Assignment-01

* **Aim:-**

- Develop a responsive web design using HTML5, containing a form. Style the pages using CSS, Use of tag selector, class selector and id selectors. Use inline, Internal and External CSS, Apply Bootstrap CSS.

* **Objectives:-**

1. To understand HTML Tags.
2. To learn the styling of web pages using CSS.
3. To learn Bootstrap frontend framework.

* **Theory:-**

1] Define Responsive Web Design (RWD). What is its primary goal?

→ RWD is an approach to web development that ensures websites automatically adjust their layout, images and other elements to provide an optimal viewing experience across different devices and screen sizes. It's primary goal is to make web pages usable and visually appealing on all devices (desktops, tablets and smartphones) without requiring separate versions.

2] Role of <meta name="viewport" ...> in RWD.

→ - The purpose of <meta name="viewport" ...> is to tell the browser how to control the page's dimensions and scaling.
- It is essential because without it, mobile browsers

often zoom out to show the desktop-sized page, making text small, and layouts broken. This tag ensures layouts scale to the device width. eg:-

<meta name="viewport" content="width=device-width, ~~inital~~ initial-scale=1.0">

Q 3] How bootstrap helps create a responsive layout?

→ • Bootstrap provides a 12-column grid system that divides the page into flexible columns.
• Developers can specify how many columns elements take up on different screen sizes, eg col-12 for mobile & col-md-6 for medium devices.
• The grid ~~accepts~~ adapts automatically, stacking on smaller screens and aligning side by side on larger ones.

eg)
```
<div class="row">
    <div class="col-12 col-md-6">Left</div>
    <div class="col-12 col-md-6">Right</div>
</div>
```

Q 4] Difference between Tag, Class, and ID selectors.

→

	Type	Syntax	Usage	Specificity level
①	Tag selector	p {color: red;};	Styles all elements	Lowest
②	Class selector	.highlight { color:blue;}	Styles elements with class = "highlight"	Medium
③	ID selector	#header { color: red;}	Styles the element with id = "header"	Highest

Qs] Three ways to apply Css.

→ 1. Inline Css → directly inside html elements using the style attribute.

eg Hello .

2. Internal Css → inside a <style> tag in html <head>.

eg
```
<style>
    p { color: blue;}
</style>
```

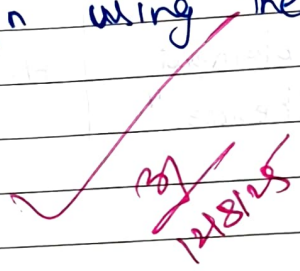
3. External Css → In a separate Css file linked with <link rel = "stylesheet">

eg <link rel = "stylesheet" href = " styles.css" >.

Page No.
Date

* Problem statement:- 4 (Roll 37 - 48)

* Conclusion → Thus, learned about various HTML tags and their scope, syntax and usage, came to know about bootstrap, various methods of CSS styling and hence implemented a responsive web design using the concepts.



Vishwesh Bhat
TY CSF
Roll no - 37
PRN - 1032232281

FSD Lab assignment - 02

* **Aim:** Develop a web application using javascript to implement sessions, cookies, DOM, Perform validations such as checking for ~~emptiness~~ emptiness, only numbers for phone numbers, special characters requirement for password, regex for certain formats of the fields, etc. Use the my SQL database.

* **Objectives:-**

1. To understand what form validation is.
2. To learn basic functioning of DOM projects.
3. To learn how to apply various techniques to implement it.

* **Theory:-**

1) Role of regular expressions (RegEx):-

- Regex are patterns used to match character combination in strings. They provide a powerful way to check whether a string fits a specific format. They are used in various applications such as validating data like phone numbers or passwords, etc.
They are suitable for ~~they~~ these tasks because, Regex can define patterns for valid ph.no. format, including country code, length etc.

eg) `/^\d{3}-\d{3}-\d{4}$/` matches 123-456-7890, a phone number of 10 digits.

- Regex can also enforce specific rules for passwords use case, where we need to define a password of 8 characters minimum, with at least 1 letter, 1 numbers & 1 special character. 3)

2) Sessions vs. cookies:-

- Session:

- A session is a server side storage mechanism used to store user data across multiple pages. A session Id is usually stored in a cookie, but the data is stored on the server.
- They allow web applications to maintain states for users as they navigate between pages.

