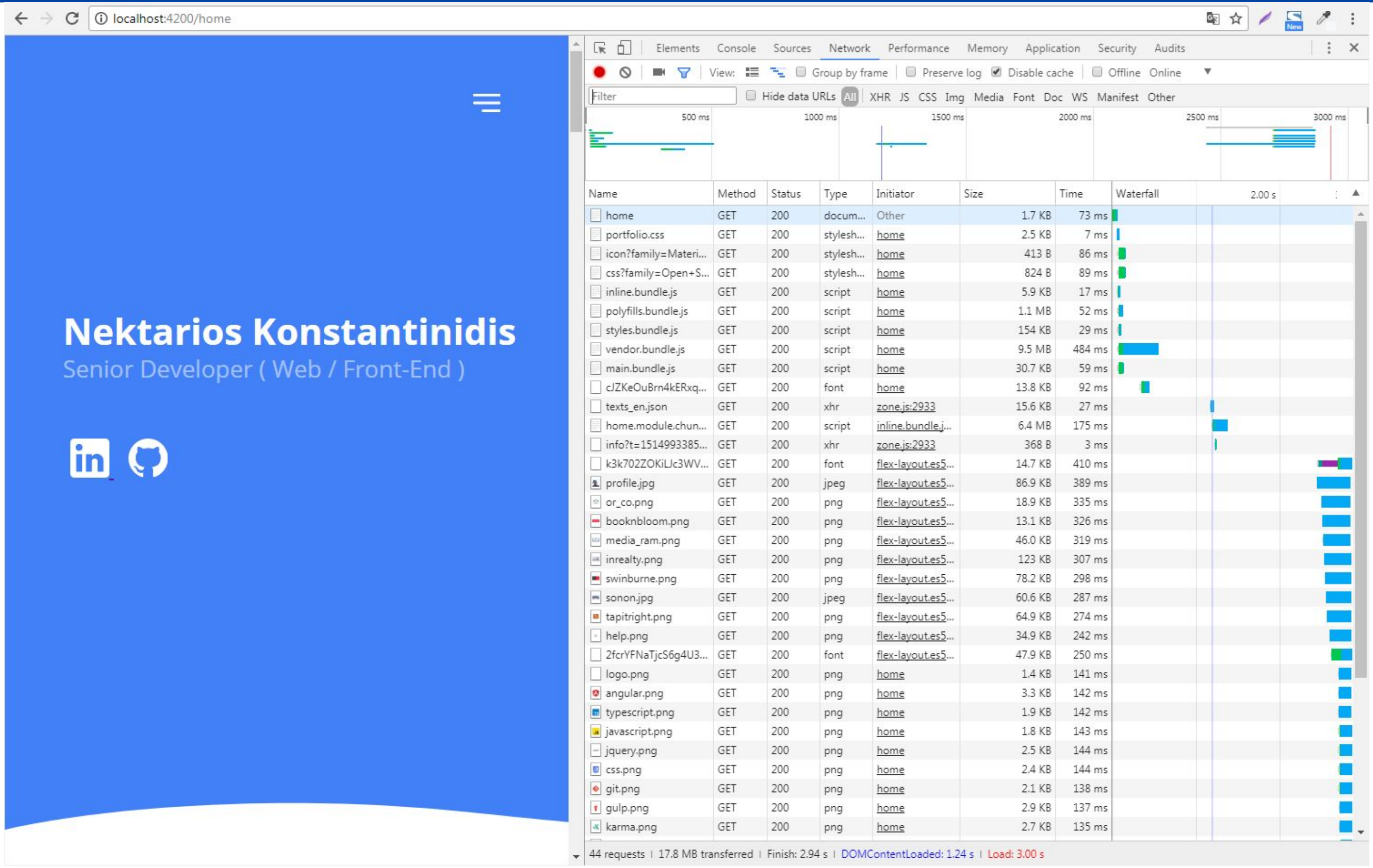
Angular Optimizations

*ngThess (meetup2)

January 5, 2018

Optimizations

1.a. Without optimizations: ng serve


localhost:4200/home
Elements Console Sources Network Performance Memory Application Security Audits
View: Group by frame Preserve log Disable cache Offline Online
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other
500 ms 1000 ms 1500 ms 2000 ms 2500 ms 3000 ms
Waterfall 2.00 s

Nektarios Konstantinidis
Senior Developer (Web / Front-End)

Name	Method	Status	Type	Initiator	Size	Time
home	GET	200	docum...	Other	1.7 KB	73 ms
portfolio.css	GET	200	stylesh...	home	2.5 KB	7 ms
icon?family=Materi...	GET	200	stylesh...	home	413 B	86 ms
css?family=Open+S...	GET	200	stylesh...	home	824 B	89 ms
inline.bundle.js	GET	200	script	home	5.9 KB	17 ms
polyfills.bundle.js	GET	200	script	home	1.1 MB	52 ms
styles.bundle.js	GET	200	script	home	154 KB	29 ms
vendor.bundle.js	GET	200	script	home	9.5 MB	484 ms
main.bundle.js	GET	200	script	home	30.7 KB	59 ms
cJZKeOuBrn4kERxq...	GET	200	font	home	13.8 KB	92 ms
texts_en.json	GET	200	xhr	zone.js:2933	15.6 KB	27 ms
home.module.chun...	GET	200	script	inline.bundle.j...	6.4 MB	175 ms
info?t=1514993385...	GET	200	xhr	zone.js:2933	368 B	3 ms
k3k702ZOKiLJc3WV...	GET	200	font	flex-layout.es5...	14.7 KB	410 ms
profile.jpg	GET	200	jpeg	flex-layout.es5...	86.9 KB	389 ms
or_co.png	GET	200	png	flex-layout.es5...	18.9 KB	335 ms
booknbloom.png	GET	200	png	flex-layout.es5...	13.1 KB	326 ms
media_ram.png	GET	200	png	flex-layout.es5...	46.0 KB	319 ms
inrealty.png	GET	200	png	flex-layout.es5...	123 KB	307 ms
swinburne.png	GET	200	png	flex-layout.es5...	78.2 KB	298 ms
sonon.jpg	GET	200	jpeg	flex-layout.es5...	60.6 KB	287 ms
tapitright.png	GET	200	png	flex-layout.es5...	64.9 KB	274 ms
help.png	GET	200	png	flex-layout.es5...	34.9 KB	242 ms
2fcrYFNaTjcS6g4U3...	GET	200	font	flex-layout.es5...	47.9 KB	250 ms
logo.png	GET	200	png	home	1.4 KB	141 ms
angular.png	GET	200	png	home	3.3 KB	142 ms
typescript.png	GET	200	png	home	1.9 KB	142 ms
javascript.png	GET	200	png	home	1.8 KB	143 ms
jquery.png	GET	200	png	home	2.5 KB	144 ms
css.png	GET	200	png	home	2.4 KB	144 ms
git.png	GET	200	png	home	2.1 KB	138 ms
gulp.png	GET	200	png	home	2.9 KB	137 ms
karma.png	GET	200	png	home	2.7 KB	135 ms

44 requests | 17.8 MB transferred | Finish: 2.94 s | DOMContentLoaded: 1.24 s | Load: 3.00 s

On serve, application size is **17.8MB.**

The large size of js files can be explained by the fact that they include sourcemaps, which are helpful and used during debugging.

Speed is still ok because everything is loaded locally from localhost.

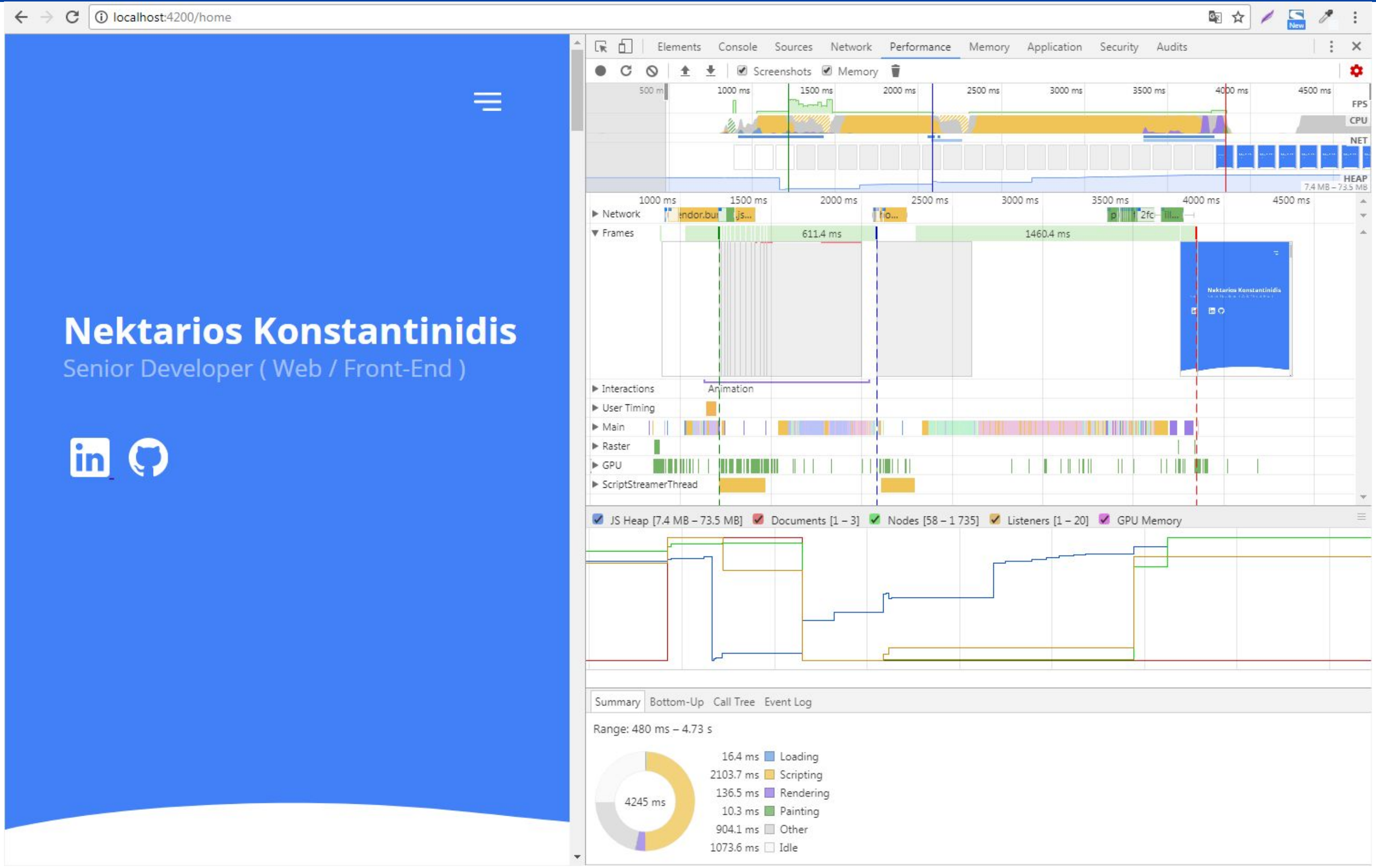
1.a. Without optimizations: ng serve

Source Maps

.map files are source map files that let tools map between the emitted JavaScript code and the TypeScript source files that created it. Many debuggers (e.g. Visual Studio or Chrome's dev tools) can consume these files so you can debug the TypeScript file instead of the JavaScript file.

This is the same source map format being produced by some minifiers and other compiled-to-JS languages like CoffeeScript.

