

Javascript

Initiation



Internet : Javascript

Objectif

JavaScript / Json / Ajax

Sommaire

1. Définition
2. Bases
3. Divers



JavaScript / Json / AJAX

1 - JavaScript - Définition + bases

1a. Le Javascript c'est quoi ?

JavaScript (souvent abrégé JS) est un langage de programmation de scripts principalement utilisé dans les pages web interactives mais aussi côté serveur.

C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés.

<http://fr.wikipedia.org/wiki/JavaScript>



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

1b. Les bases du langage

Bob est un homme (= objet)

Bob est né le 28/11/1973

Bob peut manger, lire et calculer son âge

Bob est une instance de classe

"développeur"

Bob est basé sur un autre objet appelé

"développeur"

Bob garde ses informations et les méthodes qui vont avec ses informations

Bob a ses méthodes privées

Bob/dev travaille avec Jill/graph et Jack/PM

Dev/graph/PM basés sur l'objet personne

Bob:talk – Jill:talk – Jack:talk

objet

propriétés

méthodes

classes (OOP classique)

prototype (OOP prototype)

encapsulation

privé / publique

aggrégation, composition

héritage

Polymorphisme/surcharge

Javascript / Json / AJAX

1 - JavaScript - Définition + bases

1b-i. Les primitives

Par coeur :

- 1 - Number
- 2 - String
- 3 - Boolean
- 4 - undefined
- 5 - null (≠ undefined)
- 6 - NaN
- 7 - Infinity
- 8 - function
- 9 - object

Déclarer une variable

```
var a;
```

Initialiser une variable

```
var a=1;
```

Les variables sont sensibles à la casse

Console : essayer :

```
var aa = 'test';
```

```
var AA = 'autre';
```

```
aa
```

```
AA
```

```
typeof(AA)
```



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

1b-ii. Les blocs de code

Bloc simple

```
{  
    var a = 1;  
    var b = 3;  
}
```

Blocs dans des blocs

```
{  
    var a = 1;  
    var b = 3;  
    {  
        var c = 1;  
        var d = 3;  
    }  
}
```



JavaScript / Json / AJAX

1 - JavaScript - Définition + bases

1b-ii. La syntaxe

Déclarer une variable

```
var tutu = 12; var t, u, v =15, yy="o";
```

Sensibles à la casse.

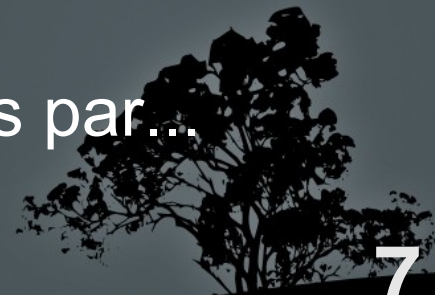
```
Tutu != tutu
```

Déclarer une fonction

```
function test() {  
}
```

Fonction avec des paramètres

La portée est par fonction et non pas par...
bloc.



JavaScript / Json / AJAX

1 - JavaScript - Définition + bases

1b-ii. La syntaxe

Erreur classique :

```
function test() {  
    return  
        (1 + 2 + 3 + 4 + 5 + 6);  
}  
  
console.log(test());
```



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

1 b-iii. Les opérateurs de calcul

<code>1 + 2</code>	<code>3</code>
<code>99.99 - 1</code>	<code>98.99</code>
<code>2 * 3</code>	<code>6</code>
<code>6 / 4</code>	<code>1.5</code>
<code>6 % 3</code>	<code>0</code>
<code>255 ^ 128</code>	<code>127</code>
<code>var a=12; a++; a</code>	<code>13</code>
<code>var b=a; b--; b</code>	<code>12</code>



1b-iv. Les opérateurs logiques

NOT logique	!xx	
ET logique	&&	
OU logique		
var b = !true; b		false
var b = !!true; b		true
var b = "one"; !b		false
var b = "one"; !!b		true
Chaine vide : var u1=''; !u1		
!null		false
!undefined		false
Nombre !0		false
Nombre !NaN		false
Booléen !false		false



1b-vi. Les opérateurs de comparaison

==	Vrai si égalité
===	Vrai si égalité + type
!=	Vrai si non-égalité
!==	Vrai si non-égalité OU types différents
>	Vrai si gauche supérieur à droite
>=	Vrai si gauche supérieur ou égal à droite
<	Vrai si gauche inférieur à droite
<=	Vrai si gauche inférieur ou égal à droite

