MEASURING THE PERFORMANCE OF SINGLE PAGE APPLICATIONS

nic jansma | SOASTA | nicj.net | @nicj

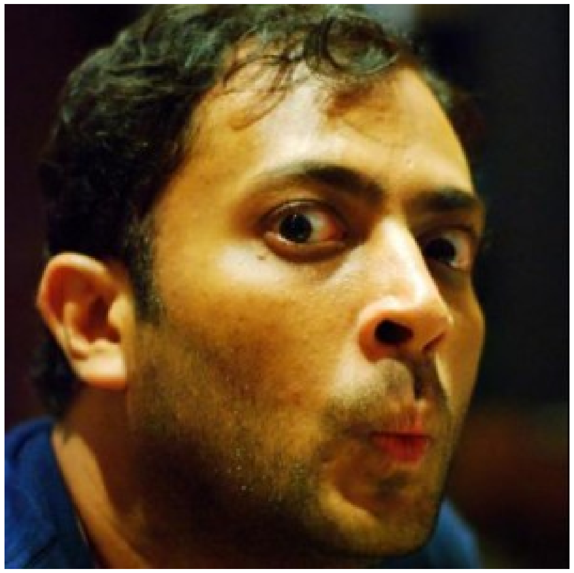
philip tellis | SOASTA | bluesmoon.info | @bluesmoon

SLIDES

slideshare.net/nicjansma/

WHO ARE WE?

Nic Jansma
SOASTA



Philip Tellis
SOASTA

DEFINITIONS

RUM

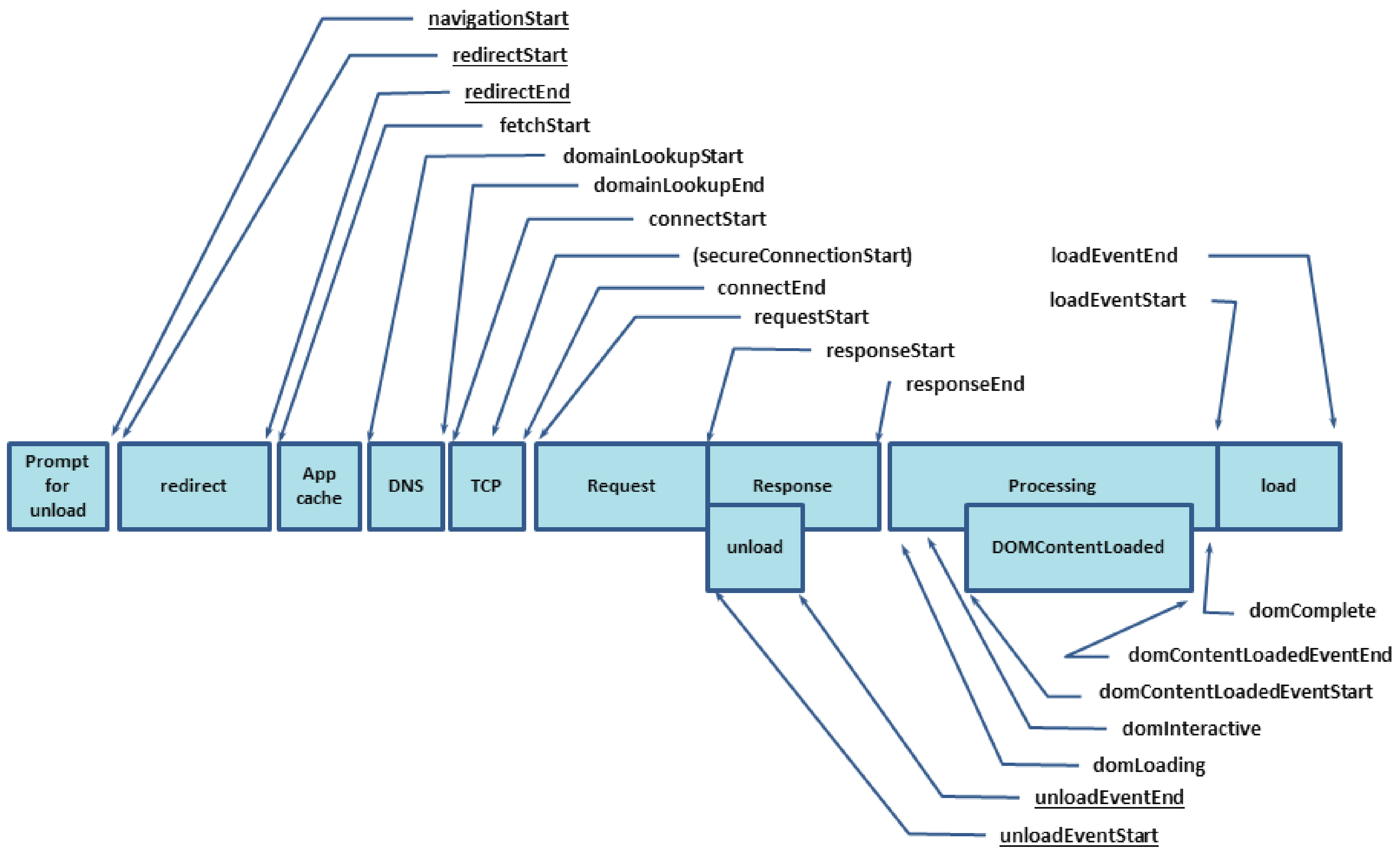
Real User Monitoring

- Gathering performance metrics from real user experiences
- Versus Synthetic Monitoring, with emulated users in a controlled environment

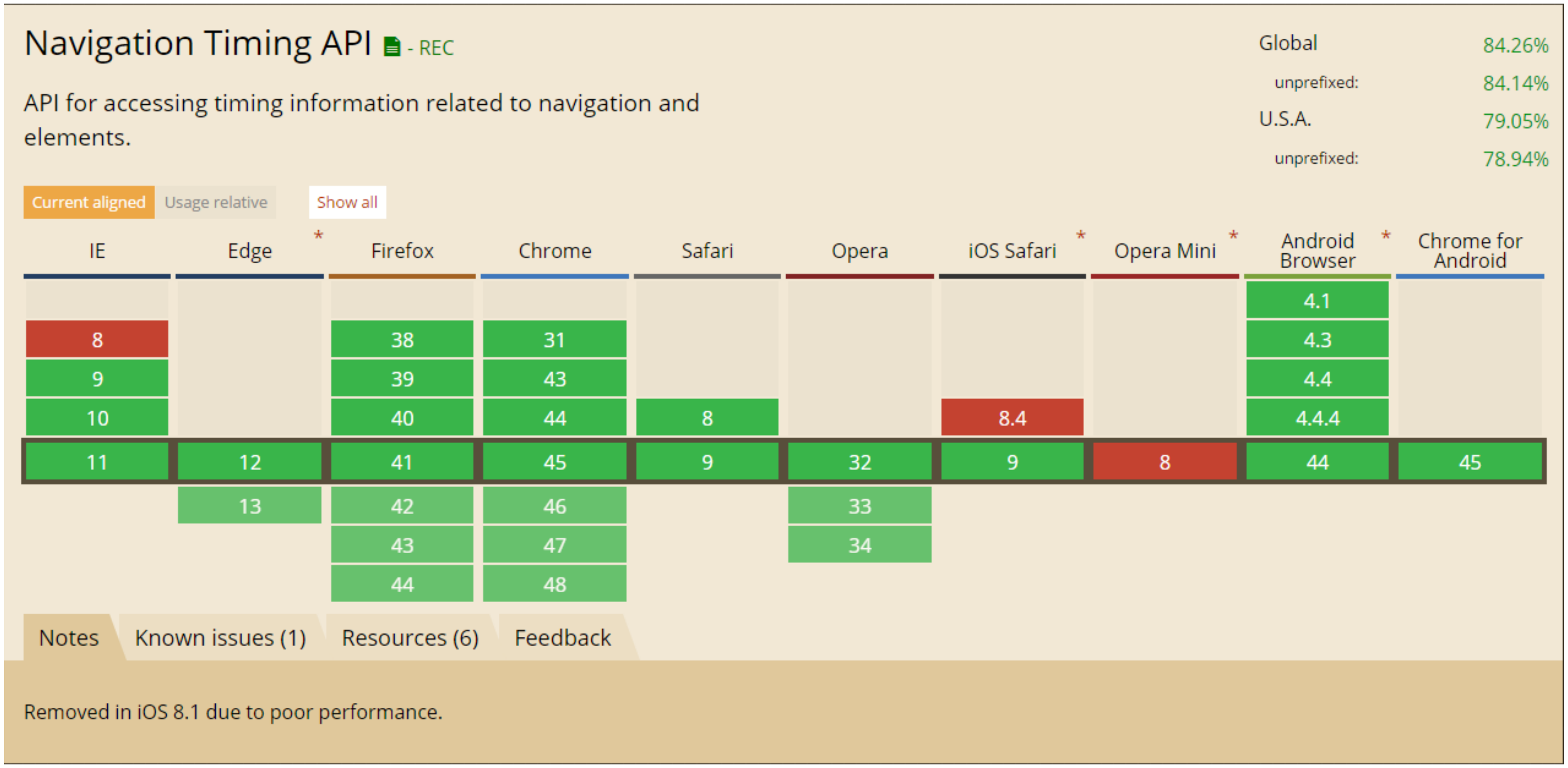
RUM: HOW IT'S DONE

- JavaScript measures the browser's events and performance interfaces
  - Listen for `readyState` changes and the `onload` event
  - Measure DNS, TCP, SSL, Request and Response times from NavigationTiming and user measurements from UserTiming (if available)
  - Gather User Agent characteristics (Version, Screen Size, etc)
- Beacon this data back to the cloud for analytics

NAVIGATIONTIMING



NAVIGATIONTIMING



Navigation Timing API - REC

API for accessing timing information related to navigation and elements.

