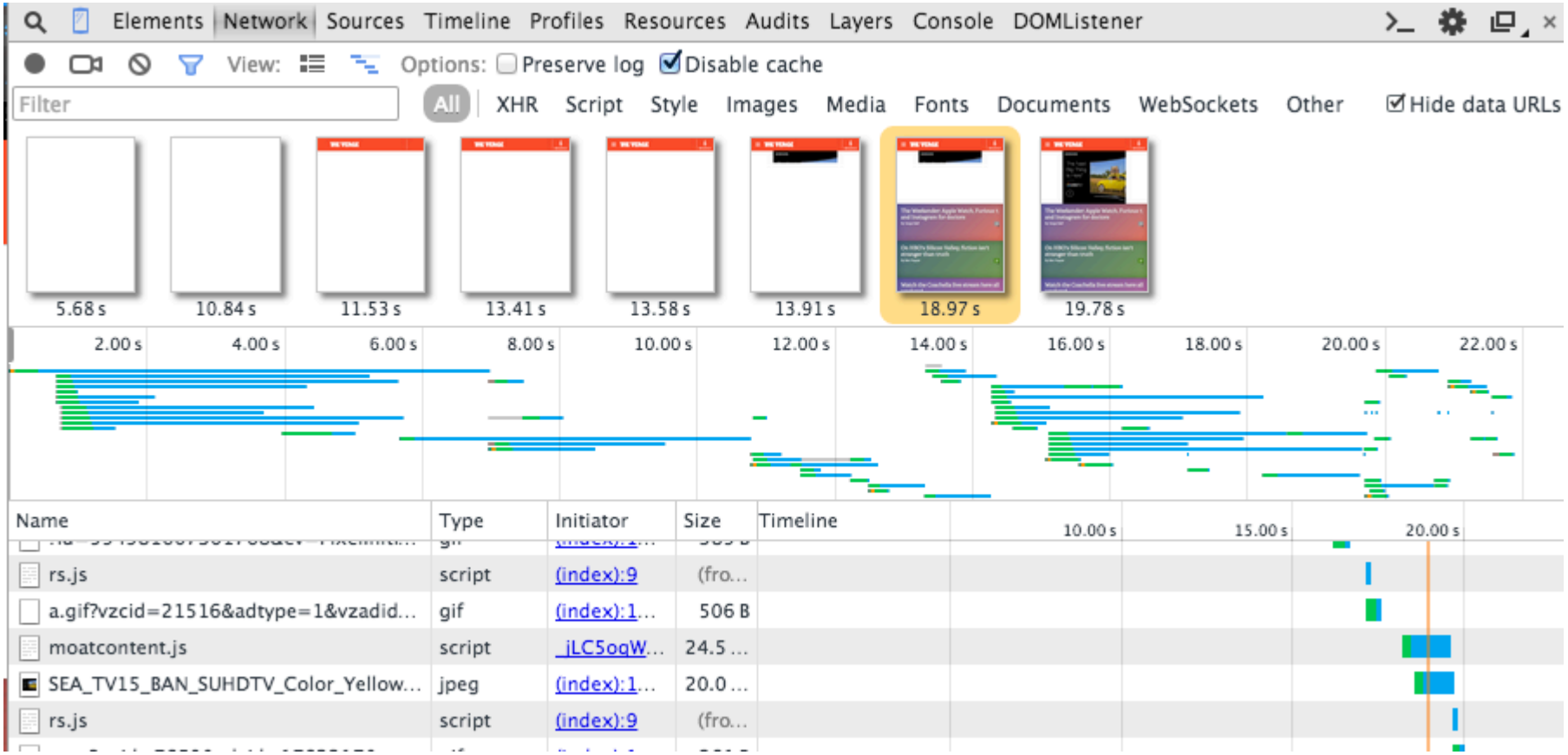
Perf Audit of theverge.com

Part 1. Paul Irish, April 2015

theverge.com - chrome canary desktop (44.0.2366.0). throttled via Good 2G

Load: Time till page appears to be interactive

19 seconds until primary page content is visible



The last <script> of the original HTML completed downloading at 10s, well before the paint at 18s.
56 requests were completed before the first paint.

Render blocking resources

The big issue here is all the render blocking resources.

The markup forces most of these to be render-blocking:

```
  <script>
//<![CDATA[

        (function() {
          var useSSL = 'https:' == document.location.protocol;
          var src = (useSSL ? 'https:' : 'http:') +
            '//www.googletagservices.com/tag/js/gpt.js';
          document.write('<scr' + 'ipt src="' + src + '"></scr' + 'ipt>');
        })();

//]]>
```

script[defer] would be a very easy addition for them, but unimplementable because IE8.

Synchronous chain of dependent scripts

Looking at network waterfall, the gpt.js series is all render-blocking and has a chain of three scripts forced synchronously via document.write. Every attempt in the world should be made to use the async snippet instead.

In addition the -head.js two JS files are render blocking but not required for the initial view.

Results after adjustments

We moved gpt.js to be async and deferred two of the <head>s JS files.
We measured the time to first meaningful paint as 100% faster.

Insights

Blink Insights

- DevTools: it's very hard to identify which of the requests are render blocking.
- Because IE8 breaks adoption of script[defer] can we ship a new name for it?
- No way to indicate [defer] on a stylesheet.
- Why doesn't <script async> work?

Next steps

- Look at input/scroll latency while page is loading.

Part 2. Kenji Baheux, July 2015

July 2015, Kenji Baheux (kenjibaheux@chromium.org)

~ Loading (first visit) ~

/ draft (peer review) /

This section looks into Loading aspects (mainly, time to first meaningful paint) when a user visits The Verge for the first time.

Setup
Google Nexus 4 on a 3G network, Chrome 45.0.2454.6
Remote debugging from Chrome dev 45 with Devtools' experimental filmstrip feature enabled.
First-ever-visit simulated by disabling HTTP cache in Devtools.

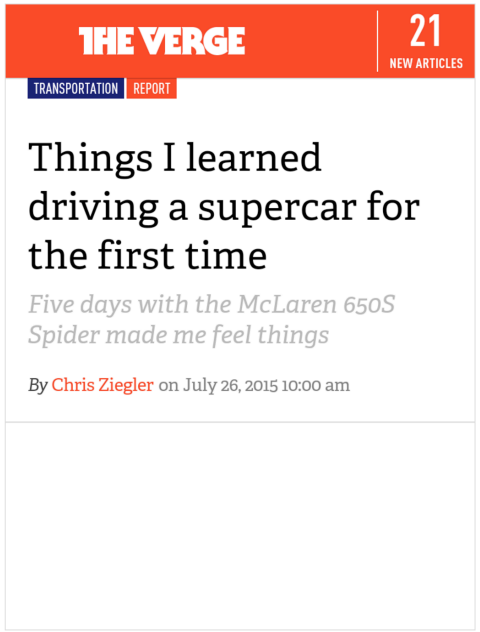
Goal
optimize for a fast *first meaningful paint*

- where "first meaningful paint" is defined as: the user can read the content above the fold

URL: http://www.theverge.com/2015/7/26/9040645/mclaren-650s-spider-first-drive
Meaningful paint:

- since this is an article page, the headline, lead and body text should be visible.
- nice to have: tag line + the time at which the article was written (in a different font)
- not required: menu iconography, image, number of new articles, tags.

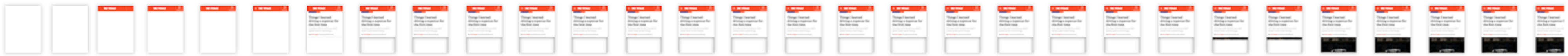
Screenshot of the meaningful paint on a particular run:



Overview (network viewpoint)

