High Performance WebSocket

@wesleyhales

SHAPE
shapesecurity.com

This talk will cover…

- Light overview of client-server comm.
- Case study of WSS+SPDY project
- Problems faced in collecting real time data (Browser -> Server)
- Performance Checklist
- Animated GIFs

WebSocket

WebSocket provides low latency delivery of text and binary application data in both directions.

Not a replacement for XHR or SSE

XHR, SSE, and WS

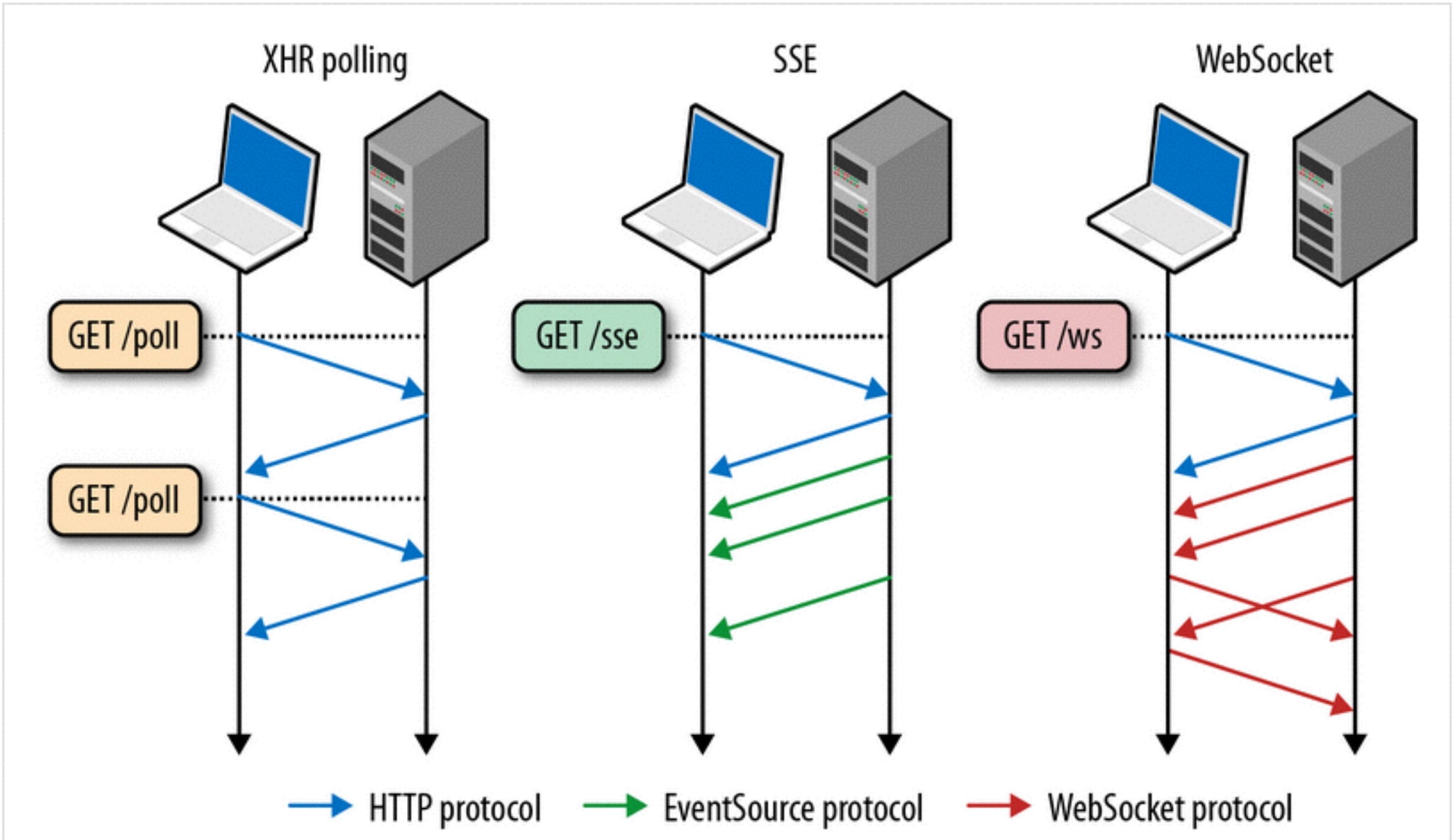


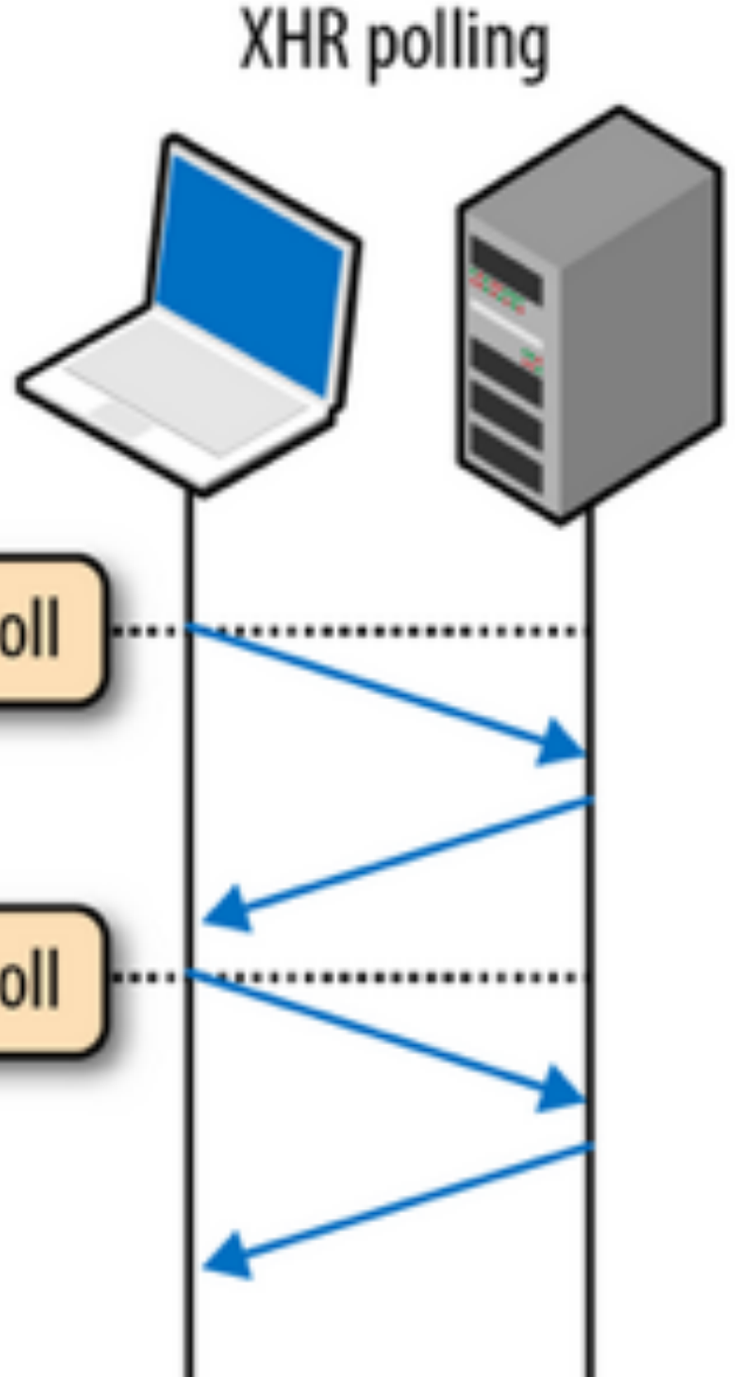
Figure 17-2. Communication flow of XHR, SSE, and WebSocket

