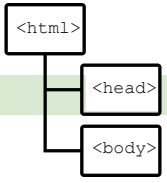
Programmation web - Client riche

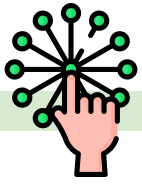
Introduction au JavaScript

Structure (HTML)



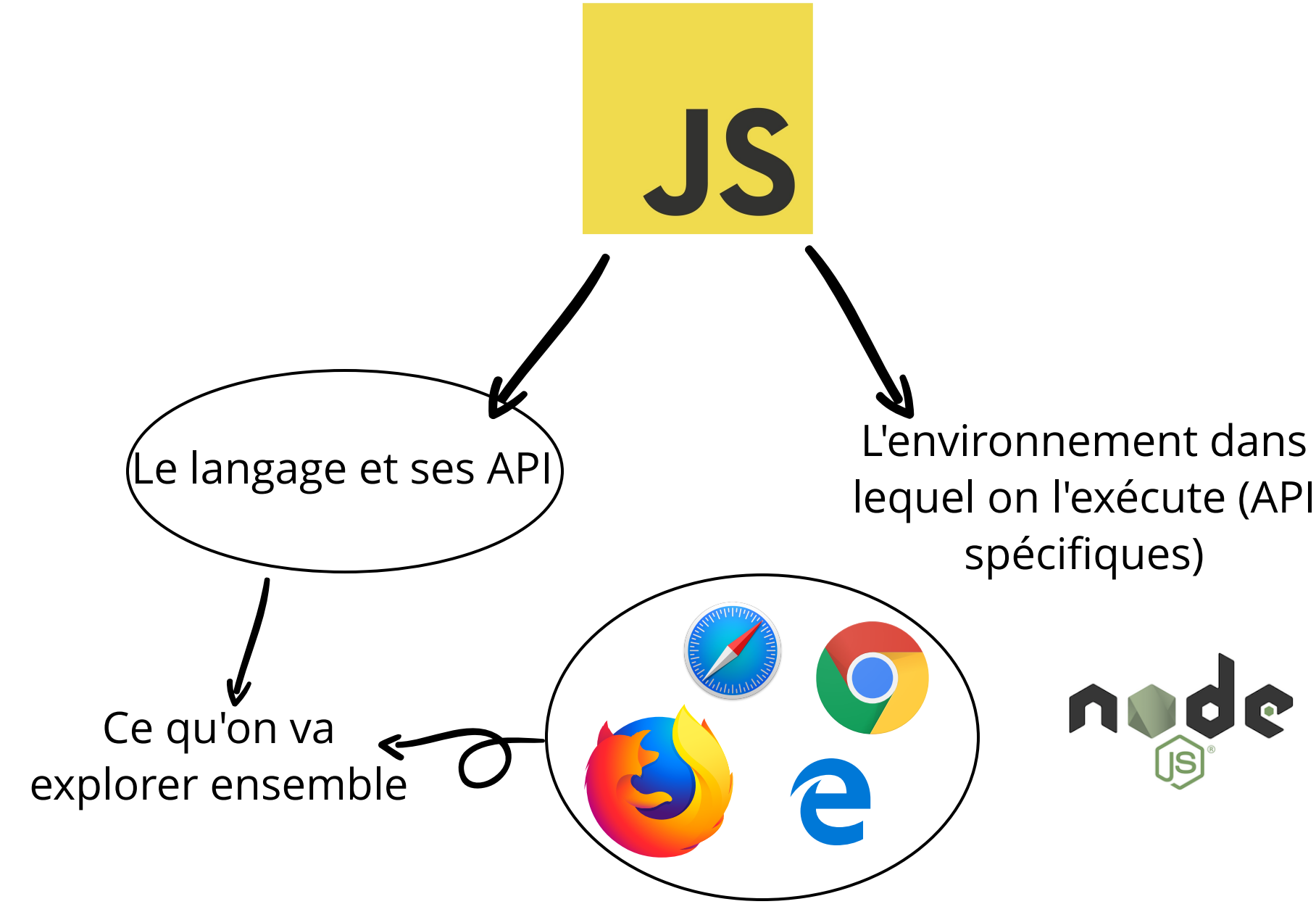
Style (CSS)

Comportement (JavaScript)



L'histoire de JS

WIKIPÉDIA
L'encyclopédie libre


JS
Le langage et ses API
L'environnement dans lequel on l'exécute (API spécifiques)
Ce qu'on va explorer ensemble
node
JS

Qu'est-ce qu'on peut faire dans le navigateur ?

