upGrad
#RahoAmbitious

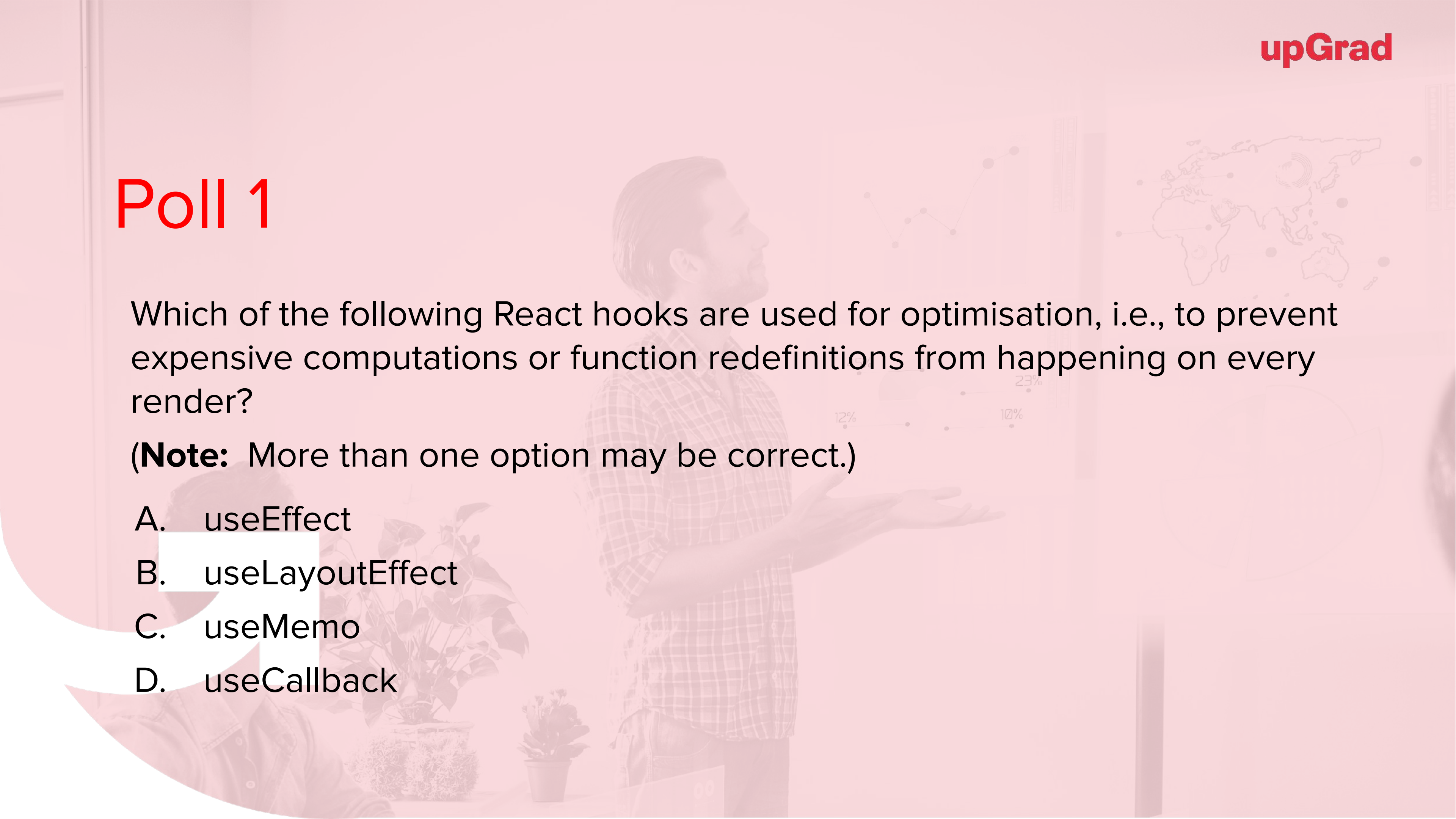
Full Stack Software Development

Course: Advanced Frontend Development Using React

Lecture on: Forms and Material-UI

In the last class, we discussed...

- useMemo and useCallback are built-in hooks in React that help with performance optimisation through memoization
- The useReducer hook is used to manage more complex state transitions and state variables which are correlated as compared to useState
- Creating our own hooks lets us extract component logic into reusable functions that can then be added into multiple components



upGrad

Poll 1

Which of the following React hooks are used for optimisation, i.e., to prevent expensive computations or function redefinitions from happening on every render?

