CS 4350: Fundamentals of Software Engineering
CS 5500: Foundations of Software Engineering

Lesson 6.2 Introduction to “React”

Jon Bell, John Boyland, Mitch Wand

Khoury College of Computer Sciences

Topic for this Lesson

- React/JS: Front-End Framework
  - Created by Facebook; released to open-source.
- Describe architecture and big ideas.

- https://reactjs.org/

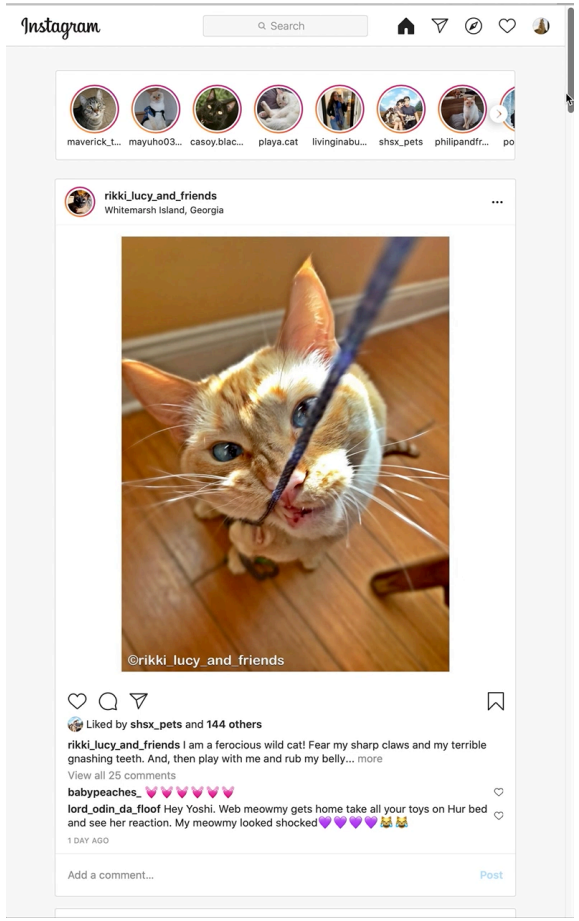
Learning Objectives for this Lesson

- By the end of this lesson, you should be able to:
  - Explain how component reuse simplifies application development;
  - Describe the three key ideas of the React framework.

HTML: Markup Language of the Web

- Language for describing structure of a document:
  - Denotes hierarchy of elements.
- What might be elements in this document?