1.a. Without optimizations: ng serve



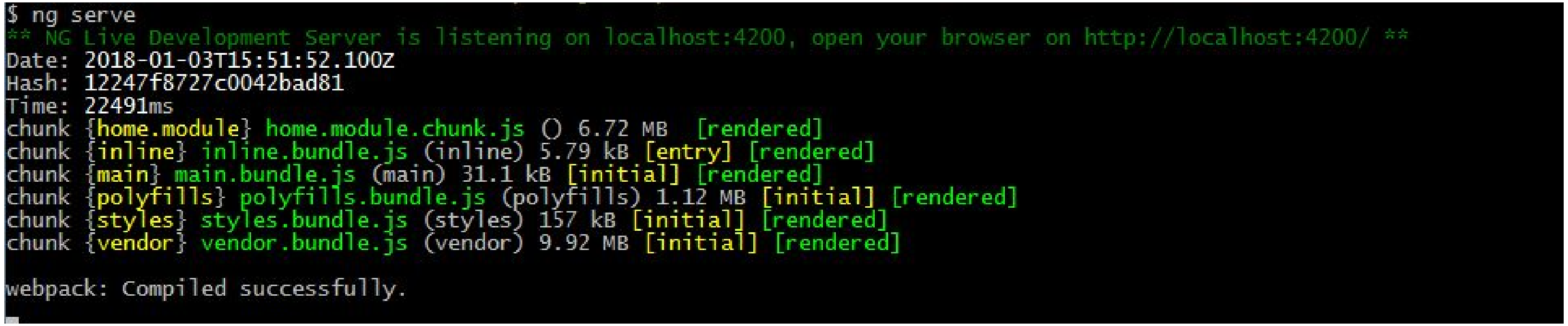
Range: 480 ms – 4.73 s

16.4 ms	Loading
2103.7 ms	Scripting
136.5 ms	Rendering
10.3 ms	Painting
904.1 ms	Other
1073.6 ms	Idle

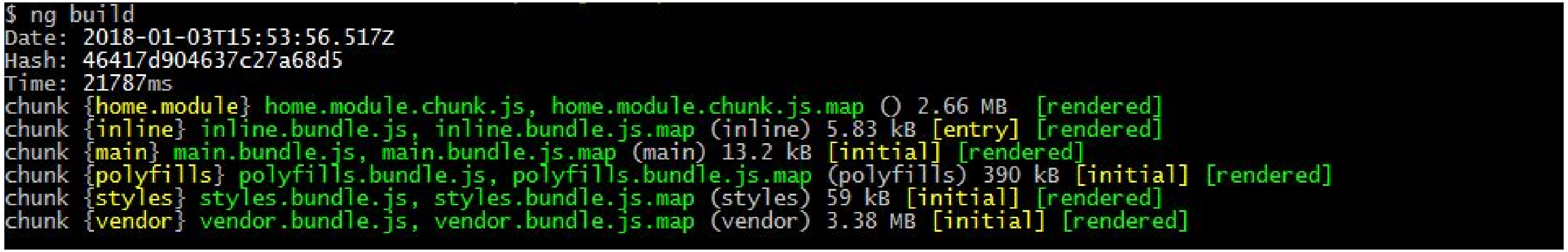
On serve the view is rendered in **3 seconds.**

The blue line represents the DOMContentLoaded event. The green line represents time to first paint. The red line represents the load event.

1.b. Without optimizations: ng build



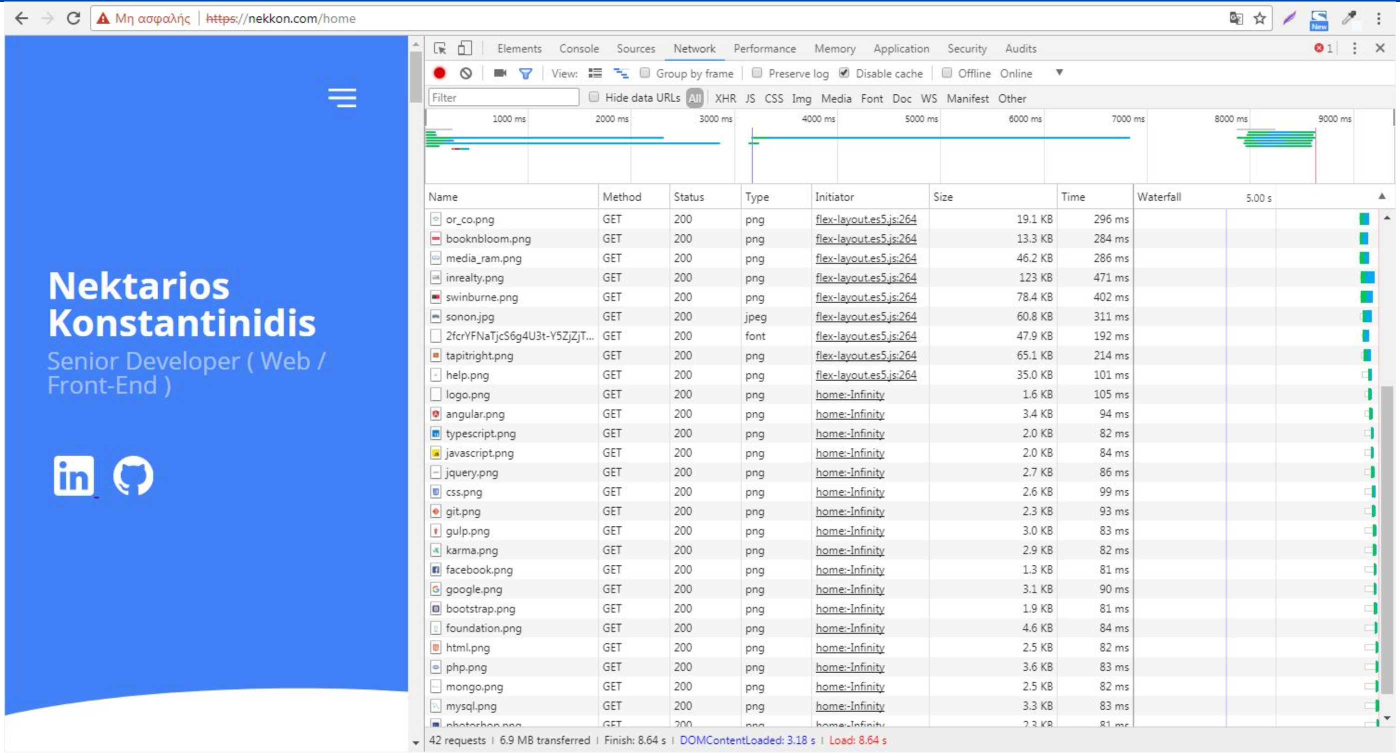
```
$ ng serve
** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
Date: 2018-01-03T15:51:52.100Z
Hash: 12247f8727c0042bad81
Time: 22491ms
chunk {home.module} home.module.chunk.js () 6.72 MB  [rendered]
chunk {inline} inline.bundle.js (inline) 5.79 kB [entry] [rendered]
chunk {main} main.bundle.js (main) 31.1 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js (polyfills) 1.12 MB [initial] [rendered]
chunk {styles} styles.bundle.js (styles) 157 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js (vendor) 9.92 MB [initial] [rendered]

webpack: Compiled successfully.
```



```
$ ng build
Date: 2018-01-03T15:53:56.517Z
Hash: 46417d904637c27a68d5
Time: 21787ms
chunk {home.module} home.module.chunk.js, home.module.chunk.js.map () 2.66 MB  [rendered]
chunk {inline} inline.bundle.js, inline.bundle.js.map (inline) 5.83 kB [entry] [rendered]
chunk {main} main.bundle.js, main.bundle.js.map (main) 13.2 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 390 kB [initial] [rendered]
chunk {styles} styles.bundle.js, styles.bundle.js.map (styles) 59 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js, vendor.bundle.js.map (vendor) 3.38 MB [initial] [rendered]
```

We can see a **63.75%** reduction of js code size from a simple build of the project. (From 17.954MB to 6.508MB). The main difference is that the build command separates code and sourcemap files.

1.b. Without optimizations: ng build



Μη ασφαλής | ~~https~~://nekkon.com/home

Nektarios Konstantinidis
Senior Developer (Web / Front-End)

Elements Console Sources Network Performance Memory Application Security Audits

View: Group by frame | Preserve log | [✓ Disable cache] | Offline Online

Filter | Hide data URLs | All XHR JS CSS Img Media Font Doc WS Manifest Other

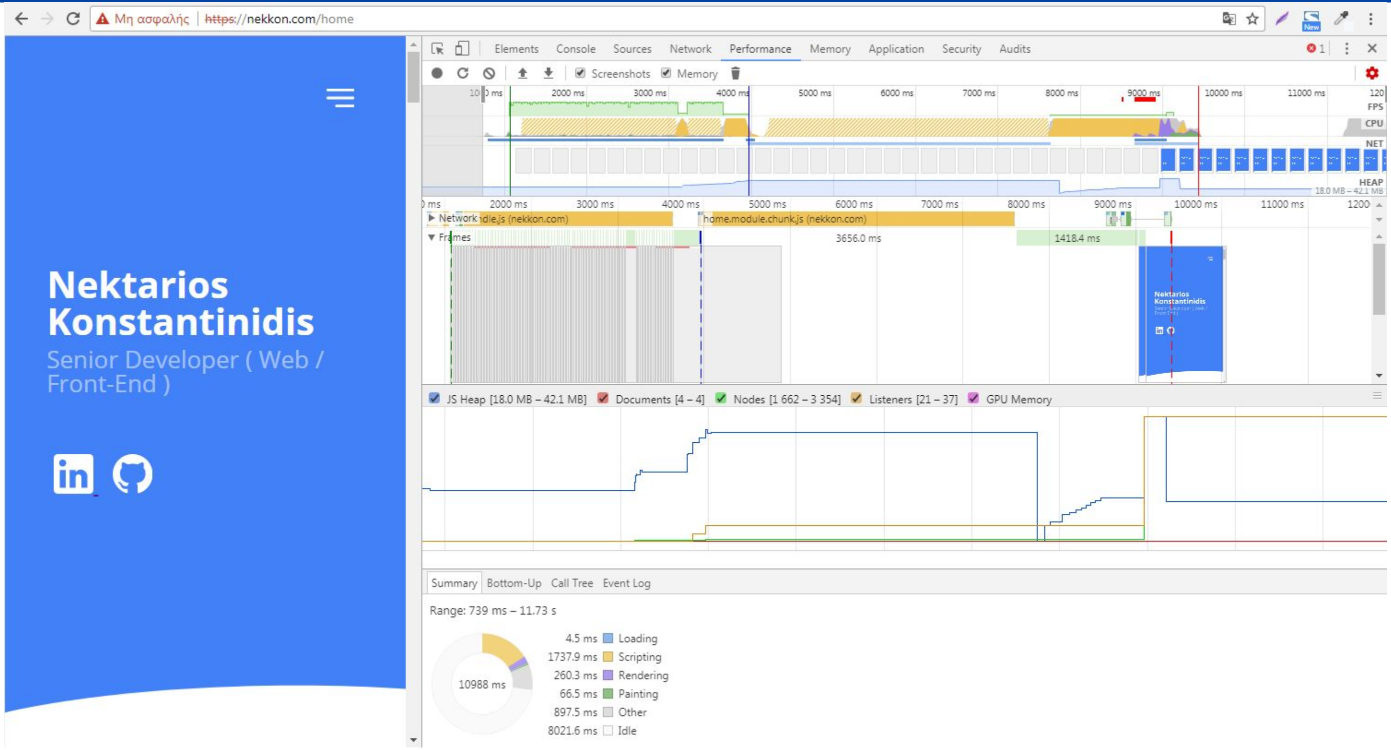
Name	Method	Status	Type	Initiator	Size	Time
or_co.png	GET	200	png	flex-layout.es5.js:264	19.1 KB	296 ms
booknbloom.png	GET	200	png	flex-layout.es5.js:264	13.3 KB	284 ms
media_ram.png	GET	200	png	flex-layout.es5.js:264	46.2 KB	286 ms
inrealty.png	GET	200	png	flex-layout.es5.js:264	123 KB	471 ms
swinburne.png	GET	200	png	flex-layout.es5.js:264	78.4 KB	402 ms
sonon.jpg	GET	200	jpeg	flex-layout.es5.js:264	60.8 KB	311 ms
2fcrYFNaTjcS6g4U3t-Y5ZjZjT...	GET	200	font	flex-layout.es5.js:264	47.9 KB	192 ms
tapitright.png	GET	200	png	flex-layout.es5.js:264	65.1 KB	214 ms
help.png	GET	200	png	flex-layout.es5.js:264	35.0 KB	101 ms
logo.png	GET	200	png	home:-Infinity	1.6 KB	105 ms
angular.png	GET	200	png	home:-Infinity	3.4 KB	94 ms
typescript.png	GET	200	png	home:-Infinity	2.0 KB	82 ms
javascript.png	GET	200	png	home:-Infinity	2.0 KB	84 ms
jquery.png	GET	200	png	home:-Infinity	2.7 KB	86 ms
css.png	GET	200	png	home:-Infinity	2.6 KB	99 ms
git.png	GET	200	png	home:-Infinity	2.3 KB	93 ms
gulp.png	GET	200	png	home:-Infinity	3.0 KB	83 ms
karma.png	GET	200	png	home:-Infinity	2.9 KB	82 ms
facebook.png	GET	200	png	home:-Infinity	1.3 KB	81 ms
google.png	GET	200	png	home:-Infinity	3.1 KB	90 ms
bootstrap.png	GET	200	png	home:-Infinity	1.9 KB	81 ms
foundation.png	GET	200	png	home:-Infinity	4.6 KB	84 ms
html.png	GET	200	png	home:-Infinity	2.5 KB	82 ms
php.png	GET	200	png	home:-Infinity	3.6 KB	83 ms
mongo.png	GET	200	png	home:-Infinity	2.5 KB	82 ms
mysql.png	GET	200	png	home:-Infinity	3.3 KB	83 ms
photoshop.png	GET	200	png	home:-Infinity	2.3 KB	81 ms

42 requests | 6.9 MB transferred | Finish: 8.64 s | DOMContentLoaded: 3.18 s | Load: 8.64 s

On build, application size is reduced to **6.9MB. (from 17.8MB)**

Speed is considered slow because everything is loaded from a server.

