QUESTION 11

Node.js

POINTS

5

✖ Delete Question

PROBLEM

Insert Images Insert Field

```
What does the following Node.js program output?

![Screen_Shot_2020-05-18_at_2.22.01_PM.png](/files/0be3d646-da78-4714-
87fe-abec9f23edeb)

|____|
```

↳Add **Subquestion**

+ Add **Question 12**

Save Assignment

Q1 React Counter

10 Points

Little Timmy just learned **React.js** and wants to build a counter component for his website with it. `<Counter>` is a dynamic button that displays the number of times it has been clicked. Help him work through some of the problems he is facing in implementing this component.

On page load:

0

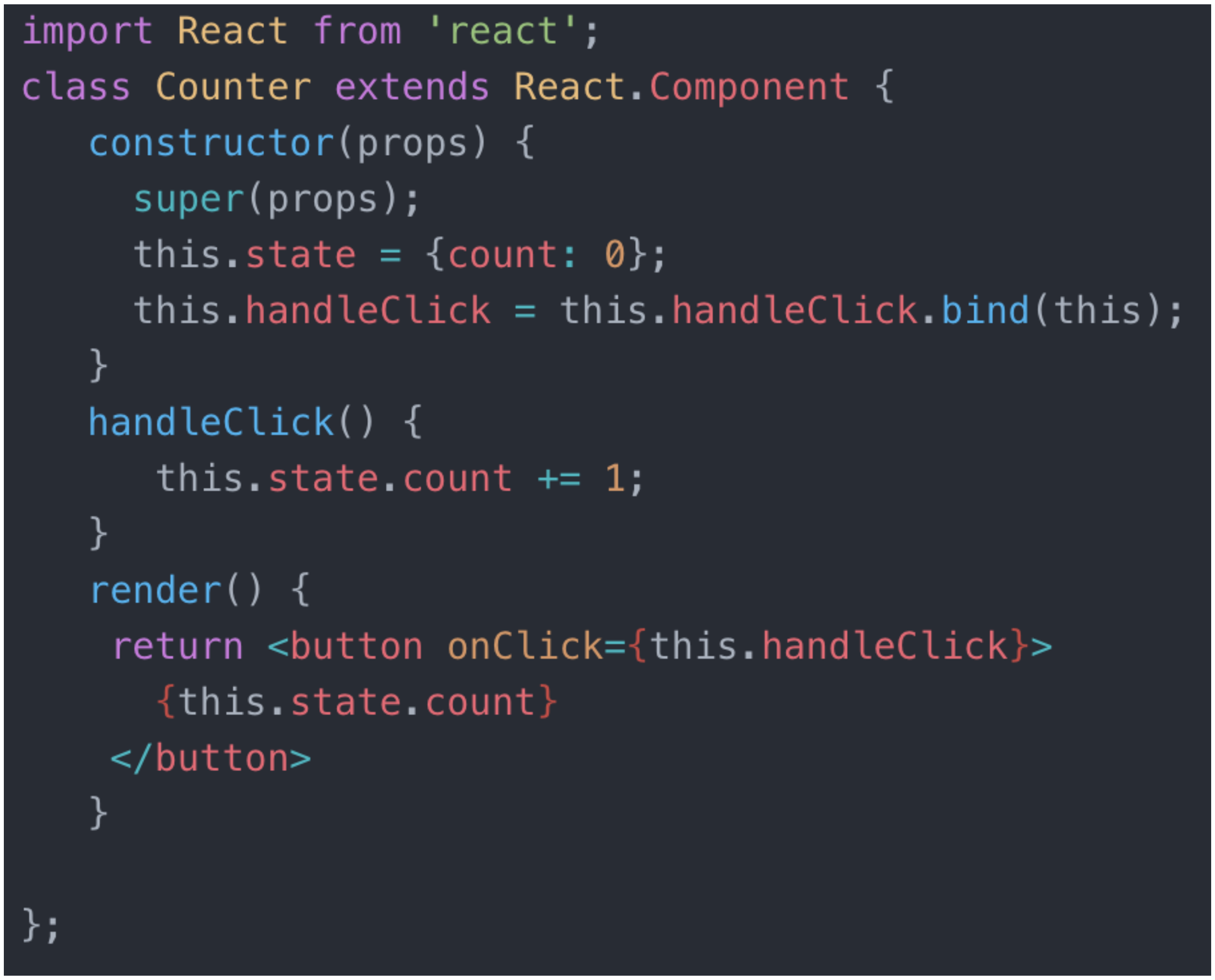
After clicking once:

