MERN

About Me

Corporate Trainer & Project Consultant (Open Source & Microsoft Technology)

Conducted 100+ training sessions for close to 1500+ participants from 50+ Indian corporate clients like ***MICROSOFT, TCS, INFOSYS , COGNIZANT, CAPGEMINI, ACCENTURE, HCL, WIPRO, LTI*** *etc. with a success results*

Certifications

Certified Trainer For Google & Microsoft Technologies
Technical Consultant for MEAN Stack, MERN Stack, DEVOPS & DRUPAL

Key Skills

MEAN STACK, MERN STACK, MEVN STACK, LAMP STACK, FULL STACK (.NET, PHP), DEVOPS, DRUPAL, .NET CORE, AWS, AZURE, SQL SERVER, MONGODB, ANGULAR, REACT, REACT NATIVE, VUE, HTML5, CSS3, JAVA SCRIPT, JQUERY, BOOTSTRAP, PHP, DRUPAL, WORDPRESS, MYSQL, SQL SERVER, DOTNET CORE, DIGITAL MARKETING,

TAPAN KUMAR

Experience

18+ years of total experience as a Web Developer, Corporate Trainer & Project Consultant. Depth exposure to Responsive Web Development, UI Development & CMS Development, Interviewer Panel & Mentor

Training

A highly motivated, enthusiastic, skilled and talented professional with rich experience. obtained MTech and Diploma in SE. Always fascinated with live projects in Web development & utilize hard earned knowledge in the field of Corporate Training.

Conducted 10+ training sessions for International corporate clients like ***AIT Global-USA, Mauritius Institute Of Education-Mauritius, KSL Dubai, NCB Bank Jaddah-Saudi Arab , Trade Development Bank-Mangolia, ENI-Egypt etc.***

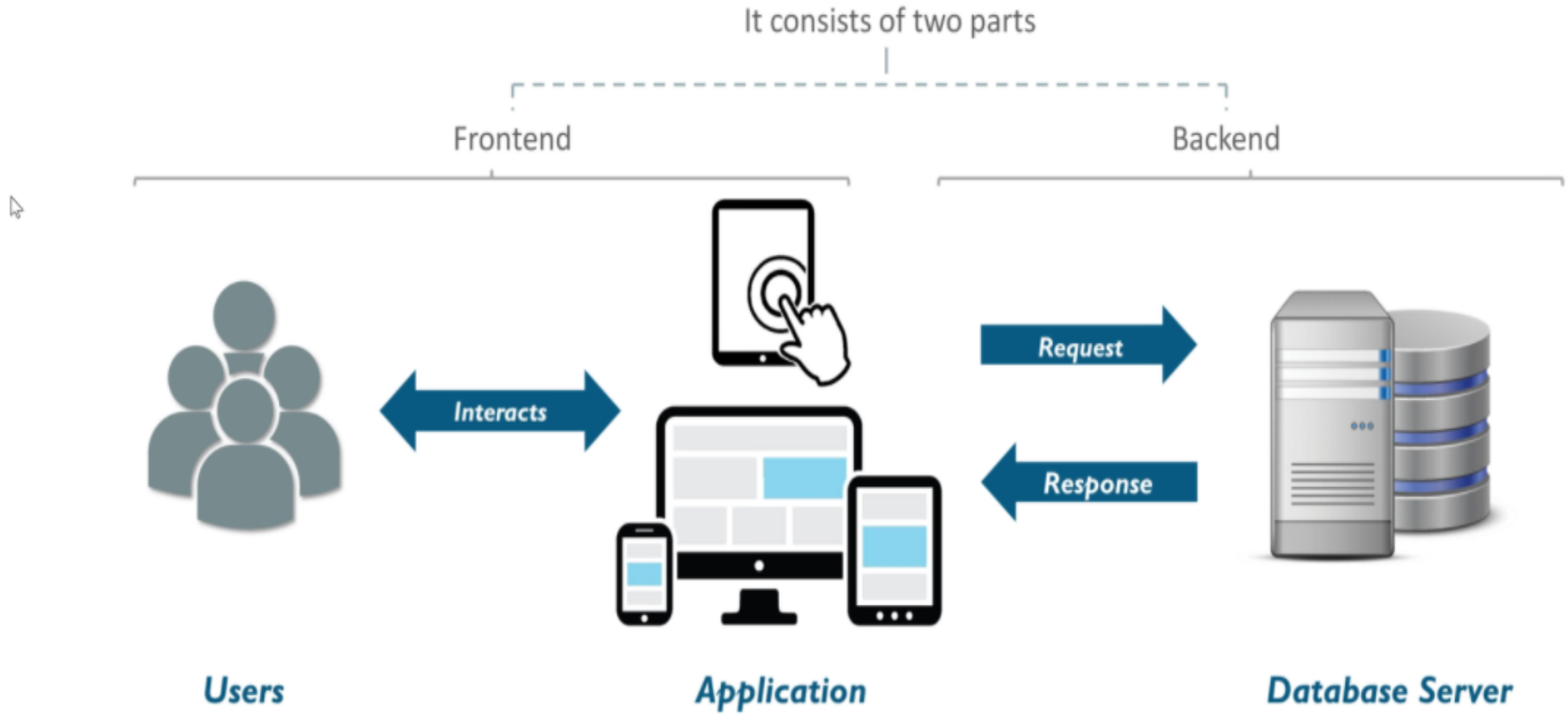
Profile at www.tapanskills.in, Emal : tapanskills@gmail.com, LinkedIn : https://www.linkedin.com/in/tapanprofile/

DRUPAL ON TRENDS

- GOOGLE TRENDS
- INDEED.COM
- NAUKRI.COM
- NPMTRENDS.COM
- GITHUB DOWNLOADS & STARS

Building Blocks Of Web Application Development

Web Application Development is the creation of an application program that reside on remote servers and are delivered to the user's device, over the internet through browser interface.



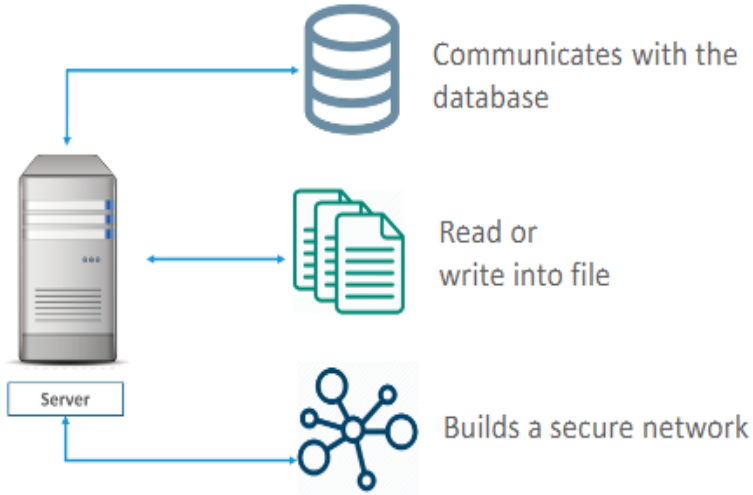
Frontend Development And Backend Development

Frontend Development refers to constructing what a user sees when they load a web application – the content, design and how you interact with it.

01

Components of Frontend

Components of Backend

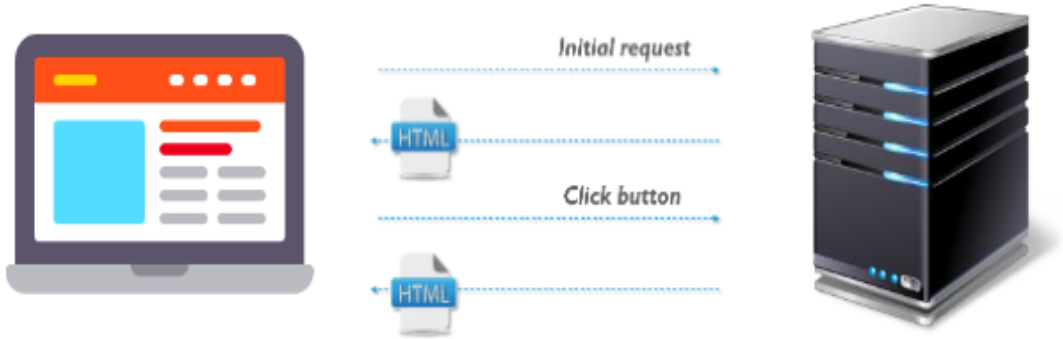


02

Backend Development refers to the server side of an application, where you have to develop a business logic to manage the content, along with security and structure, in order to ensure that application remains operational.

Multi-page Application (MPA)

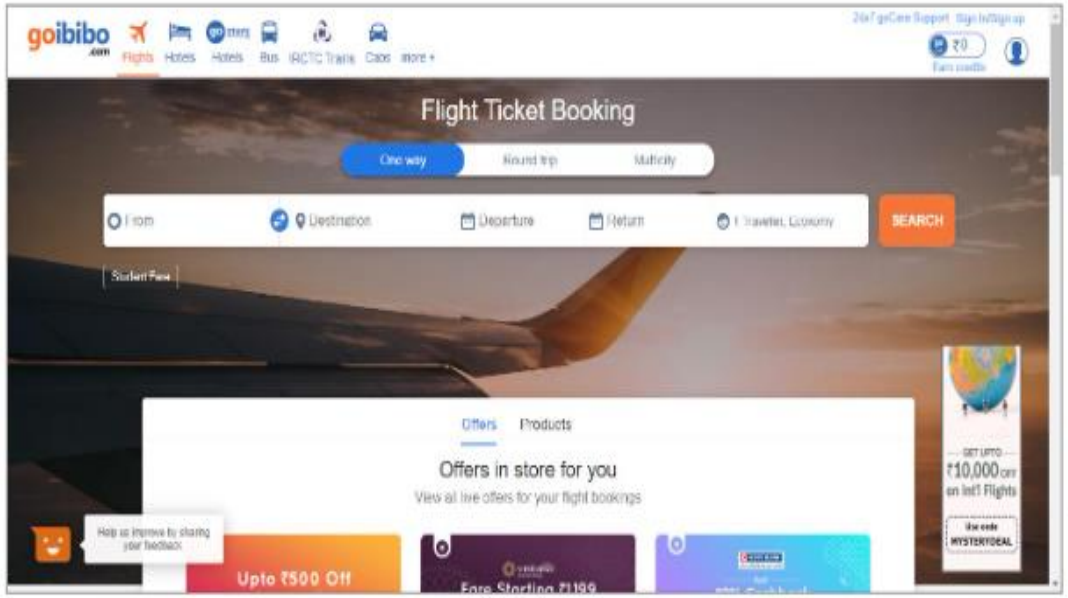
- In ***multi-page application*** each time you click on a link or interact with the application, a new page is downloaded from the server and then rendered in the web browser
- Here content is organized on individual pages that are usually **static**
- They do not change in response to user's action
- A **brand new page** with its own static content is served when a user clicks any button



Single-page Application (SPA)

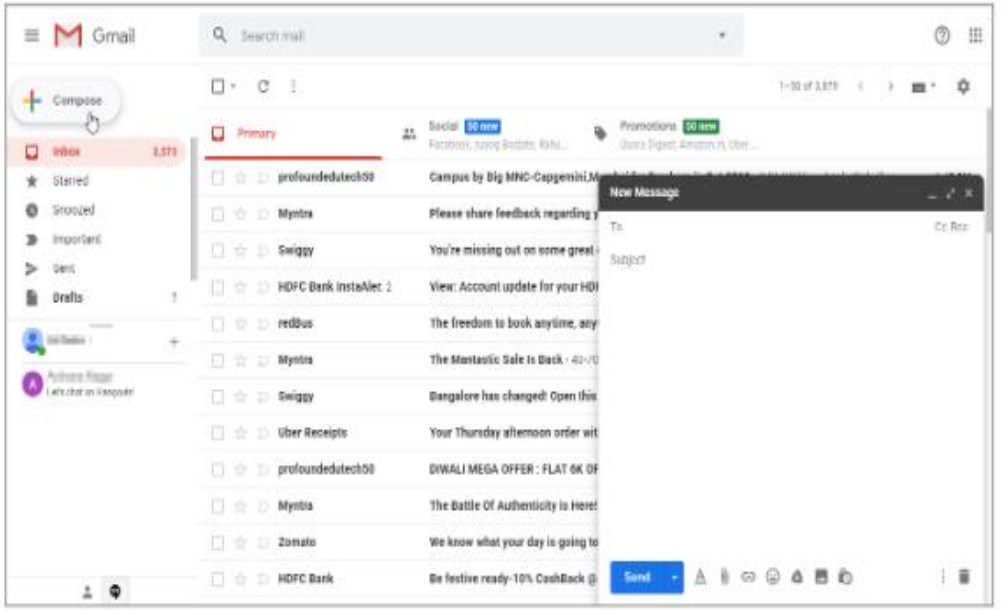
- A ***single-page application*** is an application that works inside a browser and does not require page reloading during use
- Instead of serving a brand new page to the user, SPA swaps out the old content for new in case of any user interaction
- SPA is **faster**, more **responsive**, **compact, easy to develop and deploy**
- SPA and all its content is **only loaded once**, when the user first interacts with the web page

Examples



Multi-page Web Application

Single-page Web Application



MERN

M - MONGO

E - EXPRESS

R - RECT

N - NODE

MERN -PREREQUISITES

MERN = MongoDB, Express, React , NodeJS

Client

React	• Code is flexible to change & maintainable based on MVC principle • Segregation od Concern into components. Build into JS
TypeScript	• ECMA Script Standard, created by Microsoft • Supports OOPS keywords, eg. Class, extends etc.. • Provides compile time errors , Compiles into JS
JSX = JS+HTML JavaScript XML	• Combines flexibility of JavaScript and HTML to render Views dynamically in React
BootStrap	• Created by Twitter . Open Source. Scaled UI to diff. screen sizes • Supports Responsive Web Design
JQuery	• Open Source Library, reduces no. of lines of code • Higher probability of code executing on diff. browsers • Emits JavaScript into the browser
JavaScript	• ECMA standard for calling DOM API • Client browser only has JS runtime engine
DOM	• W3C API for dynamically modifying the HTML on client side • Parser to convert HTML to tree structure for quick HTML element searches and modification
CSS 3	• W3C standard for styling the UI • Mobile Friendly
HTML 5	• W3C standard for creating page structure / UI • Mobile Friendly

MERN -PREREQUISITES

MERN = MongoDB, Express, React , NodeJS

Client

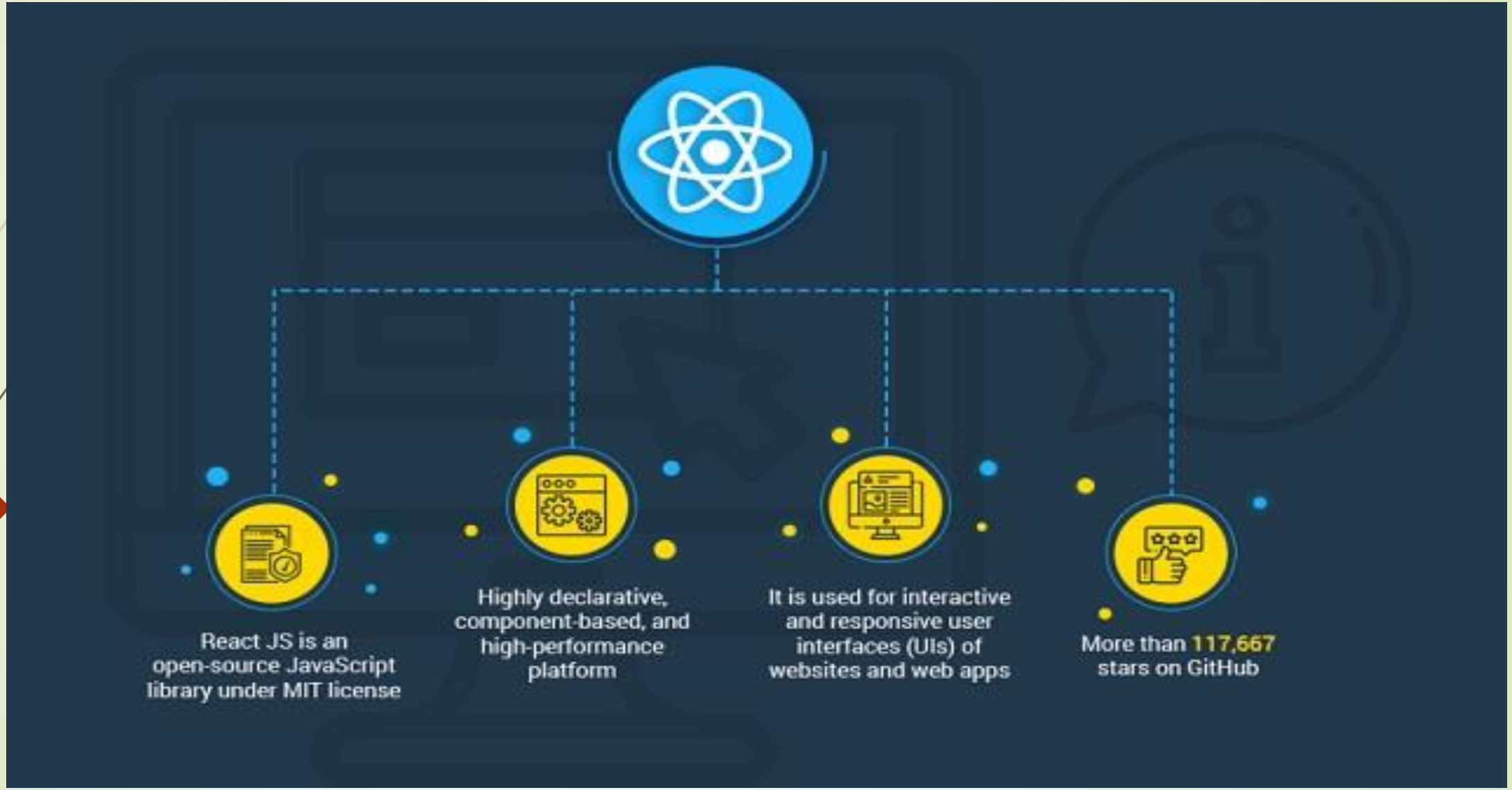
- **React**
  - Code is flexible to change & maintainable based on MVC principle
  - Segregation od Concern into components. Build into JS
- **TypeScript**
  - ECMA Script Standard, created by Microsoft
  - Supports OOPS keywords, eg. Class, extends etc..
  - Provides compile time errors , Compiles into JS
- **JSX = JS+HTML JavaScript XML**
  - Combines flexibility of JavaScript and HTML to render Views dynamically in React
- **BootStrap**
  - Created by Twitter . Open Source. Scaled UI to diff. screen sizes
  - Supports Responsive Web Design
- **JQuery**
  - Open Source Library, reduces no. of lines of code
  - Higher probability of code executing on diff. browsers
  - Emits JavaScript into the browser
- **JavaScript**
  - ECMA standard for calling DOM API
  - Client browser only has JS runtime engine
- **DOM**
  - W3C API for dynamically modifying the HTML on client side
  - Parser to convert HTML to tree structure for quick HTML element searches and modification
- **CSS 3**
  - W3C standard for styling the UI
  - Mobile Friendly
- **HTML 5**
  - W3C standard for creating page structure / UI
  - Mobile Friendly

Web Server

- **Npm Modules**
- **ExpressJS**
- **NodeJS**

- **NODEJS:** Development environment
- **ExpressJS:** Configures NodeJS as a Web Server for Deployment
- **Npm Modules** API to build SPA Apps

