

JavaScript

CM1-2 : JavaScript, types et opérateurs

Mickaël Martin Nevot

V4.2.1



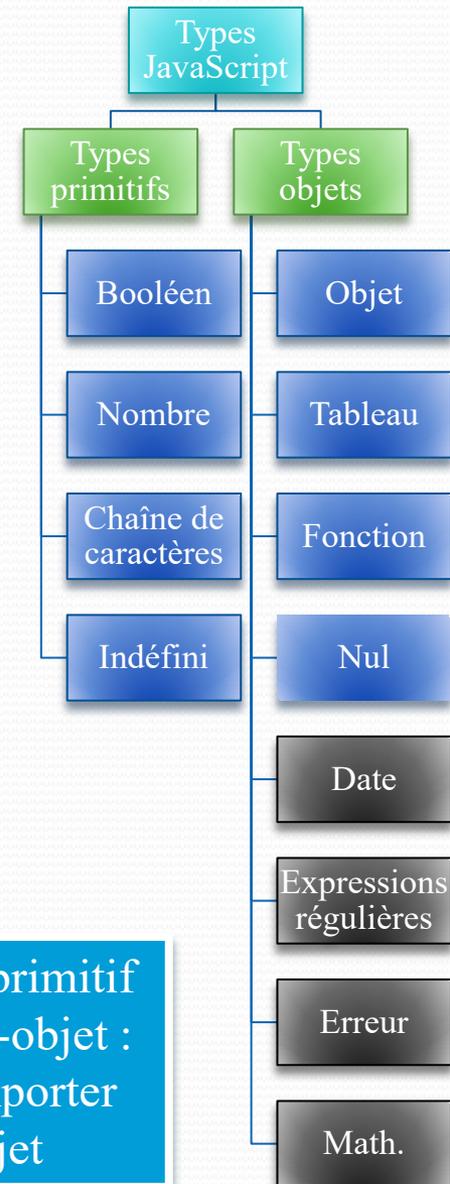
Cette œuvre de Mickaël Martin Nevot est mise à disposition sous licence Creative Commons Attribution - Utilisation non commerciale - Partage dans les mêmes conditions.

JavaScript

- I. Présentation
- II. JS
- III. Types/opérateurs
- IV. Avancé
- V. Ajax
- VI. DOM
- VII. XHR
- VIII. JSON
- IX. jQuery
- X. JS/Web 2.0

Types

- Primitifs :
 - Booléen
 - Nombre
 - Chaîne de caractères
 - Indéfini
- Objets :
 - Tableau
 - Fonction
 - Date
 - Expressions régulières
 - Nul



Chaque type primitif est un pseudo-objet : il peut se comporter comme un objet

Booléen

- Peut prendre comme valeur `true` ou `false`
- Valeurs considérées comme `false` :
 - Le booléen `false` lui-même
 - Le nombre `0`
 - Le nombre `0.0`
 - La chaîne de caractères vide `""`
 - La valeur spéciale `NaN`
 - La valeur spéciale `null`
 - La valeur spéciale `undefined`
- **Toutes** les autres valeurs sont considérées comme `true`
- Pseudo-objet : `Boolean`
- Conversion : `Boolean(val)` ← Le plus souvent pas nécessaire

Nombre

- Nombre décimal (**pas de type entier**)
- Conversion : `parseInt(val)`, `parseInt(val, base)`

```
parseInt("123"); // 123.  
parseInt("11", 2); // 3.
```

- Valeurs spéciales :
 - « Pas un nombre » (*not a number*) : NaN

- Fonction `isNaN()` :

```
let a = parseInt("hello", 10); // a = NaN.  
let b = a + 3; // b = NaN.  
let c = isNaN(b); // c = true.
```

- Infinité : `-Infinity`, `Infinity`

```
let a = -1 / 0; // a = -Infinity.  
let b = 1 / 0; // b = Infinity.
```

- Pseudo-objet : `Number`

Chaîne de caractères

- Séquence de caractères spécifiée par `"` ou `'` :

```
let str1 = "Bonjour";  
let str2 = 'Bonjour';
```

