**Lecture Review Question**

1. Stream Abstraction is a method of obtaining data which may not be present, nor may it not be present in the future. Streams represent a series of objects for modelling asynchronous, infinitely-sized data. Stream abstraction can be a very functional technique for retrieving data for when you don’t know when it will be received, or the size of the data being received.

Streams implement the observer pattern, meaning that it executes a design pattern where an object maintains a list of its dependents called observers, which are to be notified automatically of any changes in state. Streams implement the observer pattern as they have a one-to-many relationship with the abstracted data.

Streams can be constructed over any form of data including user inputs, variables, data structures and properties. Streams are useful for modelling asynchronous, infinitely-sized data. They are functional in that they supply a possible solution to the synchronization problem. This involves providing a unified abstraction of everything, through the modeling of all application states as streams. Thus, enabling faster debugging and development times. This is thanks to all event handling and DOM state changes being implemented within the same classes and using the same semantics. The most common application for streams in rich web development is the capturing of asynchronous events such as retrieving user input. Streams are very practical when the size of the abstracted data is unknown and the time of arrival is also unknown.

1. You can use the RxJS library to handle asynchronous network responses to API requests through the implementation of an observable stream, promises and functions such as .map and .fromPromise, allowing us to handle and render its response data into the DOM.

In my interface is a text field and a button which allows users to query data from an API. This button triggers an API request by creating a fromEvent observable from its onclick event and assigning it to a stream called ‘userBtn’. I would then chain .map following the fromEvent observable, allowing me to map the user’s input to the stream, ‘userBtn’. After this, I will subscribe userBtn to an observerable stream through the use of functions, fromPromise (which returns am observable stream of promises) and flatMap (which flattens all promise resolutions into a single observable stream). userBtn subscribing to this returns a response object to the original API call.

The benefits of using a streams library for networking over promises include that they are intended for handling events which may or may not occur repeatedly, whereas promises occur immediately and only occur once. Networking involves regularly fetching dynamic data over HTTP and thus is better suited to networking than promises. Another advantage of using streams library is that it can be cancelled whilst a promise cannot. This is executed through the subscription of an observable which allows it to cancel its subscription. I think the main downside associated with streams library over promises is that they cannot be chained whereas promises can. This means that a promise returns another promise whereas an observable within the stream’s library, can only return a subscription to the observable object.

1. An asynchronous function is a method which uses an event loop to return an AsyncFunction object. It accomplishes this by implementing a Promise to return its result. The consequences of these functions sharing global state are that while one global state variable is waiting for a Promise to return, the same variable could potentially be assigned a different value in another part of the program. In addition to this, debugging time will increase as sharing global state means that any bug has the potential to affect the entire system, and will subsequently be harder to find.

Good practice to alleviate these problems is to inject the required dependencies into each functions. This is done by simple passing the data they need into them. Benefits of avoiding functions using global state include that debugging and testing are quicker than they would be using global state functions.