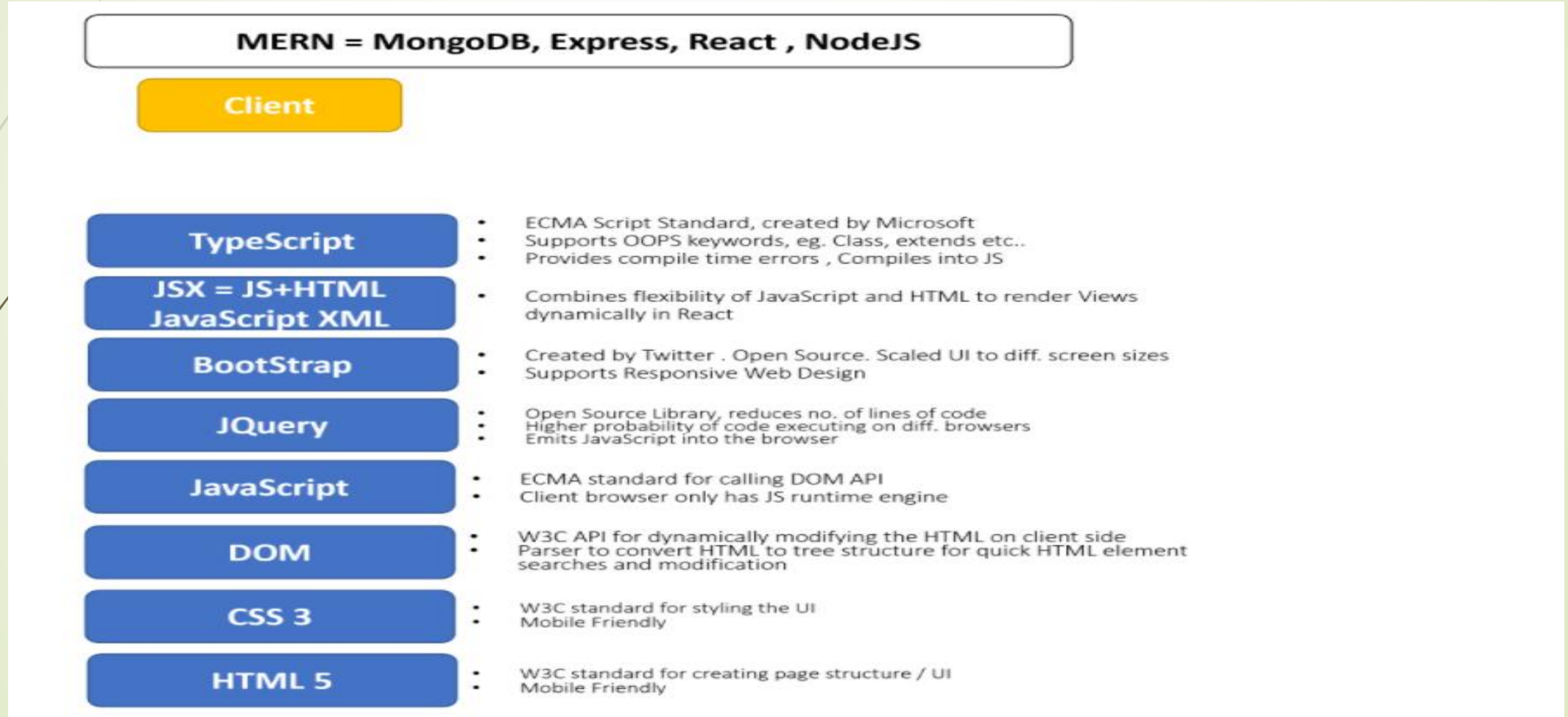
REACT


React JS is an open-source JavaScript library under MIT license
Highly declarative, component-based, and high-performance platform
It is used for interactive and responsive user interfaces (UIs) of websites and web apps
More than 117,667 stars on GitHub

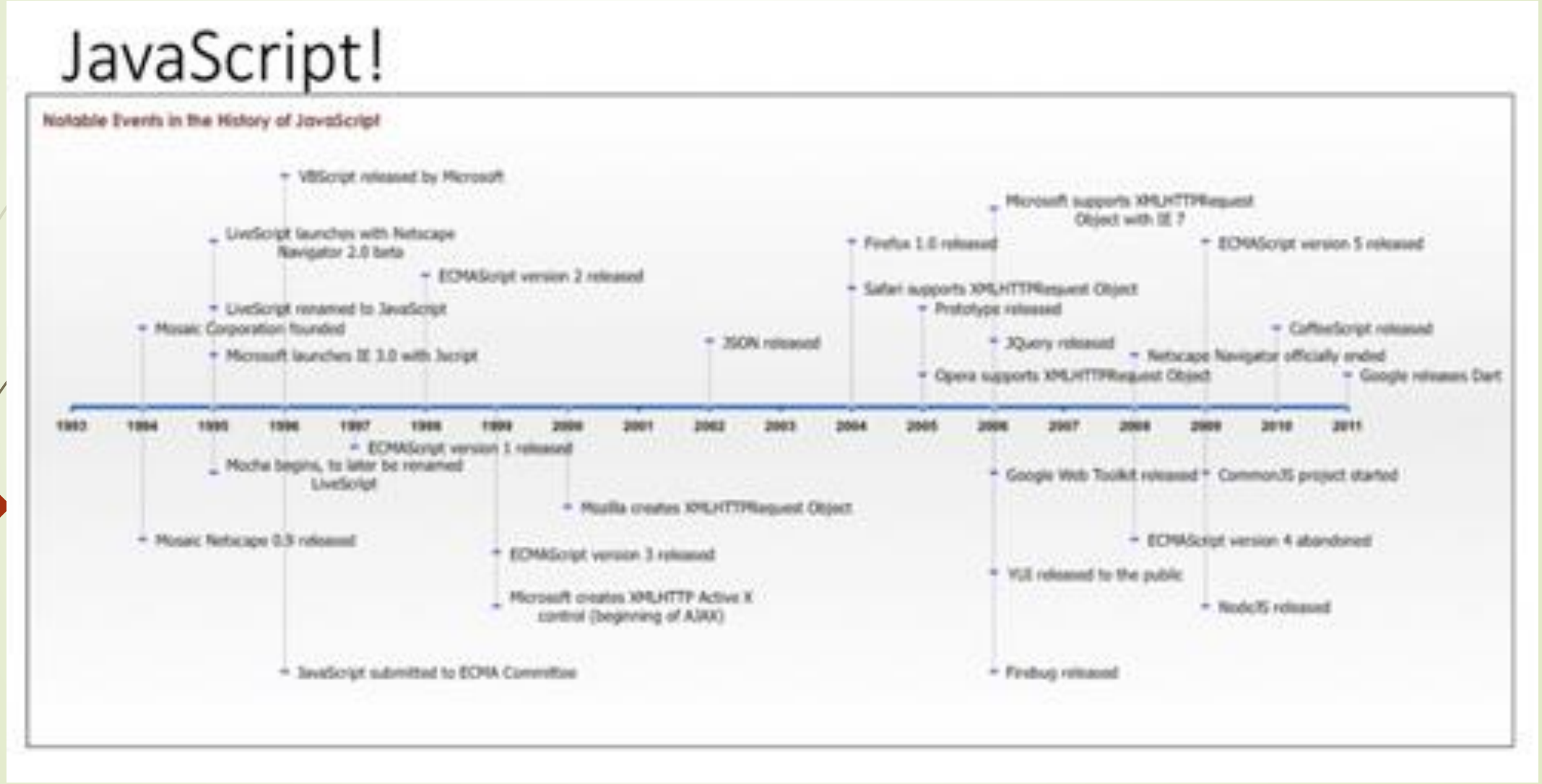
PreRequisites : REACT

- HTML5
- CSS3
- JAVA SCRIPT
- JQUERY
- BOOTSTRAP
- PROGRAMMING SKILLS
- OBJECT ORIENTED PROGRAMMING
- ES5 /ES6
- NODEJS
- JSX

REACT -PREREQUISITES



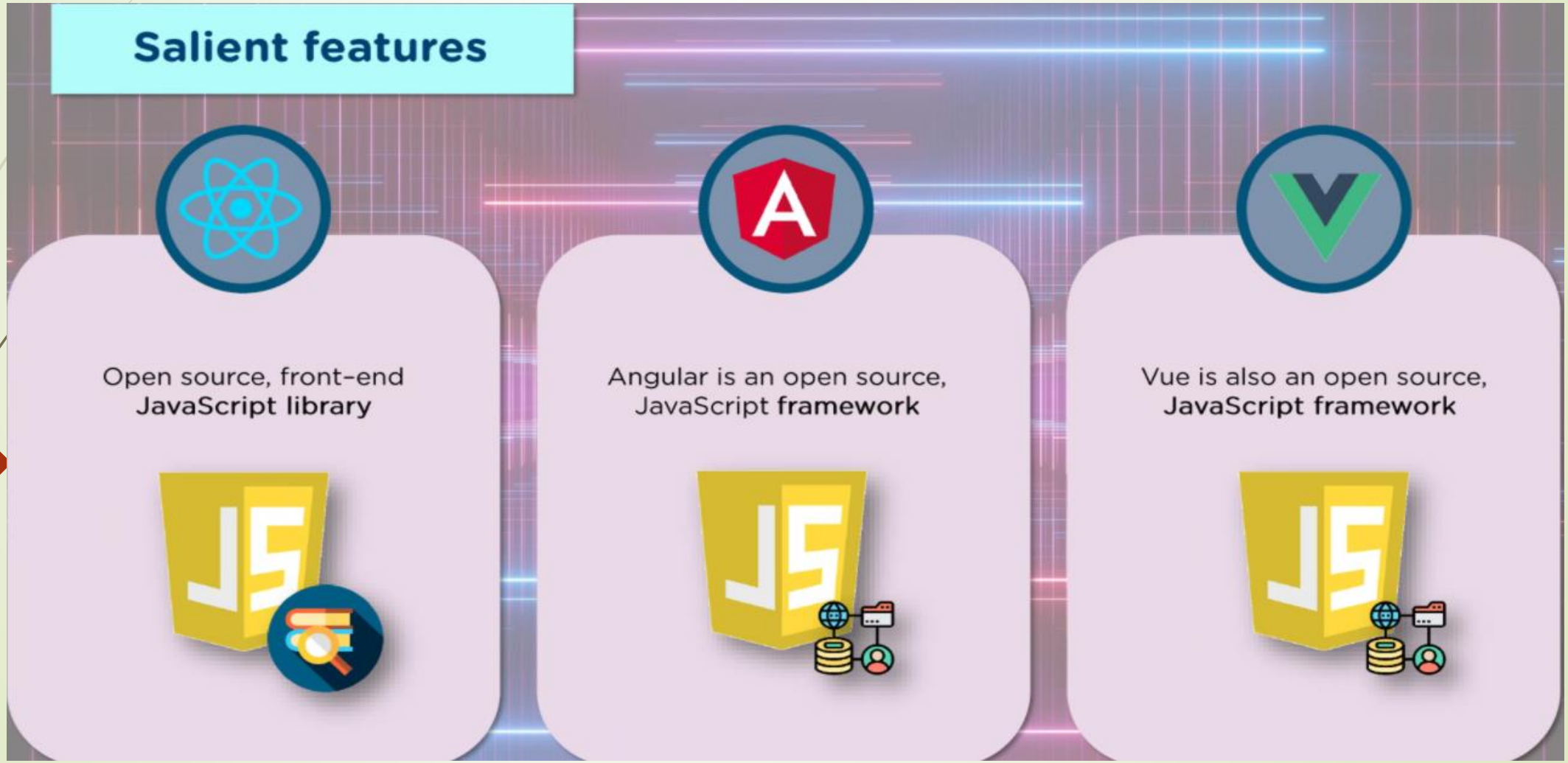
PreRequisites : REACT



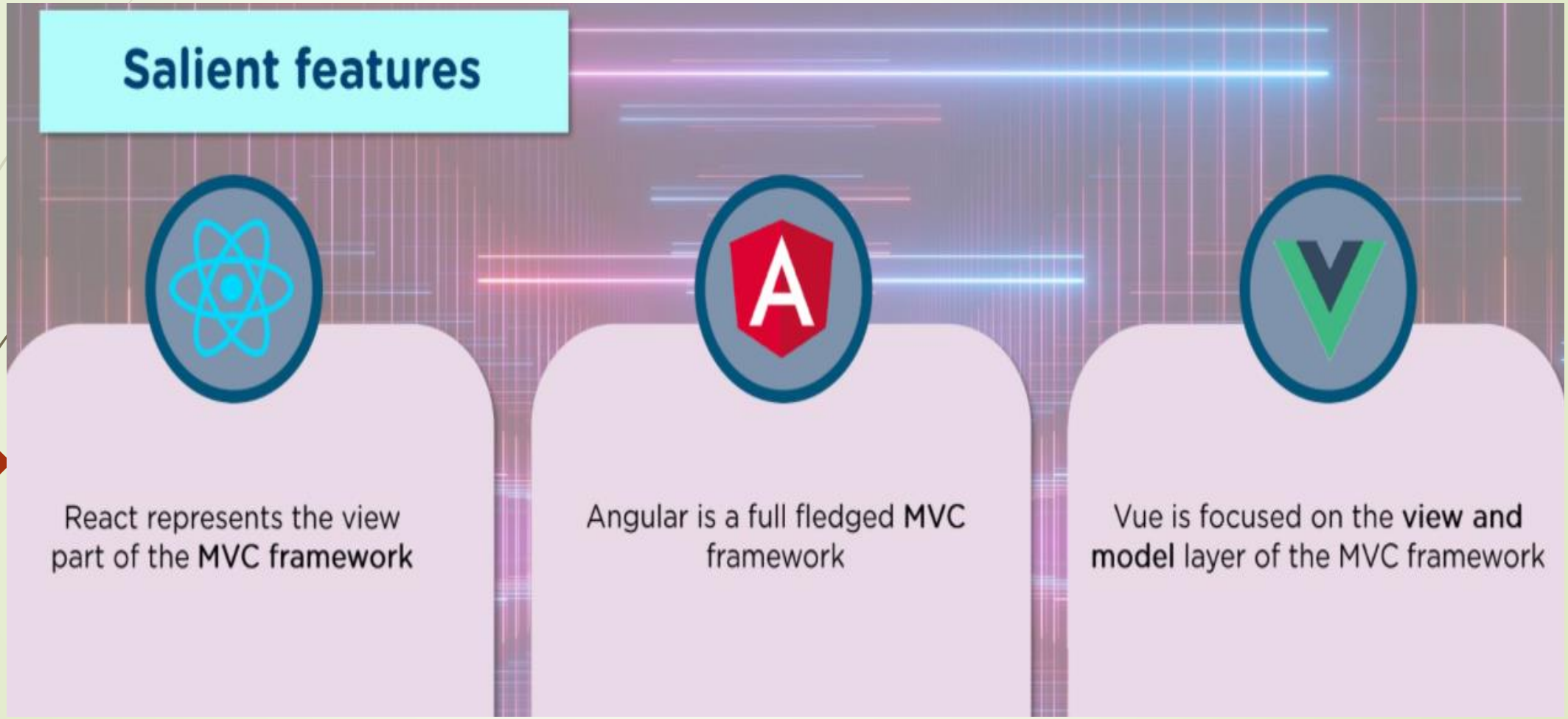
Java Script is Everywhere!

- MEAN STACK
- MERN STACK
- MEVN STACK
- Client Side
- Server Side
- Database Side
- HTML5 based API like GEO- location, Web Worker, Web Storage based on java Script
- Browser Default Script Language
- All Framework ANGULAR, REACT, VUEJS based on Java Script.
- Extending the functionality like language (PHONEGAP)
- ES standard script language
- Chrome OS Chrome Book) – Java Script based OS

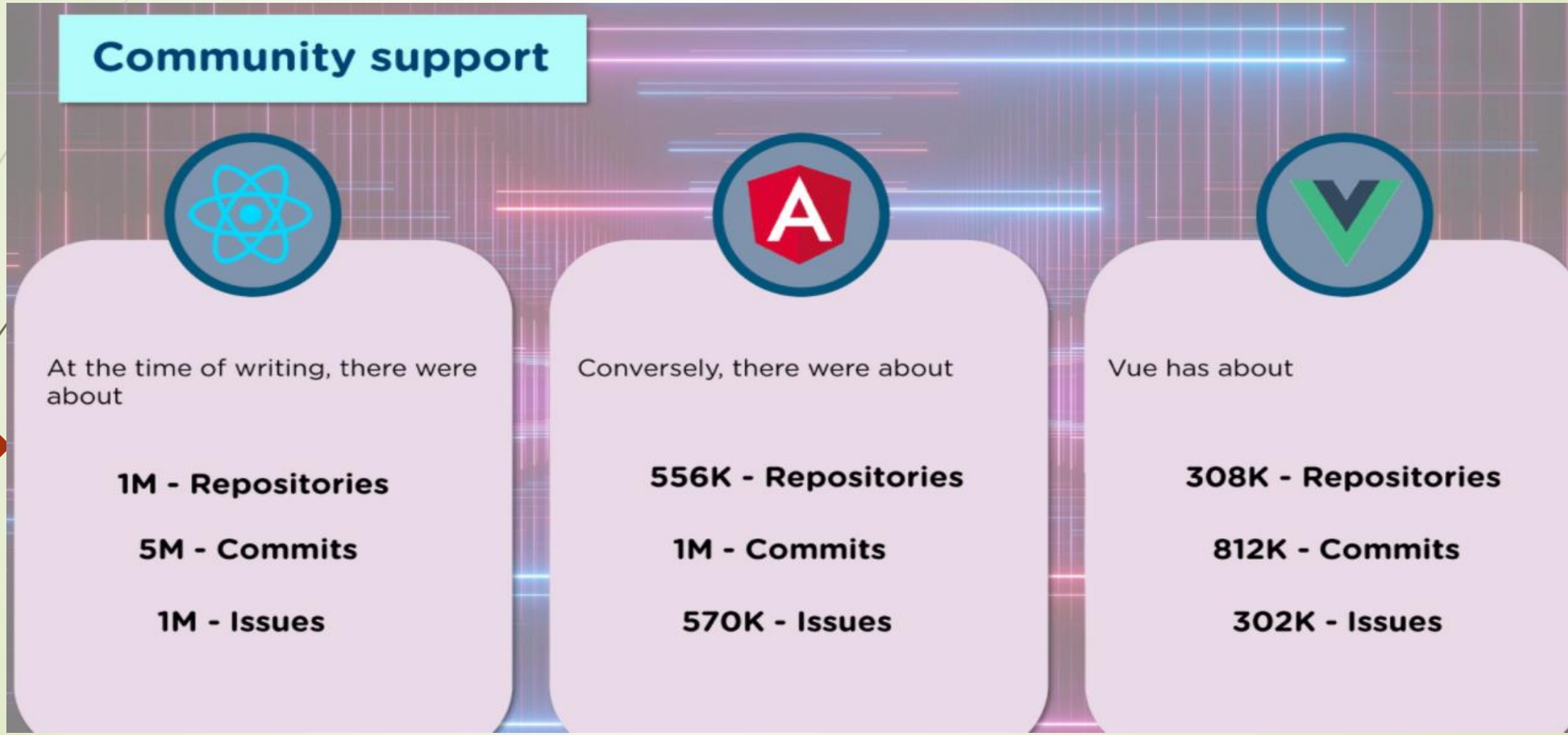
Java Script is Everywhere!



