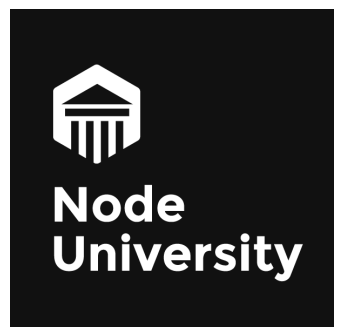


HTTP/2 with Node and Express

Implementing Future Web



Azat Mardan @azat_co



Slides

PDF+Markdown: <https://github.com/azat-co/h2-node>

HTTP/2

It's here.

Really is here

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			<div>249</div>						
			<div>2451</div>			<div>29.2</div>		<div>4.4</div>	
<div>8</div>	<div>213</div>	<div>247</div>	<div>2452</div>			<div>29.3</div>		<div>4.4.4</div>	
<div>1211</div>	<div>214</div>	<div>248</div>	<div>2453</div>	<div>239.1</div>	<div>239</div>	<div>210</div>	<div>all</div>	<div>252</div>	<div>2451</div>
		<div>249</div>	<div>2454</div>	<div>2310</div>	<div>240</div>				
		<div>250</div>	<div>2455</div>	<div>23TP</div>	<div>241</div>				
		<div>251</div>	<div>2456</div>						

<http://caniuse.com/#feat=http2>

History of H2

Started at SPDY at Google

Benefits of H2

Multiplexing

Multiple requests in a single TCP connection means browsers can request all assets in parallel.

Server push

Servers can push assets, such as CSS, JS, and images, before a browser knows it needs them

Faster load times by utilizing
render time and reducing number
of requests.

Without Server Push

1. Client requests HTML and server responds
2. Browser renders HTML
3. Browser requests additional assets: images, scripts, CSS
4. Server responds
5. Browser renders page with assets

Server Push Scenario

1. Client requests HTML and server responds
2. Server knows HTML will need certain assets, and pushes assets while browser is busy rendering HTML
3. Browser renders HTML and uses pushed assets instead of requesting them

Browser will only use pushed assets if it needs them.

Stream priority

Stream priority allows browsers to specify priority of assets. For example, browser can request HTML first to render it before any styles or JavaScript.

Header compression

All HTTP/1.1 requests have to have headers which are typically duplicate the same info.

H2 forces all HTTP headers to be sent in a compressed format.

Encryption*

* De facto mandatory encryption

Although the encryption is not required, most major browsers implement H2 only over TLS (HTTPS).

TL;DR: Old methods of
HTTP/1 might not
work and might even
harm!

Examples: Domain sharding, file
concatenation, sprites

HTTP/2 with Node and Express

H2 Express Server



Azat Mardan @azat_co



Let's Get Our Hands
Dirty

SSL Key+Cert

```
$ mkdir http2-express
$ cd http2-express
$ openssl genrsa -des3 -passout pass:x -out server.pass.key 2048
...
$ openssl rsa -passin pass:x -in server.pass.key -out server.key
writing RSA key
$ rm server.pass.key
$ openssl req -new -key server.key -out server.csr
...
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
...
A challenge password []:
...
$ openssl x509 -req -sha256 -days 365 -in server.csr -signkey server.key -out server.crt
```

H/2 and Node

>> spdy – like

>> http2 – not like

>> core http2 (based on nghttp2) – coming! ([GitHub issue](#))

spdy

```
npm init
```

```
npm i express spdy -S
```

```
const port = 3000
const spdy = require('spdy')
const express = require('express')
const path = require('path')
const fs = require('fs')

const app = express()
```

```
app.get( '*', (req, res) => {  
    res  
        .status(200)  
        .json( {message: 'ok' } )  
    })
```

