Stream API - Readable Streams

In Stream API, a readable stream is a data source from where we can read data in a sequential and asynchronous way. It is a standardized way to get data from the underlying sources. Underlying sources is the resource which present on the network. They are of following two types –

Push source – In which the data is pushed to you when you access them. You can have control of the stream like when to start or when to pause or even when to terminate the current stream. For example, video game streaming.

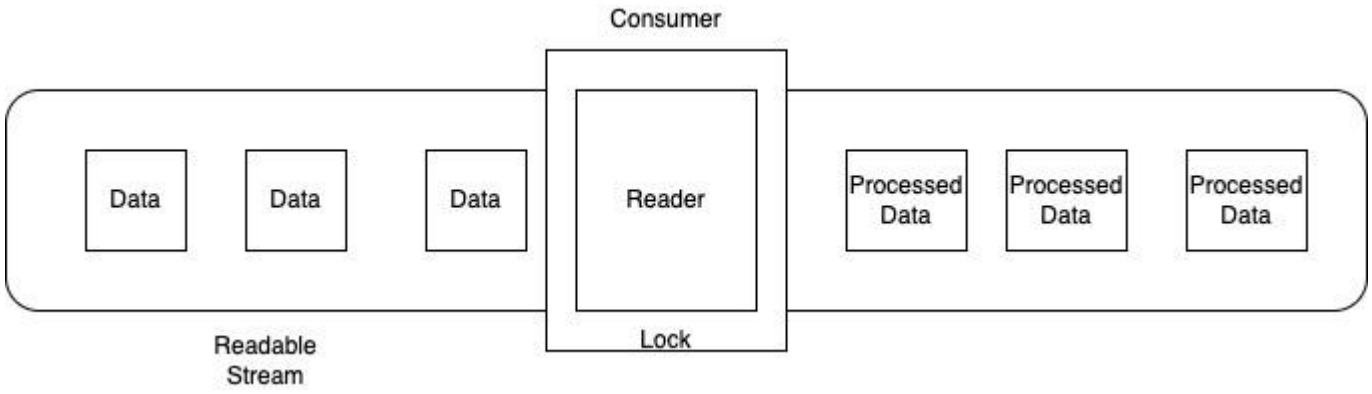
Pull source – In which you need to explicitly request the data from them. For example access files with the help of Fetch or XHR call.

In a Readable Stream, the data is in the form of small chunks so it is read sequentially, one chunk at a time. A chunk can be a single byte or can be of a larger size. Hence the size of the chunks can be different in a stream. Now lets us understand how readable stream works.

Working of Readable Stream

The working of the Readable stream is quite straightforward. In a readable stream, the chunks of data are placed in enqueue. It means the chunks are waiting in the queue to read. Here we have another queue that is an internal queue which keeps track of unread chunks. The chunks are read by the reader. It processes the data of one chunk at a time and allows you to perform operations on the data. One reader can read only a single stream at a time. When the reader starts reading the stream at that time the stream is locked for that reader means no other reader is allowed to read that stream. If you want another reader to read that stream, then you have to terminate the first reader or can create a tee stream. Also, each reader has its own controller which allows you to control the stream such as start, close, or pause.

It also has a consumer who is responsible for handling the received data from the readable stream and processing it and can able to perform operations on it.



Readable Stream Interfaces

Stream API supports three types of readable stream interfaces –

- ReableStream Interface
- ReableStreamDefaultReader Interface
- ReadableStreamDefaultController Interface

ReadableStream Interface

The ReadableStream Interface is used to represent a readable stream of data. It is generally used with Fetch API to handle the response stream. It can also handle the response stream of developer-defined streams.

Constructor

To create a readable stream object for the given handlers ReadableStream interface provides a ReadableStream() constructor.

