**Redux**

1. Intro
   1. Redux is a predictable state container for java scripts apps.
   2. Redux is not tied to React
   3. Can be used with Angular, React, Vue or even vanilla java script
   4. Redux is a library for java scripts applications
   5. Redux stores the state of application
   6. In redux, all state transitions are explicit, and it is possible to keep track of them
   7. The changes to your application’s state become predictable
   8. React-Redux is a library that provides bindings to use React and Redux together in an application
2. Three core concepts
   1. Store, Action and Reducer are three core concepts
   2. A **Store** that holds the state of your application
   3. An **Action** that describes the changes in the state of the application
   4. A **Reducer** which actually carries out the state transition depending on the action
3. Three Principles
   1. First: The state of your whole application is stored in an object tree within a single store
   2. Second: The only way to change the state is to dispatch an action, an object describing what happened.
   3. Third: To specify how the state tree is transformed by actions, you write pure reducers.
4. Actions:
   1. The only way your application can interact with the store
   2. Carry some information from your app to the redux store
   3. Plain java script objects
   4. Have a ‘Type’ property that indicates the type of action being performed
   5. The ‘Type’ property is typically defined as string constants
5. Reducer
   1. Specify how the app’s state changes in response to actions sent to the store
   2. Function that accepts state and action as arguments, and returns the next state of the application
   3. (previousState, action) => newState
6. Store
   1. One store for the entire application
   2. Responsibilities
      1. Holds application state
      2. Allows access to state via **getState()**
      3. Allows state to be updated via **dispatch(action)**
      4. Registers listeners via **subscribe(listener)**
      5. Handles unregistering of listeners via the function returned by **subscribe(listener)**
7. Combine Reducer
   1. Used to combine multiple reducer into single
8. Middleware
   1. Is the suggested way to extend redux with custom functionality
   2. Provides a third-party extension point between dispatching an action, and the moment it reached the reducer
   3. Use middleware for logging, crash reporting, performing asynchronous tasks etc.
9. Action Types:
   1. Synchronous Action
      1. As soon as an action was dispatched, the state was immediately updated
   2. Asynchronous Action
      1. Asynchronous API calls to fetch data from an end point and use that data in your application
      2. Redux-thunk is define to create async action which is a middleware we can apply in redux store
10. React Redux

**Flux**

1. Flux is an architectural pattern proposed by Facebook for building SPAs. It suggests to split the application into the following parts:
   1. Stores
   2. Dispatcher
   3. Views
   4. Action / Action Creators
2. Actions
   1. Helper methods that facilitate passing data to the dispatcher
3. Dispatcher
   1. Receives actions and broadcasts payloads to registered callbacks
4. Store
   1. Containers for application state and logic that have callbacks registered to the dispatcher
5. Controller View
   1. React Components that grab the state from stores and pass it down via props to child component