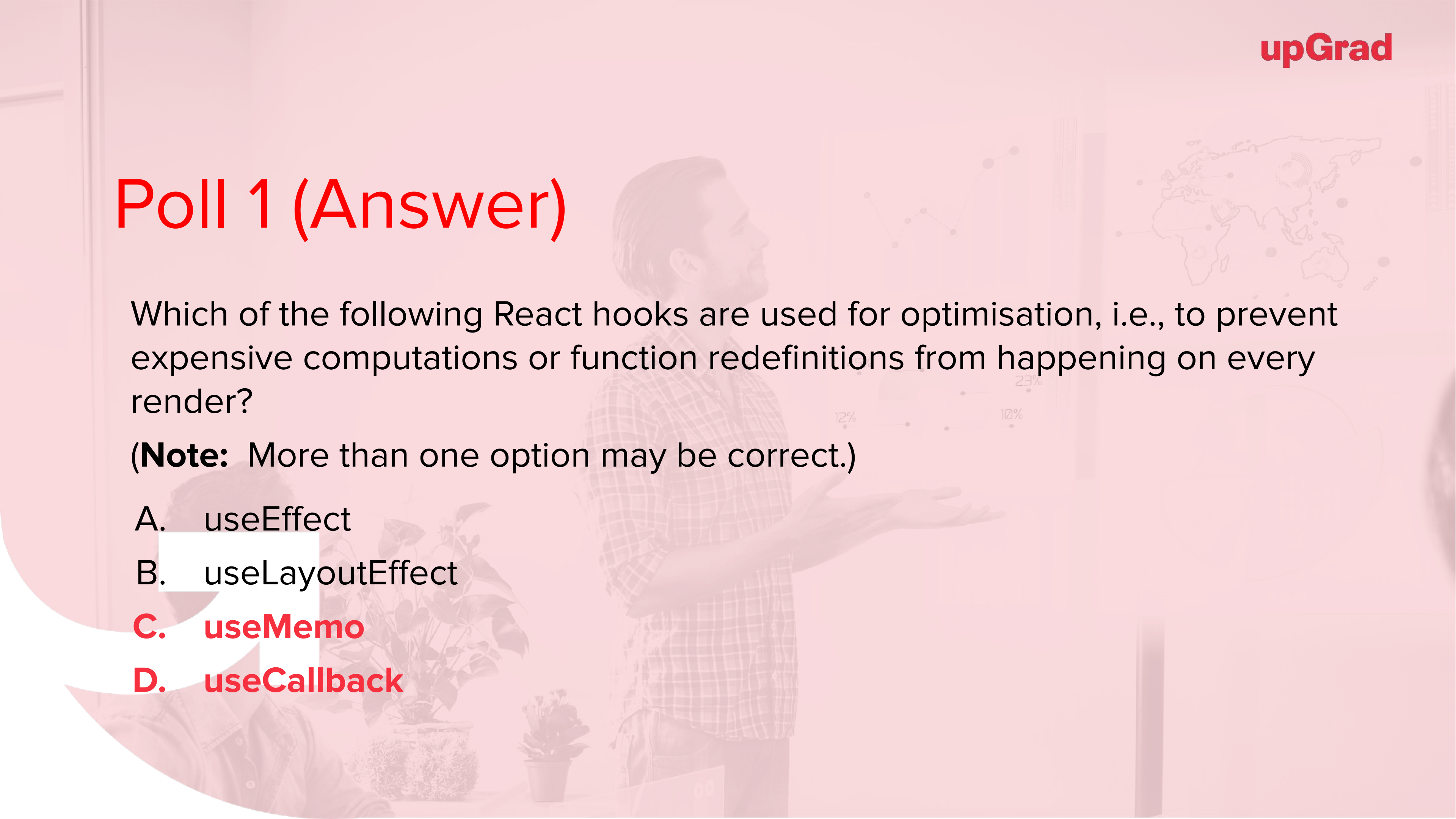
(**Note:** More than one option may be correct.)

A. useEffect

B. useLayoutEffect

C. useMemo

D. useCallback



upGrad

Poll 1 (Answer)

Which of the following React hooks are used for optimisation, i.e., to prevent expensive computations or function redefinitions from happening on every render?

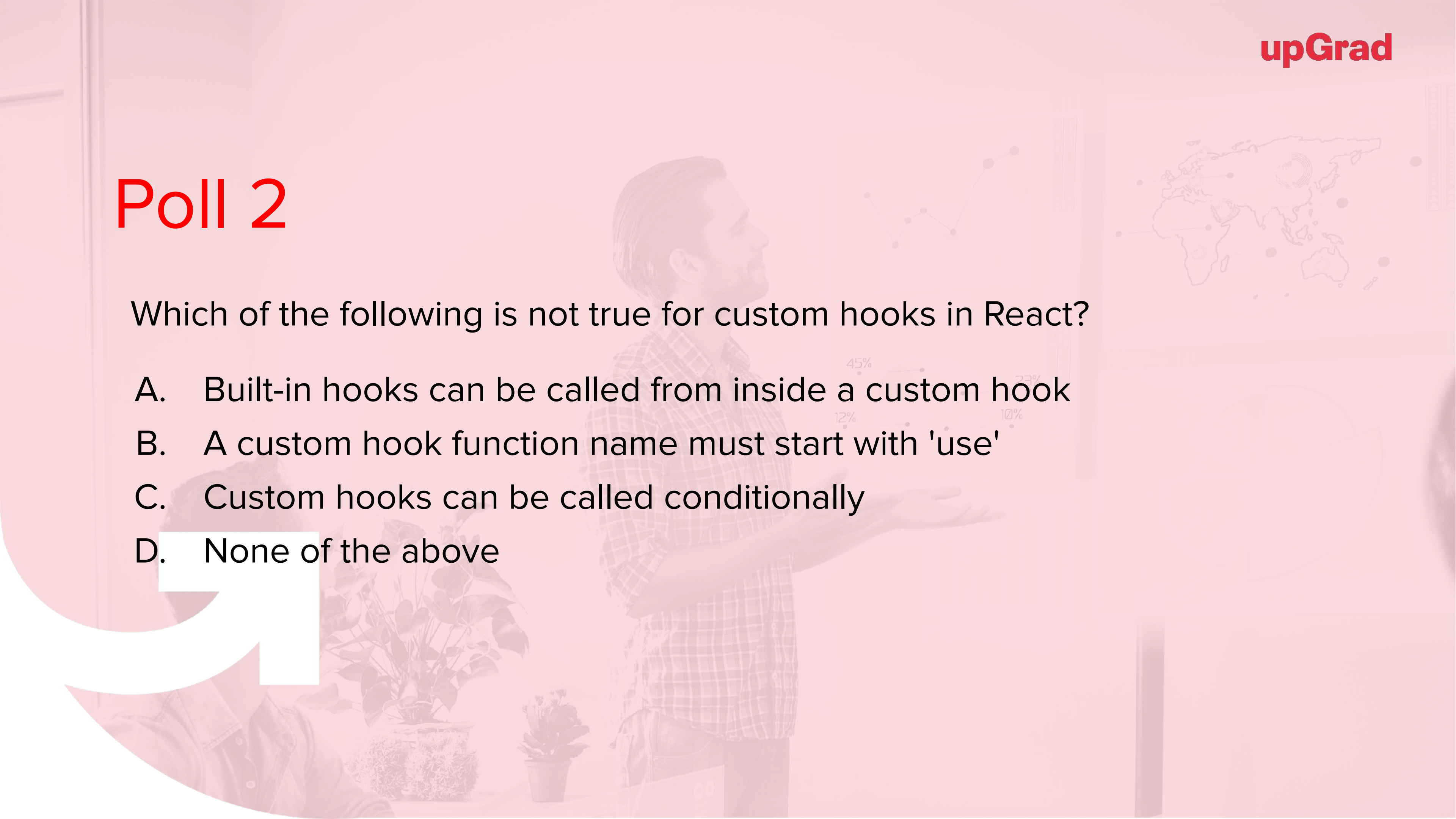
(**Note:** More than one option may be correct.)

