React

ANIL JOSEPH

Agenda

- Introduction to React
- Creating React Applications with Toolchains
- Understanding Components
- Components Deep Dive
- React Hooks
- Error Handling & Debugging
- Server-side communication with HTTP Requests
- Single Page Applications with Routing
- Redux & React Context
- Server-side rendering
- Webpack
- Deployment
- Introduction to Testing

Presented By Anil Joseph(anil.jos@gmail.com)

Anil Joseph

Introduction

- Over 20 years of experience in the industry
- Technologies
  - C, C++
  - Java, Enterprise Java
  - .NET & .NET Core
  - **UI Technologies: React, Angular, jQuery, ExtJs**
  - Mobile: Native Android, React Native
- Worked on numerous projects
- Conducting trainings for corporates (700+)

Presented By Anil Joseph(anil.jos@gmail.com)

Software

Node.js and NPM

- Version 10 or higher

Yarn(Optional)

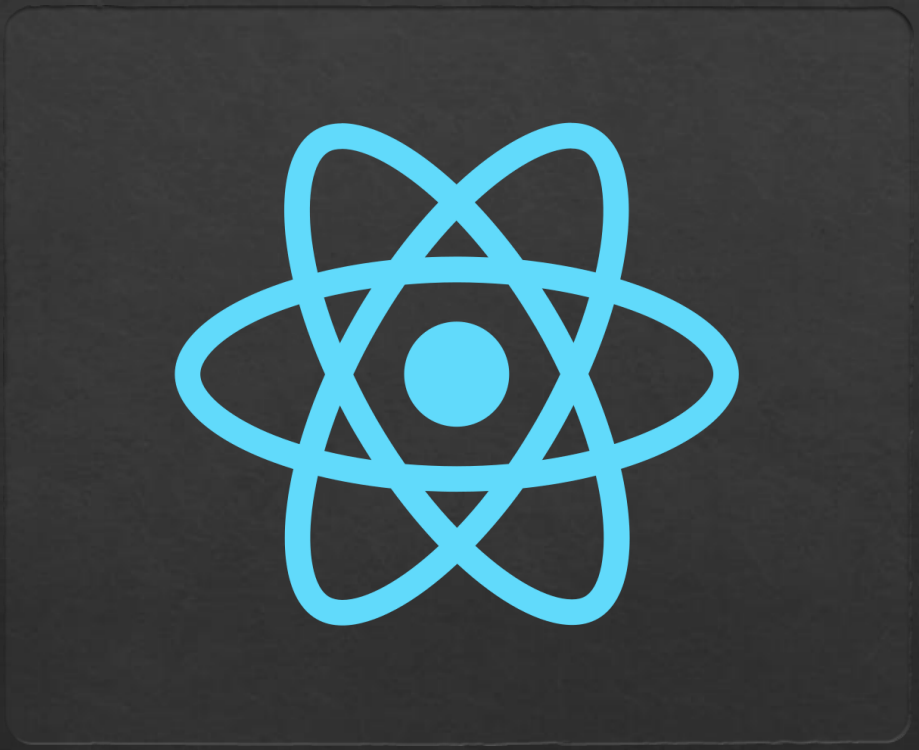
JavaScript Editor(Any one)

