## Explain what is meant by the stream abstraction. What is the relationship between streams and the observer pattern? What are streams useful for modelling and when might you use them in Rich Web development?

Stream abstraction is used to model asynchronous data resources. In other words, it distributes a sequence of data over time. Streams process the data chunk by chunk and then once it is processed it is forgotten, this is less memory intense than the usual way to store data like in an array.

The observer pattern is mainly implemented as event handling systems. When an object is interacted with it will automatically notify all its dependencies and they in turn might call a function. This is similar to the way in which streams are used. However instead of firing an action on a single event streams use a method called observable and when a button is clicked they can perform a variety of operations that an occur in that stream.

Streams are useful for when you need to model state synchronisation. Because streams can be used to model all application states which allows for complete abstraction of all elements. All inputs, keyboard and mouse clicks, DOM changes and network responses can all be processed in the same stream. This model can be used when there is a lot of processing to be done and no need to store the data.

## 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?

When a response is received from the API with the RxJS library you can subscribe to the event. When you have subscribed to the event you have other functionality which allows you to display certain data, map the data to another array or display an image.

The biggest difference between Promises and Streams are that Promises can only able to handle one asynchronous operation. Promises also only return once, if promises are used for event handling, like button clicks, it will return the first click and not any subsequent clicks. Observables in streams really helped with this issue by being able to subscribe to a stream of data rather than once piece of data. I don’t think there are many downsides to using observable over promises other than promises are a great at handling operations in a structed way.