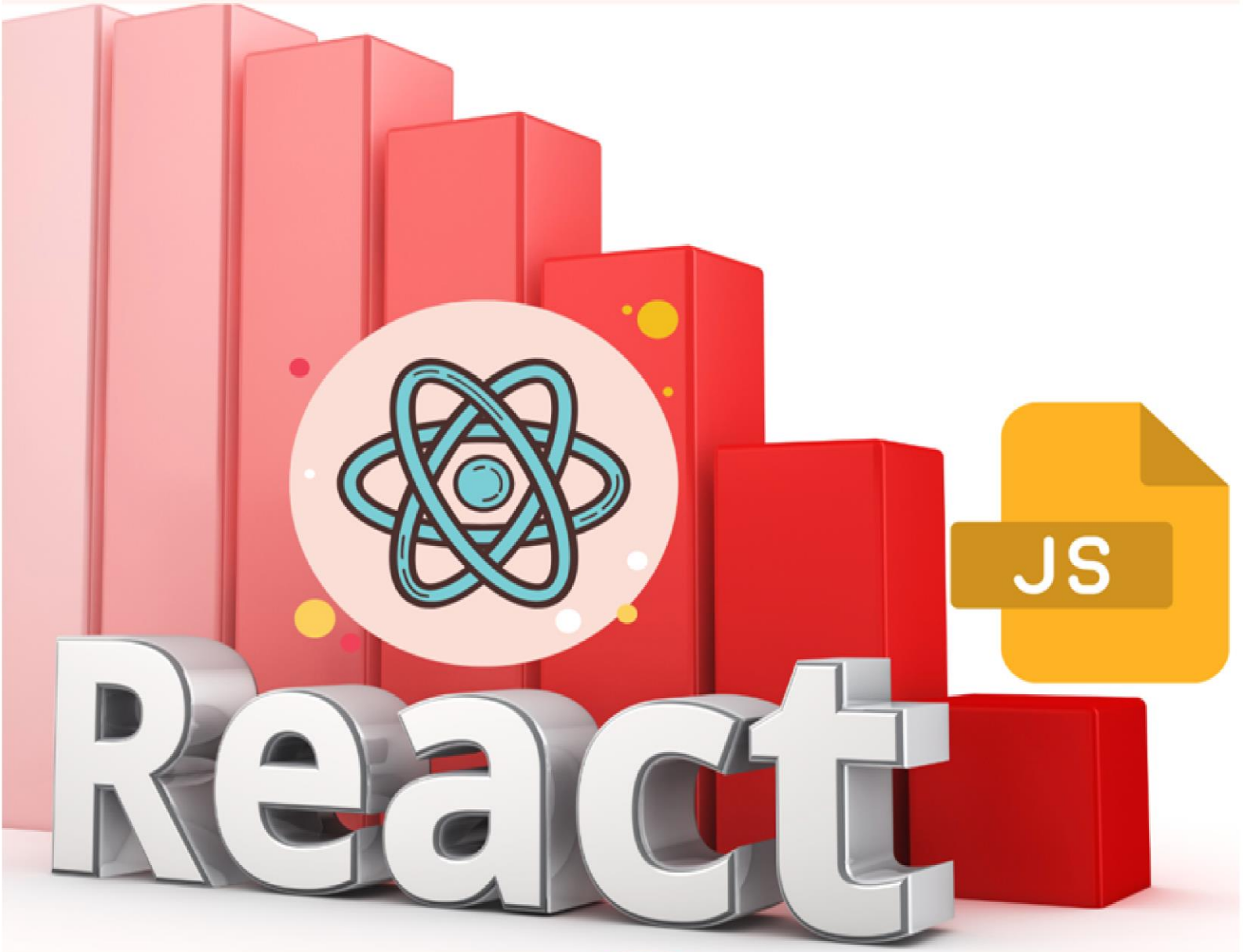
SADIQ TECH SOLUTIONS



React JS

Prerequisites:

- HTML
- CSS
- JavaScript ES6

Topics

- What is React?
- Why React?
- Difference between React, Angular, Vue etc.
- Features and Limitations
- Understanding React Application Architecture
- Components, Props and State
- Styling with Material UI
- Debug React Apps
- Life Cycle
- React Hooks
- Ajax Calls / Http Requests
- Integration with Server Side [MERN]
- Routing
- Forms and Validations
- Deploying
- Redux
- Error Handling
- Unit Testing
- Web Pack
- React Native
- End to End Integration

Which one I have to choose? [Angular or React]

Projects – Which are completely integrated with lot of server-side interactions and every concept is handled at server level only they are looking for good UI to make faster interactions then "React JS".

Projects – Which need to control their application flow both client side and server side – Angular.

React JS is library

Angular is Framework

Mill of Users are accessing some content using content. Only few interactions like message, chat. Not complete EPR. [Not Querying – Transactions, complex operations] – React is Great Twitter,

Instagram

Java - Spring

PHP - Cake PHP, Code Igniter Python - Django, Flask

Ruby - Ruby on Rails

.NET - ASP.NET MVC

JavaScript - SPINE, Angular

MVC – Model View Controller

React is only with "View" UI – Interactions

Web Development 3+ Years: Angular, React

Java, .NET, PHP, Python 3+ years [Angular, React]

What is React and How it Works?

- React JS is a library.
- Client Side we have languages, libraries and frameworks.
- **Client-Side Languages**
  - JavaScript
  - TypeScript
- **Client-Side Libraries**
  - jQuery
  - RxJS
  - jQlite
  - React JS
- **Client-Side Frameworks**
  - Angular JS
  - Knockout JS
  - Backbone JS
  - Angular

What is difference between Language and Library?

- Language requires functions defined explicitly to handle various interactions.
- Language requires lot of references.
- Library is a set of factories.
- Factory is a set of pre-defined functions.
- You can implement the existing function and define the functionality.
- Library reduces the compatibility issues.
- React JS is a library.

What is difference between Library and Framework?

- Library can build application.
- Library is passive [not-active]
- Library can't control the application flow.
- It requires browser events to make library active.

Framework is a software architectural pattern that provides set of libraries to build application and

can also control the application flow.

What is React JS?

