8. Streams

Monday, June 12, 2023 1:49 PM

Used to process (read & write) data I/O
piece by piece (chunks),
without completing the whole read or write operation,
and thus without keeping all the data in memory.

When we read a file using streams:
It reads part of the data -> do sth with it->
free memory -> repeat until whole file is processed

Similar to Netflix & Youtube, every clip
is read piece by piece => return to users asap
when it **finishes reading the parts**

4 types of Streams:	Description	Example	Important Events & Funcs
1. Readable streams	Streams from which we can consume data All streams can emit & listen to named events	-http requests -fs read streams	-data -end **pipe() read()
2. Writable streams	Streams to which we can write data	-http responses -fs write streams	-drain -finish write() end()

3. Duplex streams	Streams are both readable & writable	net web socket	
4. Transform streams	Duplex streams that transform data as it is written or read	zlib Gzip creation	

Streams are instances of the EventEmitter class!
We'd rather learn 'how to consume Streams',
instead of implementing Streams!

====Streams in Practice
Reading a large size .txt file => send to clients

streams.js:

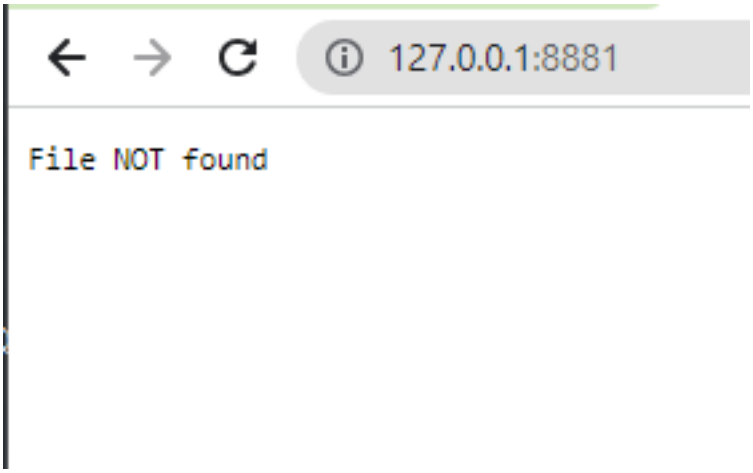
```
const fs = require('fs');
const server = require('http').createServer();
// Listen to a request event
server.on('request', (req, res) => {
    // Solution 1
    // Node.js will have to load entire large text file before
    // sending back data to clients
    // App will crash
    // fs.readFile('./test-file.txt', (err, data) => {
    //     if (err) console.log(err);
    //     res.end(data);
    // });

    // Solution 2
    // Create a Stream to consume data piece by piece
    // Return each chunk of data to clients
    const readable = fs.createReadStream('./testtt-file.txt');
    // readable.on('data', chunk => {res.write(chunk)}) -->
    // readable.on('end', ()=>{res.end()})
    readable.on('data', chunk => {
        // Write it to a writable stream
        res.write(chunk);
    })
    // When stream is done reading entire file
```

