Encryption and encoding

Update the website's HTML, without using JavaScript or CSS, to make use of semantic elements so that:

* The classless outer *div* element is replaced with a more appropriate element.
* The divs with the *image* and *caption* classes are replaced with self-contained content elements.
* The divs with the *lorem-ipsum* and *description* classes are replaced with elements, so that by default only the contents of the *description* element are shown. When the contents of the description element are clicked, the visibility of the rest of the *lorem-ipsum* element is toggled.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Semantics</title>

</head>

<body>

<div>

<h1>Lorem Ipsum</h1>

<div class="image">

<img src="https://goo.gl/zF9eky" alt="Lorem Ipsum">

<div class="caption">Lorem Ipsum</div>

</div>

<div class="lorem-ipsum">

<div class="description">Lorem ipsum dolor sit amet, consectetur adipiscing elit...</div>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Curabitur vitae hendrerit mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Mauris lacinia scelerisque nibh nec gravida.

Duis malesuada nec nibh sit amet pulvinar.

Phasellus congue porttitor arcu, ut suscipit nibh aliquam vel.

Nunc arcu lectus, egestas ut sem ac, euismod porttitor eros.

Phasellus tincidunt consequat pharetra. Maecenas sodales purus at nulla finibus dapibus.

Nullam varius at nisl vel euismod. Fusce aliquet ligula non tempor fermentum.

Nam fermentum posuere mauris, quis aliquam nibh dictum sed.</p>

</div>

</div>

</body>

</html>

* Classless div element: Wrong answer
* Self contained content: Wrong answer
* Toggling lorem-ipsum visibility: Wrong answe

Every user on your website has an image avatar that is displayed when they post a comment. You want to style these images differently from other images on your site. Add a CSS class named avatar that fulfils the following requirements:

1. The avatar's border is rounded, so that it appears as a circle.
2. The avatar's width and height are both 150px.
3. The avatar has a solid border, has a width of 2px, and be colored gray.

For example, the avatar in the template should look like

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Avatar</title>

<style>

/\* Write your CSS solution here (do not edit the surrounding HTML) \*/

</style>

</head>

<body>

<img class="avatar" src="https://goo.gl/khGCrk" alt="avatar" />

</body>

</html>

*The TextInput*component renders an *input* element in the DOM and accepts a ref that is forwarded to that input element. Finish the *FocusableInput*component:

* The component should accept a *focused*prop.
* When the *focused*prop is changed from false to true, and the *input*is not focused, it should receive the focus.
* If on mounting the *focused*prop is true, the input should receive the focus.

class Input extends React.PureComponent {

render() {

let {forwardedRef, ...otherProps} = this.props;

return <input {...otherProps} ref={forwardedRef} />;

}

}

const TextInput = React.forwardRef((props, ref) => {

return <Input {...props} forwardedRef={ref} />

});

class FocusableInput extends React.Component {

ref = React.createRef()

render() {

return <TextInput ref={this.ref} />;

}

// When the focused prop is changed from false to true,

// and the input is not focused, it should receive focus.

// If focused prop is true, the input should receive the focus.

// Implement your solution below:

