Loading Mobile Web Pages in 2 Seconds or Less

Page Construction Guidelines for Mobile Websites

Keynote Analytics

Version 4

April 2014

Contents

Summary

- 2010: Google begins factoring site performance into its Page Rank formula.[1]
- Early 2013: Google reports that global mean time for real users to load pages is 7 to 8 seconds.[2]
- Mid-2013: Strangeloop reports that 85 percent of mobile users expect sites to load at least as fast or faster than sites on their desktop.[3]
- Late 2013: Google recommends that mobile pages render above-the-fold content in less than 1 second.[4] Meanwhile, only 16 percent of Fortune 100 websites come close to that threshold.[5]

Talk about setting the bar high.

Mobile Web page performance is a tough nut to crack as it is. Users expect desktop-like responsiveness despite the lower throughput and higher latency of mobile networks. High performance on a mobile site results in more stickiness, greater ad revenue and increased sales, while poor performance leads users to abandon the site or, worse yet, complain about it.

A 2-second page load is ambitious, let alone Google's recommendation for 1 second.

Site designers may have little control over the mobile network itself, but they have plenty of control over how their pages behave on both the device and the server. Nobody can guarantee 2-second page load times, but techniques abound for rendering satisfying, above-the-fold (i.e., initial page render) content on a mobile browser with minimal delay. Above all, applying those techniques to reduce the number of network round-trips per page should be the site designer's top priority.

This white paper encapsulates the recommendations of the Keynote Analytics team in analyzing thousands of Keynote customer sites since 2001. Designers, developers and DevOps professionals familiar with Web terminology can use the paper as a roadmap toward 2-second page load times on their own mobile sites.

Main Messages:
-The site designer's top priority is to reduce the number of network round-trips per page.
-Keynote's top three recommendations for best practices in mobile page design are:
- *Eliminate or reduce redirections*
- *Reduce the number of DNS lookups per page*
- *Reduce the number of new HTTP requests per page*

[1] Google, "Using site speed in web search ranking," April 2010. http://googlewebmastercentral.blogspot.com/2010/04/using-site-speed-in-web-search-ranking.html
[2] Google, "Is the web getting faster?" Apr 2013. http://analytics.blogspot.com/2013/04/is-web-getting-faster.html
[3] Strangeloop Networks, "Mobile Device Users Expect Sites to Load Fast," 2013. http://www.strangeloopnetworks.com/resources/infographics/web-performance-and-user-expectations/mobile-device-users-expect-sites-to-load-fast/
[4] Google, "Mobile Analysis in PageSpeed Insights," updated November 2013. https://developers.google.com/speed/docs/insights/mobile
[5] The Search Agency, "Mobile Experience Scorecard: Fortune 100 Companies," November 2013. http://info.thesearchagency.com/TheSearchAgencysMobileExperienceScorecardFortune100Companies.html

Introduction – Focus above the Fold

To achieve the optimal user experience, the immediately visible, above-the-fold content on the page's critical rendering path needs to display and be functional to the site visitor as soon as possible.

The sequence in which the browser requests resources on the page affects the visitor's experience. Keynote recommends the following order to render a useful, above-the-fold page most quickly:

1. Base page HTML
2. CSS
3. Any JavaScript required for initial page render
4. Images
5. All other JavaScript
6. Third-party tags

Speeding up the delivery of both above-the-fold and below-the-fold page content requires that designers deal with six broad areas of the Web, each of which has its own set of variables:

1. Improving Overall Architecture
2. Accelerating Initial Render
3. Dealing with Latency on Mobile Networks
4. Shortening Render Times
5. Reducing Page Size
6. Enhancing the Return Visitor Experience

We examine each of those areas and variables below.

1. Improving Overall Architecture

All pages follow the same basic steps while loading, and there is room for improvement at each step:

1. Perform DNS lookups
2. Establish TCP connection(s) to the web server(s)
3. Establish SSL connections (for secure page content)
4. Load the base page HTML document
5. Execute application calls
6. Load content

Use no more than 2 domains per page. Sites routinely use multiple domains to accommodate mobile and non-mobile content, third-party tags for analytics and ads, secure and non-secure content, assets pulled from internal business units, and intentional domain sharding (which Keynote does not recommend). However, every additional domain on a page requires an additional DNS lookup – particularly costly on a mobile network – and at least one additional TCP connection to a Web server.