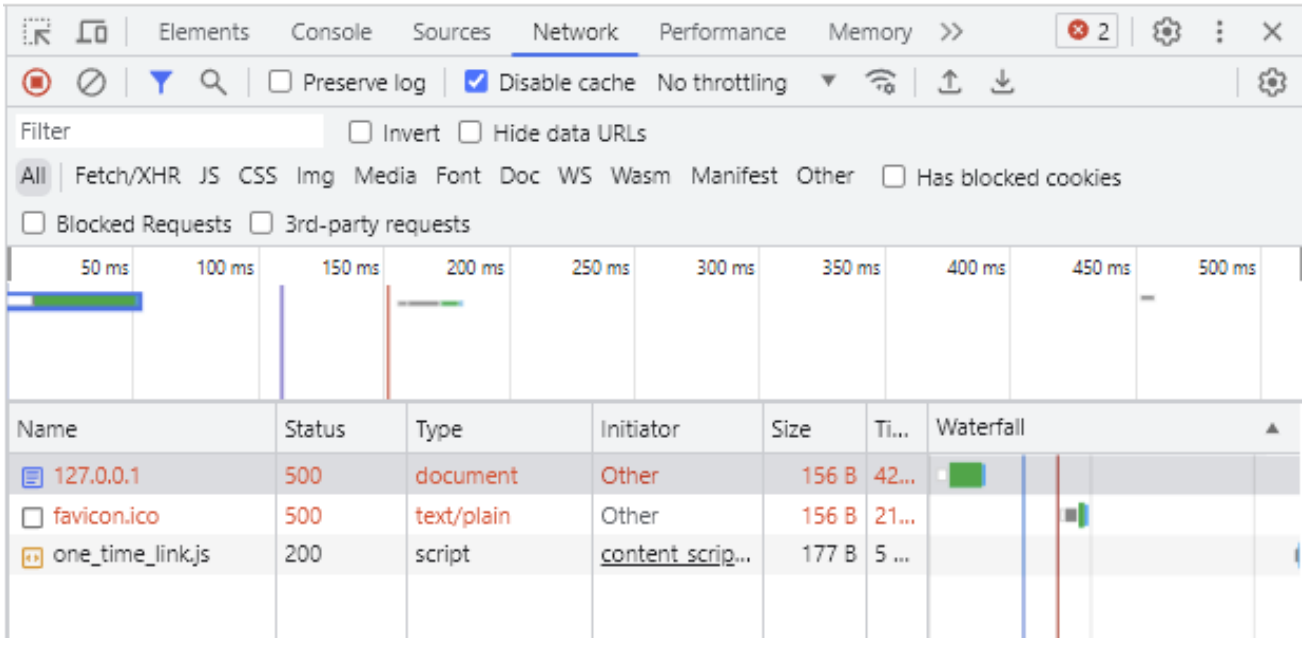
```
    readable.on('end', () => {
        res.end();
    });
    // Error
    readable.on('error', err => {
        console.log(err);
        // if using express.js
        // res.status(500);
        res.writeHead(500, {
            'Content-type': 'text/html',
            'Custom-header': 'ooops, page NOT found :(',
            'Status-code': res.statusCode = 500,
        })
        res.statusCode = 500; // Server error
        res.end('<h1>File NOT found<h1>');
    });


     // Solution 3
     // To overcome Back Pressure issues
     // Use pipe() on all 'Readable Streams'
     // to pipe OUTPUT of Readable Streams right into INPUT of Writable
     // Auto-handle speed of coming in & speed of going out
     const readable = fs.createReadStream('./test-file.txt');
     readable.pipe(res);
     // readableSource.pipe(writeableDestination);
     // readableSource.pipe(duplexStream);
     // readableSource.pipe(transformStream);

});
const localhost = '127.0.0.1';
const port = 8881;
server.listen(port, localhost, () => {
    console.log(`Server has been started on ${localhost}:${port}`);
})
```

