Breaking up components into the controller / view pattern

Composition

Components get composed together

Allows scalability

Follows the unix philosophy

Concept of ownership

Parents “own” children

Owners set props for children

Props are immutability

Controller View

React component w/ child components

Top-level react component

Controls data flows for all child components by setting props

Also interacts w/ stores

You \_can\_ next controller views, but not recomended as it can cause multiple data flows and multiple render calls

Put markup is separate component from state management

Model state

State

Used to model data that changes over time

Should be owned by a component that can then pass that state down to children

State owner should be parent of all components that need that state

If can’t decide which component should own state, you can just create a data component and make sure it’s above all components that need it’s state.

Props

Way that parent components pass state down to children

Smart components vs dump components

Smart components get data from API’s, etc

Dumb components just get data from parents via props

## Prop validation

PropTypes

Schema for a component

Require certain props be passed in

Validate type of props

PropTypes are like asserts

They are only run in DEV

Generate nice warnings in browser console

Otherwise, you can get less clear js errors

Set expectations for a given component

An example:

propTypes: {

author: React.PropTypes.object.isRequired,

onSave: React.PropTypes.func.isRequired,

validate: React.PropTypes.func..isRequired,

errors: React.PropTypes.object,

hasErrorsr: React.PropTypes.func.isRequired

}

This is:

requiring an author object

Require a function to be called when user clicks save

Require a function to be called to validate input

Declare any errors passed into our form be of type object

Require a function to determine if any errors have occurred.

Optional types:

optionalArray: React.PropTypes.array,

optionalBool: React.PropTypes.bool,

optionalFunc: React.PropTypes.func,

optionalNumber: React.PropTypes.number,

optionalObject: React.PropTypes.object,

optionalString: React.PropTypes.string

Any prop that isn’t required should have a value assigned in getDefaultProps

var foo = React.createClass({

propTypes: {

authors: React.PropTypes.array.isRequired

}

});

Failure to pass a prop named authors of type array will lead to error in client console

[Warning] Warning: Failed propType: Required prop `authors` was not specified in `AuthorList`. Check the render method of `Authors`. (bundle.js, line 44716)

Dev vs Production mode

Prod mode is minified

Minified version runs in prod mode by default

## Best Practice

Keep variable you export and the name of the file in-synch

## Component API

this.isMounted()

Can be invoked in componentDidMount to verify component was mounted

Prop validation w/ PropTypes

## Mixins

Mixins for cross-cutting concerns