Delivering Optimized Images on the Modern Web

Velocity/Europe, Barcelona

November 17, 2014

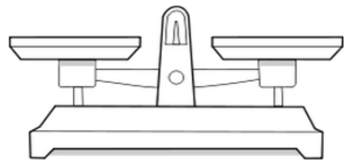
Joshua Marantz jmarantz@google.com

Google Make the Web Fast team

Balancing UX, Speed and Ease of Development

Desktop / Retina Tablet Users
Fast WIFI connection
Can Display large, dense images
Powerful CPUs
Long-lasting Batteries
Varying Browser Capabilities

Smartphone Users
High-latency connectivity
Relatively small screens
Slow CPUs, thin pipe to GPUs
Batteries Stretched Thin
Varying Browser Capabilities

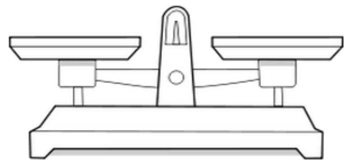


Website Developers
Limited Time --> Prioritize Content
User-Generated Content
Confusing Array of Changing Standards
Browsers Updating Constantly
Getting Up To Speed on “Responsive” Techniques

Balancing UX, Speed and Ease of Development

Desktop / Retina Tablet Users
Fast WIFI connection
Can Display large, dense images
Powerful CPUs
Long-lasting Batteries
Varying Browser Capabilities

Smartphone Users
High-latency connectivity
Relatively small screens
Slow CPUs, thin pipe to GPUs
Batteries Stretched Thin
Varying Browser Capabilities



Website Developers
Limited Time --> Prioritize Content
User-Generated Content
Confusing Array of Changing Standards
Browsers Updating Constantly
Getting Up To Speed on “Responsive” Techniques

Delivering A Great Mobile Web Experience

UX = Performance + Quality

Mobile Web Performance

Above-the-fold content delivered in one round trip, 15kb compressed

Ilya Grigorik: Optimizing the Critical Rendering Path for Instant Mobile Websites - Velocity SC - 2013

*This includes all critical JavaScript, CSS, and **Images***

→ ***inlined base64 low-resolution data URLs***

→ ***use the most compact format available on client***

Mobile Web Quality

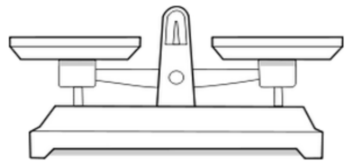
Deliver images with enough quality to make Retina displays look great

→ ***replace low-res images with high-res ones on zoom***

Balancing UX, Speed and Ease of Development

Desktop / Retina Tablet Users
Fast WIFI connection
Can Display large, dense images
Powerful CPUs
Long-lasting Batteries
Varying Browser Capabilities

Smartphone Users
High-latency connectivity
Relatively small screens
Slow CPUs, thin pipe to GPUs
Batteries Stretched Thin
Varying Browser Capabilities



Website Developers
Limited Time --> Prioritize Content
User-Generated Content
Confusing Array of Changing Standards
Browsers Updating Constantly
Getting Up To Speed on “Responsive” Techniques

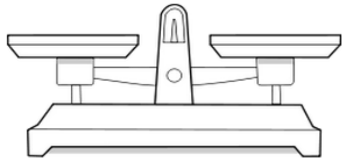
Delivering a great desktop / tablet experience

- Image-resolution sensitive to pixel density
- Image-resolution sensitive to zoom-level
- Maximize system performance using CDNs and proxy-caches

Balancing UX, Speed and Ease of Development

Desktop / Retina Tablet Users
Fast WIFI connection
Can Display large, dense images
Powerful CPUs
Long-lasting Batteries
Varying Browser Capabilities

Smartphone Users
High-latency connectivity
Relatively small screens
Slow CPUs, thin pipe to GPUs
Batteries Stretched Thin
Varying Browser Capabilities



Website Developers
Limited Time --> Prioritize Content
User-Generated Content
Confusing Array of Changing Standards
Browsers Updating Constantly
Getting Up To Speed on “Responsive” Techniques

Website Developers

Most Website developers are not **performance** experts

Most Website developers are not **mobile** experts

Most Website developers are not **responsive-design** experts

Most Website developers don’t come to Velocity

Many websites are not owned or maintained by their developers

- contractors, sons-in-law, nephews...

Web Developers View of Web Images

A simple `<img src=....>` tag is the easiest way to put images on your site

Consider sites with lots of web pages

Each of those have lots of images

Getting simple ‘img’ tags right may be all we can get in the wild

But by itself it is not going to make all of your users happy.

