React Basics-1

Module 1: Anatomy of React

By the end of this module I will be able to :

- Explain the concepts behind React and component architecture.
- Describe how to use assets within an app to apply styling and functional components.
- Create a component to service a specific purpose.
- Create a folder and demonstrate how to create and import files within that folder.
- Use and manipulate props and components to effect visual results.

Module 2: Data and State

By the end of this module I will be able to:

- Use common methods to manage state in React.
- Detail the concept and nature of state and state change.
- Describe the hierarchical flow of data in React.
- Describe how data flows in both stateful and stateless components.
- Use an event to dynamically change content on a web page.
- Describe some common errors associated with events and the syntax required to handle them.

Module 3: Navigation Updating and Assets in React

By the end of this module I will be able to:

- Use media assets, such as audio and video, with React.
- Demonstrate how to manipulate image assets using reference paths.
- Explain the folder structure of a React project in terms of embedded or referenced assets.
- Demonstrate the conditional implementation and rendering of multiple components.
- Create and implement a route in the form of a navbar.

- Describe navigation design in React, with a focus on single and multi-page navigation.

Week-1

Learning Objectives

- Explain the concepts behind React and component architecture
- Create a component to serve a specific purpose
- Create a component folder and demonstrate how to create and import files within that folder
- Use and manipulate props and components to effect visual results
- Describe how to use assets within an app to apply styling and functional components

Introduction

- standalone bits are components
- components are merged and as a whole ends up as a website.
- State: values of all variables that are used in an application.
- Real time apps- airbnb, IG, FB etc.

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “components”.

Need for fast, responsive UI lead to meta using react.

React is open-sourced.

Refresher

HTML

HTML is used to describe the structure of Web pages.

Users need to be able to interact with elements on a web page. This means that an HTML document must be represented in a way that JavaScript code can query and update it. And that's the function of the DOM. It's a model of the objects in your HTML file.

Essential HTML tags and concepts

Layout & Style

- <html>
- <head>
- <body>
- <div>

Text formatting & lists

- <h1>…<h6>
- <p>
- <ul><li>
- <b><i>

Images and links

- <img src="">
- <a href="">

Linking and Meta

- <link>
- <title>
- <meta>

Semantic

- <header>

CSS

CSS styling options:

- Font styling (font size, font color, etc.)
- Flex Box Layout (Layout of items using CSS Flex Box Layout)
- CSS Selectors
- Position, Padding, Margins and Display
- Colors, Background and Icons

JS

- Data types
- Using var, let and const
- Conditionals and Loops
- Using objects, arrays and functions
- ES6 Arrow functions
- In-built functions such as map(), forEach() and promises.
- Destructuring Arrays and Objects
- Error Handling

Package Manager (Node + npm)

- Installation command to install npm modules in your project
- Installing a package as a dev dependency
- Start command
- Updating npm version

- Navigating around the package.json file

Module Exports

there are two ways to export modules in JavaScript:

1. Using default exports
2. Using named exports

Named exports are a way to export only certain parts of a given JavaScript file.

A module can only have **one default export**, but **as many named exports as you'd like** (zero, one, two, or many

---

React Components & where they live

why React?

- reusable components.
- client side library.
- flexible custom UIs

React.js Overview

- Can be used to create a SPA. SPA is A technology that loads a single web page and performs updates to the DOM on this single web page based on user interaction with this web page in known as a SPA. Indeed, React allows developers to build SPAs.

component based arch

  - building software based on reusable code. Can be inserted anywhere.

- devs can work on different components without interfering with out dev’s components. standalone component on UI.
- website is a collection of components.
- Each component has its own HTML, CSS, JS code to create dynamic content.

React.js Streamline building and composing components. and reducing load on browser via component rendering.

React provide Virtual DOM to ensure update to actual DOM is as minimal as possible.

Functional Components

React provides 2 types of components : 1. functional and 2. class components.

every react app’s index.js must have atleast 1 component. it is called the “root” component.

A React component is indeed much like a regular JavaScript function.

Render Syntax: <ComponentName/>

the. root component contains other component. this is ultimately convert to a DOM fragment/element and added the DOM with some “id”.

JSX (javascript xml)

looks very similar to HTML. allows us to write js code inside HTML like element. It helps makes website/content dynamic.

A react component won’t run unless it has a JSX element inside it. Analogous to function declaration and function call.

Component is a JS file.

React treats components differently based on its letter is capitalized or not. if it is not, treats as HTML element instead of react element.

Transpiling ; converting JSX to HTML.

Reading Material

Basic concepts of flexbox - CSS: Cascading Style Sheets | MDN

The flexible box layout module, usually referred to as flexbox, was designed as a one-dimensional layout model, and as a method that could offer space distribution between items in an interface and powerful

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox

it has 2 axis - main , and cross.

4 directions- row, row-reverse, column and column-reverse.

it also has different wraps for line/content.

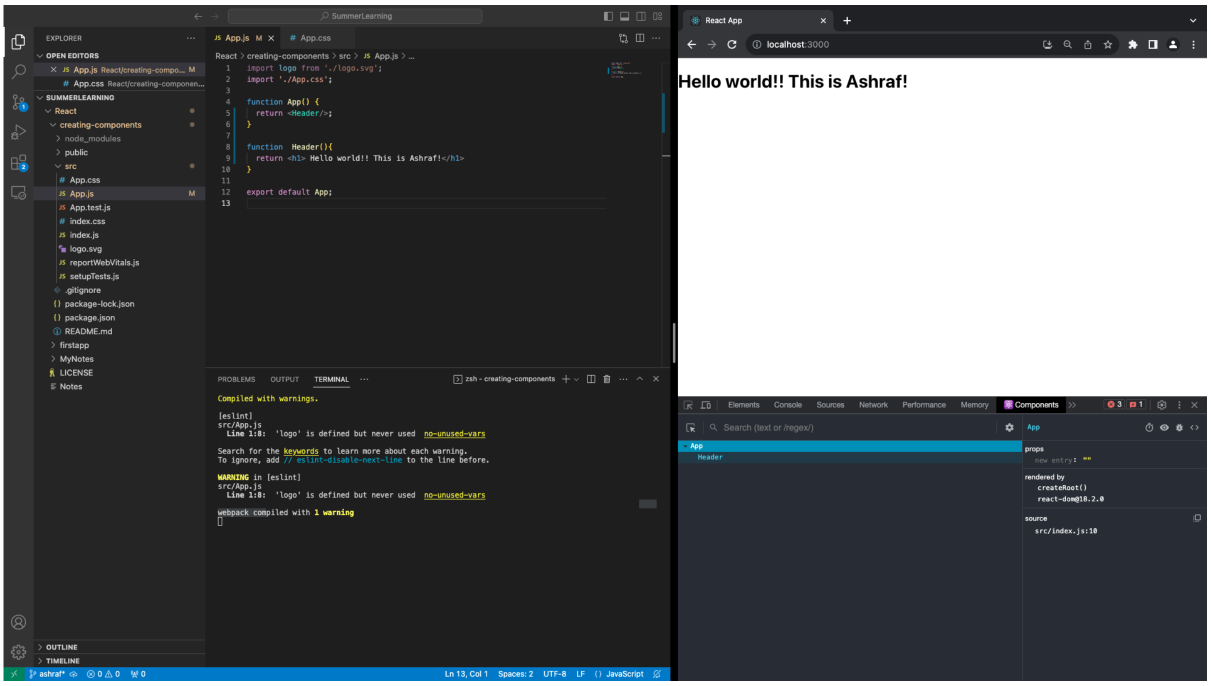
Provide control on expanding/shrinking of content, alignment, etc.

