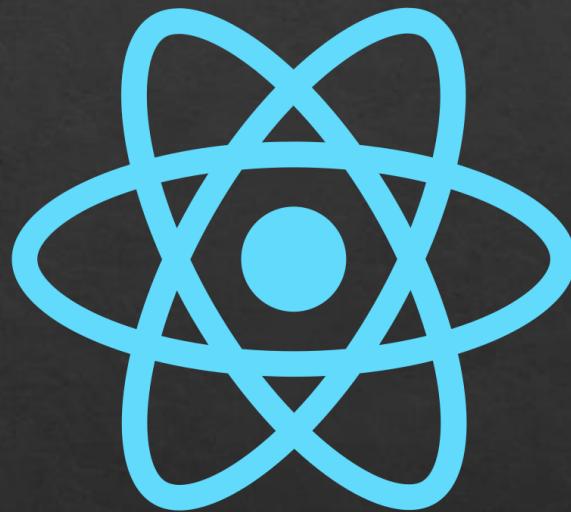
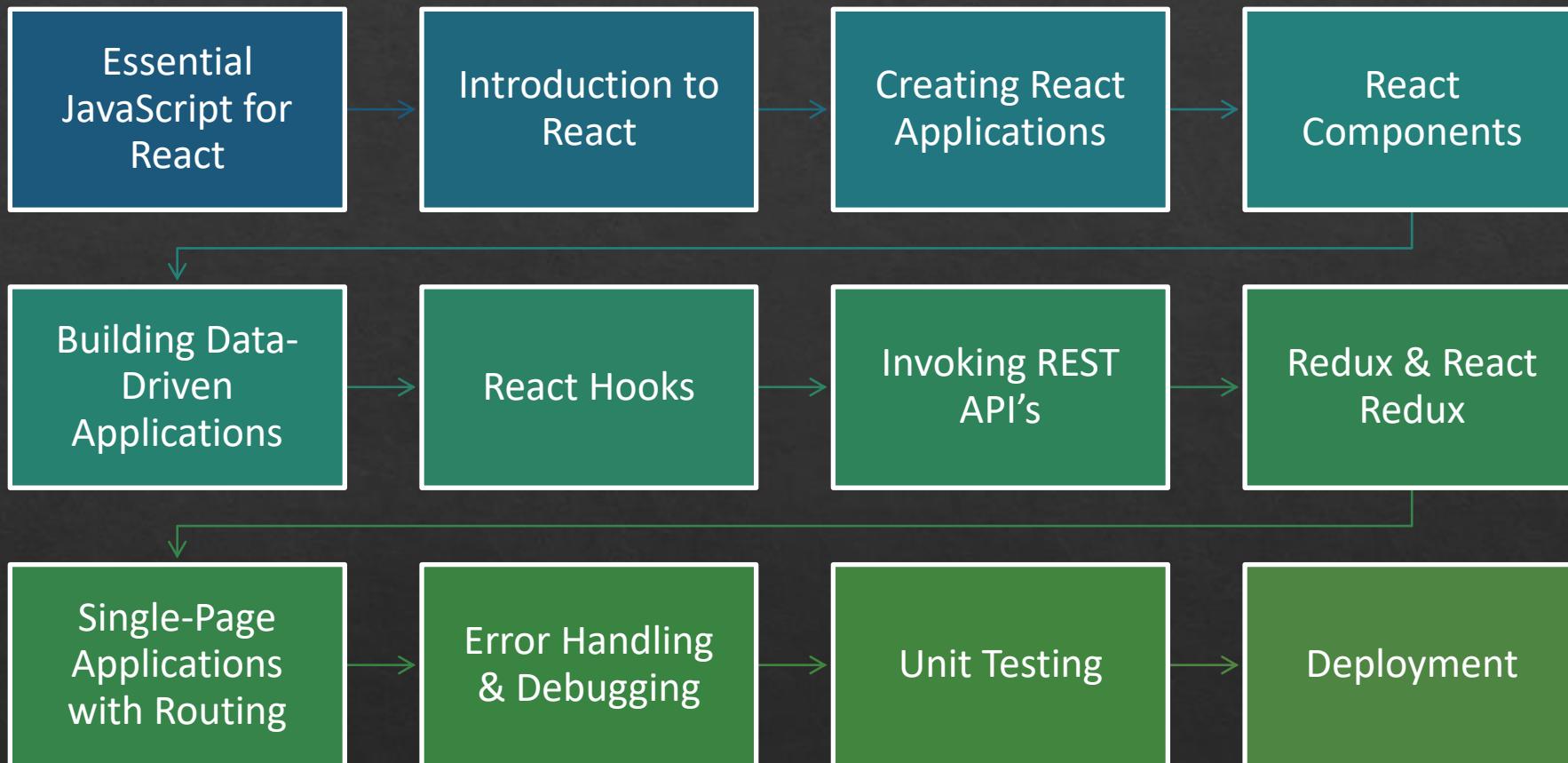


# React



ANIL JOSEPH

# Agenda



# Software

Node.js & NPM

Yarn  
(Optional)

**Visual Studio**

Browsers(Chromium)

# JavaScript

---

An interpreted language

---

Dynamic

---

Object-oriented

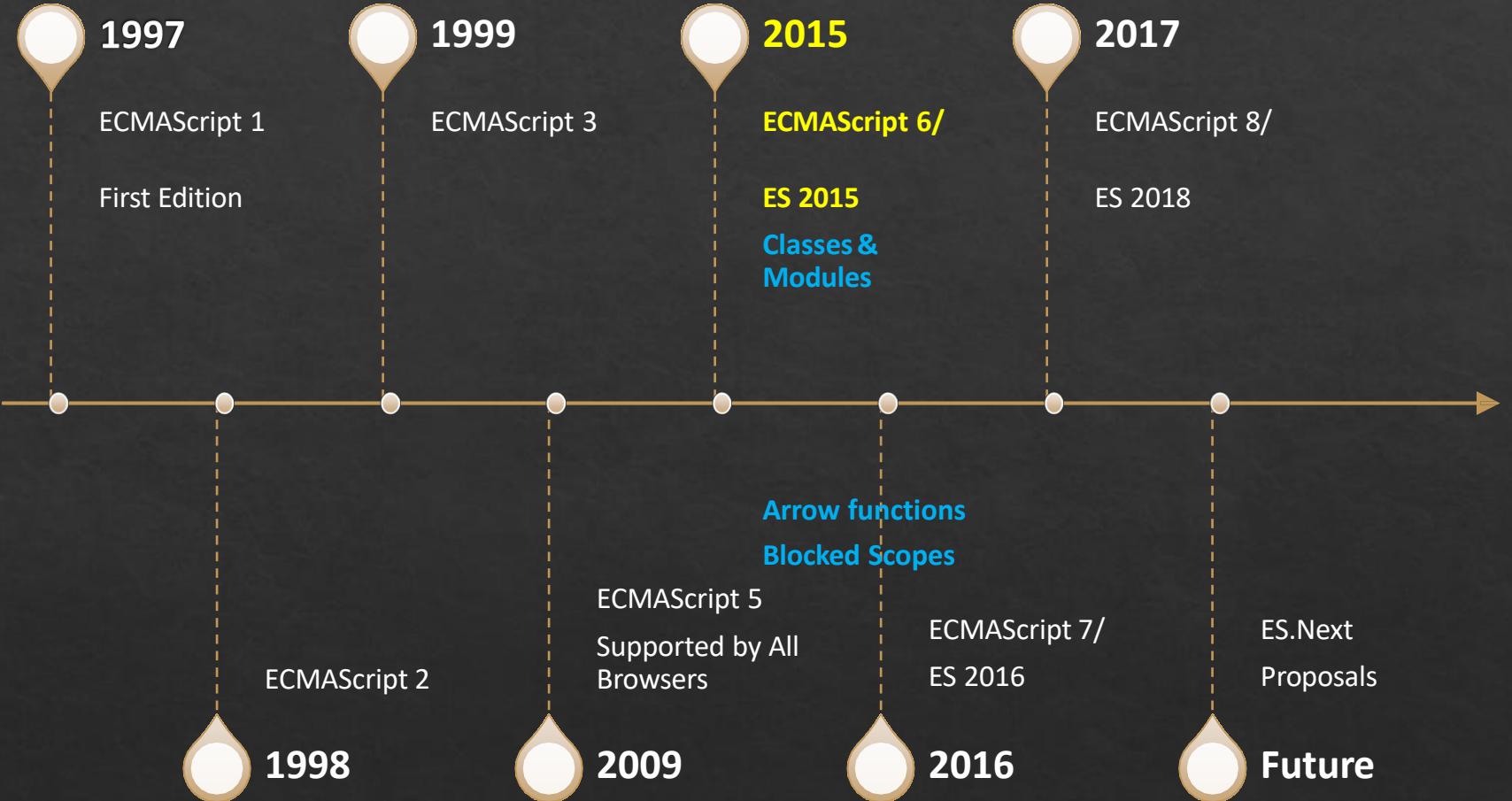
---

Supports Functional style of programming

---

Available on the browsers and Node.js

# ECMAScript Versions



# ES6/7 Features (Used in React)

Block Scoped  
Constructs

Classes

Arrow functions

Spread and Rest  
Operator

Modules

# Block Scope

- ❖ Scopes determine the visibility of a declared variable, function or class
- ❖ ES5 scopes
  - ❖ global
  - ❖ function
  - ❖ lexical
- ❖ ES6
  - ❖ block
  - ❖ keywords: let & const
- ❖ let & const creates a block scope and are hoisted without initialization

# Arrow Function

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

# Spread Operator

- ❖ Allows an iterable such as an array expression or string to be expanded in places where zero or more arguments or elements are expected.