- Cookie:

- A cookie is a small piece of data stored on the client side (in user's browser). Cookies are typically used to store data such as user preferences or session IDs. 4)
- They hold a session ID to identify the user and maintain their logged in state between page loads.

* Cookies & sessions working together:

- The session is created on the server when the user logs in, and a unique session ID is assigned. The session ID is stored in a cookie on the user's browser. On subsequent requests, the browser sends the cookie to the server, where the session ID is used to retrieve the user's data, maintaining their logged-in state.

3) Client-Side vs. Server-Side ~~Valid~~ Validation:

- Client side validation:
  - happens in the user's browser before data is sent to the server. It can improve user experience by catching errors quickly.
- Server-Side:
  - Happens on the server, where data is validated after it is sent from client. It is crucial because it cannot be bypassed by user.
  - If we rely solely on client side validation, a malicious user could bypass it by disabling js. or using other browsers tools. This could allow them to submit invalid or dangerous data to the server.

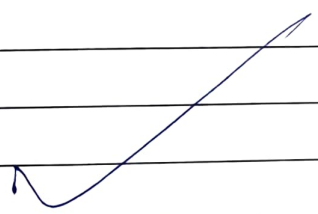
4) DOM Manipulation:

- The DOM or Document object model represents the structure of a webpage. Using javascript, we can interact with DOM to dynamically modify content.

eg) Change content of paragraph after form submission.

```
<!Doctype HTML>
<html>
<head>
    <title> DOM Manipulation </title>
</head>
<body>
    <form id = "form">
```

```
<input type="text" id="uname">
<button type="submit">Submit</button>
</form>

<p id="welcome message"></p>

<script>
document.getElementById('form').onSubmit=
    function(event){

    event.preventDefault();
var uname = document.getElementById('uname

document.getElementById('welcome message').textConten
    ="Welcome, " + uname + "!"; };

</script>

</body>
</html>
```



In the example above, when user submits the form, the script grabs the input value & updates the contents of the <p> element dynamically

5) Steps for Front End to My SQL Connectivity:

1] Setup frontend - HTML, CSS, JS,
- HTML form to collect user input.
- Js for validation.

2] Backend - PHP, nodejs, etc.
- Handles request from frontend and interacts with MySQL.
- Helps with CRUD operations.

3] Setup MySQL Database

4] Use AJAX to send requests using fetch() or XMLHTTPRequest().

5] Write a backend script to handle connection to MySQL database.

* FAQ:-

Q1] Give 3 reasons why form validations are important?

→ 1] Data integrity: Ensures that data entered into the system is accurate and formatted correctly.

2] Improved User Experience: It provides immediate feedback to user, reducing frustration and preventing mistakes.

3] Security: Prevents harmful data (eg SQL injection) from being submitted by sanitizing inputs.

Q2] Give an example of how to modify an attribute value using Dom.

→ In this eg, we will modify the src and alt attributes of <img> tag & change them dynamically using Javascript.

```
<!DOCTYPE HTML>
<html> <head>
    <title> DOM Manipulation </title>
</head>
<body>
  <img src = "image1.jpg" alt = "image1">
  <button onclick = "changeImage()"> Change Image
      </button>

  <script>
  function changeImage(){
    document.getElementById('image').src = 'image2.
    document.getElementById('image').alt = 'image 2'
    }
</script>
</body>
</html>
```

Q3) What are the Different features of Javascript?

1] Event handling ⇒ can respond to events such as click, mouse movements, submissions, etc.
2] DOM Manipulation.
3] AJAX ⇒ asynchronous communication.
4] Functions & loops.
5] Object oriented programming.
6] Error Handling.
7] Asynchronous Programming ⇒ Uses callback, promises, async/await.

FSD Lab Assignment - 3

* **Aim:-** Design an interactive frontend application using React by implementing templating using components, States and Props, Class, Events. It must be responsive to scale across different platforms.

* **Objectives:-**

To develop a responsive, interactive front-end application using React.js that effectively demonstrates the fundamental concepts of component-based architecture, state management, and event handling. The application will serve as a practical exercise in building a scalable user interface by implementing templating with components, managing dynamic data with states and props, and handling user interactions with events, ensuring a seamless user experience across various devices and screen sizes.

* **Theory:-**

1] Role of State and Props in React:

→ • **State:**
— Internal data managed by component.

- A component can update its state using setState (class) or useState.
- Used for dynamic data that changes over time. eg. input fields, counters, etc.