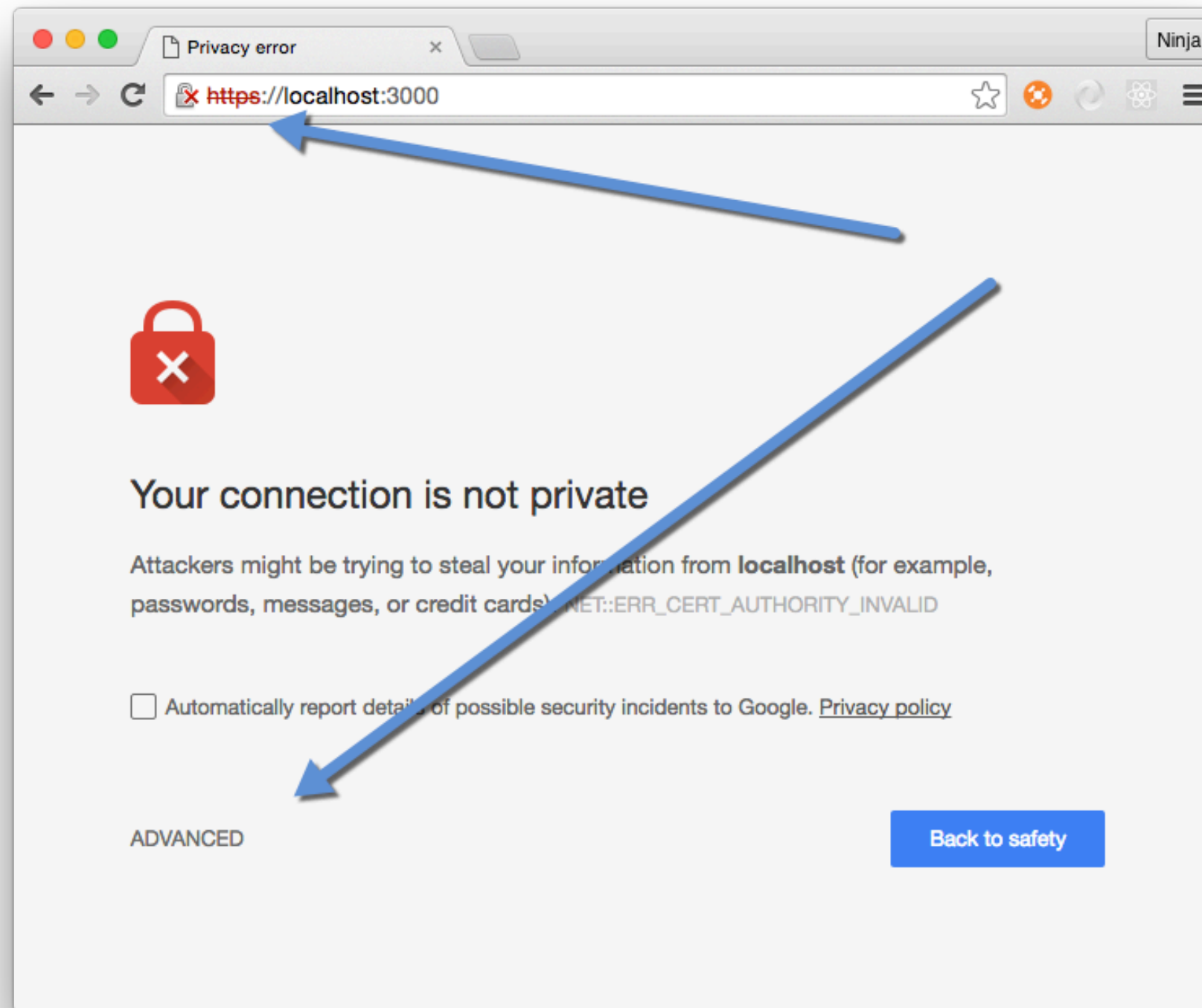
```
const options = {  
    key: fs.readFileSync(__dirname + ' /server.key'),  
    cert: fs.readFileSync(__dirname + ' /server.crt')  
}
```

