Welcome

React Js Essentials By Debangshu Nag

Course content

- ❖Overview
- ❖Setup of Environment & Editor
- ❖JSX
- ❖Components
- ❖Props Overview
- ❖Props Validation
- ❖Component API
- ❖Component Life Cycle
- ❖Forms

Course content (Cont..)

- Events
- Refs
- Arrays
- Keys
- Router
- Flux concept
- Using Flux / Redux
- Best Practices
- Next Step..

Overview

- ❖What is React Js?
  - ➢ReactJS is JavaScript library used for building reusable UI components developed by Facebook.
- ❖Features of React JS?
  - ➢JSX : JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.
  - ➢Components: React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects.

Overview (Cont..)

- Unidirectional data flow and Flux: React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your data unidirectional.
- License: React is licensed under the Facebook Inc. Documentation is licensed under CC BY 4.0.

❖ React Advantages

- Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.
- Can be used on client and server side as well as with other frameworks.
- Component and data patterns improve readability, which helps to maintain larger apps.

Overview (Cont..)

- React Limitations
  - Covers only the view layer of the app, hence you still need to choose other technologies to get a complete tooling set for development.
  - Uses inline templating and JSX, which might seem awkward to some developers.
- What is virtual DOM?
  - Virtual DOM is in memory representation of real DOM. The DOM is synced with real DOM. The synchronization happens between the executions of render function & display elements in the browser, this process are called reconciliation.
- How Virtual DOM Works?
  - If there are any changes occurs, the changes applied to virtual DOM by React JS. React then calculate the difference between the actual DOM & virtual DOM.Once the difference has been calculated then the changed portion of the real DOM will be updated only.

Overview (Cont..)

- Single Page Application (SPA)
  - It is a web application or web site that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server. This approach avoids interruption of the user experience between successive pages, making the application behave more like a desktop application.
  - In an SPA, either all necessary code – HTML, JavaScript, and CSS – is retrieved with a single page load or the appropriate resources are dynamically loaded and added to the page as necessary, usually in response to user actions.
  - The page does not reload at any point in the process, nor does control transfer to another page.

Overview (Cont..)

- ❖ Challenges in SPA
  - ➢ DOM Manipulation: How to manipulate the view efficiently?
  - ➢ History: What happens when pressing back button?
  - ➢ Routing: Readable URLs i.e. how to generate different urls for different views?
  - ➢ Data Binding: How bind data from model to view?
  - ➢ View Loading: How to load the view?
  - ➢ SEO: How to do SEO (Search Engine Optimization) properly.
- ❖ Advantages of SPA
  - ➢ Minimum page load time.
  - ➢ Code reusability & maintainability.
  - ➢ High response time increase good user experience.

Environment Setup

- We must have node js software installed in our machine
- Editors
    - Visual Studio Code
    - Sub lime Text
    - Note pad++
- Install React Js
    - npx create-react-app applicationName
    - We need to go into the folder & then run the application
    - To run the application: npm start

JSX

- Why use JSX?
  - It is faster because it performs optimization while compiling code to JavaScript.
  - It is also type-safe and most of the errors can be caught during compilation.
  - It makes easier and faster to write templates, if you are familiar with HTML.
- Lets check prog1 folder & put those in src folder
- We Can use javascript expression in JSX.
  - Lets check prog2 folder & put those in src folder
- We cannot use if else statements inside JSX, instead we can use conditional (ternary) expressions.
  - Lets check prog3 folder & put those in src folder

JSX (Cont..)

- ❖Inline Styling: React recommends using inline styles. When we want to set inline styles, we need to use camelCase syntax. React will also automatically append px after the number value on specific elements.
  - ➢Lets check prog4 folder & put those in src folder
- ❖Comments When writing comments, we need to put curly brackets {} when we want to write comment within children section of a tag.
  - ➢Lets check prog5 folder & put those in src folder
- ❖HTML tags always use lowercase tag names, while React components start with Uppercase.

