

# Deep Dive into React Portals

Souvik Basu



What is React Portal

Header

Login

showLogin

Email

Password

Login

Cancel

The diagram illustrates a login process within a green header area. A 'Login' button is located in the top right. An arrow points from this button to the text 'showLogin'. A second arrow points from 'showLogin' to a login form. The form is a light gray rectangle containing two input fields labeled 'Email' and 'Password', and two buttons labeled 'Login' and 'Cancel'.

App

Header

Login

showLogin

showLogin

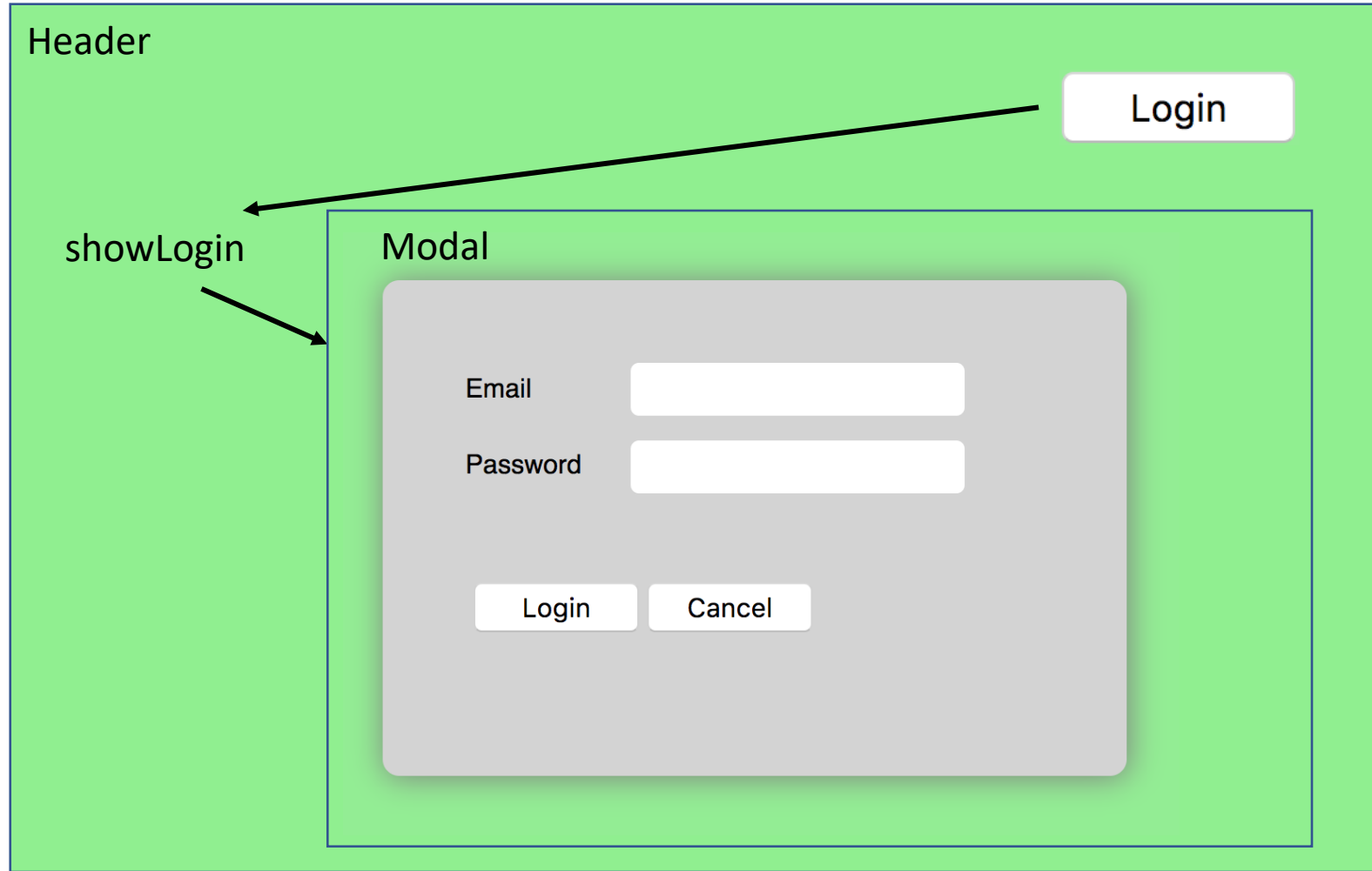
Main

Email

Password

Login

