CSE 331
Software Design & Implementation

Kevin Zatloukal
Spring 2020
Modern Web UIs
JS vs Java Classes

• JS method signatures are just the name
  – JS objects are just HashMaps
  – field names are the keys

• Java methods signatures are name + arg types
  – e.g., `avg(int, int)`

• JS has only one method with a given name
  – language allows different numbers of arguments
    • Missing arguments are undefined
  – can strengthen a spec by accepting a wider set of possible input types
TypeScript

- Type system is unsound
  - can’t promise to find prevent all errors
  - can be turned off at any point with any types

- More assumptions needed to guarantee no errors
  - in Java, no unchecked casts
  - more ways here to circumvent the type system
    - e.g., “eval” (see ugly hack using window obj)
TypeScript

- tsc performs type checking
- Creates version has type annotations removed
- Built into the tools provided in HW8
  - but can be used directly in your own projects
React

• Regain modularity by allowing custom tags

```javascript
let app = (
  <div>
    <TitleBar name="My App"/>
    <EditPane rows="80" />
  </div>);
```

• `TitleBar` and `EditPane` can be separate modules
  – their HTML gets substituted in these positions
Simple React Components

• First React example was not dynamic
• No need to have classes then:

```javascript
function EditPane(props) {
    return <textarea rows={props.rows} />;  
}
```
Structure of a React Application

- **Model** (data and invariants)
- **Listeners** (events)
- **HTML** (presentation)

React Components:
- Updates
- Presentation
Structure of Example React App

App

Quarter Picker

Class Picker

Props:
– quarter

State:
– classes

State:
– quarter

onPick

Quarter

onBack
Splitting the Model

- Custom tag also has its own events

- Updating data in a parent:
  - sends parent component new data via event
  - parent updates state with `setState`
  - React calls parent’s `render` to get new HTML
    - result can include new children
    - result can include changes to child props
Splitting the Model

- State should exist in the **lowest common parent** of all the components that need it
  - sent down to children via *props*

- Children change it via *events*
  - sent up to the parent so it can change its state

- Parent’s render creates new children with new props
React Gotchas

• Model must store all data necessary to generate the exact UI on the screen
  – react may call render at any time
  – must produce identical UI

• Any state in the HTML components must be mirrored in the model
  – e.g., every text field’s value must be part of some React component’s state
  – render produces
    <input type="text" value={...}>
React Gotchas

- **render** should not have side-effects
  - only *read* **this.state** in **render**

- **Never modify** **this.state**
  - use **this.setState** instead

- **Never modify** **this.props**
  - read-only information about parent’s state

- Not following these rules may introduce bugs that will be hard to catch!
React Gotchas

- `setState` does not update state instantly:

  ```javascript
  // this.state.x is 2
  this.setState({x: 3});
  console.log(this.state.x); // still 2!
  ```

- Update occurs after the event finishes processing
  - `setState` adds a new event to the queue
  - work is performed when that event is processed

- React can batch together multiple updates
Problems

Discussed a number of problems with basic JS app…

1. Lack of tool support
   – no checking of types, tags, etc.

2. No support for modularity
   – all the code and UI in a single file

3. More boilerplate
   – minimized JS file would change function names
   – need to call `btn.addEventListener` by hand
React Event Listeners

• Solves the problems of poor modularity

• Also removes an ugly hack in the earlier code

  <button onClick="PickQuarter(...)"> 
  window[“PickQuarter”] = PickQuarter

• Event listeners can be added in the natural way:

  <button onClick={this.onClick.bind(this)}> 
  <button onClick={evt => this.onClick(evt)}>
React Performance

• React re-computes the tree of HTML on state change
  – can compute a “diff” vs last version to get changes

• Surprisingly, this is not slow!
  – slow part is calls into browser methods
  – pure-JS parts are very fast in modern browsers
  – processing HTML strings is also incredibly fast
React Tools

• Use of compilers etc. means new tool set

• npm does much of the work for us
  – installs third-party libraries
  – runs the compiler(s)

• HW8-9 use create-react-app
  – nice but somewhat opaque
  – wrapper on webpack
  – often best to use webpack directly