A. useEffect

B. useLayoutEffect

C. useMemo

D. useCallback



Poll 2

Which of the following is not true for custom hooks in React?

A. Built-in hooks can be called from inside a custom hook

B. A custom hook function name must start with 'use'

C. Custom hooks can be called conditionally

D. None of the above

upGrad

Poll 2 (Answer)

Which of the following is not true for custom hooks in React?

A. Built-in hooks can be called from inside a custom hook

B. A custom hook function name must start with 'use'

C. Custom hooks can be called conditionally

D. None of the above

upGrad

Today’s Agenda

1. Implementing forms in React
   - Difference from traditional HTML forms
   - Controlled vs uncontrolled components
2. Form validation using Material-UI
   - Higher order validator components
   - Validation rules

upGrad

Implementing Forms in React

Forms in React

upGrad

Difference Between Forms in HTML and React

HTML

- Form data is handled by the DOM
- Data is stored directly in the DOM elements

Therefore, accessing this data involves reading it directly from the DOM. For example:

```
<form>
  <input type="text" name="email" />
</form>

const email = document.querySelector('input[name="email"]').value;
```

Forms in React

upGrad

Difference Between Forms in HTML and React

React

- Form data is managed in the components
- Data is stored in the component state

We will discuss an example of this approach shortly.

Forms in React

upGrad

Controlled Components

A ‘controlled’ React component renders a form with the following properties:

- React state acts as the single source of truth
- The component that renders the form also controls what happens on user input
- The form data is maintained in the internal state of the form component

Forms in React

upGrad

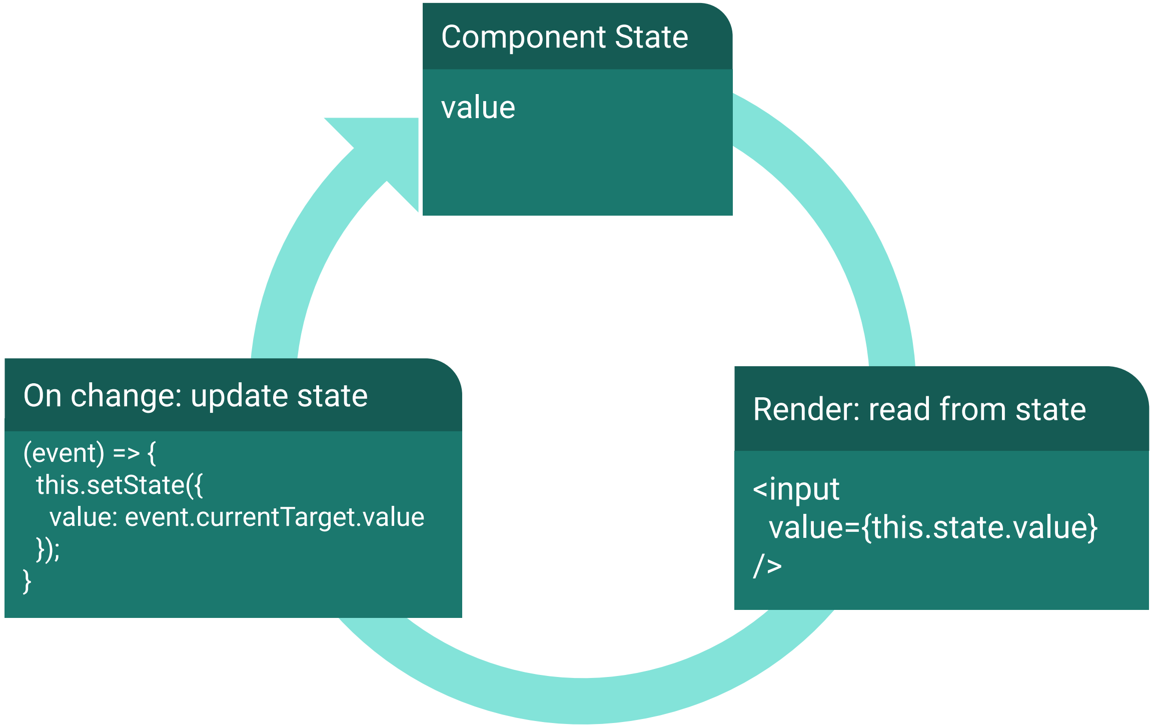
Controlled Form Component

```
class MyForm extends React.Component {
    onChange = (e) => {
      this.setState({ name: e.target.value });
    }
    onSubmit = (e) => {
      // Prevent the page from reloading
      e.preventDefault();
      // submit the data to the server...
    }
    render() {
      return (
        <form onSubmit={this.onSubmit}>
          <label>Enter name</label>
          <input type="text" name="name" value={this.state.name} onChange={this.onChange} />
          <input type="submit" value="Submit" />
        </form>
      );
    }
}
```

Forms in React

upGrad

How do Controlled Components Work?



```
(event) => {
  this.setState({
    value: event.currentTarget.value
  });
}
```

```
<input
  value={this.state.value}
/>
```

Forms in React

upGrad

Textarea

In HTML, there is no value attribute for **textarea**, but to help us turn it into a controlled component, React supports a ***value*** prop.