Cancel



Header

Hi a@b.c

Login

email

showLogin

Modal

a@b.c

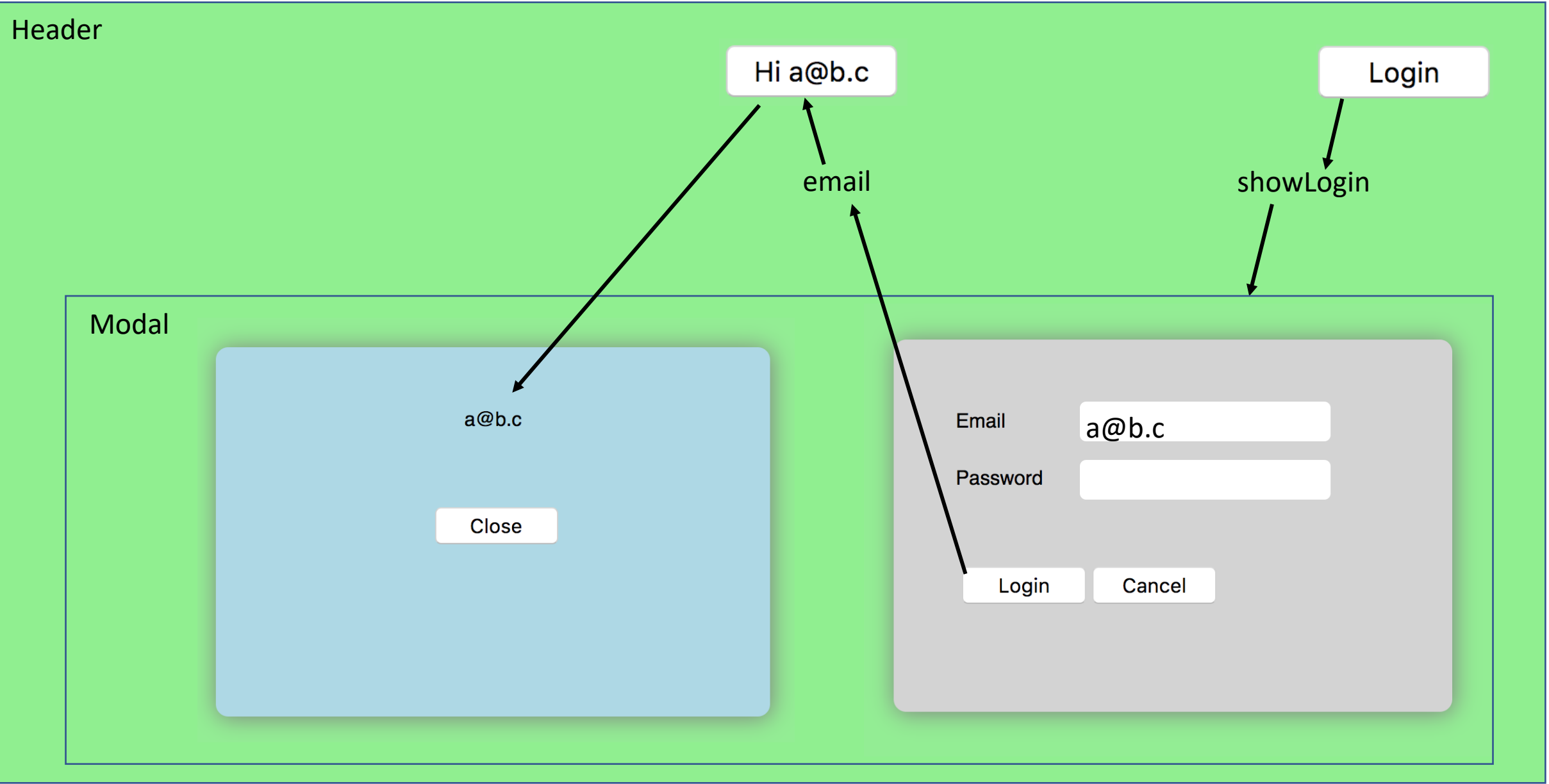
Close

Email

Password

Login

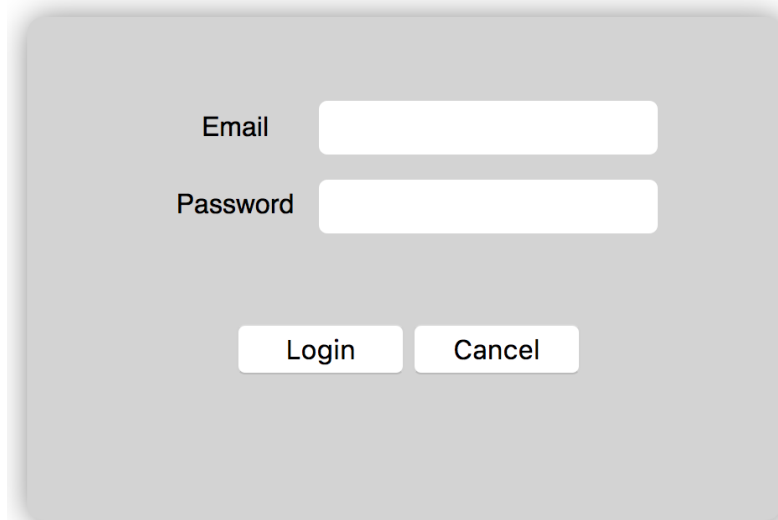
Cancel



# When to use Portals?

- Always on Top
- Child bigger than Parent

# Lightbox

A lightbox dialog box with a light gray background and rounded corners. It contains two input fields for 'Email' and 'Password', and two buttons labeled 'Login' and 'Cancel' at the bottom.

Email

Password

Login Cancel



# Children that spill out of parent div

Login

Email

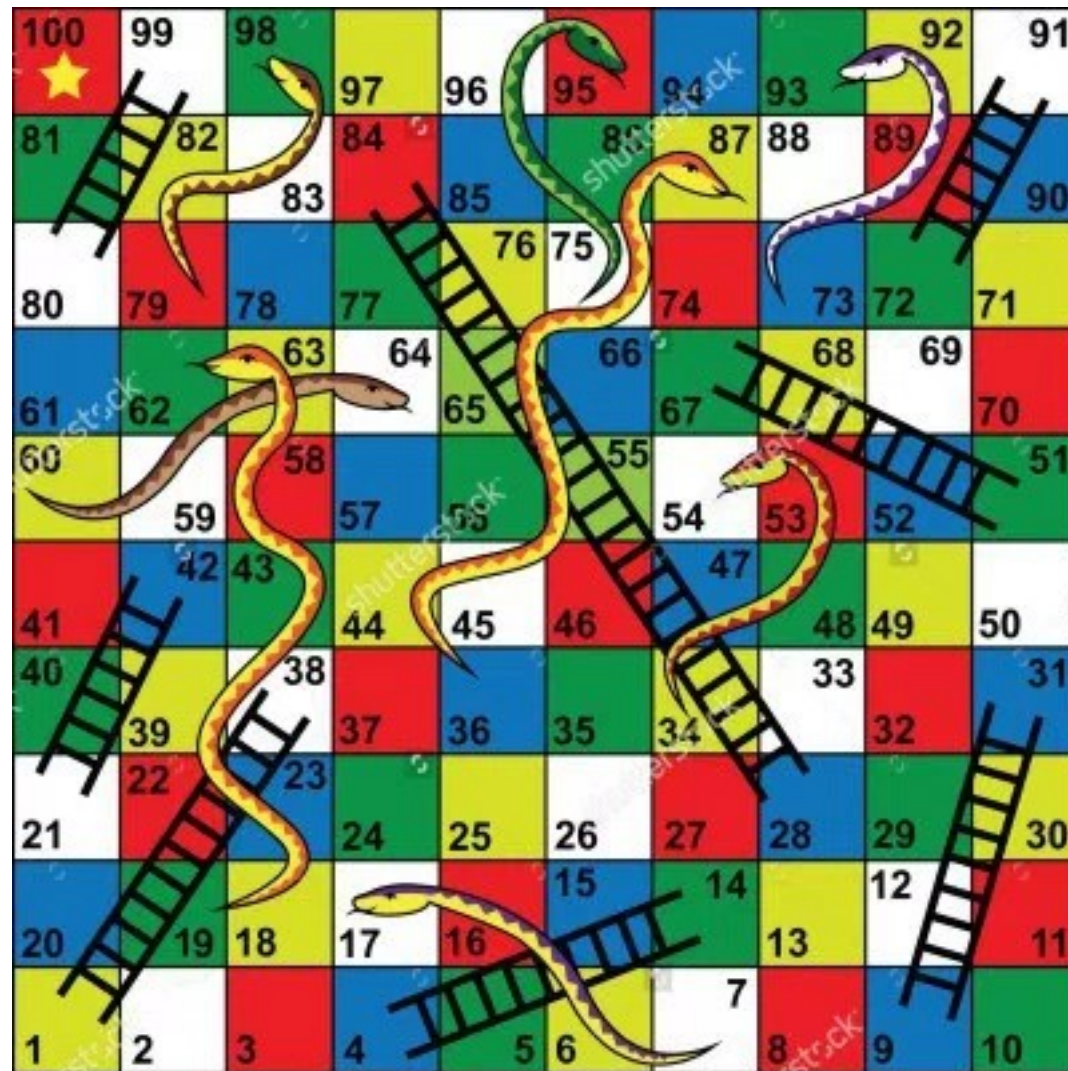
# Why to use Portals?

