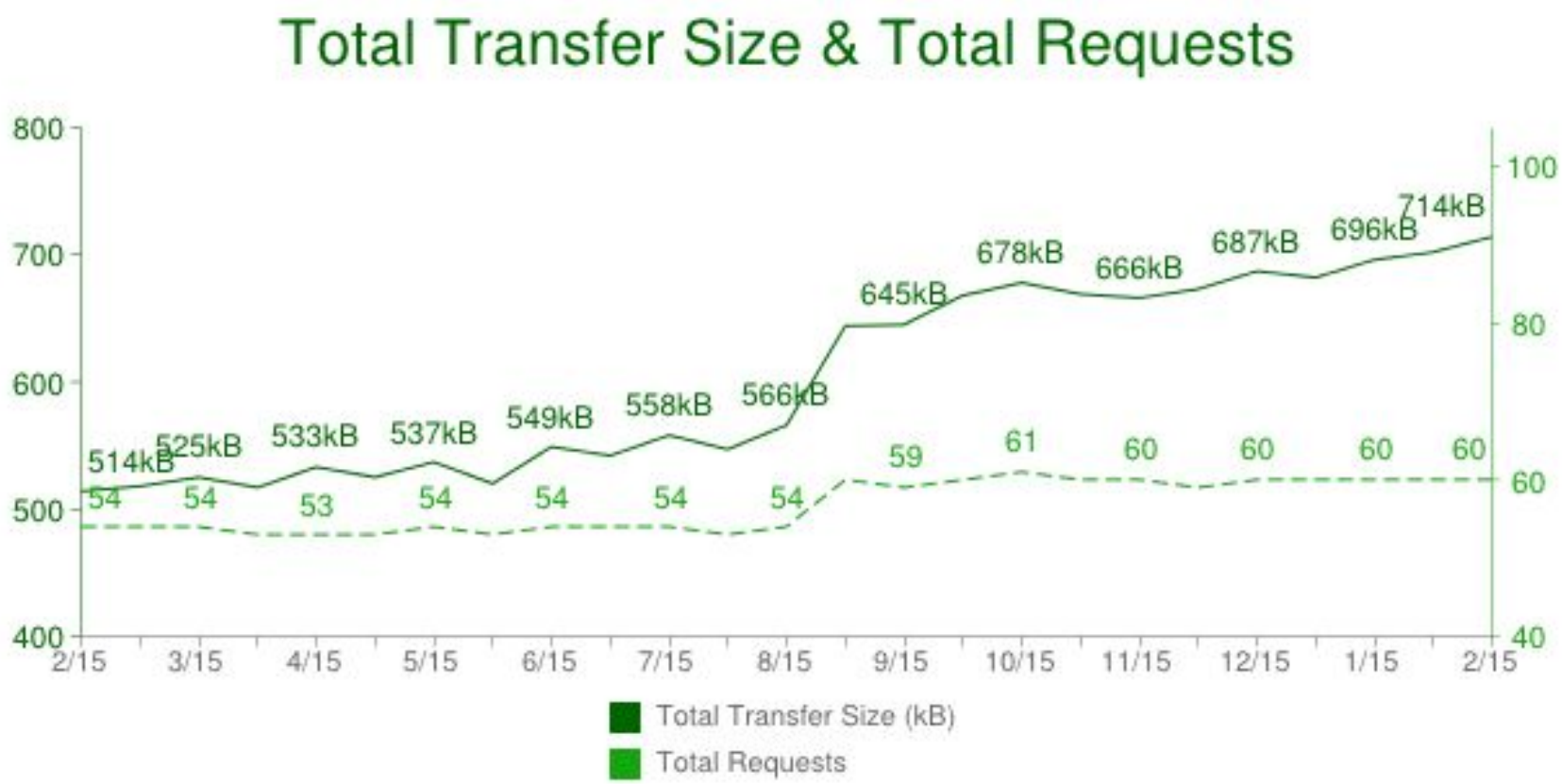
webp

enabling faster, smaller and more beautiful web

Stephen Konig skonig@google.com
Ilya Grigorik igrigorik@google.com

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



Content Type	Avg # of Requests	Avg size
HTML	6	39 kB
Images	**39**	**490 kB**
Javascript	**10**	**142 kB**
CSS	3	27 kB

69%

HTTP Archive - Mobile Trends (Feb, 2013)

It's a HiDPI world...

