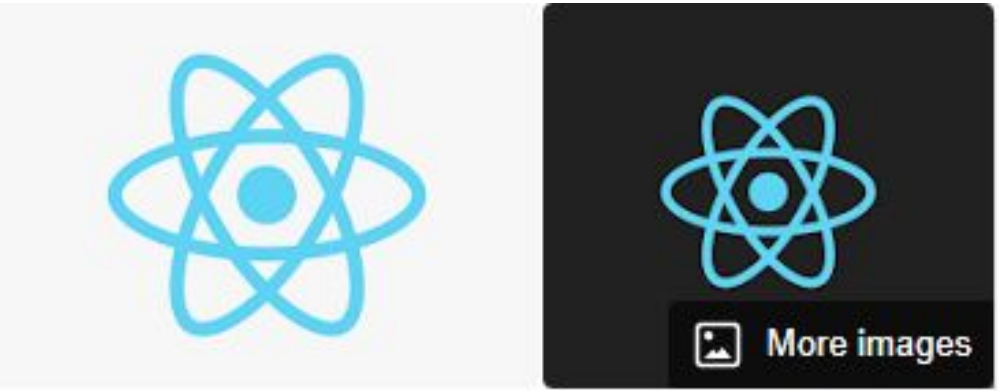


React

JavaScript library

Stable release: 17.0.2

Initial release: 2013

Platform: Web platform

Written in: JavaScript

License: MIT License

Developer: Open-source software, Facebook, Instagram

Sandeep Gupta

What is React ?

React is an open-source, front end, JavaScript library for building user interfaces or UI components.

It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications

Why should we use React

1. Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.

2. Can be used on client and server side as well as with other frameworks.

3. Component and data patterns improve readability, which helps to maintain larger apps.

React Features

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.

Unidirectional data flow − React implements one-way data flow which makes it easy to reason about your app.

License − React is licensed under the Facebook Inc. Documentation is licensed under CC BY 4.0.

Environment Setup and Configuration

Download and install - npm

Download and install - nodejs

Download and install - git

Download and install any editor - like VS Code

Starting with create-react-app

Create React App is a comfortable environment for learning React, and is the best way to start building a new single-page application in React.

It sets up your development environment so that you can use the latest JavaScript features, provides a nice developer experience, and optimizes your app for production.

npm i create-react-app

npm create-react-app my-app

cd my-app

npm start

JSX

Using JSX

JSX looks like a regular HTML in most cases. We already used it in the Environment Setup chapter. Look at the code from App.jsx where we are returning div.

React Components

1. Stateless Component
2. Stateful Component

State is the place where the data comes from. We should always try to make our state as simple as possible and minimize the number of stateful components. If we have, for example, ten components that need data from the state, we should create one container component that will keep the state for all of them.

Using State

The following sample code shows how to create a stateful component using EcmaScript 2016 syntax.

React Props

The main difference between state and props is that props are immutable. This is why the container component should define the state that can be updated and changed, while the child components should only pass data from the state using props.

Using Props

When we need immutable data in our component, we can just add props to reactDOM.render() function in main.js and use it inside our component.

Lifecycle Methods

Each component has several “lifecycle methods” that you can override to run code at particular times in the process.

Mounting -

These methods are called in the following order when an instance of a component is being created and inserted into the DOM:

constructor()

getDerivedStateFromProps()

render()

componentDidMount()

Lifecycle Methods

Updating -

An update can be caused by changes to props or state. These methods are called in the following order when a component is being re-rendered:

getDerivedStateFromProps()

shouldComponentUpdate()

render()

getSnapshotBeforeUpdate()

componentDidUpdate()

Lifecycle Methods

Unmounting

This method is called when a component is being removed from the DOM:

componentWillUnmount()

React Forms

In the following example, 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.

React Events

This is a simple example where we will only use one component. We are just adding onClick event that will trigger updateState function once the button is clicked.

onchange() - An HTML element has been changed

onclick() - The user clicks an HTML element

onmouseover() - The user moves the mouse over an HTML element

onmouseout() - The user moves the mouse away from an HTML element

onkeydown() - The user pushes a keyboard key

onload() - The browser has finished loading the page

React 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.

Using Refs

The following example shows 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.

React Router

A simple way to install the react-router is to run the following code snippet in the command prompt window.

npm install react-router

Step 2 - Create Components

In this step, 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.

