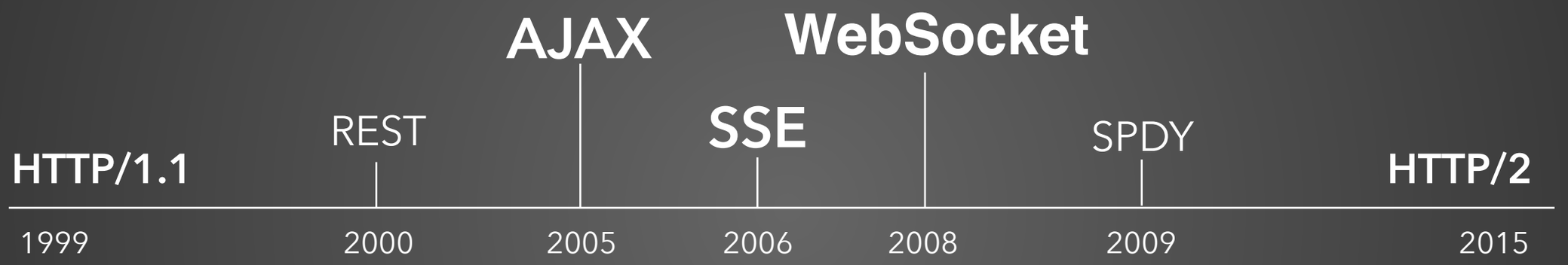
Owning Web Performance With PhantomJS 2

@wesleyhales

How long does it take the page to load?

