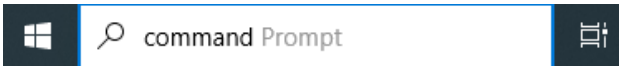
COURSE : PLATFORM BASED PROGRAMMING
SESSION : II
SUBJECT : Practicum Guide: JavaScript Basics in Node.js
STUDENT NAME : Jaenal mutakin

Practicum Objective

1. Understand the use of JavaScript in the Node.js environment.
2. Get to know the basics of the JavaScript programming language.
3. Understand the use of variables, data types in JavaScript.
4. Use built-in and third-party modules in Node.js.

1. Create a project in the following way:

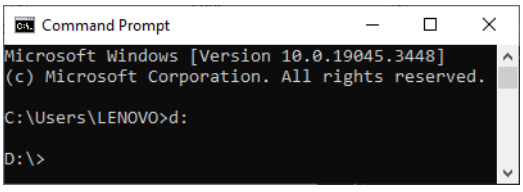
a. Open a console window, type "command" in the search input



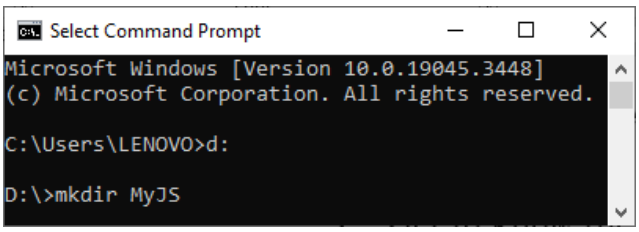
b. then the search results will appear as follows:

c. then click "Open"

d. Change the active drive according to your own hard disk, for example change it to the d:\ drive as shown below:



e. Create the **MyJS** folder using the **mkdir name-directory** syntax as shown below:

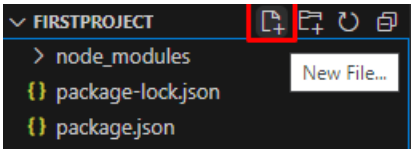


f. then change the current directory to **MyJS**, using the **cd name-directory** (change directory) syntax, as follows:

g. The next step is to initialize the project using the npm init syntax. Fill in some parameters that will be used to generate the package.json file

h. type **code .** to open the project folder into visual studio code as follows

i. Finally, create a file with the name index.js in the visual studio code, by pressing the (+) sign or new file as follows:

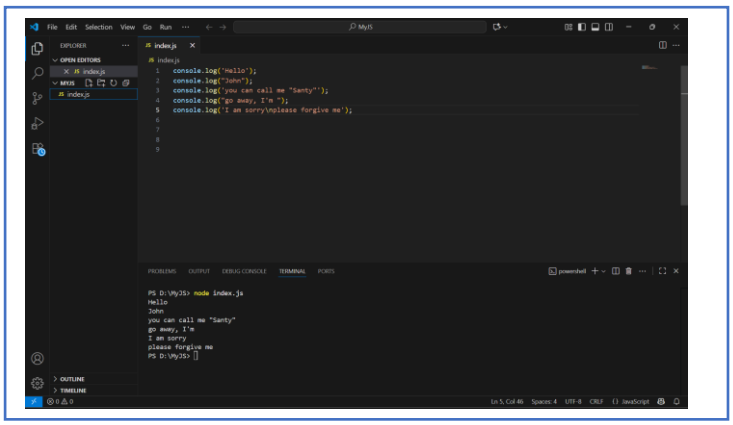


2. Making a program

a. Write the following program code !

```
console.log('Hello');
console.log("John")
console.log('You can call me "Santy"');
console.log("Go away, I'm Coding");
console.log(`I am sorry\nplease forgive me`);
```

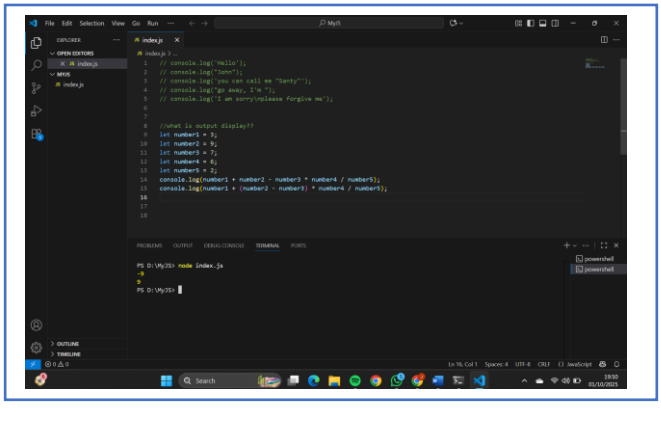
Display the results of the program code and write your analysis in the box below !