- **Visual Studio Code**
- Sublime Text
- Atom

Browsers

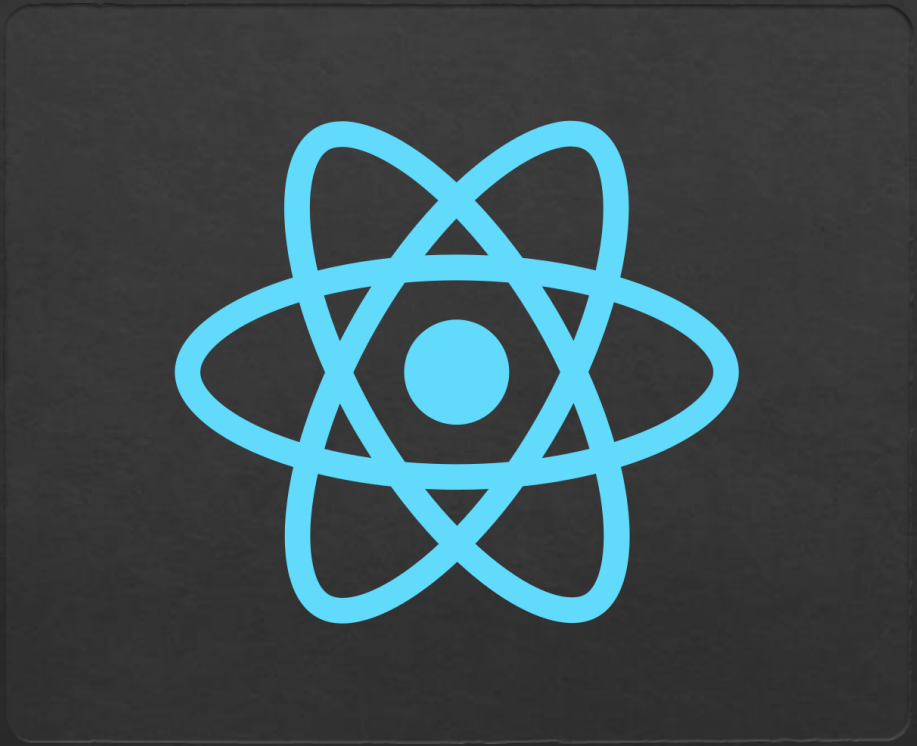
- Chrome, Firefox

What is React?



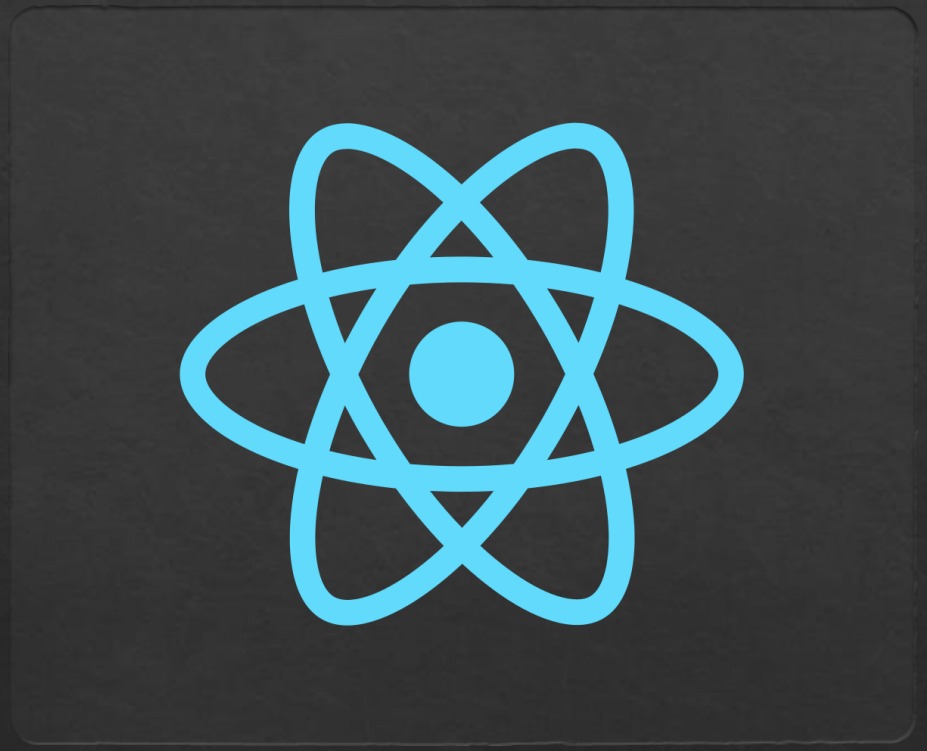
- A JavaScript library to build ***User Interfaces***.
- Building applications for the ***browser***.
- React allows to create custom HTML elements called ***components***.
- Components are used to build ***complex User Interface***
- Reacts promotes writing ***maintainable, manageable and reusable code***.

