Started on	Wednesday, 8 September 2021, 4:14 PM
State	Finished
Completed on	Wednesday, 8 September 2021, 4:16 PM
Time taken	2 mins
Grade	**7.00** out of 7.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

A functional (or function) component can only be created using the **function** keyword.

Select one:

True [✓ False] ✓

Correct. Functional components may also be created using the arrow function syntax, which does not use the function keyword.

The correct answer is 'False'.

Question **2**

Correct

Mark 1.00 out of 1.00

When the unary plus operator + is placed in front of a string, and that string is only operand, for example, like this:

```
+"3"
```

It will:

Select one:

- a. Double the string.
- b. Turn the string into a number, if it can be converted to a number, or convert it to Boolean *false* if it cannot be converted to a number.
- [✓ c.] Turn the string into a number, if it can be converted to a number, or convert it to the special value *NaN* if it cannot be converted to a number. ✓
- d. Add one to the number inside it.

Your answer is correct.

The correct answer is: Turn the string into a number, if it can be converted to a number, or convert it to the special value *NaN* if it cannot be converted to a number.

Question **3**

Correct

Mark 3.00 out of 3.00

Which of these following examples would work as a return from a functional component?

Select one or more:

[✓ a.]

```
function Example() {
    return (
        <div>
            <p>Hello World</p>
        </div>
    );
}
```

✓ Correct. You can return multiple JSX elements as long as there's only one JSX element at the top level.

b.

```
function Example() {
    return (
        <div>Hello</div>
        <div>World</div>
    );
}
```

c.

```
function Example(props) {
    return (
        <p>{this.props.name}</p>
    );
}
```

[✓ d.]

```
const Example = ({name}) => {
    return (
        <p>{name}</p>
```

✓ Correct. Assuming that **name** is a valid property of **props**, the object destructuring syntax makes the variable **name** available to be used within this component.

```
    );
};
```

[✓] e.
```
function Example() {
    return <div />;
}
```
✓ Correct. It returns a single JSX element at the top level.

[ ] f.
```
const Example = () => {
    return (
        <span>{props.name}</span>
    );
};
```

Your answer is correct.

The correct answers are:

```
function Example() {
    return <div />;
}
```

,

```
function Example() {
    return (
        <div>
            <p>Hello World</p>
        </div>
    );
}
```

,

```
const Example = ({name}) => {
    return (
        <p>{name}</p>
    );
};
```

Question **4**

Correct

Mark 1.00 out of 1.00

When do you use the **render** attribute instead of **component** in a react-router <Route> component?

Select one:

- ○ a. When you are routing to a class component only.
- ○ b. Whenever you are using an exact path for the <Route>.
- ○ c. When you need to pass the react-router match.params object.
- [✓ d. When you need to pass props to the component being routed to.] ✓ Correct. Use the **component** attribute when you do not need to pass props, and **render** when you do.

Your answer is correct.

The correct answer is: When you need to pass props to the component being routed to.

Question **5**

Correct

Mark 1.00 out of 1.00

Presentational components do not ever contain any local state information.

Select one:

○ True

[✓ False] ✓

Correct. Presentational containers can contain local state information related to the UI, such as if a modal is hidden or visible.

The correct answer is 'False'.

« »