ECS 162
WEB PROGRAMMING

FAQ

- Can we use Create React App (or, fill in tool here)?

Sure, but the assignment you hand in must be a Web server that you wrote using Node and Express, not (for instance) the Web server built into Create React App. So you could use CRA to fine-tune your .jsx files, but then you need to make sure you can turn those into a lango.js file (or whatever) that will be served by your node+express static server.

React main ideas

- Create virtual HTML tags to encapsulate widgets
- Programming illusion: DOM is re-constructed, from scratch, in software, whenever it changes
- Compile JSX into Javascript for nice programmer interface

Calling the compiler

- Installation instructions (from React documentation):

  npm init –y

  npm install babel-cli@6 babel-preset-react-app@3

- I put this command into my ~/.profile file on server (run when UNIX shell is started) (all one line):

  alias compile="npx babel --presets react-app/prod --extensions ".jsx" . --out-dir ."

- Run compiler by typing "compile" on the command line.

Component with children

```
function Card(props) {
    return <div className="textCard">
        {props.children}
    </div>;
    }
```

- React puts the children of the component in the JSX into props.children.

Using component with children

```
const main = (
      <main>
      <Card>
        <textarea onKeyPress={checkReturn} />
      </Card>
      <Card>
        <p>What she said!</p>
      </Card>
      </main>   );
```

Properties

- JSX components are functions. How to give them arguments?
- For instance, we'll want to print translated phrases on cards, and we can't hard-code them in.

```
function Txt(props) {
        return <p>{props.phrase}</p>
        }
```

Properties

```
function Txt(props) {
        return <p>{props.phrase}</p>
        }
```

- Component takes one input argument, the object props.
- props.children was a built-in object property.
- Here we use a new object property, that we made up, props phrase.

Assigning values to properties

```
<Card>
        <Txt phrase="My dog has fleas"/>
 </Card>
```

- Here we assign a value to props.phrase.
- This is where props.phrase is initialized, it is used when the function Txt is called.

Using a variable

```
const opinion = "It's a wonderful world";
...
<Card>
        <Txt phrase={opinion} />
</Card>
```

- Here we have a normal Javascript variable containing a string.
- It is evaluated by the JSX compiler when it is in {}.

Properties

- props are evaluated when a component is rendered.
- What if a prop is undefined?
- We can put any code into component functions, so we can add code to check for that.
- We just have to make sure a component function always returns some valid DOM elements.

Properties

```
function Txt(props) {
        if (props.phrase == undefined)
           { return <p>Text missing</p>; }
        else return <p>{props.phrase}</p>;
        }
```

This does **not** work

- Error when run by browser!

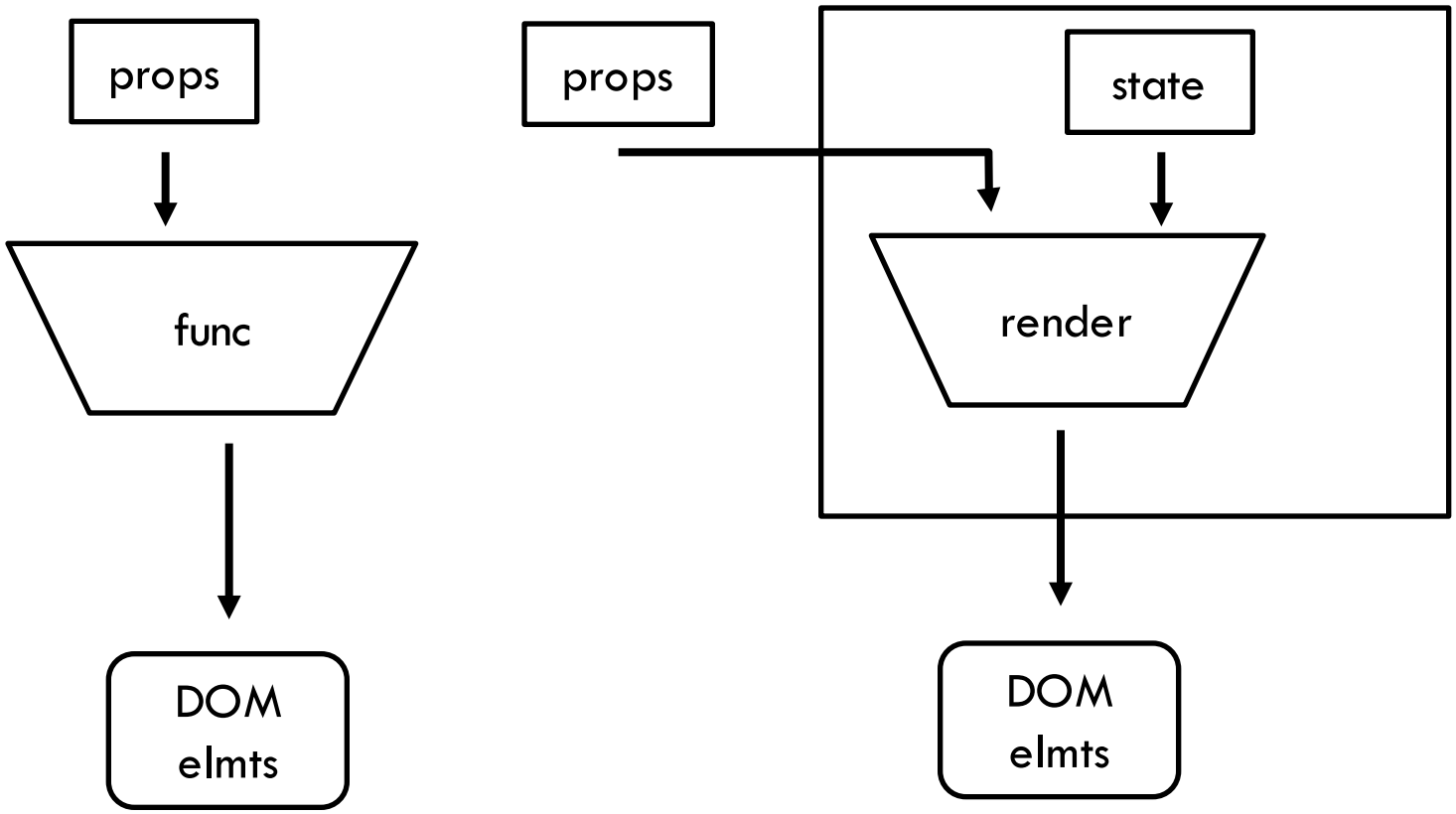
```
function Txt(props) {
        if (props.phrase == undefined) {
          props.phrase = "Text missing";
          }
        else return <p>{props.phrase}</p>;
        }
```

