St. JOSEPH'S COLLEGE OF ENGINEERING
(An Autonomous Institution)
St. JOSEPH'S INSTITUTE OF TECHNOLOGY
(An Autonomous Institution)
St. JOSEPH'S GROUP OF INSTITUTIONS
OMR, Chennai - 119

Placement Empowerment Program (PEP)
Full Stack Web Development - Syllabus

HTML – Hyper Text Markup Language

- Introduction to HTML
- HTML Document Structure
- Text Formatting
- Creating Lists
- Adding Links
- Working with Images
- Creating Tables
- Iframes
- HTML Forms
- HTML Media

Practice Exercises

- Exercise 1: Create a simple HTML page with headings, paragraphs, and lists.
- Exercise 2: Add links and images to the HTML page.
- Exercise 3: Construct a basic table to display tabular data.
- Exercise 4: Build a form with different form elements.

CSS – Cascading Style Sheets

- Introduction to CSS
- CSS Syntax: Selectors, properties, and values.
- Inline CSS, Internal CSS, External CSS
- CSS Selectors
- Basic Styling Properties
- Box Model
- CSS Display
- CSS Positioning

Practice Exercises

- Exercise 1: Apply inline CSS styles to specific HTML elements.
- Exercise 2: Create an internal CSS and external CSS and apply styles to HTML elements.
- Exercise 3: Practice using different CSS selectors to target specific elements.
- Exercise 4 Style a webpage using various CSS properties and experiment with the box model.

Advanced Selectors

- CSS Flexbox
- CSS Grid
- CSS Transitions
- Responsive CSS

Practice Exercises

- Exercise 1: Apply advanced selectors and pseudo-classes to style specific elements.
- Exercise 2: Implement a responsive design using media queries.
- Exercise 3: Create a flexible layout using CSS flexbox.
- Exercise 4: Build a grid-based layout using CSS grid.

Java Script

- Introduction to JavaScript
- JavaScript Syntax
- Variables, data types, operators, and basic expressions
- Control Flow: Conditional statements (if-else, switch) and loops (for, while).
- Functions: Defining functions, parameters, function call and return statements.
- JS Strings and String Methods
- JS Arrays and Array Methods

Practice Exercises

- Write JavaScript functions to perform simple calculations and solve basic problems like isEven, findMax, isPalindrome, factorial, reverse string.
- Create 2 new arrays called 'vegetables' with 'carrot', 'broccoli', 'spinach' and 'fruits' with 'apple', 'mango', 'banana'.

- Combine the 'fruits' and 'vegetables' arrays into a single array called 'groceries' and log the 'groceries' array.
- Sort the 'groceries' array alphabetically and log the sorted array.
- Reverse the order of the 'groceries' array and log the reversed array.

- Create a new array called 'numbers' with a sequence of numbers from 1 to 5.
  - Use a loop to iterate through the 'numbers' array and log each number
  - Create a new array called 'squaredNumbers' by squaring each number from the 'numbers'. Log the 'squaredNumbers' array.
  - Use the 'map' method to create a new array called 'doubledNumbers' by doubling each number from the 'numbers' array. Log the 'doubledNumbers' array.
  - Use the 'filter' method to create a new array called 'evenNumbers' that contains only the even numbers from the 'numbers' array. Log the 'evenNumbers' array.
  - Use the 'reduce' method to calculate the sum of all numbers in the 'numbers' array. Log the sum.
- Create an array called 'mixed' that contains a mix of strings, numbers, and boolean values.
  - Use the 'filter' method to create a new array called 'onlyStrings' that contains only the string elements from the 'mixed' array. Log the 'onlyStrings' array.

JS Objects

- Object Definitions, Properties, Methods
- Object Constructors
- Object Iterables
- Object Sets and Maps

Practice Exercises

- Create an object representing a book with properties for title, author, and year. // Log the book object to the console.
- Create an array of book objects, each representing a different book. // You should have at least 3 books in the array. // Log the array of book objects to the console.
- Write a function that takes a person object as a parameter and logs their information. // The function should log their name, age, city, and occupation (if available).
- Write a function that takes an array of book objects as a parameter and logs information about each book. // The function should loop through the array and log the title, author, and year of each book.

- Create a new object representing a car with properties for make, model, and year.// Log the car object to the console. Also Add a method to the car object that calculates the age of the car based on the current year. //The method should return the age of the car. Call the method you created to calculate the age of the car and log it to the console.

