

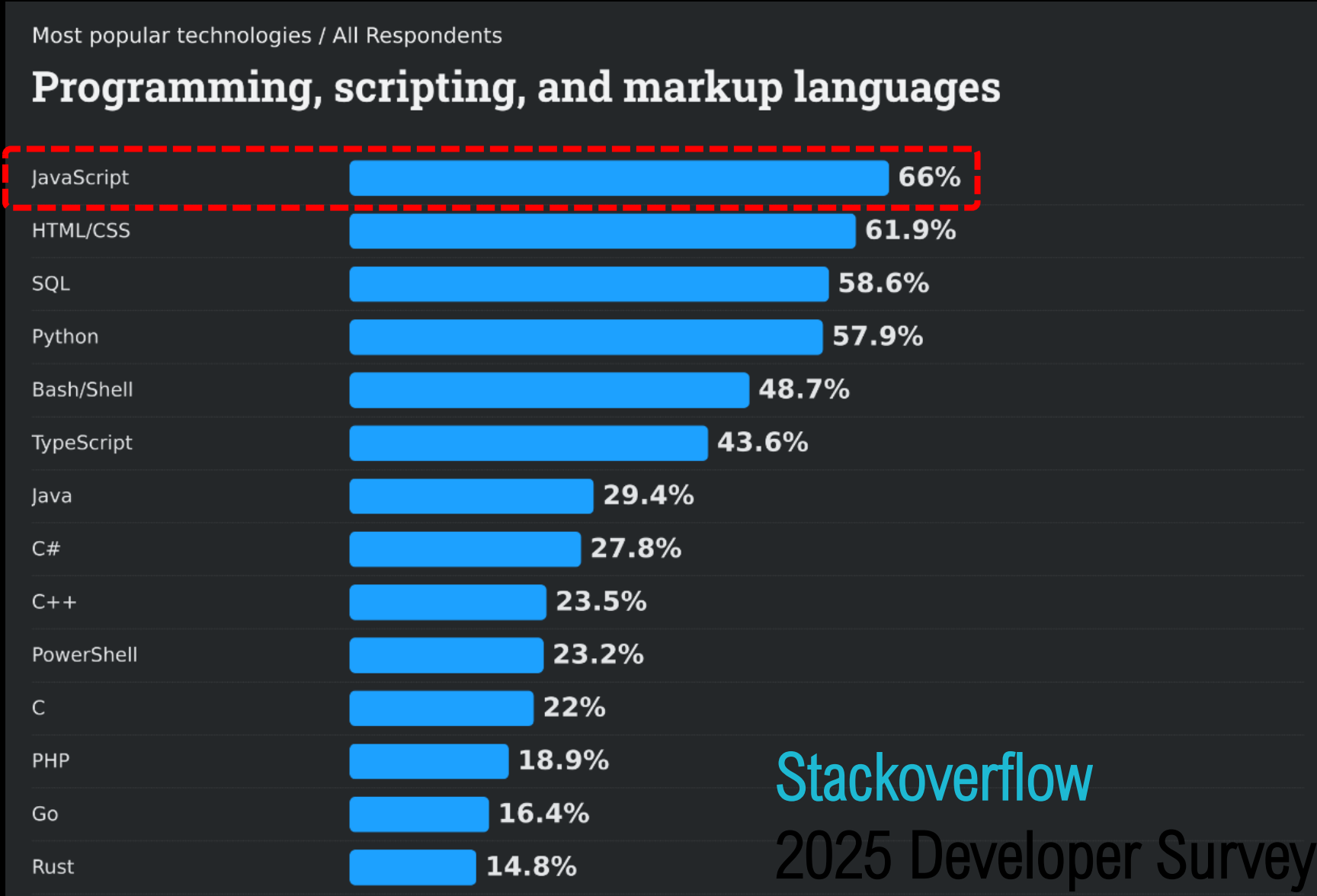
# Pengenalan JavaScript (ES6+)