componentDidUpdate(prevProps) {}

componentDidMount() {}

}

FocusableInput.defaultProps = {

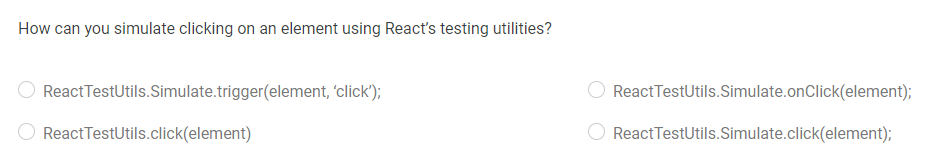
focused: false

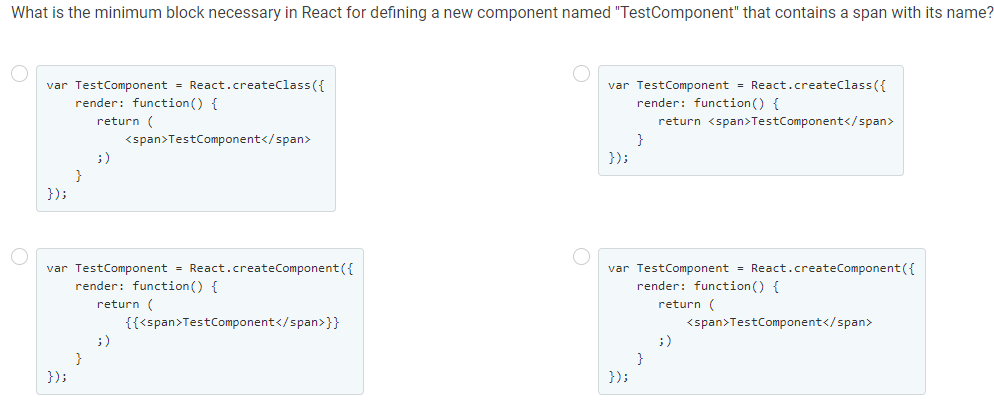
};

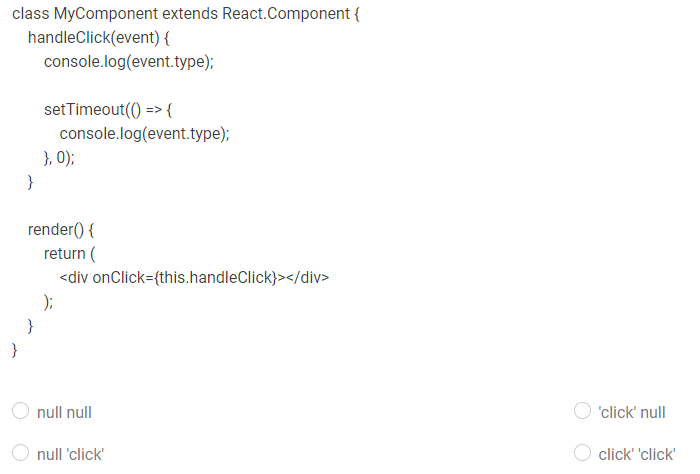
const App = (props) => <FocusableInput focused={props.focused} />;

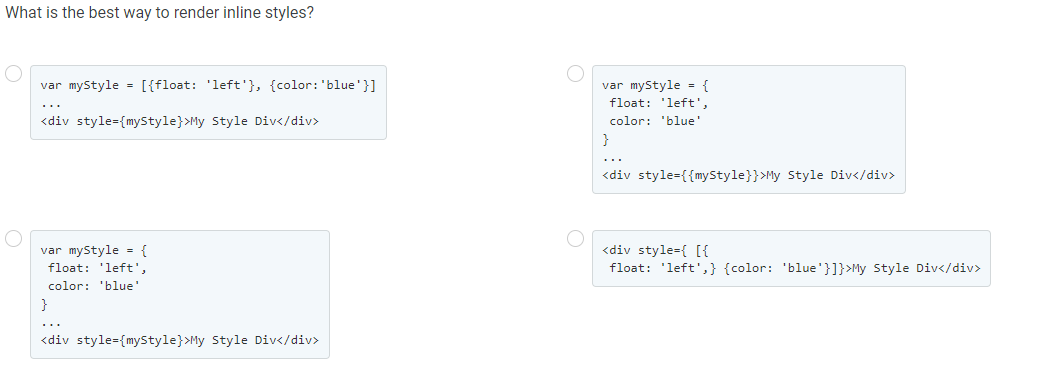
document.body.innerHTML = "<div id='root'></div>";