1.b. Without optimizations: ng build



Range: 739 ms – 11.73 s

4.5 ms	Loading
1737.9 ms	Scripting
260.3 ms	Rendering
66.5 ms	Painting
897.5 ms	Other
8021.6 ms	Idle

On build the view is rendered in **8.64 seconds.**

2. Why Optimize?

One of the things that most impacts User Experience (especially on mobile) is the application startup experience, and perceived performance. In fact, studies have shown that 53% of mobile users abandon sites that take longer than 3 seconds to load!

And this is true for all applications in general, not only mobile applications. Any application can benefit from a much better startup experience, especially if we can get that working out of the box.

One of the things that we can do to improve the user experience is to show something to the user as quickly as possible, reducing the time to first meaningful paint.

3.a Optimizations: Assets / HTTP Requests

Assets folder

The assets folder in an angular app doesn’t get optimized during builds. In the example, this folder contained images and a .json file with the texts.

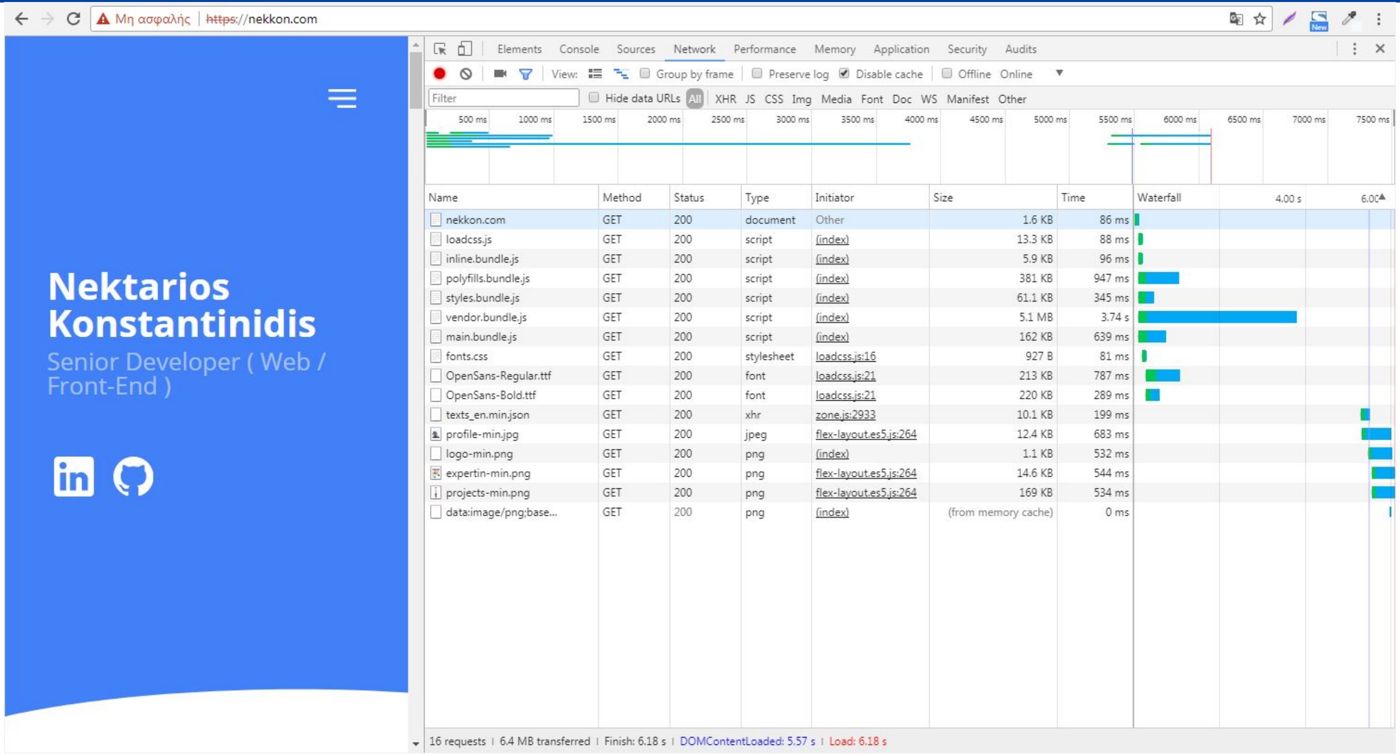
Http Requests

Once your server receives an HTTP request from a user's browser, your server then responds and delivers the files to that user's browser. The user's browser then renders the webpage.
There were many files that could be be combined in one to reduce requests during load. An example are the .png files for Expertin and Projects section.

Tools / Sites used

1. To combine images into one : http://responsive-css.spritegen.com/ and http://css.spritegen.com
2. To generally compress final images http://optimizilla.com/ . This online image optimizer uses a smart combination of the best optimization and lossy compression algorithms to shrink JPEG and PNG images to the minimum possible size while keeping the required level of quality.

3.a Optimizations: Assets / HTTP Requests

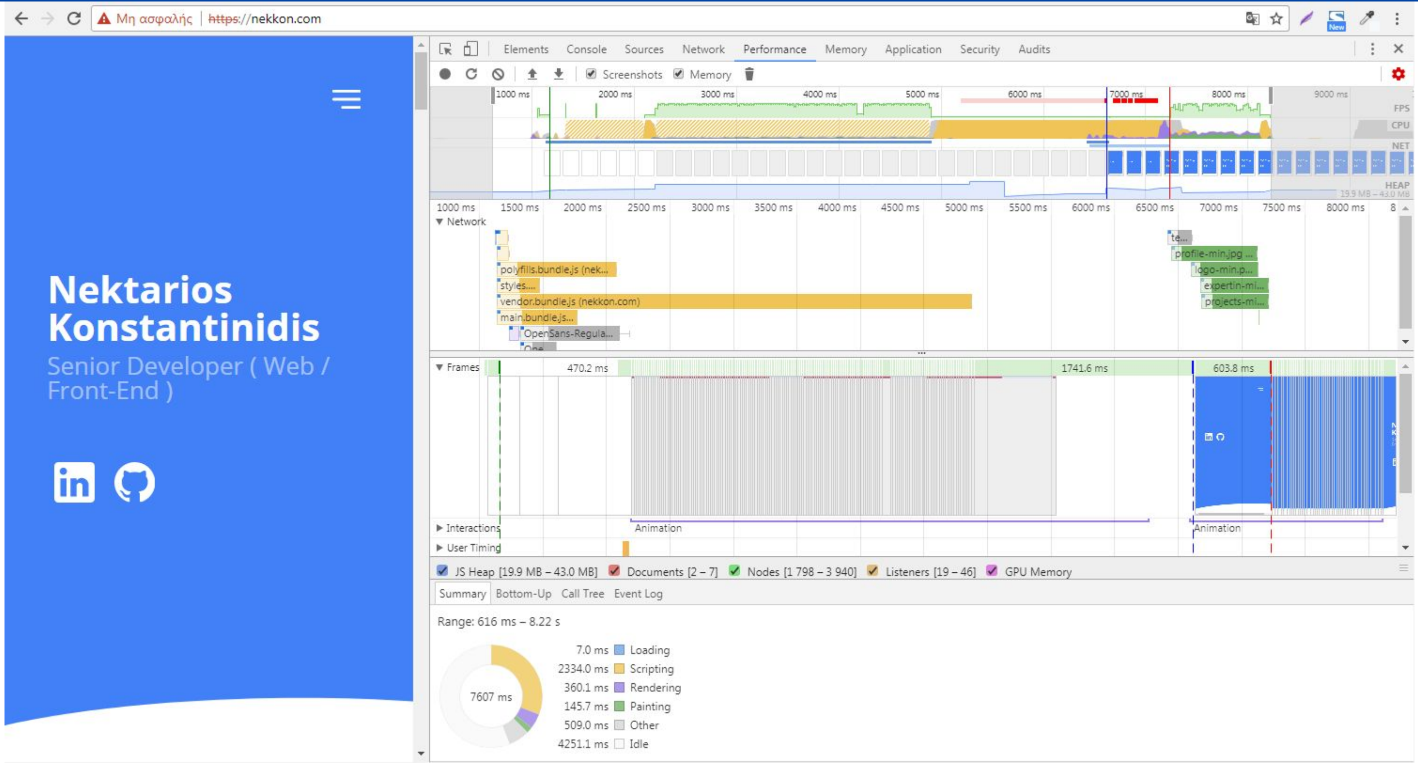

Μη ασφαλής | ~~https~~://nekkon.com
Elements Console Sources Network Performance Memory Application Security Audits
View: Group by frame Preserve log Disable cache Offline Online
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other
Nektarios Konstantinidis
Senior Developer (Web / Front-End)

Name	Method	Status	Type	Initiator	Size	Time
nekkon.com	GET	200	document	Other	1.6 KB	86 ms
loadcss.js	GET	200	script	(index)	13.3 KB	88 ms
inline.bundle.js	GET	200	script	(index)	5.9 KB	96 ms
polyfills.bundle.js	GET	200	script	(index)	381 KB	947 ms
styles.bundle.js	GET	200	script	(index)	61.1 KB	345 ms
vendor.bundle.js	GET	200	script	(index)	5.1 MB	3.74 s
main.bundle.js	GET	200	script	(index)	162 KB	639 ms
fonts.css	GET	200	stylesheet	loadcss.js:16	927 B	81 ms
OpenSans-Regular.ttf	GET	200	font	loadcss.js:21	213 KB	787 ms
OpenSans-Bold.ttf	GET	200	font	loadcss.js:21	220 KB	289 ms
texts_en.min.json	GET	200	xhr	zone.js:2933	10.1 KB	199 ms
profile-min.jpg	GET	200	jpeg	flex-layout.es5.js:264	12.4 KB	683 ms
logo-min.png	GET	200	png	(index)	1.1 KB	532 ms
expertin-min.png	GET	200	png	flex-layout.es5.js:264	14.6 KB	544 ms
projects-min.png	GET	200	png	flex-layout.es5.js:264	169 KB	534 ms
data:image/png;base...	GET	200	png	(index)	(from memory cache)	0 ms

16 requests | 6.4 MB transferred | Finish: 6.18 s | DOMContentLoaded: 5.57 s | Load: 6.18 s

On build, application size is **6.4MB.** (from 6.9MB, 7.25% less)

