

Université de Msila

1

Faculté des mathématiques et de l'informatique

Département d'informatique

3^{ième} année Licence ISIL

Rappels sur JavaScript

Présenté par : Meliouh.A

Intérêt

Exemple 1

```
<html>
  <head>
    <title>Page statique</title>
  </head>
  <body>
    <div>
      Nous sommes le 04/11/2018
    </div>
  </body>
</html>
```

Intérêt

Exemple 2

```
<html>
  <head>
    <title>Page statique</title>
  </head>
  <body>
    <script type = "text/javascript">
      date = new Date();
      document.writeln("Nous sommes le ", date);
    </script>
  </body>
</html>
```

- Le Javascript est un langage "de script" simplifié, intégrant une couche "objet" :
- Initialement élaboré par Netscape en association avec Sun Microsystems (1995: LiveScript)
- Standardisé par un comité spécialisé, l'ECMA (European Computer Manufacturers Association).
- Javascript permet de:
 - rendre **dynamique** un site internet développé en HTML :
 - Validation de formulaires, calculs, messages,
 - Modification de la page web,
 - Communication avec un serveur directement (AJAX)
 - De développer de véritables applications fonctionnant exclusivement dans le cadre d'Internet.
- Validation javascript: **jslint.com**

- ne peut pas lire ou écrire dans le système de fichier de la machine
- ne peut exécuter de programme externe
- pas de connexion autre que serveur web

- Le code JavaScript peut être **pénible** à mettre au point.
 - utilisez un débogueur ;
 - soignez la présentation (indentation du code) ;
 - ne jamais utiliser de variables globales (utiliser le mot clé var).

- **Dans l'entête de la page**

- Entre les balises `<head>` et `</head>`
- Code exécuté lors d'un événement utilisateur
- L'événement se trouve dans le corps du document.

```
<html>
<head>
<script type="text/javascript">
    fonctionf () { alert('Au revoir'); }
</script>
</head>
<body onUnload="f();"> // fermeture de la page courante
</body>
</html>
```

- Dans le corps de la page HTML
 - Entre les balises <body> et </body>
 - Code exécuté lors du chargement de la page

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


- **Dans l'entête ou dans le fichier**

- Fichier en format texte
- Avantage : réutilisation du script dans plusieurs pages
- Inconvénient : requête supplémentaire vers le serveur

```
<html>
<head>
<script type="text/javascript" src="fichier.js"></script>
</head>
<body>
<input type="button" onclick="popup()" value="Clic">
</body>
</html>
```

insertion de code JS

10



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>test JS</title>
  <script type="text/javascript" src="script.js"></script>
</head>
<body>
  <script type="text/javascript">
    <!--
      function affiche_date(){
        var aujourd'hui = new Date();
        alert(aujourd'hui.toLocaleString());
      }
    // -->
  </script>
  <noscript><p>activez JS !</p></noscript>
  <div>
    <h1>Test JS</h1>
    <a href="javascript:affiche_date()">date</a>
    <a href="javascript:alert('hello')">hello</a>
    <form action="test.php">
      <div><input type="button" value="clie" onclick="pair()"></div>
    </form>
  </div>
</body>
</html>
```

1

2

3

4



```
function pair(){
  nb = prompt('entrez un nombre', '');
  (nb%2 == 0) ? alert('pair') : alert('impair');
}
```

script.js



1



entrez un nombre <input type="text" value="3"/> OK	impair OK
--	--------------

Le code peut être placé dans :

- (1) fichier externe .js
- (2) élément script
- (3) attribut contenant URL
- (4) attribut événementiel onclick, onchange, onselect,...

- Une **console** JavaScript est présente dans les outils de développement de Firefox ou Chrome (raccourci clavier **F12**)
- **Exemple :**
 - `x; // Variable non définie`
 - `var x; x; // Variable définie, non initialisée`
 - `var x = 1; x; // Variable définie et initialisée`
- `// Affiche une variable dans la console`
`console.log(x);`
- `// Affiche un dialogue d'alerte`
`alert("Coucou !")`

- définition classique :
 - caractère suivi de caractères ou chiffres.
- on peut utiliser le caractère **souligné** (_)
- on peut également utiliser **\$** (cf frameworks Ajax)

- Les variables sont des objets
- elles peuvent être déclarées à tout moment
- on peut utiliser le mot-clé **var** pour les déclarer
- Toute valeur a un type en JavaScript. Mais ce type n'est pas spécifié lors de la déclaration d'une variable.