Author: Mocha Kei

Github: <https://github.com/imochakei/javascript-es6>

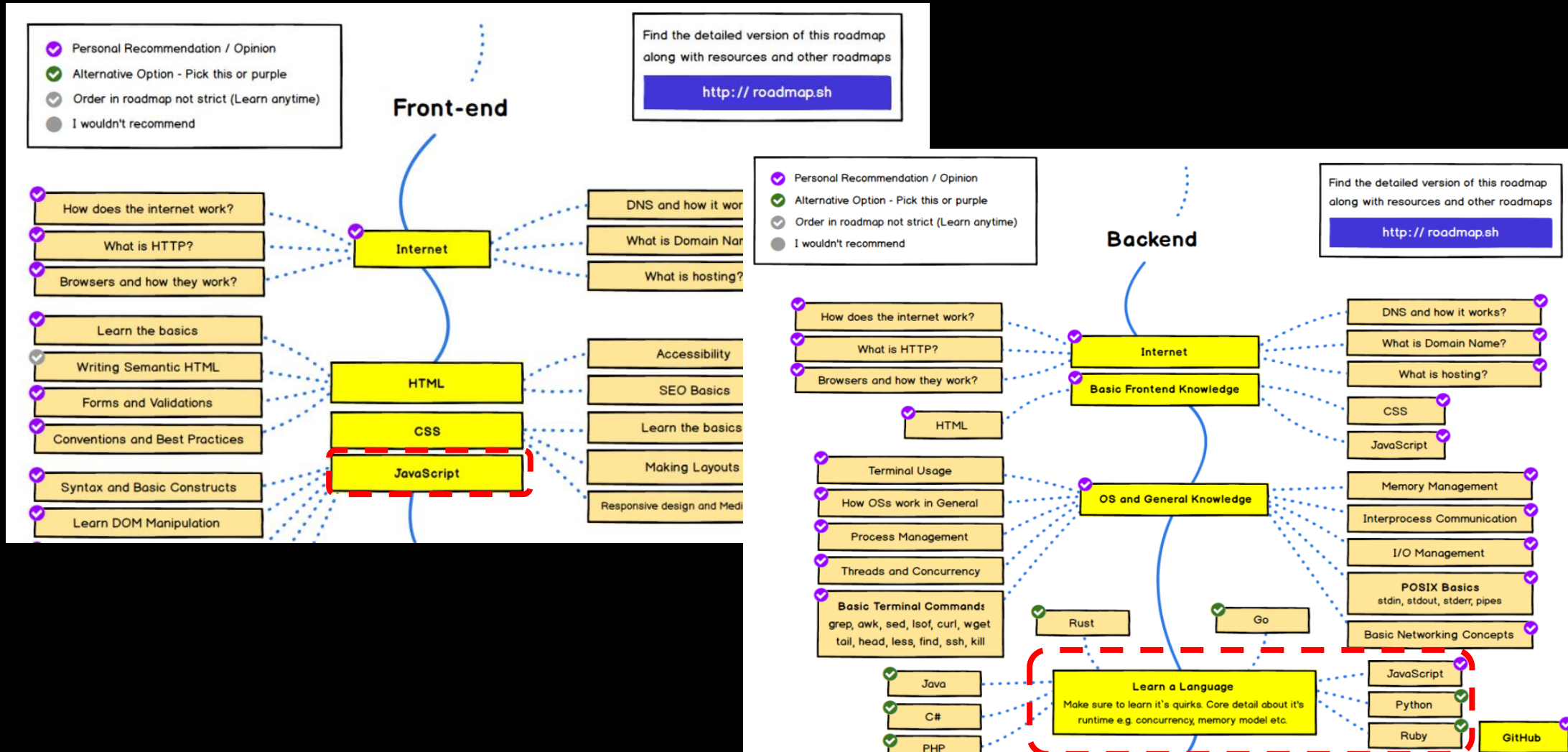


# ● Trend Pemrograman 2025

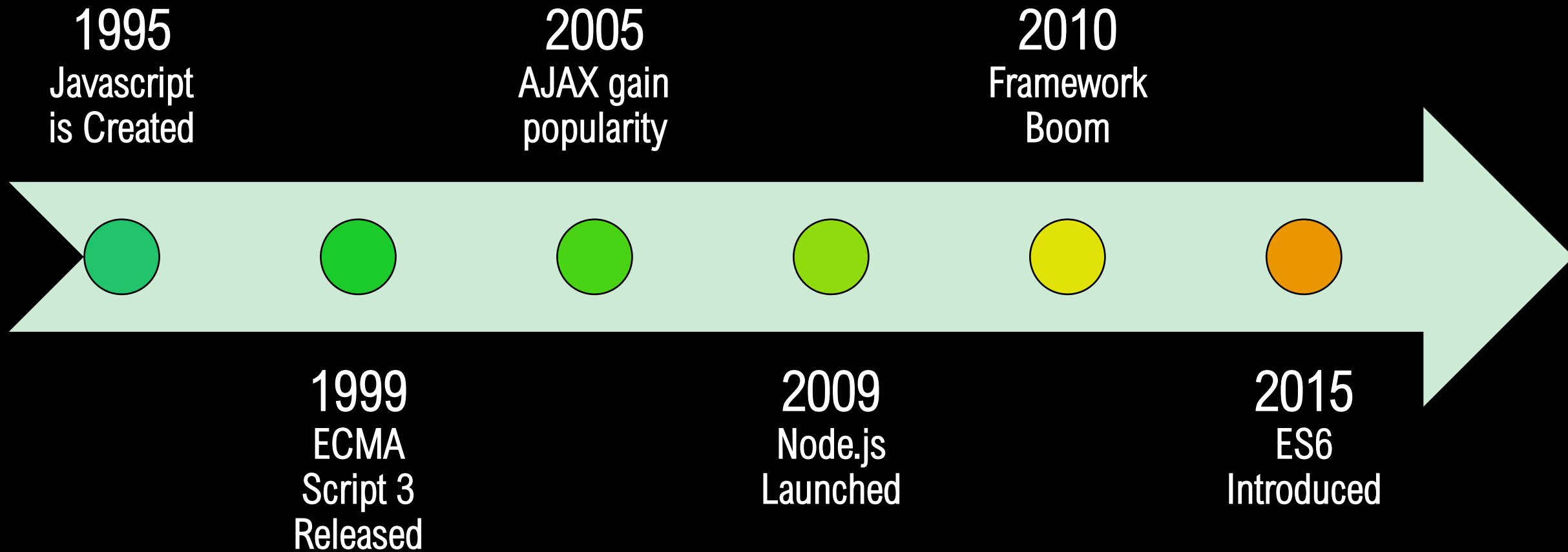