Tablet	dimension	device-width	px/inch
Nexus 7	3.75	603	**~ 160**
Kindle Fire	3.5	600	**~ 170**
iPad Mini	4.75	768	**~ 160**
PlayBook	3.54	600	**~ 170**
Galaxy 7'' (2nd gen)	3.31	600	**~ 180**
Macbook + Retina	15.4	2880	**~ 220**
Chromebook Pixel	12.85	2560	**~ 239**

HiDPI screens require 4x pixels!

Without careful optimization, this would increase the size of our pages by a huge margin - from 500KB to ~2000 KB!

Which image format should I use?

Wrong question! Instead, what if we had one format with all the benefits and features?

- Lossy and lossless compression
- Transparency (alpha channel)
- Great compression for photos
- Animation support
- Metadata
- Color profiles
- ....

That's WebP!

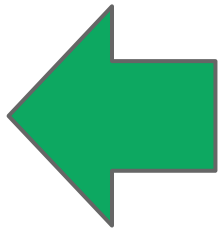
Brief history of WebP...

- **WebM** video format uses VP8 video codec
- **WebP** is derived from VP8, essentially a key frame...

- ***#protip:*** *great [GDL episode on WebM format]*

Brief history of WebP...

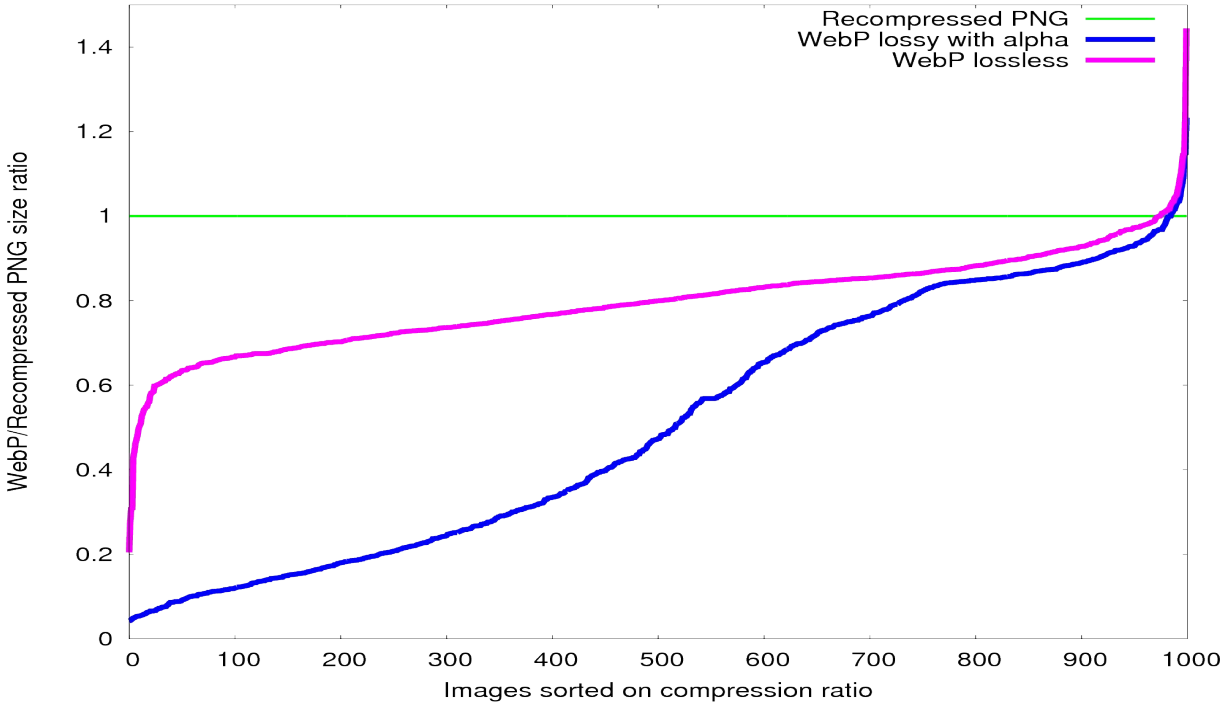
- **Initial release (2010)**
  - Lossy compression for true-color graphics
- **October, 2011**
  - Color profile support
  - XMP metadata
- **August, 2012**
  - Lossless compression support
  - Transparency (alpha channel) support
- **WIP + future...**
  - Animation + metadata
  - Encoding performance
  - Better support for ARM and mobile
  - Layer support (3D images) + high color depth images (> 8 bits)