3.a Optimizations: Assets / HTTP Requests


Μη ασφαλής | https://nekkon.com
Elements Console Sources Network Performance Memory Application Security Audits
Screenshots Memory
Nektarios Konstantinidis
Senior Developer (Web / Front-End)
Network
polyfills.bundle.js (nek...
styles....
vendor.bundle.js (nekkon.com)
main.bundle.js...
OpenSans-Regula...
profile-min.jpg ...
logo-min.p...
expertin-mi...
projects-mi...
Frames
470.2 ms
1741.6 ms
603.8 ms
Interactions
Animation
User Timing
JS Heap [19.9 MB – 43.0 MB] Documents [2 – 7] Nodes [1 798 – 3 940] Listeners [19 – 46] GPU Memory
Summary Bottom-Up Call Tree Event Log

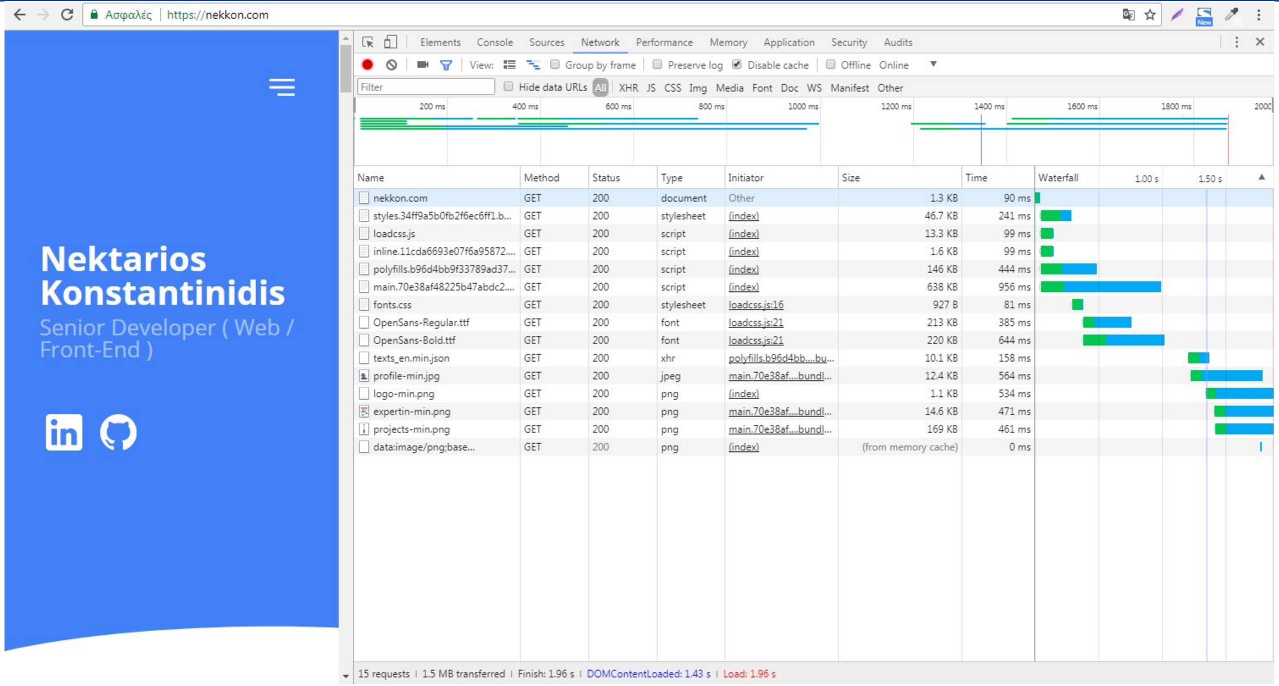
Range: 616 ms – 8.22 s

7607 ms

7.0 ms	Loading
2334.0 ms	Scripting
360.1 ms	Rendering
145.7 ms	Painting
509.0 ms	Other
4251.1 ms	Idle

On build the view is rendered in **5.8 seconds** (from 8.64 seconds, 32.87% less)**.**

3.b Optimizations: ng build -prod


Ασφαλής | https://nekkon.com
Nektarios Konstantinidis
Senior Developer (Web / Front-End)
Elements Console Sources Network Performance Memory Application Security Audits
View: Group by frame Preserve log Disable cache Offline Online
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

Name	Method	Status	Type	Initiator	Size	Time
nekkon.com	GET	200	document	Other	1.3 KB	90 ms
styles.34ff9a5b0fb2f6ec6ff1.b...	GET	200	stylesheet	(index)	46.7 KB	241 ms
loadcss.js	GET	200	script	(index)	13.3 KB	99 ms
inline.11cda6693e07f6a95872....	GET	200	script	(index)	1.6 KB	99 ms
polyfills.b96d4bb9f33789ad37...	GET	200	script	(index)	146 KB	444 ms
main.70e38af48225b47abdc2....	GET	200	script	(index)	638 KB	956 ms
fonts.css	GET	200	stylesheet	loadcss.js:16	927 B	81 ms
OpenSans-Regular.ttf	GET	200	font	loadcss.js:21	213 KB	385 ms
OpenSans-Bold.ttf	GET	200	font	loadcss.js:21	220 KB	644 ms
texts_en.min.json	GET	200	xhr	polyfills.b96d4bb....bu...	10.1 KB	158 ms
profile-min.jpg	GET	200	jpeg	main.70e38af....bundl...	12.4 KB	564 ms
logo-min.png	GET	200	png	(index)	1.1 KB	534 ms
expertin-min.png	GET	200	png	main.70e38af....bundl...	14.6 KB	471 ms
projects-min.png	GET	200	png	main.70e38af....bundl...	169 KB	461 ms
data:image/png;base...	GET	200	png	(index)	(from memory cache)	0 ms

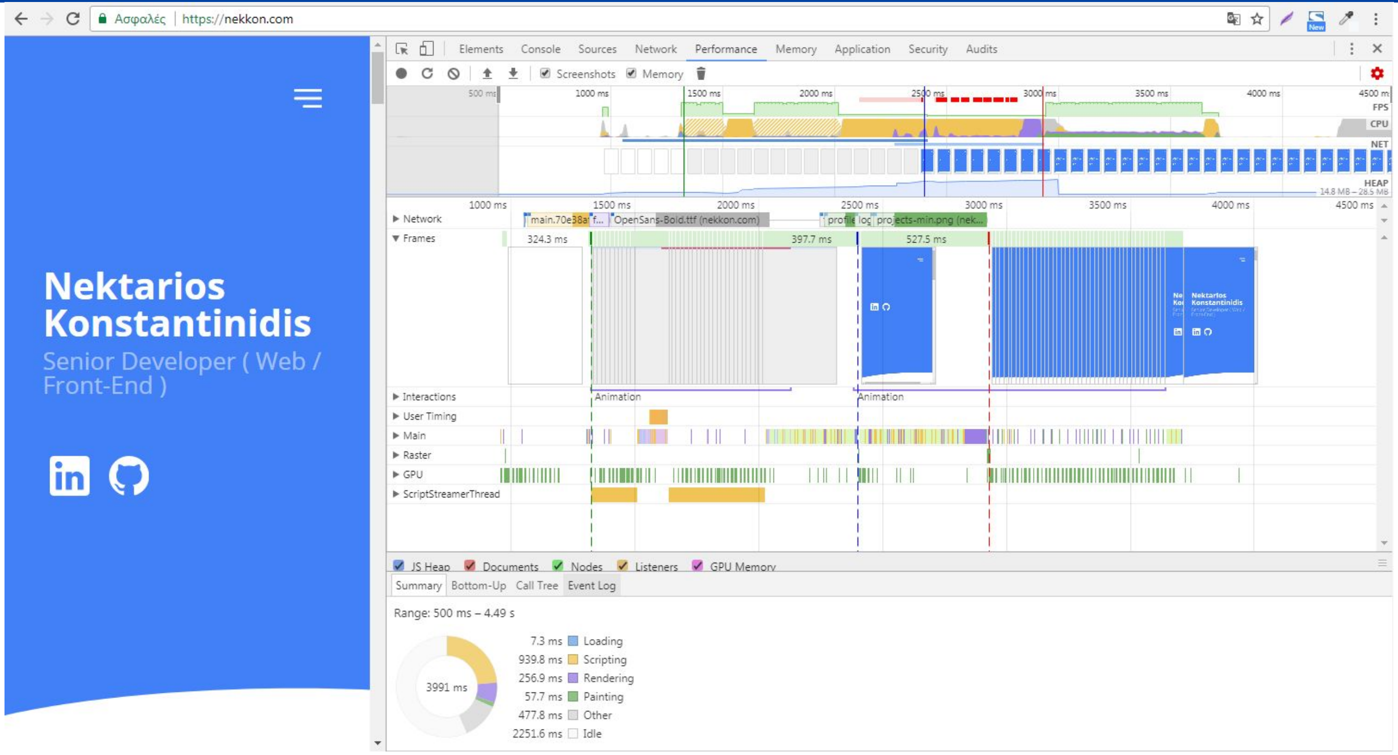
15 requests | 1.5 MB transferred | Finish: 1.96 s | DOMContentLoaded: 1.43 s | Load: 1.96 s

On build with the production flag, application size is reduced to **1.5MB.** (from 6.4MB, 76.6% less)

The --prod meta-flag engages the following optimization features.

Ahead-of-Time (AOT) Compilation: pre-compiles Angular component templates.
Production mode: deploys the production environment which enables production mode.
Bundling: concatenates your many application and library files into a few bundles.
Minification: removes excess whitespace, comments, and optional tokens.
Uglification: rewrites code to use short, cryptic variable and function names.
Dead code elimination: removes unreferenced modules and much unused code (tree-shaking).

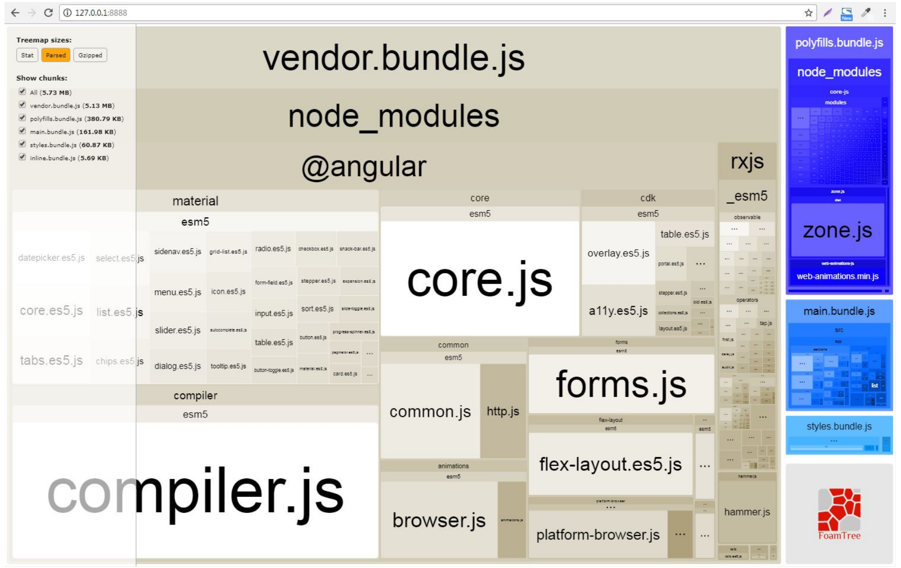
3.b Optimizations: ng build -prod


Ασφαλές | https://nekkon.com
Nektarios Konstantinidis
Senior Developer (Web / Front-End)
Elements Console Sources Network Performance Memory Application Security Audits
Screenshots Memory
FPS
CPU
NET
HEAP
Network
Frames
324.3 ms
397.7 ms
527.5 ms
Interactions
Animation
User Timing
Main
Raster
GPU
ScriptStreamerThread
JS Heap Documents Nodes Listeners GPU Memory
Summary Bottom-Up Call Tree Event Log
3991 ms

```
Range: 500 ms – 4.49 s

   7.3 ms  Loading
 939.8 ms  Scripting
 256.9 ms  Rendering
  57.7 ms  Painting
 477.8 ms  Other
2251.6 ms  Idle
```

On build with the production flag, the view is rendered in **2.5 seconds** (from 5.8 seconds, 56.9% less).