# Keunggulan Javascript

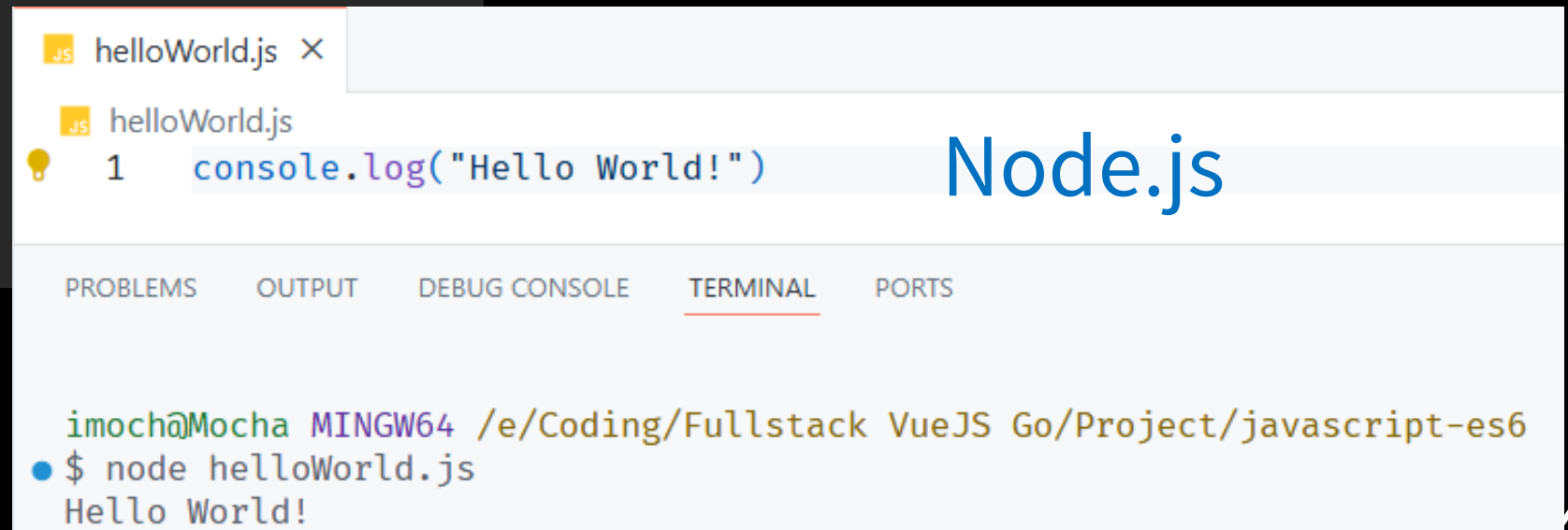
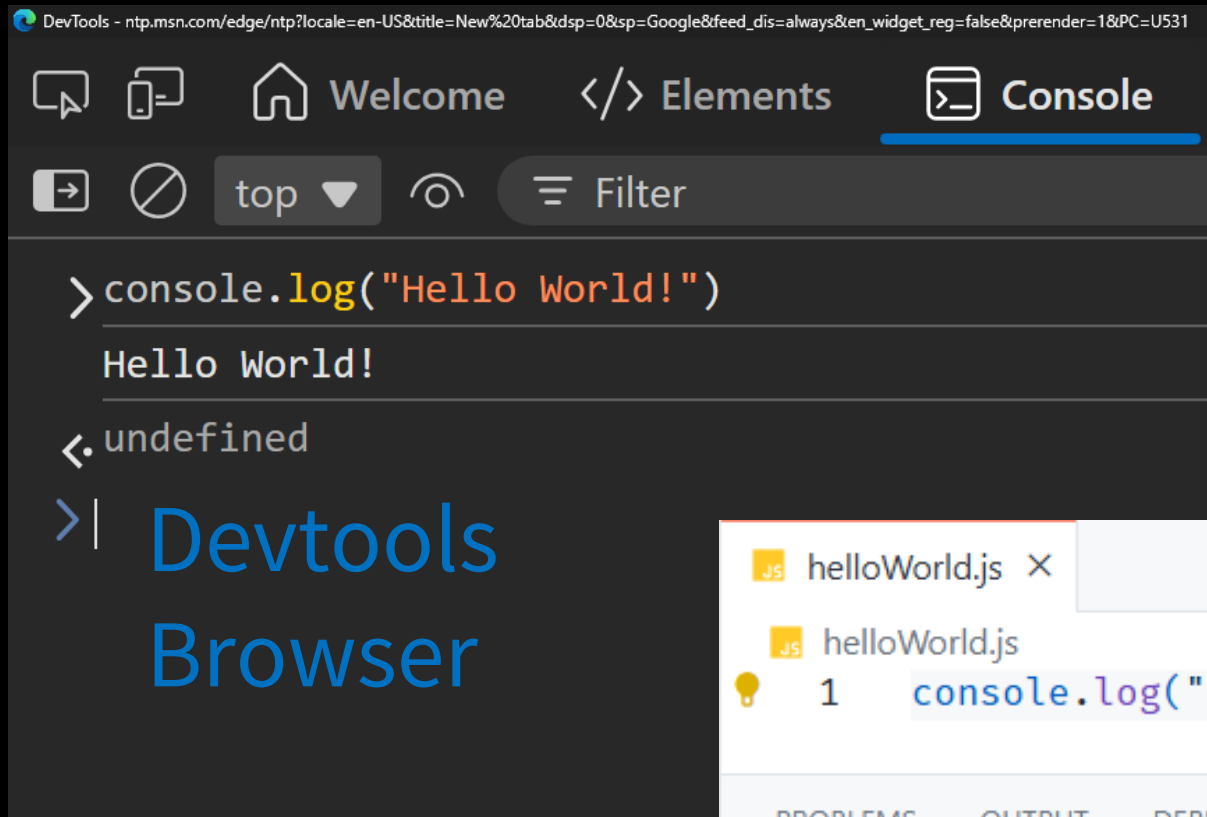
Kebebasan memilih *stack* teknologi: front-end, back-end, gamedev, iot



