Navigating the Critical Path

A Journey into Web Performance

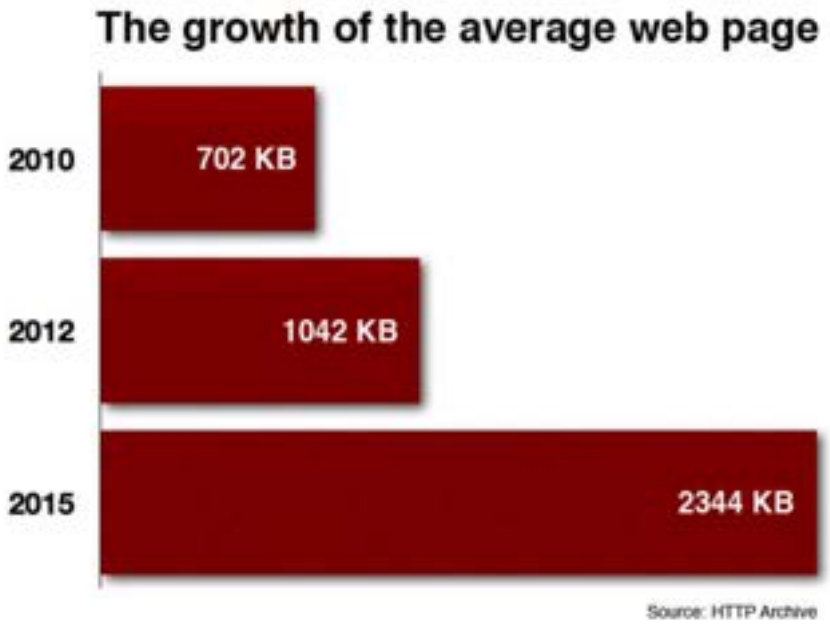
Chris Nguyen
@uncompiled

The Washington Post

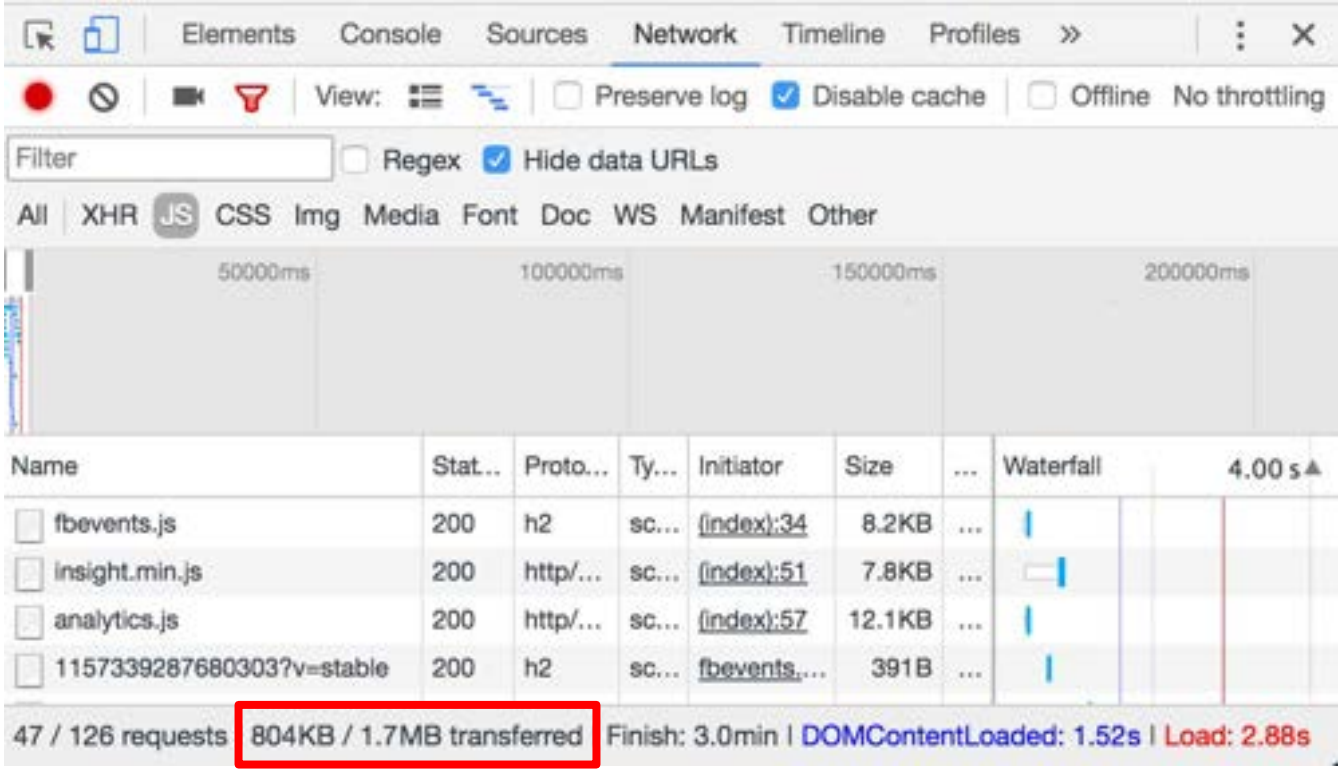
Performance + UX =

http://beartato.net/

The growing epidemic of page bloat



That article is **1.7MB**...



KLINT FINLEY BUSINESS 04.23.16 7:00 AM

THE AVERAGE WEBPAGE IS NOW THE SIZE OF THE ORIGINAL *DOOM*

ID SOFTWARE

The web is *Doom*ed.

Today the average webpage is about the same size, data-wise, as the classic computer game *Doom*, according to software engineer

MOST POPULAR

PRODUCT REVIEW
Review: August Smart

How do ***you*** optimize for performance?

Optimization Techniques

- Concatenate/bundle
- Minify
- Transpile/pre-process
- Code splitting
- Tree shaking

Why do these optimizations exist?

The Network Blocks Everything

`GET /index.html`

- DNS lookup
- TCP connection (slow start)
- TLS handshake
- Server processing
- Content transfer

github.com/davecheney/httpstat

```
▶ httpstat https://www.facebook.com

Connected to 31.13.71.36:443

HTTP/2.0 200 OK
Content-Type: text/html
Max-Age=7776000; path=/; domain=.facebook.com; httponly
Strict-Transport-Security: max-age=15552000; preload

Body discarded

  DNS Lookup   TCP Connection   TLS Handshake   Server Processing   Content Transfer
[     34ms  |          13ms  |        118ms  |             140ms  |             14ms  ]
            |                |               |                    |                   |
   namelookup:34ms           |               |                    |                   |
                       connect:48ms          |                    |                   |
                                   pretransfer:166ms              |                   |
                                                     starttransfer:306ms              |
                                                                                 total:321ms
```

DNS Lookup

```
▶ httpstat https://www.facebook.com

Connected to 31.13.71.36:443

HTTP/2.0 200 OK
Content-Type: text/html
Max-Age=7776000; path=/; domain=.facebook.com; httponly
Strict-Transport-Security: max-age=15552000; preload

Body discarded

  DNS Lookup   TCP Connection   TLS Handshake   Server Processing   Content Transfer
[     34ms  |          13ms  |        118ms  |             140ms  |             14ms  ]
            |                |               |                   |                   |
   namelookup:34ms           |               |                   |                   |
                       connect:48ms          |                   |                   |
                                   pretransfer:166ms             |                   |
                                                     starttransfer:306ms             |
                                                                                 total:321ms
```

TCP Connection

```
▶ httpstat https://www.facebook.com

Connected to 31.13.71.36:443

HTTP/2.0 200 OK
Content-Type: text/html
Max-Age=7776000; path=/; domain=.facebook.com; httponly
Strict-Transport-Security: max-age=15552000; preload

Body discarded

  DNS Lookup   TCP Connection   TLS Handshake   Server Processing   Content Transfer
[     34ms  |          13ms  |        118ms  |             140ms  |             14ms  ]
            |                |               |                    |                   |
   namelookup:34ms           |               |                    |                   |
                        connect:48ms         |                    |                   |
                                   pretransfer:166ms              |                   |
                                                     starttransfer:306ms              |
                                                                                 total:321ms
```

TLS Handshake

```
▶ httpstat https://www.facebook.com

Connected to 31.13.71.36:443

HTTP/2.0 200 OK
Content-Type: text/html
Max-Age=7776000; path=/; domain=.facebook.com; httponly
Strict-Transport-Security: max-age=15552000; preload

Body discarded

  DNS Lookup   TCP Connection   TLS Handshake   Server Processing   Content Transfer
[     34ms  |          13ms  |        118ms  |             140ms  |             14ms  ]
            |                |               |                   |                   |
   namelookup:34ms           |               |                   |                   |
                       connect:48ms          |                   |                   |
                                   pretransfer:166ms             |                   |
                                                     starttransfer:306ms             |
                                                                                 total:321ms
```

Server Processing

```
▶ httpstat https://www.facebook.com

Connected to 31.13.71.36:443

HTTP/2.0 200 OK
Content-Type: text/html
Max-Age=7776000; path=/; domain=.facebook.com; httponly
Strict-Transport-Security: max-age=15552000; preload

Body discarded

  DNS Lookup   TCP Connection   TLS Handshake   Server Processing   Content Transfer
[     34ms  |          13ms  |        118ms  |            140ms  |            14ms  ]
            |                |               |                   |                  |
   namelookup:34ms           |               |                   |                  |
                       connect:48ms          |                   |                  |
                                   pretransfer:166ms             |                  |
                                                     starttransfer:306ms            |
                                                                                total:321ms
```

Content Transfer

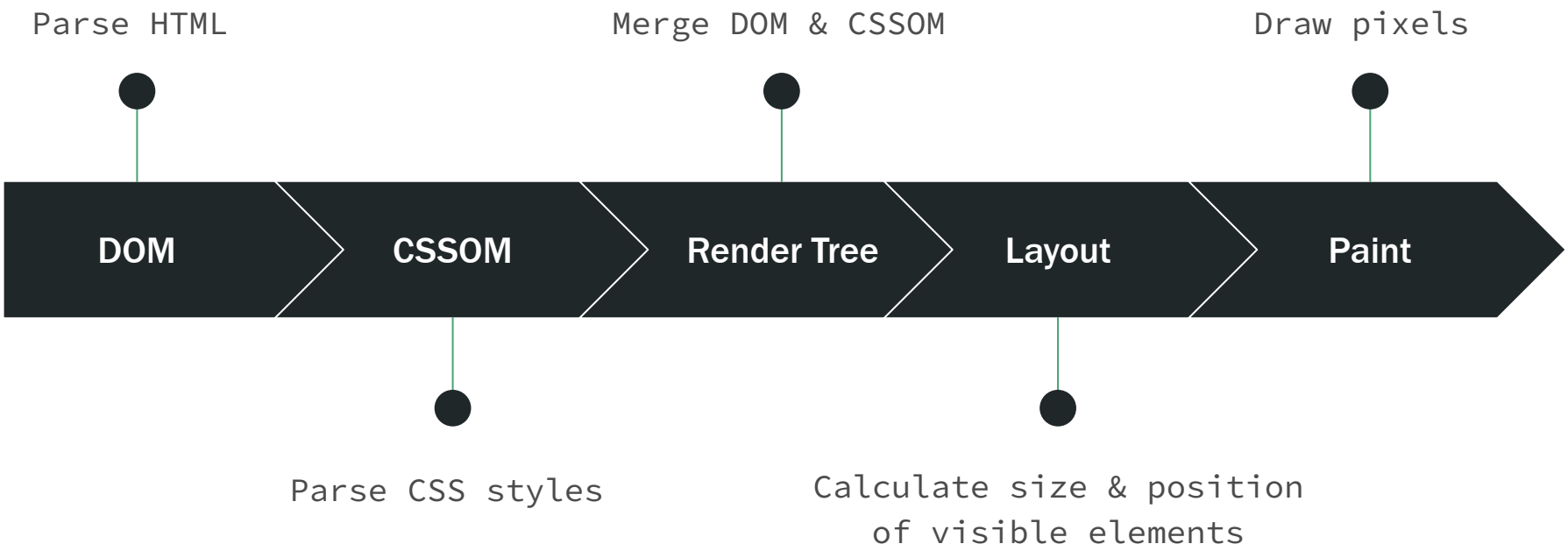
```
▶ httpstat https://www.facebook.com

Connected to 31.13.71.36:443

HTTP/2.0 200 OK
Content-Type: text/html
Max-Age=7776000; path=/; domain=.facebook.com; httponly
Strict-Transport-Security: max-age=15552000; preload

Body discarded

  DNS Lookup   TCP Connection   TLS Handshake   Server Processing   Content Transfer
[     34ms  |          13ms  |        118ms  |             140ms  |             14ms  ]
            |                |               |                   |                   |
   namelookup:34ms           |               |                   |                   |
                        connect:48ms         |                   |                   |
                                    pretransfer:166ms            |                   |
                                                      starttransfer:306ms            |
                                                                                 total:321ms
```