Best Practices for Domain Usage:
-Where possible, limit to 2 domains per page (excluding third-party)
- All elements critical to initial render should be on the same domain

Keep TCP connections persistent with *HTTP Keep Alive*. The entire process of delivering a page depends on TCP and the critical initial connection that usually takes place once per domain. But if *HTTP Keep Alive* (i.e., persistent connection) is disabled, the site forces a costly, new TCP connection for each request. When *HTTP Keep Alive* is enabled:

- the browser establishes a new TCP connection to each domain and reuses it for the life of the session;
- connections remain open unless forced to close (by a server or network device setting);
- the savings from not having to close connections can reduce page load time by seconds.

Best Practice for TCP Connections:
-Always enable HTTP Keep Alive in all situations, on all domains

Use Secure Socket Layer (SSL) sparingly. SSL enables secure communication between the server and the browser through encryption. However, each new TCP connection requires establishing SSL (via key exchange), slowing down page load. Using *HTTP Keep Alive* is especially helpful on pages served with SSL connections.

Best Practices for SSL:
-Limit the number of domains used on SSL pages (limiting the impact of setup)
-Enable HTTP Keep Alive (critical for SSL pages)
-Use SSL only when absolutely necessary

Reduce size of HTML file. The HTML code served for a page contains its core structure, but server-side generation for the HTML can be a performance bottleneck. Keynote recommends that mobile-optimized sites deliver all HTML required for initial page render in a single file 14KB or smaller. Techniques like minification (removing whitespace and trimming text) and compression (all mobile browsers support gzip compression) can reduce the size of the HTML file without compromising content. See "**Deliver base page HTML in first round trip**" below.

Best Practices for HTML File Size:
-Reduce HTML code to 50KB or less, optionally using gzip compression
-Limit the base page HTML file to 14KB or less
-Minify HTML by removing whitespace and content
-If the total of all the HTML on the page has to be greater than 50KB, or if the base page HTML cannot be 14KB or less, consider "early flush" of HTML to allow loading of CSS and

JavaScript before the entire HTML document is loaded

Retrieve HTML code in a single request. Except for the simplest web pages, all sites require application calls (*HTTP GET* or *HTTP POST*) to retrieve the core, dynamically-generated HTML or multiple HTML documents. Where possible, avoid multiple application calls to reduce the number of HTTP requests over the mobile network. Also avoid application calls that return redirections, which serialize (i.e., induce delay by forcing the browser to load assets one at a time, rather than in parallel) and slow the process of loading the HTML in the browser.

Best Practices for Application Calls:
-Return all core HTML used to render the page in a single application call
-Remove all redirections
-Defer additional or expanded functionally to load later in the page load or via asynchronous (non-blocking) application calls

Serve content optimized for connection type. A growing number of devices and mobile operating systems (Android 2.2+, BlackBerry) offer details accessible to JavaScript about the current network connection (3G, 4G, Wi-Fi) of the browser. Server-side application processes can serve optimized content based on network connection type.

Best Practice for Network Connections:
-Where available, check the network connection type in the browser and serve an appropriate version of content

2. Accelerating Initial Render

For fast initial render, all critical, above-the-fold content must be delivered as quickly as possible. Correctly incorporating HTML, Cascading Style Sheets (CSS) and JavaScript to the overall architecture of the page is a big step toward shorter page load time.

Deliver base page HTML in first round trip. To initially render the page most quickly, the browser must receive critical HTML content in the first Round Trip Time (RTT) of the initial TCP connection. With normal TCP settings, the maximum load for the first round trip of data delivery is 14KB, so if the base page (i.e., HTML required for initial page render) can fit in a 14KB file, the browser can begin work without waiting for additional round trips. Load additional HTML beyond the base 14KB from a separate file after initial render.

Best Practices for Base Page HTML File Size:
-Reduce HTML required for initial page render to 14KB or less, served in a single file
-Avoid any redirections prior to the base page HTML request

Deliver CSS as early as possible. Initial render is delayed until the browser can build the Cascading Style Sheet Object Model (CSSOM), so it is imperative to load CSS early. The browser parses HTML incrementally, but it must wait for the entire CSS file to load before it can process any of it. Deferring non-critical CSS content to files loaded later can speed up the initial page render.

Best Practices for CSS:
-Deliver CSS as early as possible (either as part of the base page HTML content or as the very next content requested)
-Place all render-critical CSS in line in the base page HTML response, deferring the rest

Eliminate blocking JavaScript. JavaScript can slow initial page render when it blocks the loading of all other content on the page, delays the construction of the Domain Object Model (DOM) in the browser's memory, or queries/modifies the DOM or CSSOM.

