Anil Joseph
anil.jos@gmail.com

React

ANIL JOSEPH

Agenda

Overview of ES6/7 features	TypeScript	Introduction to React	Creating React Applications with ToolChains
Understanding Components	JSX	Components Deep Dive	Unit Testing & TDD
React Hooks	Styling Components	Debugging	Server-side communication with HTTP Requests
Single Page Applications with Routing	Forms	Redux	E2e Testing
	Security	Deployment	

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

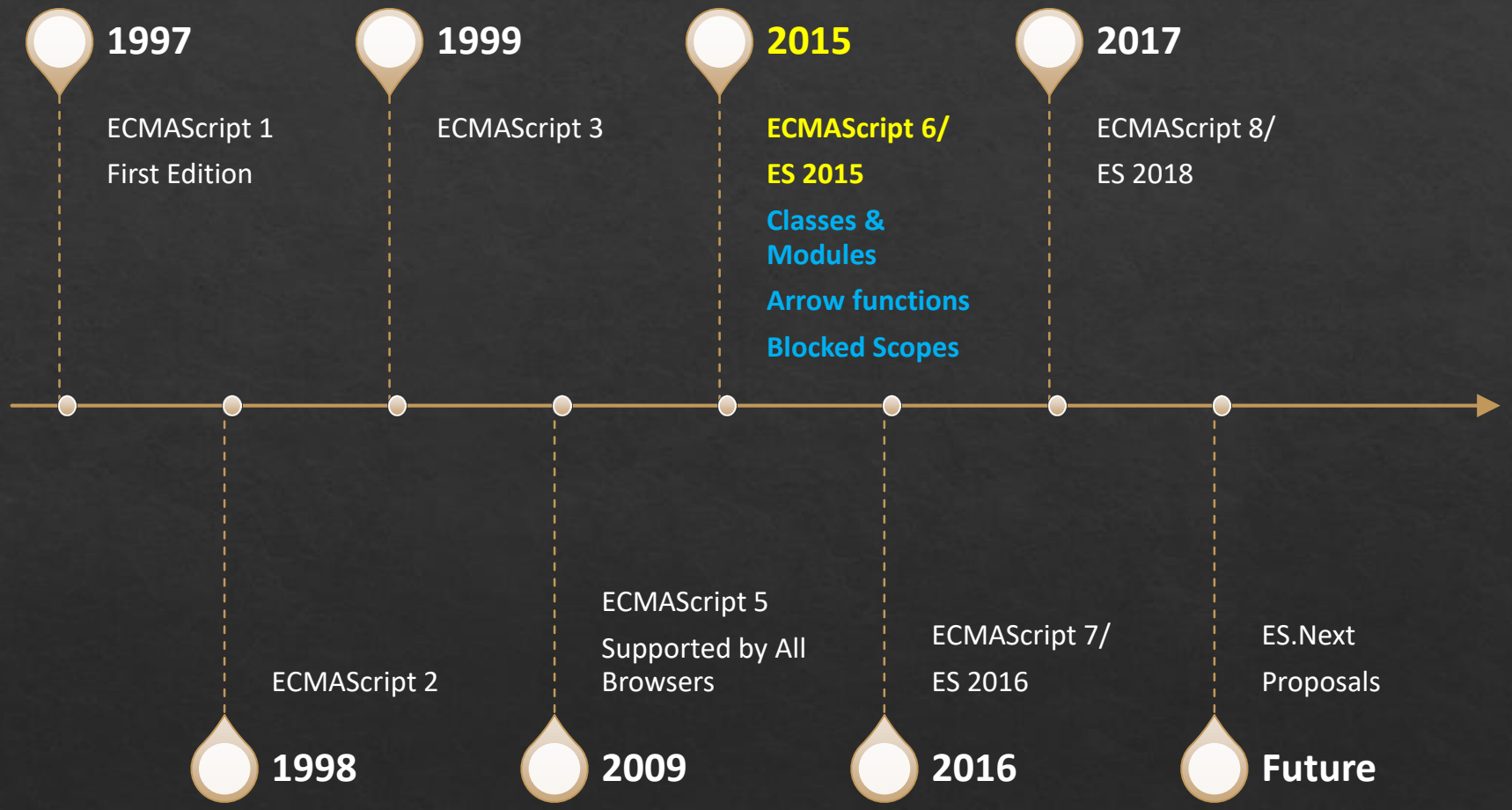
Presented By Anil Joseph(anil.jos@gmail.com)

JavaScript

- JavaScript (JS) is an interpreted programming language.
- It’s a dynamic language.
- Supports object-oriented programming.
- As part of web browsers, implementations allow
  - Client-side scripts to interact with the user.
  - Alter the document content that is displayed.
  - Control the browser.
  - Communicate asynchronously.
- JavaScript was developed by Brendan Eich at Netscape.
- Released in September 1995

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

ECMAScript Versions



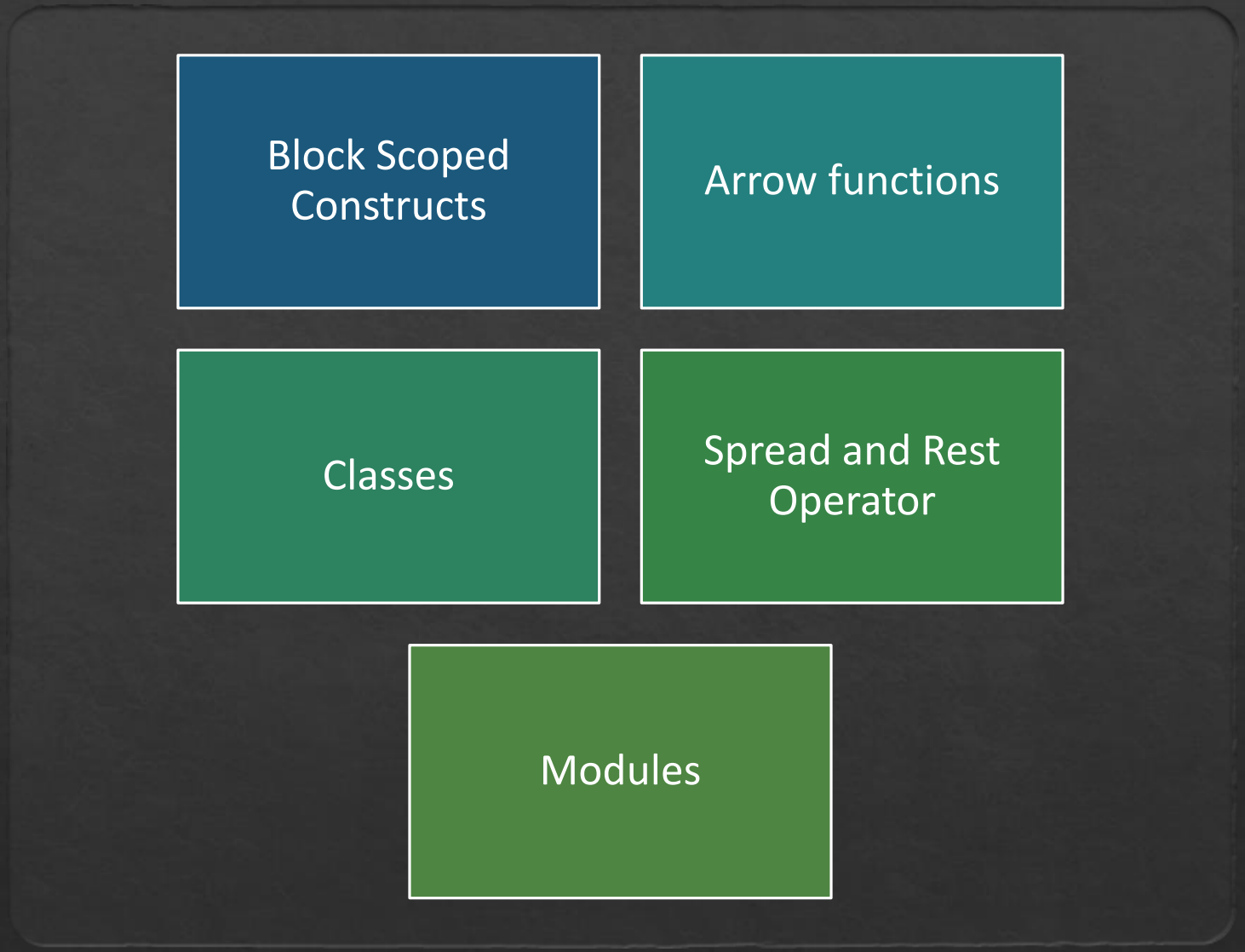
Presented By Anil Joseph(anil.jos@gmail.com)

Challenges with JavaScript

- Inconsistent browser implementations to access the DOM, XML parsing etc.
- JavaScript supports ***global variables***, which could cause conflicts when using multiple libraries.
- Creations of ***namespaces*** is not straight forward.
- Implementation of ***object-oriented code*** differs from conventional object-oriented languages.
- Some language features are complex to understand

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

ES6/7 Features (Used in React)



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

Block Scoped Constructs

- **Scope** determines the accessibility (visibility) of variables.
- ES5 supports two scopes
  - Global
  - Functional
- ES 6 introduces the block scoped constructs
  - **let**
  - **const**

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

Arrow Functions

Represents a function expression

An arrow function expression has a shorter syntax than a function expression

They do not receive the implicit arguments “this” and “arguments”

Used widely for asynchronous and functional programming

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

Spread and REST Operators

- **Spread syntax** allows
  - An iterable to be expanded in places where zero or more arguments are expected
  - Example: An array to be converted to a list of arguments of a method.
- **Rest parameter** syntax allows
  - To represent an indefinite number of arguments as an array.

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

ES6 Classes

- ES 6 supports class based object-oriented programming
- Classes in ES6
  - Constructor
  - Properties
  - Methods
  - Static methods
  - Inheritance
- Classes are just syntactic sugaring of constructor functions.