- Props is strictly used for input to a component function. Cannot be changed inside of it! We say, it is immutable.

Storing State

- When user hits return in the textarea, we want to get the translation and display it.
- Somewhere we need to store and change the text that appears on the right.
- Could use a lot of global variables like opinion, but global variables tend to be buggy.
- In order to store state, in React we move from defining components with functions to defining components with classes (classes have data!).

Function vs Object



Class creates objects that have the render function, and also store state.

Review of classes

```
class Weather {
        constructor (t,w) {
            this.fahrenheit = t;
            this.wind = w;
            this.celsius = function() {
                return (this.fahrenheit-32)*5/9;
            }
        }
}
```

Same thing

```
class Weather3 {
        constructor (t,w) {
            this.fahrenheit = t;
            this.wind = w;
        }
        celsius () {
                return (this.fahrenheit-32)*5/9;
            }
}
```

Component as class

```
class CreateCardMain extends React.Component {
  render() { return (
        <main> …cards… </main>);
    } // end of render function
  } // end of class
```

- A re-write of our previous component.
- The render() method used to be the function.
- The .js file gets a bunch of new React stuff at the top, coming from React.Component.

State

- state for a React component is an object, just like props.
- Initialize state in a constructor function for the class.

```
class CreateCardMain extends React.Component {
   constructor(props) {
      super(props);
      this.state = { opinion: "Life is a bowl of cherries" }
    } …
```

- super(props) calls the constructor for React.Component

Using a state variable

- Still has to be evaluated using {}

```
…
    <Txt phrase={this.state.opinion} />
…
```

- Recall that “this” refers to the particular object of of this class that is running the code in the methods.

Event handler

- Rather than have event handlers as globals, make them methods as well.

```
class CreateCardMain extends React.Component {
   ....
    checkReturn(event) {
        console.log(event.charCode);
     }
```

Using event handler method

```
...
    <textarea onKeyPress={this.checkReturn} />
...
```

- Again, we need “this” since now it is a method of the object created by the class.

Using textarea data

```
checkReturn(event) {
        if (event.charCode == 13) {
           let newPhrase =
       document.getElementById("inputEng").value;
           this.setState({opinion: newPhrase} );
           }
        }
```

setState()

- The setState method of a React component…
  - adds or changes some subset of its state object's properties (not all of them!)
  - and then re-renders the component, including all its children
- So there are two ways to trigger re-render a) setState(), or b) ReactDOM.render(). Usually we use a).

How to finish?

- Instead of copying English, get translation
- checkReturn should make AJAX request.
- Callback of AJAX request should call setState()
- Where should we put the callback?

How to finish?

- Instead of copying English, get translation
- checkReturn should make AJAX request.
- Callback of AJAX request should call setState()
- Where should we put the callback?

As another method of the class. That way, it can call this.setState()

User pushes “start review”

- How do we get to the flashcard review screen?
- We don’t want to switch to a different HTML page. The HTML would be exactly the same anyway!

User pushes “start review”

- How do we get to the flashcard review screen?
- We don’t want to switch to a different HTML page. The HTML would be exactly the same anyway!

Here’s one appraoch: your .jsx file would contain two possible “main” components, “CreateCardMain” and “ReviewCardMain”. Then give it a single “WholePage” component, with a state variable that is either “create” or “review”; based on this state variable, it would render either “CreateCardMain” or “ReviewCardMain”.