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 = {

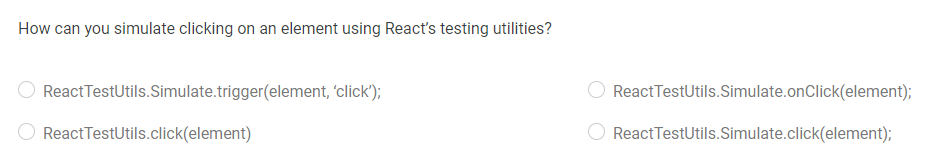
focused: false

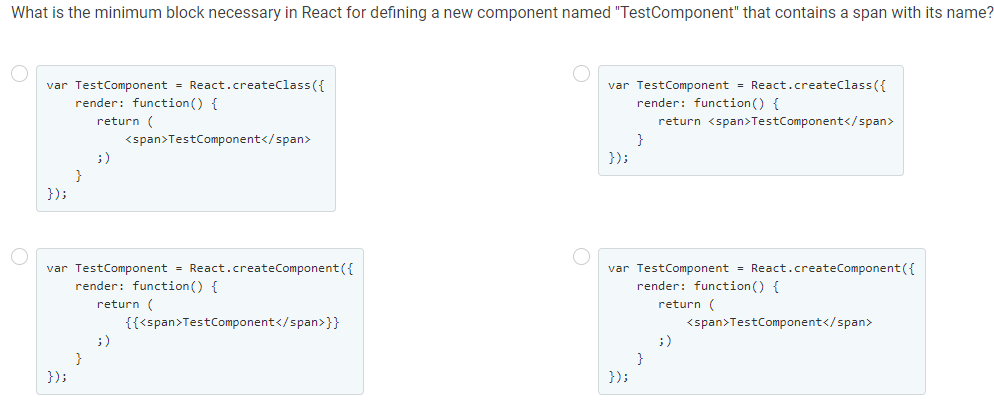
};

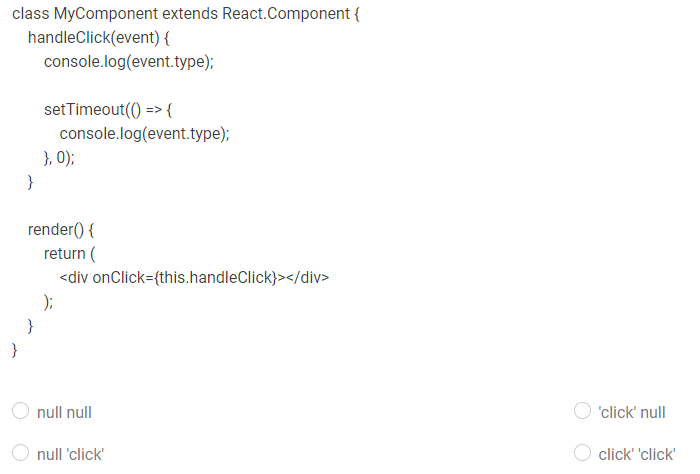
const App = (props) => <FocusableInput focused={props.focused} />;

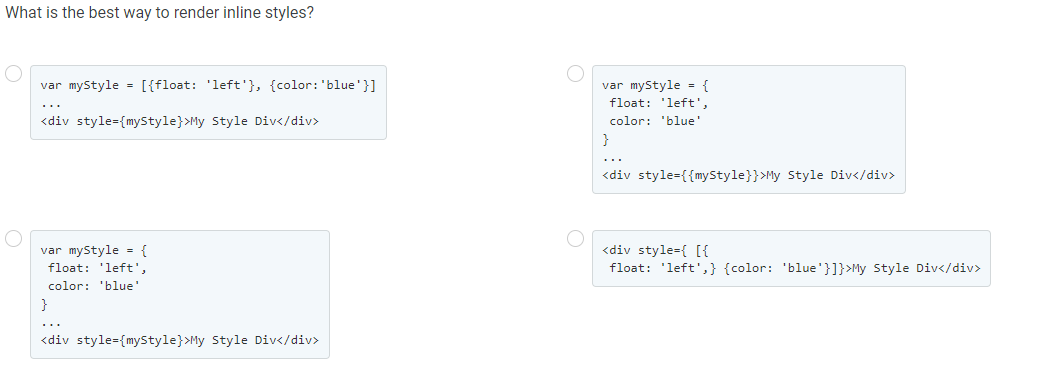
document.body.innerHTML = "<div id='root'></div>";

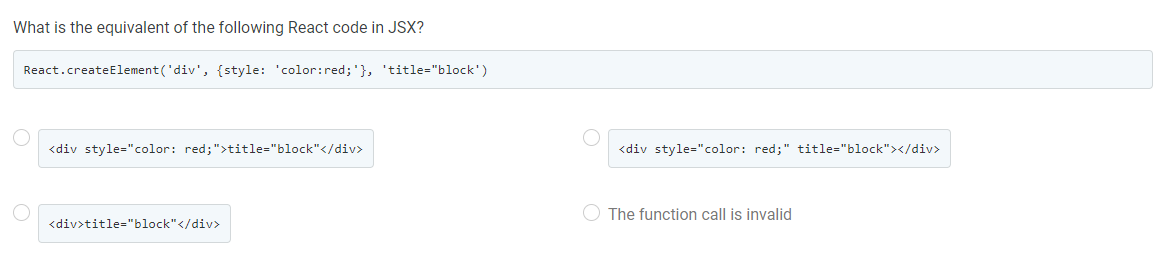
const rootElement = document.getElementById("root");

