Internet Technologies

Detailed Syllabus:

UNIT- I: Introduction to Web Technologies

Client Server Architecture, Local Host,Introduction To HTML5, W3C, web publishing, Introduction to Style sheet, types of style sheets- Inline, External, Embedded CSS, text formatting properties, CSS Box Model, CSS Border, margin properties, Positioning Use of classes in CSS, color properties, use of &, Layout Design using CSS,Bootstrapping.

UNIT- II: JavaScript

Introduction of JavaScript, JavaScript identifiers, operators, control & Looping structure, Introduction of Array, Array with methods, Math, String, Date Objects with methods User defined & Predefined functions, DOM objects, Window Navigator, History, Location, Event handling, Validations on Forms. Introduction to TypeScript (TS), TS simple,special types, TS arrays,tuples,object types,Enums,Union Types,TS Functions,TS Casting, TS classes, generic and Utility types.

UNIT- III: Client Side Technologies

Introduction to XML and JSON, Introduction to jQuery, Syntax Overview, Anatomy of a jQuery Script, Creating first jQuery script, Traversing the DOM, Selecting Elements with jQuery, Refining & Filtering Selections, Selecting Form Elements, Working with Selections, Chaining, Getters & Setters, CSS, Styling, & Dimensions, Manipulating Elements, Getting and Setting Information about Elements, Moving, Copying, and Removing Elements, Creating New Elements, Manipulating Attributes, Utility Methods, Events, Connecting Event to Elements, Namespacing Events, Event handling, Triggering Event handlers, Event Delegation, Animating effects, animate(), click(), hover(), toggle(), Plugins , Create a basic plugin, Finding & Evaluating Plugins, Writing Plugins, Tabs, Panels and Panes examples, jQuery UI and Forms.

UNIT- IV: AJAX and React JS

AJAX Overview, JQuery's AJAX related methods, Ajax and Forms, Ajax Events. React JS framework, Client side programming, server side programming, Database Programming.

UNIT- V: Server-Side Programming

Introduction to PHP Programming Language, Basic Programming Features: Expressions and Control flow and Arrays, Form Handling, Database Management using PHP and MySQL, Stateful Server Programming with Cookies and Sessions, Apache Server, Node JS programming, Use of POSTMAN, HTTP Methods with Restful Architecture.

Reference Books:

1. Internet and World Wide Web How to Program, P.J. Deitel, H.M. Deitel.
2. Jeffrey C. Jackson, Web Technologies - A Computer Science Perspective, Pearson Education, 2006.
3. Complete reference HTML.
4. JavaScript Bible.
5. HTML, DHTML, JavaScript, Perl & CGI Ivan Bayross
6. XML: How to program Deitel & Deitel.
7. Learning PHP, MySQL & Java Script, Robin Nixon, O'Reilly Publications, 2018

Suggested Exercises as per the Theory syllabus order

Unit 1:

#1.HTML5 & CSS Layout Design for a Responsive Web Page

Objective: Design a responsive web page using HTML5 and CSS, adhering to W3C standards.

Task: Create a multi-section webpage for a fictional company. The page should include a header, navigation bar, main content area, sidebar, and footer. Use HTML5 semantic elements like <header>, <nav>, <section>, <article>, and <footer>.

- Apply Inline, External, and Embedded CSS styles in different parts of your webpage.
- Use Bootstrap to make the layout responsive across different screen sizes.
- Add custom styles using CSS Grid or Flexbox to achieve a complex layout without relying solely on Bootstrap.

#2 Styling and Web Publishing

Objective: Combine knowledge of HTML5, CSS, and web publishing to create and publish a personal blog.

Task: Design a blog page using HTML5, implementing all necessary HTML tags, including and for styling. Use an External CSS stylesheet to manage the styles, including text formatting, color properties, and responsive design.

- Incorporate a custom color scheme and font choices using Google Fonts.
- Publish the blog on a free hosting service like GitHub Pages or Netlify, ensuring it adheres to W3C web standards.
- Extend the project by adding a commenting system using a third-party service like Disqus and ensure your blog is SEO-friendly.

