**React Documentation**

What is React?

React is an open-source JavaScript library for building user interfaces.

Not a framework

Focus on UI

Why React?

Created and maintained by Facebook.

More than 100k starts on GitHub.

Huge community.

In demand skillset.

Component Based Architecture

1. Header
2. SideNav
3. MainContent
4. Footer

Advantages of React

Components make it possible to write reusable code.

React is declarative.

Seamlessly integrate react into any of your application.

Portion of your page or a complete page or even an entire application itself.

React native for mobile application.

Prerequisites:

HTML, CSS, and JavaScript fundamentals.

ES6

JavaScript: ‘this’ keyword filter, map and reduce.

ES6: let & const, arrow functions, template literals, default parameters, object literals, rest and spread operators and destructing assignment.

Plan Content:

Fundamentals

HTTP

Routing

Redux

Utilities

Create Application:

Node and text editor

Node: latest version

Text editor: VS code

Create React App:

Run a command and entire react project is created for you.

Approach 1:

Step 1: npx create-react-app <project\_name>

Npx package runner

Approach 2:

Step 2 :npm create-react-app -g

Create-reat-app<project\_name>

Install package globally.

Folder Structure

Root folder consists of 3 folders and 4 files.

1. Package.json file

Contains all dependencies and scripts required for project.

1. Node\_modules folder

Contains all installed dependencies.

1. Public folder

Contains 3 files.

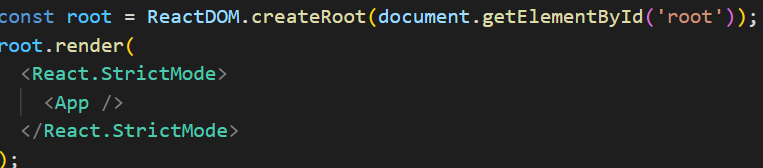
Index.html is only one html file in react project. Hence, called as single page application.



1. Src Folder

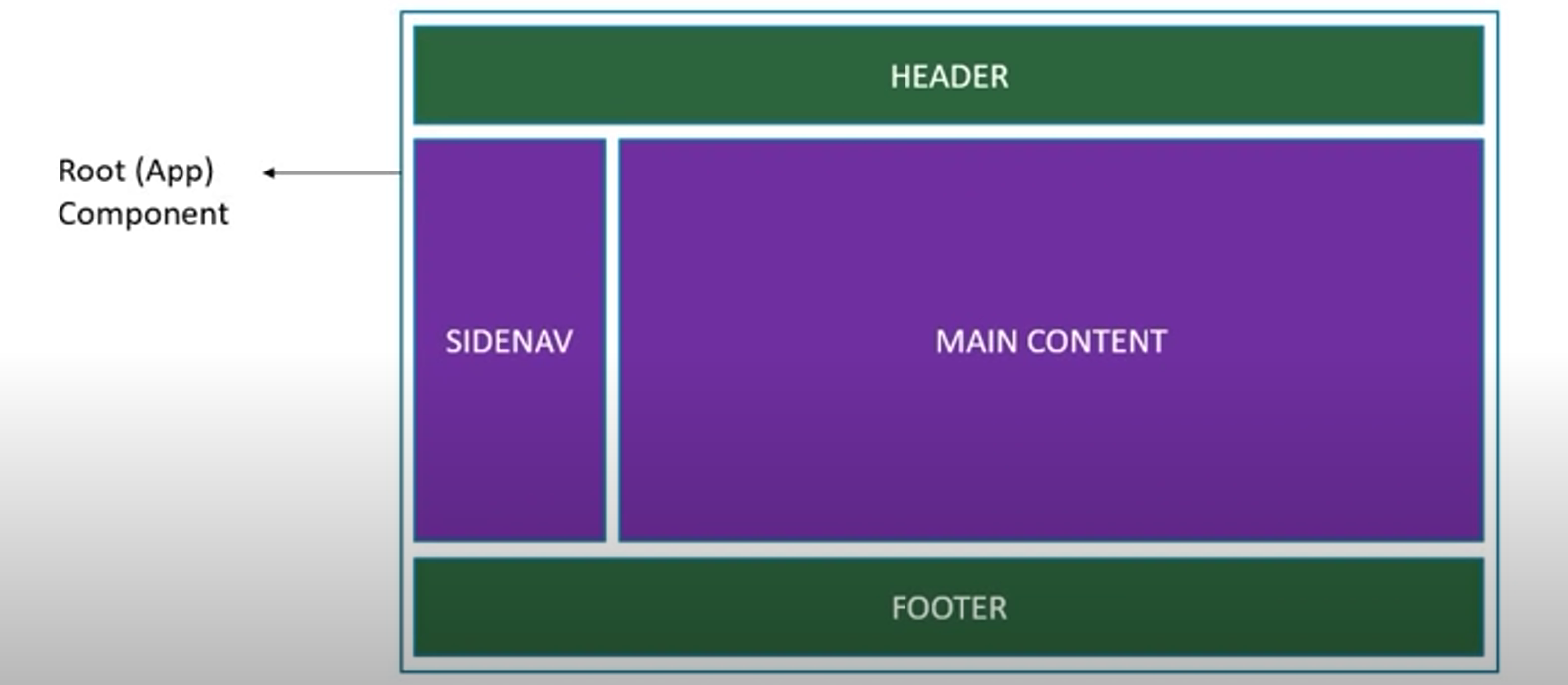
Folder working on most during development.

1. Index.js: Specify root component which is app component Dom element which will be controlled by react. Dom element in index.js is an element with an ID root which is present in index.html.



Components

Components let you split the UI into independent, reusable pieces and think about each piece in isolation.



