AMP: Hopes and Fears

These are my notes and comments from the AMP Roadshow in Sunnyvale on April 4.

The questions/issues I have are in bold throughout the rest of the article.

> at the #amproadshow learning about AMP, what's next and what cool things you can do with it.
>
> I'm ambivalent about AMP and am hoping to be convinced otherwise
>
> — Carlos Araya (@elrond25) April 4, 2018

What problem is AMP trying to solve?

Particularly since the introduction of AMP stories and the losening Javascript restrictions on adding Javascript to AMP content it's starting to look just like another optimized library to create web content and, furthermoer, the AMP cache is starting to look more and more like the old m-dot mobile sites of yesterday.

So it begs the question:

Beyond performance, what else are we trying to solve with it?

User experience

One of the things that bothers me is that AMP looks like it wants to replace most other web technologies that try to improve the user experience.

This is part of a more general question about my wondering why Google competes with itself in so many different categories? In the web space we have at least two competing technologies: AMP and PWA's that live apart or can be forced together in what the AMP team calls Progressive Web AMPs so it begs the first question in this category.

If this is for best users experiences why not support technologies that enhance what's already there instead of creating something new?

The AMP team acts as gatekeepers for what is and isn't part of the AMP stack and platform. My problem, for example, is that I can't run the Prism syntax highlighters into AMP code (mine is generated by Wordpress' `wp-amp` plugin compiled from source). It works under normal circumstances when used with non-amp content so why shuold AMP content be any different?

Since we can't use Javascript on AMP content, how can we leverage web APIs? I understand that you're working on relaxing the restriction of using Javascript in AMP content and experimenting with using workers to offload running scripts (amp-script) but, right now, I can't use any APIs that the AMP team doesn't allow. How is that putting users first?

I think I figured out the cause but not the reasoning behind it. Until the AMP team relaxes the restrictions for scripts on AMP content the AMP library itself will not accept scripts other than the ones they whitelist, so whenever scripts needs to inject content into the page they will fail and the content will not render as expected.

In my case with my code highlights I still get something that looks different enough from my content to be distinguishable from prose but not what I expected as syntax highlighted code.

I guess I could pre-process the content and insert the elements I need before the content is loaded but doesn't that defeat the purpose?

Who answers if there's a problem?

Until better tooling is available it's very hard to do good AMP development. Given all the restrictions of what you can't do and how you should do what you can, it's hard to remember that your primary IDEs for AMP are Dev Tools and the Google Search Console for your site.

Most people won't bother to check the search console to see if there is a problem with search and AMP or know how to fix it (I had a problem with linked data being wrong but since I didn't write the plugin or the AMP tags being used to generate it, I had no idea about how to solve the problem) so having AMP working is not guarantee that the tool is working properly, or at all…

So whose fault is it? Mine? The plugin developer? The AMP team?

We also don't know if AMP Validation Errors will cause the page not to render or if these are warnings of things that must be fixed to be compliant.

Garbage in, Garbage out

I host my own installation of Wordpress so this is less of an issue for me. But for the people who host content in Wordppress dot com this may come in as a surprise: ***AMP is enabled by default!*** (see AMP (Accelerated Mobile Pages) in the Wordpress dot com support site).

Twitter is also sending users to AMP version of the content (when available) by default. **Shouldn't this be an opt in experience rather than opt out or no choice at all?**

This is less of an issue for Twitter than it is for Wordpress because in the earlier, it's a concious action to linkn to a resource. In Wordpress it's a setting enabled by default so it'll publish AMP until you've actively disabled it so people may not be fully aware that they are publishing AMP and not really care about it.

So we have situations where the AMP content matches the canonical content, AMP is not the canonical content, and where there is no quality control for the AMP content itself... in the WordPress case, I doubt many users are aware that they are publishing AMP content. So if we get low-quality content pushed into AMP, who wins in this scenario?

Duplicating effort?

When AMP first came out as an optimization library I had my reservations but saw it as an interim solution to the mobile performance issues that I see documented everywhere and that have gotten progressively worse.

But it's not about improving mobile performance anymore.

The concept of ***amp stories*** and the associated elements that we must use to build them are a completely different concept and, to be honest, they frighten the hell out of me.

The concept of stories at least according to the news I've read out there, throws the concept of non-AMP sites out the window. **If there is no equivalency**

check, what will search engines do when the content is different? Which version of the content will get priority?