Global	84.26%
unprefixed:	84.14%
U.S.A.	79.05%
unprefixed:	78.94%

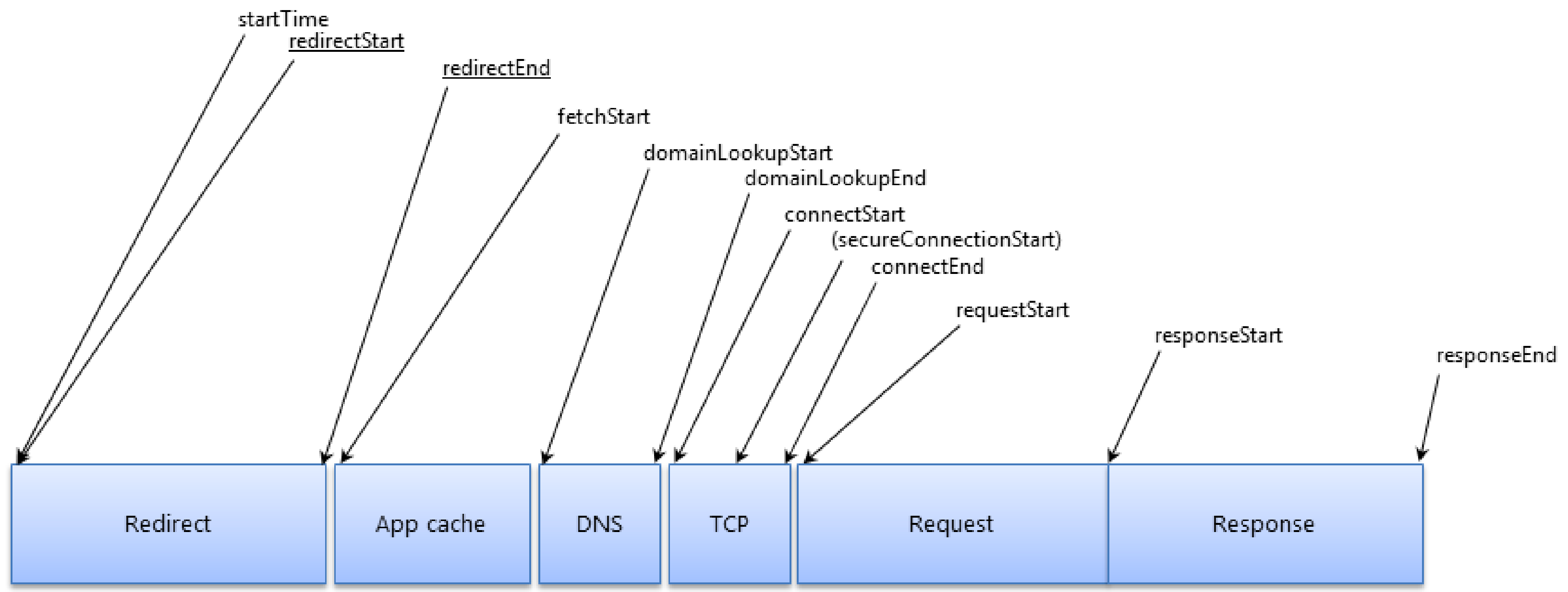
Current aligned · Usage relative · Show all

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

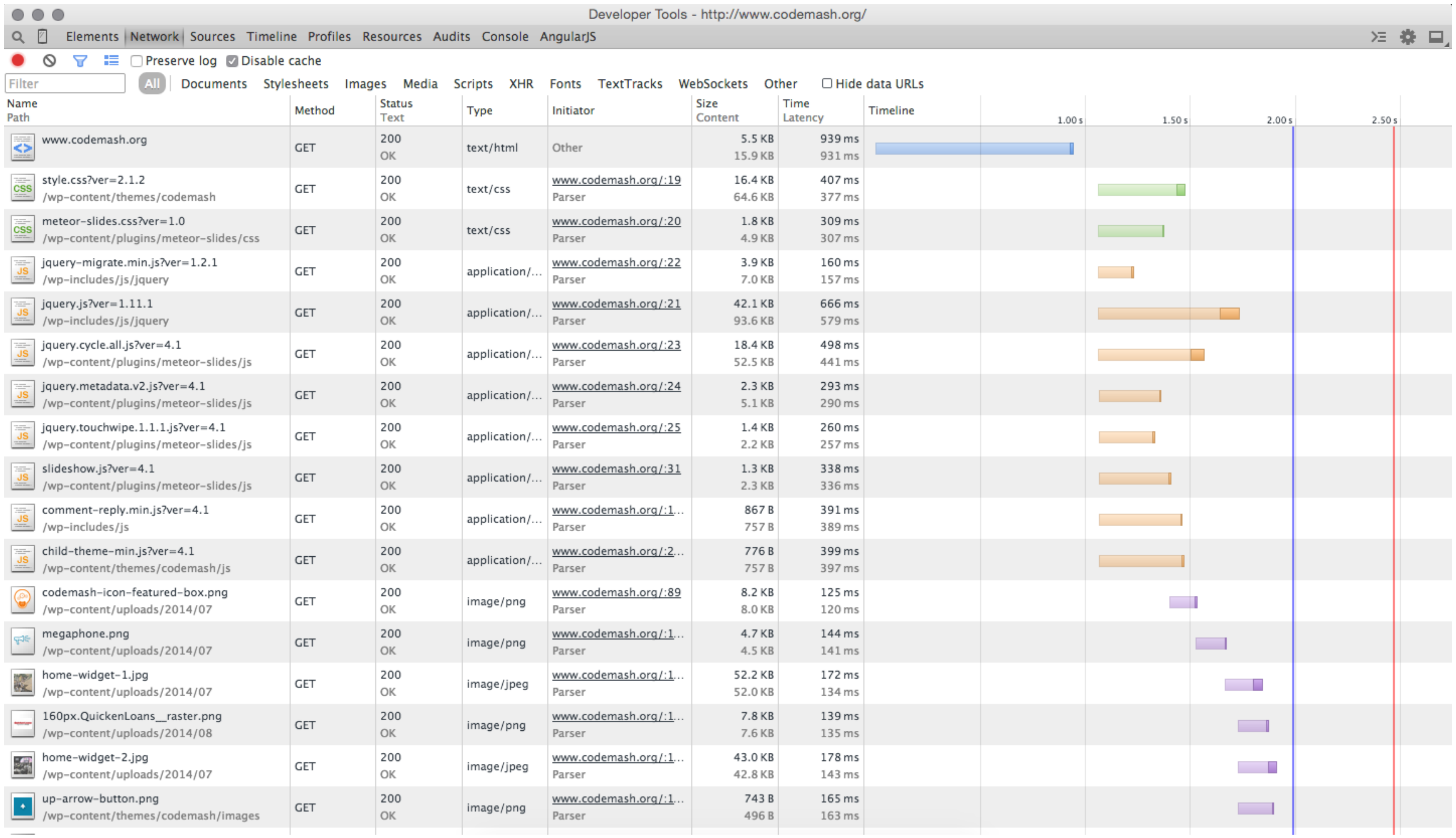
Notes · Known issues (1) · Resources (6) · Feedback

Removed in iOS 8.1 due to poor performance.

