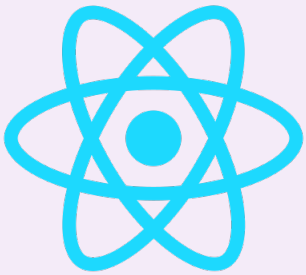
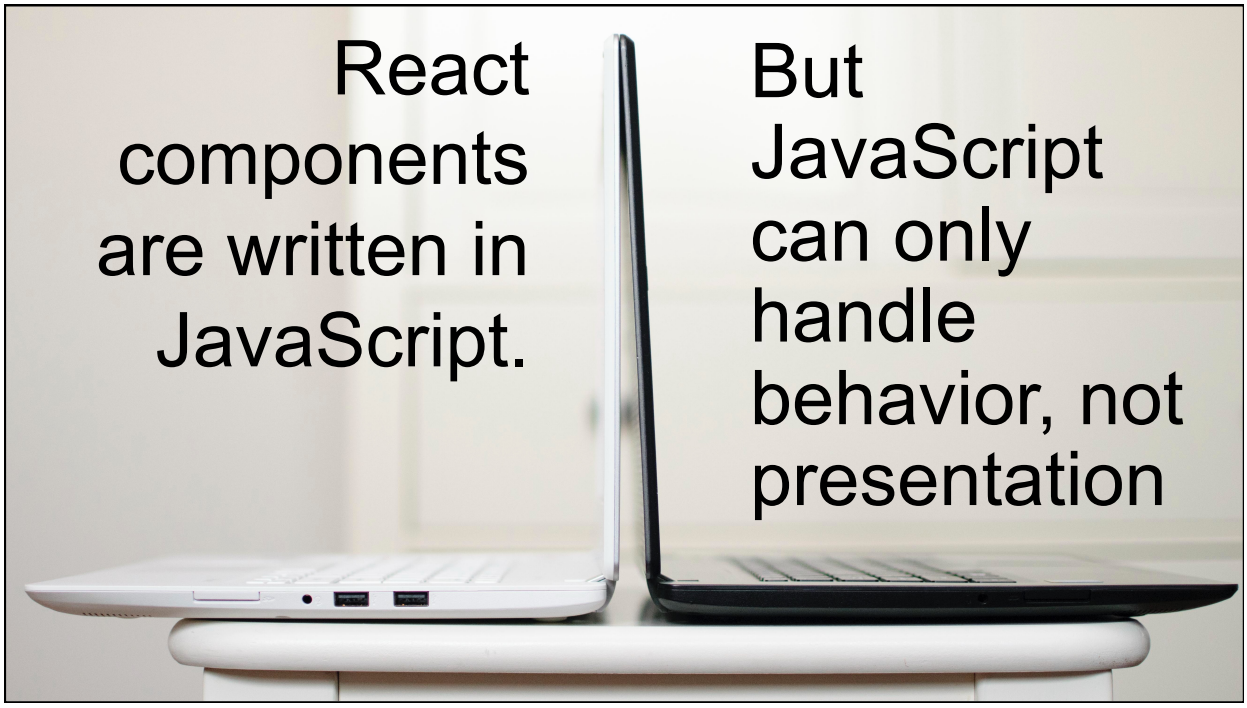


Stateless functional components

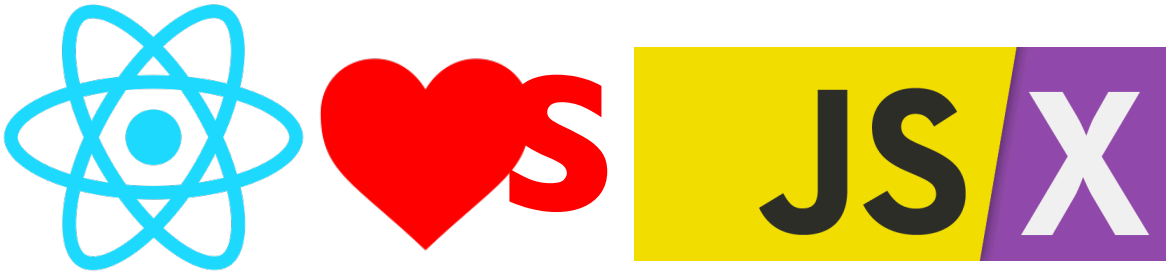
... because less is more

tl;dr

- The 7 rules of JSX
- Everything it takes to make a component
- The two ways to get a component on a page



To help with the presentation, React introduces a new DSL* called JSX



* Domain-Specific Language

JSX


JavaScript + XML = JSX
It isn't JSH. It is JSX.
HTML is forgiving ... XML is not!