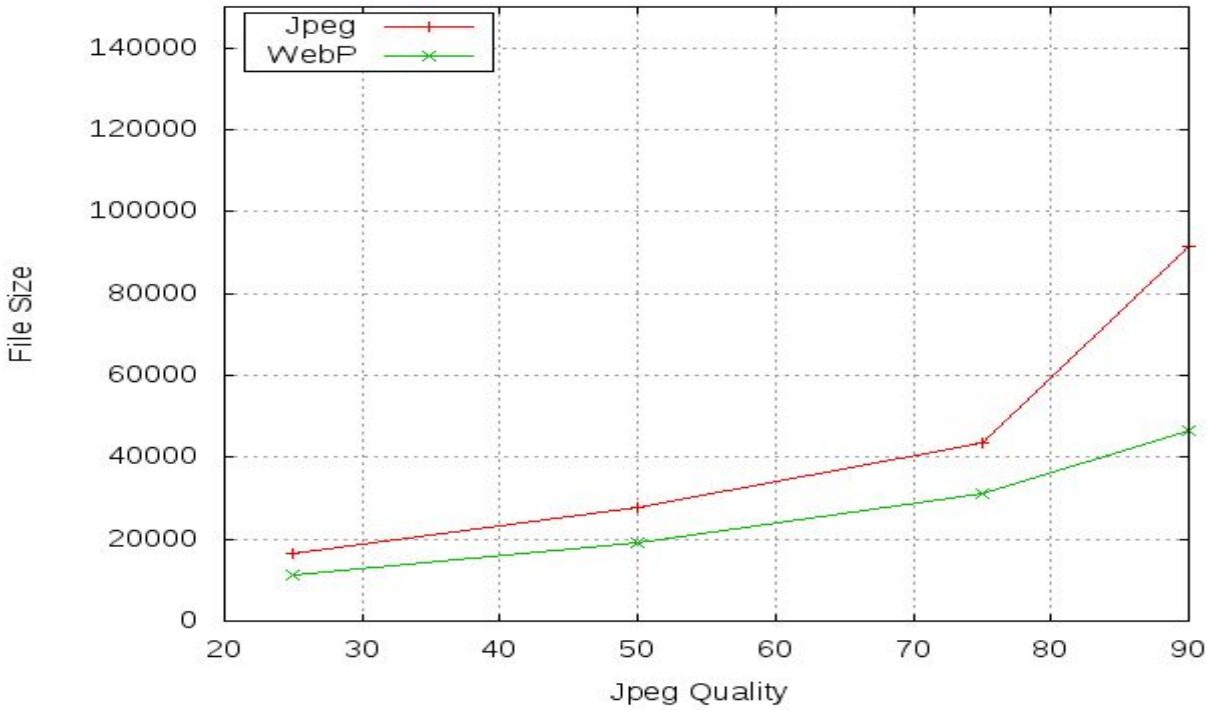


Now a viable alternative and replacement to *JPEG, PNG*

WebP *vs.* JPEG and PNG

- **30%** file size reduction over JPEG
- **up to 80%** file size reduction over PNG
- For JPEG gains increase at higher quality levels





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

Lunch == Free, or is it?

- **Better compression == *more* CPU cycles**

- Software performance today...
  - **Encoding:** 10x over JPEG
  - **Decoding:** 1.4x over JPEG

- High encoding cost may be a limitation for use cases where images are generated dynamically
  - ex, Google Maps

- Bandwidth savings **can** *(and in practice, so far... do)* outweigh the extra CPU time