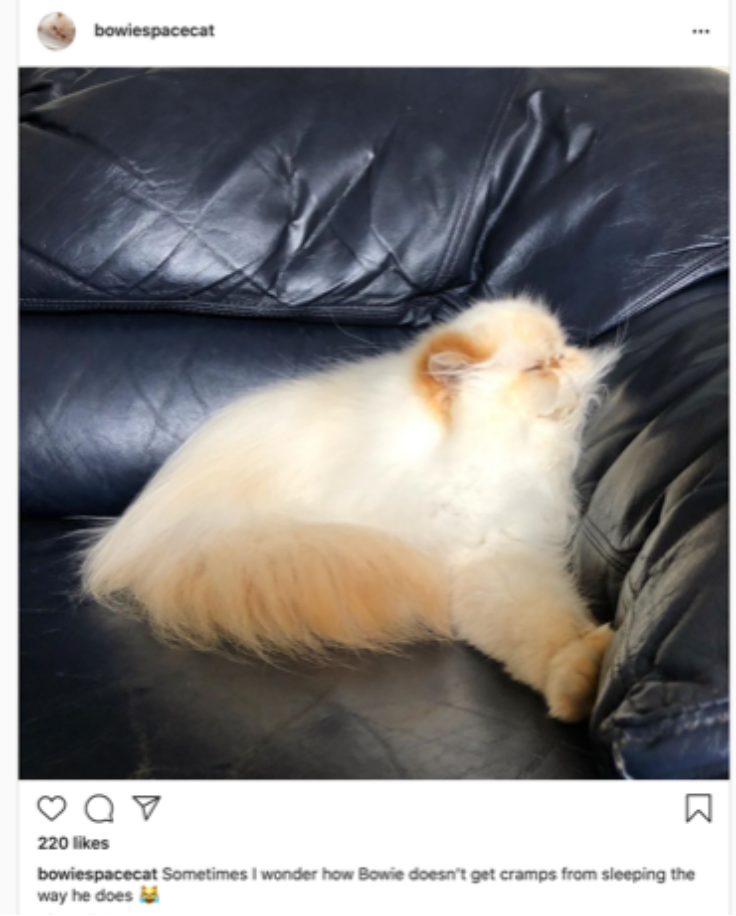
Rich Interactive Web Applications



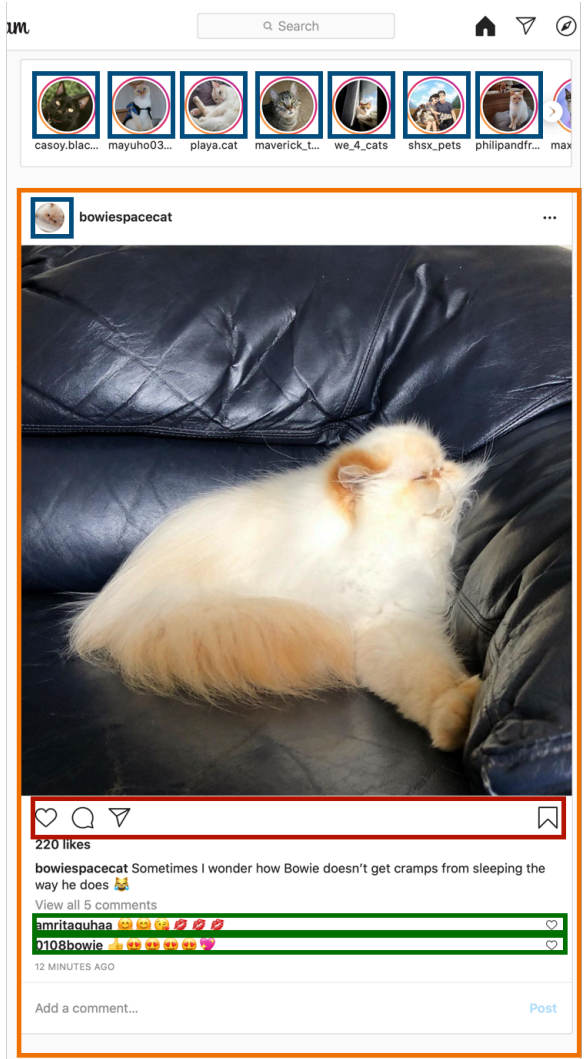
- Not just static HTML
- Infinite scrolling of cat photos.
- In video, more photos are “loaded” when we get near the bottom.

Widgets in Web UIs

- Each widget has both visual presentation & logic
  - e.g., clicking on like button executes logic related to the containing widget
  - Logic and presentation of individual widgets are strongly related,
- Widgets often occur more than once
  - e.g., comment/like widgets
- Changes to data should cause changes to widget
  - e.g., new images, new comments should show up in real time