#3 CSS Box Model and Advanced Positioning

Objective: Master the CSS Box Model, positioning techniques, and the use of & elements.

Task:Create a webpage that showcases a product gallery. Each product should be displayed within a card that includes an image, title, and description.

- Use the CSS Box Model to control the padding, borders, and margins of the product cards.
- Implement positioning techniques to create a fixed header and footer, with the gallery in a scrollable central section.
- Experiment with different positioning properties (static, relative, absolute, fixed, and sticky) to place elements in desired locations.
- Challenge yourself by creating an overlay effect on product images using position: absolute; inside the cards.

Unit 2:

#4 Dynamic Form Validation with JavaScript and Event Handling

Objective: Create a dynamic form with client-side validation using JavaScript.

Task: Design a registration form with fields such as name, email, password, date of birth, and phone number.

Implement real-time validation using JavaScript to check:

- Name: Only alphabets, no special characters.
- Email: Valid email format.
- Password: At least 8 characters, including uppercase, lowercase, number, and special character.
- Date of Birth: User must be at least 18 years old.
- Phone Number: Only digits and exactly 10 characters long.
- Utilize DOM manipulation to show error messages next to each field when validation fails.
- Handle form submission only if all fields pass validation.
- Add custom validation rules and create a password strength indicator using the Math object.

#5 Building a To-Do List Application with JavaScript Arrays and DOM Manipulation

Objective: Use JavaScript to create an interactive To-Do list application.

Task: Create a simple UI with an input field to add tasks and a list to display them.

Use JavaScript arrays to store and manage the list of tasks. Implement features to add, remove, and mark tasks as completed. Add a filter to display all tasks, only completed tasks, or only pending tasks.

Use DOM manipulation to update the UI based on the user's actions.

Extend the functionality by adding a "due date" feature, using the Date object, and sorting tasks based on the due date.

#6 Introduction to TypeScript - Refactoring JavaScript Code

Objective: Refactor a JavaScript codebase into TypeScript.

Task: Take an existing JavaScript project (like the To-Do list from Exercise 5) and convert it to TypeScript. Define types for variables, function parameters, and return values.

- Use TypeScript arrays, tuples, and object types to manage data.
- Implement a Task class to encapsulate the properties and methods related to tasks.

- Use Enums for task statuses (e.g., pending, completed).
- Compile the TypeScript code to JavaScript and test its functionality.

#7 Creating a Weather App with TypeScript and DOM Manipulation

Objective: Build a simple weather app using TypeScript that fetches data from a public API.

Task: Use the OpenWeatherMap API to fetch weather data based on the user's location.

Create a user interface with an input field to enter a city name and a button to fetch weather data.

- Use TypeScript for data handling, ensuring all API responses are correctly typed.
- Display the weather data (temperature, humidity, weather condition) dynamically on the page.
- Use DOM objects and Event Handling to update the UI based on user input.
- Add error handling for network issues and invalid city names, and use TypeScript's Union Types and Casting to handle different possible API responses.

Unit 3:

#8 Converting XML Data to JSON and Manipulating with jQuery

Objective: Work with XML and JSON data formats and manipulate them using jQuery.

Task: Create an XML file representing a catalog of books, including details like title, author, genre, price, and publication date.

- Write a jQuery script to load the XML data using $.ajax(), parse it, and convert it to JSON format.
- Display the JSON data in a styled HTML table, using jQuery to dynamically generate the table structure.
- Allow users to filter the books by genre using a dropdown menu. The filter should dynamically update the displayed table.
- Implement additional filtering options such as author and price range, using jQuery to refine the selection.

#9 Building a Dynamic Form with jQuery and JSON

Objective: Create a form that dynamically changes based on JSON data, using jQuery to manipulate the DOM.

Task: Create a JSON object that represents a form structure (e.g., a registration form with fields like name, email, password, and country).