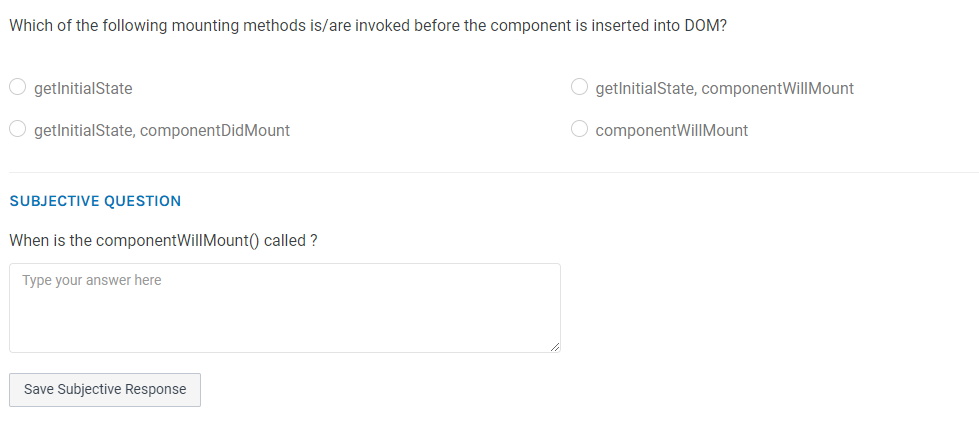
ReactDOM.render(<App />, rootElement);