Full page load

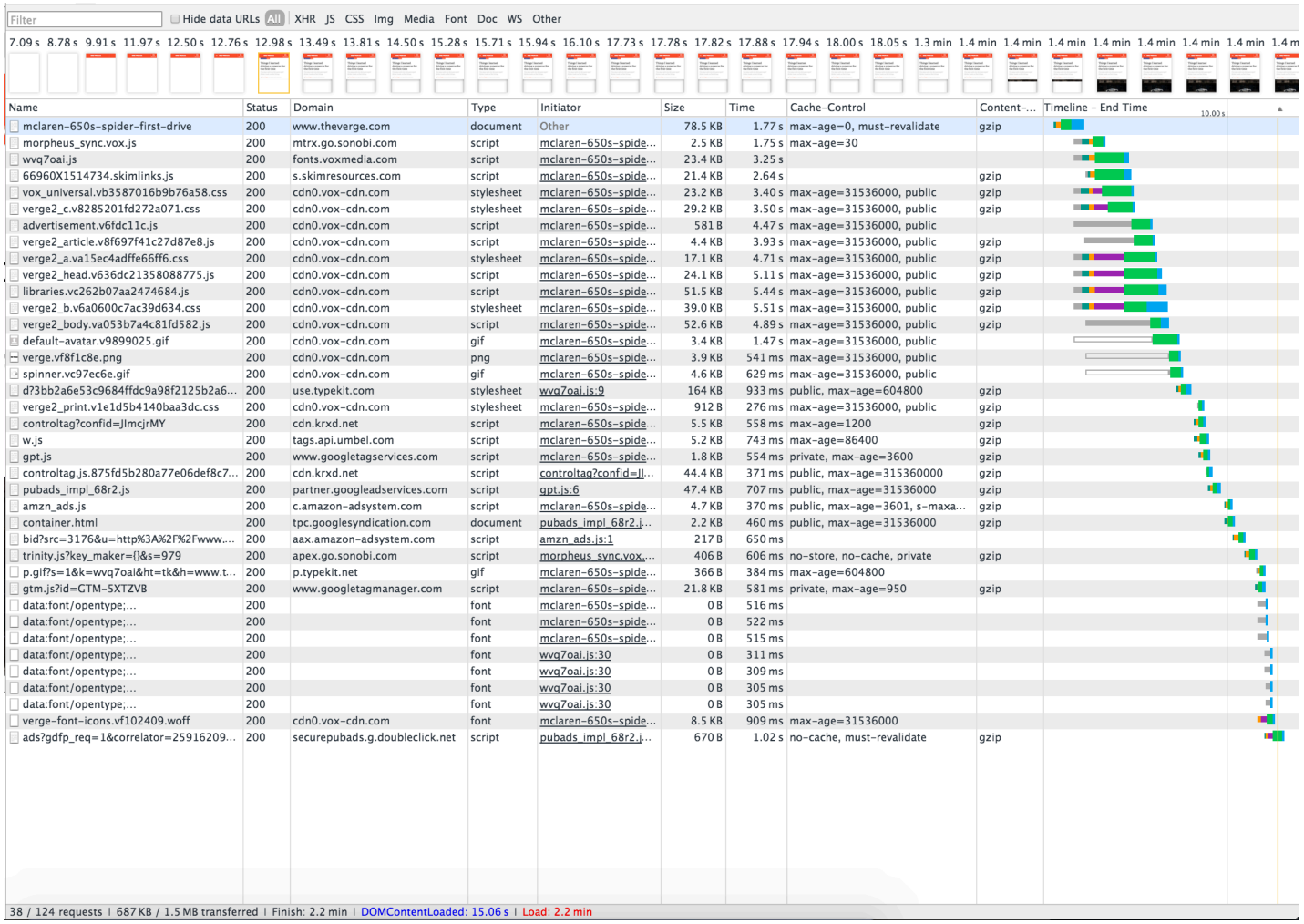
7.09 s | 8.78 s | 9.91 s | 11.97 s | 12.50 s | 12.76 s | 12.98 s | 13.49 s | 13.81 s | 14.50 s | 15.28 s | 15.71 s | 15.94 s | 16.10 s | 17.73 s | 17.78 s | 17.82 s | 17.88 s | 17.94 s | 18.00 s | 18.05 s | 1.3 min | 1.4 min | 1.4 min | 1.4 min | 1.4 min | 1.4 min | 1.4 min | 1.4 min | 1.4 m



Name	Status	Domain	Type	Initiator	Size	Time	Cache-Control	Content-...	Timeline – End Time
mclaren-650s-spider-first-drive	200	www.theverge.com	document	Other	78.5 KB	1.77 s	max-age=0, must-revalidate	gzip	
morpheus_sync.vox.js	200	mtrx.go.sonobi.com	script	mclaren-650s-spide...	2.5 KB	1.75 s	max-age=30		
wvq7oai.js	200	fonts.voxmedia.com	script	mclaren-650s-spide...	23.4 KB	3.25 s			
66960X1514734.skimlinks.js	200	s.skimresources.com	script	mclaren-650s-spide...	21.4 KB	2.64 s		gzip	
vox_universal.vb3587016b9b76a58.css	200	cdn0.vox-cdn.com	stylesheet	mclaren-650s-spide...	23.2 KB	3.40 s	max-age=31536000, public	gzip	
verge2_c.v8285201fd272a071.css	200	cdn0.vox-cdn.com	stylesheet	mclaren-650s-spide...	29.2 KB	3.50 s	max-age=31536000, public	gzip	
advertisement.v6fdc11c.js	200	cdn0.vox-cdn.com	script	mclaren-650s-spide...	581 B	4.47 s	max-age=31536000, public		
verge2_article.v8f697f41c27d87e8.js	200	cdn0.vox-cdn.com	script	mclaren-650s-spide...	4.4 KB	3.93 s	max-age=31536000, public	gzip	
verge2_a.va15ec4adffe66ff6.css	200	cdn0.vox-cdn.com	stylesheet	mclaren-650s-spide...	17.1 KB	4.71 s	max-age=31536000, public	gzip	
verge2_head.v636dc21358088775.js	200	cdn0.vox-cdn.com	script	mclaren-650s-spide...	24.1 KB	5.11 s	max-age=31536000, public	gzip	
libraries.vc262b07aa2474684.js	200	cdn0.vox-cdn.com	script	mclaren-650s-spide...	51.5 KB	5.44 s	max-age=31536000, public	gzip	
verge2_b.v6a0600c7ac39d634.css	200	cdn0.vox-cdn.com	stylesheet	mclaren-650s-spide...	39.0 KB	5.51 s	max-age=31536000, public	gzip	
verge2_body.va053b7a4c81fd582.js	200	cdn0.vox-cdn.com	script	mclaren-650s-spide...	52.6 KB	4.89 s	max-age=31536000, public	gzip	
default-avatar.v9899025.gif	200	cdn0.vox-cdn.com	gif	mclaren-650s-spide...	3.4 KB	1.47 s	max-age=31536000, public		
verge.vf8f1c8e.png	200	cdn0.vox-cdn.com	png	mclaren-650s-spide...	3.9 KB	541 ms	max-age=31536000, public		
spinner.vc97ec6e.gif	200	cdn0.vox-cdn.com	gif	mclaren-650s-spide...	4.6 KB	629 ms	max-age=31536000, public		
d?3bb2a6e53c9684ffdc9a98f2125b2a6...	200	use.typekit.com	stylesheet	wvq7oai.js:9	164 KB	933 ms	public, max-age=604800	gzip	
verge2_print.v1e1d5b4140baa3dc.css	200	cdn0.vox-cdn.com	stylesheet	mclaren-650s-spide...	912 B	276 ms	max-age=31536000, public	gzip	
controltag?confid=JlmcjrMY	200	cdn.krxd.net	script	mclaren-650s-spide...	5.5 KB	558 ms	max-age=1200	gzip	
w.js	200	tags.api.umbel.com	script	mclaren-650s-spide...	5.2 KB	743 ms	max-age=86400	gzip	
gpt.js	200	www.googletagservices.com	script	mclaren-650s-spide...	1.8 KB	554 ms	private, max-age=3600	gzip	
controltag.js.875fd5b280a77e06def8c7...	200	cdn.krxd.net	script	controltag?confid=Jl...	44.4 KB	371 ms	public, max-age=315360000	gzip	
pubads_impl_68r2.js	200	partner.googleadservices.com	script	gpt.js:6	47.4 KB	707 ms	public, max-age=31536000	gzip	
amzn_ads.js	200	c.amazon-adsystem.com	script	mclaren-650s-spide...	4.7 KB	370 ms	public, max-age=3601, s-maxa...	gzip	
container.html	200	tpc.googlesyndication.com	document	pubads_impl_68r2.j...	2.2 KB	460 ms	public, max-age=31536000	gzip	
bid?src=3176&u=http%3A%2F%2Fwww....	200	aax.amazon-adsystem.com	script	amzn_ads.js:1	217 B	650 ms			
trinity.js?key_maker={}&s=979	200	apex.go.sonobi.com	script	morpheus_sync.vox....	406 B	606 ms	no-store, no-cache, private	gzip	
p.gif?s=1&k=wvq7oai&ht=tk&h=www.t...	200	p.typekit.net	gif	mclaren-650s-spide...	366 B	384 ms	max-age=604800		
gtm.js?id=GTM-5XTZVB	200	www.googletagmanager.com	script	mclaren-650s-spide...	21.8 KB	581 ms	private, max-age=950	gzip	
data:font/opentype;...	200		font	mclaren-650s-spide...	0 B	516 ms			
data:font/opentype;...	200		font	mclaren-650s-spide...	0 B	522 ms			
data:font/opentype;...	200		font	mclaren-650s-spide...	0 B	515 ms			
data:font/opentype;...	200		font	wvq7oai.js:30	0 B	311 ms			
data:font/opentype;...	200		font	wvq7oai.js:30	0 B	309 ms			
data:font/opentype;...	200		font	wvq7oai.js:30	0 B	305 ms			
data:font/opentype;...	200		font	wvq7oai.js:30	0 B	305 ms			
verge-font-icons.vf102409.woff	200	cdn0.vox-cdn.com	font	mclaren-650s-spide...	8.5 KB	909 ms	max-age=31536000		
fbds.js	307	connect.facebook.net		Other	0 B	28 ms			
ads?gdfp_req=1&correlator=25916209...	200	securepubads.g.doubleclick.net	script	pubads_impl_68r2.j...	670 B	1.02 s	no-cache, must-revalidate	gzip	
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
analytics.js	200	www.google-analytics.com	script	gtm.js?id=GTM-5XT...	11.2 KB	404 ms	public, max-age=7200	gzip	
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
data:image/gif;base...	(data)		gif	mclaren-650s-spide...	(from cac...	0 ms			
quant.js	200	edge.quantserve.com	script	Other	3.3 KB	495 ms	private, max-age=86400	gzip	
b?c1=2&c2=7976662&ns__t=1438055...	204	b.scorecardresearch.com	text/plain	mclaren-650s-spide...	248 B	729 ms	private, no-cache, no-cache=Set...		
_jLC5oqWXzw1IWgPvd2VE0b.js	200	storage.cloud.kargo.com	script	gtm.js?id=GTM-5XT...	41.8 KB	801 ms		gzip	
linkid.js	200	www.google-analytics.com	script	analytics.js:3	1.2 KB	1.12 s	public, max-age=3600	gzip	
verge-font-icons.woff	200	www.theverge.com	font	mclaren-650s-spide...	8.3 KB	1.39 s			