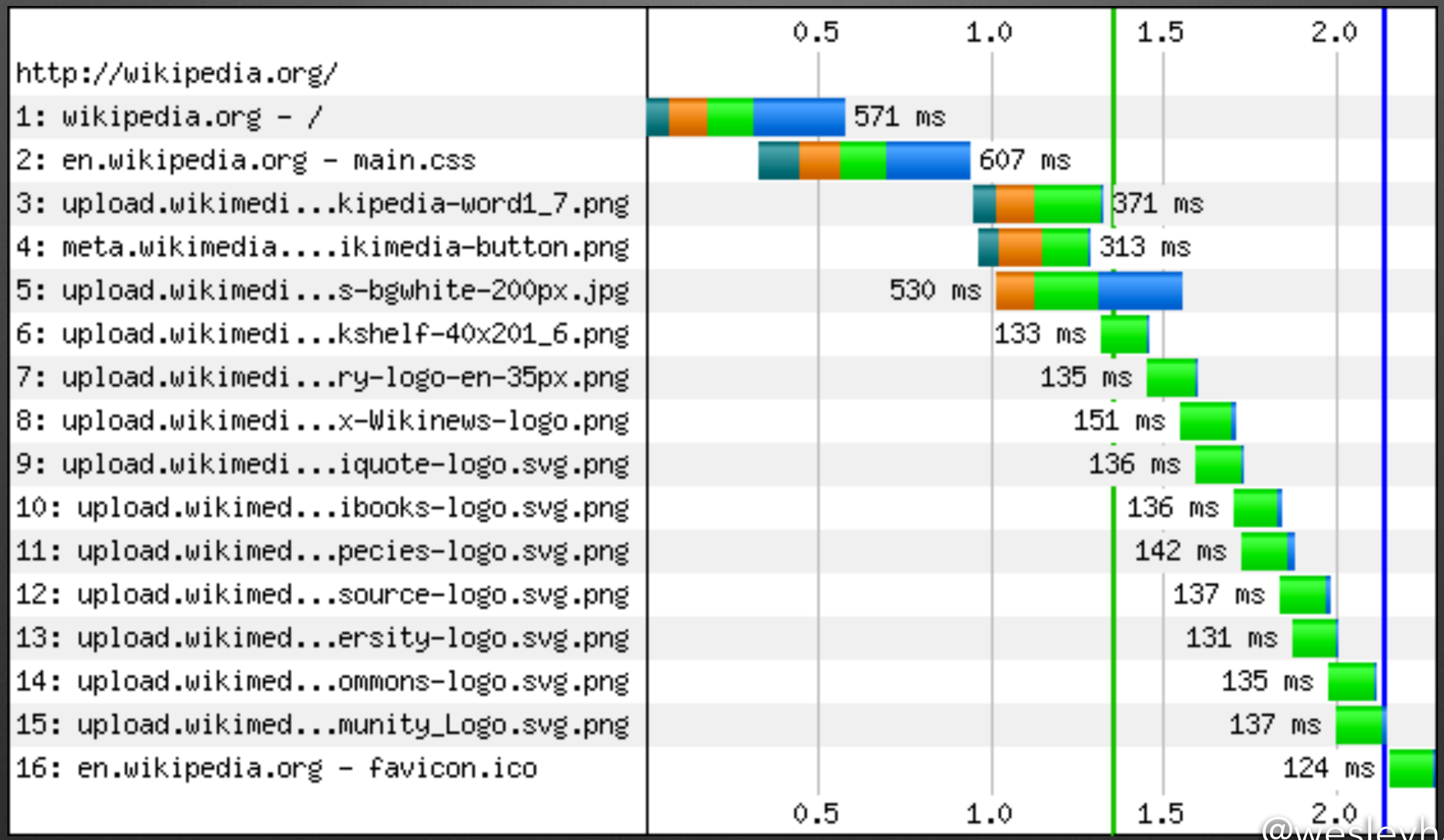
@wesleyhales

Delivering HTML, CSS, and JavaScript to the browser



@wesleyhales

Loading everything and presenting something to the user

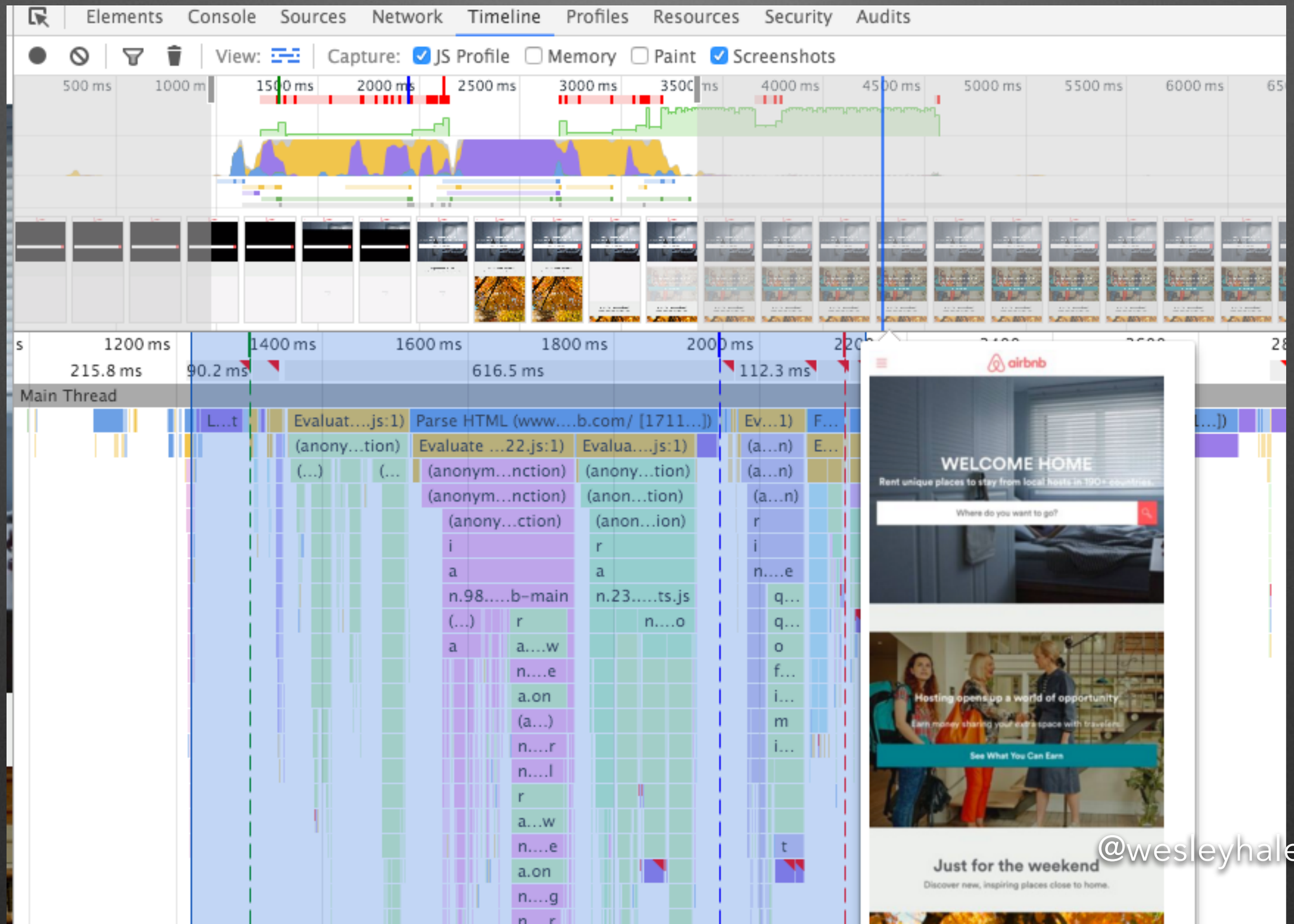


@wesleyhales

Web applications have become more complex...

@wesleyhales

Like this…

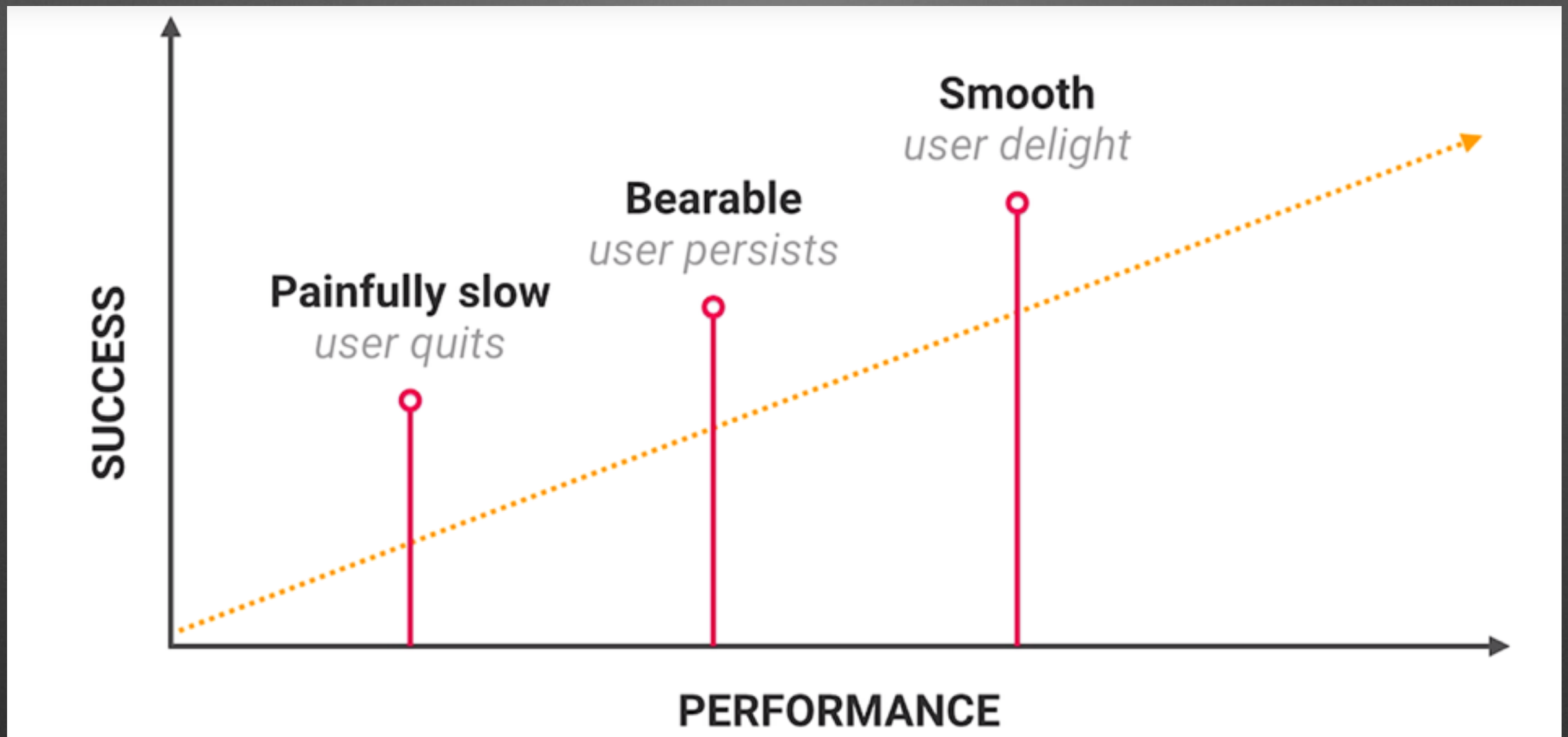

Elements
Console
Sources
Network
Timeline
Profiles
Resources
Security
Audits
View:
Capture:
JS Profile
Memory
Paint
Screenshots
500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
3500 ms
4000 ms
4500 ms
5000 ms
5500 ms
6000 ms
1200 ms
1400 ms
1600 ms
1800 ms
2000 ms
215.8 ms
90.2 ms
616.5 ms
112.3 ms
Main Thread
Evaluat….js:1)
Parse HTML (www….b.com/ [1711…])
(anony…tion)
Evaluate …22.js:1)
Evalua….js:1)
(anonym…nction)
(anon…tion)
(anony…ction)
(anon…ion)
n.98…..b-main
n.23…..ts.js
airbnb
WELCOME HOME
Rent unique places to stay from local hosts in 190+ countries
Where do you want to go?
Hosting opens up a world of opportunity
Earn money sharing your extra space with travelers
See What You Can Earn
Just for the weekend
Discover new, inspiring places close to home.
@wesleyhales

Understanding the basics

@wesleyhales

Delay	User reaction
0 - 100ms	Instant
100 - 300ms	Slight perceptible delay
300 - 1,000ms	Task focus, perceptible delay
1,000+ ms	Mental context switch
10,000+ ms	I'll come back later…

