Lecture 3: React Native

Jordan Hayashi

Previous Lecture

- Classes
- React
- Imperative vs Declarative Programming
- Props
- State
- todoApp.js
- React Native...

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

How does React Native work?

- JavaScript is bundled
  - Transpiled and minified
- Separate threads for UI, layout and JavaScript
- Communicate asynchronously through a “bridge”
  - JS thread will request UI elements to be shown
  - JS thread can be blocked and UI will still work

Differences between RN and Web

- Base components
- Style
- No browser APIs
  - CSS animations, Canvas, SVG, etc.
  - Some have been polyfilled (fetch, timers, console, etc.)
- Navigation

React Native Components

- Not globally in scope like React web components
  - Import from `'react-native'`
- `div` → `View`
- `span` → `Text`
  - All text must be wrapped by a `<Text />` tag
- `button` → `Button`
- `ScrollView`

https://facebook.github.io/react-native/docs/components-and-apis.html

Style

- React Native uses JS objects for styling
- Object keys are based on CSS properties
- Flexbox layout
  - Default to column layout
- Lengths are in unitless numbers
- `style` prop can take an array of styles
- `StyleSheet.create()`
  - Functionally the same as creating objects for style
  - Additional optimization: only sends IDs over the bridge

Event Handling

- Unlike web, not every component has every interaction
- Only a few “touchable” components
    - `Button`
    - `TouchableOpacity`, `TouchableHighlight`, `TouchableWithoutFeedback`
    - `TouchableNativeFeedback` (Android only)
- Web handlers will receive the event as an argument, but React Native handlers often receive different arguments
    - Consult the docs

Components

- Return a node (something that can be rendered)
- Represent a discrete piece of the UI
- “All React components must act like pure functions with respect to their props.”
- Two types:
  - Stateless Functional Component (SFC) a.k.a. Pure Functional Component
  - `React.Component`

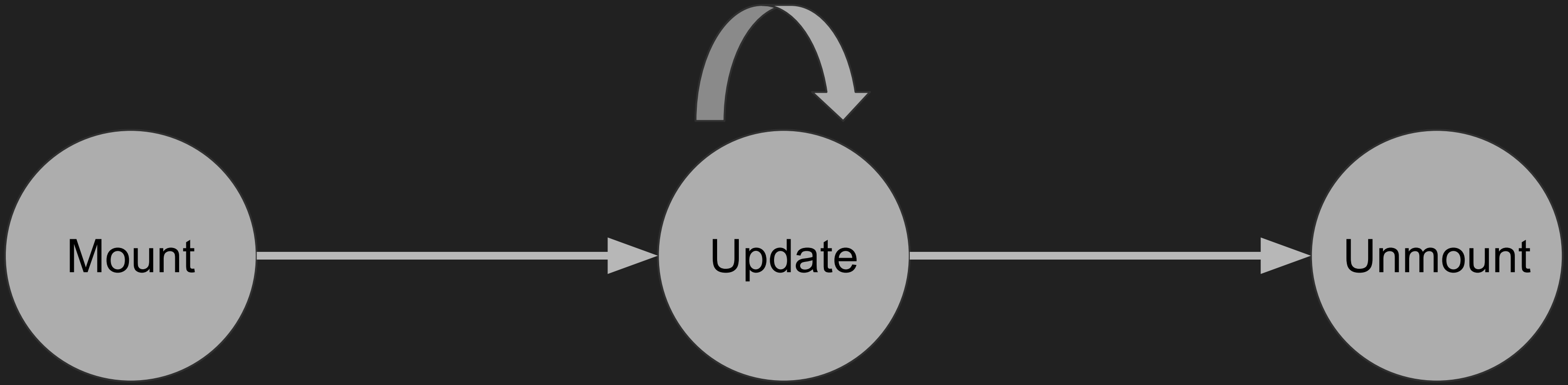
Stateless Functional Component (SFC)

- Simplest component: use when you don’t need state
- A function that takes props and returns a node
  - Should be “pure” (it should not have any side effects like setting values, updating arrays, etc.)
- Any change in props will cause the function to be re-invoked

React.Component

- An abstract class that can be extended to behave however you want
- These have additional features that SFCs don’t
  - Have instances
  - Maintain their own state
  - Have lifecycle methods (similar to hooks or event handlers) that are automatically invoked
- Rendering now becomes a function of props and class properties

Component Lifecycle



Mount

- `constructor(props)`
  - Initialize state or other class properties (bound methods, etc.)
- `render()`
  - The meat of a component
  - Return a node
- `componentDidMount()`
  - Do anything that isn’t needed for UI (async actions, timers, etc.)
  - Setting state here will cause a re-render before updating the UI

Update

- `componentWillReceiveProps(nextProps)`
  - Update any state fields that rely on props
- `shouldComponentUpdate(nextProps, nextState)`
  - Compare changed values, return true if the component should rerender
    - If returned false, the update cycle terminates
  - Almost always a premature optimization
- `render()`
- `componentDidUpdate(prevProps, prevState)`
  - Do anything that isn’t needed for UI (network requests, etc.)

Unmount

- `componentWillUnmount()`
    - Clean up
        - Remove event listeners
        - Invalidate network requests
        - Clear timeouts/intervals

Writing React Native

Expo

- “The fastest way to build an app”
- Suite of tools to accelerate the React Native development process
    - Snack - runs React Native in the browser
    - XDE - a GUI to serve, share, and publish your Expo projects
    - CLI - a command-line interface to serve, share, and publish projects
    - Client - runs your projects on your phone while developing
    - SDK - bundles and exposes cross-platform libraries and APIs

Import/Export

- Components are great for simplifying code
- We can split components into their own files
    - Helps organize project
    - Export the component from the file
- Import the component before using it in a file
- Default vs named import/export

PropTypes

- React can validate the types of component props at runtime
- Development tool that allows developers to ensure they’re passing correct props
- Helps document your components’ APIs
- Only runs in development mode

How to Read Docs

- Have a goal in mind
- See what the library/framework/API offers
- Find something that solves your problem
- Configure using the exposed API