■ JavaScript Basics & OOP

■ 1. Variables

Definition: Variables are containers used to store data.

```
let name = "John";   // block-scoped, changeable
const age = 25;      // block-scoped, constant
var city = "Delhi";  // function-scoped, old way
```

Example:

```
let x = 10;
const y = 20;
console.log(x + y); // 30
```

■ 2. Conditional Statements

Definition: Used to make decisions in code.

```
if (condition) {
   // code
} else if (anotherCondition) {
   // code
} else {
   // code
}
```

Example:

```
let age = 18;
if (age >= 18) {
  console.log("You can vote");
} else {
  console.log("You cannot vote");
}
```

■ 3. Looping Statements

Definition: Loops are used to repeat a block of code.

```
for (let i = 1; i <= 5; i++) {
  console.log(i);
}

let i = 1;
while (i <= 5) {
  console.log(i);
```

```
  i++;
}

let j = 1;
do {
  console.log(j);
  j++;
} while (j <= 5);
```

■ 4. Functions

Definition: Functions are blocks of code that perform a task.

Types:

```
// Function Declaration
function add(a, b) {
  return a + b;
}
console.log(add(2, 3)); // 5

// Function Expression
const multiply = function(a, b) {
  return a * b;
};
console.log(multiply(2, 3)); // 6

// Arrow Function
const square = (x) => x * x;
console.log(square(4)); // 16

// Anonymous Function
setTimeout(function() {
  console.log("Hello after 1 sec");
}, 1000);
```

■ 5. OOP Concepts in JavaScript

Class & Object

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  greet() {
    console.log(`Hello, I am ${this.name}`);
  }
```

```
}

let p1 = new Person("John", 22);
p1.greet(); // Hello, I am John
```

Inheritance

```
class Animal {
  sound() {
    console.log("Animal makes a sound");
  }
}

class Dog extends Animal {
  sound() {
    console.log("Dog barks");
  }
}

let d = new Dog();
d.sound(); // Dog barks
```

Polymorphism

```
class Shape {
  area() {
    return 0;
  }
}

class Circle extends Shape {
  constructor(r) {
    super();
    this.r = r;
  }
  area() {
    return Math.PI * this.r * this.r;
  }
}

let s1 = new Circle(5);
console.log(s1.area()); // 78.5
```

Encapsulation

```
class BankAccount {
  #balance = 0; // private property

  deposit(amount) {
    this.#balance += amount;
  }
```

```
  getBalance() {
    return this.#balance;
  }
}

let acc = new BankAccount();
acc.deposit(500);
console.log(acc.getBalance()); // 500
```

Abstraction

```
class Car {
  startEngine() {
    this.#fuelCheck();
    console.log("Engine started");
  }
  #fuelCheck() { // private method
    console.log("Fuel checked");
  }
}

let c = new Car();
c.startEngine();
// Output: Fuel checked
//         Engine started
```