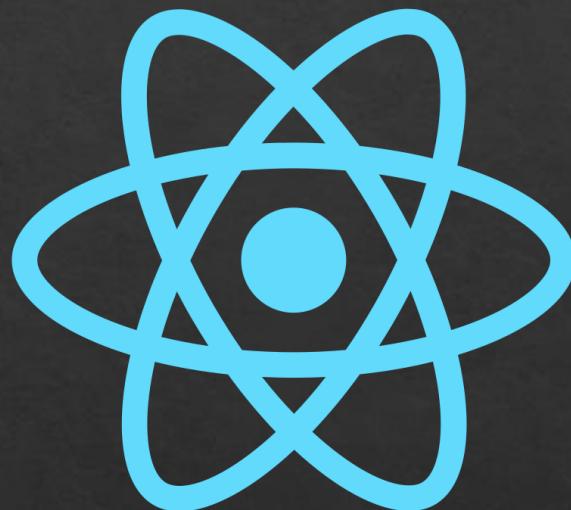


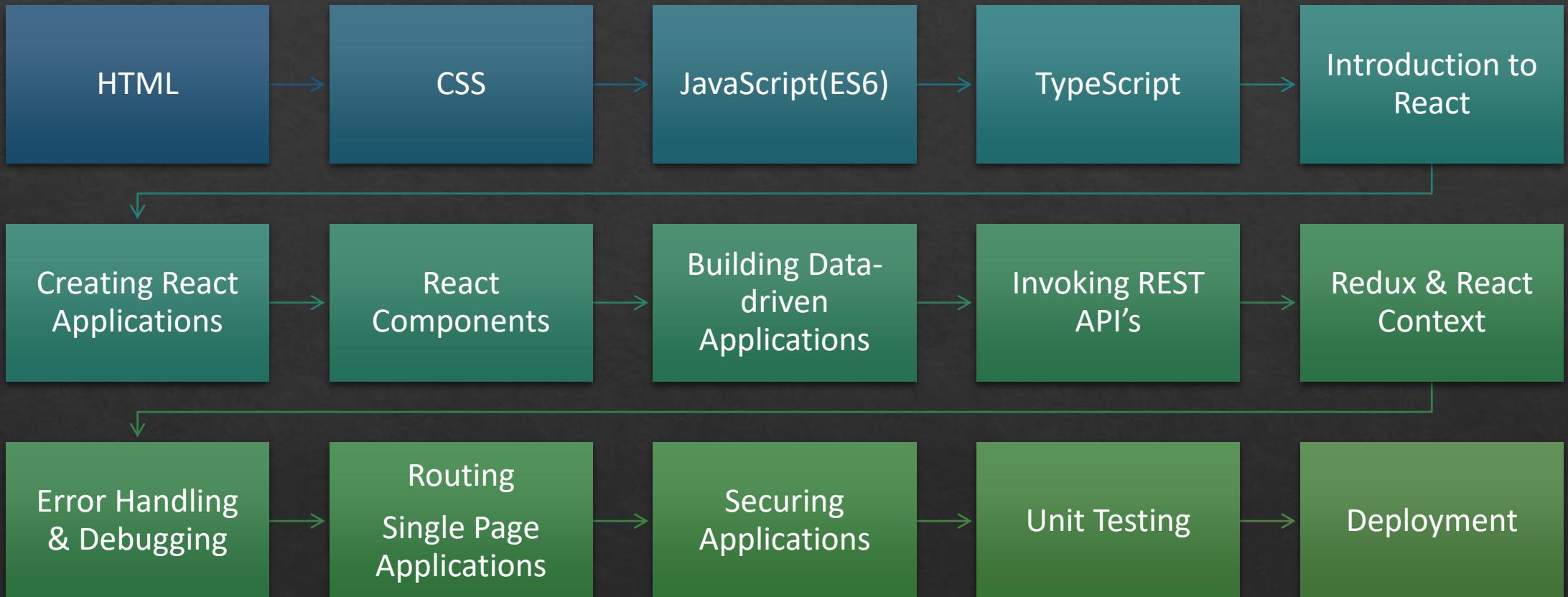
# React



ANIL JOSEPH

Presentaed by Anil Joseph(anil.jos@gmail.com)

# Agenda



# 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+)

# Software

Node.js & NPM

Yarn  
(Optional)

HTML, CSS, JavaScript,  
TypeScript Editor  
**(Visual Studio Code)**

Browsers(Chromium)

# HTML

- ❖ The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser.
- ❖ HTML was first created by Tim Berners-Lee, Robert Cailliau, and others in 1989.
- ❖ Versions
  - ❖ HTML2, HTML3, HTML4, HTML5

# What is HTML 5

- ❖ HTML 5 comprises of
  - ❖ Markup
  - ❖ JavaScript API's
- ❖ Most HTML5 features are supported by the modern browsers.

# HTML5 Features



More descriptive  
markup(Semantic)

New tags introduced that are  
more descriptive



Better Interface and  
Applications

Easier to create more richer and  
interactive applications.



Multimedia with less  
reliance on plug-ins

No requirement of using Flash  
or Silverlight for video, audio  
and vector graphics.



Backward Compatible



JavaScript libraries available for common  
functionalities like Forms, Sockets, Local Storage etc.

# HTML Markup

- ❖ html
- ❖ head
- ❖ body
- ❖ div(division)
- ❖ p (paragraph)
- ❖ form
- ❖ input
- ❖ select, option
- ❖ textarea
- ❖ button
- ❖ label
- ❖ h1,h2,h3,h4,h5,h6
- ❖ img
- ❖ table
- ❖ thead
- ❖ tbody
- ❖ th, tr, td
- ❖ ol, ul(lists)
- ❖ li (list item)

# Semantic Markup

- ◊ header
  - ◊ For an introduction of a document or section, could include navigation
- ◊ footer
  - ◊ For a footer of a document or section, could include the name of the author, the date of the document, contact information, or copyright information
- ◊ nav
  - ◊ Defines the navigation
- ◊ main
  - ◊ The main content area consists of content that is directly related to or expands upon the central topic of a document, or the central functionality of an application.
- ◊ section
  - ◊ For a section in a document. Such as chapters, headers, footers, or any other sections of the document
- ◊ article
  - ◊ Specifies independent, self-contained content, could be a news-article, blog post, forum post, or other articles which can be distributed independently from the rest of the site

# Semantic Markup

- ❖ aside
  - ❖ For content aside from the content it is placed in. The aside content should be related to the surrounding content
- ❖ canvas
  - ❖ For making graphics with a script
- ❖ figure
  - ❖ For grouping a section of stand-alone content, could be a video
- ❖ figcaption
  - ❖ The caption of the figure section
- ❖ time
  - ❖ For defining a time or a date, or both

# Semantic Markup

- ❖ audio
  - ❖ For multimedia content, sounds, music or other audio streams
- ❖ video
  - ❖ For video content, such as a movie clip or other video streams
- ❖ datalist
  - ❖ A list of options for input values
- ❖ mark
  - ❖ For text that should be highlighted
- ❖ details & summary
  - ❖ represents a disclosure widget
- ❖ progress
  - ❖ represents the completion progress of a task
- ❖ meter
  - ❖ Displaying a scalar measurement within a known range

# Getting Started

- ❖ HTML 5 pages can be viewed on any browser
- ❖ However, features of HTML 5 are implemented by only the latest browsers
- ❖ The doc type for HTML 5
  - ❖ <!DOCTYPE html>
  - ❖ Much simpler than earlier versions

# Javascript API's

- ❖ DOM Manipulation
- ❖ Forms
- ❖ Geolocation
- ❖ Web Storage
- ❖ Web Workers
- ❖ Media Capture
- ❖ Web Notifications
- ❖ Offline Web Applications
- ❖ Server Sent Events
- ❖ Web Sockets
- ❖ XMLHttpRequest
- ❖ Canvas 2D Context & WebGL

# HTML DOM

- ❖ The Document Object Model (DOM) is the data representation of the objects that comprise the structure and content of a document on the web.
- ❖ Its an object-oriented representation of the web page and can be modified with a scripting language such as JavaScript.
- ❖ The document object represents the DOM
- ❖ The DOM API's are used to access and manipulate the DOM.

# Window

- ❖ The window object represents a window/tab containing a DOM document.
- ❖ All global JavaScript objects, functions, and variables automatically become members of the window object.
- ❖ Properties available on window object
  - ❖ history
  - ❖ location
  - ❖ screen
  - ❖ navigator
  - ❖ many more...

# Events

- ❖ document:DOMContentLoaded
  - ❖ When the DOM is ready
- ❖ window:load
  - ❖ When the page and all resources are loaded
- ❖ window:beforeunload
  - ❖ When the user is leaving the page, can be cancelled
- ❖ window:unload
  - ❖ When the user is leaving the page
- ❖ document:readystatechange
  - ❖ The current state of the page

# Form Field Types

- ❖ text
- ❖ password
- ❖ email
- ❖ url
- ❖ range
- ❖ number
- ❖ date
- ❖ datetime
- ❖ color
- ❖ tel
- ❖ search

# Methods and Properties

- ❖ `boolean checkValidity();`
  - ❖ Validates the control against constraints.
  - ❖ If any constraints are not satisfied, then the user-agent fires the ***invalid*** event and the method returns false.
  - ❖ If all constraints are satisfied does nothing, the method returns true
- ❖ `readonly attribute ValidityState validity;`
  - ❖ Returns the validity state of the control
- ❖ `readonly attribute DOMString validationMessage;`
  - ❖ Return a suitably localized message that the user agent would show the user if this were the only form control with a validity constraint problem.
- ❖ `void setCustomValidity(in DOMString error);`
  - ❖ set the ***custom validity error*** message to the value of the given message argument.

# Validity States

- ◊ **valueMissing**
  - ◊ Returns true if a required element is empty.
- ◊ **typeMismatch**
  - ◊ Applies to all the new type attributes. For example, if an email value is incorrect, this attribute returns true.
- ◊ **patternMismatch**
  - ◊ Returns true when an element contains the pattern attribute and doesn't conform to the regular expression.
- ◊ **tooLong**
  - ◊ When any element surpasses its max length property this attribute will return true.
- ◊ **rangeUnderflow** and **rangeOverflow**
  - ◊ If an element's min or max attributes are above or below the specified values, this attribute will return true.
- ◊ **stepMismatch**
  - ◊ When an element with the step attribute doesn't conform to the required step value, this attribute returns true.
- ◊ **valid**
  - ◊ If any of the values listed above return true, this attribute returns false to indicate the field is invalid. Otherwise, if all the conditions are met, it will return true.

# Additional Form Support

- ◆ autofocus
  - ◆ Support for placing the focus on a specific form element.
- ◆ required
  - ◆ Support for enabling the input field as required.
- ◆ placeholder
  - ◆ Support for placing a placeholder text in a form field
- ◆ pattern
  - ◆ Define a regular expression for validation
- ◆ contenteditable
  - ◆ Support for inline content editing

# AJAX

- ❖ Asynchronous JavaScript and XML.
- ❖ Ajax is methodology to asynchronously invoke a server endpoint using the HTTP(s) protocol from a browser application using JavaScript. The data exchange is in XML format.
- ❖ Lately the preferred data format is JSON.
- ❖ All browsers provide a JavaScript object called XMLHttpRequest which is the core object for AJAX.
- ❖ There are many high-level libraries built over XMLHttpRequest
- ❖ Fetch API is one such library which is part of HTML5

# Browser Storage

- ❖ HTML 5 introduces a new mechanism for storing data on the client
  - ❖ Web Storage
  - ❖ Easy to use, powerful and reasonably secure.
- ❖ Other mechanisms
  - ❖ Web SQL Databases
  - ❖ Indexed Databases

# Web Storage

- ❖ Allows us to store information as key-value pairs on the client-side
- ❖ Unlike cookies the information stored does not get exchanged to the server.
- ❖ Browsers restricts every domain to store up to 5mb to 10mb of data on the client.
- ❖ Currently supported by most browsers.
- ❖ Web Storage gives us 2 scopes to store information
  - ❖ localStorage: A key/value pair, tied to the domain and persists across browser sessions.
  - ❖ sessionStorage: A key/value pair tied to the domain and erased when the browser session ends.
- ❖ The API's are implemented as javascript functions using the
  - ❖ `window.localStorage` object
  - ❖ `window.sessionStorage` object

# Web Storage Methods and events

- ❖ Methods

- ❖ `setItem(key, value)`
- ❖ `getItem(key)`
- ❖ `removeItem(key)`
- ❖ `clear()`

- ❖ Events

- ❖ `storage`

# CSS

## Cascading Style Sheets

# CSS 3.0

- ❖ Concepts
- ❖ Selectors
- ❖ Responsive Design
- ❖ Media Queries
- ❖ Box and Flex Model
- ❖ Bootstrap

# Cascading Style Sheets

” ”

CSS is a stylesheet language used to describe the presentation of a document written in HTML.



CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

# Selectors

Selectors are used to target the HTML elements on our web pages that we want to style.

- ◊ \*
  - ◊ Targets every single element on a page
- ◊ #x
  - ◊ Targets the elements with the id="x"
- ◊ x
  - ◊ Targets the elements of type "x"
- ◊ .x
  - ◊ This is a class selector. More reusable

# Selectors

- ❖ **x y**
  - ❖ The descendant selector
  - ❖ Selects the element “y” which is a descendant of “x”
- ❖ **x > y**
  - ❖ Selects the element “y” which is direct child of “x”
- ❖ **x + y**
  - ❖ Selects the element “y” which immediately follows (sibling) “x”

# Selectors

- ◊ `x[attr]`
  - ◊ Attribute selector
  - ◊ Selects all elements “x” which have a the attribute “attr” defined.
- ◊ `x[attr=“foo”]`
  - ◊ Selects all elements “x” which have a the attribute “*attr*” ***equal to “foo”***.
- ◊ `x[attr*=“foo”] (CSS 3.0)`
  - ◊ Selects all elements “x” which have a the attribute “*attr*” ***contains “foo”***.
- ◊ `x[attr^=“foo”] (CSS 3.0)`
  - ◊ Selects all elements “x” which have a the attribute “*attr*” ***starts with “foo”***.

# Selectors

- ❖ `x[attr $=“foo”]` (*CSS 3.0*)
  - ❖ Selects all elements “x” which have a the attribute “attr” **ends with “foo”**.
- ❖ `x[attr ~=“foo”]` (*CSS 3.0*)
  - ❖ Selects all elements “x” which have a the attribute “attr” **containing value “foo” that’s space seperated.**
- ❖ `x:checked` (*CSS 3.0*)
  - ❖ Targets an element that is checked(checkbox, radiobutton).
- ❖ `x:not(#id)`
  - ❖ Negation pseudo class
  - ❖ Selects all elements “x” which does not have the id equal to “id”

# Responsive Web Design

- ❖ Responsive design aims to make web pages render properly on a variety of devices with different screen sizes, form factors, resolution etc.
- ❖ How?
  - ❖ Media queries and media query listeners
  - ❖ A flexible grid-based layout that uses relative sizing
  - ❖ Flexible images and media, through dynamic resizing.

# CSS 3.0 Media Queries

- ❖ Media Queries is a module allowing content rendering to adapt to conditions such as screen resolution.
- ❖ It became a W3C recommended standard in June 2012.
- ❖ A media query consists
  - ❖ *media type*
  - ❖ one or more expressions

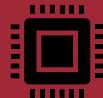
# Media types

- ❖ screen
- ❖ print
- ❖ handheld
- ❖ projection
- ❖ tv
- ❖ all

# Media Expressions

- ❖ min-width
- ❖ min-height
- ❖ max-width
- ❖ max-height
- ❖ device-width
- ❖ device-height
- ❖ orientation
- ❖ aspect-ratio
- ❖ device-aspect-ratio
- ❖ color

# CSS Box Model



All HTML elements can be considered as boxes.



In CSS, the term "box model" is used when referring to design and layout.



The CSS box model is essentially a box that wraps around HTML elements



It consists of:

Margins

Borders

Padding

actual content.

# CSS 3.0 Flexbox



In this flexbox model, the children of a box are laid out either horizontally or vertically.



Unused space can be assigned to a particular child or distributed among the children by assignment of flex to the children that should expand.



Nesting of boxes (horizontal inside vertical, or vertical inside horizontal) can be used to build layouts in two dimensions.

# Flexbox styles

## display

- values : display or inline-display

## flex-direction

- row (default): left to right
- row-reverse: right to left
- column: same as row but top to bottom
- column-reverse

## flex-wrap

- nowrap (default): single-line / left to right
- wrap: multi-line / left to right
- wrap-reverse: multi-line / right to left

## flex-flow

- This is a shorthand flex-direction and flex-wrap properties
- flex-flow: <'flex-direction'> | | <'flex-wrap'>

# Flexbox styles

## justify-content

- **flex-start** (default): items are packed toward the start line
- **flex-end**: items are packed toward to end line
- **center**: items are centered along the line
- **space-between**: items are evenly distributed in the line; first item is on the start line, last item on the end line
- **space-around**

## order(child)

- By default, flex items are laid out in the source order. However, the `order` property controls the order in which they appear in the flex container.

# Bootstrap

- ❖ A CSS framework for building responsive, mobile-first sites.
- ❖ Works on all modern browsers
- ❖ Provides CSS and HTML design templates
- ❖ Provides reusable components
- ❖ Supports LESS and SASS
- ❖ Customizable

# JavaScript

---

An interpreted language

---

Dynamic

---

Object-oriented

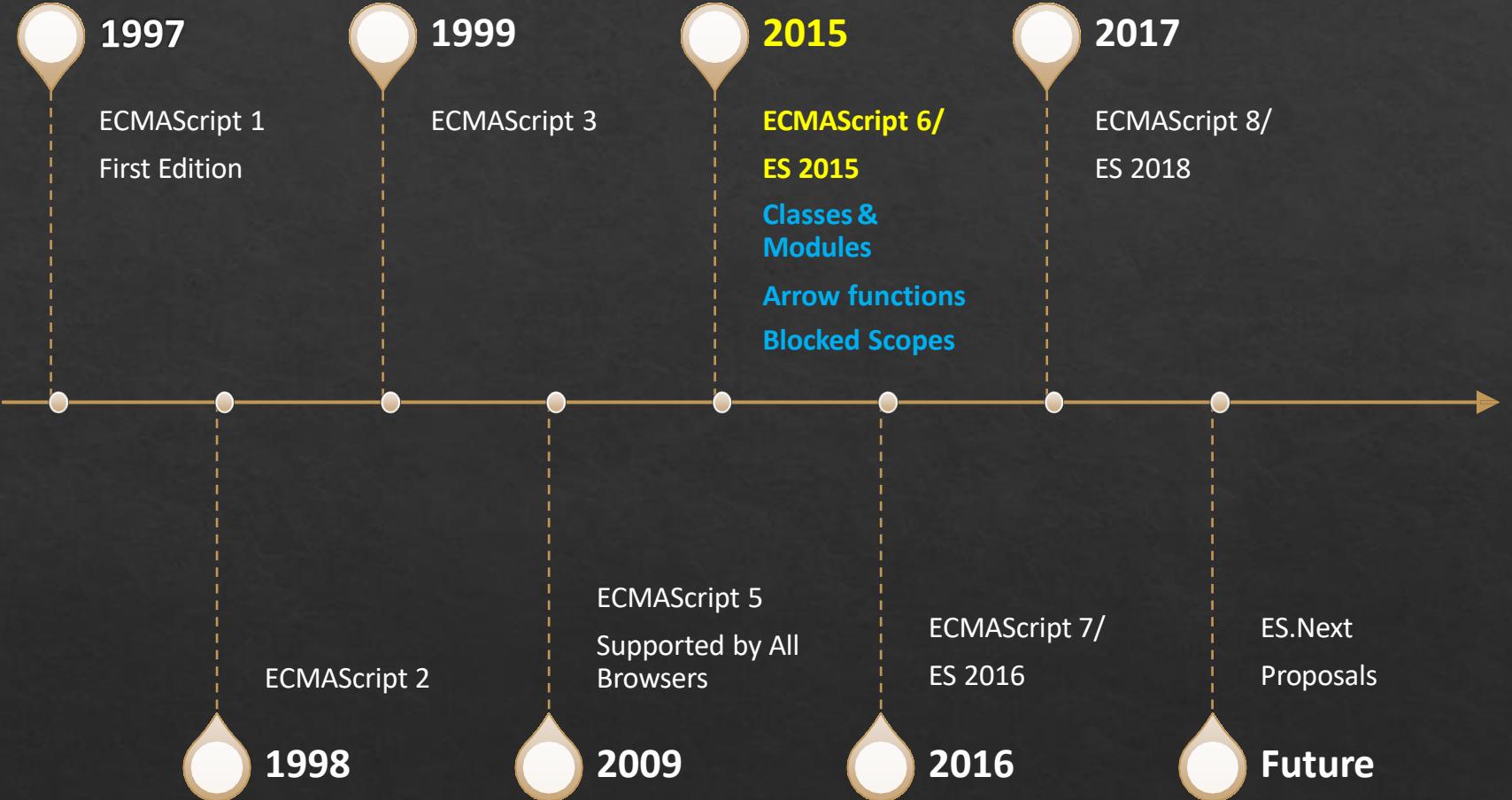
---

Supports Functional style of programming

---

Available on the browsers and Node.js

# ECMAScript Versions



# ES6 New Syntax

Block Scoping

Arrow  
Functions

Rest and  
Spread

Object Literals  
Extensions

Template  
Literals

for .. of loop

Destructuring

Modules

# 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**

# 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 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.

# Object Literal Extensions

Simplified  
Syntax for field  
assignment

Simplified  
Syntax for  
functions

Dynamic field,  
property and  
function names

for... of

New for.. Of  
loop

Iterate over  
iteratables

Example

- `for(var item of names){ }`

# Template Literals

Allows  
interpolation on a  
String template.

Defined inside the  
back ticks (` `)

## Example

- `Number: \${number}`

Allows  
Expressions

# Destructuring

*Destructuring* is a convenient way of extracting multiple values from data stored in (possibly nested) objects and Arrays.

## Example:

- var arr = [2300, 3400, 6000]
- var [a, b, c] = arr;

## Example

- var employee = {id:1, name: “Anil”, location:“Mumbai”}
- var {id, name, location} = employee;