Best Practices for JavaScript:
-Ideally, load no JavaScript at all in the critical path to initial page render
-Eliminate blocking JavaScript
-Place all render-critical JavaScript in line in the base page HTML response, deferring the rest

3. Dealing with Latency on Mobile Networks

Reducing the number of round trips per page should be the site designer's top priority.

Latency can be as much as 10 times worse on a typical 3G mobile network than on a high-speed wired network connection. Round trips for requests, responses and DNS lookups that take 20 to 50 ms on a wired network might require 250 ms or longer over a 3G network. Web designers can avoid compounding that latency by carefully using redirections, HTTP requests, images and content delivery networks (CDNs).

Redirect on the server side. Redirections are useful for moving from a mobile to a non-mobile domain, from a non-secure to a secure domain and from a generic to a customized response (e.g. iPhone- versus Android-optimized HTML). However, redirections serialize the download in the browser, especially when they precede the base page HTML delivery, making a 2-second initial render almost impossible. Except when moving from a non-secure to a secure domain, avoid redirections and their very high-latency round trips to the server.

Best Practices for Redirections:
-Remove all redirections except for necessary HTTP->HTTPS redirections
-Avoid situations where a link uses a URL that will require an HTTP->HTTPS redirection; link directly to the HTTPS URL instead

Limit HTTP requests to 20 per page. Limiting the number of elements on a page is the best way to reduce round trips between browser and servers. For sites with most visitors on 4G or better networks, Keynote recommends a maximum of 20 or fewer requested assets per page. For example:

- No redirections if possible
- 1 HTML file
- 1-2 CSS elements
- 2-4 JavaScript calls
- 12 or fewer image files
- 2 or fewer third-party or analytics tags

Best Practices for HTTP Requests:
-Total requested assets should be 20 or fewer per page
-Combine small (under 2KB) images with CSS sprites
-Combine external CSS and JavaScript into fewer files
-Replace "designer images" (e.g., rounded corners) with HTML5 effects
-Load third-party calls or analytics calls after initial render

Reduce separate requests for images. Several techniques can lead to fewer image requests:

- **CSS sprites** combine multiple small images in a grid in a single file, and CSS background image rules display appropriate portions of the image file where needed on the page.
- **Image manipulation tools in HTML5** enable effects like rotations to replace loading separate left and right versions of images, or to replace certain wait transition images altogether. HTML5 can also eliminate the need for color swatches.
- **Data URIs –** small images in-lined directly to HTML using Base64 encoding – eliminate separate round trips to the server. (Note that data URIs are not suited to large images, and that they can complicate the caching model.)

Best Practices for Images:
-Consider using CSS sprites
-Exploit HTML5 effects where appropriate
-In-line small images with Base64 encoding, but only in moderation

Tune CDN usage. Keynote encourages the owners of all sites – especially global sites – to move their static content to CDN edge servers located closer in geography and on the network to site visitors. A well-implemented CDN can improve responsiveness, even for dynamic content, and provide a form of peak-capacity load handling. (Note that most of the latency on mobile-optimized sites occurs in the last wireless hop to the device. Not all CDNs deliver the same level of benefit for mobile-optimized site visitors as for desktop site visitors.)

Best Practices for CDNs:
-Move all content (JavaScript, CSS, images, etc.) to the CDN
-Tune TTL (time-to-live) to allow for longest-term caching
-Enable optimization of non-static content (route optimization, etc.)

4. Shortening Render Times

JavaScript, third-party tags and other essentials contribute to good user experience but slow page load.

Limit external JavaScript calls to 4 per page. JavaScript calls can serialize download in the browser and slow page load, especially on mobile-optimized sites. Slow JavaScript calls can make the entire page feel slow to the user. Keynote recommends caching JavaScript, combining it into as few files as possible and auditing the code to avoid redundancies introduced by different development teams. If a JavaScript call occurs on only one page (e.g., a single geolocation function), deploy it within the HTML for a single file request; otherwise, move frequently used code to an external file instead.