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

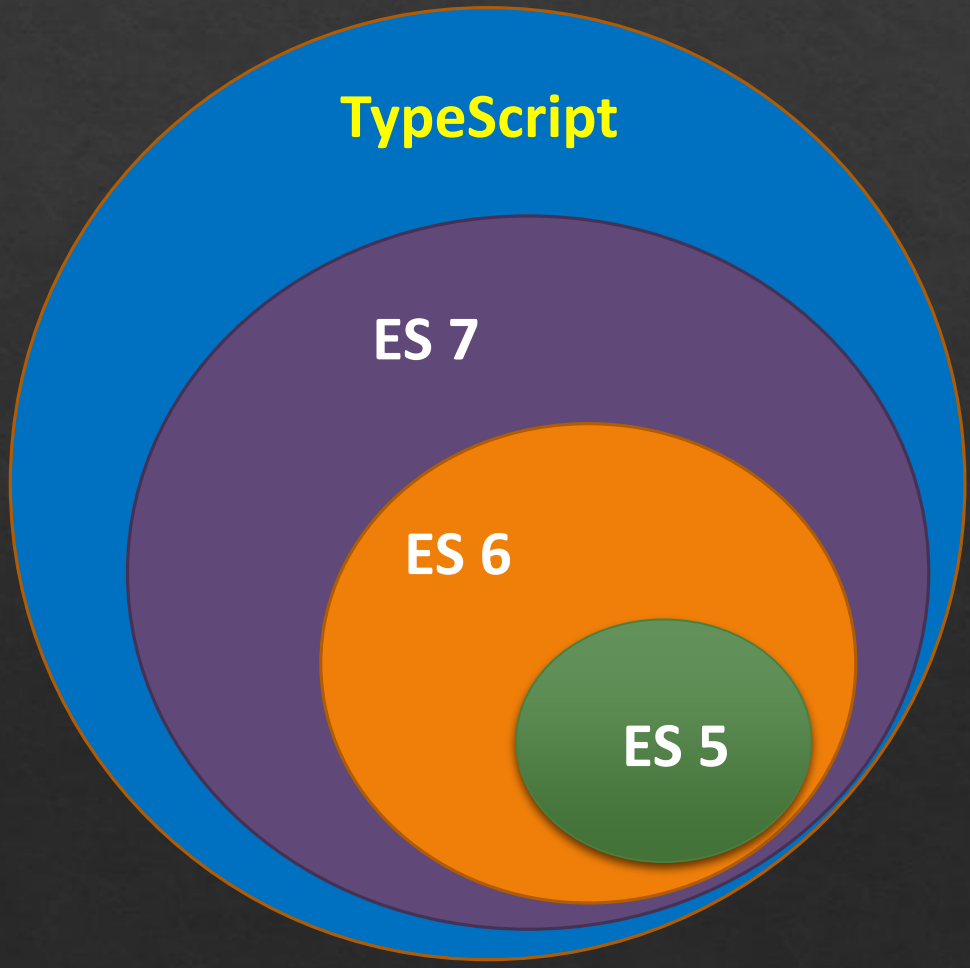
TypeScript

TypeScript is programming language developed and maintained by Microsoft.

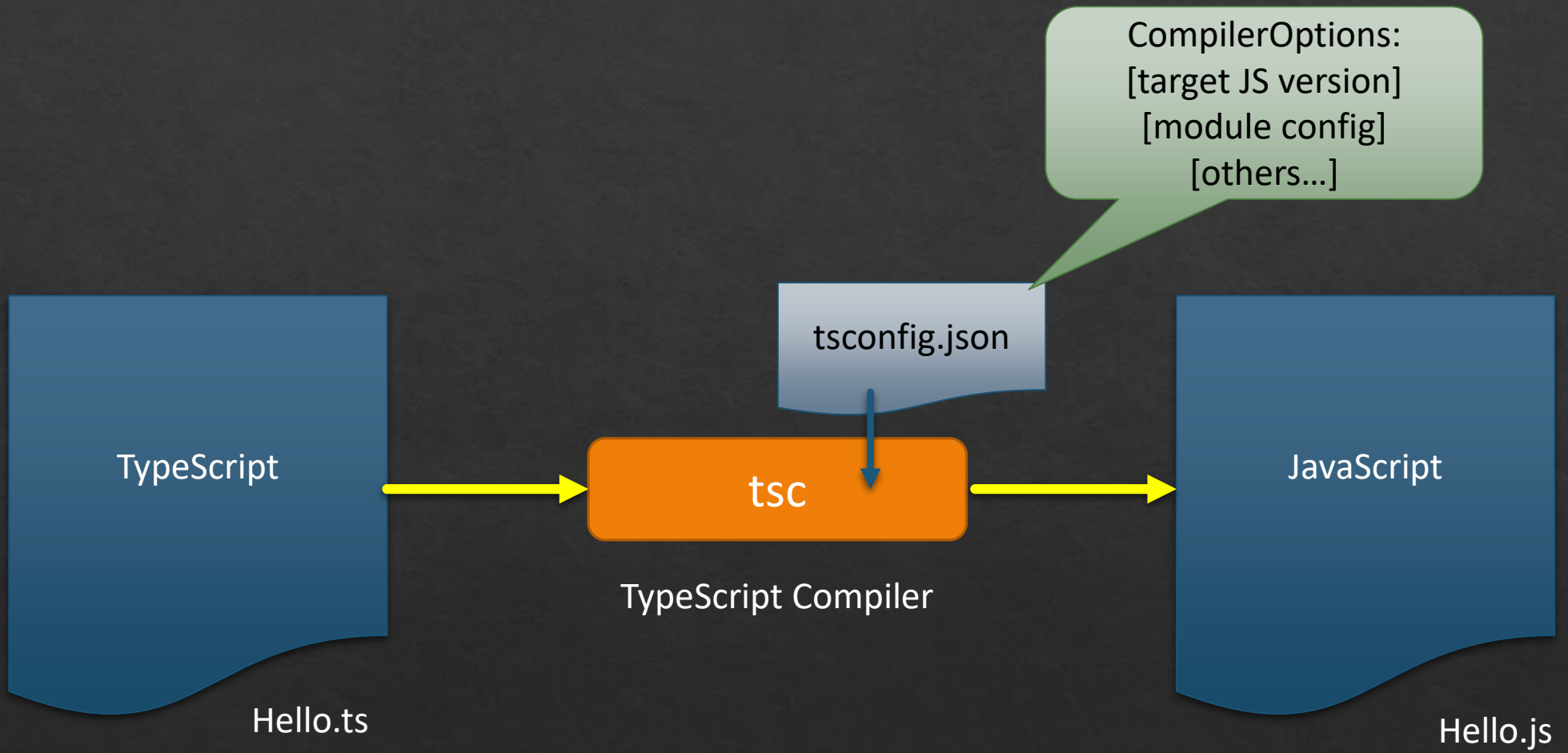
TypeScript is a typed superset of JavaScript.

Transcompiles to JavaScript.

Open Source and designed for development of large applications.

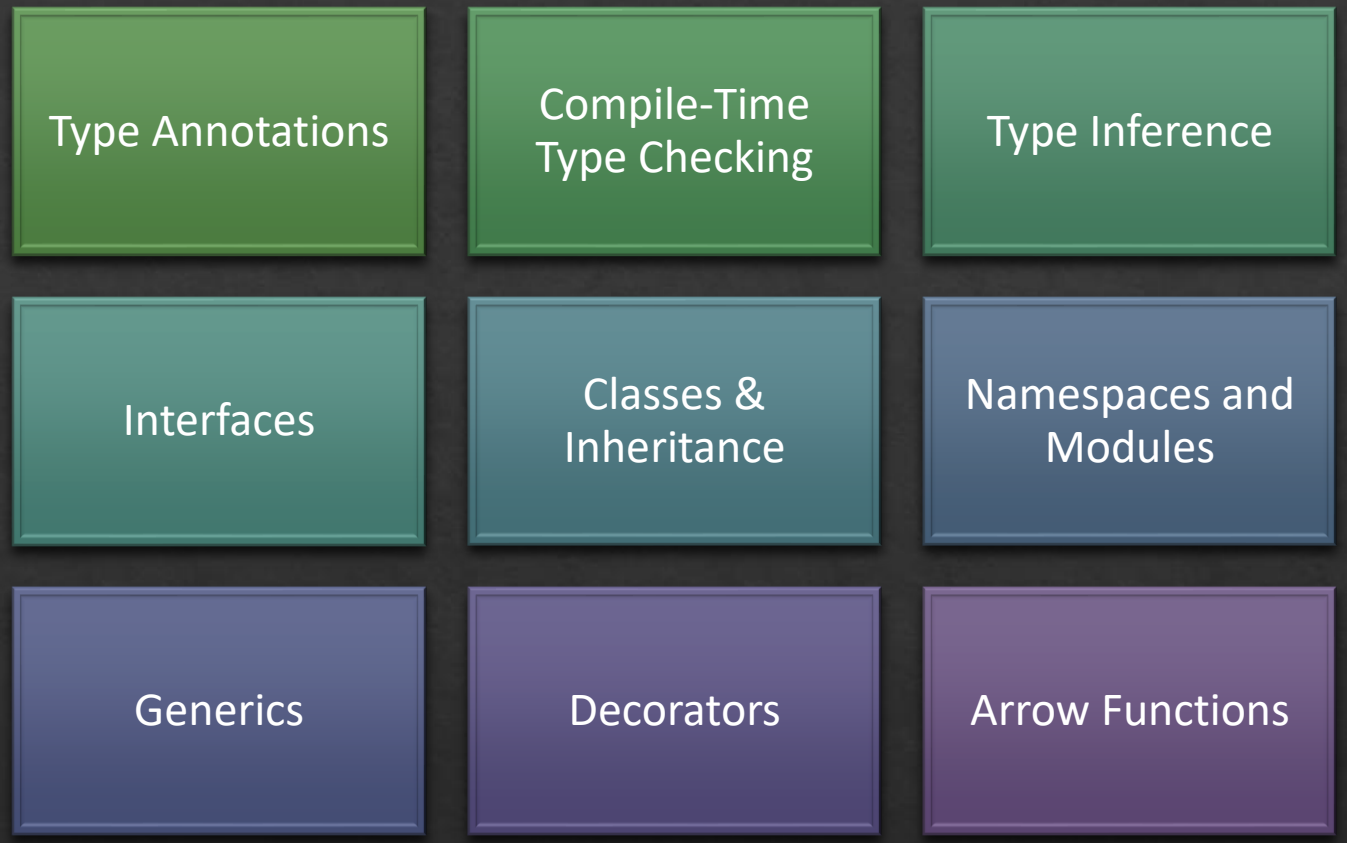


TypeScript



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

TypeScript Features



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

TypeScript Installation

Node.js

- npm install -g typescript

Plugins Available by default for

- Visual Studio 2015 & 2017
- Visual Studio Code

Plugins available for

- Eclipse
- Atom
- Sublime
- WebStrome
- Many more

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

TypeScript Types

Boolean

- **let** isAvailable: boolean = false;

