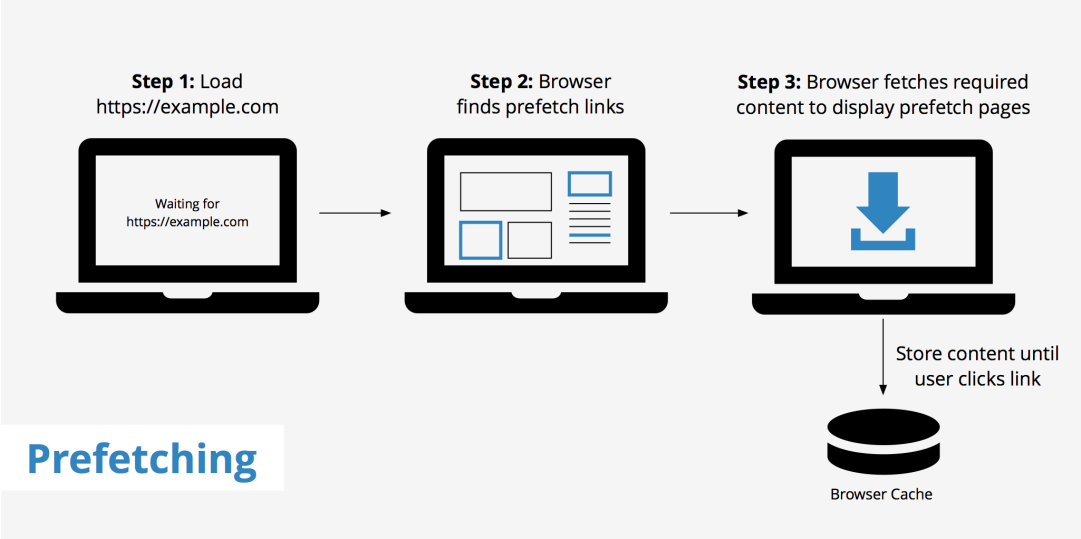
keycdn.com

What Is Prefetching and Why Use It - KeyCDN Support

6-7 minutes

What Is Prefetching and Why Use It

Updated on October 4, 2018



What is prefetching?#

There are a variety of ways web developers can help speed up a website. They can bring content closer to their visitors, optimize their content through compression, set high expiry times so that assets remain in their browser longer, and more. However, what if we were able to implement a way for content to already be almost instantaneously available to a visitor uponrequest - in comes prefetching.

Prefetching allows a browser to silently fetch the necessary

resources needed to display content that a user might access in the near future. The browser is able to store these resources in its cache enabling it to **deliver the requested data faster**. Therefore, once a web page has finished loading and the idle time has passed, the browser begins downloading other resources. Once a user clicks on a particular link that has already been prefetched, they will see the content instantly.

Types of prefetching#

There are three main types of prefetching that exist. The most popular and widely used method is link prefetching. However, DNS prefetching and prerendering are also useful options and each serves their own purpose.

Link prefetching#

Link prefetching, as discussed in the previous section, is a mechanism that allows the browser to fetch resources for content that is assumed the user will request. W3C defines it as such:

> The prefetch link relation type is used to identify a resource that might be required by the next navigation, and that the user agent SHOULD fetch, such that the user agent can deliver a faster response once the resource is requested in the future.

The browser will look for `prefetch` in either an HTML `<link>` or the HTTP header `Link`, for example:

- HTML: `<link rel="prefetch" href="/uploads/images/pic.png">`
- HTTP header: `Link: </uploads/images/pic.png>; rel=prefetch`

As can be seen in the image below, `prefetch` has been adopted by most major browsers with the exclusion of Safari, iOS Safari,

and Opera Mini.

It should be noted that prefetching only works for cacheable resources such as CSS, images, JavaScript, etc. However, once a cacheable resource has been prefetched and a user navigates to a page with that resource, it will be delivered very quickly as it was already fetched in the background.

One downside to link prefetching is that given the event that a user doesn't navigate to the page with the prefetched asset, their browser will have already unnecessarily fetched the asset in the background, thus increasing the size of its cache without any realized benefit.

DNS prefetching#

A DNS or domain name server converts IP addresses in readable website URLs such as yourwebsite.com. Whenever a user requests an asset being hosted on a particular domain they must perform a DNS lookup and find which domain name that IP address belongs to. This process takes time and the most DNS lookups that are required, the longer your visitors will be waiting for a page to load.

DNS prefetching allows the browser to perform the DNS lookups for links on a page in the background while the user browses the current page. This minimizes latency as when the user clicks on a link with DNS `prefetch` enabled, they do not have to wait for the DNS lookup to take place as it already has.

DNS `prefetch` can be added to a specific URL by adding the `rel= tag` to the link attribute like so: `<link rel="dns-prefetch" href="https://www.keycdn.com">`

Similar to link `prefetch`, DNS `prefetch` has also been adopted by most modern browsers.

Prerendering#

Prerendering is similar to prefetching in that it gathers the resources needed to display a page that the user may navigate to next. Although the main difference is that instead of just downloading the required resources, prerendering actually **renders the entire page in the background**. The page is hidden, although if the user navigates to the page, the hidden page will replace the current tab and will display the page the user requested.

The diagram below shows a visual demonstration of what happens when prerender is used.

Source: The Chromium Projects

Prerendering is the least supported type of prefetching across the major browsers. Supported only by IE, Edge, Chrome, Opera, and Chrome for Android.

Prefetching example#

Popular search engines such as Bing and Google are known for using `prefetch` to deliver results to users when a search is made. When the user enters their search query, the search engine goes ahead and delivers the results to the user. Based on which results a user typically visits (usually the first or second) the resources for these pages are then prefetched resulting in a **faster loading time** if the user clicks the link.

Although `prefetch` is a useful tool for speeding up the web, there are also a few cons to using it. For example, websites such as Google who use prefetch, are using the website owner's bandwidth to load the page that a user may **not actually visit**. The same goes for tracking page views in Google Analytics, the user may not actually visit the prefetched site, however, a page view will still be recorded.

CDNs and prefetching#

Using prefetching in conjunction with a CDN is good practice and often used to further speed up a website's load time. Although both involve a level of fetching data and storing it, they act in different capacities. A CDN fetches data from the origin server and caches it on an edge server near its visitors. On the other hand, prefetching retrieves data (either from a CDN's edge server or an origin server) and caches it in the browser.

Although both methods are different, they can work together to improve loading times. In fact, we're using prefetching on our main site as can be seen below.

Summary#

Prefetching is without a doubt a useful browser mechanism to make use of. It's ability to fetch resources to deliver content faster while the user browses their current page makes it an appealing feature for both website owners and viewers alike. For more information and a detailed list of frequently asked questions, consult Mozilla's prefetching FAQ page.