Intro to React - Summer of JS Series

Women Who Code Cincinnati

https://github.com/WomenWhoCodeCincy/2018-july-react-intro

Kat Fairbanks

Github - @katzenbar

Twitter - @DerKatzenbar

Before We Get Started...

- This talk is focused on the *concepts* needed to program in React.
- You do not need to understand every line (or most lines) of the code.
- Understanding the code will come with following tutorials.

What is React?

> React is a library for building composable user interfaces. It encourages the creation of reusable UI components which present data that changes over time.

— *https://reactjs.org/blog/2013/06/05/why-react.html*

What is React?

- Created by Facebook engineers
- Facebook Ads was a very difficult, high-risk project to maintain and update
- When Instagram was acquired by Facebook, React had to be separated from the Facebook ecosystem. This eventually led to the framework being open-sourced.
- https://stackshare.io/posts/the-react-story

What is React? — Component-Based

> Build encapsulated components that manage their own state, then compose them to make complex UIs.

— *https://reactjs.org*

What is React?

```
class Card extends Component {
  render() {
    return (
      <div className="card">
        <div className="card__header">
          {this.props.header}
        </div>
        <div className="card__content">
          {this.props.content}
        </div>
      </div>
    );
  }
}
```

What is React?

```
class LatestHeadlines extends Component {
  state = [
    { headline: "...", blurb: "..." },
    { headline: "...", blurb: "..." },
  ]

  render() {
    return (
      <div className="headlines">
        { state.map(
          (article) =>
            <Card
              header={article.headline}
              content={article.blurb}
            />
        ) }
      </div>
    );
  }
}
```

What is React? — Declarative

> Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

— *https://reactjs.org*

Templates — The usual story

In most frameworks, normally you would write an HTML file with some template tags.

View:

```
{
  "stooges": [
    { "name": "Moe" },
    { "name": "Larry" },
    { "name": "Curly" }
  ]
}
```

Template:

```
{{#stooges}}
<b>{{name}}</b>
{{/stooges}}
```

Output:

```
<b>Moe</b>
<b>Larry</b>
<b>Curly</b>
```

What is React? — Declarative

> Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

— *https://reactjs.org*

JSX

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);
```

JSX

> Instead of artificially **separating technologies** by putting markup and logic in separate files, React **separates concerns** with loosely coupled units called “components” that contain both.

JSX

```
class LatestHeadlines extends Component {
  state = [
    { headline: "...", blurb: "..." },
    { headline: "...", blurb: "..." },
  ]

  render() {
    return (
      <div className="headlines">
        { state.map(
          (article) =>
            <Card
              header={article.headline}
              content={article.blurb}
            />
        ) }
      </div>
    );
  }
}
```

JSX — Almost the same as HTML

`class` ⇒ `className`

HTML

```
<div class="button"></div>
```

JSX

```
<div className="button" />
```

JSX — Almost the same as HTML

All tags must be closed in JSX

HTML

```
<img src="cats.png">
```

JSX

```
<img src="cats.png" />
```

JSX — Almost the same as HTML

BUT WAIT THERE'S MORE!

You can embed Javascript expressions into your HTML with JSX

Quick Detour - Javascript Expressions

An expression is any valid unit of code that resolves to a value. (from MDN)

Good

```
2 + 12 * 16
```

```
"Hello"
```

```
[1, 2, 3].map(x => x * x)
```

Bad

```
var x = "Hello"
```

JSX — Using Expressions

JSX interprets text inside of `{curly brackets}` as Javascript code.

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);
```

JSX — Rendering lists

JSX expressions can also be lists.

Input

```
const list = ['apples', 'oranges', 'bananas'];

const element = (
  <div>{list}</div>
);
```

Result

```
<div>applesorangesbananas</div>
```

JSX — Rendering lists

It is usually more appropriate to return an array of JSX elements (we usually call a JSX element a **Node**).

Input

```
const listAsElements = list.map((item) => {
  return <li>{item}</li>;
});

const element = <ul>{listAsElements}</ul>;
```

Output

```
<ul>
  <li>apples</li>
  <li>oranges</li>
  <li>bananas</li>
</ul>
```

JSX — Rendering lists

We usually do not compute these lists outside of the templates. It is easier to see the whole picture if you calculate it inline like this.

```
const element = (
  <ul>
    {
      list.map((item) => {
        return <li>{item}</li>;
      })
    }
  </ul>
);
```

JSX — Rendering lists

React expects each Node in a list to have a unique `key` attribute, to help it with rendering efficiently.

```
const element = (
  <ul>
    {
      list.map((item) => {
        return <li key={item}>{item}</li>;
      })
    }
  </ul>
);
```

Getting Back to React

Actually JSX is a large part of understanding how to write simple React code! But there are a few more concepts to learn to cover the basics.