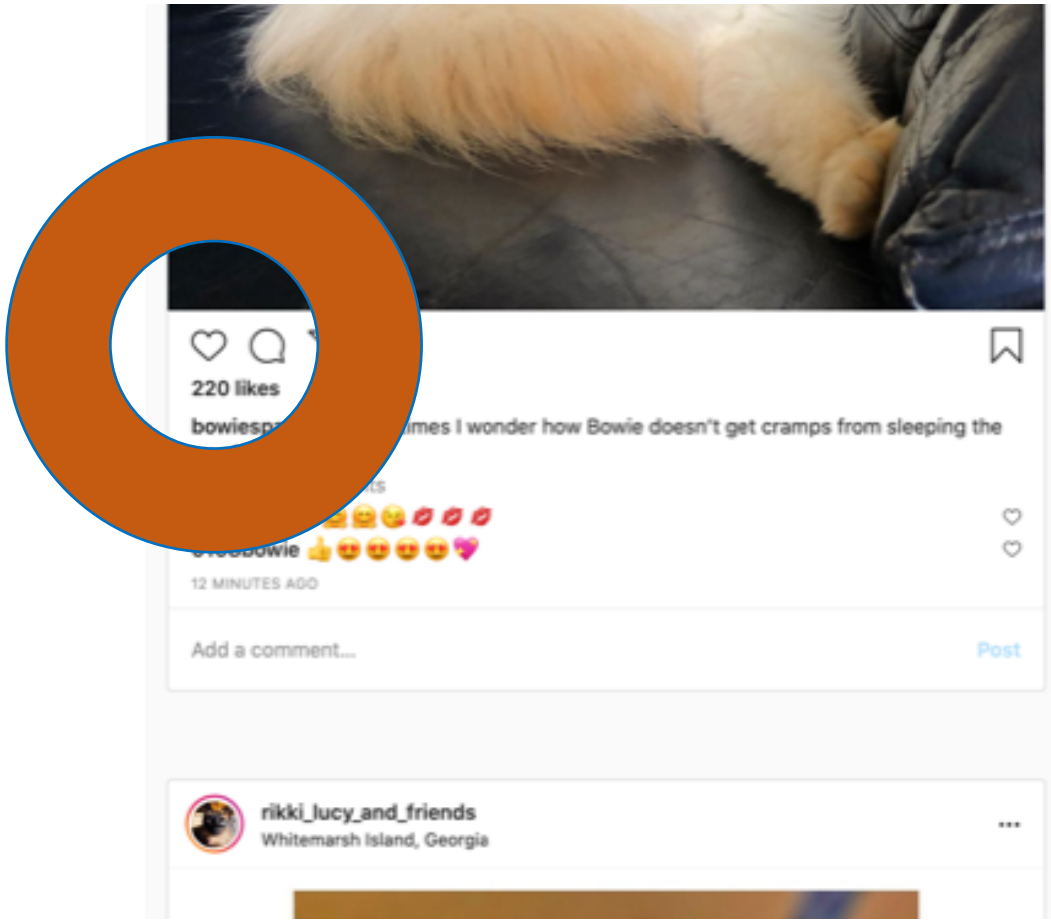
Key Idea: Components



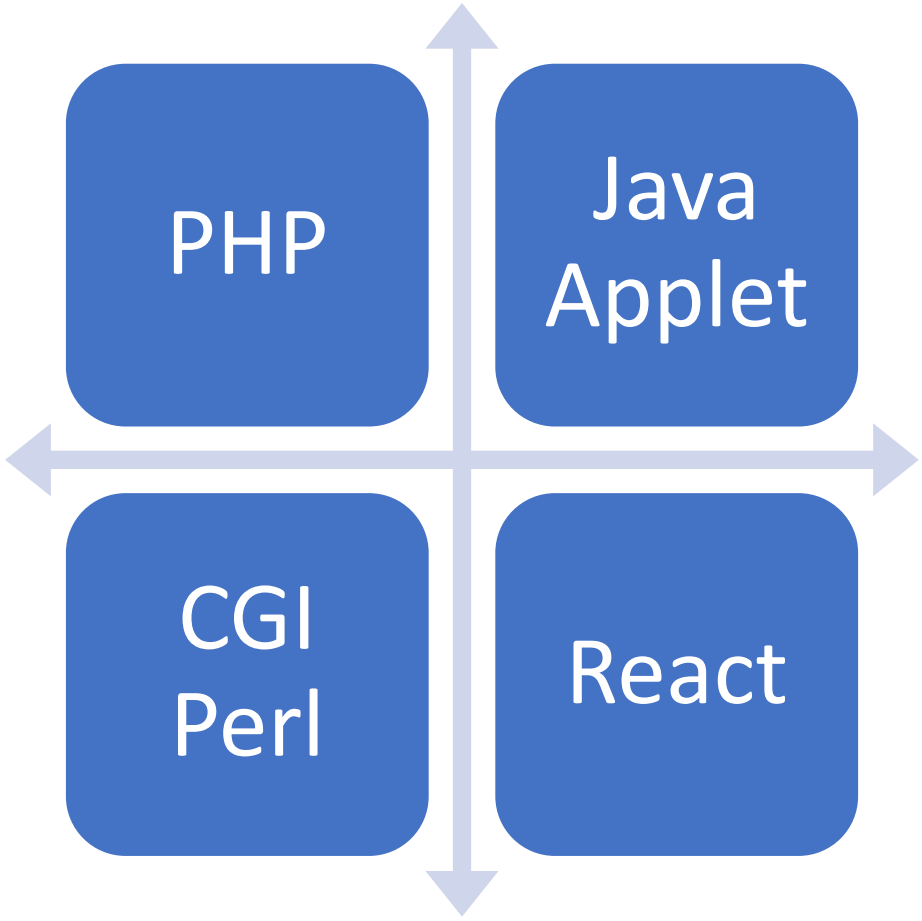
- Organize related logic and presentation into a single unit
  - Includes necessary *state* and the logic for updating this state
  - Includes presentation for *rendering* this state into HTML
- Synchronizes state and visual presentation
  - Whenever state changes, HTML should be rendered again