XHR is…

"transactional"

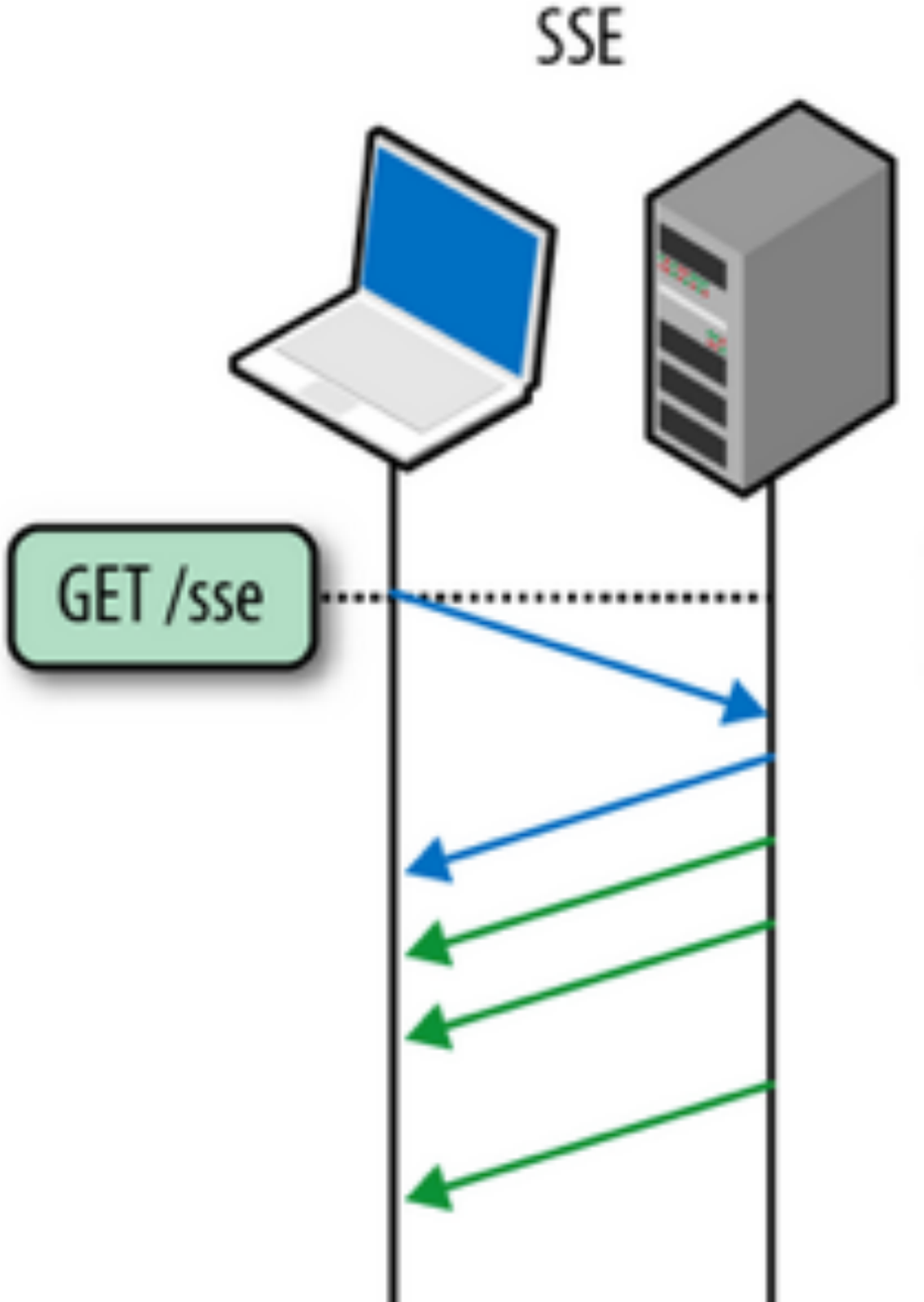
request-response

communication



SSE is…

Efficient, low-latency server-to-client streaming of text-based data



Experiment

onslyde

@wesleyhales

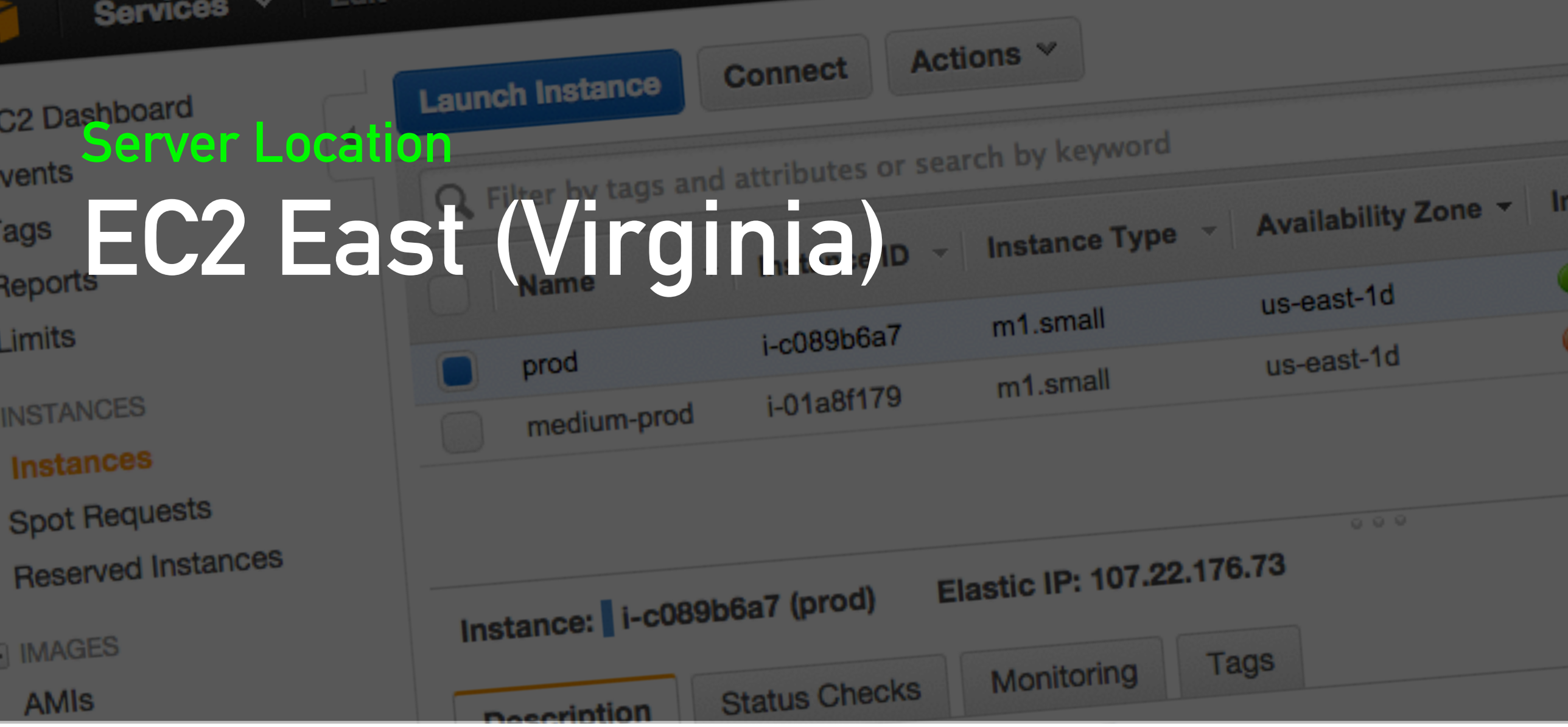
https://www.youtube.com/watch?v=uuHZ-d0Stsg#t=710

onslyde details

- Platform for real time feedback
- Started in early 2012
- Custom fallbacks no JS framework
- github.com/onslyde

Server Location

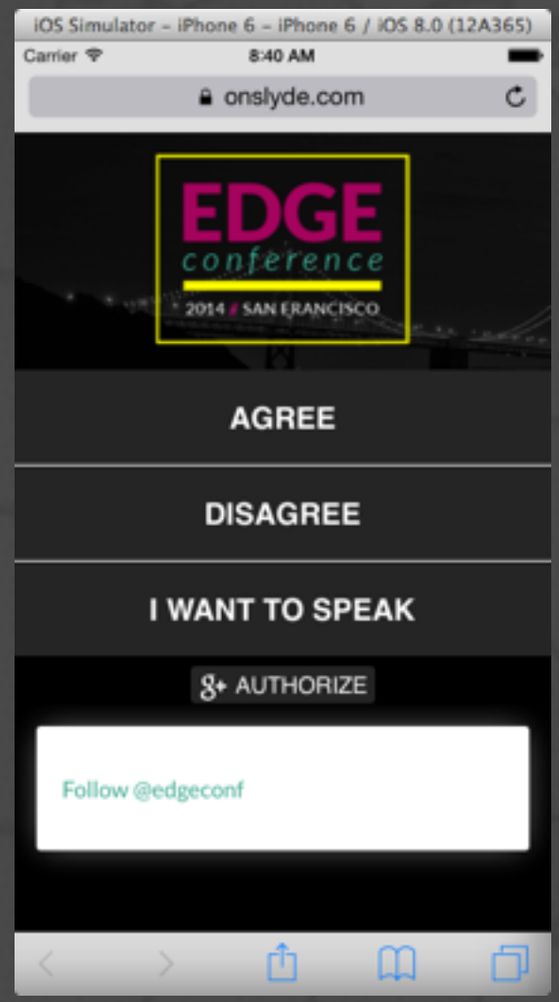
EC2 East (Virginia)



@wesleyhales

Browser Locations

Live Event Based



Live Event Based

Mobile Device Info	Sessions
Apple iPhone	140
LG Nexus 5	62
(not set)	26
Apple iPad	14
Mozilla Firefox for Android	12
BlackBerry KBD	9
Google Nexus 4	9
Mozilla Firefox OS	7
Motorola XT1053 Moto X	6
LG LGMS323 Optimus L70	5

Service Provider	Sessions
tw telecom holdings inc.	282
service provider corporation	52
(not set)	37
t-mobile usa inc.	13
cellco partnership dba verizon wireless	6
softlayer technologies inc.	5
sprint nextel corporation	5
comcast cable communications holdings	4
agile-inap	3
vodafone limited	2

Battery Consumption

Visits and Pages / Visit by Mobile Device Info

Mobile Device Info	Visits	Pages / Visit
Apple iPhone	252	3.57
Google Nexus 4	35	2.37
Apple iPad	29	2.97
Samsung Galaxy Nexus	29	2.10
Google Nexus 7	26	2.81
(not set)	18	1.39
RIM Z10	11	1.18
Samsung GT-I9300 Galaxy SIII	8	2.62
Samsung GT-N7100 Galaxy Note II	8	1.12
Samsung GT-I9505 Galaxy S IV	7	1.86

Visits by Brow

Browser
Chrome
Safari
Firefox
Android Brows
BlackBerry
Safari (in-app)