Textarea syntax in HTML:

```
<textarea>
  Input content of text area
</textarea>
```

textarea syntax in React (value prop)

```
<textarea value="Input content of text area" />
```

Forms in React

Select

React supports the value prop on select components to specify the currently selected option. For example:

```
<select value={this.state.value} onChange={e => this.setState({ value: e.target.value })}>
  <option value="foo">Foo</option>
  <option value="bar">Bar</option>
  <option value="baz">Baz</option>
</textarea>
```

Forms in React

upGrad

File Input

The value of a file input is read-only; hence, it is not possible for React to manage it.

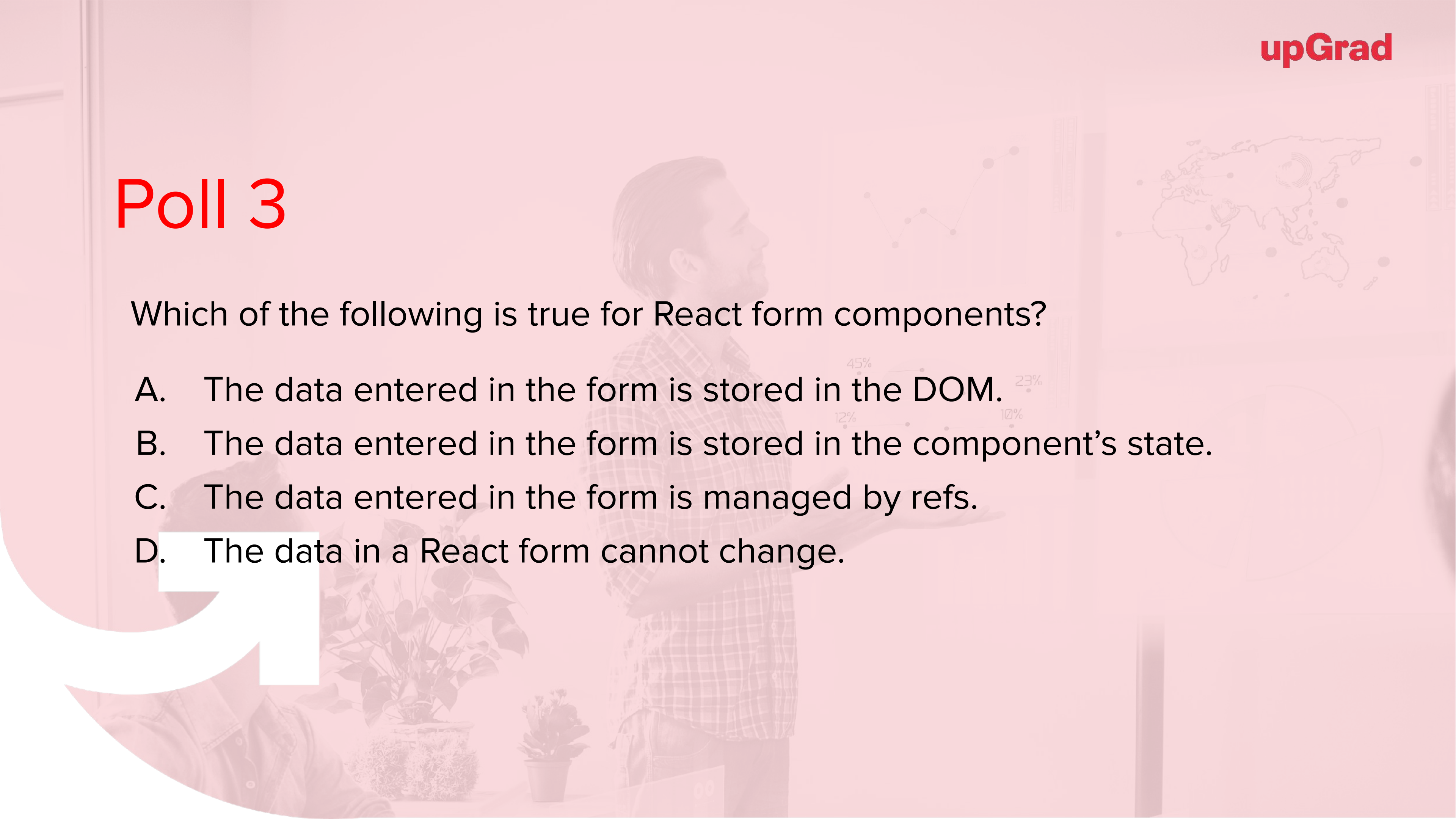
Therefore, <input type="file"> is an uncontrolled component in React.

Forms in React

upGrad

Form With Multiple Inputs (Generic)

```
class MyForm extends React.Component {
    onChange = (e) => {
      const value = e.target.type === 'checkbox' ? e.target.checked : e.target.value;
      this.setState({ [e.target.name]: value });
    }
    onSubmit = (e) => {
      e.preventDefault();
      // submit the data to the server...
    }
    render() {
      return (
        <form onSubmit={this.onSubmit}>
          <label>Enter name</label>
          <input type="text" name="name" value={this.state.name} onChange={this.onChange} />
          <label>Enter phone number</label>
          <input type="number" name="phone" value={this.state.phone} onChange={this.onChange} />
          <input type="submit" value="Submit" />
        </form>
      );
    }
}
```



upGrad

Poll 3

Which of the following is true for React form components?