The Changing Landscape: Image Formats

Lossy: JPEG, **Webp, JPEG-XR**

Lossless: PNG, **Webp, JPEG-XR**, GIF, SVG

Fewer bytes is generally better

- modulo decompression cost
- modulo memory of decompressed image

Goal: get the whole page to 15k uncompressed, 1 round trip

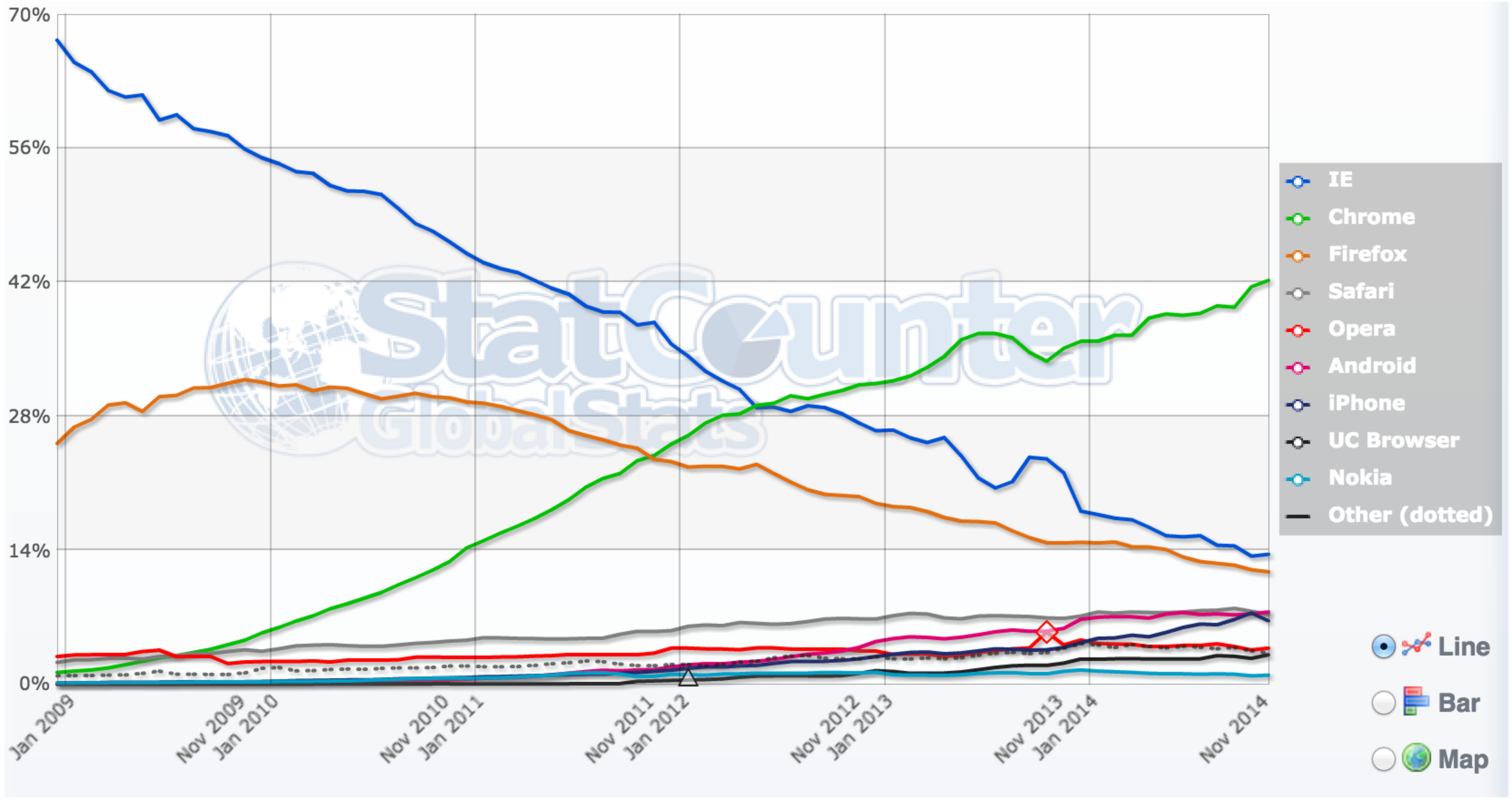
- we need to squeeze critical images
- need to serve them as base64 data URLs

http://goldfishforthought.blogspot.com/2010/10/comparison-webp-jpeg-and-jpeg-xr.html

https://developers.google.com/speed/webp/docs/webp_study

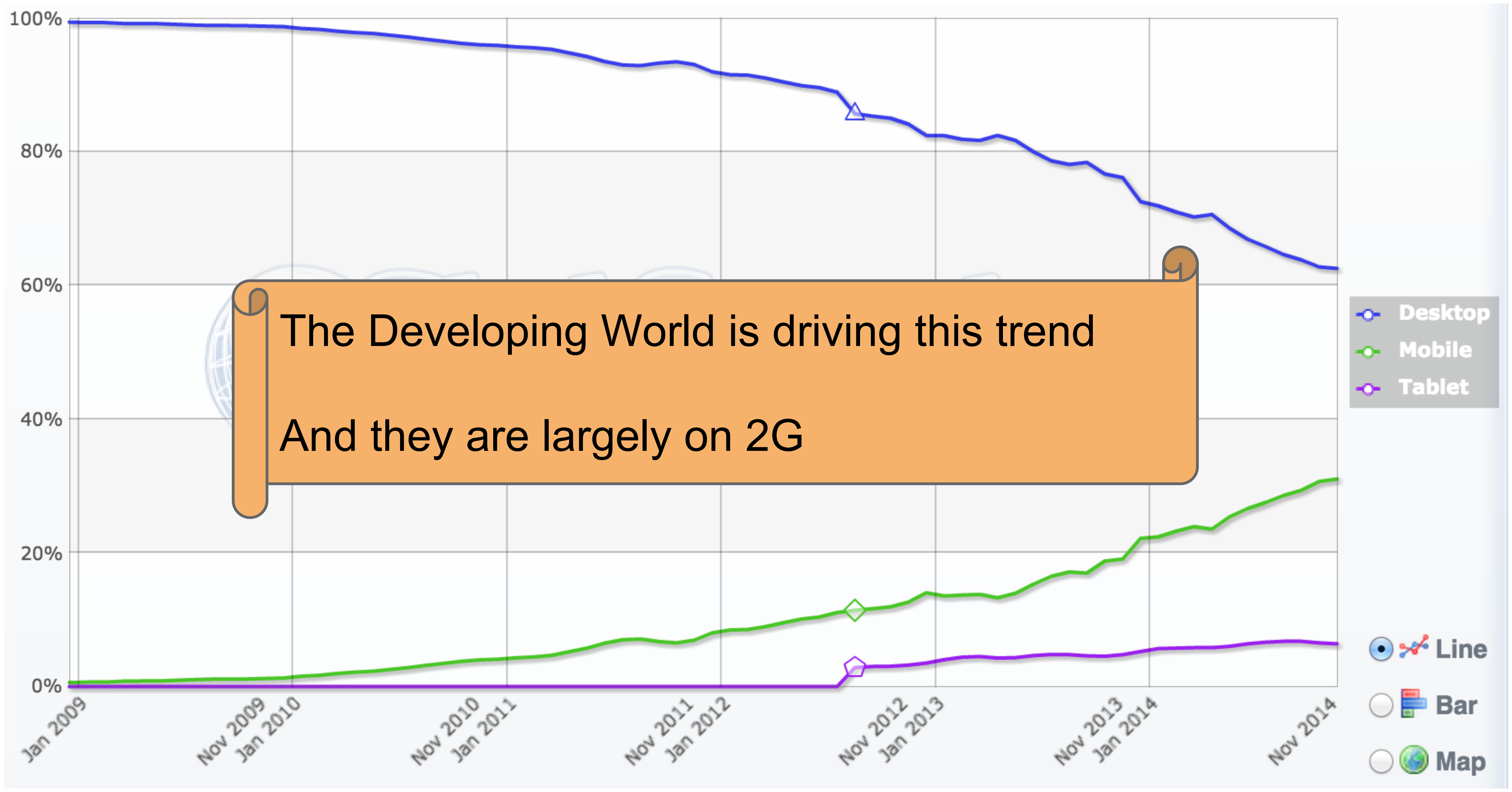
http://www.smashingmagazine.com/2014/05/14/responsive-images-done-right-guide-picture-srcset/