# ES6 Classes

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

# 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

# Modules

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

one.js

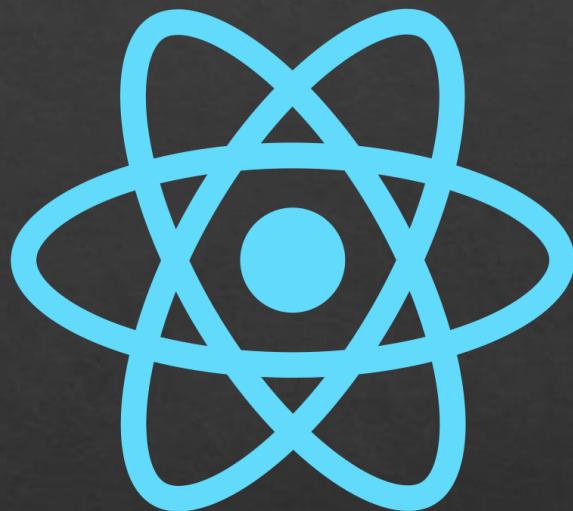
```
import {foo, bar} from './one';  
  
foo();  
bar();
```

two.js

# NPM Package vs Module

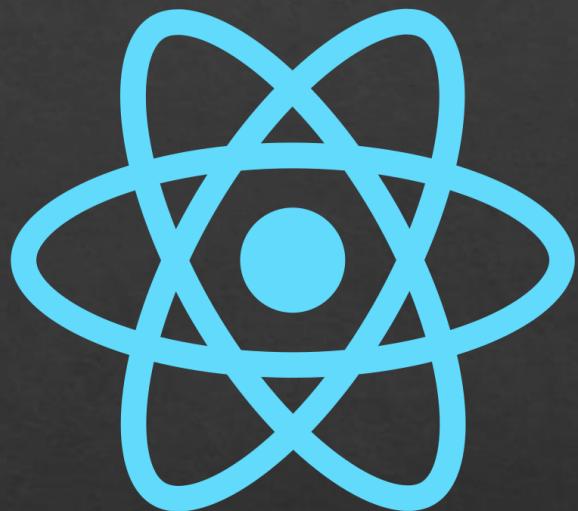
- ❖ Module
  - ❖ Single JavaScript file with some reasonable functionality
- ❖ Package
  - ❖ A directory with one or more modules
  - ❖ package.json file contains metadata about the package

# What is React?



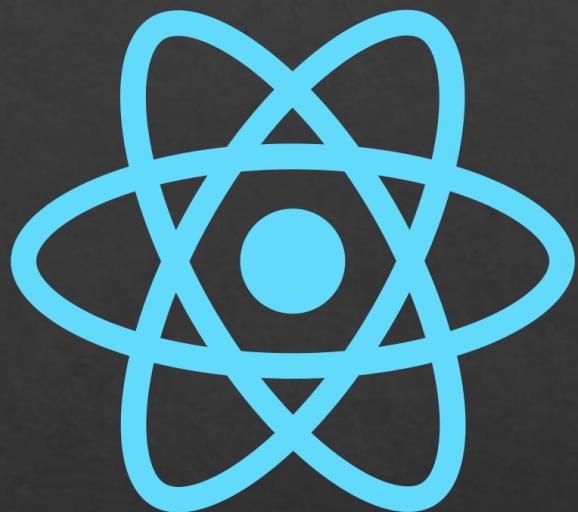
- ❖ A JavaScript library to build *User Interfaces*.
- ❖ Building applications for the *browser*.
- ❖ React allows to create custom HTML elements called *components*.
- ❖ Components are used to build *complex User Interface*
- ❖ Reacts promotes writing *maintainable, manageable and reusable code*.

# 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.

# Getting Started

## Two React Libraries

- React
- ReactDOM

## JSX

- A JavaScript extension syntax allowing quoting of HTML

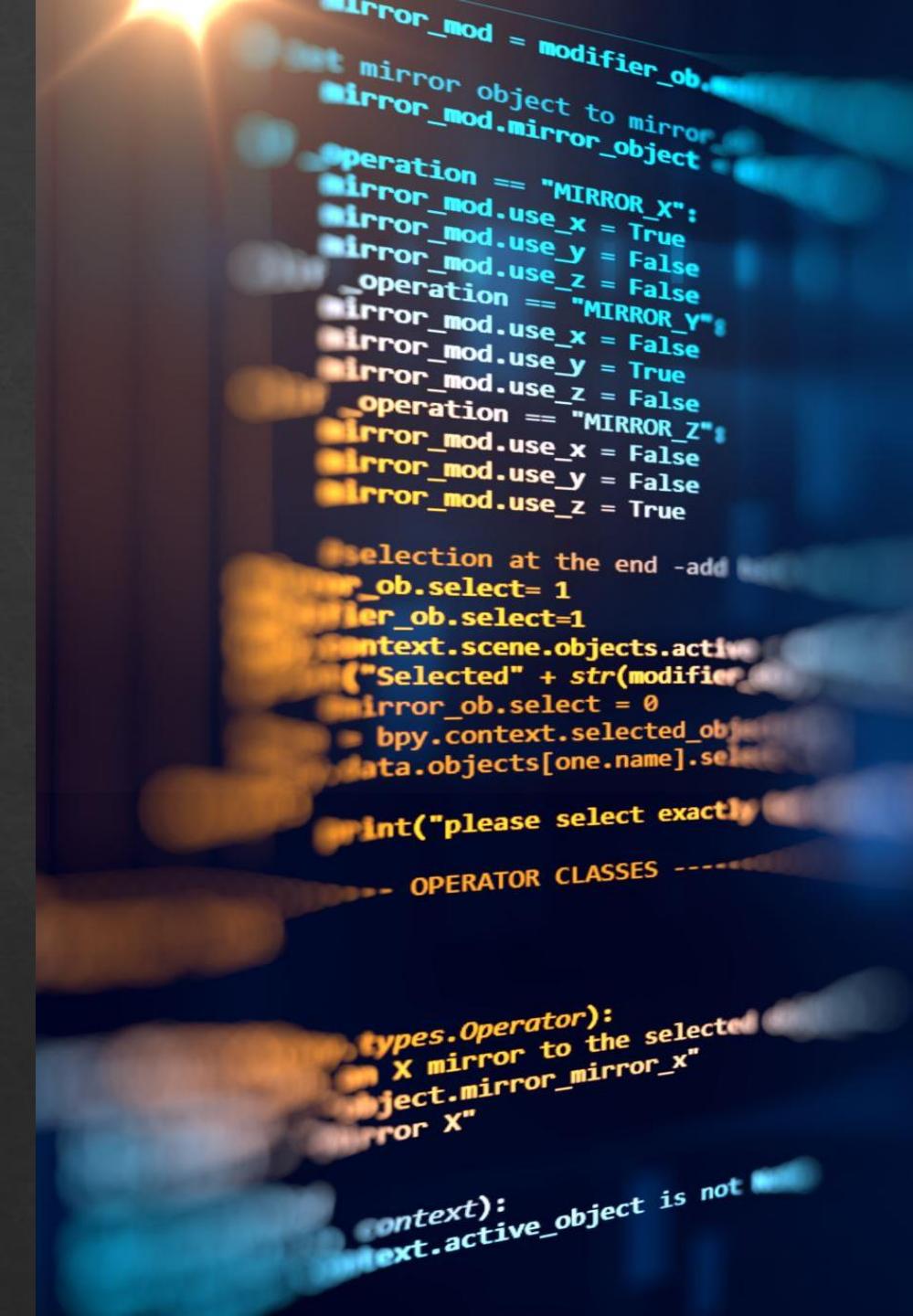
## Babel

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

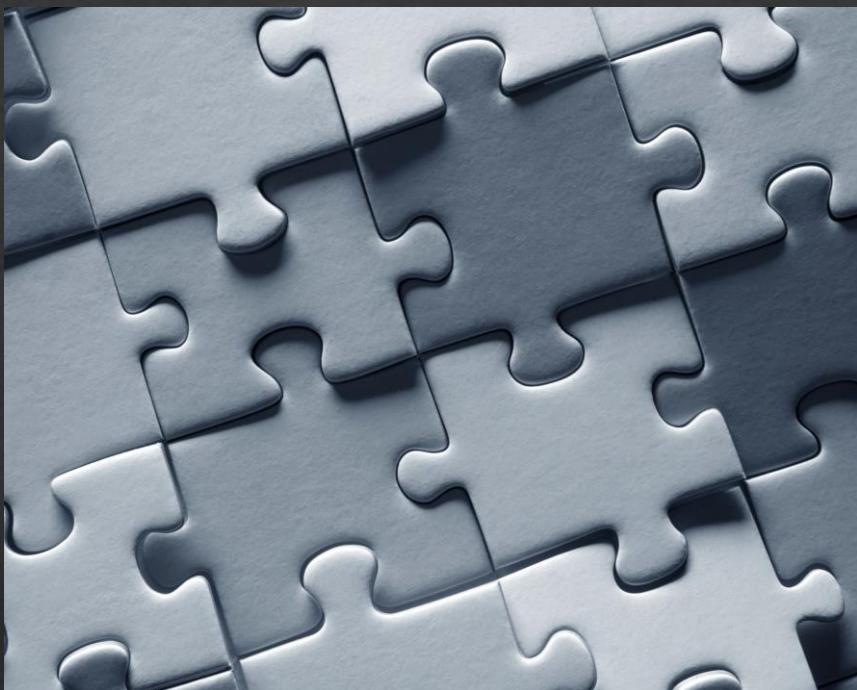
# Creating a React Project

# Tool Chains

- ❖ A toolchain is a set of programming tools that is used to perform a complex software development task or to create a software product.(Wiki)
- ❖ Using a toolchain provides a better developer experience
- ❖ Advantages of Tool Chains
  - ❖ Scaling to many files and components.
  - ❖ Using third-party libraries from npm.
  - ❖ Detecting common mistakes early.
  - ❖ Live-editing CSS and JS in development.
  - ❖ Optimizing the output for production.