Requests until first meaningful paint

tl;dr: first meaningful first paint @12.98s



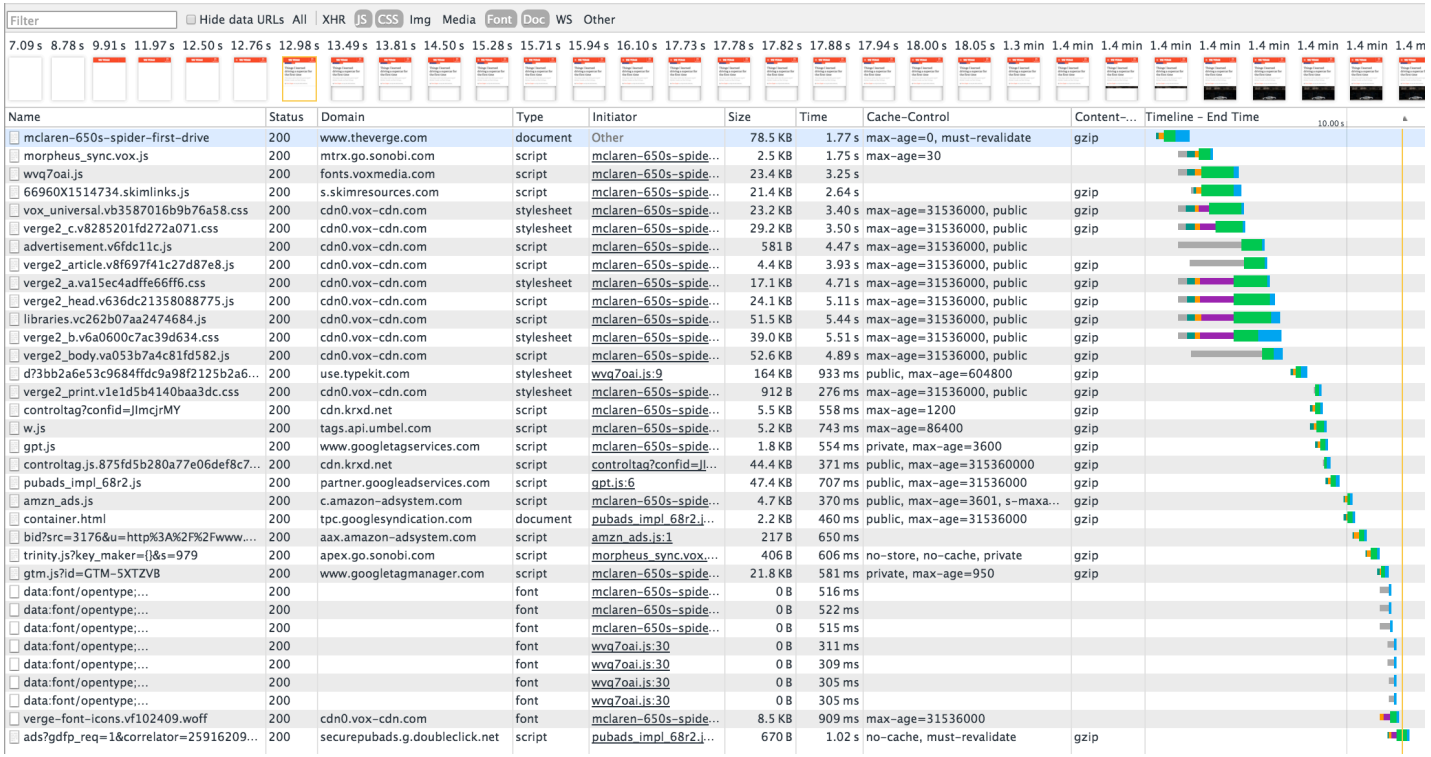
This shows all the network requests until the first meaningful paint (@12.98s), ordered by End time.

The orange line on the right hand side indicates the time at which the selected frame (here, the first meaningful paint) was rendered.

Critical path

chain of events

Assets that are *render-blocking* affect the time it takes to be able to start painting something on the screen, and typically fonts then affect the time it takes to paint something **meaningful** (text content) on the screen.
Here is a filtered view of the first network requests.



It shows the **potentially** render-blocking assets (JS, CSS, Font and Document) that started before the first meaningful paint. Figuring out which assets are actually render-blocking currently requires to backtrack from the Initiator and look for where the script/... was inserted and the absence of async/defer. Here is what I found:

Render blocking assets:

- main document
- libraries.vc262b07aa2474684.js
- verge2_head.v636dc21358088775.js
- wvq7oai.js (selfhosted TypeKit JS)
  - TypeKit's dynamically added d?3bb2... stylesheet
  -
- advertisement.v6fdc11c.js
- vox_universal.vb3587016b9b76a58.css
- verge2_a.va15ec4adffe66ff6.css
- verge2_b.v6a0600c7ac39d634.css
- sonobi.com's morpheus_sync.vox.js
  - Sonobi.com's dynamically insterted trinity.js?...
- verge2_c.v8285201fd272a071.css
- googletagservices.com's gpt.js
  - GPT's dynamically added pubads_impl_68r2.js
    - Pubads's dynamically inserted securepubads.g.doubleclick.net/gampad/ads?
- c.amazon-adsystem.com's amzn_ads.js
  - Amazon's bid?=

in body

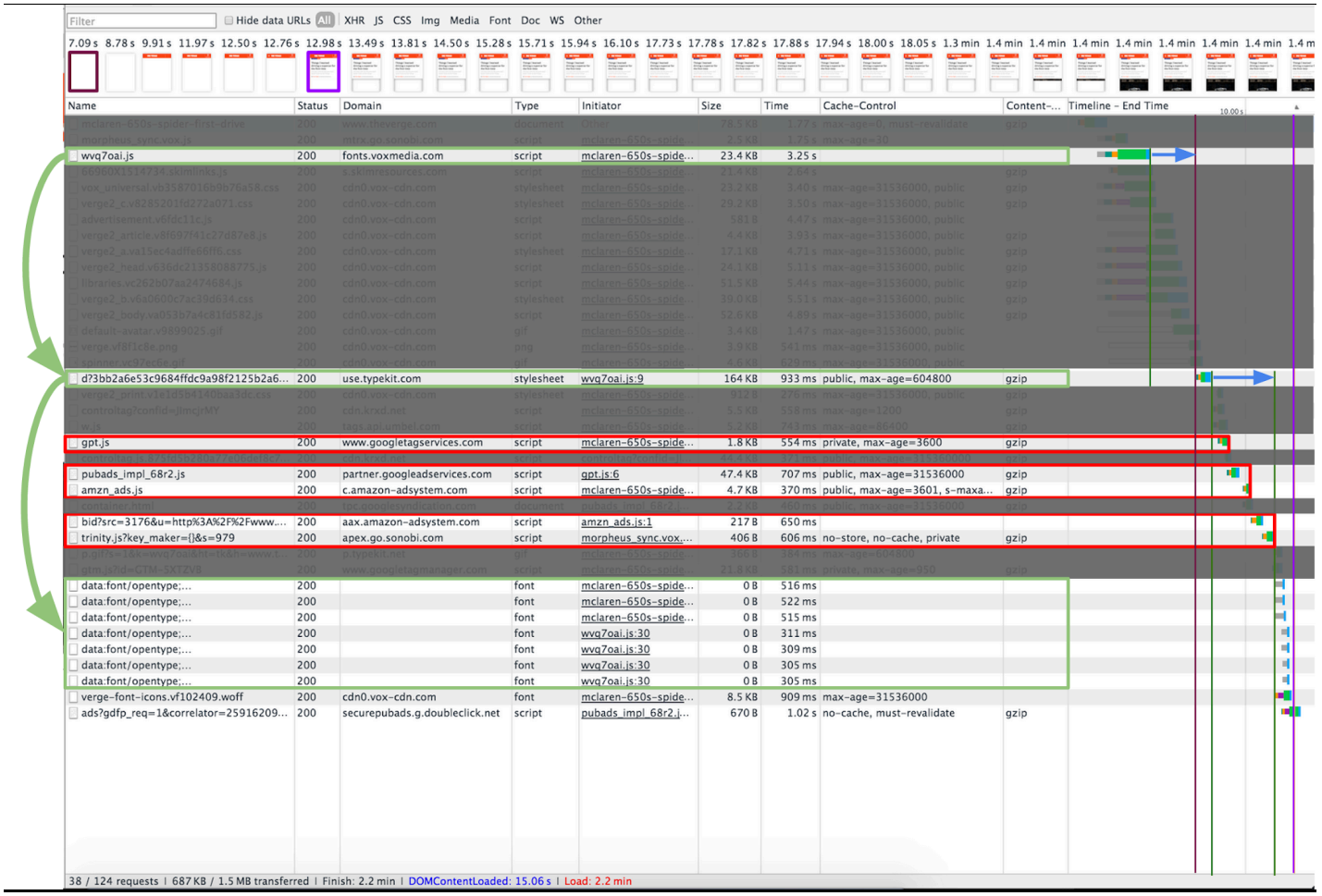
- verge2_article.v8f697f41c27d87e8.js

- verge2_body.va053b7a4c81fd582.js
- 66960X1514734.skimlinks.js

Meaningful paint blocking assets:
- **fonts served as dataURI from TypeKit**
- **selfhosted font for iconongraphy (verge-font-icons.vf102409.woff)**

Optimizing the time to *meaningful first paint* from *first paint*

I believe that the delay between first paint (@7.09s) and first meaningful paint (@12.98s) can be explained by the following diagram (bear with me ;)



Claret highlighted thumbnail:first paint (white page, pretty much meaningless but that's our start).