- Handles code complexity (easier state management)

# Concerns around usage of React Portal

- Spaghetti code
- Portals are like goto statement

goto

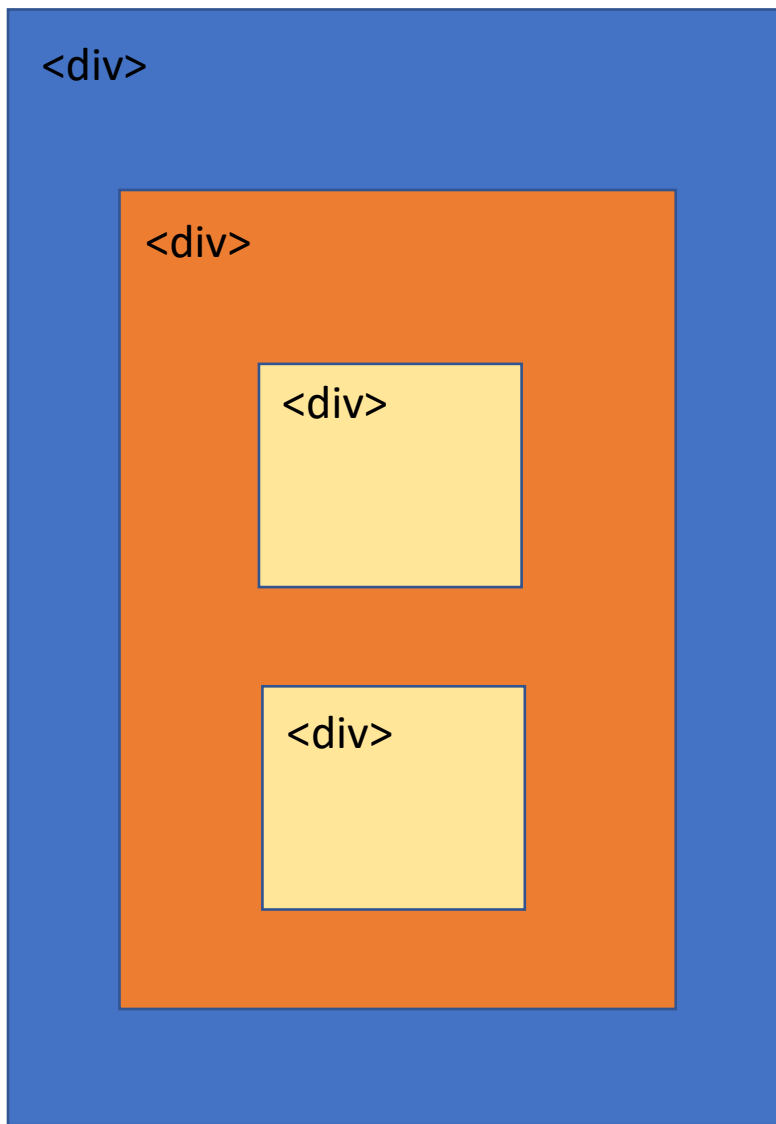


# Portal inside Portal inside Portal

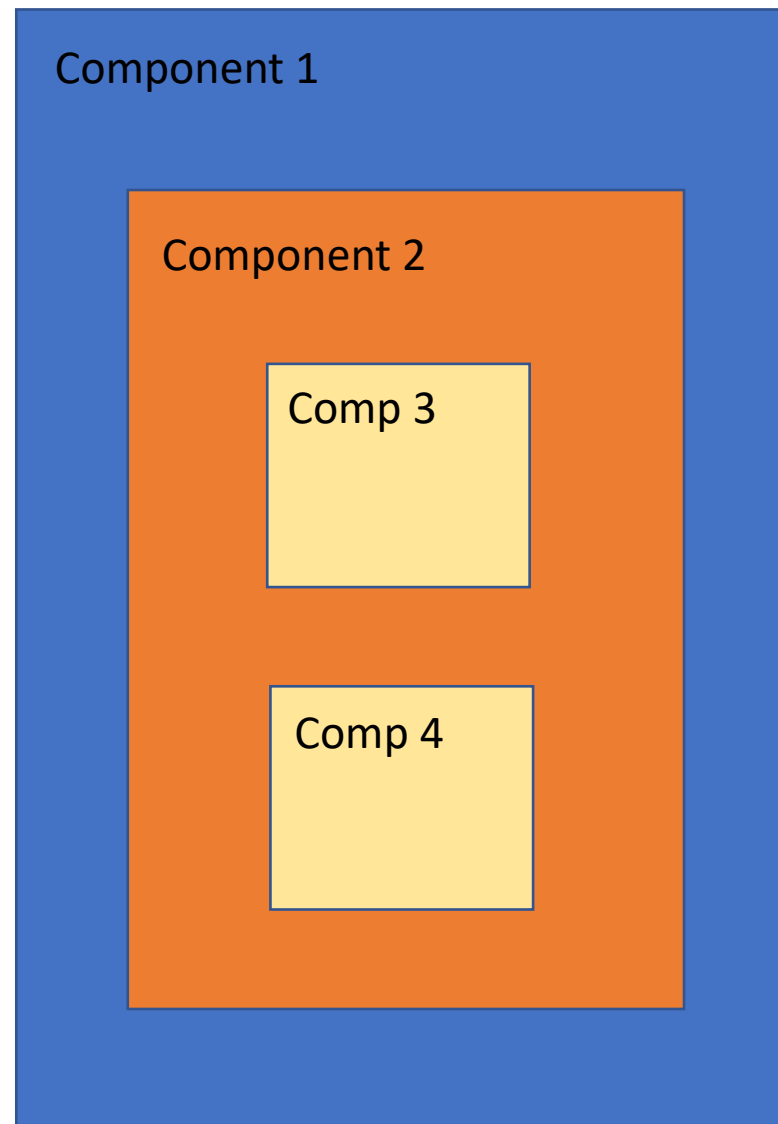


# Clean design using Portals

- Virtual DOM abstraction
- Portal enforces to think in terms of React Elements and not DOM



DOM



JSX

Email

Password

Login

Cancel

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div id="root">
      <div style="background-color: rgba(0, 0, 0, 0.1); border-radius: 10px; padding: 50px; height: 200px; width: 350px; margin-top: 100px; margin-left: auto; margin-right: auto;"> == $0
        <div>
          <label style="height: 40px; width: 100px; display: inline-block;">Email</label>
          <input style="height: 30px; width: 200px; display: inline-block; border-radius: 5px; border: none;">
        </div>
      <div>...</div>
      <div>...</div>
    </div>
    <script type="text/javascript" src="/static/js/bundle.js"></script>
  </body>
</html>
```

```
▼<App>
  ▼<Login> == $r
    ▼<div style={backgroundColor: "rgba(0,0,0,0.1)", borderRadius: 10, padding: 50, ...}>
      ▼<div>
        <label style={height: 40, width: 100, display: "inline-block"}>Email</label>
        <input style={height: 30, width: 200, display: "inline-block", ...}></input>