A. The data entered in the form is stored in the DOM.
B. The data entered in the form is stored in the component’s state.
C. The data entered in the form is managed by refs.
D. The data in a React form cannot change.

upGrad

Poll 3 (Answer)

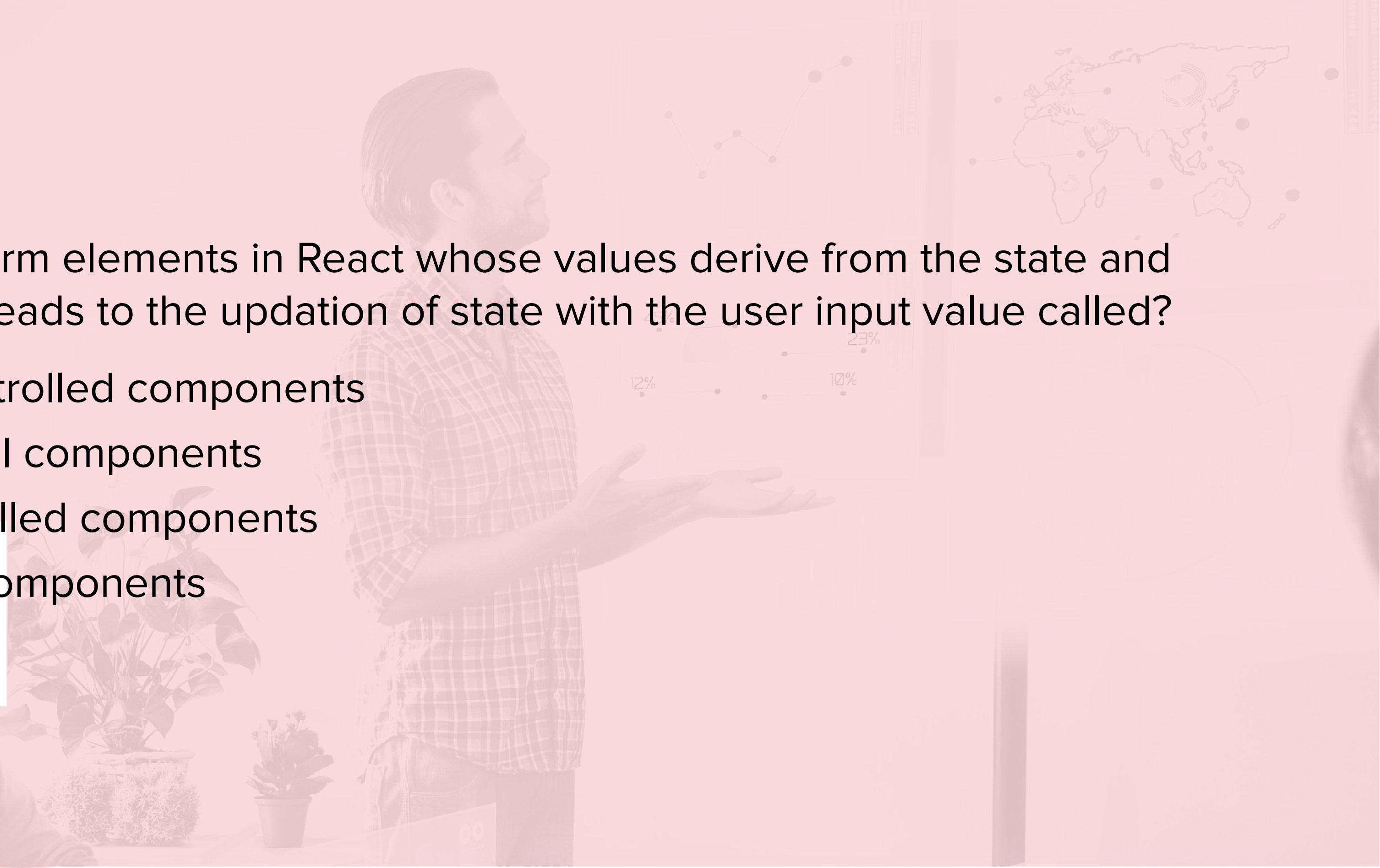
Which of the following is true for React form components?

A. The data entered in the form is stored in the DOM.

B. The data entered in the form is stored in the component’s state.

C. The data entered in the form is managed by refs.

D. The data in a React form cannot change.



Poll 4

What are form elements in React whose values derive from the state and user input leads to the updation of state with the user input value called?

A. Uncontrolled components
B. Stateful components
C. Controlled components
D. Pure components

upGrad

Poll 4 (Answer)

What are form elements in React whose values derive from the state and user input leads to the updation of state with the user input value called?

A. Uncontrolled components

B. Stateful components

C. Controlled components

D. Pure components

Form Validation Using Material-UI

Form Validation

upGrad

The Need for Validation

There are two primary reasons for which the data needs to be validated before it is sent to the server for persisting:

- Security: Being able to input random strings opens up our database and web application to a variety of attack vectors, such as SQL injection and cross-site scripting (XSS)
- Ensuring sanity of the input data: Validation helps ensure that some simple rules are being followed, such as a mandatory form field or the input format of a particular field. This, in turn, helps the user enter valid values and correct data to some extent

Form Validation

Validation in React Forms

You can add data validation logic in either:

- onChange or
- onSubmit

To this end, you already have libraries available to help you without having to reinvent the wheel!