3.b Optimizations: ng build -prod

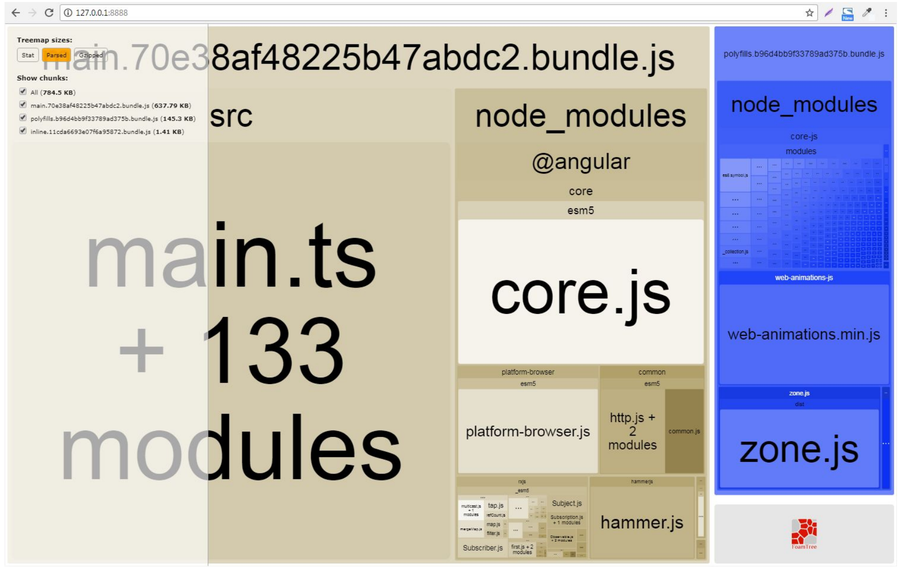


127.0.0.1:8888

Treemap sizes: Stat [✓ Parsed] Gzipped

Show chunks:

- ✓ All (**5.73 MB**)
- ✓ vendor.bundle.js (**5.13 MB**)
- ✓ polyfills.bundle.js (**380.79 KB**)
- ✓ main.bundle.js (**161.98 KB**)
- ✓ styles.bundle.js (**60.87 KB**)
- ✓ inline.bundle.js (**5.69 KB**)



127.0.0.1:8888

Treemap sizes: Stat [✓ Parsed] Gzipped

Show chunks:

- ✓ All (**784.5 KB**)
- ✓ main.70e38af48225b47abdc2.bundle.js (**637.79 KB**)
- ✓ polyfills.b96d4bb9f33789ad375b.bundle.js (**145.3 KB**)
- ✓ inline.11cda6693e07f6a95872.bundle.js (**1.41 KB**)

We can see a **86.3%** reduction of js code size from a production build of the project. (From 5.73MB to 785KB). The main difference is that only the required code from the libraries is kept and added to main.bundle.js, vendor.bundle.js does not exist.

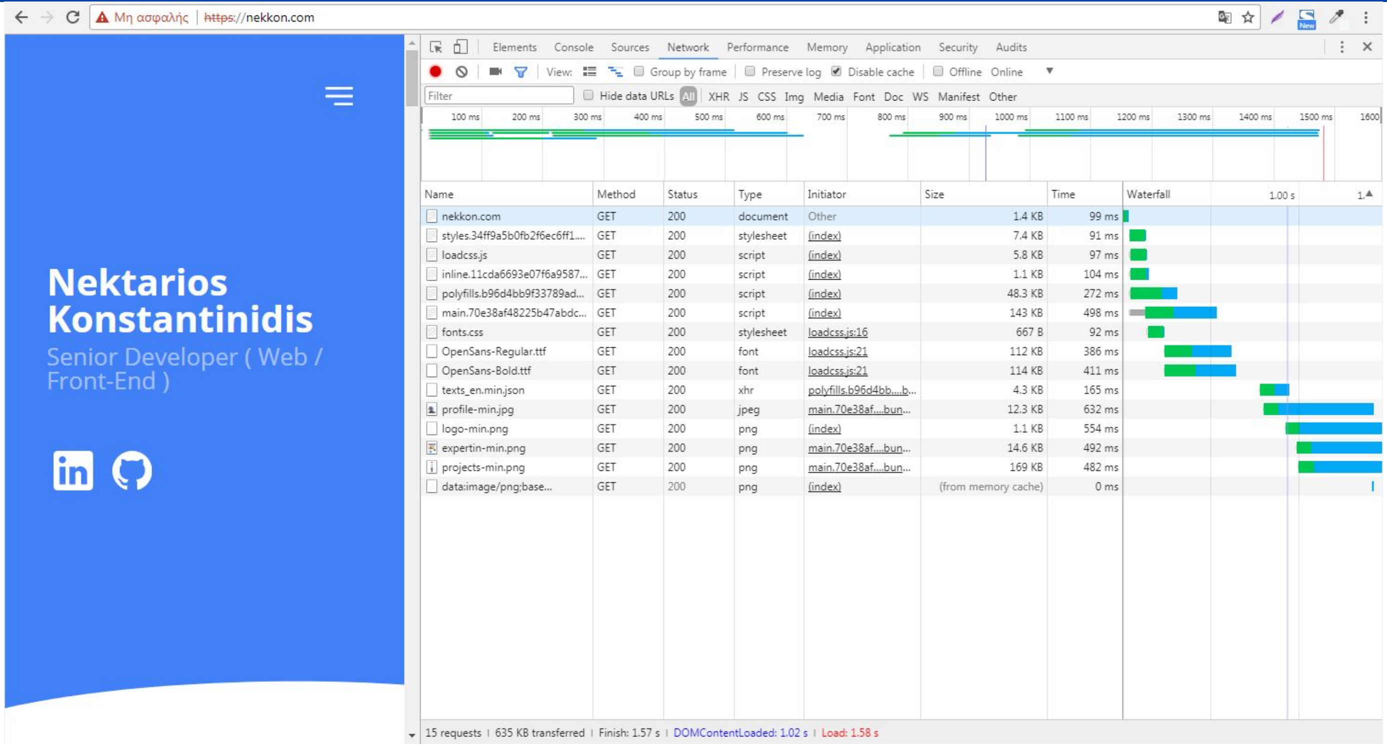
https://coryrylan.com/blog/analyzing-bundle-size-with-the-angular-cli-and-webpack

3.c Optimizations: gzip compression (server)

Server setup

1. Hosting was a VPS with a2hosting.
2. Server OS was Ubuntu 16.04
3. Node.js version 6 was installed along with nginx and pm2
   https://www.digitalocean.com/community/tutorials/how-to-set-up-a-node-js-application-for-production-on-ubuntu-16-04
4. To deploy files a simple gulp script was created by using gulp-sftp
   https://github.com/gtg092x/gulp-sftp
5. gzip was added to nginx settings
   https://www.digitalocean.com/community/tutorials/how-to-add-the-gzip-module-to-nginx-on-ubuntu-16-04
6. Installed free TLS/SSL certificate (Let's Encrypt)
   https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-16-04

3.c Optimizations: gzip compression (server)



Μη ασφαλής | ~~https~~://nekkon.com

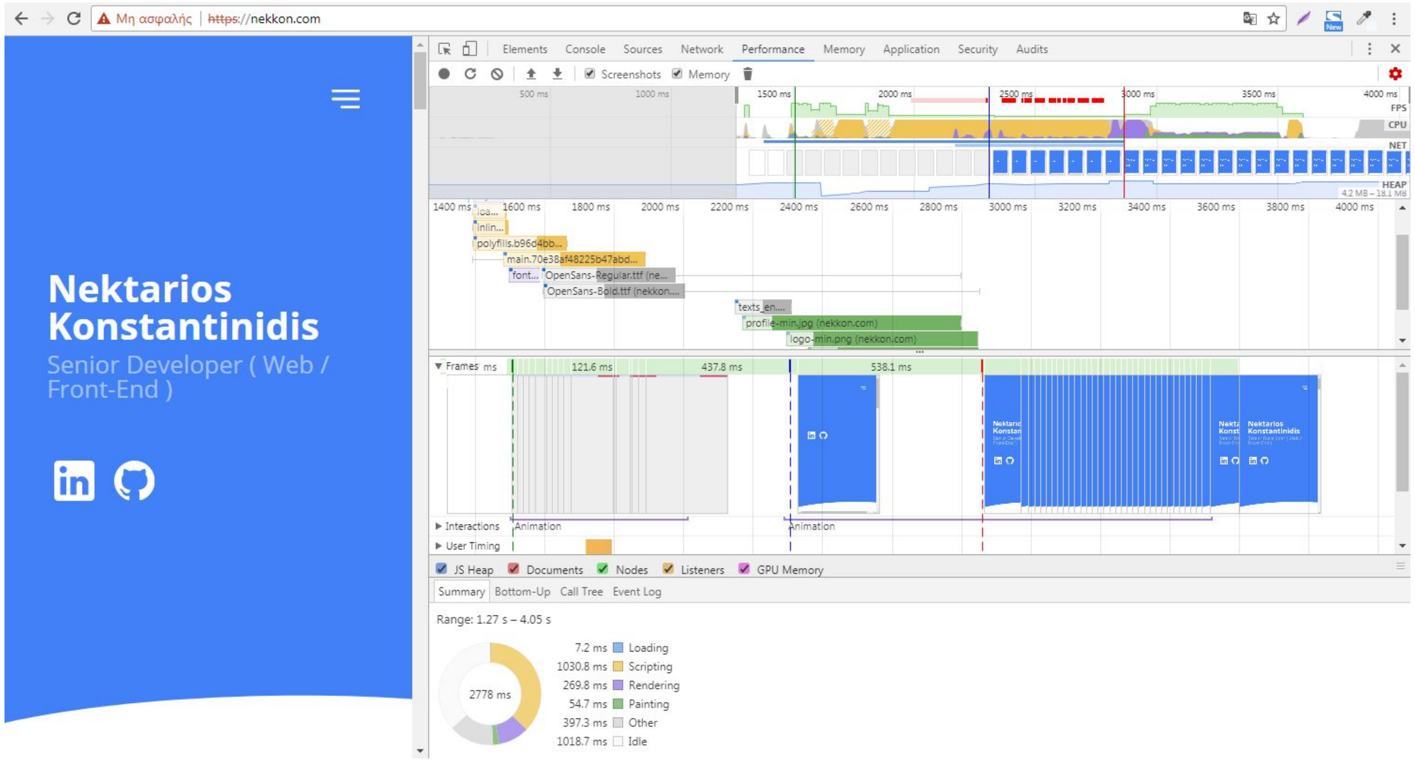
Nektarios Konstantinidis
Senior Developer (Web / Front-End)

Name	Method	Status	Type	Initiator	Size	Time
nekkon.com	GET	200	document	Other	1.4 KB	99 ms
styles.34ff9a5b0fb2f6ec6ff1....	GET	200	stylesheet	(index)	7.4 KB	91 ms
loadcss.js	GET	200	script	(index)	5.8 KB	97 ms
inline.11cda6693e07f6a9587...	GET	200	script	(index)	1.1 KB	104 ms
polyfills.b96d4bb9f33789ad...	GET	200	script	(index)	48.3 KB	272 ms
main.70e38af48225b47abdc...	GET	200	script	(index)	143 KB	498 ms
fonts.css	GET	200	stylesheet	loadcss.js:16	667 B	92 ms
OpenSans-Regular.ttf	GET	200	font	loadcss.js:21	112 KB	386 ms
OpenSans-Bold.ttf	GET	200	font	loadcss.js:21	114 KB	411 ms
texts_en.min.json	GET	200	xhr	polyfills.b96d4bb....b...	4.3 KB	165 ms
profile-min.jpg	GET	200	jpeg	main.70e38af....bun...	12.3 KB	632 ms
logo-min.png	GET	200	png	(index)	1.1 KB	554 ms
expertin-min.png	GET	200	png	main.70e38af....bun...	14.6 KB	492 ms
projects-min.png	GET	200	png	main.70e38af....bun...	169 KB	482 ms
data:image/png;base...	GET	200	png	(index)	(from memory cache)	0 ms

