Creating Content With JSX

React $\rightarrow$ Library that defines what a component is and how multiple components work together

ReactDOM $\rightarrow$ Library that knows how to get a component to show up in the browser what we write

```
<h1>Hi there!</h1>
```

Babel

This is what runs in the browser

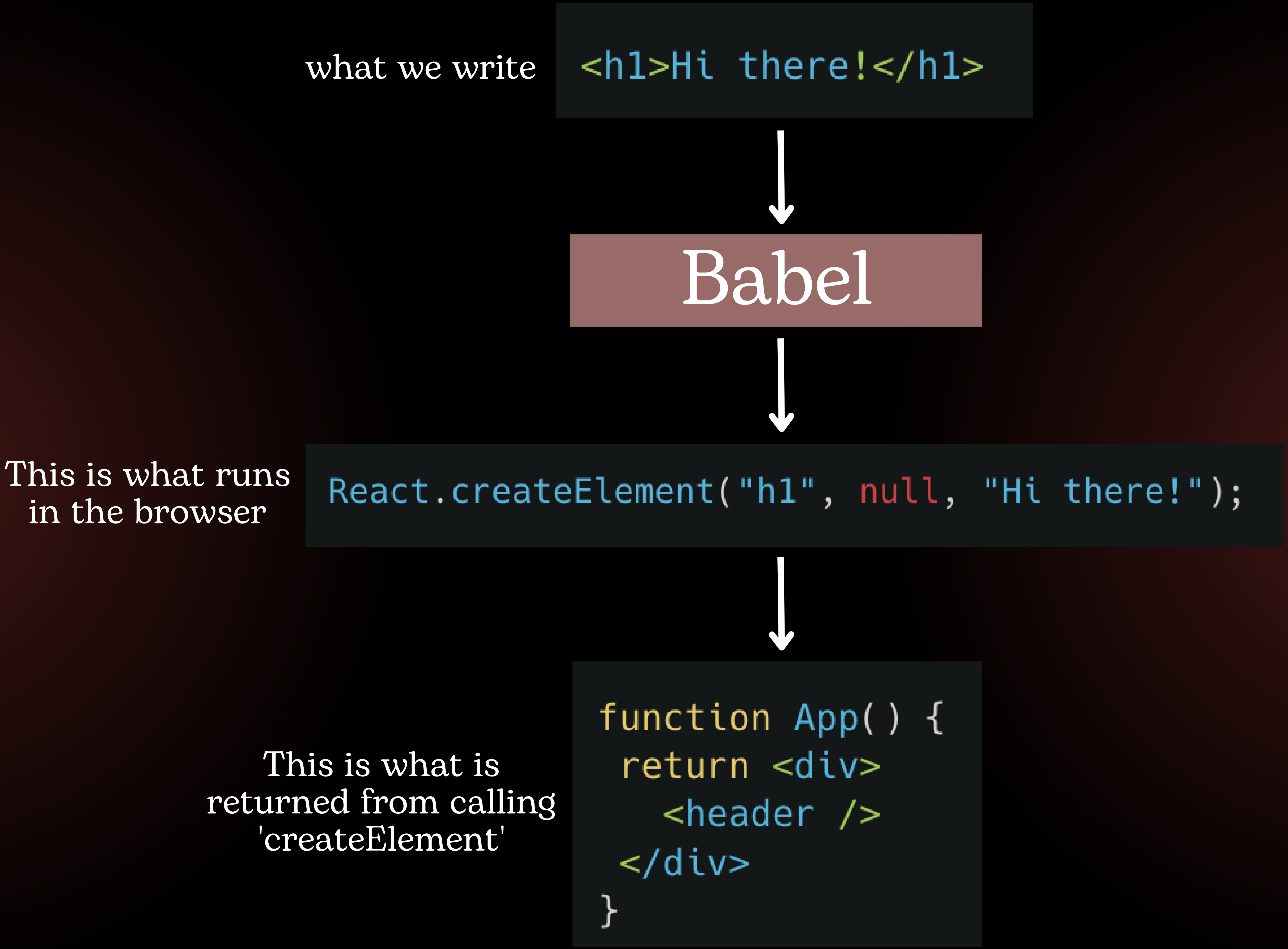
```
React.createElement("h1", null, "Hi there!");
```

This is what is returned from calling 'createElement'

```
{
  $$typeof: Symbol(react.element),
  key: null,
  props: { children: 'Hi there!' },
  ref: null,
  type: 'h1'
}
```

babeljs.io/repl

Tool to show you what your JSX
is turn into


what we write
Babel
This is what runs in the browser
This is what is returned from calling 'createElement'

```
<h1>Hi there!</h1>
```

```
React.createElement("h1", null, "Hi there!");
```

```
function App() {
 return <div>
   <header />
 </div>
}
```

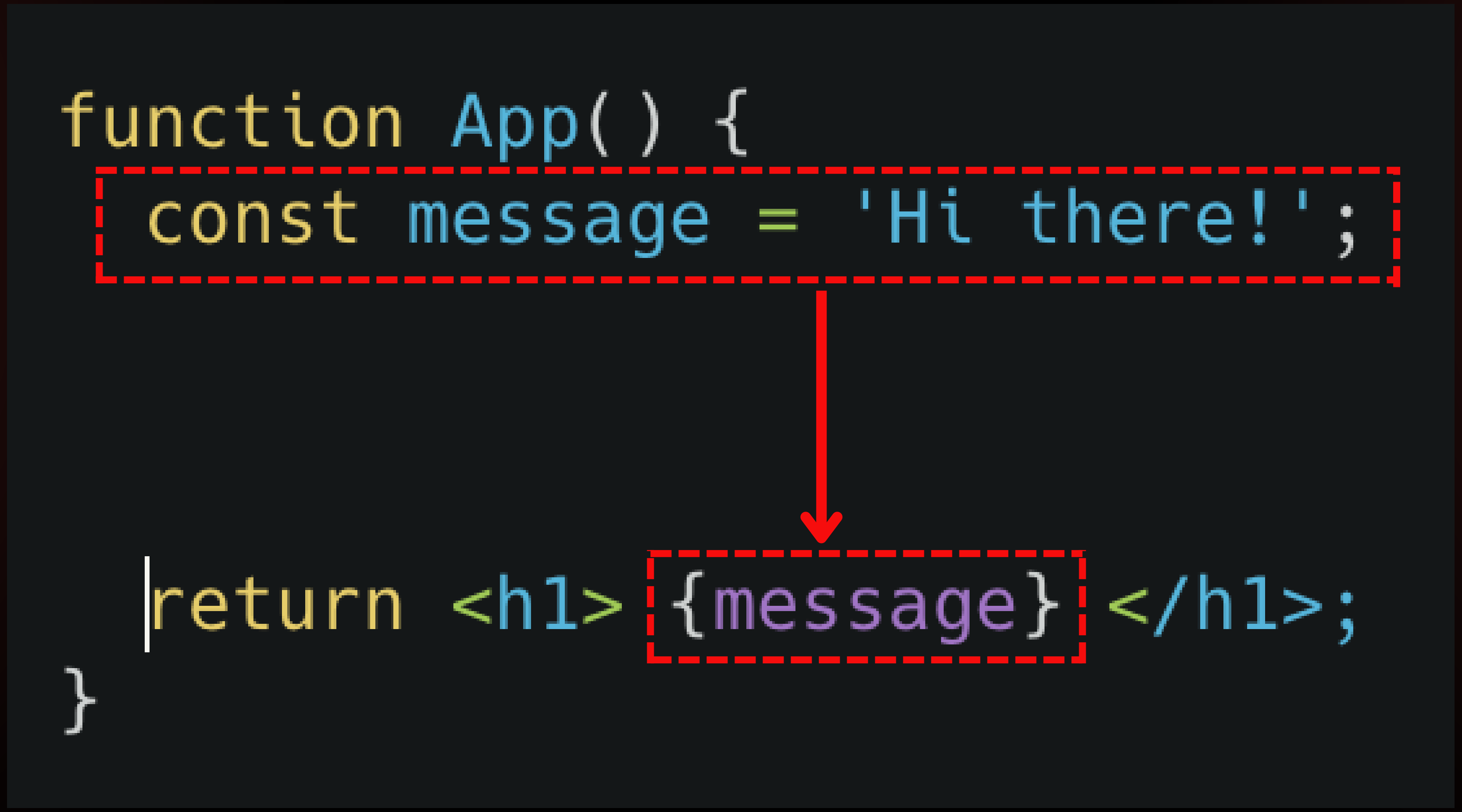
```
<h1>Hi there!</h1>
```

Writing this doesn't make anything show up in the browser automatically

This creates an **instruction** for React, telling it to make an element

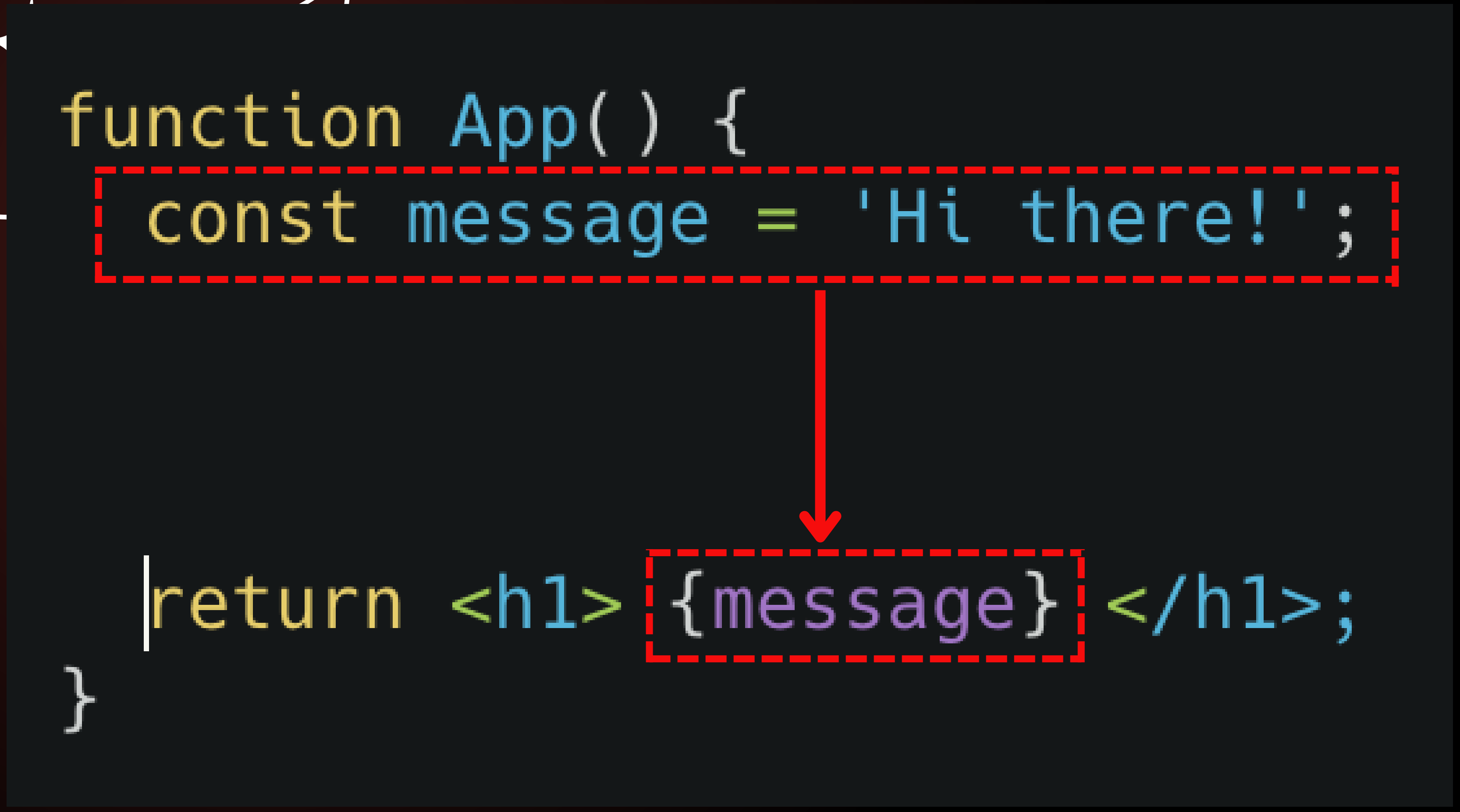
We have to **return** it from a component for React to use it