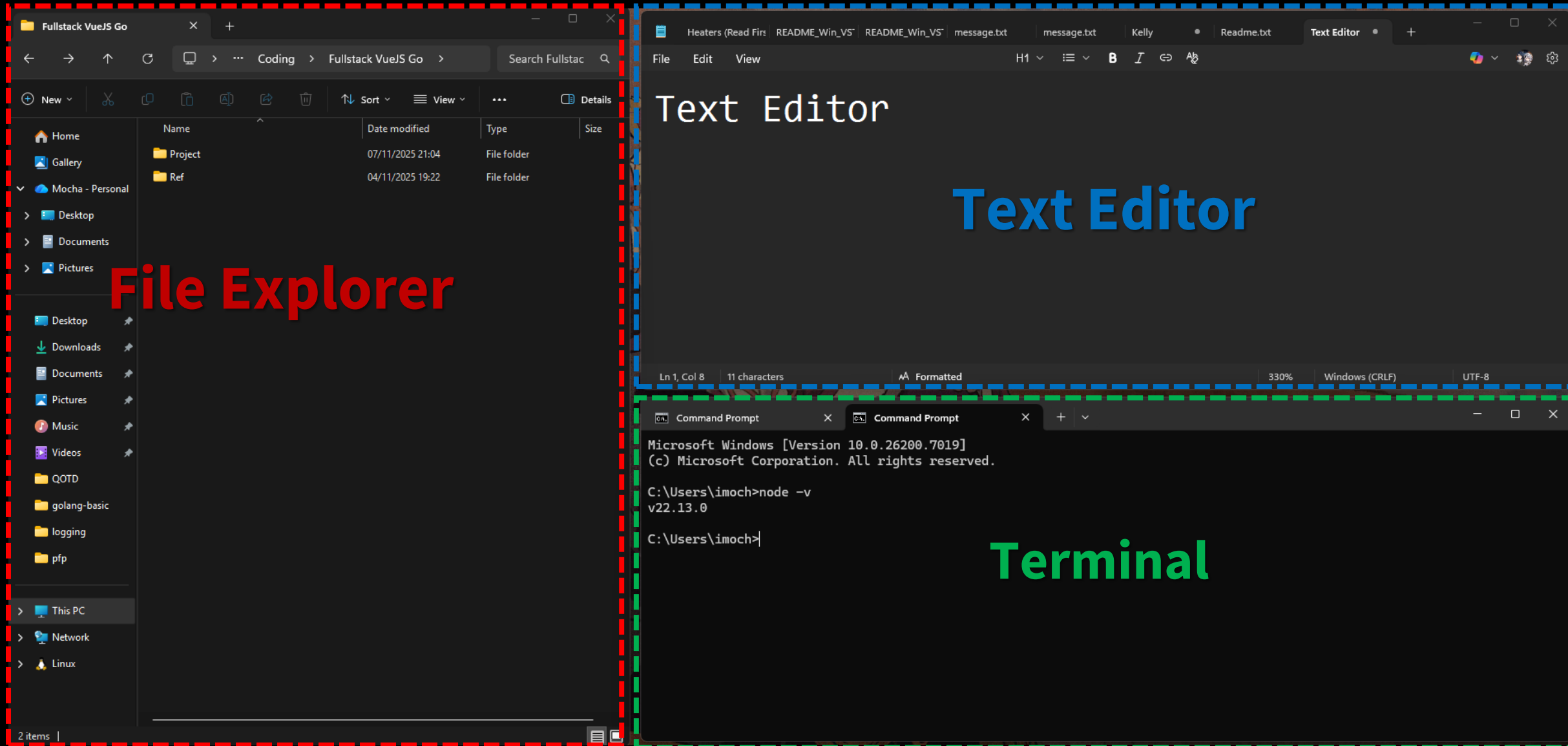
# Sejarah Javascript



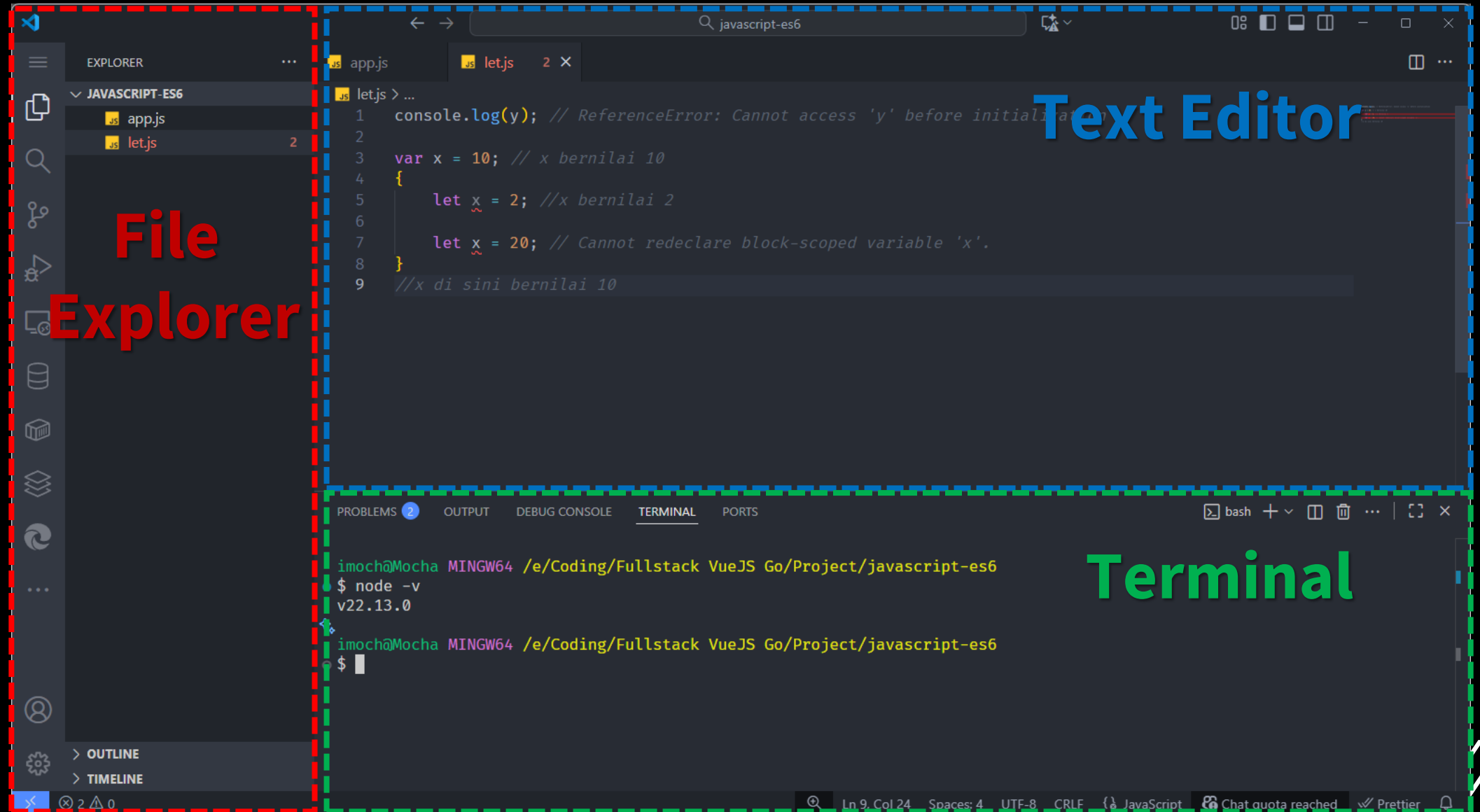
# ● Cara menjalankan javaScript



# Development Environment

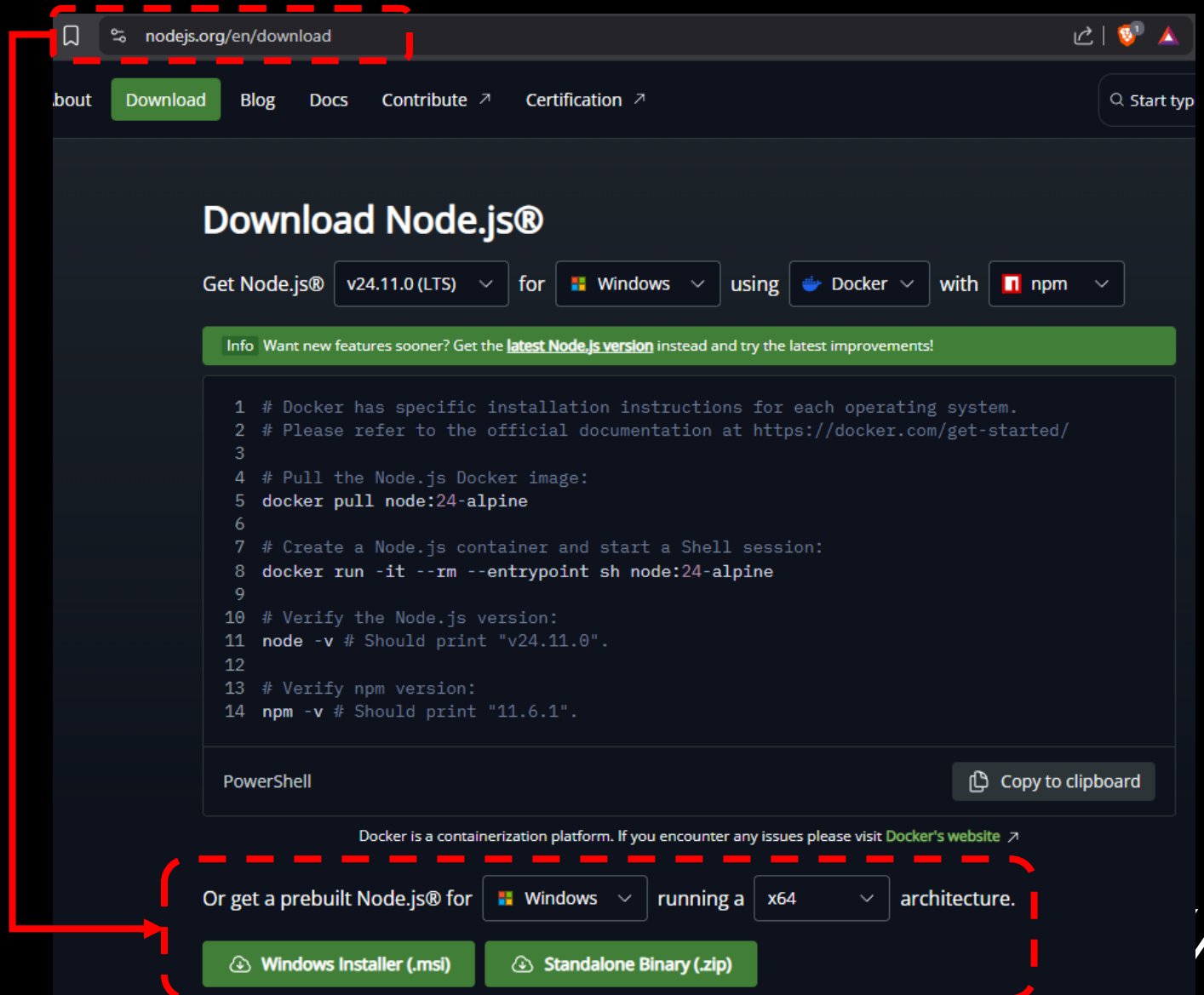


# Integrated Development Environment (IDE)



# Panduan instalasi Node.js

1. Download Node.js pada <https://nodejs.org/en/download>
2. Pilih OS yang digunakan (windows, mac, linux)
3. Download installer
4. Jalankan installer seperti biasa
5. Test instalasi dengan mengetikan `node -v` pada cmd



# Tipe Data JavaScript

```
// String
let race = "Alien";
let fullName = "Mocha Kei";
// Number
let age = 17;
let weight = 50.5;
// Boolean
let x = true;
let y = false;
// Object
const person = { firstName: "Mocha", lastName: "Kei" };
// Array object
const language = ["PHP", "Javascript", "GO"];
// Undefined
let x;
let y;
// Null
let x = null;
let y = null;
```



# ECMAScript 5 (2009) and ECMAScript 6 (2015)

- 2009: ECMAScript 5 (ES5) was released, adding important features like strict mode, [JSON](#) support, and array methods.
- 2015: ECMAScript 6 (ES6) was released, introducing modern features like [arrow functions](#), classes, modules, template literals, and promises.

| Feature           | Description  |
|-------------------|--|
| Arrow Functions   | Concise syntax for writing functions <code>() =&gt; {}</code>              |
| Template Literals | String interpolation, which allows embedding expressions inside strings    |
| Classes           | Syntactic sugar for creating objects and dealing with inheritance          |
| Promises          | A cleaner way to work with asynchronous code, replacing callback functions |