Claret line: timestamp for first paint. Observe that it happens right **before the request for TypeKit's stylesheet**.

Purple highlighted thumbnail: first meaningful paint.

Purple line: timestamp for first meaningful paint. Observe that it happens quite a while after we got **TypeKit's stylesheet**.

Green entries and lines: web font related assets and timestamps
1. TypeKit's webfont loader JS
2. TypeKit's stylesheet containing the font-face definitions and embedded fonts
3. TypeKit's embedded fonts
    - Surprisingly, these still took 300-500ms to deal with...

2 giant green arrows on the left side indicating that:
1. TypeKit's web font loader kicks the stylesheet request
2. TypeKit's web fonts are defined (and embedded) into the d?3bb… stylesheet

Red entries: additional render blocking assets dynamically inserted in <head>.

Note the first Blue arrow on the entry for wvq7oai.js. It's highlighting some significant delay between the time at which we got TypeKit's web font loader JS and the time at which TypeKit's request for fetching the stylesheet starts. More on that here.

Then note the second Blue arrow on the entry for TypeKit's stylesheet highlighting a significant delay before we start to grab the embedded fonts for displaying the text. From this view, **it seems** that Chrome is getting swamped with a succession of extra render blocking requests triggered by third party scripts which may postpone our ability to decide that we really need these fonts*. More on that here.

*: by design, web fonts are lazy loaded.

tl;dr: 4-5 seconds delay from first paint to first meaningful paint

- Unclear from Network tab what's going on, need to take at look at Timeline (see next section)

For reference: screenshots showing how the extra render blocking assets are inserted

pubads_impl_xx.js inserted by gpt:

- non async
- in head

```
        googletag.evalScripts();
    else {
        var U = (P()._vars_["#6#"] ? "https:" : "http:") + "//partner.googleadservices.com/gpt/pubads_impl_" + S() + ".js"
          , V = T.currentScript;
        if (!("complete" == T.readyState || "loaded" == T.readyState || V && V.async)) {
            var W = "gpt-impl-" + Math.random();
            try {
                T.write('<script id="' + W + '" src="' + U + '">\x3c/script>')
            } catch (X) {}
            T.getElementById(W) && (P()._loadStarted_ =
            !0)
        }
        if (!P()._loadStarted_) {
            var Y = T.createElement("script");
            Y.src = U;
            Y.async = !0;
            (T.head || T.body || T.documentElement).appendChild(Y);
            P()._loadStarted_ = !0
        }
    }
    ;
}
)()
```

amzn_ads.js:

- non async
- in head

```
</script><script src="http://mtrx.go.sonobi.com/morpheus_sync.vox.js"></script><script>
//<![CDATA[

        (function() {
          var useSSL = 'https:' == document.location.protocol;
          var src = (useSSL ? 'https:' : 'http:') +
            '//c.amazon-adsystem.com/aax2/amzn_ads.js';
          document.write('<scr' + 'ipt src="' + src + '"></scr' + 'ipt>');
        })();

//]]>
</script><script>
//<![CDATA[

        try {
          amznads.getAds('3176');
        } catch(e) { console.log(e); /*ignore*/ }

//]]>
</script><script>
//<![CDATA[
```

bid?src inserted by amzn_ads.js (call to getAds above):

- non async
- in head

```
    t.getAds = function(a, n, o, r) {
        if (r)
            return void t.doGetAdsAsync(a, n, o);
        var i = t.getScriptSource(a, n, o);
        t.log("amznads.getAds: Call to: " + i),
        aax_write(e, "<script type='text/javascript' src='" + i + "'></script>")
    }
    ,
```

```
var tk = function(a, b) {
    if (!a.w) {
        a.o = b.j;
        var c = a.T(b)
          , c = Bh(Hj(c, !1))
          , d = ++a.A;
        googletag._tmanager_.tickRepeated("start_ad_fetch_period", d, b.j[0].l);
        a.j ? Kj(a, b.j, "googletag.impl.pubads.setAdContentsBySlotForSync") : og(b.j[0], c);
        Qj();
        document.write('<script type="text/javascript" src="' + c + '">\x3c/script>')
    }
}
```

trinity.js inserted by sonobi's morpheus_... :

- non async
- in head

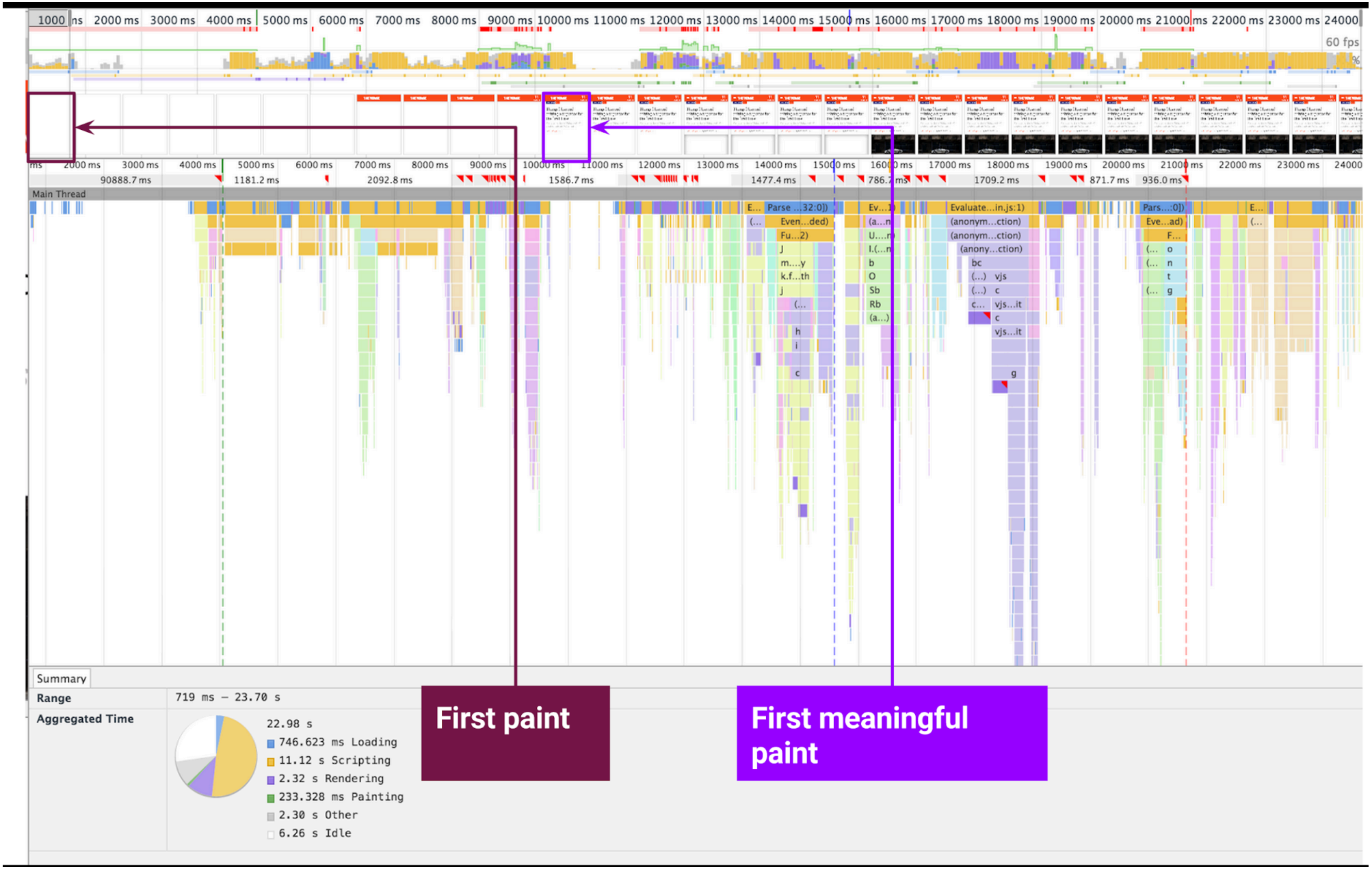
```
    launch: function (src, callback) {
        if (src == null) return;
        if (callback == null) callback = function () {};
        try {
            document.write("<scr" + "ipt type='text/javascript' src='"+src+"'></scr" + "ipt>");
            callback();
            callback = function () {};
        } catch (e) {}
    },
```

Overview (timeline viewpoint)