- **Props:**

- Short for properties, passed from parent to child components.
- Read-only; a component can't modify its own props.
- Used to pass data and behavior down the component tree.

★ **Difference:-**

- State is private and Mutable, owned by component.
- On the other hand, Props are external and immutable, provided by parent.

★ **Purpose in Data flow:-**

- React follows one way data flow (top-down). Props follow flow from Parent → Child, while State is local to a component but can be lifted up for shared data.

2] What is a react Component?

→ A react component is a reusable, self contained building block that defines how a UI should look & behave.

- Types:-

I] Class component:

- ES6 classes extending React.component.
- Uses render() method.
- Lifecycle methods (eg, componentDidMount, componentDidUpdate).

eg)
```
class MyComponent extends
    React.component {
      render(){
        return <h1> Hello,
        {this.props.name} </h1>; }}
```

II] Functional Component:

- Simple Js functions returning JSX.
- Initially stateless, but with Hooks (like useState or useEffect), they can manage state & lifecycle.

eg)
```
function MyComponent ({name}) {
    return <h1> Hello, {name} </h1>; }
```

- Advantages of Functional Components with

— Cleaner & more concise syntax.
— No need for "this" keyword
— Hooks allow stateful logic and side eff
in functions.
— Easier to reuse logic via custom hooks.

3] Components of "Templating using component

— In react, UI is built by composing com
instead of writing large static HTML
— Each component acts as a template that
be reused and customised with props.

* Advantages over HTML:-

- Reusability: Components can be used in multipl
places without duplicating code.
- Maintainability: Code is modular, easier to
debug and ~~extend~~ extend.
- Dynamic rendering.
- Separation of concerns.

eg) Instead of copying a button's HTML everyw
we can create a <Button /> componen
& reuse it.

4] Handling user events in React:

→ React uses event handlers similar to DOM events, but with CamelCase syntax and functions.

eg)
```
import React, {useState} from "react";

function counter() {
    Const [count, setCount] = usestate (0);
    const handleClick = () => {
        Setcount (count+1); }

    return (
      <div>
        <p>Count: {count} </p>
      <button onClick = {handleClick} >
    Increase </button>
      </div> ); }

    export default Counter;
```

Here, onClick binds the event handler.
The state (count) is updated when the button is clicked.

5] Responsive Web Design:

→ An approach to design, where websites adapt seamlessly across devices is called RWD.

- Importance:-

— Improves accessibility & usability
— Essential due to diverse screen sizes
— Enhances SEO & user experience.

* Implementation in React:

I] CSS Media Queries:

```
.container {
    width: 100%; }
@media (min-width: 768px){
    .container{
        width: 50% } }
```

II] CSS-IN-JS:

```
import styled from "styled-components";

const Container = styled.div`
    width: 100%;
    @media (min-width: 768px){
        width: 50%; }
```

- Both methods allow us to create layouts that adjust based on screen size, ensuring a fluid user experience.

* Conclusion: Hence, learnt about how React components, Props and events enable building interactive & responsive UIs.

FSD Laboratory - 4

* Aim:

- Enhance web page developed in earlier assignment by rendering lists, portals, Error Handling, Routers and style with React CSS, also make it a responsive design to scale well across PCs, tablets and mobile phones.

* Objectives:

- Enhance User interface and User Experience.
- Improve application robustness and Navigation.

* Theory:-

Q1] How do Lists and Keys work in React?

→ Lists in react are created by iterating over an array of data and rendering elements dynamically using the .map() function. Each item in the list must have a unique key prop.

- Keys: Help React identify which items have been changed, been added or removed. Without proper keys, React may re-render inefficiently, reducing performance.
- Lists: Allow efficient rendering of repeated UI elements like menus, tables or cards.

eg)
```
const items = ['Apple', 'Banana', 'Mango'];
const list = items.map((item, index) => <li key={index}>
    {item}</li>);
```

Page No.
Date

Q2] What is React Portal and when would you use one?

→ A react portal allows rendering of a compon content into different parts of the DOM, outs the parent component's hierarchy.

- Use case: Useful for modals, pop-ups, tooltips or overlays where UI elements should appear above other content but still be controlled by React.

eg) ReactDOM.createPortal (<Modal />, document. — getElementById('modal-root'));

Q3] Discuss the importance of error Boundaries in React.

→ Error Boundaries are React components that catch Javascript errors anywhere in their child component tree and display a fallback UI instead of breaking the entire application.