The Changing Landscape: Overall Browser Share



http://gs.statcounter.com/#all-browser-ww-monthly-200812-201411

The Changing Landscape: The Ascent Of Mobile

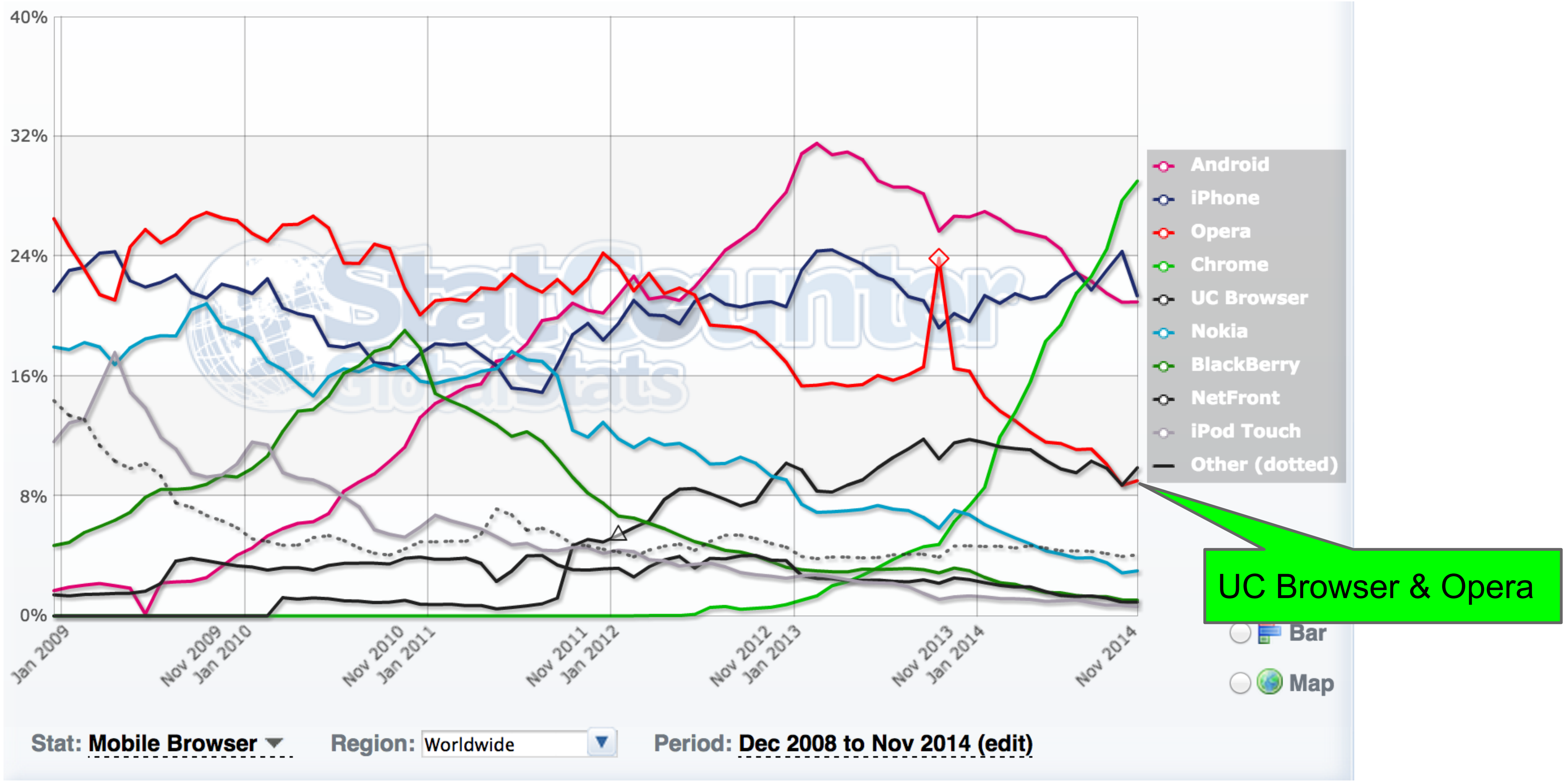


The Developing World is driving this trend

And they are largely on 2G

http://gs.statcounter.com/#desktop+mobile+tablet-comparison-ww-monthly-200812-201411

The Changing Landscape: Mobile-only Browser Share



Stat: Mobile Browser Region: Worldwide Period: Dec 2008 to Nov 2014 (edit)

http://gs.statcounter.com/#mobile_browser-ww-monthly-200812-201411

The Changing Landscape: Browser Image Support, Nov ‘14

Browser	Mobile Market Share(*)	Desktop + Tablet Market Share	WebP	Jpeg-XR	PNG, Jpeg, Jpeg-Progressive, GIF
IE	0%	20%	no	Yes (IE9+)	Yes
Chrome	29%	48%	Lossy & Lossless	no	Yes
Android	21%	0%	Lossy	no	Yes
Opera	9%	1%	Lossy	no	Yes
UC Browser	10%	0%	?	?	Yes
Safari / iPhone	22%	11%	no	no	Yes
Nokia	9%	0%	no	no	Yes

19% of Mobile Browsers designed for low bandwidth (Opera + UC)