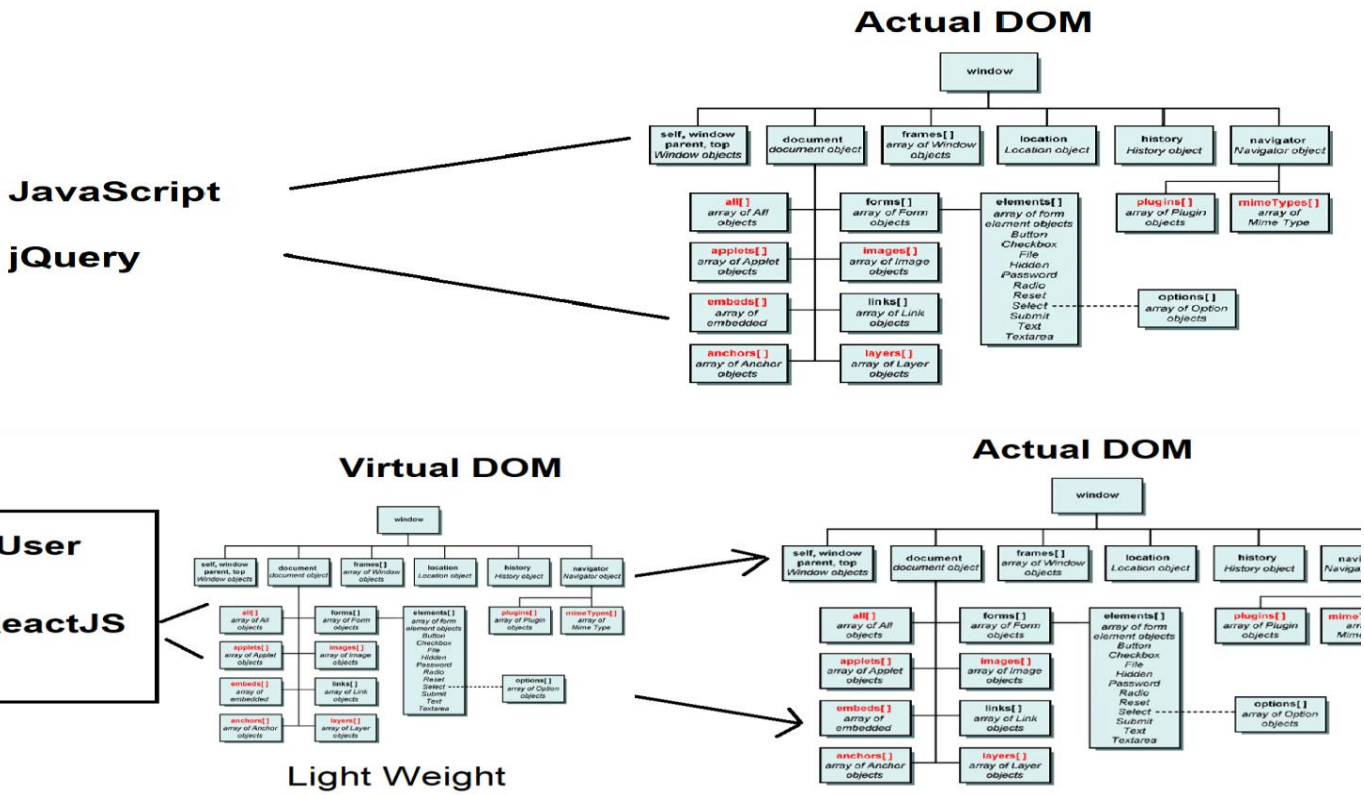
- React JS is a JavaScript library.
- Building User interfaces and handling User interactions client side.
- Facebook – Jordan Walke – for building SPA.
- 2013 Initial release.
- 2020 – 17.0.1 latest stable version.

It is not designed for what you are using. Hence limited library.

React JS library requires several external libraries to integrate and handle functionality.

Features of ReactJS

- ReactJS uses virtual DOM that makes the user experience better.



- Faster Interactions.
- React uses JSX, which is easy to handle the manipulation on DOM elements.
  - JSX is faster that JavaScript DOM manipulations.
  - Logic and markup can be defined in single file.
  - It is easy to create template.
  - Template comprises of presentation and logic.
  - React uses “One Way Data Binding”
  - Data binding is a technique used to update the component data to View [UI]
  - Any change in component data will update to UI. [HTML]
  - Changes made in UI are not directly update back to component, they are handled on

    virtual DOM.
- Component based architecture.
  - It is an alternative for legacy type library.
  - It is more asynchronous
  - It loads only what is required for the situation.
  - It improves the performance of application.
  - It makes your library modular.

Concerns with ReactJS

- Very high pace of development.
- Problem for developers to catch and update the pace of developing features.
- Poor documentation.
- Poor SEO
- JSX [Presentation with Logic] – Insecure content and ignore

  XSS [Cross Site Scripting Attacks]
- Need lot of external libraries to handle various interactions.

