# Lecture Revision Questions 1:

* **Stream Abstraction** is a method used to model asynchronous data sources. A stream is a sequence of data elements that is available to the user overtime (Not all at once) so they data arrives at different times. This could be any source of input or destination for output e.g. keyboard, screen, CD, network ports etc.
* **Streams and Observer Pattern** have a relationship based on the “observation” of data. When we are “listening” to a stream of data it is known as **subscribing** and the functions we are we are using are defined as observers, therefore the stream is what is being observed and this is what creates the relationship between the two.
* **Streams are useful for Modelling** asynchronous or indefinitely-sized data, this is the case because we are unaware of the size of the data or when the data might arrive so with the use of streams it makes it easier to model this data. This might be used in Rich Web development whenever data is expected without an idea for how large the data is or when it will arrive. An example of a stream of data is with I/O devices, as they can produce or consume near infinite amounts of data overtime. Regular functions can’t handle this data, so we need streams to model the data for us.

# Lecture Revision Questions 2:

* **Streams library (RxJS) for Networking** may be used for networking over promises as they use observables, observables have all the same features that promises have and more. Promises can be used to handle a single event when an asynchronous operation completes or fails, however, observables can handle multiple events which makes them more flexible than promises. Observables also have an important feature which Promises don’t and that’s the ability to be cancellable, which means that if the call to a server is not needed anymore the Observable allows it to cancel the subscription while a promise would still call the server.

Alongside all that was mentioned above, RxJS library has many more features that promises do not. The Observable provides many operators such as **map**, **forEach**, **reduce**, **filter** etc. These add many ways to deal with the data from the streams directly without having to use arrays. From what I have learned and read I don’t think there are any downsides to using RxJS and Observables over promises as they have all the same functionality as promises and more. The operators can be useful and should be considered when choosing between Observables and Promises, however if you have no need for the operators using

Promises should be sufficient.