



Javascript

Partie 2

Les objets



Objets offerts par le langage

Objet Math

- L'objet **Math** possède des méthodes et des propriétés prédéfinies.
- Les méthodes sont des fonctions.
- Les propriétés sont des nombres spécifiques.
 - Exemple avec la valeur de pi (3,1416).

propriété **Math.PI**

```
var rayon = 2;
```

```
var superficieducercle = Math.PI * rayon * rayon;
```

Math : méthodes

abs
acos
asin
atan
atan2
ceil
cos
exp
floor
log
max
min
pow
random
round
sin
sqrt
tan

- **Math.abs(*number*)**
- **Math.max([*number1*, *number2*... [, *numberM*]])**
 - Renvoie la plus grande de plusieurs expressions numériques.
- **Math.floor(*number*)**
 - Renvoie le plus grand entier < ou = à son argument numérique.
- **Math.min([*number1*, *number2*... [, *numberM*]])**
 - Renvoie le plus petit de plusieurs expressions numériques.
- **Math.random()**
 - Renvoie un nombre pseudo-aléatoire compris entre 0 et 1.
- **Math.sqrt(*number*)** Renvoie la racine carrée d'un nombre.

3

Objet String (et 'Literal')

```
ch1 = "bonjour";           // crée une 'simple' variable ('Literal')
ch2 = new String("Hello"); // crée un objet String
```

- **Remarque:**
les méthodes de String peuvent être appelées sur une chaîne. JavaScript convertit alors temporairement le littéral en un objet String.

4

String: propriétés et méthodes

Instruction	Description
• <code>length</code>	• Propriété: entier qui indique la longueur de la chaîne de caractères
• <code>charAt()</code>	• Méthode qui permet d'accéder à un caractère isolé d'une chaîne.
• <code>indexOf()</code>	• Méthode qui renvoie la position d'une chaîne partielle à partir d'une position déterminée (en commençant au début de la chaîne principale soit en position 0).
• <code>lastIndexOf()</code>	• Méthode qui renvoie la position d'une chaîne partielle à partir d'une position déterminée(en commençant à la fin c.a.d en position <code>length</code> moins 1).
• <code>substring(x,y)</code>	• Méthode qui renvoie un string partiel situé entre la position <code>x</code> et la position <code>y-1</code> .
• <code>toLowerCase()</code>	• Transforme toutes les lettres en minuscules
• <code>toUpperCase()</code>	• Transforme toutes les lettres en Majuscules.

5

Propriété `length`

- La propriété `length` retourne un entier qui indique le nombre d'éléments dans une chaîne de caractères. Si la chaîne est vide (`""`), le nombre est zéro.
- Syntaxe:


```
x = variable.length;
x = ("chaîne de caractères").length;
```

6

Méthode `charAt (index)`

- Renvoie le caractère à l'index spécifié.

`strObj.charAt (index)`

- **`strObj`** est une variable String
- **`index`** : la position du caractère demandé. Les valeurs correctes sont admises entre 0 et la longueur de la chaîne moins 1.
- les caractères sont comptés de gauche à droite et la position du premier caractère est 0. La position du dernier caractère est donc la longueur (length) de la chaîne de caractère moins 1.

7

Exemple de `charAt ()`

- `var str="Javascript";`
- `var chr=str.charAt(0);`
- `var chr="Javascript".charAt(0);` → J

- `var str="Javascript";`
- `var chr=str.charAt(9);` → t

- `var str="Javascript";`
- `var chr=charAt(str,13);` → "

8

Fonction `indexOf (...)`

- Renvoie la position du caractère où s'est produite la première occurrence d'une sous-chaîne à l'intérieur d'un objet **String**.

strObj.indexOf(subString[, startIndex])

- **strObj**: littéral ou objet **String**
- **subString** : sous-chaîne à rechercher dans l'objet **String**.
- **startIndex**:
 - Facultatif. Valeur entière facultative spécifiant l'index marquant où commencer la recherche dans l'objet **String**. Si cette valeur est omise, la recherche commence au début de la chaîne.
- **Notes**
 - La méthode **indexOf** renvoie une valeur entière indiquant le début de la sous-chaîne dans l'objet **String**. Si la sous-chaîne n'est pas trouvée, un -1 est renvoyé.
 - Si *startIndex* est négatif, *startIndex* est interprété comme zéro s'il est négatif, et comme le plus grand index possible s'il est trop grand.
 - La recherche est effectuée de gauche à droite.

