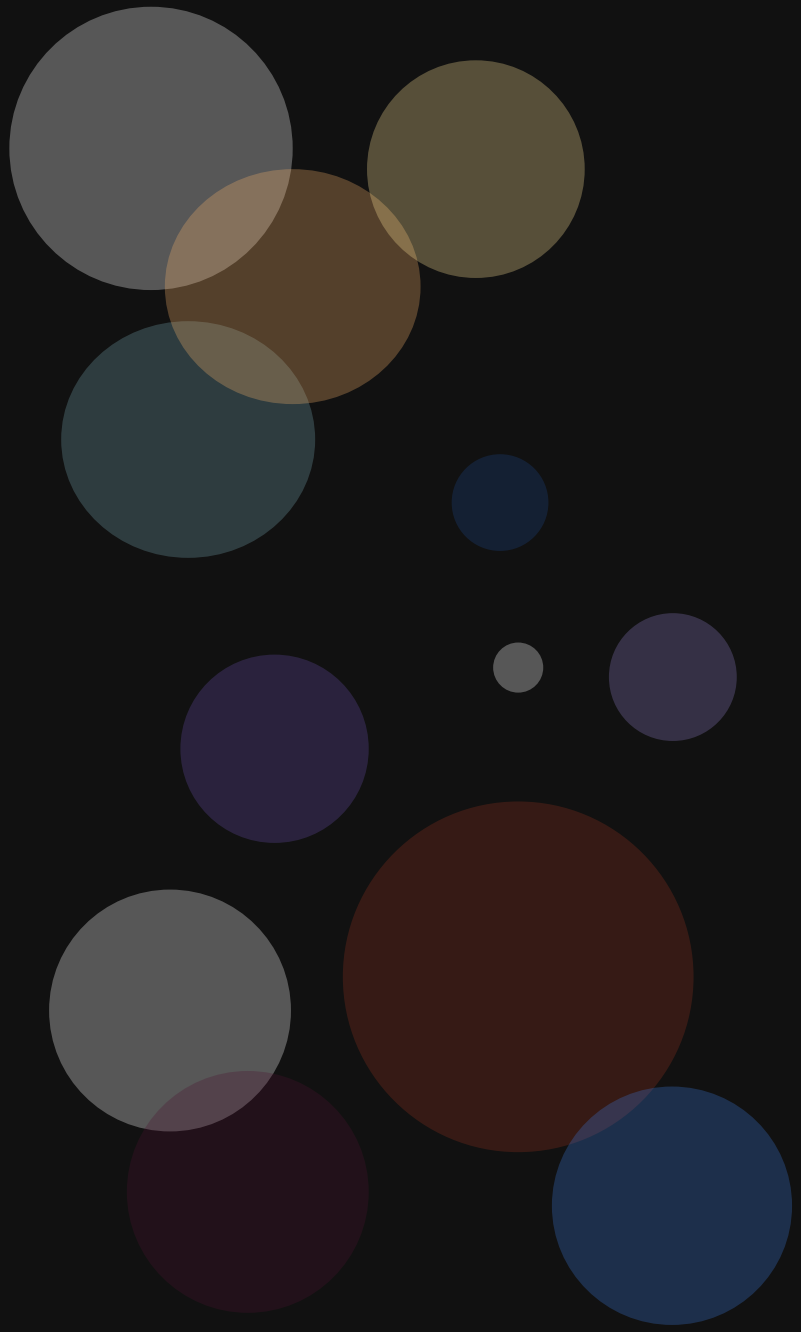


Resource hints

@贾正权 2019/05/10

Contents

- Preload（Candidate Recommendation）
- Prefetch（Working Draft）
- Preconnect（Working Draft）