15 requests | 635 KB transferred | Finish: 1.57 s | DOMContentLoaded: 1.02 s | Load: 1.58 s

On build with the production flag and after gzip compression, application size is reduced to **635KB.** (from 1.5MB, 57,7% less)

3.c Optimizations: gzip compression (server)



Μη ασφαλής | https://nekkon.com
Elements Console Sources Network Performance Memory Application Security Audits
Screenshots Memory
Nektarios Konstantinidis
Senior Developer (Web / Front-End)
Frames 121.6 ms 437.8 ms 538.1 ms
Interactions Animation
User Timing
JS Heap Documents Nodes Listeners GPU Memory
Summary Bottom-Up Call Tree Event Log

Range: 1.27 s – 4.05 s

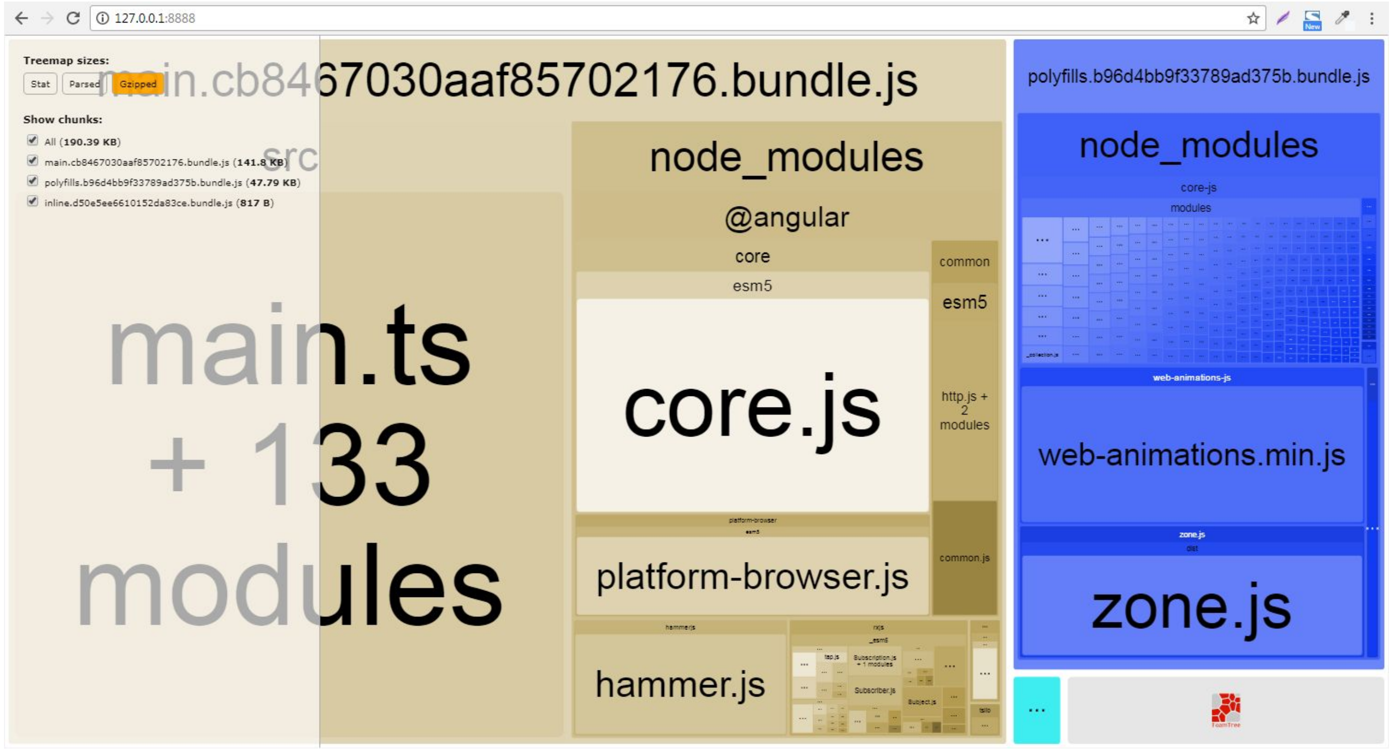
2778 ms

7.2 ms	Loading
1030.8 ms	Scripting
269.8 ms	Rendering
54.7 ms	Painting
397.3 ms	Other
1018.7 ms	Idle

With the gzip compression, the view is rendered in **1.6 seconds** (from 2.5 seconds, 36% less**).**

Although in DOM, texts are not shown cause they are waiting for custom fonts to be loaded. (FOUT) This makes the user feel that the page was loaded in 1.6 seconds instead of 1 second.

3.c Optimizations: gzip compression (server)


127.0.0.1:8888

Treemap sizes:

Stat | Parsed | [✓ Gzipped]

Show chunks:

- ✓ All (**190.39 KB**)
- ✓ main.cb8467030aaf85702176.bundle.js (**141.8 KB**)
- ✓ polyfills.b96d4bb9f33789ad375b.bundle.js (**47.79 KB**)
- ✓ inline.d50e5ee6610152da83ce.bundle.js (**817 B**)

With gzip compression, application bundle size is reduced to **190KB.** (from 785KB, 77% less)

3.d.i Universal: How to configure

Universal

A normal Angular application executes in the browser, rendering pages in the DOM in response to user actions.

Angular Universal generates static application pages on the server through a process called server-side rendering (SSR).

It can generate and serve those pages in response to requests from browsers. It can also pre-generate pages as HTML files that you serve later.

To add universal to your app follow this guide:

https://github.com/angular/angular-cli/wiki/stories-universal-rendering

To use data requested from server client side use BrowserTransferStateModule.

https://blog.savoirfairelinux.com/en-ca/2017/angular-app-with-server-side-rendering/

3.d.ii Universal: FOUT - Async custom fonts

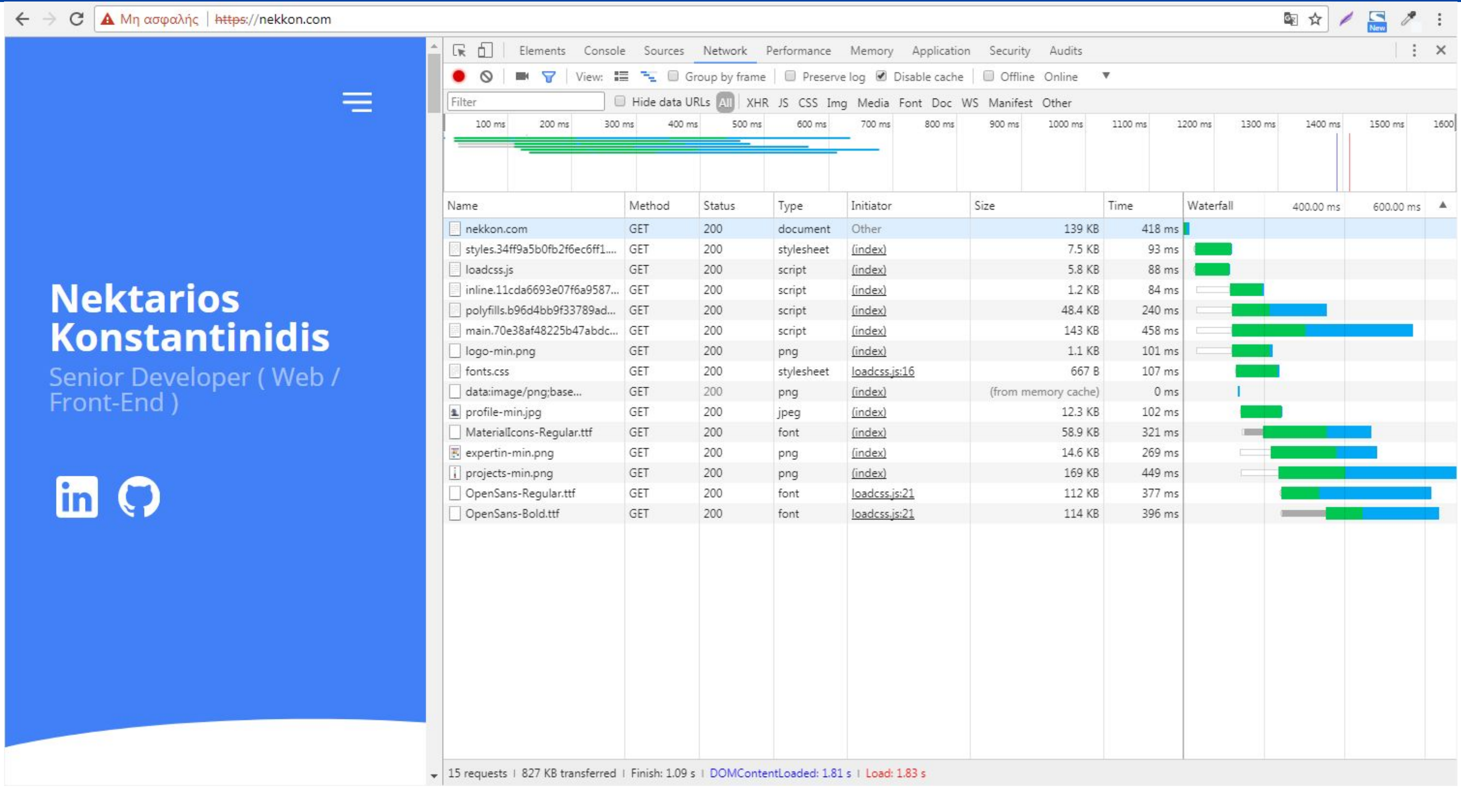
Fonts are loaded as assets into a web page—just like images or video. Depending on your browser or your connection speed, they can load quickly or lag behind the rest of the page. Different browsers handle font loading differently; for example, Safari and Chrome will refrain from displaying text set in a web font until the font has loaded, while Internet Explorer won't show anything on the page until the font loads. Meanwhile, Firefox will display the site with the fallback fonts in the font stack, and then switch to the linked fonts after they've finished loading. This results in a flash of unstyled text, or FOUT.

https://github.com/typekit/webfontloader

Web Font Loader

Web Font Loader gives you added control when using linked fonts via @font-face. It provides a common interface to loading fonts regardless of the source, then adds a standard set of events you may use to control the loading experience. The Web Font Loader is able to load fonts from Google Fonts, Typekit, Fonts.com, and Fontdeck, as well as self-hosted web fonts. It is co-developed by Google and Typekit.