Best Practices for Limiting JavaScript Calls:
-Limit external JavaScript calls to 4 per page
-Move any JavaScript critical to initial page render into the base page HTML
-Combine JavaScript files
-Structure JavaScript to enable it to be cached in the browser early in the site visit
-Consider using JavaScript inline where appropriate
-Load all non-essential JavaScript after initial page render

Optimize JavaScript. Some JavaScript is not needed on a page until after initial render, so deferring their download and execution can make the page load faster. Also, the Web Worker specification in HTML5 allows concurrent, multi-threaded execution of JavaScript in the browser for better performance, especially on limited-capability mobile processors. Large, complex development frameworks like jQuery often include code that mobile-optimized sites do not need.

Best Practices for Optimizing JavaScript:
-Defer the downloading and execution of JavaScript until after the initial page render
-Use HTML5 multi-threading where appropriate
-Avoid large frameworks optimized for desktop browsing requirements

Limit third-party tags to 2 per page. Each third-party call (e.g., for advertising, marketing analytics, tracking) requires a new round-trip request and response from a Web server and may require a DNS lookup. Furthermore, third-party providers may introduce quality problems or tags that block other page content from loading. At the very least, they occupy browser resources that other page content could be using.

Best Practices for Third-Party Calls:
-Limit third-party calls to 2 per page
-Audit tags regularly to ensure that they are still being used internally
-Load third-party tags only after all critical content on the page has been requested
-Always load third-party tags after the point of initial render
-Consider asynchronous loading of third-party content where appropriate
-Track third-party performance internally and reject services that cause performance problems

Replace click events with touch events. On the touch screens used in mobile devices, onclick events often incur delays of 300 to 500 ms after the user taps the screen, as the browser or OS waits for additional input or gestures. Replacing onclick with touch events (e.g., touchstart, touchmove, touchend) eliminates this delay and helps speed up page transitions. Sites can be designed either to support both touch events and onclick, or to identify the device type and serve only the appropriate version of the HTML.

Best Practice for Touch Events:
-Replace the onclick behaviors of the site with touch-event-driven behaviors where safe

Use Server-Sent Events for asynchronous communication. EventSource JavaScript objects and Server-Sent Events in HTML5 allow the browser to establish efficient, asynchronous communication with the server, especially over a mobile network. The techniques ensure that some page content is deferred until initial page render and functionality are complete. For sites that meter their customers' data usage, this is more responsible than auto-refreshing pages or sending polling requests.

Best Practice for Server-Sent Events:
-Consider Server-Sent Events instead of more traditional polling requests

Prefetch anticipated content. Prefetching page assets in anticipation of their use in subsequent page loads (e.g., through fixed steps in a purchase path, or a search -> search results -> search result details path) can be effective on mobile-optimized sites, provided that:

- the assets load only after the current page is completely loaded;
- prefetching does not interfere with JavaScript execution on less-powerful processors;
- the prefetched assets do not flush other assets out of the cache prematurely; and
- customers with metered usage have the option of disabling prefetching.

Best Practice for Prefetching:
-Consider moderate pre-fetching resources likely to be used on subsequent page views

5. Reducing Page Size

Although the number of requests is more important to page load time (see "**Limit HTTP requests to 20 per page**" above), heavy pages generally result in poor performance on mobile networks.

Limit overall page weight to 200KB. Keynote recommends page weights (i.e., the size of the file as sent across the mobile network) below 200KB, with a maximum 400KB for mobile-optimized sites. (Entry pages and marketing pages can be slightly larger if they employ the techniques in this paper for faster page render.) gzip compression is a server setting supported by all mobile browsers, so a 200KB compressed file meets the recommendation as well as a 200KB uncompressed file. Designers can minify text elements like HTML, CSS and JavaScript by removing whitespace in the files.

Best Practices for Overall Page Weight:
-Avoid serving pages heavier than 200KB where possible
-Compress (using gzip) all text elements (CSS, JavaScript, HTML, etc.)
-Minify all text elements

Limit image size to 10KB each. Keynote recommends a maximum of 10KB per image. Beyond that, consider optimizing or replacing the image.

- Most photographic images in JPEG format look acceptable at 85 percent quality.
- Scale images to be size- and quality-appropriate for mobile device displays.
- Image files with few colors may be smaller in GIF or PNG formats than in JPEG.

Also, put controls in place to prevent internal content owners from accidentally publishing large images on the mobile-optimized site. Inappropriately heavy content can ruin good mobile site design.