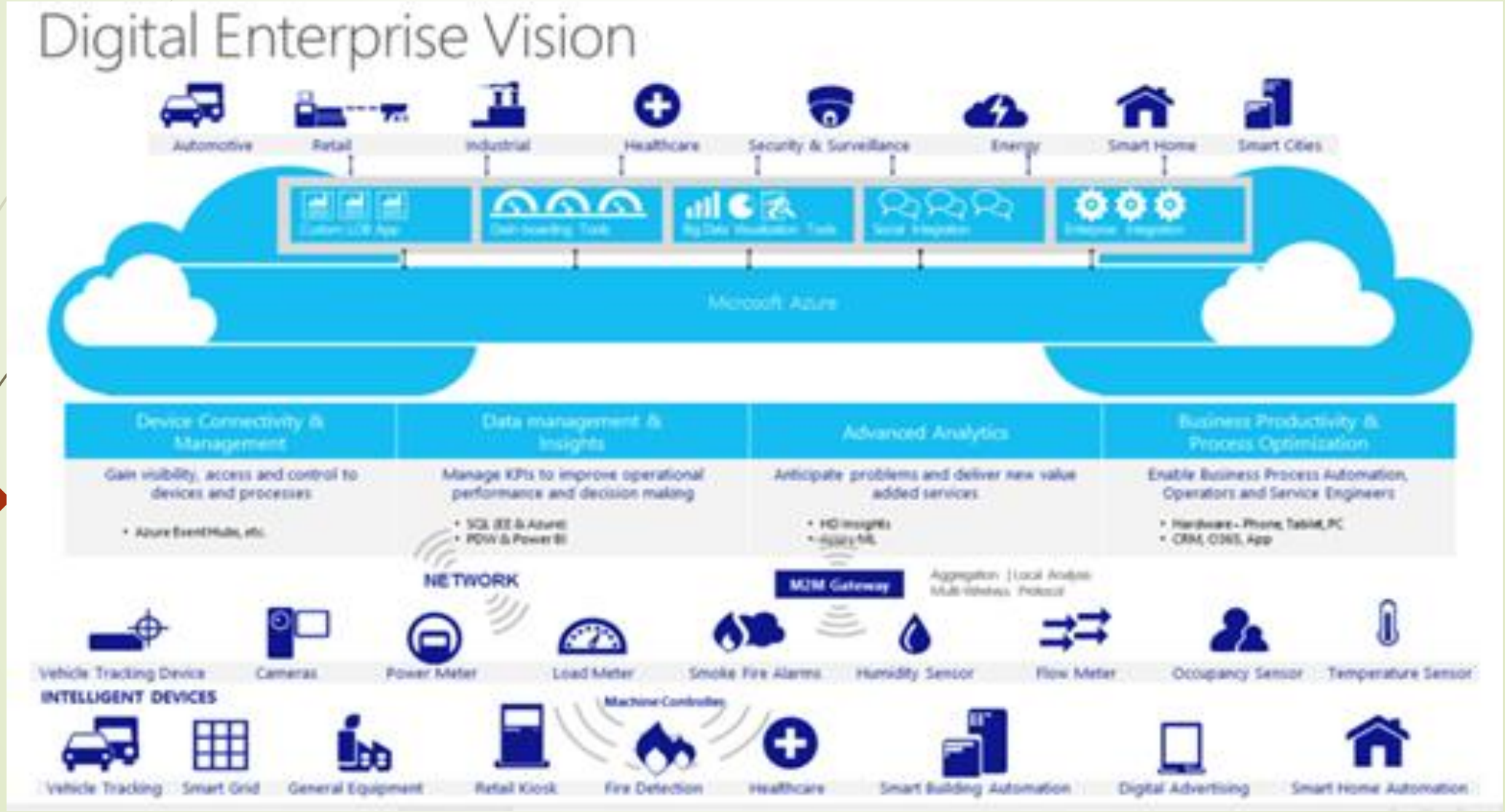
Java Script is Everywhere!



Java Script is Everywhere!



PreRequisites : REACT



PreRequisites : REACT

Standards

ABOUT 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.
- A JavaScript library for building user interfaces
- React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “components”.
- ReactJS is a JavaScript library that combines the speed of JavaScript and uses a new way of rendering webpages, making them highly dynamic and responsive to user input. The product significantly changed the Facebook approach to development.

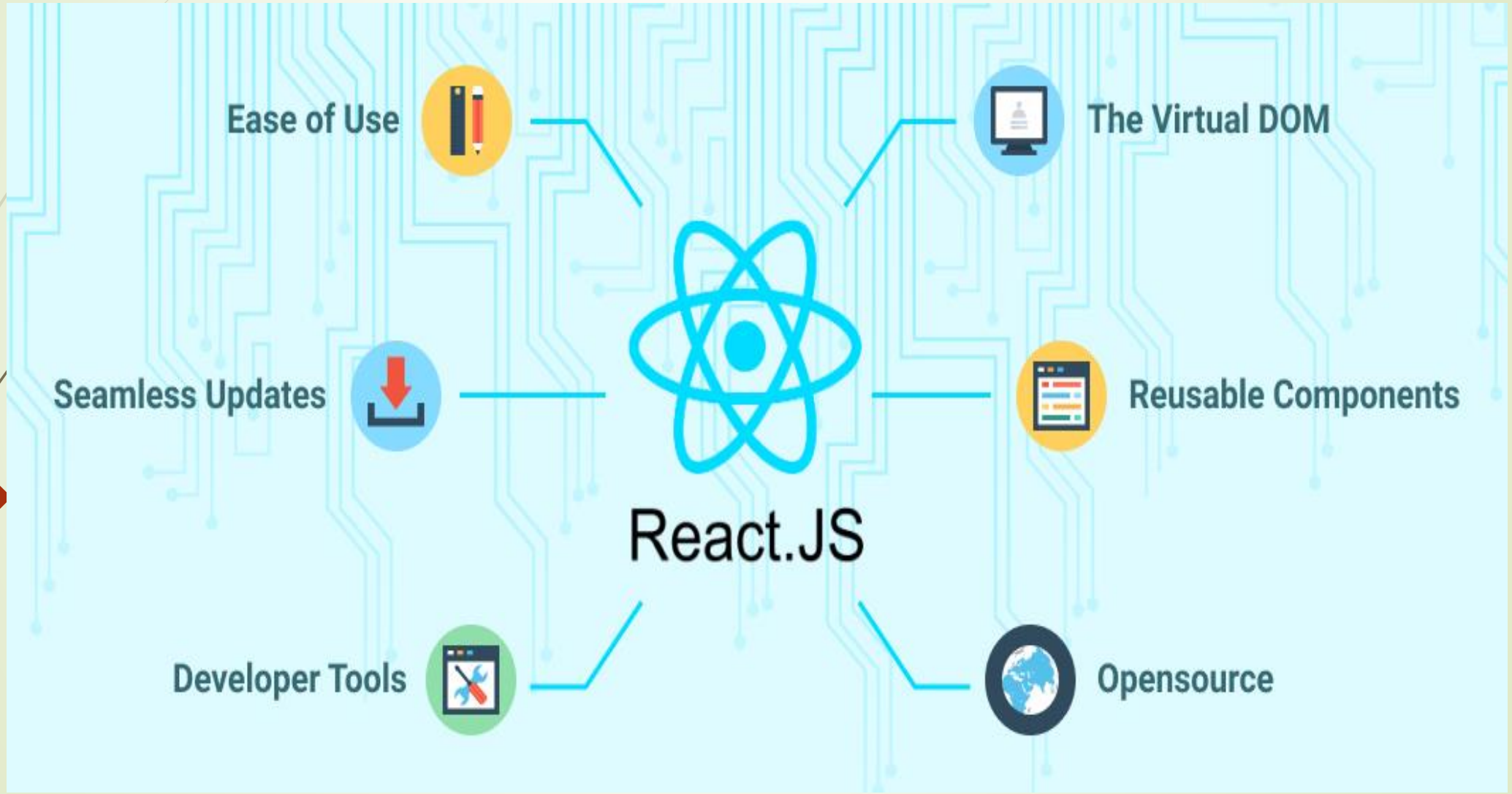
ABOUT REACT

- React is a remarkable JavaScript library that's taken the development community by storm. In a nutshell, it's made it easier for developers to build interactive user interfaces for web, mobile and desktop platforms.
- Today, thousands of companies worldwide are using React, including big names such as Netflix and Airbnb.

Is node js required for react?

It is not true; you DO NOT need **Nodejs** every time you use **React**. **Reactjs** is a library, which is only used to render the user-interfaces of your web and mobile apps. While **Reactjs** can only be used to build UI components on the frontend, **Nodejs** will take care of the data stored on the backend.

ABOUT REACT


Ease of Use
The Virtual DOM
Seamless Updates
Reusable Components
React.JS
Developer Tools
Opensource

Features of ReactJS

JSX : JSX is an extension to javascript. Though it is not mandatory to use JSX in react, it is one of the good features and easy to use.

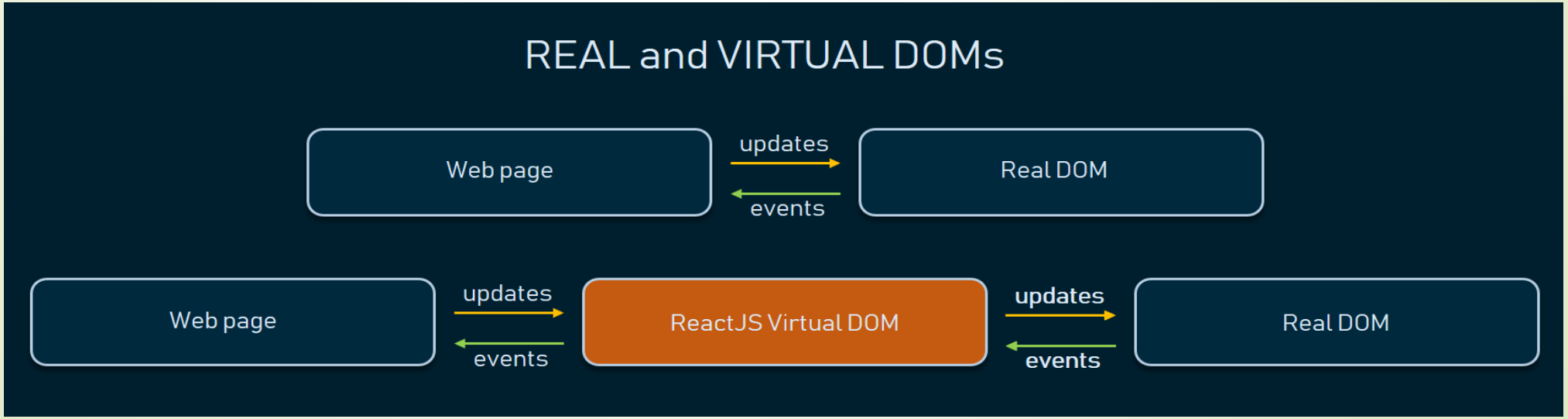
Components: Components are like pure javascript functions that help make the code easy by splitting the logic into reusable independent code. We can use components as functions and components as classes. Components also have a state, props which makes life easy. Inside a class, the state of each of the props is maintained.

Virtual DOM: React creates a virtual dom, i.e., in-memory data - structure cache. Only the final changes of DOM has later updated in the browsers DOM.

Javascript Expressions: JS expressions can be used in the jsx files using curly brackets, for example {}.

REACT : Virtual DOM

- Virtual DOM in React makes the user experience better and developer's work faster
- DOM (document object model) is a logical structure of documents in HTML, XHTML, or XML formats. Describing it in layman's terms, it is a viewing agreement on data inputs and outputs, which has a tree form. Web browsers are using layout engines to transform or parse the representation HTML-syntax into a document object model, which we can see in browsers.



One-direction data flow in ReactJS provides a stable code

React allows for direct work with components and uses downward data binding to ensure that changes in child structures don't affect their parents. That makes code stable.

Most complex view-model systems of JS-representation have a significant but understandable disadvantage – the structure of data flow. In the view-model system, child elements may affect the parent if changed. Facebook removed these issues in React, making it just the view system.

Instead of using explicit data binding, ReactJS uses one direction – downward – data flow. In such a structure, child elements cannot affect parent data. To change an object, all a developer needs to do is modify its state and apply updates. Correspondingly, only allowed components will be upgraded.

Advantages of ReactJS

Here, are important pros/benefits of using ReactJS:

ReactJS uses virtual dom that makes use of in-memory data-structure cache, and only the final changes are updated in browsers dom. This makes the app faster.

You can create components of your choice by using the react component feature. The components can be reused and also helpful in code maintenance.

Reactjs is an open-source javascript library, so it is easy to start with.

ReactJS has become very popular in a short span and maintained by Facebook and Instagram. It is used by many famous companies like Apple, Netflix, etc.

Facebook maintains ReactJS, the library, so it is well maintained and kept updated.

ReactJS can be used to develop rich UI for both desktop and mobile apps.

Easy to debug and test as most of the coding is done in Javascript rather than on Html.

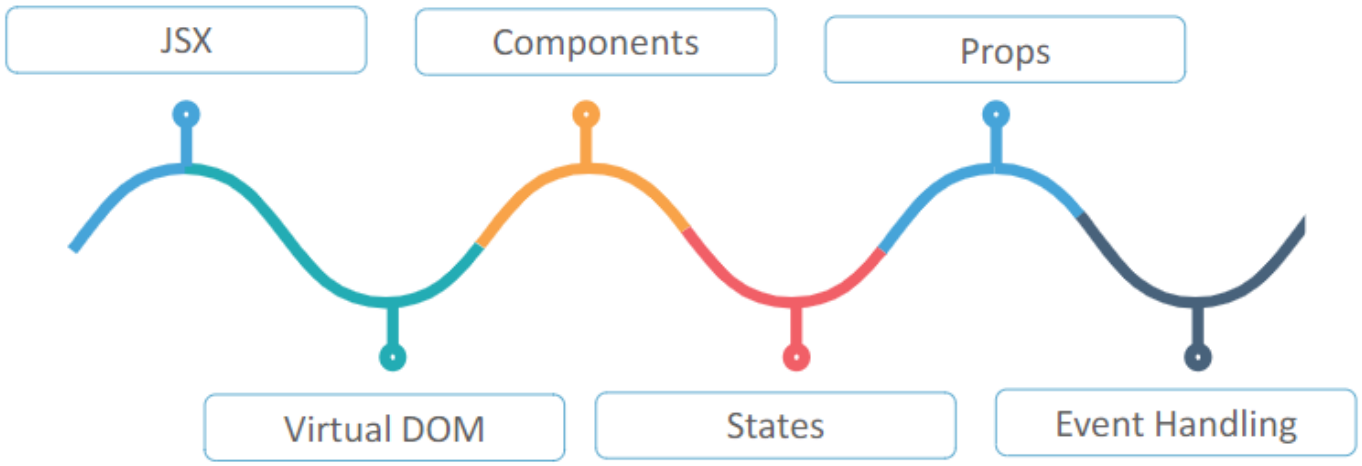
REACT

What Is React?

React (also known as React.js or ReactJS) is a JavaScript library for building user interfaces.

React is used in the development of single-page web application or mobile applications, as it is optimal for fetching rapidly changing data that needs to be recorded.

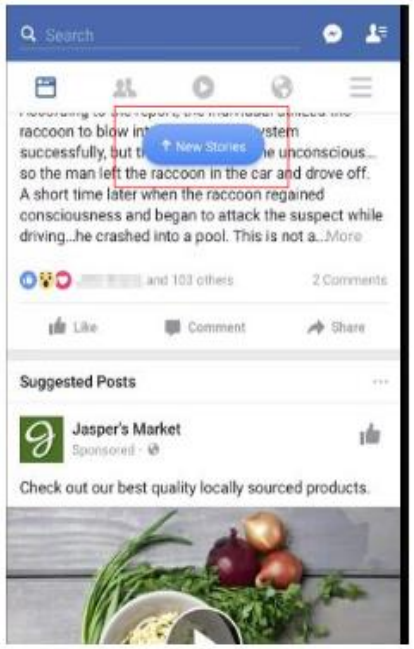
Elements of React:

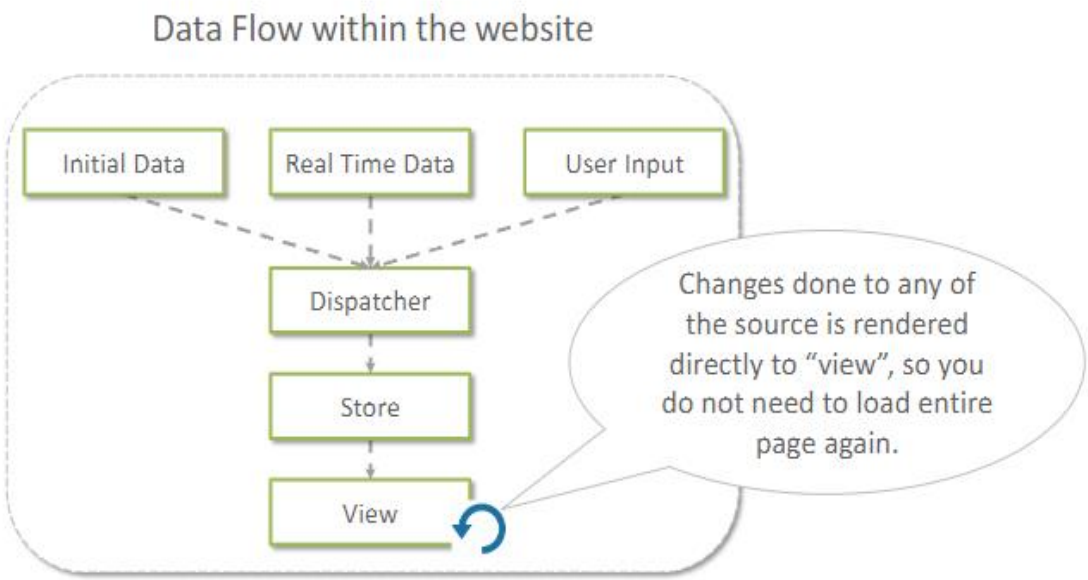


REACT

Facebook: Solution

Facebook after implementation of React became more user friendly. When new data is added, a "New Stories" notification appears. Clicking it, will automatically refresh the newsfeed section

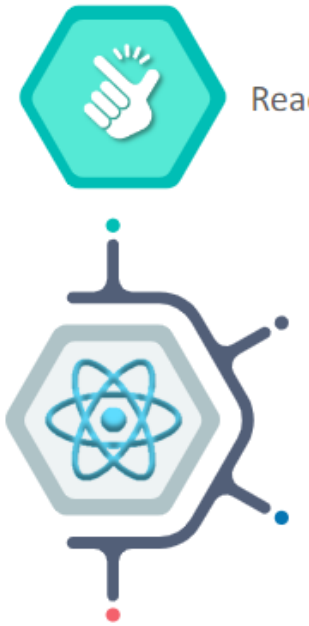




React uses ***virtual DOM*** which optimizes the memory consumption and results to faster load of web pages.

REACT

Why Should We Learn React?