Making JSX Reusable — Components

We organize our JSX templates into something React calls Components.

```
import React, { Component } from 'react';

class HelloWorld extends Component {
  render() {
    return <div>Hello, World!</div>;
  }
}
```

Props — Passing Data into Components

The syntax we use for HTML attributes becomes even more powerful in React. You can not only pass basic data into components, but you can also pass Javascript objects and functions.

```
<li key={item}>{item}</li>
```

Props — Passing Data into Components

Assume we have a `Card` component. We can pass props in like this:

```
const articleContent = """
There is an exciting group of women programming
right this moment.
""";

<Card
  heading="Ladies are programming"
  content={articleContent}
/>
```

Props — Passing Data into Components

In the `Card` component, props that are passed in can be accessed through `this.props`.

```
class Card extends Component {
  render() {
    return (
      <div className="card">
        <div className="card__header">
          {this.props.header}
        </div>
        <div className="card__content">
          {this.props.content}
        </div>
      </div>
    );
  }
}
```

Props — Passing Data into Components

Functions can be passed into components and elements, to be called by that Node.

```
class Counter extends Component {
  incrementCount = () => {
    console.log("Increment!")
  }

  render() {
    return (
      <div>
        <div>Count: {this.props.count}</div>
        <button onClick={this.incrementCount}>
          Increment
        </button>
      </div>
    );
  }
}
```

Props — Immutable

To implement `incrementCount` on the last slide, you might be tempted to try this.

```
class Counter extends Component {
  incrementCount = () => {
    this.props.count += 1;
  }
  ...
}
```

And this might make a change, but it would only be until React renders the DOM again. Props are *immutable*.

State — Mutable Data for Components

Mutating data is necessary to build an interactive application. React lets you do this though something called `state`.

```
class Counter extends Component {
  state = {
    count: 0,
  }
  ...
}
```

State — Mutable Data for Components

So since `state` is mutable, can we do this?

```
class Counter extends Component {
  incrementCount = () => {
    this.state.count += 1
  }
  ...
}
```

Not quite. React wants a message to know that you have updated the state, so it can re-render at the correct time.

State — Mutable Data for Components

State can be modified through a function on the component instance, `this.setState`.

```
class Counter extends Component {
  state = {
    count: 0,
  };

  incrementCount = () => {
    this.setState({
      count: this.state.count + 1,
    });
  }

  ...
}
```

State — Mutable Data for Components

If state is an object, you only have to pass in the keys you want to update.