Setup Environment for ReactJS Install Node JS

- Node JS provides a package manager NPM.
- Package manager is a tool used to install and configure library required for our application.
- There are several package managers like: Yarn, NPM, Bower, Ruby Gems, NuGet etc.
- Installing Node JS on your PC will get you NPM.
  https://nodejs.org/en/download/

  [Make sure that Node JS version 10+]
- After installing Node JS check the version from command prompt

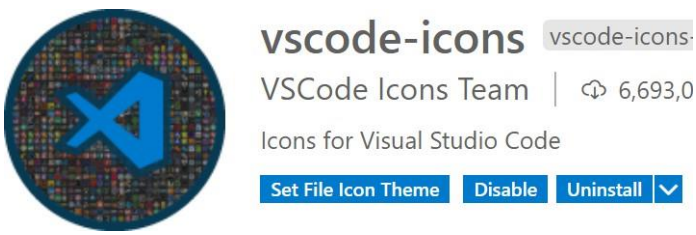
  > node -v

  > npm -v

Install Visual Studio Code [IDE – Integrated Development Environment]

- IDE provides environment to build, debug, test and deploy application.
- You can use various editors: "editorconfig.org"
- Download and install VS Code
  https://code.visualstudio.com/
- Go to "Customize" and "Install Support for JavaScript"
- Go to "Extensions" and Install following
  - **Live Server:** It required to preview your application.

  - **Vscode-icons:** It gives suitable icons for specific file extensions and names, so that developer can easily identify the file type.



- **IntelliSense for CSS Class Name:** It provides help on CSS

Creating a new Project

- Create a new folder for project by name "ReactApp" on your PC

  "C:\ReactApp"

- Open your folder in Visual Studio Code: File Menu
  -> Open Folder -> C:\ReactApp
- Open Terminal: Go to Terminal Menu and select "New Terminal" or Ctrl + ` [backtick]
- Run the command "npm init"

  C:\ReactApp> npm init
- It will generate "package.json" that comprises of project meta data.

  package name: (reactapp) reactapp version: (1.0.0)

  description: This is our first react application

  entry point: (index.js) test

  command:

  git repository:

  keywords: react shopping, smart shopping author: Naresh I Technologies

  license: (ISC) MIT

- **Install React Modules into Project**
  C:\ReactApp>npm install --save react
  C:\ReactApp>npm install --save react-dom
- Installed library will be maintained in "**node_modules**" folder

Create a React Application

> npm install -g create-react-app

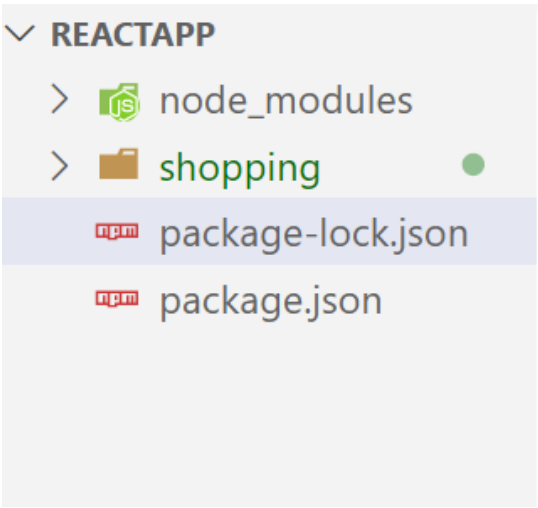
> create-react-app shopping

> Change to shopping
C:\reactapp\shopping>

> **npm start** [Start the react application]

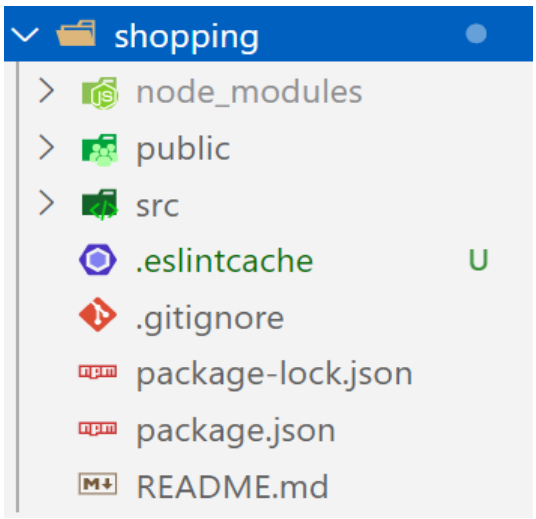
React Workspace

- It comprises of shared libraries for all your ReactJS projects.
- In workspace you can maintain multiple projects
- Workspace comprises of