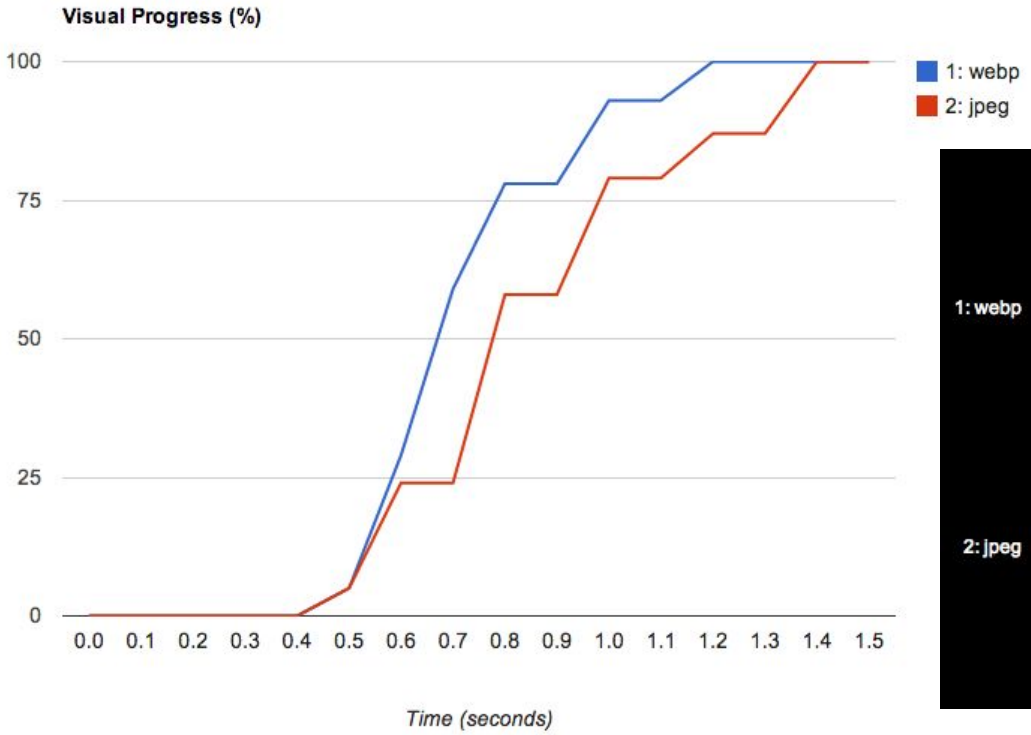
- Many users are on metered data plans
  - Data is expensive - literally.
  - E.g. Chrome data compression proxy!

WebP performance will improve with further optimizations, plus hardware support.

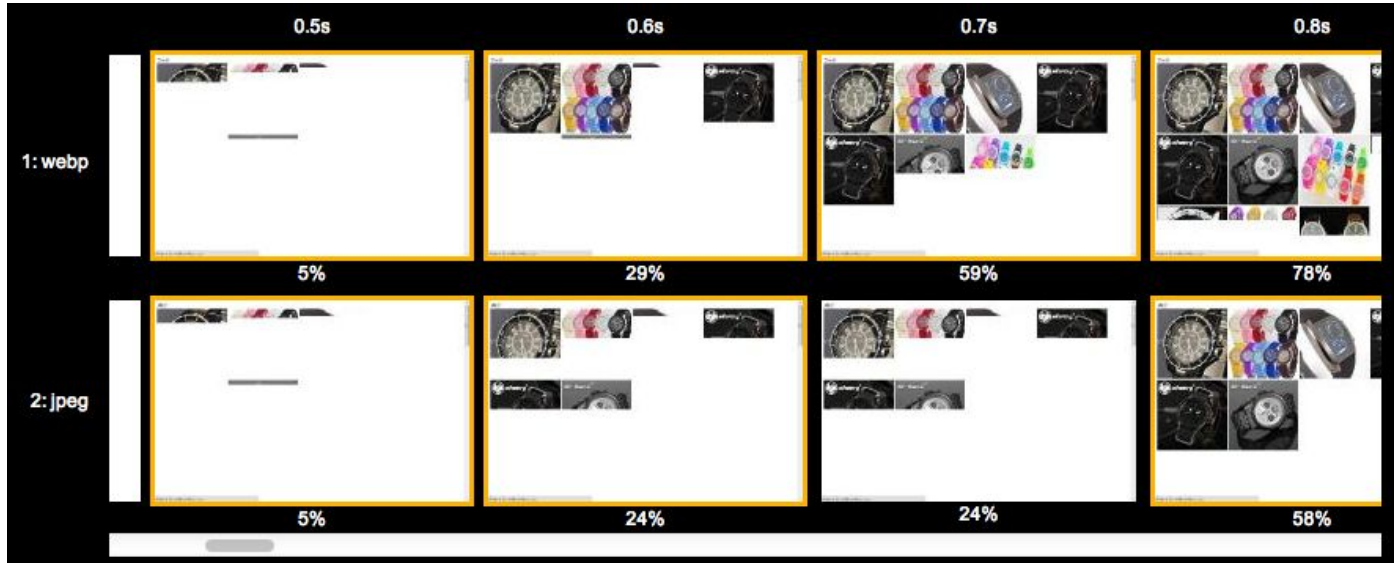
"A picture is worth a thousand words" - Ebay tech blog

"This test from webpagetest.org compares the page load time of WebP vs. JPEG. The test has one page with 50 images in the WebP format, and another page with the same 50 images in the JPEG format. ***Because the WebP page had to download fewer bytes (474,484 vs. 757,228), it completes loading much earlier compared to the JPEG page.***

If you track your web site's browser usage stats and find that Chrome/Opera users are a sizable chunk, using WebP images will improve the page load time for these users."