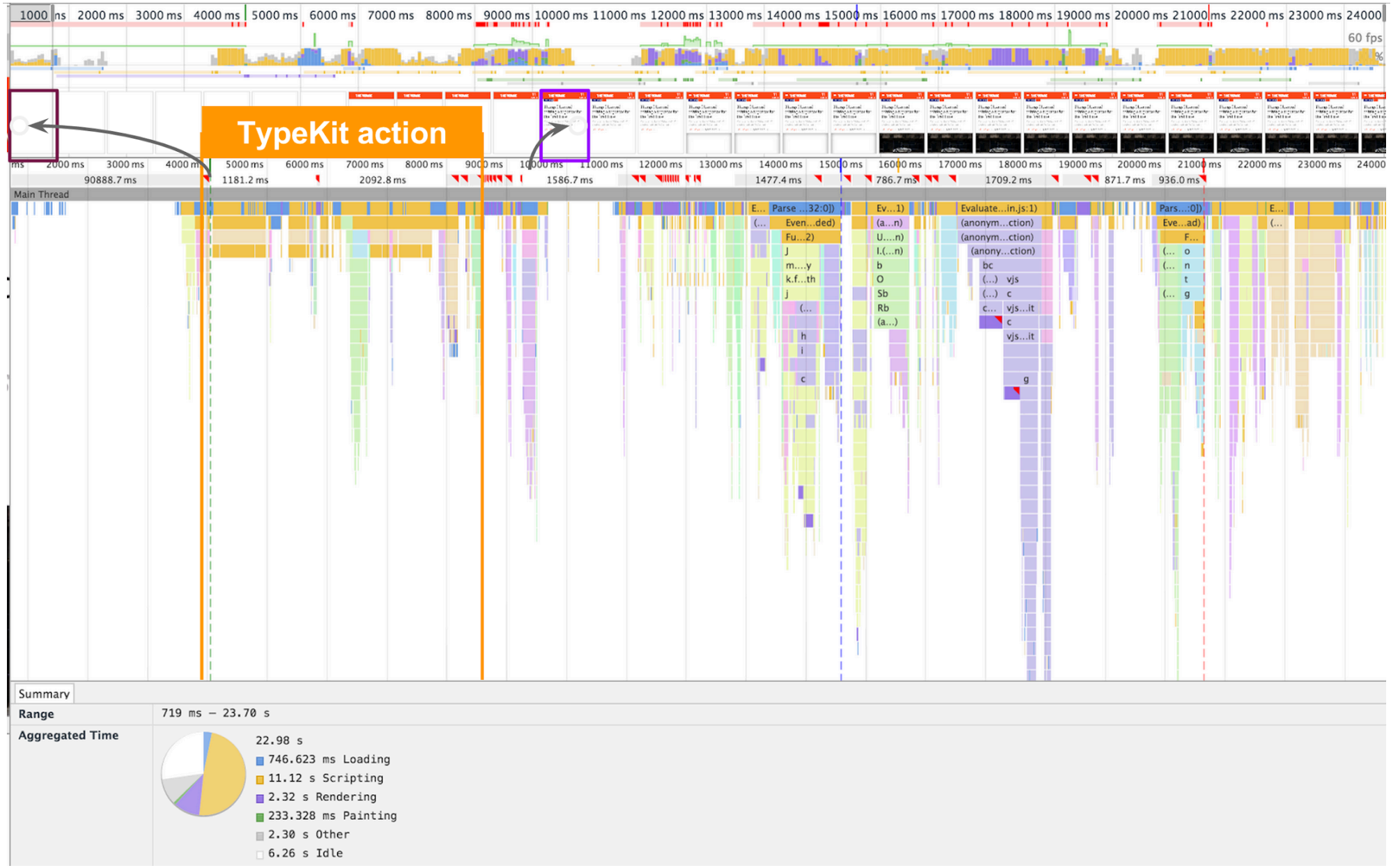
Note: this is from a different run but from the same setup. See the original slides and view in fullscreen if they are too tiny in this document.

Full page load

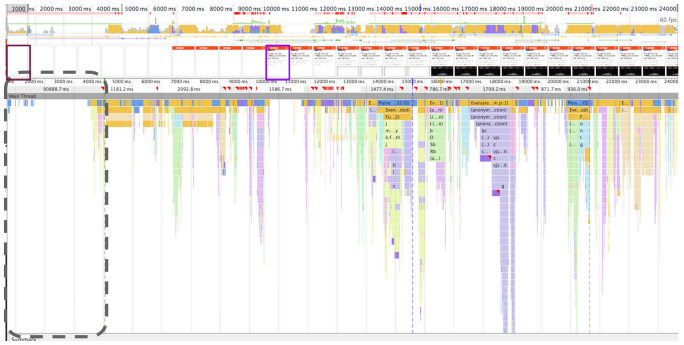
Note: start time set to the unload event @720ms



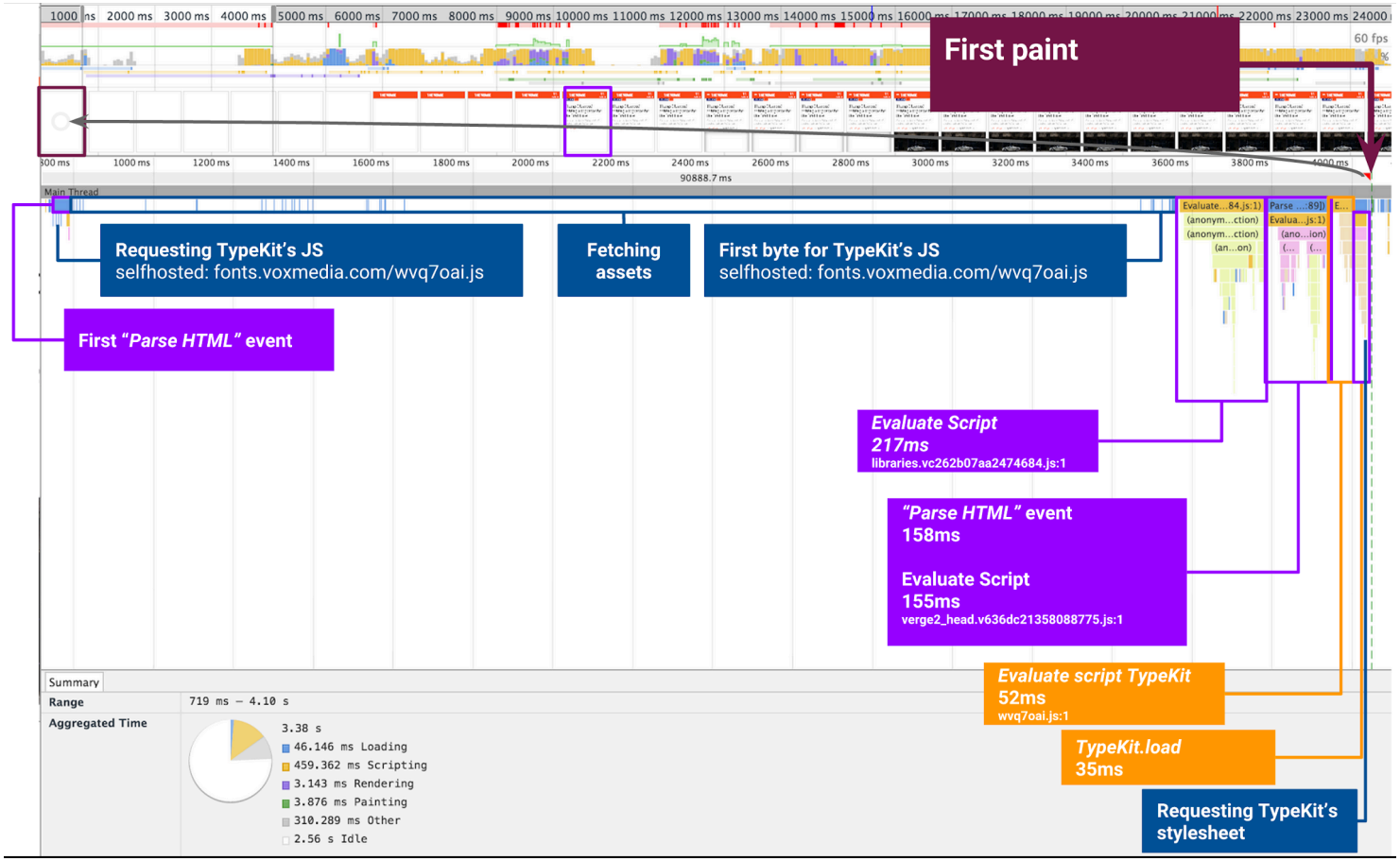
Between first paint and first meaningful paint, a large portion of the activity is TypeKit related:



Events until first paint (green dotted line ~@+3.4s)

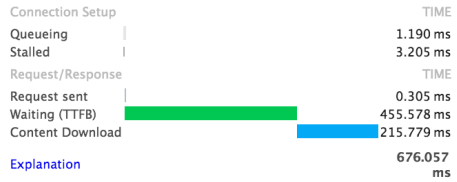


⇓



Requesting TypeKit's JS happens rather fast. Of note, The Verge is selfhosting an obsolete version of TypeKit's JS (1.7; 2014-01-09).

