Why can't you return adjacent, top-level JSX elements?

It becomes clear, once you translate the JSX code to the `React.createElement()` calls React actually executes (don't forget: The React project build workflow performs this transformation for you!)

JSX:

```
import React from 'react';

const heading = props => (
     <h1>{props.title}</h1>
     <h2>{props.subtitle}</h2>
);

export default heading;
```

This is NOT allowed because it would be translated to:

```
import React from 'react';

const heading = props => React.createElement('h1', {},
props.title) React.createElement('h2', {}, props.subtitle);

export default heading;
```

This is invalid JavaScript syntax, you're trying to return two expressions (two `React.createElement()` calls).

You are allowed to do that if you

a) return an array of `React.createElement()` calls OR
b) return a single `React.createElement()` call that wraps the other two

a)

```
import React from 'react';

const heading = props => [
     React.createElement('h1', {key: 'i1'}, props.title),
     React.createElement('h2', {key: 'i2'}, props.subtitle)
];

export default heading;
```

This is equivalent to returning an array of keyed JSX elements.

b)

```
import React from 'react';

import Aux from '../hoc/Aux';

const heading = props => React.createElement(
     Aux,
     {},
     React.createElement('h1', {key: 'i1'}, props.title),
     React.createElement('h2', {key: 'i2'}, props.subtitle)
);

export default heading;
```

This is equivalent to using `<Aux>`.

b) works because we can pass as many children (third argument to `React.createElement()`) as we want.