Number

- **let** age: number = 16;
- **let** hex: number = 0xf00d;

String

- **let** name: string = “Anil";

Array

- **let** list: number[] = [1, 2, 3];
- **let** list: Array<number> = [1, 2, 3];

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

TypeScript Types

Enum

- **enum** Color {Red, Green, Blue}
- **let** c: Color = Color.Green;

Any

- **let** x: any = 4;
- x = “hello”

Tuple

- let data: [number, string];
- data = [1, "Anil"];

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

TypeScript Types

void

```
function foo(): void {
   console.log(“foo");
}
```

Null and Undefined

- var x: string = null;
- var y:string= undefined

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

Interfaces

- Interfaces are a powerful way of defining contracts.
- Example

```
interface Vehicle{

    name: string;
    speed: number;
    gear?: number;

    applyBrakes(decrement: number): void;
}
```

- Interfaces can extend interfaces
  - (keyword extends)
- Classes implements interfaces
  - (keyword implements)

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

Classes

- Traditional JavaScript uses functions and prototype-based inheritance to build up reusable components.
- Starting with ECMAScript 2015, also known as ECMAScript 6, JavaScript introduces the object-oriented class-based approach.
- TypeScript supports classes that compile down to JavaScript
  - Works across all major browsers and platforms
  - Without having to wait for the next version of JavaScript.

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

Classes

- Example

```
class Car implements Vehicle{
    constructor(public name: string, public speed: number, public
    gear: number){

    }
    applyBrakes(decrement: number): void{

    }
}
```

- Modifiers supported
  - public, private, protected
- Can have constructors
- Supports Properties
- Supports static members
- Supports inheritance
- Classes and methods can be abstract

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

TypeScript: Modules

- Starting with the ECMAScript 2015, JavaScript has a concept of modules.
- TypeScript shares this concept.
- Modules are executed within their own scope
  - Not in the global scope

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

Modules

- Use the import and export statements.

```
let foo = function(){
   //some code
}


export default foo;
```

one.js

```
import foo from ‘./one’;

foo();
```

two.js

```
import bar from ‘./one’;

bar();
```

three.js

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

Modules

```
export let foo = function(){
   //some code
}


 export let bar = function(){
    //some code
 }
```

one.js

```
import {foo, bar} from ‘./one’;


 foo();
 bar();
```

two.js

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

NPM Package vs. Module

Module

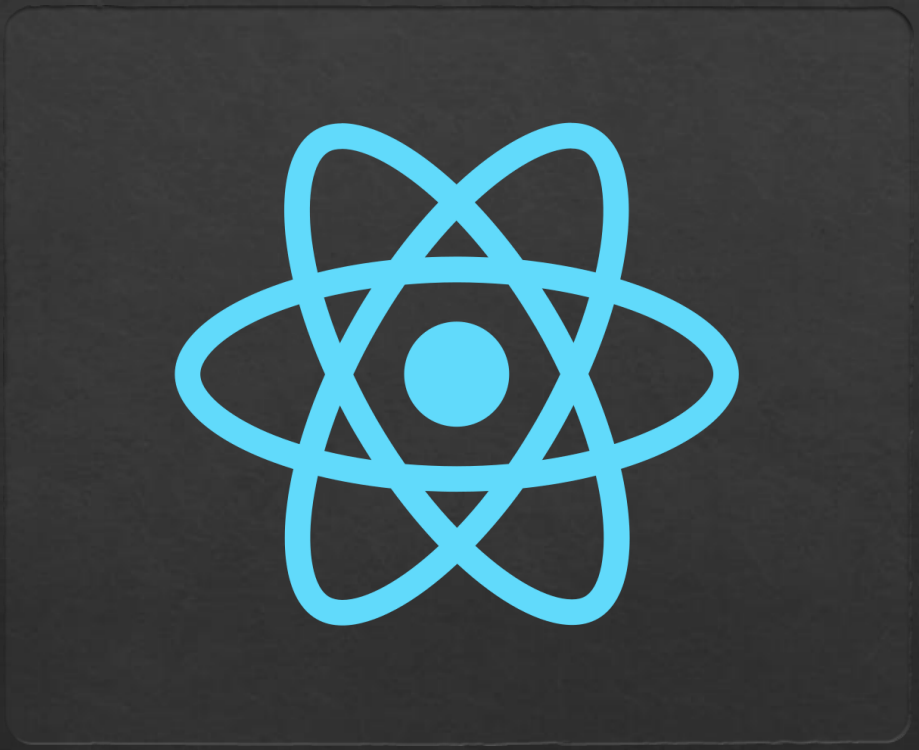
- Single JavaScript file with some reasonable functionality

Package

- A directory with one or more modules
- package.json file contains metadata about the package

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

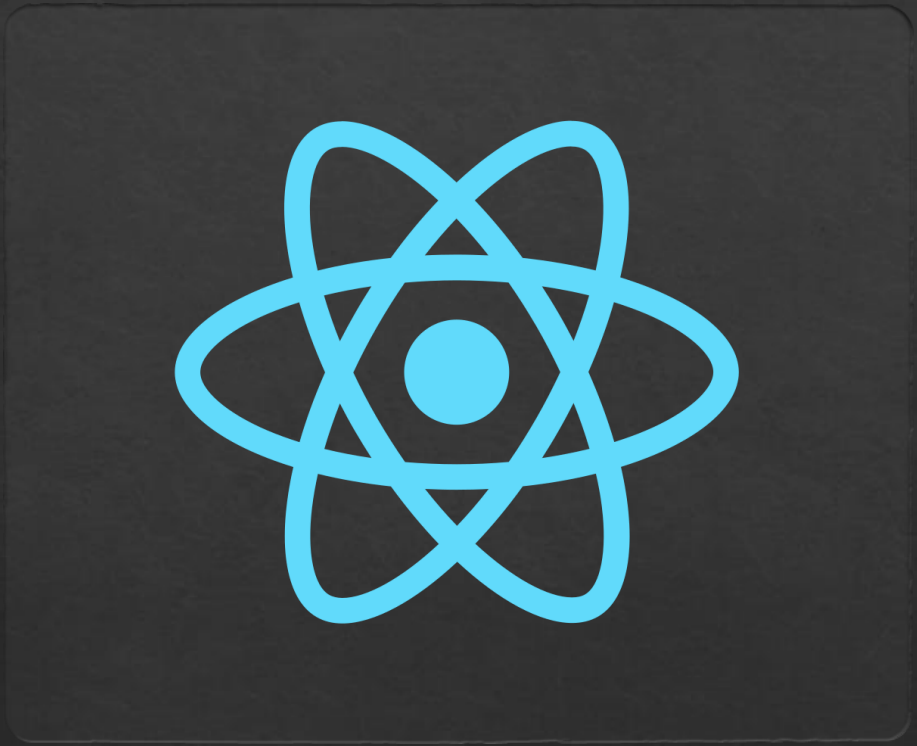
What is React?



- 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***.

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

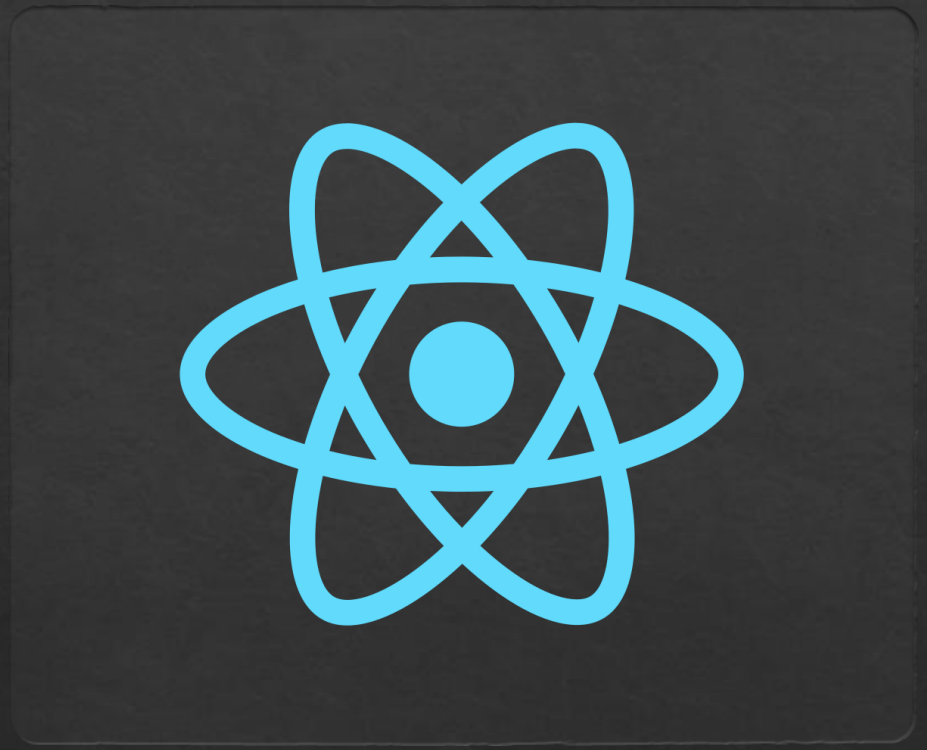
Why React?



- 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.

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

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

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

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.

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

Alternatives

Angular

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

Vue

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

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

Getting Started

The Two React libraries

- React
- ReactDOM

Babel

- The compiler for writing next generation JavaScript.

JSX

- A JavaScript extension syntax allowing quoting of HTML

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

Getting Started Playgrounds

CodePen

- https://codepen.io/pen

CodeSandbox

- https://codesandbox.io/s/new

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

Creating a React Project

Build Process: 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

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

Build Process: How?