Why React?



- React handles *State*
  - UI State is difficult to handle with JavaScript
- React focuses on business logic
  - Focus on business logic
- React is *fast*.
  - Apps made in React can handle complex updates and still feel quick and responsive.

Why React?



- React is *modular*
  - Instead of writing large, dense files of code, you can write many smaller, reusable files.
- React is *scalable*.
  - Large programs that display a lot of changing data are where React performs best.
- React has a Huge ecosystem, community

History

React was created by **Jordan Walke**, a software engineer at Facebook.

It was first deployed on **Facebook's** newsfeed in 2011

Later on **Instagram** in 2012.

It was open-sourced in 2013.

Alternatives

Angular

- Open-source front-end web application platform from Google.

Vue

- Open-source progressive JavaScript framework for building user interfaces.

Getting Started

The Two React libraries

- React
- ReactDOM

JSX

- A JavaScript extension syntax allowing quoting of HTML

Babel

- The compiler for writing next generation JavaScript.
- Compiles JSX to JavaScript

Creating a React Project

Tool-Chain: Why?

Manage Dependency

- Use multiple libraries

Optimize Code

- Small in size

Next Generation Features using ES6 and ES7

- Leaner code, Easier to read, Faster

More Productive

- CSS Auto Prefixes, Linting

Tool-Chain: How?

Dependency Management Tool

- npm and yarn

Bundler

- WebPack

Complier

- Complies ES6/ES7 to ES5
- Compiles JSX to JavaScript
- Babel + Presets

Development Server

- Web Server for development

Tool-Chain

Create-React-App

- The React team provides a toolchain called **Create-React-App**.
- Use to quickly start development on React with zero configuration
- Integrates the various tools and libraries required for the build process
  - Webpack, NPM, Yarn, Babel
- Uses the ***react-scripts*** package

Create Project(1)

1

Install NodeJs

- Runtime to run/execute JavaScript on the machine/server
- This installation also installs npm(Node Package Manager)

2

Install “create react app”

- **Tool to create react applications**
- **npm install –g create-react-app**

3

Create Application/Project

- **create-react-app the-react-app**

4

Start the Application

- **cd the-react-app**
- **npm start**

Create Project(2)

1

Install NodeJs

- Runtime to run/execute JavaScript on the machine/server
- This installation also installs npm(Node Package Manager)

2

Create Application/Project

- **npx create-react-app the-react-app**

3

Start the Application

- **cd the-react-app**
- **npm start**

Create Project(3) Typescript

1

Install NodeJs

- Runtime to run/execute JavaScript on the machine/server
- This installation also installs npm(Node Package Manager)

2

Create Application/Project

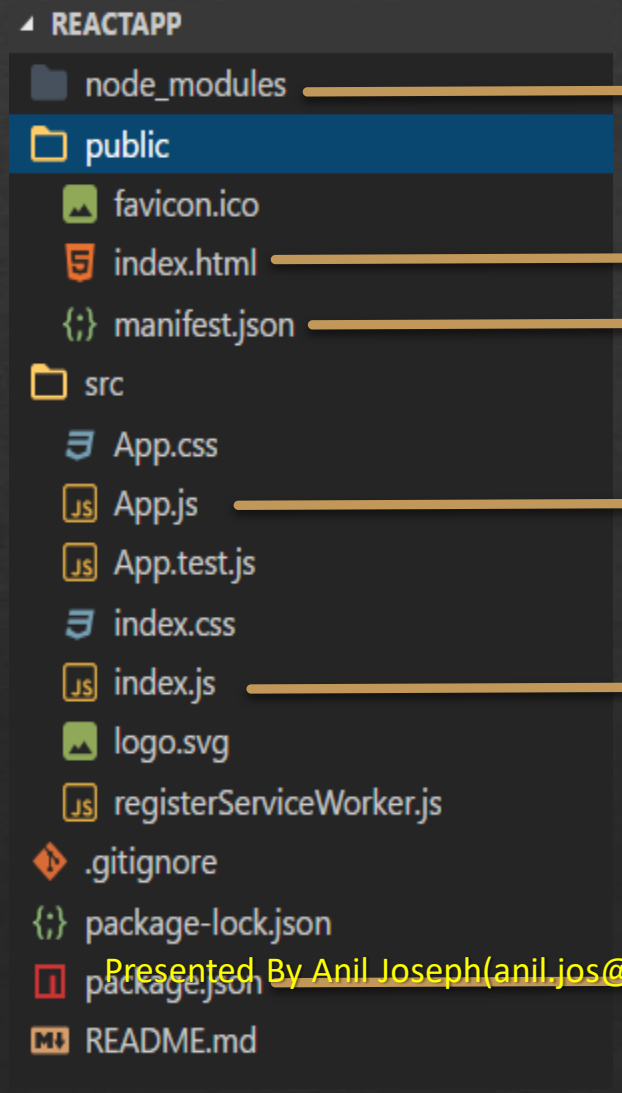
- **npx create-react-app the-react-app –template typescript**

