ECMAScript 2015

ECMAScript 5

ES5

- Premiers efforts d'harmonisation des navigateurs après 10 ans de laissé aller
- Prépare l'arrivée d'ES6/2015
- Introduit une approche plus fonctionnelle dans les APIs
- Rend le JavaScript moins permissif et plus logique

Changements syntaxiques

Changements syntaxiques

- Les virgules après le dernier élément sont autorisée
- Nan, infinity et undefined sont des constantes
- parseInt() est par défaut en base 10
- Les mots clefs du langage sont interdits en nom de propriétés

Strict mode

Le mode strict

- Le fameux `'use strict';`
- Son impact :
  - La gestion des erreurs est moins permissive
  - Interdit les variables globales implicites dans les fonctions
  - This n'est plus rattaché à l'objet global dans les fonctions
  - Interdit l'utilisation de with
  - Restreint eval
  - Rend l'utilisation d'arguments plus logique
  - Interdit les doublons dans les propriétés des objets littéraux et dans les paramètres de fonctions

➔ Plus logique et plus robuste

New APIs

Arrays - search

- Index d'un élément :

```
Array.prototype.indexOf(element)
```

- Dernier index d'un élément

```
Array.prototype.lastIndexOf(element)
```

```
var myArray = ['val1', 'val2', 'val1'];

myArray.indexOf('toto'); // -1
myArray.indexOf('val1'); // 0
myArray.indexOf('val2'); // 1
myArray.lastIndexOf('val1'); // 2
```

Arrays - iteration

- Itérer sur tous les éléments d'un tableau

```
Array.prototype.forEach(function)
```

- Check que TOUS les éléments d'un tableau respectent une condition (si la fonction retourne false une fois, every() retourne false)

```
Array.prototype.every(function)
```

- Check que au moins un des élements respecte une condition (si la fonction retourne true une fois, some() retourne true)

```
Array.prototype.some(function)
```

Arrays – itération - exemples

```
var myArray = ['val1', 'val2', 'val1'];

myArray.forEach(function(value) {
  console.log(value);
});

myArray.every(function(value) {
  return value != '';
});

myArray.some(function(value) {
  return value == 'val1';
});
```

Arrays - transformations

- Transforme un tableau en un autre tableau

```
Array.prototype.map(function)
```

- Transforme un tableau en une valeur

```
Array.prototype.reduce(function)
Array.prototype.reduceRight(function)
```

- Retourne un nouveau tableau filtré

```
Array.prototype.filter(function)
```

Arrays – transformations - exemples

```
var myArray = ['val1', 'val2', 'val3'];
myArray.map(function(value, index, array) {
  return value.toUpperCase();
}); // ['VAL1','VAL2','VAL3']

myArray.reduce(function(initialVal, value, index, array)
{
  return initialValue + ' ' + value;
},''); // " val1 val2 val3"

myArray.reduceRight(function(initialValue, value, index,
array) {
  return initialValue + ' ' + value;
},''); // " val3 val2 val1"

myArray.filter(function(value, index, array) {
  return value == 'val1';
}); // ['val1']
```

Arrays - misc

- Check si une variable est un tableau

```
Array.isArray(tableau)
```

```
Array.isArray([]); // true
Array.isArray(['val1','val2']); // true
Array.isArray(129); // false
```

Fonctions

- Créé une nouvelle dans un autre contexte d'execution

```
Function.prototype.bind(context, param1, param2,..., paramN)
```

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

var sum2 = sum.bind(undefined,2);

function showThis() {
  console.log(this);
}

var globalShowThis = showThis.bind(window);
var localShowThis = showThis.bind(this);
```

Date

- Renvois le timestamp courant

```
Date.now(); // current timestamp
```

- Convertis la date en date ISO

```
Date.prototype.toISOString()
```

```
var today = new Date("05 October 2011 14:48 UTC");
today.toISOString(); // "2011-10-05T14:48:00.000Z"
```

Chaines

- Supprime les blancs

```
String.prototype.trim()
```

```
var myStr = "   test   ";

myStr.trim(); // "test"
```

Objets - instanciations

- Instancier un objet héritant d'une classe parente

```
Object.create(parent);
```

- Configurer les propriétés

```
var child = Object.create(parent, {
  dataDescriptor: {
    value: "value",
    writable: true,
    enumerable: true
  },
  accessorDesc: {
    get: function () { return accessorDesc; },
    set: function (val) { accessorDesc = val; },
    configurable: true
  }
});
```

Objets - protections

- Empêcher un objet d'avoir de nouvelles propriétés

```
Object.preventExtensions(obj);
```

- Empêcher un objet d'être reconfiguré

```
Object.seal(obj);
```

- Rendre un objet immutable

```
Object.freeze(obj);
```

- Vérifier l'état

```
Object.isExtensible(obj);
Object.isSealed(obj);
Object.isFrozen(obj);
```

Objets - introspection

- Récupérer la liste des propriétés enummérables propres à l'objet

```
Object.keys(obj);
```

- Récupérer la liste des propriétés propres à l'objet

```
Object.getOwnPropertyNames(obj);
```

```
var myArray = ['val1','val2'];

Object.keys(myArray); // [0,1]
Object.getOwnPropertyNames(myArray); // [0,1,length]
```