RESOURCETIMING

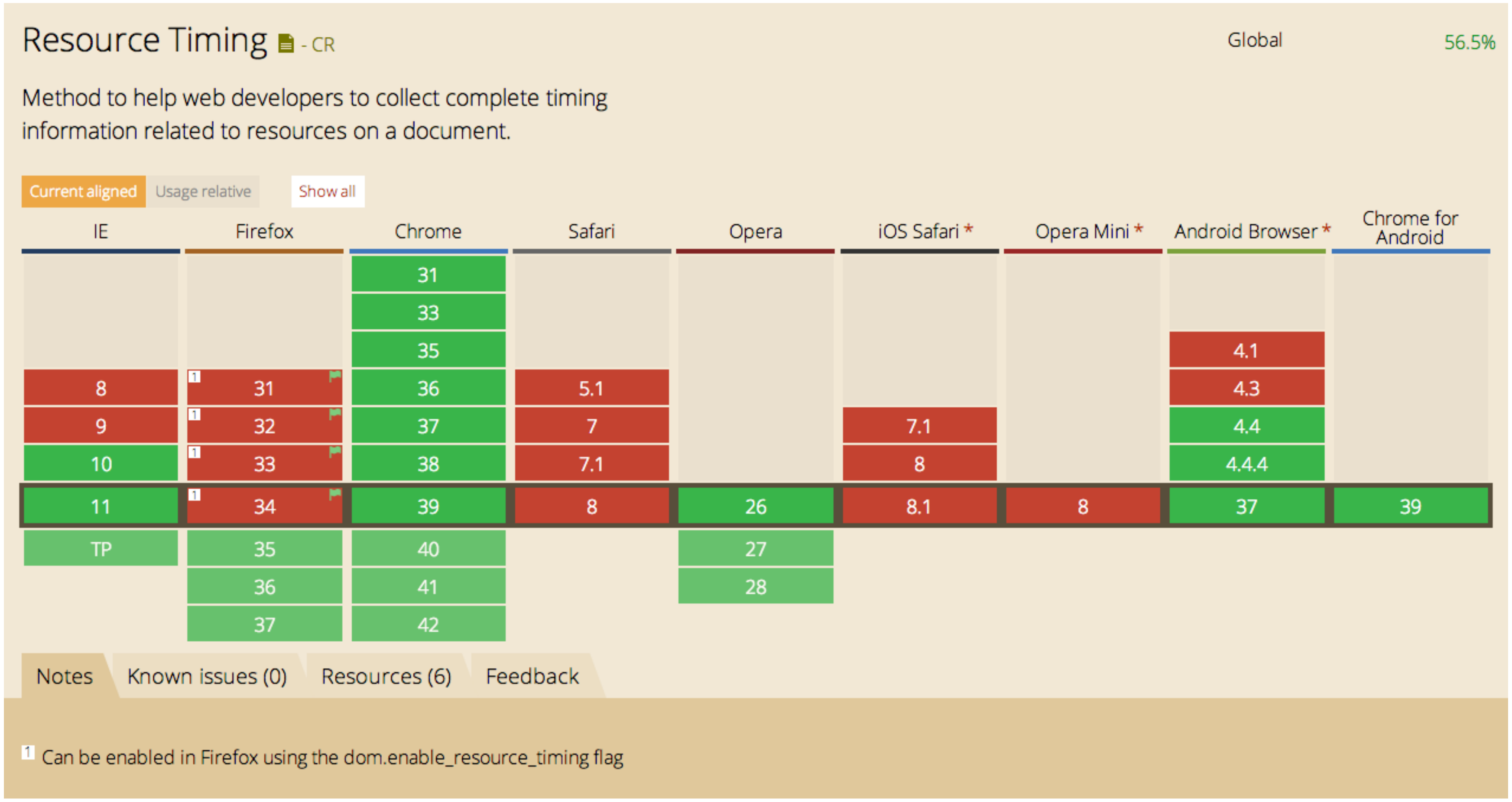


RESOURCETIMING



Name / Path	Method	Status / Text	Type	Initiator	Size / Content	Time / Latency	Timeline
www.codemash.org	GET	200 / OK	text/html	Other	5.5 KB / 15.9 KB	939 ms / 931 ms	
style.css?ver=2.1.2 /wp-content/themes/codemash	GET	200 / OK	text/css	www.codemash.org/:19 Parser	16.4 KB / 64.6 KB	407 ms / 377 ms	
meteor-slides.css?ver=1.0 /wp-content/plugins/meteor-slides/css	GET	200 / OK	text/css	www.codemash.org/:20 Parser	1.8 KB / 4.9 KB	309 ms / 307 ms	
jquery-migrate.min.js?ver=1.2.1 /wp-includes/js/jquery	GET	200 / OK	application/...	www.codemash.org/:22 Parser	3.9 KB / 7.0 KB	160 ms / 157 ms	
jquery.js?ver=1.11.1 /wp-includes/js/jquery	GET	200 / OK	application/...	www.codemash.org/:21 Parser	42.1 KB / 93.6 KB	666 ms / 579 ms	
jquery.cycle.all.js?ver=4.1 /wp-content/plugins/meteor-slides/js	GET	200 / OK	application/...	www.codemash.org/:23 Parser	18.4 KB / 52.5 KB	498 ms / 441 ms	
jquery.metadata.v2.js?ver=4.1 /wp-content/plugins/meteor-slides/js	GET	200 / OK	application/...	www.codemash.org/:24 Parser	2.3 KB / 5.1 KB	293 ms / 290 ms	
jquery.touchwipe.1.1.1.js?ver=4.1 /wp-content/plugins/meteor-slides/js	GET	200 / OK	application/...	www.codemash.org/:25 Parser	1.4 KB / 2.2 KB	260 ms / 257 ms	
slideshow.js?ver=4.1 /wp-content/plugins/meteor-slides/js	GET	200 / OK	application/...	www.codemash.org/:31 Parser	1.3 KB / 2.3 KB	338 ms / 336 ms	
comment-reply.min.js?ver=4.1 /wp-includes/js	GET	200 / OK	application/...	www.codemash.org/:1... Parser	867 B / 757 B	391 ms / 389 ms	
child-theme-min.js?ver=4.1 /wp-content/themes/codemash/js	GET	200 / OK	application/...	www.codemash.org/:2... Parser	776 B / 757 B	399 ms / 397 ms	
codemash-icon-featured-box.png /wp-content/uploads/2014/07	GET	200 / OK	image/png	www.codemash.org/:89 Parser	8.2 KB / 8.0 KB	125 ms / 120 ms	
megaphone.png /wp-content/uploads/2014/07	GET	200 / OK	image/png	www.codemash.org/:1... Parser	4.7 KB / 4.5 KB	144 ms / 141 ms	
home-widget-1.jpg /wp-content/uploads/2014/07	GET	200 / OK	image/jpeg	www.codemash.org/:1... Parser	52.2 KB / 52.0 KB	172 ms / 134 ms	
160px.QuickenLoans__raster.png /wp-content/uploads/2014/08	GET	200 / OK	image/png	www.codemash.org/:1... Parser	7.8 KB / 7.6 KB	139 ms / 135 ms	
home-widget-2.jpg /wp-content/uploads/2014/07	GET	200 / OK	image/jpeg	www.codemash.org/:1... Parser	43.0 KB / 42.8 KB	178 ms / 143 ms	
up-arrow-button.png /wp-content/themes/codemash/images	GET	200 / OK	image/png	www.codemash.org/:1... Parser	743 B / 496 B	165 ms / 163 ms	

RESOURCETIMING



Resource Timing - CR

Global 56.5%

Method to help web developers to collect complete timing information related to resources on a document.

Current aligned Usage relative Show all

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		🟩 31						
		🟩 33						
		🟩 35					🟥 4.1	
🟥 8	🟥 [1] 31	🟩 36	🟥 5.1				🟥 4.3	
🟥 9	🟥 [1] 32	🟩 37	🟥 7		🟥 7.1		🟩 4.4	
🟩 10	🟥 [1] 33	🟩 38	🟥 7.1		🟥 8		🟩 4.4.4	
🟩 11	🟥 [1] 34	🟩 39	🟥 8	🟩 26	🟥 8.1	🟥 8	🟩 37	🟩 39
🟩 TP	🟩 35	🟩 40		🟩 27				
	🟩 36	🟩 41		🟩 28				
	🟩 37	🟩 42						

Notes Known issues (0) Resources (6) Feedback

[1] Can be enabled in Firefox using the dom.enable_resource_timing flag

BOOMERANG

- Created by Philip Tellis @ Yahoo
- Gathers performance metrics and characteristics of page load and beacons data to your server (aka RUM)
- Open-source project (with contributions from SOASTA)
- https://github.com/lognormal/boomerang/

SPAS
SINGLE PAGE APPS

- Run on a single page, dynamically bringing in content as necessary
- Built with frameworks like AngularJS, Ember.js, Backbone.js, React, etc.

SPAS

HARD VS. SOFT NAVIGATIONS

- **Hard Navigation**: The first page load, which will include all static HTML, JavaScript, CSS, the SPA framework itself (e.g. `angular.js`), plus showing the initial route
- **Soft Navigation**: Any subsequent route (address bar) change
- Any URL might be loaded via *either* hard or soft navigation

3 CHALLENGES OF MEASURING THE PERFORMANCE OF SPAS

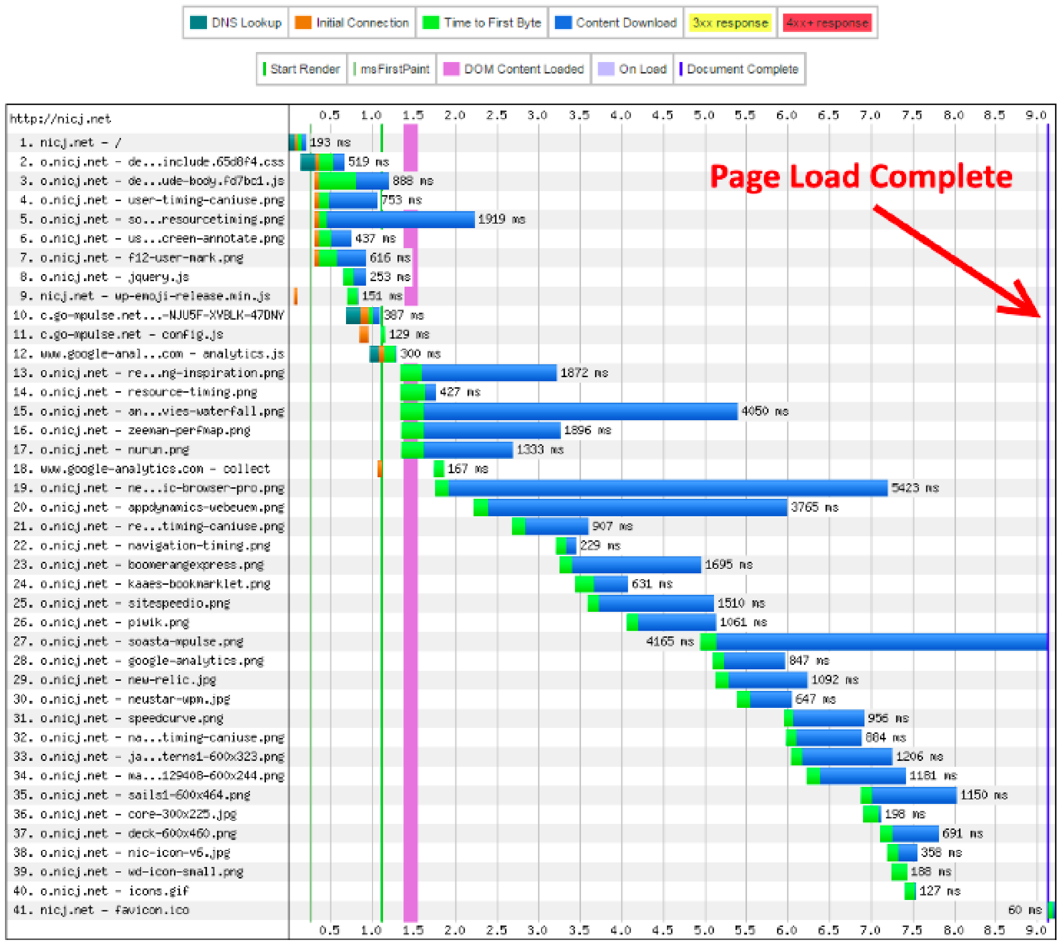
CHALLENGE #1

THE ONLOAD EVENT NO LONGER MATTERS

Traditional websites:

- On navigation, the browser begins downloading all of the JavaScript, CSS, images and other static resources
- Once all static resources are fetched, the body's `onload` event will fire
- This is traditionally what websites consider as page load complete
- This is traditionally what RUM measures