3

Start the Application

- **cd the-react-app**
- **npm start**

Project Structure



Folder with all downloaded dependencies

The page displayed in the browser

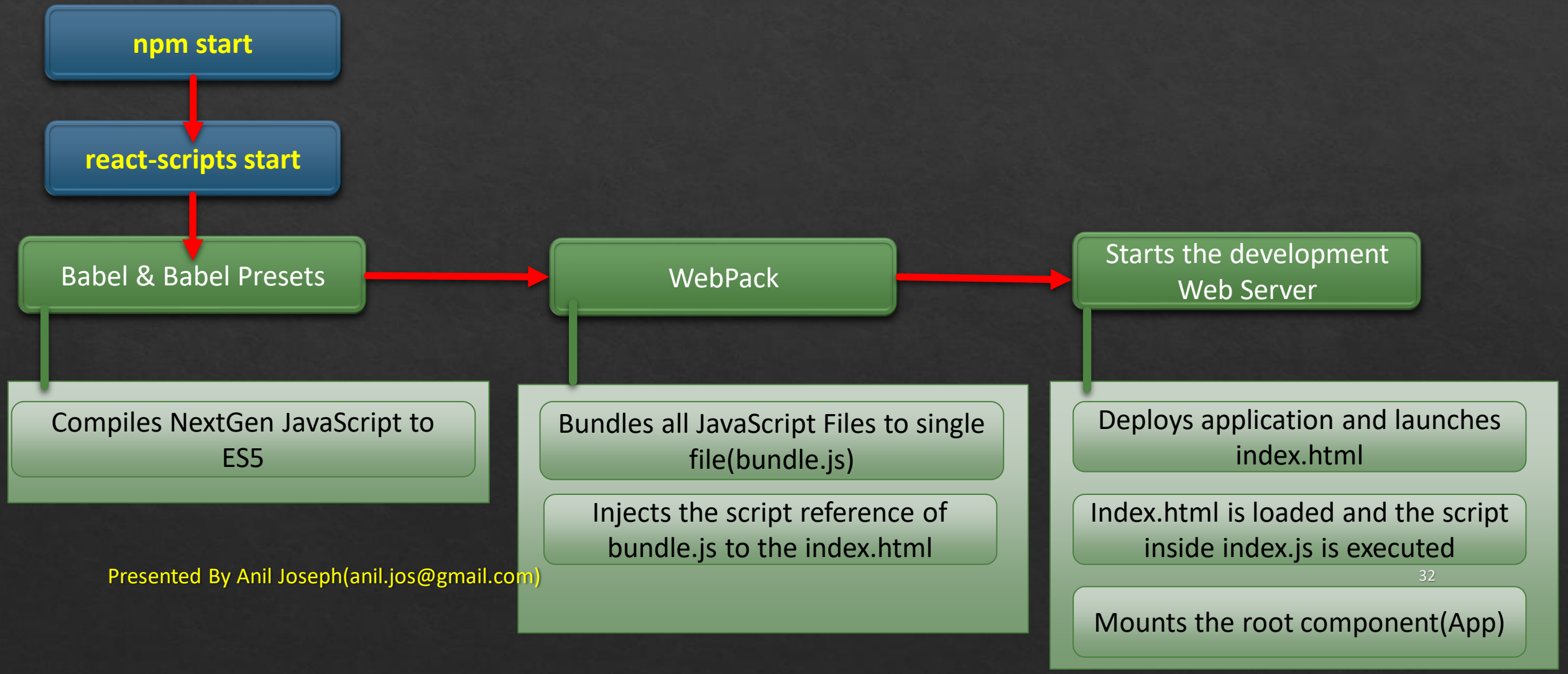
The Web Application Manifest (for installable web applications)