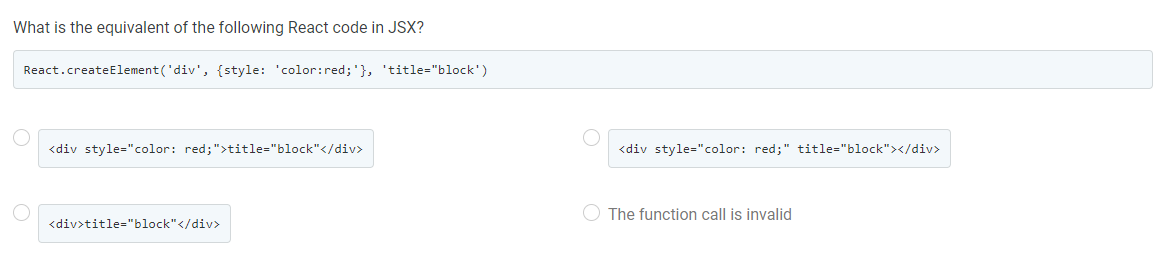
const rootElement = document.getElementById("root");

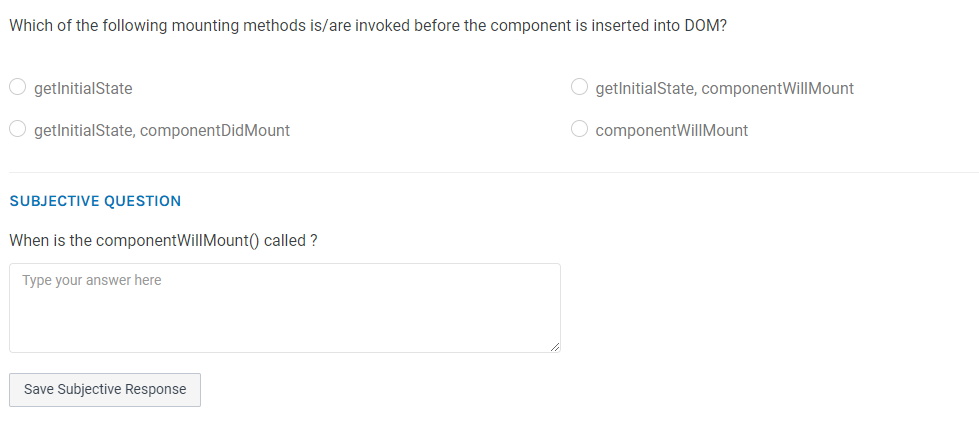
ReactDOM.render(<App />, rootElement);