TRADITIONAL WEBSITE WATERFALL



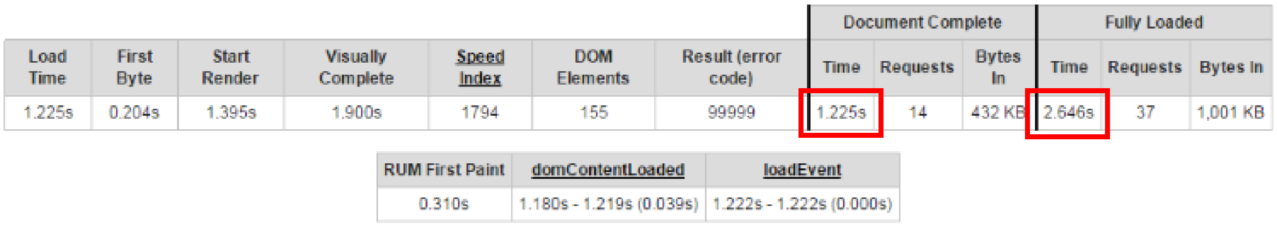
CHALLENGE #1

THE ONLOAD EVENT NO LONGER MATTERS

Single Page Apps:

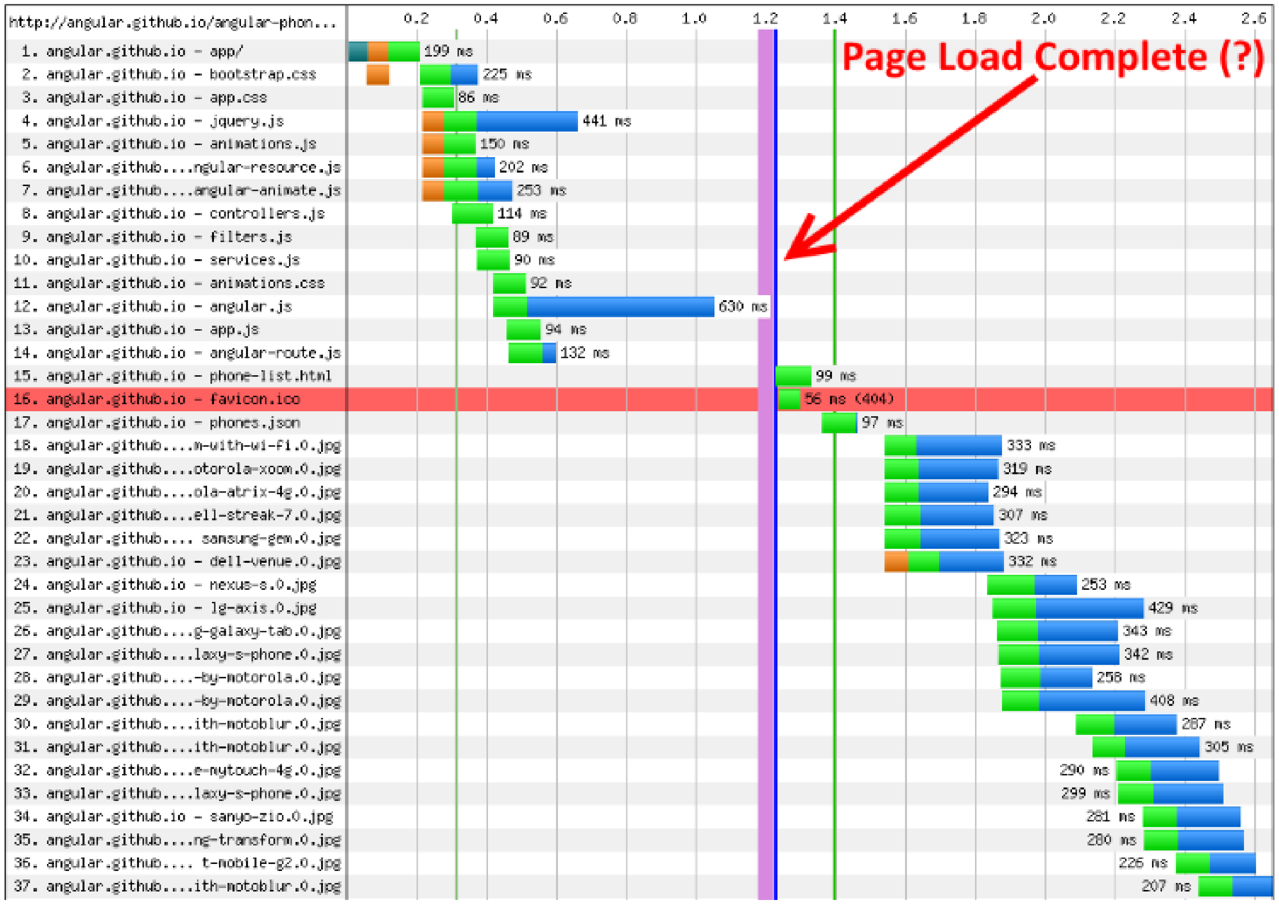
- Load all static content like a traditional website
- The frameworks' code will also be fetched (e.g. `angular.js`)
- *(the onload event fires here)*
- Once the SPA framework is loaded, it starts looking at routes, fetching views and data
- All of this content is fetched *after* the `onload` event

SPA WATERFALL



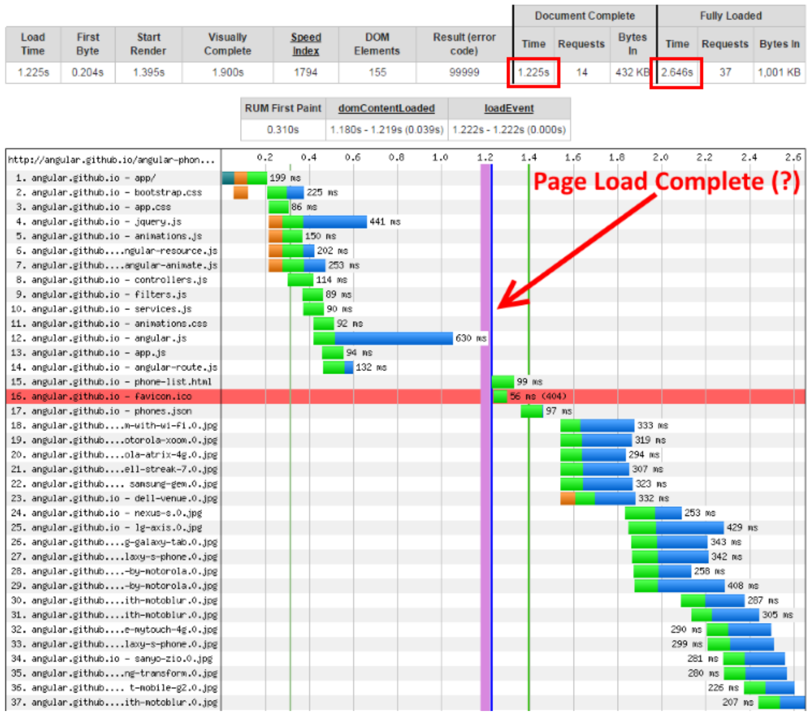
							Document Complete			Fully Loaded		
Load Time	First Byte	Start Render	Visually Complete	Speed Index	DOM Elements	Result (error code)	Time	Requests	Bytes In	Time	Requests	Bytes In
1.225s	0.204s	1.395s	1.900s	1794	155	99999	1.225s	14	432 KB	2.646s	37	1,001 KB

RUM First Paint	domContentLoaded	loadEvent
0.310s	1.180s - 1.219s (0.039s)	1.222s - 1.222s (0.000s)



http://angular.github.io/angular-phon...	
1. angular.github.io - app/	199 ms
2. angular.github.io - bootstrap.css	225 ms
3. angular.github.io - app.css	86 ms
4. angular.github.io - jquery.js	441 ms
5. angular.github.io - animations.js	150 ms
6. angular.github....ngular-resource.js	202 ms
7. angular.github....angular-animate.js	253 ms
8. angular.github.io - controllers.js	114 ms
9. angular.github.io - filters.js	89 ms
10. angular.github.io - services.js	90 ms
11. angular.github.io - animations.css	92 ms
12. angular.github.io - angular.js	630 ms
13. angular.github.io - app.js	94 ms
14. angular.github.io - angular-route.js	132 ms
15. angular.github.io - phone-list.html	99 ms
16. angular.github.io - favicon.ico	56 ms (404)
17. angular.github.io - phones.json	97 ms
18. angular.github....m-with-wi-fi.0.jpg	333 ms
19. angular.github....otorola-xoom.0.jpg	319 ms
20. angular.github....ola-atrix-4g.0.jpg	294 ms
21. angular.github....ell-streak-7.0.jpg	307 ms
22. angular.github.... samsung-gem.0.jpg	323 ms
23. angular.github.io - dell-venue.0.jpg	332 ms
24. angular.github.io - nexus-s.0.jpg	253 ms
25. angular.github.io - lg-axis.0.jpg	429 ms
26. angular.github....g-galaxy-tab.0.jpg	343 ms
27. angular.github....laxy-s-phone.0.jpg	342 ms
28. angular.github....-by-motorola.0.jpg	258 ms
29. angular.github....-by-motorola.0.jpg	408 ms
30. angular.github....ith-motoblur.0.jpg	287 ms
31. angular.github....ith-motoblur.0.jpg	305 ms
32. angular.github....e-mytouch-4g.0.jpg	290 ms
33. angular.github....laxy-s-phone.0.jpg	299 ms
34. angular.github.io - sanyo-zio.0.jpg	281 ms
35. angular.github....ng-transform.0.jpg	280 ms
36. angular.github.... t-mobile-g2.0.jpg	226 ms
37. angular.github....ith-motoblur.0.jpg	207 ms

