Page 1: Introduction to JavaScript

JavaScript (JS) is a lightweight, interpreted, and versatile programming language primarily used to create dynamic and interactive web pages.

It is part of the core web technologies, along with HTML and CSS.

Key Features:

- Dynamic typing: Variables can hold different data types at different times.
- Event-driven: Handles user interactions like clicks, hovers, and keypresses.
- Prototype-based: Uses objects for inheritance and extensibility.
- Runs in the browser: Executed by web browsers.

Example:

```
<script>
  alert("Hello, JavaScript!");
</script>
```

Page 2: JavaScript Basics

Variables:

- var: Function-scoped.

- let: Block-scoped.

- const: Block-scoped and immutable.

Example:

```
let name = "John";

const age = 30;

console.log(name, age);
```

Data Types:

- Primitive: String, Number, Boolean, Null, Undefined, Symbol.

- Reference: Objects, Arrays, Functions.

Example:

```
let fruits = ["apple", "banana"];

console.log(fruits[0]);
```

Page 3: Functions and Scope

Functions:

- Reusable blocks of code.

- Can accept parameters and return values.

Example:

```
function greet(name) {
  return `Hello, ${name}!`;
}
console.log(greet("Alice"));
```

Scope:

- Global: Accessible everywhere.

- Local: Accessible within a block or function.

- Lexical: Determined by code nesting.

Example:

```
{
  let localVar = "I'm local";
  console.log(localVar); // Works
}
console.log(localVar); // Error
```

Page 4: DOM Manipulation

The Document Object Model (DOM) allows JavaScript to interact with HTML elements.

Example:

```
document.getElementById("myButton").addEventListener("click", function() {
  alert("Button clicked!");
});
```

Common Methods:

- getElementById: Select an element by ID.
- querySelector: Select elements using CSS selectors.
- createElement: Create new HTML elements dynamically.

Example:

```
let newDiv = document.createElement("div");
newDiv.textContent = "Hello!";
document.body.appendChild(newDiv);
```

Page 5: Advanced JavaScript Concepts

1. Asynchronous Programming:

- Promises: Handle asynchronous operations.

Example:

```
fetch("https://api.example.com/data")
  .then(response => response.json())
  .then(data => console.log(data));
```

- Async/Await: Syntactic sugar for Promises.

Example:

```
async function fetchData() {
  let response = await fetch("https://api.example.com/data");
  let data = await response.json();
  console.log(data);
}
```

2. ES6+ Features:

- Arrow functions, template literals, destructuring, modules, etc.

Example:

```
let sum = (a, b) => a + b;
console.log(sum(3, 4));
```

3. Event Loop:

- Handles asynchronous tasks and events using the call stack and task queue.