# Fitur Baru ES6+

| Fitur                            | Deskripsi  |
|----------------------------------|--|
| <b>The let Keyword</b>           | The <u>let variables</u> are mutable, i.e., their values can be changed. It works similarly to the var keyword with some key differences, like scoping, which makes it a better option when compared to var. |
| <b>The const Keyword</b>         | <u>const</u> is used to declare variables with a constant value, ensuring the value cannot be reassigned.  |
| <b>Arrow Functions</b>           | <u>Arrow functions</u> provide a concise syntax for writing function expressions and automatically bind this to the surrounding context.   |
| <b>Destructuring Assignment</b>  | <u>Destructuring in JavaScript</u> basically means the breaking down of a complex structure( <u>Objects</u> or <u>arrays</u> ) into simpler parts  |
| <b>The Spread (...) Operator</b> | The <u>spread operator</u> expands an array or object into individual elements or properties.  |
| <b>The For/Of Loop</b>           | The <u>for/of loop</u> allows you to iterate over iterable objects like arrays, strings, Maps, and Sets but in a short syntax as compared to other loops.  |
| <b>Maps and Sets</b>             | Map: <u>Maps</u> store key-value pairs where keys can be any data type.<br>Set: <u>Sets</u> store unique values of any type.   |
| <b>Classes</b>                   | ES6 introduced <u>classes in JavaScript</u> . Classes in javascript can be used to create new Objects with the help of a constructor, each class can only have one constructor inside it.                    |
| <b>Promises</b>                  | <u>Promises</u> simplify handling asynchronous operations by providing .then and .catch methods.   |
| <b>Default Parameters</b>        | Allows functions to have default values for parameters.  |



# (1) JavaScript let

Dalam contoh ini, **let** di dalam kurung kurawal dianggap sebagai variabel yang berbeda, tidak mempengaruhi variabel **x** di luar blok. Ini berbeda dengan **var** yang bersifat function-scoped atau global.

memiliki **block scope**, **dideklarasikan sebelum digunakan** dan **tidak bisa dideklarasikan ulang**

```
console.log(y); // ReferenceError: Cannot access 'y' before initialization
var x = 10; // x bernilai 10
{
    let x = 2; //x bernilai 2
    let x = 20; // Error: Cannot redeclare block-scoped variable 'x'.
}
//x di sini bernilai 10
```



## (2) JavaScript const

Const memiliki aturan yang sama seperti let, namun isi dari const tidak bisa dirubah setelahnya. Kecuali untuk array dan object, kita bisa mengganti isi dari array atau property object

```
const PI = 3.141592653589793;  
PI = 3.14; // ini akan error (Reassigned)  
PI = PI + 10; // ini juga akan error
```

// Constant array:

```
const peserta = ["Mocha", "Moona", "Budi"];  
peserta[0] = "Kei"; // mengganti isi dari array  
peserta.push("Zeta"); // menambahkan isi array
```

// const object:

```
const peserta = {firstName:"Mocha", lastName:"Kei", job:"Vtuber"};  
peserta.color = "Dilan"; // Bisa mengganti property  
car.partner = "Milea"; // Bisa menambahkan property
```



# Rangkuman var, let and const

1. let and const have **block scope**.
2. let and const can not be **redeclared**.
3. let and const must be **declared** before use.
4. let and const does **not bind** to this.
5. let and const are **not hoisted**.

|       | Scope | Redeclare | Reassign | Hoisted | Binds this |
|-------|-------|-----------|----------|---------|------------|
| var   | No    | Yes       | Yes      | Yes     | Yes        |
| let   | Yes   | No        | Yes      | No      | No         |
| const | Yes   | No        | No       | No      | No         |



### (3) Arrow Functions

Arrow function adalah cara menulis fungsi pada javascript ES6 yang lebih ringkas

```
// Fungsi Javascript Tradisional (function expression)
const greet = function(name) {
    return "Halo, " + name + "!";
};
console.log(greet('Traditional'));
```

```
// Modern es6+ (arrow function)
const greetArrowFunctions = name => `Halo, ${name}!`;

console.log(greetArrowFunctions('Arrow Functions'));
```



## (4) Destructuring Assignment

Destructuring Assignment adalah sintaks ES6 untuk “membongkar” nilai dari array atau properti dari objek langsung ke variabel terpisah sehingga kode lebih ringkas dan mudah dibaca.

```
// Object destructuring
const user = { firstName: "Mocha", lastName: "Kei", age: 17 };
const { firstName, lastName } = user; // "Mocha", "Kei"
let { age } = user;
console.log({firstName, lastName, age})
```

```
// Array destructuring
const fullstacks = ["Frontend", "Backend"];
const [f, b] = fullstacks;
console.log({f, b})
```



## (5) The Spread (...) Operator

Operator spread (...) adalah sintaks ES6 untuk “menyebarkan” elemen array atau properti objek menjadi unit individual seperti di array/object atau argumen fungsi

```
// array spread
const numbers = [23,55,21,87,56];
let minValue = Math.min(...numbers); //21
let maxValue = Math.max(...numbers); //87

// object spread
const user = { firstName: "Mocha", lastName: "Kei" };
const profile = { age: 17, city: "Isekai" };
const identity = { ...user, ...profile }; // menggabungkan properti
const withFlag = { ...user, active: true }; // menambah properti
const override = { ...user, lastName: "Chiato" }; // menimpa properti
```