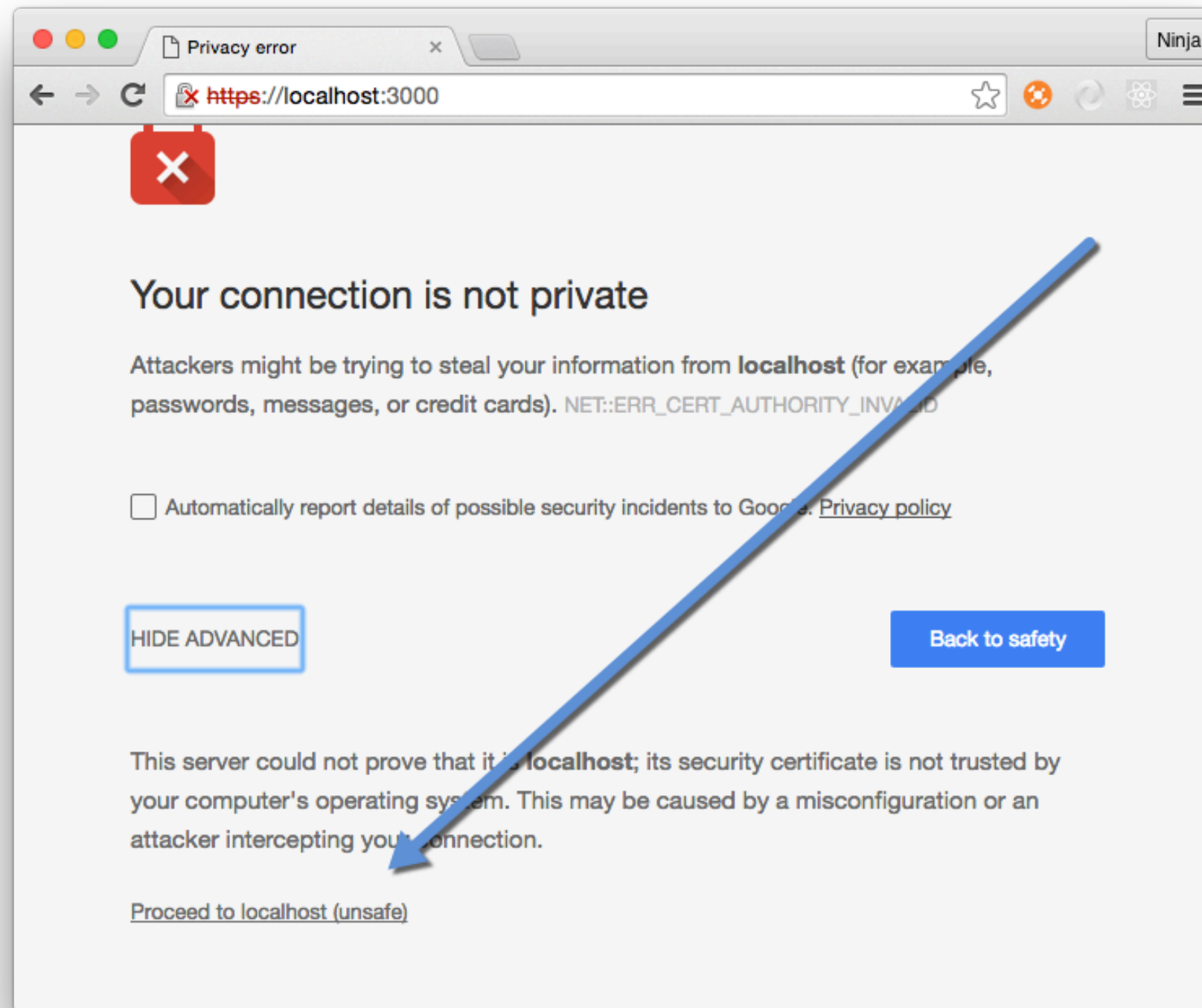

spdy

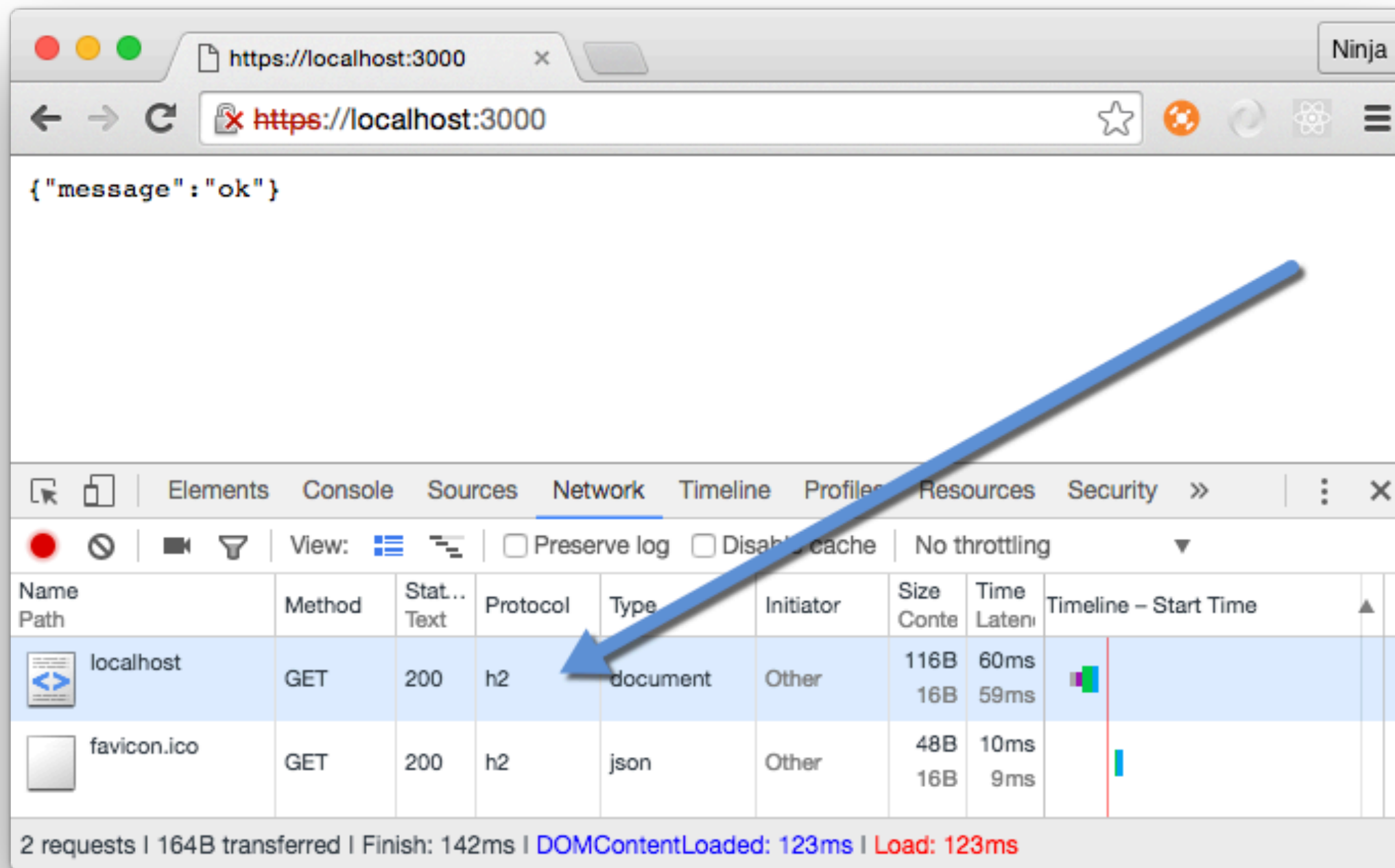
```
.createServer(options, app)  
.listen(port, (error) => {  
  if (error) {  
    console.error(error)  
    return process.exit(1)  
  } else {  
    console.log('Listening on port: ' + port + '.')  
  }  
})
```