Curly braces mean we are about to reference a JS variable or expression

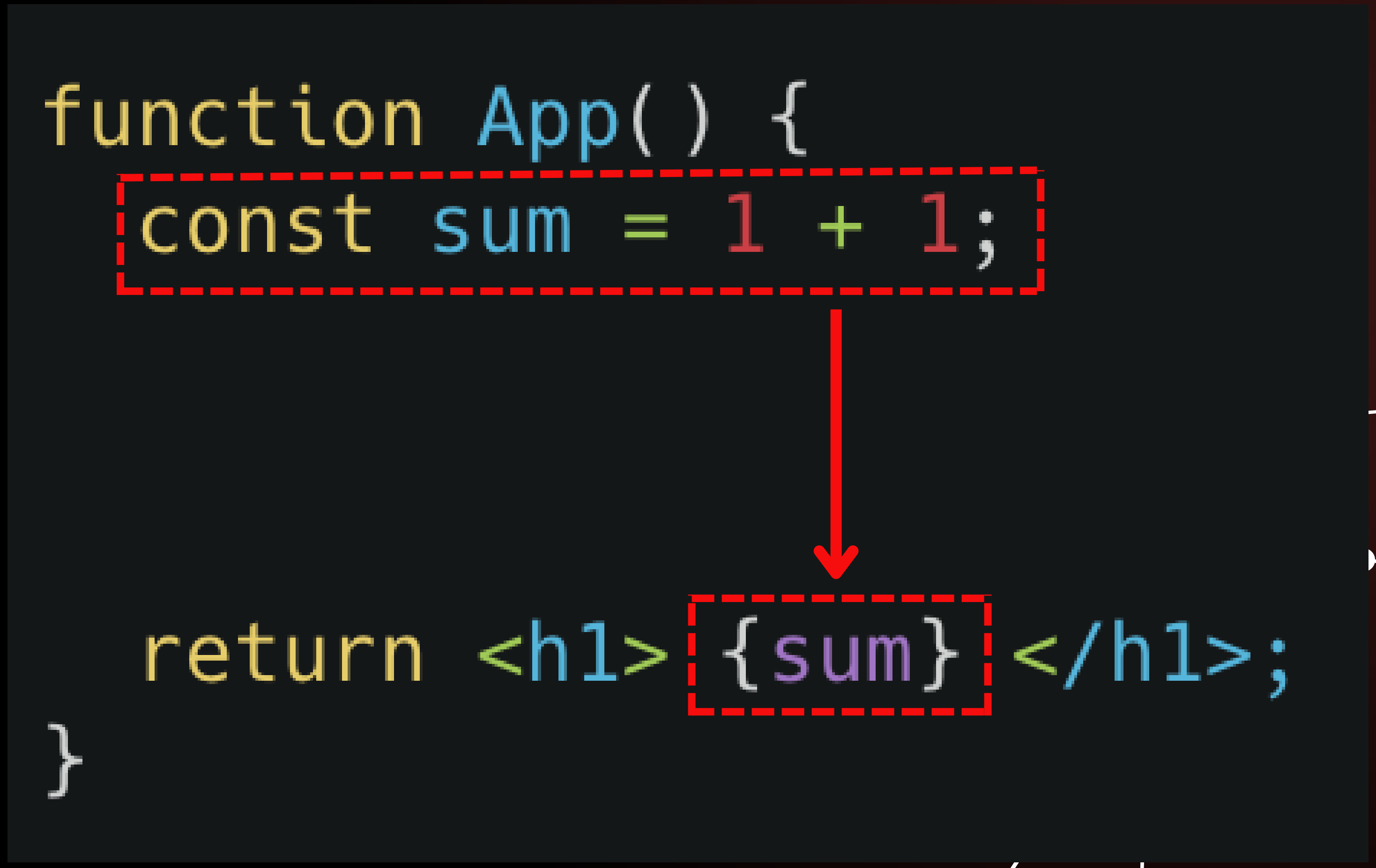


```
function App() {
  const message = 'Hi there!';



  return <h1> {message} </h1>;
}
```

We most often curly braces to show strings or number

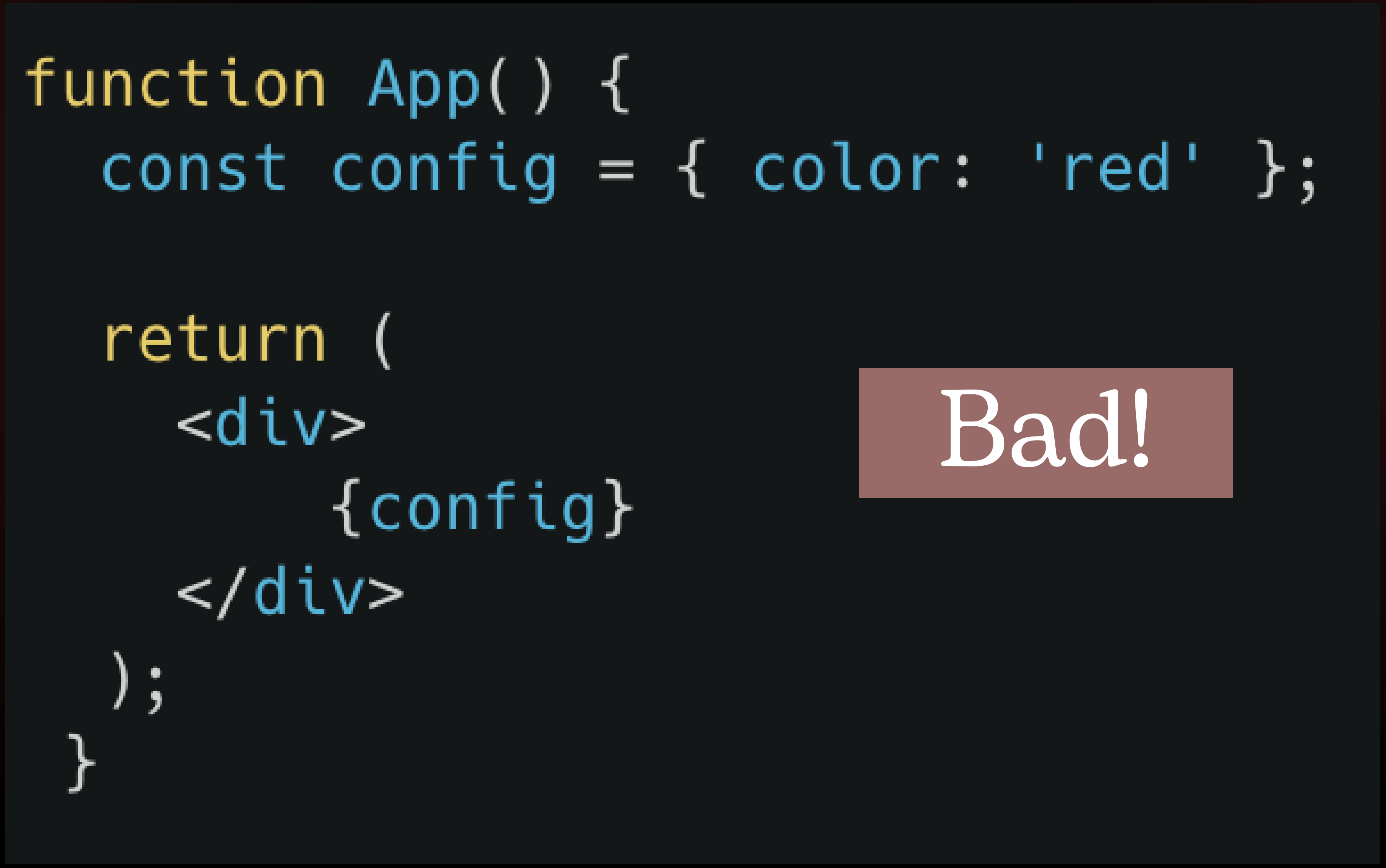


```
function App() {
  const message = 'Hi there!';


  return <h1> {message} </h1>;
}
```



```
function App() {
  const sum = 1 + 1;


  return <h1> {sum} </h1>;
}
```

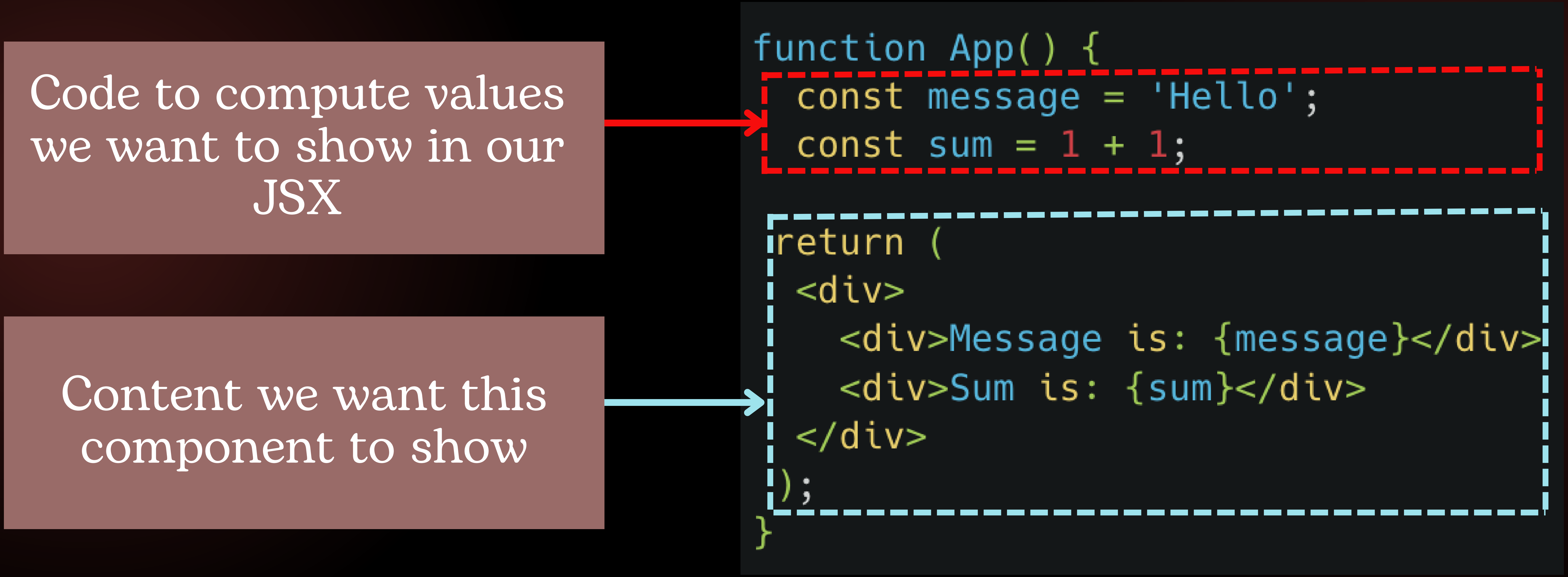
Common Error:
React cannot show an object as text content