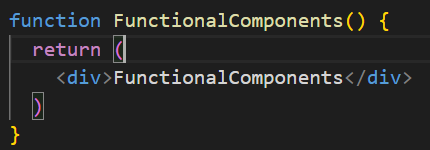
Components are reusable.

Components can contain other components.

Component Type

**Stateless Functional Component**

JavaScript Functions.



Arrow function :

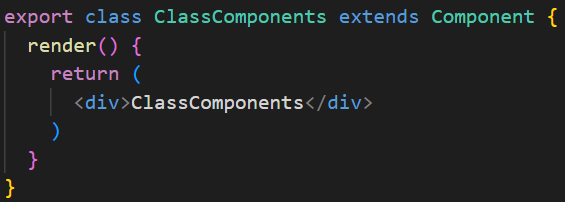


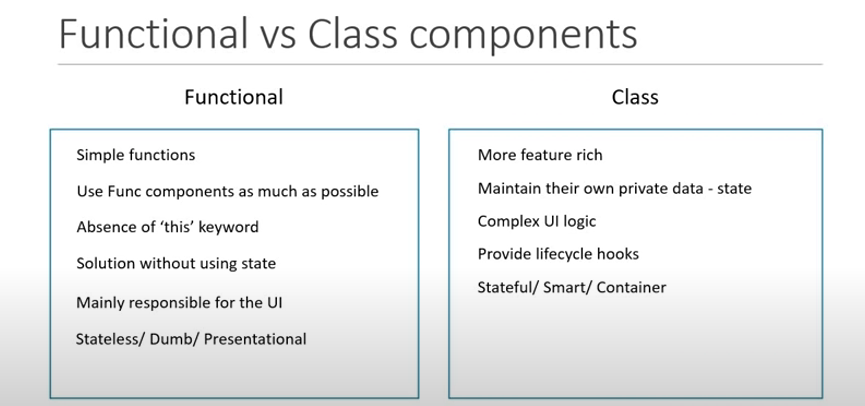
Note: If export const is used then use import { constname} from ‘path’ or else normal only.

**Stateful Class Component**

Class extending Component Class

Render method returning HTML.





JSX

JavaScript XML(JSX) – Extension to JavaScript language syntax.

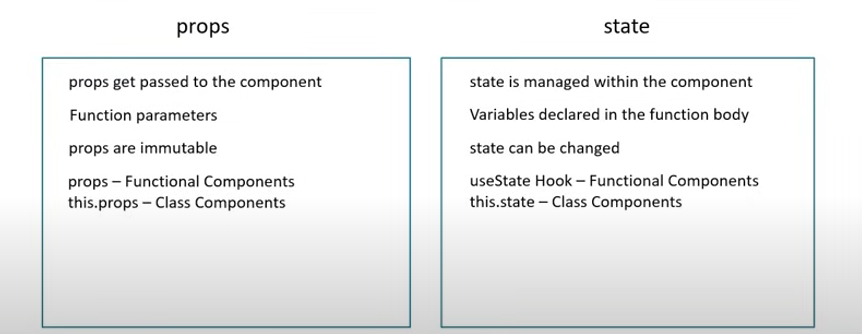
Props

Props: Properties are optional input that a component can accept that allows component to be dynamic.

Props are immutable cannot be changed.

State

State: Second way to find out what is rendered on the screen.



Changing the state object

**This.setState():** To change the value in the state object, use this method.

setState has two arguments (object, call back function)

* In call back function code for updated state is placed.
* Why is prevState used?

It allows to access the state value from last render cycle.

* To update state based on the previous state value, pass in a function as an argument instead of the regular object.

Destructing Props and State

Destructing makes the code much clearer. We must use this keyword throughout the program but using restricting, we discard by assigning them in new variable.

Event Binding

Need to bind events so that “this” keyword would not return an undefined.

List and Keys

**Lists**: Are designed to hold a collection of elements with flexibility to handle different data types.

**Keys**: Using keys, it’s helpful to update remove an item rather than changing entire list.

* Keys cannot be used as a prop.
* When there is no unique id as key index can be used as key but list should be static and should not be reordered or filtered.

CSS Styling Components

Using Stylesheets: Template literal syntax for using two classes under same div {`${attribute-name} class-name`}

Lifecycle Methods

**Mounting:** When an instance of a component is being created and inserted into the DOM.

*Methods are constructor, static getDerivedStateFromProps, render and componentDidMount*

**Updating:** When a component is being re-rendered because of changes to either its props or state.

*Methods are static getDerivedStateFromProps, shouldComponentUpdate, render, getSnapshotBeforeupdate and componentDidUpdate*

**Unmounting:** When a component is being removed from the DOM.

*Method is componentWillUnmount()*

* Method is invoked immediately before a component is unmounted and destroyed.
* Cancelling any network requests, removing event handlers, cancelling any subscriptions and invalidating timers.
* Do not call the setState method.

**Error Handling:** When there is an error during rendering, in a lifecycle method, or in the constructor of any child component.

*Methods are static getDerivedStateFromError(error) and componentDidCatch(error, info)*

When there is an error either during rendering, in a lifecycle method, or in the constructor of any child component.

Fragments

To avoid the placing of an extra div React.Fragment or <>………</> is used where the presence of empty div can be avoided.

Pure Component

A pure component implements shouldComponentUpdate with shallow props and state comparison.

This component does not re-render.

Unnecessary re-render can be one plus point with pure component results with good performance.

Never mutate the state. Always return a new object that reflects state.

This is possible for class component.

Memo

This component does not re-render same as that of pure component, but this is for functional component. This improves performance.

React Hooks

1. useState Hook: The React useState Hook allows us to track state in a function component.

useState in array/object spread operator is used because useState cannot merge data like state in class component hence spread operator is used.

1. useEffect Hook: Runs after every render