

Lecture 2:

React, Props, State

Jordan Hayashi

Previous Lecture

- ES6 and beyond
- Closures
- IIFEs
- First-Class Functions
- Execution Stack
- Event Loop
- Callbacks
- Promises and Async/Await
- `this`

Classes

- Syntax introduced in ES6
- Simplifies the defining of complex objects with their own prototypes
- Classes vs instances
- Methods vs static methods vs properties
- new, constructor, extends, super

React

- Allows us to write declarative views that “react” to changes in data
- Allows us to abstract complex problems into smaller components
- Allows us to write simple code that is still performant

Imperative vs Declarative

- How vs What
- Imperative programming outlines a series of steps to get to what you want
- Declarative programming just declares what you want

Imperative: Javascript is imperative because you tell it series of steps to get what you want. Ex. create div, then appen list to it and then label etc...

Declarative: HTML is declarative



Imperative way to create guitar would be adding pegs, strings, body and then attaching it together etc..

Declarative way to create guitar would be "I want guitar and tune strings to it"

*** Check Files for code examples ***

Imperative:
"4-imperativeGuitar.js"

Declarative:
"5-declarativeGuitar.js"

By User:Martin Möller - File:Classical Guitar two views.jpg, CC BY-SA 2.0 de, <https://commons.wikimedia.org/w/index.php?curid=40474936>

React is Declarative

- Imperative vs Declarative
- The browser APIs aren't fun to work with
- React allows us to write what we want, and the library will take care of the DOM manipulation

React is Easily Componentized

- Breaking a complex problem into discrete components
- Can reuse these components
 - Consistency
 - Iteration speed
- React's declarative nature makes it easy to customize components

****Check "9-slideshowComponents.js" for an example to better understand**

React is Performant

- We write what we want and React will do the hard work
- Reconciliation - the process by which React syncs changes in app state to the DOM
 - Reconstructs the virtual DOM
 - Diffs the virtual DOM against the DOM
 - Only makes the changes needed

Writing React

- JSX
 - XML-like syntax extension of JavaScript
 - Transpiles to JavaScript
 - Lowercase tags are treated as HTML/SVG tags, uppercase are treated as custom components
- Components are just functions
 - Returns a node (something React can render, e.g. a `<div />`)
 - Receives an object of the properties that are passed to the element

Props

- Passed as an object to a component and used to compute the returned node
- Changes in these props will cause a recomputation of the returned node (“render”)
- Unlike in HTML, these can be any JS value

State

- Adds internally-managed configuration for a component
- `this.state` is a class property on the component instance
- Can only be updated by invoking `this.setState()`
 - Implemented in `React.Component`
 - `setState()` calls are batched and run asynchronously
 - Pass an object to be merged, or a function of previous state
- Changes in state also cause re-renders

todoApp.js

But why limit React to
just web?

React Native

- A framework that relies on React core
- Allows us build mobile apps using only JavaScript
 - “Learn once, write anywhere”
- Supports iOS and Android