1

Q1.1 Component Error

4 Points

Timmy's Counter component doesn't work correctly. Here's the code and a couple of questions about it:



```
import React from 'react';
class Counter extends React.Component {
   constructor(props) {
     super(props);
     this.state = {count: 0};
     this.handleClick = this.handleClick.bind(this);
   }
   handleClick() {
      this.state.count += 1;
   }
   render() {
    return <button onClick={this.handleClick}>
      {this.state.count}
    </button>
   }

};
```

1. Describe a change you could make to allow the component to work as specified.

2. Describe what behavior the user of the broken counter (before your fix) sees.

Q1.2 Component Event Handling

4 Points

React's use of JavaScript classes to specify React Components runs into the problem that JavaScript method functions don't directly work as DOM event handler. There many different approaches to solving this problem. Most of the approaches require code either in the class constructor or how the event handler function is specified in the render()

method's JSX code. Below is a table containing snippets of constructor code and render method JSX for possible solutions to this problem

Way	constructor(props)	render()
A	`this.handleClick = this.handleClick.bind(this);`	`<button onClick={this.handle`
B	`this.handleClick2 = this.handleClick.bind(this);`	`<button onClick={this.handle`
C	`// none`	`<button onClick={this.handle`
D	`// none`	`<button onClick={() => this.`
E	`this.handleClick = this.handleClick.bind(this);`	`<button onClick={() => this.`
F	`this.handleClick = this.handleClick.bind(this);`	`<button onClick={this.handle`

Some of the ways in the above table do not actually work. Describe the bad approach or approaches stating what is wrong with them.

Q1.3 Webpack Bundle

2 Points

While Timmy was debugging the component, he realized that he was spending a lot of time running `npm run build` every time he made a change. He wanted to increase his rate of iteration in the future by avoiding having to do so. He looked at his page `index.html` and noticed this at the bottom of the page:

```
<script src="compiled/counter.bundle.js"></script>
```

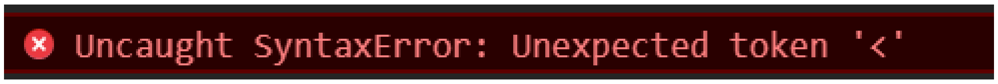
He wonders if it is faster to just include the JavaScript inline in the HTML file instead. To test his hypothesis, he runs a little experiment and made a test page:

```
<html>
<head>
  <title>Timmy's Brilliant Idea</title>
</head>
<body>
  <div id="reactapp"></div>

  <!-- Defines React.Component and ReactDOM.render -->
  <script src="./library/react.js"></script>

  <script>
    class App extends React.Component {
      render() {
        return (
          <div>This is so much faster!</div>
        );
      }
    }
    ReactDOM.render(<App />, document.getElementById("reactapp"));
  </script>
</body>
</html>
```

He loads up the page and immediately sees this error in the console:



```
Uncaught SyntaxError: Unexpected token '<'
```

What are the cause of this error and the fundamental problem with the approach?

Q2 Single-Page Applications

2 Points

The dictionary entry for the word "deep" reads:

