Cal State Fullerton

CPSC 349
React JS

By

Mahitha Pasupuleti

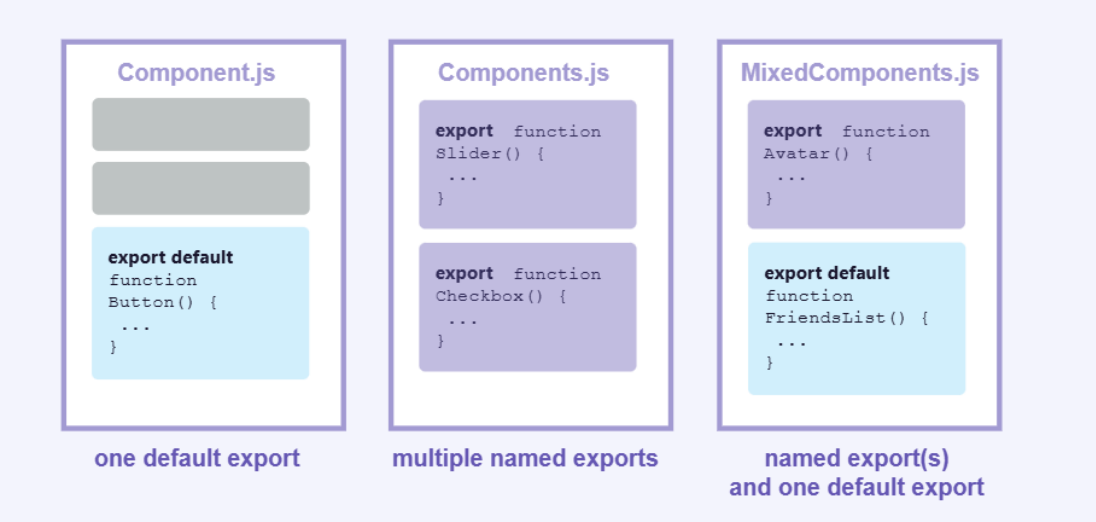
Component

- React lets you combine your markup, CSS, and JavaScript into custom “components”, **reusable UI elements for your app.**
- **React component is a JavaScript function that you can *sprinkle with markup*.**
- React component names must start with a capital letter.
- How to build a component –

 - step-1 : **Export the component**
 - step-2 : **Define the function**
 - step-3 : **Add markup**

Importing and Exporting component

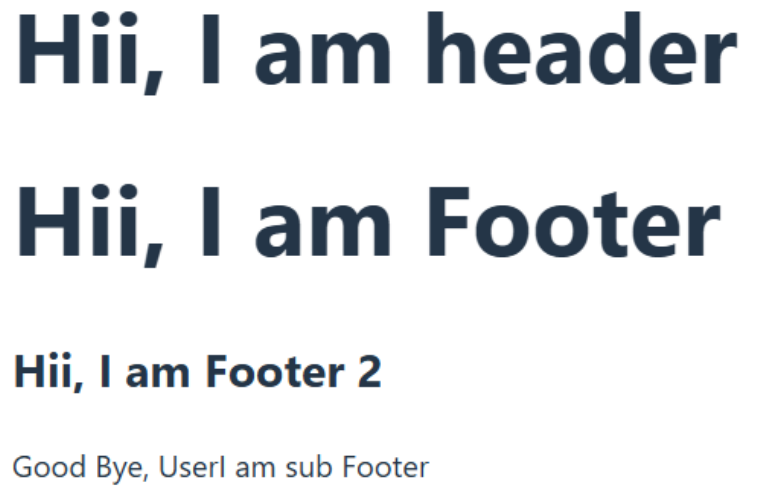
- There are two primary ways to export values with JavaScript: default exports and named exports. So far, our examples have only used default exports. But you can use one or both of them in the same file. **A file can have no more than one *default* export, but it can have as many *named* exports as you like.**



JS in JSX

- JSX lets you write HTML-like markup inside a JavaScript file, keeping rendering logic and content in the same place. Sometimes you will want to add a little JavaScript logic or reference a dynamic property inside that markup. In this situation, you can use curly braces in your JSX to open a wi aScript.

```
function Footer(){

    const name="User";
    return(
        <>
          <h1>Hii, I am Foote
          <h2>Hii, I am Foote

          Good Bye, {name}


        </>
    )
    }

    export default Footer
```



Fragment in JS

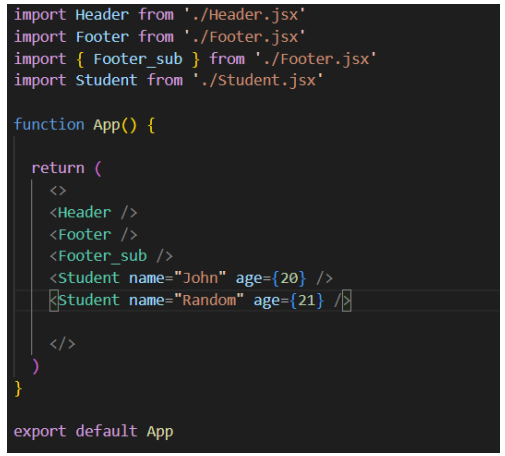
Wrap elements in <Fragment> to group them together in situations where you need a single element.
Grouping elements in Fragment has no effect on the resulting DOM; it is the same as if the elements were not grouped.
The empty JSX tag <></> is shorthand for <Fragment></Fragment> in most cases.

