Javascript

JavaScript everywhere

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 JavaScript?

- Code côté client
- Code côté serveur (NodeJS)
- Scripts (NodeJS)
- Application mobiles (React Native, Cordova, ...)
- Application desktop (Electron, React Native)
- IOT

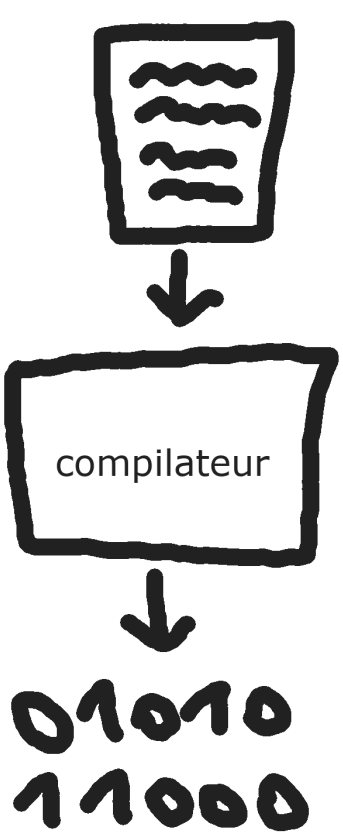
l'ECMAScript

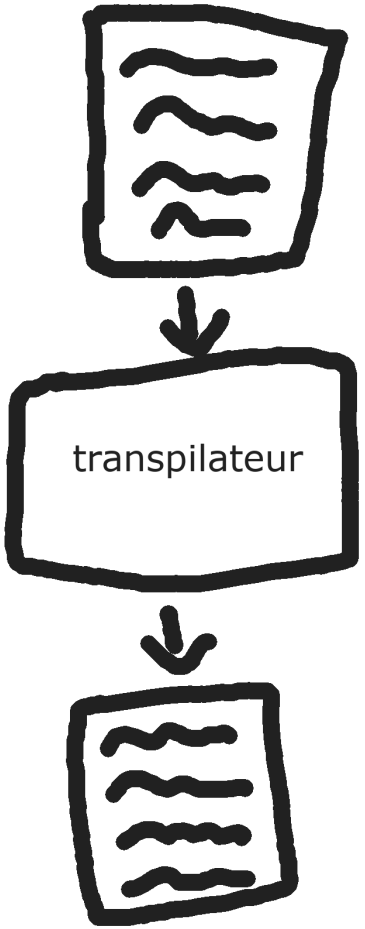
- Avant ES5, une version tous les 3-5ans
- Tous les navigateurs modernes supportent ES5
- A partir d'ES6 (ES2015), une version tous les ans
- Quelques navigateurs supportent ES6
- Aujourd'hui ES2018

Comment utiliser les dernières versions de JS ?

les transpilleurs !

- Le plus utilisé est BabelJS
- Permet de transformer un langage en un autre
- Ne pas confondre avec un compilateur
- ES2018 => ES5, plus de soucis de compatibilité !
- Les polyfills permettent également une meilleure rétrocompatibilité