Browsers supporting advanced image formats
Desktop+Tablet: 69% (Chrome + IE + Opera)
Mobile: 59% (Android + Opera + Chrome)

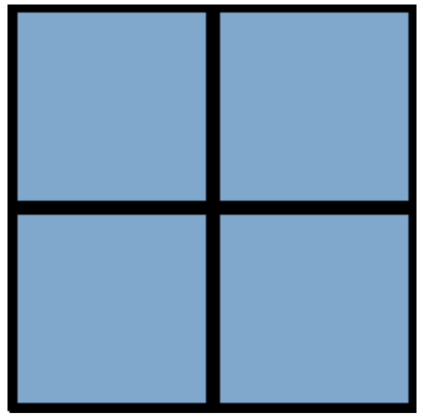
http://gs.statcounter.com/#all-br
http://gs.statcounter.com/#mobi

Changing Landscape: Devices

http://en.wikipedia.org/wiki/List_of_displays_by_pixel_density (+ zooming)

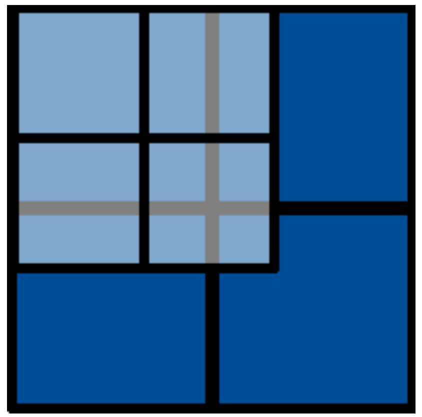
```
<img src="image-1x.jpg" width="500" height="500">
```

Old:



CSS pixel == device pixel

New:



CSS pixel ≠ device pixel

Recent Apple device resolutions

Model	Generations	Diagonal cm (in)	Resolution	PPCM (PPI)	CSS pixel ratio
iPad	1st gen, 2 gen	25 (9.7)	1024x768	52 (132)	1
iPad / iPad Air			536	104 (264)	2
iPad Mini				64 (163)	1
iPad Mini	2nd gen	20 (7.9)	2048x1536	128 (326)	2
iPhone 3GS	3 / 3rd gen	8.9 (3.5)	320x480	64 (163)	1
iPhone 4	4 / 4th gen	8.9 (3.5)	960x640	128 (326)	2
MBP 13”	2009-2012	34 (13.3)	1280x800	44 (113)	1
MBP 13” Retina	2012-2013	34 (13.3)	2560x1600	89 (227)	2

People who buy Retina iPads have beautiful devices and want to see beautiful images on them

Challenges with simple <img src=> tag

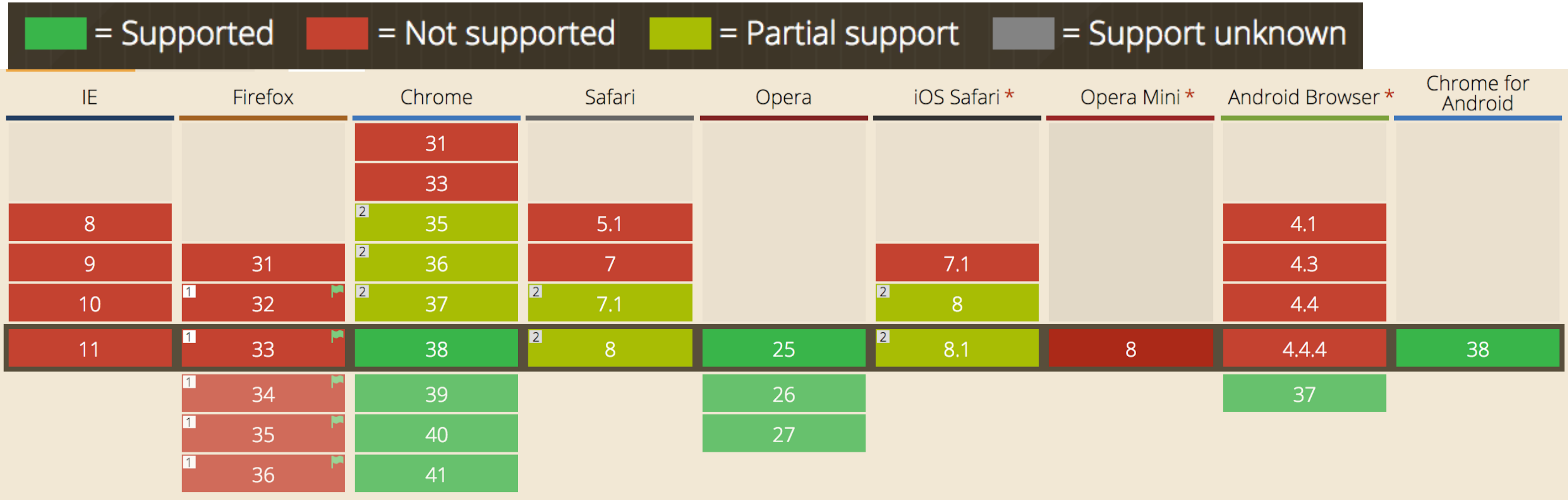
All parameters for optimization not known in one place

The browser doesn’t know the image dimensions are quality before downloading

The server does not know the pixel density or device performance based in the request

Thus was born srcset: http://www.w3.org/html/wg/drafts/srcset/w3c-srcset/ (published June 26, 2014)

srcset browser support



🟩 = Supported 🟥 = Not supported 🟨 = Partial support ⬜ = Support unknown

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		🟥 31						
		🟥 33						
🟥 8		🟨 [2] 35	🟥 5.1				🟥 4.1	
🟥 9	🟥 31	🟨 [2] 36	🟥 7		🟥 7.1		🟥 4.3	
🟥 10	🟥 [1] 32	🟨 [2] 37	🟨 [2] 7.1		🟨 [2] 8		🟥 4.4	
🟥 11	🟥 [1] 33	🟩 38	🟨 [2] 8	🟩 25	🟨 [2] 8.1	🟥 8	🟥 4.4.4	🟩 38
	🟥 [1] 34	🟩 39		🟩 26			🟩 37	
	🟥 [1] 35	🟩 40		🟩 27				
	🟥 [1] 36	🟩 41						

Fortunately, Graceful fallback to src= attribute means you can use srcset and it will not harm other browsers

http://caniuse.com/#feat=srcset

Don’t forget the users!