# Create React App



- ❖ Create React App is an officially supported way to *create single-page React applications*.
- ❖ It offers a modern build setup with no configuration.
- ❖ Sets up the development environment to use the latest JavaScript features.
- ❖ Optimizes the application for production.
- ❖ Integrates the tools: NPM, Babel, Webpack, Webpack development server

# Create Project

1

## Install NodeJs

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

2

## Install Toolchain

- **Install create-react-app**
- **npm install create-react-app -g**

3

## Create React Application

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

4

## Start the Application

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

# Create Project

1

## Install NodeJs

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

2

## Create React Application

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

3

## Start the Application

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

# Create Project-Typescript

1

## Install NodeJs

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

2

## Create React Application

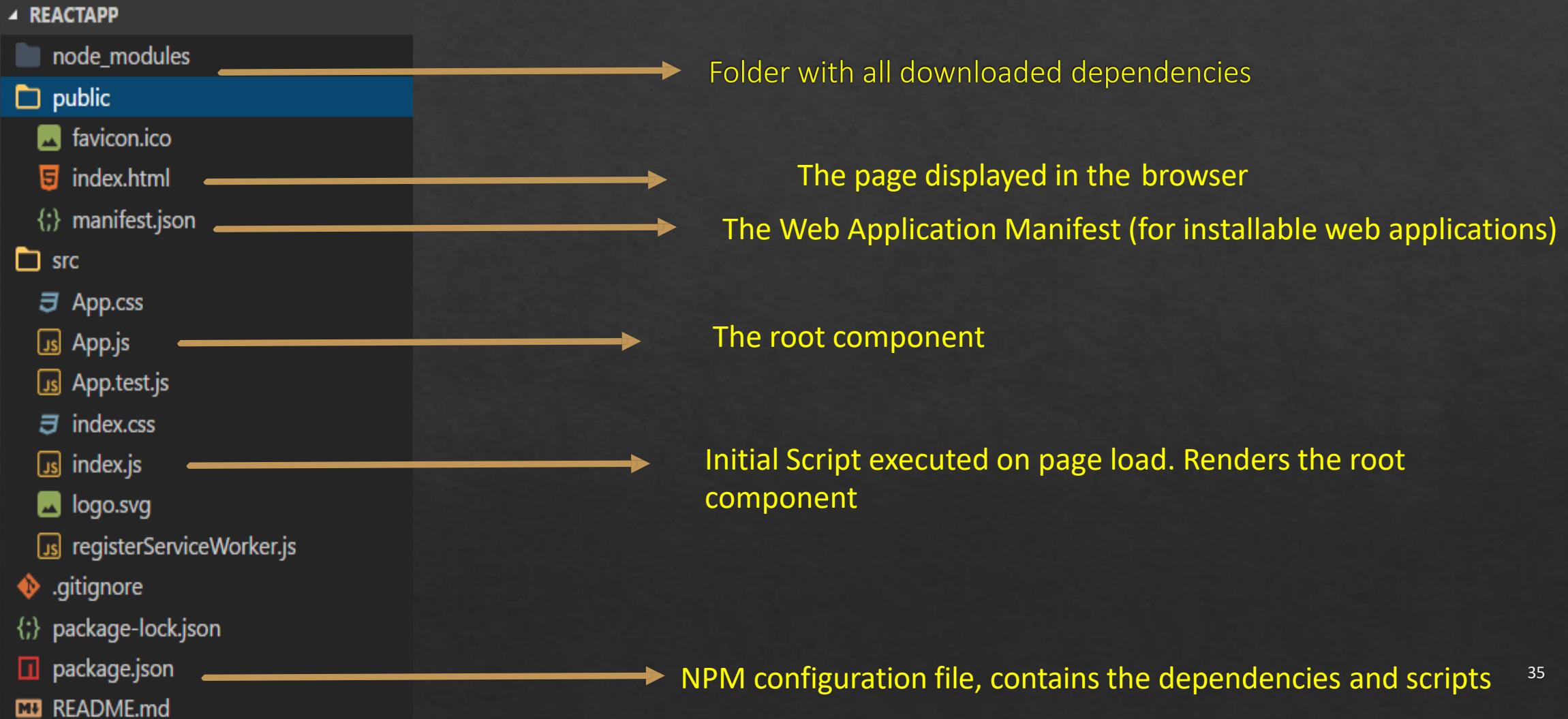
- `npx create-react-app the-react-app --template typescript`

3

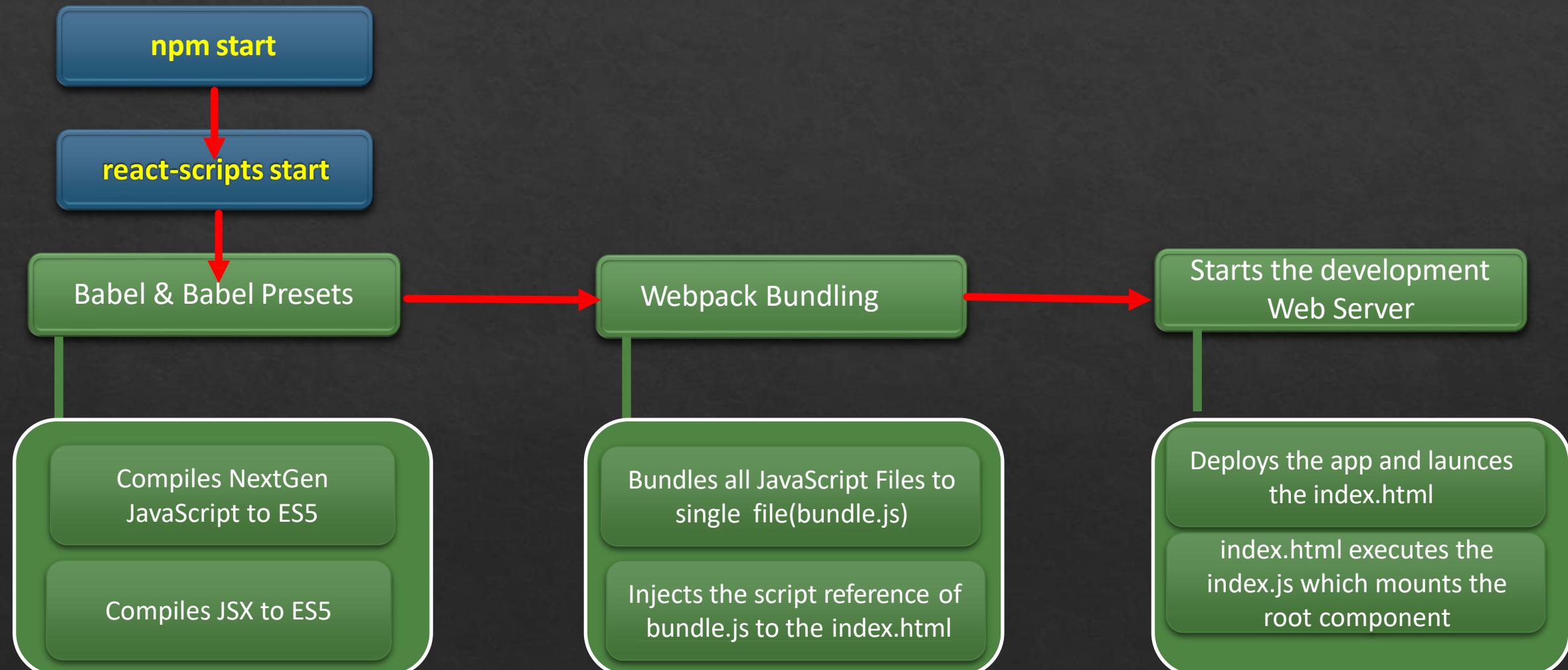
## Start the Application

- `cd the-react-app`
- `npm start`

# Project Structure

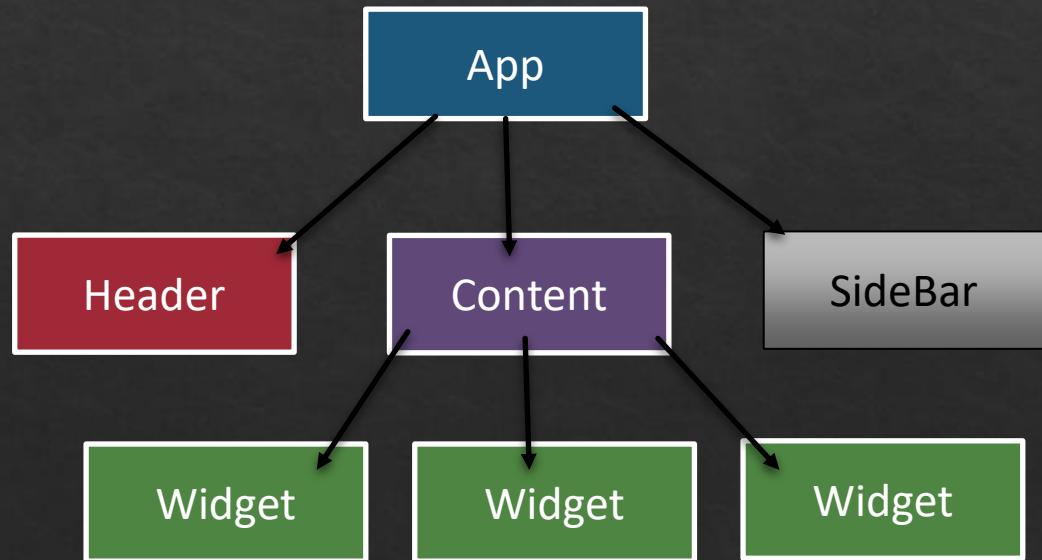
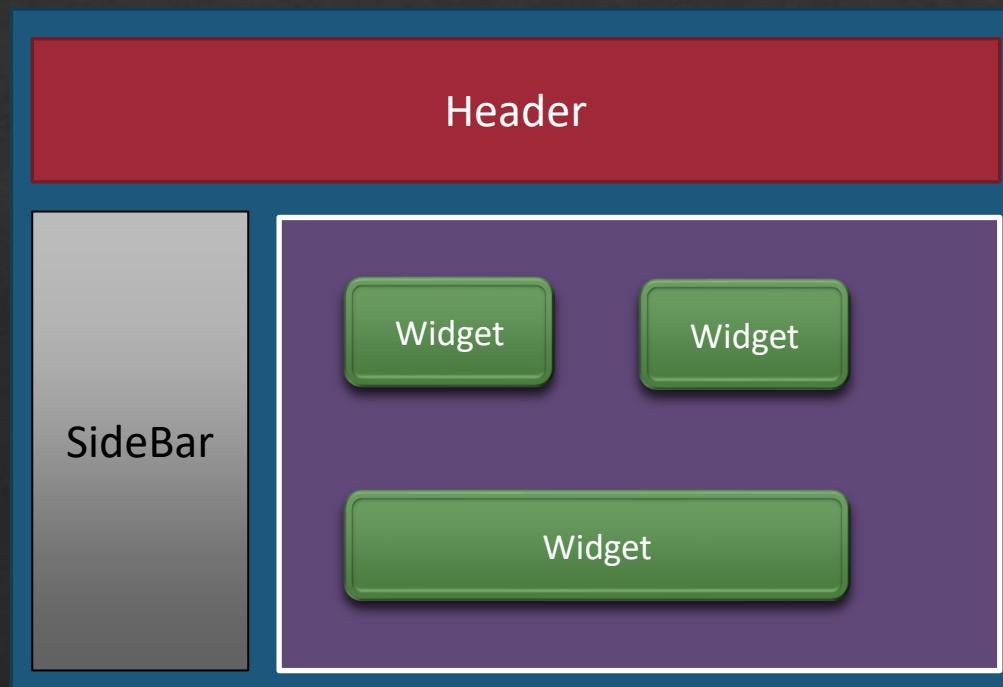


