Components

A React "Component" returns JSX/HTML

- A js function
    - "function-based component" or
    - "functional component"
- old style is "class-based"
    - We won't be using those
    - **Warning**: React docs are years out of date :(
    - See **https://beta.reactjs.org/** instead

Components are Tags

A React Component can be used as an Element in JSX

- Open/close or self-closing
    - `<Greeting></Greeting>` or `<Greeting/>`
- Consistent!
    - html elements in JSX are ALSO consistent!
- Element name matches function name
    - MixedCase, not camelCase
        - YES: `<Greeting/>` or `<CatVideos/>`
        - NO: `<greeting/>` or `<catVideos/>`

Components are not files

OFTEN a `.jsx` file is exactly 1 component

- This is not required

For this course, it IS required

- one file === one component
- filename should match component name
- once you understand more nuance, then can change

Components are a single container

A component can have any collection of nested elements/components

- But MUST have a single parent container element
- OR be a "fragment"
    - more on that later

Example of single parent container

This works:

```
function CatFacts() {
  return (
    <div className="cat-facts">
      <h1>Cat Facts</h1>
      <div className="subtitle">Number 3 will shock you</div>
      <ul>
        <li>Cats can rotate their ears 180 degrees</li>
        <li>Felines can purr or roar, but not both<li>
        <li>Humans domesticated dogs,
          but cats domesticated humans</li>
      </ul>
    </div>
  );
}

export default CatFacts;
```

Example without single parent container

This will give you an error:

```
function CatFacts() {
  return (
    <h1>Cat Facts</h1>
    <div className="subtitle">Number 3 will shock you</div>
    <ul>
      <li>Cats can rotate their ears 180 degrees</li>
      <li>Felines can purr or roar, but not both<li>
      <li>Humans domesticated dogs,
        but cats domesticated humans</li>
    </ul>
  );
}

export default CatFacts;
```

imports

Even without React, we want multiple files to organize

- big files = hard to manage/maintain

CRA includes a *bundler* (webpack)

- lets us use many files in dev
- outputs to fewer files in prod

Syntax is not browser JS

- bundler converts

Importing JSX

Write a `Test.jsx` in `src/`

```
function Test() {
  return (
    <p>Hello World</p>
  );
}
export default Test;
```

Top of `App.jsx`:

```
import Test from './Test';
```

Near end of `App.jsx`:

```
  <Test/>
  </header>
```

The parts of importing

- say what you want to export
- say what you are importing
    - and from where
- use what you've imported

We will start with discussing component imports first

- other imports are different rules

Say what you are exporting

At end of file:

```
export default VARIABLE_NAME;
```

Example:

```
function CatVideo() { /* ... */ }
export default CatVideo;
```

Should match filename for ease of use

There are other export options

- we won't use them yet

This isn't JS that works in browser

- converted by tools that CRA gives us

Say what you are importing

...and from where

```
import CatVideo from './CatVideo';
import Component from './Filename';
```

- Can be single or double quotes
- `Component` is the name you will use
    - should match the filename for ease of use
- `Filename` is the filename
    - You need a path (`./`)
    - Can be a different directory
    - Do not need a file extension
        - will import `.jsx` or `.js`

Using your imported Component

Use an imported Component in a HTML-like JSX tag:

```
import Test from './Test';

function Demo() {
  return (
    <div className="demo">
      <Test/>
    </div>
  );
}
export default Demo;
```

Any file can import other files

- Gets weird/breaks if you make a circle
    - A uses B, and B uses A
    - Don't do that

importing CSS

CRA allows you to import CSS files

```
import './App.css';
```

- Makes the CSS available on the HTML page
- filename can be anything
    - does not have to be MixedCase
    - must have `.css` extension
    - must have a path (e.g. `./`)
- Do not need to have CSS with each component
    - can use `src/index.css`
    - or put all css in css file(s) imported in `App.jsx`

React has other options for CSS, more on that later

importing images

importing images LOOKS like importing Components:

```
import someImage from './cat-pic.jpg';
```

