Index

Contents

[1. Component 2](#_Toc160908576)

[2. Basic component syntax 2](#_Toc160908577)

[3) Virtual Dom 3](#_Toc160908578)

[4) Install react library 3](#_Toc160908579)

[5) In Vs code install 3](#_Toc160908580)

[6) Create new react app 3](#_Toc160908581)

[7) install bootstrap 4](#_Toc160908582)

[8) props vs state 4](#_Toc160908583)

[9) Raising and handling events 5](#_Toc160908584)

[10. SinglesourceofTruth 5](#_Toc160908585)

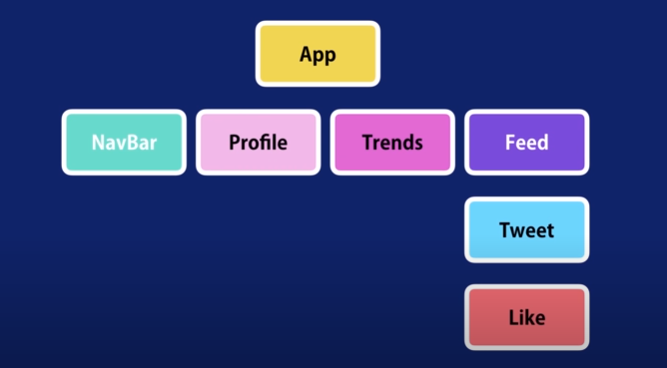
[12. Component Sync 5](#_Toc160908586)

[13. LifeCycle hooks 6](#_Toc160908587)

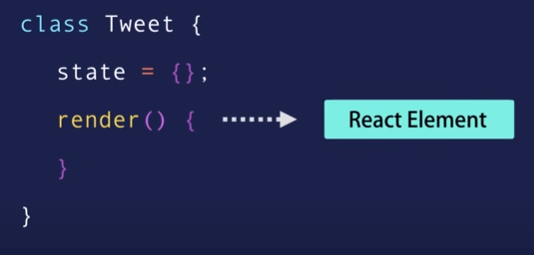
[14. ES6 Basics 7](#_Toc160908588)