Dependency Management Tool

- npm and yarn

Bundler

- WebPack

Complier

- Complies ES6 to ES5/ES3
- Babel + Presets

Development Server

- Web Server for development

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

Build Process: Tool

- The React team provides a tool 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

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

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**

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

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**

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

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**

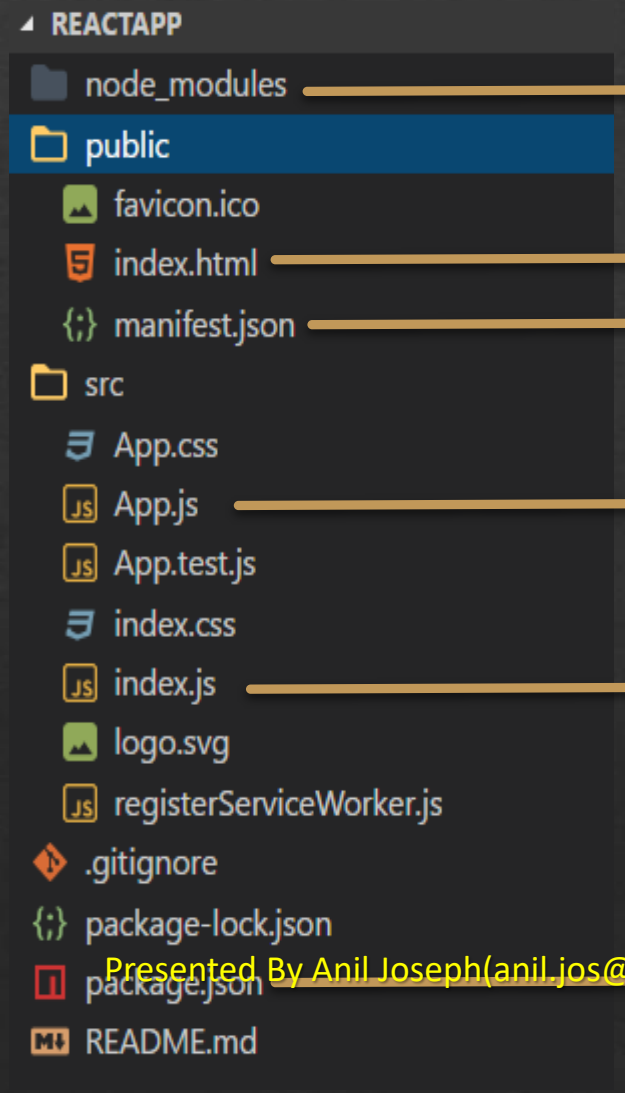
3

Start the Application

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

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

Project Structure



Folder with all downloaded dependencies

The page displayed in the browser

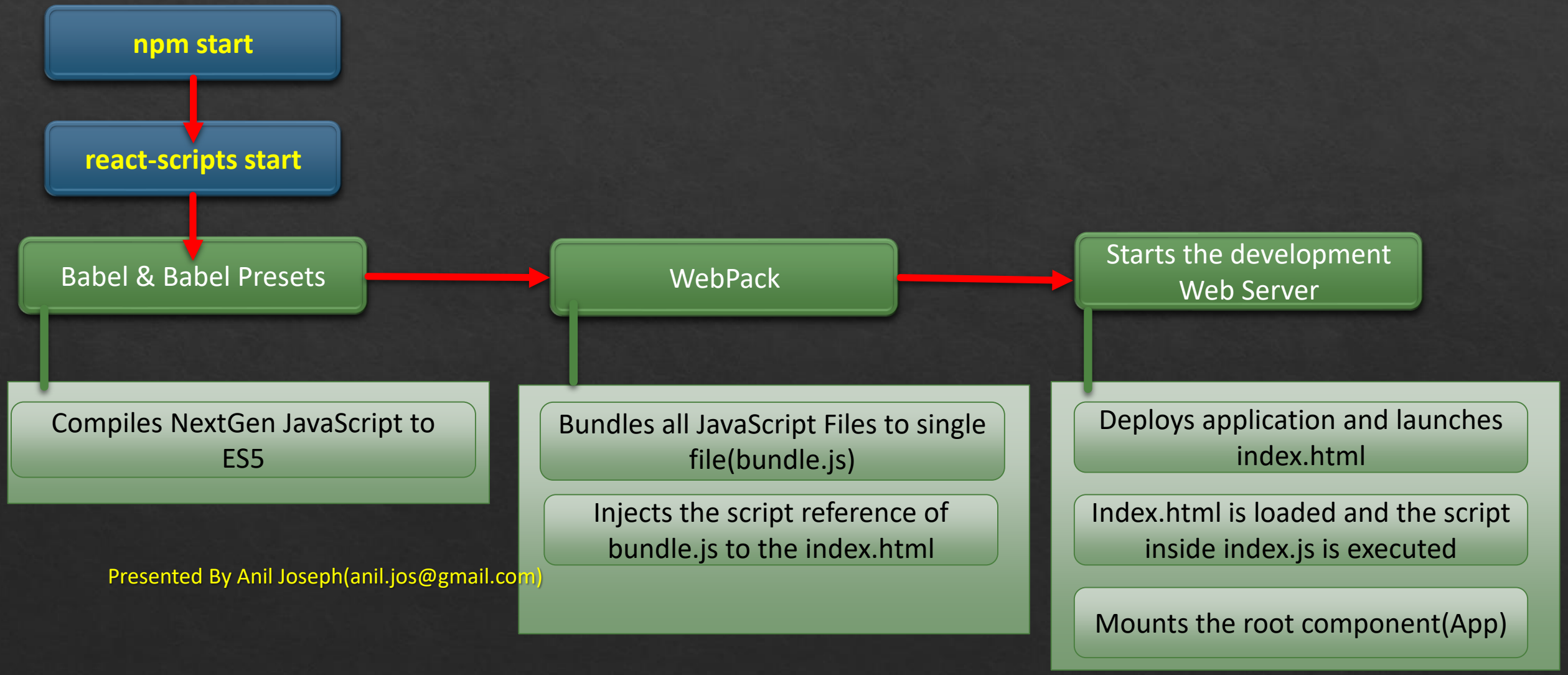
The Web Application Manifest (for installable web applications)

The root component

Initial Script executed on page load. Renders the root component

NPM configuration file, contains the dependencies and scripts

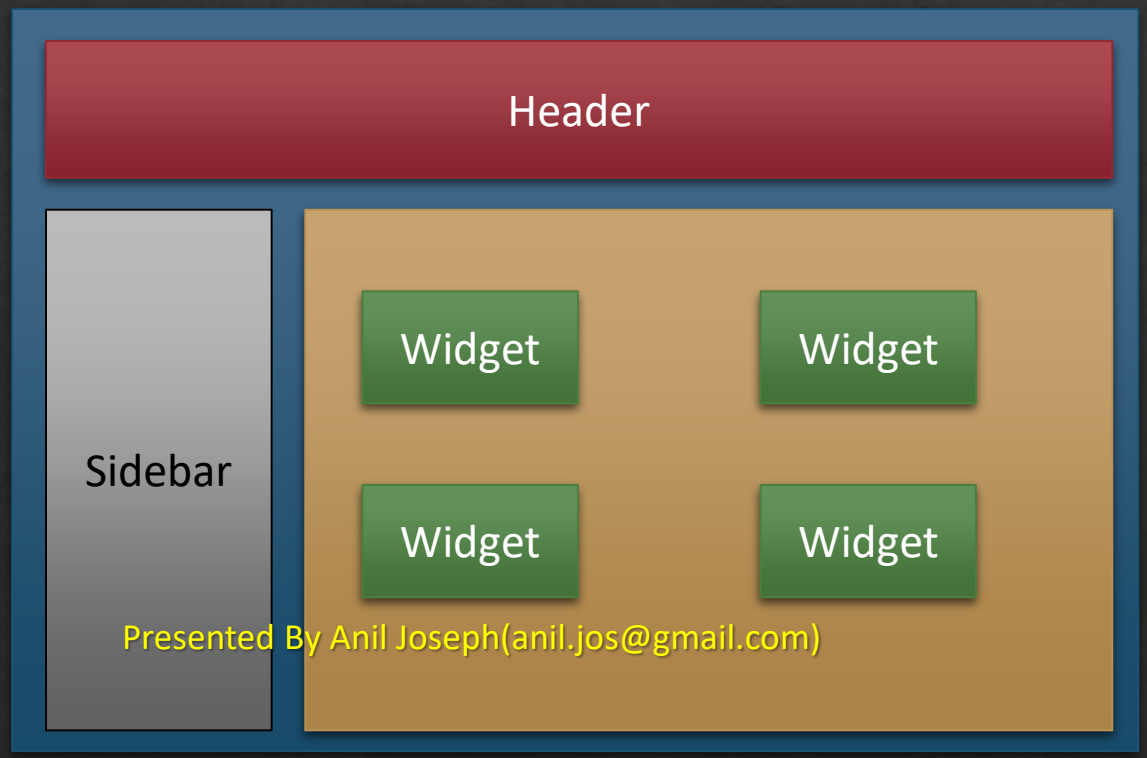
Project Execution

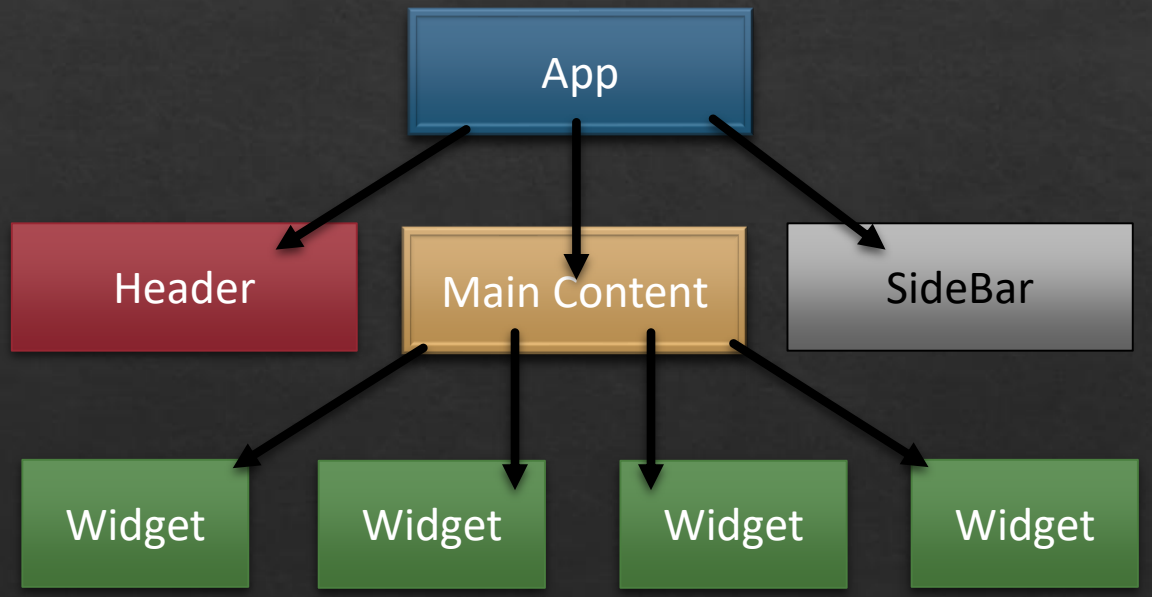


Presented By Anil Joseph(anil.jos@gmail.com)

