3 - Reactive Web Programming and Streams- D18124115

Questions and Answers

1. 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?

A stream involves breaking down into small chunks a resource that you want to receive, send, or transform.

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Stream is an abstract concept and can be presented in other languages

The observable is the Subject being observed/subscribed to You can also put it into other words: The stream is the “observable” being observed.

Streams allow a lot of control of data buffering: it is possible to detect

when streams start and end, chain streams together, handle errors and

cancel streams

The two most common use cases for streaming:

Streaming media, especially video

Real-time analytics

1. 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?

Reactive programming is an asynchronous programming paradigm concerned with data streams and the propagation of change. RxJS (Reactive Extensions for JavaScript) is a library, which allows you to work in asynchronous data programing.

Observable can help us to work and interact with different streams.

We can use the following libraries to handle network response to API request:

Map: to manipulate responses from the API

Request: API performs calls and returns Observables, basic RxJS objects that represent the HTTP response

flatMap – to create new Observable basing on the data emitted by another Observable

Promises are not able to work on multiple events.