Who are your users?
How are they connected?
What browsers do they use?
Do you care...

- whether your site is beautiful on a modern tablet / computer?
- if your site works properly on a variety of older browsers
- if your site performs well on a mobile phone on Edge/3G/LTE?

Anatomy of a reasonable Image tag

```
<img src="image-1x.jpg"
     srcset="image-2x.jpg 2x, image-4x.jpg 4x">
```

Available in Chrome, Safari, & Firefox

Extra attributes will be safely ignored by other browsers until they add this capability

Delivers the right quality images to most modern phones and tablets

srcset specification flexibility

The srcset attribute allows authors to provide a set of images to handle graphical displays of varying dimensions and pixel densities.

The attribute essentially takes a comma-separated list of URLs each with one or more descriptors giving the maximum viewport dimensions and pixel density allowed to use the image. From the available options, the user agent then picks the most appropriate image. **If the viewport dimensions or pixel density changes, the user agent can replace the image data with a new image on the fly.**

To specify an image, give first a URL, then one or more descriptors of the form 100w or 2x, where "100w" means "maximum viewport width of 100 CSS pixels" and "2x" means "maximum pixel density of 2 device pixels per CSS pixel".

Srcset Observations in Chrome & Safari

	Chrome	Safari
Selects images on Page Load	Yes	Yes
Selects images on Control-+	No	No
‘onresize’ on Control-+	Yes	Yes
‘onresize’ on Pinch-Zoom	No	Yes
‘ontouchstart’	Yes	Yes

srcset helps Responsive Design, but is not dynamic by default

Dynamic Image Refinement “polyfills”

```
function upgradeImage(img) {
  var hiRes = getSourceForCurrentResolution(img);
  if (img.src != hiRes) {
    var hiResImg = new Image();
    hiResImg.onload = function() { img.src = hiRes; }
    hiResImg.src = hiRes;
  }
}
```

Control-+ for Desktop devices	Pinch-Zoom for Mobile Devices

window.addEventListener(‘resize’, function() {
  foreach img { upgradeImage(img); }
});
	foreach img {
	  img.addEventListener(‘touchstart’, function() {
	    upgradeImage(img);
	  });
	}

Why Add This JS? Can’t Browsers Do This?

Game Plan:

Deliver a low-res view of the web page in 15k bytes compressed

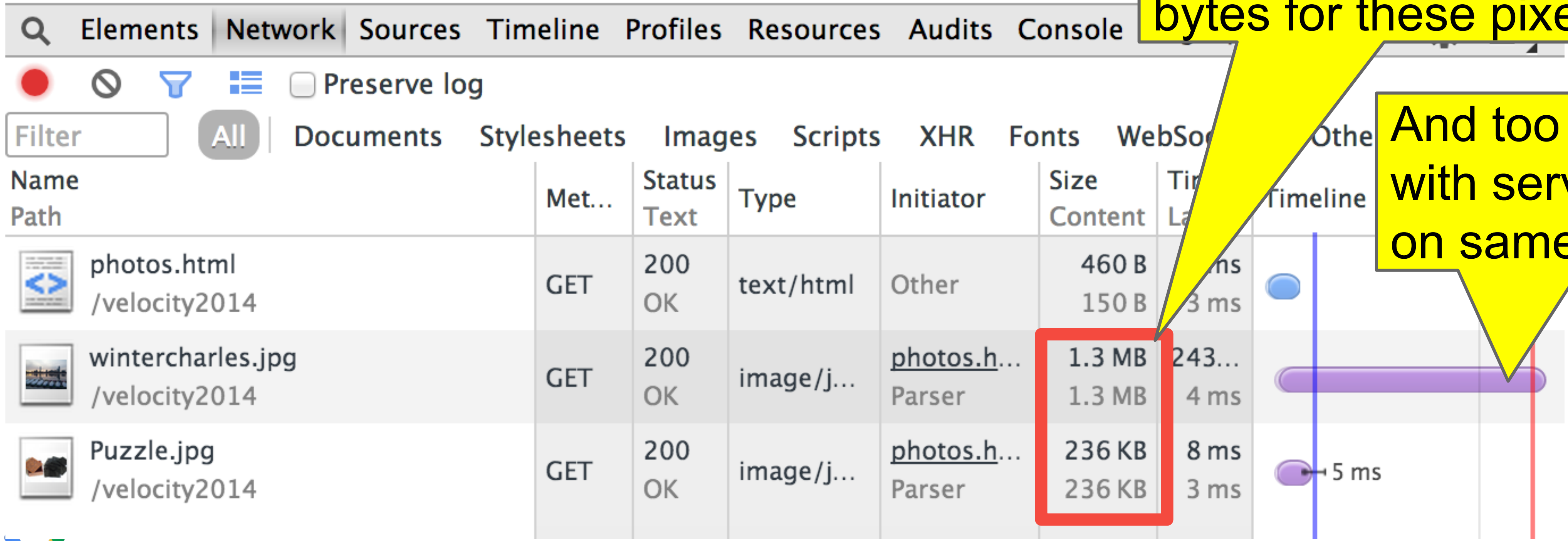
    no external resources

Fill in detail dynamically as user expresses interest

srcset is not (AFAICT) intended for this, but it could be

Resized image with simple img tag

```
<html><head/><body>
<img src="wintercharles.jpg" width="147" height="110"></img>
<img src="Puzzle.jpg" width="147" height="110"></img>
</body></html>
```