compilateur
01010
11000

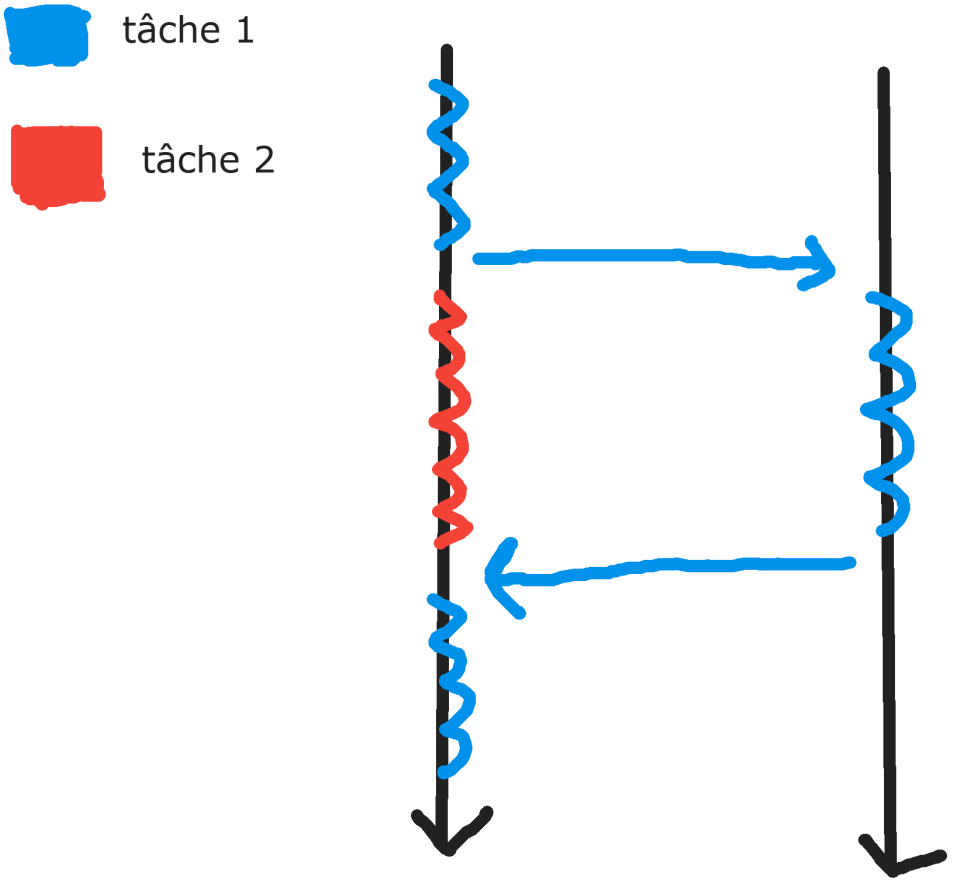

transpilateur

Quelques langages

- TypeScript
- ReasonML
- CoffeeScript
- Kotlin
- ...

L'asynchronicité et les promesses

- Spécificité de JS
- Un seul thread
- Ne pas bloquer l'interface utilisateur
- Appels réseaux, accès disque, ...


tâche 1
tâche 2

La syntaxe

```
// 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

// /!\ ne plus utiliser var
// var var4 = "tutu"
```

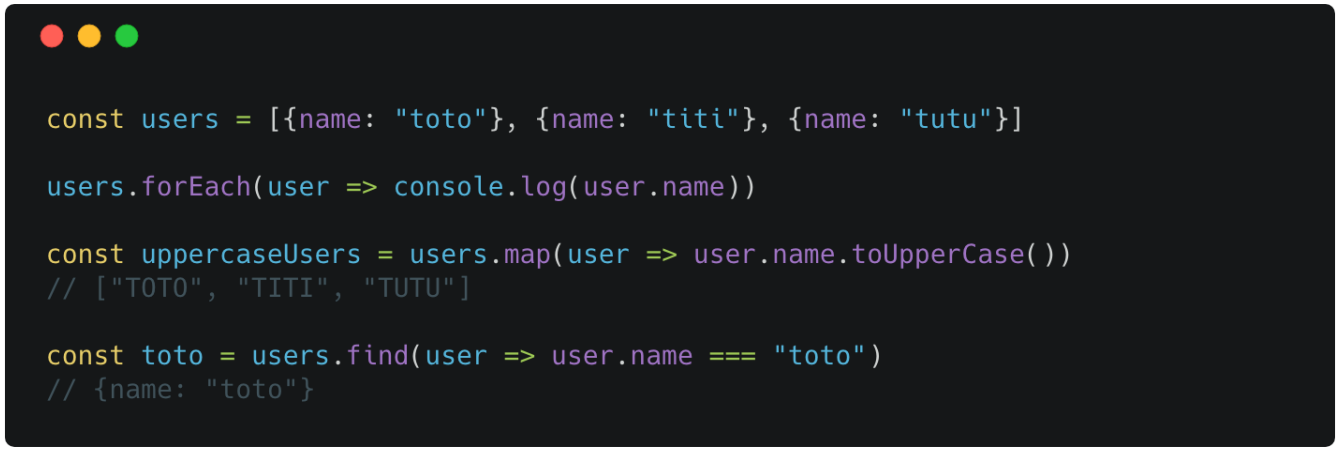
```
const i = 0

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

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

// En règle générale, il faut utiliser l'égalité stricte
```

```
let i

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

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

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

const dictionnaire = { clé1: "valeur1", clé2: "valeur2", toto: 3, tutu: "toto" }
console.log(dictionnaire["toto"])   // 3
console.log(dictionnaire.toto)      // 3
console.log(dictionnaire.titi)      // undefined

const collection = [ { toto: 3, tutu: 2 }, { toto: 1, tutu: 5 }, { toto: 4, tutu: 1 } ]
console.log(collection[0].toto)     // 3
```

```
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
console.log(add5(1,2))  // 3
```