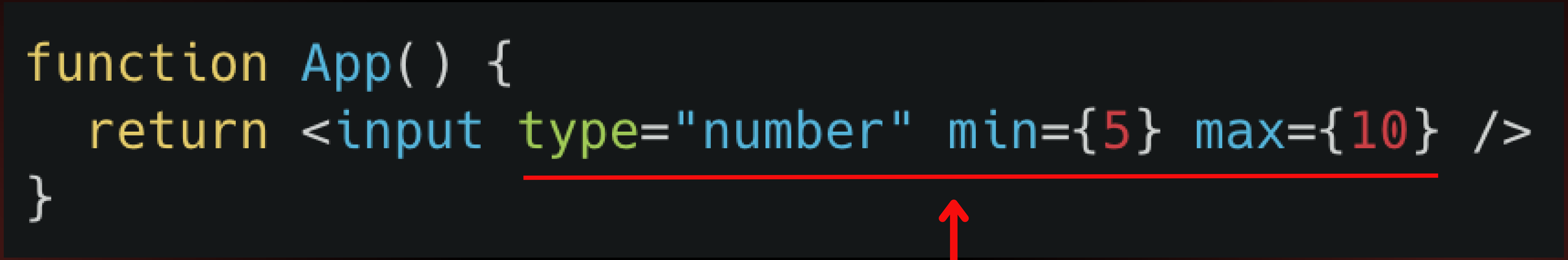
```
function App() {
  const config = { color: 'red' };

  return (
    <div>
        {config}
    </div>
  );
}
```

Bad!

Component Layout


Code to compute values we want to show in our JSX
Content we want this component to show

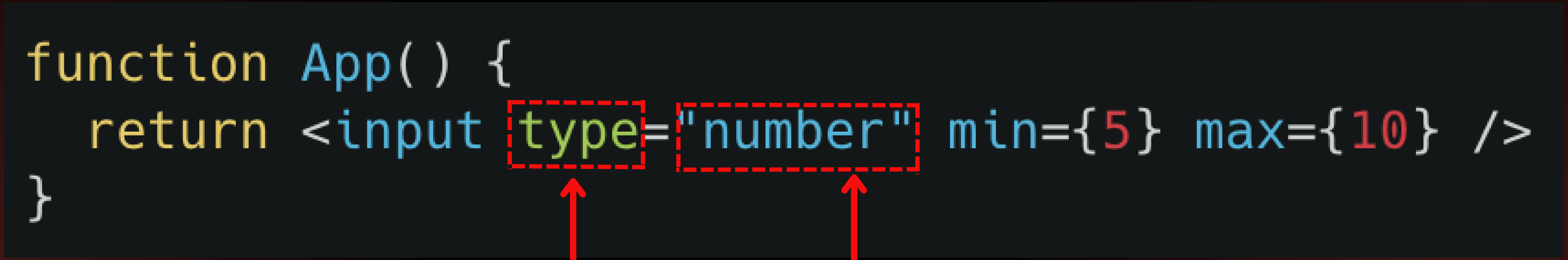
```
function App() {
  const message = 'Hello';
  const sum = 1 + 1;

  return (
    <div>
      <div>Message is: {message}</div>
      <div>Sum is: {sum}</div>
    </div>
  );
}
```



```
function App() {
  return <input type="number" min={5} max={10} />
}
```

"Props"

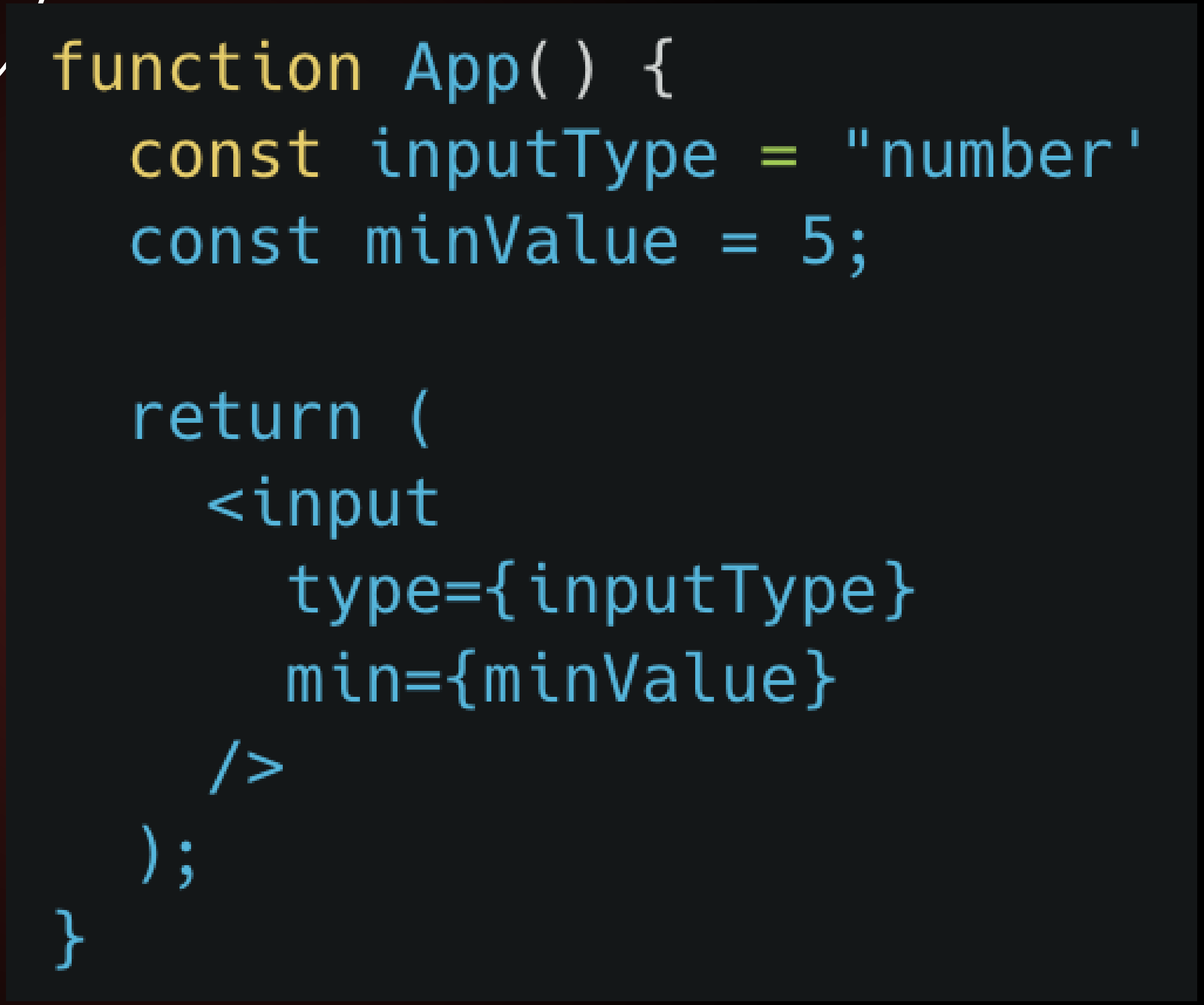
Customizes
an element



```
function App() {
  return <input type="number" min={5} max={10} />
}
```

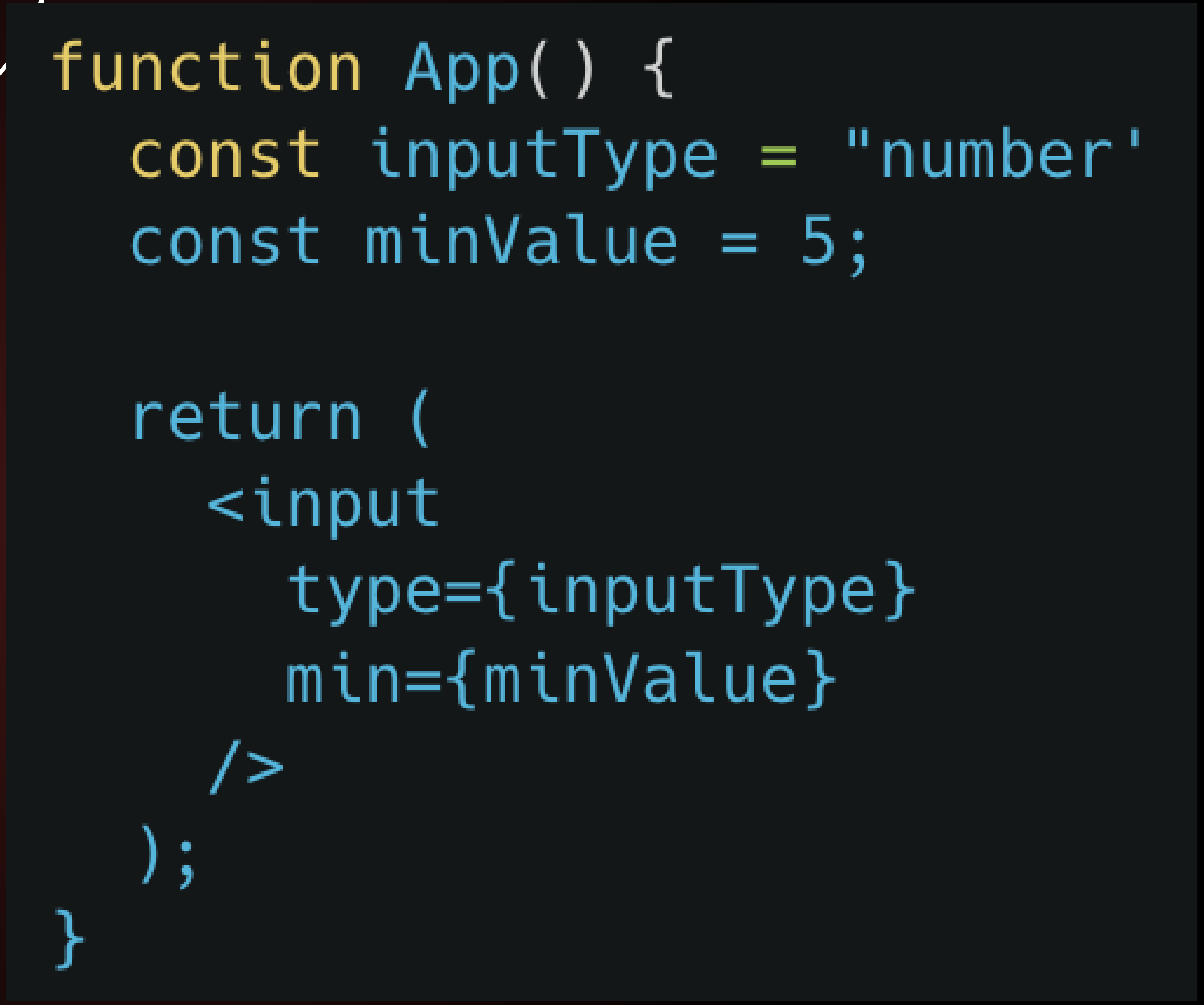
Name of the
propoty we want
to customize