# 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  
}
```

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

two.js

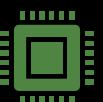
# Node.js



A server-side JavaScript platform



Created by Ryan Dahl in 2009



Provides an easy way to build scalable network applications



Built on top of Google V8 JavaScript Engine and libuv

# NPM(Node Package Manager)

- ❖ NPM is a package manager for the JavaScript programming language.
- ❖ Allows users to consume and distribute JavaScript modules that are available on the registry
- ❖ The default package manager for Node.
- ❖ Comprises of
  - ❖ Command line client, also called npm
  - ❖ An online database called the npm registry.
    - ❖ Public
    - ❖ Paid-for private packages
  - ❖ NPM website
- ❖ The package manager and the registry are managed by npm, Inc.

# NPM Packages

- ❖ Packages are reusable code published in the npm registry.
- ❖ A directory with one or more files a metadata file called package.json.
- ❖ A package is a building block that solves a specific problem.
- ❖ An application generally depends on many packages.
- ❖ Packages can be used on
  - ❖ Server side. Example: Express, Request
  - ❖ Client Side. Example Angular, React
  - ❖ Command based. Typescript, Angular CLI, JSLint

# NPM Commands

- ❖ NPM is installed with Node
- ❖ Check the version
  - ❖ `npm -v`
- ❖ Update npm
  - ❖ `npm install npm@latest -g`
- ❖ Find the path to npm's directory:
  - ❖ `npm config get prefix`
- ❖ Help
  - ❖ `npm -h`
  - ❖ `npm [command] -h`
  - ❖ `npm help [command]`
  - ❖ `npm help-search [key]`

# package.json

- ❖ A metadata file for the project
- ❖ Used to track dependencies
- ❖ Create Scripts
- ❖ Command to create package.json
  - ❖ npm init
- ❖ Setting defaults
  - ❖ npm set init-author-name 'Anil Joseph'
  - ❖ npm get init-author-name
  - ❖ npm config delete init-author-name

# Installing Packages

- ❖ Install to a project
  - ❖ npm install express
- ❖ Install and save to dependencies
  - ❖ npm install angular --save
- ❖ Install and save to development dependencies
  - ❖ npm install express --save-dev
- ❖ Install globally
  - ❖ npm install gulp -g

# Install Packages with version

- ❖ Specific version
  - ❖ npm install underscore@1.8.2
- ❖ Latest Version
  - ❖ npm install underscore@1.8.x
  - ❖ npm install underscore@1.8
  - ❖ npm install underscore@1.8.2
  - ❖ npm install underscore@1.x
  - ❖ npm install underscore@1
  - ❖ npm install underscore
- ❖ Other Options
  - ❖ npm install underscore@">=1.4.x < 1.6.x"

# Listing & Removing Package

- ❖ npm list
  - ❖ List all packages
  - ❖ Shows all dependencies of packages
- ❖ npm list --depth 1
  - ❖ List packages show dependency depth to 1 level
- ❖ npm list --global true
  - ❖ List global packages
- ❖ npm list --global true --depth 0
  - ❖ List global packages show dependency depth to 0 level

# Listing & Removing Package

- ❖ `npm list --dev true`
  - ❖ List development dependiecies
- ❖ `npm list --prod true`
  - ❖ List Production dependencies
- ❖ `npm ls`
  - ❖ ls as shortcut
- ❖ `npm uninstall underscore/ npm rm underscore/ npm un underscore`
  - ❖ Removes/Uninstall a package
- ❖ `npm uninstall underscore --save`
  - ❖ Uninstalls and updates the package.json
- ❖ `npm uninstall underscore -g`
  - ❖ Unstall global package

# TypeScript

# TypeScript

---

TypeScript is programming language developed and maintained by Microsoft.

---

TypeScript is a typed superset of JavaScript.

---

*Transcompiles* to JavaScript.

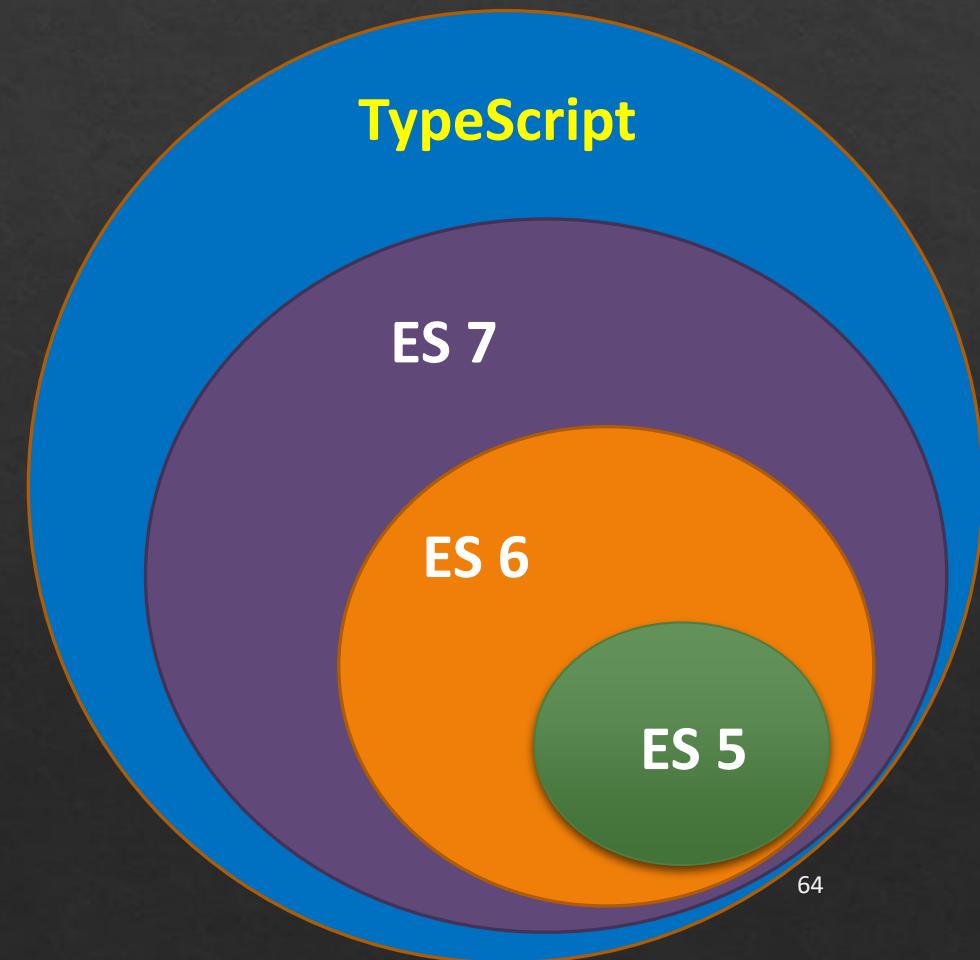
---

Designed for development of large applications.

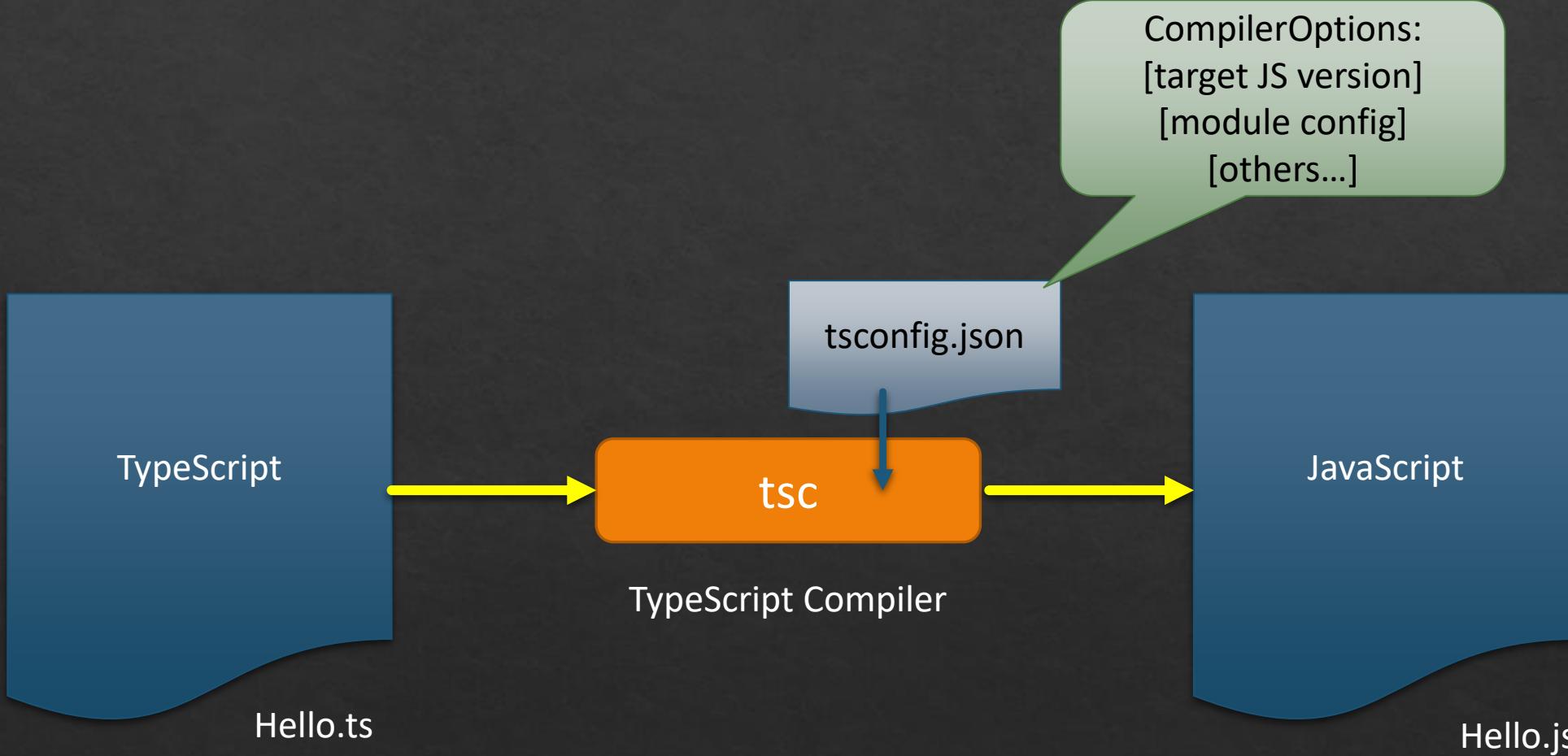
---

Open Source.

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



# TypeScript



# TypeScript Features

Type Annotations

Compile-Time  
Type Checking

Type Inference

Interfaces

Classes &  
Inheritance

Namespaces and  
Modules

Generics

Decorators

Arrow Functions

# TypeScript Installation

## Node.js

- npm install -g typescript

## Available with

- Visual Studio 2015 & 2017
- Visual Studio Code

## Plugins available with

- Eclipse
- Atom
- Sublime
- WebStorm
- Many more

# 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];

# 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"];

# TypeScript Types

void

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

Null and Undefined

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

# 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)

# 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.

# 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

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

# 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

# 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  
}
```

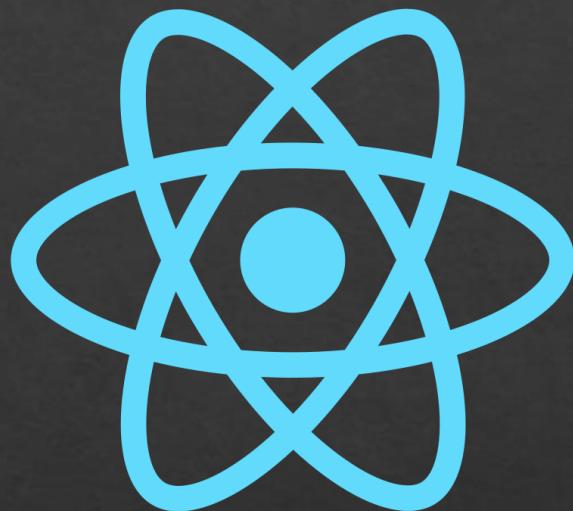
```
import {foo, bar} from './one';  
  
foo();  
bar();
```

two.js

# NPM Package vs Module

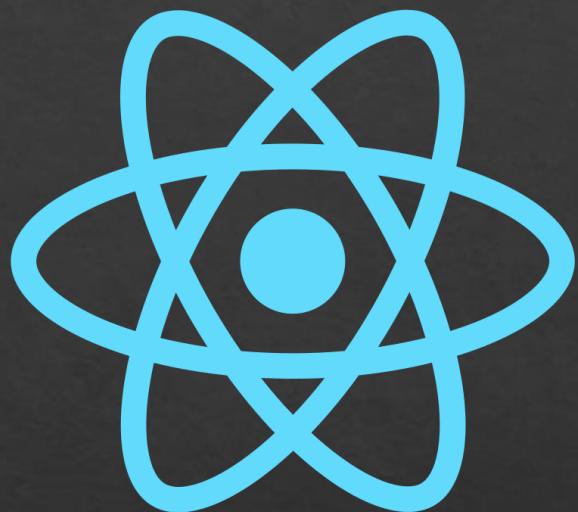
- ❖ Module
  - ❖ Single JavaScript file with some reasonable functionality
- ❖ Package
  - ❖ A directory with one or more modules
  - ❖ package.json file contains metadata about the package

# What is React?



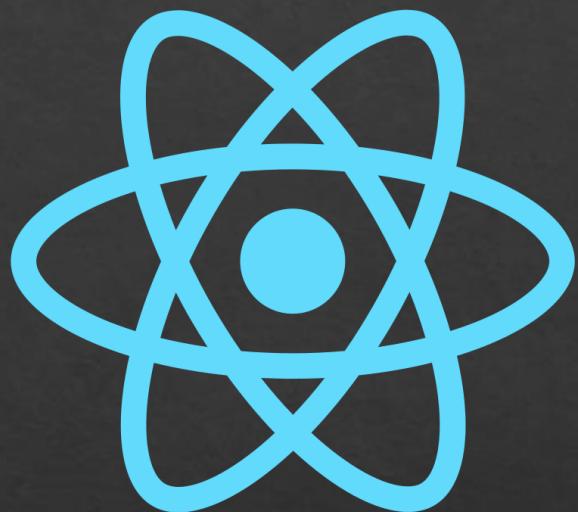
- ❖ A JavaScript library to build *User Interfaces*.
- ❖ Building applications for the *browser*.
- ❖ React allows to create custom HTML elements called *components*.
- ❖ Components are used to build *complex User Interface*
- ❖ Reacts promotes writing *maintainable, manageable and reusable code*.