De la 3D (WebGL)

Réagir à des événements utilisateur

Du son (Web Audio)

Des animations

Manipuler la structure de la page

Modifier le style des éléments HTML

Du dessin (canvas)

Envoyer une requête à un serveur (et afficher les données qu'il renvoie dynamiquement)

Écrire/lire des données dans le navigateur (cookies,...)

Utiliser la webcam et le micro de l'utilisateur

Localiser l'utilisateur

Afficher des notifications

Qu'est-ce qu'on ne peut pas faire dans le navigateur ?

Tout ce qui est "bas niveau" :

- Écrire/lire directement sur le disque de la machine
- Avoir un accès direct à l'OS de la machine
- Faire communiquer deux fenêtres/onglets s'ils n'ont aucun lien entre eux (l'un(e) a ouvert l'autre, et les domaines sont les mêmes)
- ...

La console du navigateur

Tous les navigateurs modernes embarquent des outils pour les développeurs. Généralement on y accède avec la touche **F12** ou la combinaison **CTRL+SHIFT+I**

Parmi ces outils se trouve la **console JavaScript**. Dans cette console, on peut taper du code JS et le faire **exécuter par le navigateur**.

Essayez d'ouvrir cette console, tapez-y la ligne de code suivante, puis appuyez sur Entrée :

```
alert("Hello world !")
```

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Hello world</title>
    </head>

    <body>
        <h1>Hello world</h1>

        <script>
            alert("hello from JS !")
        </script>
        <script src="/app.js"></src>
    </body>
</html>
```

```
// Les commentaires sur une seule ligne

/*
  les commentaires
  sur plusieurs lignes
*/
```

```
// Déclaration de la variable
let myVariable

// Affectation d'une valeur
myVariable = "Hello, world !"

// Déclaration et affectation
let myFirstVar = "Hello"

// Réassignation de la valeur d'une variable
let myVariable = "Hello world !"
myVariable = "Vous voulez un whisky ?"
```

```
// Déclaration et affectation
const myVariable = "Hello, world !"

// Déclaration et affectation
const myFirstVar = "Hello"

const hello = "Hello world !"
hello = "Vous voulez un whisky ?"
// => ReferenceError
```

*Vous verrez peut-être des scripts utiliser le mot-clef **var** pour déclarer une variable. C'est l'ancienne méthode. **Oubliez totalement ce mot-clef, qui est aujourd'hui remplacé par let***

Les types de variables

Number

```
const integer = 8000
const decimal = 123.456

// addition
alert(8000 + 42) // => 8042
// soustraction
alert(8000 - 42) // => 7958
// multiplication
alert(13 * 37) // => 481
// division
alert(8000 / 42) // => 190.476190476
// exponentiation
alert(2 ** 3) // => 8 (2 puissance 3, quoi)

alert(8000 / 0) // => Infinity
alert(-8000 / 0) // => -Infinity
alert("Everything but a number" / 8000) // => NaN
alert("Everything but a number" / 8000 + 42) // => NaN
```

String

```
const doubleQuotes = "Hello, world !"
const singleQuotes = 'Hello, world !'
const backticks = `Hello, world !`

const who = "world";
const doubleQuotes = "Hello, " + who + " !"
const singleQuotes = 'Hello, ' + who + ' !'
const backticks = `Hello, ` + who + ` !`
const backticks2 = `Hello ${who} !`
// => "Hello, world !"

const result = `The result of 1 + 1 is ${1 + 1}`
// => "The result of 1 + 1 is 2"
```

Boolean