React Components

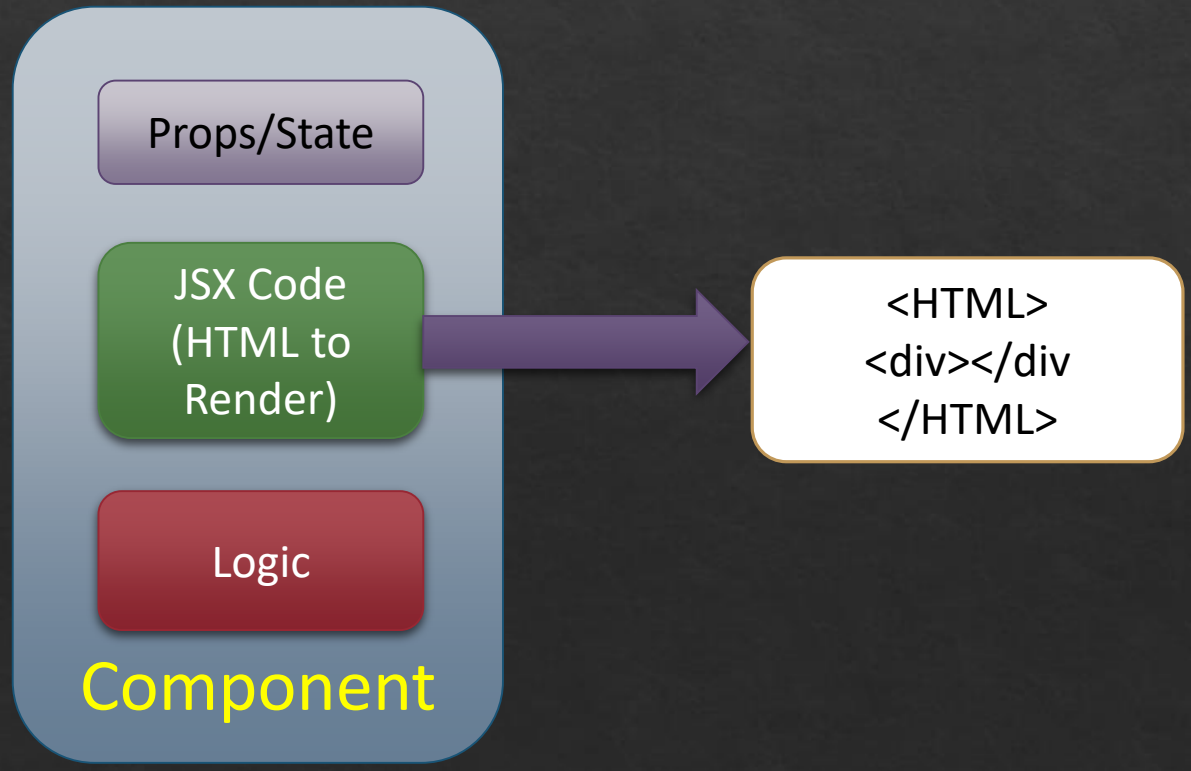
- Components are the **core building blocks of React applications**
- Creating a React Application is all about creating components.
- A React app can be depicted as a **component tree**





React Components

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



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

Types of Components

Functional

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

Class Based

- Containers
- Stateful

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

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

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

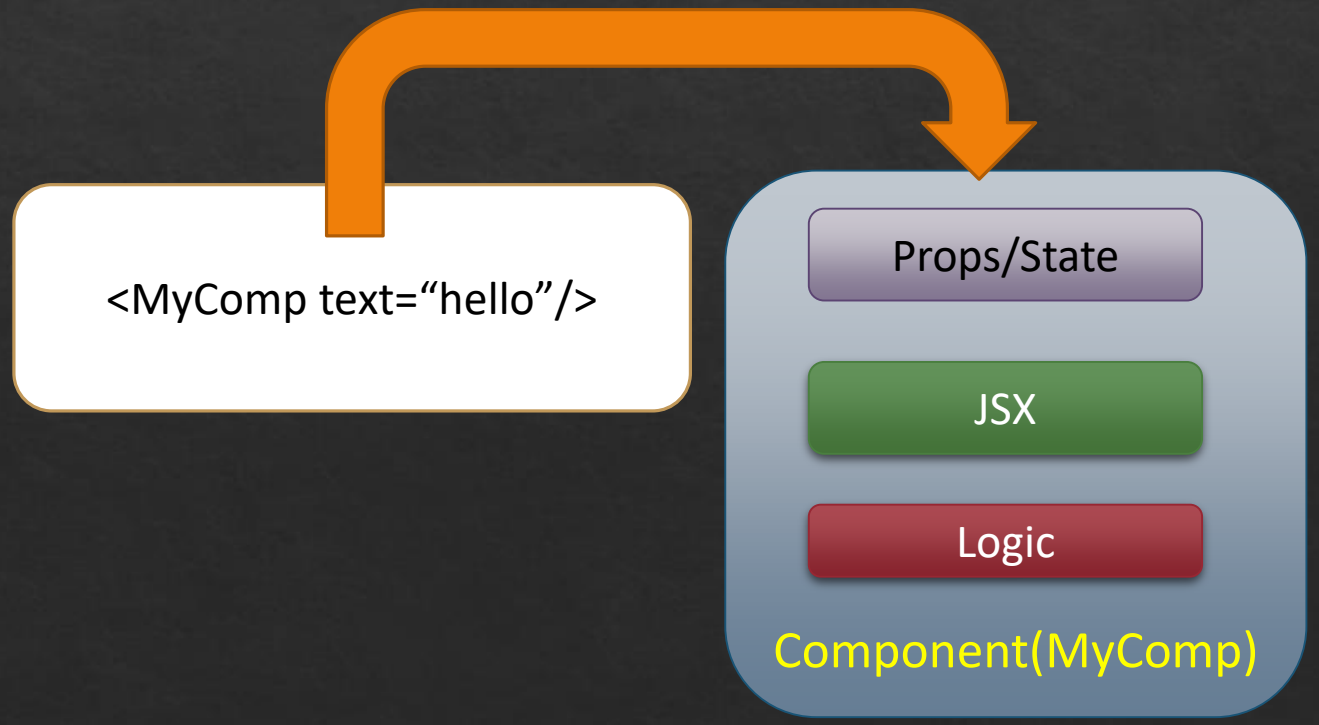
Components: Dynamic Content

- 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() }

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

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.***



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

Props Types

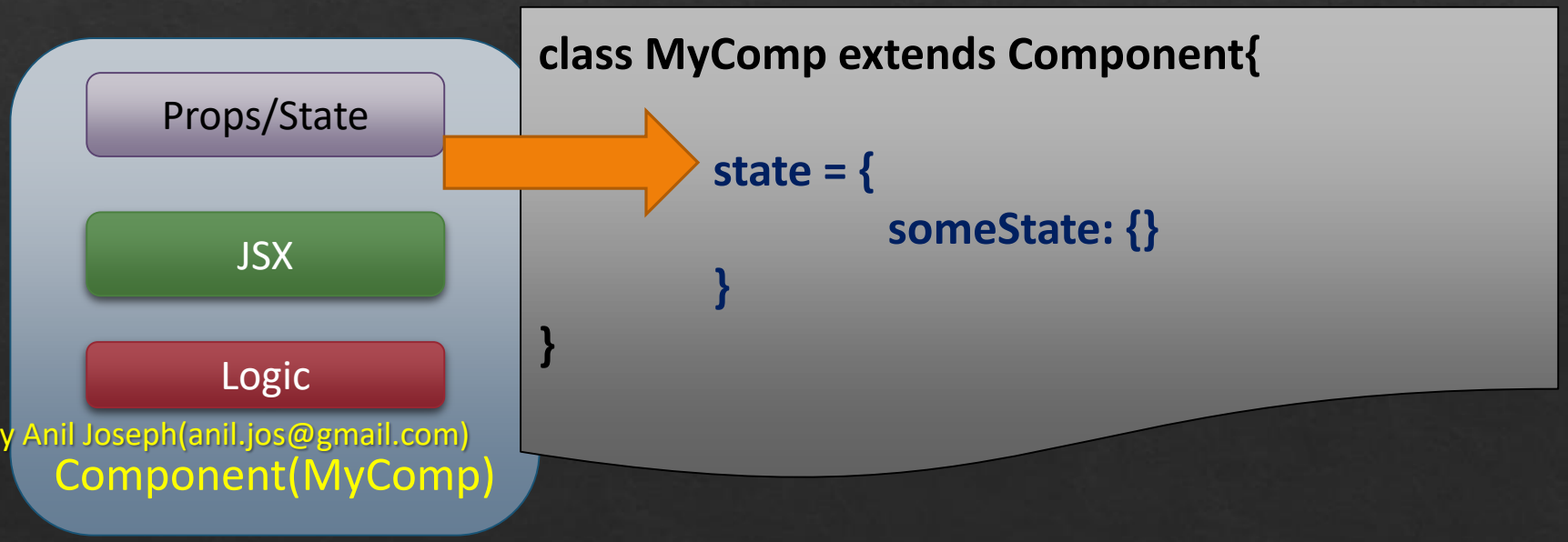
Use Prop-Types for Type Checking the Component Properties

npm install prop-types

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

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.**



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

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.