Avg. Visit Dur

Avg. Visit D

00:21:40

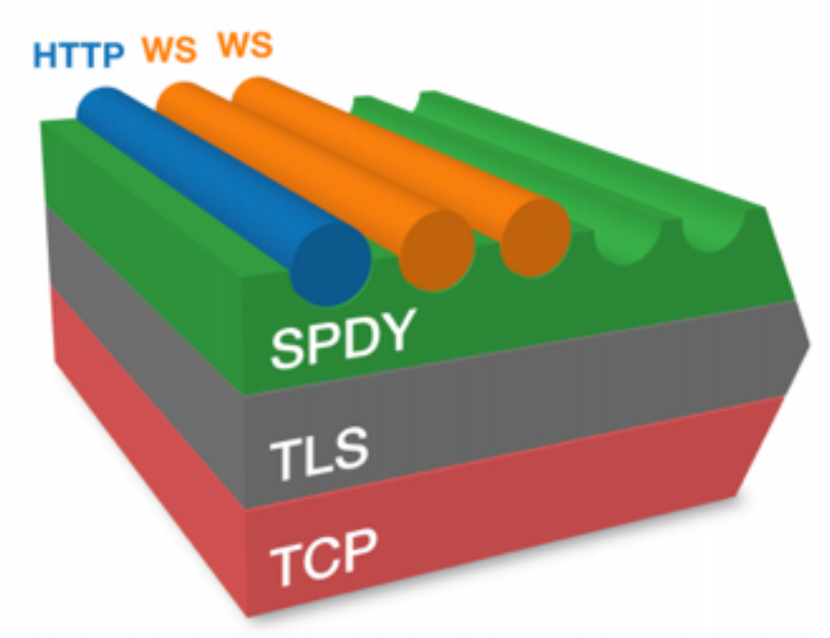
Server Stack

Supports the latest WebSocket & SPDY protocols

@wesleyhales

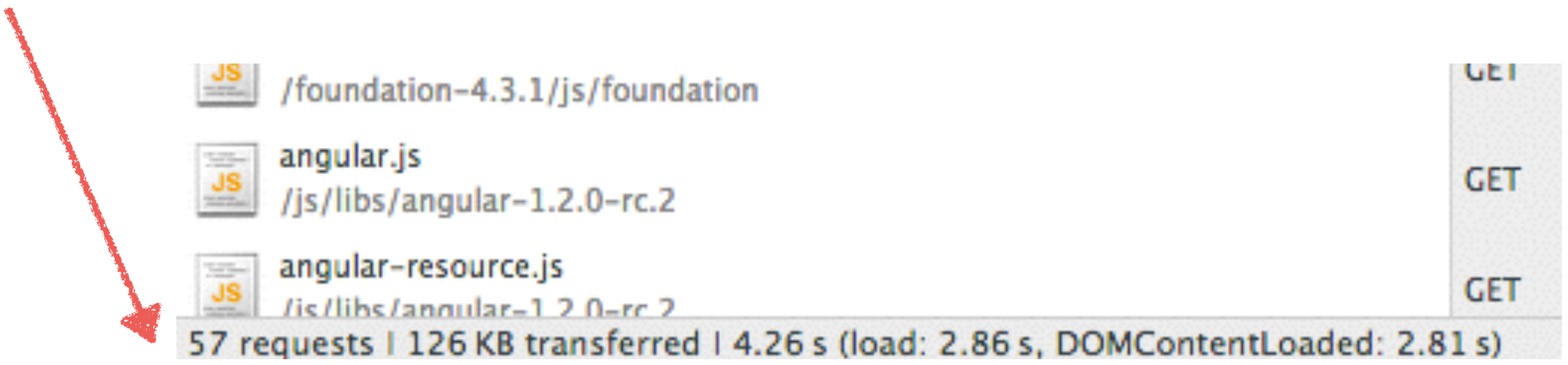
SPDY, WSS, and 443

- Layering is not currently supported in browsers
- Why do it?