The root component

Initial Script executed on page load. Renders the root component

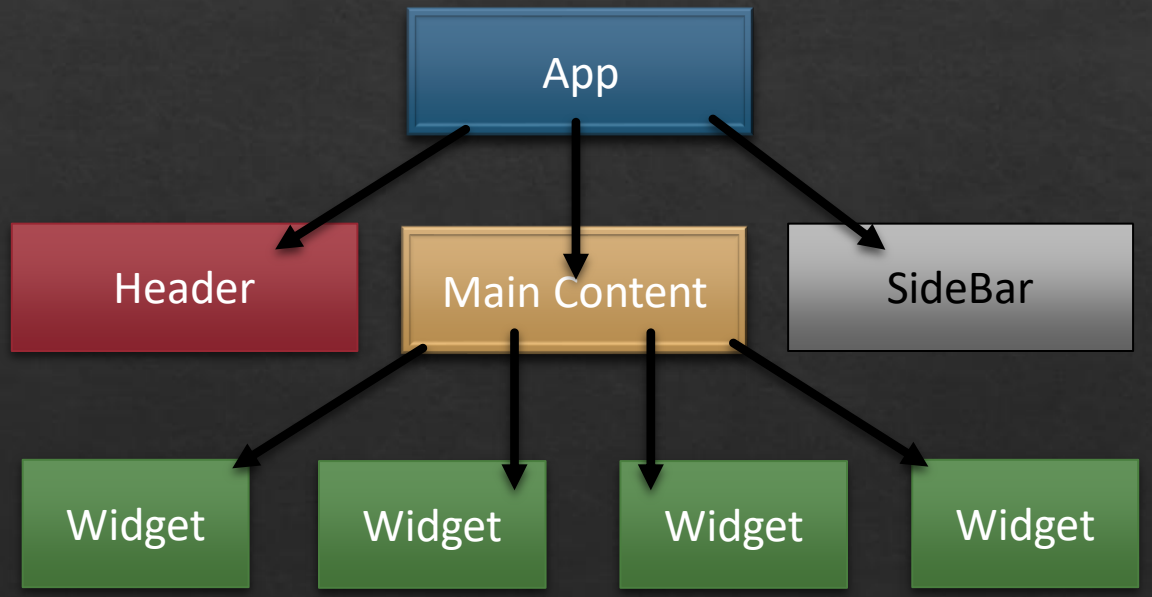
NPM configuration file, contains the dependencies and scripts