React offers a ***lower learning curve,*** all you need to know is ***JavaScript and HTML***

Elements like JSX, Components, Props, State and Virtual DOM, lets React to ***develop*** application faster with ***smaller code size, easy deployment and rich user interface***

Top corporations such as ***Facebook, New York Times, Yahoo! Mail, Netflix, Instagram*** and more use it to solve their user interface related issues

As React has highest market popularity so obviously there are ***more job availabilities for React developers*** and average salary of React Developer ranges from ***$74,033*** per year for Software Engineer Intern to ***$112,143*** per year for JavaScript Developer

Source: https://www.indeed.com/salaries/Reactjs-Developer-Salaries

REACT

Node Package Manager (NPM)

1. NPM is the world's largest Software Registry with 10 lakh code packages. Developers use NPM to share software. Many organizations also use NPM to manage private development

2. NPM is free to use, you can download all NPM public software packages without any registration or login.

3. All NPM packages are defined in a file called *package.json*. At least two fields must be present in the definition ***file***: ***name*** and ***version***

4. **NPM** is installed with **Node.js.** This means, you have to install Node.js to get NPM installed in your system. Refer LMS to install Node.js

REACT

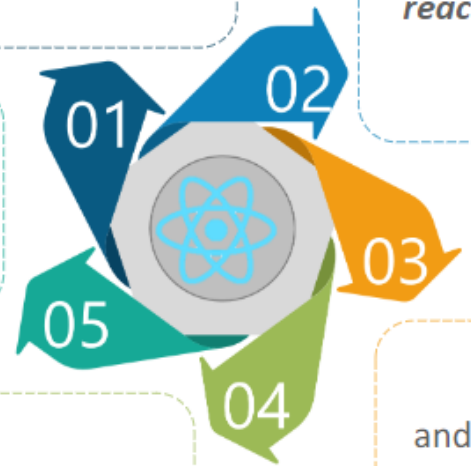
NPM Libraries Involved In React Installation

The necessary NPM libraries for starting any React application are:

react: It contains only the functionality necessary to define React components

react-dom: It serves as an entry point to the DOM

babel: It is used to convert the ES6 code to ES5 version of javaScript



webpack: It is a module bundler, and bundles JavaScript files for usage in a browser

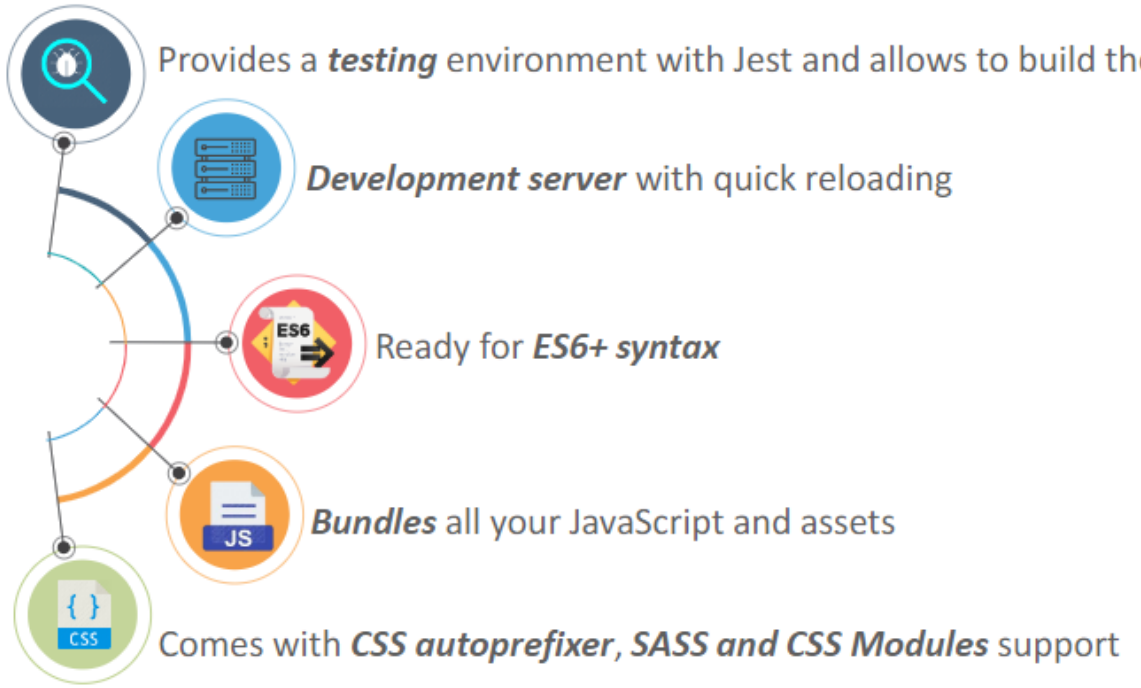
react-scripts: It includes scripts and configurations used by ***create-react-app***

REACT

create-react-app

create-react-app is a NPM package and easiest way to start up a React application.

- It provides a ready-made React application starter, so you can dive into building your application without having to deal with ***webpack*** and ***babel*** configurations
- Create-react-app setup includes:



Provides a ***testing*** environment with Jest and allows to build the React app

Development server with quick reloading

Ready for ***ES6+ syntax***

Bundles all your JavaScript and assets

Comes with ***CSS autoprefixer***, ***SASS and CSS Modules*** support

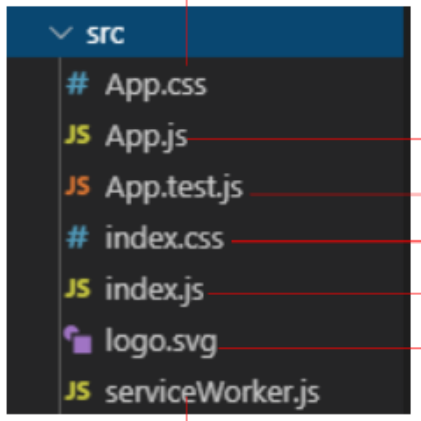
NOTE:

Install this package using the command:

npm create-react-app <app-name>

Advantages of ReactJS

Sub-Folder Structure In React



Designs the required stylings for code present in App.js

Contains the main application code

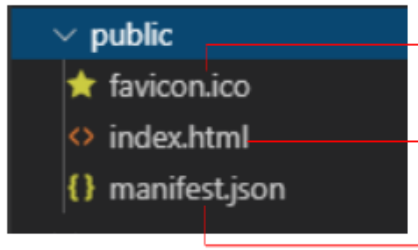
Contains the required test suite for application

Designs the required styling for code present in index.js

It provides an entry point to the web application code

It provides the React Logo

It takes care of caching and updating files for the end user. It helps in offline capability and faster page loading after the first visit



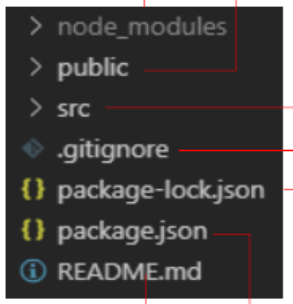
It is the header icon

It is the web template which is considered as an entry point for the web application

manifest.json file contains configurations when your application is used for Android app

REACT

Folder Structure In React



This folder contains all dependencies and sub-dependencies specified in ***package.json and*** used by React application

It contains files which don't require additional processing by ***webpack***

It contains the necessary components and application code

It is a standard file used by source control tool ***git*** to identify files and folders to be ignored while committing the code. Until and unless this file exists, the ***create-react-app command*** will not create a git repo in this folder

It maintains a ***history of changes*** in package.json, so that at any point of time, previous changes can be looked back in the ***package-lock.json file***

Contains ***dependencies and scripts*** required for building the application

Provides ***information*** about the application

REACT

package.json

A package.json in Node.js contains all the files you need for a module, where modules are JavaScript libraries you can include in your project.
package.json is used to store the metadata associated with the project as well as to store the list of dependency packages.

The metadata can be categorised into two:

Identifying metadata: It consist of the properties to ***identify the project*** such as the name, current version of the module, license, author of the project, description about the project, and more

Functional metadata: It consist of the functional properties of the project such as the entry point of the module, dependencies in project, scripts being used, repository links of project, and more

Limitations of ReactJS

- Most of the code is written in JSX, i.e., Html and css are part of javascript, it can be quite confusing as most other frameworks prefer keeping Html separate from the javascript code.
- The file size of ReactJS is large.

Using ReactJS from CDN

To start working with react, we need to first install reactjs. You can easily get started to use reactjs by using the CDN javascript files, as shown below.

Using ReactJS from CDN

CDN Links

Both React and ReactDOM are available over a CDN.

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

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

The versions above are only meant for development, and are not suitable for production. Minified and optimized production versions of React are available at:

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

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

Using ReactJS from CDN

For dev

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

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

For prod:

<script crossorigin src="https://unpkg.com/react@version/umd/react.production.min.js"></script>

<script crossorigin src="https://unpkg.com/react-dom@version/umd/react-dom.production.min.js"></script>

Using ReactJS from CDN : Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World</title>
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
    <script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
  </head>
  <body>
    <div id="app"></div>
    <script type="text/babel">
      ReactDOM.render(
        <h1>Hello, React Example without Structure!</h1>,
        document.getElementById('app')
      );
   </script>
  </body>
</html>
```

Using REACT from CDN : Example

DEMO

Configure REACT Project Using NPM

Steps :

- ✓ Download NodeJS
- ✓ Install NodeJS
- ✓ Check installed nodejs : **node –v**
- ✓ Install the package **create-react-app** globally using the command : **npm i -g create-react-app**
- ✓ Create your application using the command:

  create-react-app <name Of The Application>

  Note : start project name with small letter.
- ✓ Goto inside the Project **cd/projectname**
- ✓ **Run the Project : npm stat**
- ✓ **Congratulation You Get the Welcome React Screen on the urr : http://localhost:3000**

Using NPM Packages

- Make sure you have nodejs installed.
- Visit npmjs.com to know about the packages
- Once you have nodejs installed, create a folder reactproj/.
- To start with project setup, run command **npm init**.
- This is how the folder structure will look like:

Reactproj

package.json

Using NPM Packages

What is npm

- ✓ npm stands for node package manager, it opens up the entire world of JavaScript to individual JavaScript developers as well as to teams.
- ✓ It is currently the world's largest software registry, having approximately 3 billion downloads per week.
- ✓ If you have used any of the popular JavaScript frameworks, chances are that you accessed them using npm.

There are three distinct npm components:

1. The website
2. The command line interface (CLI)
3. The registry

Using NPM Packages

- ✓ Adapt packages of code to your own apps, or you can incorporate packages as they are.
- ✓ Download standalone tools that you can use right away.
- ✓ Run packages without downloading with npx.
- ✓ Share your code with any npm user, anywhere.
- ✓ Restrict your code to specific developers.
- ✓ Form virtual teams by with npm organization.
- ✓ Manage multiple versions of your code and your code dependencies.
- ✓ Update applications easily when underlying code has been updated.
- ✓ Find other developers that are working on similar problems and projects

Using NPM Packages

Here are the list of packages for reactjs:

- npm install react --save-dev
- Npm install –g create-react-app //install react
- npm install react-dom --save-dev
- npm install react-scripts --save-dev

Using NPM Packages

- **Webpack :**
  - A module bundler. Its main purpose is to bundle JavaScript files for usage in a browser
- **Webpack dev server**
  - Used along with webpack. It provides a development server that provides live reloading for the client side code.
  - This should be used for development only.

Using NPM Packages

NPM

- NPX comes bundled with NPM version 5.2+
- NPM by itself does not simply run any package
- If you want to run a package using NPM, you must specify that package in your package.json file.
- When executables are installed via NPM packages, NPM links to them:
- local installs have "links" created at ./node_modules/.bin/ directory.
- global installs have "links" created from the global bin/ directory (e.g. /usr/local/bin) on Linux or at %AppData%/npm on Windows.

NPX

- Npx is npm package runner.
- Let's say that you want to run some package and you run it with npx, then you would type something like this: npx some-package .
- Npx will search for that package in the local and global registry.
- When npx finds the package it will run that package from there.
- But let's say that you don't have that package installed locally or globaly.
- In that case npx will download the package files and install the package. That package will be run from the local npx cache.

Yarn

- Npm has some flaws so Facebook developers decided to build a new package manager that would represent an alternative.
- Yarn is package manager like npm,
- Yarn is faster than npm because when installing multiple packages npm installs them one at the time while yarn is installing them concurrently.

Using NPM Packages

Open the command prompt and run above commands inside the folder reactproj/.

Create a folder src/ where all the js code will come in that folder. All the code for reactjs project will be available in the src/ folder. Create a file index.js and add import react and react-dom, as shown below.

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';

ReactDOM.render(
<h1>Hello, ReactJS Example!</h1>,
    document.getElementById('root')
);
```

Using NPM Packages

Open the command prompt and run above commands inside the folder reactproj/.

Create a folder src/ where all the js code will come in that folder. All the code for reactjs project will be available in the src/ folder. Create a file index.js and add import react and react-dom, as shown below.

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';

ReactDOM.render(
<h1>Hello, ReactJS Example!</h1>,
    document.getElementById('root')
);
```

First React Project Setup

Step1

1. To start with ReactJS, we need to first import the react packages as follows.

import React from 'react';

import ReactDOM from 'react-dom';

Save the file as index.js in src/ folder

First React Project Setup

Step 2

```
ReactDOM.render(
<h1>Hello, Steps 2!</h1>,
    document.getElementById('root')
);
```

ReactDOM.render will add the <h1> tag to the element with id root. Here is the html file we are having:

```
<!DOCTYPE html>
<html>
  <head>
<title>ReactJS Demo</title>
  </head>
  <body>
    <div id = "root"></div>
  </body>
</html>
```

First React Project Setup

reactproj/

node_modules/

src/

index.js

package.json

public/

index.html

We have added the commands to compile the final file in package.json as follows:

```
"scripts": {
    "start": "react-scripts start"
  },
```

To compile the final file run following command:

npm run start

First React Project Setup

When you run above command, it will compile the files and notify you if any error, if all looks good, it will open the browser and the run the index.html file at

http://localhost:3000/index.html

Command: **npm run start:**

C:\reactproj>npm run start

> reactproj@1.0.0 start C:\reactproj

- react-scripts start

Very Tedious !!!

React Project Setup : Latest Trends

- ✓ **npx create-react-app Welcome-React**
- ✓ npx create-react-app **Welcome-React**
- ✓ cd **Welcome-React**
- ✓ **npm start**
- ✓ hen open http://localhost:3000/ to see your app.
- ✓ When you're ready to deploy to production, create a minified bundle with **npm run build.**

- ➢ Google Search : create react app
- ➢ **npm init react-app my-app**
- ➢ *npx comes with npm 5.2+ and higher*
- ➢ *npx create-react-app my-app --template typescript*

React : YARN

What is yarn.

Yarn is a package manage you can use for your code. it enables you to use and share your code with other developers around the world. Yarn will do this quickly, securely and reliably such that you don't ever have to worry.

Why yarn

Yarn enables you to use the solutions created by other developers. It makes it easier for you to develop your software.

In the event that you encounter a problem, you can report the issues or contribute back, and then you can use yarn to keep it up to date when the problem is fixed.

Your codes will be shard through something that is called a package Typically, a package will contain the code that is being shared as well as a package.json file which will describe the package.

React Project Setup : Using YARN

- npm install --global yarn
- Yarn -v
- yarn create react-app appname
- yarn start

React Project Setup : Latest Trends

npm start

Starts the development server.

npm run build

Bundles the app into static files for production.

npm test

Starts the test runner.

React Project Setup : Latest Trends

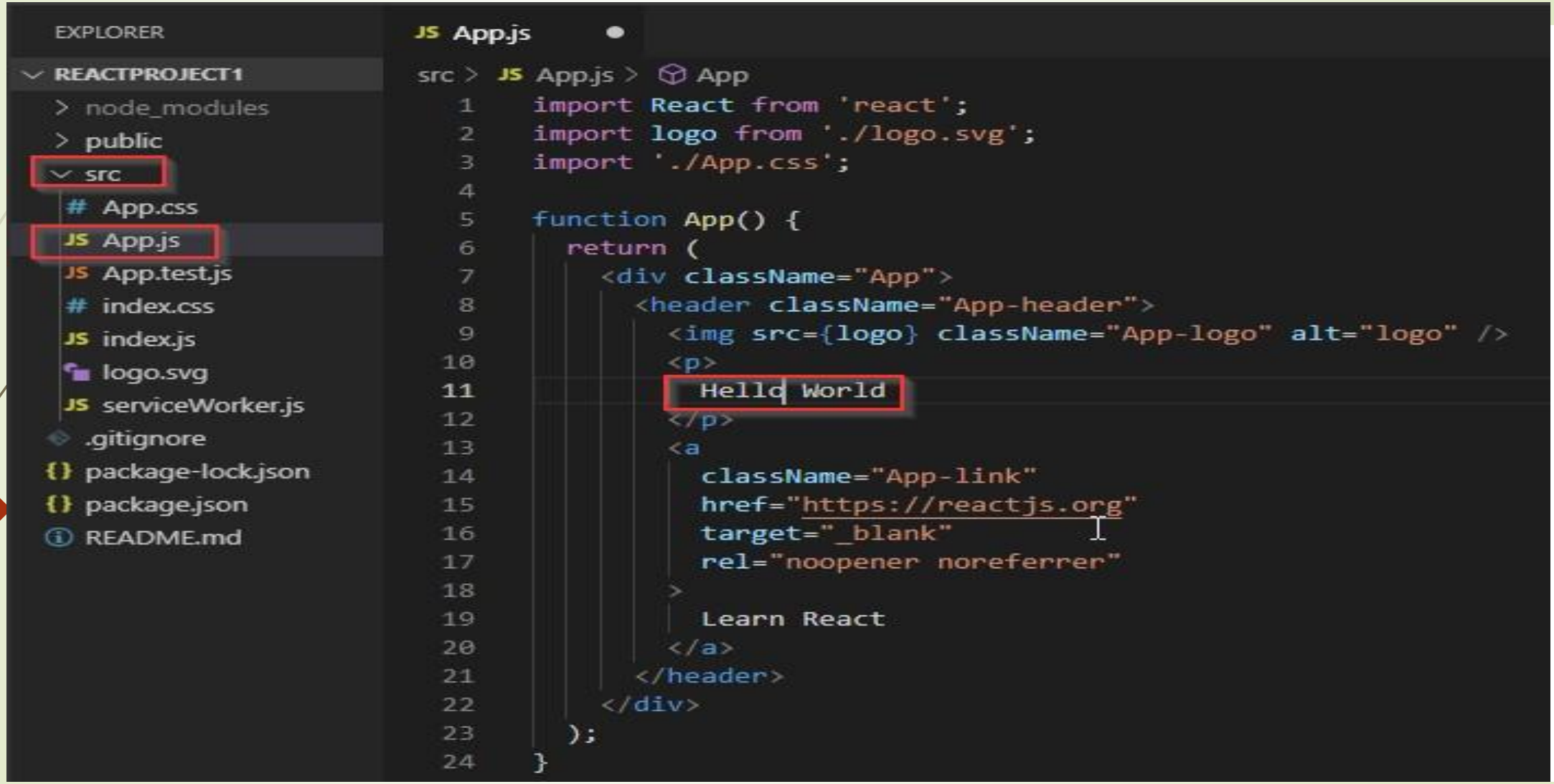
DEMO!!!

React : Assignment

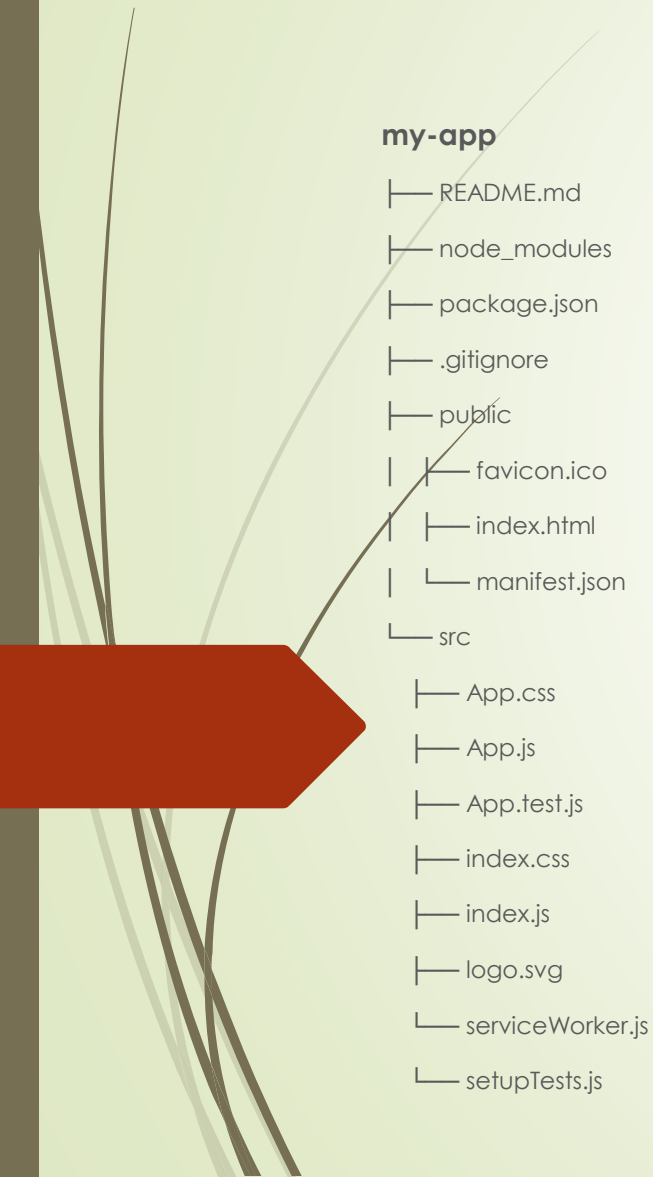
Create a React application by following the below

1. Install Nodejs and create the basic seed of React
2. Setup the Visual studio code and setup the React application in VS code
3. Create the Home Page of the application
4. Run the application in the browse

React : Hello World



React : Directory Structure