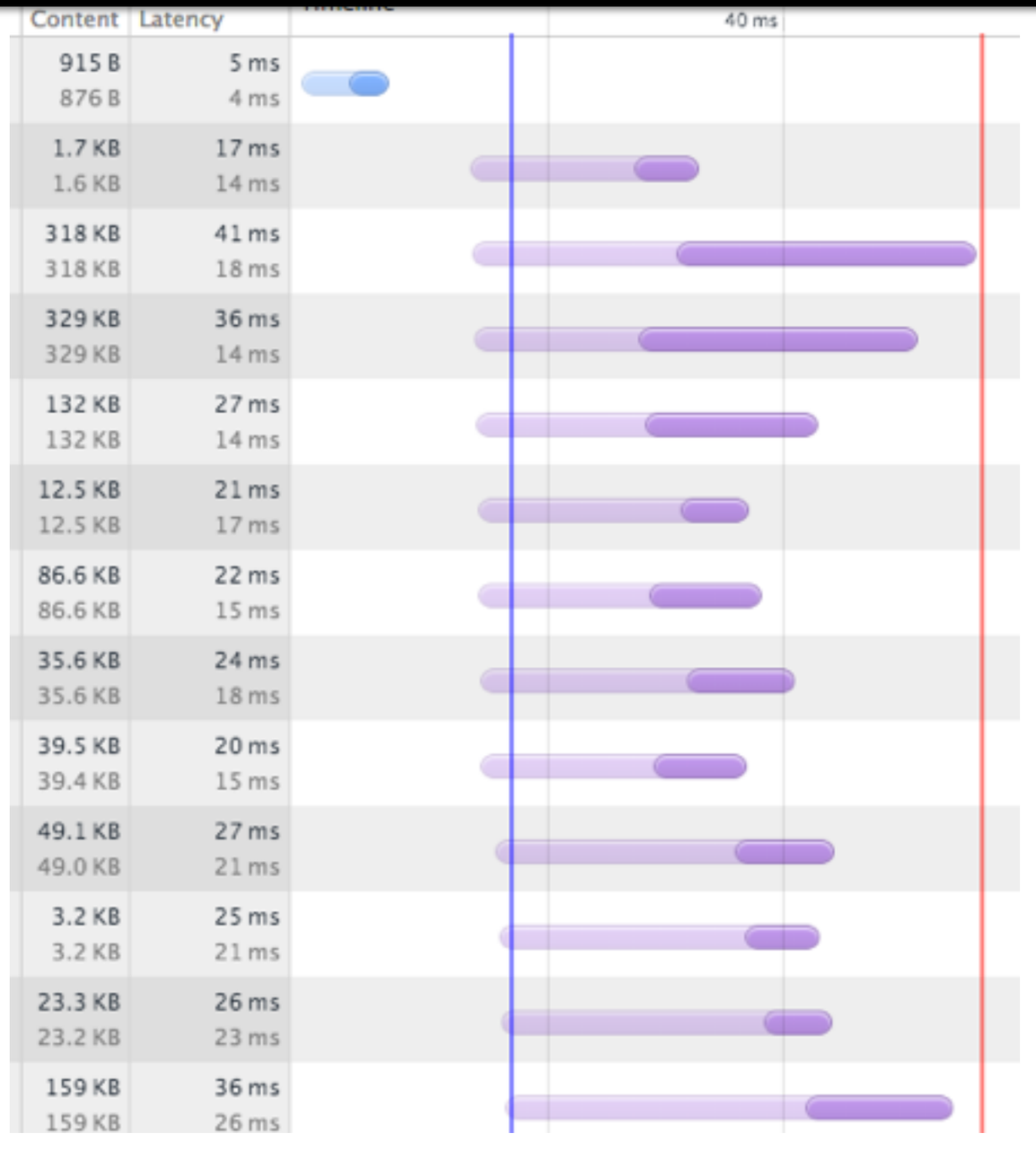
SPDY Goodness

Pre-Concat over SPDY3



Post-Concat over SPDY3

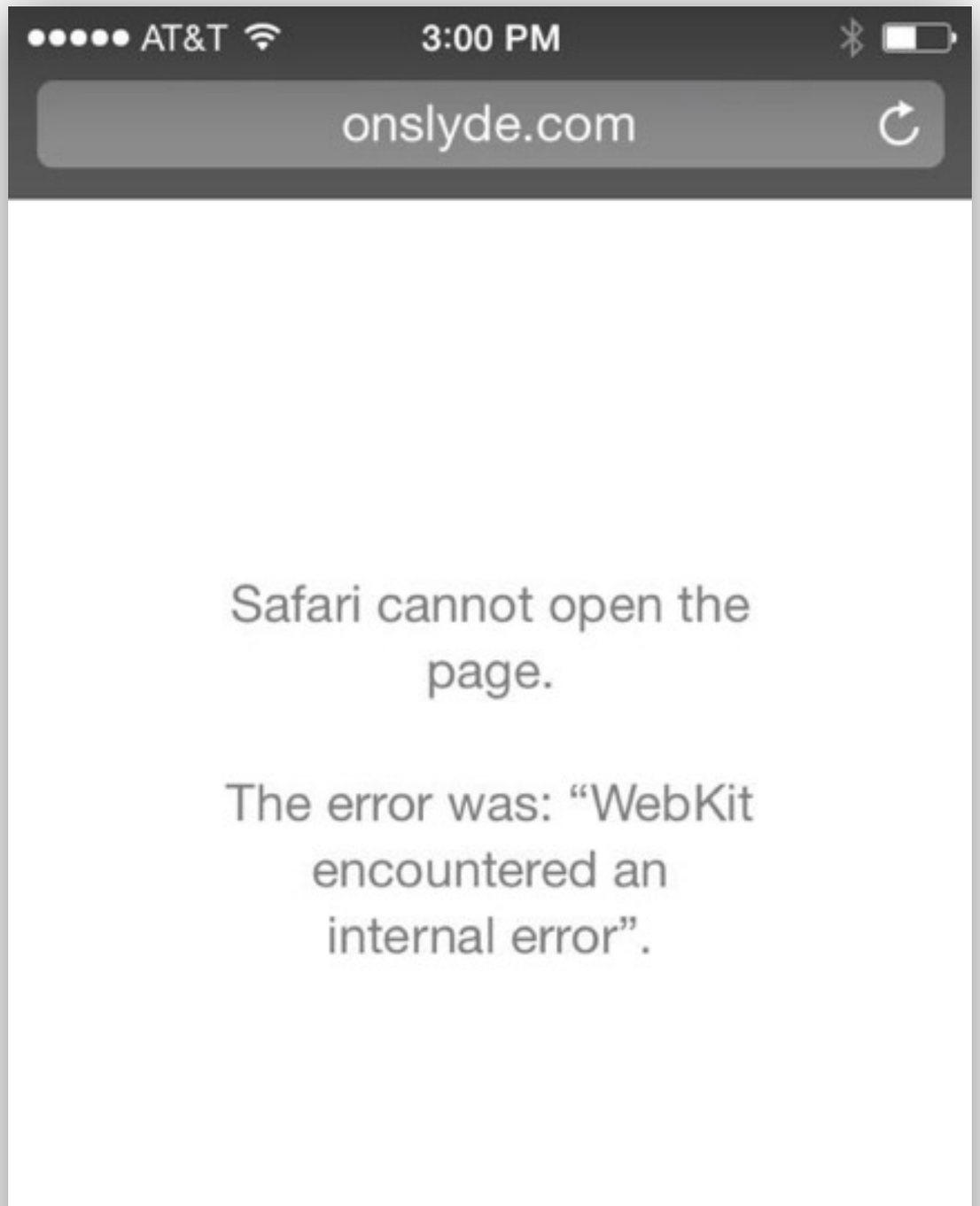
SPDY Goodness



https://www.onslyde.com/index-test.html

@wesleyhales

SPDY Badness



@wesleyhales

SPDY and HTTP 2

"HTTP/2 isn't magic Web performance pixie dust; you can't drop it in and expect your page load times to decrease by 50%."

- Mark Nottingham (chair)

About WS Data

× Headers | Frames | Cookies

Data	Length	Time
{"onslydeEvent":{"sessionID":"619","fire":function(){window.eventObjb = document.createEvent('Event');eventO…	244	10:17:04 AM
{"onslydeEvent":{"sessionID":"619","fire":function(){window.eventObjg = document.createEvent('Event');eventO…	317	10:17:04 AM
activeOptions:null,null,0:0	27	10:17:04 AM
Binary Frame (Opcode 2, mask)	0	10:17:04 AM
{"remoteMarkup":"%3Ch2 Binary Frame (Opcode 2, mask) %3EMy%20awesome%20presentation%3C%2Fh2%3E"}	86	10:17:04 AM
{"onslydeEvent":{"sessionID":"619","fire":function(){window.eventObjg = document.createEvent('Event');eventO…	358	10:17:03 AM
{"onslydeEvent":{"sessionID":"619","fire":function(){window.eventObjb = document.createEvent('Event');eventO…	244	10:17:03 AM
::connect::	11	10:17:03 AM

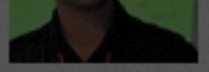
@wesleyhales

What is “High Performance” WebSocket?

Performance

32 sides to the perf coin:

- Server
- Connections
- Client side



0 DISAGREE

3 AGREE

onslyde

Demo

disag

14:03

@wesleyhales

onslyde

Measuring Millisecond Performance

@wesleyhales

Getting Server Time

Headers vs. Content rewrite

- Headers
- Content Rewrite

Sync Client with Server Time

Unfortunately, there isn't an API to give you the HTTP response headers for your initial page request.

So let's try AJAX…

Date Header

```
var oReq = new XMLHttpRequest();
oReq.onload = function(){
  var dateStr = oReq.getResponseHeader('Date');
  var serverTimeMillisUTC =
                new Date(Date.parse(dateStr)).getTime();
  var localMillisUTC =
                new Date().getTime();
  console.log(new Date(serverTimeMillisUTC),
                new Date(localMillisUTC))
};
oReq.open("HEAD", "/", false);
oReq.send();
```

