React

1.Creating React Projects
-Have Node
-Commands Used
-npx create-react-app name-of-project
-npm start
-npm uninstall -g name-of-project
-npm install

2.React JSX
-What is it?
-JSX vs Javascript

3.React Component
-What is a Component?
-Creating a component.
-Component File.
-Import and export file
-Root folder(app.js)

4.Component Styling
-External Style Sheet.
-Inline Styling
-Main Component import file.
-ClassName vs Class
-

5.Passing Dynamic Data
-What is it?
-Curly Braces {}.

6.React Event Listener
-What is it?
-onClick.

-Naming convention,Camecase.
-Event handler function.
-Event handler naming convention.
-Passing handlers as arguments
7.React Forms

9.React List and Dynamic values
-The map method
-Looping through List in React.

10.React useContext
-What is it
-useContext and useState

Creating React Project

-To create a react project make sure node.js is installed
-To install a new project run npx create-react-app name-of-project.
-To start the project run npm start.
-To uninstall project run npm uninstall -g name-of-project
-Npx create-react-app creates all the files and folders needed for a react project.
-Incases of errors we update our dependencies with npm install

Syntax

```
npx create-react-app name-of-project
npm start
npm uninstall -g name-of-project
npm install
```

Example

```
npx create-react-app emmanuel-portfolio
```

```
npm uninstall -g  emmanuel-portfolio
```
Output:-localhost3000

React JSX

-JSX makes it easier to write HTML elements in react and place them in the DOM without any createElement() or appendChild() methods.
-Without JSX it is impossible to create elements, We will need to create elements with methods.
-With JSX you can write expressions inside curly braces

Example

With JSX

```
const myElement = <h1>I Love JSX!</h1>;
const root =
ReactDOM.createRoot(document.getElementById('root'));root.render(myElement)
```

Without JSX

```
const myElement = React.createElement('h1', {}, 'I do not use JSX!');
const root =
ReactDOM.createRoot(document.getElementById('root'));root.render(myElement);
```

React Component

-Components are building blocks of a react application that accepts inputs like props and returns react elements.
-The return in the component only takes one root element, This is the main div only and inside the children are nested.
-The short cut to create the react component is rafce if you have the installed the ES7 in Vscode.

Syntax

-A Component file starts with UpperCase.
-Create a file with the .js extension in the src folder.
-Import React from react
-Create a function with the name of the component.
-Inside the function return output and export the component,Note the root in return.
-Return to Root folder(App.js) and import react component at the top.
-Render the component in JSX of root folder (app.js)

Example

Component Created

```
 import React from ‘react’
 const  Component()=>{
          return(
                  <div>
                      <h1> Hello </h1>
                   <div/>
                   )
          }
export default Component
```

Root Folder App.js

```
Import Component from ‘./Component’’
<Component><Component/>
```

Output:-Hello

React Component Styling

-Components in react are styled to make our html in JSX have a good UI when displayed.
-An external file is created with the .css extension with the same name as the component to be styled.
-className is used instead of class to reference classes defined in an external CSS stylesheet.
-className is a reserved keyword in Javascript.

Syntax

-Import style file in the component.
-Use className to access classes from the style sheet.

Example

component.js

```
import React from ‘react’;
import ‘./component.css’;
 const  Component()=>{
          return(
                   <div className=`expense-item`>
                        <h1> Hello </h1>
                    <div/>
                    )
          }
export default Component
```

Component.css

```
.expense-item{
          display:flex;
          justify-content: space-between;
}
```

React Dynamic Data in JSX

-Data in JSX can be passed dynamically instead of hard coding.

-This data can be from variables or APIs.

Syntax

-Variables with data created.

-Curly braces is used with the variable name to output data

Example

```
 const  Component()=>{
          const time = new Date(2021, 2, 28)
          const expenseTitle = `Car Insurance`

           return(
                    <div className=`expense-item`>
                         <h1>{time}</h1>
                         <h1>{expenseTitle}</h1>
                     <div/>
                     )
           }
export default Component
```

Output:-2021, 2, 28

Output:-Car Insurance

<u>**React Props/Properties.**</u>

-Props is used in components to display and share data from one component to another.

-Parent components can pass props to children components.

-Props can be many data types like numbers, strings,functions, objects.

