Soyons prêts pour ECMAScript 6

{{ softshake }} 2014
http://soft-shake.ch
@GENEVE

Philippe Charrière | **@k33g_org**

Resp. Technique

SQLI Lyon
ENTERPRISE

plutôt typé front end
mais 💙 donner son avis sur le back

Contexte

un framework MVC (MV*) avec des fonctionnalités d'ES6 "Skeleton"

codes sources ici:
https://github.com/k33g/es-6-prez-softshake

Feuille de route

models & collections & observables

views

“synchronisation”

router

démo Polymer

démo Node.js

Fonctionnement

pour chaque item
un peu de théorie
le code résultant + exécution

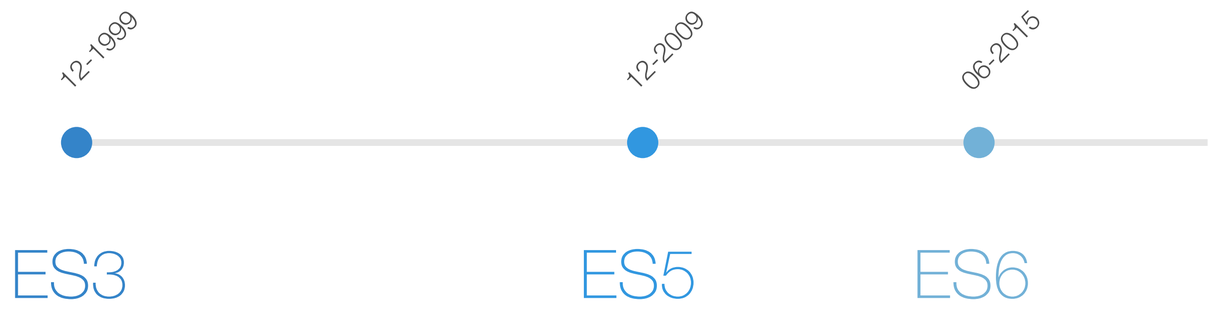
ECMAScript **6** ?

ES6

ES6 = la future version de javascript

ECMAScript c’est le nom de la version standardisée

ES6 > fin 2014 > publication mi-2015



ES6 aujourd’hui c’est possible

ES6 aujourd'hui c'est possible

> Projet **Traceur**

https://github.com/google/traceur-compiler

Installer :

```
sudo npm install traceur -g
```

Transpiler :

```
traceur --out out/Human.js --script Human-es6.js
```

ES6 aujourd'hui c'est possible

```
<html>
  <head>
    <script src="bin/traceur-runtime.js"></script>
    <script src="out/Human.js"></script>
  </head>
  <body>
  </body>
</html>
```

ES6 aujourd'hui c'est possible

```
<html>
  <head>
    <script src="bin/traceur.js"></script>
  </head>
  <body>
   <script>
    System.import("Human-es6.js");
   </script>
  </body>
</html>
```

ES6 aujourd'hui c'est possible

> Projet **Traceur**

> **6to5**

https://github.com/sebmck/6to5

> **es6-transpiler**

https://github.com/termi/es6-transpiler

> …

Un “petit bout” d’#ES6

Partie 1

les classes & les modules

& d’autres petites choses…

class

```
class Dog {
  /* mot-clé constructor + valeurs par défaut */
  constructor (name="cookie") {
    /* propriétés définies dans le constructeur */
    this.name = name;
  }
  wouaf () { /* pas de mot-clé function */
    console.log(this.name + ": wouaf! wouaf!");
  }
}

let wolf = new Dog();
wolf.wouaf()
```

extends

```
class Animal {
  constructor(name) {
    this.name = name;
  }
}

class Dog extends Animal {
  constructor (name="cookie") {
    /* on appelle le constructeur de la classe mère */
    super(name)
  }
  wouaf () {
    console.log(this.name + ": wouaf! wouaf!");
  }
}
```

export - import

```
/* Animal.js */

class Animal {
  constructor(name) {
    this.name = name;
  }
}
export default Animal;
```

```
/* Dog.js */

import Animal from './Animal';
/* pas d'extension .js */

class Dog extends Animal {
  constructor (name="cookie") {
    super(name)
  }
  wouaf () {
    console.log(this.name + ": wouaf! wouaf!");
  }
}
export default Dog;
```

```
/* main.js */

import Dog from './Dog'
let wolf = new Dog();
wolf.wouaf()
```

=>

```
/* Avant */
var sayHello = function(name) { return "hello " + name; }

/* Après */
var sayHello = (name) => "hello " + name
// ou var sayHello = (name) => { return "hello " + name; }

sayHello("Bob Morane")
```

=> !“newable”, !arguments

REST parameters

```
var sayHello = (...people) =>
    people.forEach(
      (somebody) => console.log("Hello", somebody)
    );

sayHello("Bob Morane", "John Doe", "Jane Doe");
```

=> & lexical this binding

```
/* Avant */

function Animal(friends) {
  this.friends = friends;
  this.hello = function(friend) {
    console.log("hello " + friend);
  }
  this.helloAll = function() {
    this.friends.forEach(function(friend) {
      this.hello(friend); /* error */
    });
  }
}

var wolf = new Animal(["rox", "rookie"]);
wolf.helloAll();
```