Creating component

creating a react-app using ‘npm init react-app’

For a component to render something on the page, it needs to return it as one or more JSX elements.

Creating a functional component, and rendered it in the root component ‘app.js’.



```
import logo from './logo.svg';
import './App.css';

function App() {
  return <Header/>;
}

function  Header(){
  return <h1> Hello world!! This is Ashraf!</h1>
}

export default App;
```

```
Compiled with warnings.

[eslint]
src/App.js
  Line 1:8:  'logo' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.

WARNING in [eslint]
src/App.js
  Line 1:8:  'logo' is defined but never used  no-unused-vars

webpack compiled with 1 warning
```

Hello world!! This is Ashraf!

Transpiling JSX

For React code to be understood by a browser, you need to have a **transpiling step** in which the JSX code gets converted to plain JavaScript code that a modern browser can work with.

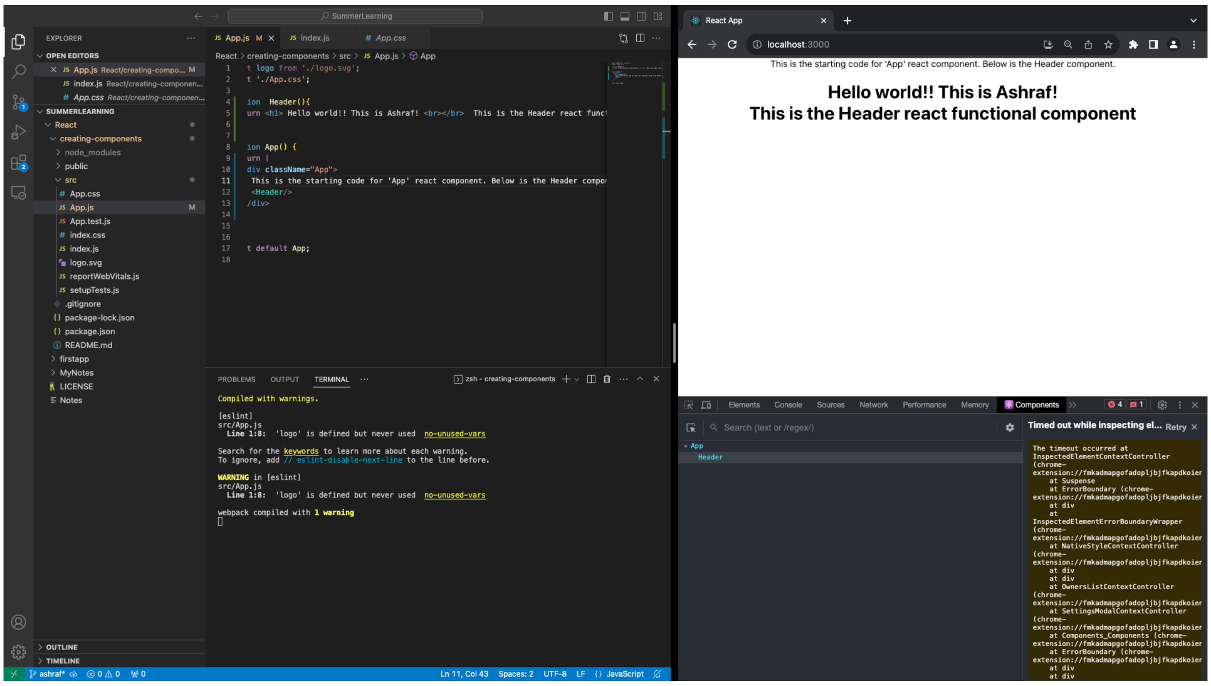
What is the minimum code that a component must have to be able to show something on the screen when rendered?

It is a functional component i.e. javascript function, with JSX element. and an export if it is in different file.

Task Completed: Created a component, and render it.

Create the App component, and header component. which is rendered inside the app component.

Added some content within the tags to delineate which react component it exists in.



```
t logo from './logo.svg';
t './App.css';

ion  Header(){
urn <h1> Hello world!! This is Ashraf! <br></br>  This is the Header react funct


ion App() {
urn (
div className="App">
 This is the starting code for 'App' react component. Below is the Header compo
 <Header/>
/div>



t default App;
```

```
Compiled with warnings.

[eslint]
src/App.js
  Line 1:8:  'logo' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.

WARNING in [eslint]
src/App.js
  Line 1:8:  'logo' is defined but never used  no-unused-vars

webpack compiled with 1 warning
```

This is the starting code for 'App' react component. Below is the Header component.

Hello world!! This is Ashraf!
This is the Header react functional component

Timed out while inspecting el...

```
The timeout occurred at
InspectedElementContextController
(chrome-
extension://fmkadmapgofadopljbjfkapdkoier
    at Suspense
    at ErrorBoundary (chrome-
extension://fmkadmapgofadopljbjfkapdkoier
    at div
    at
InspectedElementErrorBoundaryWrapper
(chrome-
extension://fmkadmapgofadopljbjfkapdkoier
    at NativeStyleContextController
(chrome-
extension://fmkadmapgofadopljbjfkapdkoier
    at div
    at div
    at OwnersListContextController
(chrome-
extension://fmkadmapgofadopljbjfkapdkoier
    at SettingsModalContextController
(chrome-
extension://fmkadmapgofadopljbjfkapdkoier
    at Components_Components (chrome-
extension://fmkadmapgofadopljbjfkapdkoier
    at ErrorBoundary (chrome-
extension://fmkadmapgofadopljbjfkapdkoier
    at div
    at div
```

React Project Structure and explanation.

Build using npm init react-app.

The project has specific directories and iels.

- node_modules: it has all the packages that are downloaded using npm.
- public: has the assets that is displayed to the user in the app.
    - Manifest.json: provide metadata
    - index.html: react app is injected into this file.
    - robots.txt : is for search engine optimization.
- src: has all the component files to ensure the app functions.
    - index.js is the most important file
- gitignore: standard git file.

- package.json: allows npm to run scripts to perform various tasks. helps to rebuild the app.
  - contains the metadata about the project and also the functional dependencies that is required by the application.
- package-lock.json: Without package.lock.json, there might be some differences in installed versions in different environments. To overcome this problem, package.lock.json is created to have the same results in every environment.
  - json file is like a one-stop solution of your entire problem. **package-lock. json is a file that is automatically generated by npm when a package is installed**. It records the exact version of every installed dependency, including its sub-dependencies and their versions.