-Data that is not string is stored in curly braces { }.

Syntax

-In the JSX function of our component we pass prop as the parameter or any name.

-Display prop to the component, By giving the component an attribute name.
-In the JSX function we call the prop with the name in the component
-We could also call the props directly using the curly brace then how we named it.

Example
App JSX File
```
 const  Component()=>{
      =
          return(
                   <div className=`App`>
                      <component text={`I am Manu`}> </component>
                    )
          }
export default Component
```

Component JSX File
```
 const  Component(props)=>{
          return(
                   <div className=`expense-item`>
                        <h1>{props.text}</h1>
                    <div/>
                    )
          }
export default Component
```

Output:- A Component with data is displayed.

Events
-An event is an action that happens when we manipulate a page.
-An event listener allows us to call functions when specified events happen.

-To use eventlistener you must have a document selected from the DOM.Thats the target
-It takes the event,function and useCapture.
-The event can be such as click,mouseover.
-Usecapture parameter is optional .

Syntax

```
target.addEventListener(event,functionCallback,useCapture)
target.addEventListener(event,Function,useCapture)
```

Example

```
button.addEventListener(`onclick`,eventlistenFunc)
function eventListen(){
        console.log(`Hello`)
}
```

Output:-Hello will be output when button is clicked.

Or

```
button.addEventListener(`onclick`, ()=>console.log(`Hello`)
```

Output:-Hello will be output when button is clicked.

__Event Parameter__

-The event functions take in a parameter which is named in any specific way.
-The normal convention of naming is `e` or `event` or anyway.
-The event contains information about the action that happened.
-We can see the events objects and properties.

Syntax

```
function eventListener(event){
     console.log(event)
}
target.addEventListeners(`click`,eventListener)
```

Example

```
function buttonListener(event){
        console.log(event)
}
button.addEventListeners(`click`,buttonListener)
```
Output:-The button object and its properties.

Event.Target

-An event.target return the DOM element that triggered a specific event so we can retrieve any property/ attribute with a value.

-We can see the element,className,Value,Type of event.,Position of Mouse.

Syntax

```
function buttonClick(e){
 e.target                    -Returns entire element
 e.target.className          -Returns the className of element
 e.type                      -Returns the type of event.
 e.offsetY                   -Returns the position of  mouse.
 e.ctrlkey

}
```

Example

HTML

```
<form>
  <input type="text"  class="inputClass"> <input/>
<form />
```

Javascript

```
const inputClass = document.querySelector('.inputClass');
function myEvent(e){
   console.log(e.target.value)
}
inputClass.addEventListeners(`keydown`,myEvent)
```

Output:-What the user inputs in the form.

React Event Listeners

-Event listeners are used to handle users interaction with the webpage such as clicking a button or hovering over a particular element.

-React Event listeners are written in Camel case syntax.

-React Event handlers are written inside curly braces inside our component, since in react eventListeners are not used.

-The Specific event receives a call back name of the function.

-Its good to use Handler naming convention when naming the function handler

-Here are common event listeners in React

onClick

-Triggers when an element is clicked

-It is used for handling clicks and other interactive elements.

onChange

-Triggers when a value of an input element is changed.

-It is used for handling form inputs such as text fields,select boxes,check boxes.

onSubmit

-Triggers when a form a submitted.

-It is used to handle form submissions and perform actions such as sending data to server.

onMouseEnter

-Triggers when the mouse pointer enters an element.

-It is used for hiding tooltips or other interactive elements.

onMouseLeave

-Triggers when the mouse pointer leaves an element.

-It is used for leaving tooltips or other interactive elements.

Syntax

```
const nameOfFunctionHandler=()=>console.log(hello);
<button> onClick = {name Of function  Handler} <button/>
```

Example

```
const ourComponent = ()={
  const  Component()=>{
        const buttonHandler = ()=>{  console.log(`Hello`)  }
         return(
                <button onClick= {buttonHandler}>Button<button/>
                )
        }
 export default NameOfComponent
```

Passing handlers as function.

-Instead of using our function handlers outside jsx we can pass them into our event listeners.

-An arrow is used inside our event listener.