```
my-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── serviceWorker.js
    └── setupTests.js
```

React App: Environment

Your environment will have everything you need to build a modern single-page React app:

- React, JSX, ES6, Typescript and Flow syntax support.
- Language extras beyond ES6 like the object spread operator.
- Auto prefixed CSS, so you don't need -weskit- or other prefixes.
- A fast interactive unit test runner with built-in support for coverage reporting.
- A live development server that warns about common mistakes.
- A build script to bundle JS, CSS, and images for production, with hashes and source maps.
- An offline-first service worker and a web app manifest, meeting all the Progressive Web App criteria.
- Hassle-free updates for the above tools with a single dependency.

React App: Examples

React App: 1st App

Home.js

```
import React from 'react';
export default class Home extends React.Component{
render()
{
    return (
        <div>
        <h1> Home Componenet</h1>
        <h3> Creating Component in REACT</h3>
        </div>
    )
}
}
```

Note : Component is based on SRP (Single Responsible Principle)

React App: 1st App

Home.js

```
import React from 'react';
export default class Home extends React.Component{
render()
{
    return (
        <div>
        <h1> Home Componenet</h1>
        <h3> Creating Component in REACT</h3>
        </div>
    )
}
}
```

Home.js bind with App.js

```
import './App.css';
import Home from './Home';
export default function App() {
  return (
    <div className="App">
      <header className="App-header">
        <Home> </Home>
        </header>
    </div>
  );
}
```

Note : Always create filename with the class name , Import the component inside the App.js & include the component as an element with the App.js

REACT : Component Class

Registration.js

```
import React from 'react';
class Registration extends React.Component{
render()
{
   return (
      <div>
      <h1> Course Registration</h1>
      <h3> Creating Class Component in REACT</h3>
      </div>
   )
}
}
export default Registration
```

REACT : Functional Class

Contact.js

```
import React from 'react';
export default function Contact()
{
return(
<div>
   <h1> Using Functional Componeent</h1>
</div>
)
}
//export default Contact;
```

REACT : Using Events

React Events

Similar to HTML, React ***executes actions*** based on ***user events***, these events mainly include: ***click, change, mouseover*** and many more

- React events are written in camelCase and event handlers are written inside curly braces-
  Example: ***<button onClick={this.click}>click here</button>***
- Using ***JSX*** you pass a function as the event handler in place of a string
- It is a good practice to always put the event handler as a ***method*** in the ***component class***

REACT : Using Events

React Events: Example

```
import React, {Component} from 'react';
import ReactDOM from 'react-dom';

class Event extends Component {
  click() {
    alert("Good One");
  }
  render() {
    return (
      <button onClick={this.click}>click here</button>
    );
  }
}

ReactDOM.render(<Event />, document.getElementById('root'));
```

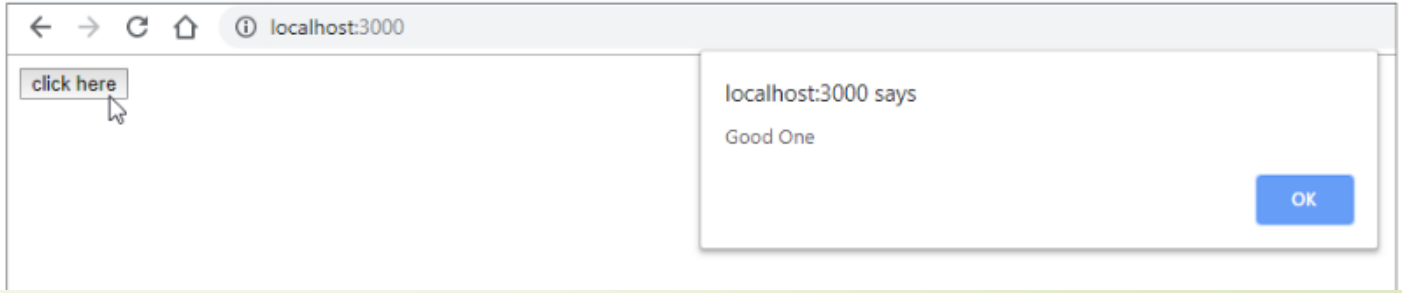
Event component

Event handler

Event

Text on button

Output:



REACT : Using Events

Contact.js

```
import React from 'react';
export default function Contact()
{
   function jfun1()
   {
      alert("Event Fired!")
   }
   //USING ARROW FUNCTION
   /*
   const jfun1=()=>
   {
      alert("Event Fired!")
   }
  */
   return(
      <h1 onClick={jfun1}> Using Event with React </h1>
      )
   }
```

REACT : Using Multiples Functions

Multifcomponent.js

```
import React from 'react';
function Multifcomponent() {
  function greeting() {
   alert("Message From Greeting() Function!")
  }
  function waveHello() {
    //console.log('Testing');
    alert("Message From waveHello() Function!")
  }
  return (
    <div>
```

REACT : Functional Class

Contact.js

```
import React from 'react';
export default function Contact()
{
return(
<div>
   <h1> Using Functional Componeent</h1>
</div>
)
}
//export default Contact;
```

REACT : Using Multiples Functions

Multifcomponent.js

```
import React from 'react';
function Multifcomponent() {
  function greeting() {
   alert("Message From Greeting() Function!")
  }
  function waveHello() {
    //console.log('Testing');
    alert("Message From waveHello() Function!")
  }
  return (
    <div>
```

REACT : Using Multiple Functions

```
<button
      onClick={() => {
        greeting();
        waveHello();
      }}>
      I'm a button
    </button>
  </div>
 );
}
export default Multifcomponent;
```

REACT : Using Multiple Functions

```
import React from 'react';
export default function Contact()
{
    function jfun1()
    {
        alert("Event Fired!")
    }
    //USING ARROW FUNCTION
    /*
    const jfun1=()=>
    {
        alert("Event Fired!")
    }
   */
    return(
        <h1 onClick={jfun1}> Using Event with React </h1>
        )
    }
```

REACT : Assignment

Create a “Todo List” Application using React Concepts

Here you should create an application where any user can list his daily tasks to be completed.

1. Take new task as input through input box
2. Add a submit button, on clicking it the task should be displayed on the screen
3. Add a check box before every task
4. On clicking the check box, the ‘check tick’ symbol should be displayed

React Hooks: an improved state management

Note : Using hook we can make functional component stateful. A **Hook** is a special function that lets you "**hook** into" **React** features. For example, useState is a **Hook** that lets you add **React** state to function components

- State is the getter & setter inside the REACT.

- React introduced its Hooks API in its 16.8 release. The greatest Hook's asset is that it allows sharing state logic between components without re-architecting the whole code block.
- Along with that, Hooks permits the reuse of logic between components without changing its structure.
- It also enables separating a specific single component into several smaller functions, based on the roles these pieces refer to, instead of breaking them according to their lifecycle methods.
- What's equally important for developers, Hooks saves them from writing classes for each component and digging deeply into the JavaScript.

React Hooks: an improved state management

```
import React, { useState } from 'react';
function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Note : This new function useState is the first “Hook”

React Hooks: an improved state management

Why ref is used in react?

Refs provide a way to access DOM nodes or **React** elements created in the render method. In the typical **React** dataflow, props are the only way that parent components interact with their children.

JSX

REACT : JSX

JSX

JSX stands for JavaScript XML, which lets you to write the HTML code within the JavaScript.

JSX is an extension to the JavaScript language syntax. It is easy to write JSX templates if you know the HTML

It is type-safe as most of the errors can be caught during compilation

It makes use of ***render function*** to return the single HTML element at a time

React components are typically written using JSX, although they do not have to be (components may also be written in pure JavaScript)

REACT JSX

- It is called **JSX**, and it is a syntax extension to JavaScript. We recommend using it with **React** to describe what the UI should look like. **JSX** may remind you of a template language, but it comes with the full power of JavaScript. **JSX** produces **React** "elements".

- **JSX** allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods. **JSX** converts HTML tags into **react** elements. You are not required to **use JSX**, but **JSX** makes it easier to write **React** applications.

REACT JSX

What is the difference between JS and JSX in react?

- **JS** is standard **javascript**, **JSX** is an HTML-like syntax that you can use with **React** to (theoretically) make it easier and more intuitive to create **React** components.
- Without **JSX**, creating large, nested HTML documents using **JS** syntax would be a large pain **in the** rear; **JSX** simply makes that process easier.
- For those unfamiliar with **React**, **JSX** is an inline markup that looks like HTML and gets transformed to JavaScript. A **JSX** expression starts with an HTML-like open tag, and ends with the corresponding closing tag. **JSX** tags support the XML self close syntax so you can optionally leave the closing tag off

REACT JSX

- In most of the cases it's only a need for the transpiler/bundler, which might not be configured to work with **JSX** files, but with **JS**! So you are forced to **use JS** files instead of **JSX**. And since **react** is just a library for javascript, it makes no difference for you to choose between **JSX** or **JS**

How is JSX different from HTML?

Basically, **JSX** is a JavaScript render function that helps you insert your **HTML** right into your JavaScript code. ... Using Vue templates is like using **JSX** in that they're both created using JavaScript. The main difference is that **JSX** functions are never used in the actual **HTML** file, while Vue templates are.

REACT JSX

Why is JSX faster?

React uses **JSX** for templating instead of regular JavaScript. ... 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 it easier and **faster** to write templates, if you are familiar with HTML.

REACT : JSX

JSX Use Case

Regular JSX

Returns the HTML Representation

```
var MyComponent =
React.createClass({
    render : function () {
        return (
            <div>
            Hello World!!!
        </div>
    );
    });
```

Specifying Attribute

```
var styles={ backgroundcolor: 'cyan'};
var MyComponent=React.createClass({
    render : function () {
        return (
            <div style={styles}>
            <h1>Header</h1>
            </div>
    );
    }
});
```

Adding Attributes

JSX Nested Elements

Returns multiple elements

<h1>,<h2>,<p> nested inside

```
var MyComponent = React.createClass( {
    render : function () {
        return (
            <div>
            <h1>Header</h1>
            <h2>Content</h2>
            <p>This is the content!!!</p>
        </div>
    );
    }
});
```

Embedding JavaScript

```
var MyComponent = React.createClass({
    render: function () {
        return(
            <div>
                <h2> {2+4} </h2>
            </div>
        );
    }
});
```

JavaScript Expression

REACT : DOM

Document Object Model

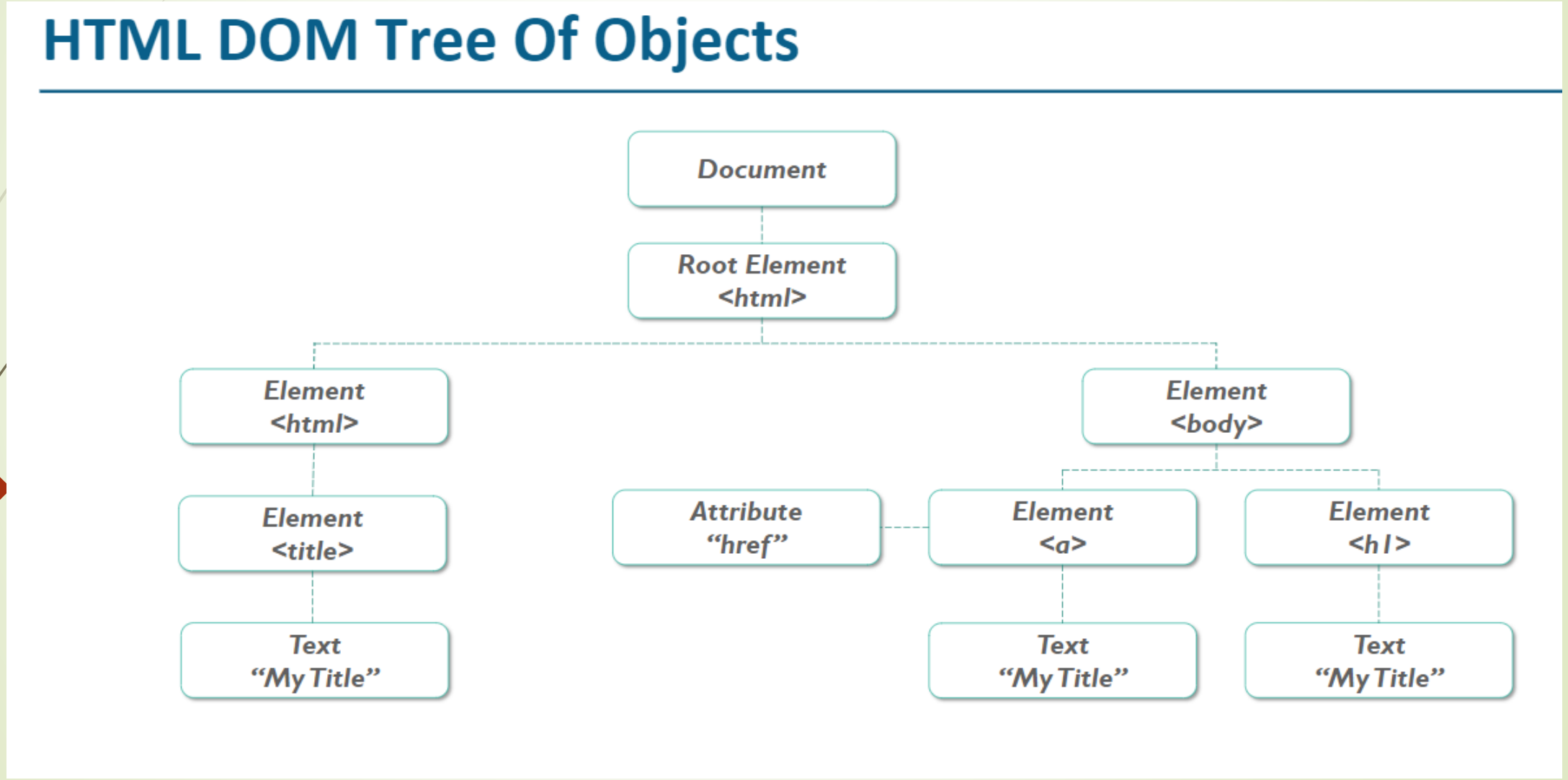
DOM is a standard logical representation of any webpage.

With the Document Object Model, programmers can ***create and build*** documents

DOM is a ***tree-like*** structure that contains all the ***elements*** and its ***properties*** as its ***nodes***

DOM provides a ***language-neutral interface*** that allows accessing and updating of the content of any element of a webpage

REACT : DOM


HTML DOM Tree Of Objects
Document
Root Element
<html>
Element
<html>
Element
<body>
Element
<title>
Attribute
"href"
Element
<a>
Element
<h1>
Text
"My Title"
Text
"My Title"
Text
"My Title"

REACT : DOM

DOM Manipulation: Disadvantage

By now we have already discussed the importance of DOM and established the fact that DOM manipulation is the heart of the modern, interactive web. Unfortunately, it does have some drawbacks

1. Let us assume that you have a list that contains ten items. You are suppose to make some changes in the fifth item. Under such circumstances, most JavaScript frameworks would rebuild *the* ***entire list again***
2. This leads to ***more work*** than necessary. Just to implement the changes of one item, even rest nine get ***rebuilt*** unnecessarily
3. Rebuilding a list is not a big deal to a web browser, but modern websites can use ***huge amounts of DOM manipulation***. To address this problem, the React popularized new aspect called ***virtual DOM***

REACT : DOM

Introduction To Virtual DOM

The virtual DOM (VDOM) is a programming concept, where "virtual" representation of a UI is kept in memory and synced with the "real" DOM by a library such as ***ReactDOM.***

In React, for every DOM object, there is a corresponding "virtual DOM object"

A virtual DOM object is a *representation* of a DOM object, like a lightweight copy

A virtual DOM object has the same properties as a real DOM object, but it lacks the power to directly change what's on the screen

Manipulating the virtual DOM is much faster, because nothing gets drawn onscreen

REACT : DOM

Features Of Virtual DOM

1. When you render a JSX element, every single virtual DOM object gets updated.
   This sounds incredibly inefficient, but the cost is insignificant because the virtual DOM updates quickly

1. **Element Rendering**

2. Once the virtual DOM is updated, React compares the virtual DOM with a virtual DOM *snapshot* that was taken right before the update

1. **Element Rendering**

2. **DOM Comparison**

3. By comparing the new virtual DOM with real DOM, React figures out *exactly which virtual DOM objects have changed*. This process is called "***diffing***." Once React knows which virtual DOM objects have changed, then React updates only those objects on the real DOM

REACT

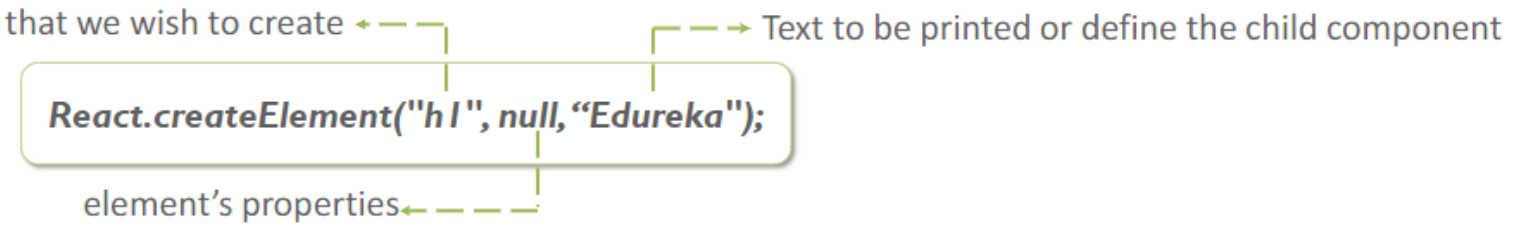
React Element

React element is an object *describing* DOM node and its desired properties.

