TYPE="Button" Value="cliquer ici" onClick="alert('Clic')">
  </BODY>
</HTML>
```

Éléments du langage et règles d'écriture

- **Événement onMouseover**
 - Analogue à onClick sauf qu'il suffit que l'utilisateur place le pointeur de sa souris sur l'un des éléments précités (lien hypertexte, image, bouton, texte, etc.) pour qu'il ait lieu
- **Événement onMouseout**
 - A l'inverse de onMouseover, cet événement se produit lorsque le pointeur de la souris quitte la zone de sélection d'un élément.

Éléments du langage et règles d'écriture

- Pour pouvoir manipuler un objet en javaScript, il doit posséder un nom
- Pour pouvoir distinguer les différents objets-éléments d'une page web, il suffit de leur donner un nom à travers de l'attribut NAME
 - <Table Name=« tableau1 »>...
 - <Table Name=« tableau2 »>...
 - <Form Name = « formulaire1 »>...
 - <Form Name =« formulaire2 »>...
 - <Textarea Name =« texte1»>...
- Dans le cas où l'objet serait unique alors pas besoin de nom pour désigner cet objet
 - Exemple : le cas de BODY(une seul BODY par document), DOCUMENT (un seul DOCUMENT par fenêtre)

Éléments du langage et règles d'écriture

- Pour adresser un objet, il ne suffit pas de donner son nom : il faut aussi préciser son « chemin d'accès » dans l'arborescence de la structure

```
<HTML>
<BODY onLoad="window.document.formulaire.zone.value='Bonjour';">
  <FORM name="formulaire">
    <INPUT NAME="zone" TYPE="text">
  </FORM>
</BODY></HTML>
```

- Si le nom de la fenêtre est omis, le navigateur utilisera par défaut la fenêtre courante
- Dans le cas de cadres (frames), il est pertinent de donner le nom de la fenêtre
- Il est possible aussi d'omettre window.document : l'adressage réussit puisqu'il n'y a qu'un seul objet « document » dans la fenêtre