Start

node server







Using CURL

curl https://localhost:3000/ -k

Make sure you got the latest version 7.46 with nghttp2

```
★ Code $ curl https://localhost:3000/ -kiv
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
* CAfile: /usr/local/etc/openssl/cert.pem
  Capath: none
* TLSv1.2 (OUT), TLS header, Certificate Status (22):
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* ALPN, server accepted to use h2
* Server certificate:
* subject: C=AU; ST=Some-State; O=Internet Widgits Pty Ltd
* start date: Jul  7 18:50:36 2016 GMT
* expire date: Jul  7 18:50:36 2017 GMT
* issuer: C=AU; ST=Some-State; O=Internet Widgits Pty Ltd
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
```


HTTP/2 with Node and Express

Server Push



Azat Mardan @azat_co



Server Push—Yeah!

```
const http2 = require('spdy')
const logger = require('morgan')
const express = require('express')
const app = express()
const fs = require('fs')
```

```
app.use(logger( 'dev' ))
```

```
app.get('/', (req, res) => {  
    res.send(`hello, http2!  
go to /pushy`)  
})
```

```
app.get('/pushy', (req, res) => {  
  var stream = res.push('/main.js', {  
    status: 200, // optional  
    method: 'GET', // optional  
    request: {  
      accept: '*/*'  
    },  
    response: {  
      'content-type': 'application/javascript'  
    }  
  })  
  stream.on('error', () => {  
  })  
  stream.end('alert("hello from push stream!");')  
  res.end('<script src="/main.js"></script>')  
})
```

```
var options = {  
  key: fs.readFileSync('./server.key'),  
  cert: fs.readFileSync('./server.crt')  
}
```

http2

```
  .createServer(options, app)  
  .listen(8080, ()=>{  
    console.log(`Server is listening on https://localhost:8080.  
You can open the URL in the browser.`)  
  })  
)
```

Result

GET /pushy 200 4.918 ms - -

https://localhost:8080/pushy x Measure Resource Loading Tir x Ninja

661px x 150px

localhost:8080 says:
hello from push stream!
☐ Prevent this page from creating additional dialogs.
OK

Initiator is Push

has green

no green

Name	Path	Met...	Sta...	Protocol	Type	Initiator	Size	Time	Timeline
			Text				Conte	Laten	
main.js		GET	200	h2	script	Push / <u>pushy:1</u> Parser	67B 33B	4ms 3ms	
pushy		GET	200	h2	docum...	Other	79B 32B	10ms 9m	

2 requests | 146B transferred | Finish: 38ms

HTTP/2 with Node and Express

Server Push Express Middleware



Azat Mardan @azat_co



Demo

<https://github.com/azat-co/http2-node-server-push/blob/master/index-advanced.js>

30-Second Summary

1. `spdy` or `http2` core (soon)
2. Express rocks
3. Server Push – yeah!
4. Server Push Express middleware 👍

Code

>> <https://github.com/azat-co/http2-express>

>> <https://github.com/azat-co/http2-node-server-push>

Slides



PDF+Markdown: <https://github.com/azat-co/h2-node>