9

Fonction `indexOf ()`

Exemple

- ```
function IndexDemo(str2){
 var str1 = "BABEBIBOBUBABEBIBOBU";
 var s = str1.indexOf(str2);
 return(s);
}
```
- Note: la position du premier caractère est 0

10

## Fonction substring(...)

- Renvoie la sous-chaîne à la position spécifiée à l'intérieur d'un objet **String**.  
**strObj.substring(start, end)**
  - **start** : entier d'index indiquant le début de la sous-chaîne désirée.
  - **end** : entier d'index indiquant la fin de la sous-chaîne désirée.
- La méthode substring renvoie une chaîne contenant la sous-chaîne depuis start jusqu'à end (exclus)
- Si start ou end est égal à NaN ou est négatif, il est remplacé par zéro.
- La longueur de la sous-chaîne est égale à la valeur absolue de la différence entre start et end. Par exemple, la longueur de la sous-chaîne renvoyée par strObj.substring(0, 3) est trois.

11

## Fonction substring( )

### *Exemple*

- str="Javascript";
- str1=str.substring(0,4);
- str2="Javascript".substring(0,4);
- str3=str.substring(6,9);

Les résultats sont ?

str1="Java"; soit les positions 0,1,2 et 3.  
 str2="Java"; soit les positions 0,1,2 et 3.  
 str3="rip"; soit les positions 6,7 et 8

12

## Objet Date

- L'objet **Date** est utilisé pour représenter les dates et heures, pour obtenir la date actuelle ("currentdate") du système et pour calculer les différences entre les dates.
- Il possède plusieurs propriétés et méthodes prédéfinies.

13

## Objet Date

- Syntaxe

```
dateObj = new Date()
```

```
dateObj = new Date(string)
```

```
dateObj = new Date(year, month, date[, hours[,
 minutes[, seconds[,ms]]]])
```

- **year**: l'année, par exemple 1976 (et non 76).
- **month**: le mois, un entier compris entre 0 et 11 (janvier à décembre).
- **date**: le jour, un entier compris entre 1 et 31.
- **hours**: Facultatif. Doit être fourni si **minutes** est spécifié. Un entier compris entre 0 et 23 (minuit à 23 heures) indiquant l'heure.
- **minutes**: Facultatif. Doit être fourni si **seconds** est spécifié. Un entier compris entre 0 et 59 indiquant les minutes.
- **seconds**: Facultatif. Doit être fourni si **milliseconds** est spécifié. Un entier compris entre 0 et 59 indiquant les secondes.
- **ms**: Facultatif. Un entier compris entre 0 et 999 indiquant les millisecondes.

14

## Date

### Méthodes

|                                |                                                                 |
|--------------------------------|-----------------------------------------------------------------|
| <code>getDate()</code>         | Retourne le jour du mois pour la date spécifiée.                |
| <code>getDay()</code>          | Retourne le jour de la semaine pour la date spécifiée.          |
| <code>getFullYear()</code>     | Retourne l'année pour la date spécifiée.                        |
| <code>getHours()</code>        | Retourne l'heure pour la date spécifiée.                        |
| <code>getMilliseconds()</code> | Retourne le nombre de millisecondes pour la date spécifiée.     |
| <code>getMinutes()</code>      | Retourne le nombre de minutes pour la date spécifiée.           |
| <code>getMonth()</code>        | Retourne le mois pour la date spécifiée.                        |
| <code>getSeconds()</code>      | Retourne le nombre de secondes pour la date spécifiée.          |
| <code>getTime()</code>         | Retourne la valeur numérique correspondant à la date spécifiée. |

**Exemple :**

```
o = new Date("1962,04,22");
document.write(o.getDate());
document.write("
");
document.write(o.getFullYear());
```

15

## Création de ses propres objets





Il existe plusieurs façons syntaxiques de créer des objets.

