Front-End Optimization on the Akamai Intelligent Platform™
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>WHAT IS FRONT-END OPTIMIZATION?</td>
<td>2</td>
</tr>
<tr>
<td>FEO IMPLEMENTATION AND CHALLENGES</td>
<td>4</td>
</tr>
<tr>
<td>Automated Front-End Optimization</td>
<td>4</td>
</tr>
<tr>
<td>FRONT-END OPTIMIZATION WITH AKAMAI</td>
<td>5</td>
</tr>
<tr>
<td>Overview: How it Works</td>
<td>5</td>
</tr>
<tr>
<td>Asynchronous Analysis Delivers Speed</td>
<td>6</td>
</tr>
<tr>
<td>Adaptively Optimized for Each End user</td>
<td>6</td>
</tr>
<tr>
<td>Overcoming Bottlenecks From End to End</td>
<td>7</td>
</tr>
<tr>
<td>Mobile Performance Optimizations</td>
<td>8</td>
</tr>
<tr>
<td>BENEFITS OF FRONT-END OPTIMIZATION WITH AKAMAI</td>
<td>9</td>
</tr>
<tr>
<td>Offload the Headache</td>
<td>9</td>
</tr>
<tr>
<td>Rely on a Proven Platform</td>
<td>9</td>
</tr>
<tr>
<td>The Best Experience for Every User, Every Time</td>
<td>9</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>10</td>
</tr>
<tr>
<td>SOURCES</td>
<td>14</td>
</tr>
</tbody>
</table>
Introduction

In the hyperconnected world, web and mobile application performance is playing a more critical role than ever in driving user adoption and engagement. Today’s savvy audiences demand richer, more interactive experiences that require more data and more processing. This is a trend of ever-increasing average web sizes. Yet consumers continue to expect pages to load faster than ever before, indiscriminately abandoning sluggish sites. At the same time, studies have repeatedly shown that enhancing site performance brings about greater page views, better audience engagement, higher conversion rates, and even improved SEO rankings. Indeed, as little as a few tenths of a second can make a difference.

Unfortunately, understanding website performance is no small task, particularly as the Internet and the applications running across it become more and more complex. Today, a surprisingly large number of factors – including network latencies, communications protocols, device capabilities and browser page rendering behaviors – play a role in determining the end user perception of website performance.

Traditionally, web performance tuning efforts focused on the application back end. This is the area over which web developers have the greatest control, but unfortunately, changes made here may not translate into big improvements for the end user. This is because back-end time – which includes the overhead involved in making database calls and generating the HTML page – actually accounts for only about 10% of user-perceived page load time for the typical web application today. The remaining 90% is due to middle mile and front end latencies. The middle mile refers to the time it takes for data to travel across the Internet, which can face significant delays due to network latency, routing problems, and Internet congestion. The front end includes the time it takes for data to cross the last mile to the user’s device and then for the browser to render the page. Website performance optimization needs to successfully address the 90% problem in order to deliver the level of responsiveness end users demand.

### How website page load time breaks down

<table>
<thead>
<tr>
<th>Back End or First Mile</th>
<th>Middle Mile</th>
<th>Front End or Last Mile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database calls</td>
<td>Retrieving page content, (including HTML, images, JavaScript, etc.,) from origin server, across the Internet (the ‘Long-hair’ across multiple networks and peering points)</td>
<td>Delivering content over cable modem, cellular network, etc., to end user</td>
</tr>
<tr>
<td>HTML page generation</td>
<td></td>
<td>Rendering page in the browser</td>
</tr>
</tbody>
</table>
What Is Front End Optimization?

As content and application delivery networks have proven effective at helping sites overcome middle mile performance issues, front end performance issues are now coming into focus and being addressed. Front end bottlenecks play a growing role in overall site performance as today’s rich and complex Internet applications make increasing use of embedded objects, sophisticated stylesheets, client side code, and third-party content – all of which slow down the rendering of a page. Mobile sites, with their sluggish networks and weaker device processors, face an even greater front end challenge. The need to address these types of performance issues has led to increased awareness about Front End Optimization (FEO), a set of best practices designed to reduce front end delays in order to improve the end user experience.