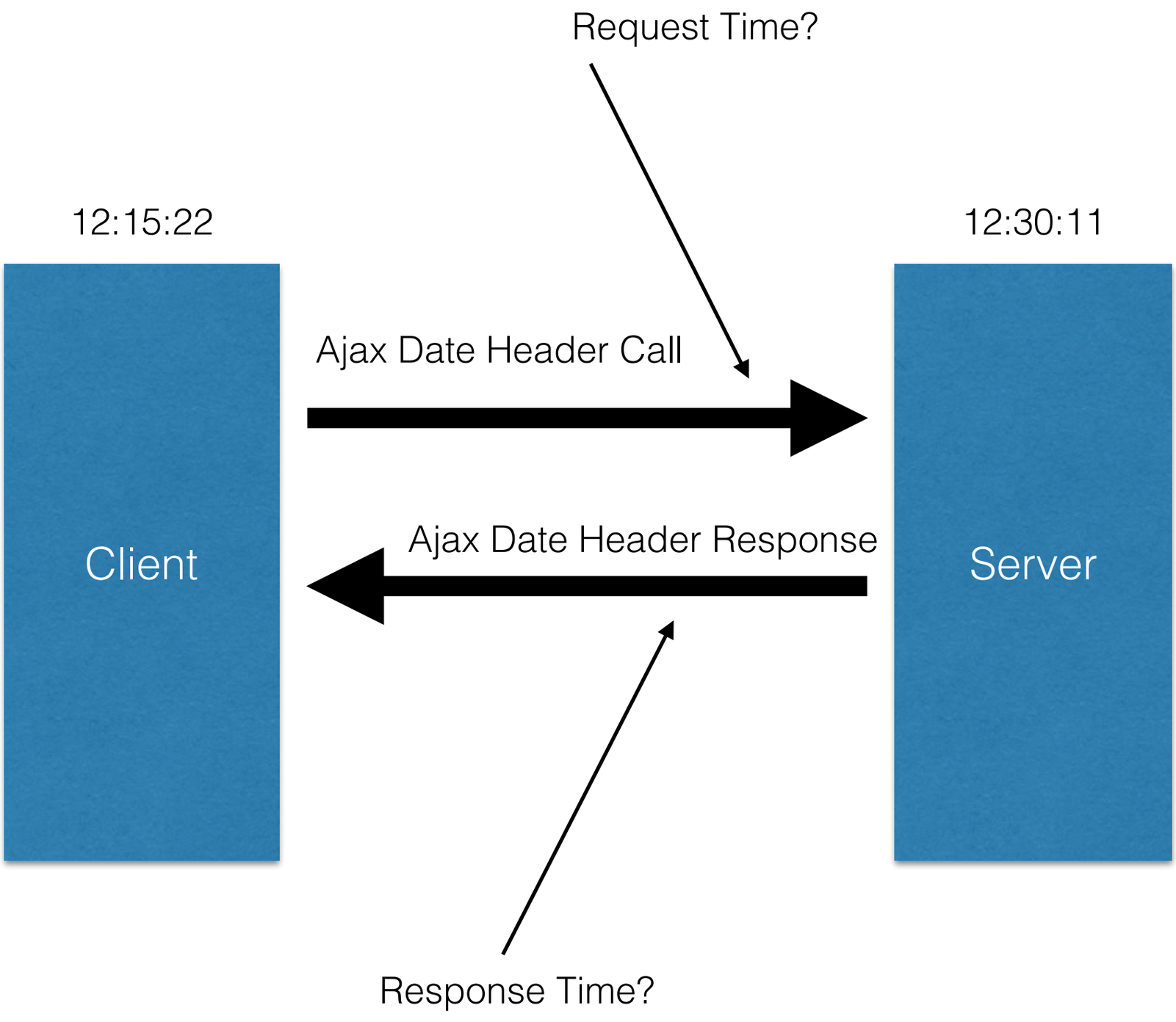
Date Header - Round Trip

```
var requestStart = new Date();
var oReq = new XMLHttpRequest();
oReq.onload = function(){
  var dateStr = oReq.getResponseHeader('Date');
  var serverTimeMillisUTC =
                new Date(Date.parse(dateStr)).getTime();
  var localMillisUTC =
                new Date().getTime();
   reqRespOffset = localMillisUTC - requestStart;
   offset =
       (serverTimeMillisUTC - localMillisUTC) + responseOffset;
};
oReq.open("HEAD", "/", false);
oReq.send();
```

Date Head d Trip

```
var reques t = new Date();
var oReq  XMLHttpRequest();
oReq.onl  function(){
  var d r = oReq.getRespons er('Date');
  var s TimeMillisUTC =
          new Date(Dat se(dateStr)).get ();
  var l illisUTC =
          new Date tTime();
   reqRe fset = lo illisUTC - requestSta
   offse
       (se illisUTC - localMillisUTC esponseOffset;
};
oReq.open("HE ", false);
oReq.send();
```