According to Search Engine Land:

> Breaking with previous guidance, Google does not intend for this type of AMP content to match your non-mobile content. Instead, this mobile-only content should be unique. I followed up on this point with a Google rep, who characterized AMP Story content this way:
>
> > AMP stories is all about encouraging new forms of expression and storytelling, so we expect most publishers to not be already publishing this kind of content out on the web currently. AMP stories is meant to facilitate this and make it easier.
>
> The rep also added that "AMP Story content should be fulfilling and standalone."
>
> Because Google has historically argued specifically against creating different content for different users, this new "separate but equal" stance surprised me. Google further explained:
>
> > This does not run counter to the idea that the publisher may also have separate expressions [of] similar content published. Say a publisher has a long-form analysis article about college admissions data. Then, say they have a companion piece that's an interactive experience letting people play with the data through charts and other visualizations. It wouldn't be correct to call the long-form piece the canonical for the interactive. They each stand on their own, although being related.

This bothers me in so many levels but I'll start with the most basic concern and

fear: Are we turning the mobile web into a single vendor experience? Will this difference between stories and web content force publishers into AMP or be left behind by not providing good results in the search page?

How about leveraging other parts of the web platform to tell the same type of stories you do with AMP? Again, isn't improving the whole web experience the name of the game?

Technical questions

There are technical questions that came up when researching AMP and writing this post a few more technical questions

Converting your content?

In talking to people at the Roadshow I now understand that AMP is a testbed to make things better and then bring it back to the wider web community; this is an important goal. I just have issues with the way AMP is doing this.

The example below is a basic sample HTML document conforming to the AMP HTML specification.

```
<!doctype html>
<html amp>
  <h<html amp>meta charset="utf-8">
    <title>Sample document</title>
    <link <title>nonical" href="./regular-html-version.html">
    <meta name="viewport" content="width=device-width,minimum-scale=1,init
    <style amp-custom><meta name="viewport" content="width=device-width,m
      h1 {color: red}
    hema.org",
      "@type": "NewsArticle",
      "headline": "Article headline",
      "image": [
        "thumbnail1.jpg"
      ],
      "datePublished": "2015-02-05T08:00:00+08:00"
    }
```

```
    </script>
    <script async custom-element="amp-carousel" src="https://cdn.ampproje
    <style amp-boilerplate>bod</script>
    {
      "@context": "http://schema.org",
      "@type": "NewsArticle",
      "headline": "Article headline",
      "image": [
        "thumbnail1.jpg"
      ],
      "datePublished": "2015-02-05T08:00:00+08:00"
    }
    body{-webkit-animation:-amp-start 8s steps(1,end) 0s 1 normal both;-m
    <p>
      Some te</h1>body{-webkit-animation:none;-moz-animation:none;-ms-ani
    <script async src="https://cdn.ampproject.org/v0.js"> data-aax_pubnam
        data-aax_src="302">
    </amp-ad>
  </body>
</html>
</amp-ad>
```

Some of the requirements and limitations of AMP are not evident. According to AMP HTML Specification an AMP document must:

- start with the doctype `<!doctype html>`
- contain a top-level `<html ⚡>` tag (`<html amp>` is accepted as well)
- contain <head> and <body> tags (They are optional in HTML)
- contain a `<link rel="canonical" href="$SOME_URL">` tag inside their head that points to the regular HTML version of the AMP HTML document or to itself if no such HTML version exists
- contain a `<meta charset="utf-8">` tag as the first child of their head tag
- contain a `<meta name="viewport" content="width=device-width,minimum-scale=1">` tag inside their head tag. It's also recommended to include initial-scale=1
- contain a `<script async src="https://cdn.ampproject.org/v0.js"></script>` tag inside their head tag
- contain the AMP boilerplate code (`head > style[amp-boilerplate]` and `noscript > style[amp-boilerplate]`) in their head tag

If this was all that AMP required for compliance I'd be ok. I've always been a gun-ho fan of proper page structure and including the appropriate elements where they should be. Death to tagsoup markup!

I love the idea of having metadata inlined in the document. This makes it easier to enforce (it's part of the AMP html spec) but it may make it harder to support in the long run as any change to the doc would require.

I may even forgive the extra attribute (other than class) in the html element.

But this is just the beginning.

Amp not only requires a certain structure from your HTML page but it changes some elements for AMP-specific ones and completely disallows others. I'm mostly OK with the tag it forbids as they are mostly legacy elements that have been kept on the specifications for compatibility (don't break the web) reasons. I mean, seriously, frameset and frames? It's 2018 people... we have better ways of doing that!

The table below (also from the AMP HTML Specification) explains the limitations and changes that AMP makes to standard HTML elements. Some of these changes help explain the structure of the example page seen earlier.