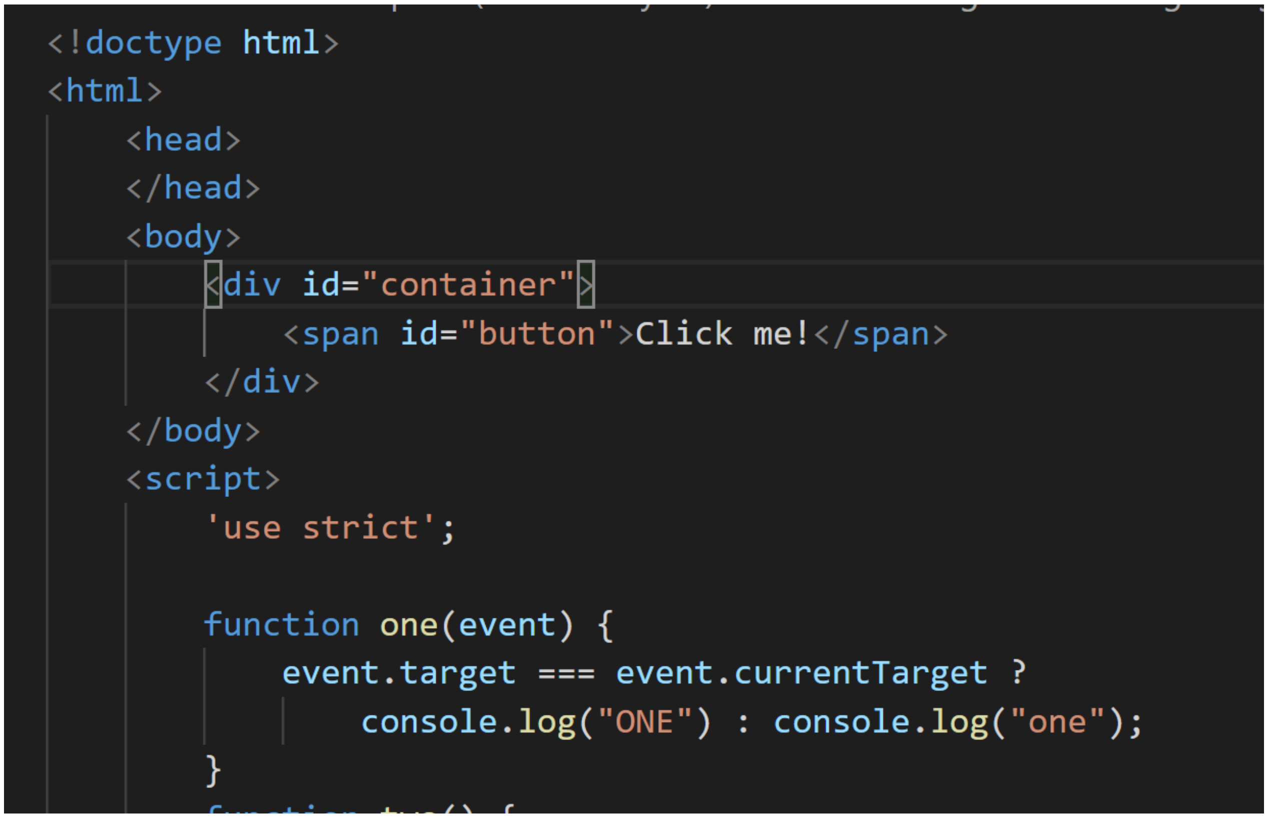
> extending far down from the top or surface

When used in the term "deep linking", what "top or surface" is being "extended far down" into? Explain your answer.

