Développement front end

CM3-1 : JavaScript

Mickaël Martin Nevot

V2.0.0

Développement front end

JavaScript

JavaScript ≠ Java

- Extension de fichier : `.js`
- Rappel : méthode recommandée d'utilisation de JavaScript :

```
<head>
    <script type="text/javascript" src="js/my-file.js"></script>
</head>
```

- Principales caractéristiques :
  - Interprété **coté client**
  - Langage de programmation de **script** sensible à la casse
  - Conçu pour le développement d'**applications web**
  - Langage **orienté objet non-typé**
  - Standard **ECMAScript** (comme ActionScript)
  - Pas de lecture/écriture ou d'exécution d'autres programmes

Commentaires JavaScript

- Non interprétés par le navigateur
- Visibles dans le code source
- Commentaires de type C/C++ et Shell Unix
- Exemple :

```
// Commentaire JavaScript(une seule ligne).
# Autre commentaire JavaScript (une seule ligne).
/* Autre commentaire JavaScript (une ou plusieurs lignes). */
/*
    Commentaire JavaScript
    (sur plusieurs lignes).
*/
```

Variable et constante

- Définition de type :
  - **Pas de déclaration explicite** du type d'une variable
  - Type d'une variable déterminé par le **contexte d'utilisation**
  - **Conversion automatique**
- Variable : `var`

```
var a;
var name = "John";
```

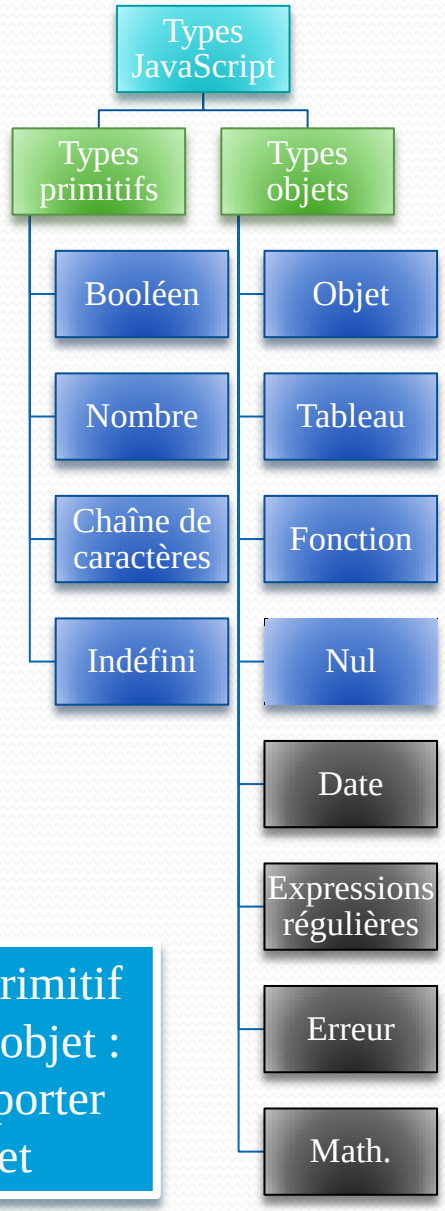
- Constante (variable non modifiable) : `const`

```
const B = 3;
B = 2;   // Erreur !
const C; // Erreur !
C = 4;   // Erreur !
```

- Portée : lexicale
  - Globale : durée de vie dans tout le script
  - Locale : durée de vie limitée au bloc de sa déclaration

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()` :

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

  - Infinité : `-Infinity`, `Infinity`

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

- Pseudo-objet : `Number`

Chaîne de caractères

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

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

- Pseudo-objet : `String`

Indéfini vs nul

Indéfini

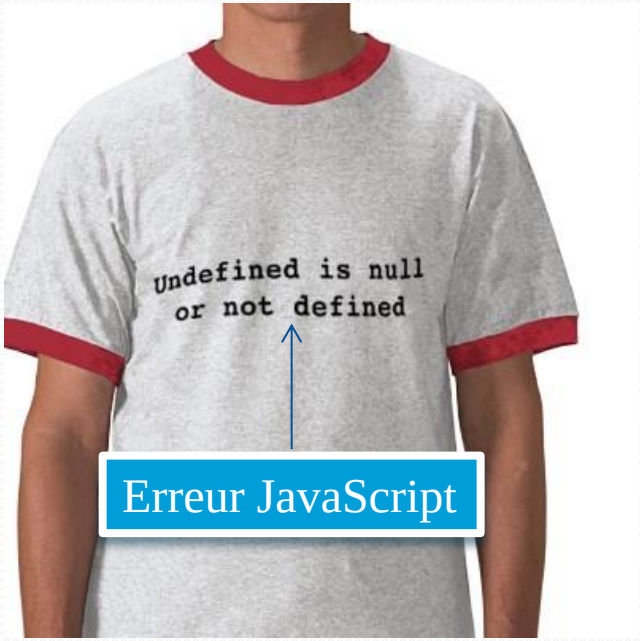
- Mot clef : `undefined`
- Type primitif
- Signification : non défini
  - Variable non déclarée
  - Variable non initialisée

```
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 :

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

- Accès au champ/à la méthode :

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

Syntaxe littérale

- Objet vierge :