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

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

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

Component Lifecycle

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

Component Lifecycle (Mount)

constructor

- Setup state
- Don’t Cause Side-Effects

componentWillMount

- Update State
- Don’t Cause Side-Effects
- Configuration

render

- Prepare and Struture JSX code

Render Child Components

ComponentDidMount

- Cause side effects
- **Initialize anything that relies on the DOM**

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

Component Lifecycle (Update)

componentWillReceiveProps(nextProps)

- Sync State to Props
- Don’t Cause Side-Effects

shouldComponentUpdate(nextProps,nextState)

- Decide whether to Continue or Not
- Don’t Cause Side-Effects

componentWillUpdate(nextProps, nextState)

- Sync State to Props
- Don’t Cause Side-Effects

render()

Update Child Component Props

componentDidUpdate()

- Cause Side-Effects
- Don’t Update State

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

Component Lifecycle (Unmount)

componentWillUnmount

- Remove Listeners
- Cancel Active network calls
- Invalidate Timers

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

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.

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

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

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

axios methods

axios.request({config})

axios.get(url, {config})

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

axios.delete(url, {config})

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

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

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';

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

React Hooks

React hooks was introduced in version 16.8

Many React features like state, lifecycle hooks etc. were available only with class-based components prior to 16.8.

Hooks let us use state and other React features in a functional component.

React team recommends the functional components over class-based since they can be highly optimized by the tools.

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

React Hooks

State Hooks(useState)

- Equivalent of state in class-based components

Effect Hooks(useEffect)

- Used to perform side-effects in a function component
- Equivalent to lifecycle hooks in class-based components

Context Hooks(useContext)

- Used to access the React Context;

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

React Hooks

Callback Hooks(useCallback)

- Returns a memoized callback.
- Used to optimize the components

Memo Hooks(useMemo)

- Returns a memoized value.
- Used to optimize the components

Ref Hooks(useRef)

- Returns a mutable ref object

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

Debugging

- Error Messages
  - Messages generated by React during development mode
- Browser Developer Tools
- React Tools
  - Tools for Chrome and Firefox
- Error Boundaries
  - Error boundaries are React components that **catch JavaScript errors anywhere in their child component tree**
  - **Log errors**
  - **Display a fallback UI**

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

Immutability

- React components have the concept of *state*.
- Whenever state changes React re-renders the component.
- If state complicated (like with nested objects) it'd be hard to check whether your state changed or not.
  - Have to use *deep equality check*
- Immutability
  - *Whenever your object has to be mutated , don't mutate it*
  - *Create a copy*
- Immutability makes it easier to detect changes and update the UI

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

Virtual DOM

The Virtual DOM (VDOM)

- Where a “virtual”, representation of a UI is kept in memory
- This is synced with the “real” DOM.
- This process is called reconciliation.

Reconciliation

- When the render() function is called it creates a tree of React elements.
- On the next update render() will return a different tree
- React then figures out how to efficiently update the UI to match the most recent tree.
- Uses the Diff algorithm

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

Higher-order Components

- A Higher Order Component is just a React Component that wraps another one.
- Higher Order Components is a Pattern used extensively with React
- Uses
  - Code reuse, logic and bootstrap abstraction
  - Render Highjacking
  - State abstraction and manipulation
  - Props manipulation
- Its basically a functions that returns a class component with the Wrapped Component.

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

Single Page Applications(SPA)

Single-Page Applications are applications that load a **single** HTML **page** and dynamically update that **page** as the user interacts with the **app**.

SPAs use AJAX and HTML5 to create fluid and responsive Web **Applications**

Do not have constant **page** reloads.

Much of the work happens on the client side, in JavaScript.

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

Routing

- Routers allow to navigate between the different views in the application using JavaScript on the client-side.
- Navigations are updated to the browser history which allows to go back and forward
- Usage of path and search parameters are allowed
- React does not provide any API for routing.
- Many other Libraries available which integrates with React
  - React-router(The de-facto standard)
  - Director
  - Aviator
  - Finch

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

React Router v4

- React Router is a collection of **navigational components** that compose declaratively with your application.
- Installation
  - npm install react-router-dom

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

React Router Components

<BrowserRouter>

- A <Router> that uses the HTML5 history to keep your UI in sync with the URL.

<HashRouter>

- A <Router> that uses the hash portion of the to keep your UI in sync with the URL.

<Route>

- The Route component is perhaps the most important component in React Router

<Link>

- Provides declarative, accessible navigation around your application.

<NavLink>

- A special version of the <Link> that will add styling attributes to the rendered element when it matches the current URL.

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

State Management

React Context

- Available from React 16.3

React Redux

- Library to manage state

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

Types of State

Local UI State

- Show/Hide UI
- Handled By the Component

Persistent State

- Orders, Blogs
- Stored on Server, can be managed by Redux or React Context

Client State

- IsAuntenticated, Filter Information
- Managed by Redux or React Context

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

Redux

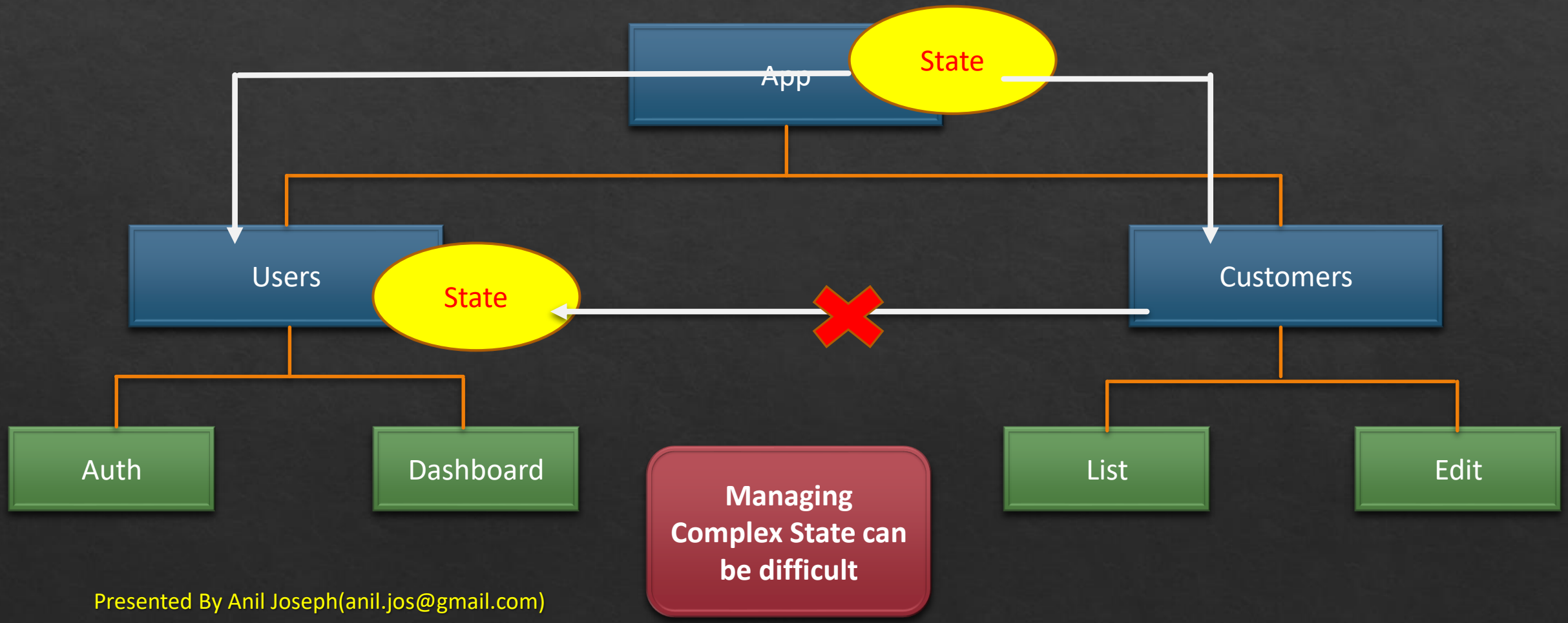
Redux is an open-source JavaScript library designed for managing application state.

It is primarily used together with React or Angular for building user interfaces.

Redux was built on top of functional programming concepts.