- Pseudo-objet : `String`



Indéfini vs nul

Indéfini

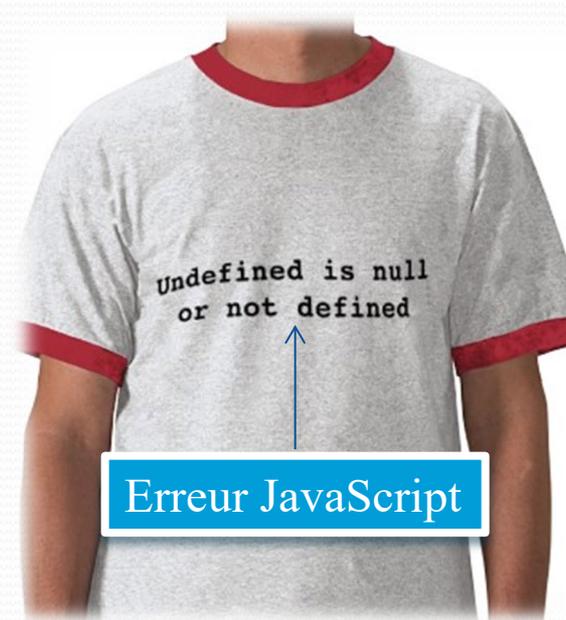
- Mot clef : `undefined`
- Type primitif
- Signification : non défini
 - Variable non déclarée
 - Variable non initialisée

```
undefined == null
```

```
undefined !== null
```

Nul

- Mot clef : `null`
- Type objet
- Signification : pas de valeur



Objet

Collection de paires nom-valeur (tableau associatif)

Syntaxe objet

- Objet vierge :

```
let obj = new Object();
```

- Accès au champ/à la méthode :

```
obj.name = "Smith";  
let name = obj.name;  
let size = obj.detail.size;  
obj.foo();
```

Syntaxe littérale

- Objet vierge :

```
let obj = {};
```

- Initialiser un objet :

```
let obj = {  
  name: "John Smith",  
  rank: 1,  
  details: {  
    color: "orange",  
    size: 12  
  }  
};
```

- Accès au champ/à la méthode :

```
obj["name"] = "Smith";  
let name = obj["name"];  
let size = obj["details"]["size"];  
obj["foo"]();
```

Tableau

Un tableau peut être indicé ou associatif et peut être multidimensionnel

- Objet spécial

- On utilise la syntaxe littérale :

```
let arr = ["monday", "tuesday", "wednesday"];  
arr[0] = "dog";
```

La création d'un tableau vierge peut aussi se faire avec la syntaxe objet

- Méthodes :

- `length()` : renvoie la valeur de l'index le plus élevé plus un
- `push(item,...)` : ajoute un/plusieurs éléments à la fin du tableau
- `pop()` : retire et renvoie le dernier élément
- `concat(item,...)` : renvoie un tableau avec les éléments ajoutés
- `slice(start, end)` : renvoie un sous-tableau
- `sort(cmpfn)` : tri (fonction de comparaison optionnelle)
- `unshift([item]...)` : insère des éléments en tête de tableau

Fonction

- Objet spécial :

```
function foo(a, b, c) {  
    // Rappel : a, b et c sont des paramètres.  
    ++a;  
    --b;  
    // Termine la fonction en retournant la valeur calculée.  
    return (a * b) / c;  
    // Le code éventuellement placé ici n'est pas exécuté.  
}  
  
let a = 5;  
let b = foo(3, a, 2); // Rappel : 3 et a sont des arguments.  
// a = 5.  
// b = 8.
```

Opérateurs

- Unaires :
 - Signe : +, -
 - Négation : !
 - Incrémentation/décrémentation (pré et post) : ++, --

```
let a = -0.5;  
++a;  
let b = --a;  
let c = !b;
```

```
var x = 2; var y = 10; var z = "5"  
document.write(eval("x * y + z + 1"))
```

Se obtiene: 2051

javascript

objeto.toString() La finalidad de este método, común

```
var meses = new Array("Enero", "Febrero", "Ma
```

Opérateurs

- Binaires :

- Arithmétiques : +, -, *, /, %
- Concaténation de chaîne : +
- Affectation : =, +=, -=, *=, /=, %=, etc.
- Comparaison : ==, >, <, >=, <=, !=, ===, !==, etc.
- Logiques : &&, ||, etc.
- Bit à bit : &, |, ^, etc.

Comparaison sur les valeurs et les types

Ou exclusif

```
let a = 5;  
let b = 10;  
let c = "a : " + a;  
let d = (a == b) && ((b % a != 0) || (b < 20));
```

Crédits

Auteur

Mickaël Martin Nevot

mmartin.nevot@gmail.com



Carte de visite électronique

Relecteur

Cours en ligne sur : www.mickael-martin-nevot.com

