**Q1) 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?**

A stream is an abstraction of a sequence of data, which is distributed in time. Usually we know sequences as arrays and lists, data distributed in space. First, we load the whole sequence into memory and then process the data. Streams allow us to process data chunk-by-chunk, as data appears in memory, pass it further and then to forget it immediately. In this case we don’t much memory.

Stream are collections of data – just like arrays or strings. The difference is that streams might not be available all at once. And they don’t have fit in memory. This makes streams really powerful when working with large amounts of data. The observer pattern is objects interface provide a generalized mechanism for push-based notification, also known as the observer design pattern. The observable object represents the object that sends notifications the provider the observer object represents the class that receives them.

**Q2) 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?**

Using Rx just for monitoring a sequence of events from a long – running task, however doesn’t take full advantage of the technology. The beauty of Rx is that it can be used for asynchronously integrating any event-based process with any other process. It uses Rx to make asynchronous calls to a web service from a button click in a web page (a button click is, effectively, a sequence of one event to use Rx in the client – side web environment, it will use Rx for JavaScript (RxJS). The API calls are almost a staple on with several advantage from offloading work to designing reliable network behaviour.

.