# Project Execution



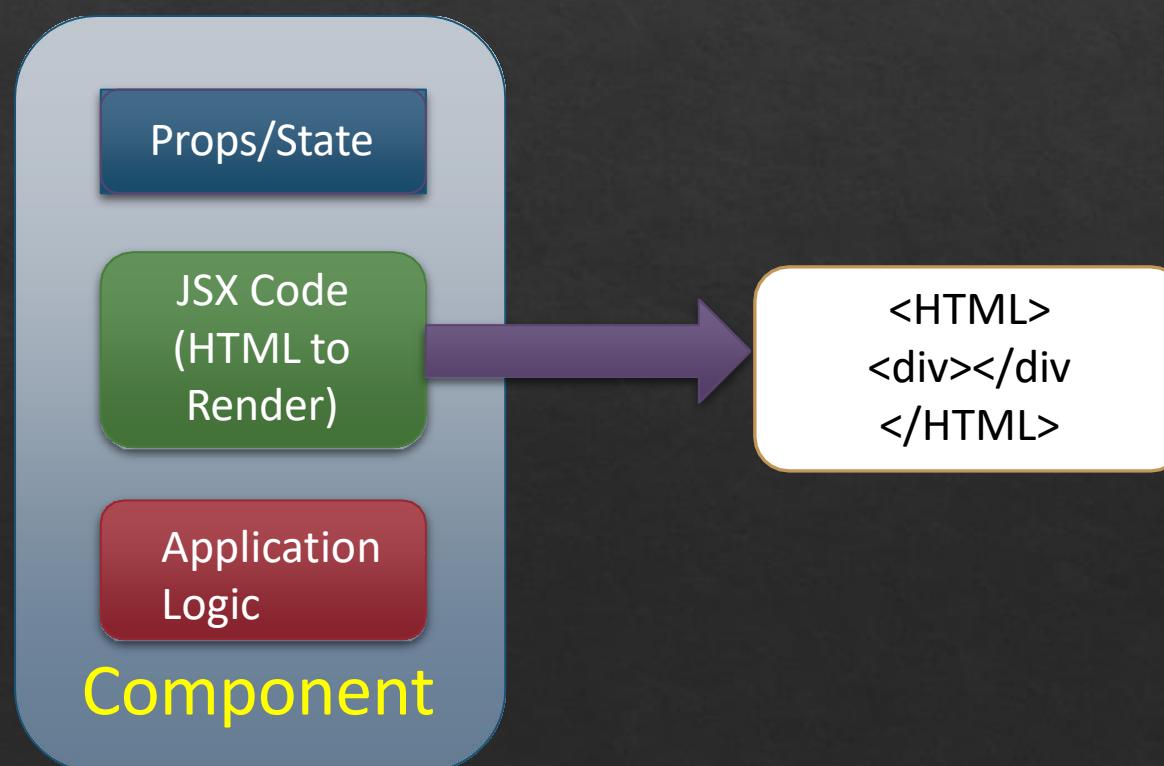
# React Components

- ❖ Components are the core building blocks in React
- ❖ Creating a React applications is all about designing and implementing components
- ❖ A React application 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() }`

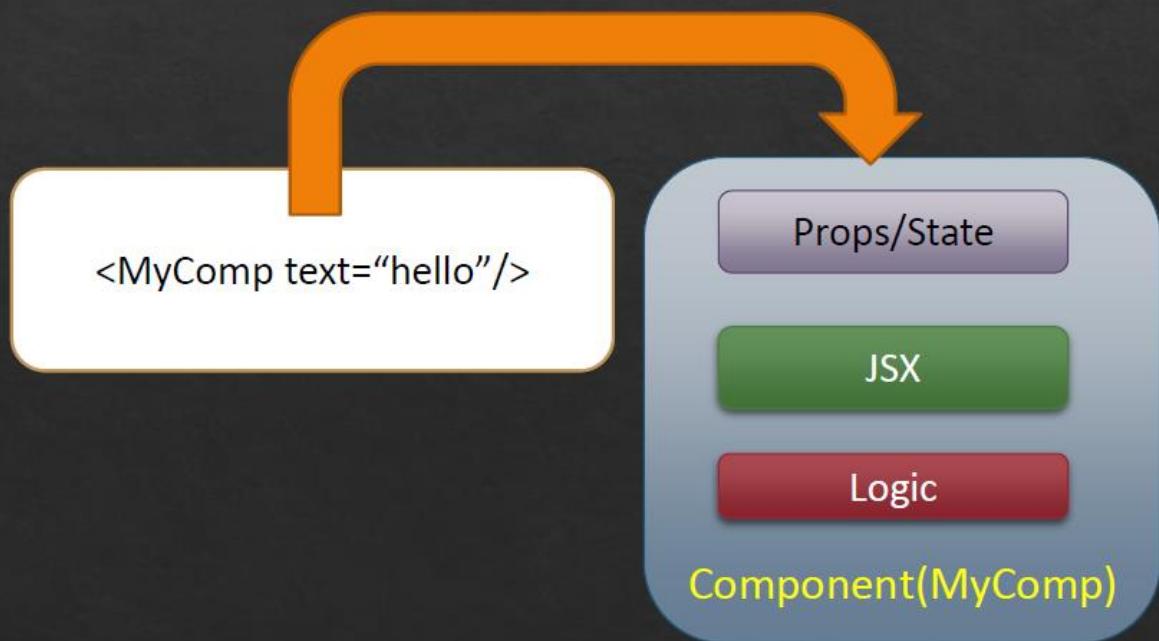
## Components: Properties(props)

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.***



# Component State

State holds information about the component

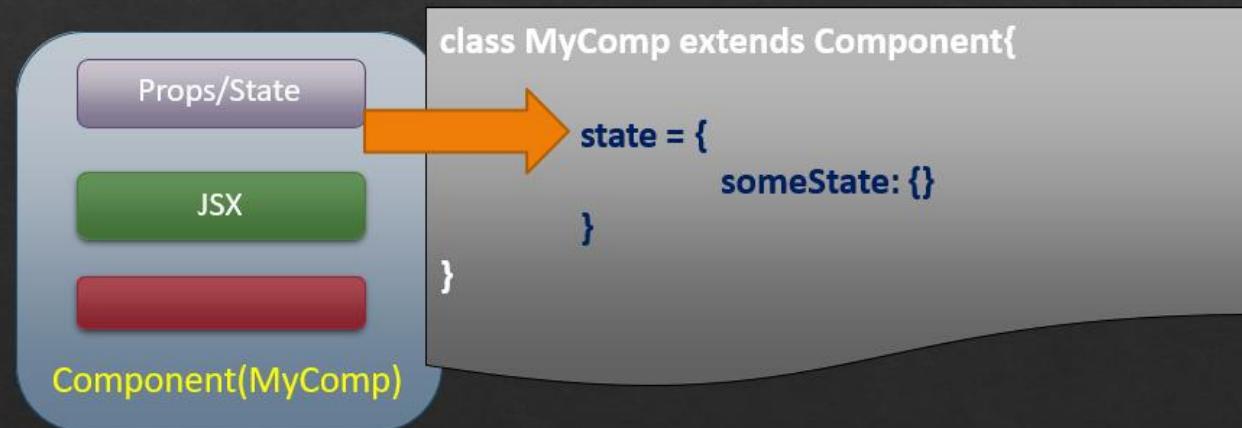
State is used when a component needs to keep track of information between renderings.

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

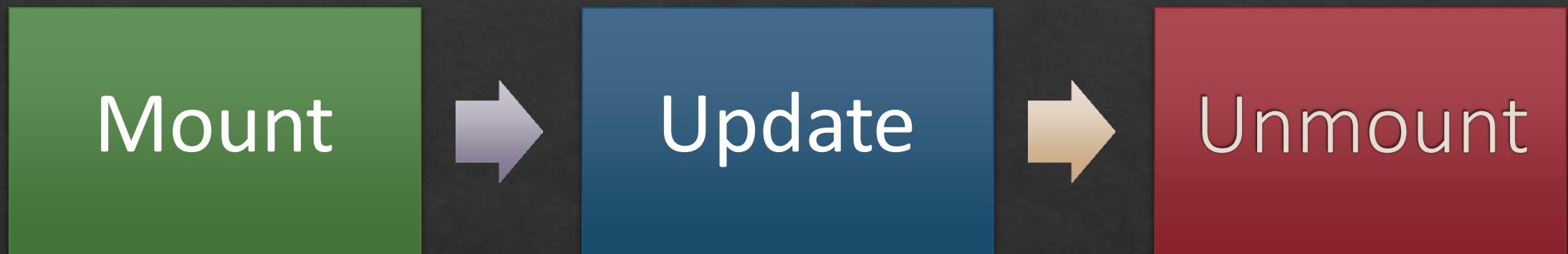
props allow you to pass data from a parent (wrapping) component to a child (embedded) component.

