Front-End Development Notes

Comprehensive Theory Notes on Front-End Development Topics

1. What is a Front-End Framework?

A front-end framework is a collection of pre-written code used to build user interfaces efficiently. It includes components, utilities, and standards for structured application development.

2. Difference Between Framework and Library:

A framework dictates the application structure and calls your code, whereas a library offers specific tools that your code calls when needed.

3. Benefits of Using a Framework:

- Accelerates development with reusable components.
- Enforces structured, maintainable code.
- Has large community support and resources.
- Handles browser compatibility issues.
- Promotes modular and testable code.

4. React, Angular, and Vue:

- React: A UI library by Facebook using JSX and Virtual DOM. Suitable for SPAs.
- Angular: A full MVC framework by Google with TypeScript and two-way binding. Ideal for enterprise applications.
- Vue: A flexible framework for the view layer, simpler than Angular, with optional JSX.

5. JavaScript Recap:

- Functions: Reusable blocks of code.
- Arrays: Indexed collections of values.
- Objects: Key-value pairs representing real-world entities.

6. Promises & async/await:

Promises represent future values. Async/await allows asynchronous code to look synchronous. Errors in async/await are handled using try-catch blocks.

7. Fetch vs XMLHttpRequest:

Fetch is modern and promise-based, cleaner than the older XMLHttpRequest which uses callbacks and verbose syntax.

8. Axios:

Axios is a promise-based HTTP client. Installation via npm, it simplifies API calls and automatically handles JSON data.

9. Intro to React:

React uses a Virtual DOM and JSX syntax. Webpack bundles JavaScript and dependencies. JSX allows mixing HTML with JavaScript logic.

10. Component Driven Approach:

- Class Components: Stateful with lifecycle methods.
- Function Components: Stateless, use hooks for state.

11. Class Component Lifecycle:

Includes constructor(), componentDidMount(), componentDidUpdate(), and componentWillUnmount() for different phases of a component's life.

12. Props:

Props are used to pass data between components. Default props can be set. Props are read-only.

13. State Management:

Class components use this.setState. Function components use useState hook to manage and update state.

14. Forms and Event Handling:

Event handlers like onChange or onSubmit are used in JSX. Form state is updated based on user input.

15. useEffect Hook:

Used for side effects like API calls in function components. Runs after the component renders.

Front-End Development Notes

16. TypeScript Introduction:

A superset of JavaScript with static typing, catching errors at compile-time. Offers better tooling and code readability.

17. TypeScript Setup:

Initialize using `tsc --init`. Use `.ts` files. Compile using `tsc` to generate JavaScript.

18. Basic Types:

Includes annotations like number, string, boolean, object, array etc.

19. Functions in TypeScript:

Support for typed parameters, optional/default parameters, rest parameters, and function overloading.

20. Enums, Generics, Tuples:

- Enums define a set of named constants.
- Generics allow functions/classes to work with different types.
- Tuples define fixed-length arrays with known types.

21. Classes and Interfaces:

Classes can implement interfaces. Interfaces define contracts that classes must follow.

22. DOM Manipulation in TypeScript:

DOM elements are selected with strict types for better safety and IntelliSense.

23. DOM Events in TypeScript:

Event listeners use specific event types like MouseEvent, reducing runtime errors.

24. Error Handling in DOM:

Use null checks and optional chaining to safely access and modify DOM elements.