      </div>
      ▼<div>
        <label style={height: 40, width: 100, display: "inline-block"}>Password</label>
        <input style={height: 30, width: 200, display: "inline-block", ...}></input>
      </div>
      ▼<div>
        <button style={height: 30, width: 100, display: "inline-block", ...}>Login</button>
        <button style={height: 30, width: 100, display: "inline-block", ...}>Cancel</button>
      </div>
    </div>
  </Login>
</App>
```



Login

Email

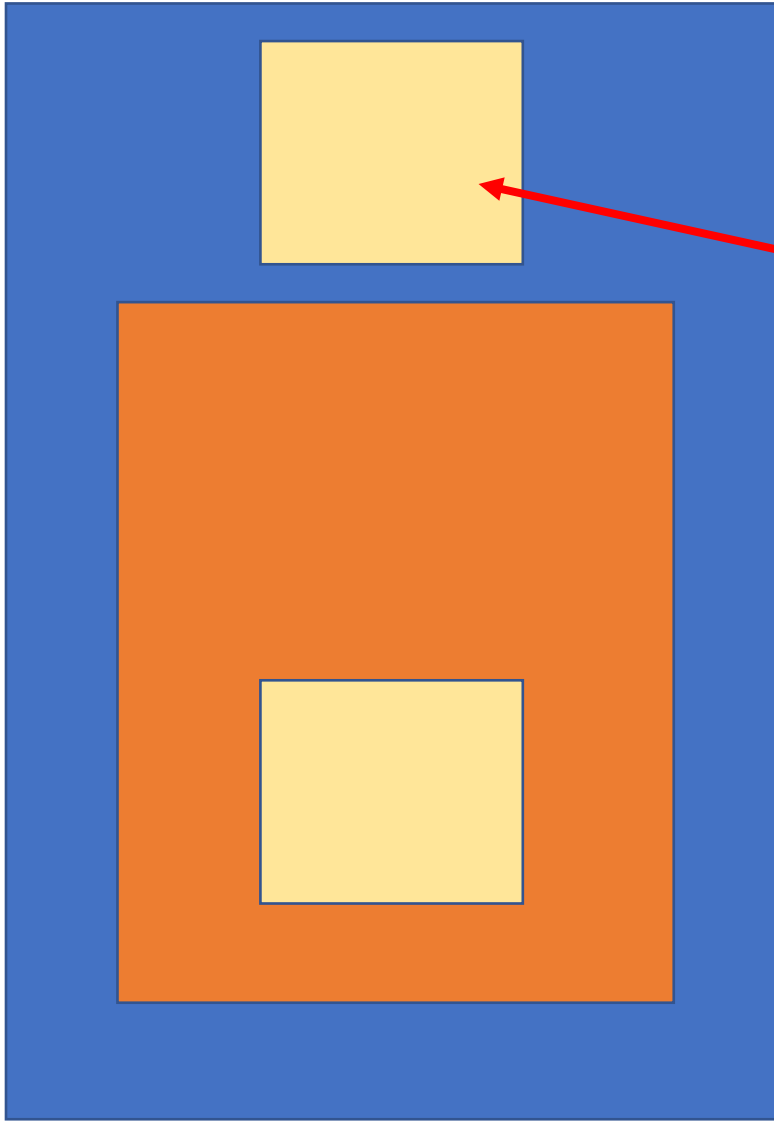
Password

Login

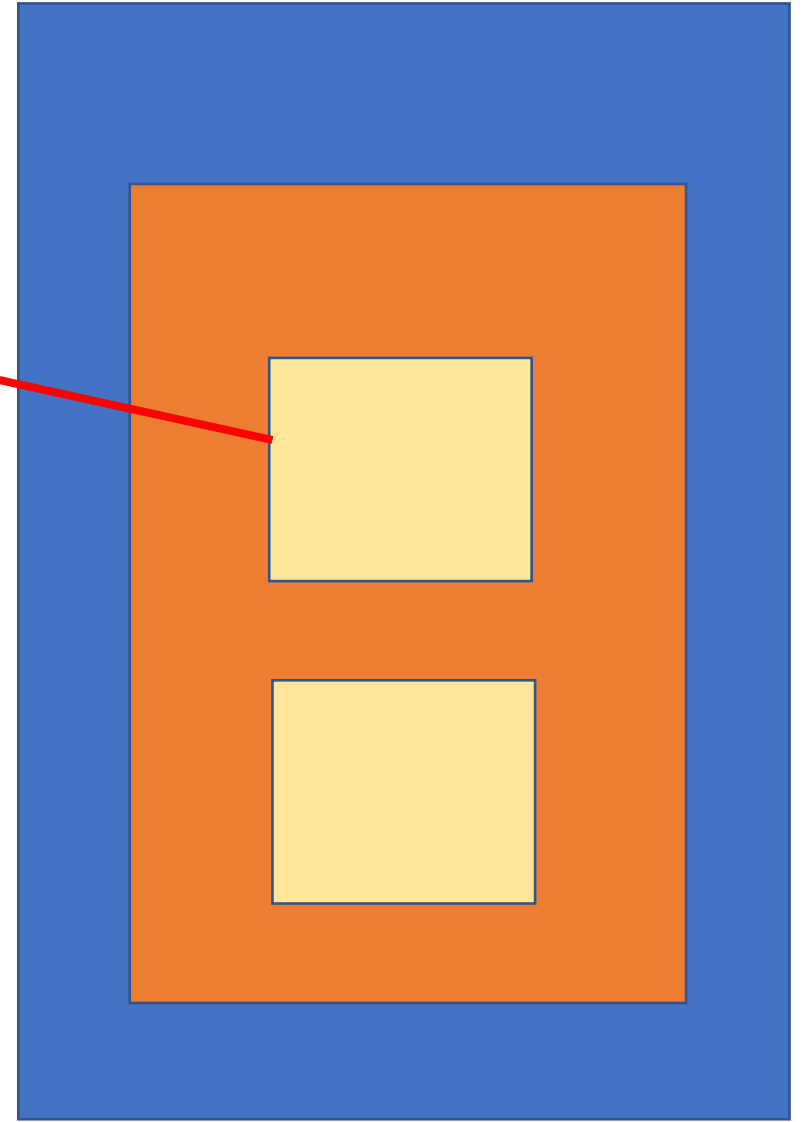
Cancel

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div id="root">
      <div>
        <div>
          <button>Login</button>
        </div>
        <div>This is the Awesome Home page</div>
      </div>
      <div id="modal"> == $0
        <div>
          <div>
            <div>
              <label>Email</label>
              <input>
            </div>
            <div>
              <label>Password</label>
              <input>
            </div>
            <div>
              <button>Login</button>
              <button>Cancel</button>
            </div>
          </div>
        </div>
      </div>
      <div>
        <script type="text/javascript" src="/static/js/bundle.js"></script>
      </div>
    </body>
  </html>
```