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>

# Component Lifecycle



# Component Lifecycle(Mount)

## constructor

- Setup State
- Don't cause side-effects

## componentWillMount

- Setup State
- Don't cause side-effects
- Configuration

## render

- Prepare and Structure the View

## Render Child Components

## componentDidMount

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

# 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/Render Child Components

componentDidUpdate

- Cause Side-Effects
- Don't update state

# Component Lifecycle(Unmount)

- ❖ componentWillUnmount
  - ❖ Remove Listeners
  - ❖ Cancel Active network calls
  - ❖ Invalidate Timers

# 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';`

# 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.

# 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;

# 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

## Custom Hooks

- A functions
- Uses Other Hooks

# 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**

# 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

# 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.

# Single Page Applications



A single-page application is an application that works inside a browser and does not require page reloading during use.



SPA is fast, as most resources (HTML+CSS+Scripts) are only loaded once throughout the lifespan of application. Only data is transmitted back and forth.

# 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

# React Router v4

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

# 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.

# State Management

## React Context

- Available from React 16.3

## React Redux

- Library to manage state

# 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

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

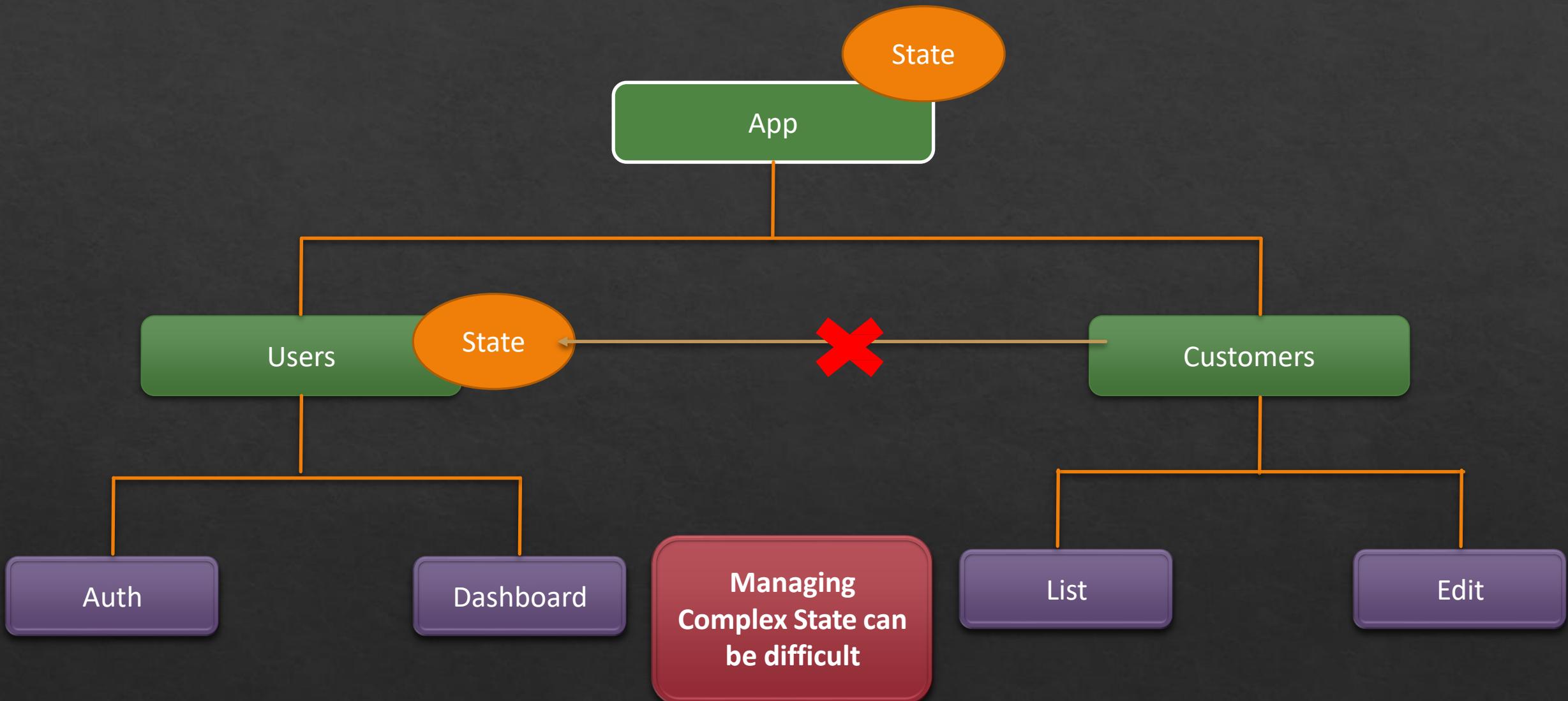
# Redux

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

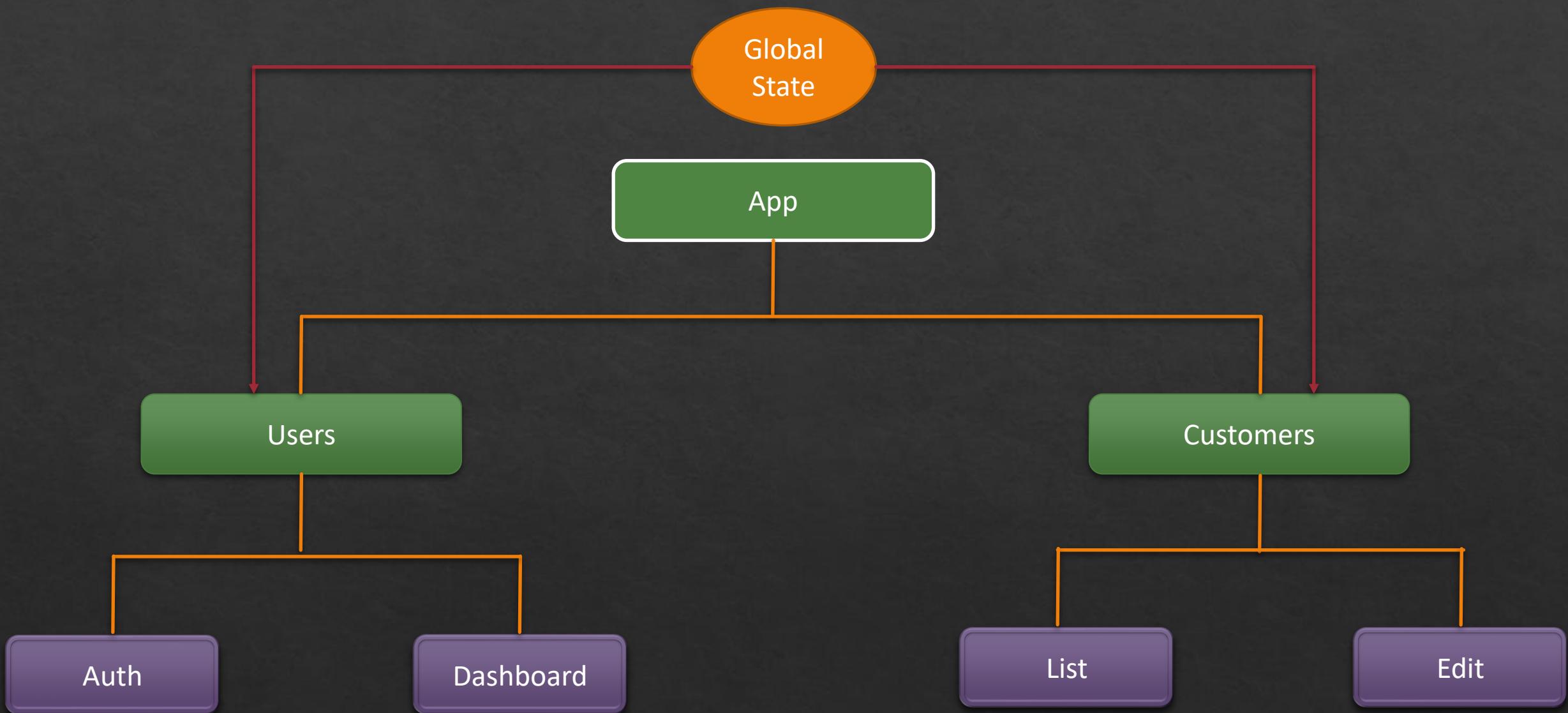
It is primarily used together with React or Angular for building user interfaces.

Redux was built on top of functional programming concepts.

