SONY
make.believe

W3C SysApps WG

TCP and UDP Socket API based on Streams

W3C San Jose F2F meeting April - 2014

Claes Nilsson
Technology Research / Sony Mobile
claes1.nilsson@sonymobile.com

SONY
make.believe

Background

- W3C SysApps TCP and UDP Socket API provides interfaces to UDP sockets, TCP Client sockets and TCP Server sockets.

- There is an ongoing rewrite of this API to be based on a general Streams API.

SONY
make.believe

Motivations for this potential re-design

- Reusing a general standardized solution for handling the complexity of sending, receiving, buffering, backpressure and other issues related to streaming and asynchronous APIs.

- Reusing a solution for piping a source stream to a destination stream.

SONY
make.believe

Streams API activities

- Ongoing work on a general Streams API:
  - W3C Streams API
  - WHAT WG Github Streams API (including node.js community)
  - Work on aligning these activities is in progress.

- Currently the work to rewrite the TCP and UDP Socket API is based on the WHAT WG Streams API

SONY
make.believe

What is a Streams API?

- A streams API provides an interface for creating, composing, and consuming streams of data.

- The work on Streams deals with similar issues as we do with the TCP and UDP Socket API, e.g.:
  - ”don’t lose data”
  - ”don’t overflow send buffers”
  - ”keep it simple for developers”

- The Streams API is designed to be used in conjunction with other APIs.

SONY
make.believe

Stream Producers

- APIs which can produce a Stream object are identified as *Producers*.
  Examples:
  - XMLHTTPRequest
  - FileReader
  - Media Capture
  - MediaStream Recording API
  - Web Cryptography API
  - WebSockets
  - RTCPeerConnection
  - TCP and UDP Sockets

SONY
make.believe

Stream Consumers

- APIs which read and act on a Stream object are identified as *consumers*.
  Examples:
  - XMLHttpRequest
  - Web Audio
  - Web Cryptography API
  - WebSockets
  - RTCPeerConnection
  - FileWriter
  - TCP and UDP Sockets

Reading push-based data sources (such as TCP) - requirements

SONY
make.believe

- Handling new data pushed from the source
- Mechanism for pausing and resuming the flow of data.
- A way to signal that the source has no more data
- A way to signal when there is an error in getting data
- Buffering logic in the stream primitive itself to assure that we don’t lose data.

Writing data - requirements

SONY
make.believe

- The Stream object must handle the complexity of buffering sequential writes, e.g. the case when the send buffer becomes full due to slow network. For example:
  - A method to write data
  - A way to signal that the buffer is getting full (reached the “high water mark”)
  - A way to signal that the buffer is drained and can receive more data
- Must be possible to signal that the underlying sink should be closed.
- Must be possible to detect “abort” signal

SONY
make.believe

Piping streams - requirements

- A common way of consuming streams is to pipe them to each other. This is one essence of streaming APIs: getting data from a readable stream to a writable one, while buffering as little data as possible in memory.

- Example: Create a read stream from a file, possibly transforming it, and pipe it to a write TCP socket stream.

How to use the Streams API for TCP and UDP Sockets? 1(5)

SONY
make.believe

Before:

```
[Constructor (DOMString remoteAddress, unsigned short remotePort,
      optional TCPOptions options)]
interface TCPSocket : EventTarget {
  readonly attribute DOMString      remoteAddress;
  readonly attribute unsigned short remotePort;
  readonly attribute DOMString      localAddress;
  readonly attribute unsigned short localPort;
  readonly attribute boolean        addressReuse;
  readonly attribute boolean        noDelay;
  readonly attribute unsigned long  bufferedAmount;
  readonly attribute ReadyState     readyState;
           attribute EventHandler   ondrain;
           attribute EventHandler   onopen;
           attribute EventHandler   onclose;
           attribute EventHandler   onerror;
           attribute EventHandler   ondata;
  void      close ();
  void      halfclose ();
  void      suspend ();
  void      resume ();
  boolean send ((DOMString or Blob or ArrayBuffer or ArrayBufferView) data);
};
```

How to use the Streams API for TCP and UDP Sockets? 2(5)

SONY
make.believe

Now:

```
[Constructor (DOMString remoteAddress, unsigned short remotePort,
      optional TCPOptions options)]
interface TCPSocket : {
   readonly attribute DOMString         remoteAddress;
   readonly attribute unsigned short    remotePort;
   readonly attribute DOMString         localAddress;
   readonly attribute unsigned short    localPort;
   readonly attribute boolean           addressReuse;
   readonly attribute boolean           noDelay;
   readonly attribute ReadyState        readyState;
   readonly attribute Promise           opened;
   readonly attribute Promise           closed;
   readonly attribute ReadableStream output;  // ReadableStream is defined by Streams API
   readonly attribute WriteableStream input;    // WritableStream is defined by Streams API
   void socketClose ();
   void socketHalfClose ();
};
```

How to use the Streams API for TCP and UDP Sockets? 3(5)

SONY
make.believe

- Each Streams API based API must provide an *adaptation layer* to the Streams API.

- The *adaptation layer* to Streams API is created through implementation of a number of functions that are given as input arguments to the constructors of the Readable/WritableStreams objects and called by the Streams API implementation.

- These functions then calls the internal methods of the Streams API to do stuff.

How to use the Streams API for TCP and UDP Sockets? 4(5)

SONY
make.believe

For example, the ReadableStream's constructor is passed the following functions that must be implemented by the TCP and UDP Socket API:

- start(): Called immediately by Streams implementation. Used to adapt to the underlying TCP implementation.
- pull(): Used to start the flow of TCP data after a "buffer getting full" condition.
- cancel(): Called when the readable stream is canceled. Used here to close the TCP connection.

How to use the Streams API for TCP and UDP Sockets? 5(5)

SONY
make.believe

- For example, the ReadableStream's constructors start() function does the following:
  - Performs TCP connection setup handshake.
  - Pushes received TCP data into the internal buffer by calling the Streams API's internal push() function.
  - When push() return value says "high watermark reached" then stops receiving TCP data through the TCP flow control mechanism.

Application code example

SONY
make.believe

```
// Echo client

var mySocket = new TCPSocket("127.0.0.1", 6789);

mySocket.input.write("Hello World").then(
  () => {
    console.log("Data has been sent to server");
    mySocket.output.wait().then(
      () => {
        console.log("Data received from server:" + mySocket.output.read());
        mySocket.socketClose();
      },
      e => console.error("Receiving error: ", e);
    );
  },
  e => console.error("Sending error: ", e);
);
```