Preload

Preload 提供了一种声明式的指令，让浏览器提前加载当前页面的资源但并不执行。

- 加载和执行分离，不阻塞渲染和 onload 事件
- 允许浏览器来**设定资源加载的优先级**

<link rel="preload">

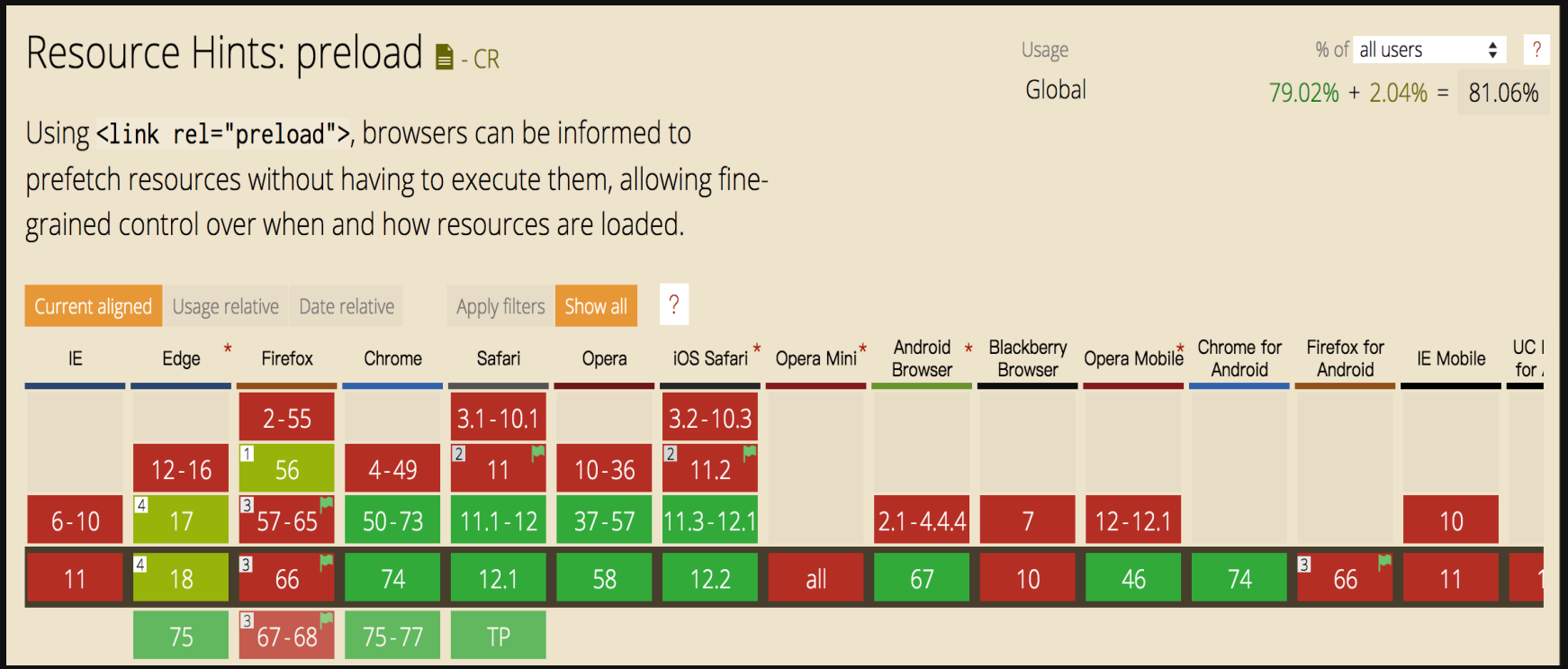
Preload

Preload 提供了一种声明式的指令，让浏览器提前加载当前页面的资源但并不执行。

- 加载和执行分离，不阻塞渲染和 onload 事件
- 允许浏览器来**设定资源加载的优先级**

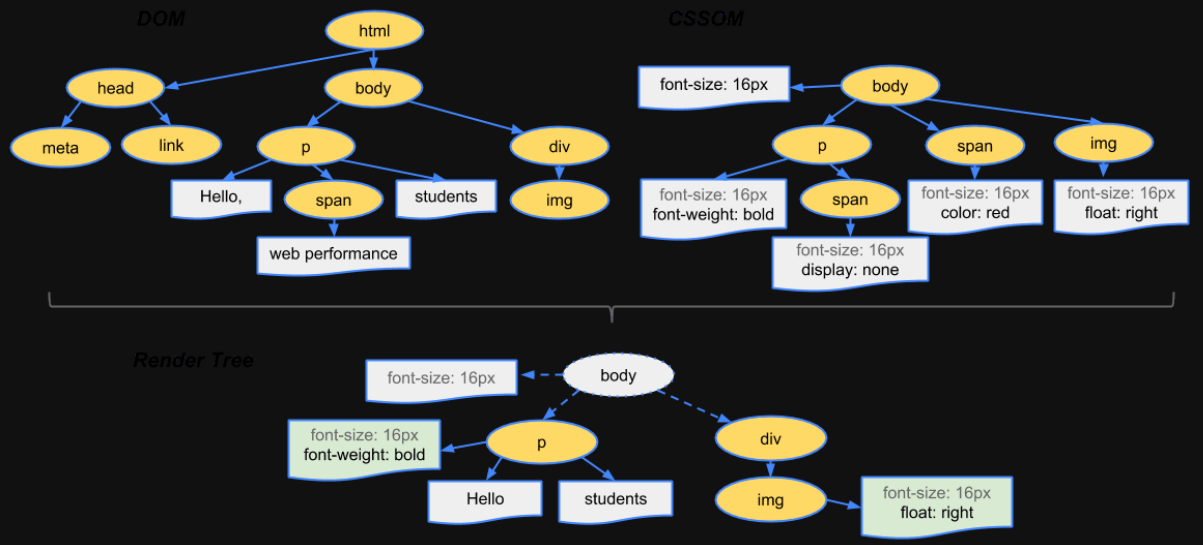
```
<link rel="preload">
```

Preload 现状



W3C Candidate Recommendation 26 October 2017

关键渲染路径



CSS 加载会阻塞页面渲染，即使 HTML 下载完成也要等待 CSS 下载完并解析才能开始构建 Render Tree。
JS 不仅能访问 DOM 还能操作样式，如果 JS 需要被执行时 CSSOM 还没有完成构建，JS 执行将会推迟。

Preload 使用

1. 使用 link 标签

```
<!-- Via markup -->
<link rel="preload" href="/css/mystyles.css" as="style">
<!-- Via JavaScript -->
<script>
    var res = document.createElement("link");
    res.rel = "preload";
    res.as = "style";
    res.href = "css/mystyles.css";
    document.head.appendChild(res);
</script>
```

2. 使用 HTTP 响应头

Link: <https://example.com/other/styles.css>; rel=preload; as=style