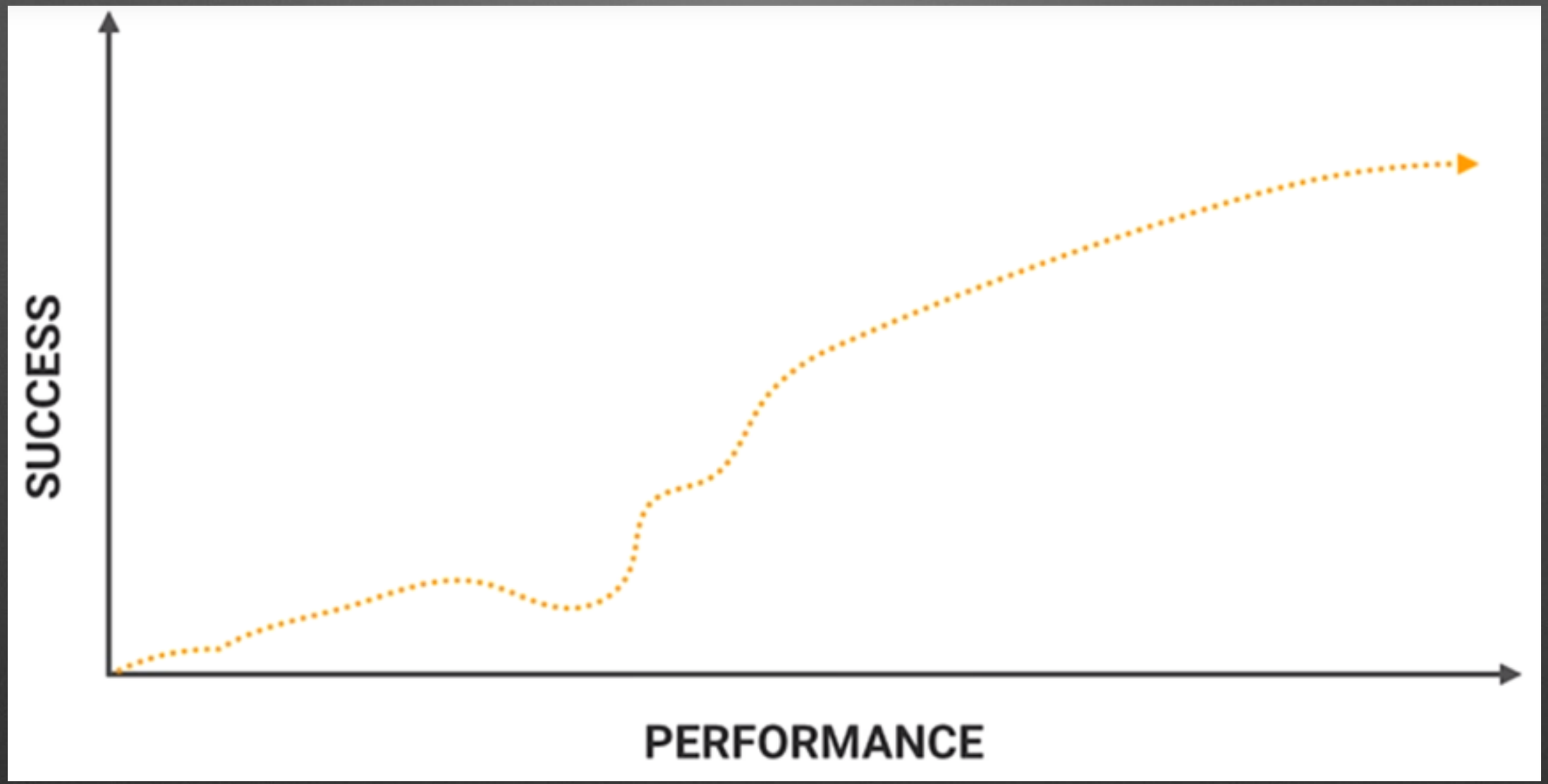
Fast is good… Faster is better



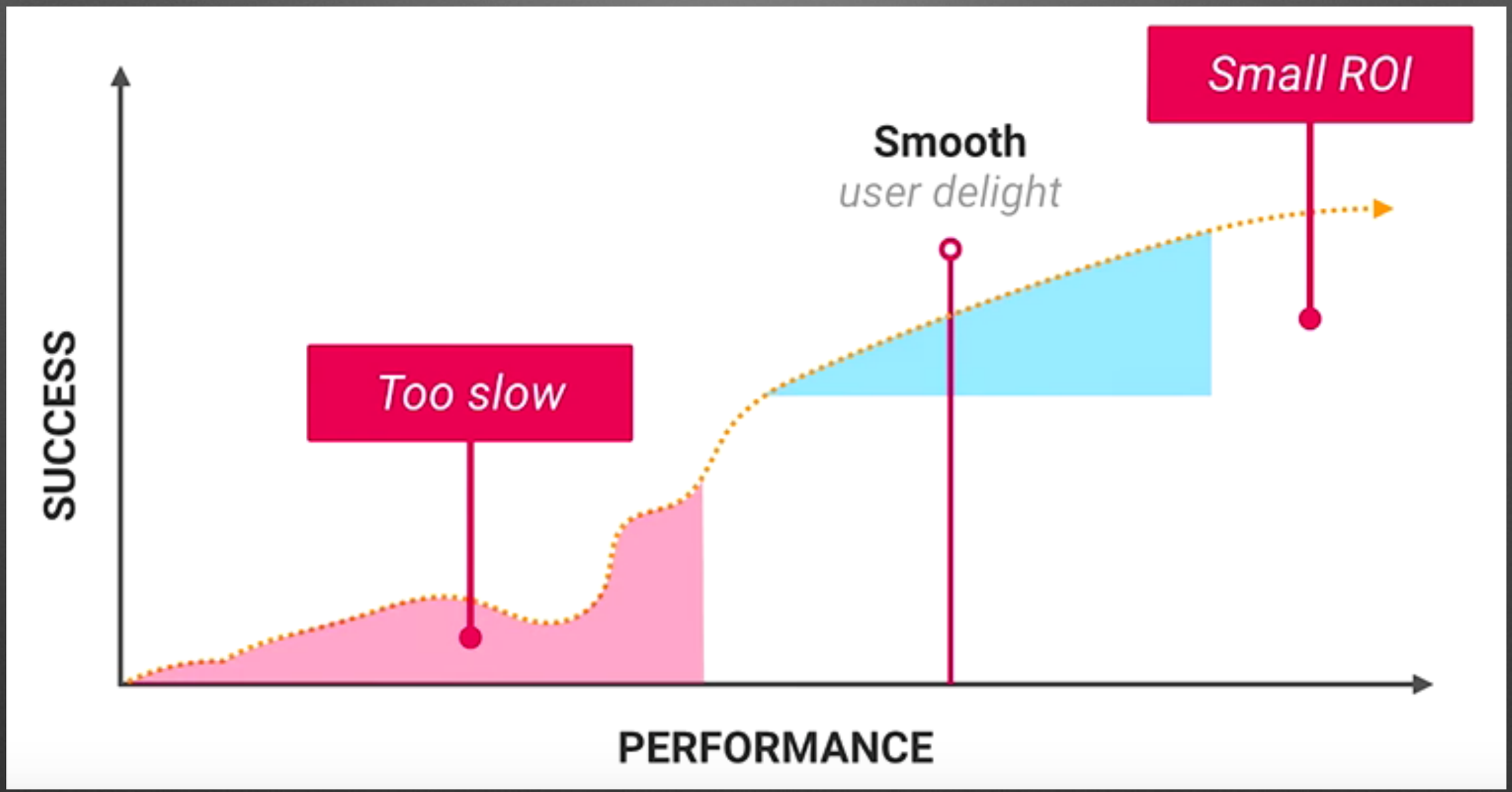
Google Dev Summit

@wesleyhales

Performance Reality



@wesleyhales



@wesleyhales

Response

Animation

Idle

Load

Google Dev Summit

@wesleyhales

RAIL

Response	Animation	Idle	Load
Tap to paint in **less than 100ms**	Each frame completes in **less than 16ms**	Use idle time to proactively schedule work	Satisfy the **Response** goals during load.
	Drag to paint in **less than 16ms**	Complete that work **in 50ms chunks**	Get first meaningful paint in **1,000ms**

Google Dev Summit

@wesleyhales

How do we measure web performance today?

@wesleyhales

Date.now() || Date.getTime()

- When did JavaScript execution begin?
  - <HEAD> vs. </BODY>
  - Get the current time
  - Profit
  - Example: Simple.html

However…

- JavaScript time is skewed by adjustments to the system clock
- NTP adjustments, leap seconds, user configuration changes, and so on can cause time inaccuracies.
- It can’t provide any data regarding the server, network, and so on.

Demo: Basic Loading and Blocking

all-old.html

@wesleyhales

DomContentLoaded

```
document.addEventListener('DOMContentLoaded', function () {
