ANGULARJS

Rappels sur JavaScript

Rappels JavaScript

Introduction et rappel

- Javascript permet de rendre dynamique un site internet développé en HTML.
- Javascript permet de développer de véritables applications fonctionnant exclusivement dans le cadre d'Internet.
- Le Javascript est un langage de script simplifié orienté objet dont la syntaxe est basée sur celle du Java.

JavaScript

- Des pages web interactives. Il peut être inclus dans un document HTML :

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript" src="script.js"></script>
<script type="text/javascript">
//<![CDATA[
/* code JavaScript */
//]]>
</script>
</head>
<body></body>
</html>
```

Les fonctions

Il est possible de définir des fonctions en JavaScript :

```
function additionner(a,b,c) {
a = b + c;
return a;
}
function multiplier(a,b) {
return a*b;
}
alert(additionner(1,2,3));
alert(mutltiplier(2,4));
```

Les variables

- **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"];
```

Les types

En JavaScript, les types sont associés aux valeurs :

```
var v = "toto"; // variable contient une chaîne
v = 'toto'; // variable contient une chaîne
v = 22; // variable contient un nombre
v = 22.12; // variable contient un nombre
v = true; // variable contient un booléen
v = ["toto", 22]; // variable contient un tableau
v = {name:"toto", age:22}; // variable contient un objet
```

Une variable qui n'a jamais été affecté est "undefined" :

```
var v; /* v est considéré comme "undefined". */
```

La valeur "null" peut être affectée à une variable. Cela signifie que la variable existe mais sa valeur ne désigne aucun objet :

```
var v = null;
```

Les booleans

Affectation et utilisation des booleans :

```
var a = true;
var b = false;
var c = a || b;
if (c) {
alert(a && b);
}
```

Les nombres

Les différentes opérations arithmétiques :

```
var v = 12;
var v = 2 + 4;
var v = 2 * 4;
var v = 4 / 2;
var v = 123 % 10;
v++; v--;
```

Les différentes affectations :

```
v=2; v+=2; v-=2; v*=2; v/=2; v%=2;
```

Les conversions entre chaînes et nombres :

```
var v = parseInt("123");
var v = parseFloat("123.12");
var s = v.toString();
```

Les comparaisons et opérateurs logiques

égal	a == b	vrai si a est égal à b
identique	a === b	vrai si a et b sont égaux et ont le même type
différent	a != b	vrai si a est différent de b
non identique ou n'ont pas le même type	a !== b	vrai si a et b sont différents
plus petit	a <	b vrai si a est strictement plus petit que b
plus grand	a > b	vrai si a est strictement plus grand que b
inférieur ou égal	a <= b	vrai si a est plus petit ou égal à b
supérieur ou égal	a >= b	vrai si a est plus grand ou égal à b

non	!a	vrai si a n'est pas vrai
et	a && b	vrai si a et b sont vrais
ou	a || b	vrai si a ou b sont vrais

```
var v = (test)?"Vrai":"Faux";
```

If/For/While/Continue/Break

```
var x = 2, y = 5;
if (x < y) document.write("<");
else document.write(">");
for (var i = 0; i < 10; i++) {
if (i == 4) continue;
document.write(i);
if (i > 7) break;
}
var i = 0;
while(i < 10) {
document.write(i);
i++;
}
```

Switch

```
function (x) {
switch (x) {
case 0 : alert("zéro"); break;
case 1 : alert("un"); break;
case 2 : alert("deux"); break;
case 3 : alert("trois"); break;
case 4 : alert("quatre"); break;
default : alert("nombre"); break;
}
}
```

Les tableaux

Déclaration d'un tableau :

```
var fleurs = new Array();
fleurs[0] = "Rose";
fleurs[1] = "Tulipe";
fleurs[2] = "Coquelicot";
/* ou */
fleurs = ["Rose", "Tulipe", "Coquelicot"];
```

Les méthodes et les propriétés:

```
var length = fleurs.length;
var position = fleurs.indexOf("Tulipe");
```