FEO techniques can be broadly grouped into four complementary approaches:

1. **Reduce the number of HTTP requests.** Before rendering a web page, browsers first make an HTTP request for the HTML page, followed by requests for additional resources required to display the page – such as images, CSS, or JavaScript. The number of requests required for a typical page has more than tripled in the last 9 years, to a current average of 100 objects per page.\(^5\)

   Unfortunately, each new HTTP request can increase unwanted delay in setting up the connection. For mobile devices, this overhead typically takes more time than downloading the page data itself. Even for desktop browsers, this overhead is one of the key factors in determining page rendering times, and reducing the number of requests can greatly improve the user’s perception of site responsiveness. FEO employs various techniques to achieve this request reduction, including consolidating multiple files into a single request, caching objects locally using HTML5’s Application Cache interface, and inlining small objects within the HTML.

2. **Reduce the size of the data.** Page weight is often the single largest factor in page download times, so reducing the number of bytes transferred can significantly speed up a site. FEO techniques are designed to reduce page weight without changing visual quality or site functionality.

   Images are typically the biggest component of page weight and can often benefit from compression or resizing without compromising quality. Adaptive image compression, where resizing is optimized based on the end user’s screen resolution and connection speed, is especially effective. Text blocks – such as HTML, JavaScript and CSS files – can also be significantly reduced in size through techniques such as compression and minification, the process of removing unnecessary characters such as whitespace and comments.
3. **Accelerate page rendering.** Different browsers exhibit different page rendering behaviors, many of which prevent pages from rendering as quickly as possible. For example, browsers often:

- Block other downloads while downloading, parsing, and executing scripts.
- Limit the number of simultaneous requests for images and other embedded content. Older versions of Internet Explorer and Firefox, for example, are limited to two parallel requests per hostname. Newer versions increase this limit to 6.
- Block page rendering while stylesheets are downloaded and parsed.

FEO techniques address these problems by recoding the HTML in a way that distributes the browser’s work so that the page can begin to render as soon as possible, delaying nonessential tasks until later. Techniques include using asynchronous JavaScript execution, which helps prevent scripts from delaying object downloads and page rendering; just-in-time or on-demand image loading, which loads only images in the current viewable area, deferring others until the user scrolls; and domain sharding, or retrieving objects from multiple subdomains to work around browser limitations on concurrent requests. While these techniques work well to minimize page rendering bottlenecks, they do need to be customized to each browser’s specific behaviors. In some cases, we may add code to take advantage of newer browser capabilities, while, in other cases, we may make changes to overcome deficiencies in older browsers.

4. **Alleviate third-party bottlenecks.** Ads, analytics code, social media widgets, and other third-party content now represent a large percentage of total page requests. Unfortunately, due to the page rendering behaviors discussed above, sluggish third-party content can significantly slow down a page or even make it appear completely unavailable to the end user, as the browser may block the progress of a page while waiting for the third-party content to load.

websites have no control over the performance of third-party content, but FEO best practices can alleviate these bottlenecks by decoupling the third-party script execution from the loading of the page. By making the third-party request asynchronous, FEO ensures that a slow third-party tag will not negatively impact the loading of the other objects and the rendering of the page.

For an overview of Akamai FEO optimizations please refer to the appendix.
FEO Implementation And Challenges

FEO evolves constantly as browser and device technology continues to change, but it is hardly a black box or secret art. Indeed, many FEO best practices are well documented across the web on sites like YSlow, PageSpeed, and Steve Souders’ High Performance web Sites blog. In theory, this means any website development team can simply implement the techniques and enjoy significant performance improvements across its site. The reality, however, can be quite different.

First, FEO is complex, making it difficult to implement and costly to maintain. The up-front investment required to manually retrofit an existing site for FEO can be on the order of many man-years – not including the ongoing resources needed to keep the site optimized. Moreover, unless all members of the development team gain FEO expertise, any implemented optimizations may be lost each time the site itself is changed or redesigned.

Second, FEO best practices are not one-size-fits-all. Different approaches are relevant in different situations, and if used incorrectly, FEO techniques can actually hurt performance. For example, domain sharding may work well for desktop browsers running on fast connections, but it may hurt performance in certain mobile scenarios. The ideal number of domains to use also depends on multiple factors.