Project Execution



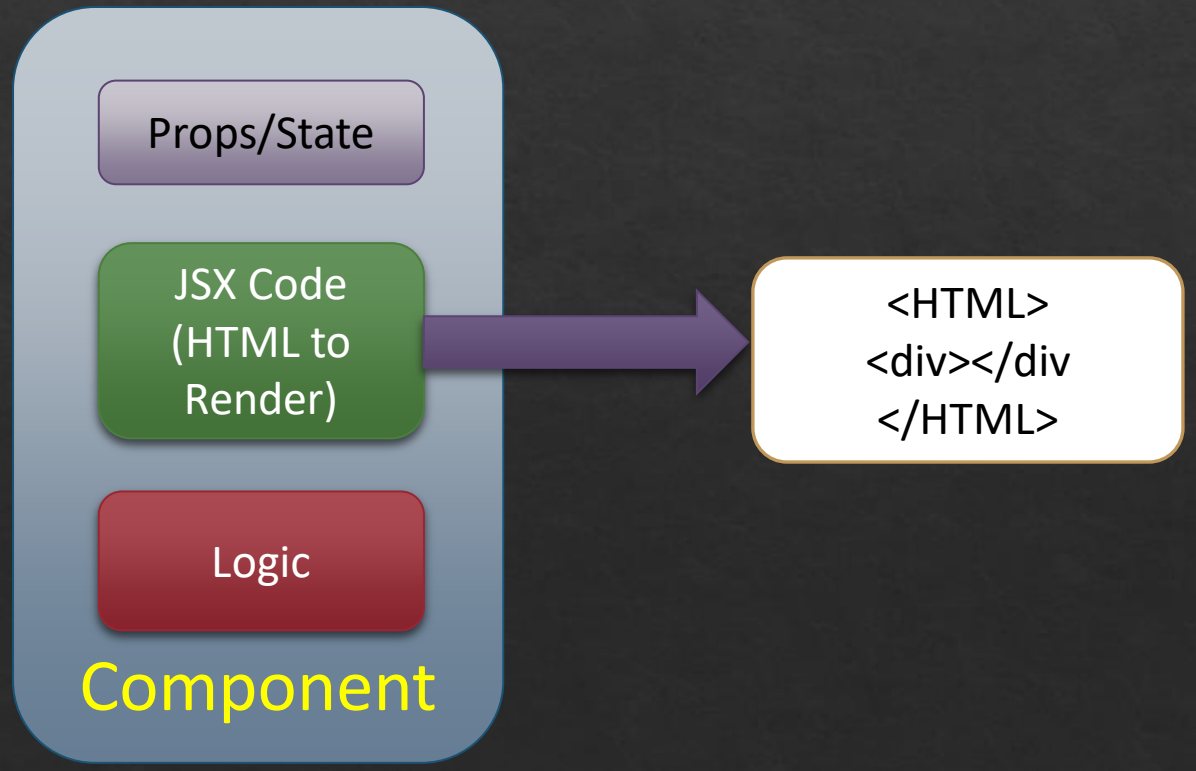
React Components

- Components are the **core building blocks of React applications**
- Creating a React Application is all about creating components.
- A React app can be depicted as a **component tree**



React Components

- The UI in a React Application is composed of ***components,*** the building blocks.
- Components are designed to be reusable.



Types of Components

Functional

- Presentational
- Stateless (till React 16.8)
- Stateful (using React Hooks)