Organizing Your Code

This is where React docs can help. They suggest two approaches:

1. Grouping by features
2. Grouping by file type

They also advise not to nest folders too deep, and to keep things simple and not overthink it.

They even say that if you're just starting out, you shouldn't spend more than five minutes setting up a project

---

Practice :

Building a Layout

Imagine that you've been given the task of building a somewhat more complex website layout using React.

At this point, you still don't know too much about how React works, but even with your limited knowledge, you can still build some relatively interesting designs.

Currently, you need to build a simple typography-focused layout for a coding blog.

This means that you will not have to use images, which simplifies your task significantly.

The layout you're supposed to build will consist of the following sections:

- Main navigation
- Promo (main advertisement)
- A list of newest posts' previews (intros)
- The footer

Created component directory in src.

Added components, nav, promo, intro1,2,3, footer js files.

Well, with JSX, it looks like HTML so much that it's easy to forget that it's actually JavaScript code - not HTML.

Some points to think about:-

- But why use **Intro1.js, Intro2.js,** and **Intro3.js**? Isn't one of the tenets of coding the DRY approach - that is, the "Don't repeat yourself" approach?
    - Indeed, it is. However, there are still a few concepts to discuss before you learn how to re-use a single component with variations in its content. This has to do with data in components, but don’t worry, we’ll be getting to that later.
- about not using the <a> element for empty links in my app.
    - The answer here depends on whether those links are "internal" - inside an app, or "external", meaning, leading to some external link, such as *https://www.coursera.org*. If the links are internal to the app - as they are envisioned here - using the <a> tag is simply not the React way of doing things. You'll learn why that is the case when discussing the use of React Router.

Importing & Creating Components

export: make component available for other components to use.

- default export is used when function name is same as the file name.
    - which mean we can have only one such component.
- named export is used when we have function with different name than the file name.
    - which mean we can have many components.

import: to get an exportable component and use it.

- import <component name> from '<location of the component file>'

note: file name extension is not required.

Difference between modules in js and component:

- Component can be thought of as small piece of functionality. And Module as series of components.

Splitting code into independent parts is the paradigm of modular programming.

It allows for independent and parallel development of different part of application.

In react, we have modularity by having component defined in their own file.

Practice: Creating & Importing components

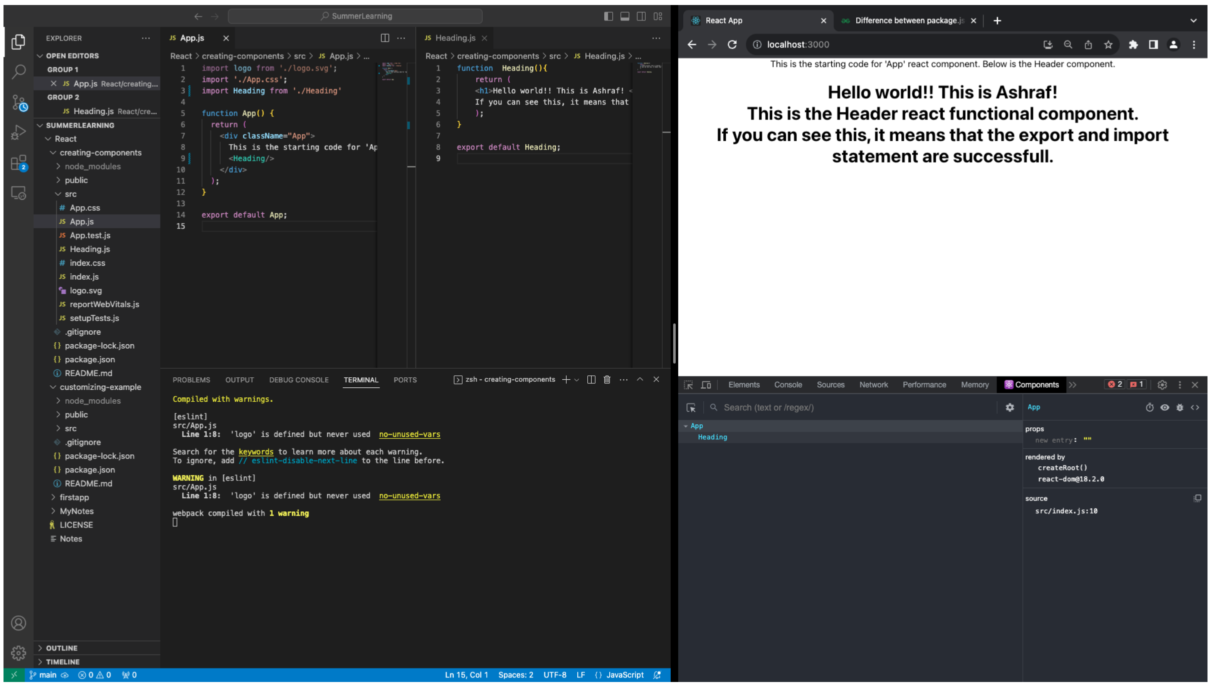
Switch directory to the other project 'creating-components'

renamed header to heading.

creating a new component file for Heading component.

changed the structure of return in heading component a little.

added the export and import statement in Heading, and App component respectively.



```
import logo from './logo.svg';
import './App.css';
import Heading from './Heading'

function App() {
  return (
    <div className="App">
      This is the starting code for 'Ap
      <Heading/>
    </div>
  );
}

export default App;
```

```
function  Heading(){
    return (
    <h1>Hello world!! This is Ashraf! <
    If you can see this, it means that
    );
}

export default Heading;
```

```
Compiled with warnings.

[eslint]
src/App.js
  Line 1:8:  'logo' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.

WARNING in [eslint]
src/App.js
  Line 1:8:  'logo' is defined but never used  no-unused-vars

webpack compiled with 1 warning
```

This is the starting code for 'App' react component. Below is the Header component.

Hello world!! This is Ashraf!
This is the Header react functional component.
If you can see this, it means that the export and import statement are successfull.

```
App
  Heading

props
  new entry: ""
rendered by
  createRoot()
  react-dom@18.2.0
source
  src/index.js:10
```

You should use the import keyword to import one component into another in React.

You can, and should, save all the components as separate files.

Component Use and Styling

Principles of Components

Just as we can make function flexible in javascript by using arguments to function. For react components, we can use properties “props” to do something similar.

JavaScript Object: store groups of related data. in the form of properties. these properties are name value pairs.

In react, we can use props to pass data between components.

props can be of multiple data types same as a js object.

Data Flow:

the component sending the props is the Parent component, and the receiver is the child component.

A single parent can share its props to multiple child components.

Important to know that it is a one-way communication.

Properties can be accessed using dot notation.

Limitations:

- one way communication. i.e. a child cannot send data to its parent component.
- Pure Function : a component/function using props can never modify the props.

Deep Dive into Props

Consider this component,

```
function App() {
  return <h1>Hello there</h1>
}
```

will be transpiled to

```
"use strict";
function App() {
return /#PURE/React.createElement("h1", null, "Hello there");
}
```

Focus on the `React.createElement("h1", null, "Hello there");` part. You can ignore the rest.