Components

- Stateless Example: We are creating two different components "Header" & "Content" App component will consume it.
  - Lets check prog5 folder & put those in src folder
- Stateful Example: It is same as previous example but in this case we are using state object
  - Lets check prog6 folder & put those in src folder

Props

- ❖ The main difference between state and props is that props are immutable but state are mutable for this reason container component i.e. parent component should define the state that can be updated and changed, while the child components should getdata from the from parent component as props.
  - ➢ Lets check prog7 folder & put those in src folder
- ❖ We can also set default property values directly on the component constructor.
  - ➢ Lets check prog8 folder & put those in src folder
- ❖ We can combine state & props
  - ➢ Lets check prog9 folder & put those in src folder

Props Validation

- ❖ Properties validation is a useful way to force the correct usage of the components. This will help during development to avoid future bugs and problems, once the app becomes larger. It also makes the code more readable, since we can see how each component should be used. We have to import “prop-types” module
  - ➢ Lets check prog10 folder & put those in src folder

Component API

- Set State: setState() method is used to update the state of the component. This method will not replace the state, but only add changes to the original state.
  - Lets check prog11 folder & put those in src folder
- Force Update: Sometimes we might want to update the component manually. This can be achieved using the forceUpdate() method.
  - Lets check prog12 folder & put those in src folder
- Find Dom: For DOM manipulation, we can use ReactDOM.findDOMNode() method. First we need to import react-dom module.
  - Lets check prog13 folder & put those in src folder

Component Life Cycle

- ❖ constructor: This gets invoked once and at first.
- ❖ getDerivedStateFromProps(): This method invoked just before calling render() & is invoked on every render. This method is used in rare use case i.e. when props changes over time like in case of animation, it can be used to check which part of the component needs to be updated.
- ❖ componentDidMount(): This method invoked after first rendering. This method is suitable place for Ajax requests, state or DOM update & setup event listener.
- ❖ shouldComponentUpdate(): It determines if component needs to be updated or not. By default it returns true. If we are sure that due to change of props or state the component no need to be updated then we can return false.

Component Life Cycle (Cont..)

- getSnapshotBeforeUpdate(): This method executed just before the render output committed to the DOM. The value of this method passed to componentDidUpdate(). This method is useful to get DOM information like scroll position.
- componentDidUpdate(): This is used to update DOM when state or props changes. This will not fire if shouldComponentUpdate() return false.
- componentWillUnmount(): It is used to cancel network request & remove event listeners associated with the component.
- Lets check prog14 folder & put those in src folder

Forms

- We should update state using setState() method & should not update state by assigning changed value directly.
- we will set an input form with **value = {this.state.data}**. This allows to update the state whenever the input value changes. We are using **onChange** event that will watch the input changes and update the state accordingly.
  - Lets check prog15 folder & put those in src folder
- Here onChange method will trigger state update that will be passed to the child input value and rendered on the screen.
  - Lets check prog16 folder & put those in src folder

Events

- ❖ We are just adding **onClick** event that will trigger **updateState** function once the button is clicked.
  - ➢ Lets check prog17 folder & put those in src folder.
- ❖ When we need to update the state of the parent component from its child, we can create an event handler (updateState) in the parent component and pass it as a prop (updateStateProp) to the child component where we can just call it.
  - ➢ Lets check prog18 folder & put those in src folder.

Refs

- The ref is used to return a reference to the element. Refs should be avoided in most cases, however, they can be useful when we need DOM measurements or to add methods to the components.
- This example will show how to use refs to clear the input field. ClearInput function searches for element with ref = "myInput" value, resets the state, and adds focus to it after the button is clicked.
  - Lets check prog19 folder & put those in src folder.

Keys

- React keys are useful when working with dynamically created components or when your lists are altered by the users. Setting the key value will keep your components uniquely identified after the change.
- If we add or remove some elements in the future or change the order of the dynamically created elements, React will use the **key** values to keep track of each element
    - Lets check prog20 folder & put those in src folder.

