Javascript

console.log(“Hello World !”)

Javascript en quelques mots

- Langage interprété
- Orienté objet, impératif, fonctionnel
- Créé le 4 decembre 1995
- Standardisé sous le nom d'ECMAScript
- N'a **AUCUN** rapport avec Java !!!

Pourquoi utiliser Javascript?

- Simple à utiliser/apprendre
- Plein de concepts sympas (fonctionnel, asynchrone)
- Populaire (1er du classement Stackoverflow Survey 2019)
  => Très grosse communauté
- Parce que c'est moi qui décide

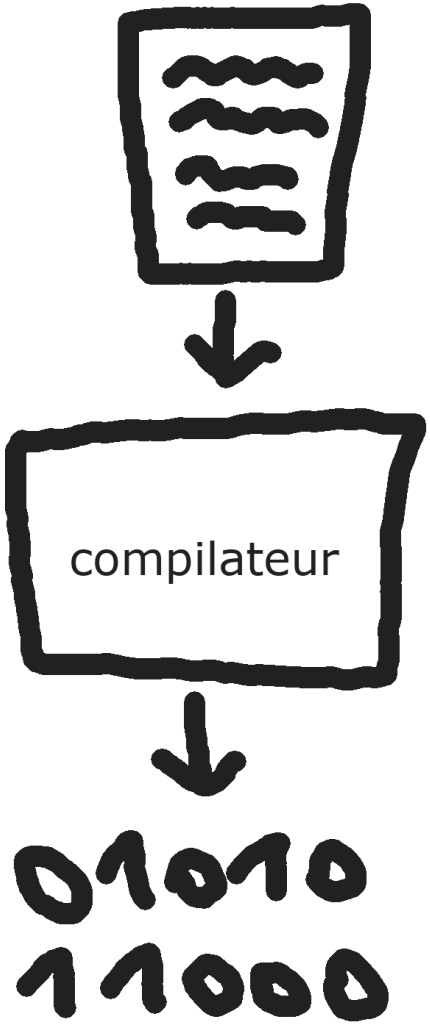
l'ECMAScript

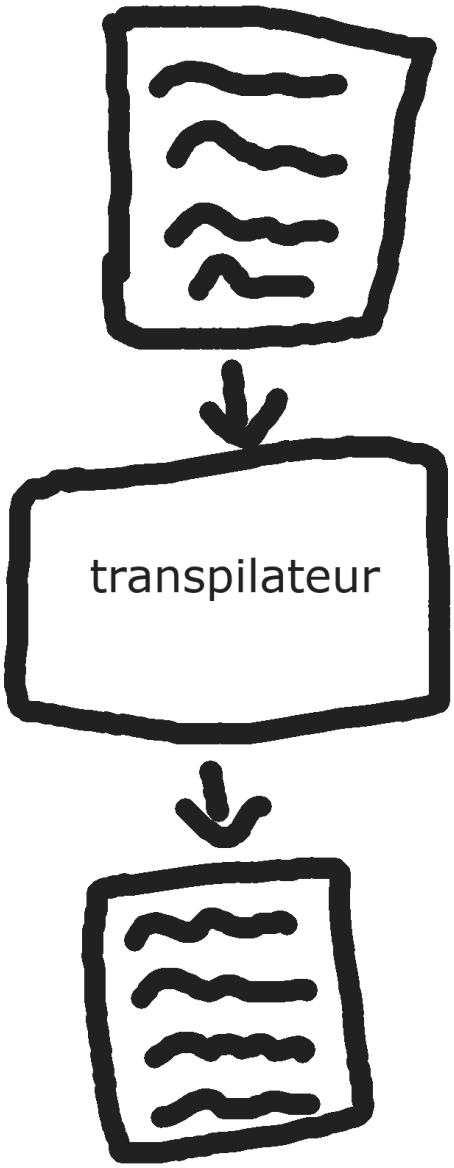
- Avant ES5, une version tous les 3-5ans
- A partir d'ES6 (ES2015), une version tous les ans
- Tous les navigateurs modernes supportent ES5
- Quelques navigateurs supportent ES6
- Aujourd'hui ES2019

Comment on utilise les dernières versions de Javascript ?

les transpileurs

- Le plus utilisé est BabelJS
- Permet de transformer un langage en un autre
- Ne pas confondre avec un compilateur
- ES2019 => ES5, plus de soucis de compatibilité !
- Les polyfills permettent également une meilleure rétrocompatibilité


compilateur
01010
11000


transpilateur

Quelques langages

- TypeScript
- ReasonML
- CoffeeScript
- et pleins d’autres…

Et ça ressemble à quoi ?

Les variables

```
// Déclarer une variable
let var1 = "toto"
let var2 = 2
let var3 = 3.5

var1 = 5
var1 = "tutu"

// Déclarer une constante
const const1 = "titi"
// const1 = "tata" /!\ IMPOSSIBLE
```

⚠️ Le mot-clé `var` a été déprécié.

Les conditions

```
const i = 0

if( i == 0 ) {
  console.log("i == 0")
} else {
  console.log("i != 0")
}

i == "0" // true
i == 0 // true
i === "0" // false
i === 0 // true
```

Les boucles

```
let i
while ( i < 10 ) {
    console.log(i)
    i++
}

for ( let j = 0; j < 10; j++ ) {
    console.log(j)
}
```

Les fonctions

```
function add ( val1, val2 ) {
    return val1 + val2
}
console.log(add(1, 2))  // 3

const add2 = function (val1, val2) {
    return val1 + val2
}
console.log(add2(1, 2)) // 3

const add3 = (val1, val2) => {
    return val1 + val2
}
console.log(add3(1, 2)) // 3

const add4 = (val1, val2) =>  val1 + val2
console.log(add4(1, 2)) // 3

const add5 = add4
```

Les structures de données

```
const tableau = [1, 2, "toto", "tutu", 3.4]
console.log(tableau.length)     // 5
console.log(tableau[0])         // 1

const dictionnaire = {tutu: "toto", titi: 1, toto: "tutu"}
console.log(dictionnaire[toto]) // "tutu"
console.log(dictionnaire.toto)  // "tutu"
console.log(dictionnaire.titi)  // 1

const collection = [ {toto: 3, tutu: 2}, {toto: 5, tutu: 2}, {toto: 7, tutu: 1}]
console.log(collection[0].toto) // 3
```

Quelques méthodes utiles

```
const users = [{name: "toto", age: 18}, {name: "titi", age: 22}, {name: "tutu", ag
users.forEach(user => console.log(user.name))
// toto titi tutu

const uppercaseUsers = users.map(user => user.name.toUpperCase())
// ["TOTO", "TITI", "TUTU"]

const majorUsers = users.filter(user => user.age >= 18)
// [{name: "toto", age: 18}, {name: "titi", age: 22}]

const toto = users.find(user => user.name === "toto")
// {name: "toto", age: 18}
```

Mais Javascript n’est pas fait pour être exécuté sur un navigateur ?!

Nodes JS

- Interpréteur Javascript (machine virtuelle)
- Utilise le V8 de Google créé en 2009
- est livré avec `npm` (Node Package Manager)

Comment ça marche ?

```
# Lancer un script
node fichier.js

# Initialiser un projet
npm init

# Installer une dépendance
npm install dependance

# Lancer un script
npm run script
```

package.json

```
{
  "name": "nom",
  "description": "description",
  "version": "0.1.0",
  "scripts": {
    "start": "echo hello world",
  },
  "dependencies": {
    "hello-world": "^0.3.2"
  },
}
```