■ Notes – Promises in JavaScript

■ What is a Promise?

A Promise represents a value that will be available in the future. It is used for asynchronous operations (like API calls, file reading, setTimeout).

■ States of a Promise

1. Pending → waiting (not completed yet).
2. Fulfilled (Resolved) → completed successfully.
3. Rejected → failed with an error.

■ Creating a Promise

```
let myPromise = new Promise((resolve, reject) => {
let success = true;
if (success) {
resolve('Success!'); // fulfilled
} else {
reject('Error!'); // rejected
}
});
```

■ Consuming a Promise

```
myPromise
.then(result => console.log(result)) // runs on resolve
.catch(error => console.log(error)); // runs on reject
```

■ Example with setTimeout

```
function getData() {
return new Promise((resolve) => {
setTimeout(() => {
resolve('Data received!');
}, 2000);
});
}

getData().then(data => console.log(data));
```

■ Output after 2 sec: Data received!

■ Why use Promises?

- Handle async code cleanly.
- Avoid callback hell.

■ Shortcut meaning: Promise = A guarantee of future value (success or failure).