Q3 Events

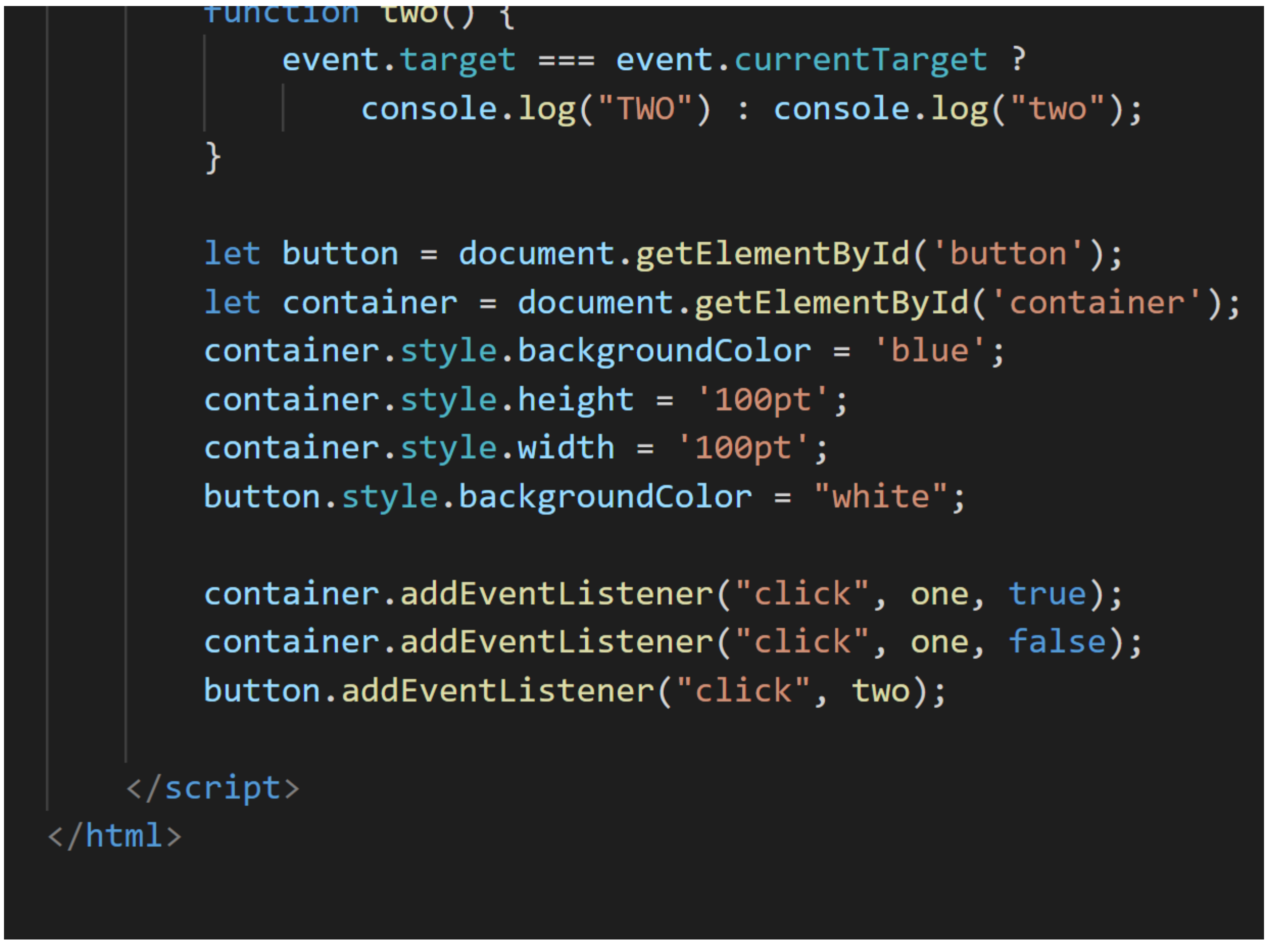
6 Points

The following HTML is loaded into the browser:



```
<!doctype html>
<html>
    <head>
    </head>
    <body>
        <div id="container">
            <span id="button">Click me!</span>
        </div>
    </body>
    <script>
        'use strict';

        function one(event) {
            event.target === event.currentTarget ?
                console.log("ONE") : console.log("one");
        }
```



```
        function two() {
            event.target === event.currentTarget ?
                console.log("TWO") : console.log("two");
        }

        let button = document.getElementById('button');
        let container = document.getElementById('container');
        container.style.backgroundColor = 'blue';
        container.style.height = '100pt';
        container.style.width = '100pt';
        button.style.backgroundColor = "white";

        container.addEventListener("click", one, true);
        container.addEventListener("click", one, false);
        button.addEventListener("click", two);

    </script>
</html>
```

Q3.1 Blue

3 Points

What will happen when the user clicks on the blue region around the button?

Q3.2 White

3 Points

What will happen when the user clicks "Click me!"?

