Streams, Buffers, & Pipes

Terminology

Buffer

Temporary storage location for data in the process of being moved from one place to another.

Intentionally kept small.

Stream

A sequence of pieces of data made available over time that eventually form into a whole.

Can be from one computer to another or from one process to another inside the same computer.

Chunk

A piece of data being sent through a stream.

Pipe

A connection between two streams.

Binary Data

Data stored in binary (0s and 1s).

Buffers in Node

Buffers

Created on the C++ part of Node and made available to the JavaScript portion via the Buffer functionality.

Buffer is available to Node applications globally and does not need a require statement to import it.

Buffer Content

Buffers hold raw binary data, but we can apply encodings in order to convert the binary data into forms that make sense to people.

Despite being stored as binary data, when raw buffer content is logged, it logs as hexadecimal.

```
<Buffer 54 68 69 73 20 69 73 20 73 6f 6d 65 20 74 65 78 74 0a>
```

Buffer Creation

```
var buf1 = Buffer.from('This is a new buffer');
var buf2 = Buffer.from(buf1);
```

More Buffer Examples

```
var buf1 = Buffer.from('Buffer Demo');

console.log(buf1);
console.log(buf1.toString());
console.log(buf1.toJSON());
console.log(buf1[2]);

buf1.write(' No 1');
console.log(buf1.toString());
```

Buffer Use

We won't often create buffers ourselves, but rather, buffers are something that we receive from another operation and have to handle.

However, we need to know what they are and how to use them.

Streams in Node

Streams and Data

Data is split into chunks and sent through a stream.

Data isn't being downloaded in its entirety before being processed or operated upon in some manner.

Often used with a buffer so that whatever process is handling the data has enough to operate on before passing that data along and collecting more.

Event Emitter

The Stream module in Node inherits from the event emitter.

Any streams created have access to methods such as on or emit.

Types of Streams

- Writable Streams
  - Streams to which data can be written.
- Readable Streams
  - Streams from which data can be read.
- Duplex Streams
  - Streams that are both readable and writable.
- Transform Stream
  - Streams that are both readable and writable that can also modify data as it is read or written.

Working with Streams

Instead of working directly with Stream, or even Readable, Writable, etc., when using streams, you create your own custom stream object that will inherit from the type of stream you want to create.

These custom streams are usually created for the purpose of accomplishing a specific task, and via inheritance, it will have all the functionality of the built in stream and specific stream type, as well as the event emitter.

Readable Stream Demo

Writable Stream Exercise

Pipes in Node

Pipes and Streams

Pipes are used for connecting a Readable Stream to Writable Stream, sending the data from one to the other.

If the source of the Readable Stream is larger than the buffer, the data transfer will happen in chunks.

Pipe Chaining

If the Writable Stream is also Readable (Duplex Stream), it can be piped again to another Writable Stream.

Pipes can be chained indefinitely so long as you start out with a stream that is readable, and subsequent streams should be both writable and readable.

Pipe Demo

Pipe Chain Demo

Pipe Exercise