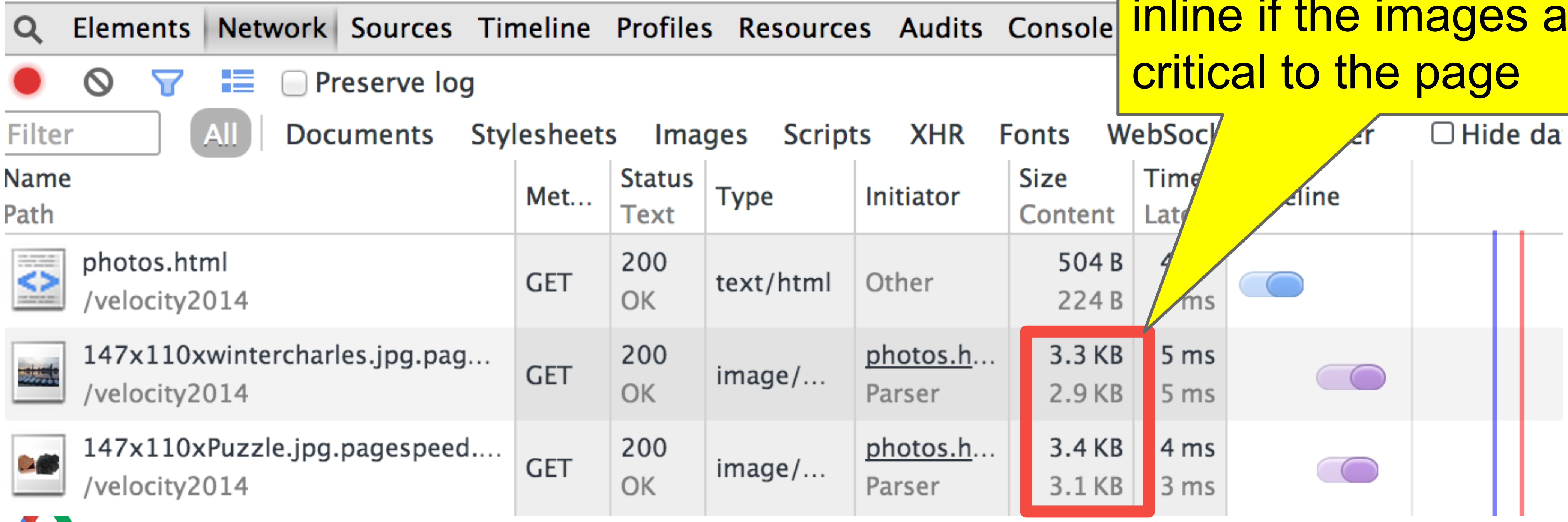
Name / Path	Met...	Status / Text	Type	Initiator	Size / Content	Ti / La	Timeline
photos.html /velocity2014	GET	200 OK	text/html	Other	460 B / 150 B	ms / 3 ms	
wintercharles.jpg /velocity2014	GET	200 OK	image/j...	photos.h... Parser	1.3 MB / 1.3 MB	243... / 4 ms	
Puzzle.jpg /velocity2014	GET	200 OK	image/j...	photos.h... Parser	236 KB / 236 KB	8 ms / 3 ms	5 ms

1.3 meg is way too many bytes for these pixels

And too slow, even with server & client on same machine

Optimized/sized images are 90% smaller & look great...

```
<html><head/><body>
<img src="147x110xwintercharles.jpg.pagespeed.ic.u8jO388Uqr.webp"
width="147" height="110"></img>
<img src="147x110xPuzzle.jpg.pagespeed.ic.SQCe96qVx2.webp"
width="147" height="110"></img>
</body></html>
```



Elements Network Sources Timeline Profiles Resources Audits Console

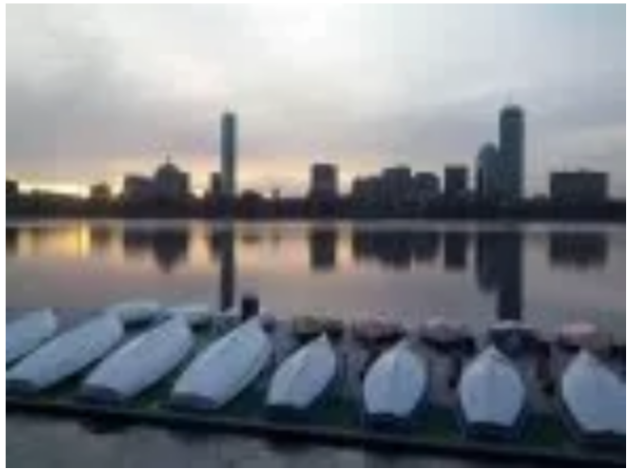
Preserve log

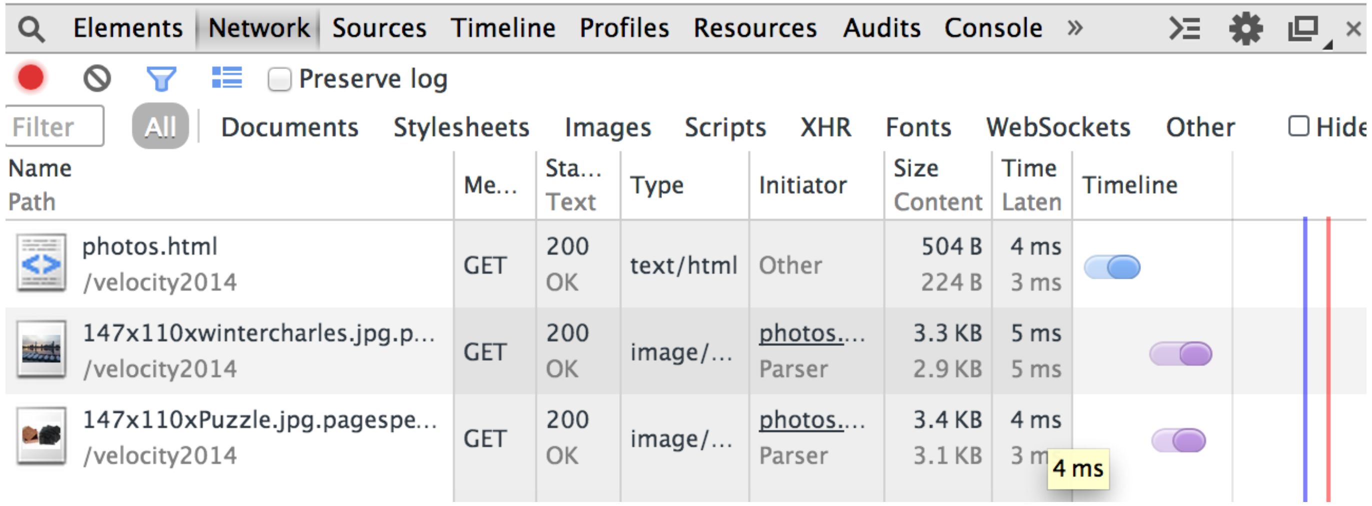
Filter All Documents Stylesheets Images Scripts XHR Fonts WebSoc... Hide da...