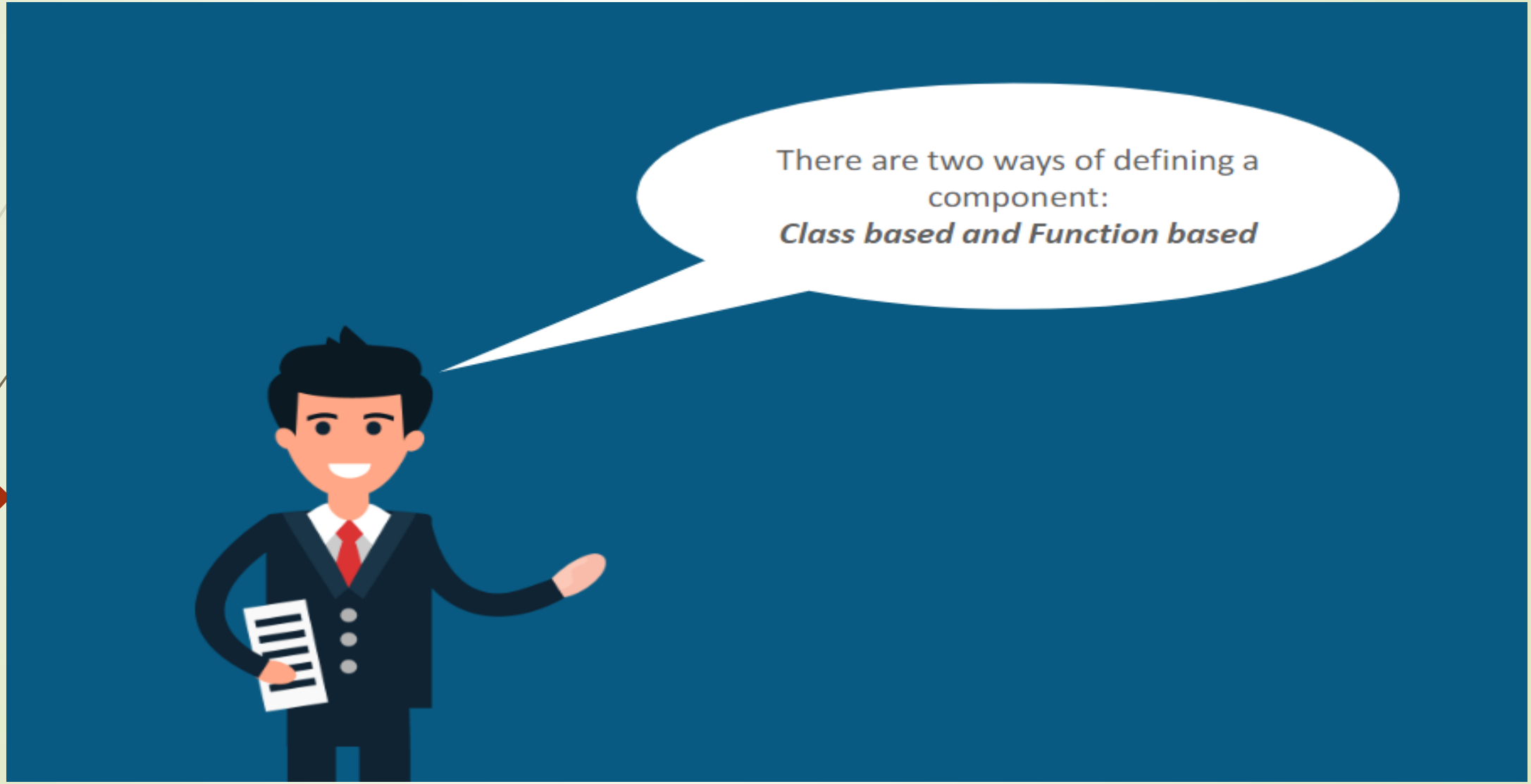
It contains information about the component type (for example, a Button), its properties (for example, its colour), and any child component inside it

React elements and DOM elements are not the same. React elements are converted to DOM elements using Render function

Given below is the ***Syntax*** of writing React Elements, here we create a React element to represent ***h1 DOM*** element using ***React.createElement*** function



REACT

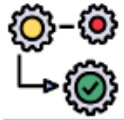

There are two ways of defining a component:
Class based and Function based

REACT

Components

Components are independent and reusable bits of code, which returns React elements that describes how a section of the UI (User Interface) should appear.

Every part of a React application is a ***component***, which ***splits*** the ***UI*** into independent reusable sections



Each independent section is ***processed separately***

We can ***easily update or change*** any component of an application without ***disturbing*** rest of the application

Components must be written in ***upper case*** to avoid ambiguity with HTML tags

Note: ***Render*** function is used only by the first component and rest make use of ***export***

REACT


There are two ways of defining a component:
Class based and Function based

REACT

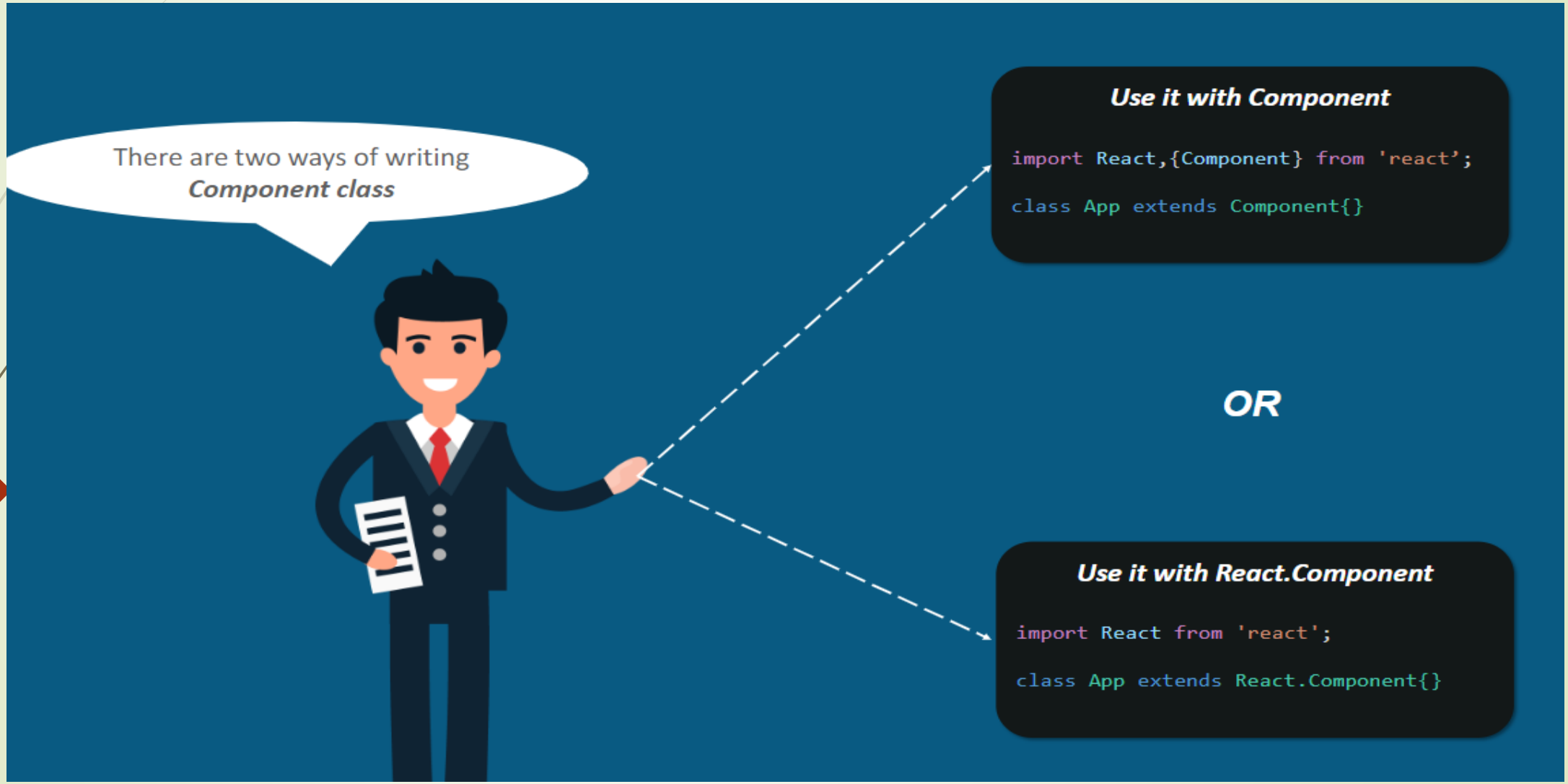
Class Component

A ***Class Component*** is defined using a *class*. It is written with '***extends React.Component'*** statement, this statement inherits ***React.Component***, and gives your component access to React.Component's functionalities.

```
import React from 'react';
import ReactDOM from 'react-dom';

class App extends React.Component {
    render() {
      return <h1>Welcome To Edureka</h1>;
    }
  }

ReactDOM.render(<App/>, document.getElementById('root'));
```

- Required packages to run the application code
- Component
- Function called to provide HTML content
- HTML content to be displayed
- Component to be rendered
- HTML element

REACT


There are two ways of writing
Component class
Use it with Component
import React,{Component} from 'react';
class App extends Component{}
OR
Use it with React.Component
import React from 'react';
class App extends React.Component{}

REACT

Use Of Constructor Within The Components

01 A ***constructor*** is a member function of a class which initializes objects of a class. It has ***same name*** as the class itself

02 It is ***called automatically called*** during the creation of an object from a class

03 When implementing the constructor for a ***React.Component***, you should call ***super(props)*** before any other statement. Otherwise, ***this.props*** will be undefined to the constructor, which can lead to bugs

04 ***super()*** method is used to ***call*** the constructor of the ***parent class***

05 If your application code ***does not contain state or props*** within the component or it is not binding any ***event handlers***, then there is ***no need to define components with constructors***

REACT

Example: constructor()

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';

class App extends Component {
  constructor(){
    super();
    this.state = {subject: "React with Redux Certification Training"}
  }
    render() {
      return <h2>Welcome to {this.state.subject}</h2>;
    }
  }

ReactDOM.render(<App/>, document.getElementById('root'));
```

- Lets App Component to receive all the functionalities provided by parent class Component
- Takes props as input parameter
- Informs the parent class to initiate the work
- Props

REACT

Functional Component

Functional components are usual JavaScript functions, which takes in ***props*** and returns ***React Element***.

Example

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';

const App = () => {
      return(
      <h1>We are learning React</h1>
      )
 }

ReactDOM.render(<App/>, document.getElementById('root'));
```

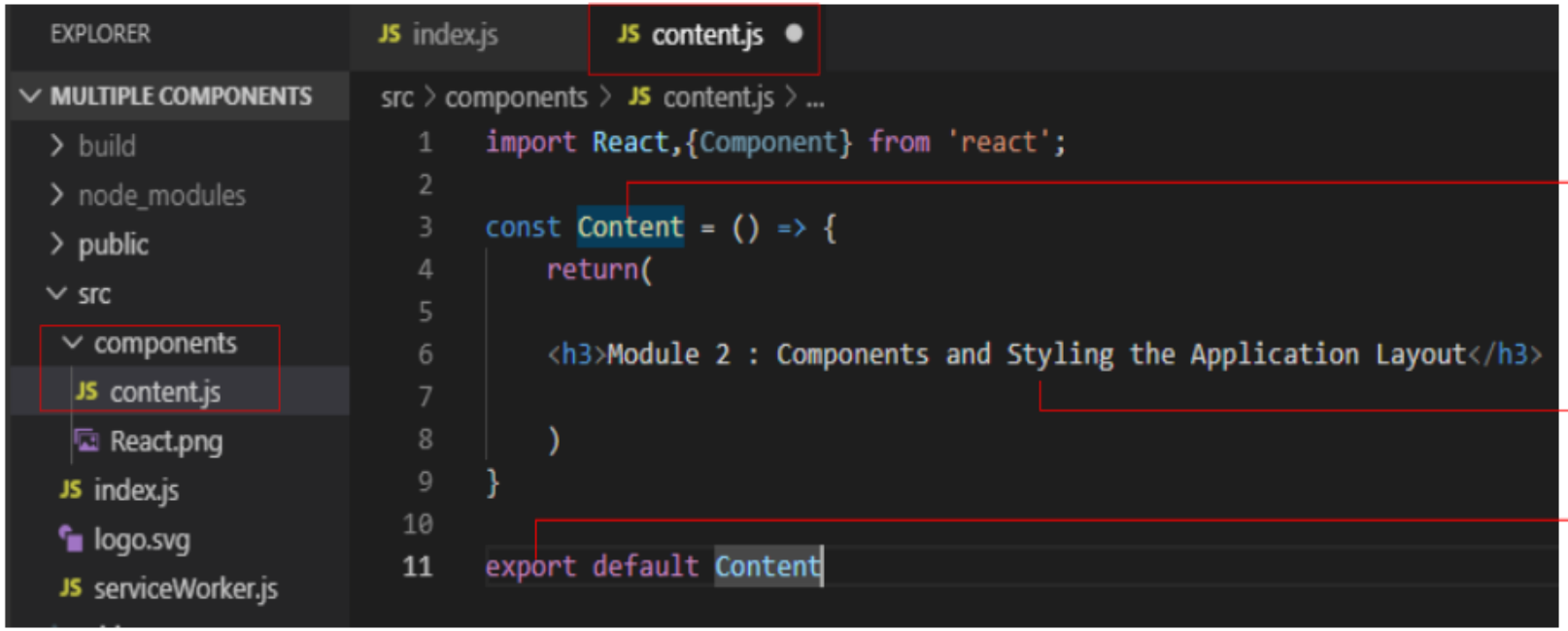
Functional component

HTML data to be displayed

REACT

How To Define Multiple Components

It is a good way to maintain a folder called ***components***, to add multiple components list.



Functional component

Content to be displayed

Used to ***connect*** the Content component to the main component

REACT

How To Define Multiple Components (Contd.)

Open ***Index.js*** file and add the path of *Content* component.

```
JS index.js   JS content.js
src > JS index.js > ...
1   import React,{Component} from 'react';
2   import ReactDOM from 'react-dom';
3   import Content from './components/content';
4
5   const App = () => {
6         return (
7             <div>
8             <h1>We are learning React</h1>
9             <Content/>
10            </div>
11        )
12    }
13  ReactDOM.render(<App/>, document.getElementById('root'));
```

Path of Content component (Child component)

Parent component

Displays data present in Content component (Child component)

Output:

REACT

Functional Component Vs Class Component

Functional Component	*Class Component*
Functional components are ***simple*** to read, understand and are written in ***few lines of code***	Class components offer ***more features***, this makes the code a little bulky than Functional components
They can ***access props***, but they ***lack*** state and life cycle hence used as ***presentational components***	They are used as ***container components***, as they access props, handle state management and lifecycle
Due to lack of states, functional components are ***stateless***	Class components are ***stateful*** and make use of constructors to initialize state
They do not need ***'this'*** keyword to access props	They make use of ***'this'*** keyword to access *props*

REACT

REACT

Props

Props are the arguments passed to the React components.

Props are usually passed via ***HTML attributes*** to components, they are used by both class and functional components

They are used to render the data ***from parent component to child component***. Hence, flow of data in react is ***unidirectional***

Props are ***immutable***, that is their value cannot be changed

Syntax - passing Props:

<ReactComponent demoProp = "Hello" />

A prop named ***demoProp*** is passed to the component named ***ReactComponent*** with a value ***Hello***

REACT

Ways Of Writing Props

There are two ways to write Props:

Props with Class Based Component

- We can access props from the component's class using: ***this.props.propName***
- '***this.props***' is a global object which stores all props of a component

Props with Function Based Component

- To access a prop from a function we do not need to use the '***this***' keyword anymore
- Functional components accept props as ***parameters*** and can be ***accessed directly***

REACT

Example: Props With Class Based Component

```
import React,{Component} from 'react';
import Child from './components/child';
import ReactDOM from 'react-dom';

class Parent extends Component {
render() {

return (
 <div>
 <Child dataFromParent = "Passing the data using props"/>
 </div>

);} }
ReactDOM.render(<Parent/>, document.getElementById('root'));
```

Class based ***Parent*** Component

Props

Output


```
import React,{Component} from 'react';

class Child extends Component {
render() {

return (
     <div>
     <h1> We are learning :{this.props.dataFromParent}</h1
     </div>
);}}
export default Child
```

Class based ***Child*** Component

Accessing Props in Child Component

REACT

Example: Props With Function Based Component

```
import React,{Component} from 'react';
import Child from './components/child';
import ReactDOM from 'react-dom';

const Parent = () => {
return (
<div>
<Child dataFromParent = "Props with function based component"/>;
</div>
);
}
ReactDOM.render(<Parent/>, document.getElementById('root'));
```

Function based ***Parent*** Component

Props

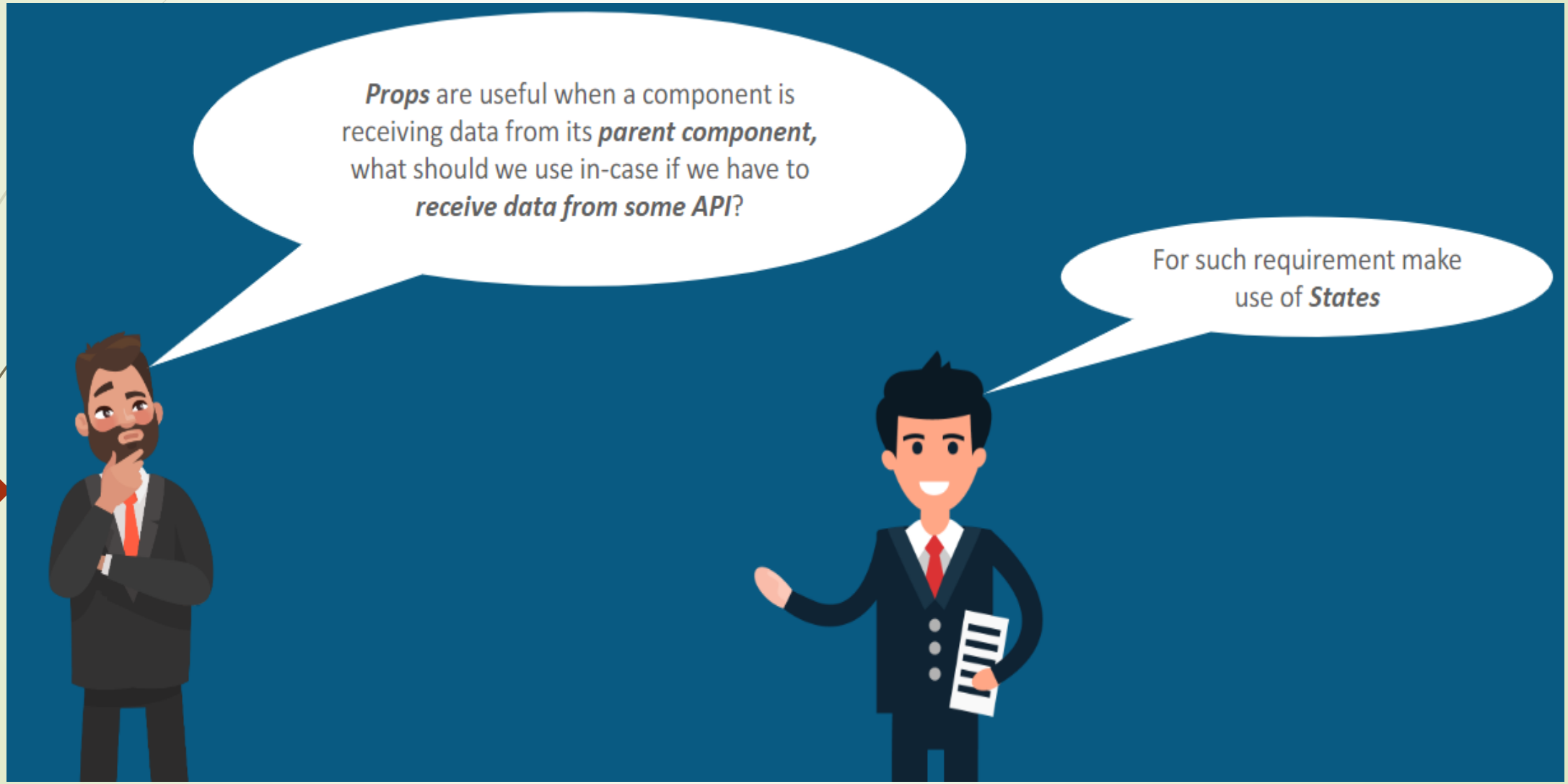
Output


```
import React,{Component} from 'react';

const Child = () => {

return (
<div>
<h1> We are learning :{props.dataFromParent}</h1>
</div>
);
}
export default Child
```

Function based ***Child*** Component

Accessing Props in Child Component

REACT


Props are useful when a component is receiving data from its ***parent component,*** what should we use in-case if we have to ***receive data from some API***?
For such requirement make use of ***States***

REACT STATES

States

React uses an observable object called ***state,*** to observe the changes made to the component and guide the component to behave accordingly.