The Network is Expensive

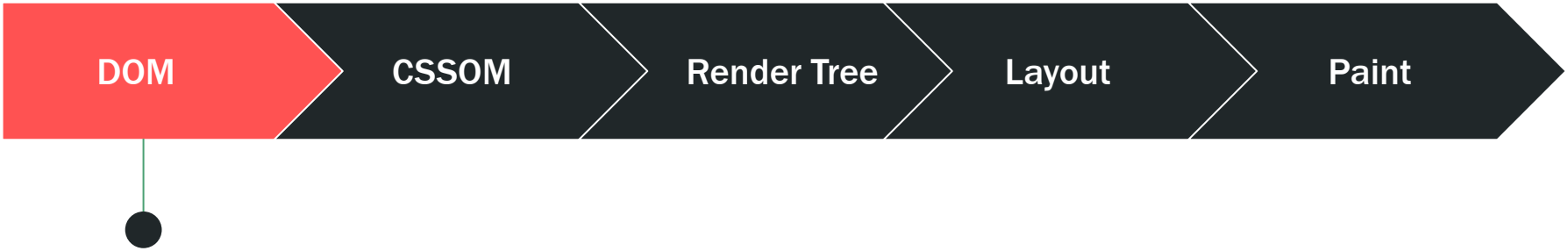
```
▶ httpstat https://www.facebook.com

Connected to 31.13.71.36:443

HTTP/2.0 200 OK
Content-Type: text/html
Max-Age=7776000; path=/; domain=.facebook.com; httponly
Strict-Transport-Security: max-age=15552000; preload

Body discarded

  DNS Lookup   TCP Connection   TLS Handshake   Server Processing   Content Transfer
[     34ms  |          13ms  |        118ms  |             140ms  |             14ms  ]
            |                |               |                    |                   |
   namelookup:34ms           |               |                    |                   |
                        connect:48ms         |                    |                   |
                                   pretransfer:166ms              |                   |
                                                    starttransfer:306ms               |
                                                                                 total:321ms
```


WHAT IS THIS?
Critical Path Planning
I DON'T EVEN
Troll.me
© Disney

Critical Rendering Path

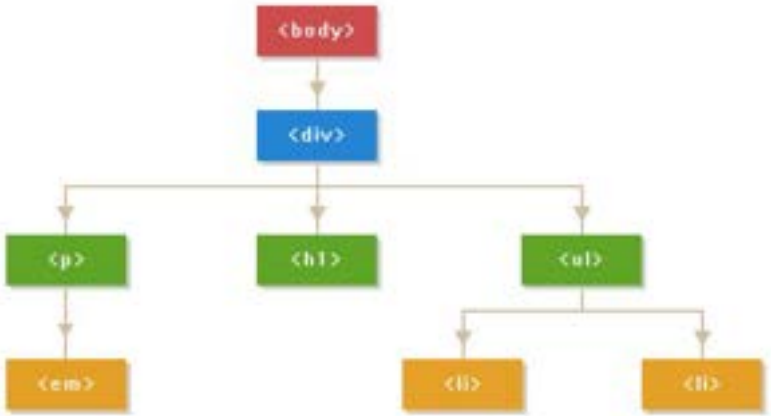


Document Object Model




```
GET /index.html
```

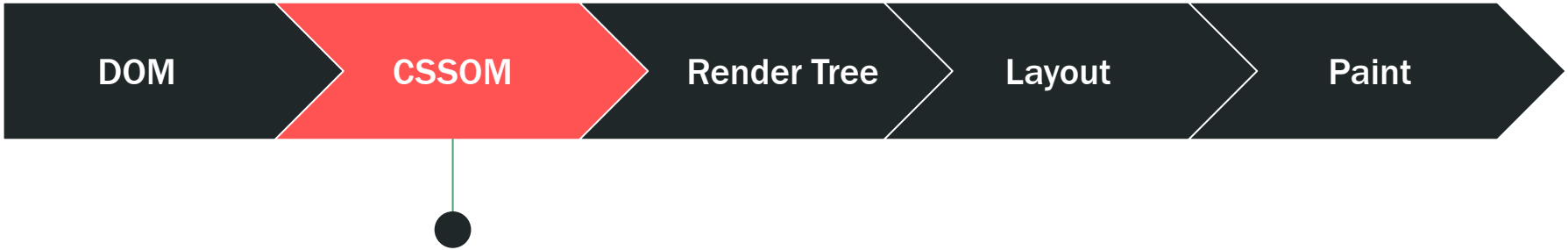
Document Object Model



Parsing HTML is incremental

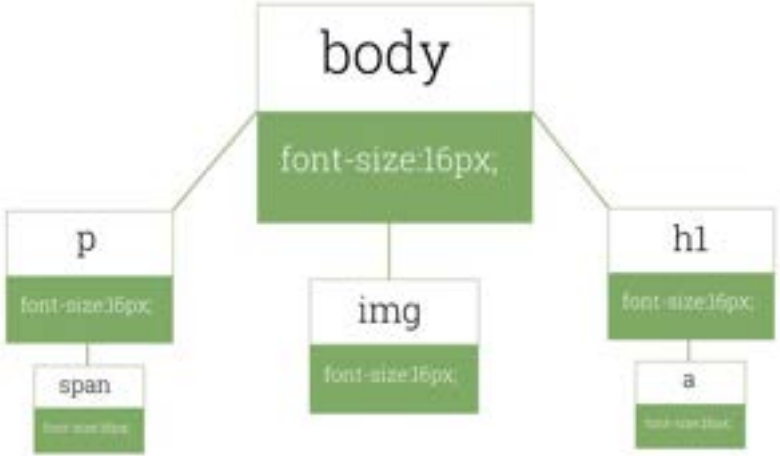
```
<html>
  <head>
    <link …>
    <script …>
  </head>
  <body>
    <div>
    …
```

CSS Object Model




```
<style type="text/css">
  body { color:red; }
</style>

<link rel="stylesheet" type="text/css" href="style.css">
```

CSS Object Model



Problem:

- CSS blocks rendering
- External CSS files require another network round trip

CSS Object Model

https://github.com/addyosmani/critical

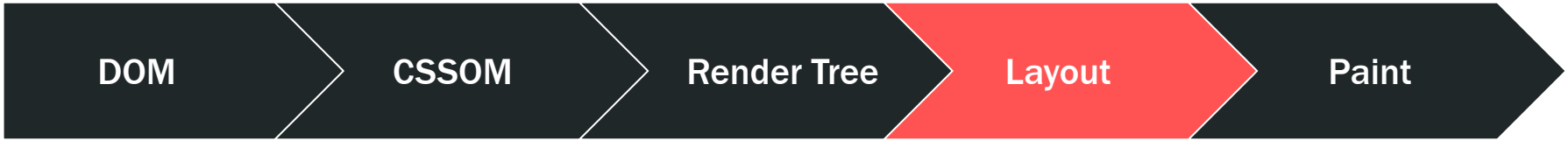
Inline critical CSS:

```
<html>
  <head>
    <style>
      /* critical styles */
    </style>
  </head>
  <body>
    …
```

Render Tree

- Contains elements that need to be rendered
- DOM and CSSOM trees are combined
- Elements with **`display: none`** would not be in the render tree

Layout



- Computes the size and position of visible elements
- **`width:50%`** must evaluate to pixels on the screen
- Layout occurs whenever the render tree or the viewport changes

Paint

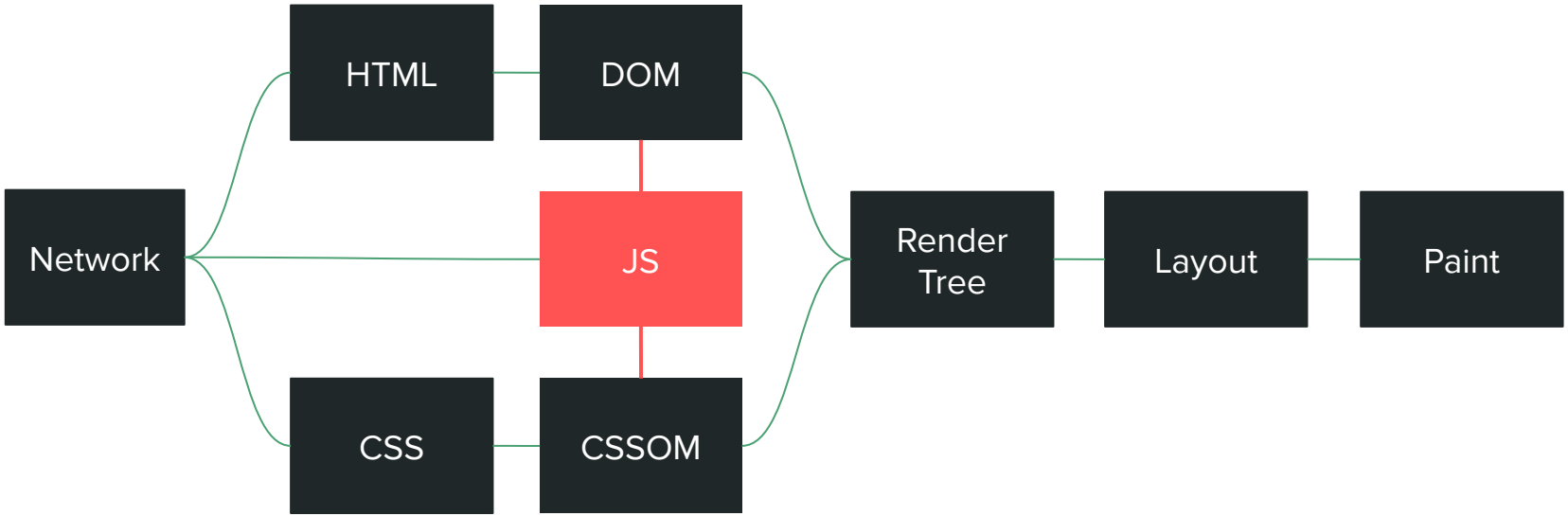
DOM → CSSOM → Render Tree → Layout → **Paint**

```
▸ clipRect(0, 0, 375, 3207, "kIntersect_Op", false)
▸ drawTextBlob(9.84375, 203, paint)
▸ drawTextBlob(9.84375, 223, paint)
▸ drawTextBlob(23.609375, 223, paint)
▸ drawTextBlob(9.84375, 252, paint)
▸ drawTextBlob(9.84375, 280, paint)
▸ drawTextBlob(126.28125, 280, paint)
▸ drawTextBlob(138.203125, 280, paint)
▸ drawTextBlob(9.84375, 309, paint)
▸ drawTextBlob(40.6875, 309, paint)
▸ drawTextBlob(44.84375, 309, paint)
▸ drawTextBlob(140.734375, 309, paint)
▸ drawTextBlob(171.109375, 309, paint)
▸ drawTextBlob(175.265625, 309, paint)
▸ drawRect(0, 0, 375, 125, paint)
  save()
▸ clipRect(8, 92, 92, 176, "kIntersect_Op", true)
▸ drawRRect(rrect, paint)
▸ drawRRect(rrect, paint)
  restore()
```


Hello, it's me...
I was wondering
if after all these years,
you'd like to
run some JavaScript?

JavaScript is...

