Data Entry with React

Goal

The goal of this lab is to create a data entry form and run validation logic when any changes are made.

The Mission

In this lab you will be building a movie data entry form. In this data entry form, you will let the user of the application edit the details of a previously selected movie.

You will also add validation logic to make sure all data entered is valid according to the applications business rules.

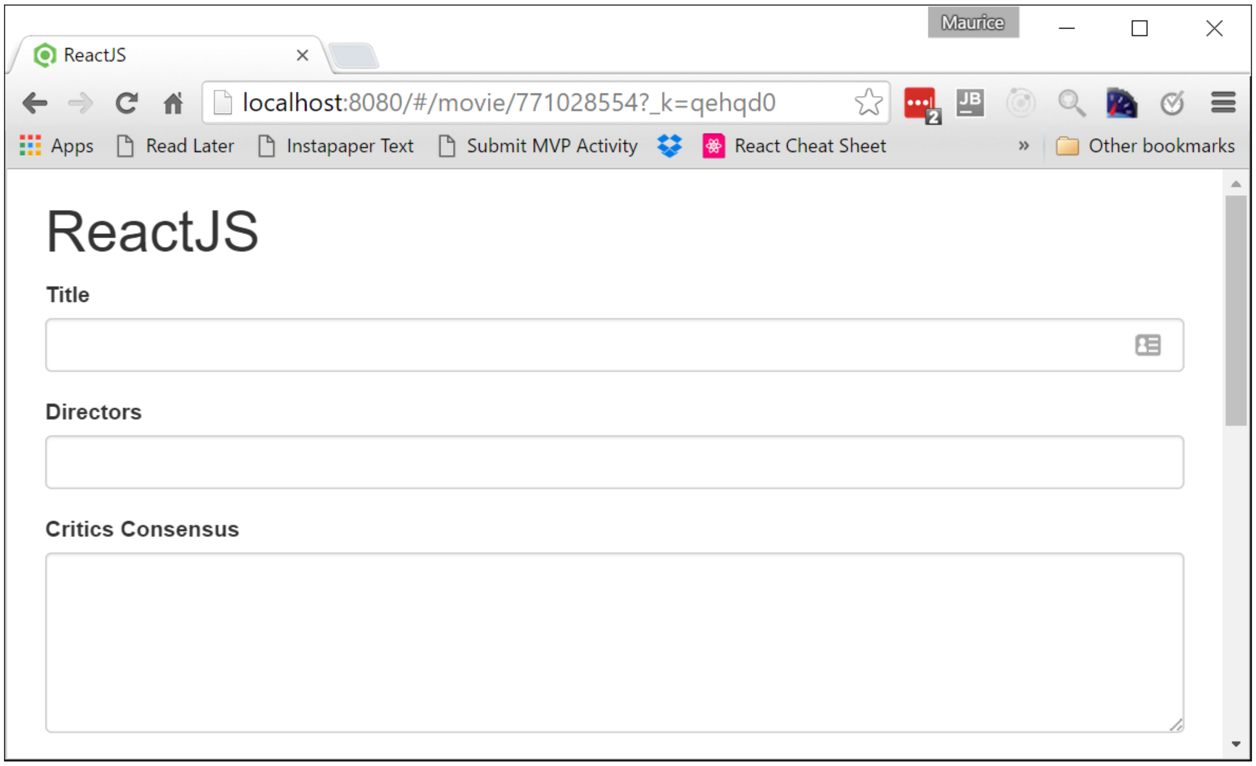
Please note that the first time you start this project it will take some time, as a number of NPM packages will download. This requires an Internet connection where the NPM registry is not blocked by a firewall.

Note: The **FINISHED** folder contains a finished version of this lab as a reference.

Type it out by hand?
Typing it drills it into your brain much better than simply copying and pasting it. Because you're forming new neuron pathways that are going to help you in the future, help them out now!

Assignment 1

Start the application by running the START.BAT file located in the in the **BEGIN** folder and observe that the main page displays a list of movies. However, when the edit button for one of these movies is clicked an empty data entry form is shown.