# 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

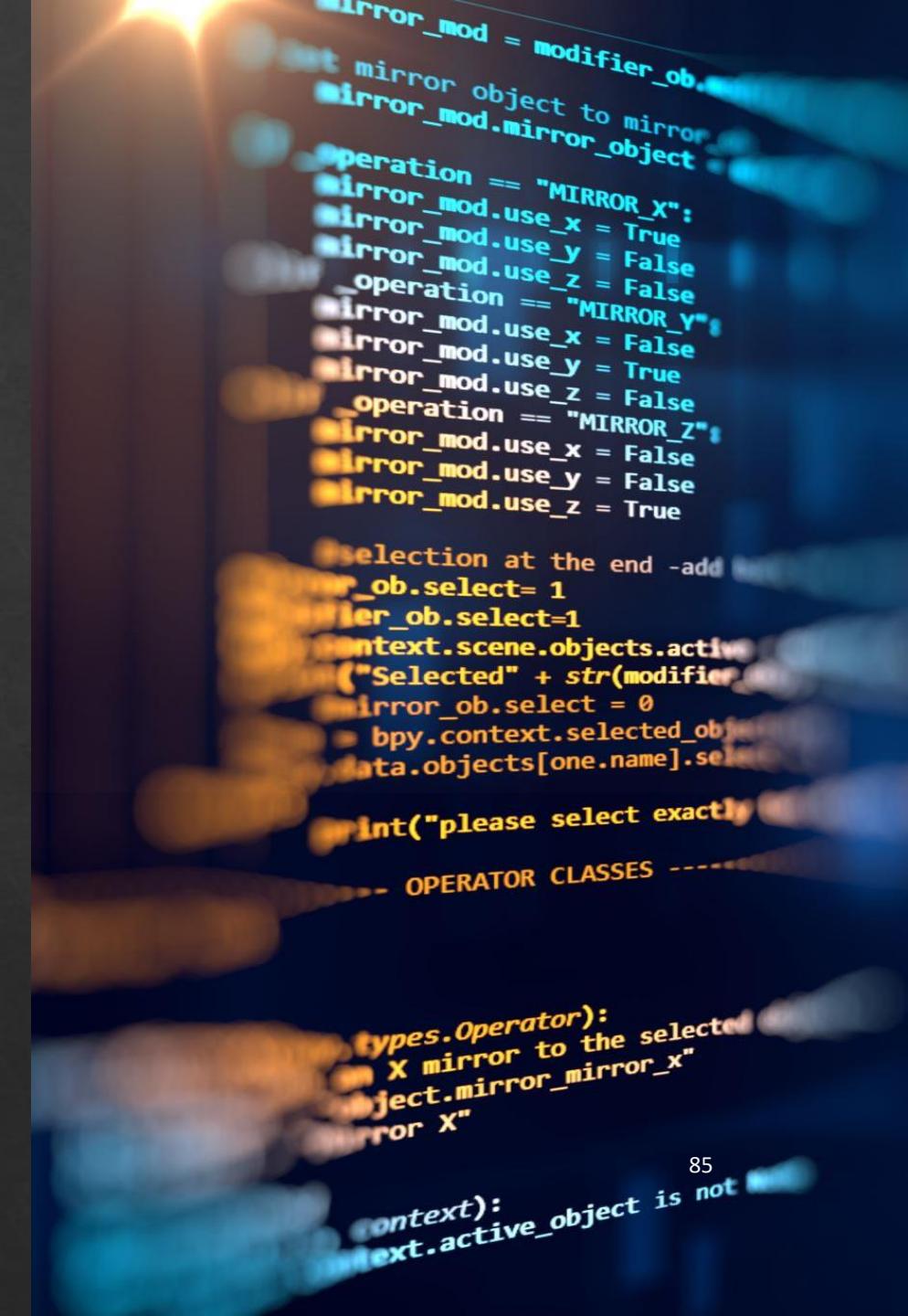
- The compiler for writing next generation JavaScript.
- Compiles JSX to JavaScript

# 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.

Presentaed by Anil Joseph(anil.jos@gmail.com)



# 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

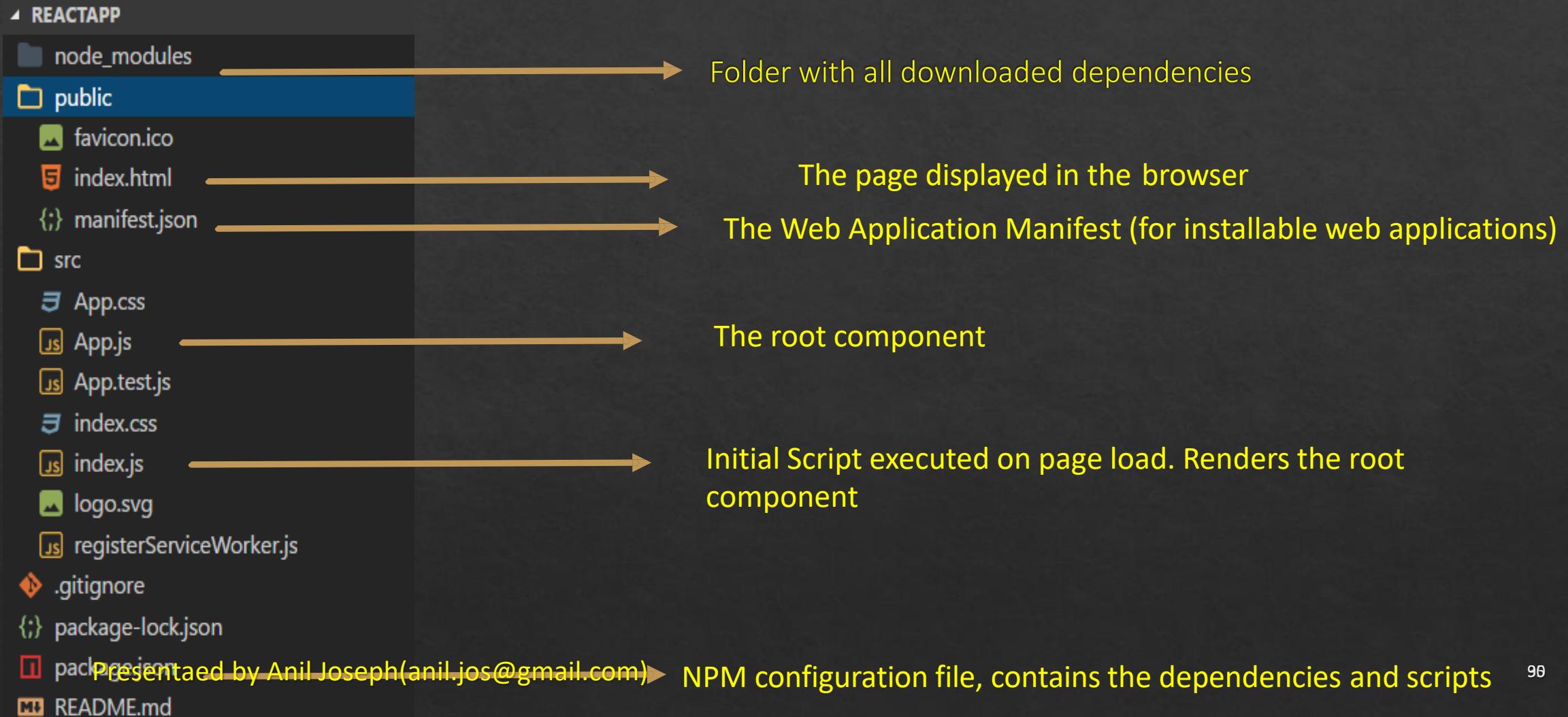
- **npx create-react-app the-react-app**

