Book Report

Javascript Basics

Rifah Sajida Deya

2nd July, 2025

Introduction

JavaScript (JS) is a high-level, dynamic, interpreted (or just-in-time compiled) programming language that is a core technology of the World Wide Web, alongside HTML and CSS. It enables developers to create dynamic, interactive, and responsive web pages by allowing scripts to run directly in the browser. JavaScript is multi-paradigm, supporting object-oriented, functional, and imperative programming styles. It features dynamic typing, prototype-based object-orientation, and first-class functions, meaning functions can be treated as variables, passed as arguments, or returned from other functions.

While JavaScript is best known for its role in client-side web development—manipulating the Document Object Model (DOM), handling events, and updating web content in real time—it is also widely used on the server side (for example, with Node.js) and in non-browser environments

Key characteristics:

- Lightweight and interpreted: Runs directly in browsers without pre-compilation.
- Event-driven and asynchronous: Efficiently handles user interactions and background tasks.
- Platform-independent: Supported by all major browsers and many server environments.
- Extensible: Enhanced by numerous libraries and APIs for varied tasks, from animation to networking.

Middleware

Middleware is software that acts as an intermediary layer between different parts of an application, facilitating communication, data management, and additional processing. In web development, especially with frameworks like Express.js, middleware refers to functions that process HTTP requests before they reach the final route handler. Middleware can:

- Modify request and response objects
- Perform tasks such as authentication, logging, or error handling
- End the request-response cycle or pass control to the next middleware using the `next()` function

Middleware is essential for separating concerns and managing complex application logic efficiently

MongoDB

MongoDB is a popular, open-source, document-oriented NoSQL database. Instead of storing data in tables and rows (as in relational databases), MongoDB stores data in flexible, JSON-like documents within collections. This allows for efficient handling of unstructured or semi-structured data, scalability, and high performance. MongoDB is widely used in modern applications for scenarios such as content management, e-commerce, analytics, IoT, and more.
Key features:

- Schema-less (flexible data models)
- High scalability (vertical and horizontal)
- Fast read/write operations
- Stores data as BSON (Binary JSON) documents

MVC (Model-View-Controller)

MVC is an architectural pattern that separates an application into three main components

Component	Role
Model	Manages data and business logic; interacts with the database
View	Handles the user interface and presentation of data
Controller	Receives user input, processes requests, updates the model, and selects the view

Benefits:

- Separation of concerns (easier maintenance and testing)
- Parallel development (teams can work on different components)
- Reusability and scalability

Client-Server Model

The client-server model is a distributed application structure where

- Clients (e.g., browsers, mobile apps) initiate requests for resources or services.
- Servers provide those resources or services, process client requests, and send back responses.

Clients and servers typically communicate over a network. The server may handle multiple clients simultaneously, and the architecture supports scaling and resource sharing.

Aspect	**MVC (Model-View-Controller)**	**Client-Server Architecture**
Definition	A software design pattern that separates an application into three components: Model (data), View (UI), and Controller (logic).	A network architecture where clients (users) send requests to a server, which processes and responds.
Purpose	Organizes code within a single application for maintainability and separation of concerns.	Structures distributed applications, separating service requesters (clients) from service providers (servers).
Scope	Focuses on how code is structured inside an application.	Focuses on how systems communicate over a network.
Components	- Model: Handles data and business logic - View: Handles UI and presentation - Controller: Handles input and updates Model/View	- Client: Initiates requests (e.g., browser, app) - Server: Listens, processes, and responds to requests
Interaction	Components interact within the same application/process.	Clients and servers interact over a network (often HTTP).

Example	Web app where MVC separates database logic, UI, and request handling.	Web browser (client) requests a web page from a web server.
Relationship	MVC can be used inside a client, a server, or both; it's about internal structure.	Client-server is about system architecture and data exchange.
Usage Together	You can use MVC within the server of a client-server system to organize code.	MVC does not replace client-server; they address different concerns.

Analogy:
Think of client-server as the city's postal system (how mail is sent and received between houses), and MVC as how you organize your mail inside your house (where you keep bills, invitations, and junk mail separately).

How Express.js Works with MongoDB

Express.js is a minimal and flexible Node.js web application framework that simplifies building web servers and APIs. MongoDB serves as the database for storing and retrieving application data. Here's how they work together:

- Express handles incoming HTTP requests, routes them to appropriate handlers, and sends responses.
- Middleware in Express processes requests (e.g., authentication, validation) before they reach route handlers.
- To interact with MongoDB, Express uses the official MongoDB Node.js driver or an ODM (Object Data Modeling) library like Mongoose.

- Route handlers in Express perform CRUD (Create, Read, Update, Delete) operations on MongoDB collections/documents, then send results back to the client.

Example flow:

1. Client sends a POST request to add a user.
2. Express receives the request, passes it through middleware (for validation/auth).
3. The route handler uses the MongoDB driver to insert the user document into a collection.
4. Express sends a response with the result.

This integration allows developers to build RESTful APIs and dynamic web applications with efficient data storage and retrieval.

Promise:

A Promise in JavaScript is an object that represents the eventual completion (success) or failure of an asynchronous operation and its resulting value. It acts as a proxy for a value that may not be available yet, allowing you to write asynchronous code that looks and behaves more like synchronous code. A promise can be in one of three states:

- Pending: Initial state, neither fulfilled nor rejected.
- Fulfilled: The operation completed successfully.
- Rejected: The operation failed.

When a promise is fulfilled or rejected, handlers attached with `.then()` (for success) or `.catch()` (for errors) are called to handle the result or error.

Callback Function:

A callback function is a function passed as an argument to another function, which is then invoked inside the outer function to complete some kind of routine or action. Callbacks are a fundamental way to handle asynchronous operations or to execute code after another function has finished its task. They can be executed synchronously (immediately) or asynchronously (after some time or event).

For example, callbacks are commonly used with array methods (`forEach`, `map`) and asynchronous operations like `setTimeout` or handling events.

Event Handling:

Event handling in JavaScript is the process of listening for and responding to events—actions that happen in the browser, such as user clicks, key presses, or page loads. When an event occurs on an HTML element, an event handler (or event listener) is a function that executes in response. Event handlers are attached to elements using methods like `addEventListener`, or by assigning a function to an element's `onevent` property (e.g., `onclick`). The event handler receives an event object containing details about the event, allowing developers to create interactive and dynamic web pages.

Conclusion

Thank you for reading and all the best 🌸