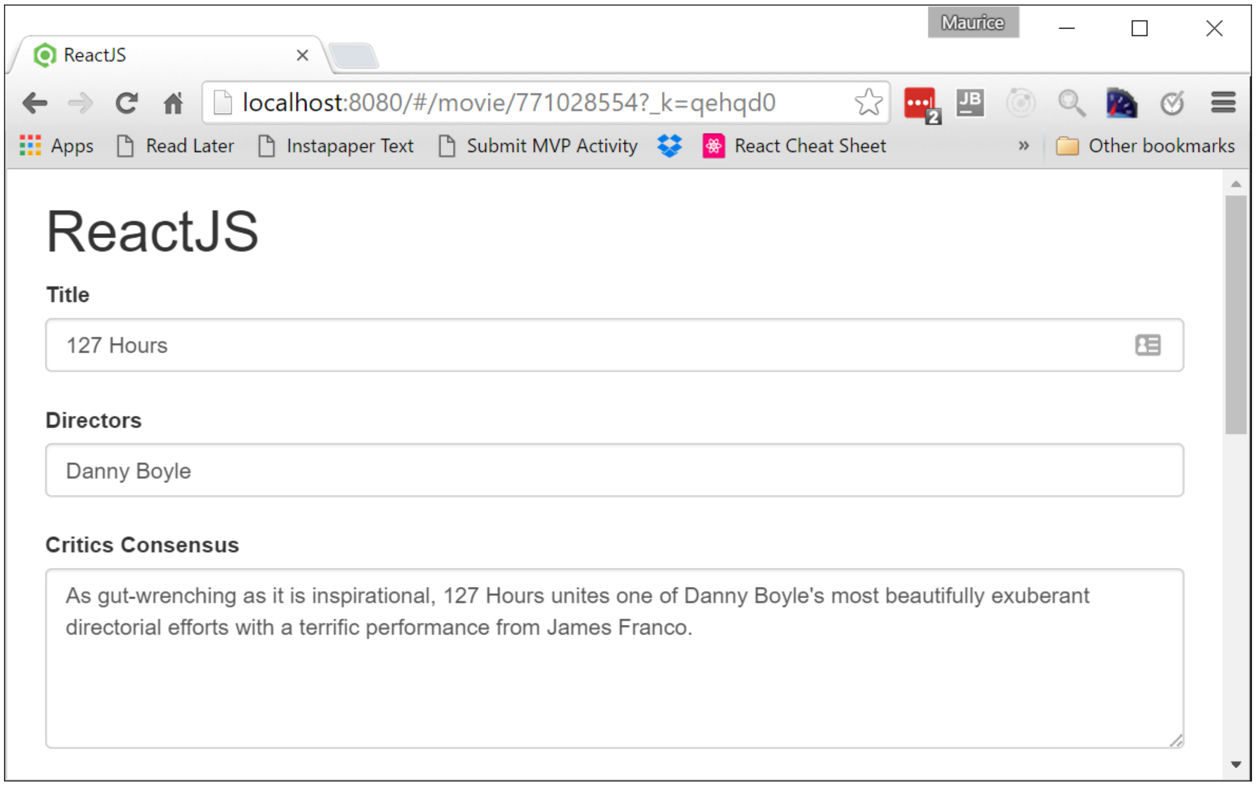
The first task is to implement the data binding to make this data entry form fully functional

Open MOVIEEDIT.JSX and notice there are a number of instances of *<InputText>* created. Each instance is passed the property name, the current value and a method to call when the value is updated. There are also two instances of *<TextArea>* with the same properties passed in. Open INPUTTEXT.JSX and notice the *onChange*, *name* and *value* props are declared but not used yet.

Update the *<InputText>* component and implement the data binding require to make these fully functional. Implement the *<InputText>* as a controlled component. Set the value property of the input element to *this.props.value*. Add an *onChange* event handler to the input element and call *this.props.onChange()* with an object containing the new value and the name of the prop being changed.

Test the application and make sure the input elements are fully functional now.

Open TEXTAREA.JSX and update the *<TextArea>* component with the same changes. With these changes in place the data entry form should be fully functional and you should be able to view/edit each movie property.



ReactJS

Title

127 Hours

Directors

Danny Boyle

Critics Consensus

As gut-wrenching as it is inspirational, 127 Hours unites one of Danny Boyle's most beautifully exuberant directorial efforts with a terrific performance from James Franco.

Assignment 2

The second task is to implement validation logic and display any reported errors.

In this task you are going to add validation logic to the movie editor. Each movie property can have zero, one or more error messages associated with it. Every time a change is made any resulting error messages need to be shown in the data entry form.