    console.log('DOMContentLoaded', new Date().getTime());
});
```

- Document has been completely loaded and parsed.
- Stylesheets, images, and subframes have not finished loading

load || unload

```
window.addEventListener('load', function (event) {
```

- Every modern framework and app uses it.
- The load event is fired when a resource and its dependent resources have finished loading.
- The load event is easy to measure. Unfortunately, it isn't a very good indicator of the actual end-user experience.


```
$( window ).load(function() {
    // Run code
});
```

readyState

- "loading" while the document is loading
- "interactive" once it is finished parsing (but still loading sub-resources)
- "complete" once it has loaded

```
document.onreadystatechange = function () {
if (document.readyState == "interactive") {
      ...
   }else if (document.readyState == "loading") {
      ...
   }else if (document.readyState == "complete") {
      ...
   }
};
```

```
1 | // A $( document ).ready() block.
2 | $( document ).ready(function() {
3 |     console.log( "ready!" );
4 | });
```

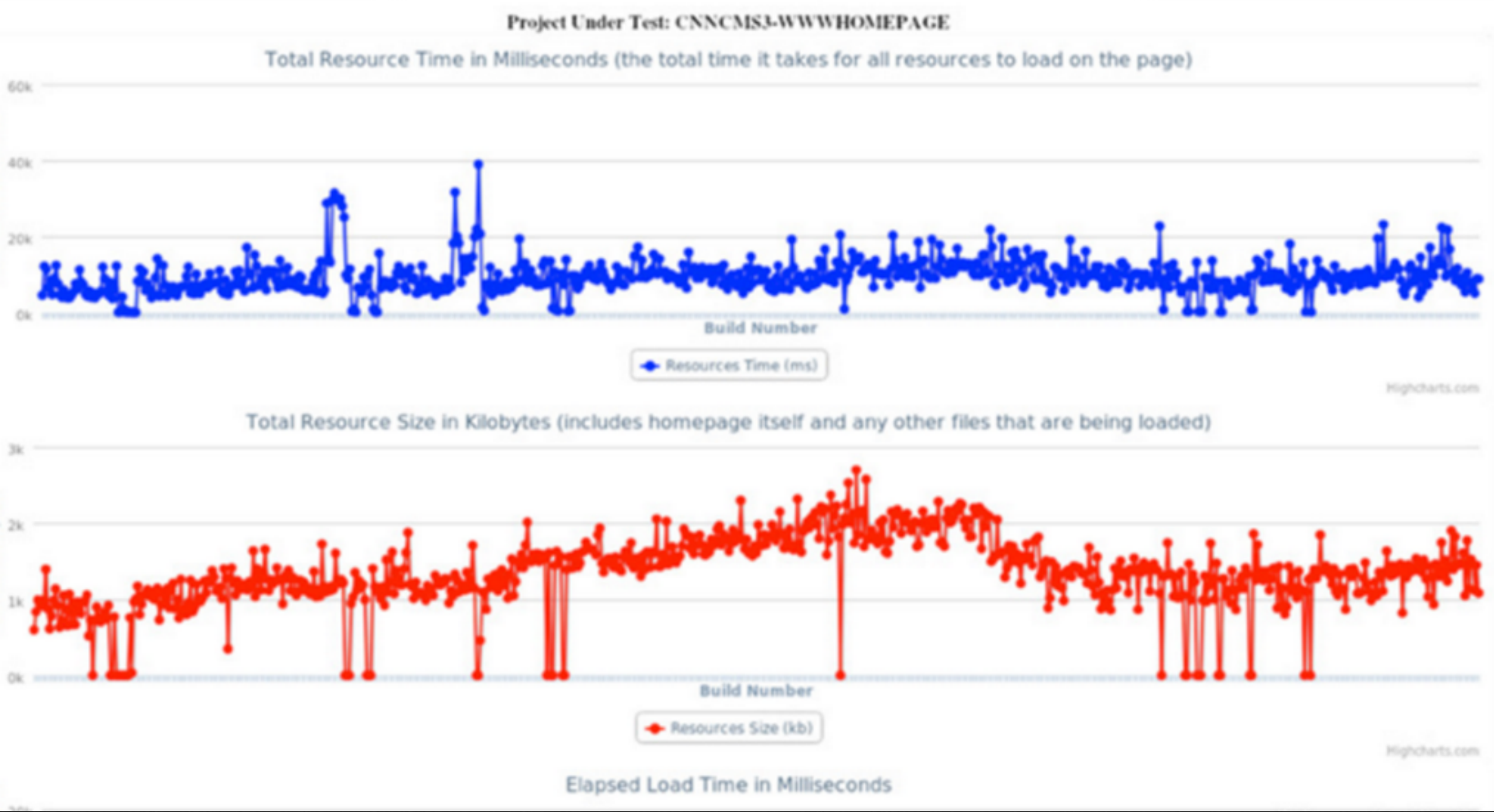
readyState == “complete”

- This will usually include any activity that is triggered by javascript after the main page loads.