Fragments are useful because grouping elements with a Fragment has no effect on layout or styles,
unlike if you wrapped the elements in another container like a DOM element.
If you inspect this example with the browser tools, you'll see that DOM nodes appear as siblings without wrappers around them

```jsx
import Header from './Header.jsx'
import Footer from './Footer.jsx'

function App() {

  return (
    <>
    <Header />
    <Footer />
    </>
  )
}

export default App
```

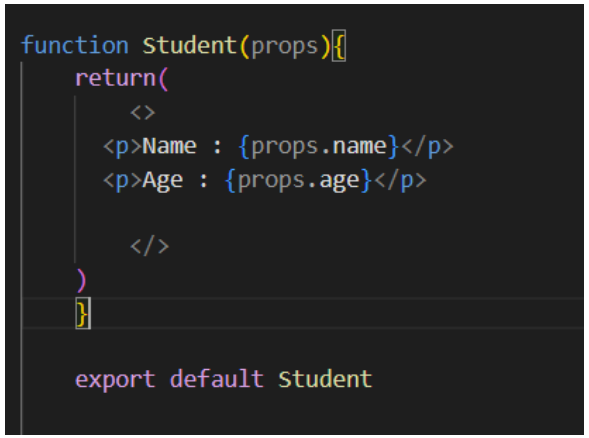
Props

- React components use *props* to communicate with each other. Every parent component can pass some information to its child components by giving them props. Props might remind you of HTML attributes, but you can pass any JavaScript value through them, including objects, arrays, and functions.
- You can think of props like “knobs” that you can adjust. They serve the same role as arguments serve for functions—in fact, props *are* the only argument to your component! React component functions accept a single argument, a `props` object:



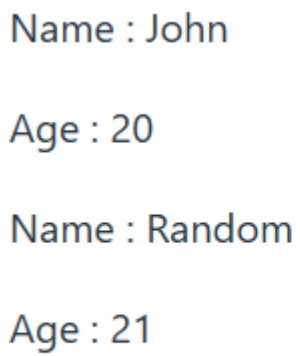
```jsx
import Header from './Header.jsx'
import Footer from './Footer.jsx'
import { Footer_sub } from './Footer.jsx'
import Student from './Student.jsx'

function App() {

  return (
    <>
    <Header />
    <Footer />
    <Footer_sub />
    <Student name="John" age={20} />
    <Student name="Random" age={21} />

    </>
  )
}

export default App
```

App.jsx



```jsx
function Student(props){
    return(
        <>
      <p>Name : {props.name}</p>
      <p>Age : {props.age}</p>

        </>
    )
    }

    export default Student
```

Student.jsx



Output

Conditional Rendering

Your components will often need to display different things depending on different conditions.
In React, you can conditionally render JSX using JavaScript syntax like `if` statements, `&&`, and `? :` operators.

```
function Student(props){
    return(
        <div>
            Name : {props.name}
            Age : {props.age}

            {props.isStudent ?<>Hi Student</>:<> {null}</>}
        </div>
    )
    }

    export default Student
```

Hii, I am header

Hii, I am Footer

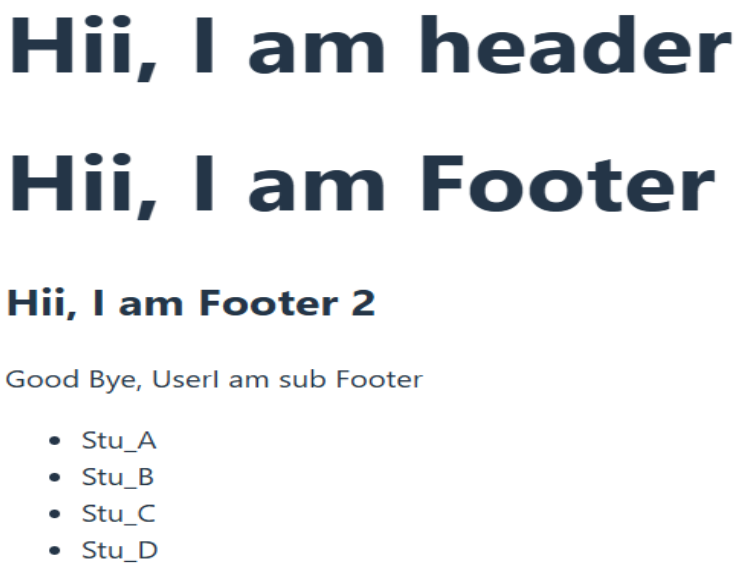
Hii, I am Footer 2

Good Bye, UserI am sub Footer
Name : JohnAge : 20Hi Student
Name : RandomAge : 21

Rendering List

You can store that data in JavaScript objects and arrays and use methods like map() and filter(),
to render lists of components from them.
JSX elements directly inside a map() call always need keys!
Keys tell React which array item each component corresponds to, so that it can match them up later.

```
let stu_list=[
    {
        id:0,
        name:"Stu_A"
    },
    {
        id:1,
        name:"Stu_B"
    },
    {
        id:2,
        name:"Stu_C"
    },
    {
        id:3,
        name:"Stu_D"
    }];

function StudentList(){
    let render_stu= stu_list.map((student)=>{
        return(<li key={student.id}>{student.name}</li>);

    })

    return(
        <>

        <ul>{render_stu}</ul>
        </>
    )
    }

    export default StudentList
```



References -

- https://react.dev/learn
- https://www.w3schools.com/REACT/DEFAULT.ASP