Critical Rendering Path



JS blocks DOM construction

Browser assumes the worst, unless declared otherwise

- `async` attribute executes JS asynchronously, as soon as it is available
- `defer` attribute executes JS when the HTML has finished parsing

When in doubt, use `defer`.

https://calendar.perfplanet.com/2016/prefer-defer-over-async/









ember

```
<html>
  <head>
    …
  </head>
  <body>
    <div id="😱"></div>
    <script src="app.js">
  </body>
</html>
```

Episode VII

THE HTML AWAKENS

A long time ago, in a galaxy far,

far away....

Web applications shipped HTML

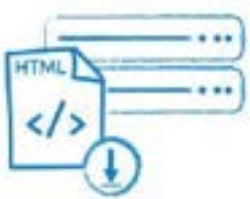
and when users interacted with

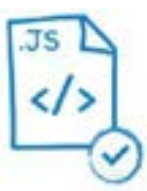

SENORGIF.COM

Client Side Rendering

Server sends bare HTML → Browser fetches JS → Browser executes JS → DOM is inflated → Rendered & interactive

Server Side Rendering



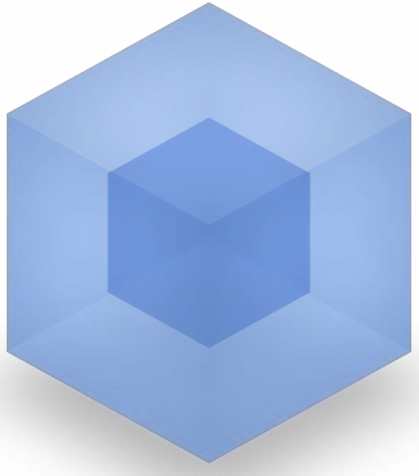


Server sends pre-rendered HTML → Browser renders HTML & fetches JS → Browser executes JS → App is interactive

Should **you** SSR?

Code Splitting

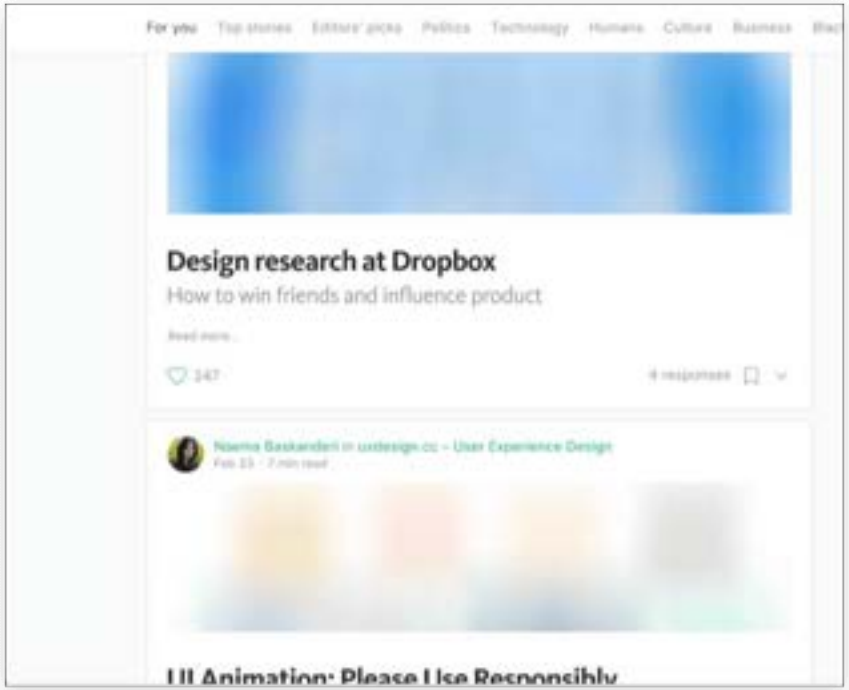
- Small bites are easier to swallow
- webpack support
  - entrypoints
  - CommonsChunkPlugin
  - import()

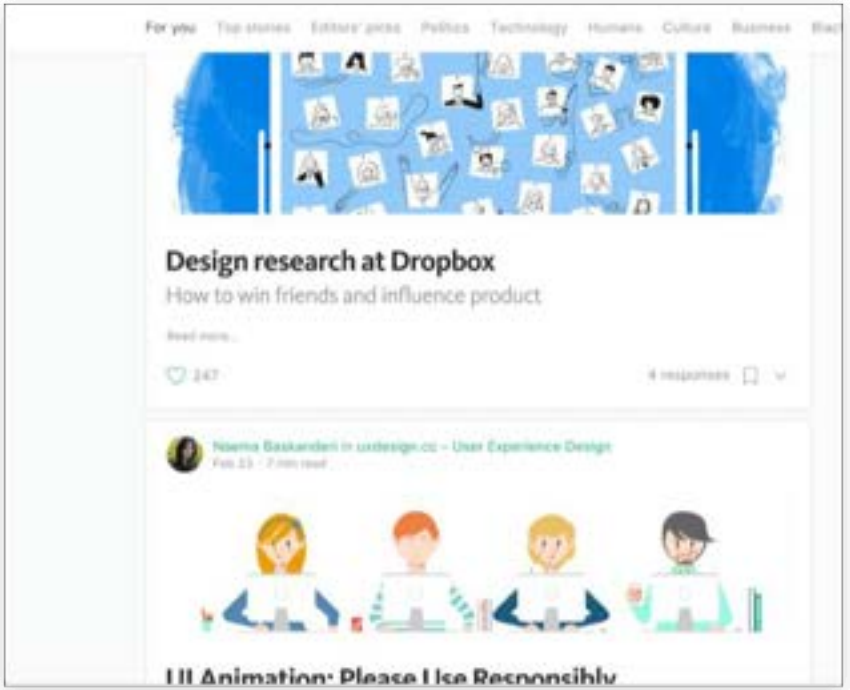


webpack
MODULE BUNDLER

https://webpack.js.org/guides/code-splitting/

Lazy Loading





Service Worker

- Client-side programmable network proxy
- Registered against a domain and path
- JS runs in a background thread

Once you load something, don't load it again!

https://github.com/w3c/ServiceWorker


HTTP/2 SERVER PUSH
2FAST 2PERFORMANT

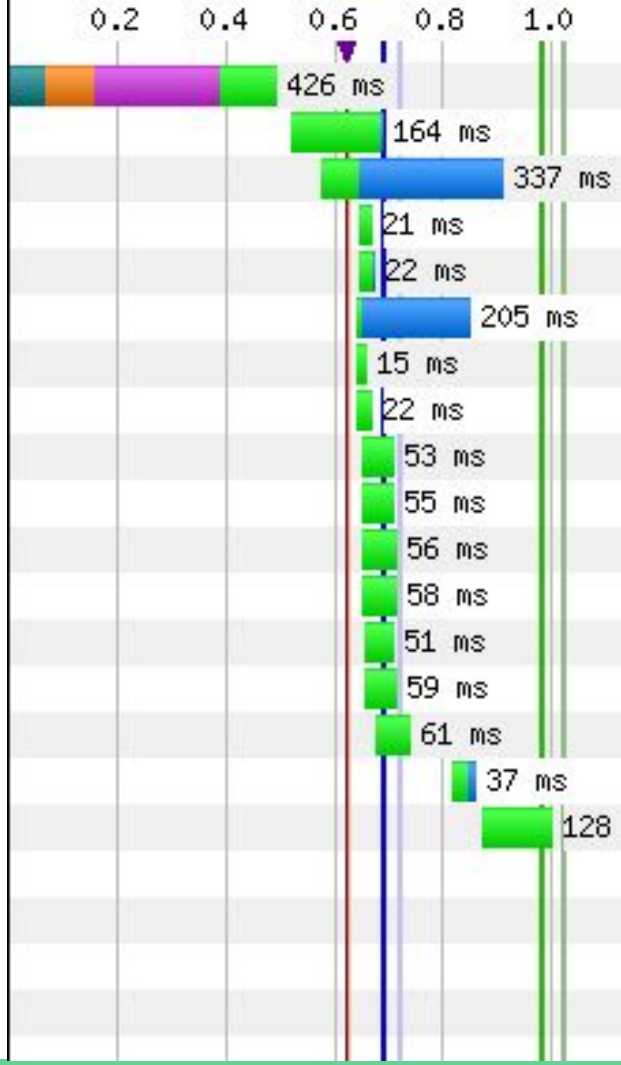
Server Push

GET /index.html

index.html

style.css

app.js



If you don't **measure** it,
you can't **improve** it.

Metrics

- DOMContentLoaded
- Load Event
- Time to First Paint
- First Contentful Paint
- First Meaningful Paint
- Time to Interactive

Metrics

- ~~DOMContentLoaded~~
- ~~Load Event~~
- Time to First Paint
- First Contentful Paint
- First Meaningful Paint
- Time to Interactive

Event reference

DOMContentLoaded

The `DOMContentLoaded` event is fired when the initial HTML document has been completely loaded and parsed, without waiting for stylesheets, images, and subframes to finish loading. A very different event `load` should be used only to detect a fully-loaded page. It is an incredibly popular mistake to use `load` where `DOMContentLoaded` would be much more appropriate, so be cautious.

> **Note:** Synchronous JavaScript pauses parsing of the DOM.

> **Note:** There are also plenty of general-purpose and standalone libraries that offer cross-browser methods to detect that the DOM is ready.

Speeding up

If you want DOM to get parsed as fast as possible after the user had requested the page, some things you could do is turn your JavaScript asynchronous and to optimize loading of stylesheets which if used as usual, slow down page load due to being loaded in parallel, "stealing" traffic from the main html document.

Event reference

Languages Edit

load

See also

Events

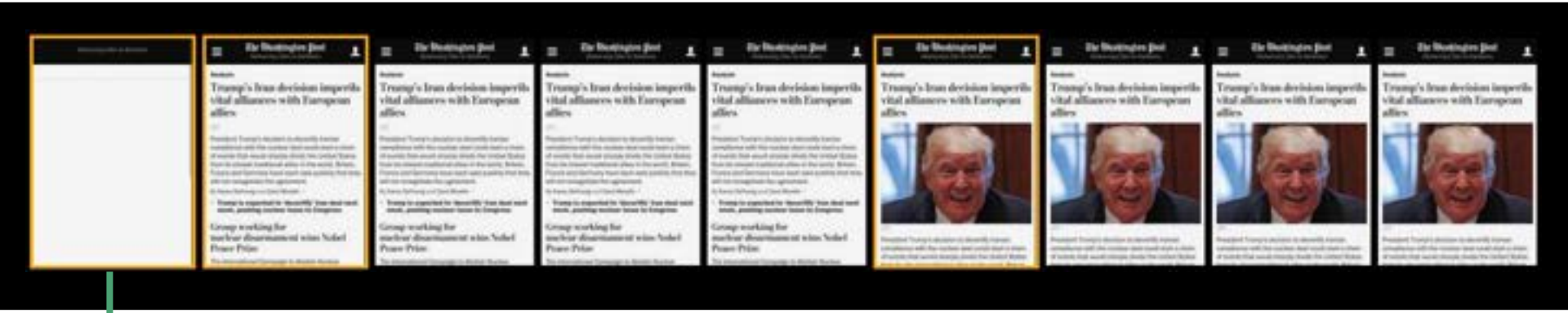
- DOM events
- File API events
- HTML DOM events
- XMLHttpRequest events

The `load` event is fired when a resource and its dependent resources have finished loading.

General info

Specification	DOM L3
Interface	UIEvent
Bubbles	No
Cancelable	No
Target	Window
Default Action	None.