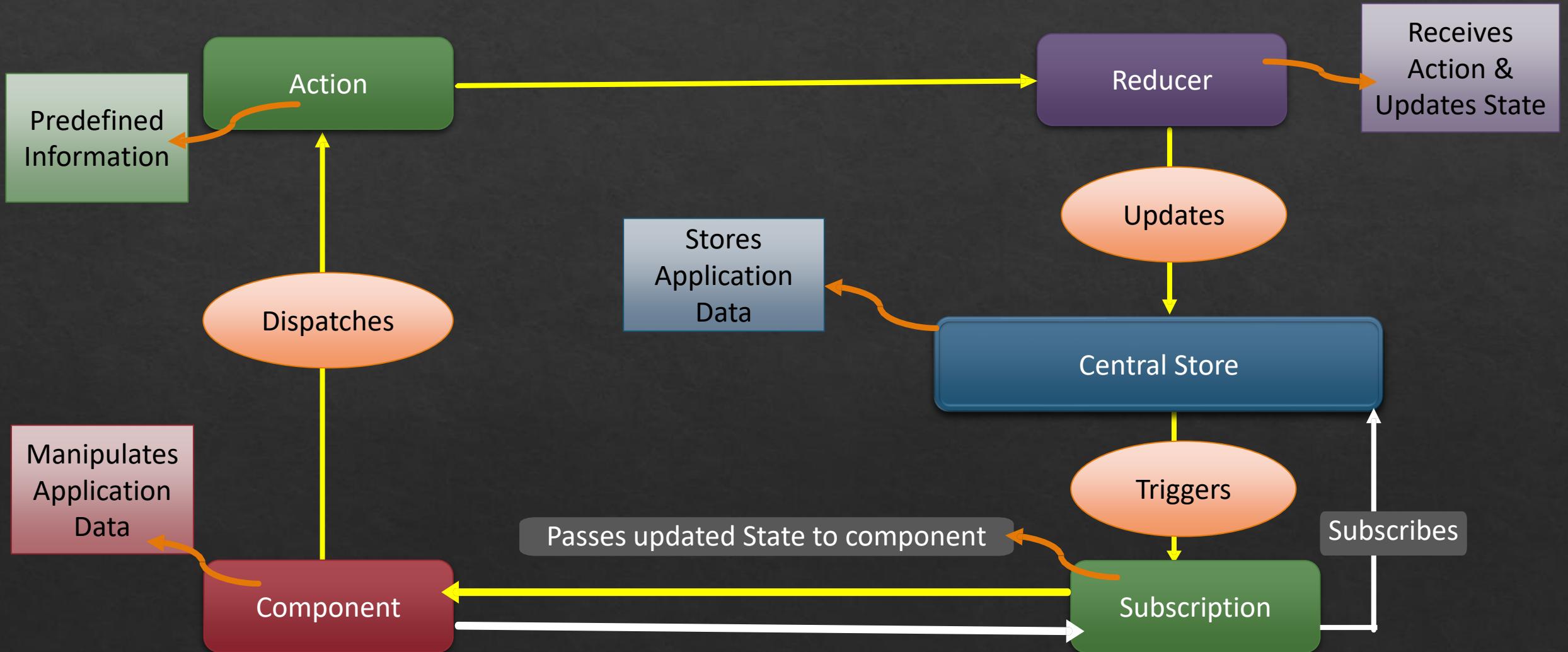
# Why Redux?



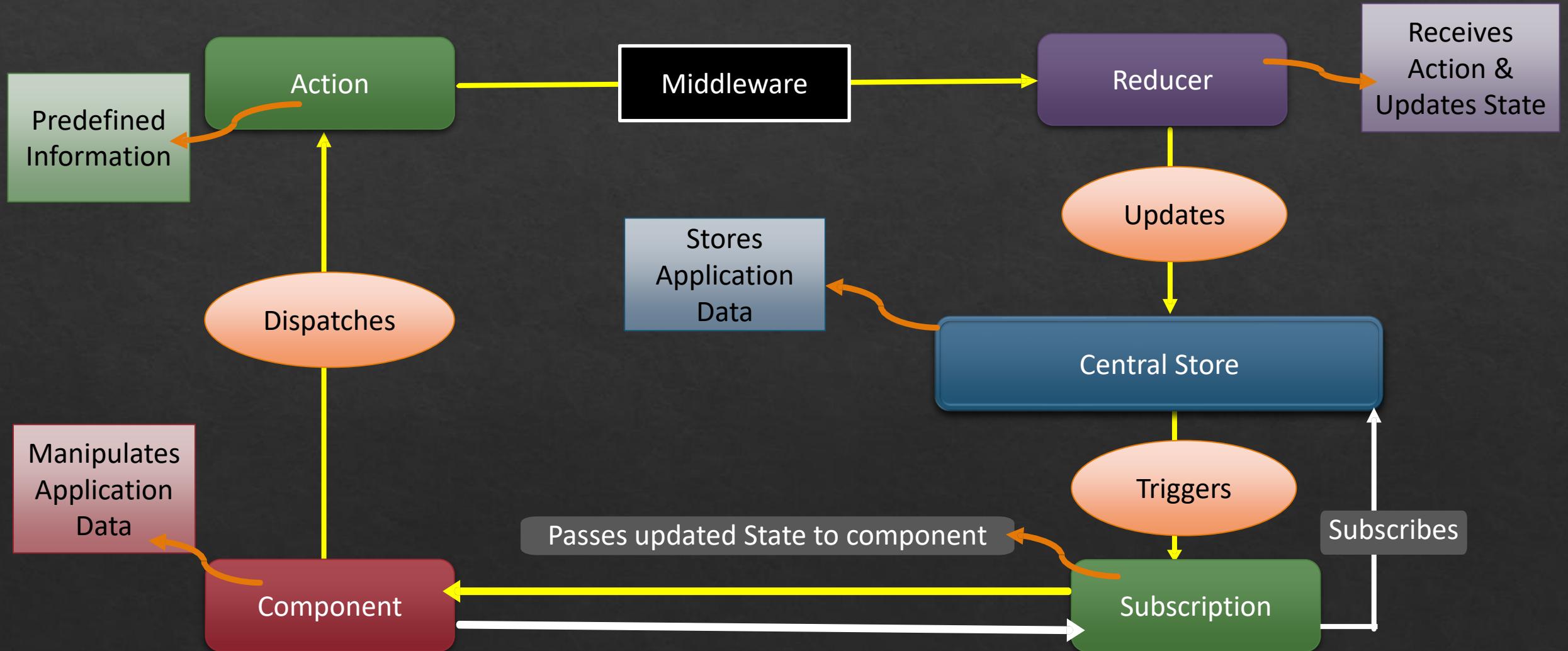
# Why Redux?



# Redux Flow



# Redux Flow



# React Redux

- ❖ Redux react is a library that integrates Redux to a React Application
- ❖ Comprises of Components & Functions
- ❖ Installation
  - ❖ npm install react-redux



# 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()

# 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`

# Code Splitting

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

# Deployment

Set basePath of Router

- <BrowserRouter basename="/app/">

Set homepage in package.json

- "homepage": "/app"

Build and Optimize npm run build

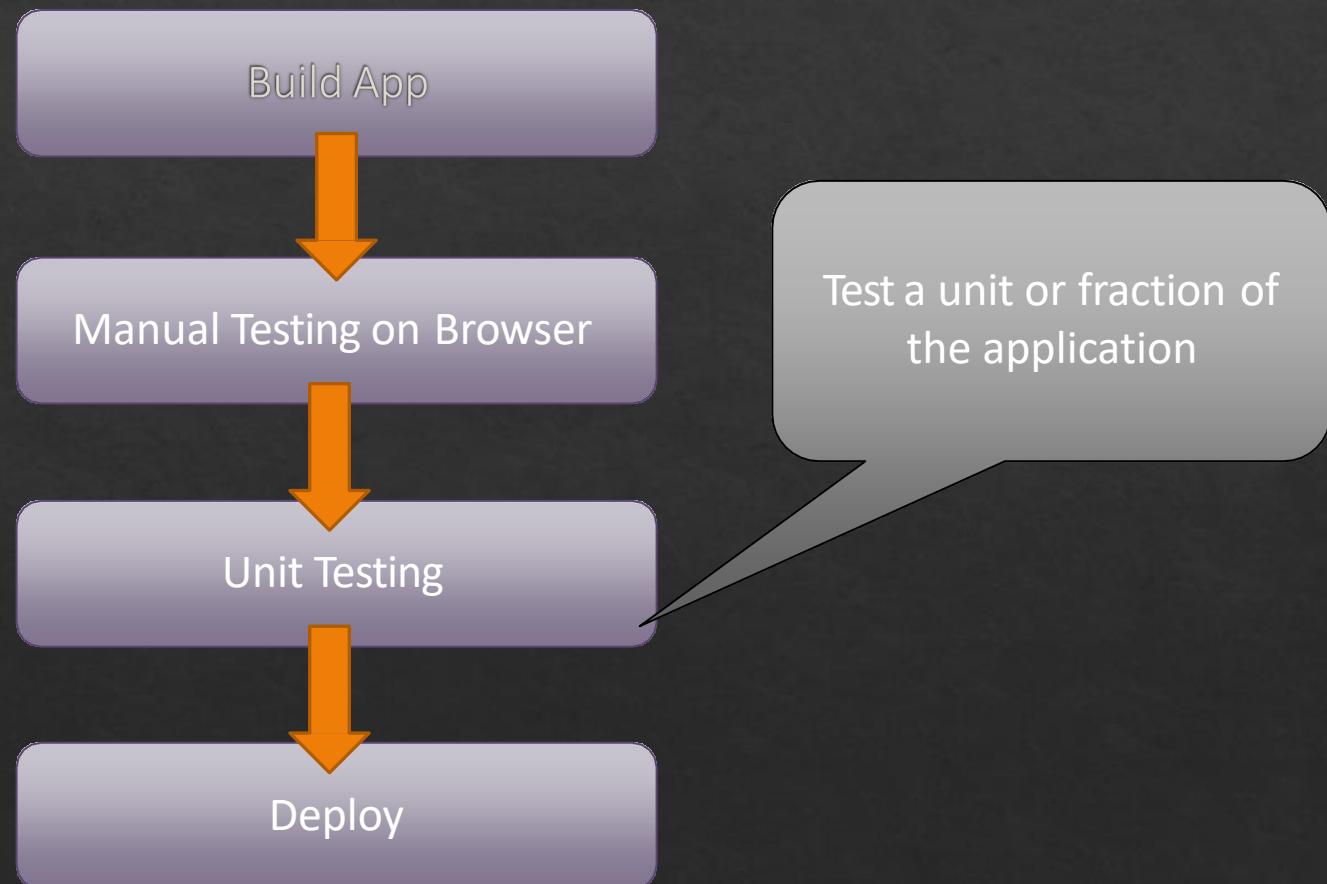
Server must Always serve index.html

Deploy artifacts to Web Server

# Using Sass

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

# Testing



# 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**

# 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

# Jest: Identifying Test files

Any files inside a folder named `__tests__` are considered tests

`__test__ / *.js`



Any files with `.spec` or `.test` in their filename are considered tests

`*.spec.js`

`*.test.js`

# 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

# 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)

---

# What to Test?

Test Isolated  
Components

Test Conditional  
Outputs

Don't Test Library  
Functions

Don't Test Complex  
Connection

# Installation

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

# 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

# E2E Tools



Cypress



Puppeteer



Selenium Webdriver



Nightwatch.js

# 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

# Deployment

Set basePath of Router

- <BrowserRouter basename="/app/">

Set homepage in package.json

- "homepage": "/app"