Q4 DOM

6 Points

Suppose you have an HTML document’s body that looks like this:

```
<body>
  <div id="myDiv"></div>
  <script type="text/javascript" src="myCoolJS.js"></script>
</body>
```

And myCoolJS.js contains:

```
let myDiv = document.getElementById("myDiv");
let p1 = document.createElement("p");
let text1 = document.createTextNode("CS142");

let a1 = document.createElement("a");
let text2 = document.createTextNode("Link");

let div1 = document.createElement("div");
let text3 = document.createTextNode("Quiz2");

p1.appendChild(text1);
a1.appendChild(text2);
div1.appendChild(text3);

myDiv.appendChild(div1);
myDiv.appendChild(a1);
myDiv.insertBefore(p1, div1);

a1.setAttribute("href", "http://localhost:3000/");

myDiv.childNodes.forEach(function(item, index){
    item.setAttribute("class", "cs142-style-" + index);
});
```

Write the HTML equivalence of the content in the `div` tag with id “myDiv”.

Q5 Web Servers

2 Points

Modern web servers for JavaScript frameworks and single page applications are often considered to be more ‘light-weight’ than those for older web apps; in other words, they usually don’t have to do as much processing as before. Describe one major piece of functionality that moved from the web server to JavaScript in the browser in single page applications.

Q6 Server Communication

2 Points

Mendel has some changes he’s very excited to push to the CS142 class website. He makes the changes, but the next day, when a student loads the website on her laptop, the website looks just like it did before. Which of the following could explain the difficulty? (Select all that apply.)

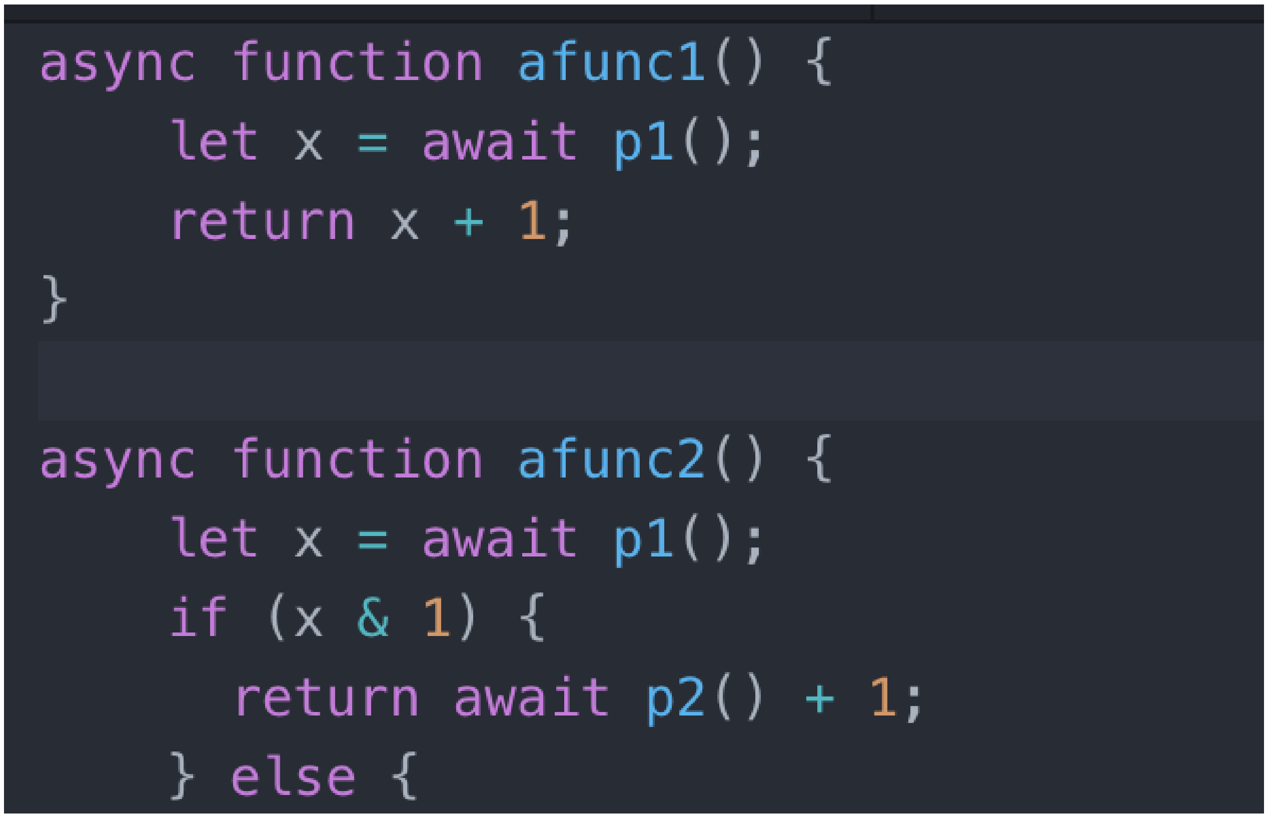
○ Without the real-time capabilities of WebSockets, we can’t expect the changes to be pushed to people’s browsers within a day of being updated.