JS HTML DOM

- DOM Manipulation: Accessing and modifying DOM elements using JavaScript.
- Event Handling: Adding event listeners and responding to user interactions

Practice Exercises

- Build an interactive webpage using JavaScript to modify DOM elements and respond to user events.
- Get a reference to the "Change Text" button and the paragraph element with the id "message." Add a click event listener to the "Change Text" button. When clicked, change the text of the paragraph to "Text changed!"
- Get a reference to the "Change Color" button and the <h1> element. Add a click event listener to the "Change Color" button. When clicked, change the text color of the <h1> element to a random color. You can generate a random color using JavaScript.
- Get a reference to the "Load Image" button. Add a click event listener to the button. When clicked, load an image dynamically and display it within the with the id "imageContainer."
- Create an <img> element dynamically using JavaScript. Set its src attribute to the URL of an image of your choice. Append the <img> element to the "imageContainer" to display the image when the button is clicked. Optionally, add a "Remove Image" button below the loaded image. When clicked, remove the image from the DOM.
- Image Manipulation
    - Add buttons to resize the image to different dimensions (e.g., small, medium, large).
    - Add buttons to apply image filters (e.g., grayscale, sepia) using CSS classes.
    - Shake the image

Asynchronous Java Script

- Introduction to Asynchronous JavaScript: Overview of asynchronous programming concepts
- Callback Functions: Understanding callback functions and their use in asynchronous operations.

Practice Exercise

- Use callbacks to handle asynchronous operations, such as making API calls and updating the DOM based on the response.

Fetch API, JSON and Error Handling

- Working with Fetch API to make HTTP requests and handle responses
- JSON: Syntax, Data types, Parse and Stringify, Objects and Arrays
- Error Handling: Handling errors in asynchronous JavaScript.

Practice Exercises

- Make a GET request to the JSONPlaceholder API to fetch a list of users.
    - Parse the JSON response and extract user data (e.g., name, email) for each user.
    - Create <li> elements for each user and append them to the <ul> with the id "userList." Display at least 5 users.
    - Handle any errors that may occur during the API request and display an error message if needed.
    - Display more user details (e.g., phone number, address).
    - Add pagination to load and display more users.
    - Implement search functionality to filter users by name or email.
    - Style the user list to make it more visually appealing.
- Implement error handling in your previous exercises, handle different types of errors, and display appropriate error messages

React

- Introduction to React, setting up a React Development Environment: Installation and configuration of React tools.
- React Components: Class & Functional
- Component Lifecycle Methods
- ES6 & JSX Syntax: Introduction to JSX syntax and its use in building React components.
- Rendering Elements: Rendering React elements to the DOM.
- Component Props

Practice Exercise

- Build a simple React component that displays a list of items passed as props and handles user interactions such as clicking on an item.

States & Events

- Component State: Understanding state in React components.
- Updating State: Modifying component state using setState().
- Component Lifecycle: Overview of component lifecycle methods in React.

- Handling Events: Adding event handlers to React components.
- Conditional Rendering: Conditionally rendering components based on certain conditions and React routing

Practice Exercises

- Create a React component that includes a counter functionality. Implement the logic to increment and decrement the counter value using state and event handlers.
- Create routes between React components and add navigation links with styles applied

React Forms & Styling

- Controlled Components: Managing form input state in React.
- Form Submission: Handling form submission and validation in React.
- Data Flow in React: Understanding the unidirectional data flow in React.
- React Inline Styling, CSS in JS libraries
- Components Interacting and Conditional/Dynamic Rendering

Practice Exercises

- Build a simple form component that captures user input and displays the submitted data on the screen using React state and controlled components.
- Build a React application that includes styled components, demonstrates components interacting and passing data between them, and utilizes conditional/dynamic rendering based on user input or data.

Node.js

- Understanding Node.js
- Package management and NPM
- Callbacks-Promises- Async-Await
- Event Loop - Event Emitter
- File System
- Timers in Node JS
- MySQL - Manipulating and Accessing MySQL Database from Node.js

Practice Exercise

- Develop a Command Line Application for an online super market using NodeJS & MySQL to perform: a) search based on product id or name b) On retrieving the results, display the product details of different brands in table format with the Price field in sorted order

Express.js

- Express Framework
- Configuring Routes
- Express JS Request – Response – GET - POST
- Processing URLs - Processing Query Strings and Form Parameters
- Cookies & Sessions
- REST API - Rendering JSON Data

Practice Exercise

Create a basic CRUD operation API by following REST syntax for a given model student with the following fields [field names] using MySQL & Express JS

MongoDB

Introduction and Setup

- Overview of MongoDB and its role in modern applications.
- Installation of MongoDB on various platforms (Windows, macOS, Linux).
- Configuration and setup of a local MongoDB server.

Basic CRUD Operations

- Creating a MongoDB database and collections.
- Inserting documents and performing basic CRUD operations & Querying for documents using different operators and criteria.

Practice Exercise

Developing a hands-on project demonstrating CRUD operations

Mongo DB – Optional Topics

Advanced CRUD Operations and Indexing

- Working with complex data types and arrays in MongoDB.
- Understanding and creating indexes for efficient querying.