## (6) The For/Of Loop

Gunakan pola for (**variable** of **iterable**) { ... } di mana variable akan berisi nilai saat ini pada setiap iterasi, dan variable dapat dideklarasikan dengan **let**, **const**, atau **var** sesuai kebutuhan. Sintaksnya ringkas dan mendukung kontrol alur seperti **break** dan **continue**

```
const users = [
  { firstName: "Mocha", lastName: "Kei" },
  { firstName: "Moona", lastName: "Hoshinova" }
];

const fullNames = [];
for (let u of users) {
  fullNames.push(`${u.firstName} ${u.lastName}`);
}
console.log({ fullNames })
```



## (7) Maps and Sets

**Map** menyimpan pasangan **key–value** dengan key dari tipe apa pun, sedangkan **Set** menyimpan kumpulan nilai unik tanpa duplikasi

```
// Map
const users = new Map([
  [101, { firstName: "Mocha", lastName: "Kei" }],
  [102, { firstName: "Moona", lastName: "Hoshinova" }],
]);
console.log(userById.get(101)); // { firstName: "Mocha", lastName: "Kei" }
console.log(userById.has(999)); // false
console.log(userById.size);      // 2
```



## (7) Maps and Sets

**Map** menyimpan pasangan **key–value** dengan key dari tipe apa pun, sedangkan **Set** menyimpan kumpulan nilai unik tanpa duplikasi

```
// Set
const tags = new Set();
tags.add("javascript");
tags.add("es6");
tags.add("javascript"); // diabaikan karena sudah ada (harus unik)
console.log(tags.has("es6")); // true
console.log(tags.size);      // 2

const names = ["Moona", "Mocha", "Moona", "Zeta"];
const uniqueNames = [...new Set(names)]; // ["Moona", "Mocha", "Zeta"]
```



# (7) Maps and Sets

Melakukan filter unique dengan menggunakan set berdasarkan fullname

```
// Map: id → user
const users = new Map([
  [101, { firstName: "Mocha", lastName: "Kei" }],
  [102, { firstName: "Moona", lastName: "Hoshinova" }],
  [103, { firstName: "Mocha", lastName: "Kei" }] // data mirip, id beda
]);
// Ambil daftar fullName
const fullNames = [];
for (const [, u] of users) {
  fullNames.push(`${u.firstName} ${u.lastName}`);
}
// ["Mocha Kei", "Moona Hoshinova", "Mocha Kei"]

// Set: pastikan unik
const uniqueFullNames = [...new Set(fullNames)]; // ["Mocha Kei", "Moona Hoshinova"]
```



## (8) Classes

Class seperti **cetakan kue**, dimana kue yang dihasilkan dari cetakan kue itu adalah **object**  
Warna kue bisa bermacam-macam meskipun berasal dari cetakan yang sama (**object memiliki sifat independen**)

```
//inisiasi class
class Person {
  //constructor
  constructor(name, age) {
    this.name = name; //attribut nama
    this.age = age; //attribut umur
  }
  // perilaku/behavior dituangkan dalam bentuk fungsi
  speak() {
    console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);
  }
}

let mocha = new Person("Mocha Kei", 17); //object
mocha.speak(); //Hello, my name is Mocha Kei and I am 25 years old.
```



## (9) Promises

Promise adalah objek operasi asynchronous yang memisahkan “kode yang menghasilkan hasil” dan “kode yang menunggu hasil” melalui handler **then**, **catch**, dan **finally**. `resolve(value)` menandakan operasi sukses dan meneruskan value, sedangkan `reject(error)` menandakan gagal dan meneruskan error.

### Kombinasi beberapa Promise

- `Promise.all([...])` menunggu semua Promise selesai; jika salah satu gagal, hasil akhirnya `rejected`, dan jika sukses semua, hasilnya berupa array nilai dalam urutan input.
- `Promise.race([...])` menyelesaikan sesuai Promise yang paling cepat, baik sukses maupun gagal, sehingga cocok untuk batas waktu atau balapan sumber data.
- `Promise.allSettled([...])` menyelesaikan setelah semua settle tanpa “gagal total”, mengembalikan status per item; `Promise.any([...])` terpenuhi oleh Promise sukses pertama dan hanya gagal jika semua gagal.



## (9) Promises

```
const fetch = () => {  
    return new Promise((resolve, reject) => {  
        setTimeout(() => resolve("Data fetched"), 2000);  
    });  
};
```

```
const delay = (ms) => new Promise(resolve => setTimeout(resolve, ms));
```

```
fetch().then(data => console.log(data));  
// delay 500 akan menampilkan hasil terlebih dahulu  
// walaupun pemanggilannya fetch dahulu baru delay  
delay(500).then(() => console.log("setengah detik berlalu"));
```



## (10) Default Parameters

Nilai default hanya dipakai ketika argumen bernilai undefined untuk kemudian diberikan nilai bawaan pada parameter fungsi

```
const halo = (name = "Zeta") => console.log(`Hai, ${name}!`);
```

```
halo();          // "Halo, Dilan"  
halo("Moona");  // "Halo, Milea"
```



Terima Kasih

<https://www.youtube.com/@mochakei>