Class Based

- Containers
- Stateful

JSX

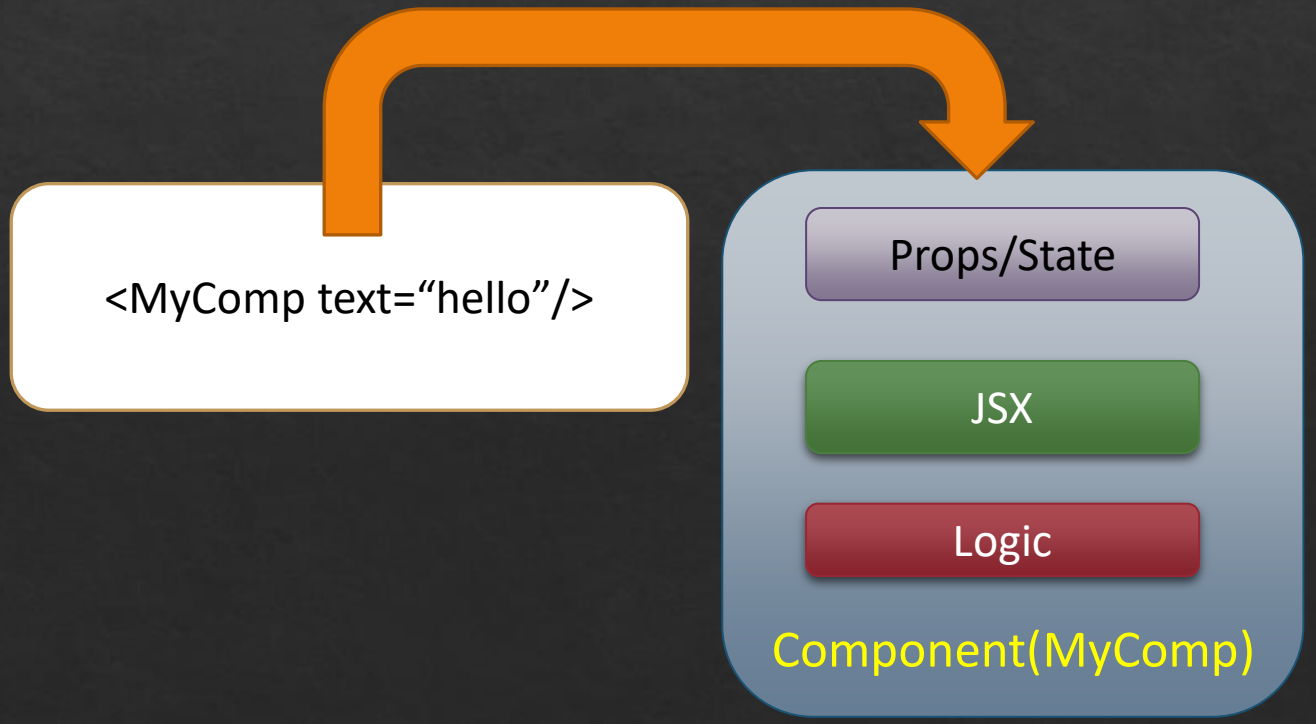
- *JSX* is a syntax extension for JavaScript.
- It was written to be used with React.
- JSX code looks a lot like HTML.
  - **It's actually JavaScript**
- A *JSX compiler* will translate any JSX into regular JavaScript.
- JSX elements are treated as JavaScript *expressions*.
  - They can go anywhere that JavaScript expressions can go.
- That means that a JSX element can be
  - Saved in a variable
  - Passed to a function
  - Stored in an object or array

Components: Dynamic Content

- Dynamic content is outputted in the JSX using an expression.
- The syntax
  - { expression }
- This can be any one line expression
- Complex functionalities can be done by calling functions
  - { invokeSomeMethod() }

Components: Properties(props)

- Most components can be customized with different parameters when they are created.
- These creation parameters are called ***"props"***.
- ***Props*** are used similar to HTML attributes.
- ***Changes to props will automatically re-render the component.***