Generally, FEO techniques involve balancing tradeoffs among many different considerations. For instance, one might assume that inlining all content – that is, requesting all page resources in a single download – is ideal for performance, because it reduces HTTP requests down to one. In reality, this is not optimal for most situations, as you lose the often-greater performance benefits of being able to cache images or use just-in-time image loading to reduce page download size and accelerate page rendering. The balancing point between these tradeoffs can be a complex and moving target, as browser, devices and networks evolve – and proliferate – at a rapid pace. Trying to keep site optimizations up to date in the face of never-ending changes, both in the device/browser landscape and within the application itself – is a daunting and expensive proposition.

Automated Front End Optimization

In an attempt to address some of the challenges associated with manual FEO implementation, a crop of young companies are now providing automated FEO technologies, available either as an appliance to be deployed in front of a site’s servers, or as a service that intercepts and responds to requests on behalf of the site. These FEO technologies attempt to tackle only the front end piece of the overall performance puzzle; they do not address network latency, congestion, and reliability issues that can severely hamper the delivery of any website or application, for example. However, they can work in conjunction with content and application delivery networks to provide a more comprehensive solution.

Unfortunately, both appliance and service-based FEO solutions often become bottlenecks of their own. Automated FEO solutions sit in the line of fire, taking the HTML code generated by a site and modifying it before delivering it to the end user. Being in the critical path, the reliability, scalability and performance of the FEO solution is paramount. For service-based solutions, the website is at the mercy of its service provider. For appliance-based solutions, the website must architect and plan for capacity and failover as well as maintain their new boxes. Either way, there is a cost and risk associated.

In addition, these FEO solutions often have integration or compatibility issues with the other components of the website’s overall performance strategy, such as the content or application delivery network. When cobbling together two disparate solutions, companies are left to fend for themselves if something goes wrong, troubleshooting in the no-man’s land in between the cracks. These situations can create support nightmares for the site, particularly with a technology that modifies website code on the fly. Thus, while stand-alone automated FEO solutions have the potential to provide some performance benefits without the high cost of manual FEO coding, they also carry a number of risks that make them far from ideal.
Front End Optimization With Akamai

Akamai takes an end-to-end approach to solving website performance, offering a cohesive, multi-layered performance solution designed to help businesses overcome origin, middle mile, and front end bottlenecks automatically and effortlessly. For every end user request, Akamai’s proven technologies are dynamically applied in a way that optimizes performance for that unique scenario, taking into account real-time website, network, and end user conditions. Akamai’s FEO capabilities are an integrated part of these solutions, working in concert with our other performance, security, and availability offerings to deliver the best possible experience for every user, on every device, every time.

Akamai’s FEO capabilities are based on advances pioneered by Blaze, an industry-leading FEO company that Akamai acquired in early 2012. Blaze’s cutting-edge FEO solution was designed from the ground up to work in synergy with a distributed content and application delivery network, providing an ideal architecture for incorporation into the Akamai Intelligent Platform. Built on this foundation, Akamai is now able to offer a uniquely robust and scalable automated FEO solution as a seamlessly integrated part of our comprehensive web performance solutions.

Overview: How it Works

In the diagrams below, we illustrate how Akamai’s FEO works:

**Part I. First Request**

1. User requests a page
2. No front end optimizations are available
3. Page is accelerated across the middle mile and delivered to first-request user without front end optimizations
4. Subsequently, page is queued with the FEO Analysis Engine to be analyzed

**Part II. Offline Analysis**

1. FEO Analysis Engine evaluates page to determine the best ways to optimize it, and produces optimized transformation rules
2. FEO Analysis Engine makes optimized transformation rules available across the Akamai Intelligent Platform
As always, end user requests are efficiently routed to a nearby Akamai server. This server may be able to deliver the requested content straight from cache or will fetch it from the origin server using route, protocol, and application optimizations to accelerate delivery over the long-haul Internet. The server also applies the relevant front end optimizations before delivering the content to the end user, which enables speedy page rendering. The end result is an exceptional end user experience that is rich, reliable, and responsive, regardless of device or Internet conditions.

**Asynchronous Analysis Delivers Speed**

Understanding which front end optimizations to apply to a dynamic site requires complex and computationally intensive analysis. Our sophisticated, offline analysis engine does this work asynchronously, precomputing transformations to be applied to the various static and dynamically generated pages of a site. This information is then provided to servers across the Akamai Intelligent Platform, allowing them to rapidly apply the transformations in real time as they deliver content to end users. By separating the complex analysis from the real-time application of the optimizations, Akamai delivers cutting-edge FEO capabilities without sacrificing speed.

