**React**

JavaScript library for user interface designing

Declarative

Component based-class based and Functional component

Consist of JSX

Pre-requisite

Intall node js

To install react

npx create-react-app my-app

To run application

npm start

simple code:

import ‘./App.css’;

const App=()=> {

return(

<div className=”App”>

<h1>Hello {name}!</h1>

</div>

);

}

}

export default App;

React fragment used to render multiple/different element of html

{

name?(

<> 🡨 react fragment

test

</>

): (

<>

<h1>test</h1>

<h2>test2</h2>

</>

)

}

Creating a functional component

Const Person = () => { //component name to be started with capital

Return (

<>

<h1>test</h1>

<h2>test2</h2>

<h1>test3</h1>

<h2>test4</h2>

</>

)

}

const App=()=> {

return(

<div className=”App”>

<Person />

<Person />

<Person />

</div>

);

}

}

Props

Allow u to pass navig data through react components. Passed via attributes.

Const Person = (props)=> {

Return (

<>

<h1>Name: {props.name}</h1>

<h2>Last Name: {props.lastName}</</h2>

<h2>Age: {props.age}</</h2>

</>

)

}

const App=()=> {

return(

<div className=”App”>

<Person name={‘John’} lastName={‘Doe’} age={25}/>

<Person />

<Person />

</div>

);

}

}

State

Plain JavaScript object represent a piece of information about the components current situation

Pre-requiste

import {useState} from ‘react’;

Hook

Whenever we start a function and we start with use then it is called hook

import {useState} from ‘react’;

import ‘./App.css’;

const App=()=> {

const [counter, setCounter] = useState(0);

useEffect(()=>{

setCounter(100);

},[]);

return(

<div className=”App”>

<button onclick={()=> setCounter((prevCount)=> prevCount -1)}>-</button>

<h1>{counter}</h1>

<button onclick={()=> setCounter((prevCount)=> prevCount +1)}>+</button>

</div>

);

}

}

export default App;

Building application

App.css-from git

search.svg-form git

Index.js

import React from ‘react’;

import ReactDOM form ‘react-dom’;

import App from ‘./App’;

ReactDOM.render(<App/>, document.getElemenyById(‘root’));

MovieCard.jsx-react component

Import React form ‘react’;

Const MovieCard=({movie})=>{//used movie1 instead of props, if props used should be mentionsed everywhere

return(

<div className=”movie”>

<div>

<p>{movie.Year}</p>

</div>

<div>

<img src={movie.Poster}!==’N/A’? movie.Poster: ‘https://via.placeholder.com/400’ alt={movie.Title}/>

</div>

<div>

<span>{movie.Type}</span>

<h3>{movie.Title}</h3>

</div>

</div>

)

}

Export deault MovieCard;

App.js

Import React from ‘react’;

Import ‘./App.css’;

Import SearchIcon from ‘./search.svg’

Import MovieCard from ‘./MovieCard’;

const API\_URL=”https:www.omdbi.com?apikey=num.”;//num to be put when created an account

const movie1={

json from git

}

const App=()=> {

const [movies, setMovies] = useState([]);

const [searchTerm, setSearchTerm] = useSate([]);

const searchMovies = async(title)=> {

const response = await fetch(‘$(API\_URL)’&s=$(title)’);

const data = await response.json();

console.log(data.Search);

}

useEffect(()=> {

searchMovies(‘Spiderman’);

},[]);

return (

<div className=”app”>

<h1>MovieLand</h1>

<div className=”search”>

<input

Placeholder=”Search for movies”

Value={searchTerm}

Onchange={(e)=>setSearchTerm(e.target.value)}

/>

<img

Src={SearchIcon}

Alt=”search”

Onclick={()=>searchMovies(searchTerm)}/>

</div>

{

movies?.length > 0

?(

<div className=”container”>

{movies.map((movie)=>(

<MovieCard movie1={movie}/>

))}

</div>

): (

<div className=”empty”>

<h2>No movies found</h2>

</div>

)