XML must be ...

- Well-formed
  - Conforms to rules
- Valid
  - Conforms to its DTD

Rules of well-formed XML

- All XML elements must have a closing tag.
- XML tags are case-sensitive.
- All XML elements must be properly nested.
- All XML documents must have a root element.
- Attribute values must always be quoted.

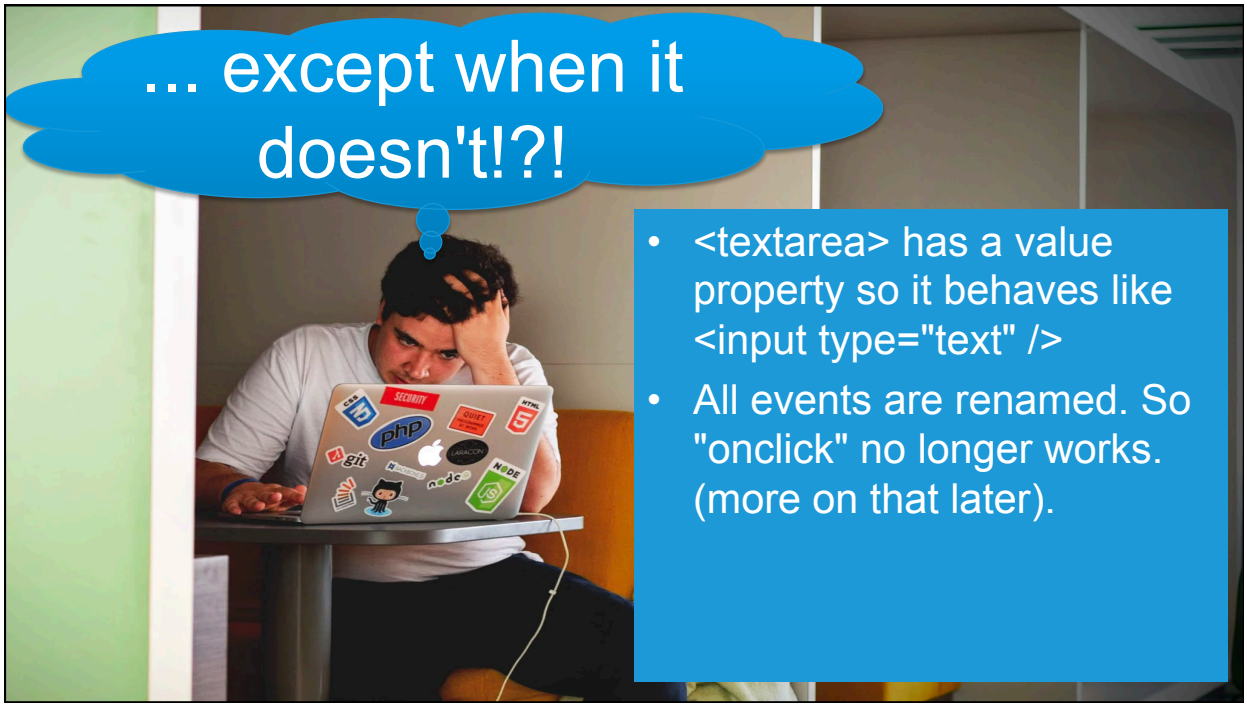
Alright, how about that valid thing?

Well, there is no DTD, but it does follow rules ...

- All tags must be pre-defined either ...
  - Valid W3C-approved elements with proper attributes
  - Valid pre-defined React components that you wrote
- Note: it uses casing to determine which is which.
  - W3C tags are lower-cased.
  - React components begin with an upper-case character

- Got compile errors in your JSX? Ask yourself "Is this strictly good W3C HTML?" And fix it where it is not standard.
- If you want to use a non-standard attribute, precede it with data-

JSX must conform to W3C standards ...



Quiz: What's wrong with this?

```
let class = "Programming 101";
let for = "because I want to graduate";
```

- Therefore, with JSX you can't go ...

```
<label class="big" for="firstname">First</label>
<input id="firstname" />
```

- Instead you have to ...

```
<label className="big" htmlFor="firstname">First</label>
<input id="firstname" />
```

- Warning in the console. Not a compile error.