- Write a jQuery script to dynamically build the form based on the JSON data.

- Use jQuery to add interactivity, such as displaying additional fields when the user selects a specific country (e.g., showing a "State" dropdown when "USA" is selected).
- Validate the form fields using jQuery before submission, displaying error messages next to invalid fields.
- Extend the form to include conditional logic, such as showing different sets of fields based on the user's selections, and use jQuery to handle complex validation scenarios.

#10 Creating a jQuery Plugin for Tabbed Navigation

Objective: Develop a custom jQuery plugin that implements tabbed navigation for a web page.

Task: Design a simple webpage with three content sections, each associated with a tab (e.g., "Home", "About", "Contact").

- Write a jQuery plugin that enables tabbed navigation between these sections. The plugin should handle the showing and hiding of content, and update the active tab's styling.
- Include options in the plugin for customizing the active tab's CSS class, animation speed, and default active tab.
- Use the plugin to initialize tabbed navigation on your webpage, ensuring that the content loads smoothly.
- Add features like URL hash navigation (so users can bookmark specific tabs) and keyboard accessibility (e.g., using arrow keys to navigate tabs).

#11 Advanced Event Handling and Animations with jQuery

Objective: Create a feature-rich interactive webpage using advanced jQuery event handling and animations.

Task: Develop a web page with a list of items (e.g., products). Each item should have a "Details" button that reveals more information about the item when clicked.

- Use jQuery to implement the click event, with smooth slide-down animation to reveal the details.
- Implement event delegation so that new items added dynamically to the list also have the "Details" functionality.
- Add hover effects using jQuery to highlight items as the user moves their mouse over them, and toggle the item's background color when clicked.
- Enhance the page by adding a toggle feature that allows users to switch between list and grid views, with jQuery handling the layout changes and animations.

Unit 4

#12 Building a Full-Stack Application with AJAX and jQuery

Objective: Develop a complete web application that integrates client-side and server-side programming with AJAX.

Task: Create a server-side API using Node.js and Express that manages a simple inventory of products. The API should support CRUD operations (Create, Read, Update, Delete) for the products.

- Develop a front-end interface using HTML, CSS, and jQuery. Use jQuery's AJAX methods to asynchronously fetch, add, update, and delete products from the server-side API.
- Implement a form to add new products, and use AJAX to submit the form data to the server without refreshing the page. Display the updated product list dynamically.
- Handle errors and display appropriate messages to the user for failed operations (e.g., network issues, invalid inputs).
- Add user authentication and authorization using JSON Web Tokens (JWT) and ensure that certain operations (like deleting a product) require the user to be logged in.

#13 React JS with AJAX for Real-Time Data Fetching

Objective: Build a React application that fetches and displays real-time data using AJAX.

Task: Create a React application that displays real-time weather information for a user's location.

- Use the fetch API or Axios to retrieve weather data from a public API like OpenWeatherMap. Implement the data fetching in a React component's lifecycle method (e.g., useEffect).
- Display the weather data in a user-friendly format, including temperature, humidity, and weather conditions.
- Implement a search feature that allows users to input a city name and update the displayed weather data based on their search, using AJAX to fetch the new data.
- Add additional features such as a 5-day weather forecast and implement error handling for invalid city names or API request failures.

#14 Dynamic Form Submission with AJAX and jQuery in a React App

Objective: Integrate jQuery's AJAX methods into a React application for dynamic form submission.

Task: Build a React-based contact form that includes fields for name, email, subject, and message.

- Use jQuery's AJAX methods within the React component to submit the form data to a server-side endpoint (e.g., built with Node.js).
- Upon successful form submission, display a confirmation message without reloading the page. If the submission fails, show an error message.
- Implement form validation on both the client-side (using React state) and server-side (using Express middleware) to ensure all fields are correctly filled out.
- Extend the form to include file upload functionality, allowing users to attach files with their message. Use AJAX to handle the file upload and provide feedback on the upload status.