On this particular the TTFB was particularly excruciating. Here is a more typical example:



- ~460ms setup + TTFB
- ~215ms downloading

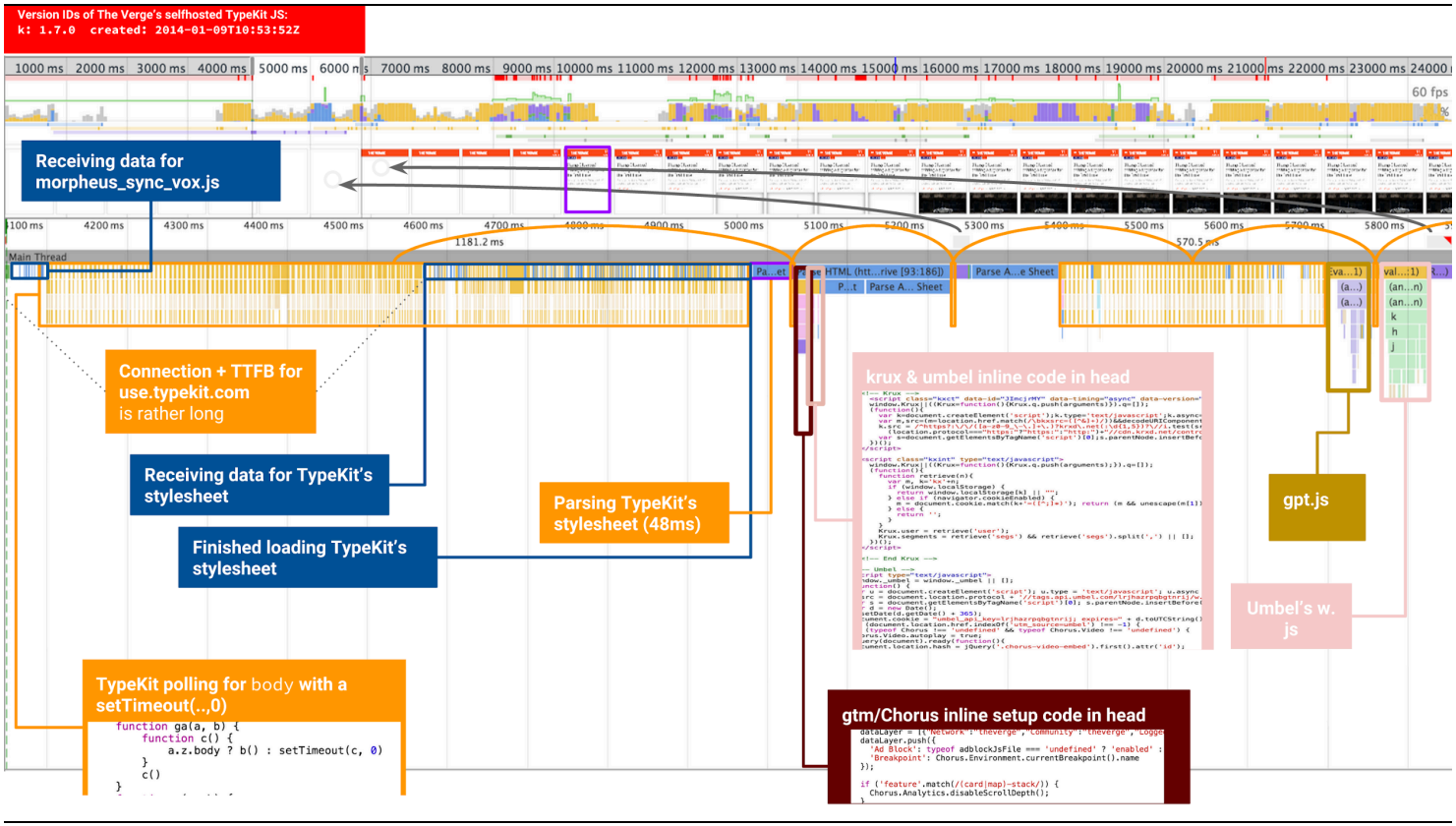
After TypeKit's JS is downloaded, we spend:

- 217ms evaluating *libraries… .js*
- 155ms evaluating *verge2.. .js*
- 52ms evaluating TypeKit's JS
- 35 ms running .load() which triggers the request for the stylesheet containing the font-face definitions.

This explains the long delay (~460ms) observed in the network view.

Of note, the stylesheet is hosted by TypeKit at use.typekit.com, the same domain they use for hosting their JS asset.

From first paint to first meaningful paint

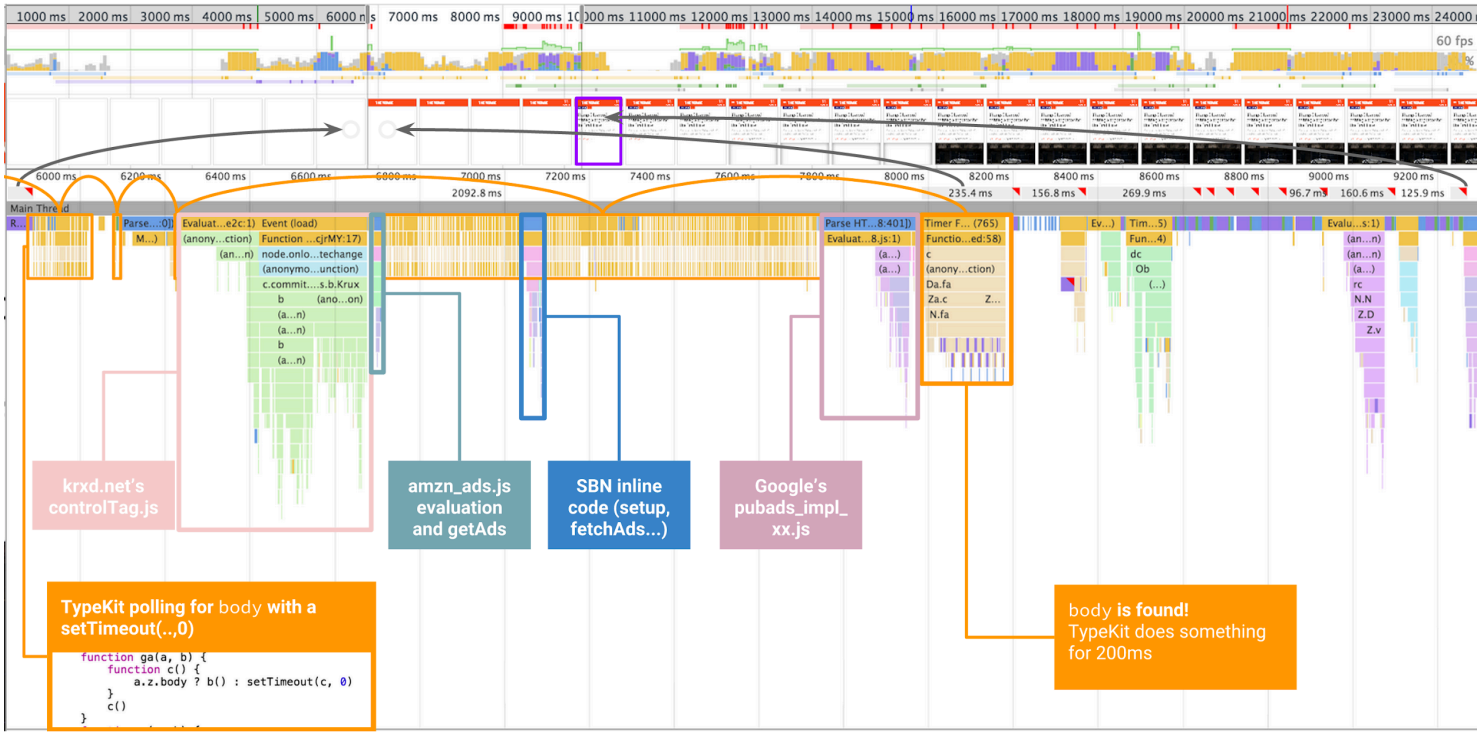


More assets are fetched and more scripts are run as the HTML of the main document is being parsed.

TypeKit-wise
On this particular run, it took quite a while (1.3s for) to get an *Initial connection* to use.typekit.com. Here is a more typical example:

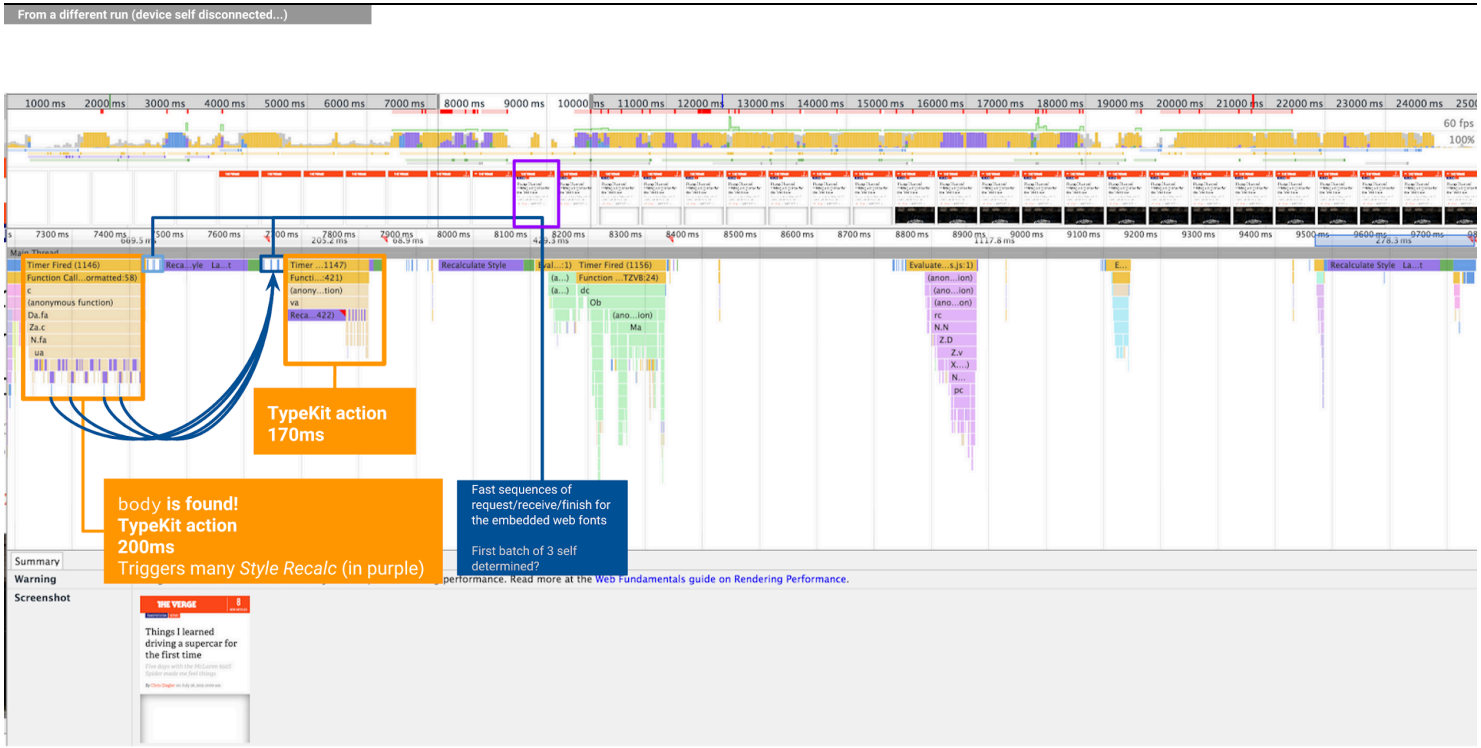
- ~485ms setup + TTFB
- ~428ms downloading