```
var x = 1;  
typeof x; // → number  
var y = "Vous êtes bien réveillés ?";  
typeof y; // → string
```

- Les types sont **dynamiques**

```
var x;           // Now x is undefined  
var x = 5;      // Now x is a Number  
var x = "Yahia"; // Now x is a String
```

- Il y a 6 types en JavaScript :
 - 5 types **primitifs** (**Boolean, Number, String, Null, Undefined**)
 - les objets **Object**.
- **Undefined** est le type des variables qui n'ont pas de valeur (e.g. une variable non initialisée).

```
var x;  
typeof x;  
// → undefined
```

- **Portée des variables**
 - **Locale** (uniquement dans le script ou la fonction)
 - **var** vloc = 0 ;
 - **Globale** (en tout point du document)
 - vglob = 0 ;

- **affectation**

`+=, -=, *=, /=, %=, &=, |=, <<=, >>=`

- **comparaison**

`==, !=, <, <=, >, >=, ===, !==`

- **arithmétique**

`%, ++, --`

- **logique** (évaluation paresseuse)

`&&, ||, !`

- **bit**

`&, |, ^ (XOR), <<, >>, >>>`

- Javascript possède les structures de contrôle du langage C :
 - if else
 - for, while, do.. while
 - break, continue
 - switch
- plus le **for ... in** (foreach) pour les tableaux
- et les exceptions throw, try, catch

- Déclaration comme en Java, C++

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

- Les variables **peuvent stocker** des fonctions !
Le code ci-dessus est équivalent à:

```
var square = function (x) { return x * x;};
```

- Les fonctions sont des **objets de «première classe»** : elles peuvent être manipulées et échangées comme tous les autres objets JavaScript.

```
function square(x) { return x * x; };  
var varfunc = square; // Affectation de la variable  
// Exécution de la fonction avec l'opérateur ()  
varfunc(2); // → 4
```

- Une fonction renvoie toujours quelque chose. Par défaut, la fonction renvoie **undefined**.

```
function mauvaiscarre(x) {  
  var y = x * x;  
    // Le développeur a oublié d'écrire return y;  
}  
mauvaiscarre(2);  
    // → undefined
```

- Une fonction peut prendre en argument une fonction

```
function boum() {alert('Boum!');}  
setTimeout(boum,2000);  
// setTimeout exécute la fonction dans la variable boum après 2s
```

- **Les fonctions principales**

- **eval(string)** : évalue le code Javascript
- **Number(var)** : convertir en nombre
- **String(var)** : convertir en chaîne
- **int parseInt(string[,radix])** : convertir en entier en fonction de la base
- **float parseFloat(string)** : convertir en réel
- **encodeURIComponent(uri)**
- **decodeURI(uri)**

Exemple avec des fonctions principales

```
1 print(parseInt('ff',16)); //255
2
3 var str=' 256';
4 var x = 1 + str;
5 print(x); // '1 256'
6
7 var x =1 + Number(str);
8 print(x); // 257
9
10 print(eval('2 + 2 * 8 - 3')); //15
11
12 //JSON
13 var person=eval("({ prenom : 'Sid', nom : 'Meyer' })");
14 print(person); // object Object
15 print(person.prenom+' '+person.nom); // Sid Meyer
16
17 var uri=encodeURIComponent('http ://www.site.fr/x\n=1?value=2<3');
18 print(uri); // http ://www.site.fr/x%0a=1?value=2%3c3
19
20
```

- Les **timers** permettent d'exécuter une action après un certain délai, exécuter une action toutes les x secondes,...
- Pour activer un timer est une opération simple.
- **setTimeout() et clearTimeout()**

```
window.setTimeout("mafonction()",1000);
```

- Ce code va appeler la fonction `mafonction()` après une seconde (le temps est exprimé ici en millièmes de secondes).
- On peut arrêter un timer avant son timeout, en utilisant la fonction **clearTimeout()**. Mais il faut alors "nommer" notre timeout, comme ceci :

```
montimer=window.setTimeout("mafonction()",5000);
```

- Ce qui permettra de le stopper comme ceci :

```
window.clearTimeout(montimer);
```

- Les **timers** permettent d'exécuter une action après un certain délai, exécuter une action toutes les x secondes,...
- Pour activer un timer est une opération simple.
- **setInterval() et clearInterval()**
- La logique est la même, sauf qu'ici on appelle une fonction à intervalles réguliers.

