COMP6080: Web Front-End Programming

Tutorial 7–ish

Sidney Pham
sidney.pham@unsw.edu.au

UNSW

Week 7, Term 3 2020

Welcome Back!

Hope you enjoyed your flexibility week, and great work getting through Assignment 2!

Are there any topics anyone wants further clarified?

I'm thinking of trying to cover closures and the event loop (properly).

Exercise: Using the GitHub API

We have a basic application that loads JSON information from GitHub (via `fetch`) to display how many public repositories exist within the Microsoft GitHub organisation.

Firstly, let's display the number of public repos collected from the fetch into the disabled input.

Adding Button Click Handlers

Let's make the buttons work now. What's going to happen?

We're getting weird behaviour because this `fetch` call (and the chain of `thens`) runs *every* render! Every time the component renders (e.g. on first load, when state changes), the entire function runs. How do we fix this?

We use the `useEffect` hook!

The `useEffect` Hook

Firstly, I think the official React documentation is really good.

Essentially, it lets us run some code after render. By default, it runs both after the first render and after every update. However, by passing an array as the second argument, we can only run the effect when a value in the array changes between renders.

Let's see how this works for our buttons.

Updating the Organisation

Introduce a new input underneath that allows users to enter a string of the organisation they want to find the number of public repos of.

After 500ms of the last keyup on this field, re-fire the fetch to get the new data and populate the input again. However, don't directly call fetch — instead, modify the `useEffect` capture.

Aside: The Comma Operator

Imagine you've got something like this:

```
const f = x => g(x);
```

but you're wondering why `f` isn't working as expected. You might want to test its inputs. You *could* grudgingly do:

```
const f = x => {
  console.log(x);
  return g(x);
};
```

Or instead use the comma operator:

```
const f = x => (console.log(x), g(x));
```

It allows you to evaluate a chain of expressions and only return the rightmost one. For example,

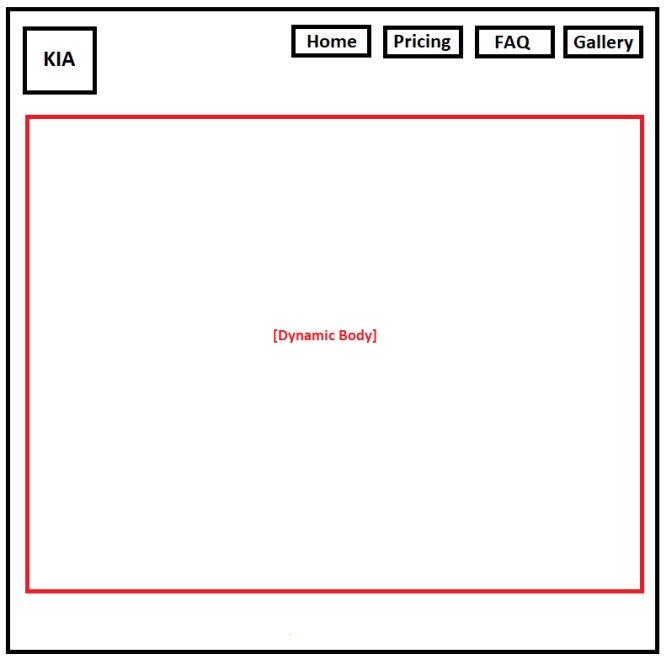
```
return (a, b, ..., z);
```

will evaluate all of `a, b, ...` and return only `z`. Only use this for debugging!

Exercise: A Simple Routed Web App

Build a basic page that consits of a home icon (any logo — we put KIA as a placeholder), and 4 navbar items.

Using `useState`, have each of these 4 navbar items load up a different ReactJS component into the app. You can stub each component with some simple text. The home icon takes you to the same page as the home item in the navbar.



Exercise: Using React Router

Now, let's modify the code to swap between these page states using React Router.