Reactive Web Programming and Streams

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?

The term stream abstraction refers to the concept of reducing the synchronisation problem in web development to a time-ordered list of items which you read and transform values in some application-specific order. Streams are useful for modelling asynchronous, infinitely-sized data. An obvious application for streams is as an abstraction of asynchronous events these events could range from mouse clicks, keyboard input, network responses, timers to DOM state changes.

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?

One could use the Observable.just() method to make a request to the API, one could then use the flatMap() method to retrieve a promise which could then be subscribed to and bespoke code could run whenever the data is retrieved from the API. sPromises are really a subset of Observables better suited to dealing with problems with callbacks. Promises are generally resolved to individual values where, streams are better for dealing with sequences of asynchronous events. The main downside to using stream is that there currently is no standard for them, so browser support is limited. Another downside to streams is the added complexity they bring when compared to simply using promises.