“Like” Button Component

- What does it keep track of?
  - Is it liked or not?
  - What post is it associated with?
- What logic does the button have?
  - When changing “like” status, send update to server.
- How does the button look?
  - Filled in if liked, hollow if not.
- Problem: how do we automatically update the button to look filled in when it’s liked?



Design Architecture Possibilities



Code embedded in HTML

HTML embedded in Code

Server

Client

Embedding

Code in HTML

```
<p>Counting to three:</p>
<% for (int i=1; i<4; i++) { %>
 <p>This number is <%= i %>.</p>
<% } %>
<p>OK.</p>
```

- Convenient, but …
- Code infeasible to statically check (it is broken up in different HTML comments).

HTML in Code

```
return “<p> Items:” + is +
  “\n<b>Total: “ + total +
  “</b></p>\n”;
```

- Code has primacy (and can be checked).
- Creation of HTML is error-prone.

Where Does Code Run?

On Server (back end)

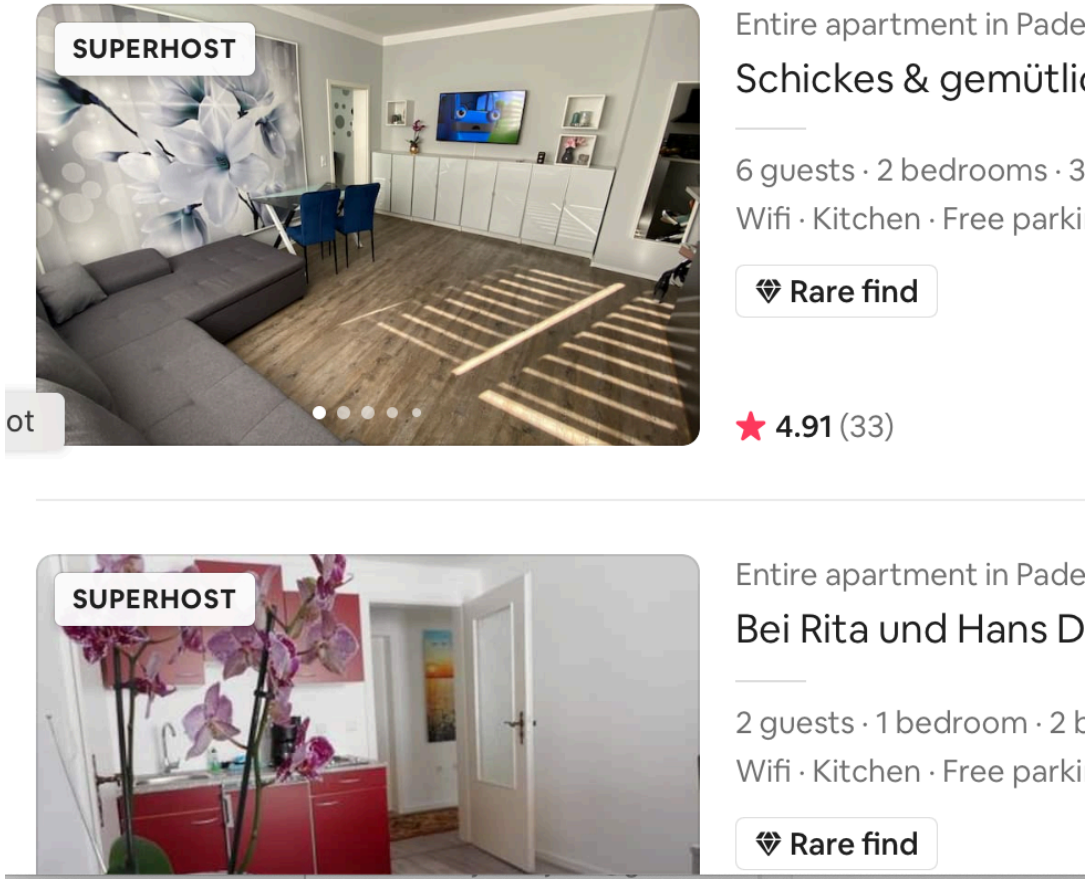
- If it runs on the server, we have full control of the HTML generated and can (in principle) use private state.
- But we have no control on the rendering process for the HTML:
  - Incrementality is on client.
- And have to push changes to client.

On Client (front end)

- If on the client, the code runs in a variety of (perhaps adversarial) contexts,
- But we can control incrementality.

React: Front End Framework for Components

- Key concepts:
  - Embed HTML in JavaScript;
  - Track application “state”;
  - Automatically and efficiently re-render page in browser based on changes to state.
- React developed by Facebook:
  - Also used in airbnb, Uber, Pinterest, Netflix, Twitter and 8855 more



Embed HTML in JavaScript/TypeScript

```
return <div>Hello {person.name}</div>;
```

- Can create HTML by using HTML syntax:
  - Inside braces { … } we can put arbitrary code, the result of which will be converted to a string in the HTML.
  - All open tags must be closed (as in XML).
- Can create components with Capitalized tags:

```
return <Card> <p>Adriel</p> </Card>;
```

  - Here “Card” is a user-defined component.
- Syntax is transpiled back to JavaScript (as is TS).

Example Component Definition

```
import React from 'react';
export interface GreetOpts {
  name : string;
}
export const Greet =
    (opts : GreetOpts) => {
  return <p>
    Hello {opts.name},
    nice to meet you!
  </p>;
}
```