Tester :

1==1

1 === '1'

1 != '2'

1 !== '2'

1 > 1

1 <= 1

1 < 1

1 <= 1



1b-vii-a. Les structures conditionnelles

Conditions "if"

```
if (a>3) {  
    result = 'ok';  
}
```

Conditions "if-else"

```
if (a>3) {  
    result = 'ok';  
} else {  
    result = 'erreur';  
}
```

Conditions "if-else-if"

```
if (note>15) {  
    result = 'bon';  
} else if (note>10) {  
    result = 'moyen';  
} else {  
    result = 'mauvais';  
}
```



1b-vii-b. Les structures conditionnelles

"if" imbriqués

```
if (a>10) {  
    if (a<15) {  
        result = 'moyen';  
    } else {  
        result = 'bon';  
    }  
} else {  
    result = 'mauvais';  
}
```

Opérateur ternaire

```
var result = (a==1 ? 0 : 1);  
A utiliser avec parcimonie
```

```
switch (note) {  
    case 0:  
        result = 'exclu';  
        break;  
    case 1:  
    case 2:  
    case 3:  
        result = 'bidon';  
        break;  
    case 19:  
    case 20:  
        result = 'normal';  
        break;  
    default:  
        result = 'à refaire';  
        break;  
}
```

Javascript / Json / AJAX

1 - JavaScript - Définition + bases

1b-viii. Les boucles

```
var i = 0;
do {
  i++;
} while (i<10);
```

```
var i=0;
while (i<10) {
  i++;
}
```

```
var p='';
for (var i = 0; i<100; i++) {
  p += 'test;';
}
```

```
for (var i = 0, p='';
     i<100;
     i++, p += 'test;') {
  /* rien */
}
```

```
var i = 0, p='';
for (;;) {
  if (++i== 100) { break; }
  p += 'test';
}
```



1b-ix. Les variables et leurs portées

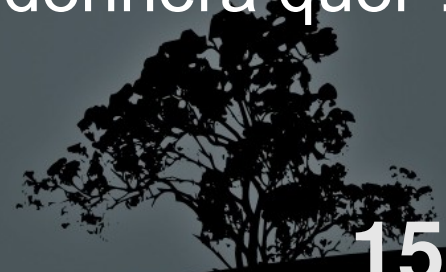
La portée des variables n'est pas par blocs.
La portée des variables est par fonction.

```
var global=1;
function f() {
    var local = 2;
    global++;
    return global;
}
```

f(); donnera quoi ?
Encore f(); donnera quoi ?
local donnera quoi ?

```
var a = 123;
function f() {
    console.log(a);
    var a = 1;
    console.log(a);
}
```

f(); donnera quoi ?



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

1c-x. Les chaînes - Syntaxe

```
var maChaine="des caracteres";  
var maChaine='des caracteres';  
var maChaine="\ntest\n123\n456";  
console.log(maChaine);  
donne quoi ?
```

\ Caractère d'échappement

\\ Écrire un \

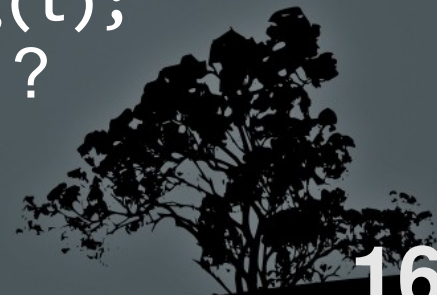
\n Écrire un retour chariot

\t Écrire une tabulation

\u Écrire un caractère unicode

```
var maChaine="\ttest\t123\t456";  
console.log(maChaine);  
donne quoi ?
```

```
var t='I \u2661 JavaScript!';  
console.log(t);  
donne quoi ?
```



1C-xi. Les chaines - Conversion

```
var s1 = "un";  
var s2 = "deux";  
var s = s1 + s2;  
s; donnera quoi ?  
typeof s; donnera quoi ?
```

```
var s1 = "un";  
var s2 = "deux";  
var s = s1 * s2;  
s; donnera quoi ?  
typeof s; donnera quoi ?
```

```
var s = "un";  
S = 3 * s  
s; donnera quoi ?  
typeof s; donnera quoi ?
```

```
var s = "3";  
s = s * 3;  
s; donnera quoi ?  
typeof s; donnera quoi ?
```



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

1C-xii. Les tableaux

```
var a=[];                var a=new Array();  
a; donnera quoi ?  
typeof a; donnera quoi ?
```

```
var a = ["12", 1, 2, 3, 5, "un"];  
a; donnera quoi ?  
a[3] = [5];  
a; donnera quoi ?
```



1c-xiii. Les tableaux — Opérations / fonctions

```
var a=[1,89,9];
```

Mettre à jour un élément

```
a[1] = 12;
```

a; donnera quoi ?

