* **[Streams](https://nodejs.dev/learn/nodejs-streams)**
* [**Stream LinkedIn Videos**](https://www.linkedin.com/learning/advanced-node-js/using-readable-streams?autoAdvance=true&autoSkip=true&autoplay=true&resume=false&u=2113185)
* [**Buffer vs Streams in nodejs**](https://medium.com/developers-arena/streams-and-buffers-in-nodejs-30ff53edd50f#:~:text=A%20buffer%20is%20a%20temporary,works%20on%20String%20and%20Buffer%20.)

Streams work on a concept called buffer.

A buffer is a temporary memory that a stream takes to hold some data until it is consumed.

In a stream, the buffer size is decided by the highWatermarkproperty on the stream instance which is a number denoting the size of the buffer in bytes.

Consider the following:

* You have a bucket named “Source” filled with water
* You need to transfer that water to bucket “Destination”
* One way is to have an intermediary bucket “Buffer”
* Source -> Buffer -> Destination, as shown below. But this is tedious

Graphical user interface

Description automatically generated with low confidence

What is other alternative ?

Connect source vs destination using hose.

A picture containing funnel chart

Description automatically generated

We can read a video file and send it to response without streams ,

Text

Description automatically generated

Disadvantages of this is

1. Task manager 🡪 node process 🡪 memory imprint.
2. Run the following command “node –trace\_gc buffer.js”
3. A picture containing text

   Description automatically generated
4. There are two many mark-sweep , which means momentarily node process is paused for garbage collection, scavenge doesn’t collect the garbage. scavenge meant only for small objects.
5. Scavenge Vs Marksweep : <https://andrejsabrickis.medium.com/monitor-node-application-garbage-collection-8e92980bb855>

The same approach with streams

* Less memory imprint
* Less mark-sweeps
* Reason is in streams we are sending the content bit by bit as chunks,

.Text

Description automatically generated

**Backpressure:**

What is backpressure in streams ?

Consider “source” bucket is pouring water at steady rate, and its been consumed by destination bucket as shown,

Chart, funnel chart

Description automatically generated

Now consider the adverse case

Source bucket is pouring at faster rate, at which destination bucket unable to consume in that case what will happen , the hose will overflow … as shown

Icon

Description automatically generated

To overcome this,

Pour the water from source to hose till the hose is full, and pause , till the water is consumed by destination.

Once the hose is empty, pour again from the source , till all the water is complete from the source bucket. This is referred to as **BACK PRESSURE**

The amount of water that a HOSE can handle is called, **WATERMARK** in nodejs , the parameter is called as **highWaterMark.**

**Note**: The highWaterMark value indicates the size of the hose that is used to transfer water from source to destination. As shown below , look at the size of the hose, highwatermark

A picture containing icon

Description automatically generated

[**PIPES**](https://www.becomebetterprogrammer.com/what-does-pipe-mean-in-node-js-how-to-use-it-practical-guide/)**:**

* Pipe() is available only obn readable stream, it is NOT there on writable stream

Text

Description automatically generated

* Pipes for streams are wat middleware is to request and response

**Duplex Streams:**

**Readstream + writestream = duplex**

* Duplex stream refer to middlesection of the pipeline.

const {Duplex} = require(‘stream’)

Class DuplexDemo extends Duplex{

//We will have to implement both read and write methods

\_write(){}

\_read(){}

}

NOTE: Node has built in duplex called as **PassThrough** const {Passthrough} = require(‘stream’)