This means that the createElement function receives three arguments:

1. The wrapping element to render.
2. A null value (which is there to show an absence of an expected JavaScript object value).
3. The inner content that will go inside the wrapping element.

How does it look when the JSX is more complex, a comp within a comp.

Consider,

```
function App() {
  return (
    <div>
    <h1>Hello there</h1>
    </div>
  )
}
```

transpiled return statement in plain JavaScript again returns two createElement functions.

```
function App() {
  return React.createElement(
    "div",
    null,
React.createElement("h1", null, "Hello there")
  );
}
```

we can replace null with {}. it becomes a props object.

In essence, the createElement looks like this,

```
React.createElement(
  type,
  [props],
  [...children]
)
```

Props make it for data flow to be dynamic instead of static.

Using props in components

how to use props in component and use them to transfer data.

Practice task:

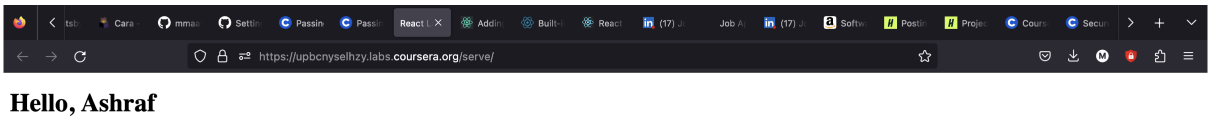
created a Heading component in src directory.

Updated Heading component to use the prop argument and use it to display the FirstUserName property within the content of the h1 tag.

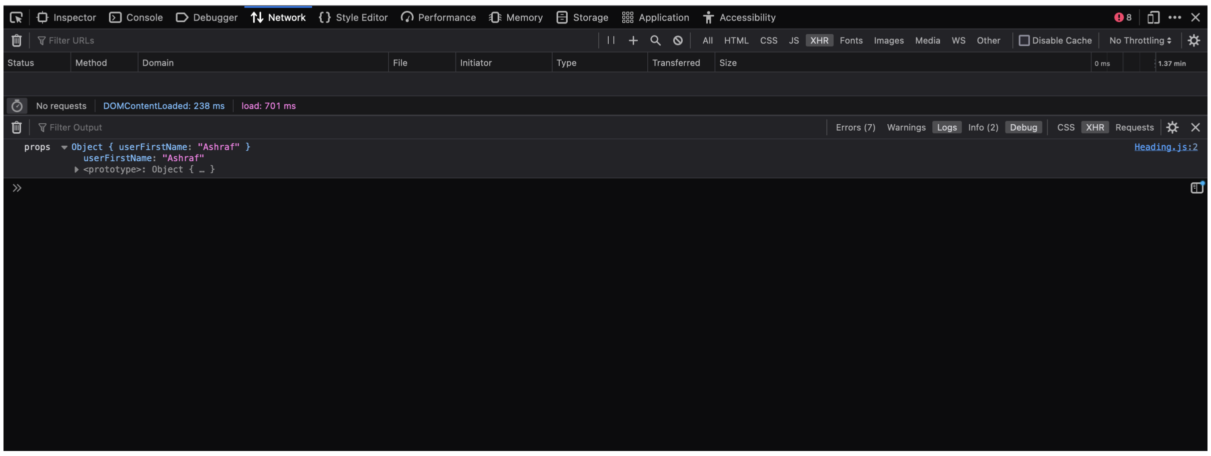
```
function Heading(props) {
    return (
        console.log("props ",props),
        <h1>Hello, {props.userFirstName}</h1>
    )
}

export default Heading;
```

Pass the prop’s param value from app.js component.

```
import Heading from "./Heading";

function App() {
    return (
        <div className="App">
            <Heading userFirstName="Ashraf"/>
        </div>
    );
};

export default App;
```



Hello, Ashraf



```
No requests   DOMContentLoaded: 238 ms   load: 701 ms

props  ▾ Object { userFirstName: "Ashraf" }                Heading.js:2
           userFirstName: "Ashraf"
         ▸ <prototype>: Object { … }
```

JSX in detail from styling perspective

allows developers to express what they want in an easier format.

It allows use to write html code directly inside javascript code.

React lets us use props object to make content dynamic.

A javascript function tells React how to render a component.

the curly braces in return statement allows us to insert js code not just placeholder for data.

to return HTML code on multiple lines, the return should have () braces.

CANNOT use class keyword in JSX. as it is reserved keyword by css. so we use className.

Props and Children

there is also a special prop known as props.children**,** which is automatically passed to every component.

Note:

The html code in JSX is placed inside a div. However,

when you do NOT want to add another div to the DOM. You CAN enclose use a fragment <> </> .

when you see a JSX element wrapping some other JSX element, you can easily understand that it's all just props.children in the background.

Reference -

the question is this: Let's say you want to have a Bag component, which can be used to "carry" Apples or Pears. How would you do that?

```
function Apples(props) {
  return (
    <div className="promo-section">
        <div>
            <h2>These apples are: {props.color}</h2>
            </div>
            <div>
            <h3>There are {props.number} apples.</h3>
        </div>
    </div>
  )
}
export default Apples
```

```
function Pears(props) {
  return (
    <h2>I don't like pears, but my friend, {props.friend}, does</h2>
  )
}
```

Solution:

```
function Bag(props) {
    const bag = {
        padding: "20px",
        border: "1px solid gray",
        background: "#fff",
        margin: "20px 0"
    }
    return (
        <div style={bag}>
            {props.children}
        </div>
    )
}
export default Bag
```

```
<Bag children={<Pears friend="Peter" />} />
<Bag children={<Apples color="yellow" number="5" />} />
```

or

```
<Bag>
    <Apples color="yellow" number="5" />
</Bag>

<Bag>
    <Pears friend="Peter" />
</Bag>
```

Another important concept to be aware of is finding the right amount of modularization. What does this mean?