- Importance: They improve application robustnes by isolating errors, preventing from the app crashing completely.
- Typically implemented using lifecycle methods like componentDidCatch and getDerived — StateFromError in class components.

eg) Used around components that may fail, eg, external API based widgets.

4] How does React Router enable Single Page Application (SPA) functionality?

→ React Router is a library for managing navigation in React applications. In a Single page application (SPA), instead of reloading the whole page, only specific components are rendered based on the route.

- How it works: It listens to URL changes and dynamically loads components without refreshing the browser, giving a seamless User experience.

eg)
```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<Home />} />
  </Routes>
</BrowserRouter>
```

Here we set the homepage path

5] Explain Different ways to style a react application.

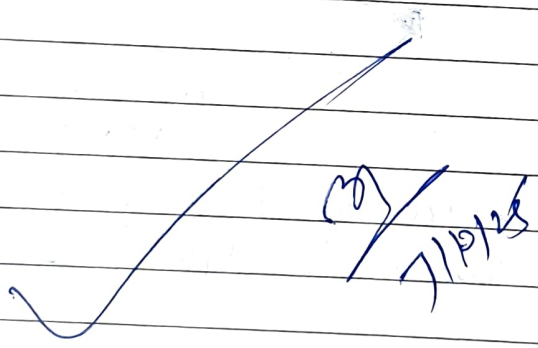
→ There are multiple ways to style a react application:

- CSS stylesheets: Traditional .css files imported into components.
- Inline Styling: Using the style attribute with Javascript objects.
- CSS Modules: Scoped CSS files to avoid naming conflicts.
- CSS-in-JS: Libraries like styled-components or Emoticon allow writing CSS inside Javascript files with dynamic theming.
- Frameworks/Libraries: Using Tailwind CSS, Bootstrap, etc for prebuilt responsive design components.

— Together, these approaches help create visually appealing, responsive & maintainable React applications.

Page No.
Date

* Conclusion:- The experiment ~~ex~~ enhanced the application with lists, portals, error boundaries, routing and styling; improving robustness and navigation.



FSD Laboratory – 5

* **Aim:-** Develop a responsive web design using Express framework to perform CRUD operations and deploy with Node.js, use MongoDB.

* **Objectives:-**

- Develop a full stack web application.
- Demonstrate backend development and deployment ~~profiency~~ proficiency.

* **FAQs:-**

Q1] What is the role of Express.js as a web framework for Node.js?

→ Express.js is a lightweight and flexible web application framework built on top of Node.js. It simplifies the process of building server-side applications by providing features for routing, middleware support, and HTTP request/response handling.

- **Routing** – Organizes application endpoints for different URLs and HTTP methods.
- **Middleware** – Enables adding functionality such as authentication, logging or error handling.
- **Advantage** – Reduces ~~Repeff~~ repetitive code (boilerplate) and speeds up backend development compared to using Node.js alone.

Page No.
Date

Q2] Explain the concept of CRUD operations in the cont of a web application.

→ • Create: Add new records, eg, adding new uses.
• Read: Retrieve and display stored data, eg, viewi users profiles.
• Update: Modify existing records, eg, editing a po
• Delete: Remove records, eg, deleting comments.
— These operations ensure complete interaction w a database and from the backbone of most web applications.

Q3] Why is MongoDB a suitable choice for this pr

→ MongoDB is a ~~No sql~~ No SQL database that stores data in flexible JSON-like documents rather than rigid tables. So it is suitable for the following reasons:
1) Flexibility: Schema-less structure allows easy storage of complex or evolving data.
2) Scalability: Handles large amounts of data efficient through horizontal scaling.
3) Integration: Works seamlessly with Node.js/ Express.js via ~~lab~~ libraries like Mongoose, making it ideal for JavaScript-based full-stack applications.

Q4] What steps are involved in deploying a Node.js and Express application?

→ Deployment involves making the application accessible on the internet. Typical steps include

1) Prepare the application - Finalize code, configure .env variables, ensure app runs locally.
2) Set up a server - Use AWS, Heroku, etc.
3) Install Dependancies.
4) Configure environment
5) Connect to MongoDB or any other DB.
6) Run the server.
7) Testing & scaling - Verify endpoints, enable HTTPS, configure Load Balancing, hire an ethical hacker to pentest it (optional).

* Conclusion :-

This experiment showcased full-stack development using Express, Node.js and MongoDB to perform CRUD operations and deploy a responsive web application.

7/10/01