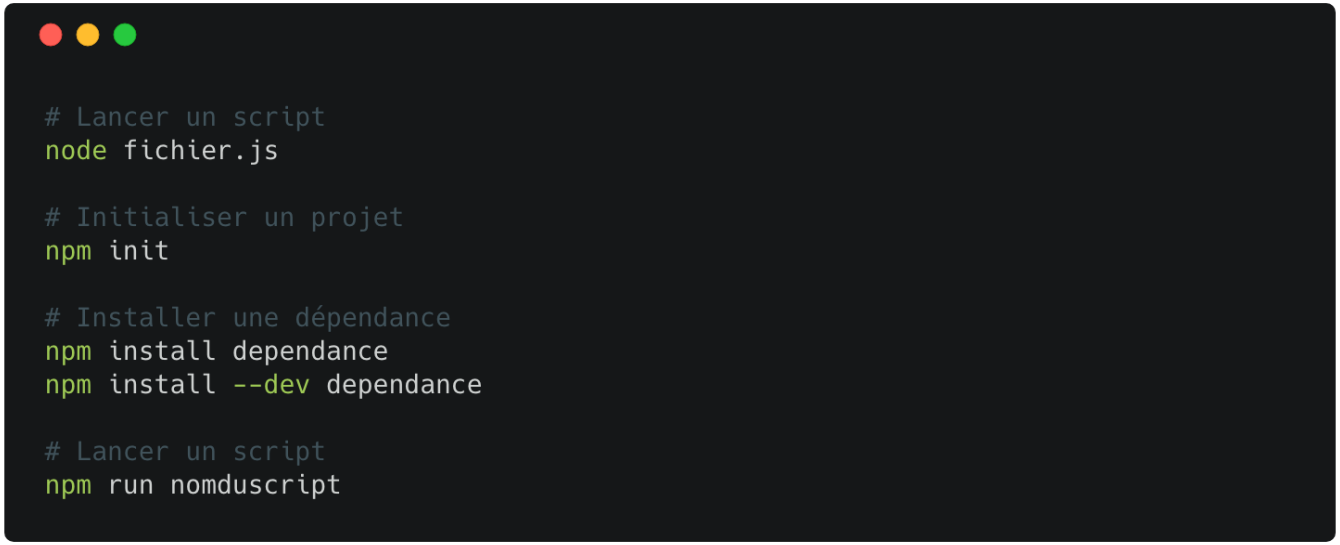
```
// src/api/users
import axios from 'axios'   // lib to make http calls easily

const fetchAllUsers = () => axios.get('/users')

export fetchAllUsers

// src/index.js
import { fetchAllUsers } from './api/users'

console.log("toto")
fetchAllUsers()
  .then(result => console.log("titi"))
  .catch(error => console.error("errors"))
console.log("tutu")
```



```
const users = [{name: "toto"}, {name: "titi"}, {name: "tutu"}]

users.forEach(user => console.log(user.name))

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

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

Nodes JS

- Interpréteur JavaScript
- Utilise le V8
- Créé en 2009
- Utilisé avec NPM



```
# Lancer un script
node fichier.js

# Initialiser un projet
npm init

# Installer une dépendance
npm install dependance
npm install --dev dependance

# Lancer un script
npm run nomduscript
```

```
{
  "name": "nom",
  "description": "description",
  "version": "0.1.0",
  "dependencies": {
    "dependance": "1.0.0"
  },
  "devDependencies": {
    "dev-dependance": "1.0.0"
  },
  scripts: {
    "nomduscript": "echo lol"
  }
}
```