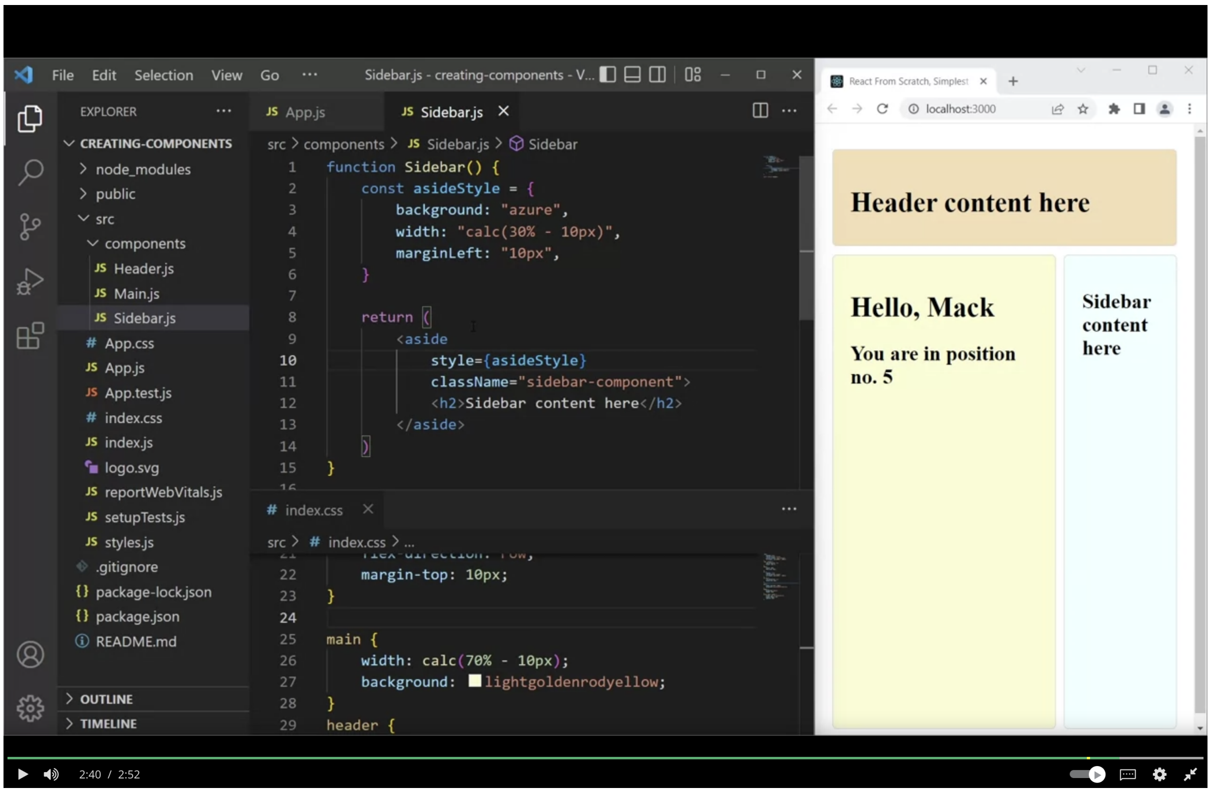
- Imagine having several small bags, each of which can only carry a single apple or pear. You would need to wrap each piece of fruit in a bag, which doesn't make much sense. You can apply the same logic to components that make up your layouts.
- You don't want to have an entire layout contained in a single component, as that would be difficult to work with. On the other hand, if you make each HTML element in your layout a separate component, it would be too hard to work with, even though the layout would be modular.
- Instead, you should organize your layouts by dividing them into meaningful areas of the page and code each of these areas as separate components. This would achieve the right amount of modularity. To further illustrate this point, think about how a person would describe a website: there's a menu, a footer, a shopping cart, and so on.

Styling JSX elements

3 ways

1. inline using style attribule of the html element
2. style lement in the head section of html
3. link the style sheet

Using React, you can easily convert a CSS rule to a JavaScript object, where each key-value pair describes a CSS declaration.s



```
function Sidebar() {
    const asideStyle = {
        background: "azure",
        width: "calc(30% - 10px)",
        marginLeft: "10px",
    }

    return (
        <aside
            style={asideStyle}
            className="sidebar-component">
            <h2>Sidebar content here</h2>
        </aside>
    )
}
```

```
    margin-top: 10px;
}

main {
    width: calc(70% - 10px);
    background: lightgoldenrodyellow;
}
header {
```

Header content here

Hello, Mack

You are in position no. 5

Sidebar content here

JSX syntax and the arrow function

Both of the 2 things achieve the same thing:-

1. Using a function declaration as a component.
2. assign anonymous function declaration to a variable, and use it to invoke.

```
function Nav(props) {
    return (
        <ul>
            <li>{props.first}</li>
        </ul>
    )
}

const Nav = function(props) {
    return (
        <ul>
            <li>{props.first}</li>
        </ul>
    )
}
```

The component is, for the most part, the same. The only thing that's changed is that you’re now using an anonymous (nameless) function, and assigning this anonymous function declaration to a variable declared.

One of the main benefits of using arrow functions is its shorter syntax.

```
const Nav = (props) => {
    return (
        <ul>
            <li>{props.first}</li>
        </ul>
    )
}
```

- The arrow itself can be thought of as the replacement for the function keyword.
- The parameters that this arrow function accepts are listed before the arrow itself.
- when you write arrow functions, **for any number of parameters other than a single parameter, using parentheses around parameters is compulsory**.
- the implicit return works if your entire component is a single line of code

JSX expression

Recall that JSX is a syntax extension to JavaScripts that is used with React.

It allows developers to write HTML as part of their component code, and is frequently used in React as it offers greater flexibility.

when this JSX code executes, the result variable will contain a react element that can then be inserted into the webpage. This is one of the key features of JSX.

Allows to inserted js values into HTML component of react element.

Expression can be used for HTML Attributes as well. Example the url for img tag.

Ternary operators and functions in JSX

Normal way:

```
let name = 'Bob';
if (name == 'Bob') {
    console.log('Hello, Bob');
} else {
    console.log('Hello, Friend');
};
```

Ternary way to do it is:

```
let name = 'Bob';
name == 'Bob' ? console.log('Hello, Bob') : console.log('Hello, Friend');
```

format to use it is as follows:

<Condition> ? “execute if TRUE” : “execute if FALSE”

comparison ? true : false

Function can also be written as a function declaration, or as a function expression. It does not have to be an arrow function. And we can use them in the JSX component in the braces.

Expressions as props

In React, we can pass any kind of expression as props.

Example:

To the Example component, three props are being passed to it: toggleBoolean, math, and str. The toggleBoolean is unchanged, and the math prop and the str prop have been added.

```
const bool = false;
const str1 = "just";

function Example(props) {
    return (
        <div>
            <h2>
                The value of the toggleBoolean prop is:{props.toggleBoolean.toString()}
            </h2>
            <p>The value of the math prop is: <em>{props.math}</em></p>
            <p>The value of the str prop is: <em>{props.str}</em></p>
        </div>
    );
};

export default function App() {
    return (
        <div className="App">
            <Example
                toggleBoolean={!bool}
                math={(10 + 20) / 3}
                str={str1 + ' another ' + 'string'}
            />
        </div>
    );
};
```

Learnt how to Embed a JS expression in attribute.

Practice: Multiple Components

Task: You've learned that components in React allow for modularity and reuse, with the ability to make data more dynamic by passing it from the parent to the child using props.

In this exercise, you'll render one component multiple times, to practice using different props and observing what exactly "reusing" components means.

Created a React component Card.

Added a function with props and default export.

Added a return statement.