SPA WATERFALL

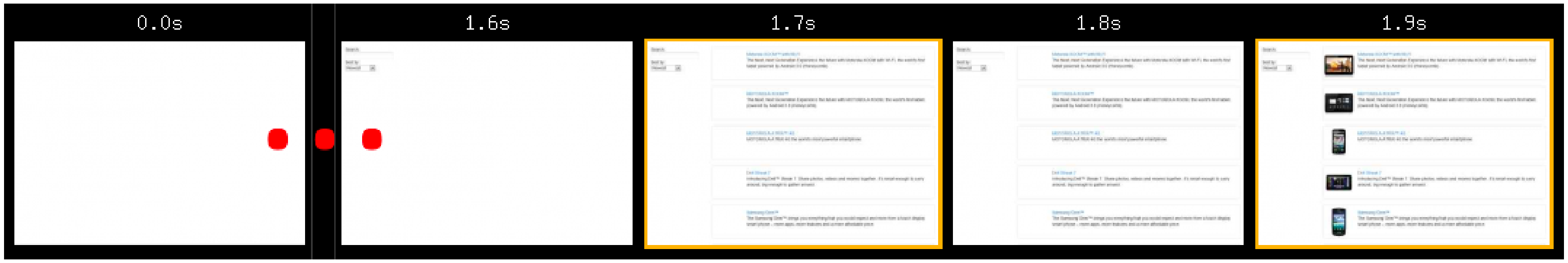


Load Time	First Byte	Start Render	Visually Complete	Speed Index	DOM Elements	Result (error code)	Document Complete Time	Document Complete Requests	Document Complete Bytes In	Fully Loaded Time	Fully Loaded Requests	Fully Loaded Bytes In
1.225s	0.204s	1.395s	1.900s	1794	155	99999	1.225s	14	432 KB	2.646s	37	1,001 KB

RUM First Paint	domContentLoaded	loadEvent
0.310s	1.180s - 1.219s (0.039s)	1.222s - 1.222s (0.000s)

http://angular.github.io/angular-phon...	
1. angular.github.io - app/	199 ms
2. angular.github.io - bootstrap.css	225 ms
3. angular.github.io - app.css	86 ms
4. angular.github.io - jquery.js	441 ms
5. angular.github.io - animations.js	150 ms
6. angular.github....ngular-resource.js	202 ms
7. angular.github....angular-animate.js	253 ms
8. angular.github.io - controllers.js	114 ms
9. angular.github.io - filters.js	89 ms
10. angular.github.io - services.js	90 ms
11. angular.github.io - animations.css	92 ms
12. angular.github.io - angular.js	630 ms
13. angular.github.io - app.js	94 ms
14. angular.github.io - angular-route.js	132 ms
15. angular.github.io - phone-list.html	99 ms
16. angular.github.io - favicon.ico	56 ms (404)
17. angular.github.io - phones.json	97 ms
18. angular.github....m-with-wi-fi.0.jpg	333 ms
19. angular.github....otorola-xoom.0.jpg	319 ms
20. angular.github....ola-atrix-4g.0.jpg	294 ms
21. angular.github....ell-streak-7.0.jpg	307 ms
22. angular.github.... samsung-gem.0.jpg	323 ms
23. angular.github.io - dell-venue.0.jpg	332 ms
24. angular.github.io - nexus-s.0.jpg	253 ms
25. angular.github.io - lg-axis.0.jpg	429 ms
26. angular.github....g-galaxy-tab.0.jpg	343 ms
27. angular.github....laxy-s-phone.0.jpg	342 ms
28. angular.github....-by-motorola.0.jpg	258 ms
29. angular.github....-by-motorola.0.jpg	408 ms
30. angular.github....ith-motoblur.0.jpg	287 ms
31. angular.github....ith-motoblur.0.jpg	305 ms
32. angular.github....e-mytouch-4g.0.jpg	290 ms
33. angular.github....laxy-s-phone.0.jpg	299 ms
34. angular.github.io - sanyo-zio.0.jpg	281 ms
35. angular.github....ng-transform.0.jpg	280 ms
36. angular.github.... t-mobile-g2.0.jpg	226 ms
37. angular.github....ith-motoblur.0.jpg	207 ms

- Browser fires `onload` at 1.225 seconds
- All visual resources (.jpgs) aren't complete until after 1.7 seconds
- Filmstrip confirms nothing is shown until around 1.7 seconds
- `onload` fired 0.5 seconds too early!



CHALLENGE #1

THE ONLOAD EVENT NO LONGER MATTERS

Single Page Apps:

- Core problem is that most of the interesting stuff (e.g. fetching images, JavaScript, CSS and XHRs for the route) happens after the onload
- The browser doesn't fire any "fully loaded"-style events after `onload`

CHALLENGE #2

SOFT NAVIGATIONS ARE NOT REAL NAVIGATIONS

- Each route change, user interaction, or visual update is dynamically fetched from the server
- There are APIs to change the URL (and detect changes) in the address bar without actually navigating
- New content is dynamically swapped in over the old content
- The browser is no longer doing a traditional navigation, where it's tearing down the old page

CHALLENGE #2

SOFT NAVIGATIONS ARE NOT REAL NAVIGATIONS

- This is great for performance
- The browser is no longer re-rendering the same header, footer or common components
- The browser is no longer re-parsing the same HTML, JavaScript and CSS

CHALLENGE #2

SOFT NAVIGATIONS ARE NOT REAL NAVIGATIONS

Bad for traditional RUM tools:

- Stop caring after the measuring the "one" navigation
- Won't run again until the next time it loads on a full navigation
- Browser events (`readyState`, `onload`) and metrics (NavigationTiming) are all geared toward a single load event

CHALLENGE #3

THE BROWSER WON’T TELL YOU WHEN ALL RESOURCES HAVE BEEN DOWNLOADED

- The browser fires `onload` only once
- The `onload` event helps us know when all static content was fetched
- In a soft navigation scenario, the browser does not fire the `onload` event again, so we don't know when its content was fetched

CHALLENGE #3

THE BROWSER WON'T TELL YOU WHEN ALL RESOURCES HAVE BEEN DOWNLOADED

SPA soft navigations may fetch:

- Templates
- Images
- CSS
- JavaScript
- XHRs
- Videos

CHALLENGE #3

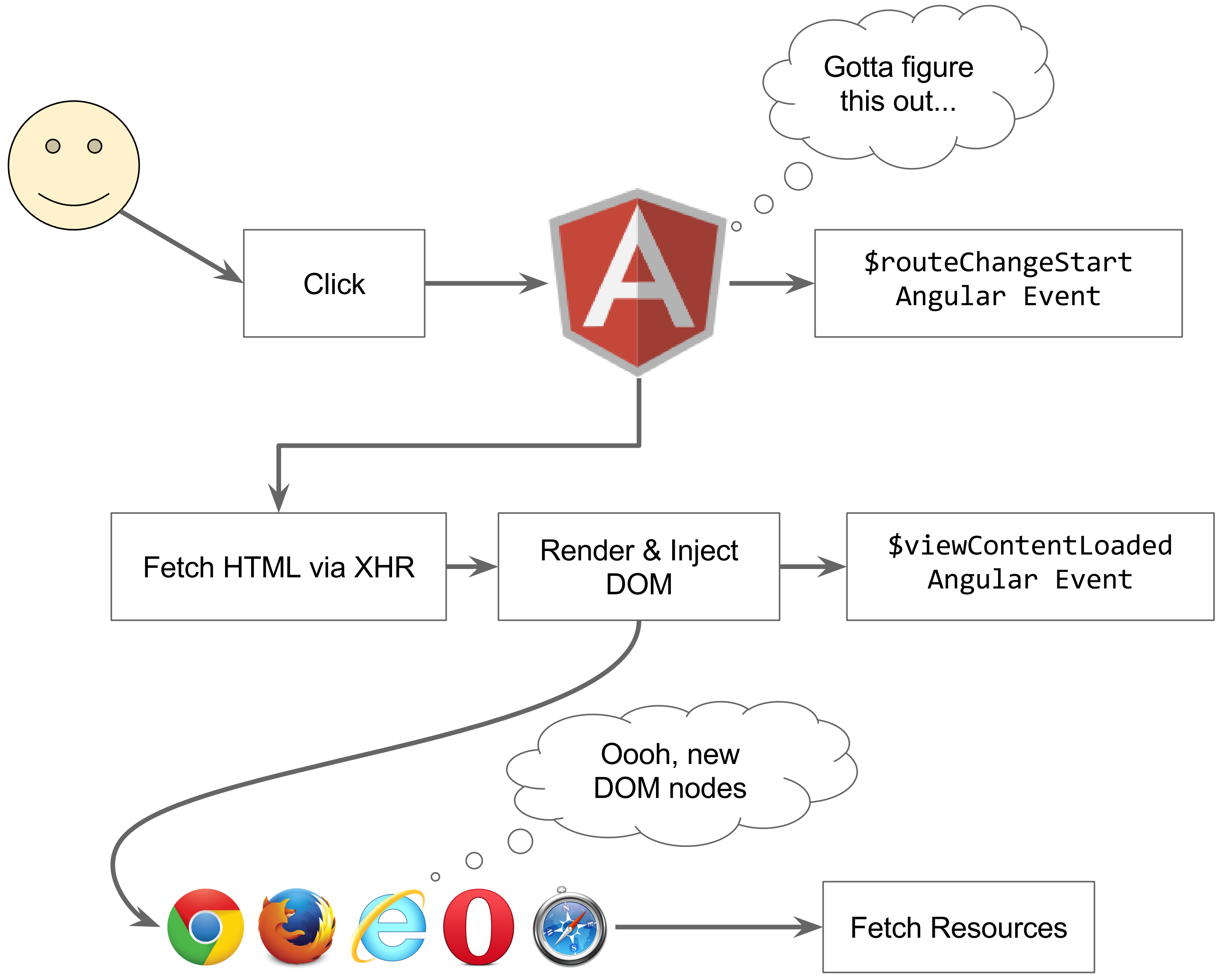
THE BROWSER WON'T TELL YOU WHEN ALL RESOURCES HAVE BEEN DOWNLOADED

SPA frameworks often fire events around navigations. AngularJS events:

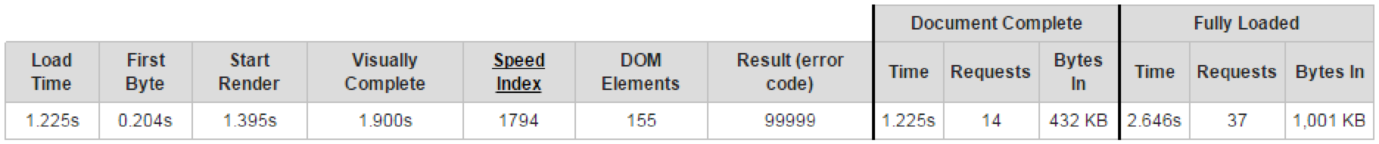
- `$routeChangeStart`: When a new route is being navigated to
- `$viewContentLoaded`: Emitted every time the ngView content is reloaded

But neither of these events have any knowledge of the work they trigger, fetching new IMGs, CSS, JavaScript, etc!