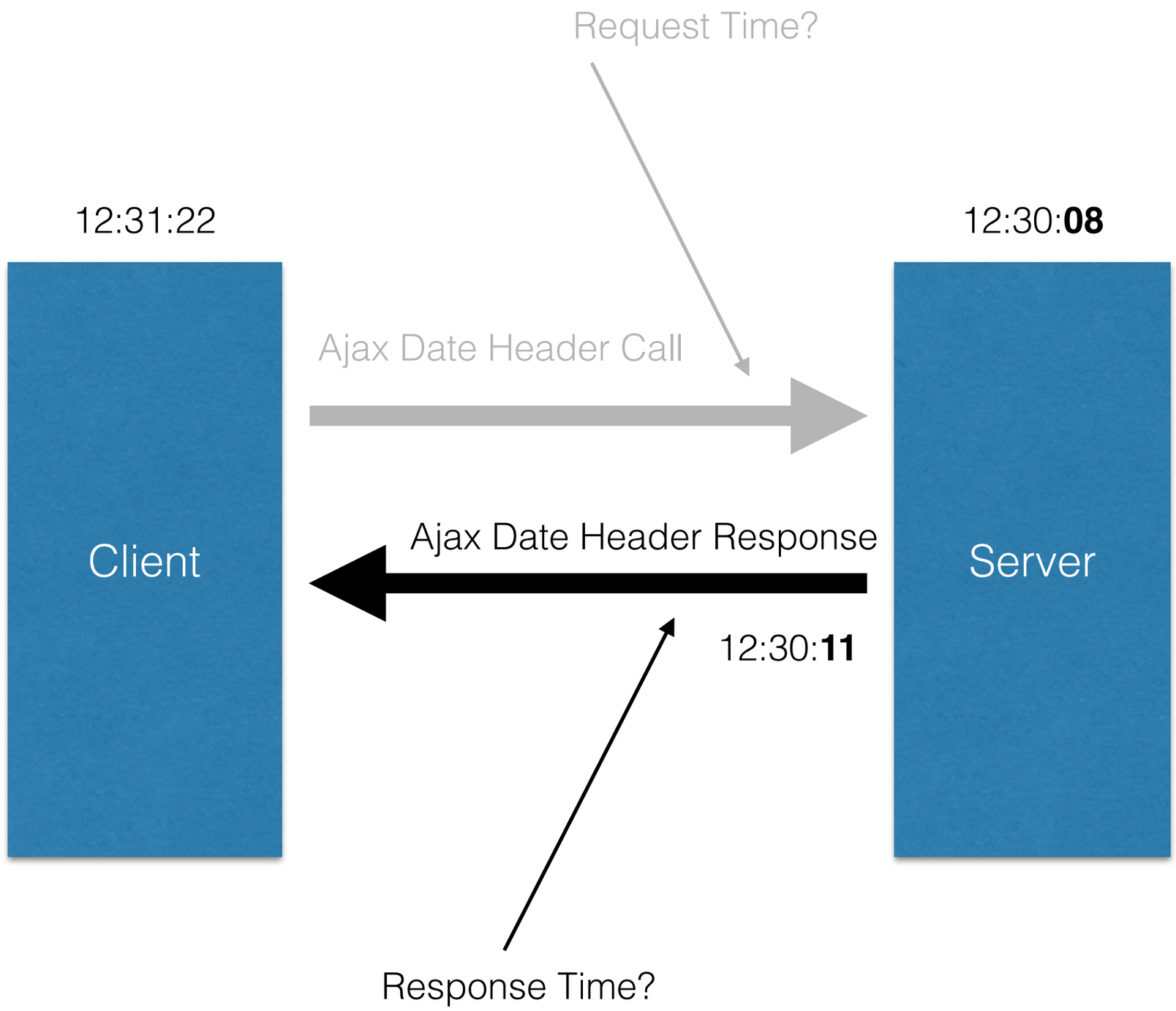
Instrumentation



Response Latency

- First request (out of sync with server) has to be measured.
- Add total response time to offset.

Instrumentation



Naviagtion Timing

Why can't we just pull the date header from the initial GET request and calculate with Navigation Timing API?

Perf Timing - Response Only

```
var requestStart = performance.timing.now;
var oReq = new XMLHttpRequest();
  oReq.onload = function(){
  var dateStr = oReq.getResponseHeader('Date');
  var serverTimeMillisUTC =
       Date.parse(new Date(Date.parse(dateStr)).toUTCString());
  var localMillisUTC =
       Date.parse(new Date().toUTCString());
  responseOffset =
        performance.timing.responseEnd - performance.timing.responseStart;
  offset = (serverTimeMillisUTC -  localMillisUTC) + responseOffset;
};
oReq.open("HEAD", "/", false);
oReq.send();

function getServerTime() {
  var date = new Date();
  date.setTime(date.getTime() + offset);
  return date;
}
```

Perf Timi…se Only

```
var requestSt…erformance.timing.now
var oReq = …ttpRequest();
  oReq.onlo…unction(){
  var date…oReq.getResponseHeader('Date…
  var ser…MillisUTC =
       Da…se(new Date(Date.parse(d…).toUTCString()
  var loc…isUTC =
       Da…se(new Date().toUTCSt…);
  respons…t =
       p…ance.timing.resp… - performance.timing.r…Start;
  offset …verTimeMillisU…ocalMillisUTC) + response…;
};
oReq.open(…"/", fa…
oReq.send()…

function getS…{
  var date = ne…
  date.setTime(da…() + offset);
  return date;
}
```

Nav Timing - Response Only

AJAX requests don't have the Performance Timing API.

Solution (Hack)

- Write the epoch timestamp to the content.
- Use the navigation timing API with that timestamp to calculate the offset.

Perf Timing - Response Only

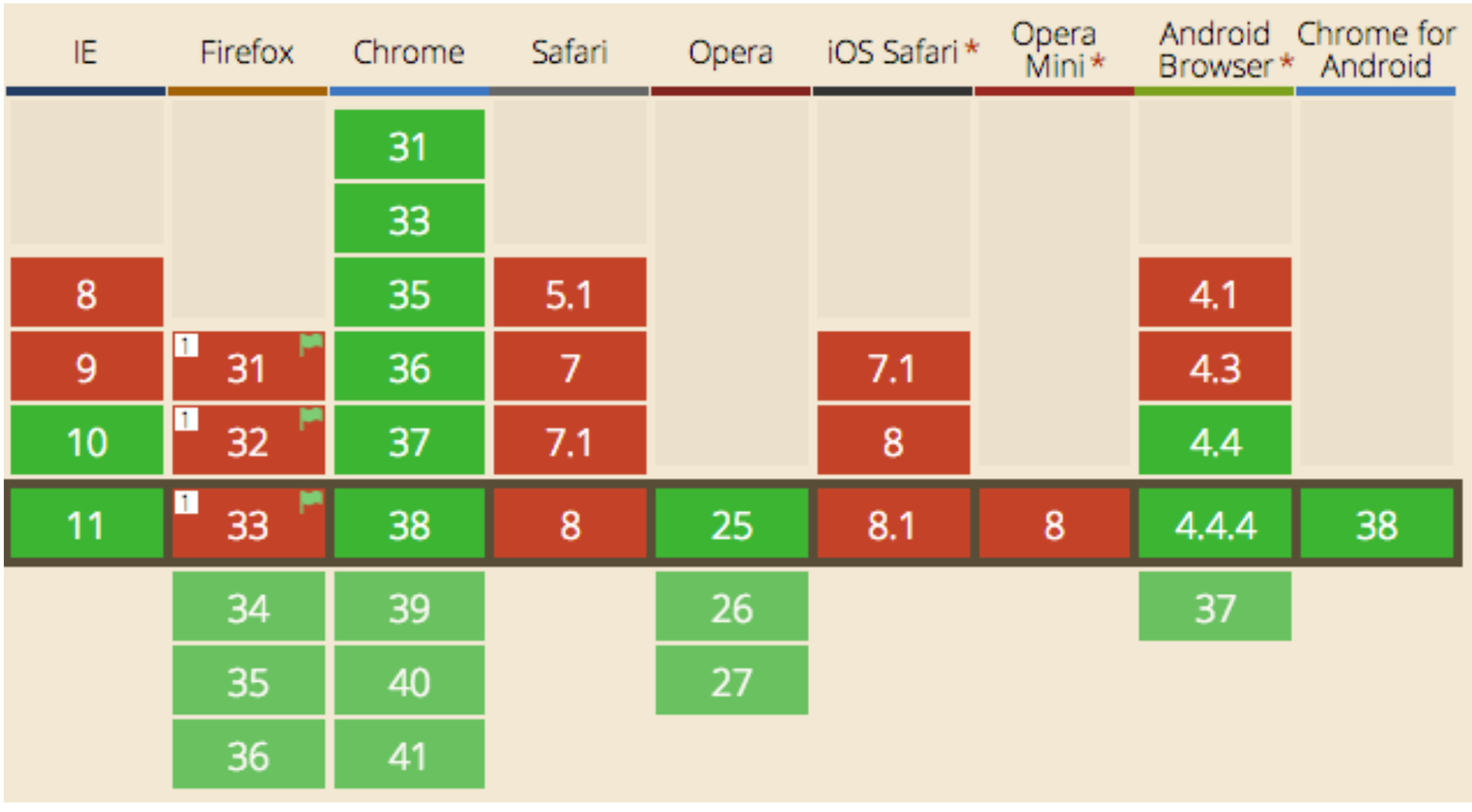
```
var serverTimeMillisUTC =
              new Date(document.getElementByID('time-holder'));
var localMillisUTC = new Date().getTime;
responseOffset =
       performance.timing.responseEnd -
       performance.timing.responseStart;
var offset =
   (serverTimeMillisUTC -  localMillisUTC) + responseOffset;

function getServerTime() {
  var date = new Date();
  date.setTime(date.getTime() + offset);
  return date;
}
```

