CSS/JS 阻塞 DOM 解析和渲染

CSS DOM JavaScript 性能 2016/11/26

我们知道页面样式一般写在HTML头部，而页面脚本放在HTML尾部。 这是因为脚本和样式会阻塞DOM渲染。 本文具体分析了包括脚本和样式在内的资源元素对DOM渲染的影响， 并给出具体的示例代码。

> 本文只讨论服务器端渲染的DOM（以下称为同步渲染）资源载入时机。 关于动态插入HTML标签（异步渲染）的阻塞情况请参考 异步渲染DOM元素的加载时机一文。

TL;DR

- CSS（外链或内联）会阻塞**整个**DOM的渲染（Rendering），然而DOM解析（Parsing）会正常进行
- 很多浏览器中，CSS会延迟脚本执行和`DOMContentLoaded`事件
- JS（外链或内联）会阻塞**后续**DOM的解析（Parsing），延迟 `DOMContentLoaded`，后续DOM的渲染（Rendering）也将被阻塞
- JS前的DOM可以正常解析（Parsing）和渲染（Rendering）

CSS阻塞DOM渲染

无论是外链CSS还是内联CSS都会阻塞DOM渲染（Rendering），然而DOM解析（Parsing）会正常进行。 这意味着在CSS下载并解析结束之前，它后面的HTML都不会显示。 这也是为什么我们把样式放在HTML内容之前，以防止被呈现内容发生样式跳动。 当然代价就是显示延迟，所以性能攸关的站点都会内联所有CSS。

TOP

然而，很多浏览器中CSS还会延迟脚本执行和`DOMContentLoaded`事件触发（该事件就是jQuery的dom ready）。 下表列出了不同浏览器是否会延迟脚本执行，具体的解释可参考 CSS载入与DOM ContentLoaded事件延迟 一文。

渲染引擎	样式表之前的脚本	样式表之后的外部脚本	样式表之后的行内脚本
Presto (Opera)	否	否	否
Webkit (Safari, Chrome)	否	是	是
Gecko (Firefox)	否	是	是
Trident (MSIE)		是	是

有些情况下，可以尝试添加媒体查询来避免不必要的阻塞。 尤其是响应式站点可以做此优化：

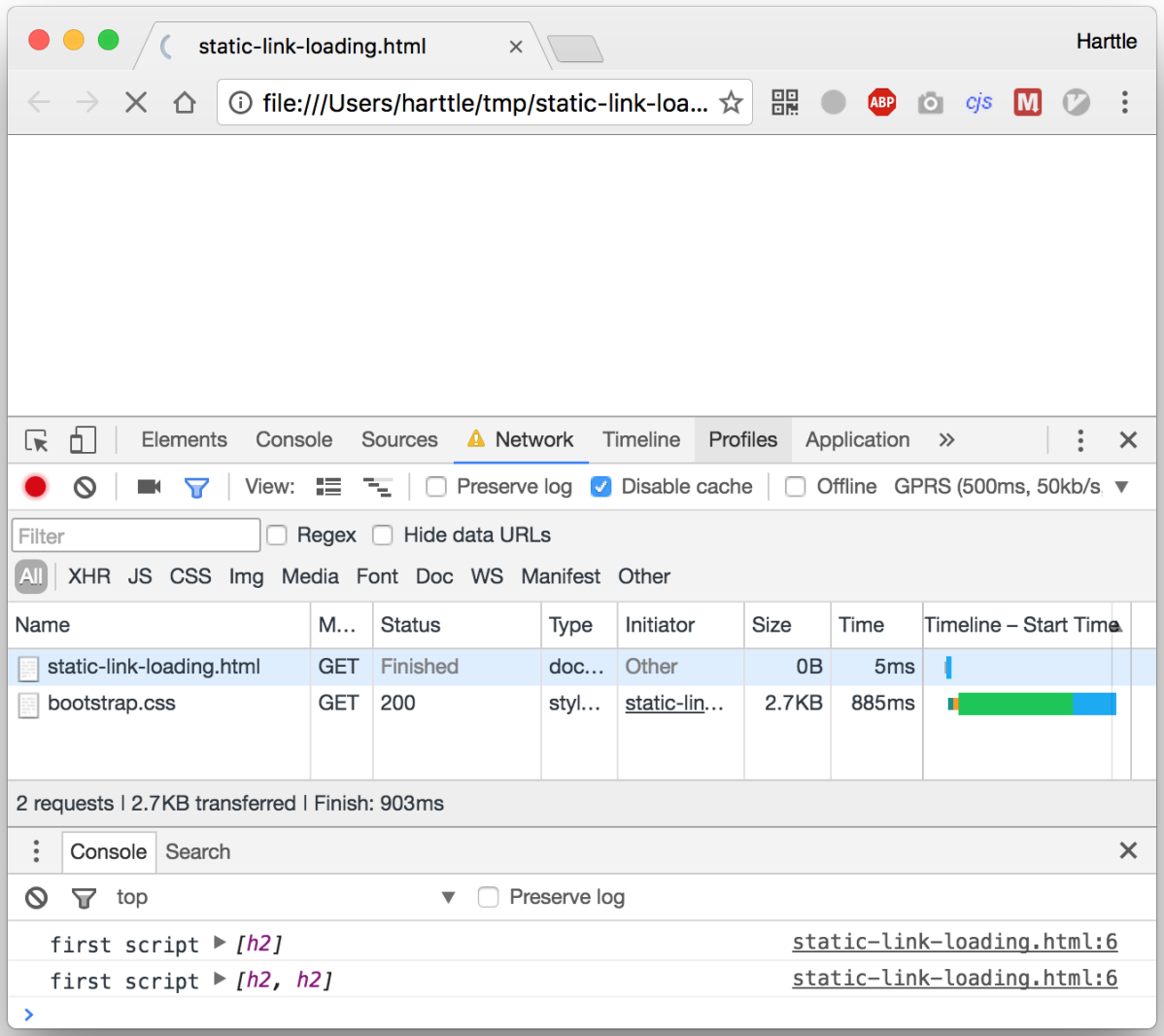
```
<link href="other.css" rel="stylesheet" media="(min-width: 40em)">
```