There are important differences:

- You pick a variable name to import as
- The filename needs to be complete
    - including file extension
    - and path
- Variable holds the path to the image as a string:
    - `<img src={someImage} alt="smug cat"/>`

importing JS

We will cover this more later

- but all components are JS

Component Props

Components have attribute-like values:

```
<Greeting target="world"/>
```

These are called "props"

- Allow you to pass values to Components
- Allows for flexibility and reuse

```
<Greeting target="class"/>
<Greeting target="world"/>
```

```
<p>Hello class</p>
<p>Hello world</p>
```

Prop values

Unlike HTML, props can hold more than strings

- non-strings must be in `{}`

Unlike HTML, props should ALWAYS have a value

- not there/not there like `disabled` or `checked`

```
<MovieSequels count={3} />
```

```
<ul class="sequels">
  <li>Cats: The Musical</li>
  <li>Cats: The Musical 2</li>
  <li>Cats: The Musical 3</li>
</ul>
```

Reading passed props

A Component function is passed an object of all props

```
<MovieSequels count={3} />
```

```
function MovieSequels( props ) {
  const list = [];

  for(let sequel = 1; sequel <= props.count; sequel  += 1) {
    const title = sequel === 1 ? '' : sequel;
    list.push( <li>Cats: The Musical {title}</li> );
  }

  return (
    <ul className="sequels">
      {list}
    </ul>
  );
}
export default MovieSequels;
```

Destructuring props

Common to **destructure** props object to get variables

```
function MovieSequels( { count } ) {
  const list = [];

  for(let sequel = 1; sequel <= count; sequel  += 1) {
    const title = sequel === 1 ? '' : sequel;
    list.push( <li>Cats: The Musical {title}</li> );
  }

  return (
    <ul className="sequels">
      {list}
    </ul>
  );
}
export default MovieSequels;
```

Events

Components are JS that outputs HTML

- So how do we attach event listeners to HTML?

"on" Handlers

```
function doMeow() {
  console.log('meow');
}

function Meow() {
  return (
    <p onClick={doMeow}>Meow</p>
  );
}

export default Meow;
```

But WAIT!

Didn't we say NOT to use "onclick" in HTML?!

Yes!

- but this isn't HTML
- it LOOKS like HTML, but isn't
- Differences are subtle but real

Comparing

Bad:

```
<p onclick="function() { console.log('meow') }">Meow</p>
```

- Editing JS in HTML
  - hard to find
  - hard to edit

Good:

```
<p onClick={doMeow}>Meow</p>
```

- Editing JS in JSX (which is just JS)
  - right where you would put it
- notice function is an actual function value

Only HTML elements can get events

Events don't happen to Components

- but you can pass props
- component can apply to returned element

```
function Meow({ onClick }) {
  return (
    <p onClick={onClick}>Meow</p>
  );
}

export default Meow;
```

Summary - Components

Components:

- Functions that return HTML/JSX
    - or class-based component
- Can be nested
- passed "props"
- must have a single parent element
    - or be "fragment"
- must be named in MixedCase
- FOR THIS COURSE:
    - 1 component per `.jsx` file (must be `.jsx`)
    - Filename matches component name

Summary - imports/exports

- A component can be exported from a file
- A component can be imported from an export
- A CSS file can be imported
  - many options on how to organize/approach
  - CSS imports do not need to be in all component
- An image path can be imported
- All imports need a path

FOR THIS COURSE:

- CSS classes should be `kebab-case`

Summary - props

Components have "props" passed in JSX

- received in "props" object passed to JS function
    - often destructured to named variables
- props can hold string or non-string values
- event handler props don't work on components
    - but can be put on HTML elements

Summary - event handlers

Event handlers go on HTML tags in JSX

- Looks like HTML JS attributes
    - but aren't
- must be `onEVENT` syntax
    - EVENT is a MixedCase event name
    - e.g. `onClick`, `onInput`, `onChange`
- event handler props don't work on components
    - but can be put on HTML elements