|  |  |
| --- | --- |
| **Angular** | **React** |
| * HTML centred JavaScript framework * Renders the application using the real DOM * MVVM organised structure * A subset of html * Angular is platform-independent and hence is compatible to work in any platform. Helps building SPA faster with less code * Can be deployed across multiple targets such as web, mobile web, native mobile etc. * Offers both one-way and two-way data binding * Customised HTML elements * Fully organised Architecture * Dependency Injection * Reusability of components * Extended HTML components * Testable, scalable | * JavaScript library * Faster rendering ability because of the virtual DOM concept * MVC structure * Uses JSX for rendering .Hence pure JavaScript based. * Can be rendered on mobile servers using react native * Reusable UI components which provides data which changes over time * One-way data binding * It needs Redux for proper state management * Reconciliation * High security * Components are maintained separately and rendered independently. * Reusability of components * Merged easily with third-party components. * Testable, Scalable |
| * Core Design Patterns:  1. Observer Pattern- a software design pattern in which an object, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods.   Example: Angular $emit and $broadcast.   1. Chain of Responsibilities Pattern: Chain of Responsibilities Pattern consists of a source of command objects and a series of processing objects. Each processing object contains logic that defines the types of command objects that it can handle; the rest are passed to the next processing object in the chain.   Example: Angular Filters   1. Composite Pattern: The composite pattern describes that a group of objects are to be treated in the same way as a single instance of an object. 2. Modular and Singleton | * Core Design Patterns:  1. Command Pattern-command Pattern is very useful to design decoupled objects. The motivation of this pattern is to execute some piece of business logic at some point in the future.   Example: Redux store is receiver (consists of all reducers)   1. Container Pattern:   In react redux, the container component alone will be passing data to the presentational components and also perform action functions which send and receive data from a server  Example: mapDispatchToProps(), mapStateToProps() method of container   1. Modular and Singleton |
| When to choose Angular?Angular can be chose when the web apps need to be developed faster with separation of concerns with definite architecture. | When to choose to react?  * To develop reusable components that changes over time. * If Performance and Security are major concerns |