React – practical exercises

The template (jsonpFLIKR.html) for the practical work is in the ReactExerciseFlkr folder. Use the template as your starting point, it gets a JSON feed that you can use to populate your React front end.

Exercise 1

* Familiarise yourself with the contents and format of the JSON data
* Create a component that will display a panel which shows a title, an image and some text
* Add or modify the existing styling to your choice

Exercise 2

* Add in some default props for the panel component
* If you have used a complex (nested) object passed to your panel component, consider how you might refactor your code so that a complex object type is not needed by the panel component e.g. how can you ‘flatten’ the data?

Exercise 3

* Create a wrapper component that will call the panel component for each item in the array
* Bookmark for future reading - Notes on proptypes and complex object defaultprops - <https://stackoverflow.com/questions/40300016/typechecking-nested-object-properties-with-proptype>

Exercise 4

* Add a like button and a likes counter display to your panel
* Use state to store the number of likes
* Optional bonus – add sort capability for the panels

Exercise 5

* Look at the examples in the electronic handouts
  + Owner/ownee callback - **9-newEventsCallback.html**
  + Event emitter – (look at this in the React Dev tools to see the changes) -   
    **9-newEventsEmitter.html**
* Add a Favourite icon to the panel, the functionality is –
  + Only one favourite is allowed, store the info in state on the owner component
  + Store an identifier for the favourite in localStorage and use that to set the favourite icon when the page is loaded
* Research - Some examples of touch applications are available here … <https://react.rocks/tag/Touch>

Exercise 6

* Create a search box component – it should contain a text box and a button
* Display this near the top of the webpage
* Use the ref to automatically select the text when the textbox has focus
* When the text box contents change, change a state property on the component
* When the button is pressed change a state property on the container to reflect the text box contents

**NOTE: we do not yet reread the JSON data from Flikr based on the new search term …. Lets think about that …**

We could put something like this in the SearchChange for the container …

var s = document.createElement('script');

var sval = "http://api.flickr.com/services/feeds/photos\_public.gne?jsoncallback=myfeed&format=json&tags=" + newSearch;

s.setAttribute( 'src', sval );

document.body.appendChild( s );

(That would dynamically add a new script tag for the search term – messy)

But then we would need to re-render the React components …

We could put the data into state and update the state with the array …

this.setState({what:newSearch, data:myArray});

But because the JSON loads asynchronously we would need to press the search button twice to get the new data or put a slight pause in to hope that the data has arrived.

setTimeout(() => {this.setState({what:newSearch, data:myArray})}, 500);

What else could we do? Pub/sub, promises, Redux – if we were writing our own AJAX code to fetch our own JSON data we would have more control over this.

**Refactor** …. Add jQuery getJSON to make life easier …. Remove all the separate script tags and put it all in the main (babel) script tags …

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"> </script>

$.getJSON("http://api.flickr.com/services/feeds/photos\_public.gne?jsoncallback=?&format=json&tags=planets", function(json){

myArray = json.items;

init(myArray);

});

Exercise 7

* Refactor your previous work to add functionality for the search option
  + Based on the search string, get new JSON data and display it on the webpage
  + You could use the jQuery getJSON as shown on the previous page or you could use a different technique of your choice

Exercise 8

* Refactor you previous work to add Redux to your webpage