```
window.setInterval("mafonction()",1000);
```

- Ce code appellera la fonction `mafonction()` toutes les secondes jusqu'à ce que la page soit fermée ou que le timer soit stoppé par la fonction **clearInterval()**. Mais là aussi, il est nécessaire de nommer le timer.

```
montimer=window.setInterval("mafonction()",5000);  
window.clearInterval(montimer);
```

Méthodes des objets

- **Question** : Comment définir une méthode d'un objet ?
- **Réponse** : Il suffit d'assigner une valeur de type **function** à un attribut.

```
var point = {coord1:1, coord2:3, size:"normal"};
```

```
point.print = function() {
```

```
    alert("Mes coordonnées sont " + this.coord1 + "," +  
    this.coord2
```

```
);
```

```
}
```

```
point.print(); // → "Mes coordonnées sont 1,3"
```

Déterminer le type d'un objet

- l'instruction **typeof**: permet de déterminer le type d'une variable ou d'un identificateur.

Exemple typeof

```
1 var b=true ;
2 var s= 'coucou' ;
3 var i=10 ;
4 var k ;
5 var tab= [1, 2, 3] ;
6 var p=null ;
7 var regex=/\w+/ ;
8 function f() { }
9
10 print(typeof b) ; // boolean
11 print(typeof s) ; // string
12 print(typeof i) ; // number
13 print(typeof f) ; // function
14 print(typeof k) ; // undefined
15 print(typeof tab) ; // object
16 print(typeof dummy) ; // undefined
17 print(typeof p) ; // object
18 print(typeof regex) ; // function
```


Création d'objets

- On peut créer les objets de 3 manières différentes
 1. utilisation de la classe **Object** puis associer des attributs et fonctions
 2. création d'un **constructeur** puis d'une instance
 3. création d'une instance grâce au format **JSON**

Exemple avec Object

```
1 // create object
2 person=new Object () ;
3
4 // add properties
5 person.first_name='Joe' ;
6 person.last_name='Dalton' ;
7
8 // add method
9 person.display=function () {
10     print(this.first_name+' '+this.last_name) ;
11 }
12
13 // use method
14 person.display () ;
```

Exemple avec Object (fonction prédéfinie)

```
1 // create object
2 person=new Object () ;
3
4 // add properties
5 person.first_name='Joe' ;
6 person.last_name='Dalton' ;
7
8 // add predefined function
9 function display() {
10     print(this.first_name+' '+this.last_name) ;
11 }
12
13 person.display=display ;
14
15 person.display() ;
```

Exemple avec constructeur

```
1 function Person(first, last) {
2   this.first_name=first;
3   this.last_name=last;
4   this.display=function() {
5     print(this.first_name+' '+this.last_name);
6   }
7 }
8
9 person=new Person('Joe', 'Dalton');
10
11 person.display();
```

- **Utilisation du format JSON (1/3)**

- Définition (*JaSON - JavaScript Object Notation*)

1. il s'agit d'un format de données (texte) qui permet la sérialisation des objets
2. facilité de lecture
3. mise en œuvre simplifiée (par rapport à XML)
4. reconnu nativement par Javascript

Utilisation du format JSON (2/3)

Exemple instance JSON

```
1 var joe={
2   "first_name" : "Joe",
3   "last_name"  : "Dalton",
4   "display"    : function() {
5     print(this.first_name+' '+this.last_name);
6   },
7   brothers    : [
8     { "name" : "Jack", age : 30 },
9     { "name" : "Averell", age : 33 },
10    { "name" : "William", age : 31 }
11  ]
12 }
13
14 joe.display();
15
16 for (var i=0;i<joe.brothers.length;++i) {
17   print(joe.brothers[i].name);
18 }
```

- **Utilisation du format JSON (3/3)**
- **IMPORTANT**
- Lorsque l'on échange des données (serveur - client), il n'est pas possible de définir des fonctions (comme dans l'exemple précédent).

- **Objet en tant que tableau associatif**
- un objet javascript est en fait un tableau associatif d'attributs et méthodes identifiés par leurs noms.

objet et tableau associatif