Exemple

```
obj = new Object();
obj.attribut = "valeur1";
document.write(obj.attribut + "
");
```

*Rappel :  
On distingue les types primitifs des types composés ( objets ) :  
les premiers sont manipulés par valeur, tandis que les autres le sont par référence.*

Autre exemple syntaxique de création d' objet avec 3 propriétés:

```
function _peripherique()
{
 this.imprimante='HP';
 this.clavier='Microsoft';
 this.souris='Cherry';
}

monPeripherique=new _peripherique();
document.write(monPeripherique.imprimante);
```

Autre exemple syntaxique de création « d'objet » avec la syntaxe « Json »  
Attention, ici il n'y a pas de notion de classe qui précède l'objet.  
L'objet est créé sans classe.

```
var obj =
{
 mapropriete: "valeur",
 mamethode: function(argument)
 {
 alert("argument: " + argument);
 }
}

alert("Valeur de propriete: " + obj.mapropriete);
obj.mamethode("Ma valeur d'argument");
```

Exemple avec un 'constructeur'

```
function soldat(nom)
{
 this.nom = nom;
 this.combat = function(arme)
 { return this.nom + ' se bat avec : ' + arme + '
'; }
}
monSoldat = new soldat('Lancelot');
document.write(monSoldat.combat('Epée'));
```

\* La propriété **prototype**

Cette propriété spéciale s'applique à une classe déjà construite et permet de lui ajouter de nouvelles propriétés et méthodes.


On peut effectuer ces ajouts sur des classes prédéfinies ou définies par le programmeur.

Comme on ne passe pas par le constructeur de l'objet, il faudra attribuer une valeur aux propriétés ajoutées de cette manière.

Syntaxe : `nomClasse.prototype.nomPropriété=valeurInitiale;`


```
function Vehicule(unNom)
{
 this.nom = unNom;
 this.demarrer = function()
 { document.write("Vroum vroum
"); }
 this.arreter = function()
 { document.write(this.nom + ' en arrêt
'); }
}
vehicule1 = new Vehicule("Voiture");
vehicule1.demarrer();vehicule1.arreter();
vehicule2 = new Vehicule('Bateau');
vehicule2.demarrer();vehicule2.arreter();

Vehicule.prototype.typeMoteur='mécanique';
document.write(vehicule1.typeMoteur + '
');
document.write(vehicule2.typeMoteur + '
');
```

Voir l'exécution de  
cet exemple 

Ajout de méthodes via 'prototype'

```
function Vehicule(unNom)
{
 this.nom = unNom; this.etat="";
 this.demarrer = function()
 { document.write("Vroum vroum
"); this.etat='ok'; }
 this.arreter = function()
 { document.write(this.nom + ' en arrêt
'); this.etat='ko'; }
}
Vehicule.prototype.recapitulatif = function()
{
 var info = 'Nom : ' + this.nom;
 info += ' | Moteur : ' + this.typeMoteur;
 if (this.etat == 'ok') info += ' | Etat : Démarré';
 else info += ' | Etat : Arrêté';
 document.write(info);
}
Vehicule.prototype.typeMoteur='mécanique';
vehicule = new Vehicule("Titine");
vehicule.demarrer();vehicule.recapitulatif();
```

Voir l'exécution de  
cet exemple 

Exemple élégant qui étend la classe array en lui ajoutant une fonction qui mélange un tableau.

```
Array.prototype.shuffle = function()
{
 var s = [];
 while (this.length) s.push(this.splice(Math.random() * this.length, 1));
 while (s.length) this.push(s.pop());
 return this;
}
```

Exemple d'utilisation

```
var monTableau= [0,1,2,3,4,5,6,7,8,9];
monTableau.shuffle();
```

*Exemple vu sur*

<http://javascript.about.com/library/blshuffle.htm>

### Travaux pratiques

Lecture de

**Programmation orientée objet avec le langage JavaScript (1ère partie)**

→Jspoo1.pdf

Lecture de

**Programmation objet en JS**

→c-prog-objet.html

Voir

<http://www.toutjavascript.com/reference>

En particulier l'objet [window.document](#)

"Essayer" de lire

<http://www.pompage.net/traduction/classe-et-heritage-en-javascript>

( Si le temps butiner un peu sur [https://developer.mozilla.org/fr/Guide\\_JavaScript\\_1.5](https://developer.mozilla.org/fr/Guide_JavaScript_1.5) )

## Exercice

Coder une 'classe' Individu' qui prend en signature 6 arguments :  
Taille, Poids, Age, Yeux, Cheveux, Nom

La classe offrira une méthode AnNaiss qui calcule l'année de naissance à partir de l'age.

Exemple :

```
quidam = new Individu("1,80", "77kg", "28", "bleus", "bruns", "Dugenoud");
document.write(quidam.Nom + ' est né en ' + quidam.AnNaiss());
```



**BONUS**

<http://www.hongkiat.com/blog/tools-to-coding-online/>

En particulier **JsFiddle**

exemple : <http://jsfiddle.net/codepo8/XJcpg/1/>

(<http://www.petit-kiwi.com/jsfiddle-outil-javascript>)