States are ***variables*** declared within the class component which holds some information that may change over the lifetime of the component

They are ***mutable***, as they hold the data that change over time and controls the behaviour of the component after each change

They are generally updated by ***event handlers*** and are ***modified*** using ***setState()*** method

We can define state in any class as below:

```
Class Sample extends React.Component
{
   constructor()
     {
       super();
       this.state = { attribute : "value" };
     }
}
```

REACT STATES

Demo: Working Of States

Demo Steps

- In this demo, you will learn how to change the displayed text using state method
- Create a component called ***Text*** and add its path to the main component
- Later add the below snippet and execute the code

src > components > JS text.js > ...

```
1   import React,{Component} from 'react';
2   import ReactDOM from 'react-dom';
3
4   class Text extends Component{
5       constructor(){
6           super()
7           this.state = {
8               text: 'Welcome students'
9           }}
10
11          changeText() {
12          this.setState({
13              text:'This is Class 2 of React'
14          })
15      }
16      render(){
17          return(
18
19                      <div>
20                      <h1>{this.state.text}</h1>
21                      <button onClick={() => this.changeText()}>Next</button>
22                      </div>
23                  );
24              }
25  }
26  export default Text
```

Class Component

An object holding the data

Props

Method called to update current state

New text to be printed on click of button

Props

Accessing the state

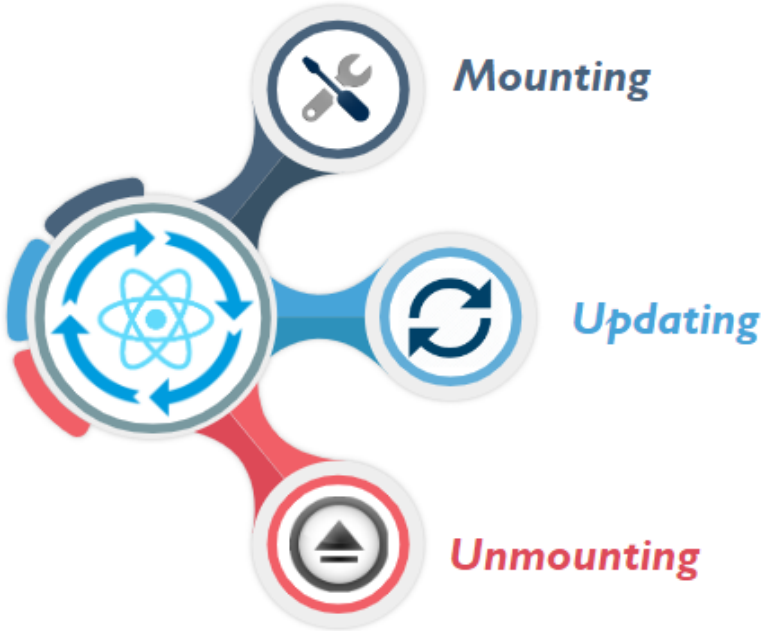
Handler

Event

REACT COMPONENT LIFE CYCLE

React Component Lifecycle

Every ***React Component follows a lifecycle***, where a series of methods are invoked in different stages.
These stages are as mentioned below:

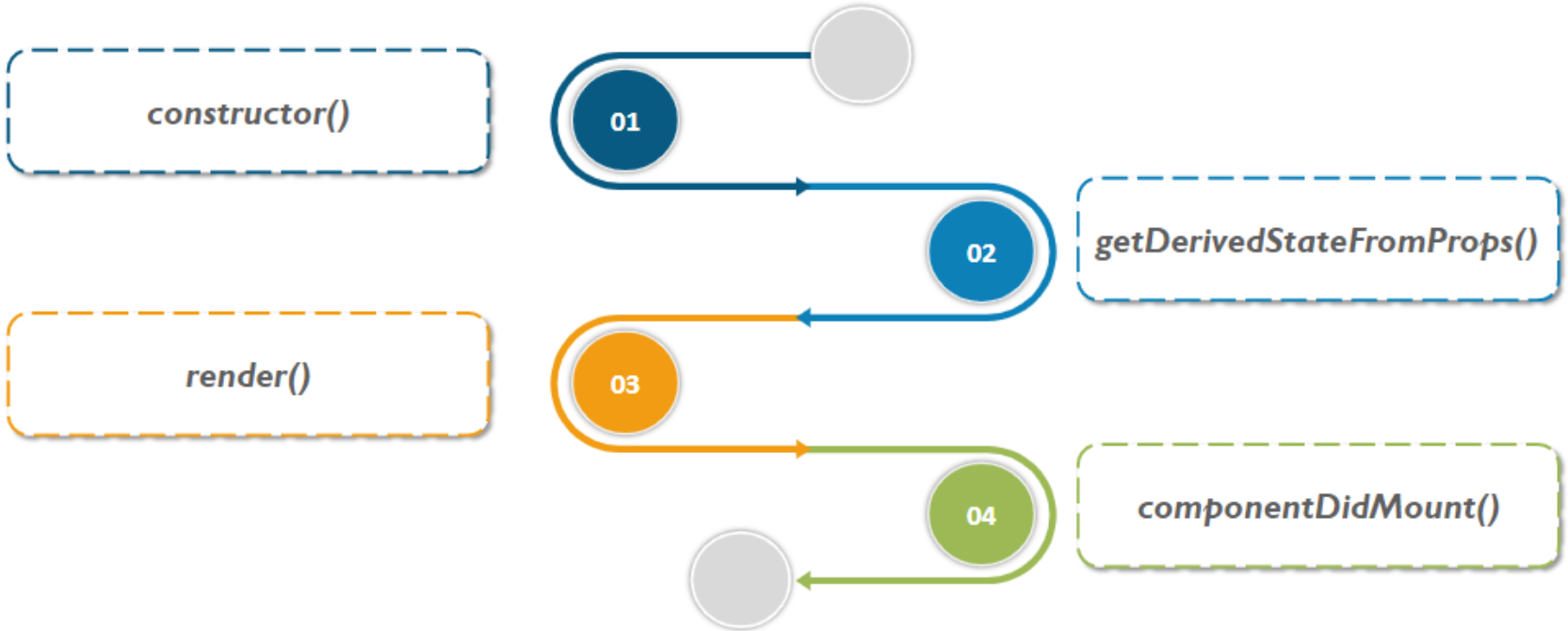


Component Lifecycle also know as Lifecycle-Hook is required when you want to control the flow of your application code.

REACT COMPONENT LIFE CYCLE

Mounting

Mounting is the phase where elements are ***added*** to DOM. During Mounting phase, four in-built methods are called simultaneously-



REACT COMPONENT LIFE CYCLE

Mounting: constructor

When a component is initiated a ***constructor*** is called to set up the props and states within the component.

- *constructor()*
- *getDerivedStateFromProps()*
- *Render()*
- *componentDidMount()*

Example

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';

class Music extends Component {
  constructor(props) {
    super(props);
    this.state = {instrument: "Guitar"};
  }
  render() {
   return (
   <h1>I know how to play {this.state.instrument}</h1>
   );
  }
}
ReactDOM.render(<Music />, document.getElementById('root'))
;
```

Output:

REACT COMPONENT LIFE CYCLE

Mounting: getDerivedStateFromProps()

This method is called before sending the element to the DOM. It takes props and returns an object along with changes to the state. It is useful in cases where it is vital to have the previous and new value for comparison

- constructor()
- getDerivedStateFromProps()
- Render()
- componentDidMount()

Example

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';

class Music extends Component {
    constructor(props) {
      super(props);
      this.state = {instrument: "Guitar"};
    }
    static getDerivedStateFromProps(props, state) {
      return {instrument: props.New };
    }
    render() {
      return (
        <h1>I know how to play {this.state.instrument}</h1>
      );
    }
  }
  ReactDOM.render(<Music New="Drums"/>, document.getElement
ById('root'));
```

Output:

Above example starts with the instrument Guitar, but when the getDerivedStateFromProps() method is called, it updates the instrument based on the passed props “New”

REACT COMPONENT LIFE CYCLE

Mounting: render()

Render() method is required to transform React Components into the DOM

- *constructor()*
- *getDerivedStateFromProps()*
- *Render()*
- *componentDidMount()*

Example

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';

class Music extends Component {
  constructor(props) {
    super(props);
    this.state = {instrument: "Guitar"};
  }
  render() {
    return (
      <h1>I know how to play {this.state.instrument}</h1>
    );
  }
}
ReactDOM.render(<Music />, document.getElementById('root')
);
```

Output

REACT COMPONENT LIFE CYCLE

Mounting: componentDidMount()

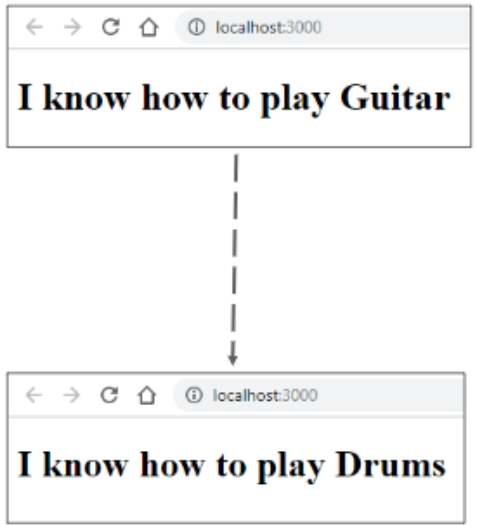
componentDidMount() method is called when component is rendered to DOM. It confirms whether the component is placed in DOM.

- *constructor()*
- *getDerivedStateFromProps()*
- *Render()*
- *componentDidMount()*

Example

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';

class Music extends Component {
    constructor(props) {
      super(props);
      this.state = {instrument: "Guitar"};
    }
    componentDidMount() {
      setTimeout(() => {
        this.setState({instrument: "Drums"})
      }, 1000)
    }
    render() {
      return (
        <h1>I know how to play {this.state.instrument}</h1>
      );
    }
  }

  ReactDOM.render(<Music />, document.getElementById('root'));
```

Output

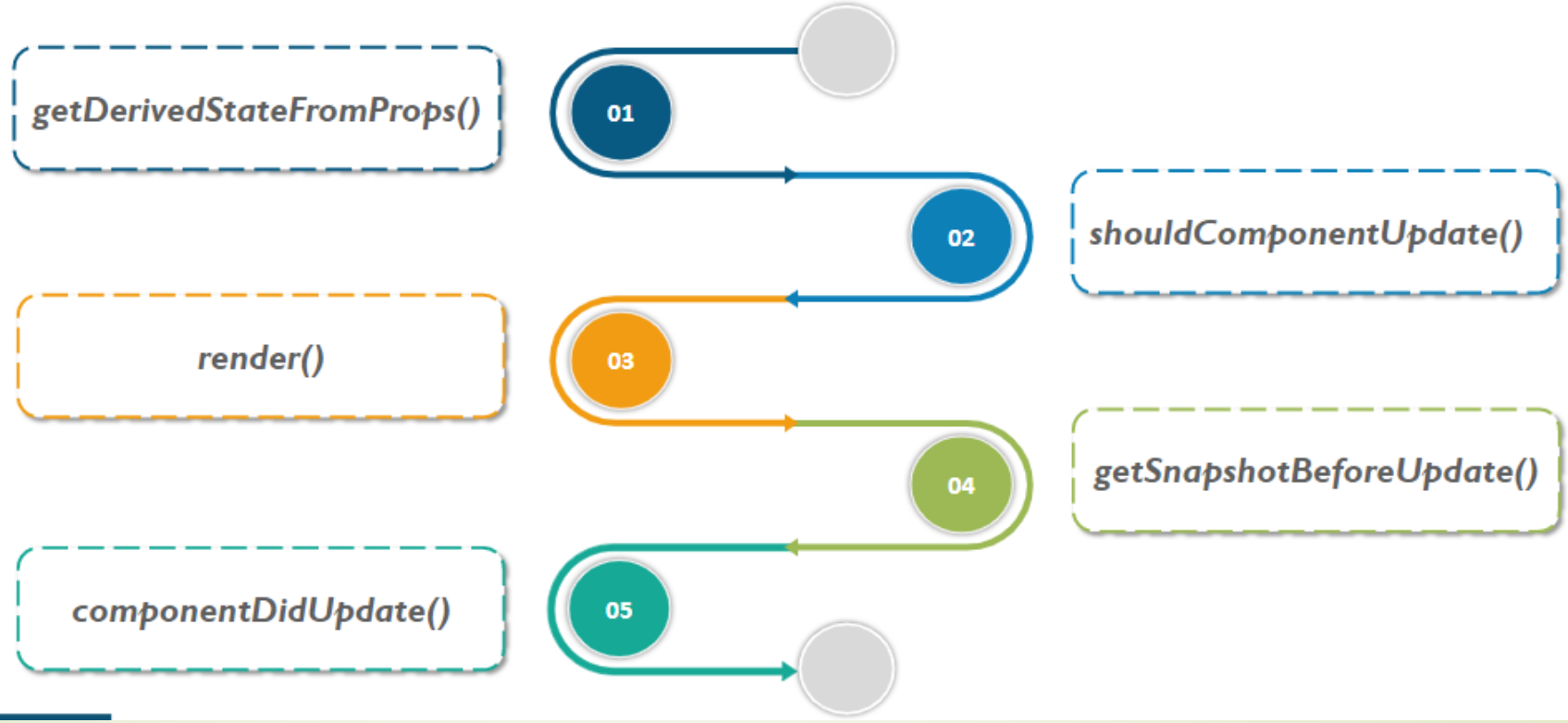


REACT COMPONENT LIFE CYCLE

Updating

Updating is the phase where the states and props of a component are updated due to some user events such as clicking or pressing any key on keyboard.

During Updating phase below in-built methods are called in order:



REACT COMPONENT LIFE CYCLE

Unmounting

Unmounting is the phase where component is supposed to be removed from the DOM.

01 ***componentWillUnmount()*** is the only method used to remove component from the DOM

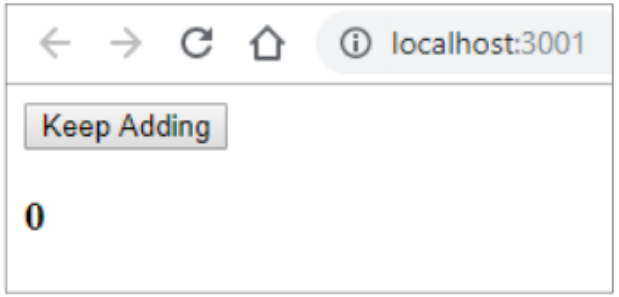
02 ***ReactDOM.render(<Component />, container)*** is the usual method of adding components to DOM, to unmount that Component from the container clean up all the attached event handlers and state

03 For unmounting the component you can make a call to {***React.unmountComponentAtNode(container_name)***}

Example:

```
setTimeout(() => {
    ReactDOM.unmountComponentAtNode(document.getElementById('root'));}, 10000);
```

Here every time on clicking the button the component is incremented by 1, after 10000 sec the component is unmounted

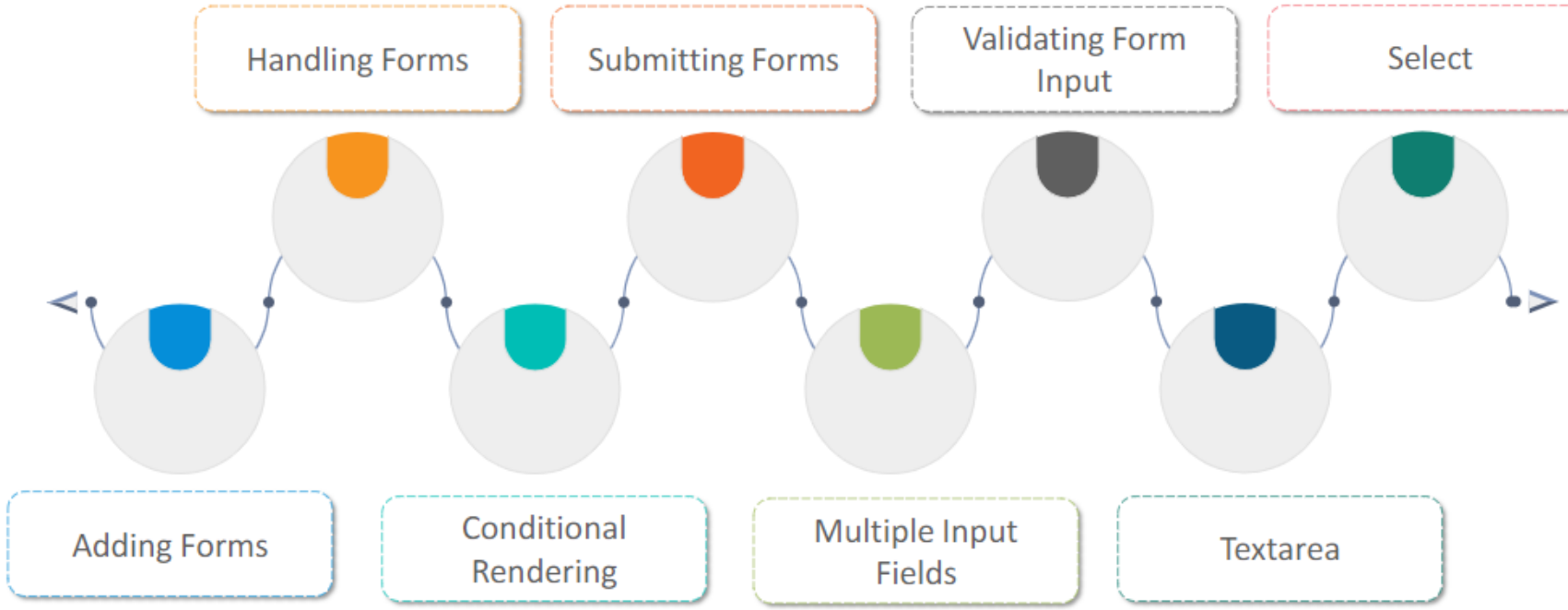


REACT FORMS

React Forms

React Forms are designed to let users interact with a ***Web Page***

Different activities associated with React forms are:



REACT FORM

Adding Form

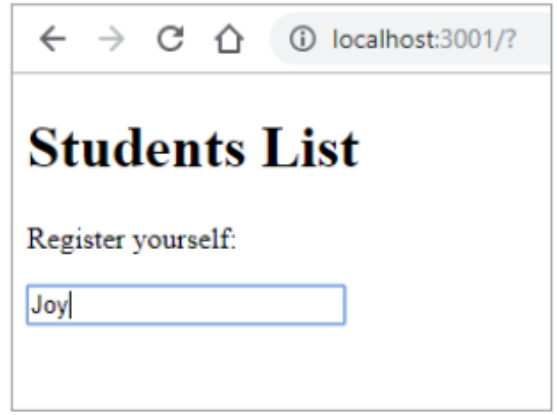
Below is the example of form which accepts the user inputs.

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';

class Form extends Component {
 render() {
  return (
   <form>
    <h1>Students List</h1>
    <p>Register yourself:</p>
    <input
     type="text"
    />
   </form>
  );
 }
}
ReactDOM.render(<Form />, document.getElementById('root'));
```

HTML element to create form

Input field where the user can enter data

Output



REACT FORM

Handling Forms

Handling forms refers to managing the data on submission or when the values are changed.