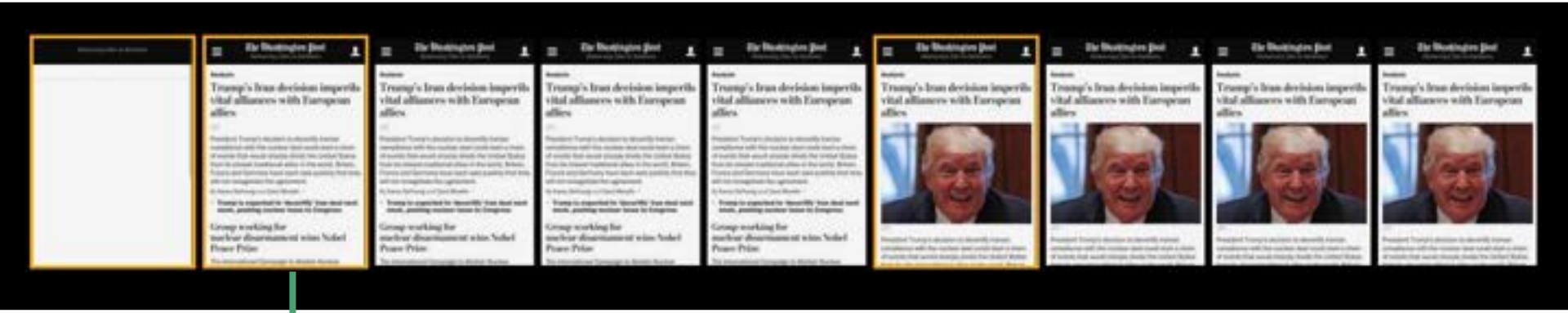
Time to First Paint



The time it takes to show the user **anything.**

Is it happening?!

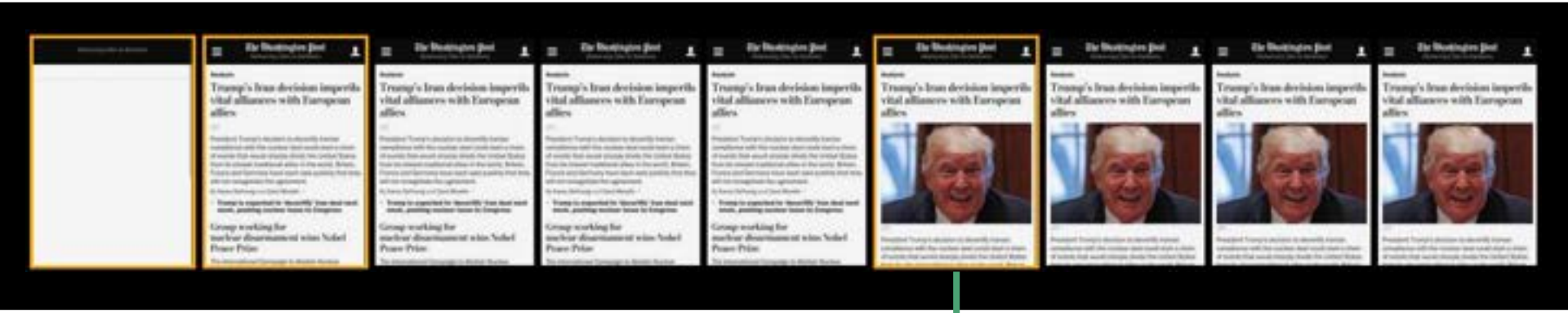
First Contentful Paint



When the browser starts rendering elements from the DOM.

Is it useful?

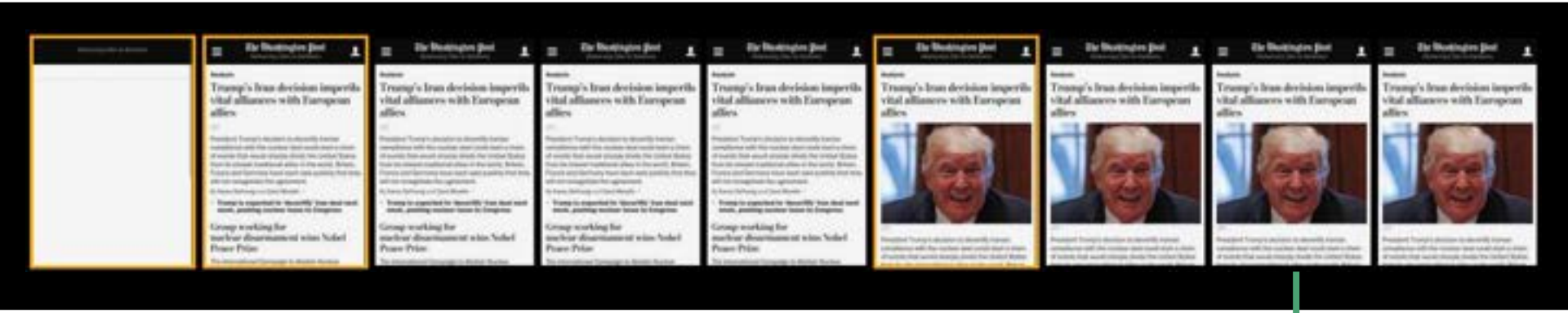
First Meaningful Paint



When the user sees what they came for.

Is it useful?

Time to Interactive



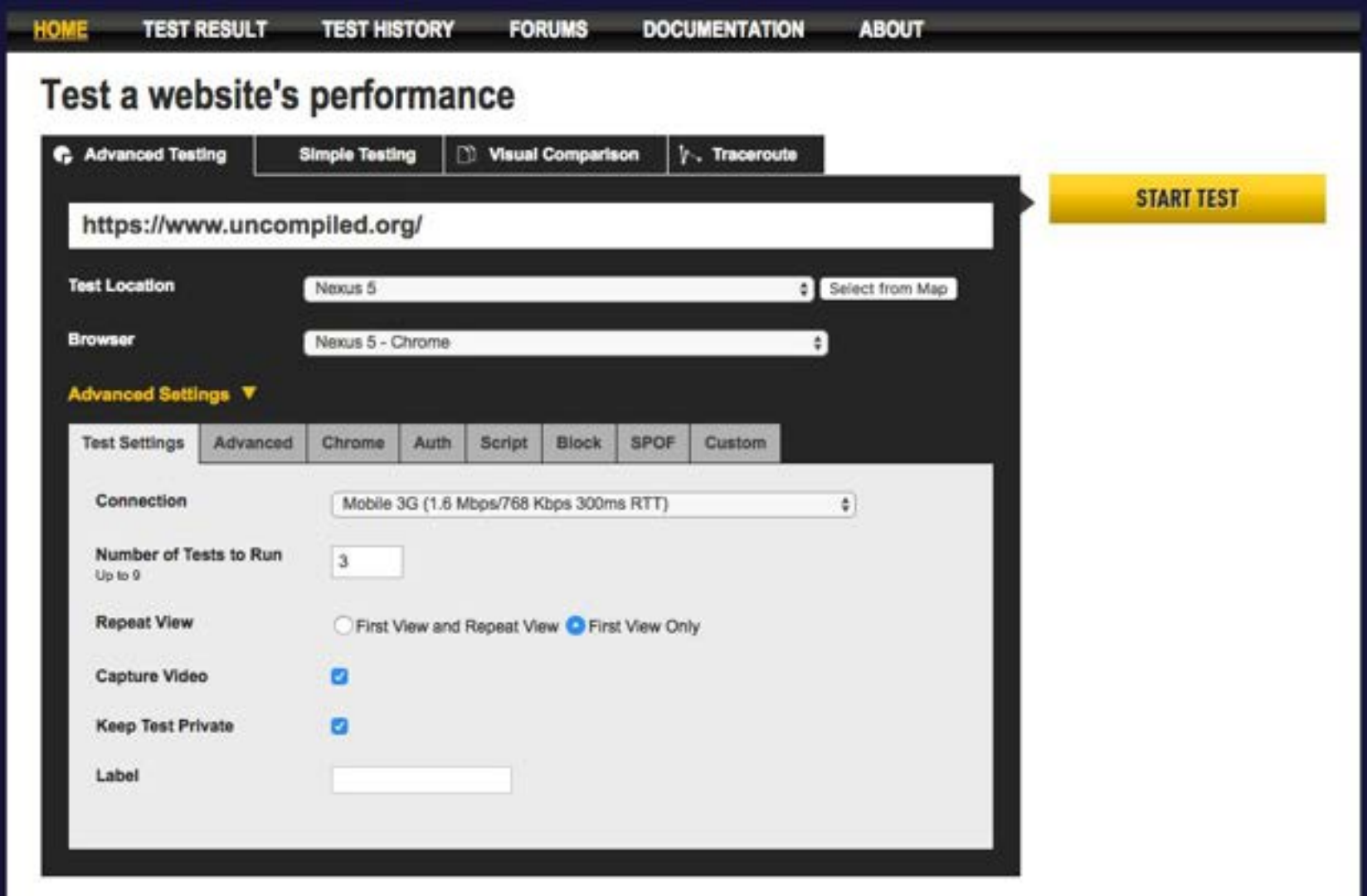
When the user can interact and engage.

Is it useable?

https://github.com/GoogleChromeLabs/tti-polyfill

Audits

- WebPageTest.org
- Developer Tools
- Lighthouse
- Performance Timing APIs


HOME
TEST RESULT
TEST HISTORY
FORUMS
DOCUMENTATION
ABOUT
Test a website's performance
Advanced Testing
Simple Testing
Visual Comparison
Traceroute
START TEST
https://www.uncompiled.org/
Test Location
Nexus 5
Select from Map
Browser
Nexus 5 - Chrome
Advanced Settings
Test Settings
Advanced
Chrome
Auth
Script
Block
SPOF
Custom
Connection
Mobile 3G (1.6 Mbps/768 Kbps 300ms RTT)
Number of Tests to Run
Up to 9
3
Repeat View
First View and Repeat View
First View Only
Capture Video
Keep Test Private
Label

Web Page Performance Test for

https://www.uncompiled.org

From: Dulles, VA - **Nexus 5 - Chrome - 3G**
9/24/2017, 10:48:03 AM

Need help improving?

First Byte Time	Keep-alive Enabled	Compress Transfer	Compress Images	Cache static content	Effective use of CDN
A	A	B	A	F	✓

Summary | Details | Performance Review | Content Breakdown | Domains | Processing Breakdown | Screen Shot | Image Analysis NEW

Tester: Nexus5_3-192.168.0.173
First View only
Test runs: **3**
Re-run the test

Raw page data - Raw object data
Export HTTP Archive (.har)
View Test Log

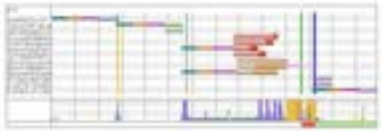
Performance Results (Median Run)

							Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	User Time	Speed Index	First Interactive (beta)	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (Run 2)	4.800s	1.355s	4.712s	4.794s	4717	> 5.009s	4.800s	10	191 KB	6.009s	13	192 KB	$----

Plot Full Results

Test Results

Run 1:

	Waterfall	Screen Shot	Video
First View (5.621s) Timeline (view) Processing Breakdown Trace (view)		 Algorithms for Everyone	Filmstrip View - Watch Video

	Load Time	First Byte	Start Render	User Time	Visually Complete	Speed Index	First Interactive (beta)	Result (error code)	Document Complete			Fully Loaded		
									Time	Requests	Bytes In	Time	Requests	Bytes In
First View (Run 1)	5.621s	1.349s	5.330s	5.625s	5.393s	5333	5.630s	0	5.621s	10	191 KB	6.879s	13	192 KB