- This code defines how to render

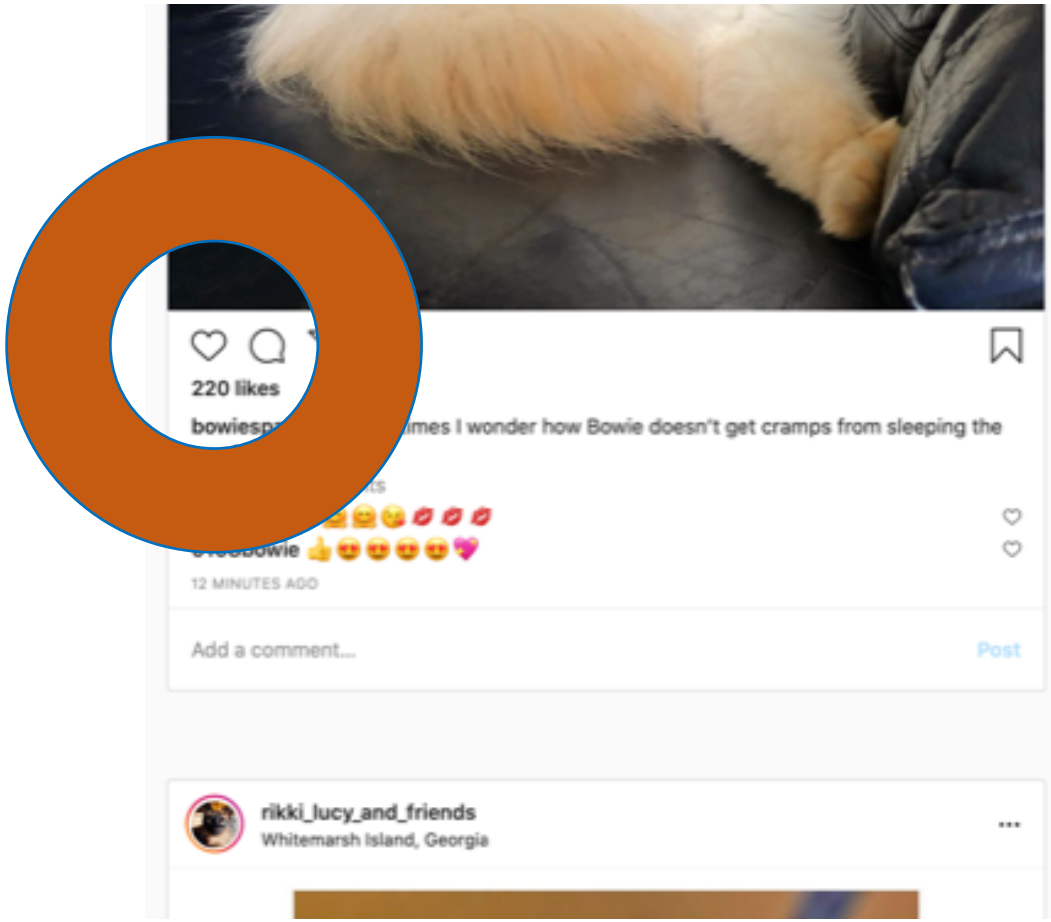
  `<Greet name="Chris"/>`

- Each component needs own file.
- If it has properties, export an interface defining them.
- Component defined as a function taking properties and returning HTML.
- Properties are immutable.

Components can also be implemented with classes.

State vs. Properties

- State **changes** to reflect the current state of the component.
  - Can (and should) change based on the current data of component.
- A ”like” button keeps track of:
  - Is it liked or not (**state**)
  - What post this is associated with (**property**)
- If component is a function, how do we represent the state?



Hooks Give Access to State

- Replace the body of the function with:

```
const [formal, setFormal] =
  useState(true);
if (formal) {
  return <p>Hello, {opts.name},
    how do you do?</p>;
} else {
  return <p>Hi, {opts.name},
     what's up?</p>;
}
```

Warning: The setter is currently unused!

- The "useState" function …
  - … declares a state variable, …
  - … with an initial value.
- The "useState" function returns an array of two values:
  1. The current value;
  2. A setter taking a new value.
- Each time you call it, you get a new state variable.
  - Only call at top level of function!

Reacting to change

- How does the greeting update?
  1. If the setter is called, the function is invoked again by framework.
  2. Then the framework *diffs* output of render with *previous* call to render, updating only that part of DOM (Document Object Model) that *changed.*
- The last step, “reconciliation,” is a key idea of React.

Reconciliation: Efficient Update

- React updates the DOM (HTML) each time the components change.
- Basically, change is based on order of components
  - Second child of Card is destroyed.
  - First child of Card has text mutated.

- Before:

```
<Card>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
</Card>
```

- After:

```
<Card>
  <p>Paragraph 2</p>
</Card>
```

Reconciliation is much more complicated.

Review: Learning Objectives for this Lesson

- You should now be able to:
  - Explain how component reuse simplifies application development;
  - Describe the three key ideas of the React framework.

Looking ahead

- The next part of Lesson 6 includes a tutorial building a simple TODO app in React.