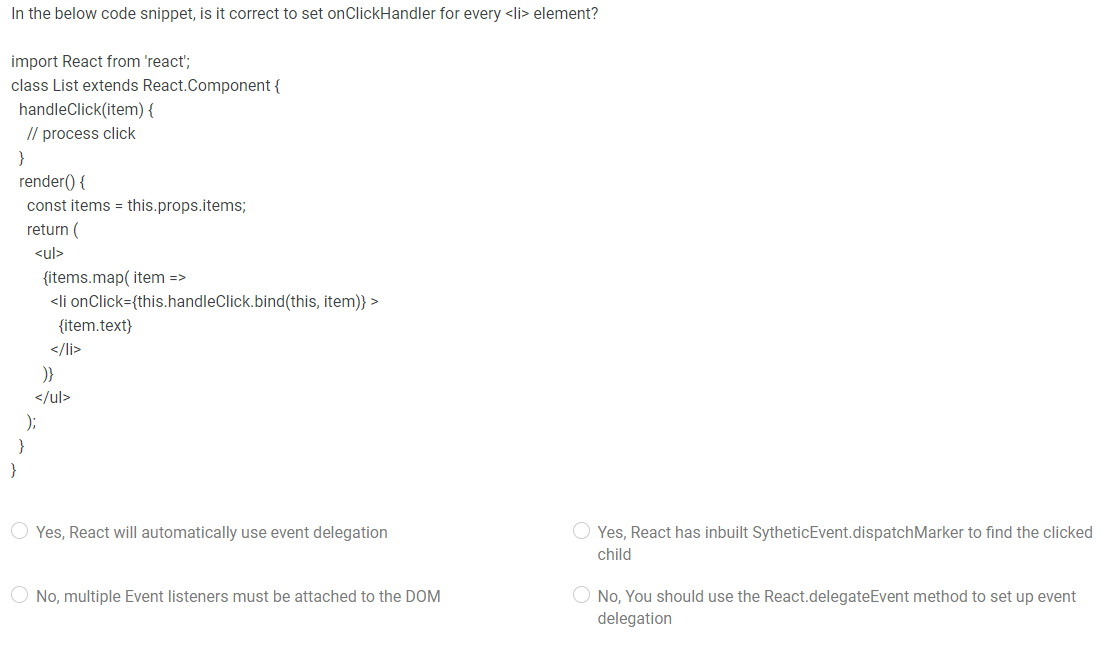


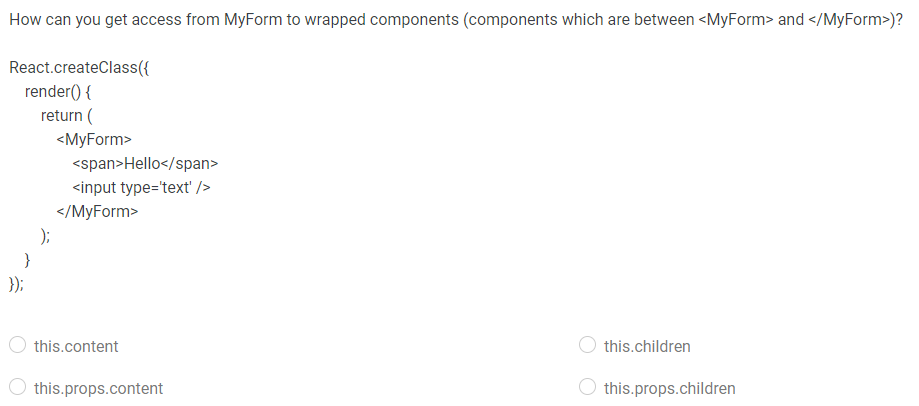


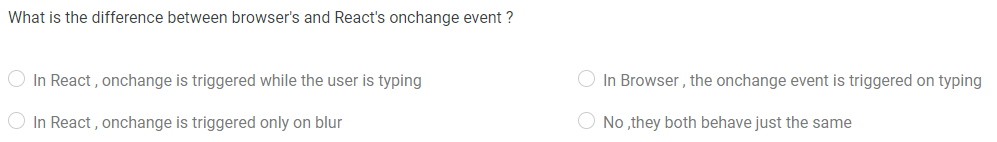




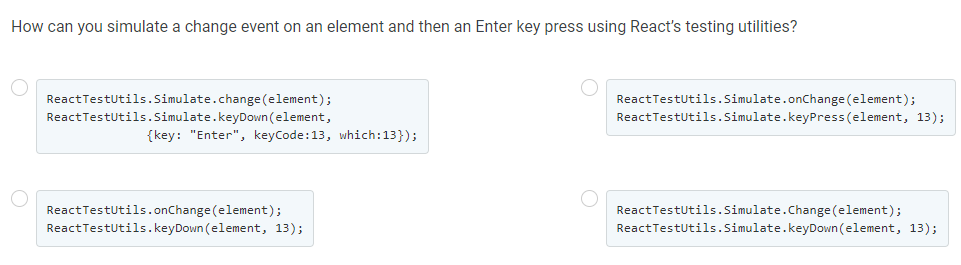


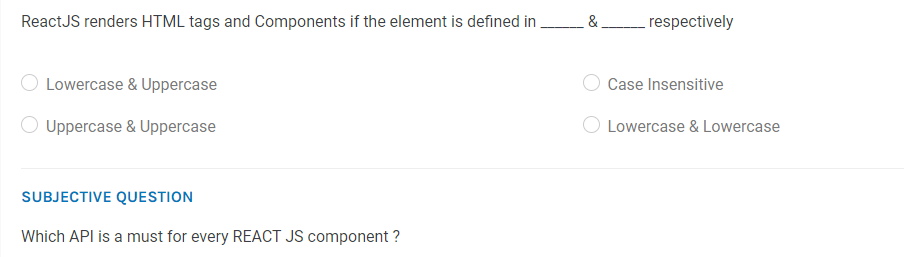




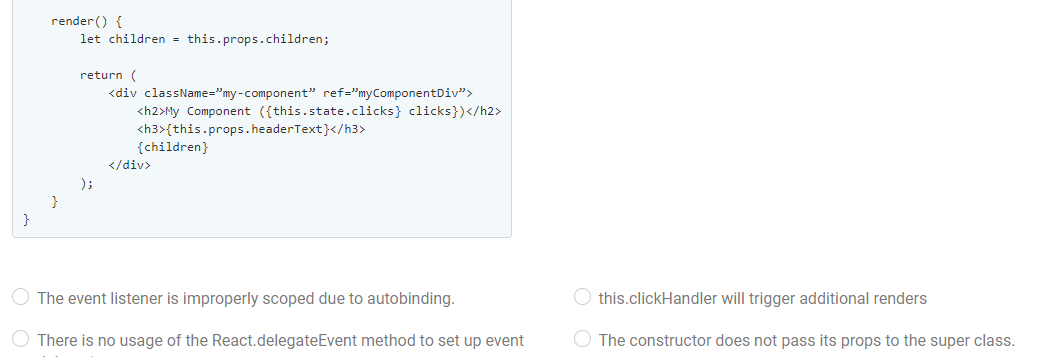


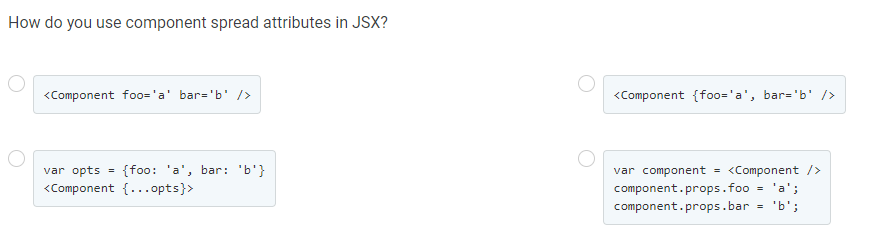












const TodoItem = (props) => <li onClick={props.onClick}>{props.item.text}</li>

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);