**React JS**

-open source js library for ui

-component based architecture

-reusable code

-react is declarative

-react will handle updating and rendering of components efficiently.

-easily intgratable to any application.

npx:- npm package runner

npx create-react-app appname

files in react project

-package.json:- contains dependencies and scripts required in the project.

-packege-lock.json:- ensure consistent installation of dependencies.

-gitignore:-is to ensure that certain files not tracked by Git remain untracked.

-node\_module folder:- folder where all dependencies are installed.

-Public folder:

?.....-manifest.json:- concerned with progressive web apps.

-index.html:-contains the root DOM.

-src folder:-

-index.js:-renders the app component into the root DOM node

-App.js:- contains the html displayed in browser.

Components

-Components represents a part of the user interface.

-root (App.js) component: - contains all the components in the app.

-components are reusable & components can contain other components. The code is in a .js file

-Stateless functional component & stateful class component.

- Stateless functional component: - is js functions which return html

-stateful class components: - regular es6 classes that extend component class from react library.

-must include a render method to return html.

Functional Components

-can receive properties & return html that describes the ui

-simple functions

-we use it as much as possible

-Absence of “this” keyword

-Hooks provide states and other features. {

-hooks breaking changes

-completely opt-in & 100% backwards-compatible

}

-mainly responsible for UI

-Are called Stateless/dumb/presentational

-no longer stateless because we can use HOOKs

There are 2 types export default export & named export, if it is named export we use {} while importing.

While creating the component we must make the name of the component capitalized.

**Class components**

-can receive properties & return html that describes the ui

-first, we create a class that extends Class “Component”, then in we use the render function and inside the function we return the html content.

-more feature rich

-maintain their own private data we call state.

-complex UI logic

-provide lifecycle hooks

-stateful/smart/container

**JSX(javascript xml]**

-JSX tags have a tag name, attributes and children.

-JSX ultimately transpiles to pure js which is understood by the browser.

-JSX translates into the react.createElement this is why we need to import react.

-In JSX -Class->className

for->htmlFor

onclick->onClick

tabindex-> tabIndex

Example with JSX:-

const Hello = () => {

//     return (

//         <div>

//             <h1>Hello ppl</h1>

//             </div>

//     )

// }

Example without JSX:-

const Hello = () => {

return React.createElement('div' ,{id=hi}, React.createElement('h1',null, 'Hi'));

}

React.createElement() accepts a minimum of 3 elements

First parameter-string which specifies the html tag to be rendered in our case ‘div’.

2nd parameter-optional properties we used {id=’hi’, className:’hello’}.

3rd parameter- are children for the html element in the first parameter.<h1> hi</h1>

We use props in React **to pass data from one component to another**

Props are immutable.

 The state is **a built-in React object that is used to contain data or information about the component**.

Props vs state

Graphical user interface, application

Description automatically generated

**Using Component State**

-create state object inside class constructor

Text

Description automatically generated

this.setState({

accepts objects like state

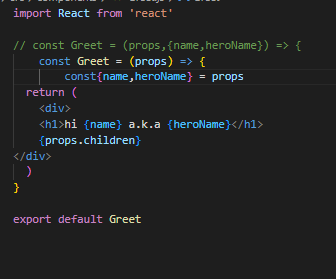
})

Graphical user interface, text, application

Description automatically generated

Destructuring: -to unpack values from arrays or properties from objects into distinct variables.

Destructuring props: -



Destructuring states:-

A screenshot of a computer

Description automatically generated with medium confidence

Event handling in functional components: -

Text

Description automatically generated

Event handling in class components:-

Text

Description automatically generated

Binding event handlers: - ‘this’ keyword is undefined in Event handlers, so we use binding.

1st option to bind: - use the bind key word and bind handler in render method.

Don’t use this because of performance.

Text

Description automatically generated

2nd option is using arrow function in the render method & calling the event handler in the function body.

Text

Description automatically generated

3rd option binding the event handler in the constructor. The best one

Text

Description automatically generated

4th option to use an arrow function in the class property

Text

Description automatically generated

Using methods from Parent component in the child component by passing the method as a prop to the child component: -  
Parent:- Text

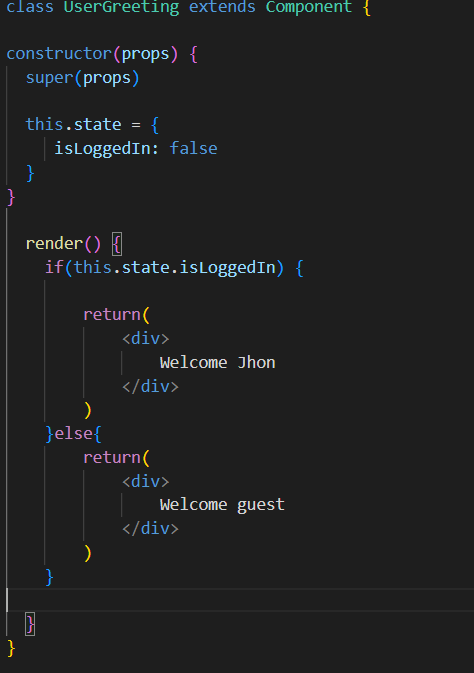
Description automatically generated

Child:- Text

Description automatically generated

**Conditional Rendering**:

1-if/else



2-element variables:-



3-Ternary conditional operators:-

Text

Description automatically generated

4-Short circuit operator:-

Text

Description automatically generated

List rendering: -

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

So the previous code becomes

Text

Description automatically generated

Using index as keys: -

Text

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

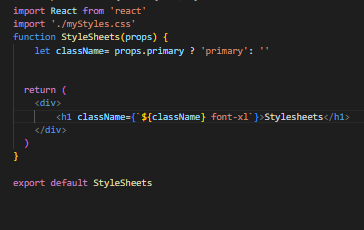
If not, we can use the npm package to generate unique id or hash out one of the properties.

Styling components: -

Graphical user interface, text, application

Description automatically generated

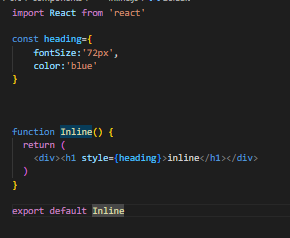
When styling we can use template literals to use stylesheets: -



Graphical user interface

Description automatically generated

When using inline CSS, we use camel case while setting the style properties like fontSize:’72px’:



When using CSS modules, we use filename.module.css and in modules the classes are locally scoped meaning we can use CSS stylesheets in child components but not modules,

In app.js: -

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

**Form Handling**

Controlled Components: - form elements whose value is controlled by react.

1st step:- create the html in render

<div>

            <label>Comments</label>

          </div>

2nd step:- assign the state created to the value of the html input

  this.state = {

     username:'',

     comments:''

  }

 <div>

            <label>Comments</label>

            <textarea value={this.state.comments}></textarea>

          </div>

3rd assign the changehandler to assign the change:

handleCommentsChange=(event)=>{

    this.setState({

        comments:event.target.value

    })

}

     <div>

            <label>Comments</label>

            <textarea value={this.state.comments} onChange={this.handleCommentsChange}></textarea>

          </div>

To submit the data:-

          <button type='submit'> Submit</button>