02

OPEN ORIENTED

凹 凸 实 验 室

前端预加载

Paul

前端预加载

02

<link rel=“preload”>

与现有的类似技术的区别

02

<link rel="preload"> VS <link rel="prefetch">

与现有的类似技术的区别

02

<link rel="prefetch">

下一页

优先级低

Preload会更好

02

<link rel="preload">

浏览器可以设置正确的资源加载优先级，这种方式可以确保资源根据其重要性依次加载， 所以，Preload既不会影响重要资源的加载，又不会让次要资源影响自身的加载。

浏览器可以确保请求是符合内容安全策略的，比如，如果我们的安全策略是Content-Security-Policy: script-src 'self'，只允许浏览器执行自家服务器的脚本，as 值为 script 的外部服务器资源就不会被加载。

浏览器能根据 as 的值发送适当的 Accept 头部信息

浏览器通过 as 值能得知资源类型，因此当获取的资源相同时，浏览器能够判断前面获取的资源是否能重用。

Preload特点

02

无阻塞

优先级高

安全策略

提前加载

…

Preloader简介

02

HTML 解析器在创建 DOM 时如果碰上同步脚本，解析器会停止创建 DOM，转而去执行脚本。所以，如果资源的获取只发生在解析器创建 DOM时，同步脚本的介入将使网络处于空置状态，尤其是对外部脚本资源来说，当然，页面内的脚本有时也会导致延迟。

预加载器（Preloader）的出现就是为了优化这个过程，预加载器通过分析浏览器对 HTML 文档的早期解析结果（这一阶段叫做“令牌化（tokenization）”），找到可能包含资源的标签（tag），并将这些资源的 URL 收集起来。令牌化阶段的输出将会送到真正的 HTML 解析器手中，而收集起来的资源 URLs 会和资源类型一起被送到读取器（fetcher）手中，读取器会根据这些资源对页面加载速度的影响进行有次序地加载。

as 属性

02

as 属性的作用是告诉浏览器被加载的是什么资源，可能的 as 值包括：

- "script"
- "style"
- "image"
- "media"
- "document"

更多参考 https://fetch.spec.whatwg.org/#concept-request-destination

忽略 as 属性，或者错误的 as 属性会使 preload 等同于 XHR 请求，浏览器不知道加载的是什么，因此会赋予此类资源非常低的加载优先级

对字体的提前加载

02

web 字体对页面文字的渲染资至关重要，但却被深埋 CSS 中，即便是预加载器有解析 CSS，也无法确定包含字体信息的选择器是否会真正应用在 DOM 节点上

简单的一段代码就能搞定字体的预加载

```
<link rel="preload" href="font.woff2" as="font" type="font/woff2" crossorigin>
```

需要注意的一点是：crossorigin 属性是必须的，即便是字体资源在自家服务器上，因为用户代理必须采用匿名模式来获取字体资源

动态加载，但不执行


```
var link = document.createElement("link");
link.href = "myscript.js";
link.rel = "preload";
link.as = "script";
document.head.appendChild(link);
```

上面这段代码可以让你预先加载脚本，下面这段代码可以让脚本执行

```
var script = document.createElement("script");
script.src = "myscript.js";
document.body.appendChild(script);
```

标记语言的异步加载

下面代码利用preload的onload事件修改rel属性

```
<link rel="preload" as="style" href="asyncstyle.css" onload="this.rel='stylesheet'">
```

举个例子，你想尽可能快的加载一段统计页面访问量的代码，但又不愿意这段代码的加载给页面渲染造成延迟从而影响用户体验，关键是，你不想延迟 window 的 onload 事件

```
<link rel="preload" as="script" href="async_script.js" onload="var script = document.createElement('script'); script.src = this.href; document.body.appendChild(script);">
```

响应式加载

02

通过 Preload，我们可以提前加载资源，利用 media 属性，浏览器只会加载需要的资源

移动端&PC端加载地图示例

```
<link rel="preload" as="image" href="map.png" media="(max-width: 600px)">
<link rel="preload" as="script" href="map.js" media="(min-width: 601px)">
```

