1. JavaScript Basics

1.1 Variables and Data Types
1.2 Type Conversion
1.3 Operators and Expressions
1.4 Control Flow Statements

2. Functions and Execution

2.1 Function Declarations and Expressions
2.2 Arrow Functions
2.3 Default Parameters
2.4 The `this` Keyword

3. Data Structures and Manipulation

3.1 Arrays and Objects
3.2 Destructuring (ES6)
3.3 Spread and Rest Operators (ES6)
3.4 Template Literals

4. Array Utilities and ES6 Methods

4.1 Common Array Methods:

- `map()`
- `filter()`
- `reduce()`
- `forEach()`

5. Asynchronous JavaScript

5.1 Callbacks
5.2 Promises
5.3 Async and Await
5.4 Error Handling with try...catch

6. Advanced JavaScript Concepts

6.1 Closures
6.2 Hoisting
6.3 ES6 Modules (import/export)
6.4 Classes and Inheritance
6.5 Event Loop and Call Stack

1. JavaScript Basics

1.1 Variables and Data Types

JavaScript uses `let`, `const`, and the older `var` to declare variables. `let` and `const` are block-scoped and prevent common bugs. `const` is used for constants or functions that don't need reassignment.

Example:

```
js
CopyEdit
let name = "John";
const age = 25;
```

Data Types include:

- Primitive: `string`, `number`, `boolean`, `null`, `undefined`
- Reference: `object`, `array`, `function`

1.2 Type Conversion

JavaScript allows conversion between data types using functions like `Number()`, `String()`, or automatic (implicit) conversions.

Example:

```
js
CopyEdit
let num = "123";
let converted = Number(num);  // 123
```

1.3 Operators and Expressions

Operators are used for arithmetic, comparison, logical decisions, and assignments.

Example:

```
js
CopyEdit
let a = 5;
let b = 10;
let result = a > b ? "A is greater" : "B is greater";
```

1.4 Control Flow Statements

Conditional logic uses `if`, `else`, `switch`. Loops like `for`, `while` repeat actions. `break` and `continue` modify loop flow.

Example:

```
js
CopyEdit
for (let i = 1; i <= 5; i++) {
  if (i === 3) continue;
  console.log(i);
}
```

---

2. Functions and Execution

2.1 Function Declarations & Expressions

Functions organize code into reusable blocks. You can define functions traditionally or as expressions.

Declaration:

```
js
CopyEdit
function greet(name) {
  return "Hello " + name;
}
```

Expression:

```
js
CopyEdit
const greet = function(name) {
  return "Hello " + name;
};
```

2.2 Arrow Functions (ES6)

Arrow functions offer a shorter syntax and inherit `this` from the parent scope.

Example:

```
js
CopyEdit
const sum = (a, b) => a + b;
```

2.3 Default Parameters (ES6)

You can set default values for function parameters.

Example:

```
js
CopyEdit
function greet(name = "Guest") {
  console.log("Hello, " + name);
}
```

2.4 The `this` Keyword

The value of `this` refers to the object from which the function is called.

Example:

```
js
CopyEdit
const person = {
  name: "Alice",
  greet() {
    console.log("Hi, I'm " + this.name);
  }
};
person.greet(); // Hi, I'm Alice
```

---

3. Data Structures and Manipulation

3.1 Arrays and Objects

Arrays hold lists of values, objects store key-value pairs.

Example (Array):

```
js
CopyEdit
let colors = ["red", "blue", "green"];
```

Example (Object):

```
js
CopyEdit
let user = { name: "Nikhil", age: 24 };
```

3.2 Destructuring (ES6)

Destructuring allows extracting values from arrays or objects into variables.

Example (Object):

```
js
CopyEdit
const { name, age } = user;
```

Example (Array):

```
js
CopyEdit
const [first, second] = colors;
```

3.3 Spread and Rest Operators

The spread operator (. . .) expands arrays or objects. The rest operator groups values.

Spread Example:

```
js
CopyEdit
const nums = [1, 2, 3];
const newNums = [...nums, 4, 5];  // [1, 2, 3, 4, 5]
```

Rest Example:

```
js
CopyEdit
function add(...args) {
  return args.reduce((a, b) => a + b);
}
```

3.4 Template Literals

Template literals use backticks and `${}` for easy string building.

Example:

```
js
CopyEdit
const name = "Nikhil";
console.log(`Hello, ${name}`);
```

---

4. Array Utilities and ES6 Methods

4.1 Common Array Methods

- `map()`: Transforms elements
- `filter()`: Filters based on a condition
- `reduce()`: Accumulates values
- `forEach()`: Loops through elements

Example:

```
js
CopyEdit
const nums = [1, 2, 3];
const doubled = nums.map(n => n * 2);  // [2, 4, 6]
```

---

5. Asynchronous JavaScript

5.1 Callbacks

A callback is a function passed into another function to run later.

Example:

```
js
CopyEdit
```

```
function greet(name, callback) {
  console.log("Hi " + name);
  callback();
}
greet("Nikhil", () => console.log("Callback executed"));
```

5.2 Promises

Promises represent future values with states: pending, resolved, rejected.

Example:

```
js
CopyEdit
let promise = new Promise((resolve, reject) => {
  resolve("Success");
});

promise
  .then(data => console.log(data))
  .catch(err => console.log(err));
```

5.3 Async and Await

`async` declares a function that returns a promise. `await` pauses execution until the promise resolves.

Example:

```
js
CopyEdit
function fetchData() {
  return new Promise(resolve => setTimeout(() => resolve("Done"), 2000));
}

async function loadData() {
  const result = await fetchData();
  console.log(result);
}
loadData();
```

5.4 try...catch for Error Handling

Use `try...catch` to handle errors cleanly, especially with `async/await`.

Example:

```
js
CopyEdit
async function load() {
  try {
    let data = await fetchData();
    console.log(data);
  } catch (err) {
    console.log("Error:", err);
  }
}
```

6. Advanced Concepts

6.1 Closures

A closure is a function that remembers the scope in which it was created.

Example:

```
js
CopyEdit
function outer() {
  let count = 0;
  return function inner() {
    return ++count;
  };
}

const counter = outer();
console.log(counter()); // 1
console.log(counter()); // 2
```

6.2 Hoisting

JavaScript moves function and variable declarations to the top of their scope before code runs.

Example:

```
js
CopyEdit
console.log(x); // undefined
var x = 5;
```

6.3 ES6 Modules

Use `export` and `import` to split code into separate files.

math.js:

```
js
CopyEdit
export function add(a, b) {
  return a + b;
}
```

main.js:

```
js
CopyEdit
import { add } from './math.js';
console.log(add(2, 3));
```

6.4 Classes and Inheritance

ES6 classes offer a cleaner way to write constructor-based code and inheritance.

Example:

```
js
CopyEdit
class Animal {
  constructor(name) {
    this.name = name;
  }
  speak() {
    console.log(this.name + " makes a sound.");
  }
}

class Dog extends Animal {
  speak() {
    console.log(this.name + " barks.");
  }
}
```

6.5 Event Loop and Call Stack

The event loop handles asynchronous operations in JavaScript by managing the call stack and task queue. It ensures non-blocking behavior using callbacks and promises.

Final Learning Order:

```
markdown
CopyEdit
1. JavaScript Basics
2. Functions and Execution
3. Data Structures (Arrays, Objects)
4. ES6 Enhancements (Arrow, Destructuring, Spread/Rest)
5. Array Methods
6. Async JavaScript (Callbacks → Promises → Async/Await)
7. Error Handling
8. Advanced Concepts (Closures, Classes, Event Loop)
```

Dotask()