Name Path	Met...	Status Text	Type	Initiator	Size Content	Time Lat...
photos.html /velocity2014	GET	200 OK	text/html	Other	504 B 224 B	4... ...ms
147x110xwintercharles.jpg.pag... /velocity2014	GET	200 OK	image/...	photos.h... Parser	3.3 KB 2.9 KB	5 ms 5 ms
147x110xPuzzle.jpg.pagespeed.... /velocity2014	GET	200 OK	image/...	photos.h... Parser	3.4 KB 3.1 KB	4 ms 3 ms

3.3 K: small enough to inline if the images are critical to the page

...until you zoom in; browser scaling looks terrible

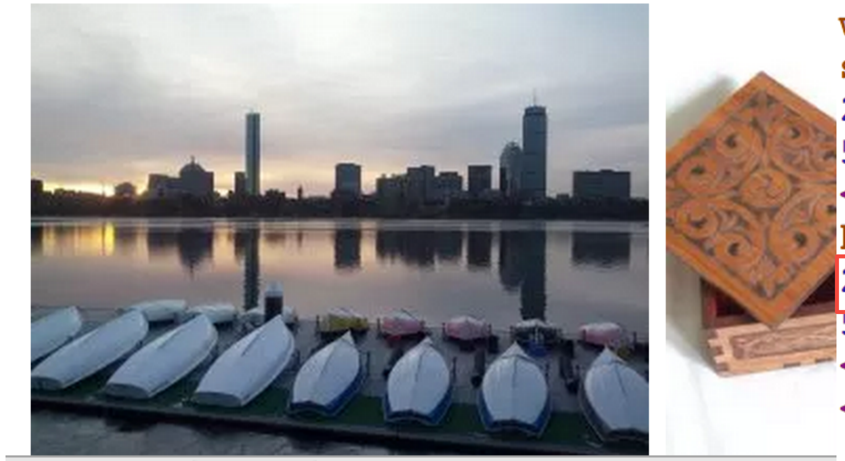




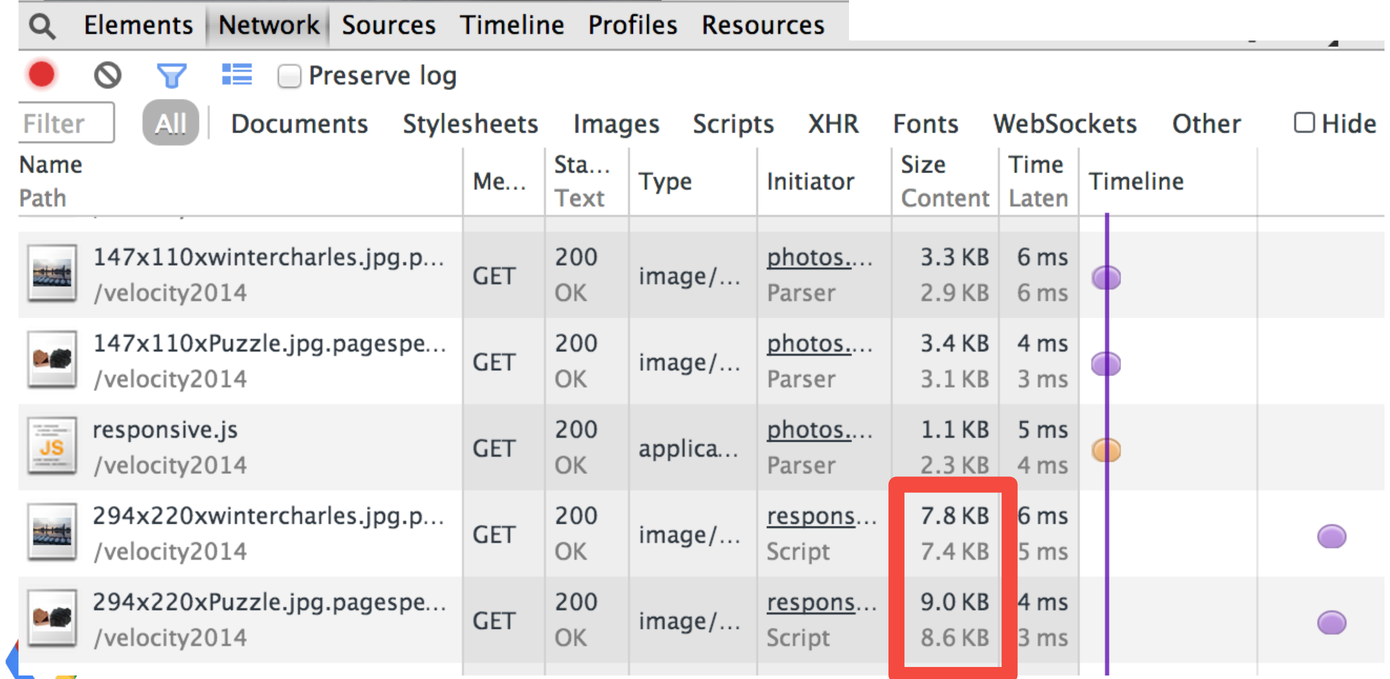
Name / Path	Me…	Sta… / Text	Type	Initiator	Size / Content	Time / Laten
photos.html /velocity2014	GET	200 OK	text/html	Other	504 B 224 B	4 ms 3 ms
147x110xwintercharles.jpg.p… /velocity2014	GET	200 OK	image/…	photos.… Parser	3.3 KB 2.9 KB	5 ms 5 ms
147x110xPuzzle.jpg.pagespe… /velocity2014	GET	200 OK	image/…	photos.… Parser	3.4 KB 3.1 KB	4 ms 3 ms