=> & lexical this binding

```
/* Avant */

function Animal(friends) {
  this.friends = friends;
  this.hello = function(friend) {
    console.log("hello " + friend);
  }
  this.helloAll = function() {
    this.friends.forEach(function(friend) {
      this.hello(friend);
    }.bind(this)); /* ou var that = this */
  }
}

var wolf = new Animal(["rox", "rookie"]);
wolf.helloAll();
```

=> & lexical this binding

```
/* Après */

class Animal {
  constructor (friends=[]) {
    this.friends = friends;
  }

  hello(friend) { console.log("hello " + friend); }

  helloAll() {
    this.friends.forEach((friend) => this.hello(friend));
  }
}
```

let versus var

```
let bob = {
  firstName:"Bob", lastName:"Morane"
}

let bob = { foo:"foo" }
/* Duplicate declaration, bob */
```

“Transpilation” … “à la volée”

```
<script src="node_modules/traceur/bin/traceur.js"></script>

<script>
  traceur.options.experimental = true;
</script>

<script>
  System.import('js/main').catch(function (e) {console.error(e);});
</script>
```

Partie 1: début d’un framework MVC

modèles & collections

puis: observables

cf. code source

Partie 2

Strings, “mixins”, …

Template strings (``backtick)

```
let firstName = "Bob", lastName = “Morane";

console.log(`Hello I'm ${firstName} ${lastName}`);

// Hello I'm Bob Morane
```

Multiline strings

```
let firstName = "Bob", lastName = "Morane";

console.log(`
  Hello I'm
  ${firstName}
  ${lastName}
`
);
/*
  Hello I'm
  Bob
  Morane
*/
```

Object.assign

```
let tonyStark = {
  firstName:"Tony", lastName:"Stark"
};

let armorAbilities = {
  fly:() => console.log("I'm flying")
};

Object.assign(tonyStark, armorAbilities);

tonyStark.fly(); // I'm flying
```

Array.from

```
/* Avant */
var items = [].slice.apply(document.querySelectorAll("h1"));

// var items = Array.prototype.slice.apply(document.querySelectorAll("h1"));

items.forEach(function(item) { item.innerHTML = "Hello"; });

/* Maintenant */

Array.from(document.querySelectorAll(“h1"))
  .forEach((item) => item.innerHTML = "Hello")
```

Partie 2: on complète le framework
les vues

cf. code source

Partie 3
Les promises

Promises

```
let doSomeThing = new Promise((resolve, reject) => {

  // faites quelque chose (d'asynchrone par ex.)

  let allisfine = true; // essayez avec false

  if (allisfine) {
    resolve("Hello World!");
  }
  else {
    reject(Error("Ouch"));
  }
});

doSomeThing
  .then((data) => { console.log(data); })
  .catch((err) => { console.log(err); });
```

http://www.html5rocks.com/en/tutorials/es6/promises

Partie 3: on complète le framework **“discuter” avec le serveur**

cf. code source

Partie 4
Maps

Map

```
let map = new Map();

map.set("one",{firstName:"John", lastName:"Doe"});
map.set("two",{firstName:"Jane", lastName:"Doe"});

console.log(map.has("one"));  // true
console.log(map.get("one"));  // Object {firstName: "John", lastName: "Doe"}
console.log(map.size);        // 2
```

Parcourir une Map

```
for (let key of map.keys()) {
  console.log("Key: %s", key);
}
/* Key: one, Key: two */

for (let value of map.values()) {
  console.log("Value: %s %s", value.firstName, value.lastName);
}
/* Value: John Doe, Value: Jane Doe */

for (let item of map) {
  console.log("Key: %s, Value: %s", item[0], item[1].firstName, item[1].lastName);
}
/* Key: one, Value: John Doe, Key: two, Value: Jane Doe */
```

… et aussi

```
let myOtherMap = new Map([
  ["one",{firstName:"John", lastName:"Doe"}],
  ["two",{firstName:"Jane", lastName:"Doe"}],
  ["three",{firstName:"Bob", lastName:"Morane"}]
]);

myOtherMap.delete("three")

myOtherMap.forEach((item)=>{
  console.log(item)
})
/*
 Object {firstName: "John", lastName: "Doe"}
 Object {firstName: "Jane", lastName: "Doe"}
*/
```

Le saviez-vous ?

Vos classes peuvent hériter des types javascript

… mais pas toujours

Partie 4: on complète le framework **Router** (rudimentaire)

cf. code source

Démo
Utilisation avec Polymer

Démo
Utilisation avec Node.js

Faut-il utiliser ES6 aujourd’hui ? & pourquoi ?

Angular 2 avec Traceur

ECMAScript 6+ (design)

All code in Angular 2 is already being written in ES6. As ES6 doesn't run in browsers today, we're using the Traceur compiler to generate the nice ES5 that runs everywhere. We're working with the Traceur team to build support for a few extensions like annotations and assertions.

Ember (next) : les modules avec ES6 Module Transpiler

Backbone est déjà ES6 compliant : https://github.com/addyosmani/todomvc-backbone-es6/blob/gh-pages/js/todo-app.js

Qui suivre ?

Addy Osmani (@addyosmani)

https://github.com/addyosmani/es6-tools

Axel Rauschmayer (@rauschma)

http://www.2ality.com/search/label/esnext

Nicholas Zakas (@slicknet)

https://github.com/nzakas/understandinges6

Merci à vous + ?