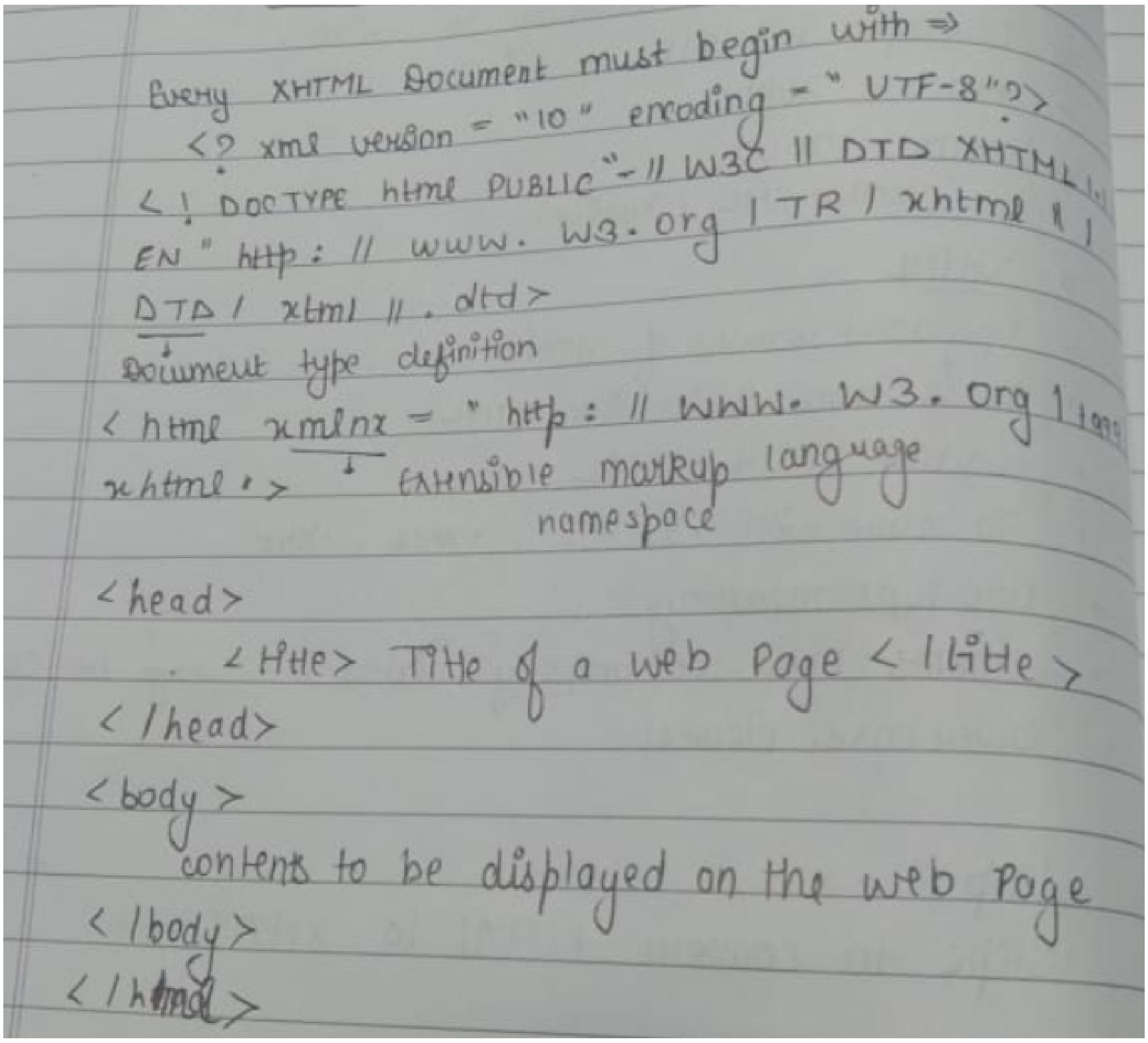
Assignment-1

Subject: Web Development using MERN Stack **Paper Code : FSD-322T**

Q1(a). Explain XHTML's Abstract Syntax. Discuss HTML forms attributes.



Every XHTML Document must begin with =>

```
<? xml version = "10" encoding = " UTF-8"?>
<! DOCTYPE html PUBLIC "-// W3C // DTD XHTML 1.
EN " http : // www. w3.org / TR / xhtml 1 /
DTD / xtml 11 . dtd >
↓
Document type definition
< html xmlnx = " http : // www. w3. org / 1999 /
xhtml " >        ↓  extensible markup language
                      namespace

<head>
    <title> Title of a web Page </title>
</head>
<body>
    contents to be displayed on the web Page
</body>
</html>
```

In XHTML, the Abstract Syntax is typically modeled as a **Document Object Model (DOM)**. The DOM is a tree-like structure where each element, attribute, or piece of text is represented as a node.

In the Abstract Syntax:

- <html> is the root element.
- <head> and <body> are child elements of <html>.
- <title> is further nested under their respective parents.
- Each of these elements and the textual content (like "Welcome") is represented as a node in the DOM.

Below are the key **HTML form attributes**:

- **action**

Specifies the URL where the form data will be sent when the form is submitted.

<form action="submit_form.php" method="POST">

- **method**

Defines the HTTP method (either GET or POST) used to send form data.

<form action="submit_form.php" method="POST">

- **name**

Gives the form a name, allowing it to be referenced in scripts or from other forms.

<form name="contactForm">

(b). Differentiate between DOM and Virtual DOM. How is Redux different from Flux.