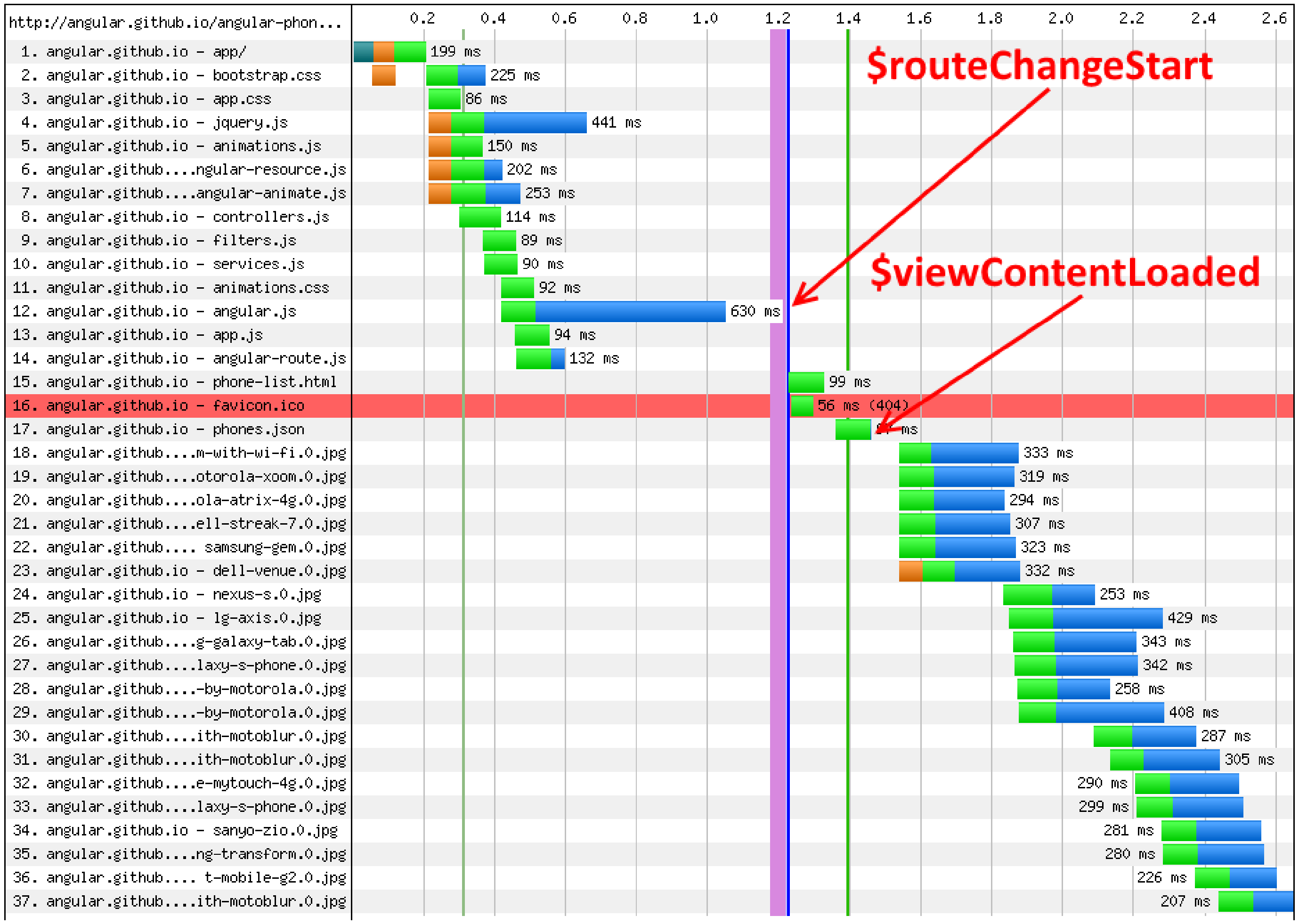
ANGULAR TIMELINE



ANGULARJS EVENT WATERFALL



							Document Complete			Fully Loaded		
Load Time	First Byte	Start Render	Visually Complete	Speed Index	DOM Elements	Result (error code)	Time	Requests	Bytes In	Time	Requests	Bytes In
1.225s	0.204s	1.395s	1.900s	1794	155	99999	1.225s	14	432 KB	2.646s	37	1,001 KB



http://angular.github.io/angular-phon...	
1. angular.github.io - app/	199 ms
2. angular.github.io - bootstrap.css	225 ms
3. angular.github.io - app.css	86 ms
4. angular.github.io - jquery.js	441 ms
5. angular.github.io - animations.js	150 ms
6. angular.github....ngular-resource.js	202 ms
7. angular.github....angular-animate.js	253 ms
8. angular.github.io - controllers.js	114 ms
9. angular.github.io - filters.js	89 ms
10. angular.github.io - services.js	90 ms
11. angular.github.io - animations.css	92 ms
12. angular.github.io - angular.js	630 ms
13. angular.github.io - app.js	94 ms
14. angular.github.io - angular-route.js	132 ms
15. angular.github.io - phone-list.html	99 ms
16. angular.github.io - favicon.ico	56 ms (404)
17. angular.github.io - phones.json	[illegible] ms
18. angular.github....m-with-wi-fi.0.jpg	333 ms
19. angular.github....otorola-xoom.0.jpg	319 ms
20. angular.github....ola-atrix-4g.0.jpg	294 ms
21. angular.github....ell-streak-7.0.jpg	307 ms
22. angular.github.... samsung-gem.0.jpg	323 ms
23. angular.github.io - dell-venue.0.jpg	332 ms
24. angular.github.io - nexus-s.0.jpg	253 ms
25. angular.github.io - lg-axis.0.jpg	429 ms
26. angular.github....g-galaxy-tab.0.jpg	343 ms
27. angular.github....laxy-s-phone.0.jpg	342 ms
28. angular.github....-by-motorola.0.jpg	258 ms
29. angular.github....-by-motorola.0.jpg	408 ms
30. angular.github....ith-motoblur.0.jpg	287 ms
31. angular.github....ith-motoblur.0.jpg	305 ms
32. angular.github....e-mytouch-4g.0.jpg	290 ms
33. angular.github....laxy-s-phone.0.jpg	299 ms
34. angular.github.io - sanyo-zio.0.jpg	281 ms
35. angular.github....ng-transform.0.jpg	280 ms
36. angular.github.... t-mobile-g2.0.jpg	226 ms
37. angular.github....ith-motoblur.0.jpg	207 ms

HOW CAN WE MEASURE SPA NAVIGATIONS?

We need to figure out at what point the navigation started (the **start event**), through when we consider the navigation complete (the **end event**).

THE START EVENT

For **hard navigations**:

- The start event is when the browser starts the process of loading the next page
- This is the same time as with traditional web app navigations
- We can use NavigationTiming's `navigationStart` if available, to know when the browser navigation began
- If NavigationTiming isn't available, and the user is navigating between pages on the same site, you can use cookies to measure when the navigation began (see Boomerang for an implementation)

THE START EVENT

Challenge #2: Soft navigations are not real navigations

- We need to figure out when the user's view is going to significantly change
- The browser **history** is changing
- SPA framework **routing events** can give us an indicator that the view will be changing
- Other important events that might indicate a view change are a user **click**, or an **XHR** that triggers DOM changes

THE START EVENT: HISTORY STATE

The **`window.history`** object can tell us when the URL is changing:

- When `pushState` or `replaceState` are being called, the app is possibly updating its view
- When the user hits Back or Forward, the `window.popstate` event is fired, and the app will possibly update the view
- *(future events will give us more info)*

THE START EVENT: ROUTING

SPA frameworks fire **routing** events when the view is changing:

- **AngularJS**: `$rootScope.$on("$routeChangeStart")`
- **Ember.js**: `beforeModel` or `willTransition`
- **Backbone.js**: `router.on("route")`

THE START EVENT: CLICKS

- When the user has **clicks** something, they might be doing simple interactions (e.g. a drop-down menu)
- Or, they might be triggering a UI update
- *(future events will give us more info)*

