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?

- Stream abstraction is a reactive programming way of modelling asynchronous data sources.
- The relationship between streams and the observer pattern is co-operative, streams implement the observer pattern to provide a trigger that will allow data to be realised using the subscribe operation.
- Streams are useful for modelling asynchronous data sources and we are most likely to use streams in rich web development when we need to process data in a time ordered fashion e.g. processing video or log files.

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 RxJS library can help handle asynchronous network responses to API requests in the following way:

- Creating observable streams from HTTP requests
- Handling HTTP error responses
- Handling out of order HTTP request completion
- Throttling user input

In my opinion, the benefits and downsides of using a stream library for networking over, for example, promises are as follows:

Benefits: Supports a wide range of data sources, regardless of type or whether there is latency or not. Using a stream library can allow us to react to the speed of the reader, making the control of flow seamless.

Downsides: There are extensive amount of operators and the library is huge, which can become overwhelming and a bit of an overkill.