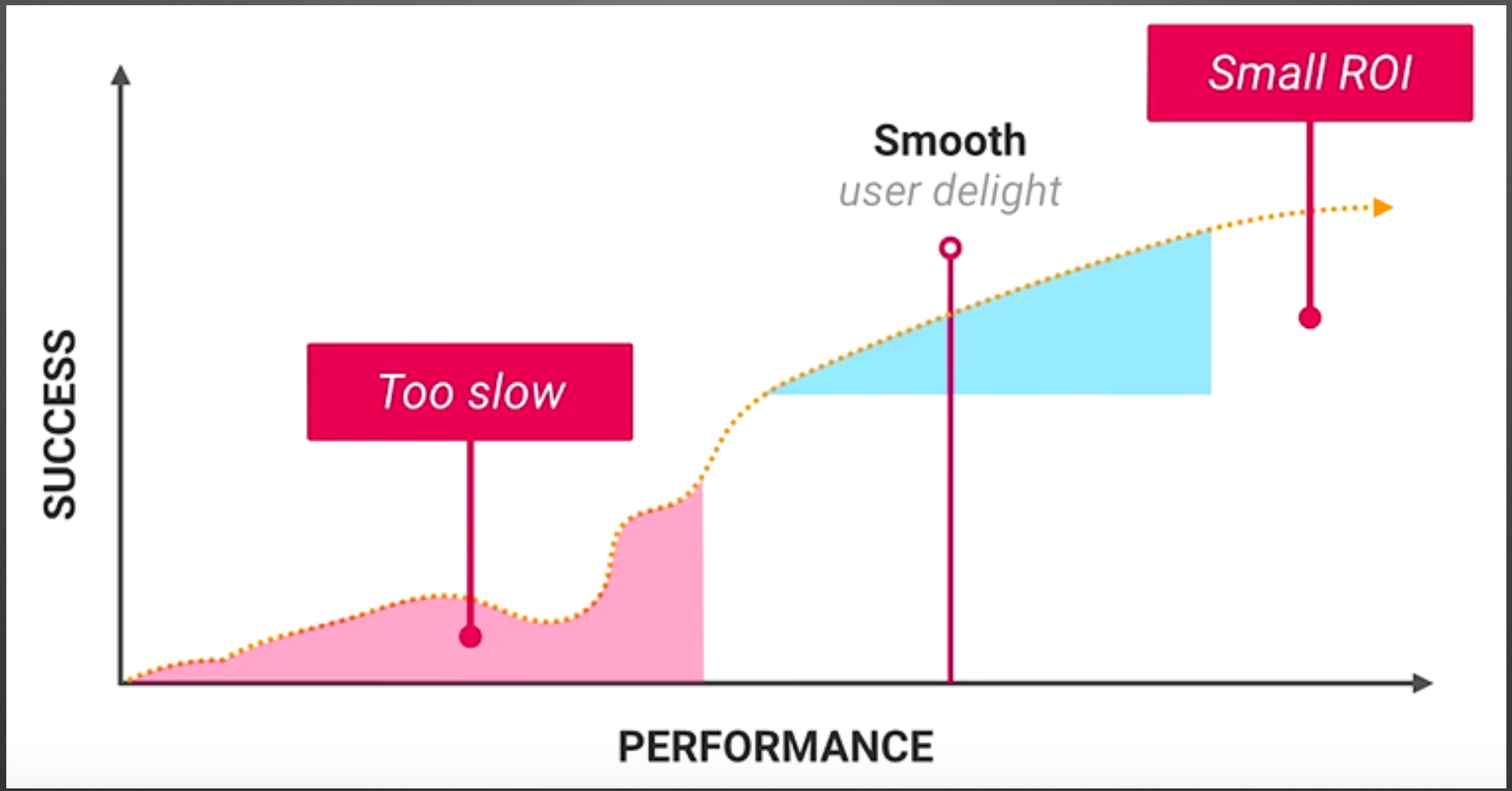
```
document.onreadystatechange = function () {
if (document.readyState == "interactive") {
        ...
    }else if (document.readyState == "loading") {
        ...
    }else if (document.readyState == "complete") {
        ...
    }
};
```

all-old.html (again)

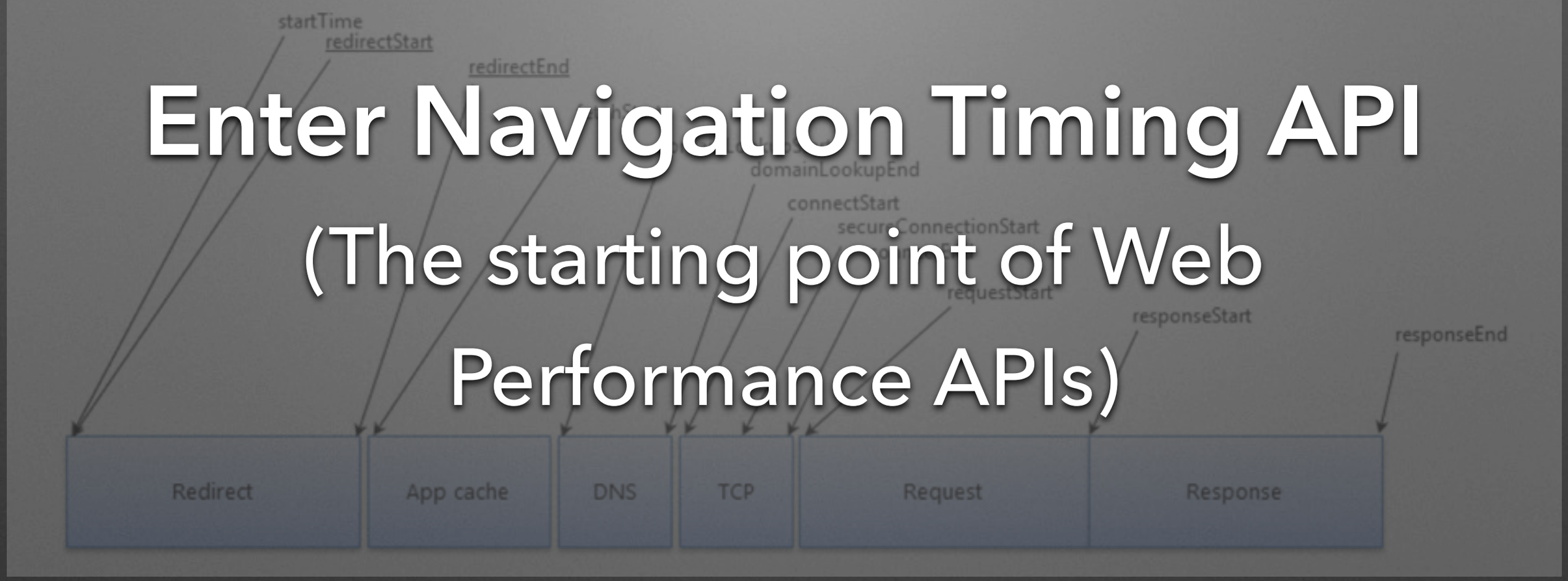
Loadreport.js (2012-2015)



Remember this…



@wesleyhales



Enter Navigation Timing API

(The starting point of Web Performance APIs)

@wesleyhales

Navigation is about how user agents convert the requested HTML, CSS, and JavaScript into rendered pixels, which is one of the most critical steps for users to navigate a document.