- Mendel may be using a “stateful” rather than a “stateless” web server, which usually causes the server to send clients the same version of the site they loaded previously.
- [✓ The student may have a cached version of the website, which her computer loaded rather than fetching the newest version.]

Q7 Promises

6 Points

You have a job at a company coding in JavaScript using Promises for blocking functions. A fellow employee was recently fired for checking in some code that used the JavaScript async function and await syntax your common boss hates. Your first job is to rewrite the functions to use the traditional pre-await way of working with promises (i.e. .then() method). Code:



```
async function afunc1() {
    let x = await p1();
    return x + 1;
}

async function afunc2() {
    let x = await p1();
    if (x & 1) {
      return await p2() + 1;
    } else {
        return await p2() + await p3();
    }
}
```

The functions pN (`p1`, `p2`, and `p3`) used in the code run for N seconds before returning N (e.g. `p1()` returns a promise that resolves in 1 second and the promise resolves to 1).

Q7.1 afunc1

2 Points

Rewrite `afunc1` to not use `await`

Q7.2 afunc2

2 Points

Rewrite `afunc2` to not use `await`

Q7.3 Timing

2 Points

If you ran timing tests on function calls to `afunc1` and `afunc2`, how long do you think the call will take? Express your answer rounded to the closest number of whole seconds. For example, we are asking about the time the following statement would take:

```
let z = afunc1();
```

`afunc1()` call time rounded to whole seconds:

[✓ 0 seconds]

- 1 seconds
- 2 seconds
- 3 seconds
- 4 seconds

`afunc2()` call time rounded to whole seconds:

- [✓ 0 seconds]
- 1 seconds
- 2 seconds
- 3 seconds
- 4 seconds

Q8 REST

2 Points

You see the following form on Twitter that enables you to create a tweet and see it live:

```
<form action="/tweet/create" method="get" autocomplete=”off” target=”_blank” nov
  <label for="tweet"> I have a bike</label><br>
  <input type="text" name="tweet" placeholder=”Your tweet here.” />
  <input type="submit" value="Submit">
</form>
```

The form works as expected when submitted. Something included in the form’s HTML suggests, however, that Twitter’s (hypothetical) API is not fully RESTful as a result. What is it? Explain.

Q9 Node.js

2 Points

A student in CS142 implemented the following lines in a Node.js program to read and store the contents of three files in the JavaScript object `fileContents`. The console.log statement prints the number of properties in the object `fileContents`.

```
var fileContents = {};
['f1','f2','f3'].forEach(function (fileName) {
     fs.readFile(fileName, function (error, dataBuffer) {
        assert(!error);
        fileContents[fileName] = dataBuffer;
     });
});
console.log(Object.keys(fileContents).length);
```

The console.log statement will output:

[✓ 0]

1

3

Can't tell from the above code - Timing dependent

Explain your answer:

Q10 Node.js

2 Points

JavaScript is single-threaded yet Node.js using the JavaScript runtime supports many concurrent operations. Describe how a web server implemented on Node.js can process requests concurrently even though only a single thread is used for execution.

Q11 Node.js

5 Points

What does the following Node.js program output?

```
const events = require("events")
let result = 0;
let eventEmitter = new events.EventEmitter();

eventEmitter.on("foo", function () { console.log("Hello, world!") });
eventEmitter.on("bar", function (a, b) {
  result += (a + b);
  console.log(result);
});
eventEmitter.on("bar", function (a, b) {
  result *= (2 + b - a);
  console.log(result);
});
eventEmitter.on("error", function () { console.log("Error Detected!!") });

eventEmitter.emit("foo");
eventEmitter.emit("bar", 1, 3);
eventEmitter.emit("who");
```