Alexis Ducerf – alexis.ducerf@DeerCoders.com

Histoire du JavaScript

- Créé en 1995 par Brendan Eich
- Standard ECMAScript
- Actuellement sur ECMAScript 7 (depuis juin 2016)
- Typage faible & dynamique
- Popularisé notamment grâce au moteur V8 de Chrome qui sera utilisé par la suite par NodeJS

Histoire de NodeJS

- Créé en 2009 par Ryan Lienhart Dahl
- Basé sur le moteur V8 de Google
- Langage de bas niveau
- Développement et maintenance fait par Joyent

Les points forts

- Asynchrone
- Multiplateforme
- Création d'applications lourdes (Electron)
- Serveur web intégré
- Monté en charge
- Événementiel
- Gestion des streams
- Communauté

Les variables

- La déclaration d'une variable se fait via le mot clé *var* si le mot clé n'est pas précisé la variable sera globale
- Depuis ES6 il est possible de déclarer des variables via *let* et *const*, *let* permet de créer une variable qui ne sera valable que dans le contexte actuel et *const* sera une constante.
- Les variables sont sensibles à la casse.
- Une variable déclaré dans une fonction ne sera pas accessible depuis l'extérieur. Exemple :

```
var myVar = 2;

function myFunction () {
  var myVar = 3;
  console.info(myVar); // 3
}

myFunction();

console.info(myVar); // 2
```

Chaînes de caractère

Il est possible depuis ES6 d'avoir des gabarits, ainsi :

```
var a = "tototo " + maVariable +
"tatata " +
"titititit ";
```

Devient :

```
var a = `tototo ${maVariable}
tatata
titititit `;
```

Les fonctions

En JavaScript les fonctions peuvent êtres nommées ou non.

Exemple :

```
var myFunction = function () {
  console.log('coucou');
};

function myFunction2 () {
  console.log('coucou 2');
}

myFunction();
myFunction2();
```

Arrow functions

Le JavaScript utilise énormément les fonctions pour gérer les callbacks, c'est-à-dire les fonctions exécutées lorsque le travail est fini. Depuis ES6 il est possible de simplifier cela.

```
setTimeout(function(){
  console.log('coucou');
}, 1000);
```

Devient :

```
setTimeout(_ => console.log('coucou'), 1000);
```

Les boucles

- while

```
var n = 0;

while (n < 3) {
  n++;
}
```

- for

```
for (let i = 0; i < 9; i++) {
    console.log(i);
}
```

Les boucles

- do while

```
var i = 0;
do {
   i += 1;
   console.log(i);
} while (i < 5);
```

- foreach

```
[1, 2, 3].forEach((val, index, array) => {
  console.log(val, index, array);
});
```

Les conditions

- If/else if/else

```
if (i === 2) {
  console.log(2);
} else if (i === 3) {
  console.log(3);
} else {
  console.log(0);
}
```

- Switch

```
switch(myVar){
  case 'un':
    console.log('un');
    break;
  case 'deux':
    console.log('deux');
    break;
  default:
    console.log('none');
}
```

- Ternaire

```
var value = (myVar)?'true':'false';
```

Les tableaux

- Pour déclarer un tableau il faut passer par *[]*, on ne passera pas la déclaration *new Array()* qui est dépréciée
- Pour rajouter des éléments dans le tableau on passera par la méthode array.push()
- Pour afficher un élément il faudra passer par une boucle ou par un accès direct *via* par exemple array[0]
- Il existe un grand nombre de méthodes pour interagir avec un tableau tel que *split(), shift(), pop(), join(), toString()*

Les objets

La création d'un objet peut être faite de quatre façons :

- *Via* un JSON (JavaScript Object Notation)

```
var maVoiture = {
  fabricant: "Ford",
  modèle: "Mustang",
  année: "1969"
}
```

- Via la création d'un Object()

```
var maVoiture = new Object();
maVoiture.fabricant = "Ford";
maVoiture.modèle = "Mustang";
maVoiture.année = 1969;
```

Les objets

- *Via* un constructeur :

```
function Voiture(fabricant, modèle, année) {
  this.fabricant = fabricant;
  this.modèle = modèle;
  this.année = année;
}

var maVoiture = new Voiture('Ford','Mustang','1969');
```

- *Via* une classe

```
class Voiture {
  constructor(fabricant, modèle, année) {
    this.fabricant = fabricant;
    this.modèle = modèle;
    this.année = année;
  }
}

var maVoiture = new Voiture('Ford', 'Mustang', '1969');
```

Pour accéder aux informations il est possible de faire ensuite *mavoiture.fabricant*

Le prototypage

JavaScript est un langage objet à prototype ce qui permet de modifier un objet *via* un constructeur et non *via* une instance de classe.