```
1 person=new Object();  
2 person['first_name']='Joe';  
3 person['last_name']='Dalton';  
4 person['display']=function () {  
5     print(this.first_name+' '+this.last_name);  
6 }  
7  
8 person.display();
```


Accès aux propriétés de l' objet

- `person1 = new person (" Mohamed ", " Yahia ", 20)`
- `Var nom= person1.firstName;` //notation pointée
- `Var prenom = person1 [" lastName "];` //tableau associatif
- //toutes les propriétés:
- **for (var i in person1){**
 `alert(" Attribut : " + i + " , valeur: "+ person1 [i]);`
}

Objets Dynamiques

- **Lire**: `var nom= person1.firstName;`
- **Modifier**: `person1.firstName = Amine;`
- **Ajouter**: //ajout a l'objet (unique) person1
`person1.taille=1,75;`
- **Supprimer**: **delete** `person1.age;`
- **Ajouter à toute les instances produites par un constructeur**
`Person.prototype.taille=1,70;`

Objet	Description
Array	Permet de manipuler les tableaux.
String	Permet de manipuler les chaînes de caractères.
Math	Fournit des méthodes pour traiter les fonctions mathématiques usuelles, telles que log, exp, aussi bien que les fonctions trigonométriques.
Date	Permet de travailler avec la date courante; fournit également des méthodes pour faire des opérations sur les dates, les heures, les minutes et les secondes.
Boolean	Permet de manipuler les valeurs logiques.
Number	Permet de manipuler les valeurs numériques et les constantes.
Function	Permet d'emballer un code dans une fonction.

- l'utilisateur peut créer des pseudo-classes avec des propriétés et des méthodes.
- de pseudo-classes, car en JavaScript, il n'y a pas de notion d'héritage, ni de polymorphisme.
- Il existe cependant des pseudo-classes prédéfinies qui appartiennent à deux catégories :
 - les pseudo-classes usuelles
 - les pseudo-classes de fenêtrage

- Il y en a quatre : String, Date, Math, Array

<code>substring(debut, fin)</code>	extraction d'une sous chaîne	debut est l'index du caractère de départ (le premier caractère a l'index 0) et fin est l'index du caractère suivant le dernier caractère à prendre en compte
<code>indexOf("chaîne")</code>	recherche d'une chaîne dans une chaîne	renvoie l'index du premier caractère
<code>charAt(index)</code>	extraction d'un caractère	le caractère est défini par son index

- pseudo-classe Date : Cette pseudo-classe permet de manipuler des dates ; elle ne possède que des méthodes (pas de propriété) :

<code>getDate()</code>	renvoie le jour du mois		<code>setDate()</code>	fixe le jour du mois
<code>getDay()</code>	renvoie le jour de la semaine (0=dimanche)			
<code>getHours()</code>	renvoie l'heure		<code>setHours()</code>	fixe l'heure
<code>getMinutes()</code>	renvoie les minutes		<code>setMinutes()</code>	fixe les minutes
<code>getMonth()</code>	renvoie le mois (0=janvier)		<code>setMonth()</code>	fixe le mois
<code>getSeconds()</code>	renvoie les secondes		<code>setSeconds()</code>	fixe les secondes
<code>getTime()</code>	renvoie le nombre de millisecondes écoulées depuis le 1er janvier 1970		<code>setTime</code> (millisecondes)	fixe une date d'après le nombre de millisecondes écoulées depuis le 1er janvier 1970
<code>getFullYear()</code>	renvoie le nombre d'années depuis 1900		<code>setYear(années)</code>	fixe une année selon le nombre d'années écoulées depuis 1900

- pseudo-classe Array : elle permet de définir des tableaux à 1 indice
- `matrice=new Array(20) //définit 20 éléments numérotés de 0 à 19`
- `matrice[15]=4.56 //valeur d'un élément du tableau`
- Cette pseudo-classe ne possède qu'une seule propriété : ***length*** et trois méthodes :
- ***join()*** : concaténation de tous les éléments en une chaîne (séparateur à préciser, sinon virgule)
- ***sort()*** : tri avec critère (facultatif) à préciser
- ***reverse()*** : transposition

- Ces pseudo-classes sont relatives aux objets d'affichage manipulés par le navigateur