Value for the
property



```
function App() {
  const inputType = "number'
  const minValue = 5;

  return (
    <input
      type={inputType}
      min={minValue}
    />
  );
}
```

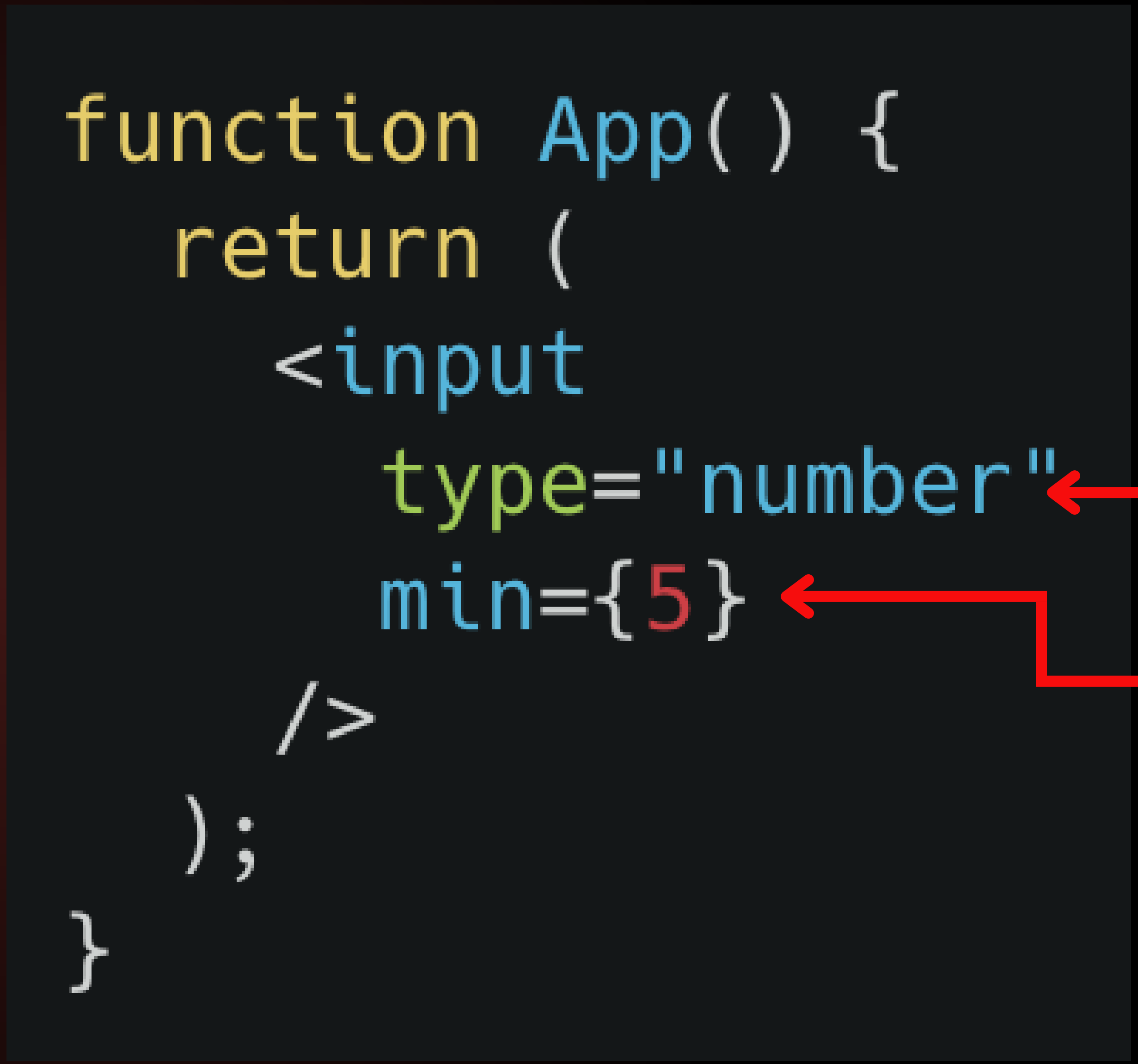
Props can refer to a variable using the same curly braces syntax



```
function App() {
  const inputType = "number'
  const minValue = 5;

  return (
    <input
      type={inputType}
      min={minValue}
    />
  );
}
```

Props can refer to a variable using the same curly braces syntax

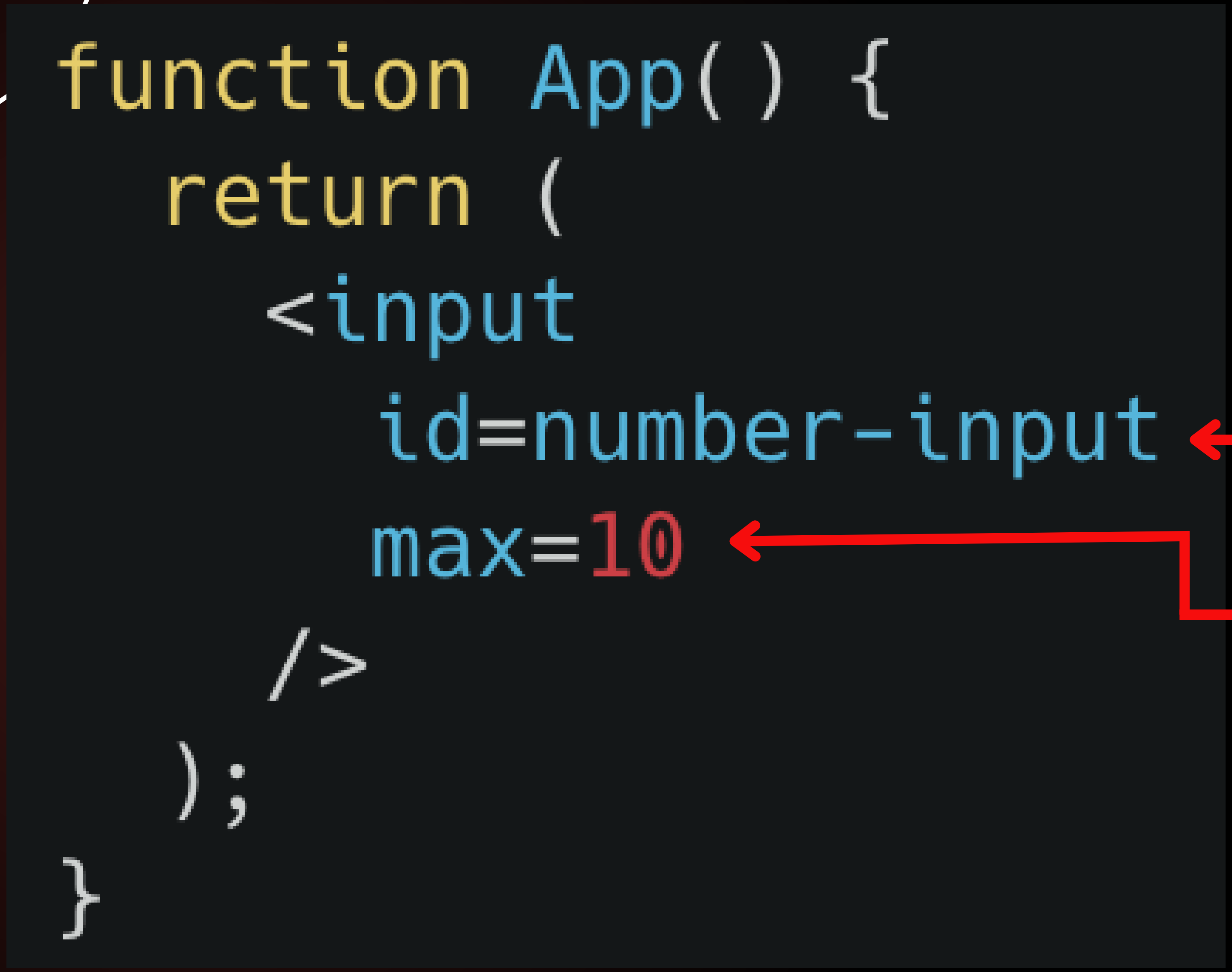
Props don't have to be defined as variables



```
function App() {
  return (
    <input
      type="number"
      min={5}
    />
  );
}
```

Strings - Wrap with **double** quotes

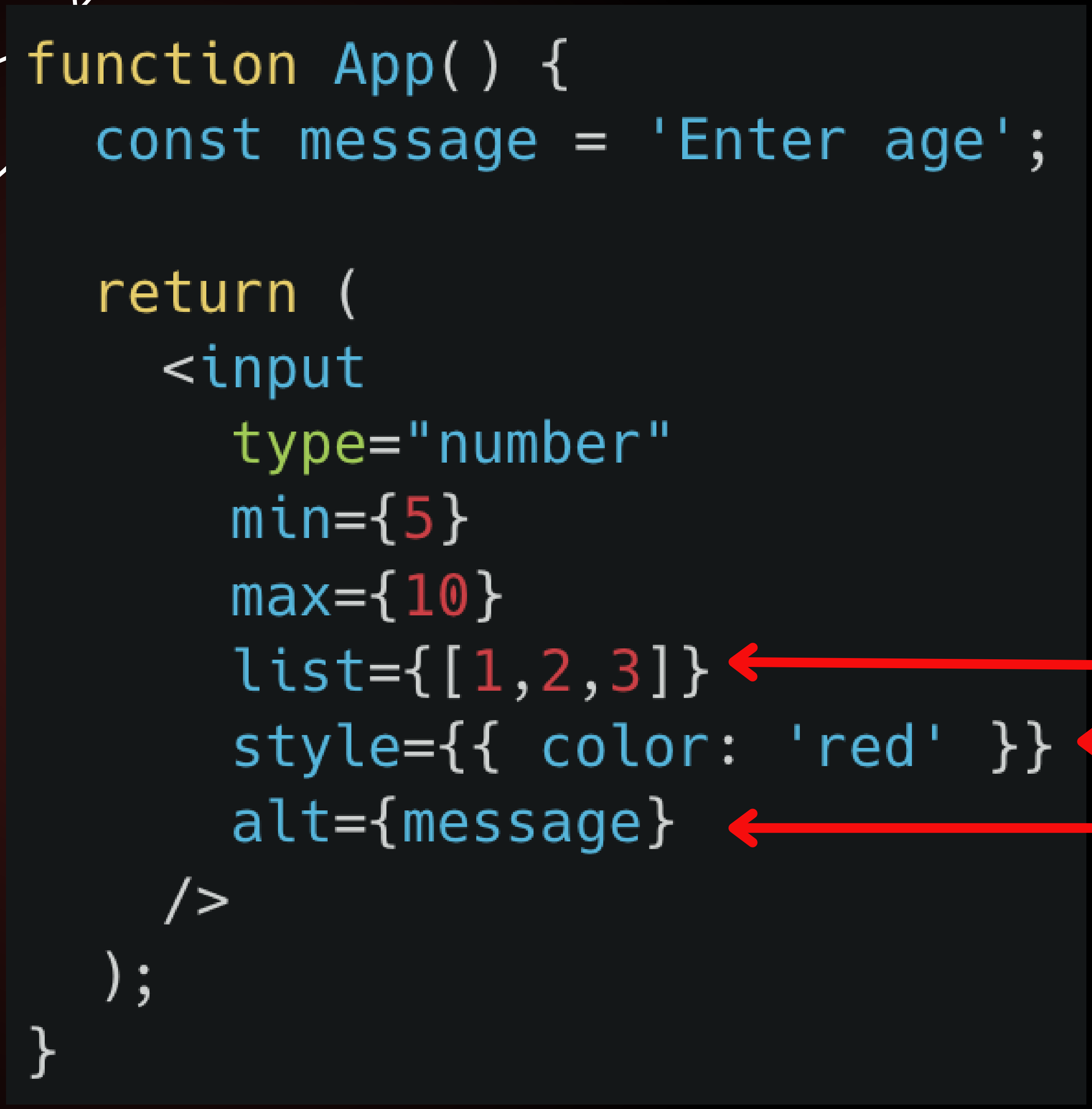
Numbers - Wrap with **curly** braces



```
function App() {
  return (
    <input
      id=number-input
      max=10
    />
  );
}
```