Material-UI

upGrad

Component Library

A React component library/design system: https://material-ui.com/

Material Design is a design language developed by Google in 2014. According to Google, it is a visual language that synthesises the classic principles of a good design, with the innovation of technology and science.

Provides a set of commonly used components with built-in theming support and other common functionalities.

Material-UI

upGrad

Component Example

For example, you can implement a delete button with a corresponding icon in your component with a few lines of code depicted below-

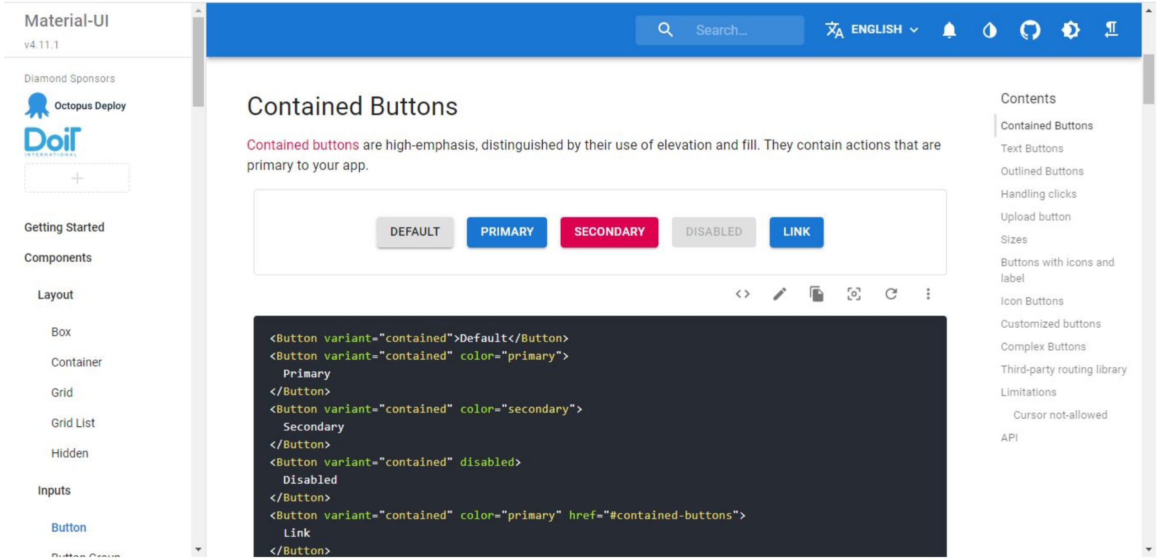
```
import Button from '@material-ui/core/Button';
import DeleteIcon from '@material-ui/icons/Delete';

<Button
  variant="contained"
  color="secondary"
  className={classes.button}
  startIcon={<DeleteIcon />}
>
  Delete
</Button>
```

Material-UI

upGrad

Components

On the Material-UI website, there is full-fledged documentation of all the different variants of the components and how to configure them as per our use.



Contained Buttons

Contained buttons are high-emphasis, distinguished by their use of elevation and fill. They contain actions that are primary to your app.

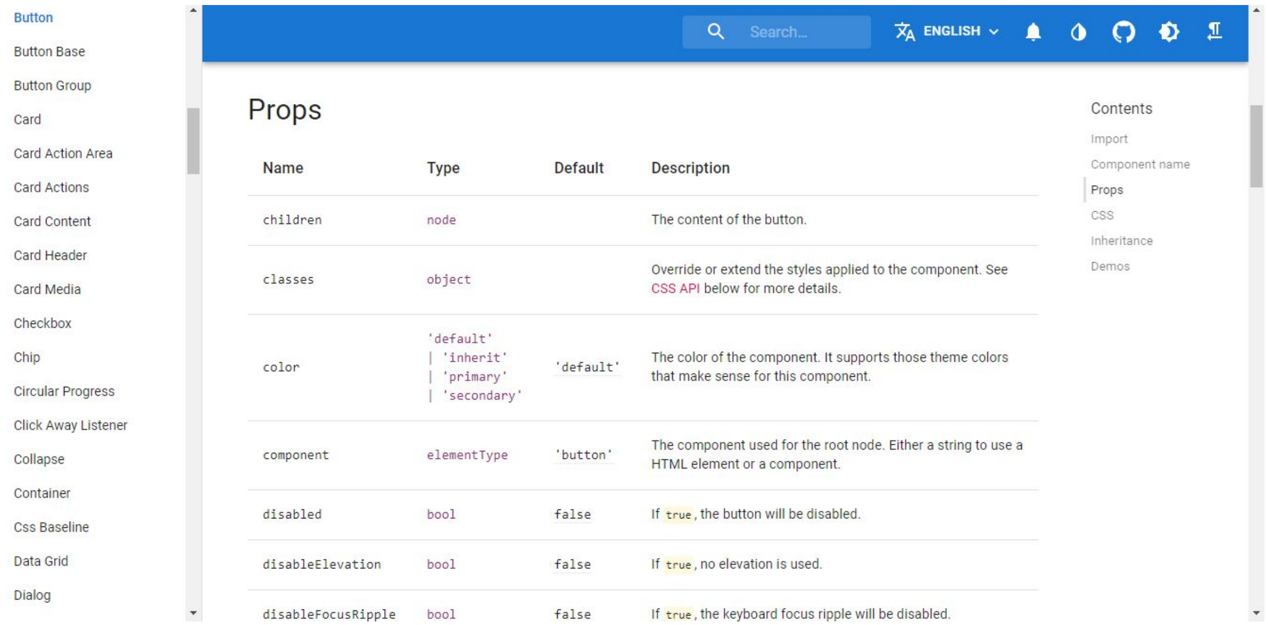
```
<Button variant="contained">Default</Button>
<Button variant="contained" color="primary">
  Primary
</Button>
<Button variant="contained" color="secondary">
  Secondary
</Button>
<Button variant="contained" disabled>
  Disabled
</Button>
<Button variant="contained" color="primary" href="#contained-buttons">
  Link
</Button>
```

Material-UI

upGrad

Component API

Detailed API (props) of the individual components is also well-documented.