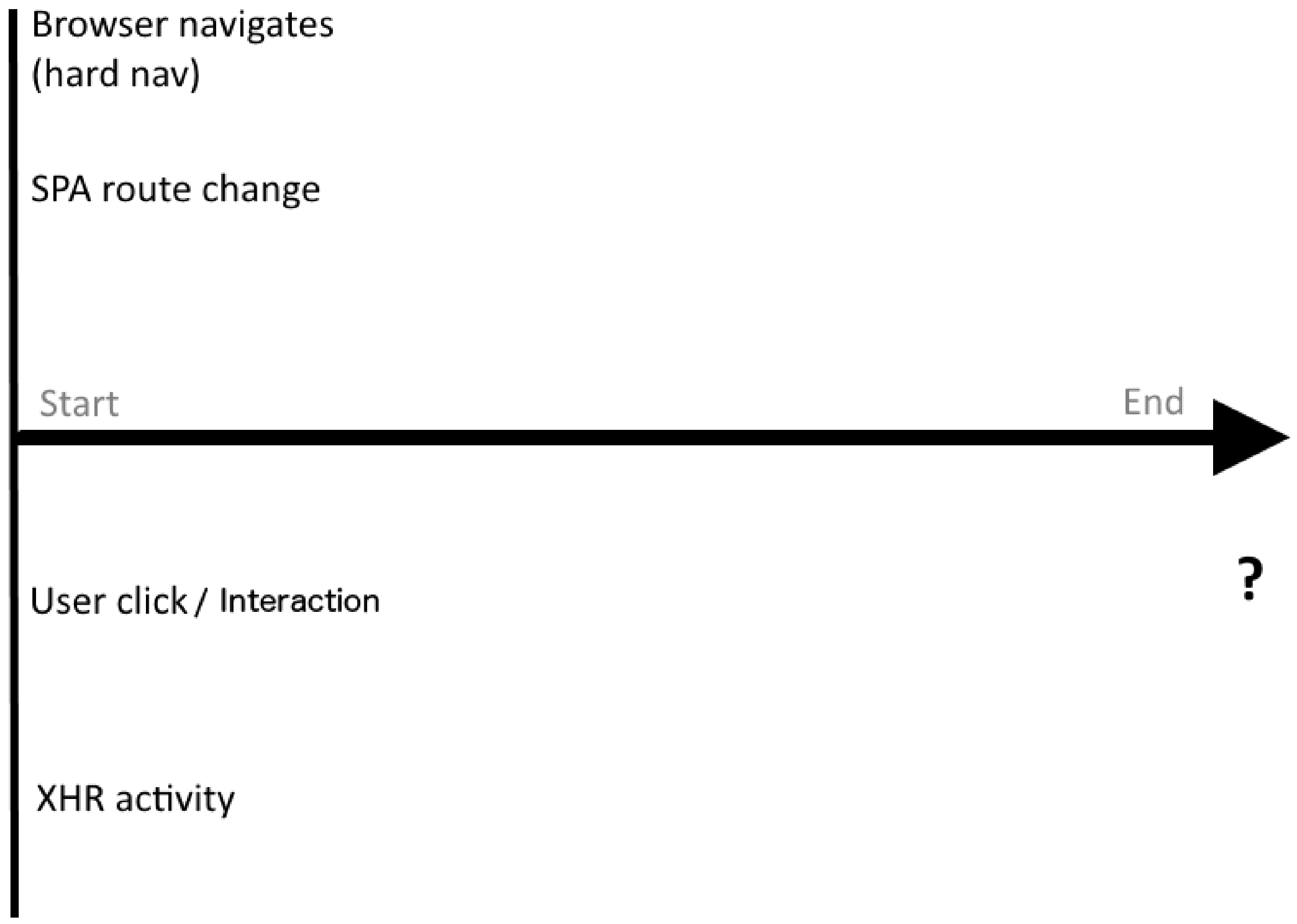
THE START EVENT: XHRS

- An **`XMLHttpRequest`** (network activity) might indicate that the page's view is being updated
- Or, it could be a periodic poller (e.g. a scoreboard update)
- Or, it could be in reaction to a user interaction (e.g. autocomplete)
- *(future events will give us more info)*

THE START EVENT

- To determine if a user **click** or **XHR** is really triggering a navigation, we can listen to what happens next
- If there was a lot of subsequent network activity, we can keep on listening for more events
- If history (address bar) changed, we can consider the event the start of a navigation
- If the DOM was updated significantly, we can consider the event the start of a navigation
- If nothing else happened, it was probably just an insignificant interaction

SPA NAVIGATIONS



THE END EVENT

When do we consider the SPA navigation complete?

There are many definitions of complete:

- When all networking activity has completed
- When the UI is visually complete (above-the-fold)
- When the user can interact with the page

THE END EVENT

Traditional RUM measures up to the `onload` event:

- This is when all resources have been fetched
- The page isn't fully loaded until *at least* then
- The UI might have been above-the-fold visually complete already
- It's traditionally when the user can fully interact with the page

SINGLE POINTS OF FAILURE (SPOFS)

Which resources could affect visual completion of the page?

- External JavaScript files
- External CSS files
- Media (images, video)

THE END EVENT

For **hard navigations**, the `onload` event no longer matters (Challenge #1)

- The `onload` event only measures up to when all static resources were fetched
- The SPA framework will be dynamically loading its UI only after the static JavaScript has been loaded
- We want to mark the end of the hard navigation only after all of the resources were fetched and the UI is complete

THE END EVENT

For **soft navigations**, the browser won’t tell you when all resources have been downloaded (Challenge #3)

- The `onload` only fires once on a page
- APIs like ResourceTiming can give you details about network resources after they've been fetched
- But to know when to stop, we need to know if there are any **outstanding** resources
- So let's monitor all network activity!

THE END EVENT

Let's **make our own** SPA `onload` event:

- Similar to the body `onload` event, let's wait for all network activity to complete
- This means we will have to **intercept** both implicit resource fetches (e.g. from new DOM elements) as well as programmatic (e.g. XHR) resource fetches

MONITORING XHRS

`XMLHttpRequests` play an important role in SPA frameworks

- XHRs are used to fetch HTML, templates, JSON, XML, data and other assets
- We should monitor to see if any XHRs are occuring
- The `XMLHttpRequest` object can be **proxied**
- Intercept the `.open()` and `.send()` methods to know when an XHR is starting

MONITORING XHRS

Simplified code ahead!

Full code at github.com/lognormal/boomerang/blob/master/plugins/auto_xh

MONITORING XHRS

```
var orig_XHR = window.XMLHttpRequest;
window.XMLHttpRequest = function() {
    var req = new orig_XHR();
    orig_open = req.open;
    orig_send = req.send;

    req.open = function(method, url, async) {
        // save URL details, listen for state changes
        req.addEventListener("load", function() { ... });
        req.addEventListener("timeout", function() { ... });
        req.addEventListener("error", function() { ... });
        req.addEventListener("abort", function() { ... });
        orig_open.apply(req, arguments);
    };

    req.send = function() {
        // save start time
        orig_send.apply(req, arguments);
    }
}
```

MONITORING XHRS

By proxying the XHR code, you can:

- Know which URLs are being fetched
- Know when a XHR has started
- Know when a XHR has completed, timed out, error or aborted
- Measure XHR states even on browsers that don't support ResourceTiming
- Most importantly, know if there are any **outstanding** XHRs

MONITORING XHRS

Downsides:

- Need additional code to support XDomainRequest
- Timing not as accurate when browser is busy (rendering, etc) as callbacks will be delayed
- You can fix-up timing via ResourceTiming (if available)

OTHER RESOURCES

XHR is the main way to fetch resources via JavaScript

- What about Images, JavaScript, CSS and other HTML elements that trigger resource fetches?
- We can't proxy the `Image` object as that only works if you create a `new Image()` in JavaScript
- If only we could listen for DOM changes...

MUTATION OBSERVER

http://developer.mozilla.org/en-US/docs/Web/API/MutationObserver:

MutationObserver provides developers a way to react to changes in a DOM

Usage:

- `observe()` for specific events
- Get a callback when mutations for those events occur

MUTATIONOBSERVER

Simplified code ahead!

Full code at

github.com/lognormal/boomerang/blob/master/plugins/auto_xh

```javascript
var observer = new MutationObserver(observeCallback);
observer.observe(document, {
    childList: true,
    attributes: true,
    subtree: true,
    attributeFilter: ["src", "href"]
});
```

```
function observeCallback(mutations) {
    var interesting = false;
    if (mutations && mutations.length) {
        mutations.forEach(function(mutation) {
            if (mutation.type === "attributes") {
                interesting |= isInteresting(mutation.target);
            } else if (mutation.type === "childList") {
                for (var i = 0; i < mutation.addedNodes.length; i++) {
                    interesting |= isInteresting(mutation.addedNodes[i]);
                }
            }
        });
    }
    if (!interesting) {
        // complete the event after N milliseconds if nothing else happens
    }
});
```

MUTATIONOBSERVER

Simplified workflow:

- Start listening when an XHR, click, route change or other interesting navigation-like event starts
- Use `MutationObserver` to listen for DOM mutations
- Attach `load` and `error` event handlers and set timeouts on any `IMG`, `SCRIPT`, `LINK` or `FRAME`
- If an interesting element starts fetching keep the navigation "open" until it completes
- After the last element's resource has been fetched, wait a few milliseconds to see if it kicked off anything else
- If not, the navigation completed when the last element's resource was fetched

MUTATIONOBSERVER

What's interesting to observe?

- Internal and cached resources may not fetch anything, so you have to inspect elements first
- `IMG` elements that haven't already been fetched (`naturalWidth==0`), have external URLs (e.g. not `data-uri:`) and that we haven't seen before.
- `SCRIPT` elements that have a `src` set
- `IFRAMEs` elements that don't have `javascript:` or `about:` protocols
- `LINK` elements that have a `href` set

MUTATIONOBSERVER

Downsides:

- Not 100% supported in today's market
- Can't be used to monitor *all* resources (e.g. fonts from CSS)

MUTATIONOBSERVER

Polyfills (with performance implications):

- github.com/webcomponents/webcomponentsjs
- github.com/megawac/MutationObserver.js

WHY NOT RESOURCETIMING?

Doesn't `ResourceTiming` have all of the data we need?

- ResourceTiming events are only added to the buffer **after** they complete
- In order to extend the SPA navigation end time, we have to know if any resource fetches are *outstanding*

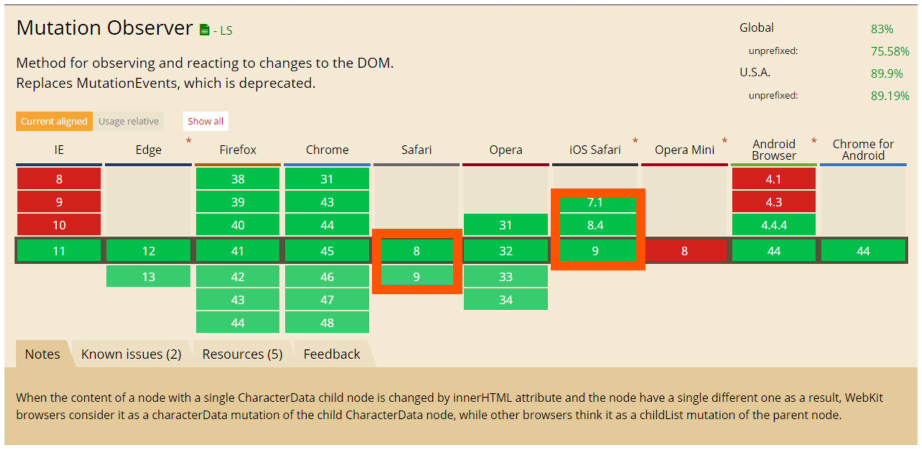
MUTATIONOBSERVER

Polyfill ResourceTiming via MutationObserver

For extra credit, you could use the data you gathered with Mutation Observer to create a Waterfall for browsers that don't support ResourceTiming but do support MutationObserver (e.g. iOS).