```
class Form extends Component {
 constructor() {
  super();
  this.state = { participate: '' };
 }
 changeHandler = (event) => {
  this.setState({participate: event.target.value});
 }
 render() {
  return (
   <form>
   <h1>Welcome {this.state.participate}</h1>
   <p>Register your name:</p>
   <input
    type='text'
    onChange={this.changeHandler}
   />
   </form>
  );}}
ReactDOM.render(<Form />, document.getElementById('root'));
```

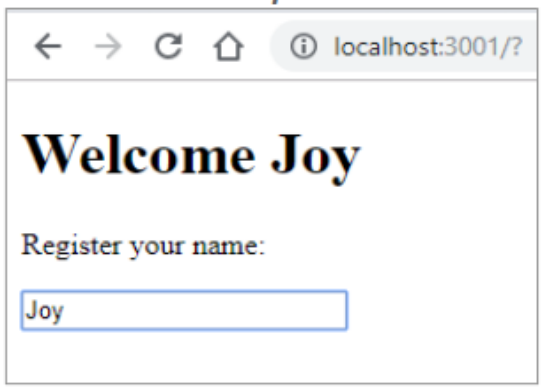
- React form is maintained by the React components and stored in component state
- These changes can be controlled by the addition of ***onChange*** attribute

State storing the data

Records the changes

Updated or submitted value

Output



REACT FORM

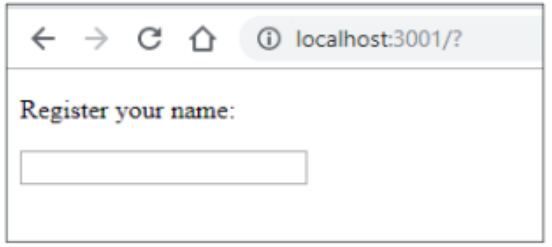
Conditional Rendering

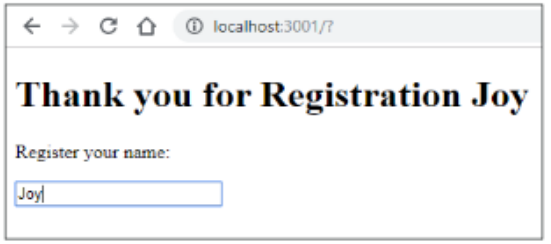
```
class Form extends Component {
  constructor(props) {
    super(props);
    this.state = { participate: '' };
  }
  changeHandler = (event) => {
    this.setState({participate: event.target.value});
  }
  render() {
    let header = '';
    if (this.state.participate) {
      header = <h1>Thank you for Registration {this.state.participate}</h1>;
    }
    return (
      <form>
      {header}
      <p>Register your name:</p>
      <input
        type='text'
        onChange={this.changeHandler}
      />
      </form>
    );}}
ReactDOM.render(<Form />, document.getElementById('root'));
```

Conditional Rendering is usually preferred to display the data after user interaction (submission).

Condition to ***render Header*** after participate registration

Output





REACT FORM

Forms Submission

```
class Form extends Component {
  constructor() {
    super();
    this.state = { participate: '' };
  }
  submitHandler = (event) => {
    event.preventDefault();
    alert(this.state.participate + " Registered" );
  }
  changeHandler = (event) => {
    this.setState({participate: event.target.value});
  }
  render() {
    return (
      <form onSubmit={this.submitHandler}>
      <h1>Welcome</h1>
      <p>Register your name and click on submit:</p>
      <input
        type='text'
        onChange={this.changeHandler}
      />
      <input
        type='submit'
      />
      </form>
    );}}
ReactDOM.render(<Form />, document.getElementById('root'))
```

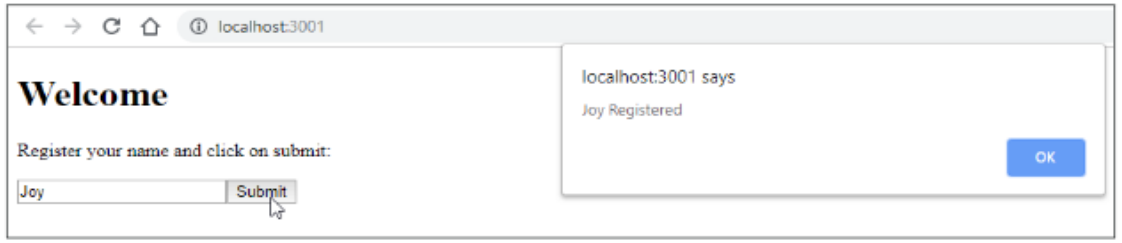
Form Submission refers to submission of data with user confirmation by clicking the submit button.

Event to be submitted after clicking submit button

Submits the entered data

Defines a submit button which submits all form values to a form-handler

Output



REACT FORM

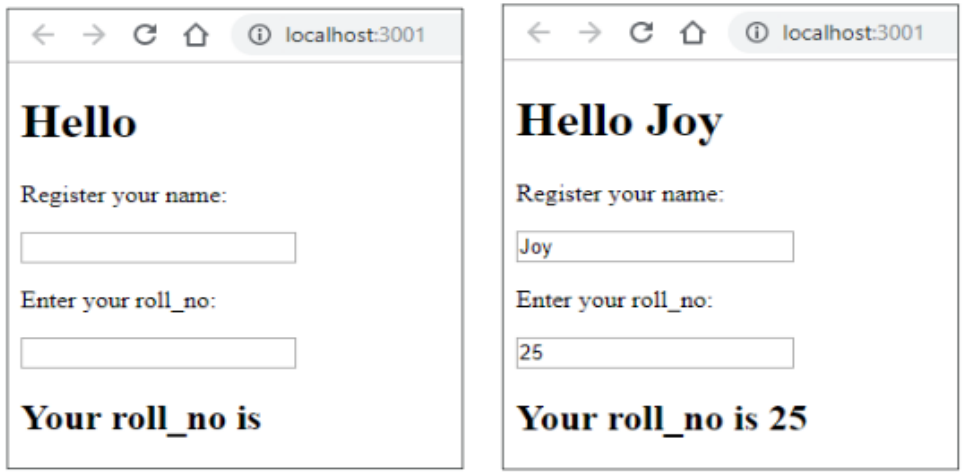
Multiple Input Fields

```
class Form extends Component {
  constructor() {
    super();
    this.state = {
      participate: '',
      roll_no: null,
    };}
  changeHandler = (event) => {
    let nam = event.target.name;
    let val = event.target.value;
    this.setState({[nam]: val});
  }
  render() {
    return (
      <form>
      <h1>Hello {this.state.participate} </h1>
      <p>Register your name:</p>
      <input
        type='text'
        name='participate'
        onChange={this.changeHandler}
      />
      <p>Enter your roll_no:</p>
      <input
        type='text'
        name='roll_no'
        onChange={this.changeHandler}
      />
      <h2>Your roll_no is {this.state.roll_no}</h2
>
      </form>
    );}}
ReactDOM.render(<Form/>, document.getElementById('
root'));
```

Multiple input fields include different categories to be mentioned in the form.

- Manages the updated values of name and roll_no
- Collects the username
- Collects the user roll_no

Output



REACT FORM

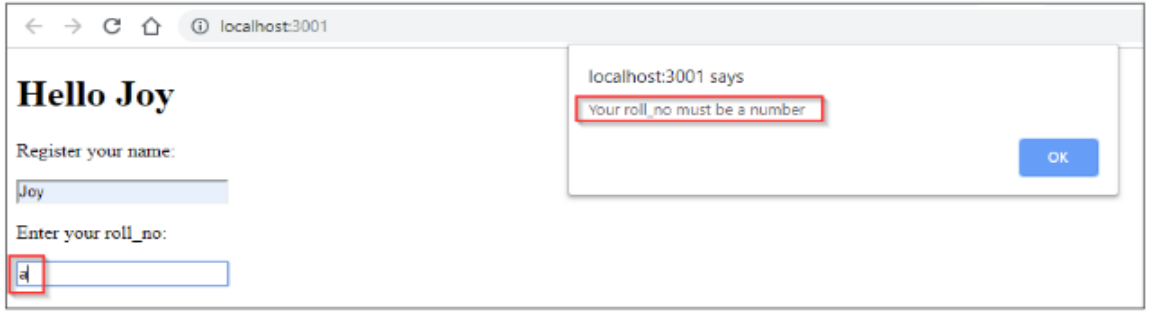
Validating Form Input

```
class Form extends Component {
  constructor() {
    super();
    this.state = {
      participate: '',
      roll_no: null,
    };}
  changeHandler = (event) => {
    let nam = event.target.name;
    let val = event.target.value;
    if (nam === "roll_no") {
      if (!Number(val)) {
        alert("Your roll_no must be a number");
      } }
    this.setState({[nam]: val});
  }
  render() {
    return (
      <form>
      <h1>Hello {this.state.participate} </h1>
      <p>Register your name:</p>
      <input
        type='text'
        name='participate'
        onChange={this.changeHandler}
      />
      <p>Enter your roll_no:</p>
      <input
        type='text'
        name='roll_no'
        onChange={this.changeHandler}
      />
      <h2>Your roll_no is {this.state.roll_no}</h2>
      </form>
    );}}
ReactDOM.render(<Form />, document.getElementById('root
'))
```

Form validation refers to entering the right input, if user enters some wrong values then the input is not accepted.

Output: When you enter right data

Output: When you enter wrong data



REACT FORM

Textarea

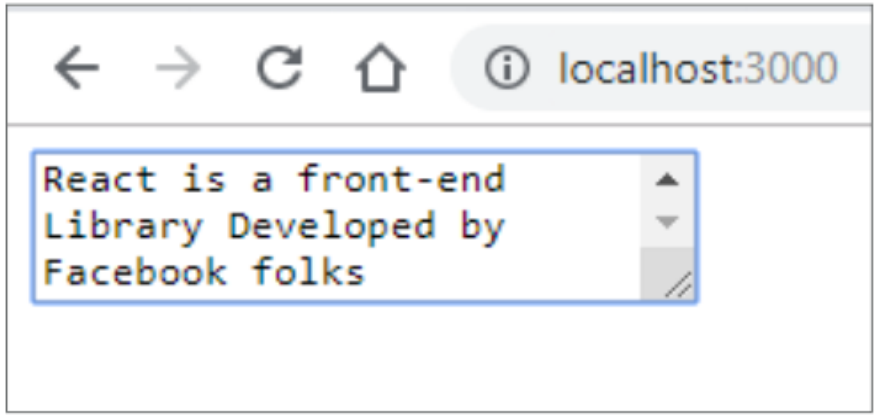
Textarea is one of the features of form, where data can be entered in textbox.

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';

class Form extends Component {
  constructor() {
    super();
    this.state = {
      description: 'React is a front-
end Library Developed by Facebook folks'
    };
  }
  render() {
    return (
      <form>
      <textarea value={this.state.description} />
      </form>
    );}}
ReactDOM.render(<Form />, document.getElementById('root'));
```

Note

In React the value of a textarea is placed in a ***value attribute***

Output



REACT FORM

Select

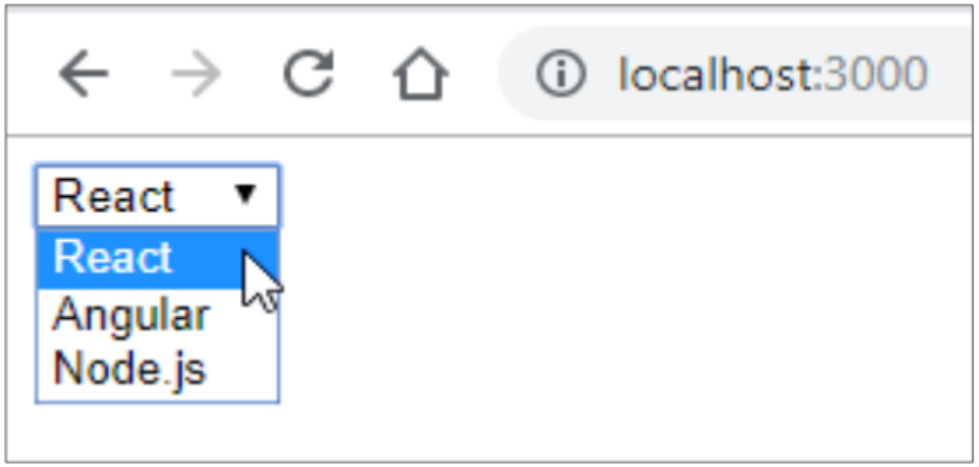
Select feature offers list of options, where user is supposed to make a choice of appropriate option.

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';

class Form extends Component {
  constructor() {
    super();
    this.state = {
      myTraining: "choose"
    };
  }
  render() {
    return (
      <form>
      <select value={this.state.myTraining}>
        <option value="React">React</option>
        <option value="Angular">Angular</option>
        <option value="Node">Node.js</option>
      </select>
      </form>
    );}}
ReactDOM.render(<Form />, document.getElementById('root'));
```

Note

In React, the selected value is defined with a value attribute on the select tag

Output



REACT -STYLING

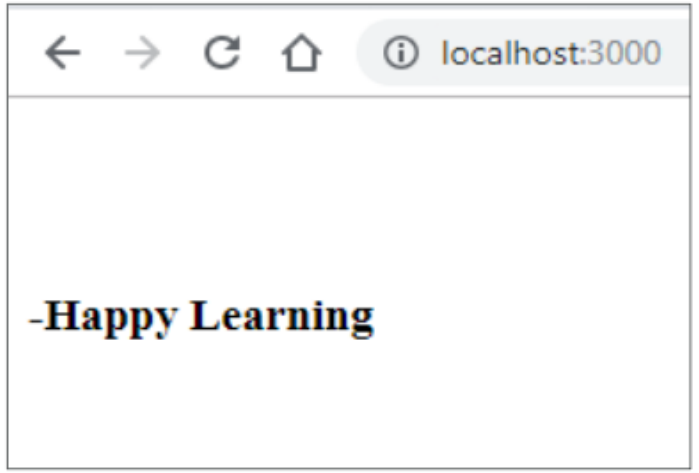
Inline Styling

Inlining CSS means putting your **CSS** into your HTML file instead of an external **CSS** file.

To style an element with the inline style attribute, the value must be a JavaScript object

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';

class Title extends Component {
  render() {
    return (
      <div>
      <h3 style={{color: "black"}}>-Happy Learning</h3>
      </div>
    );
  }
}

ReactDOM.render(<Title />, document.getElementById('root'));
```

Output



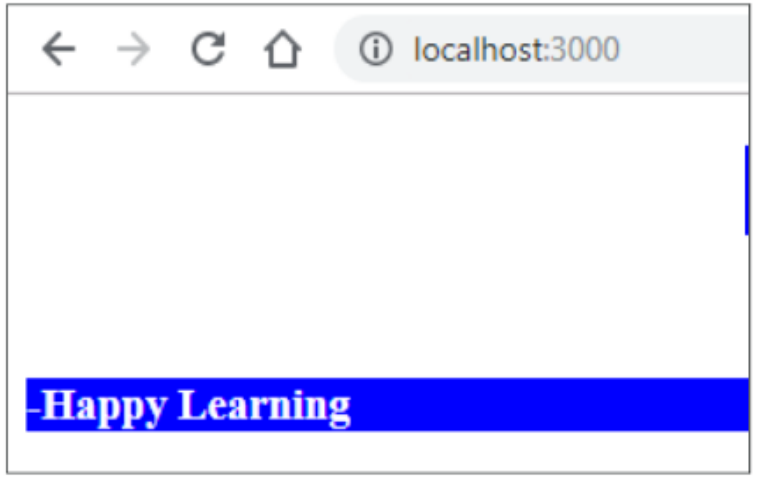
In JSX, JavaScript expressions are written inside curly braces, and since JavaScript objects also use curly braces, the styling in the example above is written inside two sets of ***curly braces {{}}***

REACT -STYLING

Adding Background Color To Text

In Inline CSS, properties with ***two words*** like ***background-color***, must be written in camelCase syntax.

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';

class Title extends Component {
  render() {
    return (
      <div>

      <h3 style={{backgroundColor: "blue",color: "white"}}>-Happy Learning</h3>
      </div>
    );
  }
}

ReactDOM.render(<Title />, document.getElementById('root'));
```



REACT -STYLING

CSS Stylesheet

This is an another way where CSS styling is written in a separate file and saved with the ***.css file extension***, which later you can import it in your application.

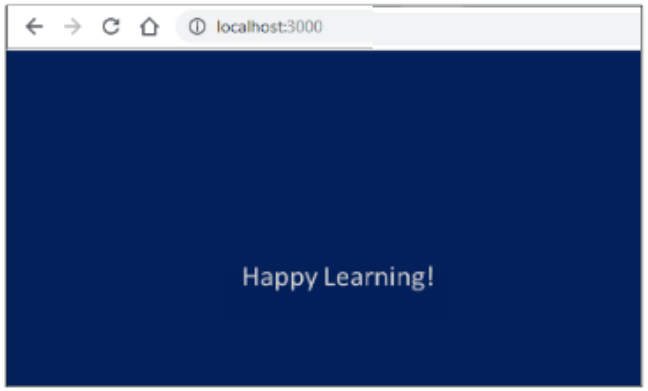
App.css

```
body {
    background-color: #03205a;
    color: rgb(255, 255, 255);
    padding: 100px;
    font-family: 'Gill Sans';
    text-align: center;
  }
```

Index.js

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';
import './App.css';

class Title extends Component {
  render() {
    return (
      <div>

      <p>Happy Learning!</p>
      </div>
    );
  }
}

ReactDOM.render(<Title />, document.getElementById('r
oot'));
```

Output



REACT : REDUX

Redux: convenient state container

- ✓ Redux is framework-agnostic and you can happily use it with Angular or Vue
- ✓ You can use Redux with Angular, but the probability of a React developer knowing Redux is much higher than knowing Angular. And you'll find more community support for tackling the React-Redux learning curve.
- ✓ Redux simplifies storing and managing component states in large applications with many dynamic elements where it becomes increasingly difficult.
- ✓ Redux stores application state in a single object and allows every component to access application state without dealing with child components or using callbacks.
- ✓ For instance, when you have two components that share the same and stand apart in the tree, without Redux, data has to be passed through multiple intermediary components with all the problems that go with it.

REACT : REDUX

REACT : REDUX

REACT : REDUX

NATIVE-REACT

- What is **React Native**? **React Native** is a framework created by Facebook to enable a smooth and easy cross-platform mobile development. It basically means that you don't have to create an iOS and Android app separately
- **React Native** is an entire platform allowing you to build **native**, cross-platform mobile apps, and **React. js** is a **JavaScript** library you use for constructing a high performing UI layer. **React. js** is the heart of **React Native**, and it embodies all **React's** principles and syntax, so the learning curve is easy.
- React Native is an exciting framework that enables web developers to create robust mobile applications using their existing **JavaScript** knowledge. It offers faster mobile development, and more efficient code sharing across iOS, Android, and the Web, without sacrificing the end user's experience or application quality

NATIVE-REACT

- React Native is an exciting framework that enables web developers to create robust mobile applications using their existing **JavaScript** knowledge. It offers faster mobile development, and more efficient code sharing across iOS, Android, and the Web, without sacrificing the end user's experience or application quality.
- **React Native** is based on **React**. **js**, which is written in JavaScript. ... While Java and Swift/Objective-C are strongly-typed, compiled **languages**, JavaScript is interpreted and often called an untyped **language**.

NATIVE-REACT

React. **js** focuses on a **better** UI, so those benefits remain. You don't have to build the same application for iOS and Android, separately as **React Native** allows your developers to reuse the common logic layer. ... **Native** app development usually means inefficiency, slower time to deployment, and less developer productivity.

Can we convert react to react native?

- There's no quick way to **convert react to react-native**. A possible alternative if **you** want your **react** app to run on a mobile device without rewriting your codebase is to use Cordova. ... Use xcode to open open your **react**-cordova .

NATIVE-REACT

NATIVE-REACT

- **React**. **js** focuses on a **better** UI, so those benefits remain. You don't have to build the same application for iOS and Android, separately as **React Native** allows your developers to reuse the common logic layer. ... **Native** app development usually means inefficiency, slower time to deployment, and less developer productivity.
- If you **need to** develop an app for both iOS and Android, **React Native** is the best tool out there. It can reduce the codebase by about 95%, saving you time and money. On top of that, **React Native** has a number of open-source libraries of pre-built components which can help you further speed up the development process.

React App: Environment

Summary:

- The core features of ReactJS includes JSX, components(functional components and class-based components), the life cycle of a component, props, and state support for a component, working with javascript expressions.
- Project setup of ReactJS is explained using CDN files and also using npm packages to build the project.
- JSX is an extension to javascript. It is a template script where you will have the power of using Html and javascript together.
- Components are like pure javascript functions that help make the code easy by splitting the logic into reusable independent code.

React App: Environment

Summary:

- A state is a javascript object similar to props that have data to be used with the reactjs render. The state data is a private object and is used within components inside a class.
- Props are properties to be used inside a component.
- A component life cycle is divided into Initialization, Mounting, Update, and UnMounting stages.
- In reactjs html input elements like <input />, <textarea /> and <select /> has their own state and needs to be updated when user intereacts using the setState() method.
- Working with events in reactjs is same as how you would have done in javascript. You can use all the event handlers that are used in javascript. The setState() method is used to update the state when the user interacts with any Html element.
- ReactJS allows you to work with external css as well as inline css using javascript expression.