Solution (Legit)

- Use Resource Timing

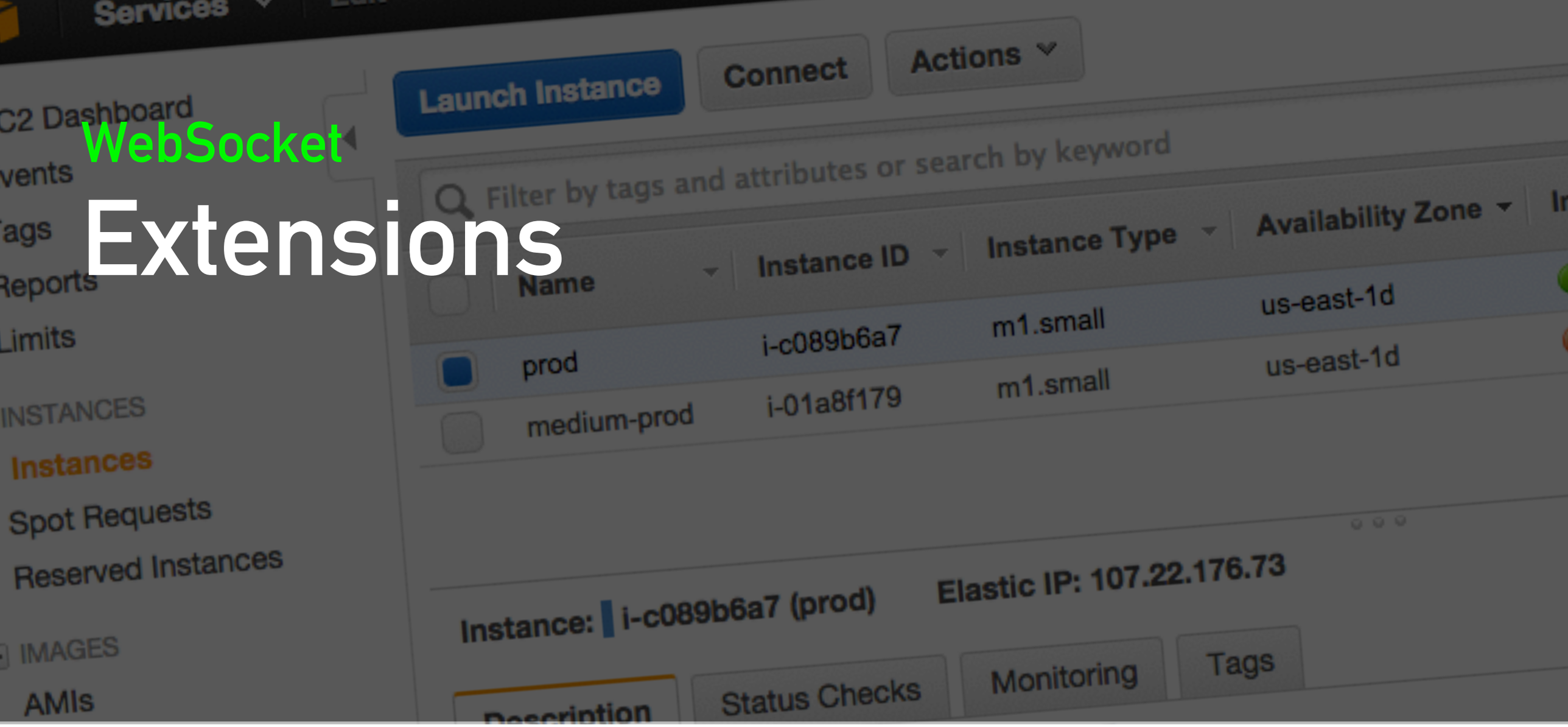
Resource Timing

- Browser Support



Resource Timing on AJAX

```
var serverTimeMillisUTC;
var oReq = new XMLHttpRequest();
oReq.onload = function(){
  var dateStr = oReq.getResponseHeader('Date');
  serverTimeMillisUTC =
                new Date(Date.parse(dateStr)).getTime();
};
oReq.open("HEAD", "/?foo123", false);
oReq.send();

function getServerTime() {
  window.performance.getEntriesByType(“resource").
        reduceRight(function(previousValue, currentValue, index, array) {
      if(currentValue.name.indexOf('foo123') > 0){
          return serverTimeMillisUTC + (currentValue.responseEnd -
                                        currentValue.responseStart);
      }
  })
}
```



WebSocket

Extensions

@wesleyhales

WS Multiplexing and Head-of-Line Blocking

- Frames from different messages can't be interleaved.
- A large message, even when split into multiple WebSocket frames, will block the delivery of frames associated with other messages.

WebSocket Extensions

- WebSocket specification allows for protocol extensions
- "Multiplexing Extension for WebSocket"
- "Compression Extensions for WebSocket”

WebSocket

Performance Checklist

@wesleyhales

Use secure WebSocket (WSS over TLS) for reliable deployments.

Use compression.

```
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
Sec-WebSocket-Key: /LqLDugB0+EK9Bnp6d3gqw==
Sec-WebSocket-Version: 13
```

Pay close attention to polyfill performance (if necessary).

Optimize binary payloads to minimize transfer size.

https://github.com/onslyde/onslyde/blob/master/js/deck/onslyde-1.0.0.deck.js#L1581

Consider compressing UTF-8 content to minimize transfer size.

https://www.igvita.com/2013/11/27/configuring-and-optimizing-websocket-compression/

Split large application messages to avoid head-of-line blocking.

Use subprotocol negotiation

http://chimera.labs.oreilly.com/books/1230000000545/ch17.html#_subprotocol_negotiation

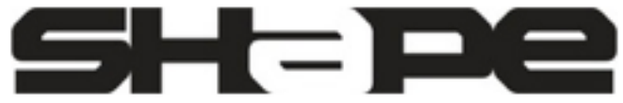
Parting Words

HTTP2, SPDY, SSE, and WebSocket are all different. They could be combined.

Are you willing to polyfill your WebSocket Implementation?

Thanks!

@wesleyhales



shapesecurity.com