Les tableaux

- Parcourir un tableau :
- **for (var i = 0; i < fleurs.length; i++)**
- document.write(i+"->"+fleurs[i]);
- **for (var i in fleurs)**
- document.write(i+"->"+fleurs[i]);
- **for (var fleur of fleurs)** ***/* Expérimental */***
- document.write(fleur);

Les objets

- Un objet est un élément nommé ayant des
  - Propriétés : paramètres que vous vérifier et modifier.
  - Méthodes : actions que l'objet est capable d'effectuer.
  - Événements : choses qui arrivent à l'objet, auxquelles celui-ci peut répondre automatiquement par une action.

Exemple de création d'Objet

- Exemple 1 :

```
//instantiation de l'objet via la notion d'objet  vide
var pers2 =  {};
pers2.nom = "Karim";
pers2.prenom = "Mohammed";
pers2.getInfo = function getInfo() {
        return 'Info :  ' + pers2.nom+ ' ' + pers2.prenom+ '.';
    };
alert(pers2.getInfo()); //affiche  'Info :  Karim Mohammed.'
```

Exemple de création d'Objet

- Exemple 2 :

```
function Personne(name) {
    this.nom = name;
    this.prenom = "";
    this.getInfo = function getInfo() {
        return 'Info :  ' + this.nom+ ' ' + this.prenom+ '.';
    };
}

//instantiation de l'objet  via la méthode constructeur
var pers = new Personne ('Ahmed');
pers.prenom = "Mohammed";
alert(pers getInfo()); //affiche  'Info :  Ahmed Mohammed.'
```

Les objets

En JavaScript, il n'y a pas de classe. Les instance sont créée directement :

```
person=new Object();
person.name="Bob";
person.age=22;
```

On peut également utiliser une description littérale de l'objet :

```
person = {name : "Bob", age : 22};
```

Nous sommes en présence de références :

```
var bob = {name: "Bob", age:22}
var jim = {name: "Jim", age:23}
var joe = {name: "Joe", friends : [bob, jim]}
jim.age = 43;
alert(joe.friends[1].age); // affiche 43
```

JSON

JSON (JavaScript Oject Notation) est un format de données dérivé de la notation des objets et tableaux de ECMAScript (donc de JavaScript) :

```
{"machin": {
"taille": 12,
"style": "gras",
"bidule": {
"machin": [
{"style" : "italique" },
{"style" : "gras" }
]
}
}}
```

L'avantage de JSON est qu'il est reconnu nativement par JavaScript.

JSON

Les éléments en JSON :

- Les objets : {chaîne : valeur, chaîne : valeur...}
- Les tableaux : [valeur, valeur, ...]
- Les valeurs : chaîne, nombre, objet, tableau, true, false, null
- Les chaînes : "abcdef" ou "abcd\n\t"
- Les nombres : -1234.12

```
{"unObjet": {
"unTableau": [12, 13, 53],
"unNombre" : 53,
"unChaîne" : "truc\n"
"unObjet" : { "style" : "gras" }
}}
```

JSON

En JavaScript, il est très facile de sérialiser une valeur en JSON :

```
var bob = {name: "Bob", age:22}
var jim = {name: "Jim", age:23}
var joe = {name: "Joe", friends : [bob, jim]}
JSON.stringify(joe);
```

Le code précédent génère la chaîne de caractères suivante :

```
{
"name":"Joe",
"friends":[
{"name":"Bob","age":22},
{"name":"Jim","age":23}
]
}
```

JSON

Inversement, il est facile de construire une valeur à partir d'une chaîne de
caractères respectant le format JSON en utilisant la fonction *eval :*

```
var json = '{'+
'"name":"Joe",'+
'"friends":['+
'{"name":"Bob","age":22},'+
'{"name":"Jim","age":23}'+
']'+
'}';
var joe = eval('('+json+')');
for (var i = 0; i < joe.friends.length; i++)
alert(joe.friends[i].name);
```