Tag	Status in AMP HTML
script	Prohibited unless the type is `application/ld+json`. (Other non-executable values may be added as needed.) Exception is the mandatory script tag to load the AMP runtime and the script tags to load extended components.
noscript	Allowed. Can be used anywhere in the document. If specified, the content inside the `<noscript>` element displays if JavaScript is disabled by the user.
base	Prohibited.
img	Replaced with `amp-img`. Please note: `<img>` is a Void Element according to HTML5, so it does not have an end tag. However, `<amp-img>` does have an end tag `</amp-img>`.

	[Or put in other terms, custom elements must have a closing tag.]
video	Replaced with `amp-video`.
audio	Replaced with `amp-audio`.
iframe	Replaced with `amp-iframe`.
frame	Prohibited.
frameset	Prohibited.
object	Prohibited.
param	Prohibited.
applet	Prohibited.
embed	Prohibited.
form	Allowed. Require including amp-form extension.
input elements	Mostly allowed with exception of some input types, namely, `<input[type=image]>`, `<input[type=button]>`, `<input[type=password]>`, `<input[type=file]>` are invalid. Related tags are also allowed: `<fieldset>`, `<label>`
button	Allowed.
style	Required style tag for amp-boilerplate. One additional style tag is allowed in head tag for the purpose of custom styling. This style tag must have the attribute `amp-custom` Do not use `@import` to load fonts in your CSS, use `amp-font` to load fonts for your page
link	`rel` values registered on microformats.org are allowed. If a `rel` value is missing from our whitelist, please submit an issue. `stylesheet` and other values like `preconnect`, `prerender` and `prefetch` that have side effects in the browser are disallowed. There is a special case for fetching stylesheets from whitelisted font providers.

meta	The `http-equiv` attribute may be used for specific allowable values; see the AMP validator specification for details.
a	The `href` attribute value must not begin with `javascript:`. If set, the `target` attribute value must be `_blank`. Otherwise allowed
svg	Most SVG elements are allowed.

But it's the changes that they make to standard HTML elements that worry me. Do we really need an image custom element? video? audio?

I don't think this solves the underlying problems of the web. It doesn't change the fact that loading speed and payload size numbers are getting slower and bigger, not faster and smaller for the mobile web.

See the HTTP Archive comparison data between January 1, 2017 and March 15, 2018 for an idea of what I'm talking about.

Granted, the HTTPArchive crawl may not catch AMP pages in its mobile results and that the pages in the crawl may not have a link to the AMP version of the page so the numbers may not reflect the actual improvements (if any) that AMP has made on the web

But AMP doesn't change the fact that we still have megabytes of images (even if they are compressed), several hundred KBs of fonts, and huge bundles of Javascript that will take tens of seconds to load and unblock page rendering. Even though people, companies and doc sites have been promoting best practices for years there has been no real improvement on how we work on the web

There are many more solutions outside of AMP that will give you performant web content. I think the biggest issue is that people may not know or may not care about these new technologies and smaller libraries that will take care of some performance bottlenecks.

Generating AMP?

Hand writing AMP is different than writing HTML but how do we generate AMP when working with CMS other than Wordpress? How much do we need to retool our systems based on the restrictions imposed by AMP?

Condé Nast has published information on how they generate the AMP content

for each of their publications: Allure, Architectural Digest, Ars Technica, Backchannel, Bon Appétit, Brides, Condé Nast Traveler, Epicurious, Glamour, Golf Digest, GQ, Pitchfork, Self, Teen Vogue, The New Yorker, Vanity Fair, Vogue, W and Wired.

Their system is fairly intricate as the image below illustrates:

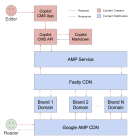

Figure 1: AMP Generation Process at Condé Nast. From The Why and How of Google AMP at Condé Nast

And the AMP generation process itself looks like it's rather complex, running from Markdown to React that then converts the Markdown content into valid AMP HTML:

Figure 2: Condé Nast AMP Service Pipeline. From The Why and How of Google AMP at Condé Nast

So now we have to insert a way to identify when we want our CMS to serve AMP content and an AMP conversion process into our pipeline... but only for our mobile content, right? I go back to the question I asked earlier about creating AMP/mobile

only sites: Are they necessary?

Furthermore, I'm concerned about the ammount of work needed to get AMP into a CMS. If you're not using a commercially available platform, how do you build AMP in addition to the standard content? As we saw in `Converting your content?` the restrictions of an AMP page are different than those of normal HTML (even if some of the restrictions are sensible and make sense).