Ajouter un élément

```
a[10] = 0.25;
```

a; donnera quoi ?

Supprimer un élément

```
delete a[7];
```

a; donnera quoi ?

A connaître par coeur :

```
a.push();
```

```
a.pop();
```

```
a.slice();
```

```
a.splice();
```

```
a.sort();
```

```
a.join();
```

```
a.length
```



1C-xiii. Les tableaux - Boucles

Différence entre ces deux boucles ?

```
for (var i=0; i<a.length; i++) {  
    console.log(a[i]);  
}
```

```
for (var i in a) {  
    console.log(a[i]);  
}
```



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

1C-xv. Les objets - Déclaration

Tapez ce qui suit :

```
var o = {  
  A: 12,  
  B: "test"  
};  
o  
typeof(o);
```

Devinez les réponses.

Tapez ce qui suit :

```
var o = {  
  A: 12,  
  B: "test",  
  a: 3.25,  
  b: "autre"  
};  
o  
typeof(o);
```

Devinez les réponses.



JavaScript / Json / AJAX

1 - JavaScript - Définition + bases

1C-xvi. Les objets - Syntaxe

Un objet dans un objet :

```
var livre = {  
  titre: "Le chien des Baskerville",  
  publication: 1902,  
  auteur: {  
    civilite: "Sir",  
    nom: "Doyle",  
    prenom: "Arthur Conan"  
  }  
};
```

Donnez quatre façons de sortir avec `console.log()` :
Le chien des Baskerville (Sir Artur Conan Doyle)

Javascript / Json / AJAX

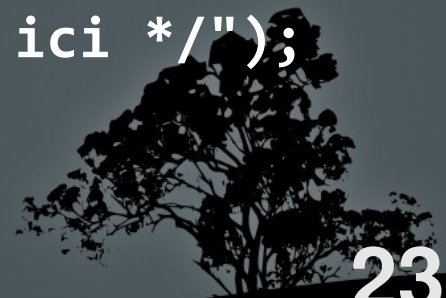
1 - JavaScript - Définition + bases

1C-xvi. Les objets - Boucles

```
var o = {  
  A: "test",  
  B: 2.565,  
  a: "autre",  
  b: "valeur"  
};
```

Terminez le code pour afficher toutes les valeurs des propriétés :

```
for (var i in o) {  
  Console.log("/ * Terminer le code ici */");  
}
```



JavaScript / Json / AJAX

1 - JavaScript - Définition + bases

1C-xvii. Les objets - Prédéfinis

Object
Array
Number
Boolean
String
Date
RegExp

(!) Les bases JavaScript sont ici...
La documentation
exhaustive est impossible
dans le cadre d'une initiation.

Objet particulier qui n'accepte pas "new" :
Math



1C-xviii. Les fonctions – Déclarations

```
var fn = function () {  
    console.log('bonjour');  
};
```

```
var o={  
    a: fn  
};
```

Comment appeler `fn` qui est dans l'objet `o` ?
Citez les deux possibilités



10-xix. Les fonctions — Déclarations

Une fonction est une donnée

```
function fn() {  
    console.log('bonjour');  
};
```

```
var fn = function () {  
    console.log('bonjour')  
};
```

```
var tab=[1, 2, fn, "aa"];
```

Comment appeler fn qui est dans le tableau tab ?

1d-i. Fonctions - Bases

En général :

```
function f() { return 1; }
```

En JavaScript :

```
var f = function() { return 1; }
```

`typeof f` donne quoi ?

```
var sum = function(a,b) { return a+b; }
```

```
var add = sum;
```

`delete sum` donne quoi ?

`typeof add` donne quoi ?

`add(1,2)` donne quoi ?