1. Component: Component are core building block and reusable web component.
2. Virtual Dom:

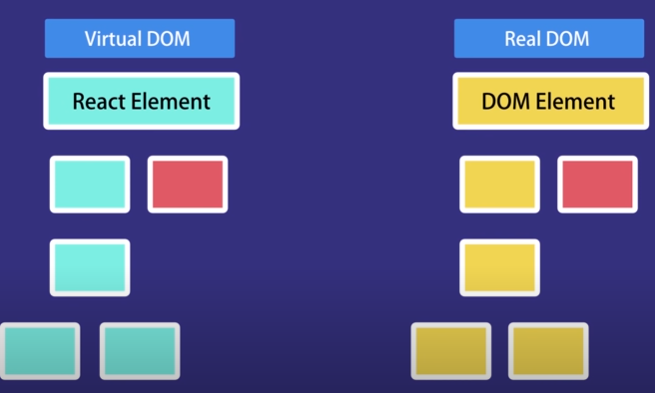
1. Component Hierarchy



1. Basic component syntax



3) Virtual Dom



4) Install react library

C:\Users\kelz>npm i -g create-react-app@4.0.0

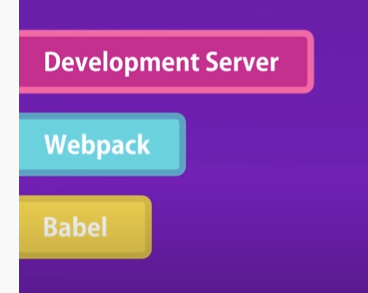
5) In Vs code install

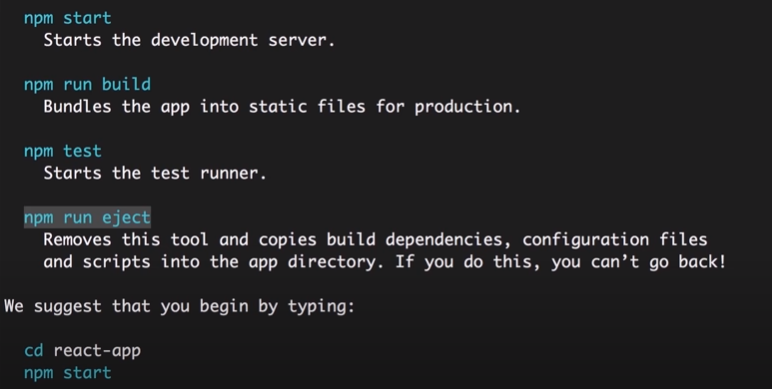
Install extensions

1. Simple react snippets
2. Prettier

6) Create new react app

>create-react-app react-app





7) install bootstrap

>npm i [bootstrap@4.5.3](mailto:bootstrap@4.5.3)

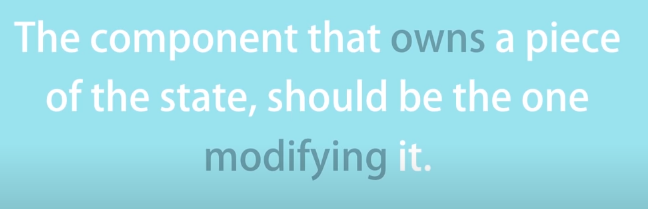
8) props vs state

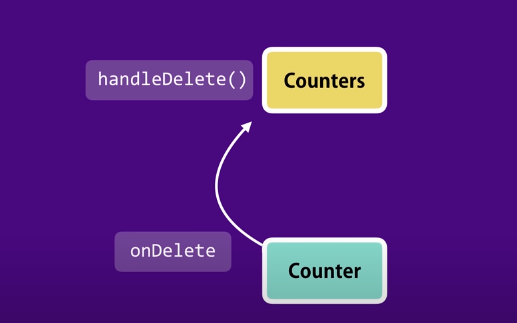
Props - passing data between components. It’s read only

State – data private to that component

9) Raising and handling events

Thumb Rule:

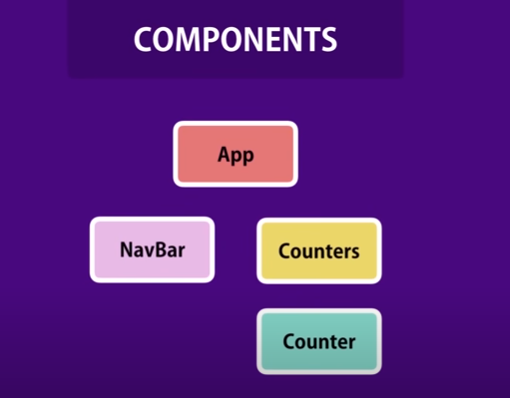




10. SinglesourceofTruth

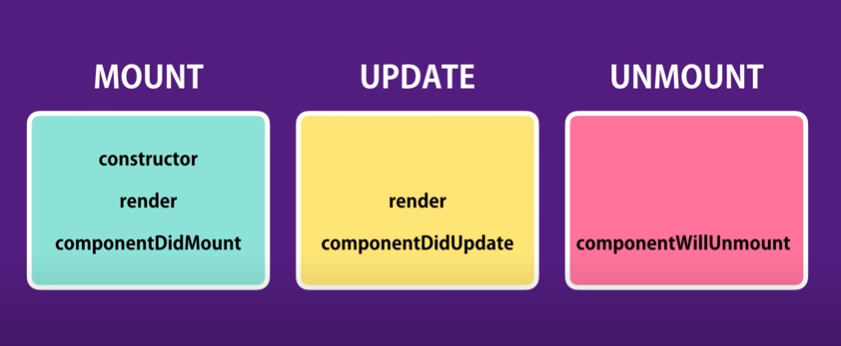
11. Removing local state (controlled component): The Child component is been controlled by parent component