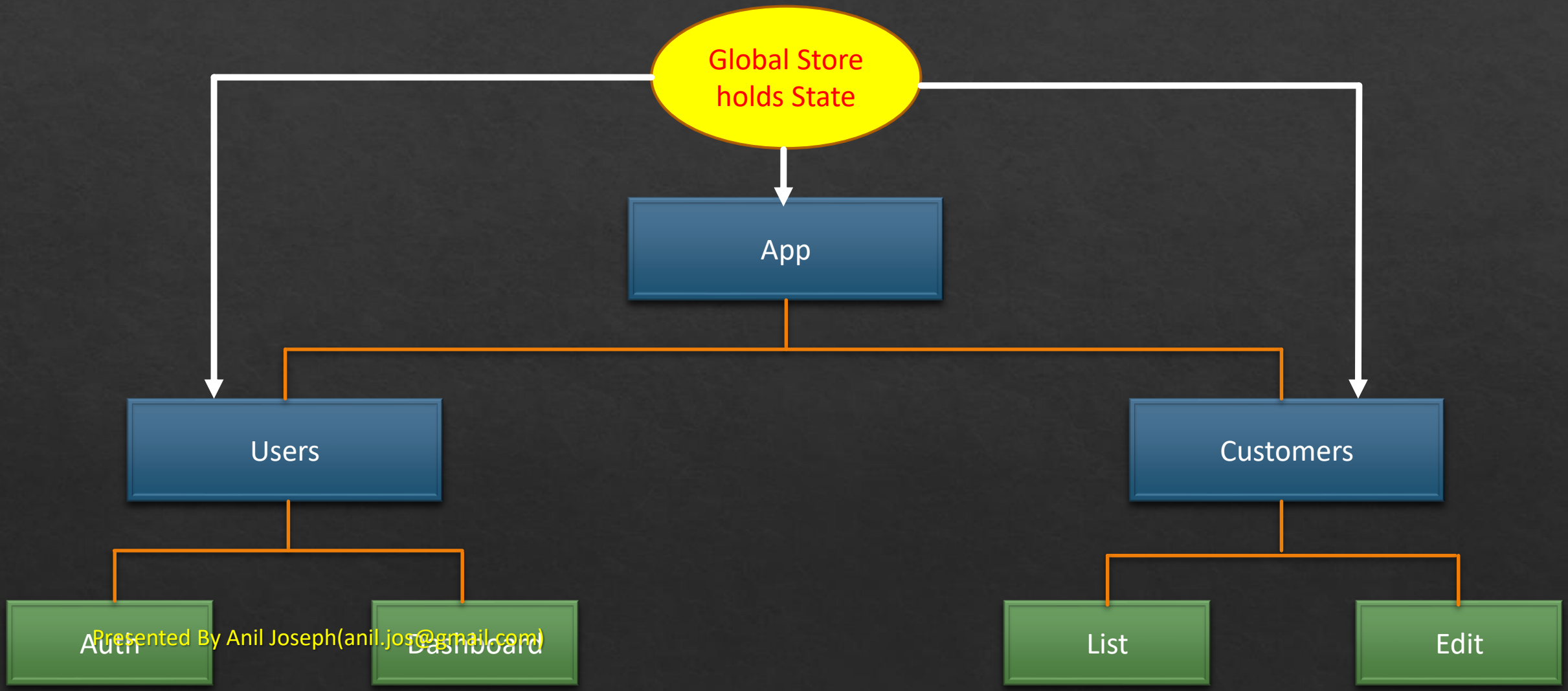
Presented By Anil Joseph(anil.jos@gmail.com)

Why Redux?



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

Redux State Management



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

Redux Parts

Store

- Store is the object that holds the application state
- The entire state is represented by a single store.

Actions

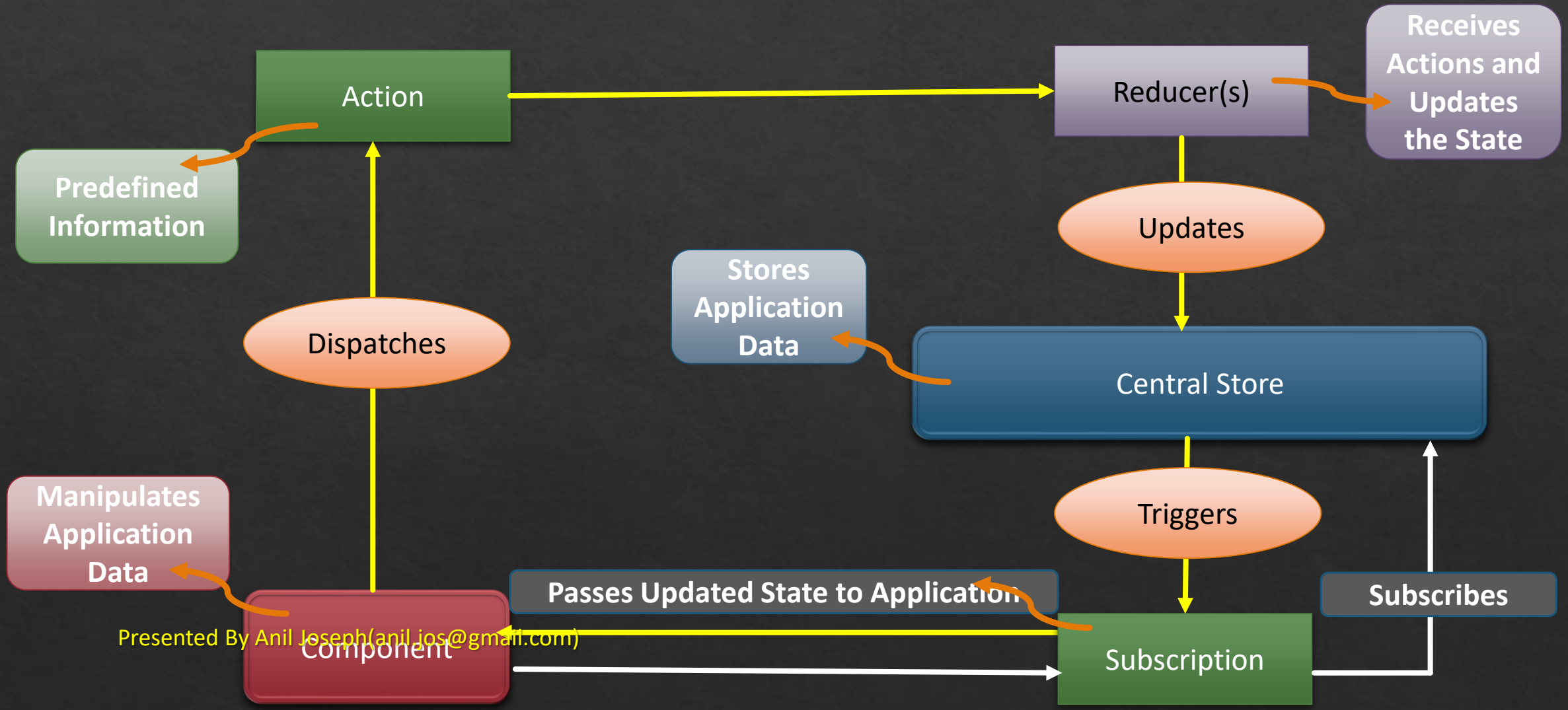
- Actions are events.
- Actions send data from the application (user interactions, internal events such as API calls, and form submissions) to the store.

Reducers

- In Redux, reducers are functions (pure)
- Its takes the current state of the application and an action and then return a new state.

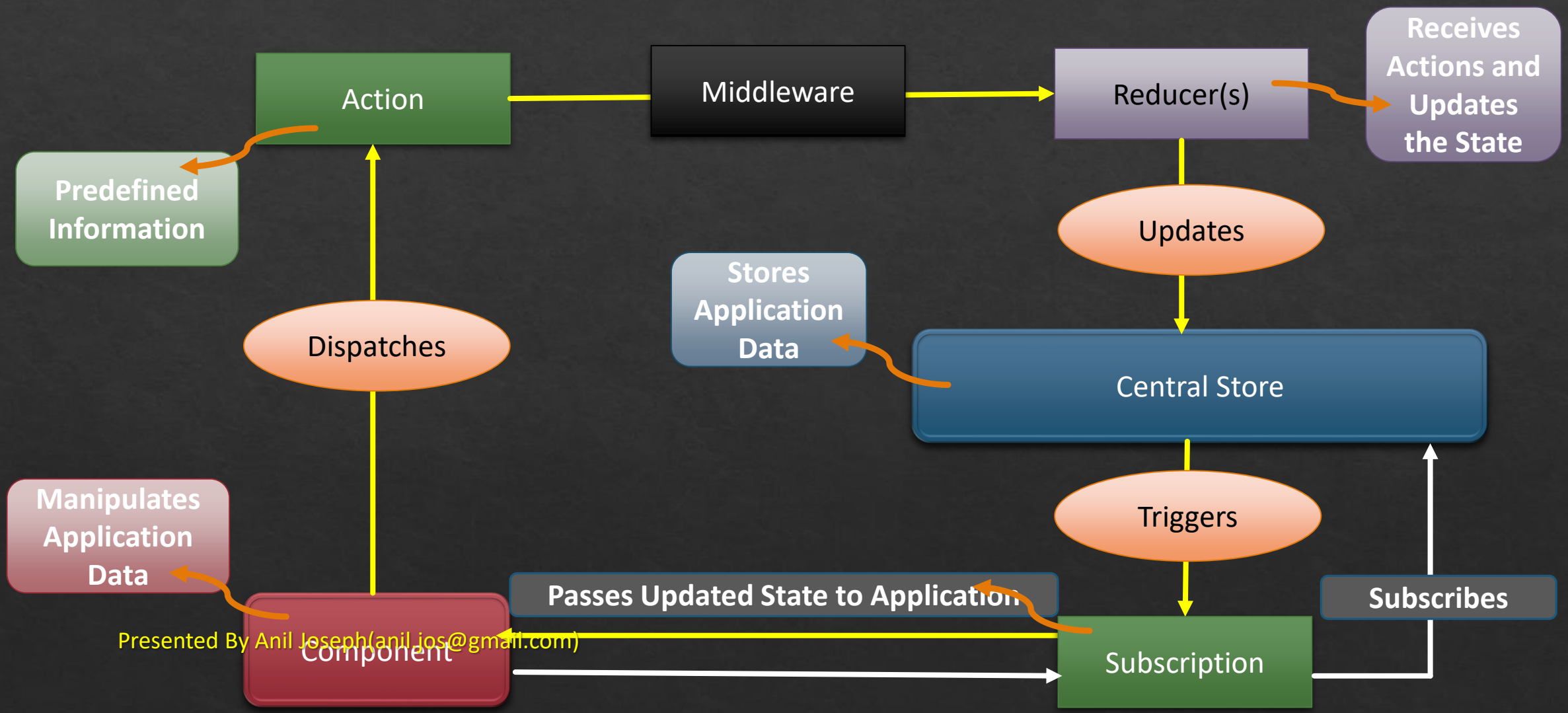
Presented By Anil Joseph(anil.jos@gmail.com)

Redux Flow



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

Redux Flow



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

Redux React

Redux react is a library that integrates Redux to a React Application

Comprises of Components & Functions

Installation

npm install react-redux

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

Error Boundaries

- Error boundaries are React components that catch JavaScript errors anywhere in their child component tree
  - Used to log errors
  - Display a fallback UI
- Error boundaries do not catch errors for:
  - Event handlers
  - Asynchronous code
  - Server side rendering
  - Errors thrown in the error boundary itself
- A class component becomes an error boundary if it defines(either one)
  - componentDidCatch
  - static getDerivedStateFromError()