- les principales classes:
- La pseudo-classe **window** concerne une zone d'affichage
- La pseudo-classe **document** porte sur le ou les contenus d'une fenêtre
- La pseudo-classe **history** mémorise la succession des sites visités par leur URL
- La pseudo-classe **location**

La pseudo-classe **window**

propriétés	
defaultStatus	message par défaut de la barre d'état
frames	tableau des divisions de la fenêtre
length	nombre de divisions
name	nom de la fenêtre
parent	fenêtre parent
self	fenêtre active
status	message de la barre d'état
top	fenêtre de premier plan

méthodes	
alert()	création d'une fenêtre de dialogue
open()	ouverture d'une fenêtre
close()	fermeture d'une fenêtre
clear()	effacement du contenu d'une fenêtre
prompt()	affichage d'une boîte de dialogue avec un texte
setTimeout()	évaluation d'une expression ou appel d'une fonction après l'écoulement d'un délai fixé

Pseudo-classes document

propriétés

anchors	accès aux ancrs
forms	accès aux formulaires
lastModified	date de dernière modification
links	accès aux liens hypertextes
location	URL du document
title	titre du document en cours

méthodes

clear()	effacement du contenu du document
open()	ouverture du buffer du document
close()	fermeture du buffer du document
write()	écriture dans le document
writeln()	idem avec retour à la ligne

Pseudo-classes history

propriété	
length	nombre d'URL chargées

méthodes	
back()	précédent
forward()	suisant
go()	aller à

Pseudo-classes location

propriétés	
host	URL du document
href	URL complet

Programmation événementielle

44

description	événement
désélection de la cible	onBlur
click gauche de la souris	onClick
modification du contenu de la cible	onChange
activation de la cible	onFocus
changement d'une page HTML	onLoad
passage de la souris sur la cible	onMouseOver
sélection de la cible	onSelect
soumission de formulaire	onSubmit
sortie de page HTML	onUnload
annulation du chargement d'une image	onAbort
glissement d'un objet vers la fenêtre du navigateur	onDragDrop
chargement causant une erreur	onError
frappe d'une touche clavier	onKeyDown
appui sur une touche clavier	onKeyPress
relâchement d'une touche clavier	onKeyUp
appui sur le bouton souris	onMouseDown
déplacement du curseur de la souris	onMouseMove
sortie du curseur de la souris de la zone sensible	onMouseOut

relâchement du bouton de la souris	onMouseUp
déplacement d'une fenêtre	onMove
reset d'un formulaire	onReset
changement de taille d'une fenêtre	onResize

Pour créer un gestionnaire d'événement, il suffit d'utiliser une balise et d'ajouter le mot clé de l'événement avec un code JavaScript indiquant l'action à entreprendre si l'événement se produit

<BALISE onQuelquechose="code javascript">

- Les cookies sont des informations qui sont écrites sur le poste client, sur le disque dur et donc mémorisées en vue d'une utilisation ultérieure lors d'une connexion future.
- Le fichier qui contient les cookies est usuellement appelé **cookies.txt** (c'est un fichier texte) qui se trouve, soit dans le répertoire **Netscape**, soit dans le sous-répertoire **cookies** du répertoire Windows (il porte un autre nom de la forme [nom_utilisateur@javascript.txt](#)).
- L'écriture et la lecture dans ce fichier peut être effectuée notamment par un petit programme JavaScript.

• La structure d'un cookie est la suivante :

nom=valeur [; expires=date][; domain=nom_domaine][; path=chemin][;secure]

- nom est une variable et valeur est sa valeur instantanée
- expires=date exprime la date de validité du cookie ; par défaut, le cookie expire à la fin de la session.
- domaine est le nom du domaine de validité du cookie (par défaut nom du serveur)
- path définit le chemin du cookie
- secure définit si le cookie est transmis de façon sécurisée.

- <http://www.info.univ-angers.fr/~richer/ensl3i.php#WEB>