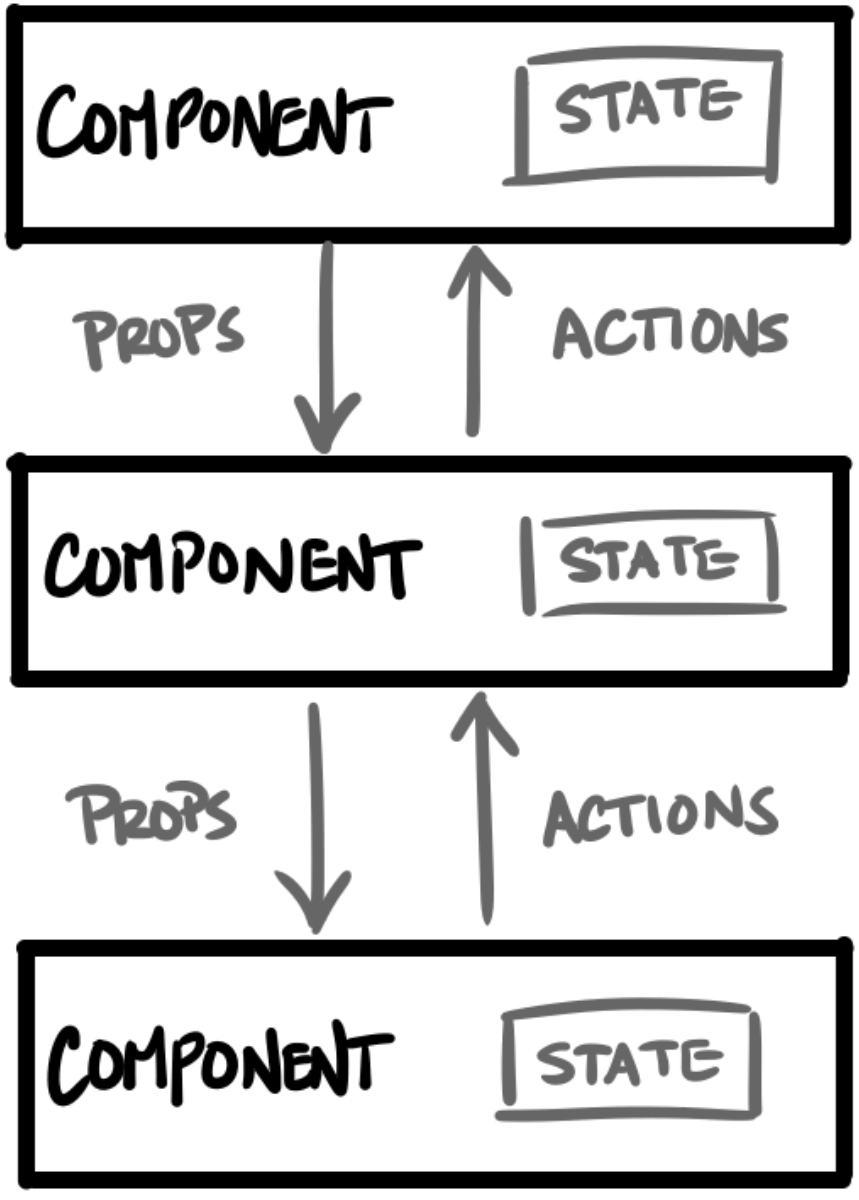
```
state = {
  count: 0,
  incrementCount: 0,
  decrementCount: 0,
};

incrementCount = () => {
  this.setState({
    count: this.state.count + 1,
    incrementCount: this.state.incrementCount + 1,
  });
}
```

State — A Double-Edged Sword

When building applications, it is a best-practice to keep state in the upper-most components. If you don't, you often end up moving state higher and higher up as needed, or find yourself without access to the data you want.

So how do we change state from the components we use?



DATA GOES DOWN

(THROUGH PROPS)

ACTIONS COME UP

(TO MODIFY STATE)

Actions Up

As an example, let's take the counter button and make it into a component.

```
class Button extends Component {
  render() {
    return (
      <button onClick={this.props.onClick}>
        {this.props.label}
      </button>
    );
  }
}
```

Actions Up

```
class Counter extends Component {
  ...
  render() {
    return (
      <div>
        <div>Count: {this.state.count}</div>
        <Button
          onClick={this.incrementCount}
          label="Increment"
        />
      </div>
    );
  }
}
```

A Real-World Example

Let's take a look at how I have been using React at work, and break down why React and Components have been helpful.

Handbuilt Porcelain Tea Se...

CURRENT BID:
$48

FOLLOW

2 HOURS LEFT

Vintage Hand-Painted Hob...

CURRENT BID:
$0

FOLLOW

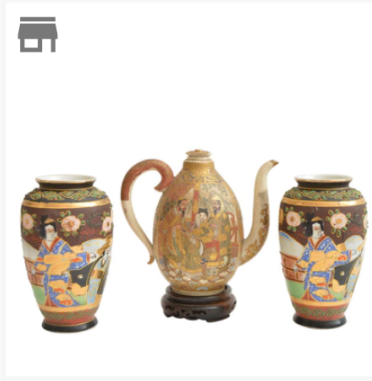
3 DAYS LEFT

Hand-Painted Coimbra Port...

CURRENT BID:
$40

FOLLOW

104 MINUTES LEFT



Vintage Satsuma Japanese...

CURRENT BID:
$15

FOLLOW

1 DAY LEFT

Vintage Japanese Lusterw...

CURRENT BID:
$1

FOLLOW

3 DAYS LEFT

Royal Limoge, "Iena" Tea Set

CURRENT BID:
$12

FOLLOW

2 DAYS LEFT

Vintage Crystal, Silver Plat...

CURRENT BID:
$36

FOLLOW

1 DAY LEFT

Porcelain Teapot and Chin...

CURRENT BID:
$10

FOLLOW

2 DAYS LEFT

Hand-Painted Coimbra Port...

CURRENT BID:
$51

FOLLOW ♡

● 76 MINUTES LEFT

Vintage Satsuma Japanese...

