THEORY EXERCISE

Introduction to React.js

Question 1: What is React.js? How is it different from other JavaScript frameworks and libraries?

* **React.js** is **component-based**, meaning UIs are broken down into small, reusable pieces called **components**.
* It uses a **virtual DOM** (Document Object Model) to efficiently update and render only the parts of the page that change.
* React focuses primarily on the **view layer** (the "V" in MVC).
* It uses **JSX**, a syntax extension that lets you write HTML-like code in JavaScript.

| **Feature** | **React.js** | **Angular** | **Vue.js** | **jQuery** |
| --- | --- | --- | --- | --- |
| **Type** | Library (UI focused) | Framework (full-fledged) | Framework (progressive) | Library |
| **Developed By** | Meta (Facebook) | Google | Evan You (community driven) | jQuery Team |
| **DOM Handling** | Virtual DOM | Real DOM with change detection | Virtual DOM | Direct DOM manipulation |
| **Architecture** | Component-based | Component-based (with services, DI) | Component-based | Imperative scripting |
| **Learning Curve** | Moderate | Steep (due to TypeScript & concepts) | Easy to moderate | Easy |
| **Data Binding** | One-way (with props/state) | Two-way (via ngModel) | Two-way (optional) | Manual |
| **Opinionated?** | No (you choose routing, state, etc.) | Yes (comes with everything) | Moderate | No |
| **Use Case** | SPAs, interactive UIs | Enterprise-level apps | Lightweight SPAs or apps | DOM manipulations, legacy apps |

Question 2: Explain the core principles of React such as the virtual DOM and componentbased architecture

**Virtual DOM**:

* React creates a lightweight copy of the real DOM.
* It updates only the parts of the DOM that change, making UI updates fast and efficient.

**Component-Based Architecture**:

* UI is built using reusable, self-contained **components**.
* Components can have their own **state** and receive **props**.

**Declarative UI**:

* You describe what the UI should look like, and React handles how to render it.

**One-Way Data Flow**:

* Data flows from **parent to child** via props, making app behavior more predictable.

**Hooks (in functional components)**:

* Tools like useState and useEffect let you manage state and side effects in functional components.

Question 3: What are the advantages of using React.js in web development?

**Fast Performance**

* Uses **Virtual DOM** to efficiently update UI.

**Reusable Components**

* Build modular, maintainable code with **component-based architecture**.

**Declarative Syntax**

* Easier to understand and debug UI by describing **what** it should look like.

**Unidirectional Data Flow**

* Predictable and controlled data handling.

**Strong Community & Ecosystem**

* Lots of libraries, tools, and community support.

**SEO-Friendly**

* Can be optimized for search engines using tools like **Next.js**.

**Cross-Platform Development**

* Use **React Native** to build mobile apps with the same principles.

2. JSX (JavaScript XML)

Question 1: What is JSX in React.js? Why is it used?

**JSX (JavaScript XML)** is a **syntax extension** for JavaScript used in React. It allows you to **write HTML-like code inside JavaScript**.

* **Makes UI code easier to write and understand**
* **Allows mixing HTML with JavaScript logic**
* **Helps visualize the component structure clearly**
* **Gets compiled to React.createElement() calls**

Question 2: How is JSX different from regular JavaScript? Can you write JavaScript inside JSX?

* **JSX** looks like **HTML**, but it’s actually **syntactic sugar for JavaScript**.
* Browsers **can’t read JSX directly** — it gets compiled into regular JavaScript using tools like **Babel**.

Question 3: Discuss the importance of using curly braces {} in JSX expressions.

* In JSX, **curly braces {} are used to embed JavaScript expressions** inside HTML-like code.
* They let you **insert variables, functions, or expressions** directly into the JSX.

const age = 25;

return <p>You are {age} years old.</p>;

3. Components (Functional & Class Components)

Question 1: What are components in React? Explain the difference between functional components and class components.

* **Components** are the **building blocks** of a React application.
* They are **reusable** pieces of UI that manage their own **state** and **logic**.

| **Feature** | **Functional Component** | **Class Component** |
| --- | --- | --- |
| Syntax | Simple function | ES6 class extending React.Component |
| State Support | Uses **Hooks** (e.g. useState) | Uses this.state and this.setState() |
| Lifecycle Methods | Uses **Hooks** like useEffect() | Has built-in lifecycle methods |
| Code Length | Shorter and cleaner | More verbose |
| Recommended? | ✅ Modern standard | Legacy (still supported) |

Question 2: How do you pass data to a component using props?

* **Props (short for properties)** are used to **pass data from a parent component to a child component**.
* Props are **read-only** and help make components **reusable**.

Question 3: What is the role of render() in class components?

* The \*\*render()\*\* method is **required** in every **React class component**.
* It **returns the JSX** that defines what should be displayed on the screen.

4. Props and State

Question 1: What are props in React.js? How are props different from state?

* **Props** (short for properties) are **inputs** passed from a parent component to a child component.
* They are **read-only** and used to customize or configure components.

**🔸 Difference Between Props and State**

| **Aspect** | **Props** | **State** |
| --- | --- | --- |
| Who Controls? | Controlled by **parent** component | Controlled **within the component** itself |
| Mutability | **Immutable** (read-only) | **Mutable** (can change over time) |
| Purpose | Pass data and config to components | Manage dynamic data and UI changes |
| Usage | Used to pass info from parent to child | Used to track internal component data |

Question 2: Explain the concept of state in React and how it is used to manage component data

* **State** is a built-in object used to **store and manage data** that can change over time within a component.
* When state changes, React **re-renders** the component to update the UI.

**🧾 How it’s used:**

* In **class components**, state is managed with this.state and updated with this.setState().
* In **functional components**, state is managed using the useState hook.

Question 3: Why is this.setState() used in class components, and how does it work?

* this.setState() is used to **update the component’s state**.
* It **merges the new state with the old state** and **triggers a re-render** to update the UI.

5. Handling Events in React

Question 1: How are events handled in React compared to vanilla JavaScript? Explain the concept of synthetic events.

* In React, events are handled using **camelCase** syntax (e.g., onClick) instead of lowercase (onclick).
* You pass **functions** as event handlers instead of strings.

**🔹 Synthetic Events**

* React uses **Synthetic Events**, which are **cross-browser wrappers** around native events.
* They provide a **consistent API** across different browsers.
* Synthetic events improve performance through **event delegation**.

Question 2: What are some common event handlers in React.js? Provide examples of onClick, onChange, and onSubmit.

* **onClick**: Handles mouse click events.
* **onChange**: Handles changes in input fields.
* **onSubmit**: Handles form submission.

<button onClick={handleClick}>Click me</button>

<input type="text" onChange={handleChange} />

<form onSubmit={handleSubmit}>

<button type="submit">Submit</button>

</form>

Question 3: Why do you need to bind event handlers in class components?

* In class components, **this is not bound by default** in event handlers.
* Binding ensures this correctly refers to the component instance inside the handler.
* Without binding, this would be undefined or incorrect.