File / Folder	Description
node_modules	- It comprises of project "dependencies" and
	"devDependencies". - The library required for your react application is in "node_modules". - These dependencies can be shared to all projects in workspace.
package.json	- It comprises of information about the packages installed in your workspace.
packagelock.json	- It comprises of packages meta data.
Shopping	- It is your react application in the workspace.

The files and folders of React Application



File / Folder	Description
node_modules	It contains local dependencies.
public	It comprises of all our project static files, like Images, html, text etc.
src	It comprises of all our project dynamic files like, CSS, JavaScript etc. [LESS, SASS]
.eslintcache	It is used to track and identify the issues in your JavaScript code. It can configure the rules for JavaScript.
.gitignore	It specifies the dependence to ignore while deploying of GIT.
Packagelock.jso n	Local dependencies meta data
Package.json	Local dependencies information.
README.md	Help document.

Files in Public folder

favicon.ico	Shortcut icon used for bookmarking.
Index.html	It is the start up page.
manifest.jso n	It comprises of metadata. Information about react application. Browsers can understand your application by accessing details from manifest.
Robots.txt	It comprises of User Agent details, which is information about the browsers supported.

Files in SRC folder

App.css	It comprises of global styles used for app component.
App.js	It comprises the logic required for default component that is "app" component. Every project contains set of several components. React application by default comes with "app" component. Application starts with "app" component. Every component comprises of 3 files - .css : Styles - .js : Logic (JSX – HTML and JS) - .spec/test.js : It is test file
App.test.js	It is the test file for "app" component.
Index.css	Styles for Index Page
Index.js	JSX for Index Page
Setuptests.js	It configures the testing environment for application testing.
reportWebVitals.js	Error reporting

What is Toolchain for React?

- Toolchain is a set of programming tools that are used in performing a complex software development task or to create a software product.
- In general, "toolchain" is a set of tools used for building/developing an application.

  Ex:

  Language: used for writing the logic Compiler : for translating in machine format.
  Debugger : to identify and fix the bugs. [error/issues]
  Testing : a framework for testing
  Deployment : tool for deploying application on production.

What is Recommended toolchain for ReactJS Application?

- Package Manager [NPM, Yarn]
- Bundler [Webpack, Parcel]
- Compiler [Babel – differential loading]

Package Manager	Installing and Maintaining Dependencies EX: NPM, Yarn
Bundler	Writing modular code and bundle it together for application. Ex: Webpack, Parcel
Compiler	It allows to handle differential loading. Library will be classified according to browser. [Legacy , Modern]

What are the popular toolchains used for ReactJS?

- Neutrino
- Nx
- Parcel
- Razzle

When we need the individual toolchains?

- When we are trying to integrate ReactJS into existing application.

What is NPX?

- It is a tool that comes with NPM.
- It can directly create react application.
- We don't require "create-react-app" to install if we are using "npx".

> *npx create-react-app yourAppName*

How to add "ReactJS" into existing application?

- Create a new folder for project "C:\HtmlReactApp"
- Open your folder in "Visual Studio Code"
- Open Terminal [Ctrl + `] or Open folder location in command prompt
- Run the command

> npm init {

"name": "htmlreact",

"version": "1.0.0",

"description": "HTML Website with React", "main": "index.js",

"scripts": {"test": "echo \"Error: no test specified\" && exit 1"},

"keywords": ["HTML","Application", "React","Application"],

"author": "Naresh-I-Technologies", "license": "MIT"}

"package.json" file is created

- Add a new folder into project by name "public"
- Add a new file into public folder "index.html"
- You can install dependencies for project using "npm"

