2. Variables and Data Types

`var`, `let`, `const`

- `var` — Function-scoped, can be redeclared.
- `let` — Block-scoped, can be updated but not redeclared.
- `const` — Block-scoped, **cannot** be updated or redeclared.

```
var name = "Divya";
let age = 30;
const city = "Nashik";

// let and const are block scoped
{
  let age = 25;  // different from outer 'age'
  console.log(age);  // 25
}
console.log(age);  // 30
```

Primitive Types

```
let str = "Hello";           // string
let num = 42;                // number
let isTeacher = true;        // boolean
let empty = null;            // null
let notDefined;              // undefined
let bigIntVal = 1234567890123456789n; // bigint
let sym = Symbol("id");      // symbol
```

Type Coercion and Conversion

```
console.log('5' + 2);      // "52" (string)
console.log('5' - 2);      // 3 (number)
console.log(Number("123")); // 123
console.log(String(123));   // "123"
```

---

3. Operators

Arithmetic

```
let x = 10, y = 3;
console.log(x + y);  // 13
console.log(x % y);  // 1
```

Assignment

```
x += 5;  // x = x + 5
```

Comparison

```
console.log(5 == "5");   // true (loose equality)
console.log(5 === "5");  // false (strict equality)
```

Logical Operators

```
console.log(true && false); // false
console.log(true || false); // true
console.log(!true);         // false
```

Ternary Operator

```
let age = 18;
let canVote = age >= 18 ? "Yes" : "No";
console.log(canVote);  // "Yes"
```

---

4. Control Flow

`if`, `else`, `else if`

```
let score = 85;

if (score > 90) {
  console.log("Excellent");
} else if (score > 75) {
  console.log("Good");
} else {
  console.log("Try again");
}
```

`switch`

```
let grade = 'B';

switch (grade) {
  case 'A':
    console.log("Excellent");
    break;
  case 'B':
    console.log("Good");
    break;
  default:
    console.log("Needs improvement");
}
```

Loops

```
// while
let i = 0;
while (i < 3) {
  console.log(i);
  i++;
}

// do...while
let j = 0;
do {
  console.log(j);
  j++;
} while (j < 3);
```

```
// for
for (let k = 0; k < 3; k++) {
  console.log(k);
}

// break and continue
for (let i = 0; i < 5; i++) {
  if (i === 2) continue;  // skip 2
  if (i === 4) break;     // exit loop at 4
  console.log(i);
}
```

---

5. Functions

Declaration vs Expression

```
function greet() {
  console.log("Hello");
}

const sayHi = function() {
  console.log("Hi");
};

greet();   // Hello
sayHi();   // Hi
```

Arrow Functions

```
const add = (a, b) => a + b;
console.log(add(2, 3));  // 5
```

Parameters and Return

```
function multiply(x, y) {
  return x * y;
}
console.log(multiply(4, 5));  // 20
```

---

6. Arrays and Objects

Creating & Accessing Arrays

```
let fruits = ["apple", "banana", "cherry"];
console.log(fruits[1]);  // banana
```

Array Methods

```
let arr = [1, 2, 3];

arr.push(4);       // [1, 2, 3, 4]
arr.pop();         // [1, 2, 3]
arr.shift();       // [2, 3]
```

```
arr.unshift(0);    // [0, 2, 3]

let sliced = arr.slice(1, 3);  // [2, 3]
arr.splice(1, 1, 99);          // [0, 99, 3]

let doubled = arr.map(x => x * 2);   // [0, 198, 6]
let even = arr.filter(x => x % 2 === 0);  // [0]
let sum = arr.reduce((acc, val) => acc + val, 0);  // 102
```

Creating & Accessing Objects

```
let person = {
  name: "Divya",
  age: 30,
  greet: function() {
    return "Hi, I'm " + this.name;
  }
};

console.log(person.name);       // Divya
console.log(person['age']);     // 30
console.log(person.greet());    // Hi, I'm Divya
```