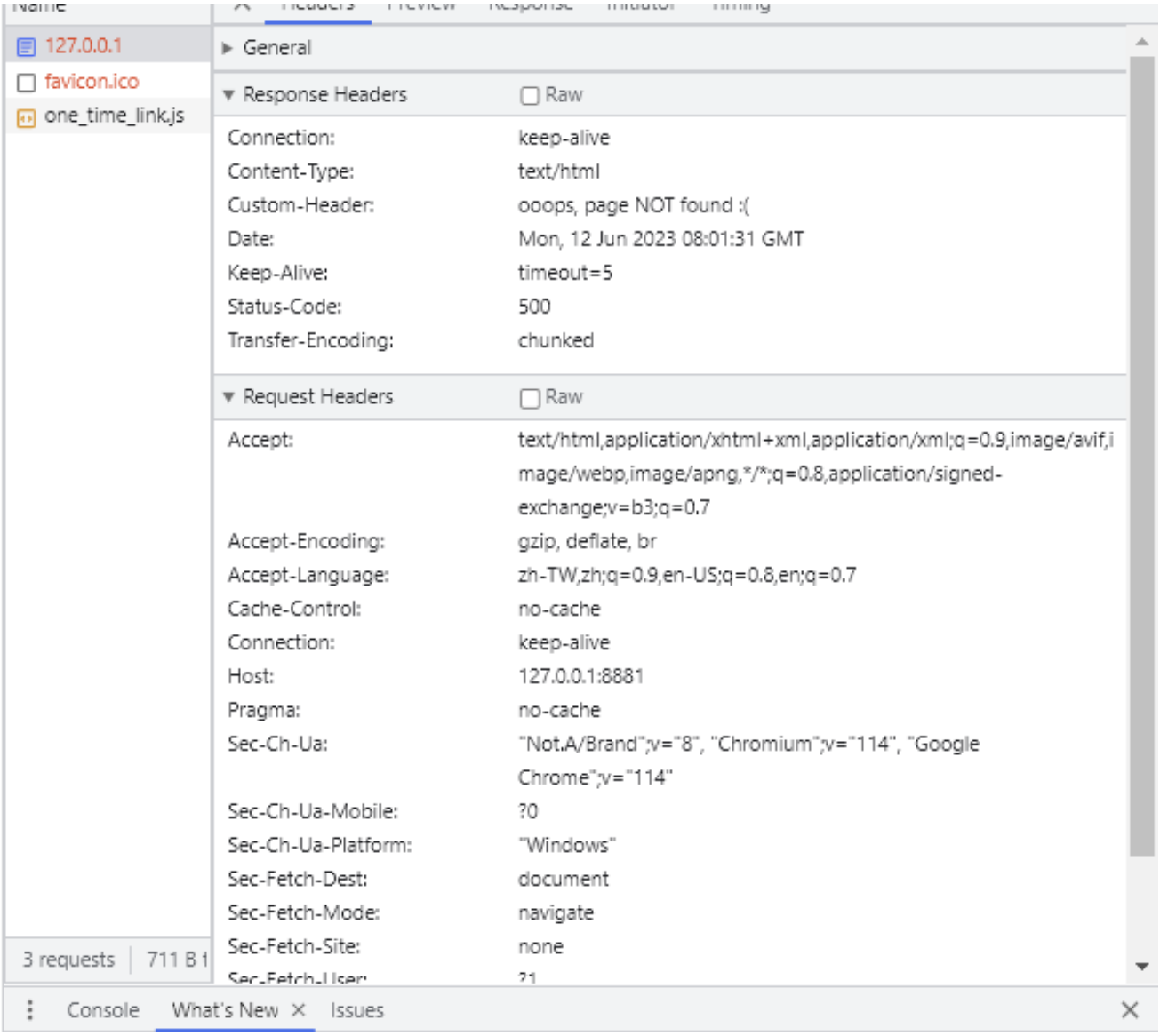

```
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
```

Streams

```
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
Node.js is the best!
```






Name
127.0.0.1
favicon.ico
one_time_link.js
3 requests
711 B t
Headers
Preview
Response
Initiator
Timing
General
Response Headers
Raw
Connection: keep-alive
Content-Type: text/html
Custom-Header: ooops, page NOT found :(
Date: Mon, 12 Jun 2023 08:01:31 GMT
Keep-Alive: timeout=5
Status-Code: 500
Transfer-Encoding: chunked
Request Headers
Raw
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-TW,zh;q=0.9,en-US;q=0.8,en;q=0.7
Cache-Control: no-cache
Connection: keep-alive
Host: 127.0.0.1:8881
Pragma: no-cache
Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="114", "Google Chrome";v="114"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Console
What's New
Issues