React

Up, Up, and Away!

By Stephen Lampa & Jesse Williamson

Today's Topics

- Component Definition
- Basic types of components
- JSX
- Props / State
- Dumb / Smart components
- Lists
- CSS
- Lifecycle
- Live Demo?

Components

Definition: Essentially, a function or class that returns how elements appear on a screen.

Note: Have the advantage of splitting the UI to make items reusable and composable

Basic types of components

Functional Component

```
function NewComponent(props) {
    return <h1>Hello, {props.world}</h1>;
}
```

- Simplest way to create a component. Takes in a single argument (props) and returns JSX

Class Component

```
class NewComponent extends Component {
    render() {
        return <h1>Hello, {this.props.world}</h1>
    }
}
```

Class Component

- Require you to extend React.Component
- Require you to have a method render()
- Has access to local state
- Has access to lifecycle methods

render(): Main (required) lifecycle method

- This is how items can shown in the view
- Mainly returns JSX, strings, or null
- If using JSX, it must return only one top level element (can't have siblings side-by-side)

Differences

- Functional components are generally used for presentation
- Class components are used mainly as “containers” that contain all the logic

Importing/Exporting

- Import to use a component from another file
  - Ex. `import { SomeComponent } from './file'
- Export to allow other components to use the current component in the project
  - Ex. `export default ComponentName`
  - Ex. `export { ComponentName }` - used for multiple things being exported in a single file

JSX

What does it look like?

JSX stands for JavaScript Syntax Extension

```
let someElement = <h1>Wowwie!</h1>;
```

Is it HTML?

NO

It is very similar though...

- Is sort of its own type (kinda like regex in other languages)
- Makes use of similar properties found in HTML
  - Also has a few of its own properties
- Can be used as expressions
- Essentially compiles down into an object
- Makes use of JavaScript expressions so that you can use JavaScript inside

CSS

Pretty much the same

- JSX elements still have concept of id and class properties
- “class” needs to change to “className” (can anyone guess why?)
- Can be imported like any other file
- Styling can still happen inline

Props

Properties or props

- The most common way to pass information from one component to another
- They come in the form of an object "{ }"
- Must stay read-only
- Passed unidirectionally
- Normal data is generally sent, but functions are also common

State
Or the self-contained prop

State

- Only works in class components*
- Generally declared in a constructor
- Can be passed down to other components as props
- When state changes, a component can update

* React hooks became a thing

Constructor - Another lifecycle method

- This is where state and bind methods get initialized
  - Optional if these are not found
  - .bind(this) is there to ensure it has the component as the parent
- Must have super(props) as the first item in constructor

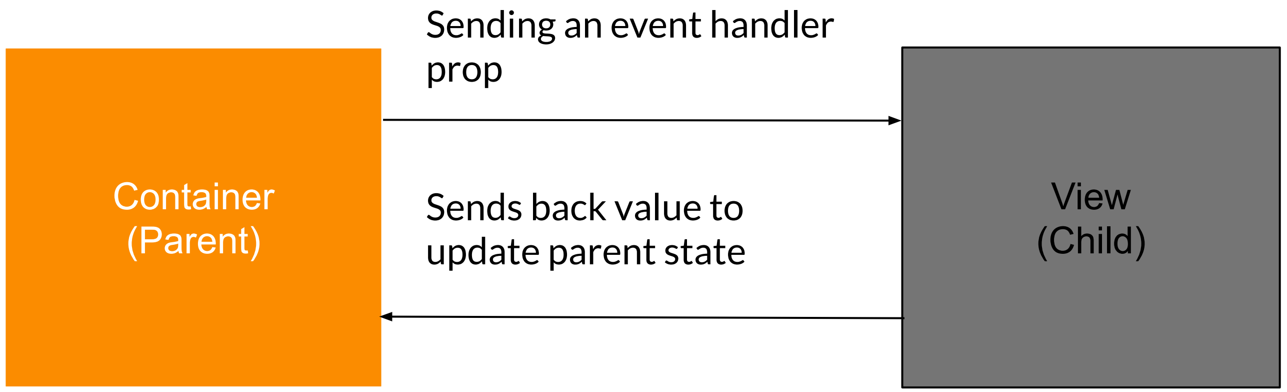
This

- Refers to, well, this
  - Refers to the object it belongs to
- In most cases this is really straightforward
- But in functions or methods, if used outside of an object, will be put in an undefined state
  - this.thing = this.thing.bind(this)
  - Alternatively, use arrow functions (lexical scope)

Event Listener/Handlers

- As the name implies, how events are handled
- React comes with a bunch of built-in events:
    - Ex. onClick, onKeyDown, onBlur, etc
    - For exhaustive list: https://reactjs.org/docs/events.html
- Takes a callback that tells how that event should be resolved

Lifting state


Sending an event handler prop
Container (Parent)
Sends back value to update parent state
View (Child)

The event handler has instructions to update the state of parent container. So when the event handler is handled, the parent becomes aware of what happens at a lower level and can change its state, which can cascade to other child (or sometimes other parent) components

Dumb/Smart
Presentational/Container
Components

Dumb/Smart Components

- Also referred to as presentational/container components (respectively)
- Dumb (presentational) only use props (if any) to display things. Do not use any real logic
- Smart (container) components carry the burden of changing/handling state and passing props to other components

Lifecycle

Lifecycle methods

- Three parts to a life of a component
  - Mounting
  - Updating
  - Unmounting
- Methods ran at its designated time in a component's life

Previous methods

- render()
- constructor()
- Best representation of the main methods: http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

componentDidMount()

- During the mounting phase, after the render, this method is called
- Good place to call fetches and other subscriptions
- Constructor should not have any side-effects due to asynchronous state behavior
- Will cause another render if state changes here

componentDidUpdate()

- During the updating phase, after the rendering and updating has happened
- Good for when a state has changed and you need to make update/patch/put/delete network calls or want to change something in the DOM based on a state change

componentWillUnmount()

- Once the component is about be gone from the view
- Just for cleaning up any lingering connections or removing event listeners

Lists

When rendering lists

- Good idea to place a unique key on each list item
- Give a component a JSX property “key”
- Helps React understand what has changed (if anything)

Questions?