Example : https://codesandbox.io/s/react-redux-application-hewdb

React New Features (Introduced in React 16.x)

There are new features introduced in react 16. Below are the new React 16 features we are going to cover.

1. React Hooks
2. React HOC
3. React Fragments

React Js Best Practices

1. Keep components small and function-specific

2. Reusability is important, so keep creation of new components to the minimum required

3. Comment only where necessary

4. Name the component Always in Camel Case after the function

React Js Best Practices

5. Name variables always in lowercase.

6. Write unit test case for each of the module.

7. keep your component small (less number of codes) and structured.

8. Keep code writing formatted well aligned by using eslint.

React References

React.org

https://reactjs.org/docs/getting-started.html

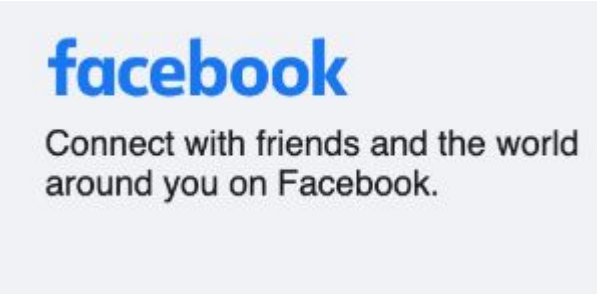
Medium

https://medium.com/easyread/how-to-get-started-with-react-js-805bf57826ad

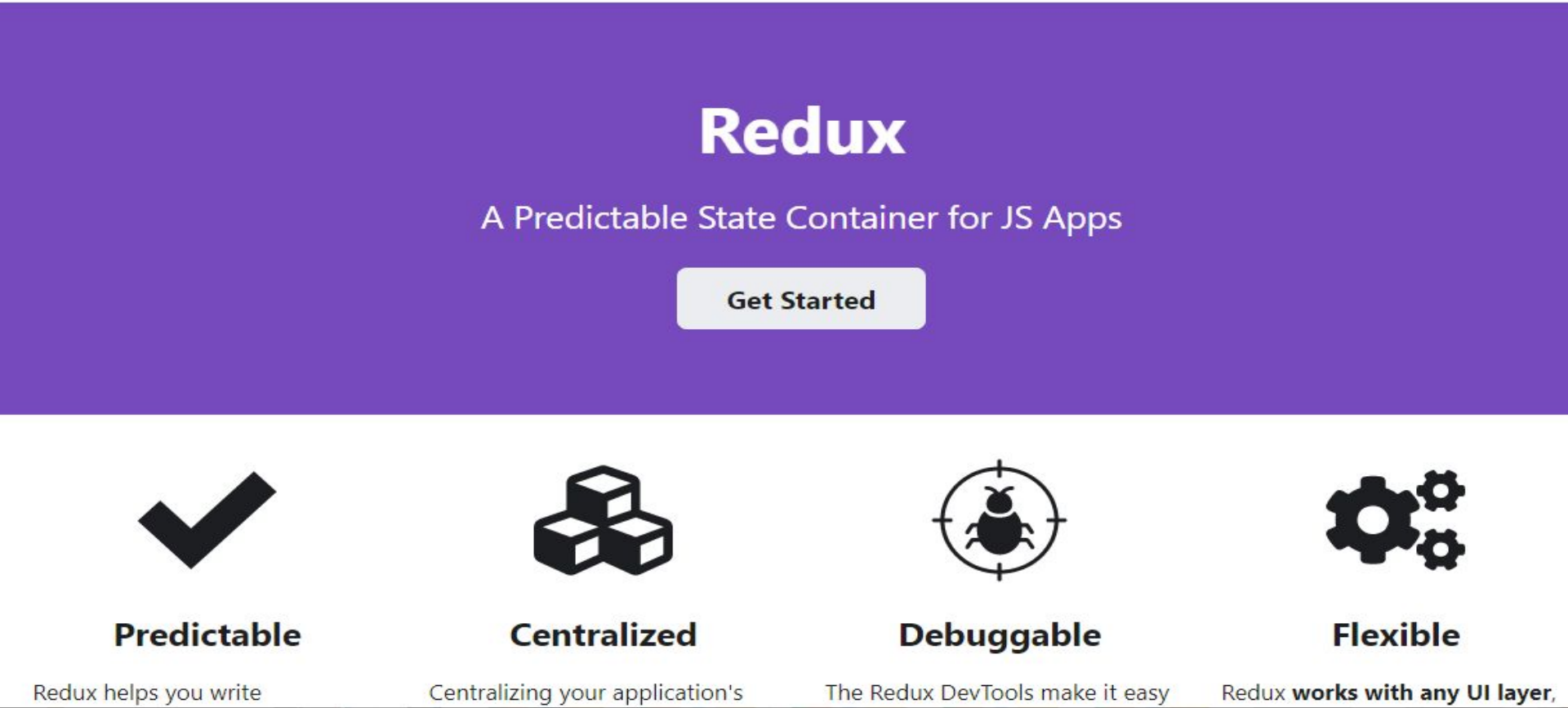
TutorialPoint

https://www.tutorialspoint.com/reactjs/index.htm

Some Famous Web App Based On React



REDUX (https://redux.js.org/)



REDUX

Redux is a predictable state container for JavaScript apps. As the application grows, it becomes difficult to keep it organized and maintain data flow. Redux solves this problem by managing application's state with a single global object called Store. Redux fundamental principles help in maintaining consistency throughout your application, which makes debugging and testing easier.