Preload 优化渲染

preload 的资源是下载与执行分离的，由于外部 CSS 样式表会阻塞渲染，JS 执行也会阻塞渲染。通过减少关键 CSS、JS 的数量，再通过 preload 加载首屏需要但是非关键的渲染资源来达到优化首屏渲染（可以认为非关键资源被延迟加载了）。

defer?

1. defer 会阻塞 DOMContentLoaded、onload 事件
2. defer 脚本的优先级是 low，preload 脚本优先级是 high

Preload 优化渲染

preload 的资源是下载与执行分离的，由于外部 CSS 样式表会阻塞渲染，JS 执行也会阻塞渲染。通过减少关键 CSS、JS 的数量，再通过 preload 加载首屏需要但是非关键的渲染资源来达到优化首屏渲染（可以认为非关键资源被延迟加载了）。

defer?

1. defer 会阻塞 DOMContentLoaded、onload 事件
2. defer 脚本的优先级是 low，preload 脚本优先级是 high

	Layout-blocking	Load in layout-blocking phase	Load one-at-a-time in layout-blocking phase		
DevTools Priority	**Highest**	**High**	**Medium**	**Low**	**Lowest**
	Main Resource				
	CSS (match)				CSS (mismatch)
		Script (early** or not from preload scanner)	Script (late**)	Script (async)	
	Font	Font (preload)			
		Import			
		Image (in viewport)		Image	
				Media	
				SVG Document	
					Prefetch
		Preload*			
		XSL			
	XHR (sync)	XHR/fetch* (async)			