@wesleyhales

Navigation Timing API - REC

Global 84.26%

unprefixed: 84.14%

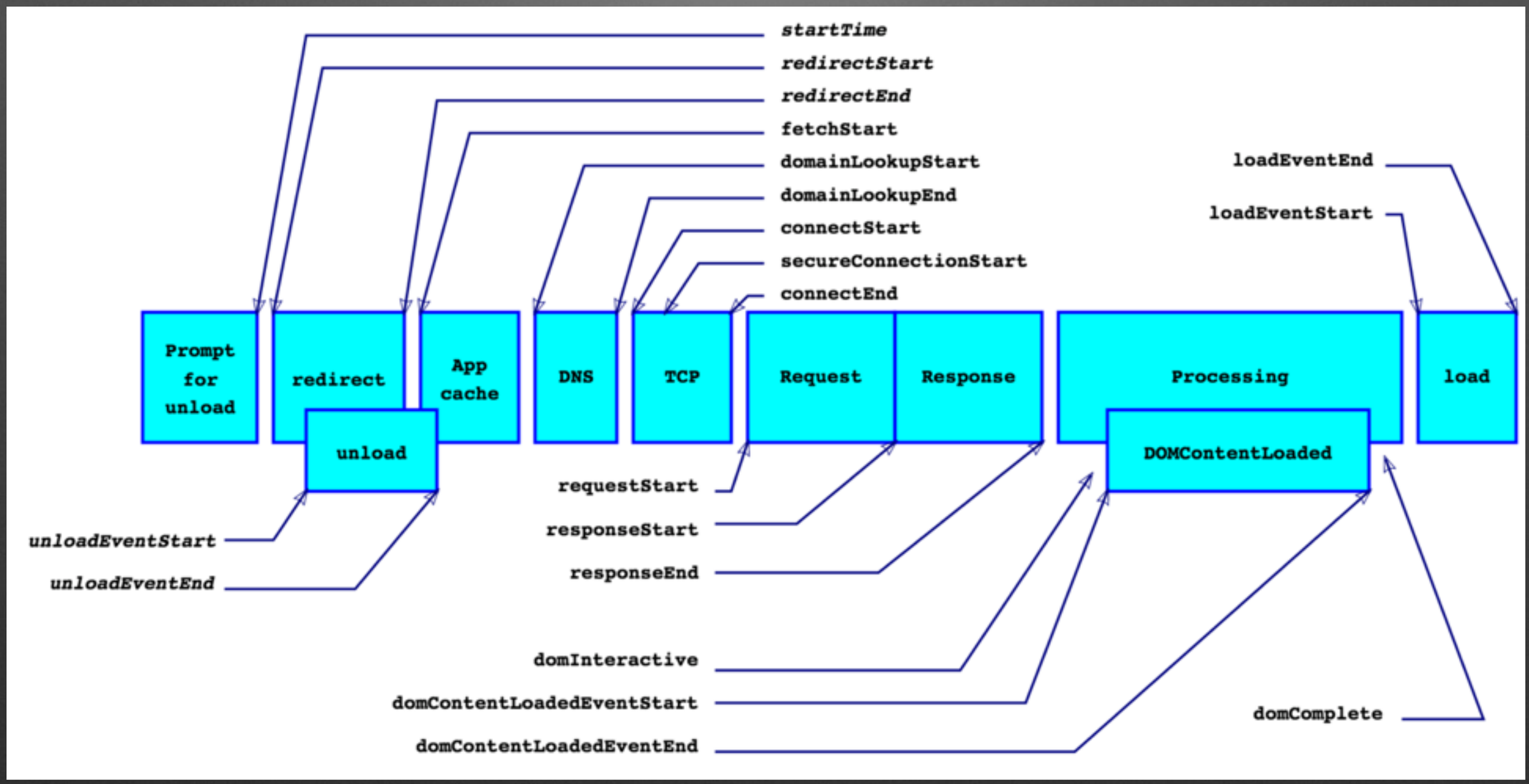
API for accessing timing information related to navigation and elements.

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			31					4.1	
8		38	43					4.3	
9		39	44					4.4	
10		40	45	8		8.4		4.4.4	
11	12	41	46	9	32	9	8	44	45
	13	42	47		33				
		43	48		34				
		44	49						

@wesleyhales

window.performance.timing process model



@wesleyhales

http://www.w3.org/2015/10/webperf-all.html

Web Performance deliverables support as of October 21, 2015

Total subtests: 105

Test	Ch48	Ed20	Ff44	Sf09
http://www.w3.org/2015/10/webperf-overview.html				
window.Performance interface exists.	PASS	PASS	PASS	PASS
window.PerformanceEntry interface exists.	PASS	PASS	PASS	FAIL
window.PerformanceObserver interface exists.	PASS	FAIL	FAIL	FAIL
window.PerformanceObserverEntryList interface exists.	PASS	FAIL	FAIL	FAIL
window.PerformanceResourceTiming interface exists.	PASS	PASS	PASS	FAIL
window.PerformanceNavigationTiming interface exists.	FAIL	PASS	FAIL	FAIL
window.PerformanceMark interface exists.	PASS	PASS	PASS	FAIL
window.PerformanceMeasure interface exists.	PASS	PASS	PASS	FAIL
window.PerformanceFrameTiming interface exists.	FAIL	FAIL	FAIL	FAIL
window.ServerEntry interface exists.	FAIL	FAIL	FAIL	FAIL
window.PerformanceTiming interface exists.	PASS	PASS	PASS	PASS
window.PerformanceNavigation interface exists.	PASS	PASS	PASS	PASS
performance.now is a function.	PASS	PASS	PASS	PASS
performance.translateTime is a function.	FAIL	FAIL	FAIL	FAIL

@wesleyhales

HRT

performance.now()

- All measurements are at microsecond precision.
- Does not depend on system clock
- The idea of High Resolution Time is to provide a monotonic, uniformly increasing timestamp suitable for interval measurements
- Example: perf.now.html

Demo: window.performance

simple-new.html

@wesleyhales

PhantomJS 2

- Released January 2015
- Headless Web Browser
- Based on WebKit

@wesleyhales

PhantomJS 2 Feature Detect

input{}
- autocomplete: true
- autofocus: true
- list: true
- placeholder: true
- max: true
- min: true
- multiple: true
- pattern: true
- required: true
- step: true

inputtypes{}
- search: true
- tel: true
- url: true
- email: true
- datetime: false
- date: false
- month: false
- week: false
- time: false
- datetime-local: false
- number: false
- range: true
- color: true

adownload: true
ambientlight: false
applicationcache: true
audio: false
audioloop: false
audiopreload: false
webaudio: false
batteryapi: false
battery-api: false

csscolumns{}
- width: true
- span: true
- fill: false
- gap: true
- rule: true
- rulecolor: true
- rulestyle: true
- rulewidth: true
- breakbefore: true
- breakafter: true
- breakinside: true

cubicbezierrange: true
displayrunin: true
display-runin: true
displaytable: true
display-table: true
ellipsis: true
cssescape: false
cssexunit: true
supports: false
cssfilters: true
flexbox: true
flexboxlegacy: true
flexboxtweener: false
flexwrap: true
fontface: true
generatedcontent: true
cssgradients: true
hsla: true
cssinvalid: true
lastchild: true
cssmask: true
mediaqueries: true