3

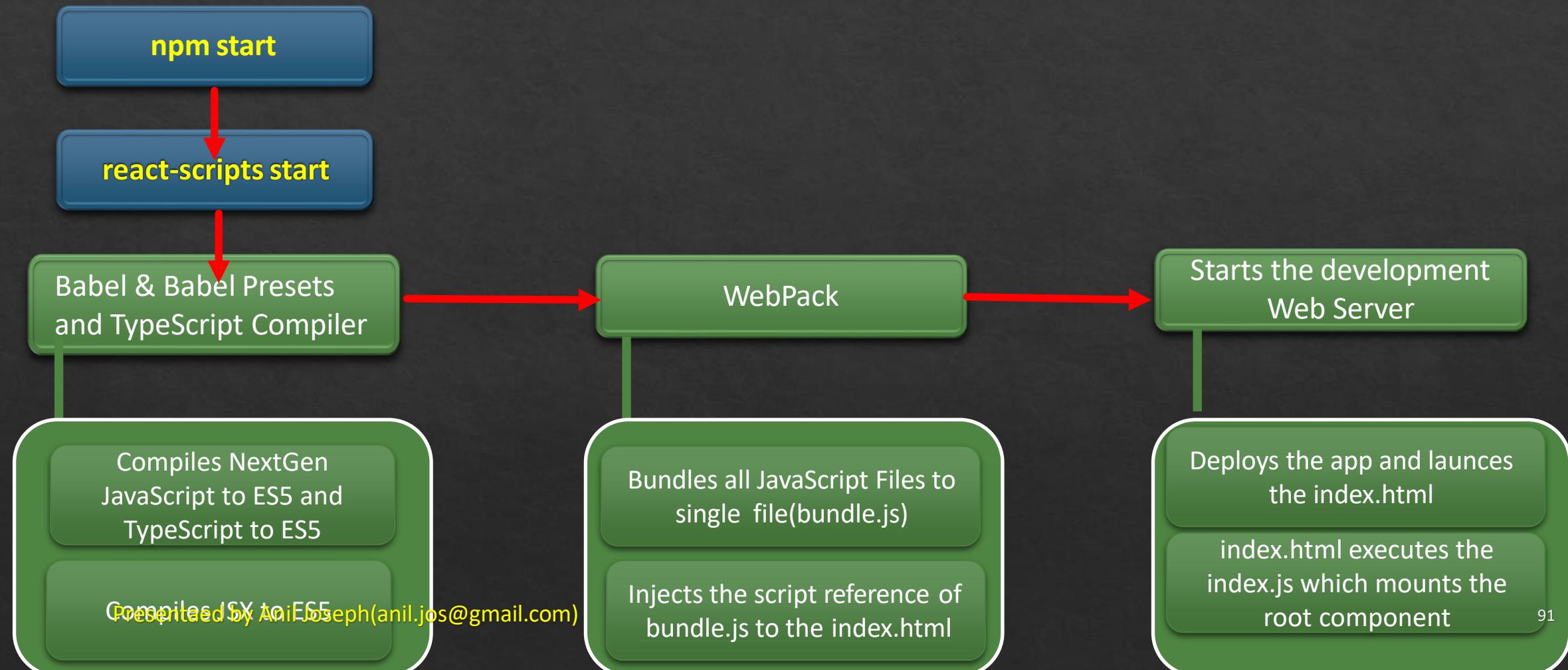
## Start the Application

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

# Project Structure

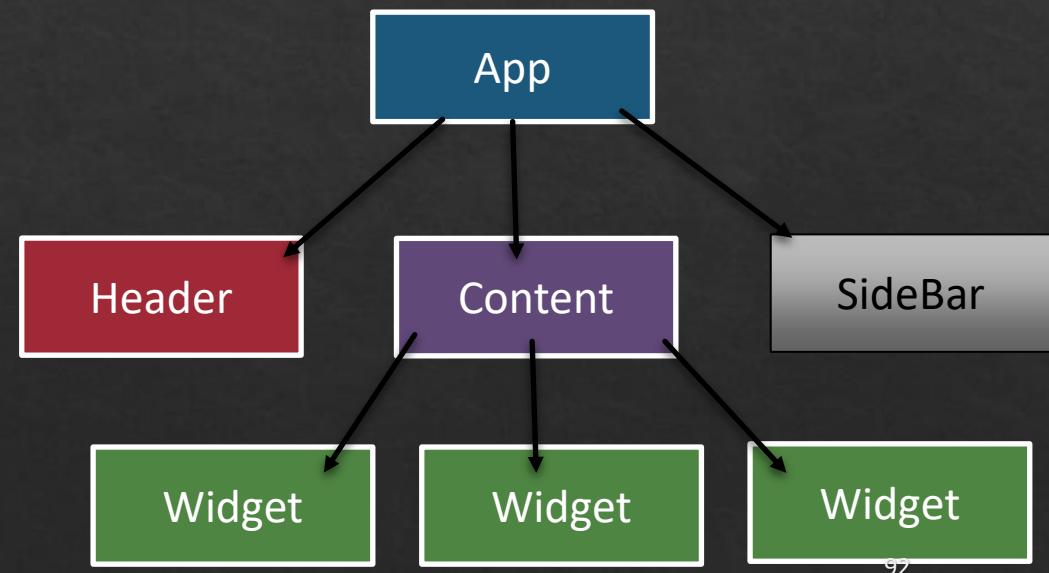
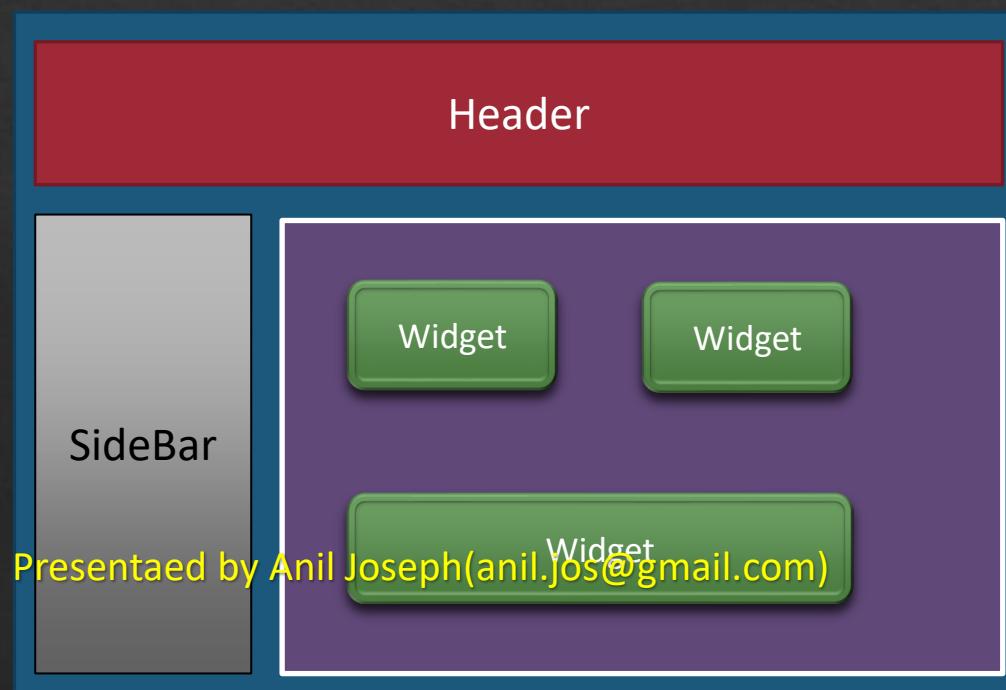


# Project Execution



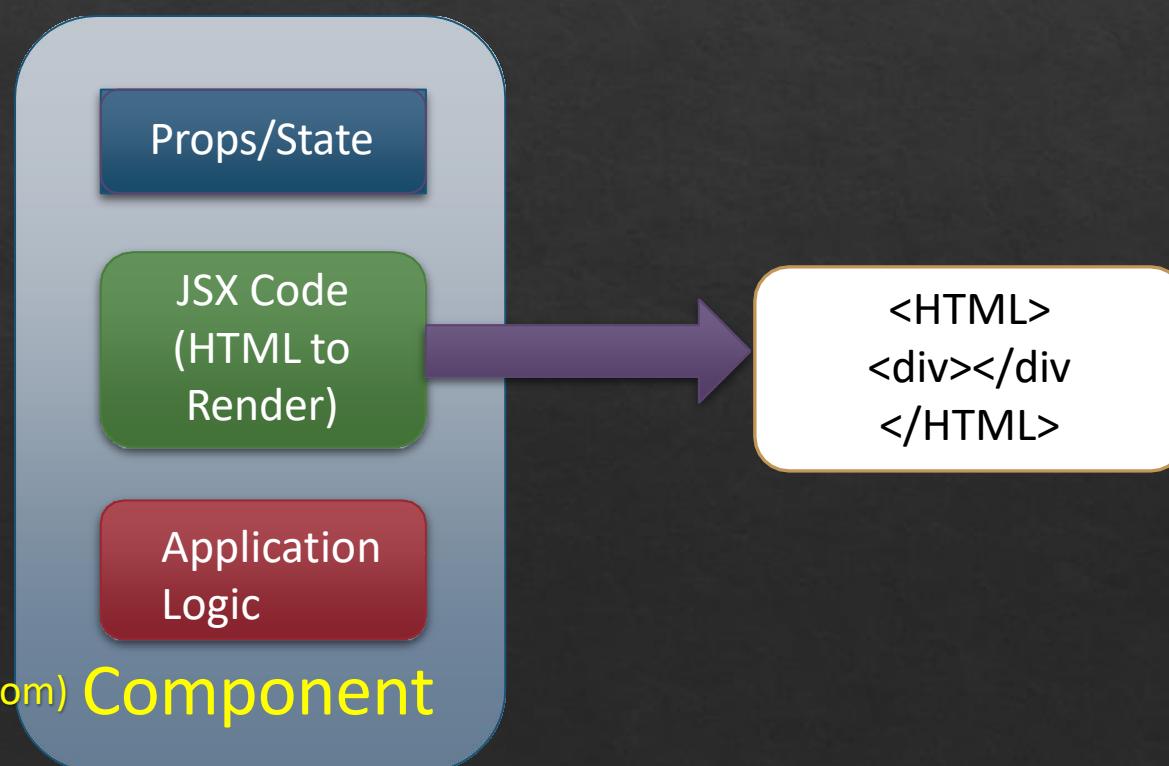
# React Components

- ❖ 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() }`

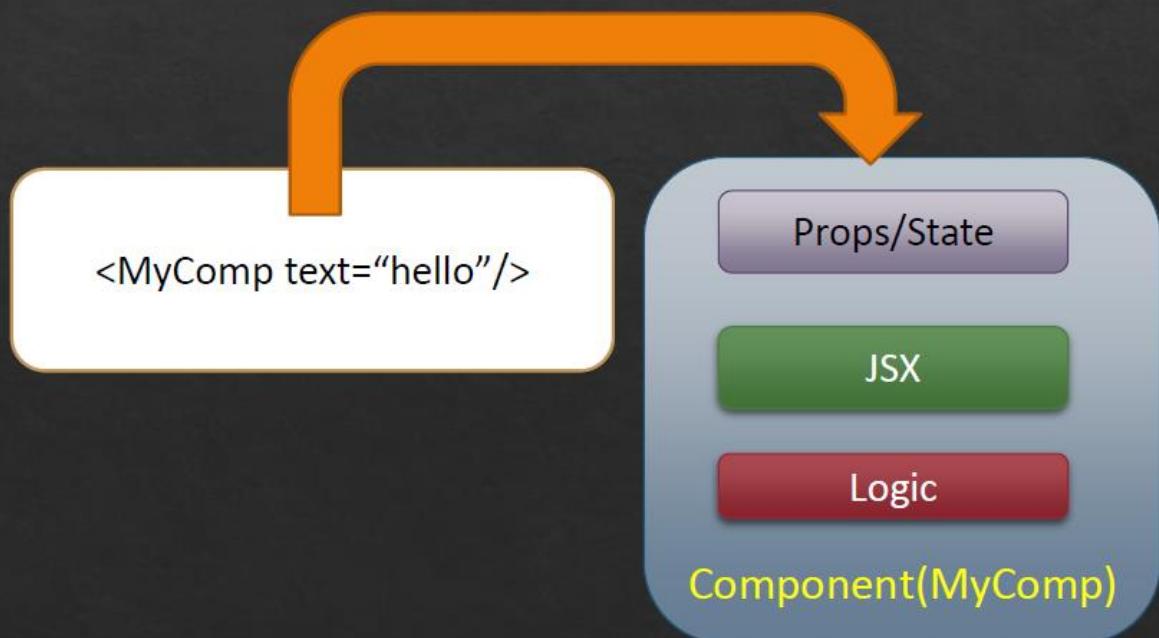
# Components: Properties(props)

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

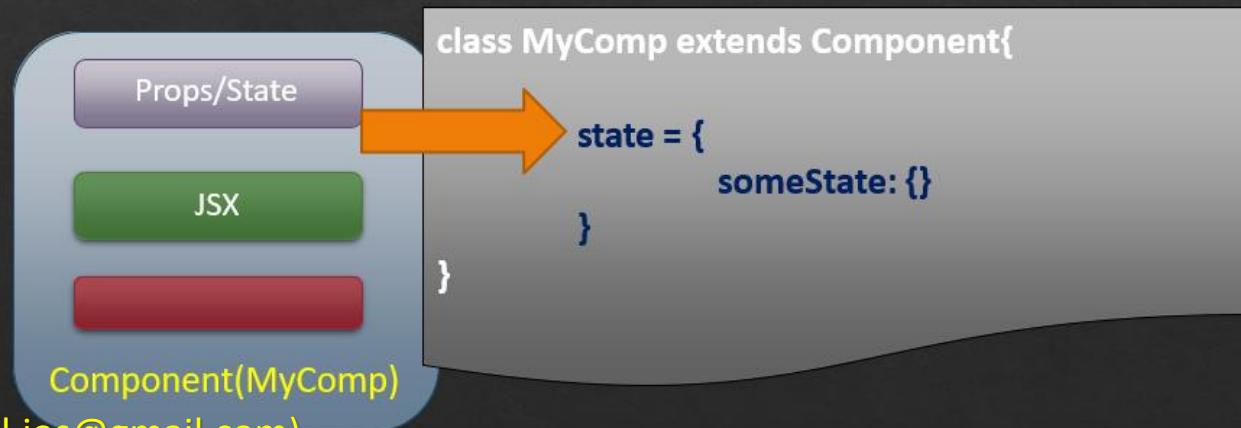
State is used when a component needs to keep track of information between renderings.

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

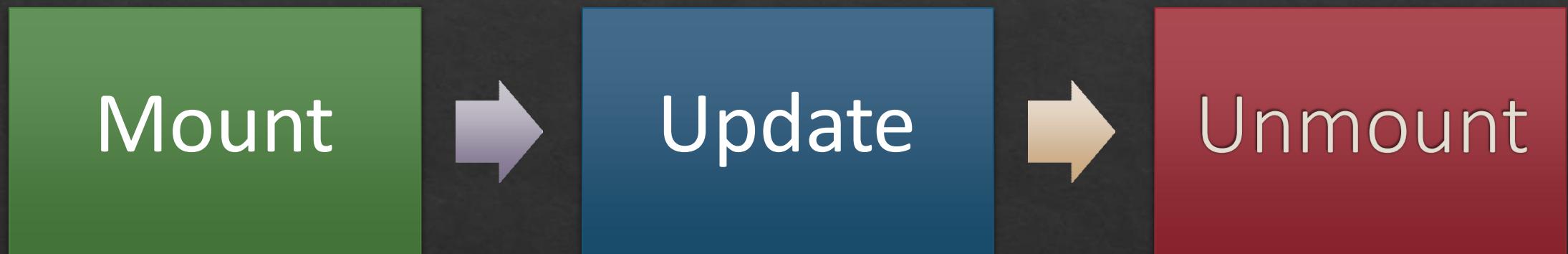
props allow you to pass data from a parent (wrapping) component to a child (embedded) component.