Props

Name	Type	Default	Description
children	node		The content of the button.
classes	object		Override or extend the styles applied to the component. See CSS API below for more details.
color	'default' | 'inherit' | 'primary' | 'secondary'	'default'	The color of the component. It supports those theme colors that make sense for this component.
component	elementType	'button'	The component used for the root node. Either a string to use a HTML element or a component.
disabled	bool	false	If true, the button will be disabled.
disableElevation	bool	false	If true, no elevation is used.
disableFocusRipple	bool	false	If true, the keyboard focus ripple will be disabled.

Material-UI

upGrad

Installation

Installation using NPM:

```
npm install @material-ui/core --save-dev
```

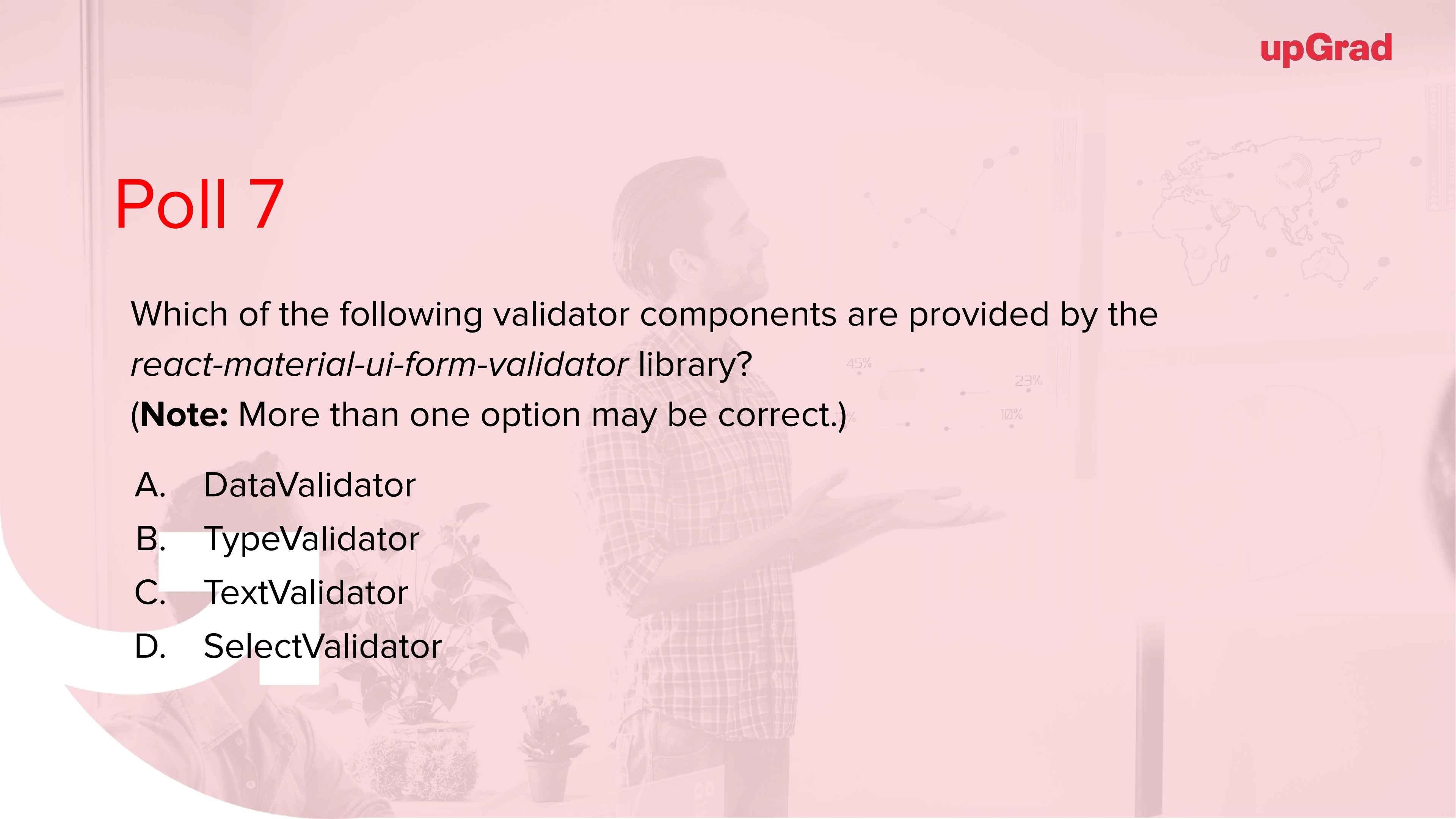
This code will download the material-ui core library as a dependency in the local project as well as add it to the project’s package.json

Material-UI

upGrad

Sample Code

```
import { Button, Input } from '@material-ui/core';

function LoginForm() {
  const onChange = () => { ... }     // event handler for user input in form
  const handleSubmit = () => { ... }  // event handler for form submit

  return (
    <form noValidate>
      <Input type="text" label="Email" onChange={onChange} />
      <Input type="password" label="Password" onChange={onChange} />
      <Button onClick={handleSubmit} color="primary">Submit</Button>
    </form>
  );
}
```