- console.log(...) → fungsi untuk **mencetak teks** ke konsol (terminal)
- 'Hello', 'John', dsb. → **string** yang akan dicetak
- \" atau \' → digunakan untuk menampilkan tanda kutip di dalam string
- \n → karakter **new line** (baris baru), jadi teks setelahnya tampil di baris berikutnya

b. Write the following program code !

```
//What is output display??
let number1 = 3;
let number2 = 9;
let number3 = 7;
let number4 = 6;
let number5 = 2;
console.log(number1 + number2 - number3 * number4 / number5);
console.log(number1 + (number2 - number3) * number4 / number5);
```

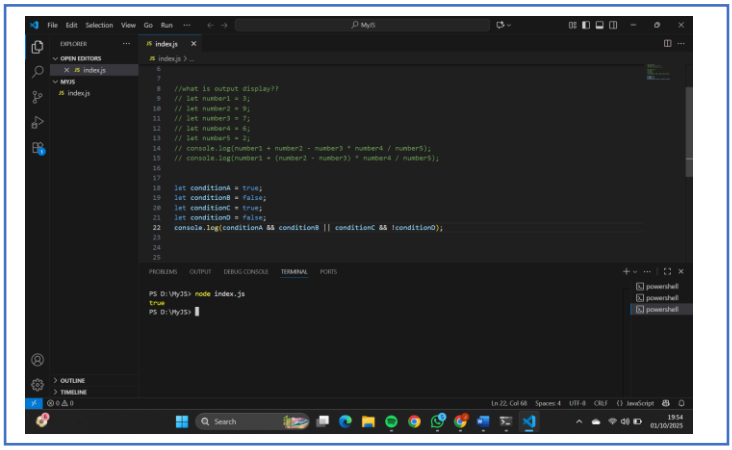


- Ada dua ekspresi matematika yang dicetak lewat console.log.
- Baris pertama: perkalian dan pembagian dilakukan **lebih dulu**, baru penjumlahan/ pengurangan.
- Baris kedua: ada tanda kurung, jadi bagian dalam kurung dihitung dulu, lalu dikalikan, dibagi, dan ditambah ke number1.
- Itulah kenapa hasilnya berbeda: satu negatif, satu positif.

c. Write the following program code !

```
let conditionA = true;
let conditionB = false;
let conditionC = true;
let conditionD = false;
console.log(conditionA && conditionB || conditionC && !conditionD);
```

Display the results of the program code and write your analysis in the box below !



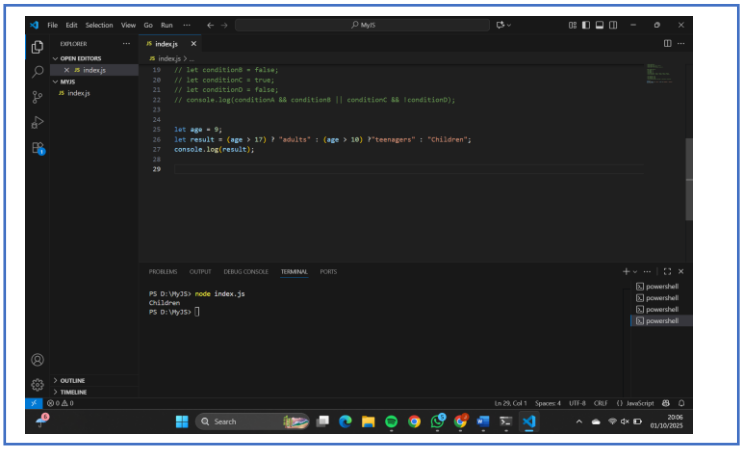
- conditionA && conditionB → true && false → **false**
- !conditionD → !false → **true**
- conditionC && !conditionD → true && true → **true**
- Jadi ekspresinya jadi: false || true → **true**

Maka outputnya adalah **true**.

d. Write the following program code!

```
let age = 9;
let result = (age > 17) ? "Adults" : (age > 10) ? "Teenagers" : "Children";
console.log(result);
```

Display the results of the program code and write your analysis in the box below !



Kode ini terdiri dari dua bagian utama:

1. Logika Boolean (Baris 19-23)

Bagian ini menghitung nilai **false** dan **true** menggunakan operator **AND (&&)** dan **OR (||)**.

- conditionB dan conditionD adalah **false**.
- Secara keseluruhan, ekspresi logika di baris 23 menghasilkan nilai **false**.

2. Penentuan Kategori Usia (Baris 25-27)

Ini adalah operator kondisional bersarang (? :) yang mengklasifikasikan usia.

- Variabel **age** disetel ke nilai **9**.
- Kode memeriksa secara berurutan:
    1. Apakah age > 17? **(9 > 17)** → **Salah** (Bukan "Adults").
    2. Apakah age > 10? **(9 > 10)** → **Salah** (Bukan "Teenagers").
- Karena kedua syarat di atas salah, nilai otomatis jatuh ke pilihan terakhir: **"Children"**.