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';`

# 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

Presentaed by Anil Joseph(anil.jos@gmail.com)

## <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

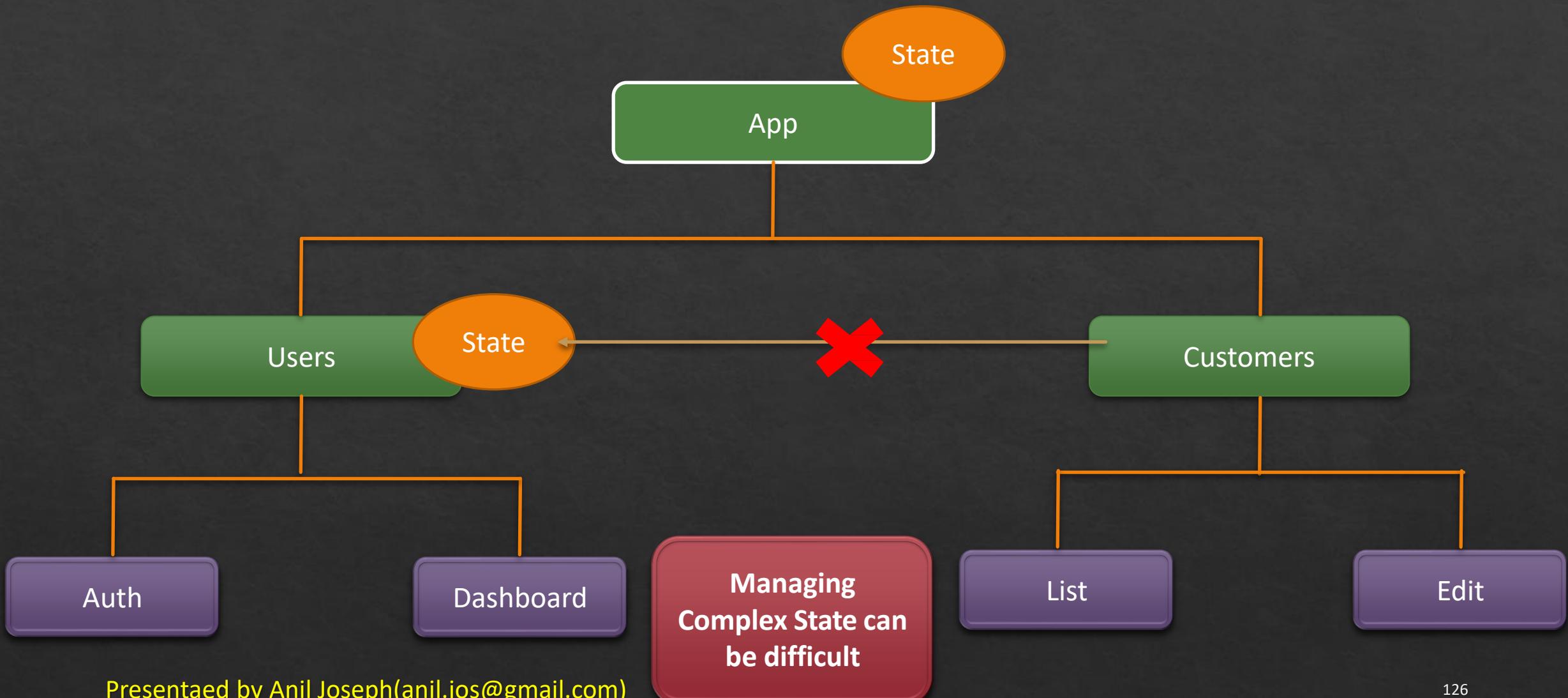
# Redux

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

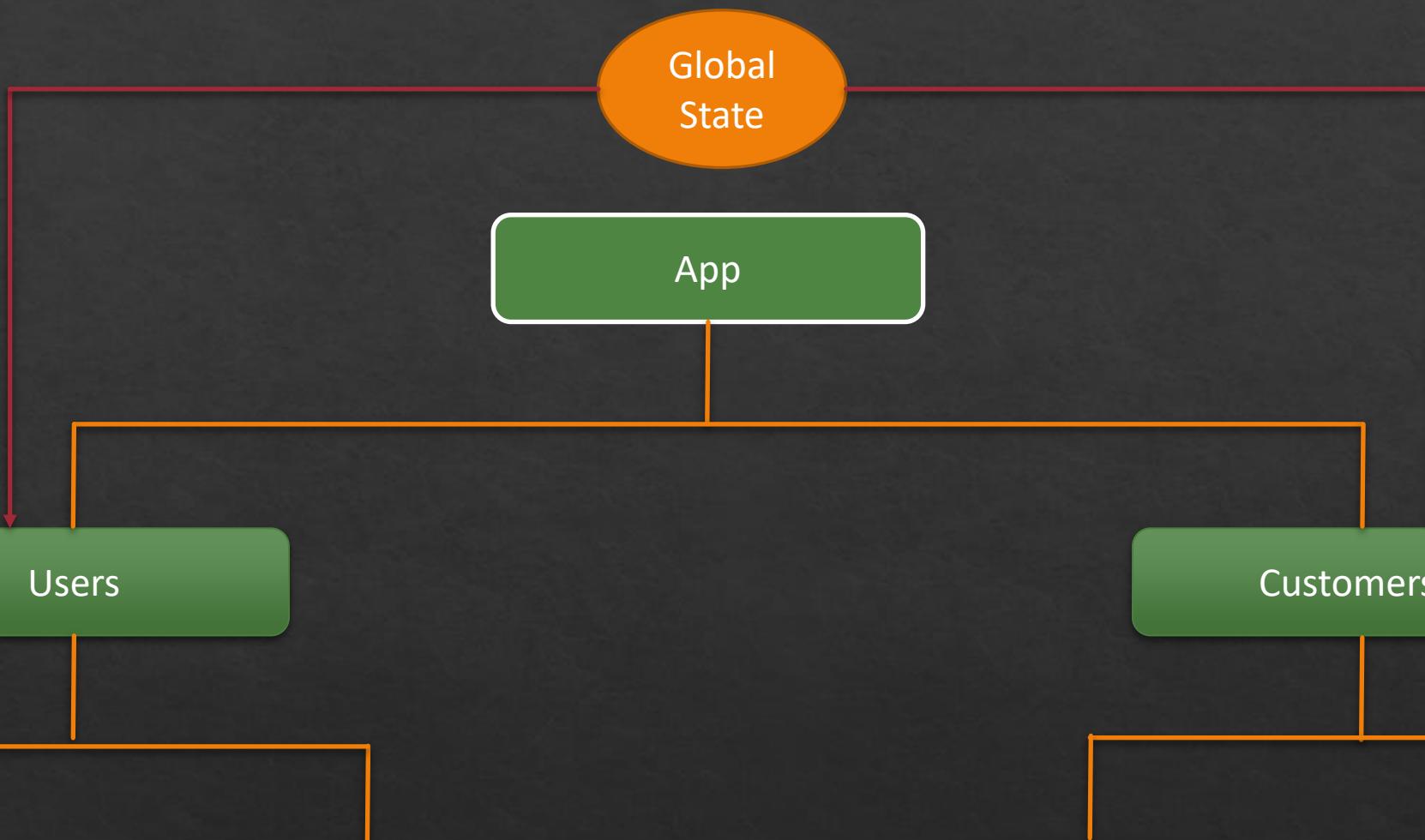
It is primarily used together with React or Angular for building user interfaces.

Redux was built on top of functional programming concepts.