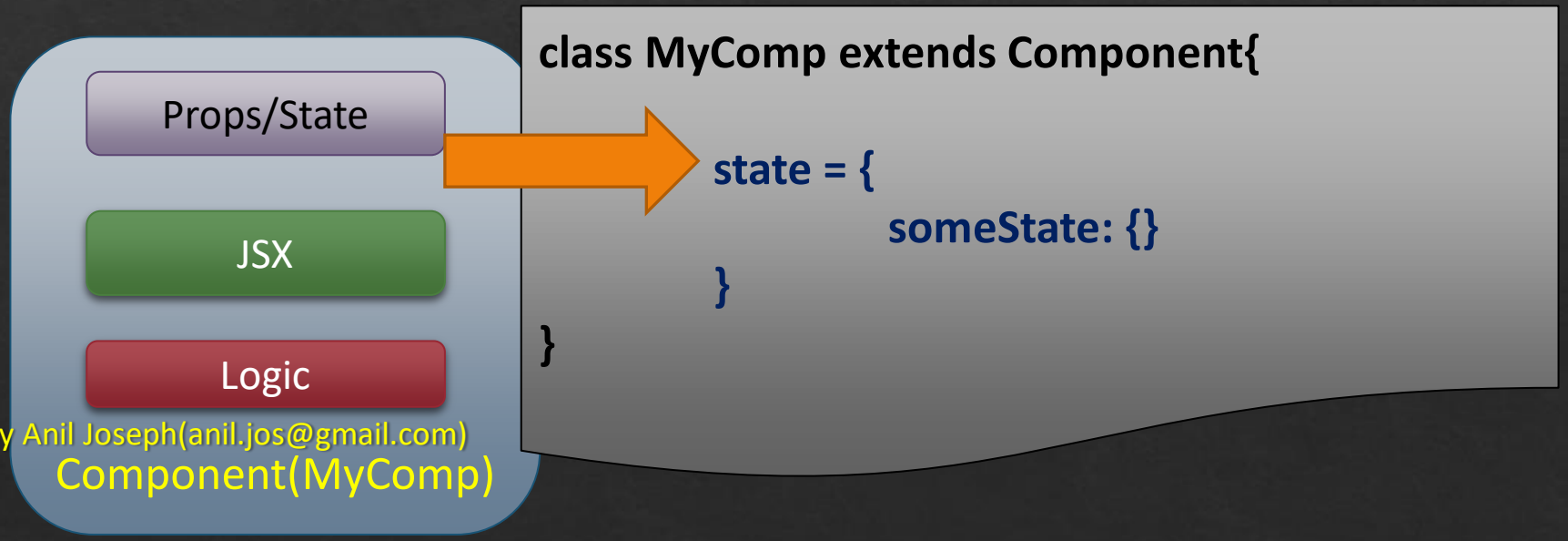
Props Types

Use Prop-Types for Type Checking the Component Properties

npm install prop-types

Component State

- State holds information about the component
- State is used when a component needs to keep track of information between renderings.
- State is created and initialized in the component itself.
- State is available only with Stateful components(Class Based).
- **Component state should be always updated using the setState method**
- **State updates trigger a rerender of the component.**



Props and State

"props" and "state" are CORE concepts of React.

Only changes in "props" and/ or "state" trigger React to re-render the components and potentially update the DOM in the browser

props allow you to pass data from a parent (wrapping) component to a child (embedded) component.

State is used to change the component from within.

Event Handling

- Handling events with React elements is very similar to handling events on DOM elements with some syntactic differences.
- React events are named using camelCase, rather than lowercase.
- With JSX you pass a function as the event handler, rather than a string.
- Event handlers will be passed instances of ***SyntheticEvent***
  - A cross-browser wrapper around the browser's native event
- Details
  - https://reactjs.org/docs/events.html

AJAX and APIs

- React does not have any API's for AJAX calls.
- We have to use an AJAX Library for server communications
- Popular Libraries
  - Axios
  - jQuery AJAX
  - Fetch API
- In a component AJAX calls to fetch data from the server should be made in the ***componentDidMount*** lifecycle method.

axios

- Promise based HTTP client for the browser and node.js
- Installation
  - npm install axios
- Features
  - Make http requests
  - Supports the Promise API
  - Intercept request and response
  - Transform request and response data
  - Cancel requests
  - Automatic transforms for JSON data
  - Client side support for protecting against XSRF

axios methods

axios.request({config})

axios.get(url, {config})

axios.post(url,{data}, {config})

axios.delete(url, {config})

axios.put(url,{data}, {config})

axios global defaults

Base URL

- axios.defaults.baseURL = 'https://abc.com’;

Headers

- axios.defaults.headers.common['Authentication'] = AUTH_TOKEN;

Headers for specific methods

- axios.defaults.headers.post['Content-Type'] = 'application/json';

Day 1 Recap

React Introduction

- A JavaScript framework to create client-side web applications
- Components are the core building blocks

Components

- Represents a UI element or widget designed to be reusable
- Types: Functional and Class-based
- Props & state
- View (defined using JSX)
- Component Logic

Day 1 Recap

Props

- Parametrize a component
- Pass data from Parent to Child and vice versa
- Props are read-only

State

- Internal to a component
- To be treated as immutable
- Use setState to update the state

Day 1 Recap

JSX

- JavaScript Extensions
- Compiled to JavaScript by Babel
- The View is written in JSX

Events

- Like HTML events
- React specific syntax

Toolchains

- Used the create-react-app toolchain to create a React project
- Create-react-app for SPA
- Chains Babel, Webpack, NPM and the Dev Server