资源加载优先级（Chrome 46+）

Preload & 跨域

浏览器加载并解析了 CSS 并不会直接下载依赖的字体资源，加载字体资源仍然要等到使用了 webfont 的选择器应用到了某些 DOM 节点上。

字体资源必须添加 crossorigin 属性，若不指定 crossorigin 属性（即使同源），则会采用匿名模式的 CORS 去加载，这将可能导致资源被加载 2 次。

Preload & 跨域

浏览器加载并解析了 CSS 并不会直接下载依赖的字体资源，加载字体资源仍然要等到使用了 webfont 的选择器应用到了某些 DOM 节点上。

字体资源必须添加 crossorigin 属性，若不指定 crossorigin 属性（即使同源），则会采用匿名模式的 CORS 去加载，这将可能导致资源被加载 2 次。

Preload 能带来的提升

Housing.com 使用 preload 加载关键脚本减少 10% TTI。
Shopify.com 使用 preload 加载字体减少 50% 文本渲染时间
https://video.twimg.com/tweet_video/C7dcmxaUwAAUhPX.mp4。
Chrome Data Saver team 使用 preload 加载 CSS 和 JS 的页面平均首屏绘制时间提升 12%。

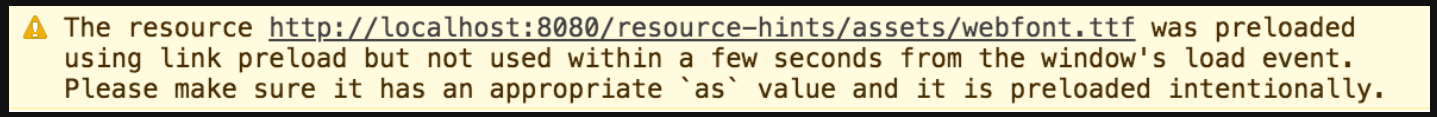
混用 Preload

preload 和 prefetch 混用的话，并不会复用利用资源，而是会重复加载。如果发生对用一资源 preload 和 prefetch 的话，会带来双倍的网络请求

```
<link rel="preload" href="a.woff" as="font">
<link rel="prefetch" href="a.woff" as="font">
```

滥用 Preload

确保 preload 加载的资源是页面所必须的资源，否则不要使用 preload 进行加载浪费带宽，这种时候 Chrome 会有一个警告。



Preload 检测

如果浏览器不支持 preload，那么一些都会失效，为了支持检测浏览器是否支持 preload，甚至更改了 DOM 规范，以提供接口支持查询 DOM 节点 rel 属性所支持的 keywords

