

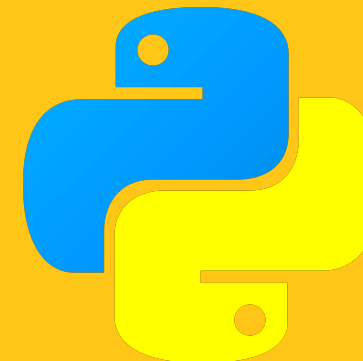
JavaScript is a high-level, prototype-based object-oriented, multi-paradigm, interpreted or just-in-time compiled, dynamic, single-threaded, garbage-collected programming language with first-class functions and a non-blocking event loop concurrency model.

JavaScript is a high-level, prototype-based object-oriented, multi-paradigm, interpreted or just-in-time compiled, dynamic, single-threaded, garbage-collected programming language with first-class functions and a non-blocking event loop concurrency model.

- HIGH-LEVEL
- GARBAGE COLLECTED
- INTERPRETED OR JIT COMPILED
- MULTI-PARADIGM
- PROTOTYPE-BASED
- FIRST-CLASS FUNCTIONS
- DYNAMIC
- SINGLE-THREADED
- EVENT LOOP



high-level

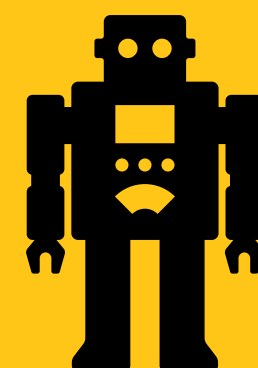


Allocation automatique des ressources



Allocation manuelle des ressources

01001000 01100101
01101100



low-level

HIGH-LEVEL

GARBAGE COLLECTED

INTERPRETED OR JIT COMPILED

MULTI-PARADIGM

PROTOTYPE-BASED

FIRST-CLASS FUNCTIONS

DYNAMIC

SINGLE-THREADED

EVENT LOOP



HIGH-LEVEL

GARBAGE COLLECTED

INTERPRETED OR JIT COMPILED

MULTI-PARADIGM

PROTOTYPE-BASED

FIRST-CLASS FUNCTIONS

DYNAMIC

SINGLE-THREADED

EVENT LOOP

```
function findKey(object, predicate) {  
  let result;  
  if (object == null) {  
    return result;  
  }  
  Object.keys(object).some((key) => {  
    const value = object[key];  
    if (predicate(value, key, object)) {  
      result = key;  
      return true;  
    }  
  });  
  return result;  
}  
  
export default findKey;
```



```
00010010010011011001000010010000111100001001000010010000  
11110000100000001110000010000000111100001000000010000000  
10000000100000001111000011110000100100001001000010010000  
11110000111000001001000011111000100100001110000001010000  
01010000001000000010000000100000111100001000000011100000  
100000001111000010100010000000100110010000000000011010000  
00010101001101000000001011110010000111100111000000000101  
011101000000000010011010000000100000100100010100111010000
```

HIGH-LEVEL

GARBAGE COLLECTED

INTERPRETED OR JIT COMPILED

MULTI-PARADIGM

PROTOTYPE-BASED

FIRST-CLASS FUNCTIONS

DYNAMIC

SINGLE-THREADED

EVENT LOOP



Paradigme: Une façon d'approcher la programmation qui dirige le style et la technique du code, ainsi que la façon d'aborder les problèmes.

- 1 Programmation Procédurale
- 2 Programmation Orientée Objet
- 3 Programmation Fonctionnelle

Ce que nous faisons depuis le début du cours.

HIGH-LEVEL

GARBAGE COLLECTED

INTERPRETED OR JIT COMPILED

MULTI-PARADIGM

PROTOTYPE-BASED

FIRST-CLASS FUNCTIONS

DYNAMIC

SINGLE-THREADED

EVENT LOOP

```
const arr = [1,2,3];  
arr.push(4);  
const hasZero = arr.indexOf(0) > -1;
```

HIGH-LEVEL

GARBAGE COLLECTED

INTERPRETED OR JIT COMPILED

MULTI-PARADIGM

PROTOTYPE-BASED

FIRST-CLASS FUNCTIONS

DYNAMIC

SINGLE-THREADED

EVENT LOOP

Array

`Array.prototype.push`

`Array.prototype.indexOf`

Prototype

Créé à partir du Prototype

```
const arr = [1,2,3];  
arr.push(4);  
const hasZero = arr.indexOf(0) > -1;
```


HIGH-LEVEL

GARBAGE COLLECTED

INTERPRETED OR JIT COMPILED

MULTI-PARADIGM

PROTOTYPE-BASED

FIRST-CLASS FUNCTIONS

DYNAMIC

SINGLE-THREADED

EVENT LOOP

Array

Array.prototype.push

Array.prototype.indexOf

Prototype

Notre tableau hérite
des méthodes de
son Prototype

Créé à partir du Prototype

```
const arr = [1,2,3];  
arr.push(4);  
const hasZero = arr.indexOf(0) > -1;
```

HIGH-LEVEL

GARBAGE COLLECTED

INTERPRETED OR JIT COMPILED

MULTI-PARADIGM

PROTOTYPE-BASED

FIRST-CLASS FUNCTIONS

DYNAMIC

SINGLE-THREADED

EVENT LOOP

👉 Dans un langage avec des fonctions de première classes, celles-ci sont traitées comme des variables. Nous pouvons donc passer des fonctions arguments dans d'autres fonctions. Nous pouvons également retourner des fonctions.

```
function sayHello() {  
  return "Hello, ";  
}  
function greeting(helloMessage, name) {  
  console.log(helloMessage() + name);  
}  
greeting(sayHello, "JavaScript!");
```



HIGH-LEVEL

GARBAGE COLLECTED

INTERPRETED OR JIT COMPILED

MULTI-PARADIGM

PROTOTYPE-BASED

FIRST-CLASS FUNCTIONS

DYNAMIC

SINGLE-THREADED

EVENT LOOP

```
let x = 23;  
let y = 19;  
x = "plage";
```


HIGH-LEVEL

GARBAGE COLLECTED

INTERPRETED OR JIT COMPILED

MULTI-PARADIGM

PROTOTYPE-BASED

FIRST-CLASS FUNCTIONS

DYNAMIC

SINGLE-THREADED

EVENT LOOP

👉 **Concurrency model:** La façon dont le moteur JavaScript gère plusieurs tâches se déroulant en même temps.

Pourquoi en a-t-on besoin?

👉 JavaScript tourne sur une seule **thread**, donc il ne peut faire qu'une seule chose à la fois.

Et si une tâche prend du temps?

👉 Le moteur utilise un **event loop**: les longues tâches sont exécutées en “arrière-plan” et sont remises dans le thread principal lorsqu'elles sont terminées.

ON EN PARLE EN DÉTAIL PLUS TARD DANS LE SEMESTRE