Build and Optimize npm run build

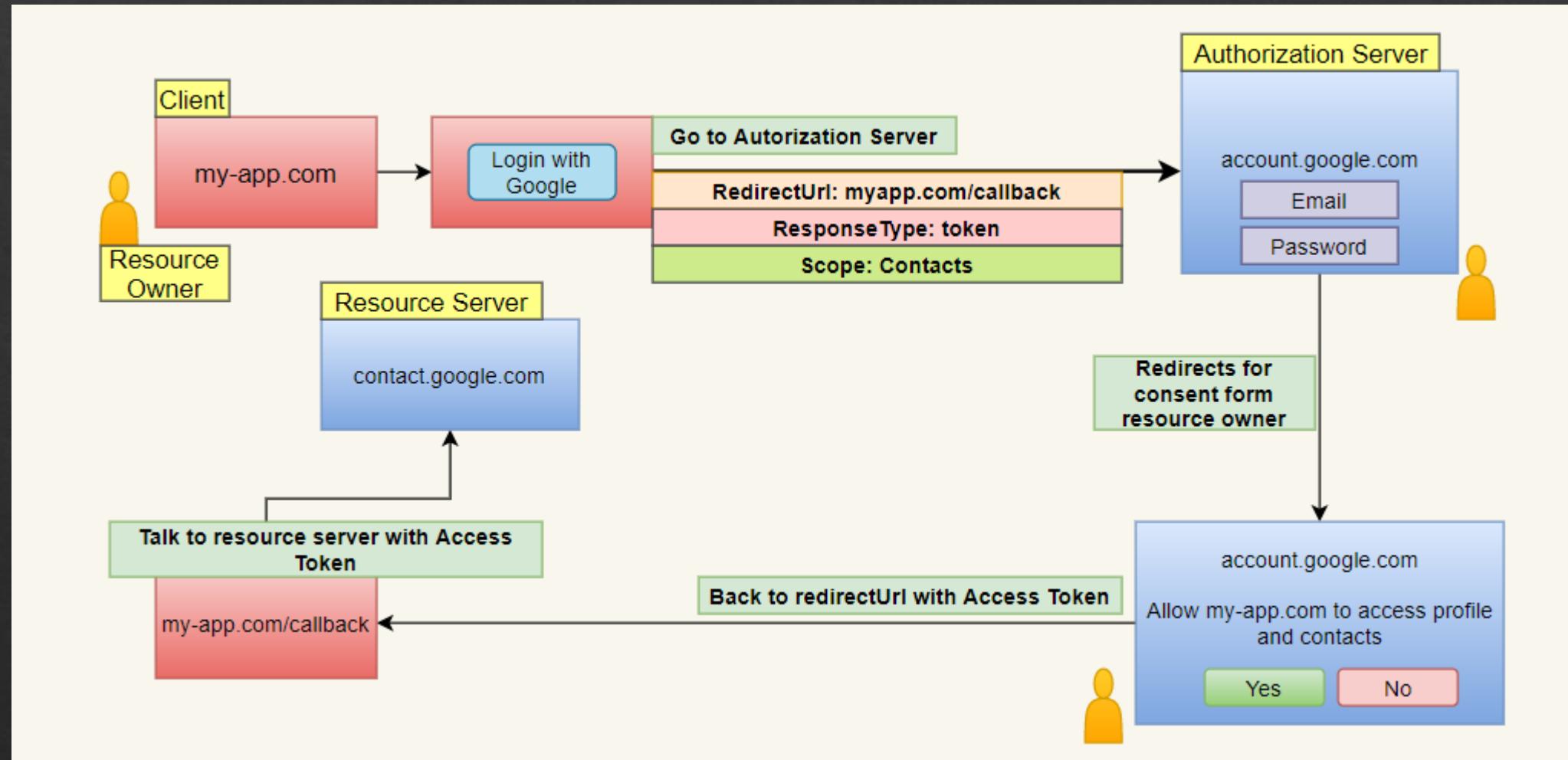
Server must Always serve index.html

Deploy artifacts to Web Server

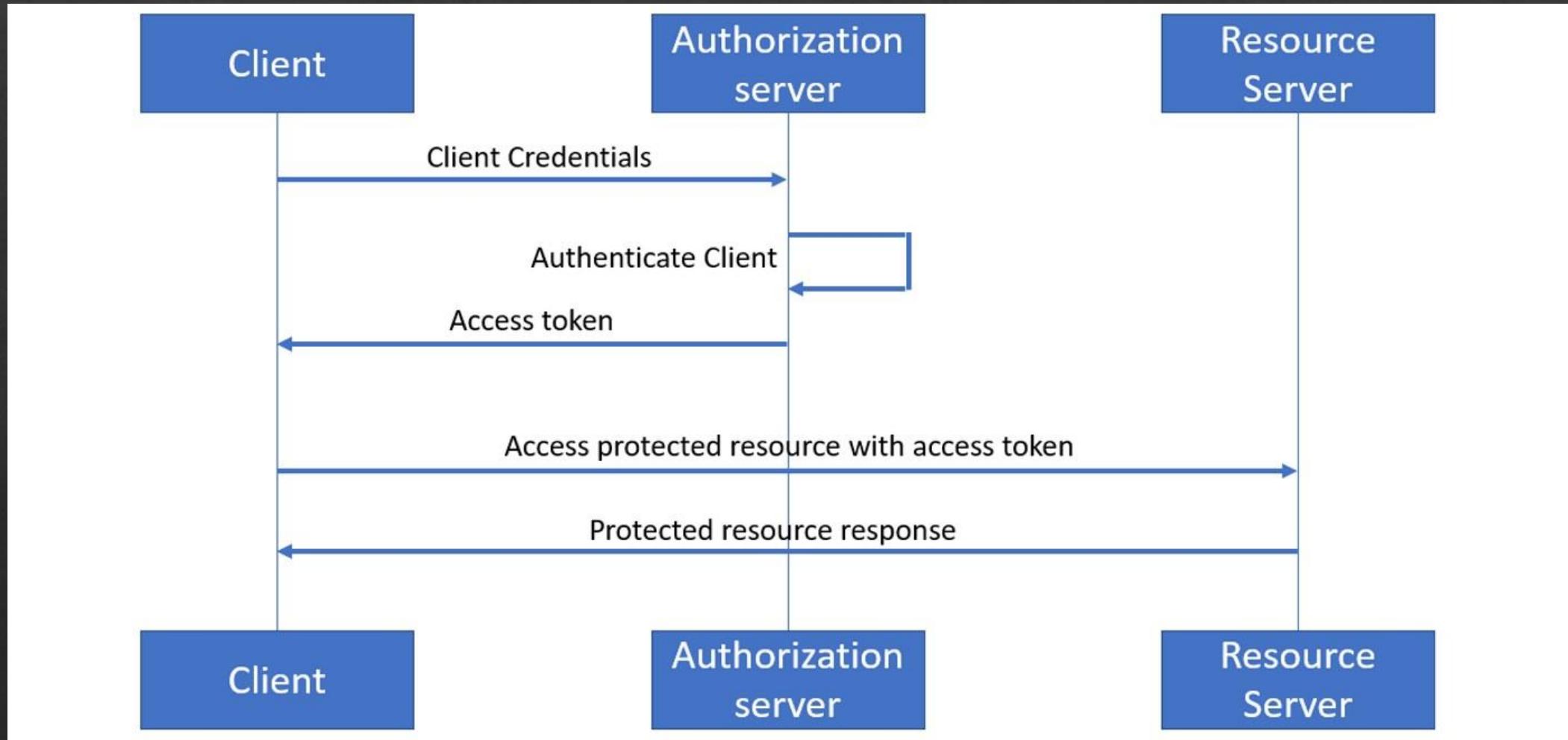
# OAuth 2.0

- ❖ OAuth is an open standard for access delegation.
- ❖ Used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords.
- ❖ Commonly used by companies such as Amazon, Google, Facebook, Microsoft and Twitter to permit the users to share information about their accounts with third party applications or websites.

