有一种优化叫做 Preload

奇技指南

今天给大家介绍一个性能优化利器：Preload。

背景

有时候为了提高网页初始加载的性能，我们会选择延迟一部分资源的加载和执行。

另一种情况是我们想要尽早加载资源，但是要等到合适的时机再执行。时机的影响因素包括依赖关系、执行条件、执行顺序等。

通常的做法是：

- 通过插入一个页面元素来声明一个资源（比如 img、script、link）。这种方式会将资源的**加载和执行耦合**。
- 用 AJAX 来加载资源。这种方式只有在时机成熟时才会加载资源，解决了执行时机问题。但是浏览器无法预解析，也就**无法提前加载**。另外如果页面有大量的阻塞脚本，就会造成**延迟**。

有没有办法既提前加载资源，又能解耦加载和执行呢？这时候就轮到 preload 大显身手啦。

什么是 Preload

preload 是一个预加载关键字。它显式地向浏览器声明一个需要提前加载的资源。使用方式如下：

- 在<head>中写入 <link rel="preload" href="some-resource-url" as="xx">（包括用 JS 创建<link>元素并插入到 <head>）
- 在 HTTP 头部加上 Link: <some-resource-url>; rel=preload; as=xx

当浏览器“看”到这样的声明后，就会以一定的优先级在后台加载资源。加载完的资源放在 HTTP 缓存中。而等到要真正执行时，再按照正常方式用标签或者代码加载，即可从 HTTP 缓存取出资源。

使用 Preload 加载资源的方式有以下几个特点：

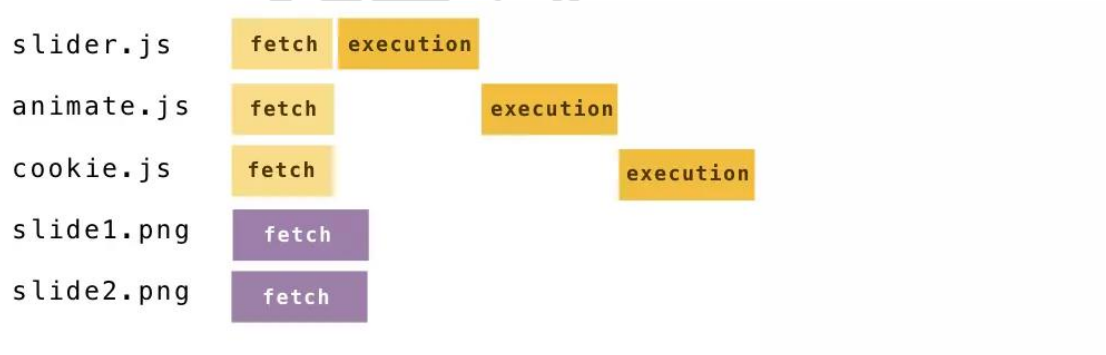
- 提前加载资源
- 资源的加载和执行分离
- 不延迟网页的 load 事件（除非 Preload 资源刚好是阻塞 window 加载的资源）

大家可能会问：Preload 跟其他提前加载资源以及加载和执行分离的方案有什么区别？

好的，满足你们的好奇心：

vs. 预测解析

浏览器很聪明，它可以在解析 HTML 时收集外链资源。并将它们添加到一个列表，在后台并行下载。预测解析也实现了提前加载以及加载和执行分离。如图所示：

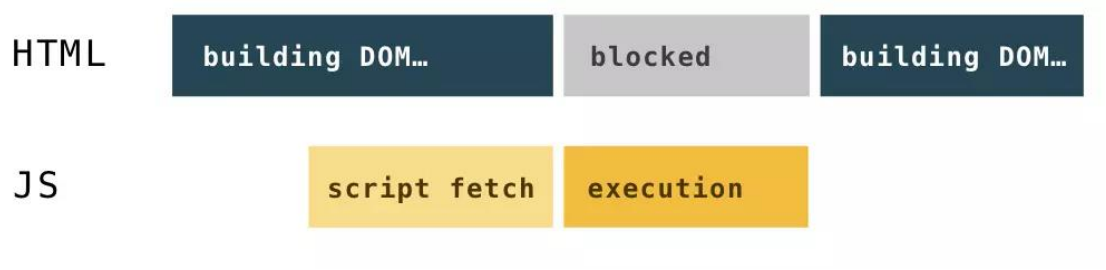


那么它跟 Preload 的区别是什么？

- 仅限于解析 HTML 时收集的外链资源。如果是程序里异步加载的资源无法提前收集到。
- 浏览器不暴露类似于 Preload 中的 onload 事件，也就无法更细粒度控制资源的执行。

vs. async

async 脚本是一加载完就立即执行，因此会阻塞 window 的 onload 事件。而且目前 async 仅限于脚本资源。



Preload 可以实现 async 一样的异步加载功能。且不局限于脚本。比如以下代码实现了加载完 CSS 文件立即作用到网页的功能：

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

注：如果页面存在同步阻塞脚本，等脚本执行完后，样式才会作用到网页。这样是因为 Preload 的资源不会阻塞 window 的 onload 事件。

vs. defer

defer 实现了资源的加载和执行分离，并且它能保证 defer 的资源按照在 HTML 里的出现顺序执行。跟 async 一样，目前也只能作用于脚本资源。

Preload 则适用多种资源类型。Preload 的资源也能像 defer 的资源一样延迟执行并保证执行顺序。

vs. Sever Push

HTTP/2 的 Server Push 也实现了资源的提前加载以及加载执行分离。不过 Server Push 节省了一个网络来回。我们可以结合 Server Push 优化 Preload，比如服务器识别到文档里的 Preload 的资源就主动推送 Preload 的资源。

如果不希望服务器推送，则可以增加 nopush 属性：

```
Link: </app/style.css>; rel=preload; as=style; nopush
```

另外 Server Push 只能推送同域资源。而 Preload 则可以支持**跨域**资源。

何时使用 Preload

任何你想要先加载后执行，或者想要提高页面渲染性能的场景都可以使用 Preload。

典型用例：

- 在单页应用中，提前加载路由文件，提高切换路由时的渲染速度。现在大型的单页应用通常会异步加载路由文件。当用户切换路由时再异步加载相应的模块存在性能问题。可以用 Preload 提前加载，提升性能。
- 提前加载字体文件。由于字体文件必须等到 CSSOM 构建完成并且作用到页面元素了才会开始加载，会导致页面字体样式闪动（FOUT，Flash of Unstyled Text）。所以要用 Preload 显式告诉浏览器提前加载。假如字体文件在 CSS 生效之前下载完成，则可以完全消灭 FOUT。

浏览器兼容性

Preload 的兼容性如下：

（图片来源：https://caniuse.com/#search=preload）

在 PC 上实现了 Preload 的浏览器包括： Chrome 50+，Saferi 11.1+ 。Edge 17+支持 HTML 方式，不支持 HTTP header 方式。移动端：iOS Safari 11.4+，Android Chrome 67

不支持 Preload 的浏览器会自动忽略它，并采用普通的加载方式。因此可以将此功能作为一种渐进增强方式用到我们的网页应用中。