Syntax

```
const newRead = new ReadableStream()
Or
const newRead = new ReadableStream(UnderlyingSource)
Or
const newRead = new ReadableStream(UnderlyingSource, QueuingStrategy)
```

Following are the optional parameters of the ReadableStream() constructor –

UnderlyingSource – This object provide various methods and properties that define the behaviour of the stream instance. The methods are: start(), pull(), and cancel() whereas the properties are: type and autoAllocateChunkSize.

QueuingStrategy – This object is used to define the queuing strategy for the given streams. It takes two parameters: highWaterMark, and size(chunk).

Instance Properties

The properties provided by the ReadableStream interface are read-only properties. So the properties provided by ReadableStream are –

Sr.No.	Property & Description
1	**ReadableStream.locked** This property is used to check whether the readable stream is locked to a reader or not.

Methods

The following are the commonly used method of the ReadableStream interface –

Sr.No.	Method & Description
1	**ReadableStream.cancel()** This method returns a promise which will resolve when the stream is cancelled.
2	**ReadableStream.getReader()** This method is used to create a reader and locks the stream to it. No other reader is allowed until this reader is released.
3	**ReadableStream.pipeThrough()** This method is used to create a chainable way of piping the current stream through a transform stream.
4	**ReadableStream.pipeTo()** This method is used to pipe the current ReadableStream to the given WriteableStream. It will return a promise when the piping process completes successfully or reject due to some error.
5	**ReadableStream.tee()** This method is used to get a two-element array which includes two resulting branches as new ReadableStream objects.

ReadableStreamDefaultReader Interface

The ReadableStreamDefaultReader interface is used to represent a default reader which will read stream data from the network. It can also be read from ReadableStream.

Constructor

To create a readableStreamDefualtReader object ReadableStreamDefaultReader interface provides a ReadableStreamDefaultReader() constructor.

Syntax

```
const newRead = new ReadableStreamDefaultReader(myStream)
```

This constructor contains only one parameter which is myStream. It will read ReadableStream.

Instance Properties

The properties provided by the ReadableStreamDefaultReader interface are read-only properties. So the properties provided by ReadableStreamDefaultReader are –

Sr.No.	Property & Description
1	**ReadableStreamDefaultReader.closed** This property returns a promise which will resolve when the stream is closed or rejected due to some error. It allows you to write a program which will respond at the end of the streaming process.

Methods

The following are the commonly used method of the ReadableStream interface –

Sr.No.	Method & Description
1	**ReadableStreamDefaultReader.cancel()** This method returns a promise which will resolve when the stream is cancelled.
2	**ReadableStreamDefaultReader.read()** This method returns a promise which will give access to the next chunk or piece in the stream's queue.

3	**ReadableStreamDefaultReader.releaseLock()** This method is used to remove the lock of the reader on the stream.

ReadableStreamDefaultController Interface

The ReadableStreamDefaultController interface represents a controller which allows us to control the ReadableStream State or internal queue. It does not provide any controller and the instance is created automatically while constructing ReadableStream.

Instance Properties

Sr.No.	Property & Description
1	**ReadableStreamDefaultController.desiredSize** This property is used to find the desired size to fill the internal queue of the stream.

The properties provided by the ReadableStreamDefaultController interface are read-only properties. So the properties provided by ReadableStreamDefaultController are –

Methods

The following are the commonly used method of the ReadableStreamDefaultController interface –

Sr.No.	Property & Description
1	**ReadableStreamDefaultController.close()** This method is used to close the related stream.
2	**ReadableStreamDefaultController.enqueue()** This method is used to enqueue the specified chunks or pieces in the related stream.
3	**ReadableStreamDefaultController.error()** This method will cause any future interaction with the related stream to the error.