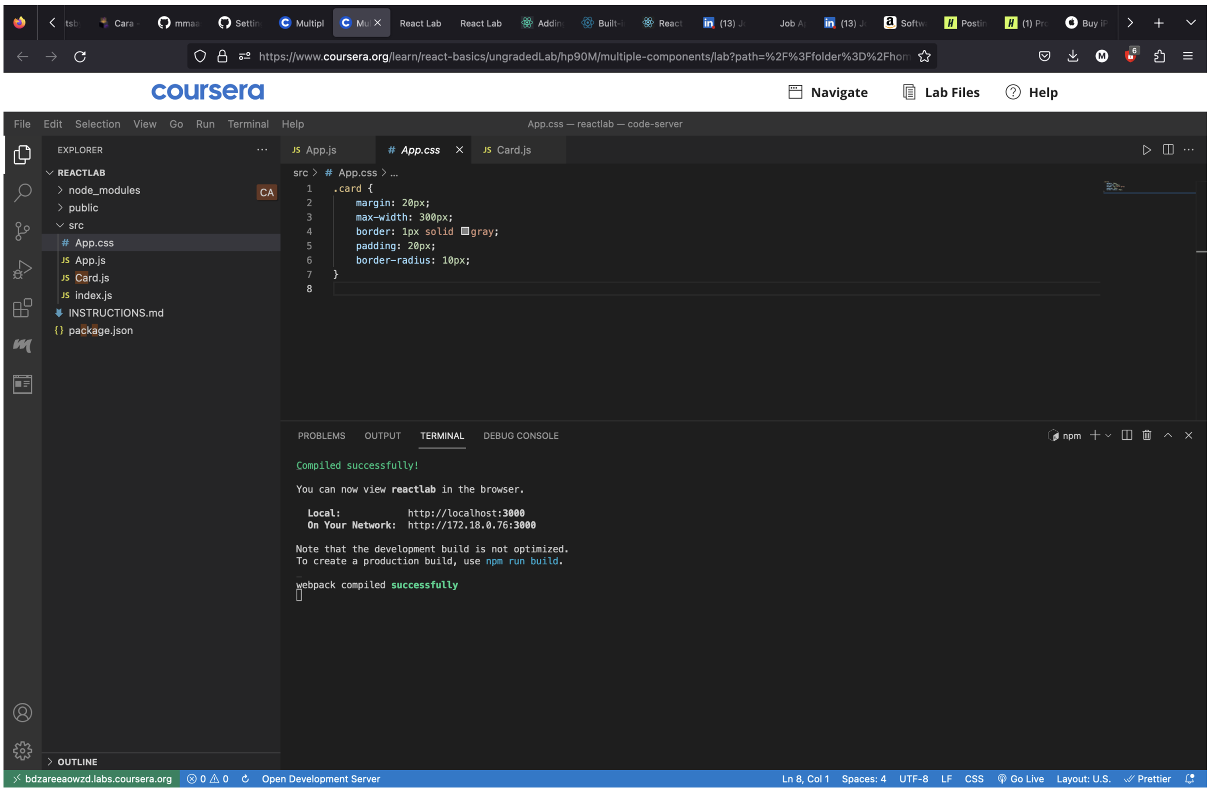
Created div

Added h2, h3 elements, with JSX expression that uses props object.

Now, back to App.js,

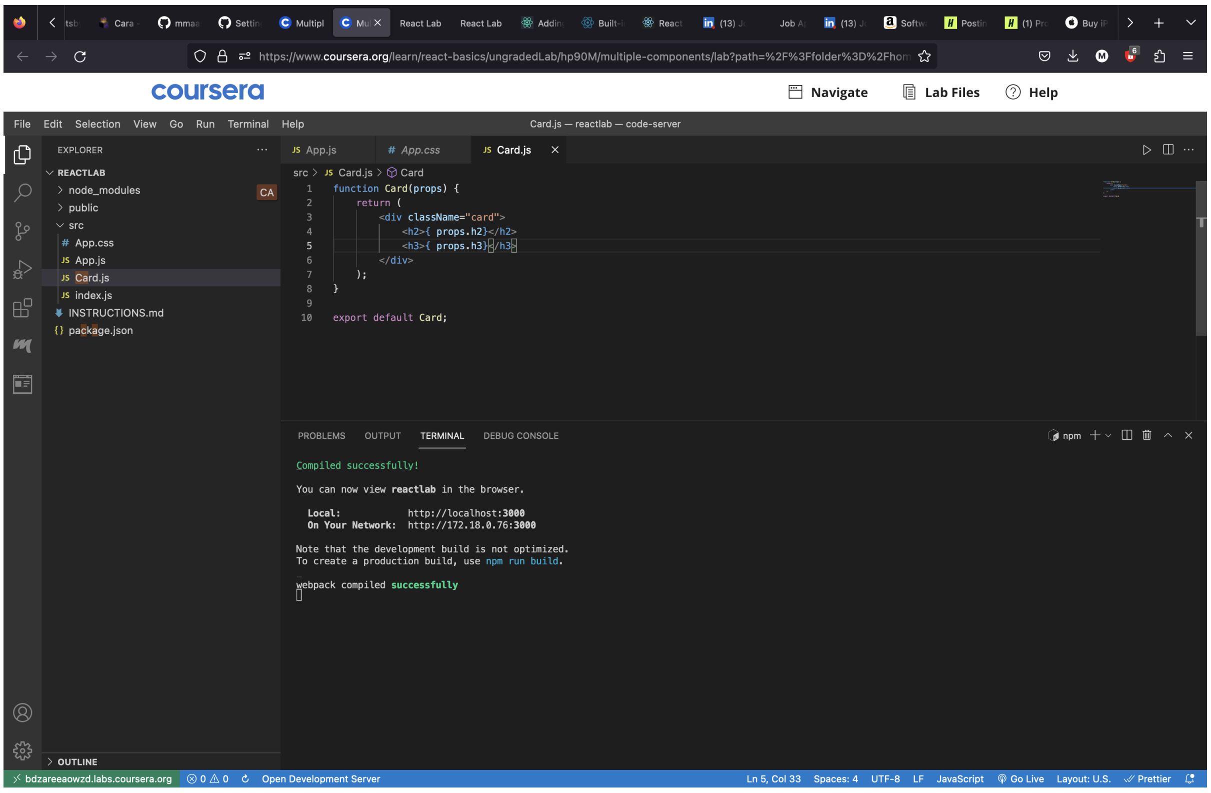
Use the above created component to render content.

add import statement, and render the Card element with paramters h2, h3.


https://www.coursera.org/learn/react-basics/ungradedLab/hp90M/multiple-components/lab?path=%2F%3Ffolder%3D%2Fhom
coursera
Navigate
Lab Files
Help
App.css — reactlab — code-server
EXPLORER
REACTLAB
node_modules
public
src
App.css
App.js
Card.js
index.js
INSTRUCTIONS.md
package.json
OUTLINE
src > # App.css > ...
bdzareeaowzd.labs.coursera.org
Open Development Server
Ln 8, Col 1 Spaces: 4 UTF-8 LF CSS Go Live Layout: U.S. Prettier