Another thing that comes to mind when it comes to generating AMP programmatically is how much of the CSS we use in our normal site can we use with AMP?

The AMP specification requires developers to use no more than 50kb of CSS and, on its face, it's a sensible requirement. That is until we realize that CSS animations and any extra prefixing we need to acomodate browser idiosyncracies would also fall under that limitation and we need to decide early on how will we make the CSS fit into the 50KB requirements or how much to take out to make the CSS fit the size restriction and still keep branding folks happy. **This is not to say we should dump hundred and hundred of lines of useless CSS into our pages** but we should be the ones who decide how much CSS we use and how the CSS we use interacts with the existing content of the page.

Same thing with Javascript.

Specific Use Cases

My specific cases are Codepen embeds and Prism.js code syntax highlighting.

The typical Codepen emebd looks like this:

```
<p data-height="395" data-theme-id="dark" data-slug-hash="aGVPvL"
data-default-tab="html,result" data-user="caraya" data-embed-version="2"
data-pen-title="Simplified Article "
class="codepen">See the Pen <a href="https://codepen.io/caraya/pen/aGVPvL/
by Carlos Araya (<a href="https://codepen.io/caraya">@caraya</a>)
on <a href="https://codepen.io">CodePen</a>.</p>
<scrip<a href="https://codepen.io/caraya/pen/aGVPvL/">bed/ei.js"></script>
</script>
```

As far as I understand AMP, this would not work in an AMP page. AMP will not let

you use `script` tags in a page, and even if AMP allows `data-*` attributes on elements they are useless because there's nothing to process them with.

Same thing with Prism. It works by adding a script and a style sheet to the page and then expects properly coded blocks of preformated text for the content to be highlighted. To validate AMP content I am not allowed to have additional script tags on the head or before the end of my document and, in theory, I should be able to merge Prism's CSS with the AMP required CSS and my site's CSS but would that work in 50KB of CSS that we're allowed to add?

CORS for my own site?

One of the biggest issues is hosting the content in a Google-based and supported CDN. This causes the large, ugly URLs (like `https://www-gq-com.cdn.ampproject.org/c/s/www.gq.com/story/gq-eye-exclusive-fatherhood-style-series---todd-snyder` or `https://www-gq-com.cdn.ampproject.org/c/s/www.gq.com/`) but also has an interesting side effect that makes CORS necessary for your own content.

When you go to a site, for example: `https://www.gq.com/story/gq-eye-exclusive-fatherhood-style-series---todd-snyder` in a mobile device it'll redirect you to the version hosted in the AMP CDN which turns into `https://www-gq-com.cdn.ampproject.org/c/s/www.gq.com/story/gq-eye-exclusive-fatherhood-style-series---todd-snyder` and caches it to guarantee preloads and seemingly instant first load experiences.

But what happens if you need to load data for your application? Say you want to load product data for your shopping cart, or a tour date list for your favorite band? Those are stored on your server, not the cache and for them to work properly you need to set CORS headers on them so they will work for origins other than yours.

On a side note, if you hit the AMP cache URL from a desktop browser it will redirect you to the desktop, however, if you hit an AMP link using an activation script like `https://www.gq.com/story/gq-eye-exclusive-fatherhood-style-series---todd-snyder/amp` it will take you to the AMP version, regardless of platform or form factor. I find this even more interesting, why restrict some ways to access the content from some form factors and not others? Am I missing the point of the URLs?

Performance

People have been working with performance for years. The early work Ilia Grigorik and Addy Osmani did and continue to do at Google, tools like UNCSS, the fact that we can precache and preload content (but not provide automatic preloading like Google does with AMP search results), the fact that we have PWA and its component technologiies available within and outside PWAs really makes me wonder how lazy we've become as developers.

I'll take three examples of things we can do now to make experiences outside of an AMP environment: Image Optimization, Resource Pioritization and Service Workers.

We know that images make up for a large fraction of what gets pushed into a web page, we don't evangelize tools like Imagemin and workflows that use these tools to generate smaller images.

Does `amp-image` really reduce the bloat problem? If you make the images smaller that means people will put even more images in a page because the size of each individual image is smaller so we can use more of them, we don't need to worry about the individual image, AMP will take care of that.

It is not hard to create a good resource prioritization strategy using a combination of H2 Push, link preload, prefetch, prerender and dns-prefetch to load resources the way we want them to. If we compare this with lazy loading content so they will only load when needed, we have a much better web experience.

Service workers can also help improve performance of your web content. We can use it to keep a persistent cache of content for our site or application that can be populated and expired as needed. Libraries like Workbok.js can help us make this process easier.