3.d.iii Universal: Server Side Rendering and Web Font Loader


Μη ασφαλής | https://nekkon.com
Nektarios Konstantinidis
Senior Developer (Web / Front-End)
Elements Console Sources Network Performance Memory Application Security Audits
View: Group by frame Preserve log Disable cache Offline Online
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

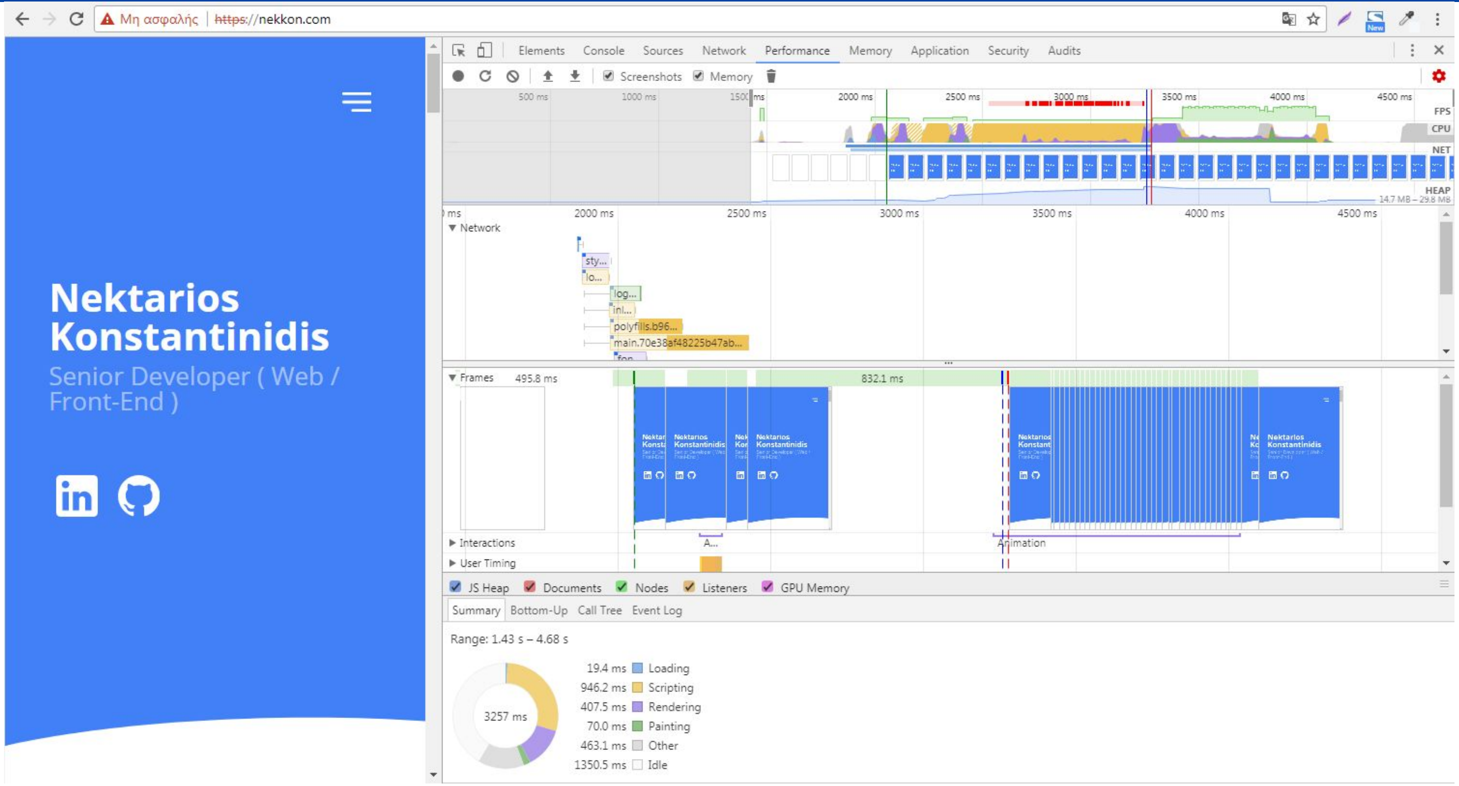
Name	Method	Status	Type	Initiator	Size	Time
nekkon.com	GET	200	document	Other	139 KB	418 ms
styles.34ff9a5b0fb2f6ec6ff1....	GET	200	stylesheet	(index)	7.5 KB	93 ms
loadcss.js	GET	200	script	(index)	5.8 KB	88 ms
inline.11cda6693e07f6a9587...	GET	200	script	(index)	1.2 KB	84 ms
polyfills.b96d4bb9f33789ad...	GET	200	script	(index)	48.4 KB	240 ms
main.70e38af48225b47abdc...	GET	200	script	(index)	143 KB	458 ms
logo-min.png	GET	200	png	(index)	1.1 KB	101 ms
fonts.css	GET	200	stylesheet	loadcss.js:16	667 B	107 ms
data:image/png;base...	GET	200	png	(index)	(from memory cache)	0 ms
profile-min.jpg	GET	200	jpeg	(index)	12.3 KB	102 ms
MaterialIcons-Regular.ttf	GET	200	font	(index)	58.9 KB	321 ms
expertin-min.png	GET	200	png	(index)	14.6 KB	269 ms
projects-min.png	GET	200	png	(index)	169 KB	449 ms
OpenSans-Regular.ttf	GET	200	font	loadcss.js:21	112 KB	377 ms
OpenSans-Bold.ttf	GET	200	font	loadcss.js:21	114 KB	396 ms

15 requests | 827 KB transferred | Finish: 1.09 s | DOMContentLoaded: 1.81 s | Load: 1.83 s

With server side rendering, application size increased to **827KB.** (from 635KB, 30% more)

This is caused by the fact that the html/css is sent rendered for the app to be loaded instantly.

3.d.iii Optimizations: Server Side Rendering and Web Font Loader



Μη ασφαλής | https://nekkon.com
Elements Console Sources Network Performance Memory Application Security Audits
Screenshots Memory
FPS
CPU
NET
HEAP
Network
Frames 495.8 ms
832.1 ms
Interactions
User Timing
Animation
JS Heap Documents Nodes Listeners GPU Memory
Nektarios Konstantinidis
Senior Developer (Web / Front-End)

Summary Bottom-Up Call Tree Event Log

Range: 1.43 s – 4.68 s

3257 ms

19.4 ms	Loading
946.2 ms	Scripting
407.5 ms	Rendering
70.0 ms	Painting
463.1 ms	Other
1350.5 ms	Idle

With Server Side Rendering and Web Font Loader, the view rendered in **570ms!** (from 1.6 seconds).

3.e Optimizations: Progressive Web Application

What is PWA?

Progressive Web App (PWA) are web applications that are regular web pages or websites, but can appear to the user like traditional applications or native mobile applications. The application type attempts to combine features offered by most modern browsers with the benefits of mobile experience.

Main Characteristics

Connectivity independent - Service workers allow work offline, or on low quality networks.
App-like - Feel like an app to the user with app-style interactions and navigation.
Fresh - Always up-to-date thanks to the service worker update process.
Safe - Served via HTTPS to prevent snooping and ensure content hasn’t been tampered with.
Discoverable - Are identifiable as “applications” thanks to W3C manifests[6] and service worker registration scope allowing search engines to find them.
Re-engageable - Make re-engagement easy through features like push notifications.
Installable - Allow users to “keep” apps they find most useful on their home screen without the hassle of an app store.
Linkable - Easily shared via a URL and do not require complex installation.

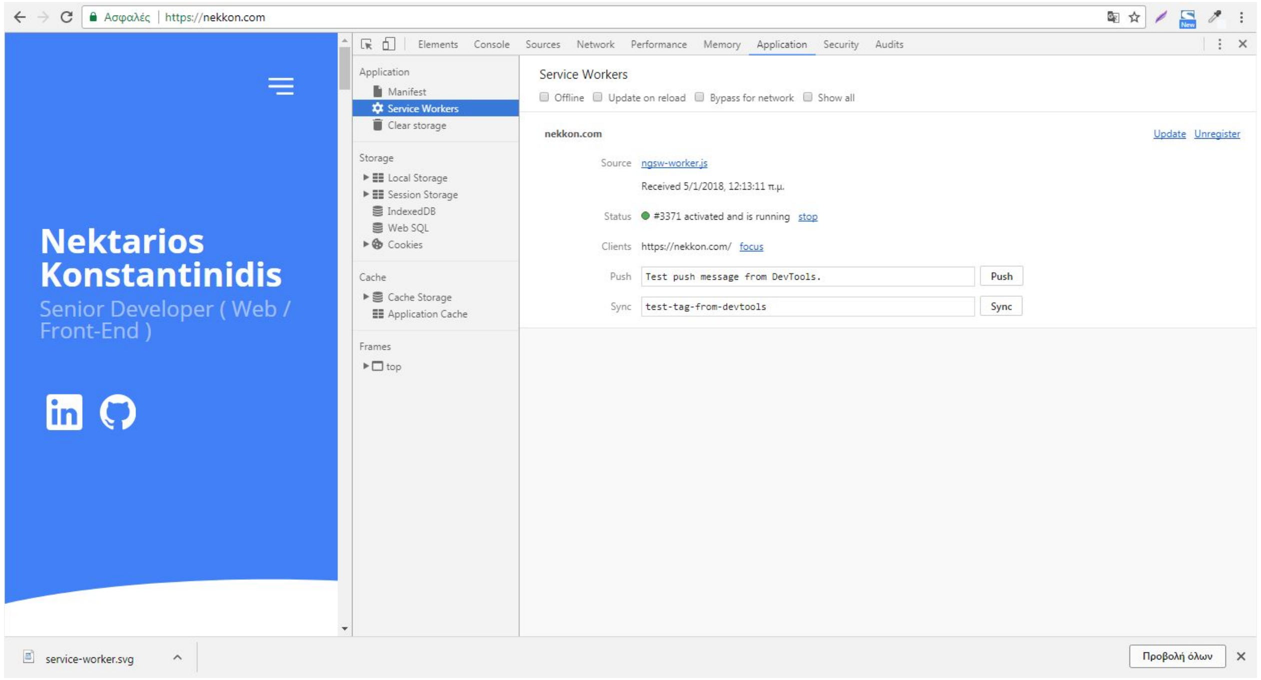
To create a new PWA with angular-cli follow this guide:

https://blog.angular-university.io/angular-service-worker/

3.e Optimizations: Progressive Web Application

Service worker

Service workers essentially act as proxy servers that sit between web applications, and the browser and the network (when available). They are intended to (amongst other things) enable the creation of effective offline experiences, intercepting network requests, and taking appropriate action based on whether the network is available, and updated assets reside on the server. They will also allow access to push notifications and background sync APIs.


Ασφαλές | https://nekkon.com

Nektarios Konstantinidis
Senior Developer (Web / Front-End)

Elements Console Sources Network Performance Memory Application Security Audits

Application
Manifest
Service Workers
Clear storage

Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies

Cache
Cache Storage
Application Cache

Frames
top

Service Workers
Offline Update on reload Bypass for network Show all

nekkon.com Update Unregister

Source ngsw-worker.js
Received 5/1/2018, 12:13:11 π.μ.

Status #3371 activated and is running stop

Clients https://nekkon.com/ focus

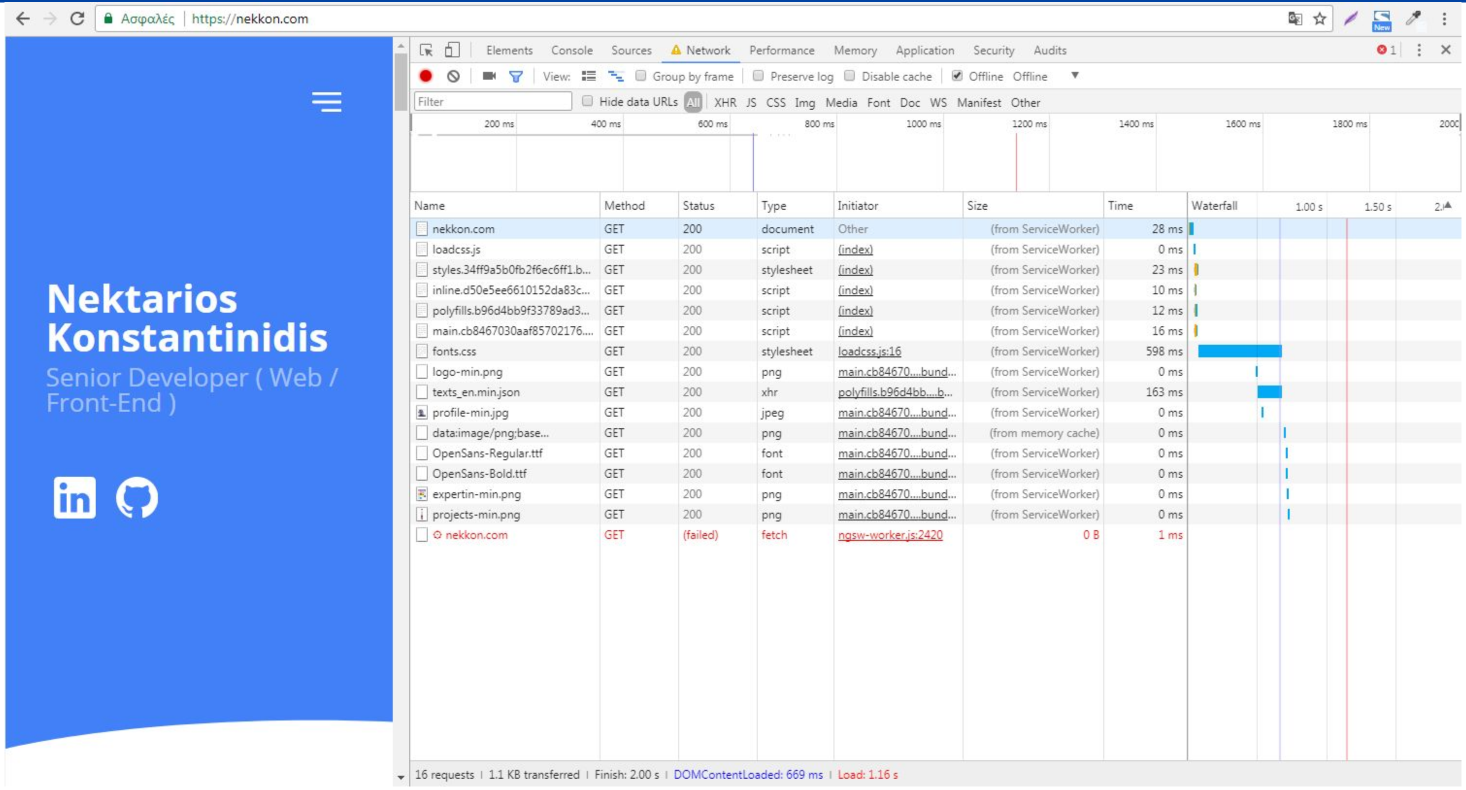
Push `Test push message from DevTools.` Push