  Ex: >npm install bootstrap

- It will generate "node_modules" and "package- lock.json"

- To integrate "ReactJS" into your page you need
  - react.development.js
  - react-dom.development.js
- You have get these file from official ReactJS website CDN links

  https://reactjs.org/docs/cdn-links.html

  - Add following scripts into head section

```
<script crossorigin
src="https://unpkg.com/react@17/umd/react.de velopment.js">
</script>
<script crossorigin
src="https://unpkg.com/react-
dom@17/umd/react-dom.development.js">
</script>
```

- Add Babel CDN to your page https://babeljs.io/docs/en/babel-standalone

```
<script src="https://unpkg.com/@babel/standalone/bab
el.min.js"></script>
```

Create React JS Component

- Components are building blocks for creating applications.
- Technically component is a template.
- It comprises of design and functionality, which you can inject and use in any page.
- Components will provide re-usable templates for React JS application.
- Components are classified into 2 major types
  - Function Component
  - Class Component
- Component is designed by using JSX [JavaScript Extension]
- React JS component can be embedded into Page or can be defined in module.
  - Embedded Technique
    Component is designed and used in the same page.
  - Module Technique
    Component is designed in a separate JavaScript [".js"]

    Component is linked to any page.

Ex: React JS components embedded into Page

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML React App</title>
        <script crossorigin
src="https://unpkg.com/react@17/umd/react.develop ment.js"></script>
        <script crossorigin src="https://unpkg.com/react- dom@17/umd/react-
dom.development.js"></script>
        <script crossorigin
src="https://unpkg.com/@babel/standalone/babel.mi n.js"></script>
        <script type="text/jsx"> ReactDOM.render(
            <p>Welcome to React JS</p>,
            document.getElementById('root'));
        </script>
        <script type="text/javascript"> function
            bodyload(){
                alert("React Integrated");}
        </script>
    </head>
    <body onload="bodyload()">
        <h2>HTML React App</h2>
        <div id="root">
            </div>
         </body>
</html>
```

Note: The MIME type of React JS "JSX" script must be "text/jsx" or "text/babel"

Syntax:

```
<script type="text/jsx"> </script>
<script type="text/babel"></script>
```

Embedded Component:

- In embedded technique the component is designed in the page.
- It is faster in access.
- It reduces the load time.
- It reduces the number of requests made to page.
- Code reusability issues
- Extensibility issues
- Maintainability and testability are the issues.

Module Technique:

- Component is designed in a separate JavaScript file ".js"
- JavaScript file is considered as a Module.
- Module comprises of classes, functions etc.
- Easy to extend
- Easy to re-use
- Easy to test
- Using an external file will always increase the number of requests made to page and also the page load time.

Ex:

- Go to "src" folder and add a new file "hello.js"
- Add the following code into "hello.js"

```
ReactDOM.render(
   <p>Welcome to React JS - External Component</p>,
    document.getElementById('root'));
```

Link to your HTML page.

```
<!DOCTYPE html>

<html>
    <head>

        <title>HTML React App</title>

        <script crossorigin
src="https://unpkg.com/react@17/umd/react.devel opment.js"></script>

        <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-
dom.development.js"></script>

        <script crossorigin
src="https://unpkg.com/@babel/standalone/babel. min.js"></script>

        <script type="text/jsx"
src="../src/hello.js"></script>

    </head>

    <body>

        <h2>HTML React App</h2>


        <div id="root">


          </div>

         </body>

</html>
```

Note:

- This is a **preview React JS e-Book** containing **only 30 pages**.
- It is provided to help you understand **how the full React JS e-Book and is structured**.
- The **complete React JS e-Book** includes detailed concepts, real-world examples, and career guidance.
- **Purchase the full React JS e-Book for just ₹249**.
- Buy now from our official website: https://topitcourses.com/react-js/