Rich Web Development Lab 3

Lecture Review Questions

1.

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?

Stream abstraction is the process of creating, processing, and consuming data in an incremental fashion. An example of this is streaming a YouTube video, as the video buffers, rather than wait for the whole video to buffer, the video is streamed through the buffer. This allows the user to watch the video as it is ready. This means streams allow a lot of control of data buffering, such as detecting when streams start, combing streams, stopping streams and handling errors. The observer pattern is a design pattern that we use in coding, which is used in streams. An observable is anything we can observe such as a click event or a livestream.

Streams are useful for modelling application state, as this solves the synchronisation problem. By using streams throughout the application, it stays in a constant state of streaming, whether it is in one stream or multiple streams of data. Examples of events that could make use of streams are mouse clicks, keyboard input, network responses, timers, and changes to the state of the DOM. Streams are also used throughout reactive programming as this is programming using asynchronous data streams.

2.

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?

The RxJS library could be utilised extensively to stream network responses to make API responses. There are a variety of different use cases depending on the kind of data you wish to send or the context of the API request, such as how often it is called. In order to trigger the API request, we could use ‘fromEvent’ to attach an event listener to the DOM. Next, the ‘from’ operator would be extremely useful as it can be used to turn a promise into an observable. Ajax is also an RxJS operator that can be used to create an observable for an AJAX request. Perhaps you want to make continuous API requests once the user triggers the fromEvent, we could use RxJS’s interval operator to make API requests every set interval. Defer is another RxJS operator we could make use of, as it can be used to create a new observable once we await a response from the API. catchError would also be useful in case of failure as it gracefully handles observables during errors. The pipe function may also be utilised before sending the API request as we may need to process the data beforehand, using some of RxJS filtering operators such as filter, first, last etc.

The benefits of using a stream library, like RxJS, make it a much better choice in comparison to using Promises. Observables have many benefits when it comes to networking. For example, observables can be asynchronous vs synchronous whereas promises are always asynchronous. Observables can also stream the data to you continuously, even as the data changes, whereas promises only to return the data once. Observables also can have multiple listeners who are viewing the data whereas promises are limited to one. Another useful feature of observables is the ability to unsubscribe and break the chain. The main downside of observables in my opinion are the complexity of implementing them. Another downfall is they do not work directly with await and async requests.

3.

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?

Sharing global state across functions is a bad practice that can make a program very unpredictable. This is because, if three asynchronous tasks A, B and C are running they can trigger events which alter the global states values. For example, A changes the global state, then we want to test asynchronous task B, the value will be different making it very hard to test these tasks, something which is essential to good programming. However, the use of observables can be a good practice to help alleviate these problems. Using observables, it is easy to inform the asynchronous tasks A, B and C of changes to state, as this is a feature of observables.