**Destructuring**

\* Extract values from arrays or objects into variables easily.

```js

const [first, second] = [10, 20];

const {name, age} = {name: "Alice", age: 25};

```

**Promises**

\* Manage asynchronous operations.

\* States: \*\*pending\*\*, \*\*fulfilled\*\*, \*\*rejected\*\*.

```js

fetch('url')

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error(error));

```

**Async/Await**

\* Syntactic sugar over Promises.

\* Write async code that looks synchronous.

```js

async function fetchData() {

try {

const response = await fetch('url');

const data = await response.json();

console.log(data);

} catch (error) {

console.error(error);

}

}

**🔹 1. Destructuring in JavaScript**

Destructuring allows you to **unpack values** from arrays or objects.

**✅ Object Destructuring**

const user = { name: "Janvi", age: 20 };

const { name, age } = user;

console.log(name); // Janvi

✅ **Array Destructuring**

const nums = [10, 20, 30];

const [a, b] = nums;

console.log(a); // 10

**🔹 2. Promises in JavaScript**

A **Promise** represents a value that may be available **now**, **later**, or **never**.

**✅ Basic Promise Example**

const fetchData = () => {

return new Promise((resolve, reject) => {

setTimeout(() => resolve({ id: 1, name: "Data" }), 1000);

});

};

fetchData().then(data => console.log(data));

**🔹 3. Async/Await (ES8)**

async/await is a cleaner way to handle Promises.

**✅ Example:**

async function getData() {

const response = await fetchData();

console.log(response);

}

getData();

**🔹 4. Destructuring with Async/Await**

You can **destructure the result of a Promise** directly inside an async function.

**✅ Example with Object**

const fetchUser = () => {

return new Promise(resolve => {

setTimeout(() => resolve({ name: "Janvi", age: 21 }), 1000);

});

};

async function showUser() {

const { name, age } = await fetchUser();

console.log(name); // Janvi

console.log(age); // 21

}

showUser();

✅ **Example with Array Destructuring**

const getCoords = () => {

return new Promise(resolve => {

setTimeout(() => resolve([40.7128, -74.0060]), 1000);

});

};

async function printCoords() {

const [lat, lng] = await getCoords();

console.log(`Latitude: ${lat}, Longitude: ${lng}`);

}

printCoords();

🔹 **5. Error Handling in Async/Await**

async function safeFetch() {

try {

const data = await fetchData(); // assume it may fail

console.log(data);

} catch (error) {

console.error("Failed to fetch:", error);

}

}