# Why Redux?



# Why Redux?



Auth

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

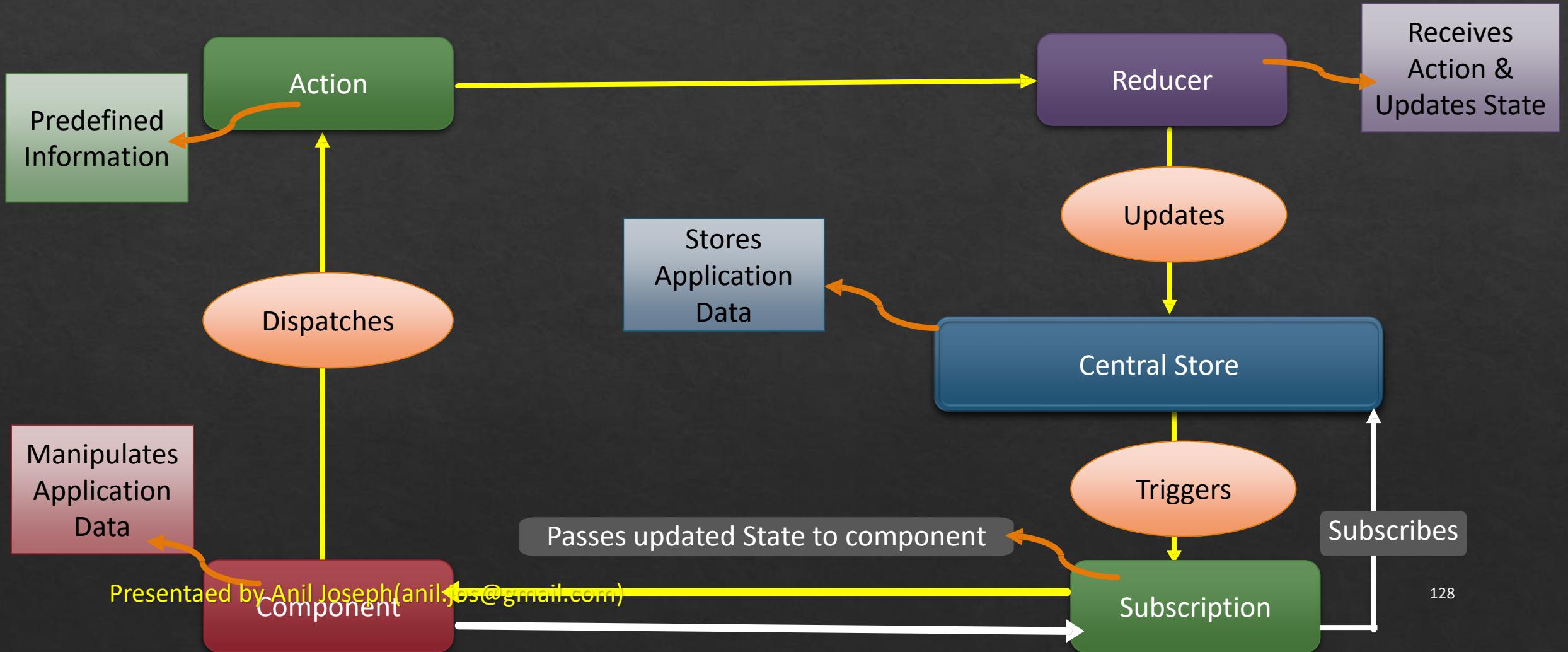
Dashboard

List

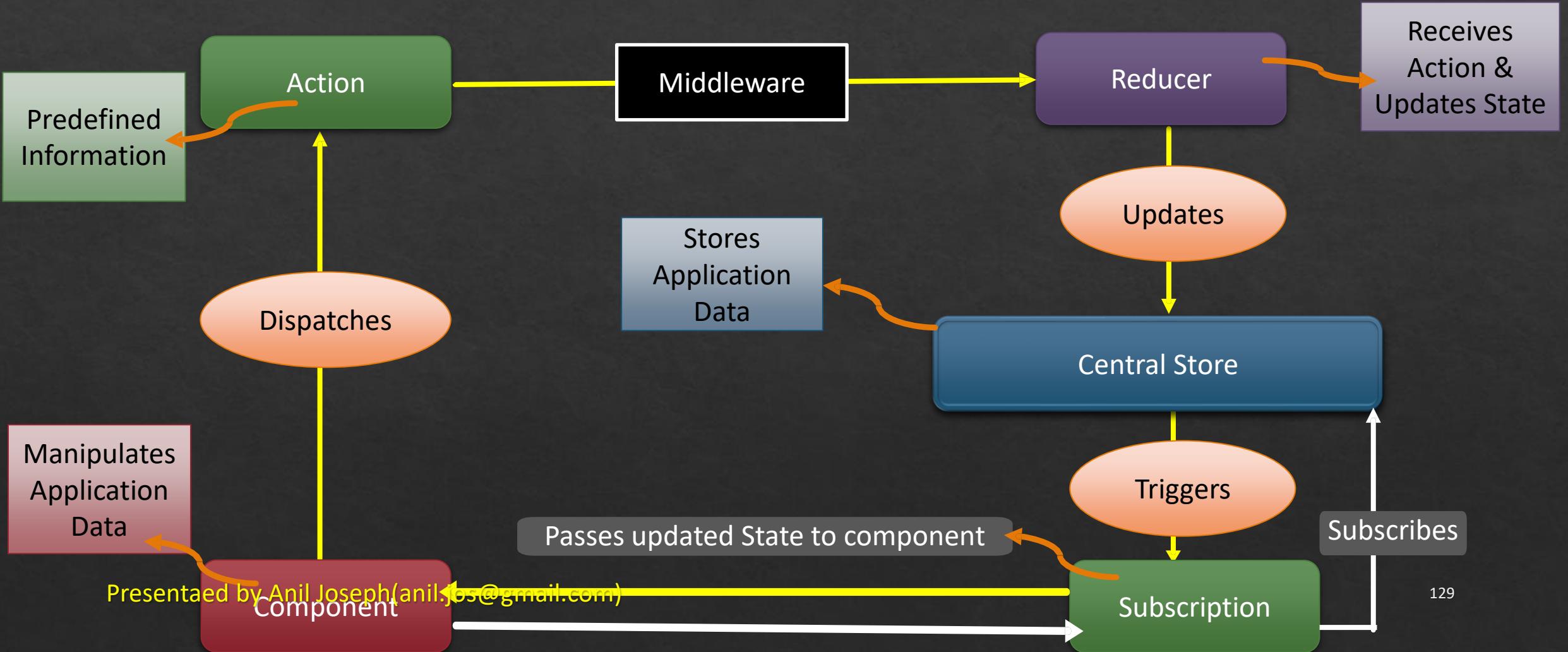
Edit

127

# 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`

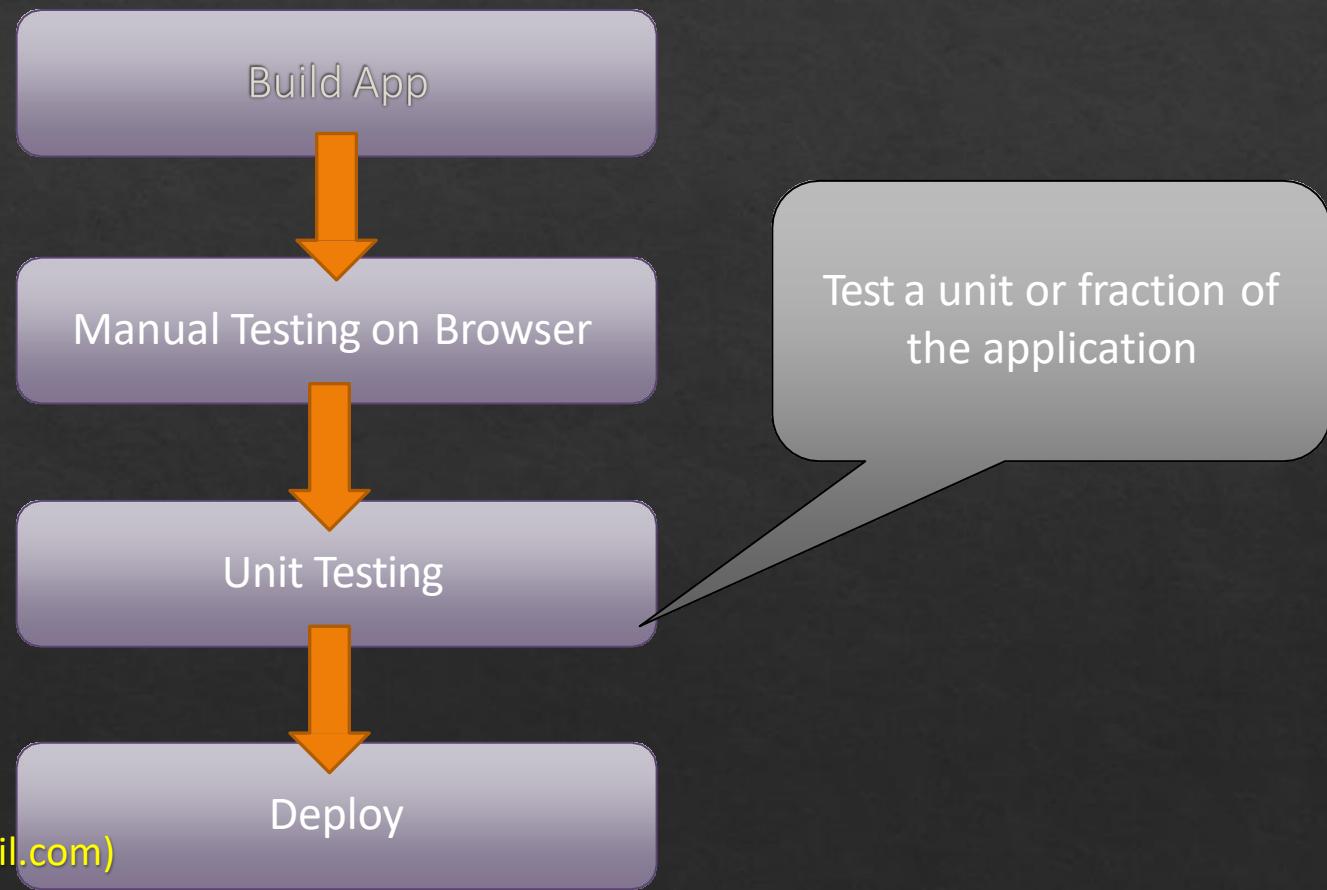
# Code Splitting

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

# 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

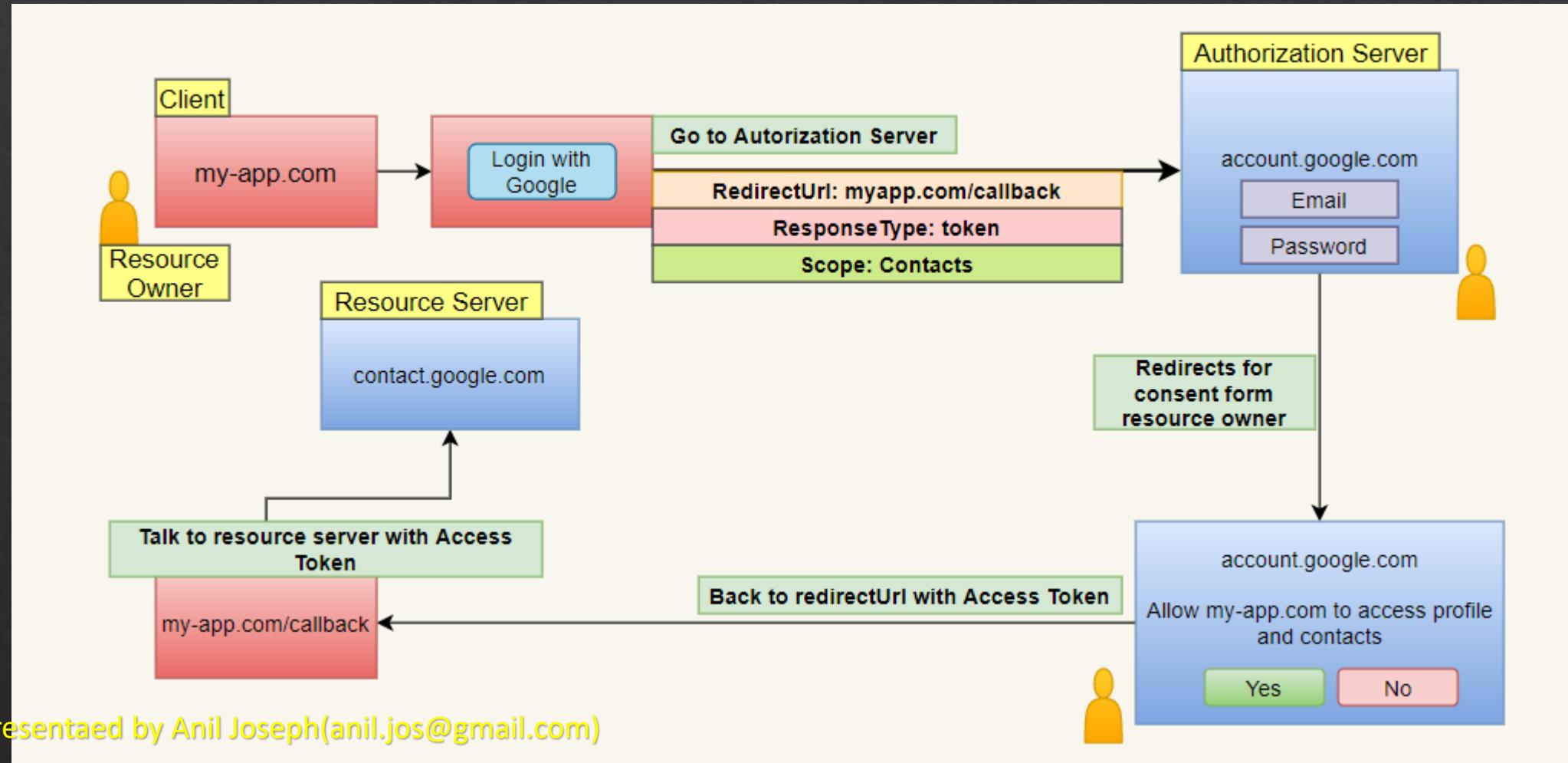
Server must Always serve index.html

Deploy artifacts to Web Server

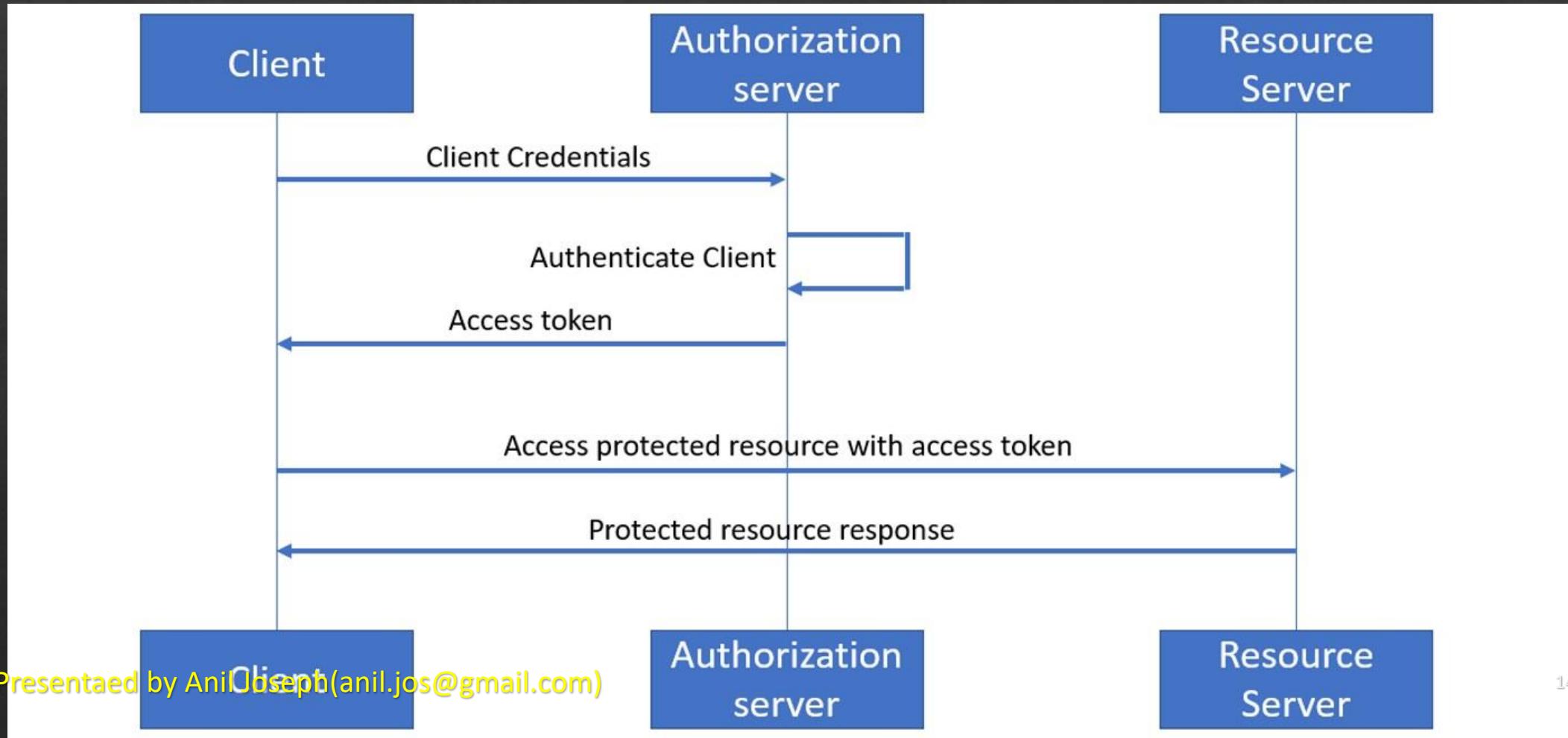
# OAuth 2.0

- ❖ 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.