But wait! What browsers support JSX?

- All of them! (Or none of them?)
- Because JSX never lands in the browser.
- It is transpiled out and replaced with equivalent JavaScript

To create a component

Here's all that is required

1. import React from 'react'
2. Create a function that returns JSX
3. Display that component inside a host.

1. import react

At the top of your component file ...

```
import React from 'react';
```

2. Create a function ...

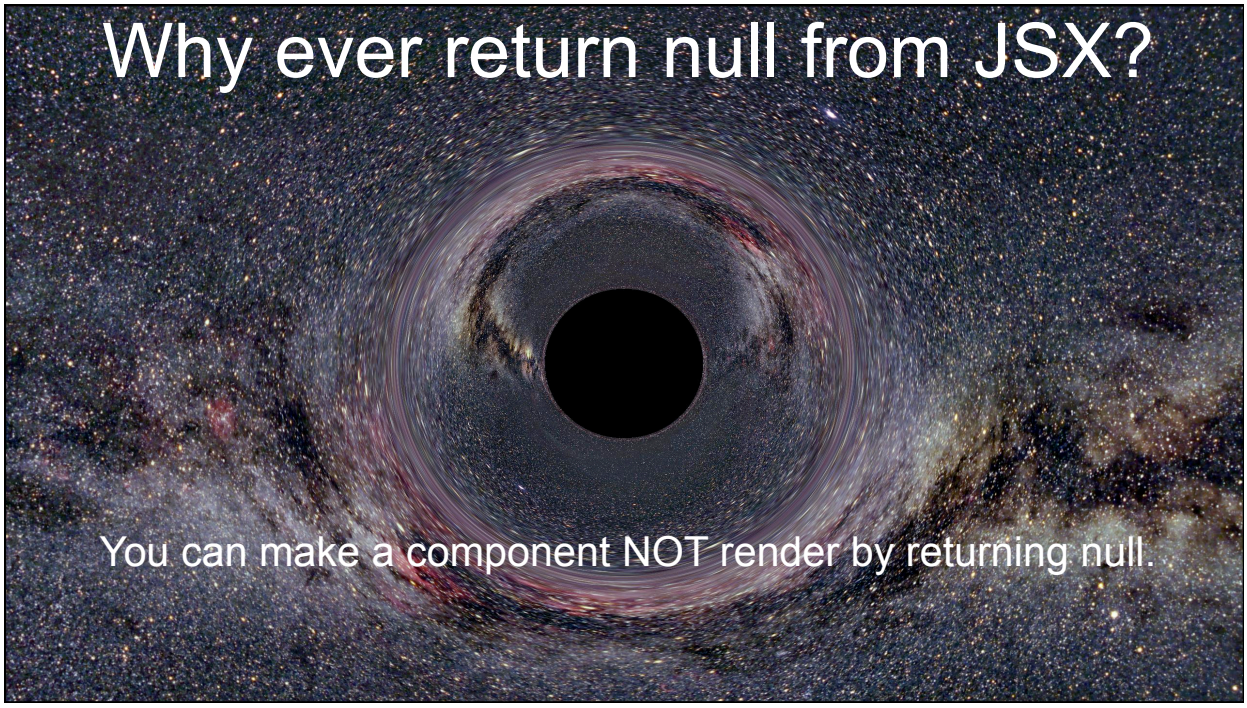
- You create a function that returns JSX

```
function Person() {
  // You can do other things here.
  return <div>Hello world</div>
}
```

- Must start with an uppercase letter! Prefer Pascal-cased

... that returns JSX

- You must return
  1. JSX with a single root element
  2. An array of JSX elements
  3. A string
  4. null
- Anything else is an error
- Tip: When you want a return and then have JSX starting on the next line, wrap them in parentheses or else Babel gets confused.



3. Host your component

You can host it inside another component or it can be the root component.

To host inside another component, put the name of this one in the parent's JSX

```
function OtherComponent() {
  return <div>
    <Person />
    <Person></Person>
  </div>
}
```

This would show two Person components.

To make this the top-level component, use ReactDOM.render()

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
ReactDOM.render(
  <Person></Person>
  , document.getElementById('root'));
```

This means that index.html must have an element with id="root".

One last thing ...

Later in the course we're going to be covering another way to write components that is more complex but it is also more capable.

tl;dr

- The 7 rules of JSX
- Everything it takes to make a component
- The two ways to get a component on a page