Router

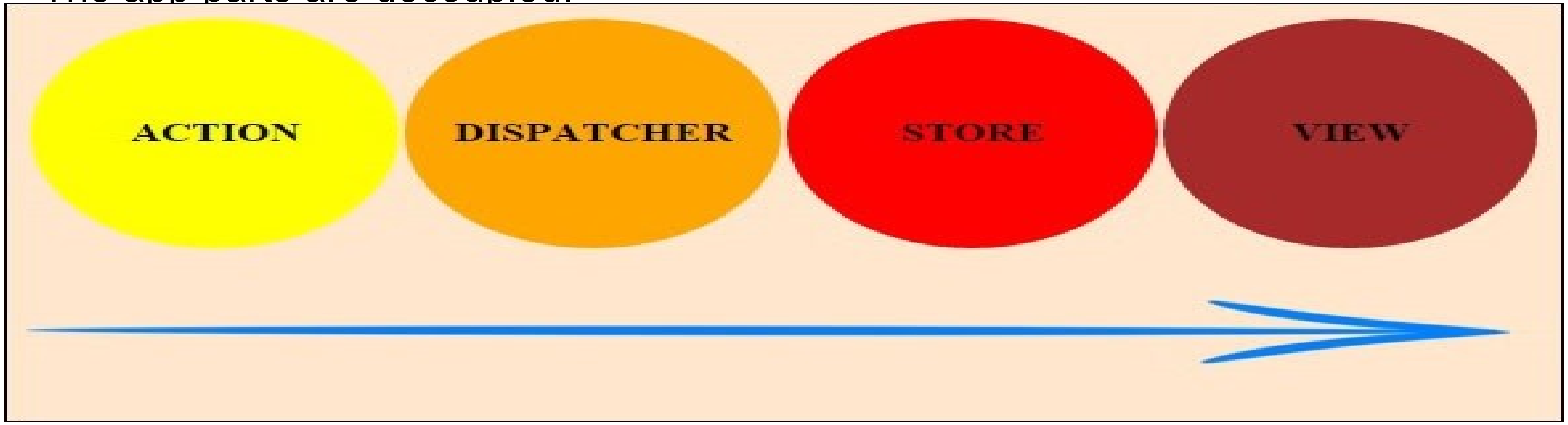
- ❖ A simple way to install the react-router-dom is to run the following code snippet in the command prompt window.
- ❖ we will create four components. The App component will be used as a tab menu. The other three components (Home), (About) and (Contact) are rendered once the route has changed.
  - ➢ Lets check prog21 folder & put those in src folder.

Flux Concept

- Flux is a programming concept, where the data is uni-directional. This data enters the app and flows through it in one direction until it is rendered on the screen.
- Flux Elements
  - Actions − Actions are sent to the dispatcher to trigger the data flow.
  - Dispatcher − This is a central hub of the app. All the data is dispatched and sent to the stores.
  - Store − Store is the place where the application state and logic are held. Every store is maintaining a particular state and it will update when needed.
  - View − The view will receive data from the store and re-render the app.

Flux Concept (Cont..)

