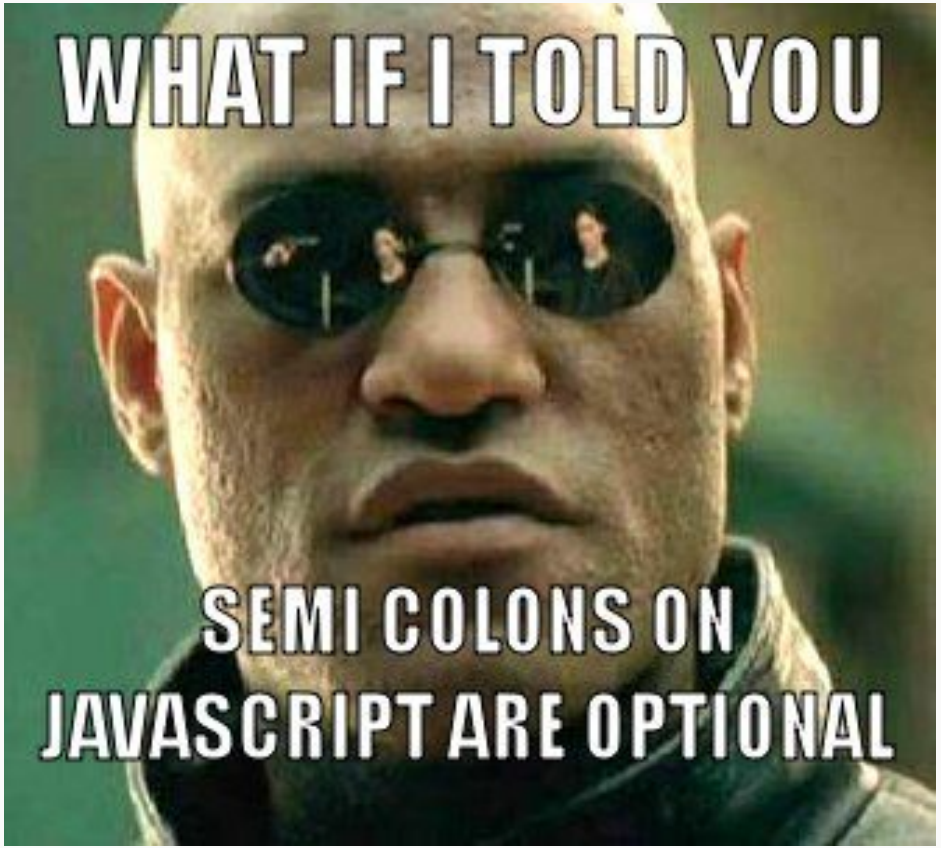
Javascript

Bases du langage

Mathieu Lallemand 01.2016


WHAT IF I TOLD YOU
SEMI COLONS ON
JAVASCRIPT ARE OPTIONAL

Notions essentielles

- Langage Programmation Orienté Objet “Prototype”
- La fonction est un “objet” (et l’objet est un fonction)
- Variables “transtypables”, un entier peut devenir une chaine
- Fonctions anonymes
- Asynchronisme / Callbacks

Éléments de langage

- Itérateurs
- Générateurs
- Définition par compréhension
- Assignation destructurante
- Tableaux à index nommés
- Let

Opérateurs, Variables, Conditions, Fonctions.

Opérateurs

+ Addition

- Soustraction

* Multiplication

/ Division

% Modulo

++ Incrément

-- Décrément

= Assignation

+= Assignation additive

-= Assignation soustractive

*= Assignation multiplicative

/= Assignation divisante

%= Assignation “Modulative?”

Variables

- Variables définies avec le mot clé “var”
- Les variables sont transtypable
    - Retours de la commande “typeof variable” :
      number, string, boolean, array, object, undefined
- On peut chaîner les déclarations de variables

```
var a=3,
b=2.2,
c=« coucou »,
d=[‘Hello’,’World’, 4];
```

Variables

- Les lignes d'instruction se terminent par un « ; » (optionnel mais recommandé)
- Let permet une redéfinition locale d'une variable
  - Attention tout de même à la lisibilité du code...

```
"use strict";
var x = 5;
for (var i=0; i<x; i++) {
  let x = "toto",
      i = 9;
  console.log(x,i);
}
```

Bloc d’instruction

- Les blocs d’instructions sont définis par des accolades « {} »
- Elles peuvent être sur la même ligne
- L’accolade ouvrante est sur la ligne de la fonction
- L’identation de bloc se fait sur 4 espace
- L’accolade fermante est sur une ligne dédiée
    - Parfois partagée avec la parenthèse fermante d’un callback

Test - IF-THEN-ELSE

```
"use strict";
if (boolean) {
  console.log("boolean is true");
}
else {
  console.log("boolean is false");
}


// Forme courte
var r = (test) ? action_1 : action_2;
ou
return (test) ? “msg_1” : “msg_2”;
```

Test - SWITCH-CASE

```
switch(variable) {
  case "TOTO":
  case "TATA":
    console.log("TOTO ou TATA");
    break;
  case "TITI":
    console.log("TITI");
    break;
  case "TUTU":
    console.log("TUTU");
  default:
    console.log("Fini !");
}
```

Boucle - FOR

```
// Initialisation, Condition, Incrément
for (var i=0; i<100; i++) {
  console.log("i="+i);
}

// Dans un object
var o = {a:"toto",b:"tata",c:"titi"};
for (var i in o) {
  console.log(i, o[i]);
}
```

Boucle - WHILE

```
while(true_boolean) {
  // Faire un truc
}
```

```
do {
  // Faire un truc
} while (true_boolean)
```

Gestion - TRY-CATCH-FINALLY

```
try {
  // Code pouvant générer une exception
}
catch (e) {
  // Gestion de l'exception
  console.log(e);
}
finally {
  // Code exécuté même en cas d'exception
}
```

Fonctions

Déclaration d’une fonction

- Les fonctions sont déclarées à l’aide du mot clé « function ».
- Une fonction peut être anonyme.
- Une fonction peut être définie n’importe où.

```
function hello() {
  console.log ("Hello World !");
}
```

```
var hello = {
  world : function() { return "Hello World"; }
}
```

Paramètres d’une fonction

- Les paramètres sont définis sans types

```
function hello(name) {
  console.log ("Hello "+name+" !");
}
```

- On peut aussi ne pas spécifier de variables, les arguments sont alors transmis dans la fonction sous le mot clé ‘arguments’ qui est un tableau.

```
function hello() {
  console.log (arguments);
  console.log ("Hello "+arguments[0]+" !");
}
```

Appel d’une fonction

- Nom de la fonction suivi de “()”
- () exécute une fonction, où qu’elle se trouve dans le scope disponible.

```
toto();
function toto () { console.log ("Hello World !"); }
```

- Il est possible d'exécuter une fonction à sa définition (closure)

```
(function toto () {
  console.log ("Hello World !");
})();
```

Scope d’une fonction

```
// code here can not use carName

function myFunction() {
    var carName = "Volvo";

    // code here can use carName

}
```

```
var carName = " Volvo";

// code here can use carName

function myFunction() {

    // code here can use carName

}
```

Scope d’une fonction

Le scope d’une fonction est constitué de :

- son/ses contenants
- son contenu propre

```
var x=5;
function toto(a,b,c) {
  console.log( this.x + a + b + c );
}
toto(1,2,3); // Affiche 11;

x=10;
toto(1,2,3); // Affiche 16;
```

Le noeud supérieur est accessible via le mot clé : “this”

Attention : “this” est dynamique.