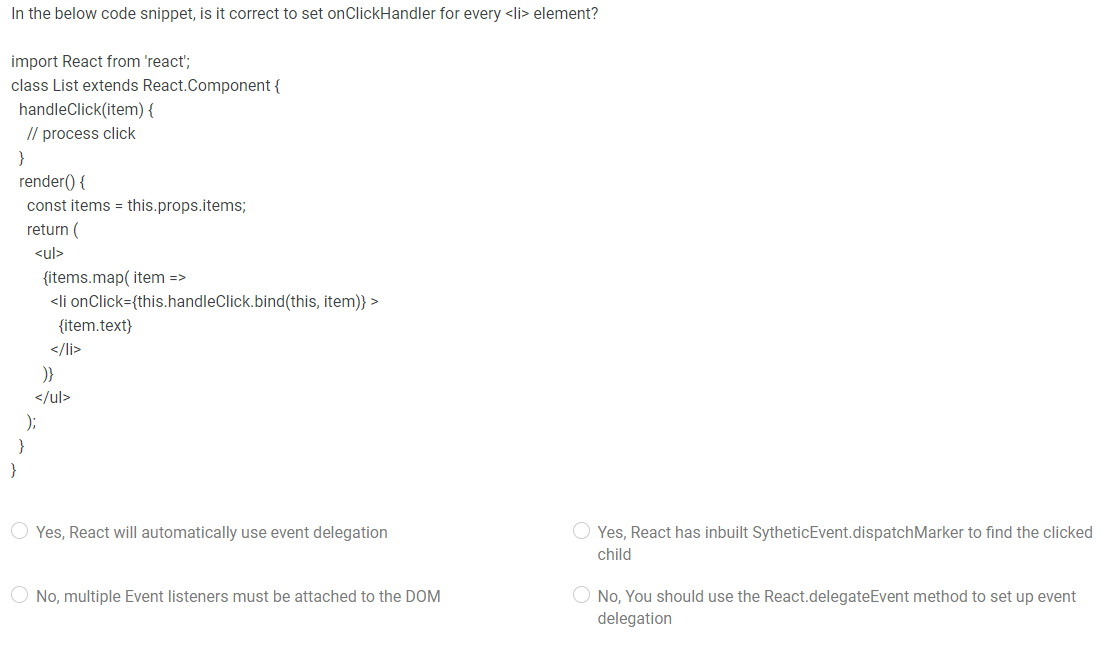


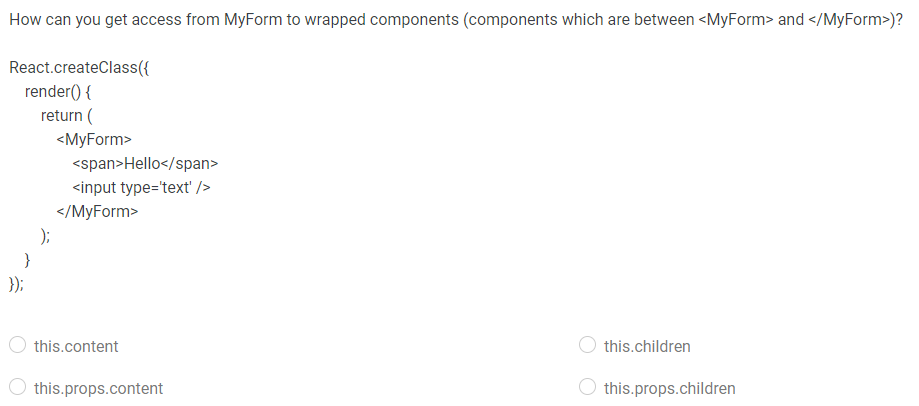


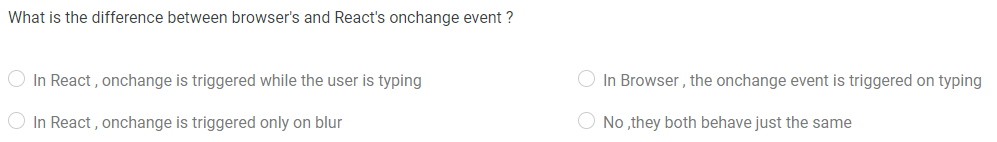




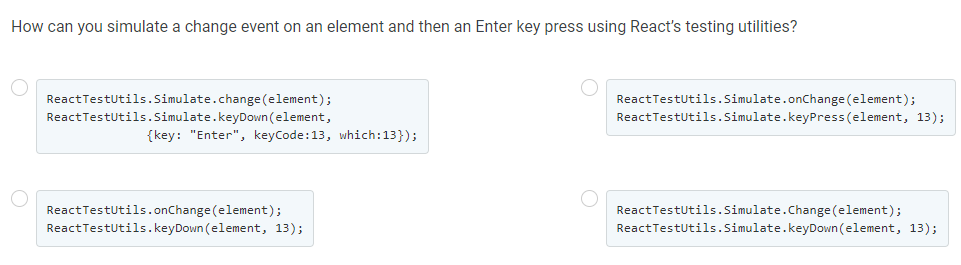


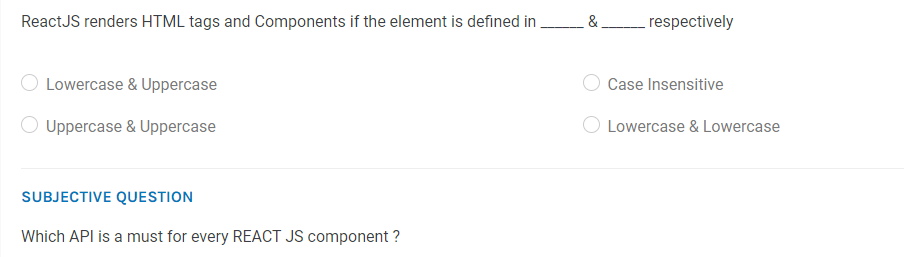




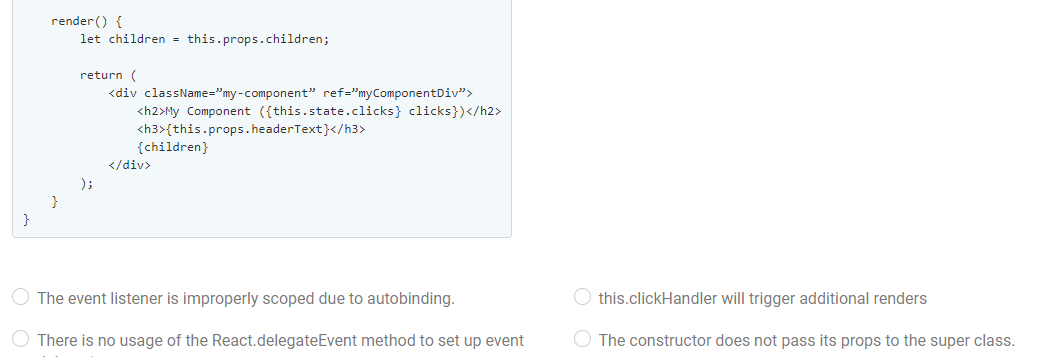


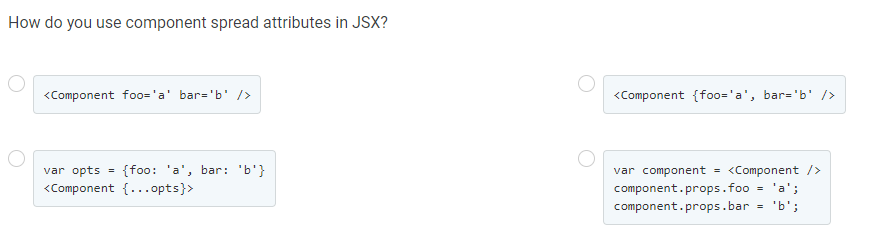












const TodoItem = (props) => <li onClick={props.onClick}>{props.item.text}</li>

class TodoList extends React.Component {

render() {

const { items, onClick } = this.props;

return (<ul onClick={onClick}>

{items.map((item, index) =>

<TodoItem item={item} key={index} onClick={this.handleItemClick.bind(this, item)}/>)}

</ul>);

}

handleItemClick(item, event) {

console.log(item)

event.preventDefault();

event.stopPropagation();

}

}

const items = [ { text: 'Buy grocery', done: true },

{ text: 'Play guitar', done: false },

{ text: 'Romantic dinner', done: false }

];

const App = (props) => <TodoList

items={props.items}

onItemClick={(item, event) => { console.log(item, event) }}

/>;

document.body.innerHTML = "<div id='root'></div>";

const rootElement = document.getElementById("root");

ReactDOM.render(<App items={items}/>, rootElement);

class Input extends React.PureComponent {

render() {

let {forwardedRef, ...otherProps} = this.props;

return <input {...otherProps} ref={forwardedRef} />;

}

}

const TextInput = React.forwardRef((props, ref) => {

return <Input {...props} forwardedRef={ref} />

});

class FocusableInput extends React.Component {

ref = React.createRef()

render() {

return <TextInput ref={this.ref} />;

}

// When the focused prop is changed from false to true,

// and the input is not focused, it should receive focus.

// If focused prop is true, the input should receive the focus.

// Implement your solution below:

componentDidUpdate(prevProps) {}

componentDidMount() {}

}

FocusableInput.defaultProps = {

focused: false

};

const App = (props) => <FocusableInput focused={props.focused} />;

document.body.innerHTML = "<div id='root'></div>";

const rootElement = document.getElementById("root");

ReactDOM.render(<App />, rootElement);