**Adaptively Optimized for Each End-User**

To be effective, FEO optimizations need to be customized for the end user environment, because different browsers and devices behave differently. For example, browsers differ in their support for features like asynchronous JavaScript, HTTP pipelining or the number of simultaneous downloads they can make, suggesting different optimization behaviors. Images may also be resized differently for different devices and screen resolutions. In addition, a device connected over a loss-y, high-latency 3G network may benefit from different types of optimizations than a device running over a fast Internet connection.

Akamai’s platform handles these different situations seamlessly, creating multiple possible optimizations in the analysis phase, and then intelligently applying the appropriate ones to each request based on real-time factors such as the end user's browser version, device and network speed. This means we take advantage of the full capabilities of each user’s specific setup to deliver the best possible experience – rather than simply settling for the lowest common denominator supported across all browsers – or targeting only the most popular browsers while leaving other users out in the cold.
Moreover, Akamai continually updates and improves its optimization engine to reflect the state-of-the-art in FEO innovations as well as the constantly evolving browser and device landscape. This enables our customers’ development teams to focus simply on creating innovative content and features, rather than getting bogged down by the ever-changing details around which versions of which browsers support what features. Your team creates and maintains just one version of your site; Akamai enables it to work optimally for every single user across an increasingly heterogeneous end user environment.

**Optimizations Targeted to Device and Network Conditions**

![Diagram showing developer creates and maintains one version, Akamai's offline analysis engine creates multiple optimization transformations, front-end optimization html optimizers targeted to device and network conditions.]

**Overcoming Bottlenecks From End to End**

By integrating FEO capabilities directly into the Akamai Intelligent Platform, these front end optimizations work seamlessly with Akamai’s entire suite of offerings, providing complementary performance benefits that overcome bottlenecks from end to end on a platform that is highly secure and proven to help deliver reliable, consistent web experiences to end users. FEO has particularly strong synergies with Akamai’s route protocol and connection optimizations. These capabilities are designed to deliver every byte faster while Akamai FEO works at the same time to reduce the amount of bytes needed. This holistic, multi-layered approach to website performance is designed to help businesses deliver exceptional end user experiences with unmatched performance, reliability, and scale.

**End-to-end Performance Benefits with Akamai**

**Without Akamai:** Many requests, many bytes, many long round-trips

**With FEO:** Fewer requests, fewer bytes

**With FEO + Dynamic Site Accelerator (DSA):** Fewer requests, fewer bytes, faster round-trips
Mobile Performance Optimizations

Because of the more restrictive capabilities of its devices and networks, the fast-growing mobile web faces a number of unique performance challenges in the last mile, shown in the table below. Yet consumer expectations remain high: a recent study showed that 71% of mobile users feel sites should load as quickly on their mobile devices as on their desktops. ⁶

Front-End/Last Mile Bottlenecks For Mobile

<table>
<thead>
<tr>
<th>Cellular Network Limitations</th>
<th>Device Limitations</th>
<th>Browser Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>• High latency</td>
<td>• Weak CPUs mean longer script parse times and slower rendering</td>
<td>• Greater diversity of browsers and OS versions to support</td>
</tr>
<tr>
<td>• High packet loss</td>
<td>• Small device caches</td>
<td>• Browsers support fewer features</td>
</tr>
<tr>
<td>• Short-lived communications channel and high overhead for establishing</td>
<td>• Smaller screen sizes and far greater variation in screen sizes</td>
<td></td>
</tr>
</tbody>
</table>

In this type of environment, FEO can have an enormous benefit. Useful techniques include many of the optimizations we have already mentioned – such as adaptively reducing image sizes based on display size or network speed, maximizing use of local device caches, delaying script parsing, and consolidating resource requests – tuned with mobile-specific parameters, as well as additional, mobile-specific optimizations such as:

• JavaScript Pre-Execution: delivering mostly static pages, with script pre-executed on the server wherever possible

• Click On Touch: Converting hyperlinks to click events to eliminate the 300ms delay associated with standard mobile hyperlink clicks

• Cellular Keep-Alive: Keeping mobile connections open longer in between page requests in order to avoid the high overhead of re-establishing a connection

In mobile, it is particularly critical that optimizations are customized for each request. Because the Akamai Intelligent Platform knows about the end user’s device, browser, and real-time network situation, it can apply FEO transformations that can help content providers maximize that user’s experience based on his or her current environment.