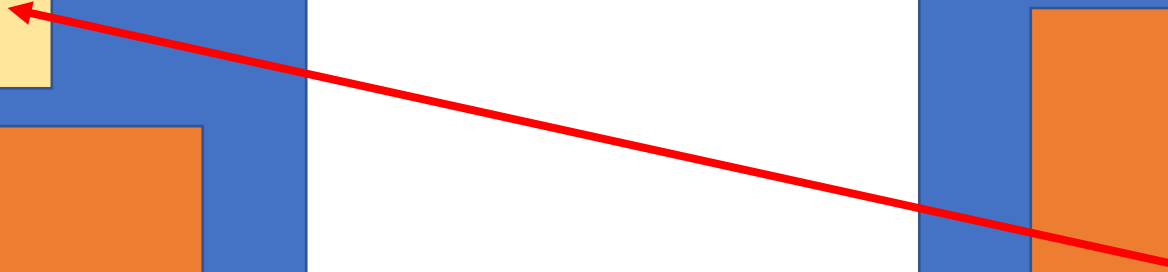
```
▼ <App>
  ▼ <div>
    ▼ <Header>
      ▼ <div style={backgroundColor: "lightgreen", padding: 10, height: 180, ...}>
        <button style={height: 30, width: 100, display: "inline-block", ...} onClick=bound showLogin()>Login</button>
        ▼ <Modal show=true>
          ▼ <ReactPortal target=HTMLDivElement{...}>
            ▼ <div style={background: "transparent", position: "absolute", top: 200, ...}>
              ▶ <Login onCancel=bound cancelLogin()>...</Login> == $r
            </div>
          </ReactPortal>
        </Modal>
      </div>
    </Header>
    ▶ <Main>...</Main>
    <div id="modal"></div>
  </div>
</App>
```