MUTATIONOBSERVER

Polyfill ResourceTiming via MutationObserver



Mutation Observer - LS

Method for observing and reacting to changes to the DOM.
Replaces MutationEvents, which is deprecated.

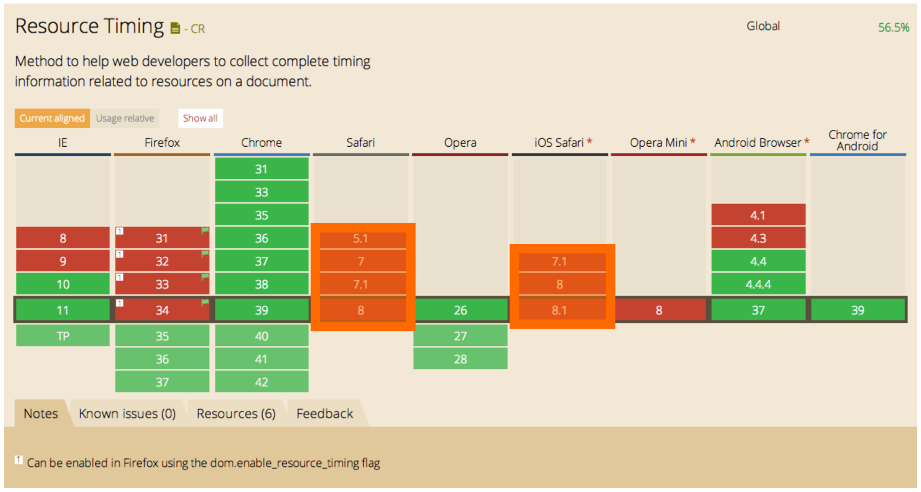
Global	83%
unprefixed:	75.58%
U.S.A.	89.9%
unprefixed:	89.19%

Current aligned · Usage relative · Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
8		38	31					4.1	
9		39	43			7.1		4.3	
10		40	44		31	8.4		4.4.4	
11	12	41	45	8	32	9	8	44	44
	13	42	46	9	33				
		43	47		34				
		44	48						

Notes · Known issues (2) · Resources (5) · Feedback

When the content of a node with a single CharacterData child node is changed by innerHTML attribute and the node have a single different one as a result, WebKit browsers consider it as a characterData mutation of the child CharacterData node, while other browsers think it as a childList mutation of the parent node.



Resource Timing - CR

Method to help web developers to collect complete timing information related to resources on a document.

Global	56.5%

Current aligned · Usage relative · Show all

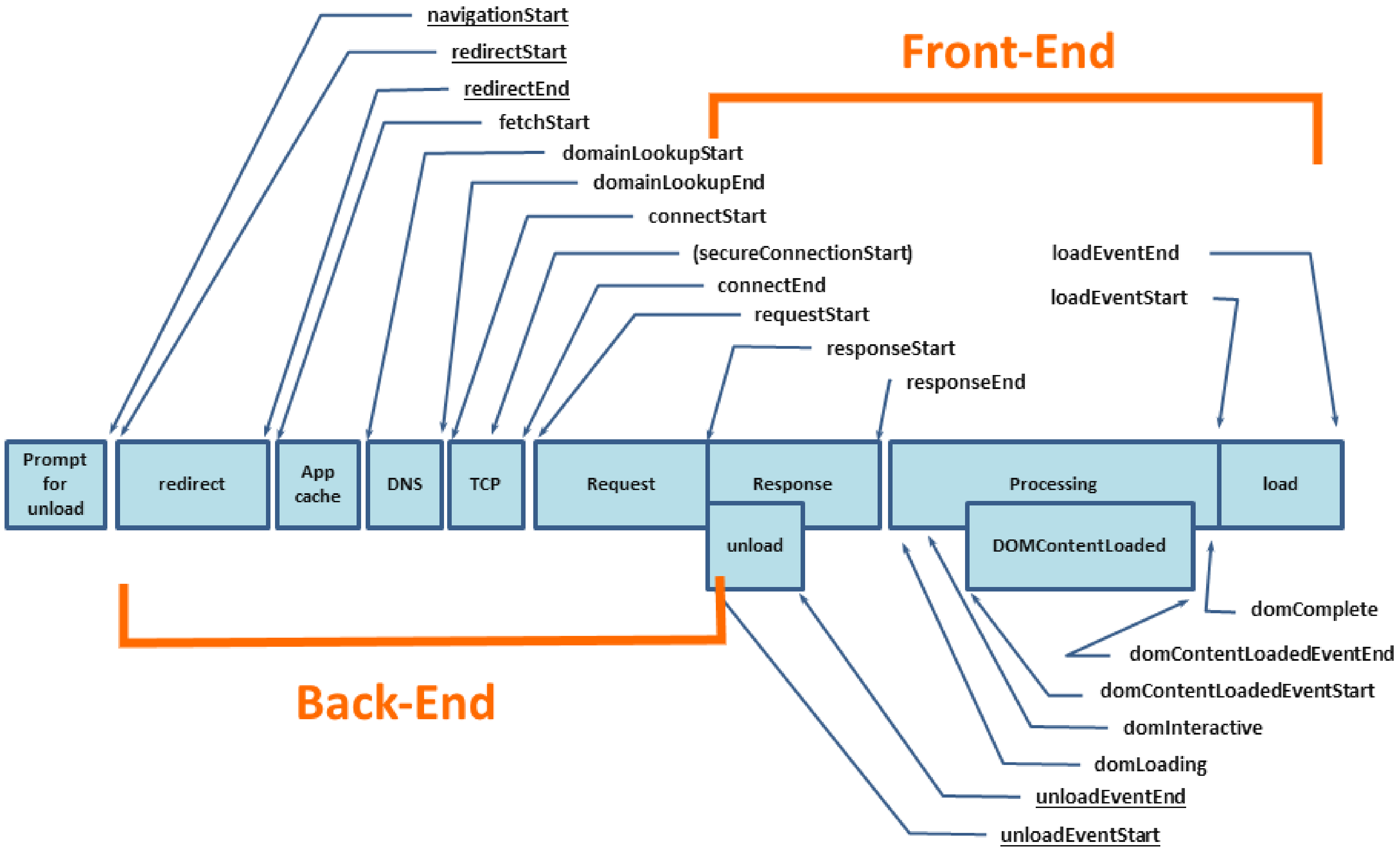
IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		31						
		33						
		35					4.1	
8	[1] 31	36	5.1				4.3	
9	[1] 32	37	7		7.1		4.4	
10	[1] 33	38	7.1		8		4.4.4	
11	[1] 34	39	8	26	8.1	8	37	39
TP	35	40		27				
	36	41		28				
	37	42						

Notes · Known issues (0) · Resources (6) · Feedback

[1] Can be enabled in Firefox using the dom.enable_resource_timing flag

FRONT-END VS. BACK-END

In a traditional page load:



FRONT-END VS. BACK-END

Traditional websites:

- **Back-End**: HTML fetch start to HTML response start
- **Front-End**: `Total Time - Back-End`

FRONT-END VS. BACK-END

Single Page Apps:

- Depends on your application's patterns, but...
- **Back-End**: Any timeslice with an XHR outstanding
- **Front-End**: `Total Time - Back-End`

MONITORING PAGE COMPONENTS

It's not just about **navigations**

What about components, widgets and ads?

- You can apply the previous techniques to page components
- For measuring performance, you need a **start time** and an **end time**
- The **start time** is probably driven by your code (e.g. a XHR fetch) or a user interaction (e.g. a click)
- The **end time** can be measured via XHR interception, MutationObservers, or callbacks from your resource fetches

MONITORING PAGE COMPONENTS

How do you measure **visual completion**?

Challenges:

- When an `IMG` has been fetched, that's not when it's displayed to the visitor (it has to decode, etc.)
- When you put HTML into the DOM, it's not *immediately* on the screen

MONITORING PAGE COMPONENTS

Use `setTimeout(..., 0)` or `setImmediate` to get a callback after the browser has finished parsing *some* DOM updates

```
var xhr = new XMLHttpRequest();
xhr.open("GET", "/fetchstuff");
xhr.addEventListener("load", function() {
    $(document.body).html(xhr.responseText);
    setTimeout(function() {
        var endTime = Date.now();
        var duration = endTime - startTime;
    }, 0);
});
var startTime = Date.now();
xhr.send();
```

MONITORING PAGE COMPONENTS

This isn't perfect:

- The browser *may* be doing layout, rendering or drawing async or on another thread
- But it's better than ignoring all the work the browser has to do to render DOM changes

LIFECYCLE

What happens **over time**?

How well does your app behave?

LIFECYCLE

It's not just about measuring interactions or how long components take to load

Tracking metrics **over time** can highlight performance, reliability and resource issues

LIFECYCLE

You could measure:

- Memory usage: `window.performance.memory` (Chrome)
- DOM Length: `document.documentElement.innerHTML.length`
- DOM Nodes: `document.getElementsByTagName("*").length`
- JavaScript errors: `window.onerror`
- Bytes fetched: `ResourceTiming2` or `XHRs`
- Frame rate: `requestAnimationFrame`

THE FUTURE!

OK, that sounded like a lot of work-arounds to measure Single Page Apps.

Yep.

Why can't the browser just tell give us performance data for SPAs in a better, more performant way?

LISTING FOR RESOURCE FETCHES

Instead of instrumenting `XMLHttpRequest` and using `MutationObserver` to find new elements that will fetch:

- W3C **Fetch** standard
- https://fetch.spec.whatwg.org/
- A Fetch Observer (https://github.com/whatwg/fetch/issues/65) that notifies us when a resource fetch starts/stops
- Less overhead than MutationObserver
- Tracks all resources rather than just DOM elements from MutationObserver

THANKS!

slideshare.net/nicjansma/ - @nicj - @bluesmoon

soasta.io/SPAperfbook