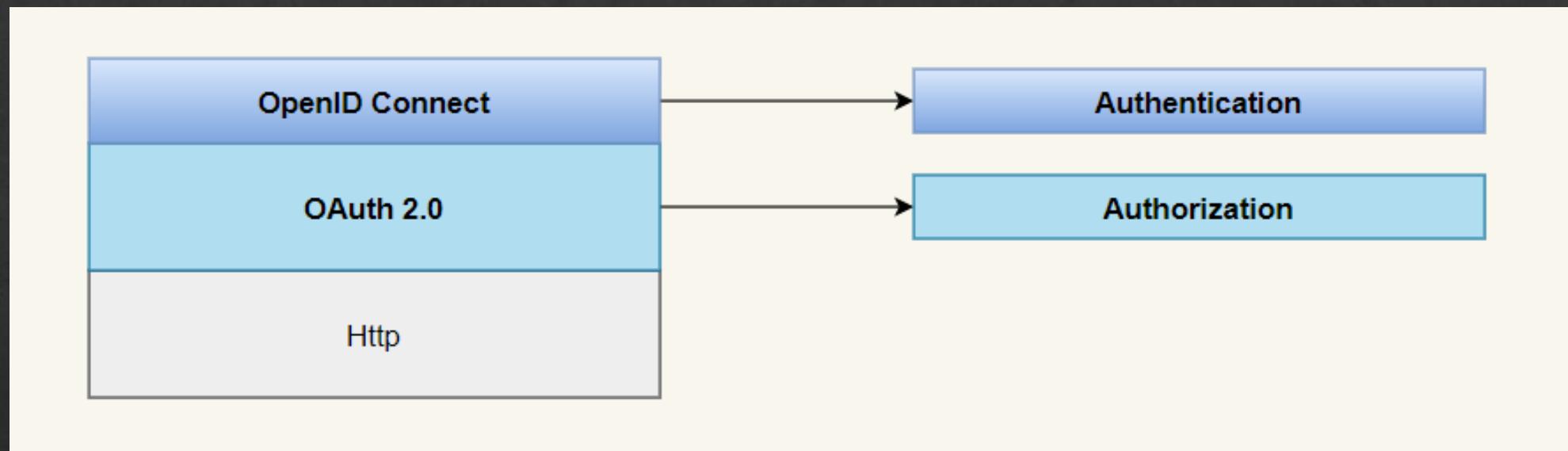
# OAuth 2.0



# OAuth 2.0



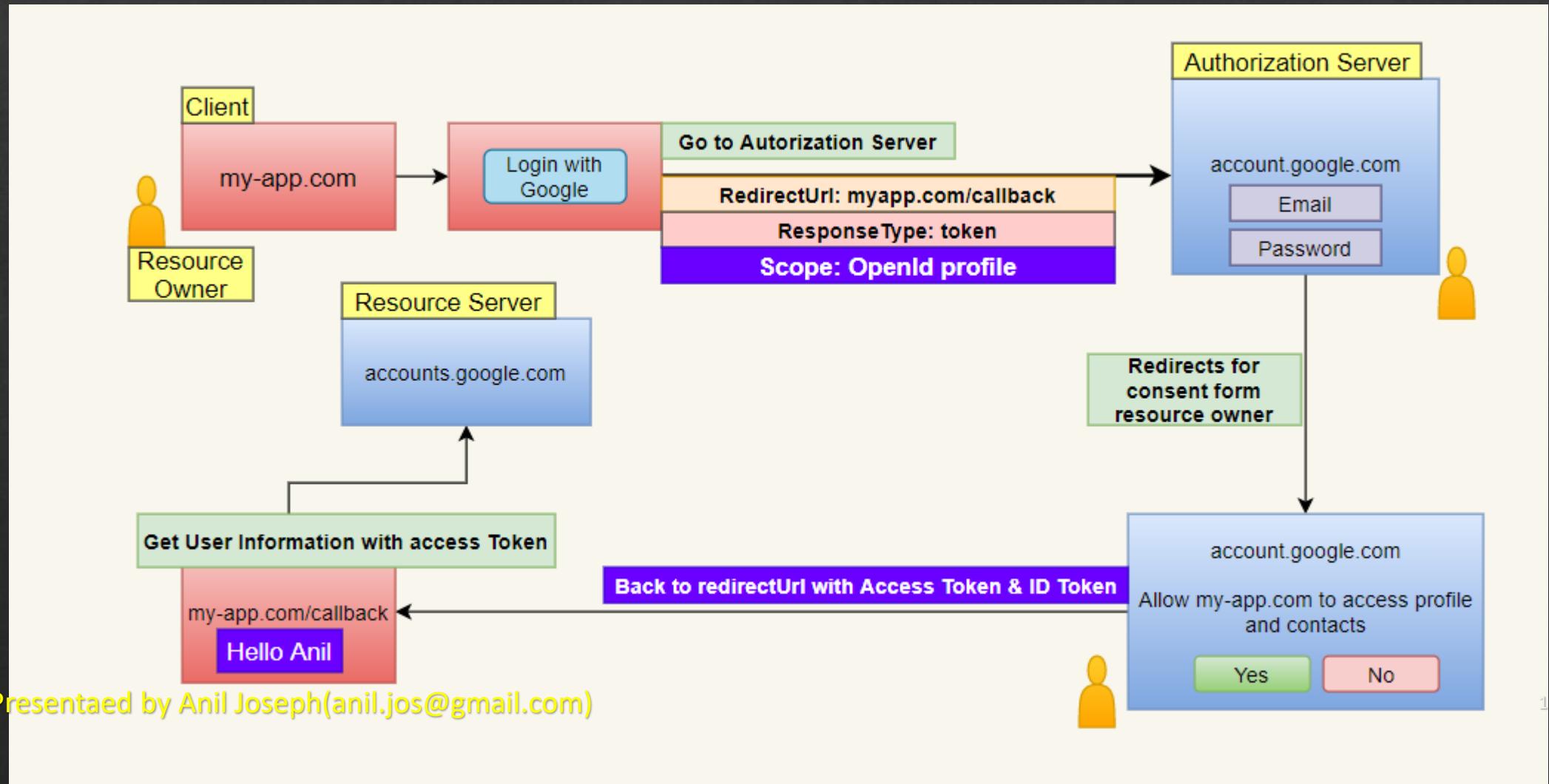
# OpenID Connect



# 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
- ❖ User for Authorization and Information Exchange

# JWT

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.

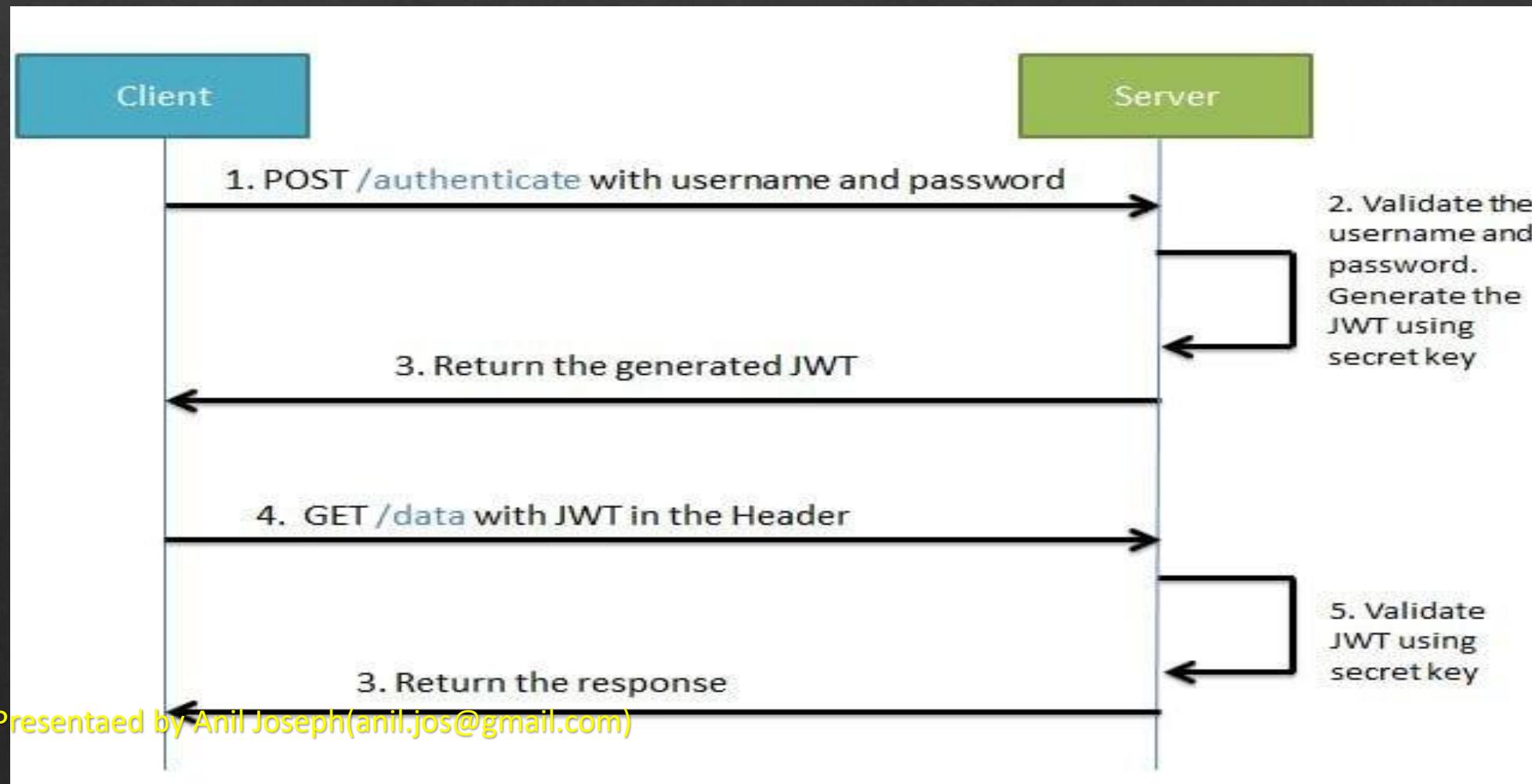
eyJpc3MiOiJPbmxbmUgSldeJ1aWxkZXIiLCJpYXQiOjE2  
MDA5OTkzMAsImV4cCI6MTYzMjUzNTMwMCwiYXVkljoi  
d3d3Lm15LWFwcC5jb20iLCJzdWIiOiJhbmlsLmpvc0BnbWF  
pbC5jb20iLCJHaXZlbk5hbWUiOiJBbmlsliwiU3VybmFtZSI6I  
kpvc2VwaClslkVtYWlsIjoiYW5pbC5qb3NAZ21haWwuY29tl  
iwiUm9sZSI6IkNvbnN1bHRhbnQifQ.

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



# Progressive Web Applications

- ❖ PWAs are web apps developed using a number of specific technologies and standard patterns to allow them to take advantage of both web and native app features.
- ❖ Key Features
  - ❖ Discoverable
  - ❖ Installable
  - ❖ Linkable
  - ❖ Network Independent
  - ❖ Progressively Enhanced
  - ❖ Re-engageable
  - ❖ Responsive
  - ❖ Secure

# Progressive Web Applications

- ❖ Service Workers
  - ❖ Service workers essentially act as proxy servers that sit between web applications, the browser, and the network (when available).
  - ❖ Enable the creation of effective offline experiences
  - ❖ Allow access to push notifications and background sync APIs
- ❖ Manifest
  - ❖ Lists all the information about the website in a JSON format
  - ❖ Information includes: Application Name, Path, Icons
- ❖ Https
  - ❖ Applications should be served using the Https protocol

# 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)