```
var obj = {};
```

- Initialiser un objet :

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

- Accès au champ/à la méthode :

```
obj["name"] = "Smith";
var name = obj["name"];
var 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 :

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

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

- 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.
}

var a = 5;
var 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) : ++, −−

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

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

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

Structures conditionnelles

Structure conditionnelle :

```
if (a == 0) {
    ++a;
} else if (a == 1) {
    --a;
} else {
    a = 0;
}
```

Branchement conditionnel :

```
switch(i) {
  case "jambon":
      alert("salé");
      break;
  case "tarte":
  case "bonbon":
  case "biscuit":
      alert("sucré");
      break;
  default:
      alert("autre");
  }
```

Boucles

- Boucle `for` :

```
var res = "";
for (var i = 0 ; i < 10 ; ++i) {
   res += i + ", ";
}
alert(res);
// 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .
```

- Boucle `for ... in` :

```
var res = "";
var arr = [1, 2, 3, 4, 5];
for (var val in arr) {
    res += (val * 2) + ", ";
}
alert(res);
// 2, 4, 6, 8, 10, .
```

- Boucle `while` :

De 0 à n fois

```
var i = 0;
while(i < 0) {
   ++i;
}
alert(i);
// 0.
```

- Boucle `do ... while` :

De 1 à n fois

```
var i = 0;
do {
   ++i;
} while (i < 0);
alert(i);
// 1.
```

Break/continue

- Sortir d'une boucle ou d'un `switch ... case :break`

```
while(1) {
   if (i == 10) {
      break;
   }
   ++i;
}
// Exemple de break dans un switch sur la diapositive de structures conditionnelles.
```

- « Sauter » à l'itération suivante d'une boucle : `continue`

```
var a = 0;
for  (var i = 0 ; i < 10 ; ++i) {
   if (i % 2 == 0) {
      continue;
   }
   a += i;
}
// ?
```

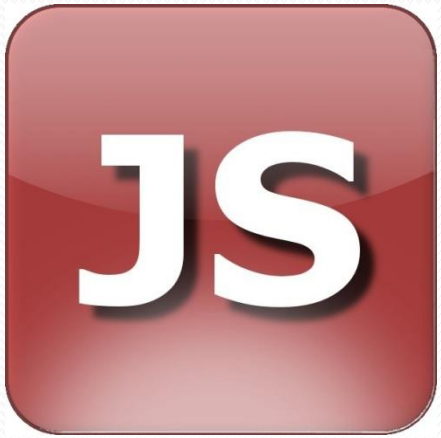
Instruction

- Se termine par un `;`
- Type d'instruction :
  - Déclaration : `var a;`
  - Affectation : `a = 10;`
  - Appel de fonction : `myFunction();`
  - Instruction conditionnelle : `if (a == b) … ;`
  - Instruction vide : `;`
  - Bloc (d'instruction) :

```
{
    instruction_1
    instruction_2
    …
}
```

Bonne utilisation

- Apport d'interactivité, de **dynamisme**
- Amélioration l'**ergonomie**
- Fonctionnalités **non critiques**
- Fonctionnalité **non sécurisées**
- Utilisation modérée



Avantages/inconvénients

- Avantages :
  - Soulage le trafic réseau
  - Bonne **réactivité**
  - **Interactivité** des sites web « statiques » (orientés client)
- Inconvénients :
  - Non pris en charge en mode dégradé
  - Problème de **compatibilité** entre navigateurs
  - Pénétration : environ 10 % de navigateurs incompatibles
  - **Insécurité** des données
  - Peut alourdir le traitement côté client
  - **Délicat à déboguer**

Anciennes versions d'Internet Explorer

Devenez un super héros

- http://superherojs.com/

Aller plus loin

- Changement du contexte de résolution : `with`
- Date
- Expression régulière
- Héritage
- Exceptions
- Fonction interne
- Prototypage
- Fermeture
- Ramasse-miettes
- Fuites mémoire
- Fonctions `apply`, `call` et `arguments`

Liens

- Document électronique :
  - http://www.alsacreations.com/tuto/liste/5-javascript.html
- Documents classiques :
  - Simon Willison. *Une réintroduction en JavaScript.*
  - Sébastien Mavromatis. *Langages de l'Internet.*
  - Vincent Riale. *Initiation au langage JavaScript.*

Crédits

Auteur

Mickaël Martin Nevot
mmartin.nevot@gmail.com

Carte de visite électronique

Relecteur

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