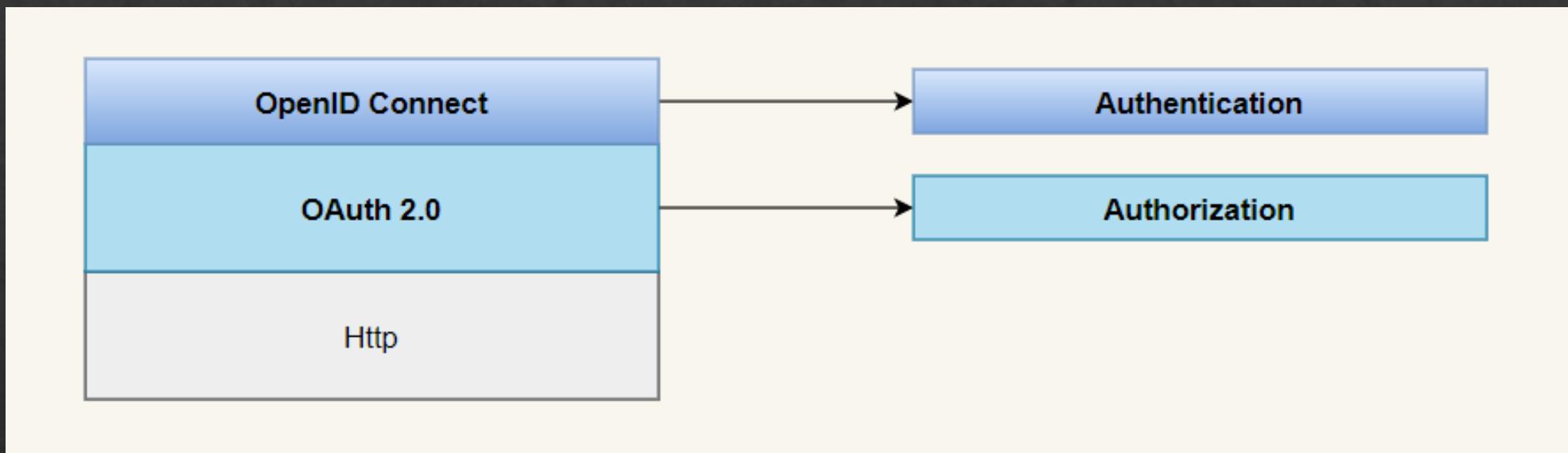
# OAuth 2.0



# OAuth 2.0



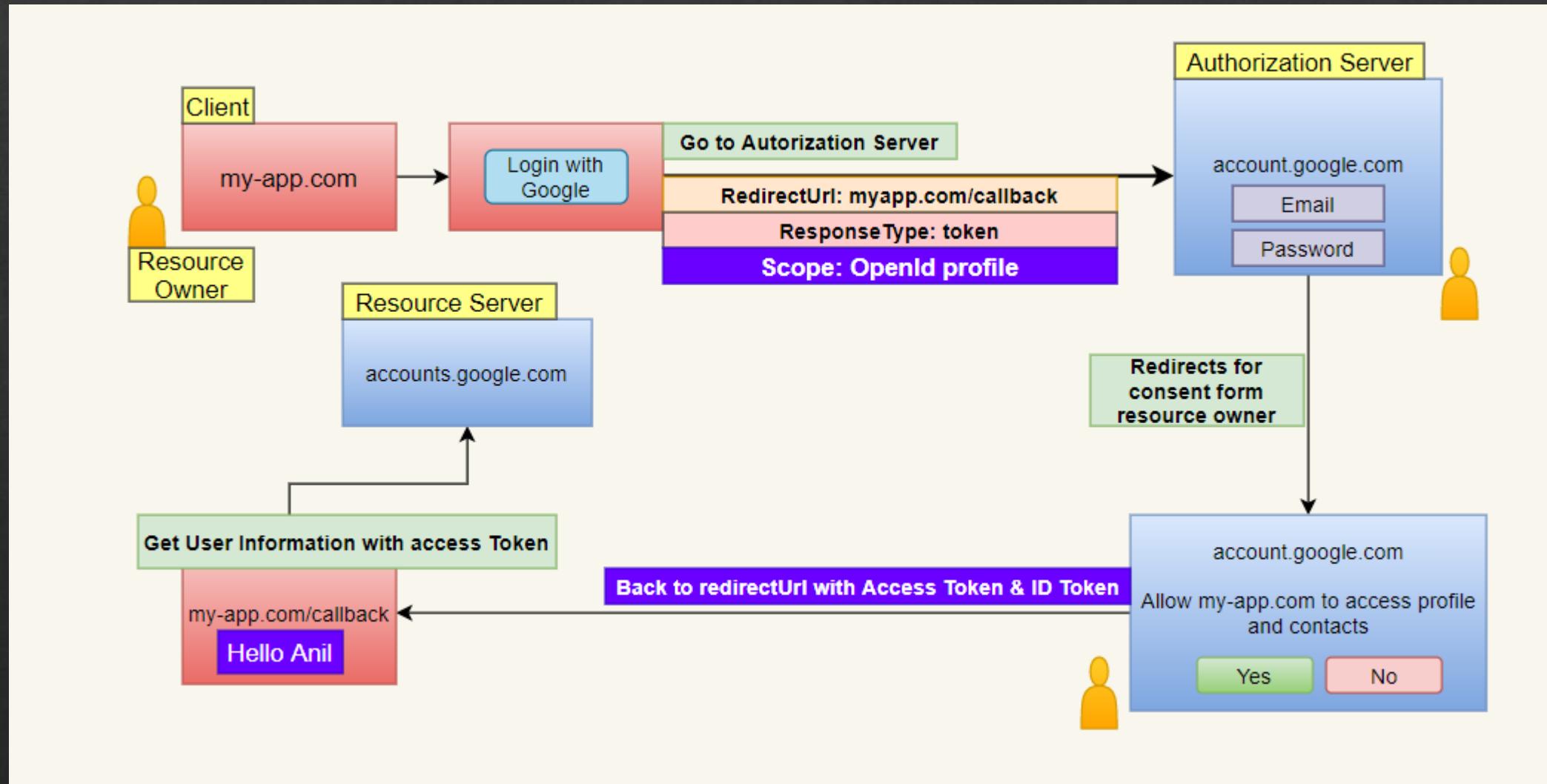
# OpenID Connect



# OpenID Connect

- ❖ OpenID Connect is a simple identity layer on top of the OAuth 2.0 protocol.
- ❖ Allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server.
- ❖ Can obtain basic profile information about the End-User in an interoperable and REST-like manner.

# OpenID Connect



# Json Web Token

- ❖ JSON Web Token (JWT) is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.
- ❖ Digitally Signed
- ❖ Used for Authorization and Information Exchange

# JWT

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.

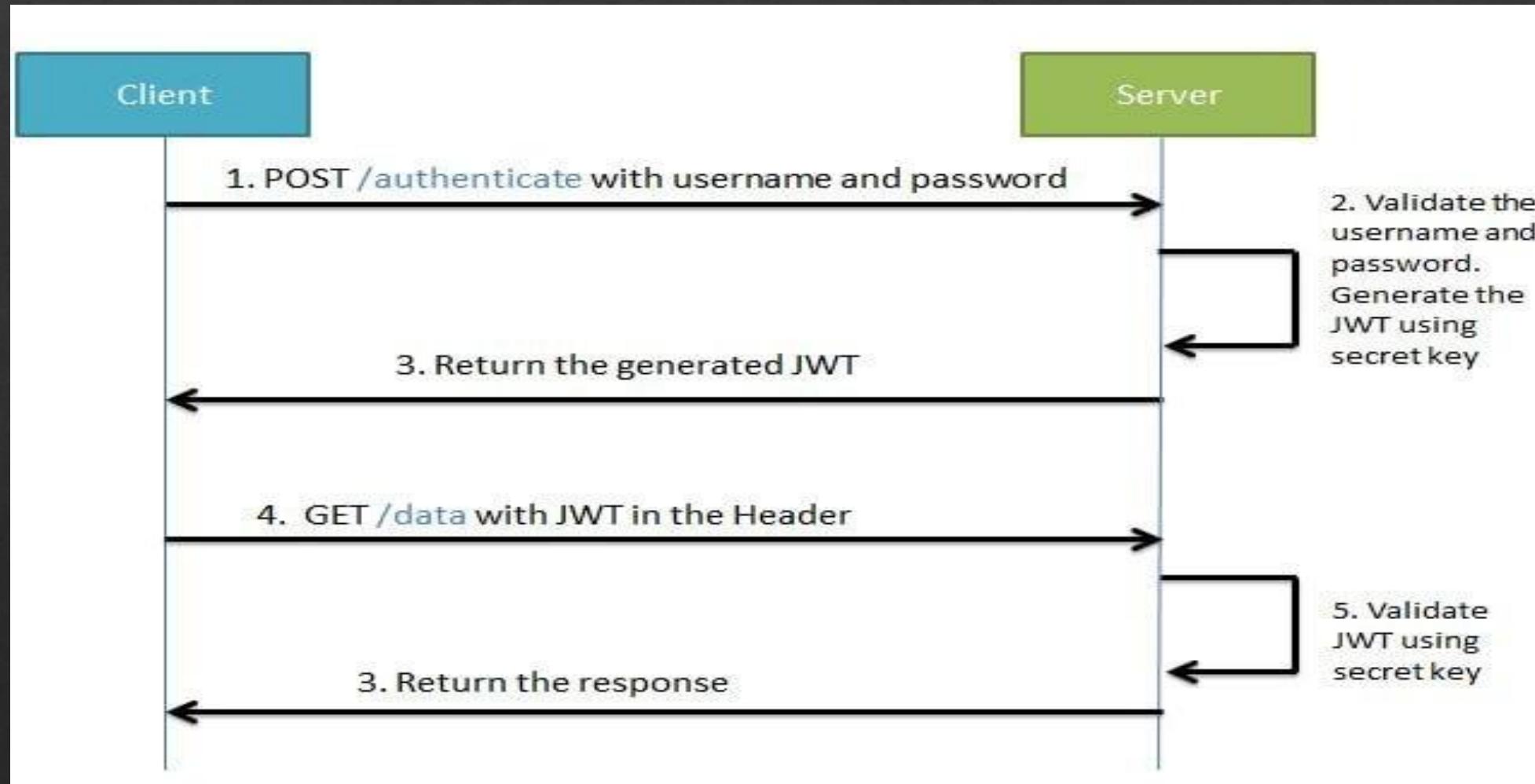
eyJpc3MiOiJPbmxbmUgSldeIEJ1aWxkZXIiLCJpYXQiOjE2  
MDA5OTkzMAsImV4cCI6MTYzMjUzNTMwMCwiYXVkljoi  
d3d3Lm15LWFwcC5jb20iLCJzdWIiOiJhbmlsLmpvc0BnbWF  
pbC5jb20iLCJHaXZlbk5hbWUiOiJBbmlsliwiU3VybmFtZSI6I  
kpvc2VwaClslkVtYWlsIjoiYW5pbC5qb3NAZ21haWwuY29tl  
iwiUm9sZSI6IkNvbnN1bHRhbnQifQ.

FrX-qAOPEDAyDh5IOJ6ipbFsrPBzaBzK7A2Hk5Yd2vk

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}  
  
{  
  "iss": "Online JWT Builder",  
  "iat": 1600999300,  
  "exp": 1632535300,  
  "aud": "www.my-app.com",  
  "sub": "anil.jos@gmail.com",  
  "GivenName": "Anil",  
  "Surname": "Joseph",  
  "Email": "anil.jos@gmail.com",  
  "Role": "Consultant"  
}
```

Signature

# JWT with Simple Forms Login



# React: Resources

- ❖ Official Site
  - ❖ <https://reactjs.org/>
- ❖ Awesome React tutorials
  - ❖ <https://github.com/enaqx/awesome-react>
- ❖ EggHead
  - ❖ <https://egghead.io/browse/frameworks/react>
- ❖ Youtube Channel(React Conf channel)
  - ❖ <https://www.youtube.com/channel/UCz5vTaEhh7dOHEyd1efcaQ>

# Thank You

ANIL JOSEPH

[anil.jos@gmail.com](mailto:anil.jos@gmail.com)