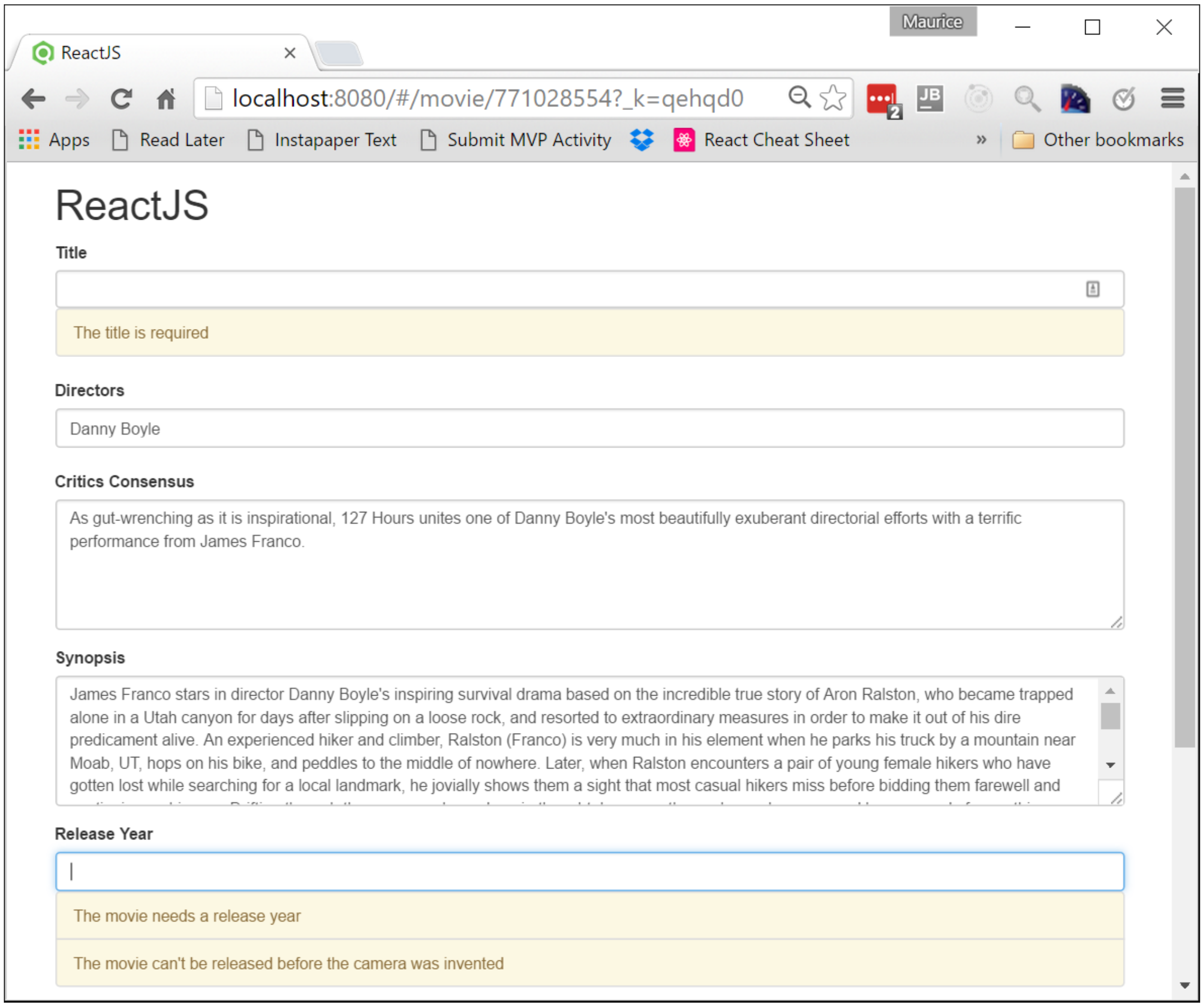
Add a new file named **MovieValidator.js** and export an object from there with a function called *validate()*. This validate function will receive two parameters, the property name being edited and the current value of that property. The function will return an array of error messages or, in the case of valid data, an empty array.

Add the following validation logic with an appropriate error message:

- *title* property:
  - The title cannot be empty
  - The title cannot be longer than 128 characters
- *year* property:
  - The movie needs a release year
  - The movie can't be released before 1845 when the camera was invented
  - The movie can't be released in a future year
- *ratings.criticsScore* and *ratings.audienceScore* properties:
  - Rating must be between 1 and 100

Open **MOVIEEDIT.JSX** and add an *errors* object to the state. This errors object will hold results of the property validation. Import the *movieValidator* and validate all changes in the *onChange()* and *onChangeRatings()* functions. Update the *errors* object with the result of the validation.

Open **INPUTTEXT.JSX** and update the *<InputText>* component to display the results of the validation. Add an *errors* array to the *propTypes* definition. Add a *defaultProps* definition so the errors default to an empty array if not passed. Display the list of errors below the input control. Use an *<ul>* with class *list-group* and *<li>* items with class *list-group-item* and *list-group-item-warning* for each of the error messages.



ReactJS

Title

The title is required

Directors

Danny Boyle

Critics Consensus

As gut-wrenching as it is inspirational, 127 Hours unites one of Danny Boyle's most beautifully exuberant directorial efforts with a terrific performance from James Franco.