Equally important, these FEO capabilities work seamlessly with Akamai’s other mobile technologies, including the Akamai Mobile Protocol, which improves communication speeds over the slow, lossy cellular last mile. The seamless layering of these capabilities is designed to result in an unsurpassed user experience for every user, every time.
Benefits Of Front-End Optimization With Akamai

Offload the Headache

With Akamai, businesses can enjoy the tremendous benefits of FEO with the virtual flip of a switch. There’s no big up-front investment, no on-going maintenance nightmare, no struggle to keep up with the latest browser changes. In fact, there are no changes at all to your Akamaized site. This means your development team can simply focus on its core mission of developing compelling content and functionality, while our high-performance platform delivers it with the speed and responsiveness your users expect.

Rely on a Proven Platform

No other FEO solution can match the proven performance, scalability, and reliability of the Akamai Intelligent Platform. With over 170,000 servers in more than 1,300 networks worldwide, Akamai has the world’s most pervasive cloud optimization platform – trusted to accelerate as much as a third of the traffic travelling over the Internet each day. Our FEO capabilities are integrated into this time-tested platform, delivering a solution that is highly secure and backed by our performance and uptime SLAs.

The Best Experience for Every User, Every Time

Akamai’s layered, integrated approach to performance means that we are able to apply the right tools at the right time for each site, across nearly every scenario. Our device-targeted front end optimizations work in synergy with our site and application acceleration services to deliver unparalleled speed from end-to-end – in a process that is as seamless to our customers as it is to their end users.

Footnote: Gateway GPRS Support Node (GGSN). The place where the public internet and the private cellular infrastructure cross over.
Appendix

For businesses looking to deliver the fastest, most engaging user experiences possible, Ion offers a fully-automated solution for situational performance by including Akamai Front-End Optimization (FEO). Akamai FEO applies optimizations based on sophisticated analysis of the web application as well as real-time conditions specific to the end user’s environment, such as browser, device, network speed and usage of third-party services. Akamai Front-End Optimization:

REDUCES REQUESTS

Each client HTTP request and server response combination represents at least one round-trip on the network. Depending on the user’s network, device and proximity to the origin server, a single round-trip request can take seconds to complete. A single web page can require dozens of HTTP requests before it can render content, with requests often delaying one another due to a limited number of connections. To reduce round-trips, Akamai FEO uses several techniques to eliminate unnecessary requests such as consolidating multiple CSS and JavaScript files and using new caching features in HTML5.

File Versioning:

Many sites only cache content for a few hours due to concerns about old content used from the cache when new content is available. This practice all but eliminates the enormous performance benefit of caching for users returning to a site later – a very common occurrence. Akamai FEO uses a technique called File Versioning to solve this problem, by giving each file a unique name. If a resource changes, the old file will be not be referenced anymore, so all content can be cached indefinitely.

Adaptive Consolidation:

Web pages hold dozens or hundreds of resources, each requiring an HTTP transaction to be fetched from the server. These transactions have considerable overhead, are time-consuming, and cannot be done all at once. Consolidation is the technique of combining multiple resources into one. Akamai FEO applies smart consolidation of JavaScript and CSS files, reducing the overall page size and accelerating load time.

Advanced Adaptive Consolidation:

While simple consolidation reduces the number of HTTP requests for first time visitors, it often leads to more bytes downloaded. The same CSS & JavaScript files are included in many different combinations across a website’s pages. Each combination is consolidated into a separate package, resulting in the same content being downloaded multiple times. Akamai FEO uses Adaptive Consolidation to fetch multiple files with one request, but cache each resource separately, thus avoiding redundant downloads. The result is the optimal number of requests, without increasing the payload.

Inlining:

Many of the external images and scripts on a page are small. Often so small, that the HTTP overhead to fetch them exceeds their own size. Inlining is the technique of embedding these resources directly into the page or resources that references them. The overall page size decreases with the reduced HTTP overhead, and the inlined resources are processed faster. Akamai FEO automatically embeds resources into pages and CSS files based on size and other considerations.

Advanced JavaScript Inlining

Inlining JavaScript into HTML is almost always faster the first time a page is loaded as the browser doesn’t have to make additional requests for JavaScript. However, basic JavaScript inlining can make subsequent page loads slower because the browser must download larger HTML (containing the inlined JS) rather than loading the JavaScript directly from the browser cache. Advanced JavaScript Inlining offers the best of both worlds. On first view JavaScript is inlined and stored in HTML5 localStorage. On the second view the JavaScript is loaded from HTML5 localStorage.