12. Component Sync



13. LifeCycle hooks

Mounting phase Updating phase Unmount phase



Constructor – called only once, perfect place of initialize the state

componentDidMount - ajax call to fetch data from backend when component render

componentDidUpdate – ajax call to get new data from server basing on condition

componentWillUnmount – called before the component removed, perfect place for clean up code

14. ES6 Basics

Refer: https://www.w3schools.com/REACT/react\_lists.asp

1. Arrow Functions:

Hello = () => {return “hello world”}

1. Array methods.

const myArray = ['apple', 'banana', 'orange'];

const myList = myArray.map((item) => <p>{item}</p>)

1. Destructuring

const vehicles = ['mustang', 'f-150', 'expedition'];

const [car,, suv] = vehicles;

1. Spread operator

const numbersOne = [1, 2, 3];

const numbersTwo = [4, 5, 6];

const numbersCombined = [...numbersOne, ...numbersTwo];

1. ES6 Modules

Import Named exports

import { name, age } from "./person.js";

import default exports

import message from "./message.js";

# Functional Component

Syntax: function-name = () => { return <div> Hi </div>;}

# React Events

function Football() {

const shoot = () => {

alert("Great Shot!");

}

return (

<button onClick={shoot}>Take the shot!</button>

);

}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Football />);

function Football() {

const shoot = (a) => {

alert(a);

}

return (

<button onClick={() => shoot("Goal!")}>Take the shot!</button>

);

}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Football />);

# React Conditionals

function Goal(props) {

const isGoal = props.isGoal;

if (isGoal) {

return <MadeGoal/>;

}

return <MissedGoal/>;

}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Goal isGoal={false} />);

# React Router

Use React Router to route to pages based on URL

import ReactDOM from "react-dom/client";

import { BrowserRouter, Routes, Route } from "react-router-dom";

import Layout from "./pages/Layout";

import Home from "./pages/Home";

import Blogs from "./pages/Blogs";

import Contact from "./pages/Contact";

import NoPage from "./pages/NoPage";

export default function App() {

return (

<BrowserRouter>

<Routes>

<Route path="/" element={<Layout />}>

<Route index element={<Home />} />

<Route path="blogs" element={<Blogs />} />

<Route path="contact" element={<Contact />} />

<Route path="\*" element={<NoPage />} />

</Route>

</Routes>

</BrowserRouter>

);

}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

# React Hooks

* Hooks allow function components to have access to state and other React features. Because of this, class components are generally no longer needed.
* Hooks allow us to "hook" into React features such as state and lifecycle methods.

useState: useState hook to keep track of application state.

## **Hook Rules**

There are 3 rules for hooks:

* Hooks can only be called inside React function components.
* Hooks can only be called at the top level of a component.
* Hooks cannot be conditional

import React, { useState } from "react";

import ReactDOM from "react-dom/client";

function FavoriteColor() {

const [color, setColor] = useState("red");

return (

<>

<h1>My favorite color is {color}!</h1>

<button

type="button"

onClick={() => setColor("blue")}

>Blue</button>

<button

type="button"

onClick={() => setColor("red")}

>Red</button>

<button

type="button"

onClick={() => setColor("pink")}

>Pink</button>

<button

type="button"

onClick={() => setColor("green")}

>Green</button>

</>

);

}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<FavoriteColor />);

[Run Example »](https://www.w3schools.com/REACT/showreact.asp?filename=demo2_react_hooks)

useEffect:

Hooks allows you to perform side effects in your components.

E.g.

Fetching data

Directly updating the Dom

Timers

Syntax: useEffect(<function>, <dependency>)

useEffect accepts two arguments. The second argument is optional.

import { useState, useEffect } from "react";

import ReactDOM from "react-dom/client";

function Timer() {

const [count, setCount] = useState(0);

useEffect(() => {

setTimeout(() => {

setCount((count) => count + 1);

}, 1000);

});

return <h1>I've rendered {count} times!</h1>;

}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Timer />);