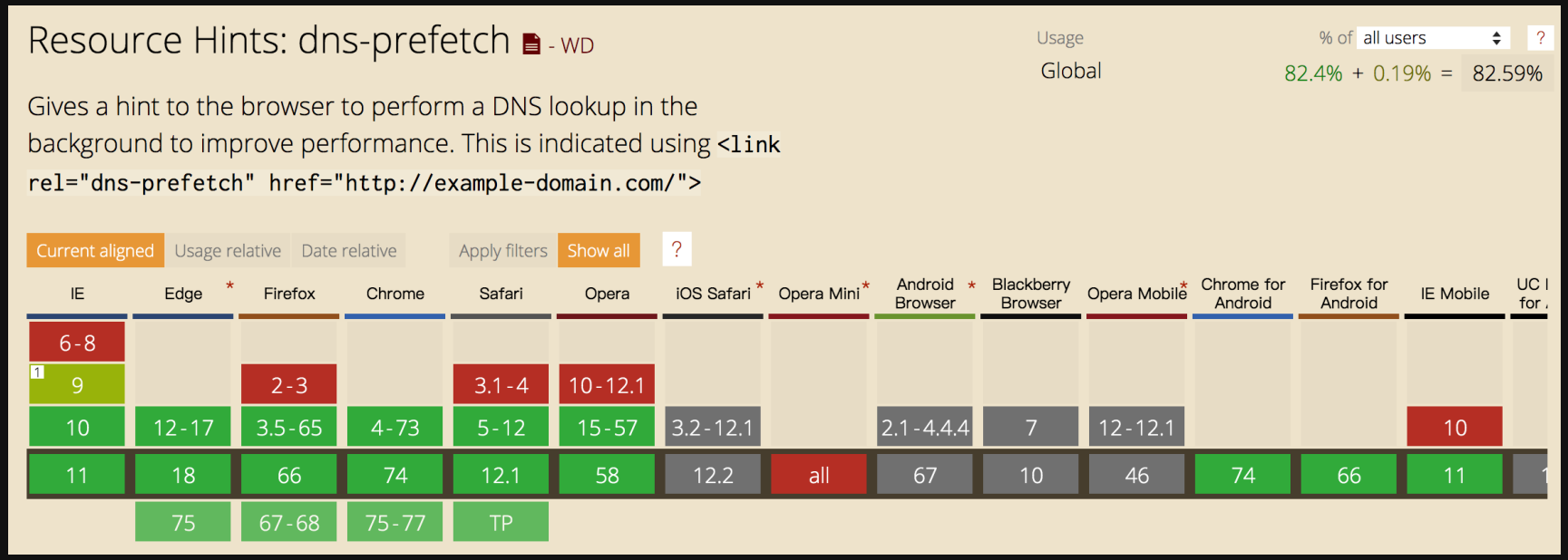
```
var DOMTokenListSupports = function(tokenList, token) {
    if (!tokenList || !tokenList.supports) {
        return;
    }
    try {
        return tokenList.supports(token);
    } catch (e) {
        if (e instanceof TypeError) {
            console.log("The DOMTokenList doesn't have a supported tokens list");
        } else {
            console.error("That shouldn't have happened");
        }
    }
};
```

Prefetch

- DNS Prefetch
- Link Prefetch
- Prerender

W3C Working Draft 07 March 2019

DNS Prefetch



X-DNS-Prefetch-Control 头控制着浏览器的 DNS 预读取功能。 DNS 预读取是一项使浏览器主动去执行域名解析的功能，其范围包括文档的所有链接，无论是图片的，CSS 的，还是 JavaScript 等其他用户能够点击的 URL。

DNS Prefetch 使用

```
1. link 标签
<meta http-equiv="x-dns-prefetch-control" content="on">

2. HTTP Header
X-DNS-Prefetch-Control: on/off

强制查询指定主机名
<link rel="dns-prefetch" href="//www.example.com">
```

另外，默认情况下，通过 HTTPS 加载的页面上内嵌链接的域名并不会执行预加载。在 Firefox 浏览器中，可以通过设置 network.dns.disablePrefetchFromHTTPS 值为 false 来改变这一默认行为。

Link Prefetch

Link Prefetch 是一种浏览器机制，其利用浏览器空闲时间来下载或预取用户在不久的将来可能访问的文档。网页向浏览器提供一组预取提示，并在浏览器完成当前页面的加载后开始静默地拉取指定的文档并将其存储在缓存中。当用户访问其中一个预取文档时，便可以快速的从浏览器缓存中得到。

```
1. link 标签
<link rel="prefetch" href="/images/big.jpeg">

2. HTTP Header
Link: </images/big.jpeg>; rel=prefetch
```