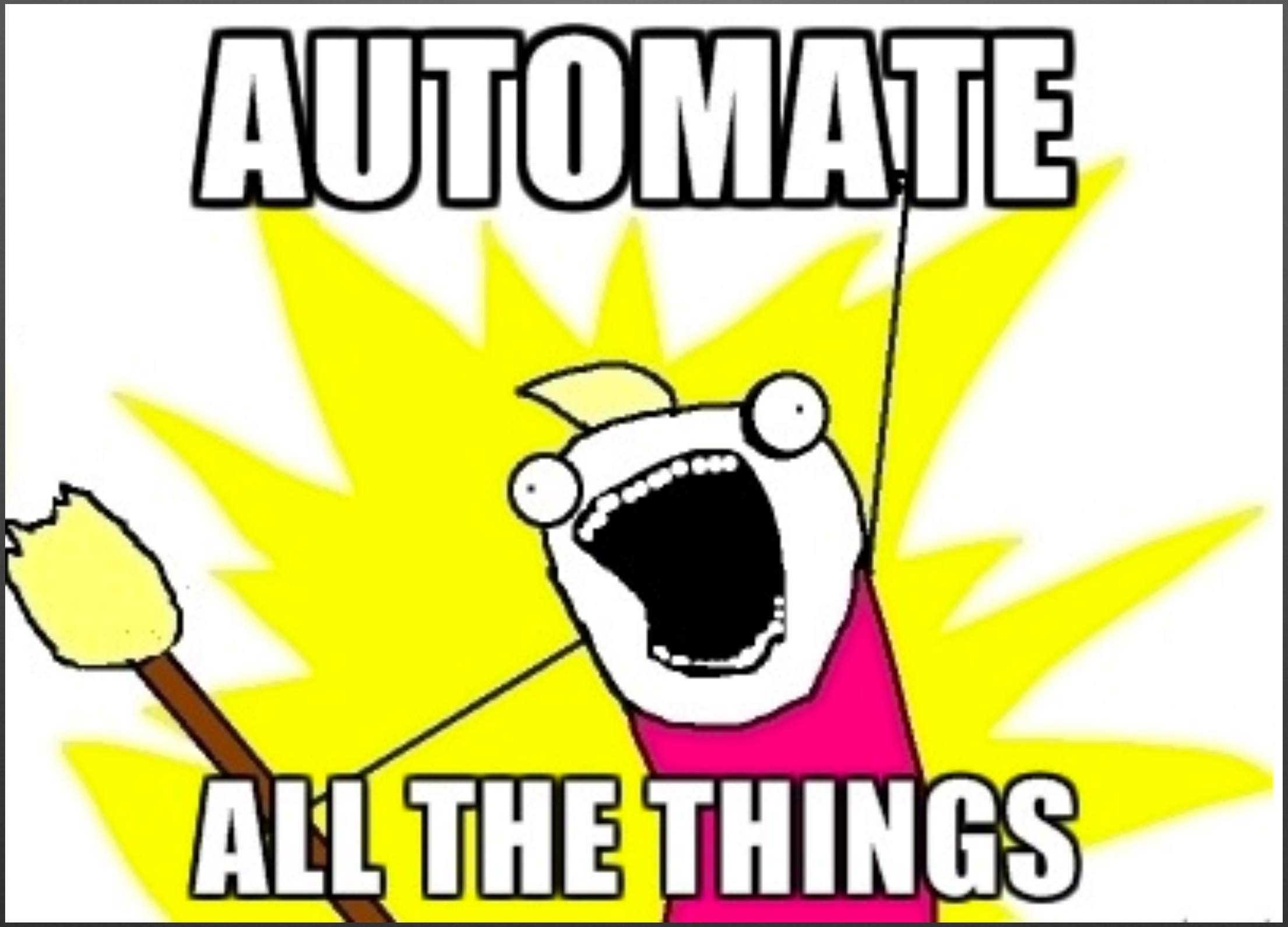
dataset: true
documentfragment: true
hidden: true
microdata: false
mutationobserver: true
draganddrop: true
datalistelem: true
details: true
outputelem: true
picture: false
progressbar: true
meter: true
ruby: true
template: false
texttrackapi: false
track: false
unknownelements: true
emoji: true
es5array: true
es5date: true
es5function: true
es5object: true
strictmode: true
es5string: true
json: true
es5syntax: true
es5undefined: true
es5: true
es6array: false
contains: false
generators: false
es6math: false
es6number: false
es6object: false

intl: false
olreversed: true
mathml: false
beacon: false
lowbandwidth: false
eventsource: true
xhrresponsetype: true
xhrresponsetypearraybuffer: true
xhrresponsetypeblob: true
xhrresponsetypedocument: true
xhrresponsetypejson: false
xhrresponsetypetext: true
xhr2: true
notification: true
pagevisibility: true
performance: true
pointerevents: false
pointerlock: false
postmessage: true
proximity: false
queryselector: true
quotamanagement: false
requestanimationframe: true
raf: true
scriptasync: true
scriptdefer: true
serviceworker: false
speechrecognition: false
speechsynthesis: false
localstorage: true
sessionstorage: true
websqldatabase: true

Demo: Basic PhantomJS Scripts

http://phantomjs.org/examples/

@wesleyhales



@wesleyhales

Speedgun.js

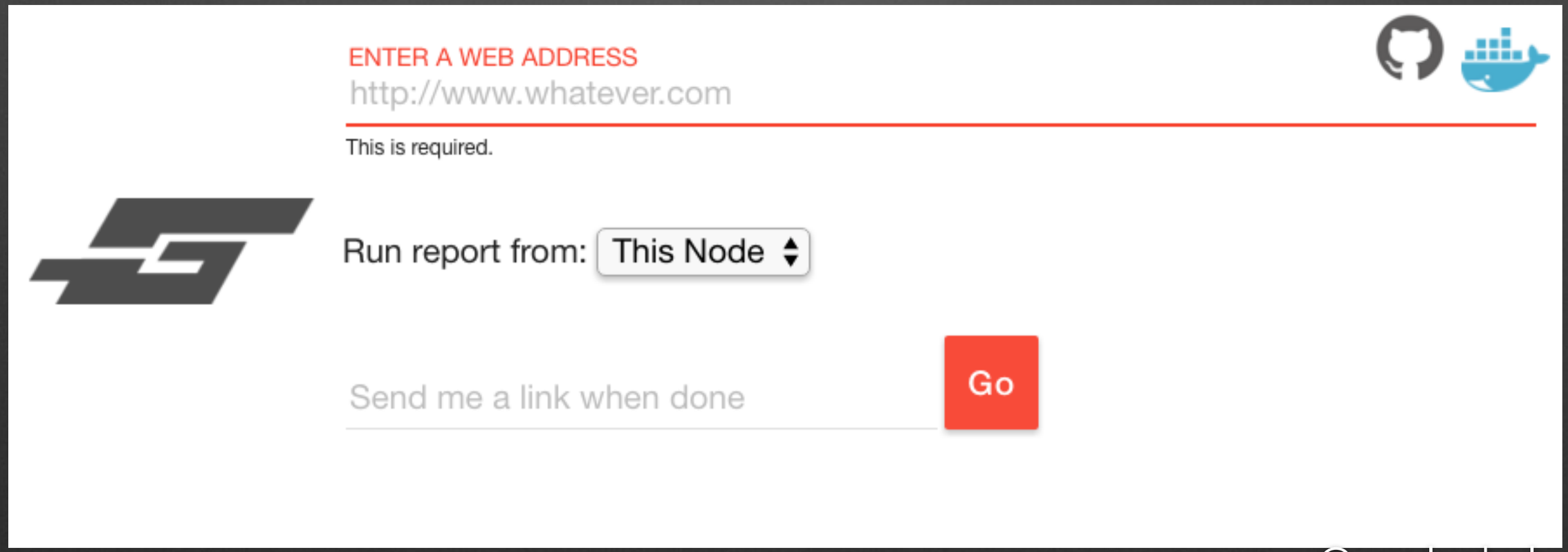
- Rewrite of loadreport.js
- Leverages all implemented PhantomJS 2 Navigation Timing APIs
- (shims resource timing)

```
[~/dev/connectJS 2015] ➔ phantomjs speedgun.js
You must supply a URL
  Usage: phantomjs --config=core/pconfig.json core/speedgun.js [options] url
  Options:
    -h, --help                  This help
    -t, --task                  Choose task (performance) [performance]
    -f, --format                How much information (detailed|simple) [simple]
    -o, --output                Output format (json|csv|junit|post) [json]
    -ua, --userAgent            Set the user agent (chrome|android|iphone) [chrome]
    -v, --version               Not implemented yet
    -u, --uuid                  only used for server side run in speedgun.io
    --verbose                   Turn on verbose logging
    --wipe                      Wipe the file instead of appending to it on each report
    --phantomCacheEnabled       Enable PhantomJS cache
```