Error! Should have double quotes

Error! Should have curly braces



```
function App() {
  const message = 'Enter age';

  return (
    <input
      type="number"
      min={5}
      max={10}
      list={[1,2,3]}
      style={{ color: 'red' }}
      alt={message}
    />
  );
}
```

Arrays - Wrap with curly braces

Objects - Wrap with curly braces

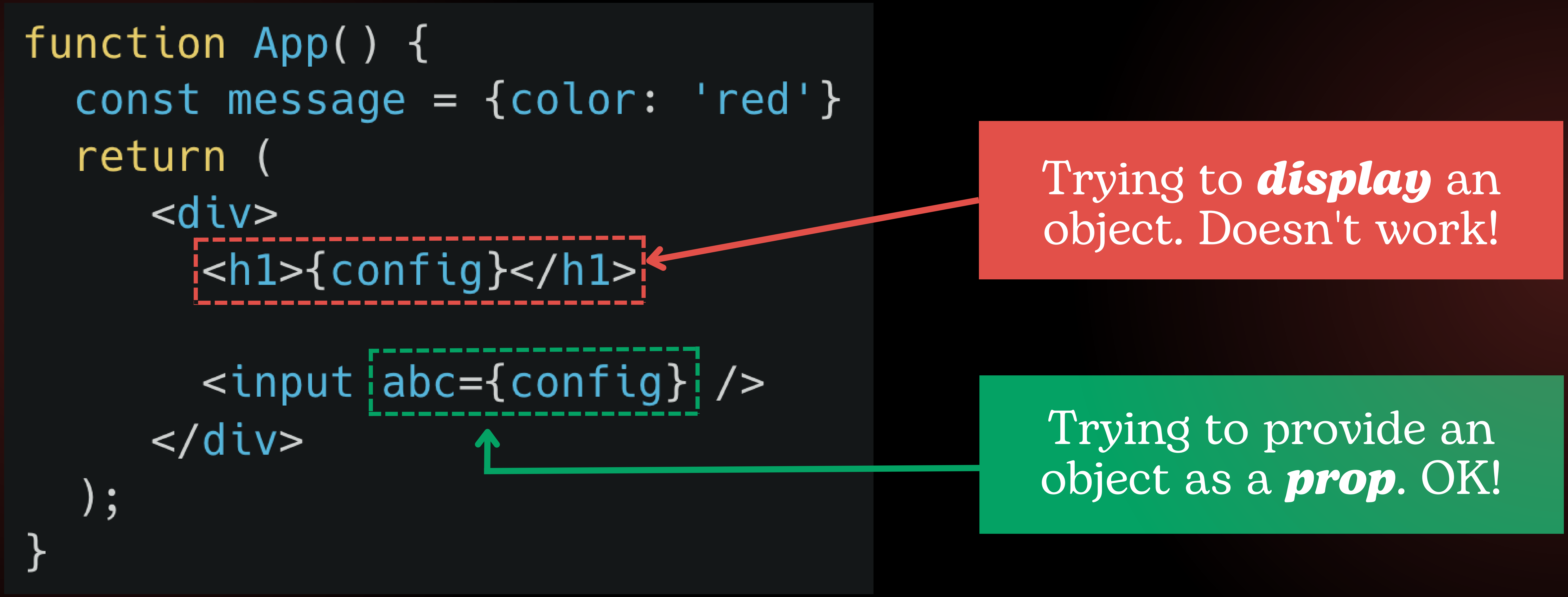
Variables - Wrap with curly braces

```jsx
function App() {
  const message = 'Enter age';

  return (
    <input
      type="number"
      min={5}
      max={10}
      list={[1,2,3]}
      style={{ color: 'red' }}
      alt={message}
    />
  );
}
```

Arrays - Wrap with **curly** braces

Objects - Wrap with **curly** braces

Variables - Wrap with **curly** braces



```
function App() {
  const message = {color: 'red'}
  return (
     <div>
       <h1>{config}</h1>

       <input abc={config} />
     </div>
  );
}
```

Trying to ***display*** an object. Doesn't work!

Trying to provide an object as a ***prop***. OK!

```
function App() {
  const message = 'Enter age';

  return (
    <input
      type="number"
      min={5}
      max={10}
      list={[1,2,3]}
      style={{ color: 'red' }}
      alt={message}
    />
  );
}
```

Arrays - Wrap with **curly** braces

Objects - Wrap with **curly** braces

Variables - Wrap with **curly** braces

Names/values of attributes that you provide to elements in HTML are *slightly different* when writing JSX

Converting HTML to JSX

1. All prop names follow camelCase
2. Number attributes use curly braces
3. Boolean 'true' can be written with just the property name. 'False' should be written with curly braces
4. The 'class' attribute is written as 'className'
5. In-line styles are provided as objects

Rule #1

In JSX, all prop names follow camelCase

HTML	Equivalent JSX
`<input maxlength="5" />`	`<input maxLength={5} />`
`<form autocapitalize />`	`<form autoCapitalize />`
`<form novalidate />`	`<form noValidate />`

Rule #2

In JSX, attributes meant to be numbers should be provided as numbers with curly braces

HTML		Equivalent JSX
`<input maxlength="5" />`	→	`<input maxLength={5} />`
`<meter optimum="50" />`	→	`<meter optimum={50} />`

Rule #3

In JSX, boolean 'true' can be written with just the property name. 'False' should be written with curly braces

HTML	Equivalent JSX
<input spellcheck="true" />	<input **spellCheck** />
<input spellCheck={false} />	<input spellCheck=**{false}** />

Rule #4

In JSX, the 'class' attribute is written as 'className'

HTML		Equivalent JSX
`<div class="divider" />`	→	`<div` **className**`="divider" />`
`<li class="item" />`	→	`<li` **className**`="item" />`

Rule #5

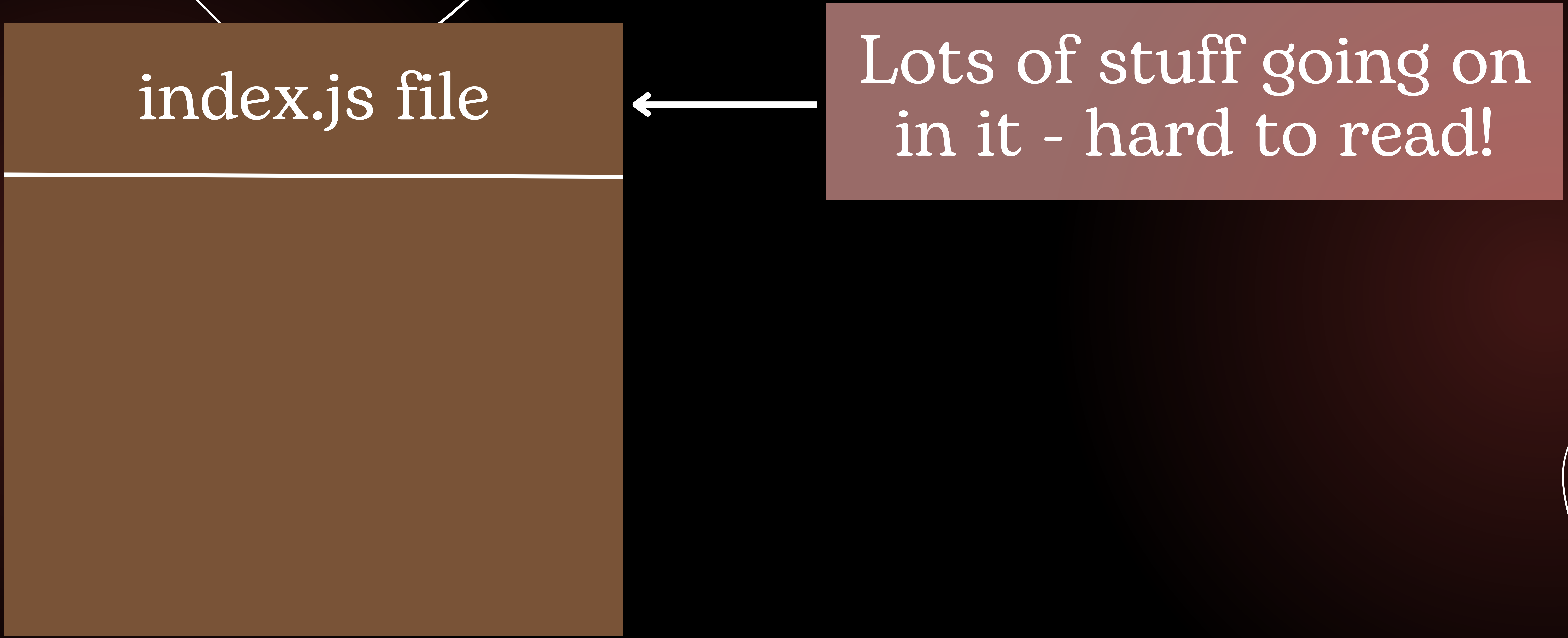
In JSX, in-line styles are provided as objects

HTML

```
<a
  style="text-decoration: 'none'; padding: '5px';"
/>
```

Equivalent JSX

```
<div
  style={{ textDecoration: 'none', padding: '5px'}}
/>
```


index.js file
Lots of stuff going on in it - hard to read!