Example - Creating ReadableStream

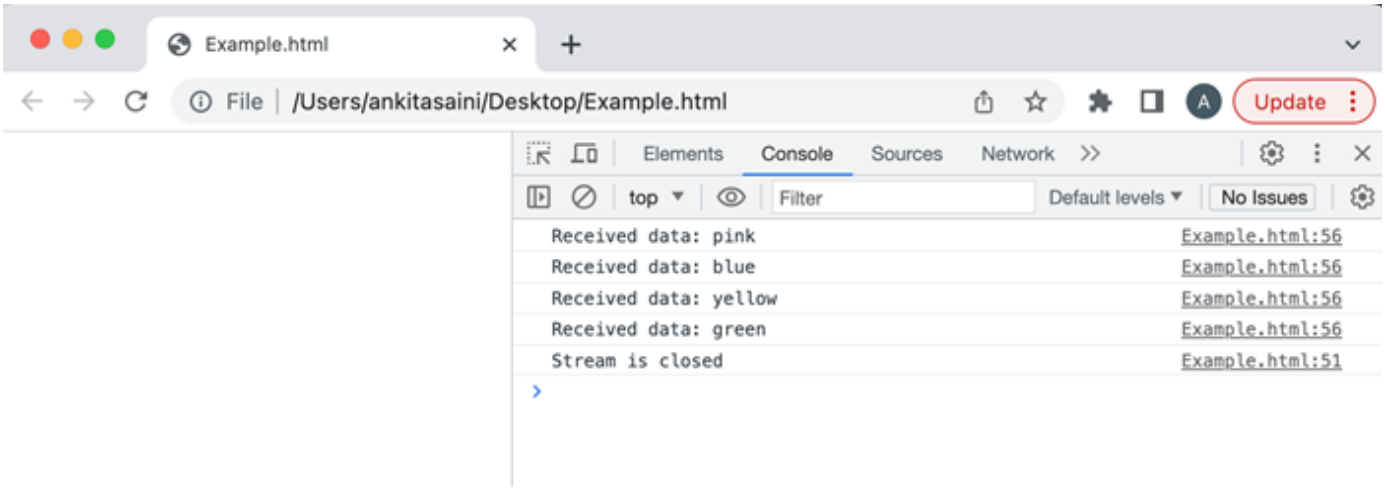
In the following program, we will create a custom readable stream using the ReadableStream constructor. So first we create a function which generates data in chunks.

Then we create a readable stream using ReadableStream() constructor containing the start() function. This start() function uses pData() recursive function which pushes data from myData() function to the consumer with the help of a controller, where a timeout is set of 1 sec between each push operation. Now we create a reader to consume the data from the stream using the getReader() function. Then we create a readMyData() function to recursively read the data from the stream with the help of the reader. When the stream ends, then the done flag is set to true and we exit from the recursive loop.

```
<!DOCTYPE html>
<html>
<body>
<script>
   // Function that produces data for the stream
   function* myData() {
      yield 'pink';
      yield 'blue';
      yield 'yellow';
      yield 'green';
   }
   // Create a readable stream using ReadableStream() function
   const readStream = new ReadableStream({
      start(controller) {
         const data = myData();

         // Adding data to the stream
         function pData() {
            const { done, value } = data.next();

            if (done) {
               // Close the stream if no more data is available
               controller.close();
               return;
            }
            // Pushing the data to the consumer
            controller.enqueue(value);

            // Continue pushing data after 1 sec
            setTimeout(pData, 1000);
         }
         // Calling the pData function to start pushing data
         pData();
      }
   });
```

```
    // Create a reader for the readable stream
    const myreader = readStream.getReader();
    function readMyData() {
       myreader.read().then(({ done, value }) => {
          if (done) {
             // Stream is closed
             console.log('Stream is closed');
             return;
          }
          // Processing the received data
          console.log('Received data:', value);

          // Continue reading the data
          readMyData();
       });
    }
    // Calling readMyData() function to start
    // reading data from the readable stream
    readMyData();
</script>
</body>
</html>
```

Output



Conclusion

So this is the readable streams in Stream API. They are the most important and most commonly used streams of the Stream API. They are supported by almost all the web

browsers like Chrome, Firefox, opera, edge, safari, etc. Now in the next article, we will learn about writable streams of Stream API.