**Rich Web Application Technology – Lab 3 Questions**

**Stream Abstraction**

While Using streams it seems as if data is flowing in and out of your program. It is not really happening this way though. Abstraction means a general idea rather than a specific thing. This is kind of like a message streaming from human interaction to the computer. Humans communicate to the computer through use of a mouse, keyboard or touch screen. The data received through these inputs are used in the program and do different things based on what inputs are used where.

**Relationship between streams and observer patterns**

The relationship between streams and observer patterns is that the streams somewhat extend the observer patterns. When a trigger happens (“click”, “keypressed” etc) on an event, you create a subscribe to a stream of such events, here actions are set to sun based on the events that occur in that stream. E.g. If I click “1” on the calculator, 1 will be added to the display and added to the eval string. This adds to the observer pattern that is a basic concept of reactive programming where a trigger just allows data to be updated when a change is made.

**What are streams useful for modelling and when might you see them used in RWD**

When writing code in an Object Orientated way, were taught to decompose problems into components, interaction and states. This breakdown occurs iteratively on so many levels. Each component will be again sub divided more so. These components will have a stream applied to them to which each component does something different based on the trigger that activates them. E.g. The calculator is a great example. Personally, I had 18 buttons to be added to the calculator. I do this iteratively in a for loop. This modelled the calculator and made each of the buttons on the calculator. Each component(Button) in the for loop has a separate function applied to it so it does different things when clicked by the mouse on the screen. For example, hitting any if the numbers buttons add the number pressed to the evaluation string so that it can be calculated. The operators are done in the exact same way. Finally, the C and the = buttons are done slightly differently in there .subscribe function. The C clears the string and display. And the = calculates the eval string and sets the string to be the answer calculated so more calculations can be done quickly. All these functions are done in the subscribe and this way is very good for modelling Rich Web Application development.

Assume that you are building an interface to an API in your Rich Web App. Describe in detail how you could use the RxJS library to handle asynchronous network responses to API requests. In your opinion, what are the benefits to using a streams library for networking over, say, promises? And what do you think are the downsides?

Initially, I would set up the observable using fromEvent and subscribe etc. Next, I would sort out the handling of the HTTP Request, such as on click of a button, subscribe a stream that triggers a http request. FlatMap would be used on the first observable, subscribed to the resulting observable, writing received data to the DOM. Using flatMap and fromPromise it’ll return a stream of promises rather than a stream of objects that the promise would emit when it resolves. It allows us to convert all the promise resolutions into a single observable stream, which we can just subscribe to. To handle out of order requests, ConcatMap can be used. Unlike flatMap it will preserve the order of the source emissions. Even if the second request completes first the first request will still be displayed first which will be a great solution for out of order requests. For efficiency in connecting to the server, “Debounce” can be used to manage the time in milliseconds that the observable should wait before emitting another request. After the given time, it will emit the last emission within the time. E.g. If typing into a search bar you can use this to smit after a few milliseconds rather than every 1/100th of a second to save processing power and server connections. Depending on the application you can also use the likes of .filter to filter out certain observables that aren’t needed in a specific condition. This will also save processing power and help increase efficiency of your application. Finally, another thing to help efficiency in the application is .merge. this allows you to combine more than one observable to a single observable. In today’s world efficiency is key to a good web application as it reduces latency time.