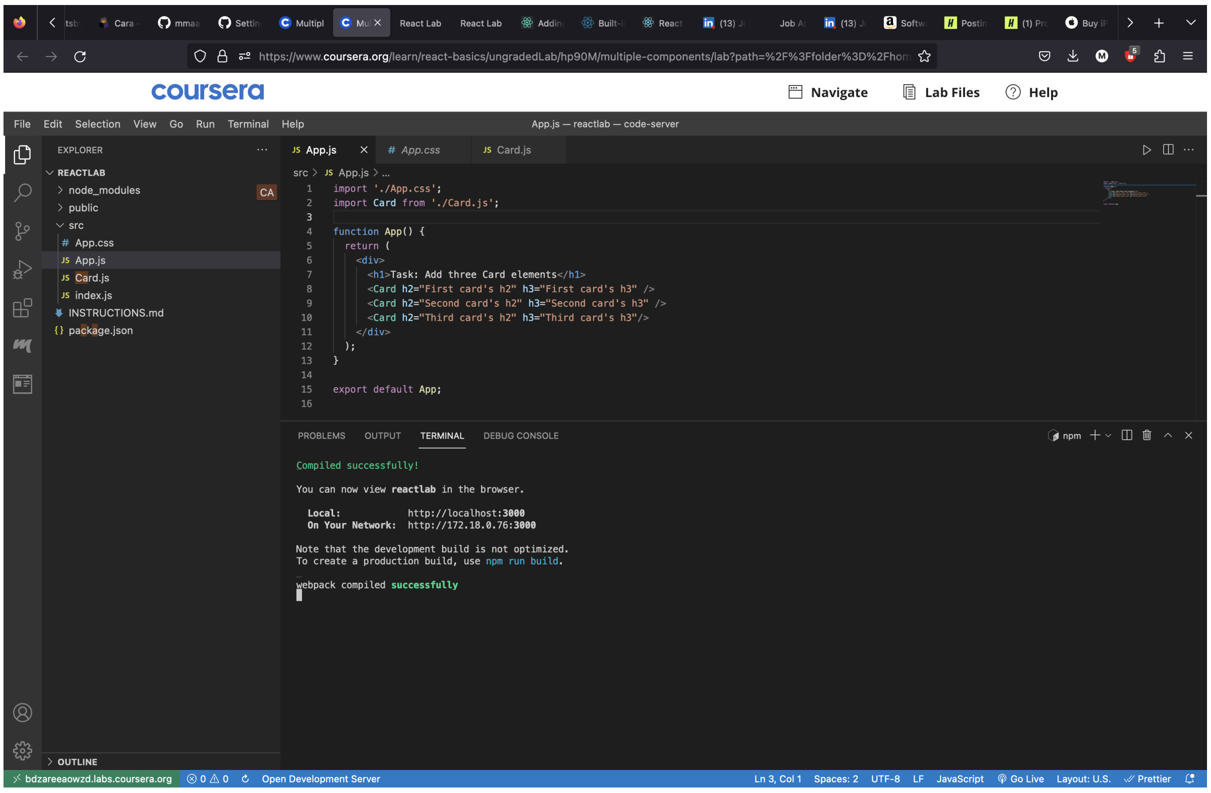
```css
.card {
    margin: 20px;
    max-width: 300px;
    border: 1px solid gray;
    padding: 20px;
    border-radius: 10px;
}
```

```
Compiled successfully!

You can now view reactlab in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://172.18.0.76:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

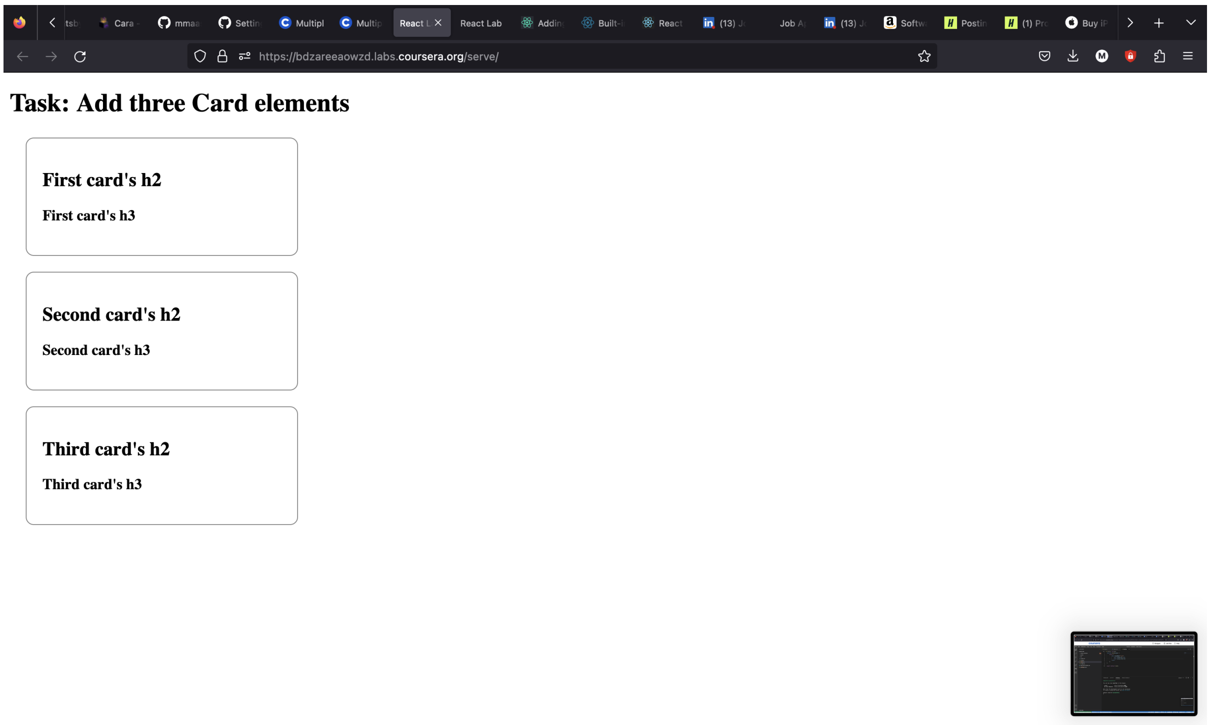

https://www.coursera.org/learn/react-basics/ungradedLab/hp90M/multiple-components/lab?path=%2F%3Ffolder%3D%2Fhom
coursera
Navigate
Lab Files
Help
Card.js — reactlab — code-server
EXPLORER
REACTLAB
node_modules
public
src
App.css
App.js
Card.js
index.js
INSTRUCTIONS.md
package.json
OUTLINE
src > Card.js > Card
bdzareeaowzd.labs.coursera.org
Open Development Server
Ln 5, Col 33 Spaces: 4 UTF-8 LF JavaScript Go Live Layout: U.S. Prettier

```javascript
function Card(props) {
    return (
        <div className="card">
            <h2>{ props.h2}</h2>
            <h3>{ props.h3}</h3>
        </div>
    );
}

export default Card;
```

```
Compiled successfully!