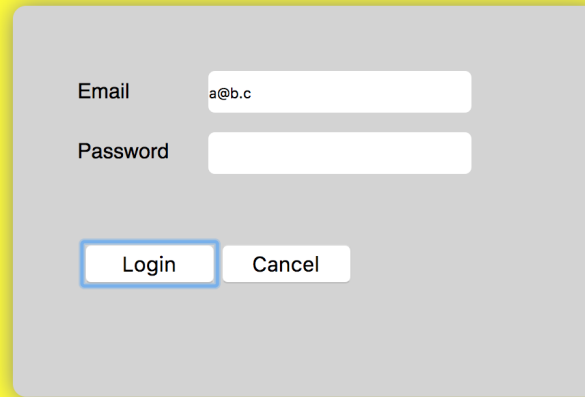
DOM



JSX

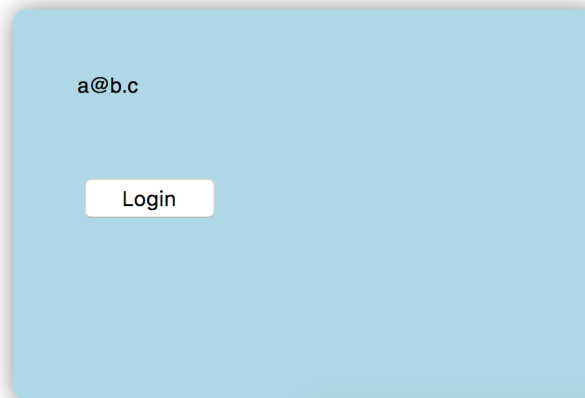


This is plain HTML page



Email

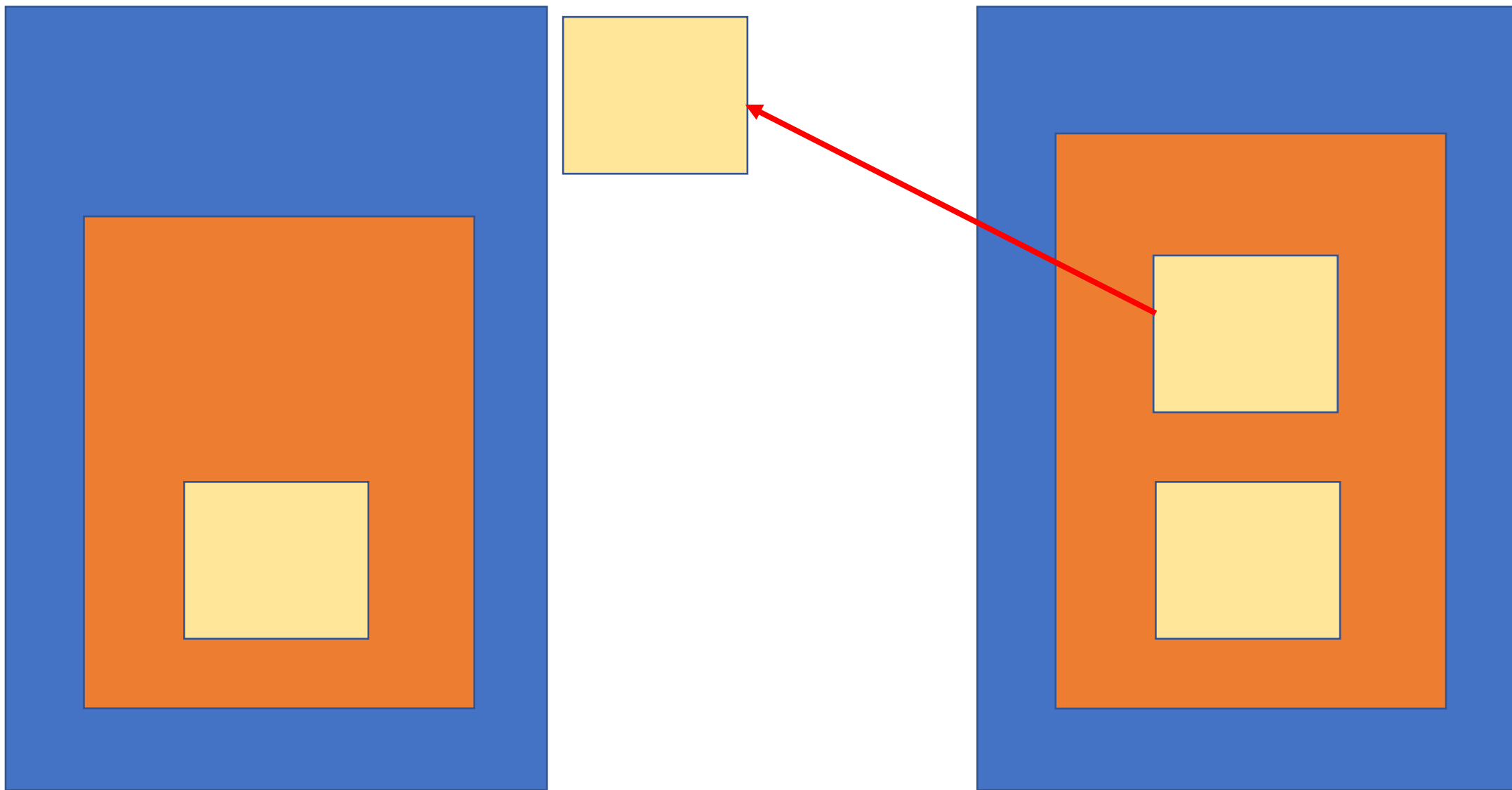
Password



a@b.c

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div id="plain_html">
      "
      This is plain HTML page
      "
    <div id="login">
      <div>
        <div>
          <label>Email</label>
          <input>
        </div>
        <div>
          <label>Password</label>
          <input>
        </div>
        <div>
          <button>Login</button>
          <button>Cancel</button>
        </div>
      </div>
    </div>
  </div>
  <div id="profile"> == $0
    <div>
      <div>
        <label>a@b.c</label>
      </div>
      <div>
        <button>Login</button>
      </div>
    </div>
    <script type="text/javascript" src="/static/js/bundle.js"></script>
  </body>
</html>
```

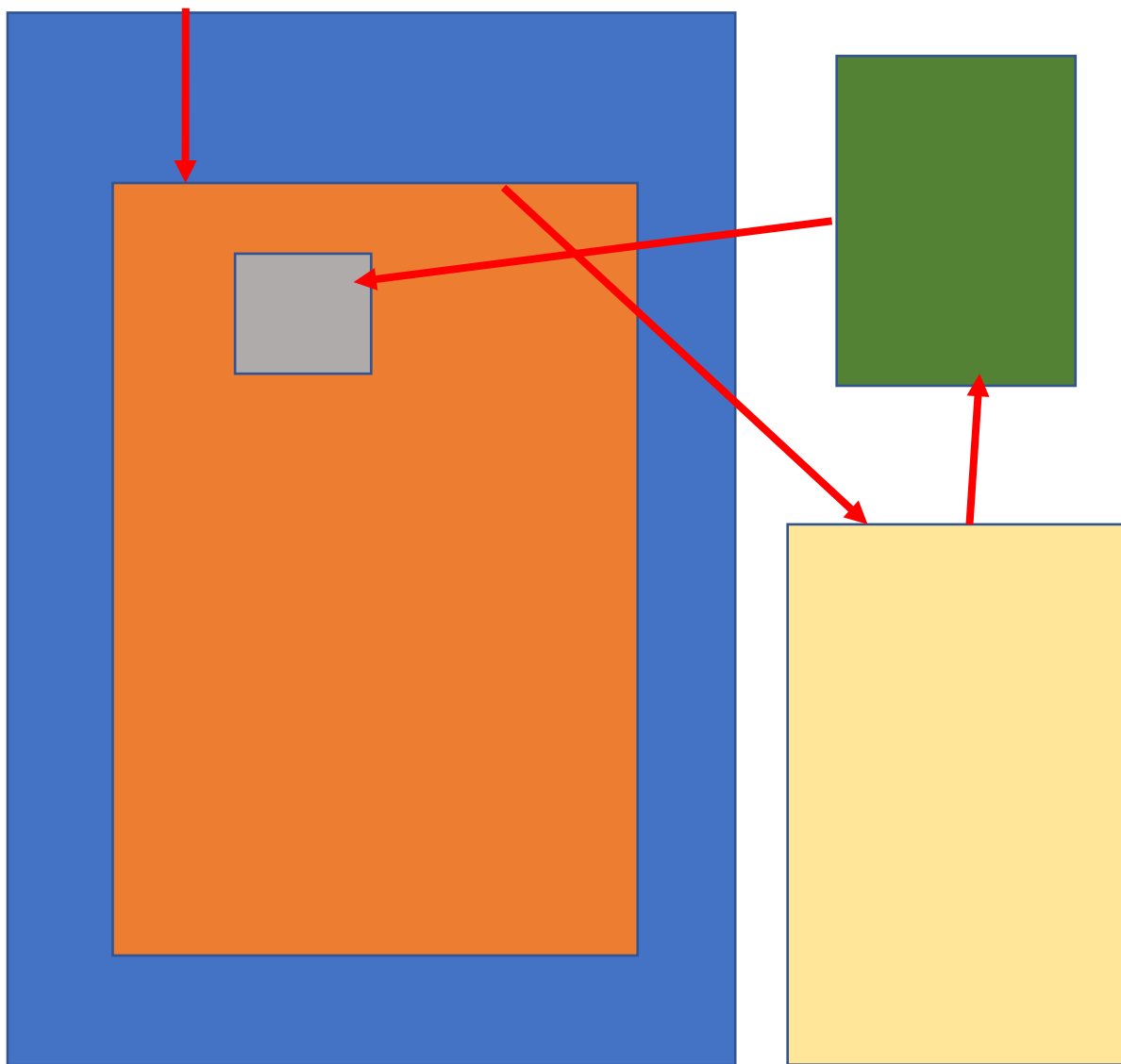
```
▼ <Profile show="true">
  ▼ <div style={backgroundColor: "lightblue", borderRadius: 10, padding: 50, ...}>
    ▼ <div>
      <label style={height: 40, width: 100, display: "inline-block"}>a@b.c</label>
    </div>
    ▼ <div>
      <button style={height: 30, width: 100, display: "inline-block", ...} onClick=bound handleLogin()>Login</button>