1d-ii. Fonctions – Anonymes

```
function invoque_et_add(a,b) {  
    return a() + b();  
}  
var un = function () { return 1; }  
var deux = function () { return 2; }  
>> invoque_et_add(un,deux); donne quoi ?  
  
>> invoque_et_add(  
    function () { return 1; },  
    function () { return 2; }  
);  
Est il possible, si oui, donne quoi ?
```



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

1e. Fonctions – Prédéfinies

<code>parseInt();</code>	<code>=> '123', 'a123', '1a23', '123a'</code>
<code>parseFloat();</code>	<code>=> '1.23', 'a1.23', '1a.23', '1.a23', '1.23a', '123e-2', '1e10'</code>
<code>isNaN();</code>	<code>=> NaN, 123, 1.23</code>
<code>isFinite();</code>	<code>=> NaN, Infinity, 1e308, 1e309</code>
<code>encodeURIComponent();</code>	<code>=> '?test=12&val=80'</code>
<code>decodeURIComponent();</code>	<code>=> 'je%20-test'</code>
<code>encodeURIComponent();</code>	<code>=> '?test=12&val=80'</code>
<code>decodeURIComponent();</code>	<code>=> 'je%20-test'</code>
<code>eval();</code>	<code>=> 'var a=12;'</code>
<code>alert();</code>	<code>=> 'bonjour'</code>
<code>console.log();</code>	<code>=> eval('var a=12;'); console.log(a);</code>

1f-i. Fonctions – Self-invoking

```
(  
    function() {  
        alert('coucou');  
    }  
) ();  
  
(  
    function(name) {  
        alert(name);  
    }  
) ('ceci est un test');
```



1f-i. Fonctions – Self-invoking

Scoping : quelles seront les sorties consoles ?

```
var test = 123;
if (true) {
    (function () {
        var test = 456;
        console.log(test);
    })();
    console.log(test);
}
console.log(test);
```



1 f-ii. Fonctions – Privées

```
var a= function (param) {  
    function b(theinput) {  
        return theinput*2;  
    };  
    return 'Résultat ' + b(param);  
};
```

```
a(2);  
a(48);  
b(test);
```



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

2a-i. POO – Bases

```
function Identite(nom, prenom) {  
    this.nom = nom;  
    this.prenom = prenom;  
}
```

```
var oo = new Identite('pons', 'olivier');  
console.log(oo.nomCompleter());
```



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

2a-ii. POO – Bases

```
String.prototype.maFonction = function () {  
    return ('Ma longueur est : '+this.length);  
};
```

```
var oo = "Test";  
console.log(oo.maFonction());
```



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

2a-iii. POO – Bases

Expliquez ce qui suit :

```
function Identite(nom, prenom) {
    this.nom      = nom;
    this.prenom   = prenom;
}
Identite.prototype.nomComplet = function () {
    return this.nom+' '+this.prenom;
};
Identite.prototype.valeur = 'test';
var oo = new Identite('a', 'c');
oo.valeur='autrevaleur';
var op = new Identite('b', 'd');
console.log(oo.valeur);
```



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

2a-iv. POO – Bases

Déterminer un type : `instanceof`

```
function Identite(nom, prenom) {  
    this.nom      = nom;  
    this.prenom  = prenom;  
}
```

```
var h=new Identite('pons', 'olivier');
```

Que donne :

`h instanceof Identite`

`h instanceof Object`



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

2a-vi. POO – Bases

Fonction cachée : le constructeur : constructor

```
function Identite(nom, prenom) {  
    this.nom      = nom;  
    this.prenom  = prenom;  
}
```

```
var h=new Identite('pons', 'olivier');
```

Que donne :
h.constructor
typeof(h.constructor)



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

2a-vi. POO – Bases

Créer des objets sans "new" :

```
function sansNew(nom, prenom) {  
    return {  
        nom: nom,  
        prenom: prenom  
    };  
}  
  
var h=sansNew('pons', 'olivier');
```

Que donne :

h.nom

h.constructor



2b. POO – Objet global - Définition

Dans les navigateurs,

Deux objets globaux à connaître :

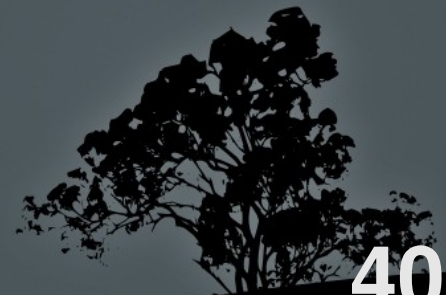
- `window`
- `document`



2c. POO – Exceptions

Essayez :

```
try {  
    console.log('a');  
    fonctionQuiNExistePas();  
    console.log('b');  
} catch(e) {  
    console.log(e.name + ' - ' + e.message);  
} finally {  
    console.log('Finally !');  
}
```