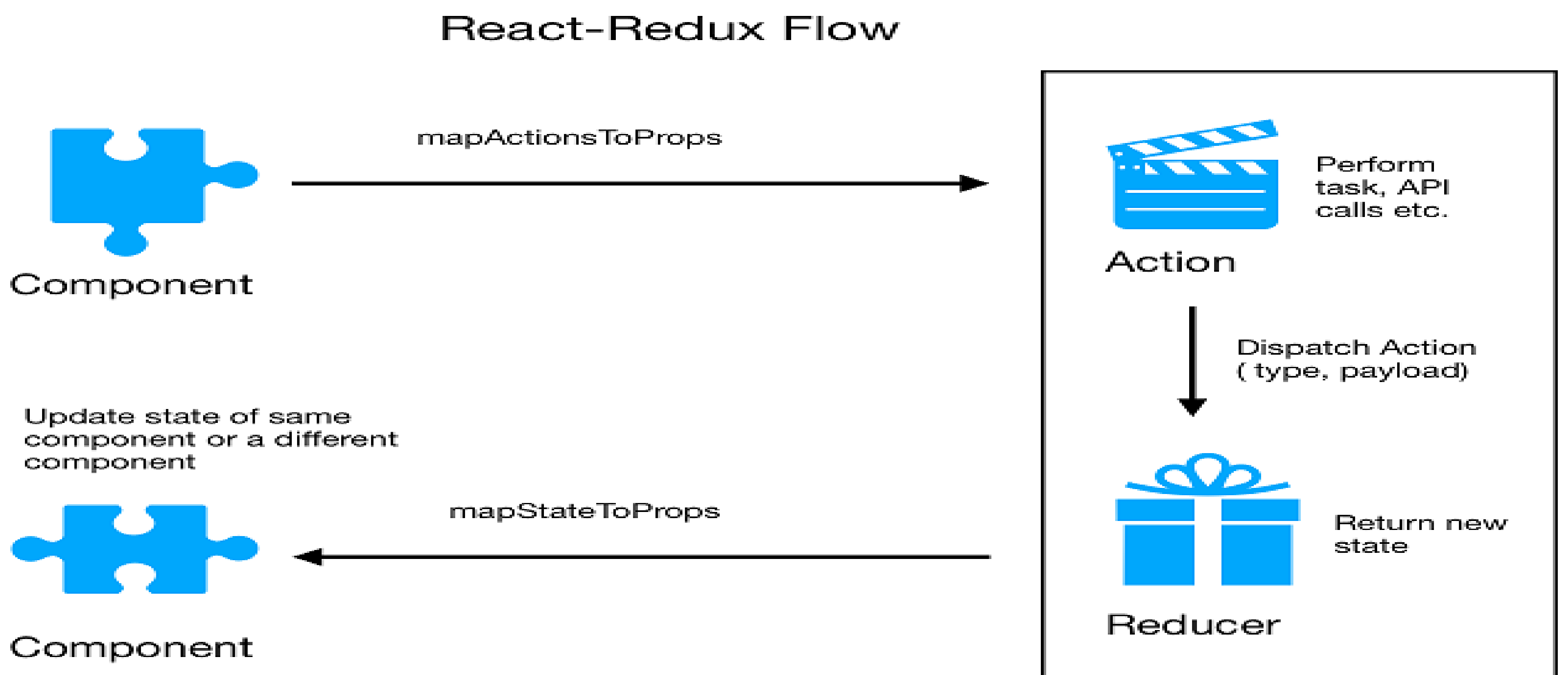
- Flux Pros
  - Single directional data flow is easy to understand.
  - The app is easier to maintain.
  - The app parts are decoupled.



Flux / Redux

- To implement flux pattern in React applications. We will use **Redux** framework.
- We have to install two modules one is redux & another is react-redux: npm i -s redux react-redux
- Lets check prog22 folder & replace the src folder.
- Need to create two folders actions, reducers
- Need to create following files
    - Actions/printMessageAction.js & action-type.js
    - reducers/printMessageReducer.js & index.js
    - store.js

Flux / Redux (Cont..)


React-Redux Flow
mapActionsToProps
Component
Perform task, API calls etc.
Action
Dispatch Action (type, payload)
Update state of same component or a different component
mapStateToProps
Component
Return new state
Reducer

Flux / Redux (Cont..)

- In action-type.js file we have created constants which are nothing but key to identify an action.
- We have created printMessageAction.js file which contains two pure functions.
- index.js of reducer file contains combine reducer which actually can combine multiple reducers.
- printMessageReducer.js contains switch case & on the basis of different actions here state data is getting updated.
- store.js create the store & return to App component.
- App component actually connecting to redux store using connect method.Here mapStateToProps & mapDispatchToProps can be any name.

Flux / Redux (Cont..)

- There are certain things that you should keep in mind while working in react-redux
    - Reducers must be pure functions. Given any input, output must always be the same.
    - The state of your whole application is stored in an object tree within a single store.
    - State is read-only. The only way to change the state is to emit an action, an object describing what happened.
    - Changes in state are made with pure functions (reducers).

Best Practices

- ❖State − It is a good practice to centralize the state and pass it down the component tree as props. Whenever we have a group of components that need the same data, we should set a container element around them that will hold the state. Flux pattern is a nice way of handling the state in React apps.
- ❖PropTypes − The PropTypes should always be defined. This will help is track all props in the app and it will also be useful for any developer working on the same project.
- ❖Render − Most of the app's logic should be moved inside the render method. We should try to minimize logic in component lifecycle methods and move that logic in the render method. If we need to calculate something from the state or props, we can do it inside the render method.
- ❖Composition − One component should only be responsible for one functionality. If some of the components have more than one functionality, we should refactor and create a new component for every functionality.

Next step

React - The Complete Guide by Maximilian Schwarzmüller from Udemy

Thank You!