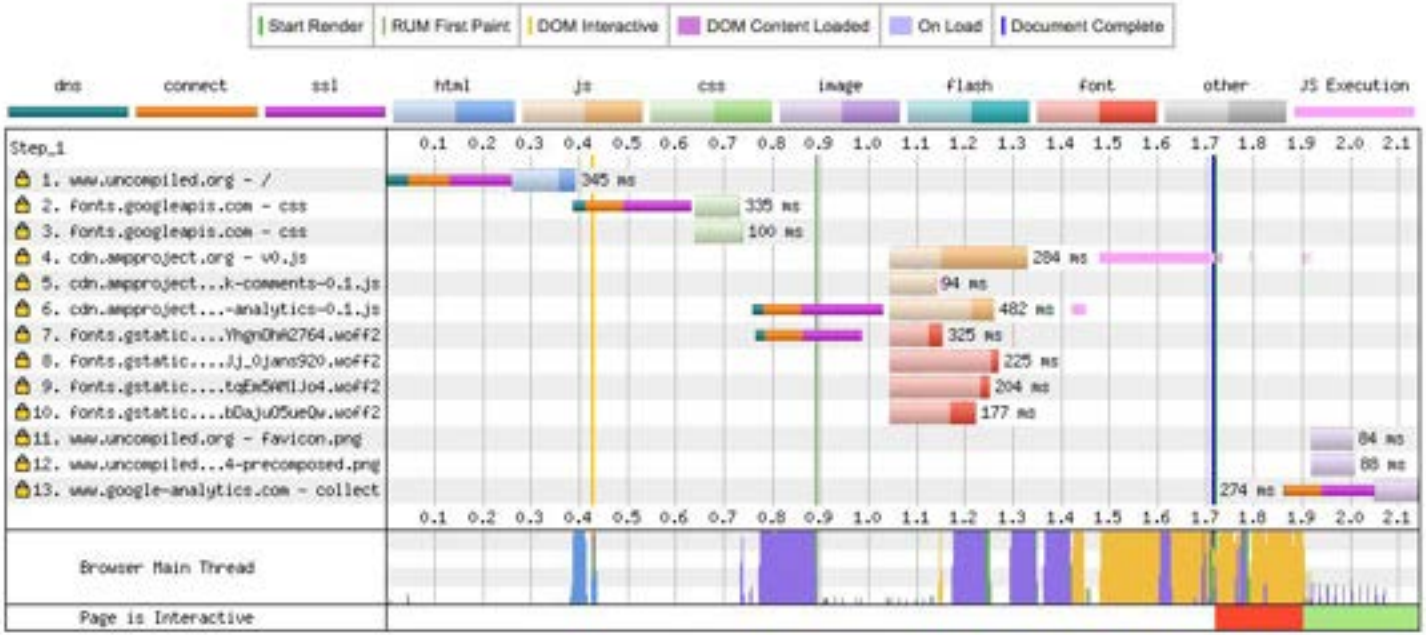
First Interactive (beta)	is	e_is	ofv	mbv	pc	oi	images	Colordepth	RUM First Paint	domInteractive	domContentLoaded	loadEvent
5.630s	5.190s	5.299s	5.332s	5.334s	5.350s	5.625s	[]	32	2.892s	1.429s	1.429s - 1.430s (0.001s)	5.626s - 5.631s (0.005s)

Waterfall View

	Load Time	First Byte	Start Render	User Time	Visually Complete	Speed Index	First Interactive (beta)	Result (error code)	Document Complete			Fully Loaded		
									Time	Requests	Bytes In	Time	Requests	Bytes In
First View (Run 3)	1.714s	0.356s	1.717s	1.713s	1.752s	1719	> 1.904s	0	1.714s	10	191 KB	2.133s	13	192 KB

First Interactive (beta)	is	e_is	ofv	mbv	pc	ol	Images	Colordepth	RUM First Paint	domInteractive	domContentLoaded	loadEvent
> 1.904s	1.555s	1.653s	1.683s	1.685s	1.699s	1.713s	[]	32	0.888s	0.426s	0.426s - 0.426s (0.000s)	1.713s - 1.717s (0.004s)

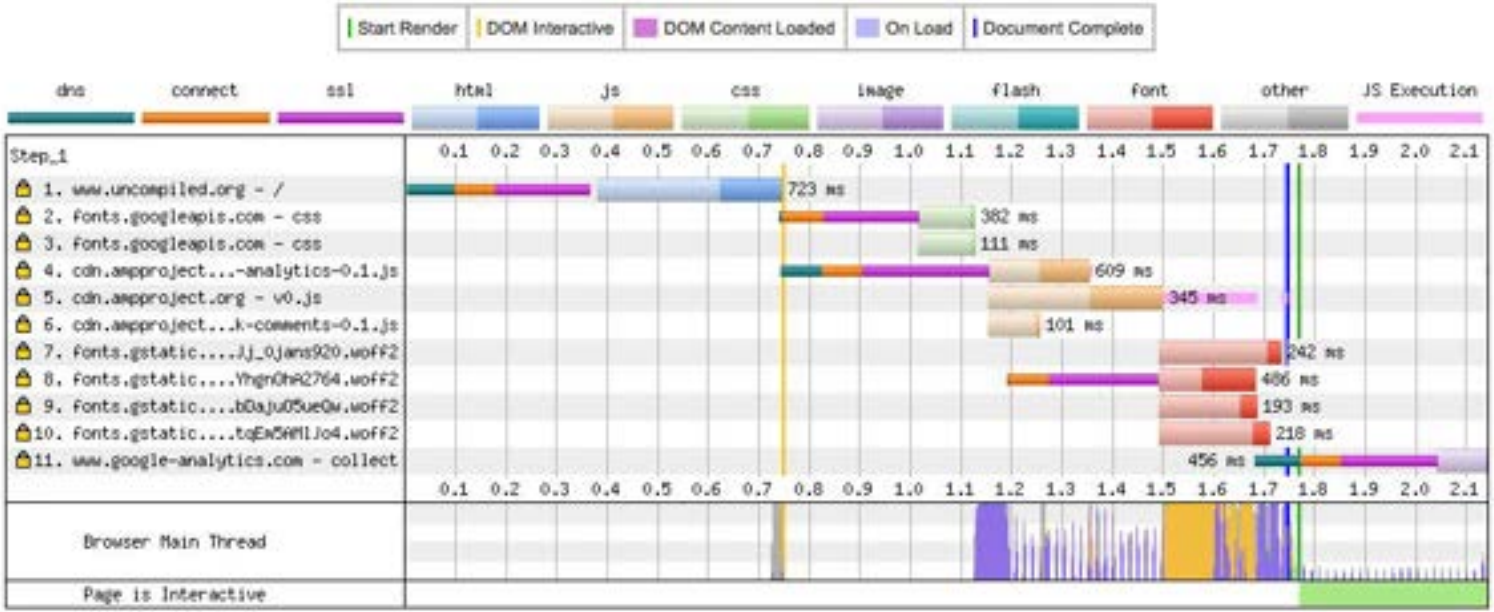
Waterfall View



	Load Time	First Byte	Start Render	User Time	Visually Complete	Speed Index	First Interactive (beta)	Result (error code)	Document Complete			Fully Loaded		
									Time	Requests	Bytes In	Time	Requests	Bytes In
First View (Run 1)	1.745s	0.621s	1.767s	1.738s	1.817s	1776	> 1.767s	0	1.745s	10	202 KB	2.138s	11	203 KB

First Interactive (beta)	is	e_is	ofv	mbv	pc	ol	Images	Colordepth	Resolution	Dpi	domInteractive	domContentLoaded	loadEvent
> 1.767s	1.541s	1.601s	1.618s	1.619s	1.629s	1.738s		32			0.748s	0.748s - 0.748s (0.000s)	1.738s - 1.748s (0.010s)