Best Practices for Image Size:
-Compress images with appropriate levels (85 percent quality)
-Resize images to optimize for smaller device displays
-Never serve images at higher resolutions than can be displayed on the screen
-Use GIF or PNG where image file sizes can be reduced
-Limit the ability to add image content to pages beyond certain size constraints

Take advantage of HTML5 and CSS 3.0. HTML5 offers new structural elements, functionality (e.g., header, navigation, article, footer), attributes for focus, and data-constrained input types such as date, time, email address and URL; most of those have traditionally required JavaScript. New effects in CSS 3.0 – gradients, rounded corners, border effects – along with animations and transitions eliminate the need for many external images. All popular mobile browsers support these new specifications.

Best Practices for HTML5 and CSS 3.0:
-Reduce JavaScript requirements with HTML5
-Reduce "designer images" and transition images with CSS 3.0

Optimize HTML and CSS content. Some development frameworks introduce validation tags to the HTML header that are unnecessary when deployed to production, adding weight without adding value. Web designers should examine all META tags for their necessity and CSS rules for their occurrence on the page to avoid sending useless bytes. Development frameworks may also automatically add attributes like *target="_self" or class=""* to anchor (*a*) and image (*img*) elements; designers can safely remove them and reduce HTML size.

Best Practices for HTML and CSS content:
-Audit all HTTP for unnecessary META tags
-Identify and remove unnecessary link and image attributes
-Avoid downloading CSS markup that is not used on the page

Trim unnecessary HTTP headers. Although it is not a part of Web page design *per se*, many sites send unnecessary HTTP headers with every request. To speed up page load, designers should remove header values for server IDs or server software versions that mobile browsers generally ignore anyway.

Best Practice for HTTP Headers:
-Audit all HTTP headers in responses to ensure that only essential headers are being sent

6. Enhancing the Return Visitor Experience

Understanding and efficiently managing the cache can reduce round trips for return visitors.

Tune caching for return visitors. Pulling an asset from cache avoids round trips and improves the return visitor experience. However, most mobile devices have relatively small caches (e.g., the iPhone 3GS has a limit of 25KB per HTML object, and most Android devices limit the total component cache size to 4 MB) that may not persist across device reboots. Other options for reducing requests include setting far future Expires headers 7 or more days – or even years – into the future, and managing versions with unique date stamps or codes in the filename to force new versions of elements to browsers on demand.

Keynote recommends avoiding the use of ETags to improve the return visitor experience on mobile-optimized sites. ETags are designed to validate content with a single server, but in multi-server environments they can actually backfire by forcing the browser to fetch from a different server a new version of a file already in cache.

Best Practices for Tuning Caching:
-Use far future Expires headers
-Remove ETags
-Limit the use of intentional "no cache" headers

Cache in offline storage. Now that popular mobile browsers support the HTML5 Web Storage specification for offline application caching, Web designers can typically count on access to 5 MB of storage per site rather than 4 MB for all sites on a typical Android component cache. Although this requires explicit management of the cache assets as key-value pairs through the localStorage JavaScript object, items in local storage persist across reboots and will not be flushed out of cache by visits to other sites.

Best Practice for Offline Application Caching:
-Use localStorage object to explicitly manage cached assets for the site

Use local storage instead of cookies. Because cookies are sent with each HTTP request, implementing cookies for domains on mobile-optimized sites hampers performance. This poses two undesirable alternatives for the return visitor experience: multiple domains, only one of which uses cookies (which will increase initial page load time with extra DNS lookups and/or TCP connections) or a single domain that uses cookies (which adds overhead to every resource request). Instead, Keynote recommends using local storage through HTML5 for storing and managing cookie-like information as efficient key-value pairs used only when explicitly necessary.

Best Practice for Avoiding the Use of Cookies:
-Use localStorage object to explicitly manage state for the site visitor

Conclusion

To get to a 2-second mobile Web page, site designers must reduce the number of round trips. Not all pages can be optimized to load that quickly, but it is a worthwhile goal. Designers must select the optimizations that make sense for their particular site.

The best way to implement the recommendations provided in this document is to collect and analyze actual performance measurements. Instead of a rules-based approach, Keynote strongly recommends taking ongoing measurements of the site and pages. By analyzing the resulting data, site designers can see exactly what is slowing page load and build their own mobile page guidelines for maximum performance.

Follow Us

Keynote runs the world's largest cloud testing, monitoring and analytics network in the world and collects over 700 million mobile and website performance measurements daily.

keynote.com

@keynotesystems

Keynote Web Performance Watch Blog

Keynote Systems