ACADE MIND

Fragments, Portals & Refs

More Tools for your Toolbox!

ACADE MIND

Module Content

JSX Limitations & Fragments

Getting a Cleaner DOM with Portals

Working with Refs

JSX Limitations

```
return (
  <h2>Hi there!</h2>
  <p>This does not work :-(</p>
);
```

React App
localhost:3000
Incognito

Failed to compile

```
./src/App.js
  Line 43:7:  Parsing error: Adjacent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fragment <>...</>?

  41 |         <CourseInput onAddGoal={addGoalHandler} />
  42 |       </section>
> 43 |       <section id="goals">
     |       ^
  44 |         {content}
  45 |       </section>
  46 |
```

This error occurred during the build time and cannot be dismissed.

JSX Limitations

```
return (
  <h2>Hi there!</h2>
  <p>This does not work :-(</p>
);
```

You **can't return more than one "root" JSX element** (you also can't store more than one "root" JSX element in a variable).

ACADE MIND

JSX Limitations

```
return (
  <h2>Hi there!</h2>
  <p>This does not work :-(</p>
);
```

You **can't return more than one "root" JSX element** (you also can't store more than one "root" JSX element in a variable).

Because this also isn't valid JavaScript

```
return (
  React.createElement('h2', {}, 'Hi there!')
  React.createElement('p', {}, 'This does not work :-(')
);
```

The Solution: Always Wrap Adjacent Elements

```
return (
  <div>
    <h2>Hi there!</h2>
    <p>This does not work :-(</p>
  </div>
);
```

Important: Doesn't have to be a <div> - ANY element will do the trick.

A New Problem: "<div> Soup"

```
<div>
  <div>
    <div>
      <div>
        <h2>Some content - yeah, this can really happen.</h2>
      </div>
    </div>
  </div>
</div>
```

In bigger apps, you can easily end up with **tons of unnecessary <div>**s (or other elements) which add **no semantic meaning or structure** to the page but **are only there because of React's/ JSX' requirement**.

ACADE MIND

Introducing Fragments

```
return (
  <React.Fragment>
    <h2>Hi there!</h2>
    <p>This does not work :-(</p>
  </React.Fragment>
);
```

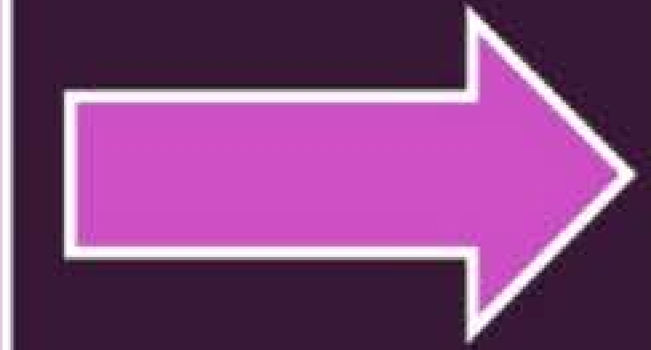
OR

```
return (
  <>
    <h2>Hi there!</h2>
    <p>This does not work :-(</p>
  </>
);
```

It's an **empty wrapper component**: It **doesn't render** any real HTML element to the DOM. But it **fulfills React's/ JSX' requirement**.

Understanding React Portals

```
return (
  <React.Fragment>
    <MyModal />
    <MyInputForm />
  </React.Fragment>
);
```



Real DOM

```
<section>
  <h2>Some other content … </h2>
  <div class="my-modal">
    <h2>A Modal Title!</h2>
  </div>
  <form>
    <label>Username</label>
    <input type="text" />
  </form>
</section>
```

Understanding React Portals

```
return (
  <React.Fragment>
    <MyModal />
    <MyInputForm />
  </React.Fragment>
);
```

Real DOM

```
<section>
  <h2>Some other content … </h2>
  <div class="my-modal">
    <h2>A Modal Title!</h2>
  </div>
  <form>
    <label>Username</label>
    <input type="text" />
  </form>
</section>
```

Semantically and from a "clean HTML structure" perspective, having this nested modal isn't ideal. It is an **overlay to the entire page** after all (that's similar for side-drawers, other dialogs etc.).

ACADE MIND

Understanding React Portals

It’s a bit like styling a <div> like a <button> and adding an event listener to it: It’ll work, but it’s not a good practice.

```
<div onClick={clickHandler}>Click me, I’m a bad button</div>
```

Understanding React Portals

```
return (
  <React.Fragment>
    <MyModal />
    <MyInputForm />
  </React.Fragment>
);
```

Real DOM

```
<section>
  <h2>Some other content … </h2>
  <div class="my-modal">
    <h2>A Modal Title!</h2>
  </div>
  <form>
    <label>Username</label>
    <input type="text" />
  </form>
</section>
```

Understanding React Portals

```
return (
  <React.Fragment>
    <MyModal />
    <MyInputForm />
  </React.Fragment>
);
```

Real DOM

```
<div class="my-modal">
  <h2>A Modal Title!</h2>
</div>
<section>
  <h2>Some other content … </h2>
  <form>
    <label>Username</label>
    <input type="text" />
  </form>
</section>
```