On-demand Image Loading:

Pages are often much larger than the size of the browser window, especially on a small mobile screen. And yet, the browser will load all the images on the page regardless of which ones are visible to the end user. Akamai FEO’s On-demand Image Loading optimization will cause a page to only load the images that are visible within the current viewport. As the user scrolls down, new images are loaded on demand. On-demand Image Loading helps improve page load time and also reduces bandwidth for cases where a user doesn’t actually scroll down a page.
**HTML5 Advanced Cache:**
Caching previously downloaded page elements on users’ browsers is one of the best ways to improve performance of repeat views. However, since the cache size on mobile browsers is very small, few files last in the cache until the user returns to the site. Even on desktop browsers, users cycle through their cache in days. Akamai FEO cache optimization leverages a new capability in HTML5 to bypass the shared browser cache and create a dedicated cache for each site.

**REDUCES BYTES**
The math is simple: the larger a web page (measured by bytes), the longer it will take a browser to render the content. Akamai FEO keeps file size in check by adjusting image formats, improving cache management, compressing files, and removing metadata from files, (such as comments, whitespaces and image metadata). Akamai FEO also helps avoid excessively large images, maintaining image quality at the edge of what the user can perceive, on a small and large screen.

Optimizations that contribute to page-size shrinkage include:

**Compression:**
Compression is an effective way to reduce page sizes and resources without impacting their content. HTML, JavaScript and CSS files are textual and compress very well, as do AJAX responses. Compression can often reduce 60%-80% of the total page size. Akamai FEO automatically compresses all text content, and sometimes even images, while ensuring compressed content is only served to clients that support it.

**Lossless Image Compression:**
Images often hold significant amounts of metadata, ranging from where a picture was taken to layering of its graphics for easy future editing. In addition, images are often saved in the format most suited for image editing software – not for the web. Image Compression removes metadata and uses the optimal image format for the web, resulting in smaller file sizes with no difference in image quality.

**Resize images to HTML dimensions:**
HTML and CSS designs are often out of sync with the management of images. Files are often uploaded to a Content Management System without knowing the size they’ll be displayed in, and the same image is often used in multiple sizes. Akamai FEO makes sure the browser will not download a large image, just to have it resized by the browser due to styles and markups. Such images will be resized ahead of time, avoiding downloading of redundant bytes.

**Browser-Specific Image Optimizations - WebP (Chrome/Opera), JPEG XR (IE9+) & JPEG2000**
New browser-specific image formats allow the same quality of experience to be delivered in a reduced payload size, when compared to standard JPEG. Akamai FEO recognizes situations where JPEG images can be replaced by browser-specific image formats resulting in a significantly better user experience.

**Minification:**
HTML, JavaScript and CSS files contain comments and whitespace that, while useful to developers, are not needed for the page’s operation. Minification is the process of removing such components and reducing the total download size. Akamai FEO automatically minifies all page resources, reducing the size of pages without modifying their functionality.

**Responsive Images:**
Most images are formatted for desktop browsers on larger screens. Such images are unnecessarily large for mobile devices. Akamai FEO’s Responsive Images technology automatically delivers an optimally-sized image for the phone, tablet or desktop browser that requested it, resulting in much faster page load with no perceived difference to the user.

**Intelligent Image Quality:**
The improvements in digital photography often lead to web page images saved at a higher pixel density than what most displays can support. Such images make the pages bigger and thus hurt the user experience, while the user cannot appreciate the higher quality picture. Akamai FEO optimizes the pixel density to reduce size without a perceptible impact on quality.
ACCELERATES RENDERING

Processing a web page is a complicated process. Browsers employ complex logic during load time, making decisions such as which files to download serially vs. in parallel, which resource types block rendering, and how to manage their connections. At the same time, they need to parse and execute complicated HTML, CSS and JavaScript code, which is often not well defined.

Unfortunately, the browser doesn’t know sites in advance and is forced to employ generic logic when processing a page. This logic changes between old and new browsers, is limited by backward compatibility, and is not customized to a site.

Akamai FEO combines the knowledge it has of a site—learned by automated analysis, inspection and user configuration—to identify the best way to load that page. Techniques like deferring print stylesheets, keeping social buttons from blocking rendering and prefetching the next page are used to guide the browsers into doing the right thing. As a result, users can get a truly fast user experience, attuned to their needs.