#15 Full-Stack CRUD Application with React, AJAX, and Database Integration

Objective: Create a full-stack CRUD (Create, Read, Update, Delete) application using React, AJAX, and a database.

Task: Set up a server-side API using Node.js, Express, and MongoDB (or another database) to manage a collection of users.

- Build a React front-end that displays a list of users fetched from the database. Use AJAX calls to interact with the server-side API for retrieving, adding, updating, and deleting users.
- Implement a form to add and edit users, with the form data submitted via AJAX to the API. Update the React state to reflect changes in the user list dynamically.
- Ensure the application supports pagination for the user list, and handle sorting and filtering operations on the client side using React.
- Implement optimistic UI updates, where the React UI updates immediately after an operation is initiated, and then rolls back if the operation fails on the server.

Unit5

#16 Developing a User Authentication System with PHP and MySQL

Objective: Build a complete user authentication system using PHP, MySQL, and stateful session management.

Task: Set up a MySQL database with a users table to store user information, including username, email, password (hashed), and registration date.

- Create a PHP script to handle user registration. Validate the form data, hash the passwords, and store the user information in the database.
- Implement a login system using PHP sessions. When a user logs in, validate their credentials, and if correct, start a session to keep them logged in.
- Build a profile page that displays the logged-in user's details and allows them to update their information. Use PHP sessions to ensure that only logged-in users can access the profile page.

- Add a "Remember Me" feature using cookies to keep the user logged in across sessions, and ensure security by preventing session hijacking.

#17 Building a Content Management System (CMS) with PHP and MySQL

Objective: Create a simple CMS using PHP for managing articles in a MySQL database.

Task: Set up a MySQL database with a posts table containing fields for id, title, content, author, and date.

- Develop a PHP backend that allows users to create, edit, and delete articles. Use PHP form handling to process the form data and interact with the MySQL database.
- Implement a frontend interface to display the list of articles, with pagination. Allow users to click on an article title to view its full content on a separate page.
- Add user authentication and authorization, so that only authenticated users can create, edit, or delete articles.
- Implement an image upload feature for articles, storing the images on the server and linking them in the article content.

#18: Creating a RESTful API with Node.js and Testing with POSTMAN

Objective: Build a RESTful API using Node.js and test it with POSTMAN.

Task: Set up a Node.js server using Express to create a RESTful API that manages a list of products.

- Implement CRUD operations: Create, Read, Update, Delete for the products, where each product has a name, description, price, and category.
- Connect the API to a MongoDB or MySQL database to persist the product data.
- Use POSTMAN to test all API endpoints, ensuring that each operation works correctly. Document the API using POSTMAN collections.
- Add user authentication to the API using JWT, and ensure that only authenticated users can create, update, or delete products.

#19 Building a RESTful Web Service with Client-Server Architecture

Objective: Create a simple RESTful API that follows the Client-Server Architecture using Node.js and Express.

Task: Develop a RESTful API for a basic task management system. The API should support the following HTTP methods: GET, POST, PUT, and DELETE.

- Implement endpoints to add, update, delete, and retrieve tasks. Each task should include an ID, title, description, and status (e.g., pending, completed).
- Test your API using Postman or a similar tool to ensure all HTTP methods are correctly implemented.

#20 Stateful Server Programming with PHP Sessions and Cookies, Integrated with Apache Server

Objective: Develop a web application that utilizes stateful server programming using PHP sessions and cookies.

Task: Create a PHP application hosted on an Apache server that tracks a user's browsing history within the application using sessions.

- Implement a shopping cart feature where users can add items to their cart, with the cart state maintained across pages using PHP sessions.
- Use cookies to remember user preferences, such as theme settings (dark mode or light mode), and apply these settings every time the user visits the site.
- Develop an admin panel that allows administrators to view user activity (e.g., products viewed, items added to the cart) by analyzing session data.
- Implement session timeout management, where inactive users are automatically logged out after a certain period of inactivity, and use cookies to store the user's preferred language settings across different sessions.