CSE 331
Software Design & Implementation

James Wilcox & Kevin Zatloukal
Fall 2022
Modern Web GUIs
React

• Improve 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
React

• Custom tags implemented using classes

```java
class TitleBar extends React.Component {

• Attributes (name="My App") passed in props arg

• Method render produces the HTML for component

• Framework joins all the HTML into one blob
  – can update in a single call to innerHTML = ...
```
React Example

register-react/…
React Components

- Each React component renders into HTML elements

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

- React components corresponds to *portions* of the document
  - TitleBar is one subtree
  - EditPane is another subtree
  - App contains the two of those
React State

• Last example was not dynamic
  – there was no model!

• Components become dynamic by maintaining state
  – stored in fields of `this.state`
  – call `this.setState({field: value})` to update

• React will respond by calling `render` again
  – will automatically update the live HTML to match
  – will only update the parts that changed
Structure of Example React App

App
  State:
  – quarter

Quarter Picker
  Props:
  – quarter

Class Picker
  State:
  – classes
Example 5

register-react2/…
React State

• 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
Structure of Example React App

- **App**
  - State:
    - quarter
  - **onPICK**

- **Quarter Picker**
  - **onBack**
  - Props:
    - quarter
  - State:
    - classes

- **Class Picker**
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*
Remaining Problems

• Code is extremely **verbose**
  – can be improved using Lambdas

• Code is **not sufficiently modular**
  – one JS mixes data, display, interaction

• **Too much work** involved with laying out elements

• **Poor tool support**
  – No compile-time types
  – HTML is created in strings!
Event Listener Gotchas

- Recall the issue with “this” in JavaScript.
  - **do not** write `onClick={this.handleClick}`

- Three ways to do this properly:
  1. `onClick={(e) => this.handleClick(e)}`
  2. `onClick={this.handleClick.bind(this)}`
  3. Make `handleClick` a field rather than a method:
     ```javascript
     handleClick: (e) => { ... };
     ```
     Then `this.handleClick` is okay.
React setState 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
Other React Gotchas

• State 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 React component’s state
  – e.g., every text field’s value must be part of some React component’s state
  – render produces
    ```html
    <input type="text" value={...}>
    ```
Other 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 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)