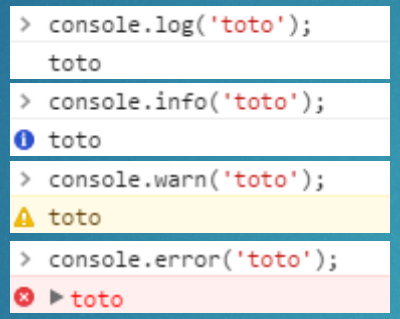
Il est donc possible par exemple de rajouter une fonction qui sera utilisable par les chaînes de caractères. Exemple :

```
String.prototype.star = function () {
  return "** " + this + " **";
};

console.log("Coucou".star()); // "** Coucou **"
```

La console

- La console permet de faire ressortir des informations au développeur. Il existe par exemple plusieurs niveaux de logs :
  - *console.log();*
  - *console.info();*
  - *console.warn();*
  - *console.error();*



- Il est aussi possible de faire des benchmarks directement via la console grâce à *console.time()* et *console.timeend()*
- Il est possible d'afficher toute la stacktrace grâce à *console.trace()*
- Il est aussi possible de faire des tests grâce à *console.assert()*

Accès fichiers

- Tout est fait grâce à la bibliothèque fs
- Accès aux fichiers en synchrone ou asynchrone
- Possibilité de passer par des streams

- Gestionnaire de paquets
- Utilisation d'un package.json
- Mise à jour de paquets
- Lancement de tâches

Les modules utiles

- Babel pour passer le code en compatibilité es5
- Browserify pour utiliser des modules NodeJS directement dans le navigateur
- Gulp / Grunt pour l'automatisation de tâches
- Mongoose ODM pour MongoDB
- Sequelize ORM pour les BDD SQL
- Async pour rendre asynchrone des fonctions qui ne le sont pas
- Request outil de requête HTTP
- PM2 outil de mise en ligne

Les modules utiles

- Nsp pour vérifier les failles de sécurité *via* le package.json
- Bower pour importer les bibliothèques en front
- Mocha framework de test
- Cheerio outil pour le scrap
- Validator outil de validation de données
- Socket.io framework pour gérer les websockets
- Helmet outil de sécurisation du code *via* les headers HTTP
- Passport pour gérer les différentes authentifications (local, FB, Twitter, Google …)

Les modules utiles (express)

- Csurf outil pour bloquer les failles CSRF
- Compression permet de compresser en deflate/gzip les trames
- Session permet de gérer les sessions utilisateur
- Multer permet de gérer les multipart/form-data dans les formulaires
- Body-parser permet de gérer les formulaires

Les modules utiles (express)

- Cors permet de gérer les Cross Origin Resource Sharing
- Morgan logger pour les requêtes HTTP
- Method-override permet de gérer les verbes tel que PUT et DELETE
- Serve-favicon permet d'ajouter un favicon au site

Un code asynchrone plus propre

- AsyncJS
- Async / Await (ES7)
- Promises (bluebird si pas de compatibilité)

Express

- Micro framework
- Express Generator
- Middlewares (compatibles avec Connect)
- REST
- Communauté

Les middlewares

Un middleware se place entre l'envoi de la requête et l'envoi de la réponse. Il permet par exemple de :

- Compiler à la volée du sass en css
- Tester si la personne est connectée
- Activer des sécurités supplémentaires
- …

Gestion de BDD

- ORM pour une base de donnnées SQL
- ODM pour une base de données NoSQL (document)
- Dans notre cas nous allons utiliser Mongoose pour MongoDB et Sequelize pour SQLite

WebSockets

NodeJS gère nativement les websockets, certains frameworks comme Socket.IO permettent de développer rapidement des applications en temps réel.

Test unitaires

- Mocha
- Chai
- Chai HTTP / Frisby

- Mise en production facile grâce à PM2
- Possibilité de créer plusieurs forks facilement
- Listing simple des processus

```
[tknew:~/Unitech/pm2] master(+84/-121)+* ± pm2 list
PM2 Process listing
```

App Name	id	mode	PID	status	Restarted	Uptime	memory	err logs
bashscript.sh	6	fork	8278	online	0	10s	1.379 MB	/home/tknew/.pm2/logs/bashscript.sh-err.log
checker	5	cluster	0	stopped	0	2m	0 B	/home/tknew/.pm2/logs/checker-err.log
interface-api	3	cluster	7526	online	0	3m	15.445 MB	/home/tknew/.pm2/logs/interface-api-err.log
interface-api	2	cluster	7517	online	0	3m	15.453 MB	/home/tknew/.pm2/logs/interface-api-err.log
interface-api	1	cluster	7512	online	0	3m	15.449 MB	/home/tknew/.pm2/logs/interface-api-err.log
interface-api	0	cluster	7507	online	0	3m	15.449 MB	/home/tknew/.pm2/logs/interface-api-err.log

Un peu de lecture en plus

- https://openclassrooms.com/courses/des-applications-ultra-rapides-avec-node-js
- http://nodejs.developpez.com/tutoriels/javascript/node-js-livre-debutant/
- http://www.tutorialspoint.com/nodejs/
- https://www.grafikart.fr/tutoriels/nodejs
- https://scotch.io/tag/node-js