index.js file

1. Import React and ReactDOM
2. Import the App component
3. Create a root
4. Show the App component

App.js file

Code to create a component

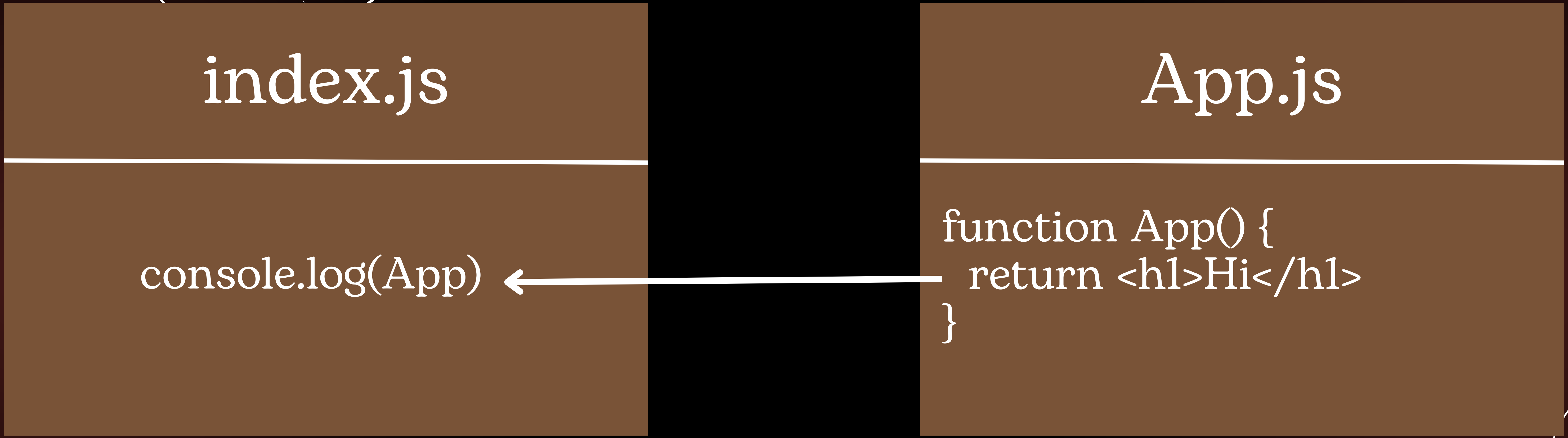
Create a Component

1. Create a new file. (By convention) File should start with a capital letter
2. Make your component. Should be a function that returns JSX.
3. Export the component at the bottom of the file
4. Import the component into another file
5. Use the component

Optional video on Module Systems (import/export statements)

Skip if you're already familiar

Challenging video. info dump. I will repeat all of this many many times throughout the course



index.js

```
console.log(App)
```

App.js

```
function App() {
  return <h1>Hi</h1>
}
```

Error!

Variables aren’t shared between files by default

index.js

```
import App from './App'
console.log(App)
```

App.js

```
function App() {
  return <h1>Hi</h1>
}

export default App
```

OK!
Variables shared between files by using import/export statements

Export Statements

index.js

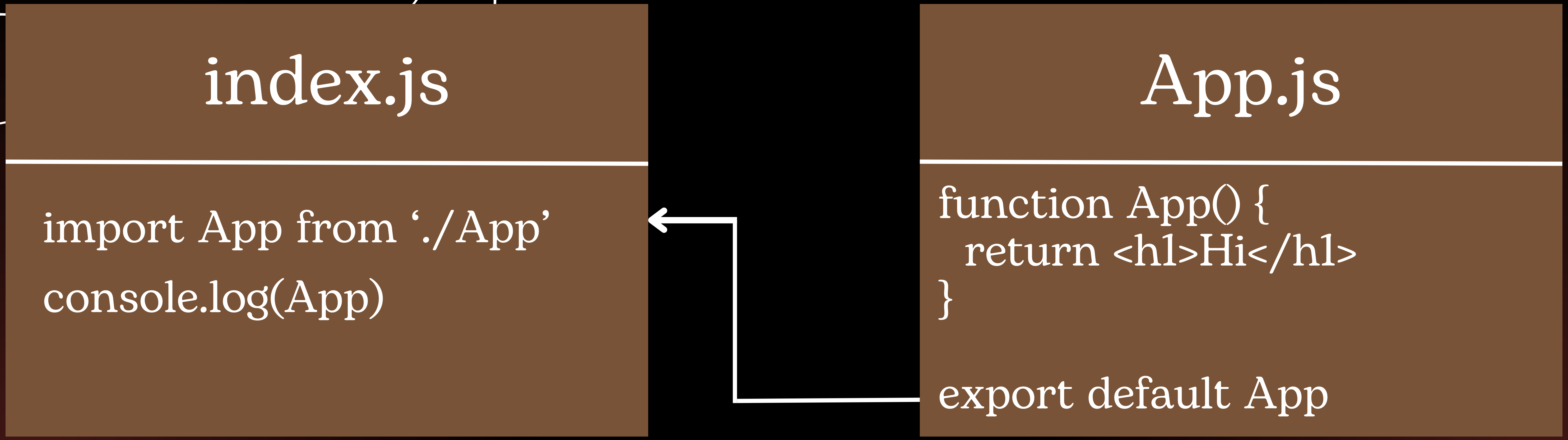
```
function App() {
  return <h1>Hi</h1>
}

export default App
```

```
export default function App() {
  return <h1>Hi</hi>
}
```

Two kinds - ‘default’ and ‘named’

A file can only have a single ‘default’ export

Two ways to write a default export



index.js

```
import App from './App'
console.log(App)
```

App.js

```
function App() {
  return <h1>Hi</h1>
}

export default App
```

Import Statements - Behind the Scenes

1. Declare a variable called App
2. Find the default export from App.js
3. Assign the default export to App variable

index.js

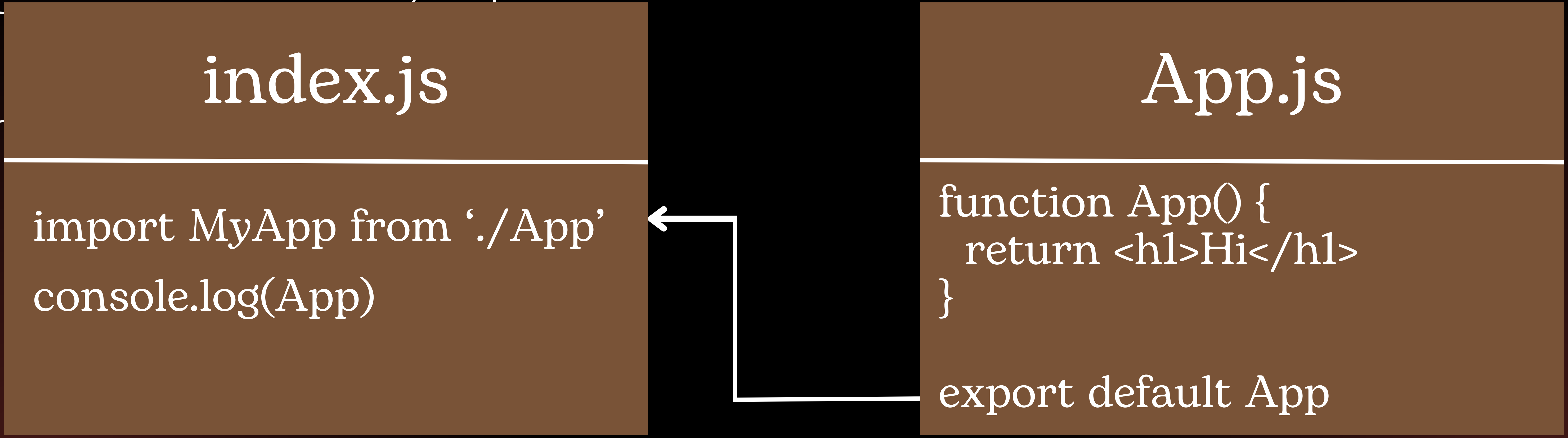
```
import App from './App'
console.log(App)
```

App.js

```
function App() {
  return <h1>Hi</h1>
}

export default App
```

Confusing Thing! Default exports can be renamed in the importing file !!!

1. Declare a variable called App
2. Find the default export from App.js
3. Assign the default export to App variable



index.js

```
import MyApp from './App'
console.log(App)
```

App.js

```
function App() {
  return <h1>Hi</h1>
}

export default App
```

Confusing Thing!
Default exports can be renamed in the importing file !!!

1. Declare a variable called App
2. Find the default export from App.js
3. Assign the default export to MyApp variable

index.js

```
import MyApp from './App'
const App = 5;
```

App.js

```
function App() {
  return <h1>Hi</h1>
}

export default App
```

Confusing Thing! Default exports can be renamed in the importing file !!!

1. Declare a variable called App
2. Find the default export from App.js
3. Assign the default export to MyApp variable

Named Export Statements

App.js

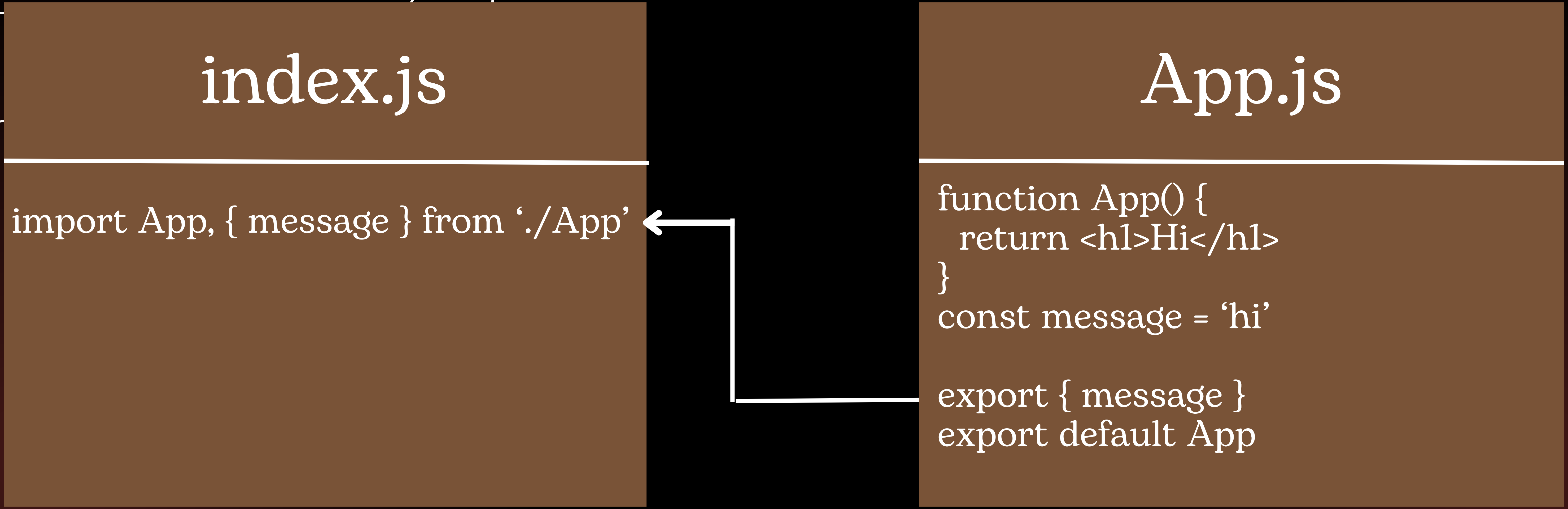
```
function App() {
  return <h1>Hi</h1>
}

export default App

const message = 'hi'
export  {message }
```

```
export default function App() {
  return <h1>Hi</hi>
}

export const message = 'hi'
```