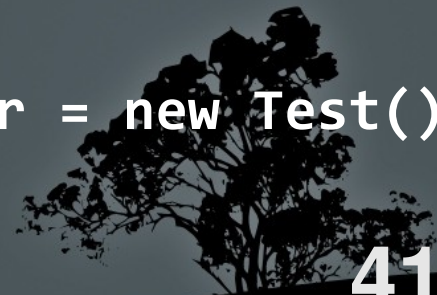


3-i. Subtilités

Deux codes : un seul fonctionne.
Lequel ? Pourquoi ?

```
function Test() {  
    this.t=function() {  
        var self = this,  
            other = -1,  
            self.tutu = 15;  
        console.log(self);  
    }  
}  
var olivier = new Test();
```

```
function Test() {  
    this.t=function() {  
        var self = this,  
            other = -1;  
        self.tutu = 15;  
        console.log(self);  
    }  
}  
var olivier = new Test();
```



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

3h-ii. Subtilités

Tapez ce qui suit :

```
var tab = [];  
tab['a'] = 12;  
tab['b'] = "test";
```

Puis

```
tab['b']  
tab.length
```

Devinez les réponses.

Tapez ce qui suit :

```
var tab = {};  
tab['a'] = 12;  
tab['b'] = "test";
```

Puis

```
tab['b']  
tab.length
```

Devinez les réponses.

Il y a deux incohérences. Lesquelles ? Pourquoi ?

Javascript / Json / AJAX

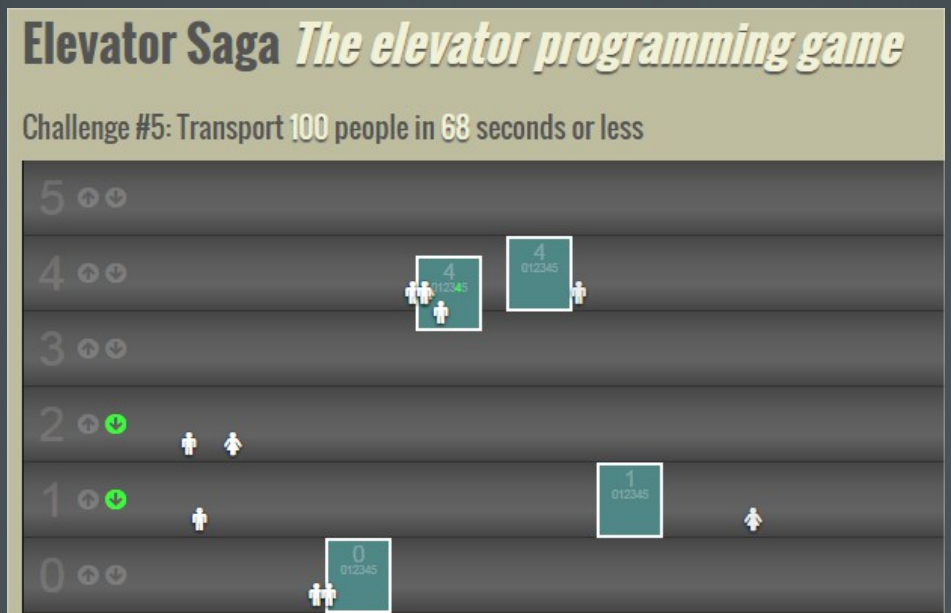
1 - JavaScript - Définition + bases

3h-i. Divers

Apprenez en jouant :

<http://play.elevatorsaga.com/#challenge=1>

<https://codecombat.com/>



Javascript / Json / AJAX

1 - JavaScript - Définition + bases

3h-ii. Divers

<http://nodeschool.io/>
`npm install -g functional-javascript-workshop`
`functional-javascript-workshop`

```
» Hello World
» Higher Order Functions
» Basic: Map
» Basic: Filter
» Basic: Every Some
» Basic: Reduce
» Basic: Recursion
» Basic: Call
```

```
FUNCTIONAL JAVASCRIPT IS GOOD
-----
» Hello World
» Higher Order Functions
» Basic: Map
» Basic: Filter
» Basic: Every Some
» Basic: Reduce
» Basic: Recursion
» Basic: Call
» Partial Application without Bind
» Partial Application with Bind
» Implement Map with Reduce
» Function Spies
» Blocking Event Loop
» Trampoline
» Async Loops
» Recursion
» Currying
» Function Call
-----
HELP
EXIT
```