SONY
make.believe

W3C SysApps WG

Raw Socket API based on Streams

W3C TPAC November - 2013

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

SONY
make.believe

Proposal for Raw Socket API

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

- Reviewers from the node.js community have proposed that the Raw Socket API should be based on a general Streams API.

- *I am investigating how the SysApps Raw Socket API could be reworked to be based on a Streams API. However, this would mean a significant redesign of the Raw Socket API.*

SONY
make.believe

Motivations for this potential re-design

- Reusing a general standardized solution for handling the complexity of 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)
  - These proposals are unfortunately very different and not coordinated.

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 Raw 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
  - WebSockets
  - Raw Socket

SONY
make.believe

Stream Consumers

- APIs which read and act on a Stream object are identified as *consumers*. Examples:
  - WebAudio
  - WebSockets
  - Raw Socket

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 Raw Sockets? (Extremely preliminary!) 1(3)

SONY
make.believe

Today:

```
[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 Raw Sockets? (Extremely preliminary!) 2(3)

SONY
make.believe

Tomorrow ? (extremely preliminary….):

```
[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 ReadyState        readyState;

           attribute EventHandler      onopen; // Not sure if we should use Stream API handlers instead
           attribute EventHandler      onclose; // Not sure if we should use Stream API handlers instead
           attribute WritableStream    out;
           attribute ReadableStream in;

  void      close ();       // Not sure if we should use Stream API methods instead
  void      halfclose ();  // Not sure if we should use Stream API methods instead
};
```

How to use the Streams API for Raw Sockets? (Extremely preliminary!) 3(3)

SONY
make.believe

Example: Writing the contents of a readable stream to the console as fast as it can.

```
var mySocket = new TCPSocket("127.0.0.1", 6789);
pump ();
function pump() {
  while (mySocket.in.readableState === "readable") {
    console.log (mySocket.in.read());
  }
  if (mySocket.in.readableState === "finished") {
    console.log("--- all done!");
  } else {
    mySocket.in.waitForReadable().then(pump, e => console.error(e));
  }
}
```