4 ms

Add “srcset” and polyfill for Zoom Sensitivity



```
<html><head/><body>
<img src="147x110xwintercharles.jpg.pagespeed.ic.u8jO388Uqr.webp"
width="147" height="110"
srcset="147x110xwintercharles.jpg.pagespeed.ic.u8jO388Uqr.webp 2x,
294x220xwintercharles.jpg.pagespeed.ic.TsN8kStZoG.webp 4x,
588x440xwintercharles.jpg.pagespeed.ic.gqGnlvZ164.webp 8x"></img>
<img src="147x110xPuzzle.jpg.pagespeed.ic.SQCe96qVx2.webp" width="147"
height="110" srcset="147x110xPuzzle.jpg.pagespeed.ic.SQCe96qVx2.webp 2x,
294x220xPuzzle.jpg.pagespeed.ic.TNrvmneew0.webp 4x,
588x440xPuzzle.jpg.pagespeed.ic.jGQVaWuUtJ.webp 8x"></img>
<script src="/velocity2014/responsive.js" type="application/javascript">
</script></body></html>
```



Name / Path	Me...	Sta... / Text	Type	Initiator	Size / Content	Time / Laten	Timeline
147x110xwintercharles.jpg.p... /velocity2014	GET	200 OK	image/...	photos.... Parser	3.3 KB 2.9 KB	6 ms 6 ms	
147x110xPuzzle.jpg.pagespe... /velocity2014	GET	200 OK	image/...	photos.... Parser	3.4 KB 3.1 KB	4 ms 3 ms	
responsive.js /velocity2014	GET	200 OK	applica...	photos.... Parser	1.1 KB 2.3 KB	5 ms 4 ms	
294x220xwintercharles.jpg.p... /velocity2014	GET	200 OK	image/...	respons... Script	7.8 KB 7.4 KB	6 ms 5 ms	
294x220xPuzzle.jpg.pagespe... /velocity2014	GET	200 OK	image/...	respons... Script	9.0 KB 8.6 KB	4 ms 3 ms	

1.1k JavaScript “polyfill” replaces lower-res images with higher-res images when user zooms in with Control-+.
+ Enables ‘srcset’ on all browsers with JS!

Demo

Delivering the best image format in markup

<picture> and <source> can be combined to select webp images to compatible browsers (Chrome, Opera)

https://html.spec.whatwg.org/multipage/embedded-content.html#introduction-3:attr-picture-source-type

```
<picture>
  <source srcset="/uploads/100-marie-lloyd.webp" type="image/webp">
  <source srcset="/uploads/100-marie-lloyd.jxr" type="image/vnd.ms-photo">
  <img src="/uploads/100-marie-lloyd.jpg" alt="" width="100" height="150">
</picture>
```

Delivering the best image format in the server

Change the URL

- Web Performance Optimization

  mod_pagespeed etc, Akamai Aqua Ion, Radware
- Embedded in Web App (Facebook, Google+)
- Risks: subtly interfering with JavaScript introspection

http://mzoom.org/588x440xwintercharles.jpg.pagespeed.ic.iVgBCAHftt.webp

Leave the URL alone

- Serve webp but to .jpg URL
- Use Vary:Accept
- Proxy-caches & CDNs need to cache responses with Vary:Accept

Hints From The HTTP Request

	User-Agent	**Accept**
IE 11	Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv: 11.0) like Gecko	text/html, application/xhtml+xml, */*
Chrome	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35. 0.1916.153 Safari/537.36	text/html,application/xhtml+xml, application/xml;q=0.9,**image/webp**,*/*; q=0.8
Firefox	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:30.0) Gecko/20100101 Firefox/30.0	text/html,application/xhtml+xml, application/xml;q=0.9,*/*;q=0.8
Safari	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.76.4	text/html,application/xhtml+xml, application/xml;q=0.9,*/*;q=0.8
Opera	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35. 0.1916.153 Safari/537.36 OPR/22.0.1471.70	text/html,application/xhtml+xml, application/xml;q=0.9,**image/webp**,*/*; q=0.8

Content Delivery Networks

Proxy Caches situated closer to users than your origin

Maps URLs + X? -> Content (Content-Type)

But what if you want to serve different content to modern browsers?

Vary:Accept

Vary:User-Agent

One concern: Proxy Cache Fragmentation

Local Proxy Cache

Handles cached requests much faster than typical origin servers

Varnish
Squid
Apache Traffic Server
nginx

Same issues with CDNs: Vary handling

Getting the Details Right: Internet Explorer

- Add Vary:Accept in your response headers
  - IE will have to validate browser cache hits (!!)

- Omit Vary:Accept in your response headers
  - Proxy caches and CDNs can deliver the wrong content, so use Cache-Control:private
  - Strip Vary:Accept at the last point you control: proxy cache or CDN

```
ModPagespeedEnableFilters in_place_optimize_for_browser
ModPagespeedPrivateNotVaryForIE off
```

https://developers.google.com/speed/pagespeed/module/system#in_place_optimize_for_browser

Getting the Details Right: Amazon EC2

HTTP Requests for origin images from EC2 are served with User-Agent + Accept headers stripped (!!!)

→ We must serve distinct URLs for webp, avoid optimizing in-place for sites using EC2 as a CDN

Making A Better Web

→ Evangelize web development best practices
→ Make faster browsers
→ Make better CDNs
→ Automate content generation to exploit browsers & CDNs

Delivering Optimized Images On The Modern Web

In 2014, browsers, CDNs, and WPO tools are make this easier

Effective use of srcset (with help of dynamic polyfills)

Try It Yourself: http://mzoom.org [through November 2014]

Sending superior image formats to modern browsers

Effective Use of CDNs and Proxy Caches