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

A.

* Streams in themselves are an abstraction used to model asynchronous data sources. It is a powerful technique when processing data when you may not know the data’s potential size or you don’t know when it will arrive into the application.
* The observer pattern is a pattern in which an object maintains a list of its dependants called observers. If any state changes it notifies them automatically. The relationship between the streams and the observer pattern manifests itself in subscribing to a said stream within a function. Those functions that are defined are the observers of said stream.
* This would be very useful when dealing when processing data of potentially unknown size that is constantly being fed into the application. In regards to Rich Web this could be done by making some API request. Data would arrive asynchronously without holding up the user interface.

Ref 21- Streams DIT Lectures by Brian Gillespie [Internet] Available from DIT Webcourses.

Ref The introduction to Reactive Programming you’ve been missing, Andre Staltz, [Internet] Available from: <https://gist.github.com/staltz/868e7e9bc2a7b8c1f754>

B. 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?

* In order to aid the asynchronous processing an RxJS (just) operator can be used to create a stream on the URL of the API end point.
* Then using a mergeMap function execute all nested API requests immediately as they pass through the stream. What it essentially does is flatten the stream into stream of responses. Because of this parallel execution it will be quick.
* I would then subscribe and use that data from the API in the DOM itself however it would be required.

As in example above I believe it is easier to manage multiple promises from multiple sources and it is easier to keep track of them with Observables, however I believe that the complexity and the additional RxJS dependencies that come with this approach may sometimes be out weighted by the native JS promise implementation.