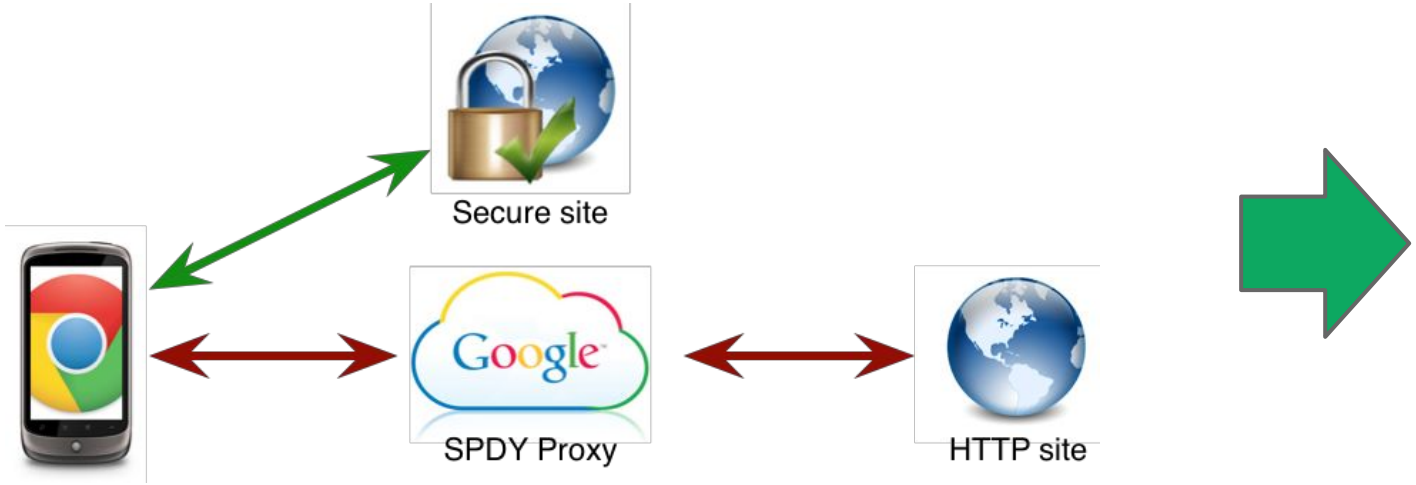
Faster "time to glass" with WebP!



Data compression proxy in Chrome for Android *(Dev Preview)*

- Original content **>** PageSpeed **>** **WebP** **>** your device

- Early tests show 50% data compression improvement, and much faster load times!
  a. Download Chrome Beta on Play Store
  b. Enable **"data compression proxy"** in **chrome://flags**!



Bandwidth Usage

	Session	Total
Original (KB)	172915	172915
Received (KB)	64805	64805
Savings (KB)	108110	108110
Savings (%)	62	62

https://developers.google.com/chrome/mobile/docs/data-compression

State of WebP adoption today

	Android	iOS	Chrome	Opera	IE	Safari	Firefox
WebP	4.x.x+	Library	✔	✔	Chrome Frame or JS	Plugin or JS	WIP Patch or JS

- 2010-2012: focus on feature support + performance.
- ***2013+ … focus on adoption and deployment!***

- **Chrome**, **Opera**, and working closely with Firefox team…
- 3rd party plugins for **Safari**
- Chrome Frame for **Internet Explorer**
- JavaScript decoder fallback

- **Android** lossy (ICS+) and lossless (JB+)
- **iOS** native apps via libraries

State of WebP adoption today

All major Google properties have, or in the process of investigating WebP support:

- Chrome Web Store - *saving TB's of traffic per day! Plus GMail, Drive, Picasa, Instant Previews, Google+, Google Play Magazines, Shopping, Google Play, Image Search, YouTube & StreetView*

3rd party site adoption:

- PageSpeed Service + mod_pagespeed + ngx_pagespeed == ***300,000+ WebP sites***
- Torbit, EdgeCast + other commercial FEO products
- ...