Synopsis

James Franco stars in director Danny Boyle's inspiring survival drama based on the incredible true story of Aron Ralston, who became trapped alone in a Utah canyon for days after slipping on a loose rock, and resorted to extraordinary measures in order to make it out of his dire predicament alive. An experienced hiker and climber, Ralston (Franco) is very much in his element when he parks his truck by a mountain near Moab, UT, hops on his bike, and peddles to the middle of nowhere. Later, when Ralston encounters a pair of young female hikers who have gotten lost while searching for a local landmark, he jovially shows them a sight that most casual hikers miss before bidding them farewell and

Release Year

The movie needs a release year

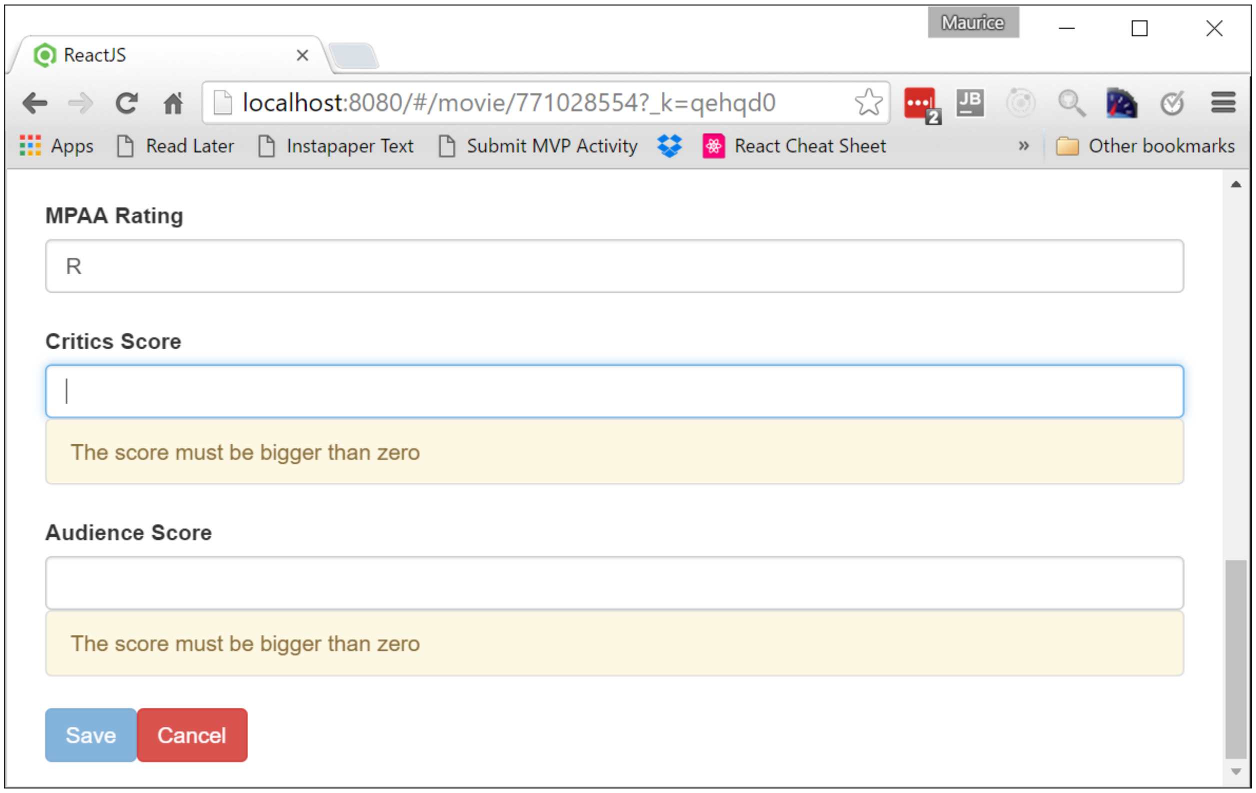
The movie can't be released before the camera was invented

Test the application and make sure the input elements are fully functional now. If you delete a movie title an error message should appear. Make sure you see both error messages if you empty the release year.

The third task is to disable the save button when there are validation errors.

With these changes the movie edit form will display validation errors. However, the user can still choose to ignore these errors and click on the save button.

Add a function *hasErrors()* to the *<MovieEdit>* component. This function should return true if there are any validation errors or false if there are none. Update the **Save** button and disable it if there are any data validation errors.



MPAA Rating

R

Critics Score

The score must be bigger than zero

Audience Score

The score must be bigger than zero

Test the application and make sure you can't save an invalid movie.

Assignment 3 (Optional)

Right now, data is only checked and errors displayed when the user makes a change. If data loaded from the server is already invalid no errors are shown. Update the *componentDidMount ()* function to validate a movie when loaded and display the result in the same way as after an edit action.

The *<TextArea>* component needs to have the same validation logic as the *<InputText>* component. As we don't want to duplicate this logic it is best to turn the error display into a separate component. Use this new component in both the *<TextArea>* and *<InputText>* components. Add validation logic that the **Synopsis** is required with a length between 25 and 500 characters.

Test the application and make sure the Synopsis field is properly validated.