You can now view reactlab in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://172.18.0.76:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```


```
import './App.css';
import Card from './Card.js';

function App() {
  return (
    <div>
      <h1>Task: Add three Card elements</h1>
      <Card h2="First card's h2" h3="First card's h3" />
      <Card h2="Second card's h2" h3="Second card's h3" />
      <Card h2="Third card's h2" h3="Third card's h3"/>
    </div>
  );
}

export default App;
```

```
Compiled successfully!

You can now view reactlab in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://172.18.0.76:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



Task: Add three Card elements

First card's h2

First card's h3

Second card's h2

Second card's h3

Third card's h2

Third card's h3

You must wrap all the returned BlogCard components in a wrapping element, such as a div.

Learned to create a single page application using react.

refresh html css, js.

setup and layout of directory and files on a react project.

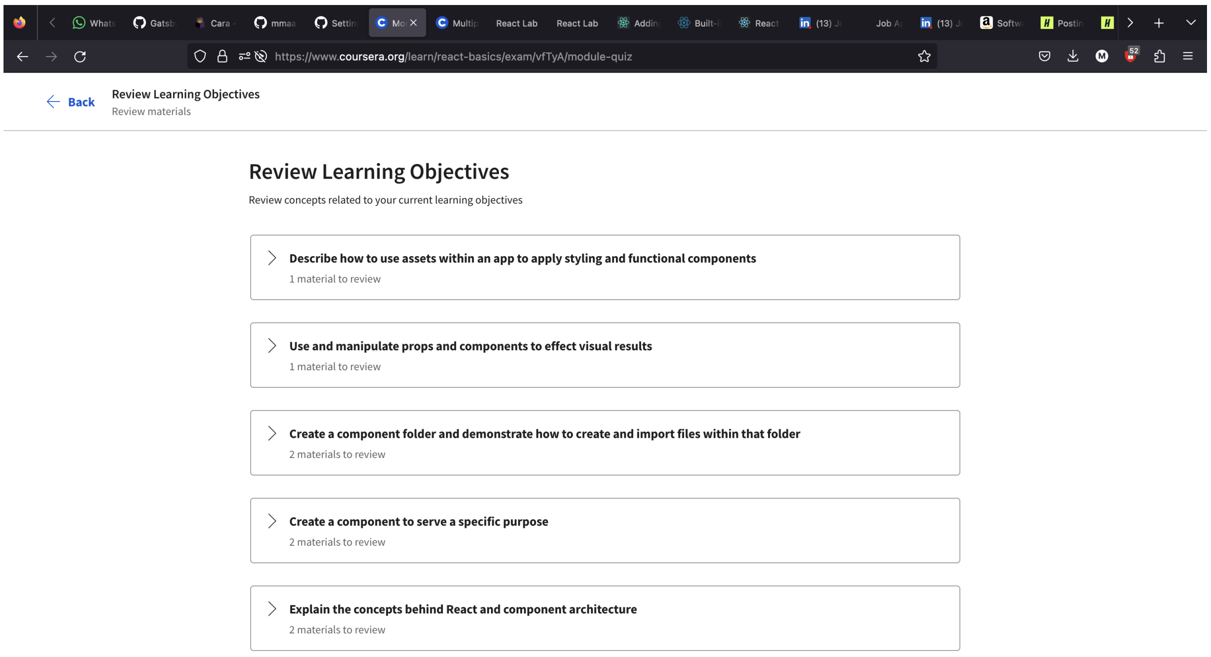
About react, component based architecture. component typele. virtual dom, how to create components, JSX, how component is built using JSX, creating components, importing and exporting, props, render components using props, jsx expressions.

---

quiz results

```
**Receive grade**

To Pass 80% or higher

### Your grade

90%

We keep your highest score

# Module Quiz

Graded Quiz. • 30 min

DueSep 17, 11:59 PM PDT

## Congratulations! You passed!

Grade received 90%

Latest Submission Grade 90%

To pass 80% or higher

**1.**Question 1

Why is React using the concept of components?

1 / 1 point

It improves the styling of your pages.
It helps accessibility readers for people who are visually impaired.

It allows the browser to render your pages faster.

It allows you to build more modular apps.

Correct

Correct! Components make React apps a lot more modular.

**2.**Question 2

What is the absolute minimum code that a component must have to be able to show somethin
g on a screen when rendered?

1 / 1 point

A named function declaration.

A named function declaration and an array of items inside of the function's body.

A named function declaration and some variables in the function’s body.

A named function declaration and a return statement with at least a single element with
 some text inside of it.

Correct

Correct! This is the absolute minimum of code a component must have to end up being rend
ered in the browser.

**3.**Question 3

What are the benefits of using props?

1 / 1 point

Props allow parent components to pass data to children components.

Props allow children components to update the values of each prop independent from their
parent component.

Props allow developers to write custom HTML tags.

Correct

That’s correct! Props facilitate the passing of data from parents to children component
s.

**4.**Question 4

You are tasked with building a web layout using React. The layout should have a header,
 a footer, and three products showing various data in the main part of the page. Choose
 the preferred component structure.

1 / 1 point

It should all fit into a single component named App component.
It should have the following components: Header, Main, Product, Footer (with the Product
component being imported into Main and rendered three times).

It should have a separate component for each link, paragraph, heading, etc.

Correct

That’s correct. Having such component structure would make all the building blocks of yo
ur React app at a correct amount of modularity.

##

**5.**Question 5

Which of the following keywords can you usually find in a React component?

1 / 1 point

function, props, return, export, default

module, function, prop, exported, default

modular, expression, prop, default

function, props, export, import, contain

Correct

That’s correct! These are all valid keywords, and they are commonly found in most React
 components.

**6.**Question 6

What is create-react-app?

1 / 1 point

It’s a stand-alone application on the web.

It’s an npm package used to build a boilerplate React app.

It’s a command you run when you want to serve your app in the browser.

It’s a command you can use in a component.

Correct

That’s correct! The create-react-app is a npm package that you can use to build a boiler
plate React app.

**7.**Question 7

Imagine you want to build a brand new React app, named “example”. Choose the correct com
mand to build a starter React app to work off of.

0 / 1 point

npm init react-app example
npm install react-app example

npm initialize react-app example

node init react-app example

Incorrect

Not quite. The word node is not used. Please go back and review *Setting up React projec
t in VS Code –* in lesson 1 of *Course Introduction*.

##

**8.**Question 8

True or false? When you write arrow functions, for any number of parameters other than a
single parameter, using parentheses around parameters is compulsory.

1 / 1 point

True.

False

Correct

Correct! When you write arrow functions, for any number of parameters other than a singl
e parameter, using parentheses around parameters is compulsory.

**9.**Question 9

True or false? You can use function calls in JSX.

1 / 1 point

True

False

Correct

Correct! You can use function calls in JSX.

**10.**Question 10

True or false? When an arrow function has a single parameter, you do not need to add par
entheses around the item parameter (to the left of the arrow).

1 / 1 point

True

False

Correct

Correct. When an arrow function has a single parameter, you do not need to add parenthes
es around the item parameter (to the left of the arrow).
```

Objectives completed



https://www.coursera.org/learn/react-basics/exam/vfTyA/module-quiz

Back

Review Learning Objectives

Review materials

Review Learning Objectives

Review concepts related to your current learning objectives

Describe how to use assets within an app to apply styling and functional components

1 material to review

Use and manipulate props and components to effect visual results

1 material to review

Create a component folder and demonstrate how to create and import files within that folder

2 materials to review

Create a component to serve a specific purpose

2 materials to review

Explain the concepts behind React and component architecture

2 materials to review