CURRENT BID:
$15

FOLLOW ♡

● 1 DAY LEFT

CONDITIONALLY RENDERED
IF PICKUP AVAILABLE

Vintage Satsuma Japanese...

CURRENT BID:
$15

FOLLOW

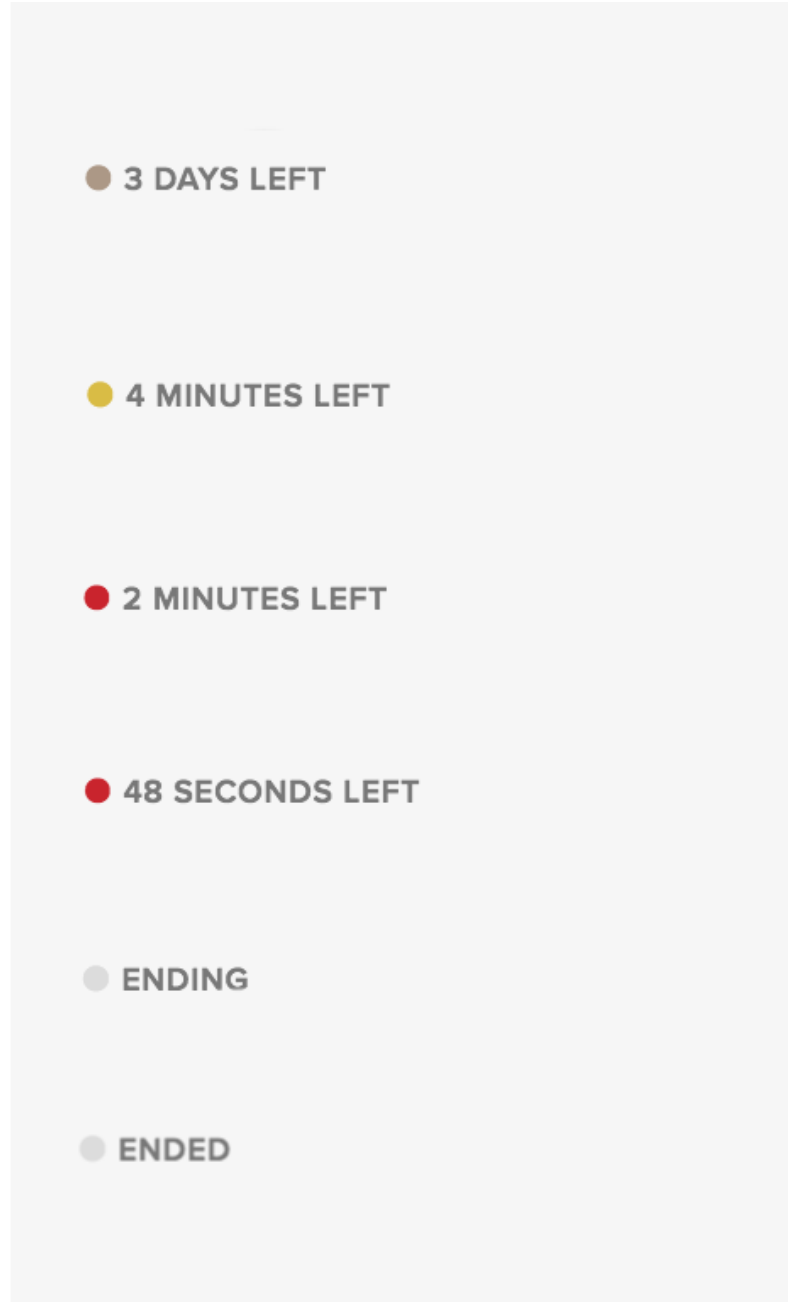
1 DAY LEFT

CURRENT BID IS UPDATED
LIVE USING WEB SOCKETS

FOLLOW BUTTON SENDS A
REQUEST TO THE SERVER

REMAINING TIME IS A COMPONENT!