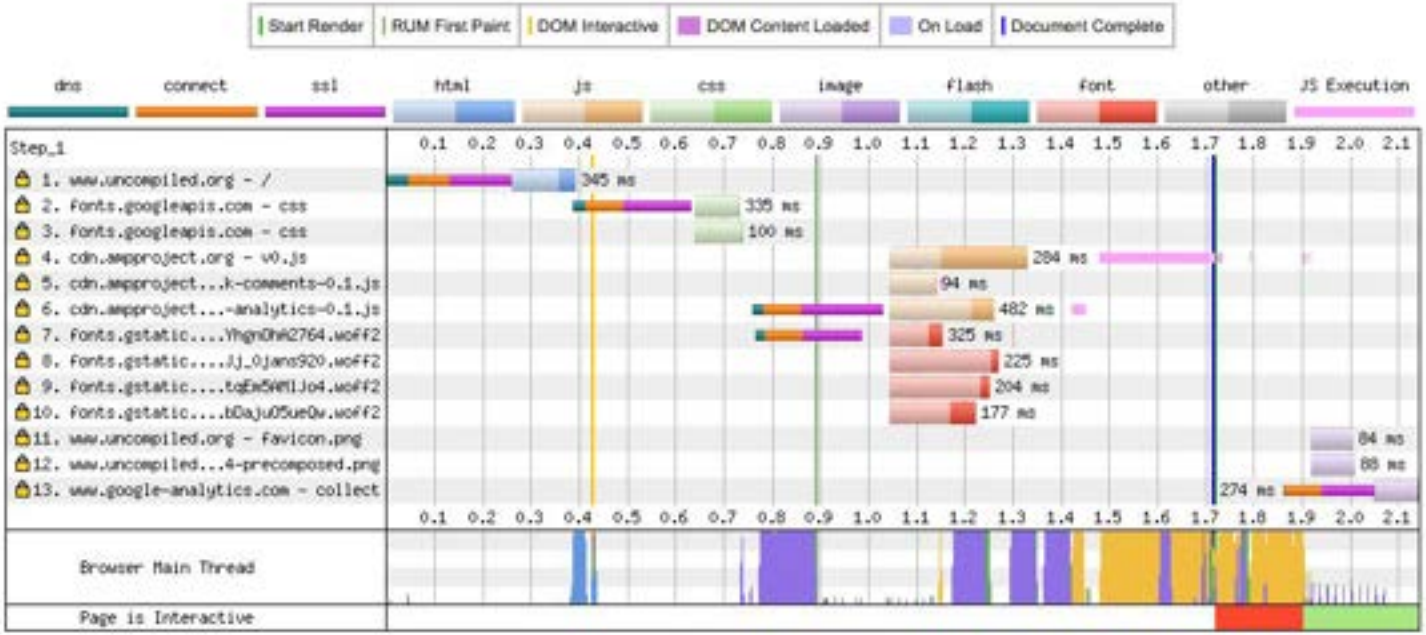
Waterfall View

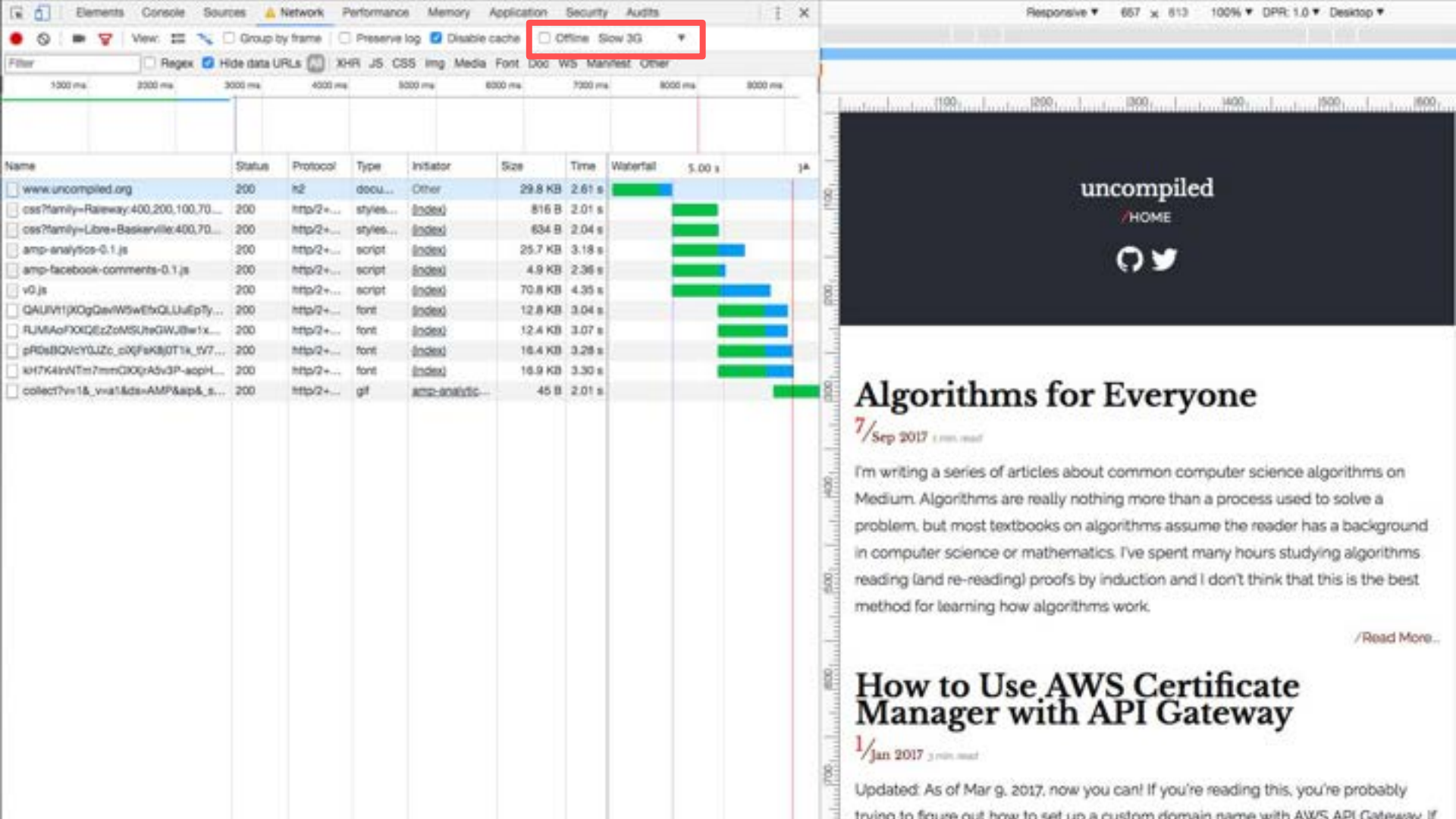


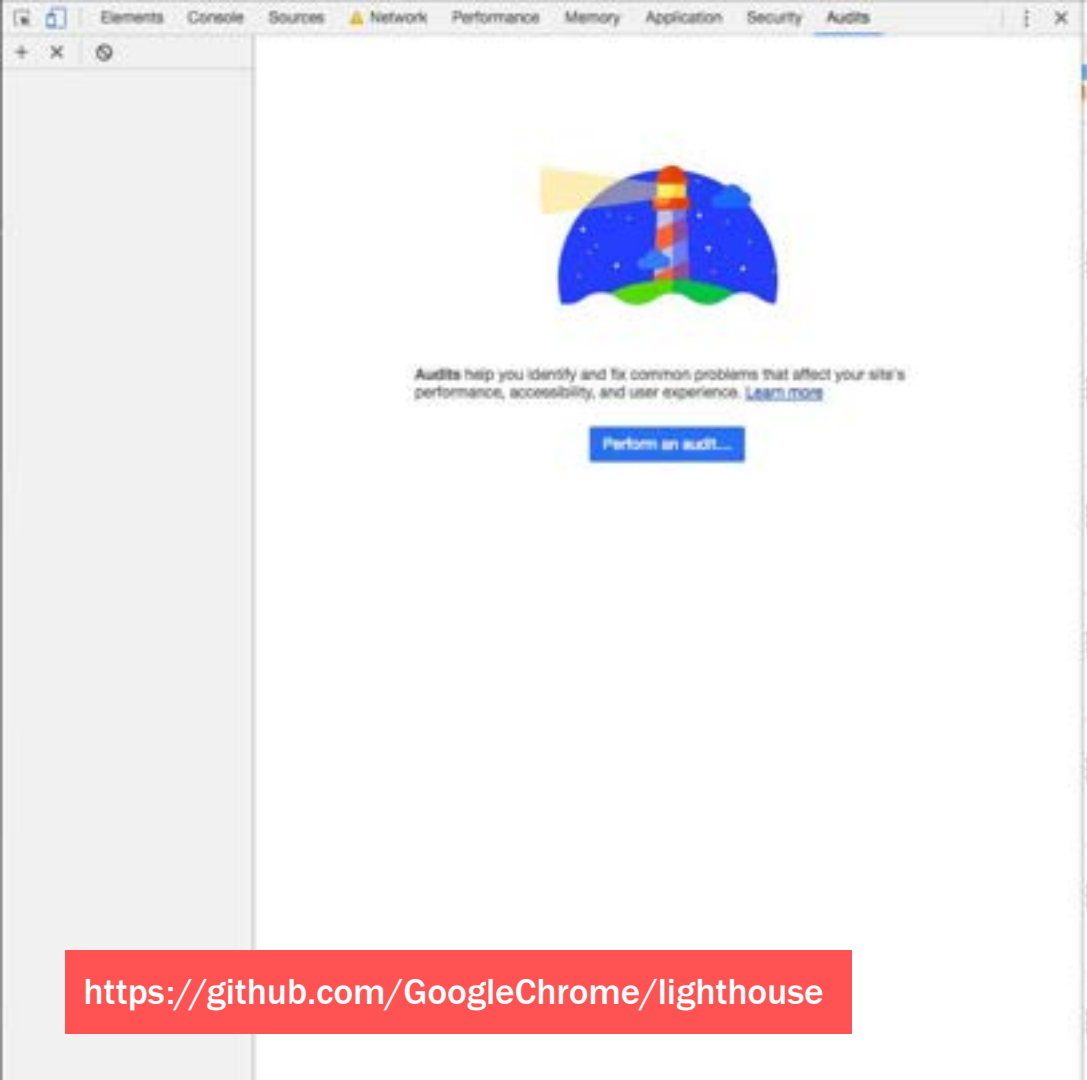
	Load Time	First Byte	Start Render	User Time	Visually Complete	Speed Index	First Interactive (beta)	Result (error code)	Document Complete			Fully Loaded		
									Time	Requests	Bytes In	Time	Requests	Bytes In
First View (Run 3)	1.714s	0.356s	1.717s	1.713s	1.752s	1719	> 1.904s	0	1.714s	10	191 KB	2.133s	13	192 KB

First Interactive (beta)	is	e_is	ofv	mbv	pc	ol	Images	Colordepth	RUM First Paint	domInteractive	domContentLoaded	loadEvent
> 1.904s	1.555s	1.653s	1.683s	1.685s	1.699s	1.713s	[]	32	0.888s	0.426s	0.426s - 0.426s (0.000s)	1.713s - 1.717s (0.004s)