Syntax

```
<button> onClick = {()=>console.log(`Hello`)} <button/>
```

Example

```
const ourComponent = ()={
         return(
                <button onClick= { ()=>{  console.log(`Hello`) }>Button<button/>
                )
        }
 export default NameOfComponent
```

React States

-States is an object that holds data or information for a component.

-Without States our user interface will never change.

-To use is first import it into the component.

-Destructuring is used in useState,It accepts two value

-Current state and function that updates state.

-useState hook is called inside the component but not inside a nested function.

Syntax

```
Import {useState } from “react”;
const[currentState, updateState]=useState(`Update me`)
```

Example

```
const  Component()=>{
  const[title,setTitle]=useState(props.title)

const buttonHandler = ()=>{
        setTitle(`Am Manu`)
}
return(
        <div className=`App`>
                <h1>{title}<h1/>
                <button onClick= {buttonHandler}>Button<button/>
         <div/>
)
}
```

Forms

-Forms are used to collect users inputs.

-The HTML <Form> is used to create an html form user input.

-The HTML <label> is used to create a text description for forms control like text field..

-The HTML <input> element is the most used form element.

-The HTML <input> is displayed in many ways depending on the type of attribute.

-Some of HTML <input> attributes are:-

-Text:- Displays a single-line text input field

-Radio:- Display a radio button for selecting one of many choices.

-Checkbox:- Displays a checkbox for selecting zero or more of man choices.

-Submit:-Displays a submit button for submitting the form.

-Button:-Displays a clickable button.

Syntax

```
<form>
  <label><label>
  <input type="text"> <input/>
<form />
```

Example

```
<form>
  <label>Username:-<label>
  <input type="text"> <input/>
  <input type="submit"> <input/>
<form />
```

<u>**React Forms**</u>

-It allows users to interact with the web page.

-Forms in React are added just like other elements.

Syntax

```
const myFormComponent  =()=>{r
    return(
           <form>
               <div>
                      <label>Title<label>
                       <input type="text">
               <div>
           <form>
)
}}
 default myFormComponent
```

React Form Handling

-Handling form is how data is handled when it changes or get submitted.
-Changes are controlled by adding event handlers.
-onSubmit is used to handle form submission its added on the form.
-onChange handling form inputs such as text fields,select boxes,check boxes.
-To prevent the browser from refreshing we use the preventDefault() method on form event.
-useState Hooks are used to keep track of each input value.
-The current value input is set using the current state.
- An event handler function is created to handle the set State new value.

Example

```
import { useState } from `react`;
const (currentTitle, setTitle) = useState(``);

const titleHandler = (event)=>{
        setTitle(event.target.value)
}

function myForm(
return(<form>
            <div>
                  <label>Title<label>
                  <input value={currentTitle} onchange ={titleHandler }type="text">
            <div>
         <form>
```

))

<u>React Form Submission</u>

-Form Submission is done by adding an event handler in the Form attribute.
-The onSubmit event handler is used.
-A function that is an event handler is created to handle.
-In the form the handler the setNew value is called to remove the current values on submission

<u>Example</u>

```
import { useState } from `react`;
const (currentTitle, setTitle) = useState(``);

const titleHandler = (event)=>{
        setTitle(event.target.value)
}

const handleSubmit = event =>{
         event.preventDefault();
         console.log(`Form Submitted with value:currentTitle`)
         setTitle(``)
}

function myForm(
  return(<form onSubmit={handleSubmit}>
            <div>
                  <label>Title<label>
                  <input value={currentTitle} onchange ={titleHandler }type=”text”>
            <div>
        <form>
))
```

<u>React List and Dynamic Values</u>

-List is render using a loop in react.
-Map Method is generally the most preferred method.
-In the li attribute the map method is used to iterate over the array.

Syntax

const array = [{id:1,value:`Manu`},{id:234}];

JSX

```
<ul>
  array.map((arr,index)=>(<li>arr<li/>)
<ul/>
```

Example

```
const people = [{id:1,name:`Manu`},
                 {id:234,name:`Koech`}
                ];

<ul>
 people.map((person,index)=>(<li>person.name<li/>))
<ul/>
```

React useContext
-It allows component to share data with other components without passing props down to multiple levels component tree.
-We create context.
-We provide a context value
-consume the context value

Syntax

-Import createContext and useState

-Initialize createrContext by storing it in a variable.

-Wrap the child components in the context provider and supply state value.