- DOT COLOR
  - DAYS = TAN
  - < DAY = RED
  - EXTENDED = YELLOW
  - ENDED = GREY
- UNIT OF TIME
- COUNTS DOWN SECONDS, GOES TO ENDING



EBTH

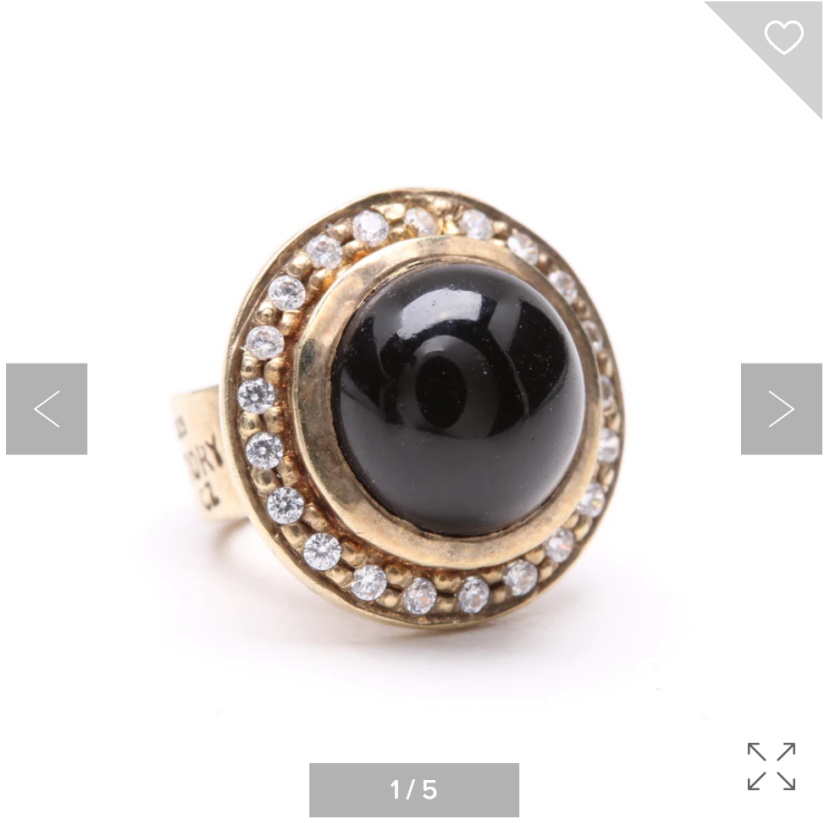
EVERYTHING BUT THE HOUSE

THE PREMIER ESTATE SALE MARKETPLACE

Help | Sell With Us | Following (0) | My Info

Search our collection of unique

CATEGORIES | SALES | ENDING | POPULAR | BOUTIQUES | FEATURED



1 / 5

View all items from Fine Jewelry, Watches & More sale

Story Gold Wash on Sterling Silver Black Onyx and Cubic Zirconia Charm

38 SECONDS LEFT

July 17th 2018 @ 7:33pm EST

CURRENT BID

$30 13 Bids

PLACE BID

DELIVERY OPTIONS More Info

Ship To Me $11.53

Deliver to **45039** EDIT ZIP

ITEM DETAILS

Jewelry Type:	Charms

EBTH

EVERYTHING BUT THE HOUSE

THE PREMIER ESTATE SALE MARKETPLACE

Help | Sell With Us | Following (0) | My Info

Search our collection of unique

CATEGORIES | SALES | ENDING | POPULAR | BOUTIQUES | FEATURED

129 ITEMS SALE #18DCC437

Fine Jewelry, Watches & More

ENDING TUESDAY, JULY 17TH 2018 @ 7:30PM

FOLLOW THIS SALE | ADD TO CALENDAR

TERMS & CONDITIONS

All Categories

Sale Items 129 Items

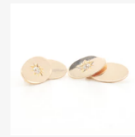
FILTER




CINCINNATI, OH

Antiques, Collectibles, Decor and More

● 4 DAYS LEFT

ASHBURN, VA

Collectibles, Toys, Home Furnishings

● 4 DAYS LEFT





BLUE ASH, OH

Art, Home Furnishings, Décor & More

● 6 DAYS LEFT





NORTHBROOK, IL

Art, Collectibles, Designer Accessories & More

● 4 DAYS LEFT





CHARLOTTE, NC

Art, Designer Accessories, Sterling Silver & More

● 4 DAYS LEFT

Next Steps — Recommended Reading

- https://reactjs.org/tutorial/tutorial.html
  - The offical ReactJS tutorial
- http://buildwithreact.com/tutorial
  - A good tutorial that explains these concepts in a really understandable way
- https://egghead.io/courses/the-beginner-s-guide-to-react
  - If you prefer videos, there is free material available through egghead.io.

Next Steps

Common Libraries

- Redux - https://redux.js.org/
- React Router - https://reacttraining.com/react-router/
- Apollo - https://www.apollographql.com/

Awesome Tools

- Gatsby - https://www.gatsbyjs.org/
- Storybook - https://storybook.js.org/