```
const ok = true
const notOk = false
```

Null

```
const nullValue = null
```

Undefined

```
let undefinedValue
console.log(undefinedValue) // => undefined

let explicitUndefined = undefined
```

On a le conteneur (la variable), mais absolument rien dedans (undefined)

On a le conteneur (la variable) et quelque chose de vide dedans (null)

Object

```
const me = {
  firstname: 'Cyrille',
  age: 28
}

console.log(me.firstname) // => "Cyrile"
console.log(me.age) // => 28

me.firstname = 'George'
console.log(me.firstname) // => "George"

me.job = 'developer and teacher'

const key = 'job'
console.log(me[key]) // => "developer and teacher"
```

```
typeof undefined // => "undefined"

typeof 8000 // => "number"

typeof true // => "boolean"

typeof "Hello world" // => "string"

typeof Math // => "object"

typeof null // => "object"

typeof alert // => "function"

typeof Symbol("id") // => "symbol"
```

Comparaison

```
10 < 20 // => true
10 <= 20 // => true
20 > 10 // => true
20 >= 10 // => true
20 == 20 // => true
10 != 20 // => true
10 == "10" // => true
10 === "10" // => false
20 != "20" // => false
20 !== "20" // => true
```

Opérateurs logiques

```
true && true // => true
true && false // => false
true || true // => true
true || false // => true
false || false // => false

10 && "hello" // => "hello"
0 && "hello" // => 0

10 || "hello" // => 10
0 || "hello" // => "hello"

!true // => false
!false // => true
!"hello" // => false
```

```
const name = prompt("What's your name ?")

if (name === "Cyrille") {
  alert("Me too !")
} else if (name === "Toto") {
  alert("Srsly?")
} else {
  alert(`Hello ${name} !`)
}
```

```
let i = 0
while (i < 10) {
  alert(i)
  ++i
}

let j = 0
do {
  alert(j)
  ++j
} while (j < 10)

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

// => 0 1 2 3 4 5 6 7 8 9
```

Les fonctions

```
function add(a, b) {
  return a + b
}

add(1, 10) // => 11

const difference = function (a, b) {
  return a - b
}

difference(10, 1) // => 9

const divide = (a, b) => {
  return a / b
}

divide(10, 2) // => 5

const exp = (n, p) => n ** p

exp(2, 2) // => 4
```

```
const exp = (n, p = 1) => n ** p

exp(2, 2) // => 4
exp(2) // => 2
```

Tout est objet
(enfin presque)

```
const number = 123.456
alert(number.toFixed(2)) // => 123.46

const str = "Hello world !"
alert(str.toUpperCase()) // => HELLO WORLD !
```

```
const emptyArray = []

const animals = ["cat", "dog", "mouse"]
animals[0] // => "cat"
animals[1] // => "dog"
animals[2] // => "mouse"
animals.length // => 3

animals[1] = "parrot"

animals.push("rat")
// => ["cat", "parrot", "mouse", "rat"]

animals.slice(0, 2)
// => ["cat", "parrot"]
```

```
const fruits = ["Apple", "Banana", "Strawberry"]

for (let i = 0, length = fruits.length; i < length; ++i) {
  console.log(fruits[i])
}

for (const fruit of fruits) {
  console.log(fruit)
}

fruits.forEach((fruit) => {
  console.log(fruit)
})

fruits.forEach(fruit => console.log(fruit))

// => "Apple" "Banana" "Strawberry"
```

Array (MDN)

```
class User {
  constructor(name, age) {
    this.name = name
    this.age = age
  }

  getDescription() {
    return `${this.name} / ${this.age}`
  }

  sayMyName() {
    alert(this.name)
  }
}

class Admin extends User {
  getDescription() {
    return super.getDescription() + " (admin)"
  }
}

const user = new User("John Doe", 42)
const admin = new Admin("Jane Doe", 42)

user.getDescription() // => "John Doe / 42"
admin.getDescription() // => "Jane Doe / 42 (admin)"
```

En JS, le modèle objet est dit "prototypal", ce qui est particulier. Pour en savoir plus, vous pouvez lire le guide (très complet) du Mozilla Developer Networks sur les objets en JS

Des questions ?

See you next week