CSS阻塞DOM渲染：案例

为了验证CSS阻塞渲染但不阻塞解析以及脚本延迟行为，设计下列HTML。 同步和异步地打印当前DOM内容，以及在样式表后添加测试脚本。

```
<html>
<body>
  <h2>Hello</h2>
  <script>
    function printH2(){
        console.log('first script', document.querySelectorAll('h2'));
    }
    printH2();
    setTimeout(printH2);
  </script>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0-a
  <h2>World</h2>
  <script> console.log('second script'); </script>
</body>
</html>
```

需要打开Chrome开发者工具的`Disable Cache`和`Throttling`来模拟较慢的网络。 然后在样式表载入过程中可以观察到以下现象：



1. 两个`<h2>`标签均为显示，说明样式表会阻塞和延迟整个DOM的渲染。
2. 第一次输出只有一个`<h2>`，说明脚本执行会阻塞 DOM 解析（感谢@huahua指正）。
3. 第二次输出有两个`<h2>`，说明样式载入过程中DOM已解析完毕，即样式表不会阻塞DOM解析。
4. `"second script"`未被打印出来，说明在Chrome中样式表之后的行内脚本被延迟了。

JS 阻塞 DOM 解析

TOP

> 上文都在讲渲染，这里讲解析。解析是指浏览器生成 DOM 树结构（此时用户不一定能看到，但脚本比如 `querySelectorAll` 可以访问到）；渲染是指浏览器把 DOM 树与 CSS 结合进行布局并绘制到屏幕（此时用户是可以看到的）。

不论是内联还是外链JavaScript都会阻塞后续DOM解析（Parsing），`DOMContentLoaded` 事件会被延迟，后续的 DOM 渲染（Rendering）也会被阻塞。 这意味着脚本执行过程中插入的元素会先于后续的 HTML 展现，即使脚本是外链资源也是如此。 由于 JavaScript 只会阻塞后续的 DOM，前面的 DOM 在解析完成后会被立即渲染给用户。 这也是为什么我们把脚本放在页面底部：脚本仍在下载时页面已经可以正常地显示了。

但浏览器的载入标识仍然会提示页面正在载入，这件事情其实可以Hack， 见异步脚本载入提高页面性能一文。

其实除了脚本之外，现代浏览器上外部样式表的加载也会延迟 `DOMContentLoaded`， 虽然不可思议但这是正在发生的事实，请参考：外部样式表与DOMContentLoaded事件延迟一文。

> 上一篇：异步渲染的下载和阻塞行为
>
> 下一篇：使用 rem 提供一致的字体大小
>
> 推荐阅读：浏览器的 16ms 渲染帧

TOP