浏览器检查所有这些预取提示，并将每一个独立的请求排到队列之中，然后浏览器空闲时将对这些请求进行预取。

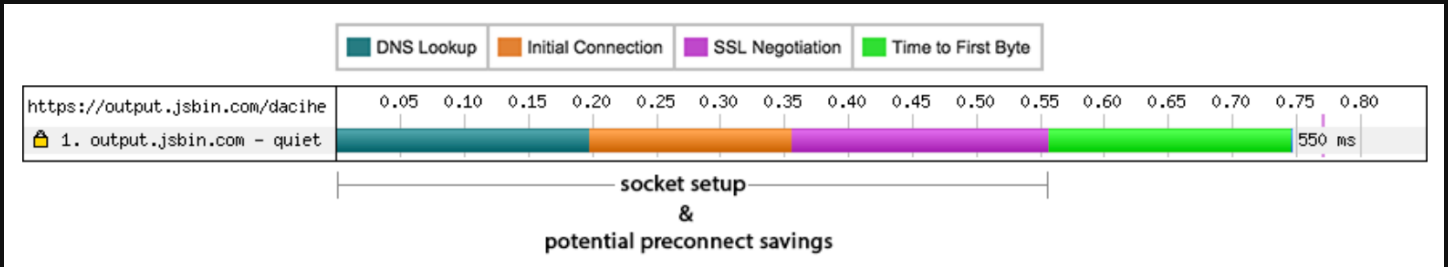
Prerender

下载下一个可能访问页面的所有资源，类似于新开了一个 tab 来加载一个页面，一般情况下我们很难预测用户下一步将会访问那个页面，prerender 一般应用的场景也会比较少。

<link rel="prerender" href="//example.com/next-page.html">

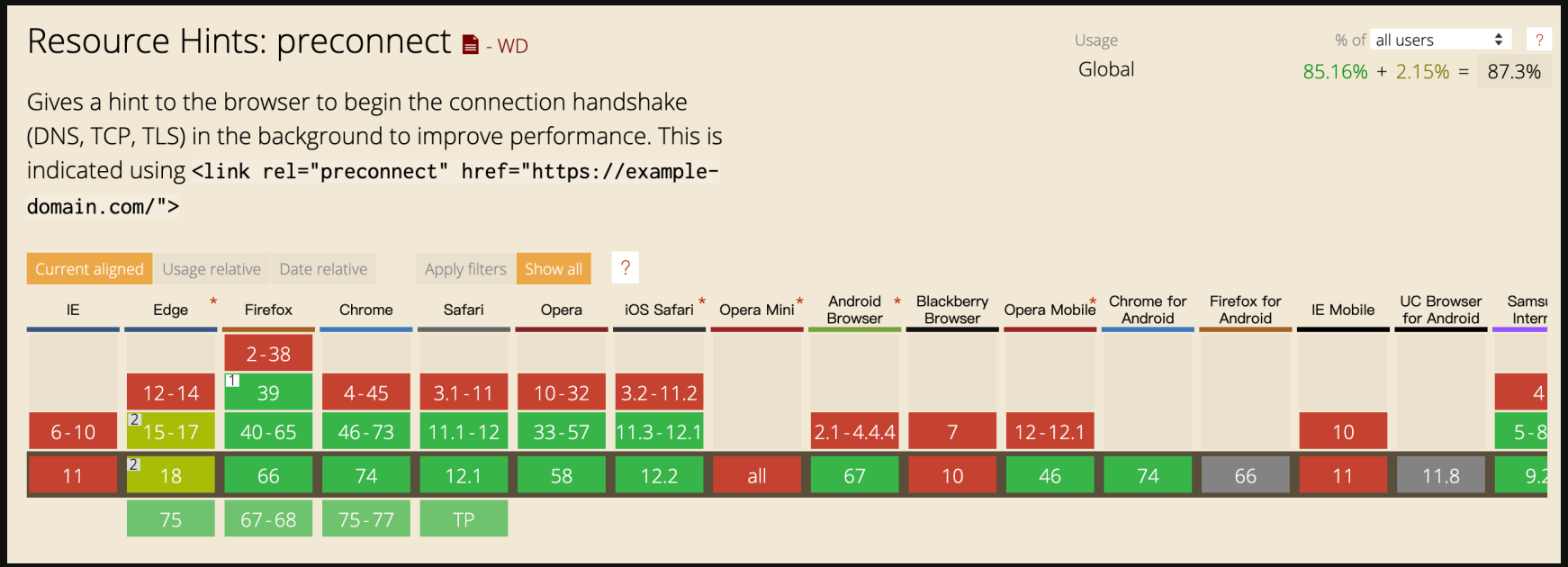
Preconnect

在实际请求字节到达服务器之前，一个 HTTP 请求会经历 DNS 解析、TCP 握手、TLS 握手（HTTPS ），连接建立的过程是需要一定时间的。尤其在移动端 DNS 解析可能上百毫秒。



