**Rich Web Lab 3 Questions**

**Questions One:**

**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 basically the process of reducing the object to its essence so that only the necessary characteristics are exposed to the user. Streams abstract some data which may or may not be present now or may or may not arrive in the future.
* A stream is the observable being subscribed to. In the diagram below you can see the relationship between streams and the observer pattern.

A piece of paper with writing on it

Description automatically generated

* Streams are useful for modelling all application states, They provide a unified abstraction of everything. Streams are offend used for handling mouse clicks, keyboard inputs, timers, network responses and DOM state changes. The reason for this is because streams allow all these things to be processed within the same logical structure using the same semantics. This means that using streams for these types of operations allows the application architecture to be reduce.

**Questions Two:**

**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 my opinion the main advantages of using a streams library for networking over the likes of promises is as follows:

* A streams library has functionality which allows it to handle more than a single event unlike promises.
* Observables can also be cancelled while a promise will eventually call the success or failed call back even when not needed.
* Streams also reduce the application architecture as streams deal with multiple events.

In my opinion these are the main disadvantages of using a streams library:

* A major disadvantage I would say from personal experience is they are hard to learn.
* They can be more memory intensive
* Most complexities have to be dealt with at the time of declaration of new services

**Questions Three:**

**Consider three asynchronous tasks, A,B & C. What are the consequences of these functions sharing global state? What is a good practice to alleviate any problems associated with this?**

The main consequence with these three asynchronous tasks, A,B and C sharing global states is that it may introduce hidden dependencies. The reason this may be bad is because it breaks multithreading which for most modern web apps is vital. The state of global variables is also unreliable, because the different task sharing the same global state can be doing anything to it affecting the other functions.

You can alleviate any problems associated with this by managing the global state correctly. In order to be able to manage the global state correctly one would have to layout and write there code very precisely.