Use when exporting multiple variables

Can have as many named exports as we want

Two ways to write a named export



index.js

```
import App, { message } from './App'
```

App.js

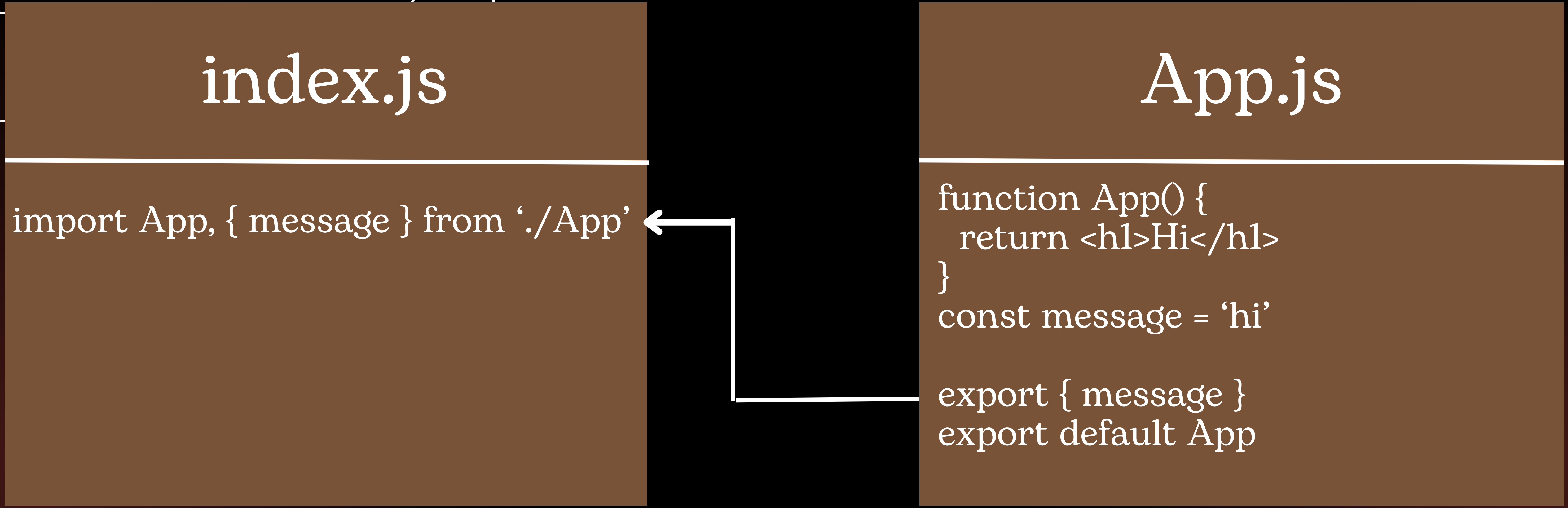
```
function App() {
  return <h1>Hi</h1>
}
const message = 'hi'

export { message }
export default App
```

Import Statements

Declare a variable called App

Find the default export from App.js

Assign the default export to MyApp variable



```
import App, { message } from './App'
```

```
function App() {
  return <h1>Hi</h1>
}
const message = 'hi'

export { message }
export default App
```

Import Statements

Named exports cannot be renamed!!!

Use when exporting multiple variables

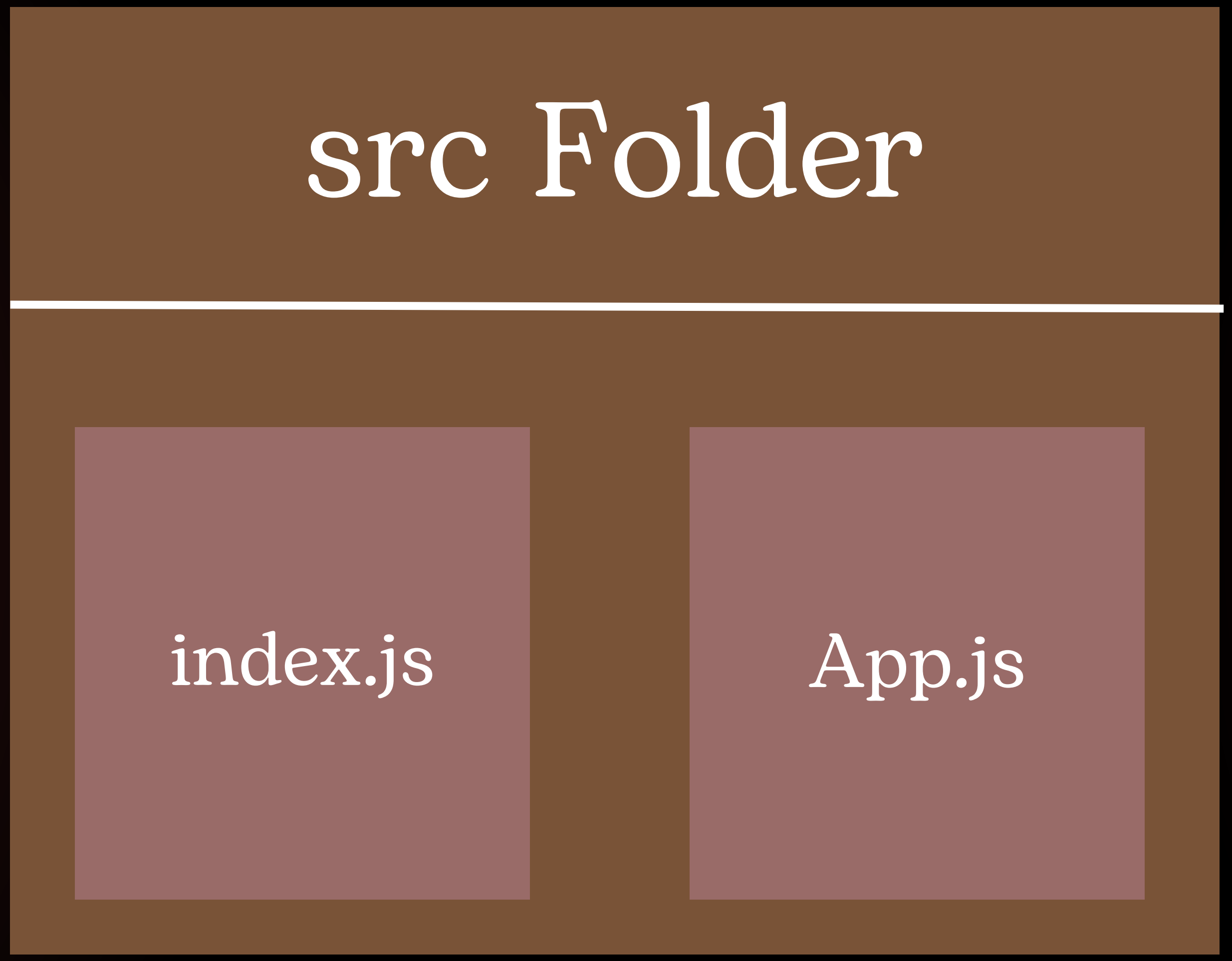
‘./’ or ‘../’ means we are importing a file that we created

Can have as many named exports as we want

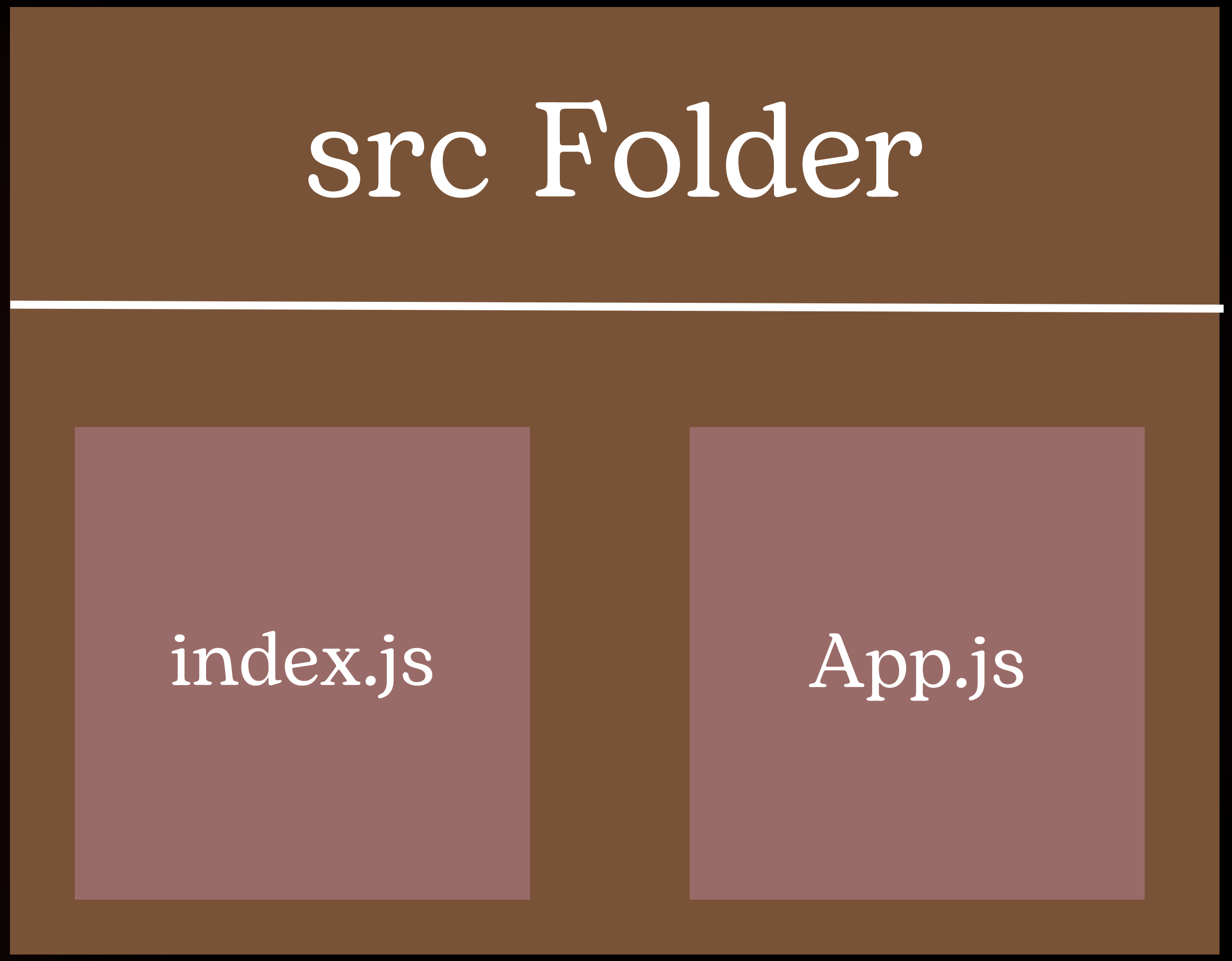
Two ways to write a named export

No ‘./’ or ‘../’ means we are importing a package

Use when exporting multiple variables $\longrightarrow$ Relative path to walk from this file to the file we are importing