Optimizations that contribute to accelerate rendering include the following:

**EdgeStart:**

EdgeStart reduces the time to deliver the first part of the HTML response, which allows the browser to download important resources such as JS, CSS and some images earlier to enhance the user experience. This optimization can be particularly helpful for mobile sites and for end users distant from the origin. EdgeStart leverages Akamai’s global footprint of Edge servers to get a response back to the end user as quickly as possible.

**Asynchronous JavaScript and CSS:**

When processing a page, browsers usually process JavaScript and CSS files as blocking resources. This means images and other resources further down the page aren’t downloaded until the JavaScript and CSS files are fully downloaded and executed. Even rendering is delayed until all CSS and preceding JavaScript is fetched and run. Akamai FEO’s Async JavaScript and CSS processing optimization modify the way scripts and style sheets are embedded into the page, making the browser process scripts, style sheets and other resources in parallel. This improves load time and makes the page start rendering much earlier, while still not showing unstyled content.

**Streaming Consolidation:**

If you consolidate 10 JavaScript files into one, you make the first one significantly slower. When downloading JavaScript and CSS, browsers do not process the content until the entire file is downloaded. As a result, consolidating resources, while it reduces HTTP requests, sometimes makes the page slower. Akamai FEO combines IFrames and its Async optimizations to overcome this problem, processing incoming JavaScript and CSS as it arrives, without additional HTTP requests.

**Invoke Click On-Touch:**

Most touch screen browsers fire an on-touch event roughly 300 milliseconds before firing an on-click event, in order to distinguish between pinch, zoom and other touch operations. Akamai FEO’s Invoke Click On-Touch leverages the on-touch event to simulate fast clicks, even on a touchscreen, resulting in a significantly more responsive page. This method applies to iPhone, iPad, and Android devices.

**JavaScript Pre-Execution:**

Executing JavaScript on mobile devices can be slow. On average, mobile browsers take ten times longer than desktop browsers to process JavaScript. When JavaScript Pre-Execution is enabled, Akamai FEO executes much of the embedded JavaScript offline and provides the requesting browser with a mostly static page. Even fast-changing scripts and interaction scripts are made faster by “statifying” other scripts which generate content. Using this method reduces the amount of JavaScript that either the client or browser will need to execute at page-request time, lessening some of the mobile device’s processing burden.

**Page Prefetching:**

This optimization loads page content ahead of time based on users’ browsing patterns. For example, say 60% of visitors to a home page click the “Hot Deals” link. While a visitor views that home page, Akamai FEO can hint to the browser to load the “Hot Deals” page without waiting for the user to request it. Then when the user does click the link, the “Hot Deals” page will display immediately.
**HTML5 Resource Prefetching:**

Resource Prefetching aims to reduce the total time it takes to navigate multiple pages across a website. When enabled, this optimization allows a page to download the JavaScript and CSS resources from other configured policies. When the user accesses those future pages, the resources will already be stored in HTML5 local storage providing a faster overall experience.

**DNS Prefetching:**

In modern web pages, there are often dozens of calls made for content on external domains. Performing a DNS resolution for each individual hostname does not take long, but with enough resources served from external domains this can lead to a performance bottleneck. DNS prefetching bypasses this by performing the DNS resolutions before objects on external domains are actually requested.

**Image Placeholders:**

On-demand Image Loading downloads images as they are needed by end users. Image Placeholders improves the user scrolling experiences by using lower resolution placeholder images until the placeholders come into view and the full quality images are needed.

Akamai continually updates and improves its FEO optimization engine to reflect the state-of-the-art in FEO innovations as well as the constantly evolving browser and device landscape. This enables our customers’ development teams to focus simply on creating innovative content and features, rather than getting bogged down by the ever-changing details around the latest FEO optimizations and which versions of which browsers support what features.

**Domain Sharding:**

Browsers impose limits on the number of parallel connections per hostnames. Although with modern browsers these limits tend to be adequately high, older browsers can be restrictive. This leads to artificial dependencies as the browser waits for otherwise independent objects to be fully delivered before proceeding with further requests. Domain sharding gets around this limit by distributing requests over multiple subdomains, ensuring that the browser does not hit the limit of parallel requests per hostname.