Demo: Speedgun.js

@wesleyhales

speedgun.io



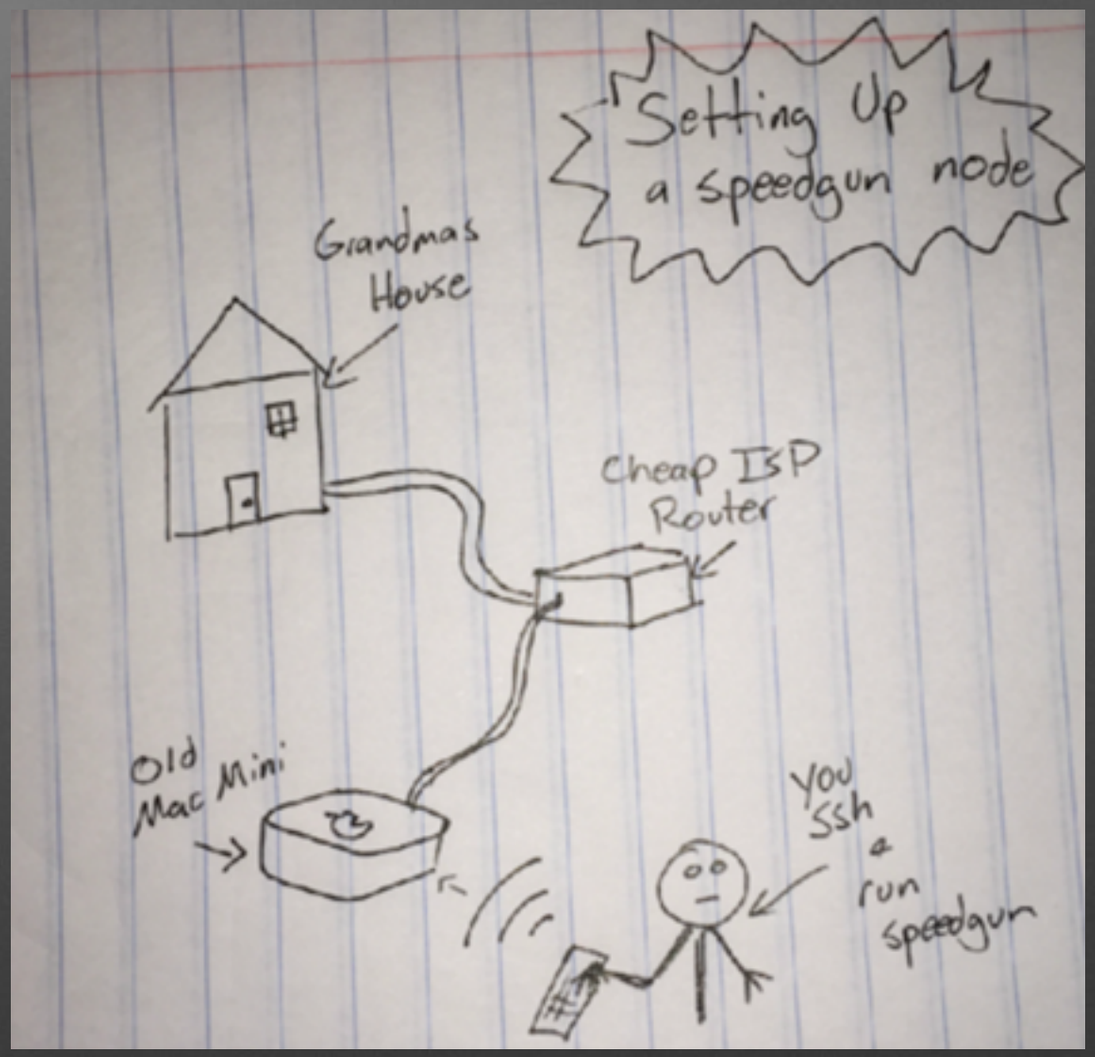
@wesleyhales

Demo: speedgun.io

@wesleyhales

Synthetic RUM

- Use Speedgun.io as centralized server
- All docker nodes send beacon with:
  - Current container CPU and memory usage
- Yes, another demo...



http://wesleyhales.com/blog/2015/04/24/Speedgun/

Synthetic RUM

- The ability to execute a script (PhantomJS in this case) in a controlled, one at a time manner.

- A REST api that allows master/slave communication and is publicly accessible.

- A beacon that sends it’s availability to a centralized(parent) server.

- Data storage of reports in a db that treats JSON as a first class citizen.

http://wesleyhales.com/blog/2015/04/24/Speedgun/

JSON Data Storage

demo on querying report data

W3C Performance Specs

[FRAME-TIMING]
Michael Blain; Ilya Grigorik. *Frame Timing*. 24 November 2015. W3C Working
[HR-TIME-2]
Ilya Grigorik; James Simonsen; Jatinder Mann. *High Resolution Time Level 2*.
[NAVIGATION-TIMING]
Zhiheng Wang. *Navigation Timing*. 17 December 2012. W3C Recommendatio
[NAVIGATION-TIMING-2]
Tobin Titus; Jatinder Mann; Arvind Jain. *Navigation Timing Level 2*. 25 Novem
[PERFORMANCE-TIMELINE-2]
Ilya Grigorik. *Performance Timeline Level 2*. 25 November 2015. W3C Workin
[RESOURCE-TIMING]
Arvind Jain; Todd Reifsteck; Jatinder Mann; Zhiheng Wang; Anderson Quach.
[SERVER-TIMING]
Ilya Grigorik. *Server Timing*. 17 July 2015. W3C Working Draft. URL: http://ww
[USER-TIMING]
Jatinder Mann; Zhiheng Wang; Anderson Quach. *User Timing*. 12 December

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

Thanks!!

- speedgun.io (github)
- Navigation Timing API
- Navigation Timing 2
- Resource Timing API

@wesleyhales