Aggregation Framework and Pipeline

- MongoDB's Aggregation Framework & Using aggregation pipeline stages for data aggregation and transformation & Aggregating data from multiple collections and sources.

Projects

HTML Projects

1) Create a basic webpage with headings, paragraphs, and links.
2) Build a form for user input and feedback.
3) Design a webpage with multimedia elements (images, videos, audio).
4) Develop a webpage using semantic HTML5 elements.
5) Build a table displaying information.

CSS Projects

1) Style a webpage using basic CSS.
2) Create a responsive webpage using CSS grid and flexbox.
3) Design a webpage with CSS animations and transitions.
4) Implement a navigation bar with dropdown menus using CSS.
5) Build a page with a fixed and sticky header.

Java Script Projects

1) Create a simple calculator using JavaScript functions.
2) Build a form validation script using JavaScript.
3) Implement a simple image slider using DOM manipulation.
4) Develop a weather app using an external API and asynchronous JavaScript.
5) Build a todo list application using local storage for data persistence.

Capstone Project: Front End – HTML, CSS, JS

- Design and develop a complete website incorporating HTML, CSS, and JavaScript skills learned throughout the course

React Projects

1. **To-Do List App with React**
   - Build a to-do list application using React, allowing users to add, edit, and delete tasks.
2. **Weather App with React**
   - Develop a weather app using React that fetches data from a weather API and displays it in a user-friendly interface.
3. **Portfolio Website with React**
   - Redesign your personal portfolio website using React to showcase your skills and projects.
4. **Recipe Finder App**
   - Create an app that allows users to search for recipes and view details using React.
5. **E-commerce Product Catalog**
   - Build an e-commerce product catalog with React, showcasing different products and categories.

Node.js Projects

1. **RESTful API for a Blogging Platform:** Create an API that allows users to create, read, update, and delete blog posts.
2. **File Upload Service:** Build an API that allows users to upload files and retrieve them.
3. **Real-time Chat Application:** Develop a real-time chat application using Socket.io and Node.js.
4. **User Authentication and Authorization API:** Create an API for user authentication and authorization using JWT.
5. **E-commerce Backend:** Build the backend API for an e-commerce platform, managing products, orders, and users.

Express.js Projects

1. **Blog Management System:** Create a blog management system where users can create, edit, and delete blog posts.
2. **E-commerce API:** Build the backend API for an e-commerce platform, managing products, orders, and users.
3. **Authentication Middleware**: Develop a custom middleware for user authentication and authorization.
4. **Real-time Chat Application:** Build a real-time chat application using Express.js and Socket.io.
5. **File Upload Service:** Create an API that allows users to upload files and retrieve them.

MongoDB Projects

1. **User Profile Management:** Create a MongoDB schema and CRUD operations for managing user profiles.
2. **Database-driven Blog:** Develop a blogging platform where articles are stored and retrieved from a MongoDB database.
3. **Product Inventory Management:** Build a system to manage product inventory using MongoDB.
4. **Authentication and Authorization:** Implement user authentication and authorization using MongoDB.
5. **Real-time Chat Application:** Create a real-time chat application with MongoDB storing chat history.

Full Stack Projects

1. **E-commerce Platform:** Build a full-stack e-commerce platform using React, Node.js, Express.js, and MongoDB, allowing users to browse products, add them to the cart, and complete the purchase.
2. **Social Media Platform:** Develop a comprehensive social media platform using HTML, CSS, JavaScript, React for the frontend and MongoDB, Node.js, and Express.js for the backend? The platform should enable user to create profiles, post updates, follow other users, and interact with content.

Course Outcomes

- Gain proficiency in creating well-structured HTML for content, styling web pages using CSS, and adding interactivity and functionality with JavaScript.

- Learn to design and develop responsive web applications that adapt seamlessly to various screen sizes and devices.

- Master popular CSS frameworks to streamline the styling process and enhance the design of web applications.

- Understand modern JavaScript concepts and ES6 features, including arrow functions, destructuring, modules, promises, and async/await.

- Acquire a strong foundation in React.js, including components, state, props, lifecycle methods, and working with forms.

- Learn to build dynamic and efficient single-page applications (SPAs) using React.js for a smooth user experience.

- Master backend development using Node.js and Express.js, including setting up servers, defining routes, and handling requests and responses and perform a simple CRUD with MySQL

- Understand the basics of MongoDB, including schema design, CRUD operations, indexing, and integrating MongoDB with Node.js applications.

- Develop RESTful APIs using Node.js and Express.js, ensuring proper request handling, error management, and API documentation.

- Combine HTML, CSS, JavaScript, React, MongoDB, Node.js, and Express.js to build a complete full-stack web application, demonstrating proficiency in both frontend and backend development.