}

/\*

<div className=”container”>

/\*

<div className=”movie”>

<div>

<p>{movie1.Year}</p>

</div>

<div>

<img src={movie1.Poster}!==’N/A’? movie1.Poster: ‘https://via.placeholder.com/400’ alt={movie1.Title}/>

</div>

<div>

<span>{movie1.Type}</span>

<h3>{movie1.Title}</h3>

</div>

</div>

\*/

<MovieCard movie1={movie1}/>

</div>

\*/

</div>

)

}

**Web dev simplified**

Props is passed into a component and props handled outside

State is inside a component and handled inside

State is re-rendered when it changes

Hook-only for function component

Cannot be put inside the conditional statement

useState

it consists of two parameter inside a array as it returns array of two parameter

const [count, setCount]= useState(4)

count is the current state and setCount is the updated state(function) to be set.

Fucntion decre(){

setCount(prevCount=>prevCount-1)

}

onclick={decre} inside a button

instead of 4 we can give an function to only run once, otherwise everytime u render

it renders the value which is not needed

const [count, setCount]= useState(()=> {

console.log(“run function”);

return 4;

})

For merging property when a single property is updated …prevState is to be used

Or can use multiple state

Const [state, setState]= useState({count: 4, theme:’blue’})

Const count = state.count

Const theme = state.theme

Fucntion decrementCount(){

setState(orevState => {  
return {…prevState, count: prevState.count-1}

})

}

useEffect

**Freecodecamp**

Why react

Composable- component based

Declarative- like variables so no need to repeatedly declare

Hireable skill

Actively maintained by skilled people

Props used as function for creating dynamic components. Like youtube home page tiles.

It cannot be just imported as it becomes static without change in data.

Static websites done till now

Dynamic to be done

Event listeners

State

Conditional rendering

Forms

Side effects

function handleClick(){

console.log(“I was clicked!”);

<button onclick={handleClick}>Click me</button>

If handleClick()-> it loads on refreshing page before clicking

Camle case to be followed

<https://legacy.reactjs.org/docs/events.html#mouse-events>

State used to update the data value correctly, react will automatically update the UI part.

Interaction like bookmarking or saving reels in apps. Uses state to update the user response.

**Props vs State**

Props refer to the properties being passed into a component for it to work correctly, like how a function receives parameters: “from above.” A component receiving props is not allowed to modify those props(immutable).

State refers to the values that are managed by the components, like variables declared inside a function. Anytime you have changing values that should be saved/displayed, you’ll likely be using state.

**Code for Employee profile//toggle favourite**

App.js

Function App()

{

Const[contact,setContact]=React.useState({

firstName:”John”,

lastName:”Doe”,

phone:”+1 (719) 555-1212”

email: [itsmyrealname@example.com](mailto:itsmyrealname@example.com)

isFavourite=false

})

Let starIcon= constact.isFavourite? “star-filled.png”:”star-empty.png”

Function toggleFavourite(){

setContact(prev=>{

return{

…prev,

isFavourite=!prev.isFavourite

}

})

}

}

Challenge:

Import Star from ‘./components/Star.js’

<Star

Key={contact.isFavourite}

isFilled={ contact.isFavourite }

/>

Star.js

Import React from ‘react’

Function Star(props)

{

Let starticon= props.isFilled?”star-filled.png”:”star-empty.png”;

Return(

<img src={‘../images/${starticon}’}

className=”class--icon”

)

}

Export default star;

//custom components can’t have dom event listeners

<Star

Key={contact.isFavourite}

isFilled={ contact.isFavourite }

handleclick={toggleFavourite}//passing function as custom properties

/>

Star.js

Import React from ‘react’

Function Star(props)

{

Let starticon= props.isFilled?”star-filled.png”:”star-empty.png”;

Return(

<img src={‘../images/${starticon}’}

className=”class--icon”

onclick={props.handleclick}

)

}

Export default star;

Passing data to components

Keep the state as close to component(local)

To pass to sibling component u need to create state in the parent component and pass down to the children

Forms

Controlled component

To have single truthy value in a components

We are giving react state to control the html values

Side Effects

What are React’s primary tasks?

