RENDER PROPS

I heard you like rendering props so I got you some render props

BUT FIRST…LET'S REVIEW JSX…

Q: WHAT DOES THIS GET COMPILED TO?

```
<MyComponent />
```

Q: WHAT DOES THIS GET COMPILED TO?

```
<MyComponent />
```

A:

```
React.createElement(MyComponent)
```

Q: WHAT IF MY COMPONENT ISN'T A COMPONENT?

```
const MyComponent = 'I am not a component.'

<MyComponent />
```

Q: WHAT IF MY COMPONENT ISN'T A COMPONENT?

```
const MyComponent = 'I am not a component.'

<MyComponent />
```

A: SAME THING:

```
React.createElement(MyComponent)
```

WHA?!?

- You may have noticed: When you fail to properly export a Component, it's *React* that complains (in the browser's console at runtime), rather than the compiler (in your terminal at build time)

HOW DOES JSX RECOGNIZE COMPONENTS?

IT DOESN’T

IT DOESN’T

- JSX doesn’t know or care if the component you name exists

IT DOESN'T

- JSX doesn't know or care if the component you name exists

IT DOESN'T

- JSX doesn't know or care if the component you name exists

- It doesn't look up components by filename

IT DOESN'T

- JSX doesn't know or care if the component you name exists
- It doesn't look up components by filename

IT DOESN'T

- JSX doesn't know or care if the component you name exists
- It doesn't look up components by filename
- The JSX compiler just generates JS *assuming* that there is (or will be) a component of that name in scope by the time the JS runs

Q: WHAT DOES THIS GET COMPILED TO?

```
<div />
```

Q: WHAT DOES THIS GET COMPILED TO?

```
<div />
```

A:

```
React.createElement('div')
```

Q: WHY THE QUOTES?

Q: WHY THE QUOTES?

A: FOR HTML TAGS, JSX CREATES ELEMENTS WHOSE TYPES ARE STRINGS

```
React.createElement('div')
```

Q: HOW DOES IT KNOW THE DIFFERENCE?

```
<Div />  // Becomes React.createElement(Div)
<div />  // Becomes React.createElement('div')
```

Q: HOW DOES IT KNOW THE DIFFERENCE?

```
<Div />  // Becomes React.createElement(Div)
<div />  // Becomes React.createElement('div')
```

A: YEP, IT LITERALLY LOOKS AT CAPITALIZATION

COMPONENT GENERATION

COMPONENT GENERATION

- If your JSX tag begins with a capital letter, it generates a React Element whose type is a Component

COMPONENT GENERATION

- If your JSX tag begins with a capital letter, it generates a React Element whose type is a Component

COMPONENT GENERATION

- If your JSX tag begins with a capital letter, it generates a React Element whose type is a Component

- When an Element is rendered, that Component will get mounted:

COMPONENT GENERATION

- **If your JSX tag begins with a capital letter, it generates a React Element whose type is a Component**

- **When an Element is rendered, that Component will get mounted:**
  - If it's a pure function, it'll get called to generate its render output

COMPONENT GENERATION

- **If your JSX tag begins with a capital letter, it generates a React Element whose type is a Component**

- **When an Element is rendered, that Component will get mounted:**
  - If it's a pure function, it'll get called to generate its render output
  - If it's a class, it'll get instantiated with \`new\`, it's lifecycle methods will get called, and \`render\` will be called to generate its render output

COMPONENT GENERATION

- **If your JSX tag begins with a capital letter, it generates a React Element whose type is a Component**

- **When an Element is rendered, that Component will get mounted:**
  - If it’s a pure function, it’ll get called to generate its render output
  - If it’s a class, it’ll get instantiated with `new`, it’s lifecycle methods will get called, and `render` will be called to generate its render output

COMPONENT GENERATION

- **If your JSX tag begins with a capital letter, it generates a React Element whose type is a Component**
- **When an Element is rendered, that Component will get mounted:**
  - If it's a pure function, it'll get called to generate its render output
  - If it's a class, it'll get instantiated with `new`, it's lifecycle methods will get called, and `render` will be called to generate its render output
- **Otherwise, it generates a React Element whose type is a string**

COMPONENT GENERATION

- **If your JSX tag begins with a capital letter, it generates a React Element whose type is a Component**

- **When an Element is rendered, that Component will get mounted:**
  - If it's a pure function, it'll get called to generate its render output
  - If it's a class, it'll get instantiated with `new`, it's lifecycle methods will get called, and `render` will be called to generate its render output

- **Otherwise, it generates a React Element whose type is a string**
- **These Elements become HTML tags of that name when they're rendered**

Q: WHAT ABOUT PROPS?

```
<MyComponent title='hello' foo={bar} />
```

Q: WHAT ABOUT PROPS?

```
<MyComponent title='hello' foo={bar} />
```

A: PROPS ARE THE SECOND ARGUMENT TO CREATE ELEMENT

```
  React.createElement(MyComponent, {
    title: 'hello',
    foo: bar
})
```

Q: WHAT ABOUT NESTED ELEMENTS?

```
<MyComponent title='hello'>
  <h1>Hi there</h1>
  <AnotherComponent />
</MyComponent>
```

Q: WHAT ABOUT NESTED ELEMENTS?

```
<MyComponent title='hello'>
  <h1>Hi there</h1>
  <AnotherComponent />
</MyComponent>
```

A: CHILDREN ARE THE THIRD ARGUMENT TO CREATE ELEMENT 😉

```
React.createElement(MyComponent, {title: 'hello'}, [
  React.createElement('h1'),
  React.createElement(AnotherComponent),
])
```

A: CHILDREN ALSO GET PUT ON PROPS, SO THIS ALSO WORKS

```
React.createElement(MyComponent, {
  title: 'hello',
  children: [
    React.createElement('h1'),
    React.createElement(AnotherComponent),
  ]
})
```

LET’S PLAY