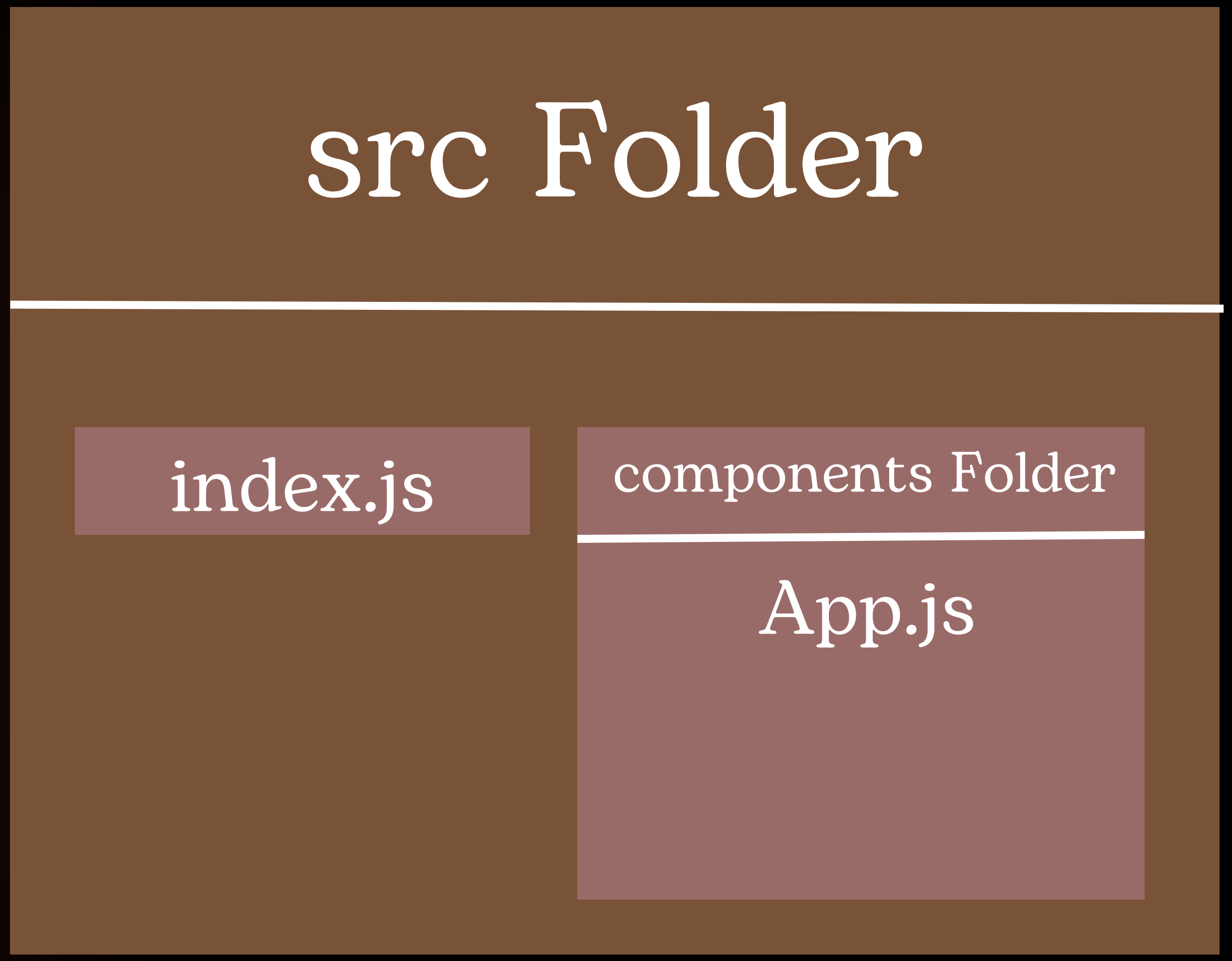


If we are in the index.js file and want to import from App.js...



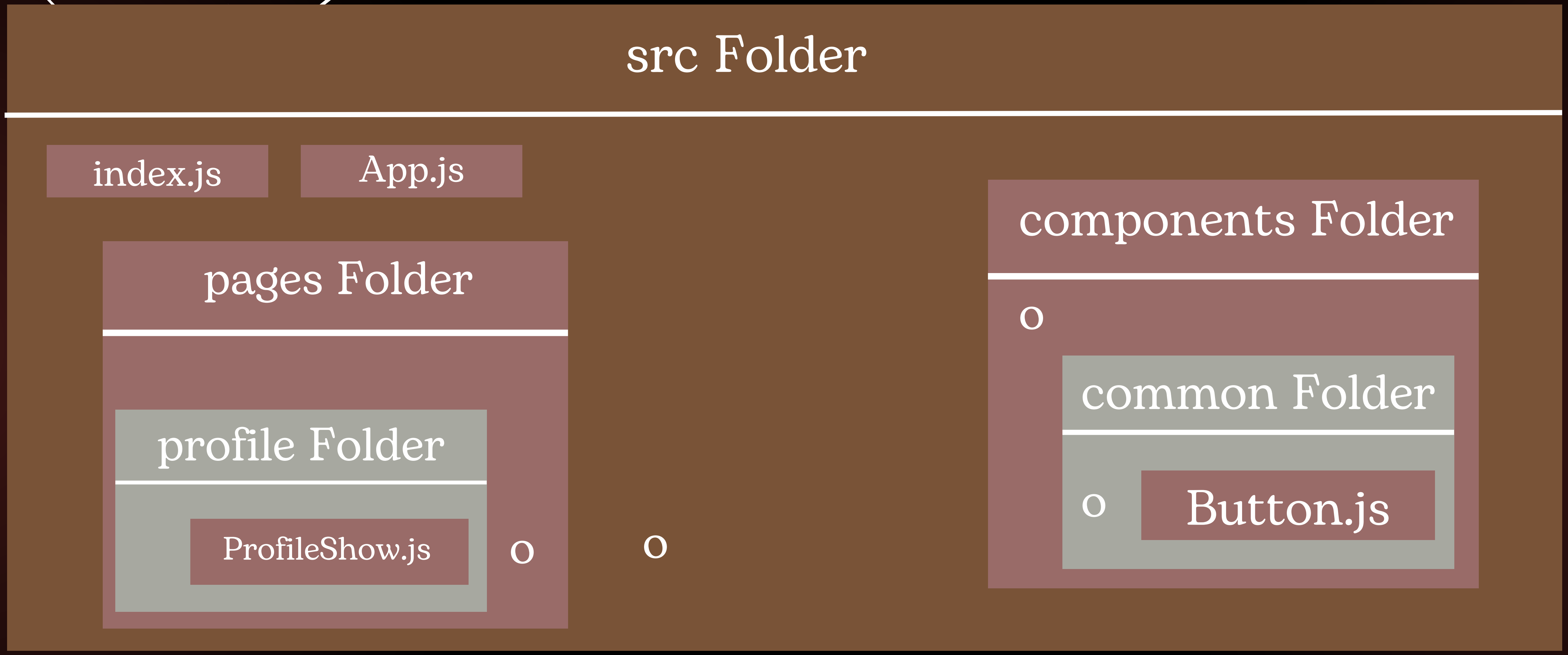
If we are in the index.js file and want to import from App.js...

```
import alskjdfj from './App';
```

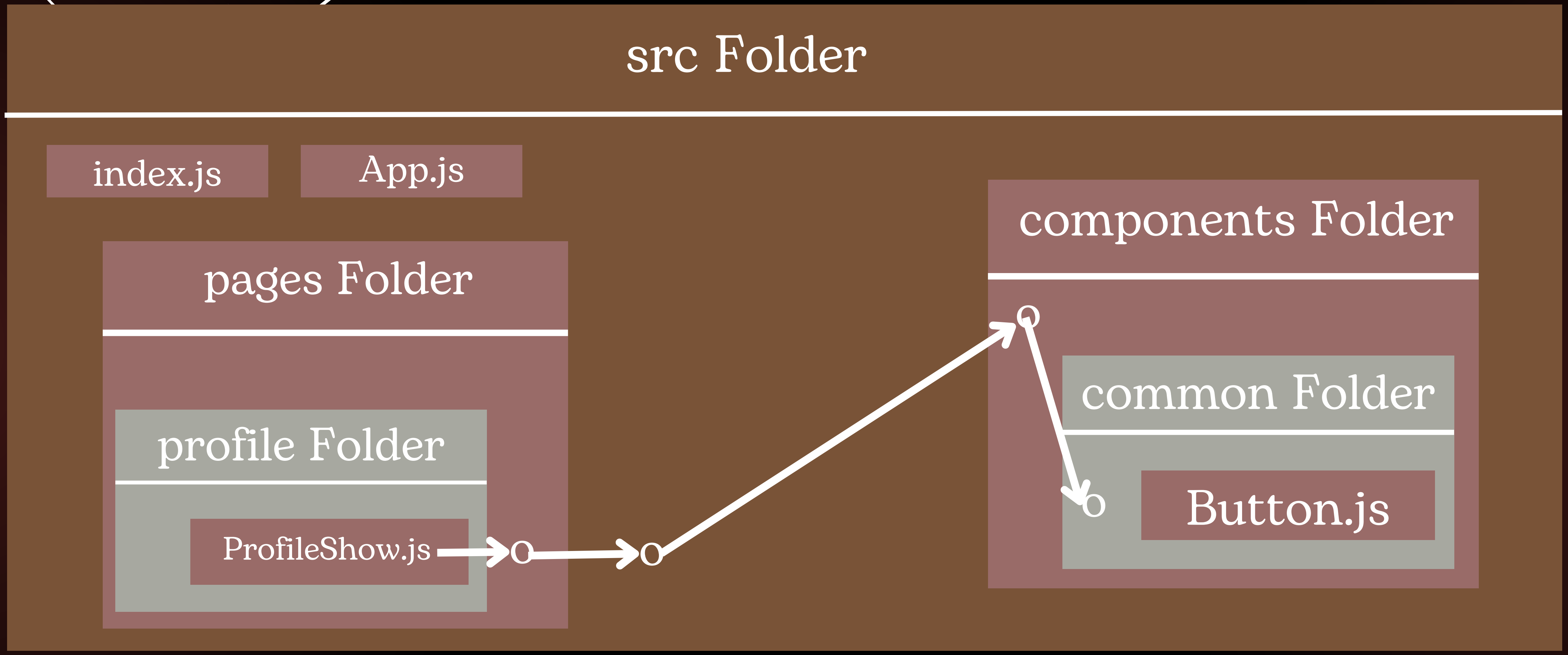


If we are in the index.js file and want to import from App.js...

```
import asdf from ‘../index’;
```



If we are in the ProfileShow.js file and want to import from Button.js....



If we are in the ProfileShow.js file and want to import from Button.js....

import asdf from '../../components/comon/Buton'

Optional video on Module Systems (import/export statements)

Skip if you're already familiar

Challenging video. info dump. I will repeat all of this many many times throughout the course