我们可以通过 preconnect 指令告诉浏览器在启动实际请求之前需要哪些 socket 连接并提前建立，以达到减少 RTT 提升资源加载速度。

Preconnect 現状



W3C Working Draft 07 March 2019

Preconnect 使用

1. 通过 Link 标签

```
<link href='https://fonts.gstatic.com' rel='preconnect' crossorigin>
<link href='https://fonts.googleapis.com/css?
family=Roboto+Slab:700|Open+Sans' rel='stylesheet'>
```

2. 通过 HTTP Header

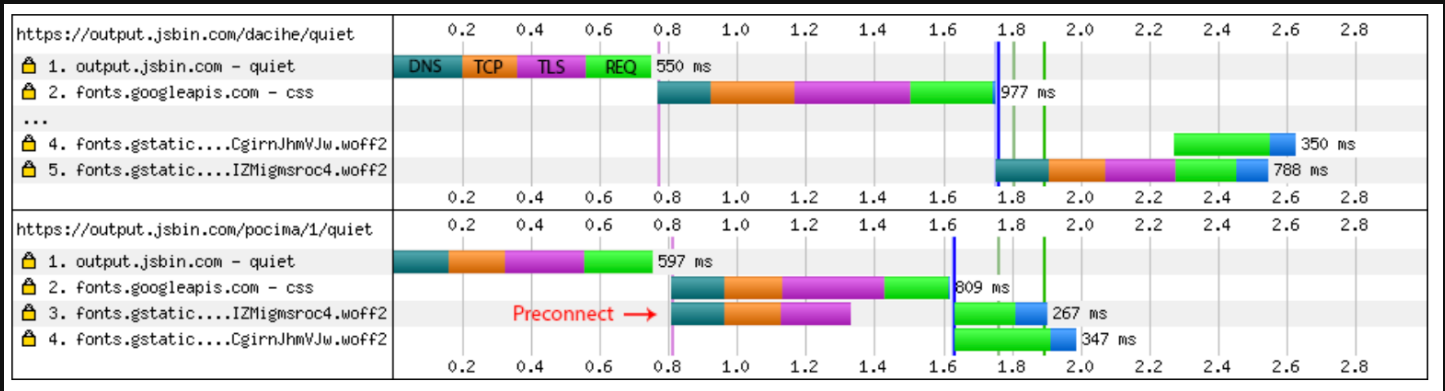
和 preload 一样 preconnect 也可以通过 HTTP Header 来让浏览器进行 preconnect，preconnect 中的连接不会阻塞页面中的请求，它是浏览器单独维护的一个连接池。

```
add_header Link '<https://google.com>; rel=preconnect'
```

3. 通过 JavaScript

```
function preconnectTo(url) {
    var hint = document.createElement("link");
    hint.rel = "preconnect";
    hint.href = url;
    document.head.appendChild(hint);
}
```

Preconnect 带来的提升




```
<link href='https://fonts.gstatic.com' rel='preconnect' crossorigin>
<link href='https://fonts.googleapis.com/css?family=Roboto+Slab:700|Open+Sans'
rel='stylesheet'>
```