    </div>
    ▼ <ReactPortal target=HTMLDivElement{...}>
      ▼ <Login show=true onLogin=bound setEmail()> == $r
        ▼ <div style={backgroundColor: "lightgray", borderRadius: 10, padding: 50, ...}>
          ▼ <div>
            <label style={height: 40, width: 100, display: "inline-block"}>Email</label>
            <input style={height: 30, width: 200, display: "inline-block", ...} onChange=bound onEmailChange()></input>
          </div>
          ▼ <div>
            <label style={height: 40, width: 100, display: "inline-block"}>Password</label>
            <input style={height: 30, width: 200, display: "inline-block", ...}></input>
          </div>
          ▼ <div>
            <button style={height: 30, width: 100, display: "inline-block", ...} onClick=bound onLogin()>Login</button>
            <button style={height: 30, width: 100, display: "inline-block", ...}>Cancel</button>
          </div>
        </div>
      </Login>
    </ReactPortal>
  </div>
</div>
</Profile>
```



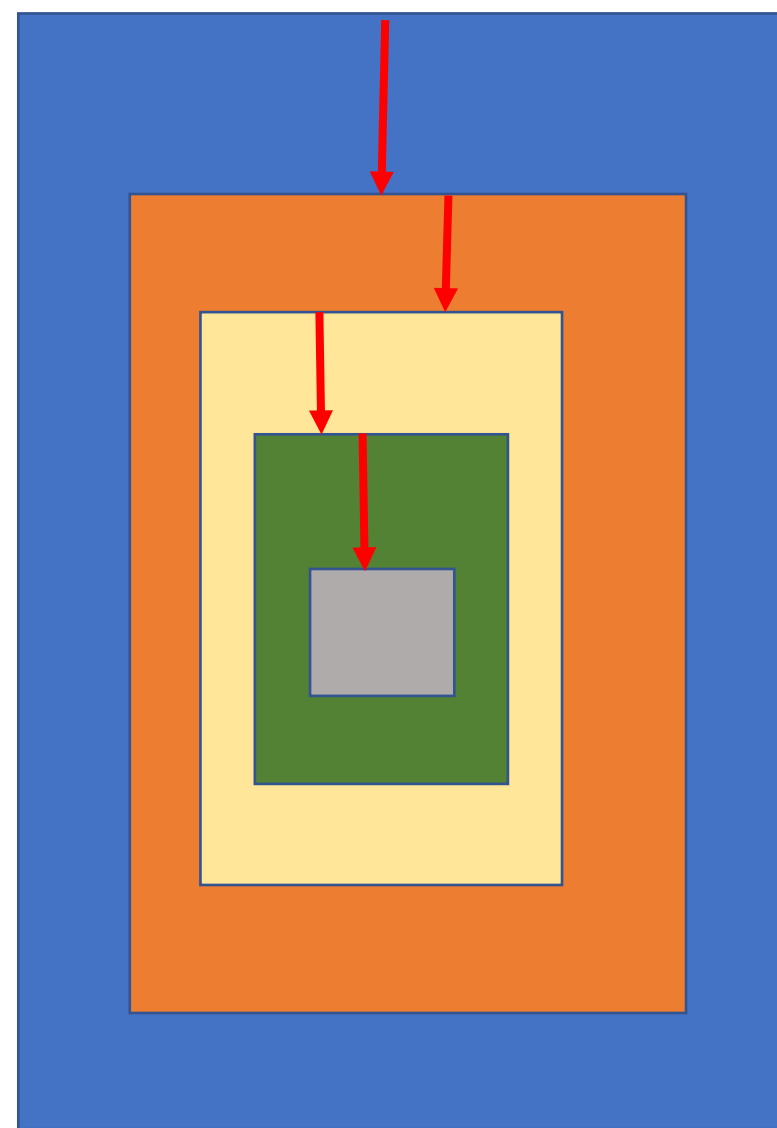
DOM

JSX





DOM



JSX

# Testing React Portals

# Testing using Enzyme

- mount -> DOM
- shallow -> Component

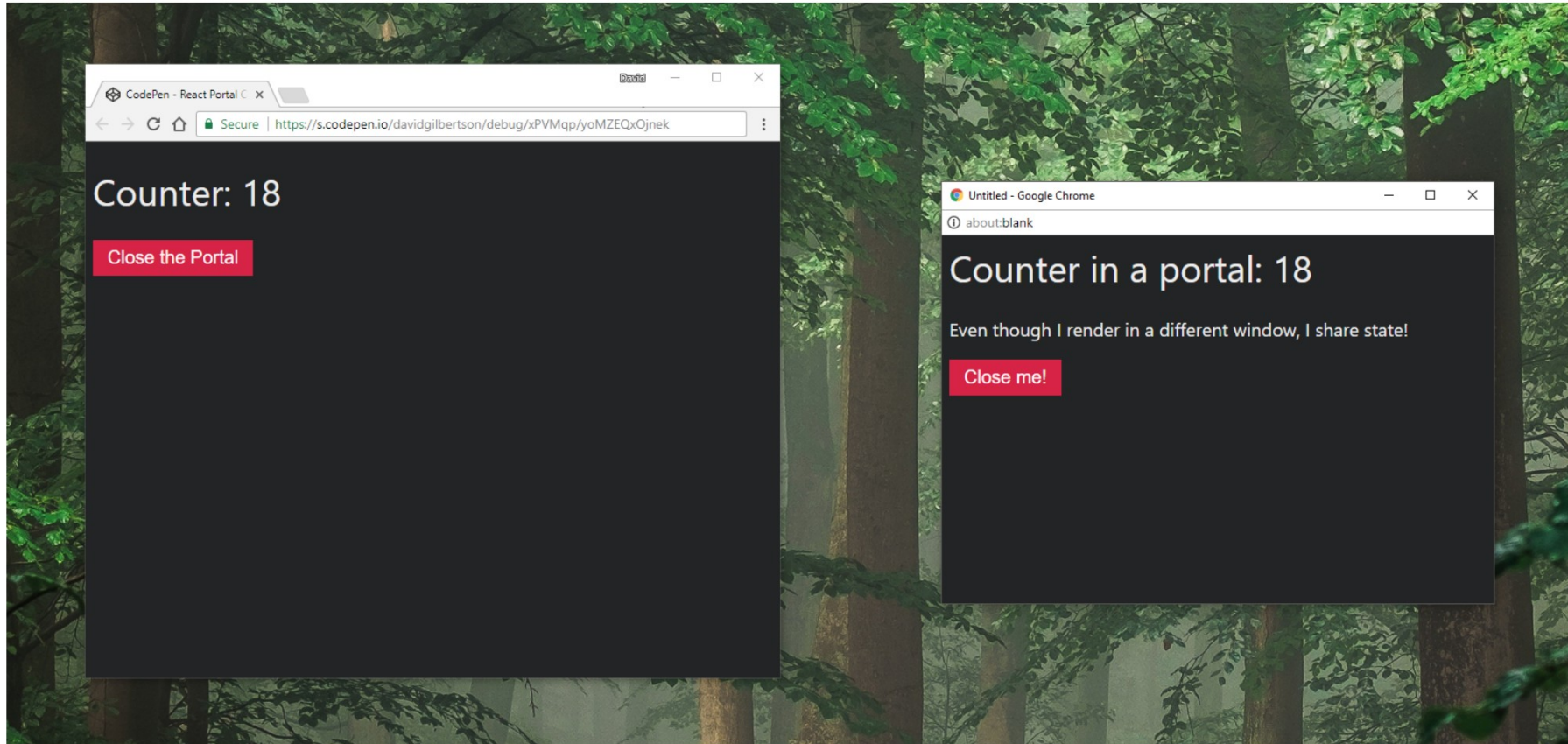
# State Management in Portals

# Gotchas with Portals

- Avoid State flow as per UI designs
- Rather think in terms of Components
- Portals is part of ReactDOM and not React

# Some cool stuff

- <https://hackernoon.com/using-a-react-16-portal-to-do-something-cool-2a2d627b0202>



```
class MyWindowPortal extends React.PureComponent {
  constructor(props) {
    super(props);
    // STEP 1: create a container <div>
    this.containerEl = document.createElement('div');
    this.externalWindow = null;
  }

  render() {
    // STEP 2: append props.children to the container <div> that isn't mounted anywhere yet
    return ReactDOM.createPortal(this.props.children, this.containerEl);
  }

  componentDidMount() {
    // STEP 3: open a new browser window and store a reference to it
    this.externalWindow = window.open('', '', 'width=600,height=400,left=200,top=200');

    // STEP 4: append the container <div> (that has props.children appended to it) to the body
    this.externalWindow.document.body.appendChild(this.containerEl);
  }

  componentWillUnmount() {
    // STEP 5: This will fire when this.state.showWindowPortal in the parent component becomes false
    // So we tidy up by closing the window
    this.externalWindow.close();
  }
}
```

Q n A

Thank you  
@souvikbasu

Code: [bit.ly/reactportals](https://bit.ly/reactportals)