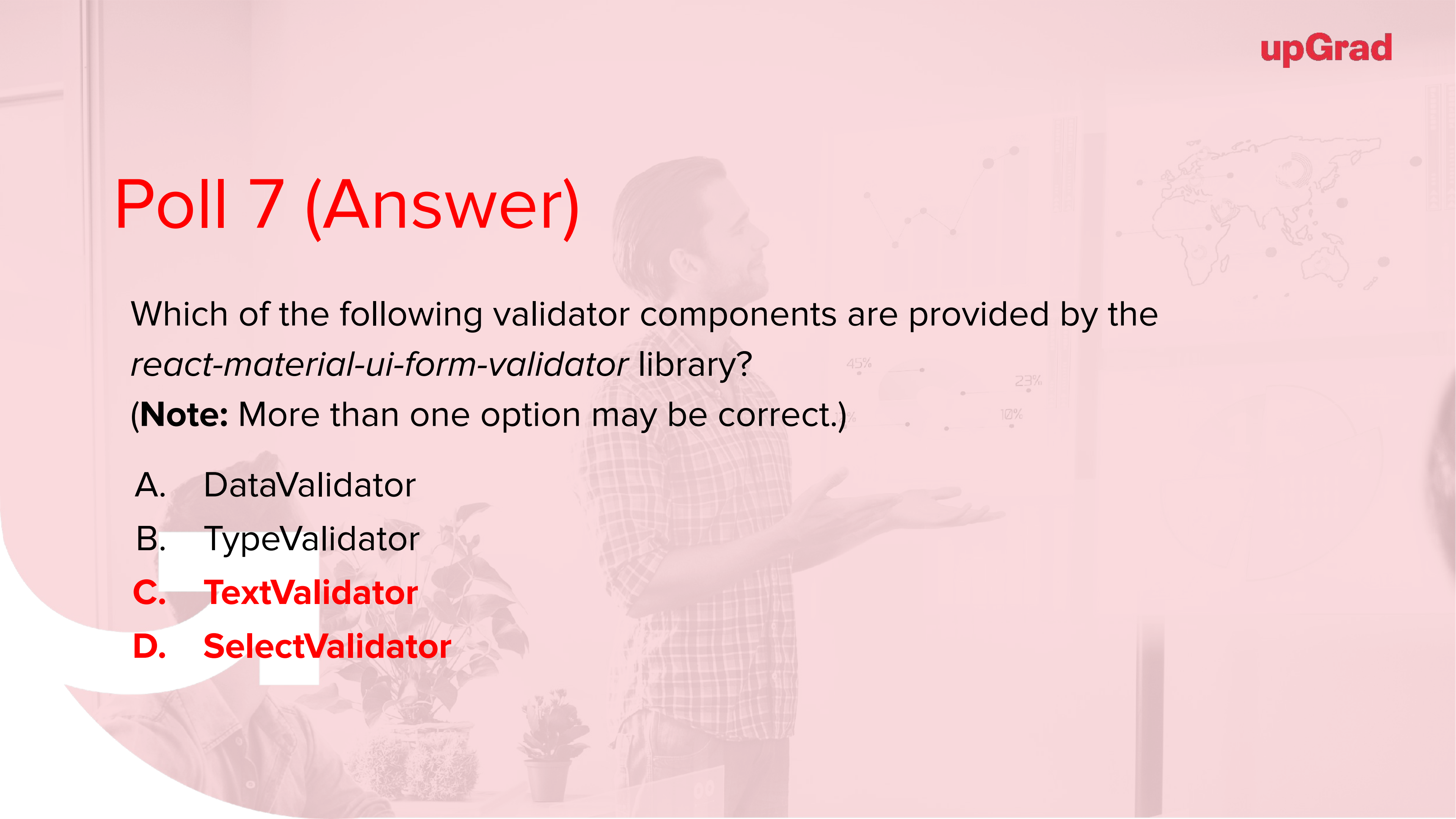


upGrad

Poll 7

Which of the following validator components are provided by the *react-material-ui-form-validator* library?
(**Note:** More than one option may be correct.)

A. DataValidator
B. TypeValidator
C. TextValidator
D. SelectValidator



Poll 7 (Answer)

Which of the following validator components are provided by the *react-material-ui-form-validator* library?
(**Note:** More than one option may be correct.)

A. DataValidator
B. TypeValidator
C. TextValidator
D. SelectValidator

upGrad

Poll 8

Which of the following is not a commonly used valid prop for validator components offered by the react-material-ui-form-validator library?

A. label

B. onValidate

C. validators

D. errorMessages

Poll 8 (Answer)

Which of the following is not a commonly used valid prop for validator components offered by the react-material-ui-form-validator library?

A. label

B. onValidate

C. validators

D. errorMessages

All the code used in today’s session can be found in the link provided below:

https://github.com/upgrad-edu/react-hooks/tree/Session9

upGrad

Doubt Clearance (5 minutes)

upGrad

Important Questions

Important Questions

upGrad

1. What are forms in React?
2. How do you create forms in React?
3. What are Pure Components?
4. What are the higher-order components in React?
5. What is the difference between controlled & uncontrolled components in React?

Key Takeaways

- Forms in HTML handle data in the DOM, whereas the React forms manage the data within the internal state of the form component itself
- Such components are called 'controlled components'
- Material-UI provides a component library for a common set of out-of-the-box but customisable components can be used to build forms and other visual components
- Material-UI Form Validator library can be used to plug in data validation into the forms created using Material-UI

These tasks are to be completed after today's session:

MCQs
Coding Questions
Course Project (Part B) - Checkpoint 3

In the next class, we will discuss

- Backend integration using APIs
  - Introduction to REST APIs
  - Session and Authentication
  - Using Fetch

upGrad
#RahoAmbitious

Thank You!