I recently came accross Tim Kladec's How Fast Is Amp Really? performance analysis of AMP content served in different ways:

1. How well does AMP perform in the context of Google search?
2. How well does the AMP library perform when used as a standalone framework?
3. How well does AMP perform when the library is served using the AMP cache?
4. How well does AMP perform compared to the canonical article?

His findings mirror my opinion about AMP not doing anything that we can't do in the open web. To me, the most, important section of Tim's article is when he talks about page weight reduction:

> The 90th percentile weight for the canonical version is 5,229kb. The 90th percentile weight for AMP documents served from the same origin is 1,553kb— a savings of around 70% in page weight. The 90th percentile request count for the canonical version is 647, for AMP documents it's 151. That's a reduction of nearly 77%.
>
> AMP's restrictions mean less stuff. It's a concession publishers are willing to make in exchange for the enhanced distribution Google provides, but that they hesitate to make for their canonical versions.
>
> If we're grading AMP on the goal of making the web faster, the evidence isn't particularly compelling. Every single one of these publishers has an AMP version of these articles in addition to a non-AMP version.
>
> From: How Fast Is Amp Really?

AMP restricts what you can do with Javascript and what you can do with custom AMP elements restricts what you can do on your pages and, regardless of the reasons why you put the restrictions in place, it shouldn't be up to a library or framework to dictate how much stuff you put in a page but it's up to the designers and developers to slim down their content... easier said than done, I know, but AMP hasn't sold me as the solution or even a good solution to this problem and neither do publishers, otherwise AMP would be the canonical version of all published web content.

Conclusion

I wrote the tweet below as a response to my original tweet.

> No, @AMPhtml hasn't fully sold me on it. Too many open issues in AMP that are easy to do in non-amp. Looking at component creation and how hard it is

> — Carlos Araya (@elrond25) April 6, 2018

As I've mentioned throughout this post the technology behind AMP is not revolutionary. It's a reaction to a problem on the mobile web and it has been taken as a way to gate keep and restrict what can be done on the web.

There are many unanswered questions about AMP and the direction it's going on.

Attempting to make AMP the canonical version of your web content worries me or simply moving away from having an HTML versionn altogether means that people may choose to learn AMP but not HTML and when AMP limitations become too restrictive will not know how to build web content that doesn't require AMP and be starting from square one again.

I mentioned earlier but having Amp Stories be completely different to any non-amp alternatives makes me worry as much if not more than making AMP the canonical version of my content. If there is no relation between my web content and the AMP stories that are built around it then what's going to be the content that gets returned by the Googlebot when you search for it? I know this is not, strictly, an AMP issue and more of a search issue but since it's an AMP product it's worth pointing out here.

I'll keep my mind open towards AMP (after all I still provide AMP for my blog content to help people with slow connections or who may have data cost issues) but there are too many open questions about the platform and its direction before I can say I agree with their assessment of moving the web forward.

Links and Resources

- AMP content
  - https://www.ampproject.org/docs/fundamentals/converting
  - Improving URLS for AMP Pages
  - AMP New Horizons
- AMP Elements/Components/Templates
  - amp-story element
  - AMP Story Example
  - amp-iframe element
  - News and Blogs Templates
- AMP Stories
  - Introducing AMP Stories, A Whole New Way To Read Wired

        - Wired Introductory AMP Story
        - Search Engine Land Piece on AMP Stories
        - Google Releases AMP Stories: What Are They All About?
- Governance
    - Githubb Discussion
- AMP at Condé Nast
    - The Why and How of Google AMP at Condé Nast
    - Evolving Google AMP at Condé Nast
- Questions, Commennts and Criticisms
    - Ethan Marcotte
        - I, for one
        - AMPlified
        - AMPersand
    - Jeremy Keith
        - AMPed up
        - The meaning of AMP
        - In AMP we trust
    - Tim Kladec
        - How Fast Is Amp Really?
        - The Two Faces of AMP
        - AMP and the Web
        - AMP and Incentives
        - CPP: A Standardized Alternative to AMP
- AMP for you (whether you want it or not)
    - Twitter
        - Twitter
        - Twitter Developer Docs
    - Wordpress
        - Wordpress Support
        - WordPress Sites Now Support Google's AMP To Make Mobile Pages Load Much Faster
            - Am I reading this correctly in assuming that it was around this date that Wordpress.com made AMP an automatic feature of the platform?
- Other
    - Taking AMP for a Spin
    - AMP News
    - Need to Catch Up on the AMP Debate?