DOM vs Virtual DOM

DOM	**Virtual DOM**
Represents the actual structure of the webpage as a tree of nodes.	A lightweight, in-memory copy of the real DOM.
Direct manipulation of the DOM in the browser is slow due to frequent re-rendering.	Updates only the parts of the DOM that need to change, improving efficiency.
Changes trigger reflows and repaints, leading to performance issues with large updates.	Uses a diffing algorithm to update only the necessary parts of the DOM.
Commonly used in traditional web development with vanilla JavaScript.	Used in modern libraries like React for optimized updates.
Re-renders the entire DOM when any change occurs, which can be inefficient.	Batch updates are made to the Virtual DOM first before applying minimal changes to the actual DOM.

Redux vs Flux

Redux and **Flux** are both state management patterns, but they differ in structure:

- **Redux** uses a **single store** for the entire application state, making it simpler and more predictable. State changes are handled by **reducers**, and **actions** are dispatched to update the state. It emphasizes **unidirectional data flow** and supports middleware for handling side effects, making it easier to debug and maintain.
- **Flux** uses **multiple stores**, each managing a specific piece of the state. Actions are dispatched to a **central dispatcher**, which updates the stores. The multiple stores approach can add complexity and make Flux harder to scale, but it offers flexibility in how state is managed across different parts of the application.

Q2(a). How to create objects in Java Script. Explain any two built in objects.

Creating Objects in JavaScript:

1. **Object Literal**:

   const person = { name: "John", age: 30 };

2. **Using** new Object():

   const person = new Object();
   person.name = "John";
   person.age = 30;

Two Built-In Objects:

1. Date **Object**: Used to work with dates and times.

   const currentDate = new Date();

2. Array **Object**: Used to store lists of data.

   const fruits = ["apple", "banana"];

(b). Discuss the different CSS link states? Can elements be overlapped in CSS. Justify this.

CSS Link States:

1. :link: Link not yet visited.

   a:link { color: blue; }

2. :visited: Link already visited.

   a:visited { color: purple; }

3. :hover: Link when hovered.

   a:hover { color: red; }

4. :active: Link when clicked.

   a:active { color: green; }

Overlapping Elements in CSS:

elements can overlap using position:

- position: absolute: Positioned relative to an ancestor.
- position: relative: Positioned relative to its original place.
- z-index: Controls stacking order.

Example:

```
.box1 { position: absolute; top: 50px; left: 50px; z-index: 1; }
.box2 { position: absolute; top: 70px; left: 70px; z-index: 2; }
```

Q3(a). Distinguish between props and state

Props vs State

Props	**State**
Passed from a parent component to a child.	Managed within the component itself.
Immutable, cannot be changed by the child.	Mutable, can be changed using setState().
Used to pass data down the component tree.	Used to track and manage dynamic data within a component.
Can be accessed but not modified by the child.	Can be modified within the component to trigger re-renders.

(b). Distinguish between var, let and const.

var vs let vs const

var	let	const
Function-scoped or globally scoped.	Block-scoped, limited to the block where it's defined.	Block-scoped, like let, but cannot be reassigned.
Can be redeclared and updated in the same scope.	Can be updated, but not redeclared in the same scope.	Cannot be updated or redeclared.
Hoisted to the top of their scope but initialized as undefined.	Hoisted but not initialized, causing a "temporal dead zone".	Hoisted but not initialized, causing a "temporal dead zone".
Generally discouraged in modern JavaScript due to scope issues.	Preferred for variables that may change value.	Preferred for variables that remain constant.

Q4 (a) Elaborate the concept of event in React App by giving an example. What are the functions of event handling?

Event in React:

Events in React represent user interactions (e.g., clicks, input changes). React normalizes these events to work consistently across browsers.

Example:

```
import React, { useState } from 'react';

function App() {
  const [count, setCount] = useState(0);

  const handleClick = () => setCount(count + 1);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={handleClick}>Click Me</button>
    </div>
  );
}

export default App;
```

Explanation: The handleClick function is triggered on button click, updating the state and re-rendering the component.

Functions of Event Handling:

1. **Triggering actions** (e.g., button clicks).
2. **Updating state** to trigger re-renders.
3. **Preventing default behavior** (e.g., form submission).
4. **Binding event handlers** (important in class components).

(b) What are the components? How props are passed between components.

Components are reusable UI building blocks in React, either functional or class-based, that manage their own state and behavior. Props are used to pass data from a **parent** to a **child** component. They are passed as attributes in JSX. Props are **immutable**.

Example:

```
// Parent Component
function Parent() {
  return <Child message="Hello" />;
}

// Child Component
function Child(props) {
  return <h1>{props.message}</h1>;
}
```

Q5 Write short notes on:

(a) Intrinsic event handling

Refers to handling events directly in HTML elements using attributes like onclick, onmouseover. Example:

```
<button onclick="alert('Hello')">Click Me</button>
```

Less flexible for complex apps compared to using JavaScript event listeners.

(b) Document Trees

The Document Tree is a hierarchical structure representing an HTML document as a DOM. The root is document, and elements like <html>, <body>, and <div> form the branches.

```
<html>
  <body>
    <div>Content</div>
  </body>

</html>
```

(c) Call Method( ) and Apply Method( )

call(): Invokes a function with a specified this and individual arguments.

greet.call(this, 'John');

apply(): Similar to call(), but arguments are passed as an array.

greet.apply(this, ['John']);

(d) ECMA Script 6

ES6 is a JavaScript update introducing:

- **let/const**: Block-scoped variable declarations.
- **Arrow Functions**: Concise function syntax.
- **Template Literals**: String interpolation.
- **Classes**: Syntax for OOP.
- **Promises**: For cleaner async handling.