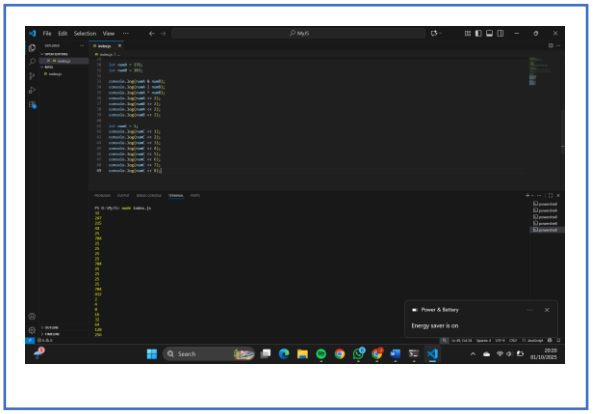
Output di terminal (Children) adalah hasil dari console.log(result) yang mencetak kategori usia 9 tahun.

e. Write the following program code!

```
let numA = 176;
let numB = 103;

console.log(numA & numB);
console.log(numA | numB);
console.log(numA ^ numB);
console.log(numA >> 2);
console.log(numB >> 2);
console.log(numA << 2);
console.log(numB << 2);

let numC = 1;
console.log(numC << 1);
console.log(numC << 2);
console.log(numC << 3);
console.log(numC << 4);
console.log(numC << 5);
console.log(numC << 6);
console.log(numC << 7);
console.log(numC << 8);
```



Kode ini menggunakan **Operasi Bitwise** pada angka. Operasi ini memanipulasi representasi biner (0s dan 1s) dari angka.

1. Logika Bitwise (&, |, ^, ~)

Ini membandingkan dua angka, **175** dan **10**.

- **& (AND), | (OR), ^ (XOR)**: Melakukan perbandingan bit per bit.
- **~ (NOT)**: Membalik semua bit (contoh: ~175 menghasilkan -176).

2. Shift Kiri (<<)

Ini mengambil angka **1** (numC) dan menggeser bit-nya ke kiri.

- Setiap kali digeser 1 posisi, nilainya setara dengan **dikali 2**.
- Baris 42 hingga 49 mencetak urutan pangkat dua: **2, 4, 8, 16, 32, 64, 128, 256**.

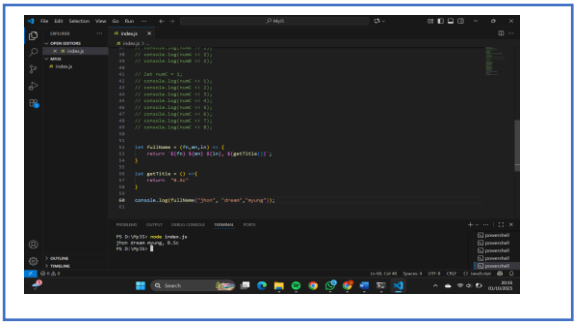
Kesimpulan: Kode ini adalah demonstrasi perhitungan dasar dalam sistem biner. *Output* di terminal menunjukkan hasil dari operasi logika dan urutan kelipatan dua.

f. Write the following program code!

```
let fullName = (fn,mn,ln) => {
    return `${fn} ${mn} ${ln}, ${getTitle()}`;
}

let getTitle = () =>{
    return "B.Sc"
}

console.log(fullName("John", "Dream","Myung"));
```



g. what is the difference between operator == and ===, give an example of the program code.

- == (**loose equality**): Compares **values**, but **allows type coercion**. That means JavaScript will try to convert the values to the same type before comparing.
- === (**strict equality**): Compares **both value and type**. No type conversion happens.

Example :

```
let a = 5;
let b = "5";

console.log(a == b);  // true → because values are equal after type coercion
console.log(a === b); // false → because number !== string
```

So, == might give you unexpected results if you're not careful with types. Use === to avoid surprises.

h. What is the difference between creating functions using functions and arrow functions?

There are several differences, but the main ones are:

1. Syntax:

Arrow functions are more concise.

```
// Traditional function
function greet(name) {
    return "Hello, " + name;
}
// Arrow function
const greet = (name) => "Hello, " + name;
```

2. this keyword behavior:

- Traditional functions have their own this.
- Arrow functions do not have their own this; they inherit it from their enclosing scope.

```
const person = {
    name: "Alex",
    traditional: function () {
        console.log("Traditional:", this.name); // refers to person
    },
    arrow: () => {
        console.log("Arrow:", this.name); // refers to outer scope (usually `window` or `undefined`)
    }
};

person.traditional(); // Traditional: Alex
person.arrow();       // Arrow: undefined
```