Sync `test-tag-from-devtools` Sync

service-worker.svg

Προβολή όλων

3.e Optimizations: Progressive Web Application


Ασφαλές | https://nekkon.com

Nektarios Konstantinidis
Senior Developer (Web / Front-End)

Elements Console Sources Network Performance Memory Application Security Audits

View: Group by frame Preserve log Disable cache ✓ Offline Offline

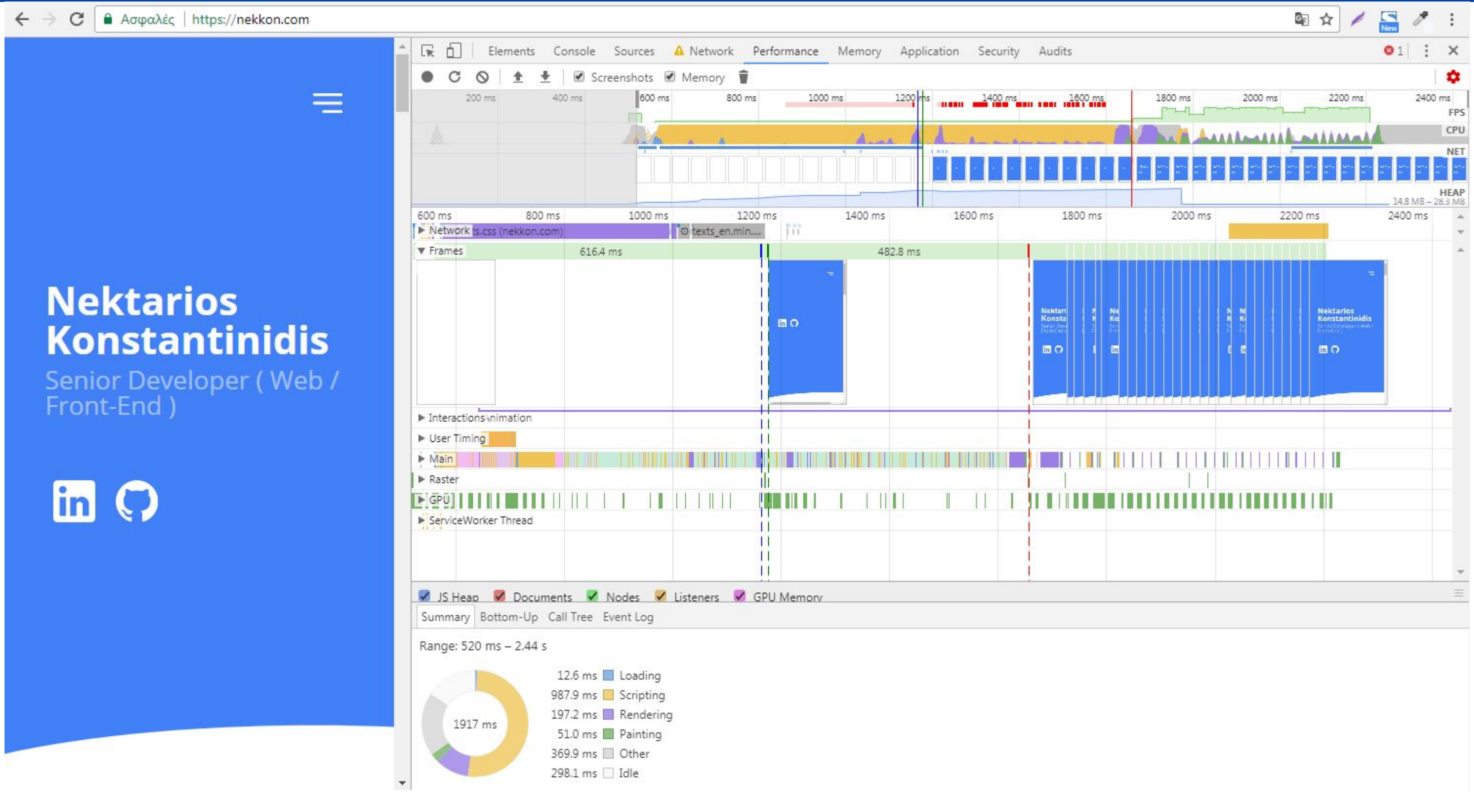
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

Name	Method	Status	Type	Initiator	Size	Time
nekkon.com	GET	200	document	Other	(from ServiceWorker)	28 ms
loadcss.js	GET	200	script	(index)	(from ServiceWorker)	0 ms
styles.34ff9a5b0fb2f6ec6ff1.b...	GET	200	stylesheet	(index)	(from ServiceWorker)	23 ms
inline.d50e5ee6610152da83c...	GET	200	script	(index)	(from ServiceWorker)	10 ms
polyfills.b96d4bb9f33789ad3...	GET	200	script	(index)	(from ServiceWorker)	12 ms
main.cb8467030aaf85702176....	GET	200	script	(index)	(from ServiceWorker)	16 ms
fonts.css	GET	200	stylesheet	loadcss.js:16	(from ServiceWorker)	598 ms
logo-min.png	GET	200	png	main.cb84670....bund...	(from ServiceWorker)	0 ms
texts_en.min.json	GET	200	xhr	polyfills.b96d4bb....b...	(from ServiceWorker)	163 ms
profile-min.jpg	GET	200	jpeg	main.cb84670....bund...	(from ServiceWorker)	0 ms
data:image/png;base...	GET	200	png	main.cb84670....bund...	(from memory cache)	0 ms
OpenSans-Regular.ttf	GET	200	font	main.cb84670....bund...	(from ServiceWorker)	0 ms
OpenSans-Bold.ttf	GET	200	font	main.cb84670....bund...	(from ServiceWorker)	0 ms
expertin-min.png	GET	200	png	main.cb84670....bund...	(from ServiceWorker)	0 ms
projects-min.png	GET	200	png	main.cb84670....bund...	(from ServiceWorker)	0 ms
nekkon.com	GET	(failed)	fetch	ngsw-worker.js:2420	0 B	1 ms

16 requests | 1.1 KB transferred | Finish: 2.00 s | DOMContentLoaded: 669 ms | Load: 1.16 s

While offline, application was loaded from cache.

3.e Optimizations: Progressive Web Application


Ασφαλές | https://nekkon.com
Nektarios Konstantinidis
Senior Developer (Web / Front-End)
Elements Console Sources Network Performance Memory Application Security Audits
Screenshots Memory
FPS
CPU
NET
HEAP
Network
Frames
616.4 ms
482.8 ms
Interactions
User Timing
Main
Raster
GPU
ServiceWorker Thread
JS Heap Documents Nodes Listeners GPU Memory
Summary Bottom-Up Call Tree Event Log

Range: 520 ms – 2.44 s

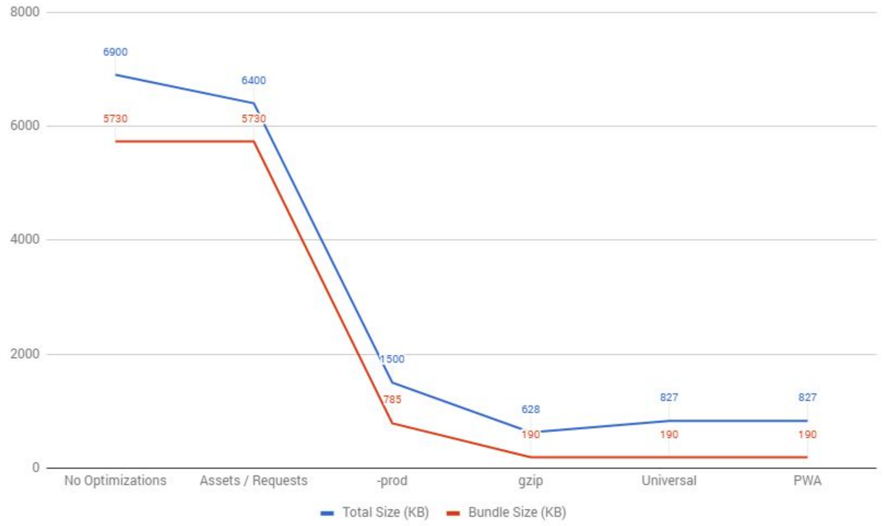
1917 ms

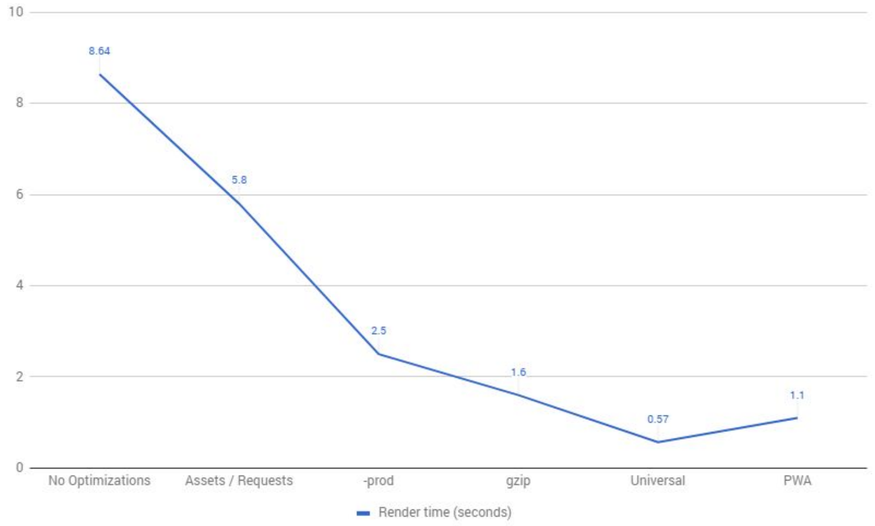
12.6 ms	🟦	Loading
987.9 ms	🟨	Scripting
197.2 ms	🟪	Rendering
51.0 ms	🟩	Painting
369.9 ms	⬜	Other
298.1 ms	⬜	Idle

While offline, the view rendered in **1100ms** (from 570ms, 48% more).

The reason is that the service worker does not cache the server-side rendered html. This could be improved with Angular App Shell.

3.f Optimizations: Results / Graphs





4. Other optimizations

Angular App Shell

To boost perceived startup performance, we want to show to the user the above the fold content as quickly as possible, and this usually means showing a menu navigation bar, the overall skeleton of the page, a loading indicator and other page-specific elements.

https://blog.angular-university.io/angular-app-shell/

ABC (Angular, Babel, Closure)

https://medium.com/@Jakeherringbone/what-angular-is-doing-with-bazel-and-closure-21f526f64a34

Useful Resources

5. Useful resources

Chrome DevTools (Browser performance)

https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/

Cheatsheet for developing ⚡lightning⚡ fast progressive Angular applications.

https://github.com/mgechev/angular-performance-checklist

Thank you!

There are polls available on meetup.com for you to vote:

1) Meetup topic
2) Meetup location
3) Meetup day/time

Feel free to contact us