More importantly, it gives you live code editing combined with a time-travelling debugger. It is flexible to go with any view layer such as React, Angular, Vue, etc.

Principles of Redux

Predictability of Redux is determined by three most important principles as given below −

Single Source of Truth

REDUX

The state of your whole application is stored in an object tree within a single store. As whole application state is stored in a single tree, it makes debugging easy, and development faster.

State is Read-only

The only way to change the state is to emit an action, an object describing what happened. This means nobody can directly change the state of your application.

Changes are made with pure functions

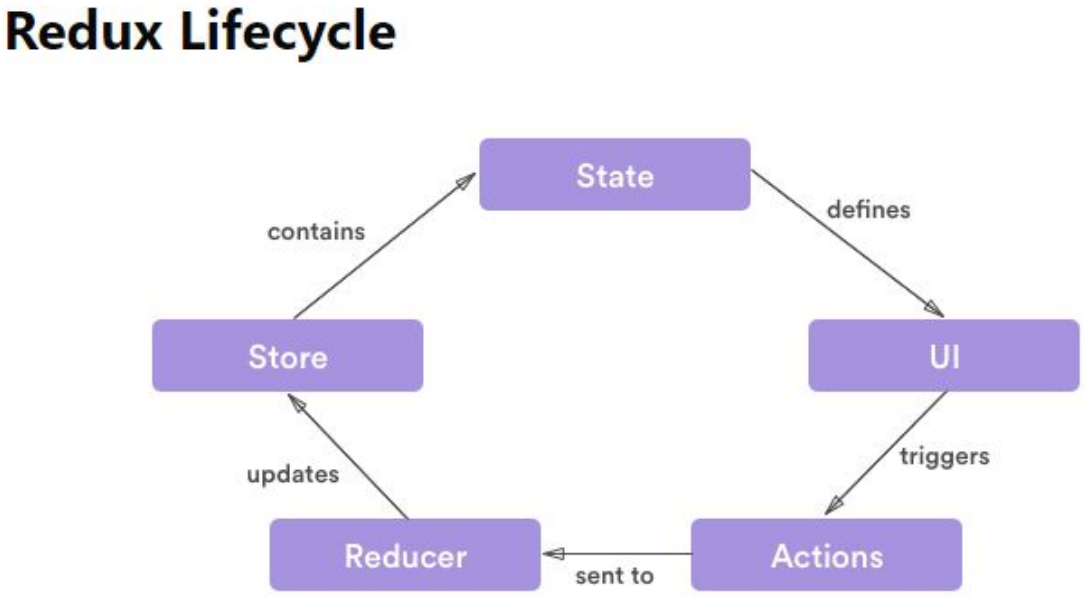
To specify how the state tree is transformed by actions, you write pure reducers. A reducer is a central place where state modification takes place. Reducer is a function which takes state and action as arguments, and returns a newly updated state.

REDUX

Redux Terminologies

- store
- state
- dispatch
- actions
- action creators
- reducers

REDUX


Redux Lifecycle
State
defines
UI
triggers
Actions
sent to
Reducer
updates
Store
contains