Also, TypeKit V1.7 is seen aggressively polling for the presence of .body with a setTimeout(...,0).



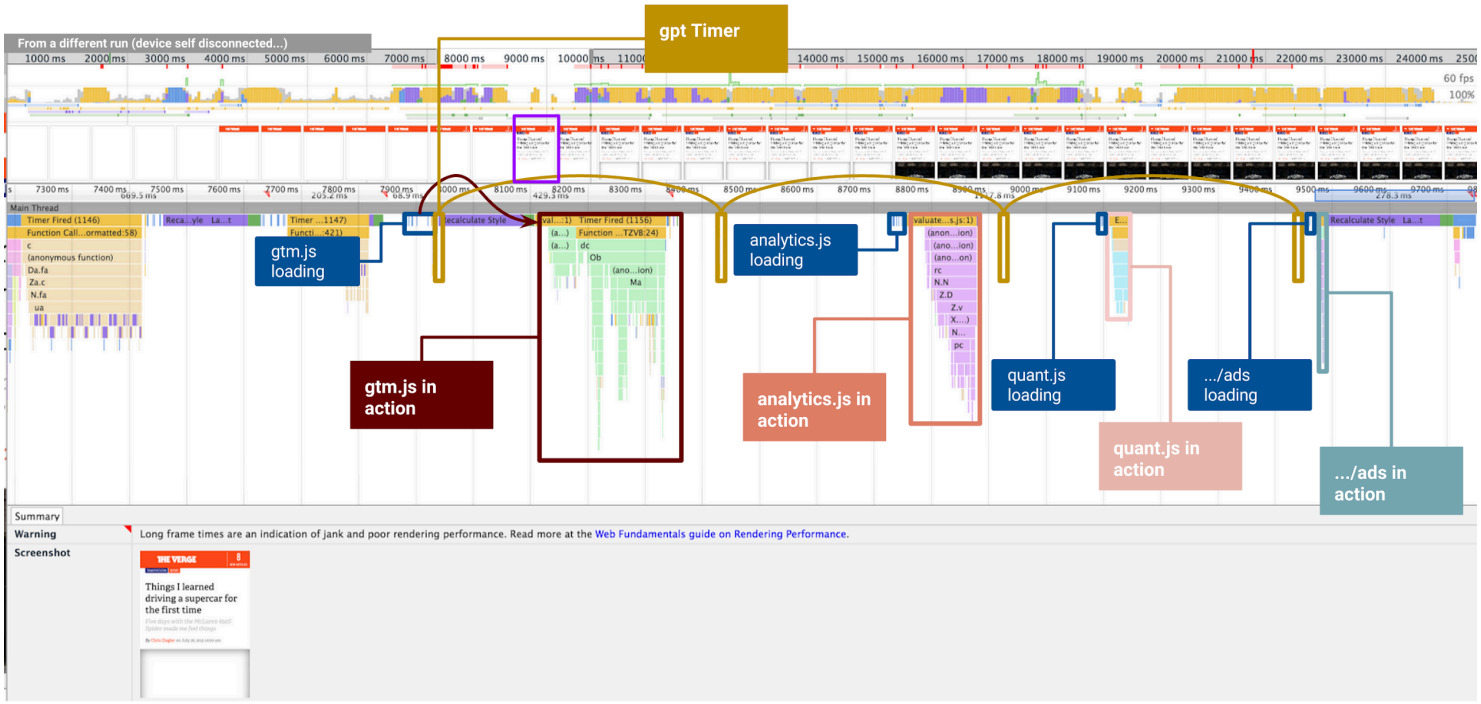
TypeKit V1.7's body polling keeps on until the 8000ms mark where body is found (Note: the Parse HT...268:401 blue chip just before that. Body is at starting from line 335).

In the meantime, more assets are being taken care of: krxd, amzn_ads, SBN inline code, pubads_impl_xx. Krxd and pubads_impl seems relatively expensive by comparison and would be worth a deep dive.



As soon as TypeKit finds out about the presence of a body, it does something for 200ms which triggers 4 web fonts requests (the fonts are embedded as dataURI in the stylesheet).

There are also 3 or so additional requests that seem to happen naturally just after TypeKit ran.



So since we have the fonts, one would expect to see a first meaningful paint happen reasonably fast, right?

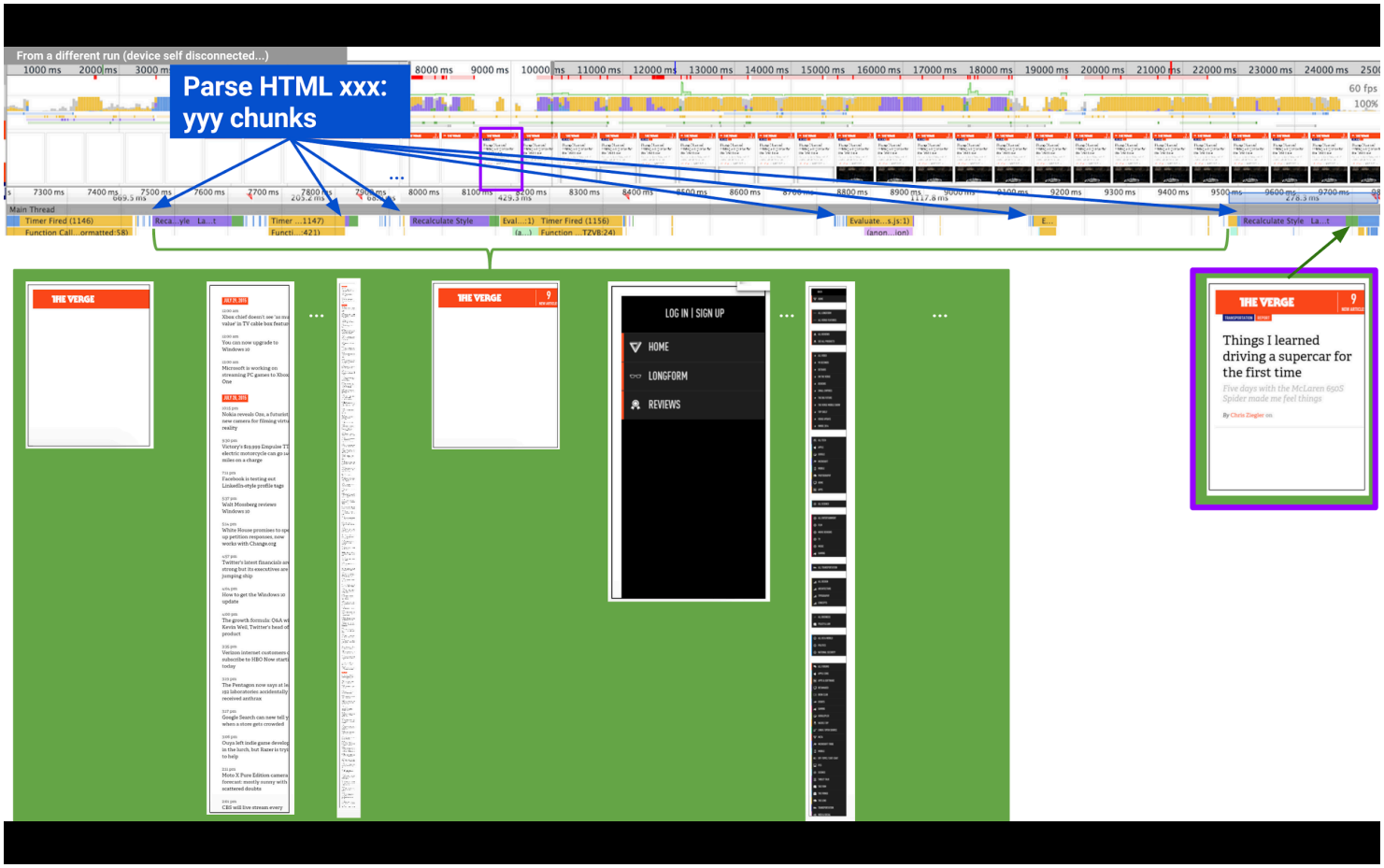
Unfortunately, it's still 2+ seconds away on this run (highlighted on the right hand side of the screenshot).