Work with the DOM/browser to render UI to the page

Manage state for us between render cycles (i.e state values are “remembered” from one render to the next)

Keep the UI updated whenever state changes occur

What can’t React handle on its own?

(Out)side effects

-local Storage

-API/database interactions

-subscription (e.g web sockets)

-syncing 2 different internal states

-basically anything that React isn’t in charge of

useEffect() – a blank canvas provided by react, to interact with the outside effect and to sync with the state

Code that runs because a component was displayed should be in Effects, the rest should be in events

You can fetch data with Effects, but you need to implement cleanup to avoid race conditions.

Putting the function as a callback function inside use Effect will only run after the UI is rendered

UseEffect takes a function as its parameter. If that function return something, it needs to be a cleanup function. Otherwise, it should return nothing. If we make it an async function, it automatically returns a promise instead of a function or nothing. Therefore, if you want to use async operations inside of useEffect, you need to define the function separately inside of useEffect, you need to define the function separately inside of the callback function, as seen below

Section 3-

Event listeners

State

Conditional rendering

Forms

Side effects

**Projects**

1. Dark mode project
2. Note-app project

-sync notes with localStorage

localStorage.getItem(“key”)

localStorage.getItem(“key”,value)

not value must be a string, so if you have a more complex value like an array or object to save, you’ll need to use:

JSON.stringify(value)

To retrieve back: JSON.parse(stringifiedvalue)

-add note summary titles

-move modified noted to the top of the list

-delete notes

**Tenzies project**

**Solo project-quizical**

**Advanced react course**

**React router**

**Css in JS**

**Next.js**

**Performance/optimization**

**More hooks**

**Websockets**

Websockets

1. Full-duplex bi-directional communication
2. WebSocket is a HTTP upgrade
3. Easy to implement and standardized
4. Only sends header once

Can I use .com

Polling & Long Polling

-Alternative to WebSockets

Much better backwards compatibility

-Polling

Send AJAX request every X amount of seconds for new data(not true real time)

-Long Polling

Send request to server and keep connection open until new data

Server Sent Events

Another ‘real-time’ alternative

Uses EventSource API to send messages from server

Not truly requires an event loop

Generally, requires an event loop

No binary message capability

Intended Use Case

WebSockets not == replacement for HTTP

WS is an upgrade for HTTP

HTTP provides automatic caching!

WS often needs special configuration of load balancing

Can’t communicate with REST

Use when you need full-duplex connection

Useful for web-based games, chatting applications, anything which needs low-latency real-time connection!

WebSocket Clients

Used to interface with WebSocket Server

Built in many languages! (including Python!)

Clients exist for MicroPy and Arduino!(IoT)

Most common clients is web based and uses Javascript

Require the Server to be able to interface WS

WebSocket Clientside Code

Const socket = new WebSocket(‘ws://localhost:8000’)

Socket.onopen=(event)=>{

Socket.send(‘PyCon AU!!’)

});

Socket.onmessage=(event)=>{

Console.log(event.data);

});

SocketIO

Javascript library for manipulating Websockets

Include fallback mechanism and auto reconnection

Handles disconnection and connection events

Namespacing and Room broadcasting

SocketIO Clientside Code

Var socket =io(‘http://localhost:8000/<MY\_NAMESPACE>’)

Socket.on(‘connect’),()=>{

Socket.emit(‘event\_on\_my\_server’,data=”PyConAU!”);

});

Socket.on(‘my\_custom\_event’,(data)=>{

})

Flask

Socket=Sockets(app)

@socket.route(‘/my\_sockets’)

Def my\_socket\_event(ws):

While not ws.closed:

Message=ws.recieve();

Ws.send(message)

Performce comparision

http and WebSocket have the same sized header

2byte/msg overhead

Socketio increases latency and intial connection under the hood starts:

Uses AJAX Long Olling initially and then upgrades

Deployment

Evenlet and Gevenet ti monkey patch async

Or just use standard threading

Async web framework are ideal

Use message queue to run multiple instances behind, load balancers with sessions workers