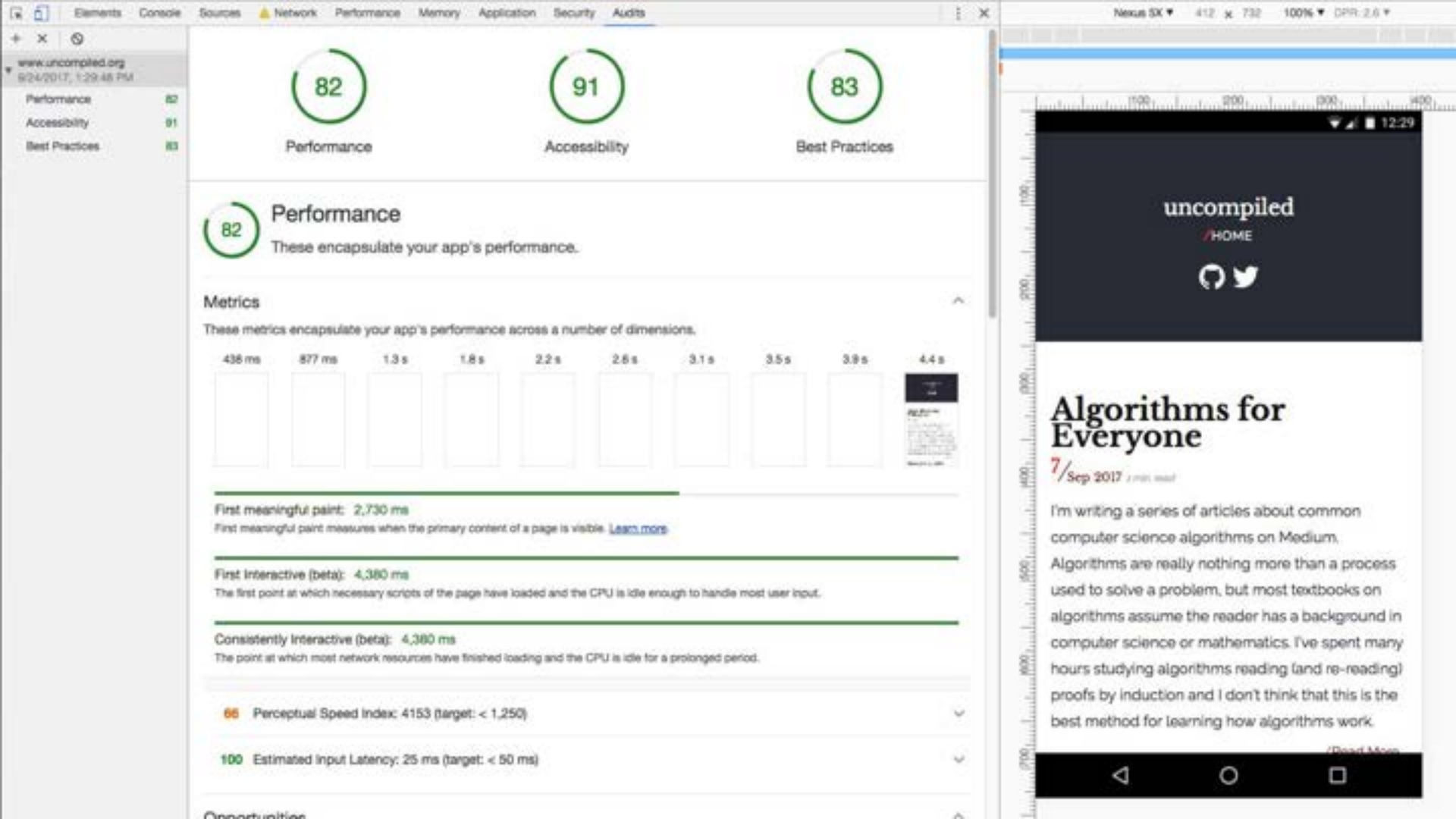
Waterfall View

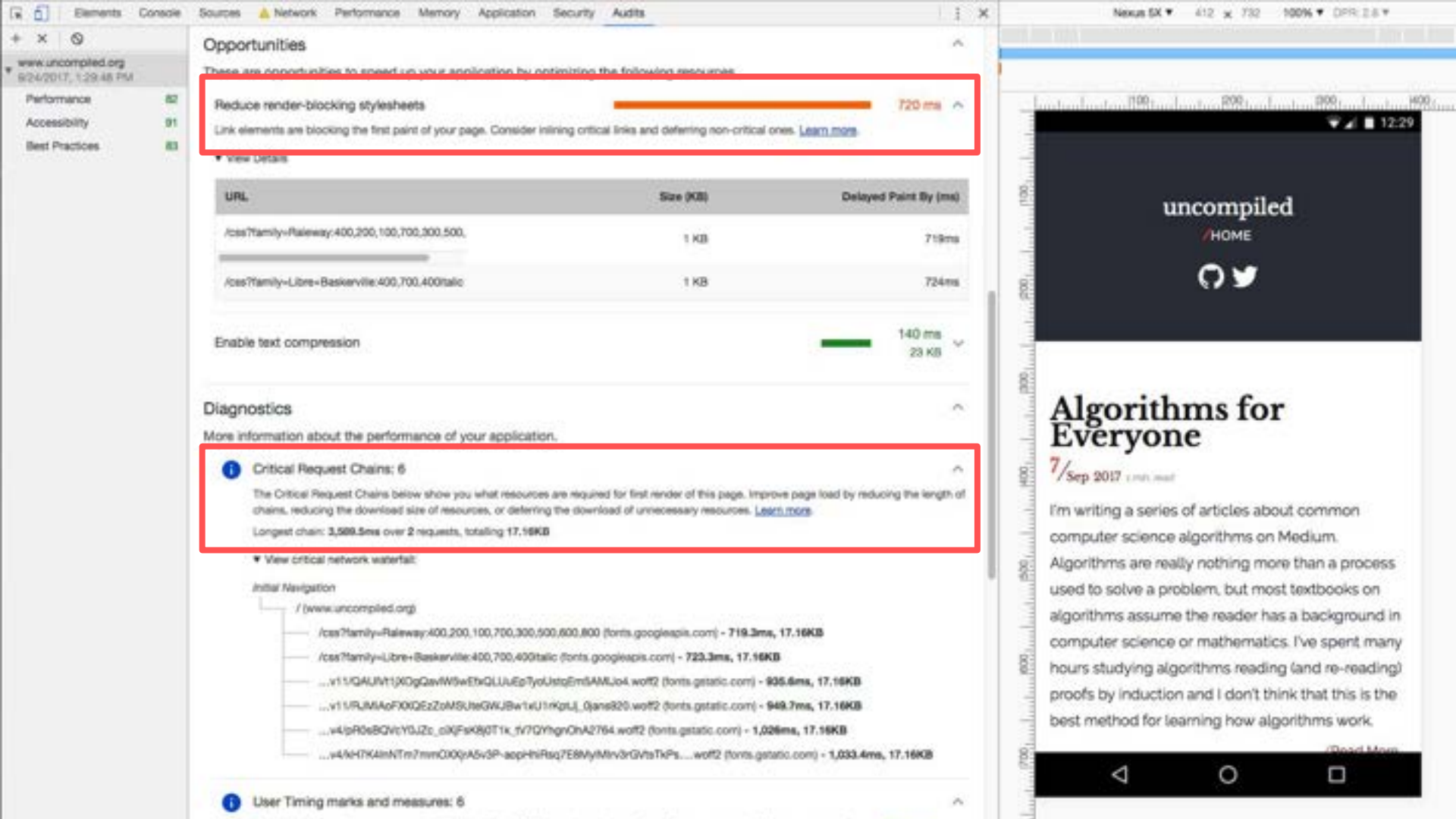



Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
Offline
Slow 3G
Disable cache
Preserve log
Group by frame
Hide data URLs
XHR JS CSS Img Media Font Doc WS Manifest Other
Name
Status
Protocol
Type
Initiator
Size
Time
Waterfall
www.uncompiled.org
css?family=Raleway:400,200,100,70...
css?family=Libre+Baskerville:400,70...
amp-analytics-0.1.js
amp-facebook-comments-0.1.js
v0.js
collect?v=1&_v=a1&ds=AMP&aip&_s...
uncompiled
/HOME
Algorithms for Everyone
7/Sep 2017
I'm writing a series of articles about common computer science algorithms on Medium. Algorithms are really nothing more than a process used to solve a problem, but most textbooks on algorithms assume the reader has a background in computer science or mathematics. I've spent many hours studying algorithms reading (and re-reading) proofs by induction and I don't think that this is the best method for learning how algorithms work.
/Read More...
How to Use AWS Certificate Manager with API Gateway
1/Jan 2017
Updated: As of Mar 9, 2017, now you can! If you're reading this, you're probably



https://github.com/GoogleChrome/lighthouse


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
www.uncompiled.org
9/24/2017, 1:29:48 PM
Performance 82
Accessibility 91
Best Practices 83
82
Performance
91
Accessibility
83
Best Practices
82
Performance
These encapsulate your app's performance.
Metrics
These metrics encapsulate your app's performance across a number of dimensions.
438 ms
877 ms
1.3 s
1.8 s
2.2 s
2.6 s
3.1 s
3.5 s
3.9 s
4.4 s
First meaningful paint: 2,730 ms
First meaningful paint measures when the primary content of a page is visible. Learn more
First Interactive (beta): 4,380 ms
The first point at which necessary scripts of the page have loaded and the CPU is idle enough to handle most user input.
Consistently Interactive (beta): 4,380 ms
The point at which most network resources have finished loading and the CPU is idle for a prolonged period.
66 Perceptual Speed Index: 4153 (target: < 1,250)
100 Estimated Input Latency: 25 ms (target: < 50 ms)
Nexus 5X
412 x 732
100%
DPR: 2.6
12:29
uncompiled
/HOME
Algorithms for Everyone
7/Sep 2017
I'm writing a series of articles about common computer science algorithms on Medium. Algorithms are really nothing more than a process used to solve a problem, but most textbooks on algorithms assume the reader has a background in computer science or mathematics. I've spent many hours studying algorithms reading (and re-reading) proofs by induction and I don't think that this is the best method for learning how algorithms work.


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
www.uncompiled.org
9/24/2017, 1:29:48 PM
Performance 82
Accessibility 91
Best Practices 83
Opportunities
Reduce render-blocking stylesheets
720 ms
Link elements are blocking the first paint of your page. Consider inlining critical links and deferring non-critical ones. Learn more
URL
Size (KB)
Delayed Paint By (ms)
/css?family=Raleway:400,200,100,700,300,500,
1 KB
719ms
/css?family=Libre+Baskerville:400,700,400italic
1 KB
724ms
Enable text compression
140 ms
23 KB
Diagnostics
More information about the performance of your application.
Critical Request Chains: 6
The Critical Request Chains below show you what resources are required for first render of this page. Improve page load by reducing the length of chains, reducing the download size of resources, or deferring the download of unnecessary resources. Learn more
Longest chain: 3,589.5ms over 2 requests, totaling 17.16KB
View critical network waterfall:
Initial Navigation
/ (www.uncompiled.org)
/css?family=Raleway:400,200,100,700,300,500,600,800 (fonts.googleapis.com) - 719.3ms, 17.16KB
/css?family=Libre+Baskerville:400,700,400italic (fonts.googleapis.com) - 723.3ms, 17.16KB
...v11/QAUIVt1jXDgQavIW5wEfxQLUuEpTyoUstqEm5AMLJo4.woff2 (fonts.gstatic.com) - 935.6ms, 17.16KB
...v11/RJMlAoFXKQEzZoMSUteGWJBw1xU1rKptJj_0jans920.woff2 (fonts.gstatic.com) - 949.7ms, 17.16KB
...v4/pR0eBQVcYGJZc_o3KjFsK8j0T1k_fV7QYhgnOhA2764.woff2 (fonts.gstatic.com) - 1,026ms, 17.16KB
...v4/NH7K4lnNTm7mmOX0rA5v3P-aopHsRsq7E8MyIMlrv3rGVtsTkPs....woff2 (fonts.gstatic.com) - 1,033.4ms, 17.16KB
User Timing marks and measures: 6
Nexus 5X
412 x 732
100%
DPR: 2.6
12:29
uncompiled
/HOME
Algorithms for Everyone
7/Sep 2017
I'm writing a series of articles about common computer science algorithms on Medium. Algorithms are really nothing more than a process used to solve a problem, but most textbooks on algorithms assume the reader has a background in computer science or mathematics. I've spent many hours studying algorithms reading (and re-reading) proofs by induction and I don't think that this is the best method for learning how algorithms work.

Tree: e29b3d6af1 ▾ personal-blog / themes / redlounge / layouts / partials / header.html

Find file | Copy path

uncompiled Add comments — e29b3d6 17 days ago

1 contributor

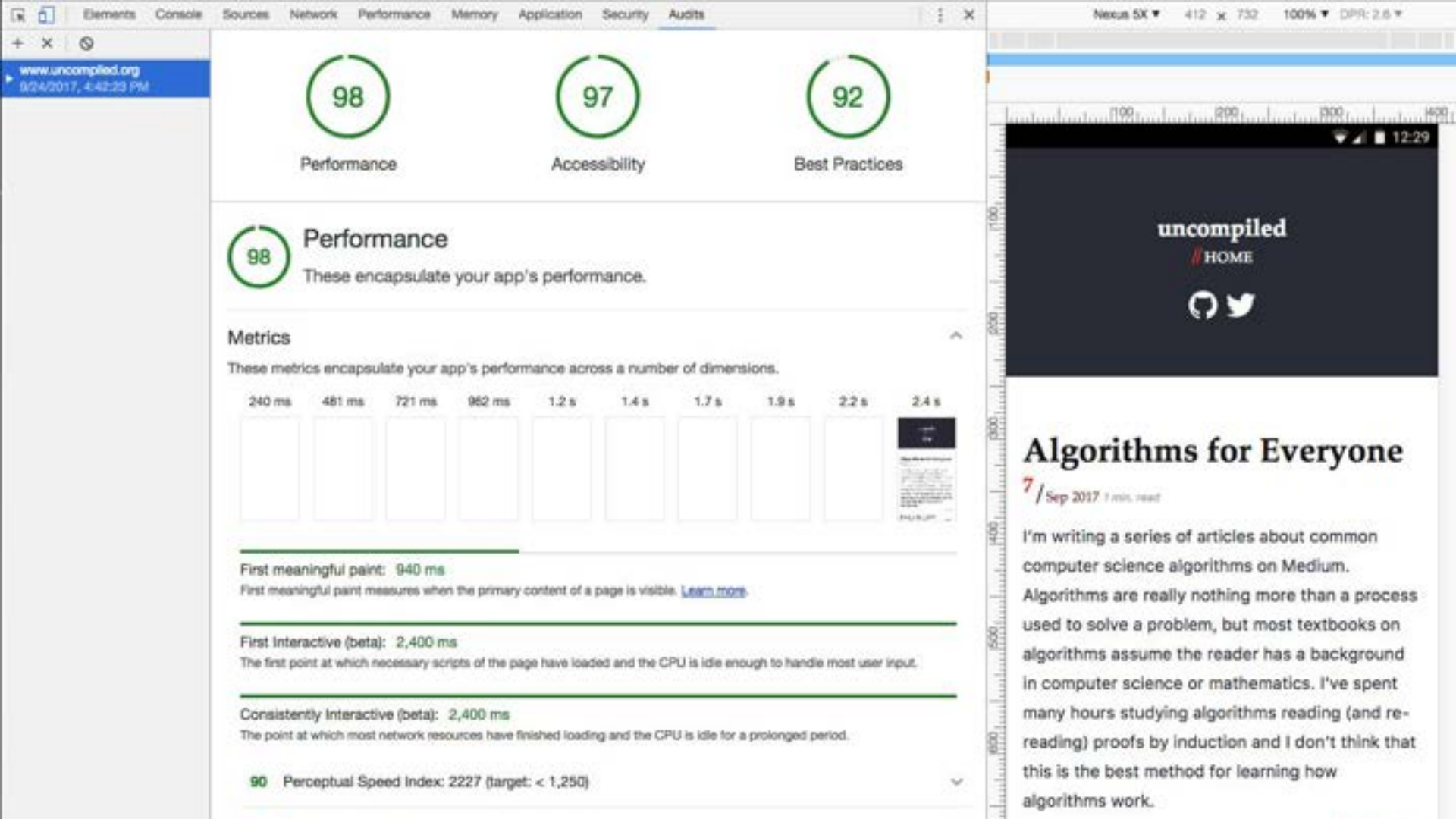
26 lines (19 sloc) | 1.72 KB

Raw | Blame | History

```
{{ partial "doctype.html" . }}
<head>
  <title>
    {{ .Title }}{{ if ne .Title .Site.Title }} // {{ .Site.Title }}{{ end }}
  </title>

  {{ partial "meta.html" . }}

  {{ partial "og.html" . }}

  <!-- CSS -->
  {{ partial "css-amp-custom.html" . }}

  <link href='https://fonts.googleapis.com/css?family=Raleway:400,200,100,700,300,500,600,800' rel='stylesheet' type='text/css'
  <link href='https://fonts.googleapis.com/css?family=Libre+Baskerville:400,700,400italic' rel='stylesheet' type='text/css'>

  <!-- Icons -->
  <link rel="apple-touch-icon-precomposed" sizes="144x144" href="/touch-icon-144-precomposed.png">
  <link rel="shortcut icon" href="/favicon.png">

  <style amp-boilerplate>body{-webkit-animation:-amp-start 8s steps(1,end) 0s 1 normal both;-moz-animation:-amp-start 8s steps(
  <script async custom-element="amp-analytics" src="https://cdn.ampproject.org/v0/amp-analytics-0.1.js"></script>
  <script async custom-element="amp-facebook-comments" src="https://cdn.ampproject.org/v0/amp-facebook-comments-0.1.js"></scrip
  <script async src="https://cdn.ampproject.org/v0.js"></script>
</head>
```