Automation FTW!

http://en.wikipedia.org/wiki/WebP#Support

Tooling and deployment

let's take a hands on look at what's available...

How do I create a WebP file?

- Download WebP converter (Linux, OSX, Windows)
  - `cwebp -quality 80 image.png -o image.webp`
  - `dwebp image.webp -o image.png`

- Download WebP Codec for Windows (Photo Viewer, Explorer, Office 2010+, ...)
- Download Photoshop plugin (by Telegraphics)
- Download GIMP plugin
- ImageMagic, Pixelmator, XnView, IrfanView, GDAL have **native support** for WebP

- Java, .NET, Flash, Python, Ruby, PHP bindings available to libwebp...

- img2webp.net online tool

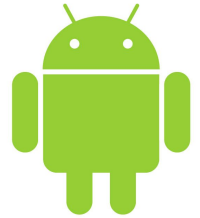
Check here for more

http://en.wikipedia.org/wiki/WebP#Support

Android + iOS

simple, easy, and big wins…

Android



- Native support for Android 4.0+

```
static {
    System.loadLibrary("webp");
}

private Bitmap webpToBitmap(byte[] encoded) {

  int[] width = new int[] { 0 };
  int[] height = new int[] { 0 };
  byte[] decoded = libwebp.WebPDecodeARGB(encoded, encoded.length, width, height);

  int[] pixels = new int[decoded.length / 4];
  ByteBuffer.wrap(decoded).asIntBuffer().get(pixels);

  return Bitmap.createBitmap(pixels, width[0], height[0], Bitmap.Config.ARGB_8888);
}
```

- webp-android-backport for Android <4.0

iOS

- Download and compile libwebp, add **WebP.framework** to your project...

```
// Get the current version of the WebP decoder
int rc = WebPGetDecoderVersion();

NSLog(@"Version: %d", rc);

// Get the width and height of the selected WebP image
int width = 0;
int height = 0;
WebPGetInfo([myData bytes], [myData length], &width, &height);

NSLog(@"Image Width: %d Image Height: %d", width, height);

// Decode the WebP image data into a RGBA value array
uint8_t *data = WebPDecodeRGBA([myData bytes], [myData length], &width, &height);
```

- Walkthrough tutorial
- WebP-iOS-example demo application

Deploying WebP on the web...

Lots of user agents, mixed support, requires a bit more work...

awesome.webp

User-Agent

(Chrome, FF, IE, ...)

Server Detection

User-Agent sniffing

Custom HTML

Cache-Control: private

work well together

Client Detection

JavaScript

Inject <img> via JS

Extra latency for img fetch

Client-side detection

- Use modernizr, or use the 1 line WebP detect function...

```
<script src="modernizr.min.js"></script>

<script>
  if (Modernizr.webp) {

    var webpImg = document.createElement("img");
    webpImg.setAttribute('src', '/awesome.webp');
    webpImg.setAttribute('alt', 'na');
    document.body.appendChild(webpImg);

  } else {
    // Fallback to non-webp, or load a JS decoder:
    // webpjs-0.0.2.min.js / webpjs-0.0.2.swf
  }
</script>
```

+ Bullet proof, custom URLs for .webp files (cache friendly), easy fallback for all clients
- Must wait for JS execution to schedule image downloads

http://webpjs.appspot.com/

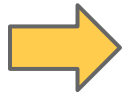
Server-side detection

- Serve different HTML based on **User-Agent** string

Cache-Control: private

```
<html>
 ...
 <img src="awesome.webp">
 ...
<html>
```

```
<html>
 ...
 <img src="awesome.jpg">
 ...
<html>
```



Or inject a polyfill library!

\+ No extra latency overhead, automated by the server (e.g. PageSpeed)

\- Returned HTML should be marked with "**Cache-Control: private**"

- **`Vary: User-Agent`** is not supported by most CDN's
- **`Vary: Accept`** is *a much better* route. Opera sends "image/webp", WIP in Chrome: crbug.com/169182

Deploying new image formats on the web

awesome.webp

Android / iOS

Android >4.0: native
Android <4.0: backport
iOS: WebP.framework

Web

Server Detection

Client (JS) Detection

#protip: *server-side automation FTW!*

Resources:

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

Mailing list: webp-discuss

Ilya Grigorik
igrigorik@google.com
Stephen Konig skonig@google.com