HTTP 头

Preload 还有一个特性是其可以通过 HTTP 头信息被呈现。也就是说上文中大多数的基于标记语言的声明可以通过 HTTP 响应头实现

```
Link: <thing_to_load.js>;rel="preload";as="script"
Link: <thing_to_load.woff2>;rel="preload";as="font";crossorigin
```

特征检查

Github有一段代码可供参考

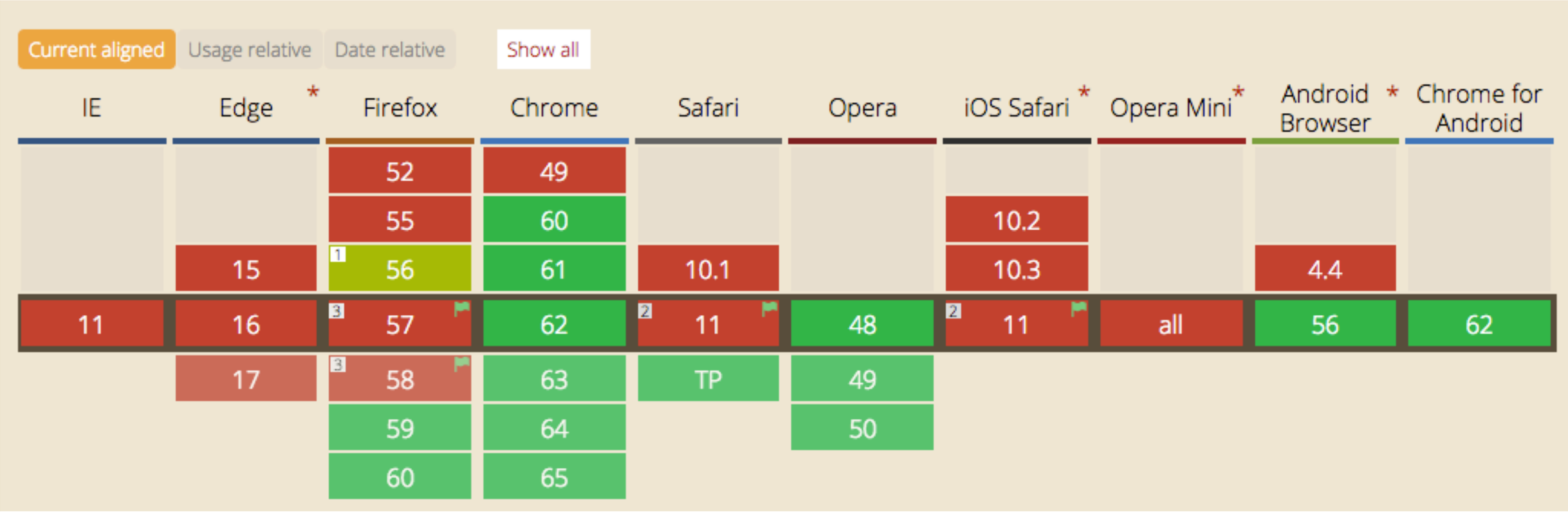
```
var DOMTokenListSupports = function(tokenList, token) {
  if (!tokenList || !tokenList.supports) {
    return;
  }
  try {
    return tokenList.supports(token);
  } catch (e) {
    if (e instanceof TypeError) {
      console.log("The DOMTokenList doesn't have a supported tokens list");
    } else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

var linkSupportsPreload = DOMTokenListSupports(document.createElement("link").relLis
if (!linkSupportsPreload) {
  // Dynamically load the things that relied on preload.
}
```

兼容性

02

浏览器支持情况



参考

https://www.w3.org/TR/preload/

http://www.jianshu.com/p/24ffa6d45087

https://developer.mozilla.org/en-US/docs/Web/HTML/Preloading_content

THANKS

FOR YOUR WATCHING