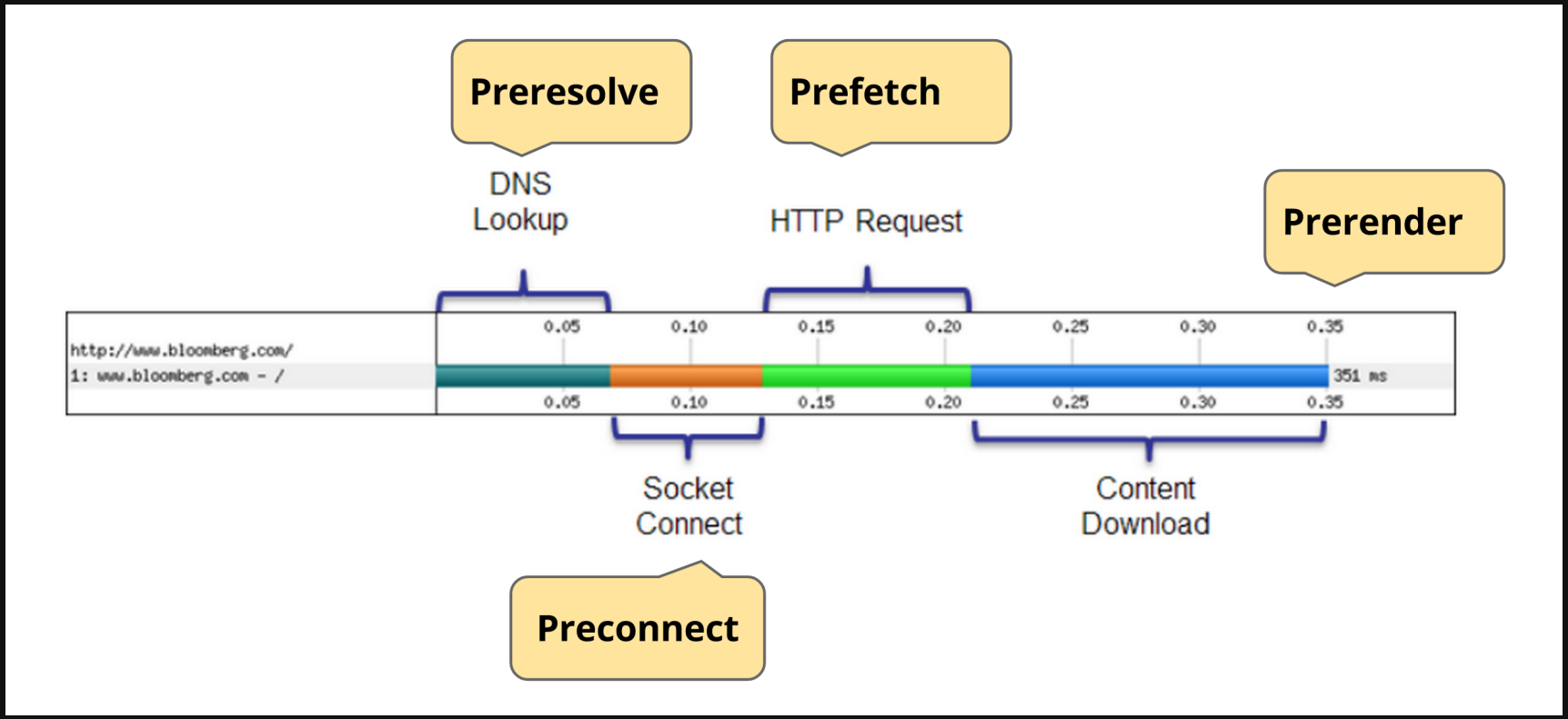
通过 preconnect fonts.gstatic.com，减少了 0.5s 延迟

合理使用 Preconnect

浏览器和服务器创建 socket 连接都是需要成本的，如果 preconnect 了没有使用到的 origin 这将造成资源浪费也不是提升性能的实践手段。

preconnect 指令被浏览器视为优化提示，不一定每一个 preconnect 指令都会起作用，浏览器还可能会调整策略比如只进行 DNS 查找或者 DNS 查找 + TCP 连接建立，这受影响于浏览器自身的实现。

Pre-*


Preresolve
Prefetch
DNS
Lookup
HTTP Request
Prerender
0.05
0.10
0.15
0.20
0.25
0.30
0.35
http://www.bloomberg.com/
1: www.bloomberg.com - /
351 ms
Socket
Connect
Content
Download
Preconnect

Reference

- https://www.w3.org/TR/resource-hints/
- https://www.keycdn.com/blog/resource-hints
- https://medium.com/reloading/preload-prefetch-and-priorities-in-chrome-776165961bbf
- https://www.igvita.com/2015/08/17/eliminating-roundtrips-with-preconnect/
- https://docs.google.com/presentation/d/18zlAdKAxnc51y_kj-6sWLmnjl6TLnaru_WH0LJTjP-o/present?slide=id.g120f70e9a_025

Thanks