Using System Fonts

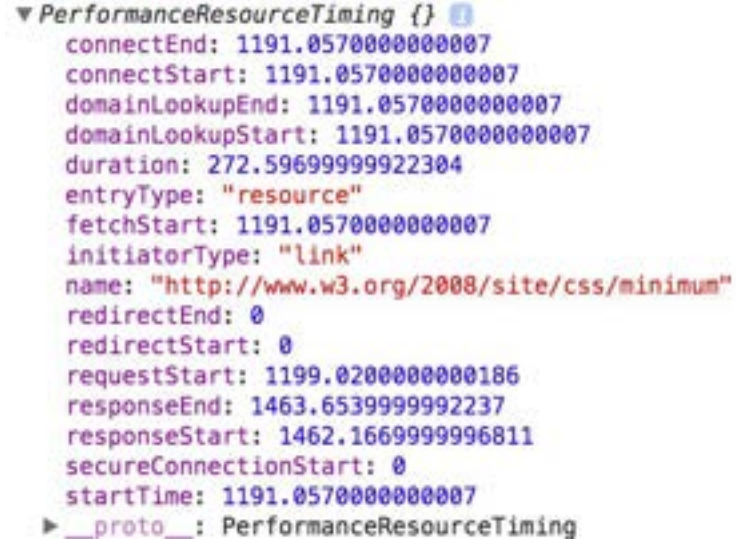
```
body {
  font-family: -apple-system,
               BlinkMacSystemFont,
               "Segoe UI", Roboto,
               Oxygen-Sans, Ubuntu,
               Cantarell, "Helvetica Neue",
               sans-serif;
}
```


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
www.uncompiled.org
9/24/2017, 4:42:29 PM
98
Performance
97
Accessibility
92
Best Practices
98
Performance
These encapsulate your app's performance.
Metrics
These metrics encapsulate your app's performance across a number of dimensions.
240 ms
481 ms
721 ms
962 ms
1.2 s
1.4 s
1.7 s
1.9 s
2.2 s
2.4 s
First meaningful paint: 940 ms
First meaningful paint measures when the primary content of a page is visible. Learn more.
First Interactive (beta): 2,400 ms
The first point at which necessary scripts of the page have loaded and the CPU is idle enough to handle most user input.
Consistently Interactive (beta): 2,400 ms
The point at which most network resources have finished loading and the CPU is idle for a prolonged period.
90 Perceptual Speed Index: 2227 (target: < 1,250)
Nexus 5X
412 x 732
100%
DPR: 2.6
12:29
uncompiled
// HOME
Algorithms for Everyone
7 / Sep 2017
I'm writing a series of articles about common computer science algorithms on Medium. Algorithms are really nothing more than a process used to solve a problem, but most textbooks on algorithms assume the reader has a background in computer science or mathematics. I've spent many hours studying algorithms reading (and re-reading) proofs by induction and I don't think that this is the best method for learning how algorithms work.

Web Performance Timing APIs

High-resolution timestamps for measuring performance

- performance.timing
- performance.getEntriesByType("resource")



https://w3c.github.io/perf-timing-primer/

Web Performance Timing APIs

User Timings

- performance.mark
- performance.measure

```
const marky = {
  mark: function mark(name) {
    if (performance.mark === undefined) {
      console.log("performance.mark Not supported");
      return;
    }
      // Create the performance mark
      performance.mark(name);
  }
}

$.get('/dreams', function(dreams) {
  dreams.forEach(function(dream) {
    $('<li></li>').text(dream).appendTo('ul#dreams');
  });
  marky.mark('dcjs_mark');
  console.log('dreams updated');
});
```

https://w3c.github.io/perf-timing-primer/

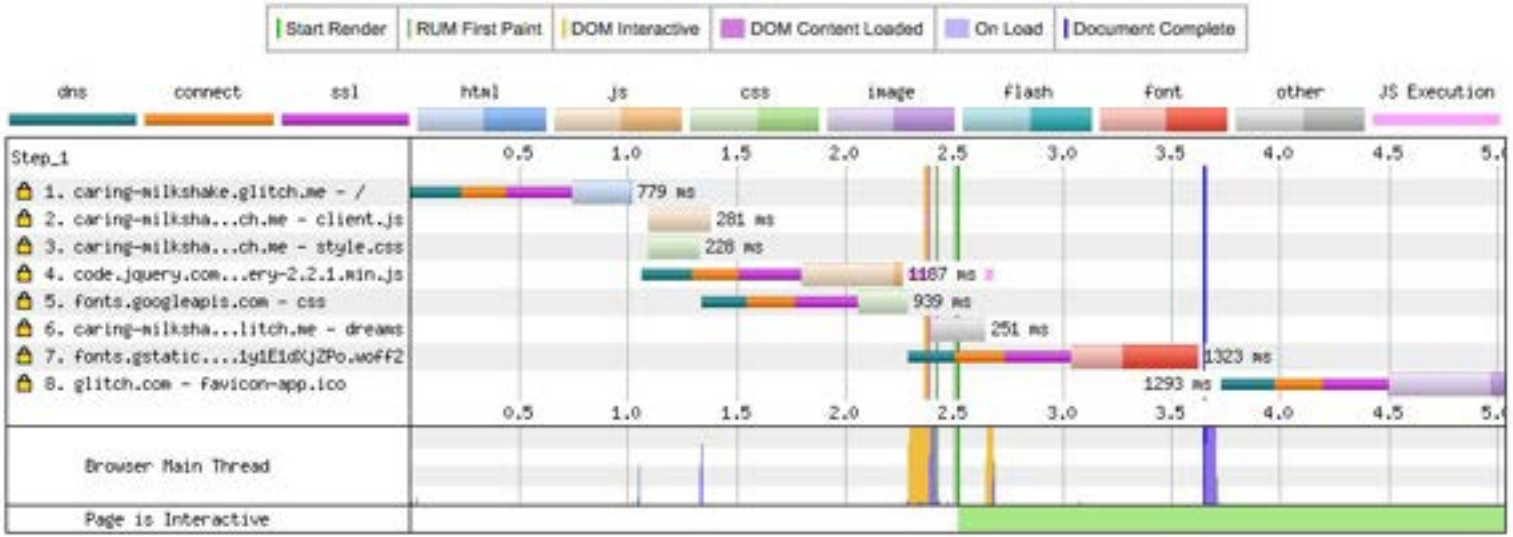
Tester: Nexus5_2-192.168.0.172
First View only
Test runs: **3**

Export HTTP Archive (.har)
Custom Metrics

	Load Time	First Byte	Start Render	User Time	Visually Complete	Speed Index	First Interactive (beta)	Result (error code)	Document Complete: Time	Document Complete: Requests	Document Complete: Bytes In	Fully Loaded: Time	Fully Loaded: Requests	Fully Loaded: Bytes In
First View (Run 1)	3.650s	1.009s	2.516s	2.672s	3.687s	2943	2.415s	0	3.650s	7	85 KB	5.030s	8	111 KB

First Interactive (beta)	dcjs_mark	Images	Colordepth	RUM First Paint	domInteractive	domContentLoaded	loadEvent
2.415s	2.672s	[]	32	2.414s	2.363s	2.363s - 2.384s (0.021s)	3.653s - 3.653s (0.000s)

Waterfall View



Maintaining Performance

- Set performance budgets
- Make everyone aware of the performance cost of adding and removing features
- Automate measurement

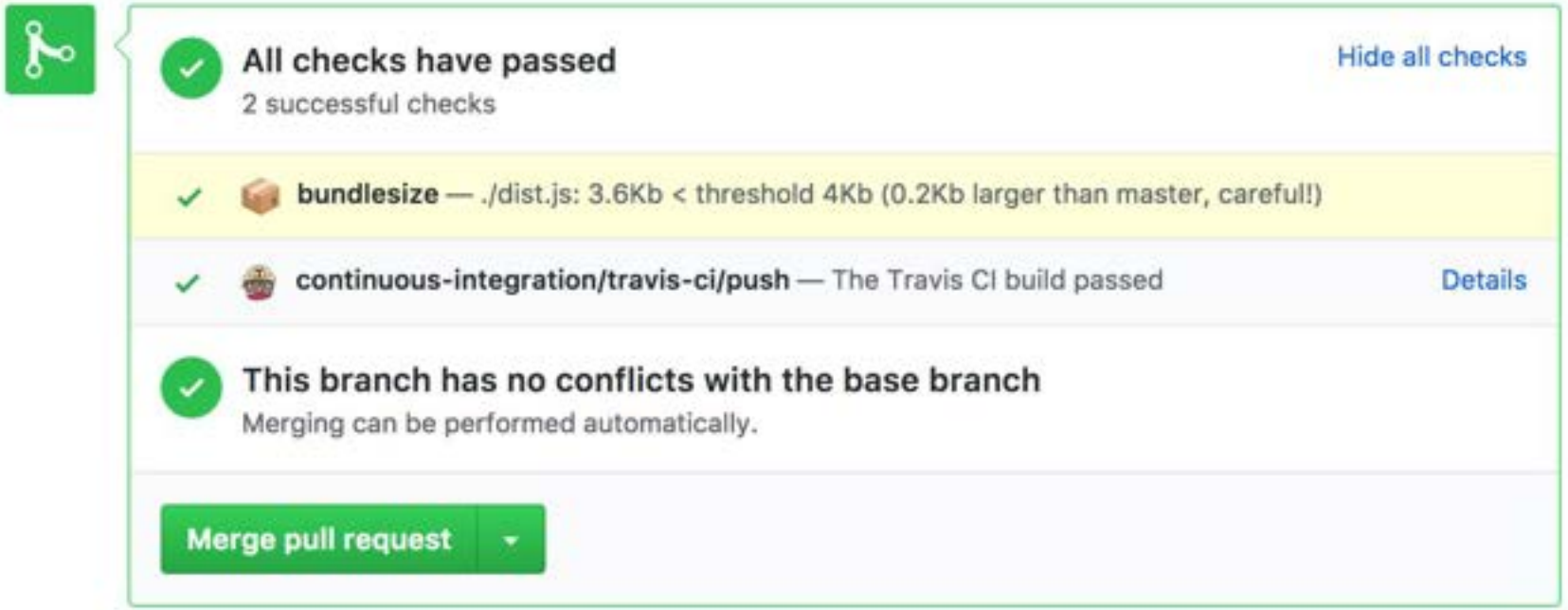
github.com/voxmedia/lightbike

Lightbike - an asset budget auditing tool

Example output

```
 ~/Projects $ lightbike ./config-example.json
Testing homepage
.-------------------------------------------------------------------.
|                  www.theverge.com - homepage                      |
|-------------------------------------------------------------------|
| Content Type | Count | Total Size | Budget Size | over/under      |
|--------------|-------|------------|-------------|-----------------|
| HTML         |     1 |     100 kb |      250 kb |     -150 kb     |
| CSS          |     3 |     112 kb |      150 kb |      -38 kb     |
| JS           |    57 |     807 kb |      500 kb |     +307 kb     |
| FONTS        |     1 |       8 kb |      200 kb |     -192 kb     |
| IMAGES       |    68 |    3442 kb |     1000 kb |    +2442 kb     |
'-------------------------------------------------------------------'

Testing homepage-6s
.-------------------------------------------------------------------.
|                 www.theverge.com - homepage-6s                    |
|-------------------------------------------------------------------|
```

github.com/siddharthkp/bundlesize



lighthouse public

> Lighthouse analyzes web apps and web pages, collecting modern performance metrics and insights on developer best practices.

Using the Node CLI

Lighthouse requires Node 6 or later.

Installation:

```
npm install -g lighthouse
# or use yarn:
# yarn global add lighthouse
```

Run it: `lighthouse https://airhorner.com/`

By default, Lighthouse writes the report to an HTML file. You can control the output format by passing flags.

CLI options

```
$ lighthouse --help
```

webpagetest public

WebPageTest API Wrapper for NodeJS

build passing npm v0.3.5 downloads 15k/month dependencies up to date

WebPageTest API Wrapper is a **NPM** package that wraps **WebPageTest** API for **NodeJS** as a module and a command-line tool.

Getting started

```
$ npm install webpagetest -g
```

Basics

Command line

```
$ webpagetest test https://twitter.com/marcelduran
```

Navigating the Critical Path

- Performance is UX
- Measure, improve, measure
- Optimize for the network
- Optimize for the critical rendering path
- Don't chase metrics: optimize for your users

Chris Nguyen
@**uncompiled**