While the network tab suggested that render blocking assets added to the head postponed our ability to make a decision about web fonts, the timeline view (at least on this particular run), says otherwise.

What actually was happening is that **it's TypeKit which was driving the delay by waiting on body**, in order to do something that leads to triggering the fonts requests. *However, for reasons we will discover in the next screenshot, this doesn't even matter.*

Here is why while we have the fonts ready, it still takes a while until something meaningful is on the screen.

First of all, we do in fact perform some paint work (green cells ■) in between dealing with more async scripts, timers and so on...



Here is a view showing what we are painting before the first meaningful frame.

As we parse more chunks of HTML from the main document, we paint what we can.

Here is how the document is structured:
<header>
<nav right>... //new articles drawer
<nav left>... // hot dog menu drawer
</header>
...
<article>...

We are painting:
1. the header

2. and then the (hidden) new article drawer* which keeps on growing as we read more HTML chunks, so we repaint it several times.

*: the drawer triggers when touching this area of the top bar:

3. third step, we start doing the same for the hotdog menu drawer (also hidden and growing as we read more chunks).

4. Finally, when we reach <article>, we start painting something meaningful.

The additional on-going action (async scripts to evaluate, timer firing...) probably doesn't help either.

tl;dr: Takeaways

the 4-5 seconds delay from first paint to first meaningful paint is due to:

1. Time to obtain chunk from main document with render blocking assets
2. Time to obtain TypeKit's JS and other render blocking assets
3. Time to run/process TypeKit's JS and other render blocking assets
4. Time to obtain subsequent render blocking assets (e.g. TypeKit's stylesheet, script dynamically added to head without ASYNC...)
5. Time to process subsequent render blocking assets (e.g. parsing TypeKit's stylesheet, evaluating and running scripts)

6. Time it takes to get the document chunks for the meaningful section of the website (i.e. <article>)

the delay from first paint to first meaningful paint is negatively influenced by:

In order of perceived importance:

1. Timers that run off and monopolize the main thread
2. TypeKit V1.7's constant polling for .body (quite likely)
3. the time it takes to layout and paint elements that comes before the meaningful section (e.g. header, nav left and right).
4. ASYNC script
5. additional network requests (e.g. usage ping)

tl;dr: Recommendations (iteration 1)

1. ~~Update to~~ **Just use TypeKit's provided latest JS :)**
    a. Selfhosting leads to obsolete / sub-optimal versions being deployed (deploy => forget)
    b. You end up paying the cost of connecting to use.typekit.com when TypeKit's JS is fetching the stylesheet containing the font-face and fonts.
    c. When it ships: consider using the LINK HTTP header with rel=preconnect in the response to the main document to prewarm connections to critical hosts (e.g. use.typekit.com).
2. Optimize the first head chunk* to contain as much meaningful paint blocking / render blocking resources as possible
3. Optimize the subsequent head chunks to contain the render blocking assets that piles on more render blocking assets.
    a. Alternatively, use ASYNC version if available or DEMAND async support if not available.
4. Restructure the document to have the meaningful section as early as possible.
    a. Alternatively/in addition, shorten/bytes-reduce the HTML for the header, left and right nav bars.

tl;dr: Insights for Blink/Chrome

1. Issue warnings in devtools for obsolete third parties (at least the popular one or problematic one)?
2. Issue warnings/tips for head optimization?
3. Pitch link preload, pre… to third parties (AI: kenjibaheux)
4. Something about layout/painting hidden elements??

For reference, first chunk on my run (not useful/reorder, useful/required, unsure/revise)

```
<!doctype html>
<!--
================================================================
== lovingly brought to you by... ===============================
================================================================
 ______    __  __    ______    ______    __  __    ______
```

```
/\  ___\   /\ \_\ \   /\  __ \   /\  == \   /\ \/\ \   /\  ___\
\ \ \____  \ \  __ \  \ \ \/\ \  \ \  __<   \ \ \_\ \  \ \___  \
 \ \_____\  \ \_\ \_\  \ \_____\  \ \_\ \_\  \ \_____\  \/\_____\
  \/_____/   \/_/\/_/   \/_____/   \/_/ /_/   \/_____/   \/_____/
==========================================================================

============================== http://www.voxmedia.com/careers ===
==========================================================================
-->

<!--[if lte IE 8]>   <html class="ie8 no-js">           <![endif]-->
<!--[if IE 9]>       <html class="ie9 no-js">           <![endif]-->
<!--[if gte IE 10]>  <html class="ie10 no-js">          <![endif]-->
<!--[if !IE]><!-->   <html lang="en-US" class="no-js">  <!--<![endif]-->
  <head data-network="verge">
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="google" value="notranslate" />
<meta name="google-site-verification" content="TYyhlycNMOtUSht2aoB7heWTK8m-H45_YJizKavkO8s" />
<meta name="google-site-verification" content="IucFf_TKtbFFH8_YeFyEteQIwYPdANM1R46_U9DpAr4" />
<meta property="twitter:account_id" content="1465737598" />
<meta name="p:domain_verify" content="85c8f3a5bb43c652bbc4414488b7b973"/>
<meta name="application-name" content="The Verge" />
<meta name="msapplication-TileColor" content="#000000"/>
<meta name="msapplication-square70x70logo" content="https://cdn0.vox-cdn.com/images/verge/livetile/small.v5254d9f.png" />
<meta name="msapplication-square150x150logo" content="https://cdn0.vox-cdn.com/images/verge/livetile/medium.v146326d.png" />
<meta name="msapplication-wide310x150logo" content="https://cdn0.vox-cdn.com/images/verge/livetile/wide.v7a20b39.png" />
<meta name="msapplication-square310x310logo" content="https://cdn0.vox-cdn.com/images/verge/livetile/large.vf4ff639.png" />
<meta name="msapplication-notification" content="frequency=30; polling-uri=http://www.theverge.com/livetile/0.xml;
polling-uri2=http://www.theverge.com/livetile/1.xml; polling-uri3=http://www.theverge.com/livetile/2.xml;
polling-uri4=http://www.theverge.com/livetile/3.xml; polling-uri5=http://www.theverge.com/livetile/4.xml" />
  <meta property="article:publisher" content="http://www.facebook.com/verge" />
  <meta property="article:published_time" content="2015-07-26T14:00:15Z" />
  <meta name="author" content="chrisziegler" />


<meta content="authenticity_token" name="csrf-param" />
<meta name="csrf-token" />

<meta data-chorus-version="bfd15b6e3870ad9894d2019725ae24f64379dbab" />


    <link rel="alternate" type="application/rss+xml" title="The Verge - All Posts" href="http://www.theverge.com/rss/full.xml" />
```

```
<link rel="shortcut icon" href="https://cdn0.vox-cdn.com/images/verge/favicon.vc44a54f.ico" />
<link rel="apple-touch-icon" href="https://cdn0.vox-cdn.com/images/verge/2.0/iphone-touch-icon.v3486ec7.png">
<link rel="apple-touch-icon" sizes="76x76" href="https://cdn0.vox-cdn.com/images/verge/2.0/ipad-touch-icon.v9e56a26.png">
<link rel="apple-touch-icon" sizes="120x120" href="https://cdn0.vox-cdn.com/images/verge/2.0/iphone-touch-icon@2x.vf9ccc4a.png">
<link rel="apple-touch-icon" sizes="152x152" href="https://cdn0.vox-cdn.com/images/verge/2.0/ipad-touch-icon@2x.v9d3fdb8.png">
<link rel="icon" sizes="196x196" href="https://cdn0.vox-cdn.com/images/verge/2.0/verge-icon-196x196.v503bbf1.png">


<!--[if lte IE 8]>
  <script src="https://cdn0.vox-cdn.com/javascripts/ie8_head.v15dfe7fec42f97b1.js"></script>
<![endif]-->
<!--[if lte IE 9]>
  <script src="https://cdn0.vox-cdn.com/javascripts/ie9_head.v8419603a28ec4bbd.js"></script>
<![endif]-->
<script type="text/javascript">
  window.Chorus = window.Chorus || {};
  window.Chorus.Context = {
      logged_in    : false
    , user_id      : 0
    , network_domain : "theverge.com"
    , network_slug : "verge"
    , community_id : 372
    , entry_id : 8804686
    , is_preview: false
    , entry_url : "http://www.theverge.com/2015/7/26/9040645/mclaren-650s-spider-first-drive"
    , emc_admin : false
    , community_domain : "theverge.com"
    , community_is_primary: true
    , page_type : "Feature"

  };
  if (!window.Vox && window.Chorus) {
    Vox = {
      Video: Chorus.Video
    }
  }
</script>

<script src="https://cdn0.vox-cdn.com/javascripts/libraries.vc262b07aa2474684.js"></script>
<script src="https://cdn0.vox-cdn.com/javascripts/verge2_head.v636dc21358088775.js"></script>

<script type="text/javascript" src="//fonts.voxmedia.com/wvq7oai.js"></script>
<script type="text/javascript" >try{Typekit.load();}catch(e){}</script>

```

~ Reading an article ~

/ planning /

This section looks into responsiveness aspects (mainly, reading an article after waiting for the first meaningful paint).

Setup
Google Nexus 4 on a 3G network, Chrome 45.0.2454.6
Remote debugging from Chrome dev 45 with Devtools' experimental filmstrip feature enabled.

Goal
optimize for Jank free, 60 FPS scrolling
URL: http://www.theverge.com/2015/7/28/9058211/amazon-new-details-plan-delivery-drone

Overview