Course: Full Stack Development

Level: BTEC Grade 12 IT

Topic: JavaScript Programming

PART A: JAVASCRIPT BASICS

1. What is JavaScript?

JavaScript (JS) is a **high-level, interpreted programming language** used mainly to create dynamic and interactive web pages.
It runs directly in the browser (client-side) but can also run on servers using Node.js.

Example:

```
<script>
  alert("Welcome to JavaScript!");
</script>
```

This small script displays a pop-up message on a webpage.

2. Variables and Data Types

Variables store data values. In JavaScript, you can declare variables using `var`, `let`, or `const`.

Example:

```
let name = "Telesphore";
const age = 18;
var isStudent = true;
```

Data Types:

- **String** – text, e.g., `"Hello"`
- **Number** – numeric values, e.g., `25`
- **Boolean** – true or false

- **Array** – list of items, e.g., `["HTML", "CSS", "JS"]`
- **Object** – stores data in key-value pairs, e.g., `{name: "John", age: 20}`

3. Operators

JavaScript supports:

- **Arithmetic:** `+ - * / %`
- **Comparison:** `== === != > <`
- **Logical:** `&& || !`
- **Assignment:** `= += -=`

Example:

```
let x = 10;
let y = 5;
console.log(x + y); // Output: 15
```

4. Conditional Statements

Conditional statements control decisions in a program.

Example:

```
let age = 18;
if (age >= 18) {
  console.log("You are an adult.");
} else {
  console.log("You are underage.");
}
```

5. Loops

Loops repeat blocks of code.

Example (For Loop):

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

Example (While Loop):

```
let count = 1;
while (count <= 3) {
  console.log("Count is " + count);
  count++;
}
```

6. Functions

Functions group reusable blocks of code.

Example:

```
function greet(name) {
  return "Hello, " + name + "!";
}
console.log(greet("King David Student"));
```

7. Arrays and Objects

Array Example:

```
let fruits = ["Apple", "Banana", "Orange"];
console.log(fruits[1]); // Output: Banana
```

Object Example:

```
let student = {
  name: "Alice",
  age: 17,
  class: "IT12"
};
console.log(student.name); // Output: Alice
```

PART B: JAVASCRIPT FOR WEB DEVELOPMENT (FRONT-END)

1. The Role of JavaScript in Web Pages

JavaScript is used to **make web pages interactive** — it works with HTML and CSS to create animations, forms, and dynamic content.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Interactive Page</title>
</head>
<body>
  <button onclick="sayHello()">Click Me!</button>

  <script>
    function sayHello() {
      alert("Hello, welcome to Full Stack Development!");
    }
  </script>
</body>
</html>
```

2. The Document Object Model (DOM)

The DOM represents an HTML document as a tree structure that JavaScript can manipulate.

Example – Changing Text:

```
<p id="message">Original text</p>
<button onclick="changeText()">Change Text</button>

<script>
  function changeText() {
    document.getElementById("message").innerHTML = "Text has been changed!";
  }
</script>
```

3. Events in JavaScript

Events are actions like clicks, typing, or hovering that trigger JavaScript functions.

Example – Mouseover Event:

```
<h2 onmouseover="highlight(this)" onmouseout="removeHighlight(this)">Hover over me!</h2>

<script>
  function highlight(element) {
    element.style.color = "red";
  }
  function removeHighlight(element) {
    element.style.color = "black";
  }
</script>
```

4. Forms and Validation

JavaScript can check if users fill out forms correctly before submission.

Example:

```
<form onsubmit="return validateForm()">
  Name: <input type="text" id="name">
  <input type="submit" value="Submit">
</form>

<script>
  function validateForm() {
    let name = document.getElementById("name").value;
    if (name == "") {
      alert("Please enter your name");
      return false;
    }
    alert("Form submitted successfully!");
    return true;
  }
</script>
```

5. Interacting with CSS

JavaScript can also change the styles of HTML elements.

Example:

```
<p id="text">Change my color!</p>
<button onclick="changeColor()">Click</button>
```

```
<script>
  function changeColor() {
    document.getElementById("text").style.color = "blue";
  }
</script>
```

PART C: JAVASCRIPT FOR FULL-STACK DEVELOPMENT (NODE.JS)

1. What is Node.js?

Node.js is a **runtime environment** that allows JavaScript to run on the **server-side** — outside the browser.
It helps developers create full-stack web applications using a single language: JavaScript.

Example:

```
// app.js
console.log("Hello from Node.js!");
```

You can run this file with:

```
node app.js
```

2. Creating a Simple Web Server

Node.js can create servers that respond to client requests.

Example:

```
const http = require('http');

const server = http.createServer((req, res) => {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Welcome to my Node.js Server!');
});

server.listen(3000, () => {
  console.log('Server running on port 3000');
});
```

When you open `http://localhost:3000` in your browser, you'll see the message.

3. Working with Modules

Modules help organize code in Node.js.

Example:

```
// math.js
exports.add = function(a, b) {
  return a + b;
}

// app.js
const math = require('./math');
console.log(math.add(5, 10)); // Output: 15
```

4. Reading and Writing Files

Node.js can handle files using the built-in `fs` (File System) module.

Example:

```
const fs = require('fs');

fs.writeFileSync('message.txt', 'Hello, JavaScript Full Stack!');
let content = fs.readFileSync('message.txt', 'utf8');
console.log(content);
```

5. Using Express.js (Simple API)

Express.js is a framework that simplifies building APIs with Node.js.

Example:

```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.send('Hello from Express.js!');
});

app.listen(3000, () => console.log('Server running on port 3000'));
```

6. Connecting to a Database (MongoDB)

Full-stack developers often connect Node.js to databases.

Example:

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/myDB', {useNewUrlParser: true});

const userSchema = new mongoose.Schema({ name: String, age: Number });
const User = mongoose.model('User', userSchema);

const newUser = new User({ name: "Alice", age: 22 });
newUser.save();
```

7. Full Stack Integration

In full-stack projects:

- **Front-end:** React, Vue, or Angular
- **Back-end:** Node.js / Express
- **Database:** MongoDB or MySQL

These parts communicate using **APIs (Application Programming Interfaces)**.