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

React Context

- Context provides a way to pass data through the component tree without having to pass props down manually at every level.
- Example of props to be passed down
  - Theme
  - Locale
  - Authenticated user
- API
  - React.createContext

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

Code Splitting

- Code splitting allows you to split your app into separate bundles which your users can progressively load.
- API
  - React.lazy
  - Suspense

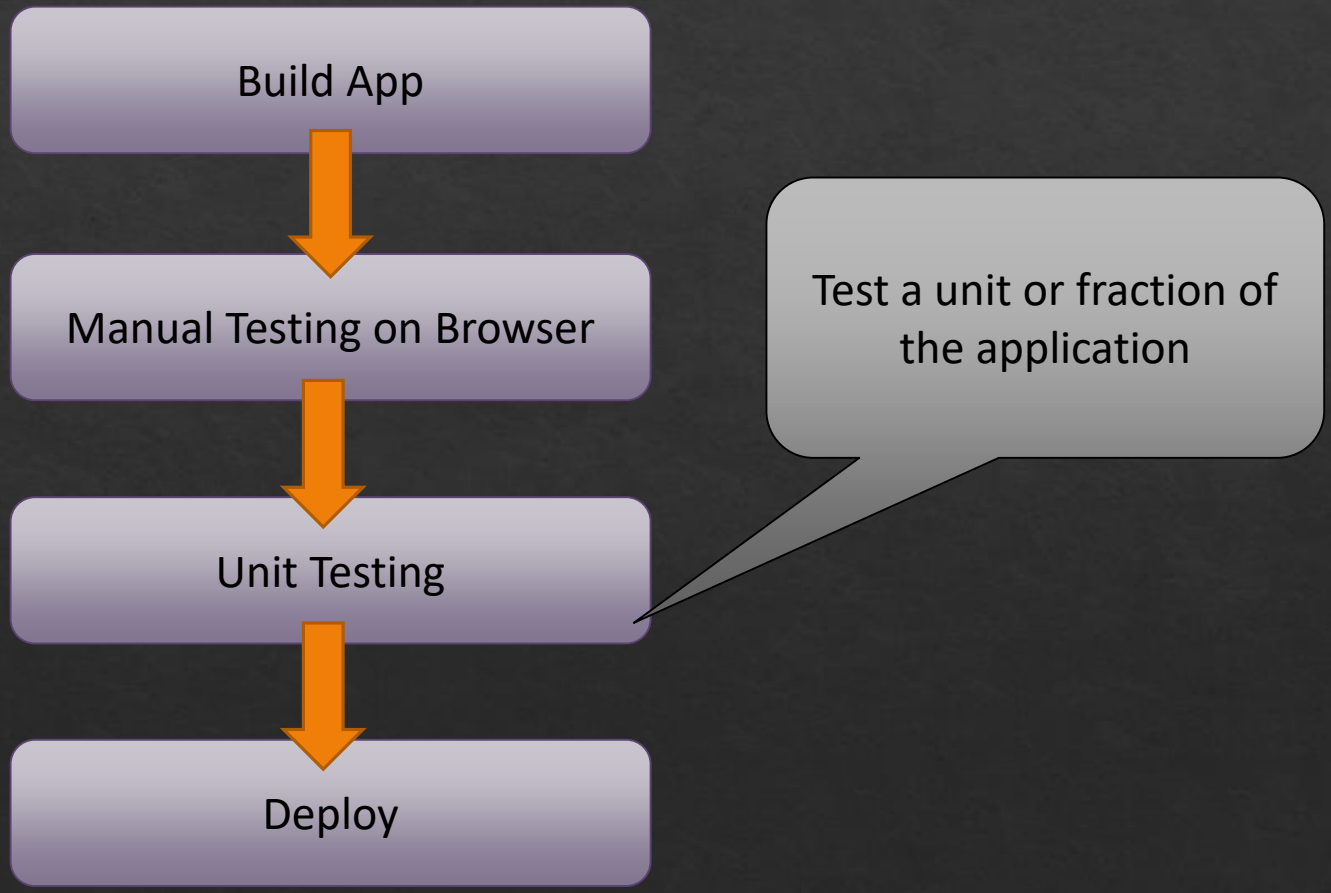
Presented By Anil Joseph(anil.jos@gmail.com)

Using Sass

- Sass is a style sheet language
- Install for create-react-app
  - npm i node-sass
- Rename .css to .scsss

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

Testing



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

Testing Tools

Testing API
&
Test Runner

Write the Unit Test.
Executes the Unit Tests.

Jest
(Built over Jasmine)

Test Utilities

Simulate the React App

react-test-renderer

enzyme

testing-library/react

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

Jest

A testing library and test runner with handy features

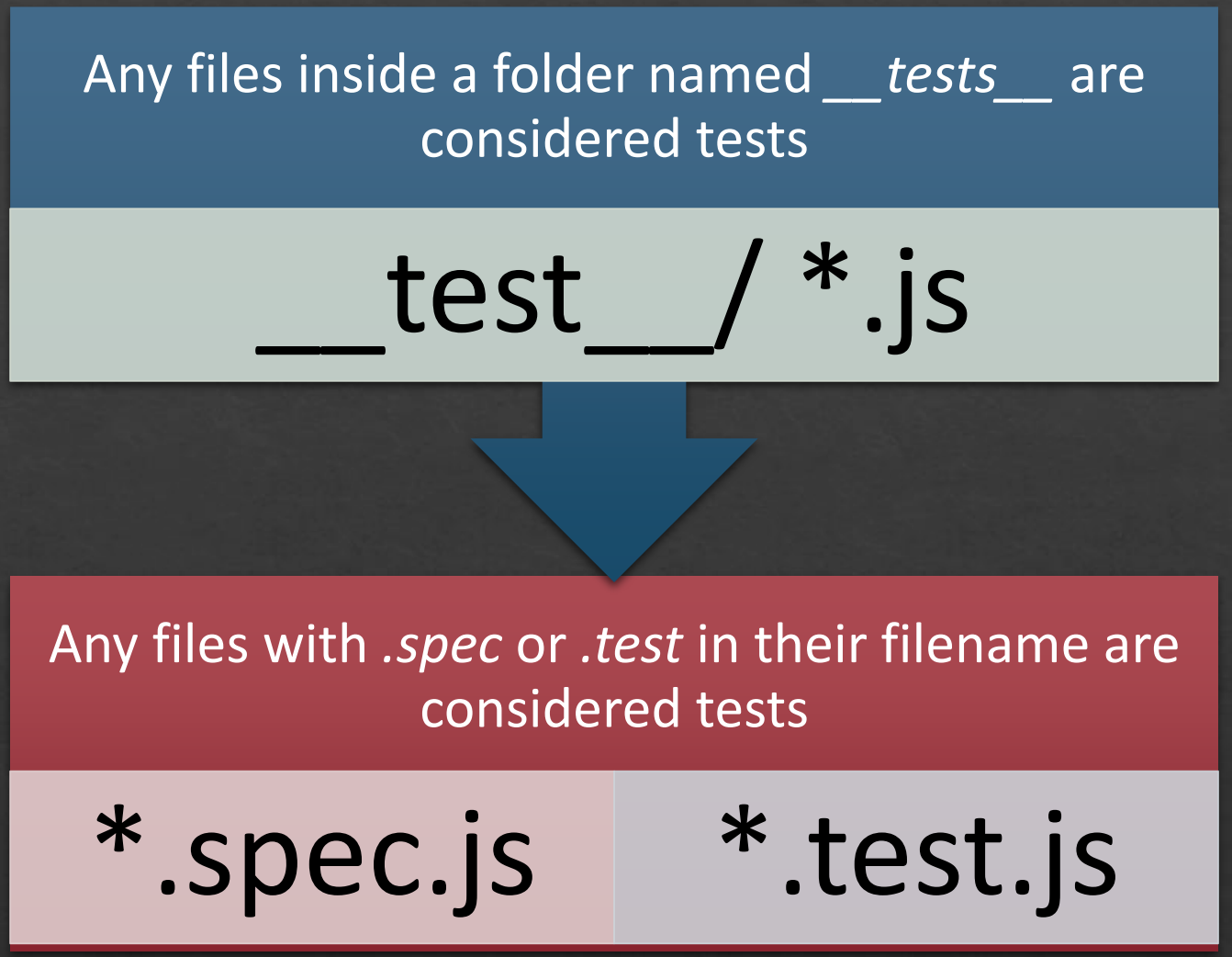
Created by the members of the React team and the recommended tool for unit testing react

Built on top of Jasmine/Mocha

Additional features like mocking and snapshot testing

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

Jest: Identifying Test files



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

Jest Global Methods

it

- Method which you pass a function to, that function is executed as block of tests by the test runner.
- Alias name: test

describe

- An optional method for grouping any number of *it* or *test* statements
- Alias name: suite

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

Setup & Teardown Global functions

beforeEach	BeforeEach runs a block of code before each test
afterEach	Runs a block of code after each test
beforeAll	BeforeAll runs code just once, before the first test
afterAll	Runs a block of code after the last test)

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

What to Test?

Test Isolated Components

Test Conditional Outputs

Don’t Test Library Functions

Don’t Test Complex Connection

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

Installation

```
npm install enzyme react-test-renderer enzyme-adapter-react-16
```

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

End to End Test

End-to-end testing is a methodology used to test whether the flow of an application is performing as designed from start to finish.

The purpose of carrying out end-to-end tests is to identify system dependencies and to ensure that the right information is passed between various system components and systems.s

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

E2E Tools

Cypress

Pupeeter

Selenium Webdriver

Nightwatch.js

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

React UI Components

- There are many UI Libraries available in the React Eco space
- Libraries
  - React-bootstrap
  - Material UI
  - Grommet
  - Ant Design React
  - React Desktop
  - React Belle

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

Deployment Steps

Set basePath of Router

<BrowserRouter basename="/app/">

Set homepage in package.json

"homepage": "/app"

Build and Optimize

npm run build

Server must Always serve index.html

Deploy build artifacts to server

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

Resources

React: https://reactjs.org/

Awesome React: https://github.com/enaqx/awesome-react

Tutorial: https://egghead.io/courses/start-learning-react

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