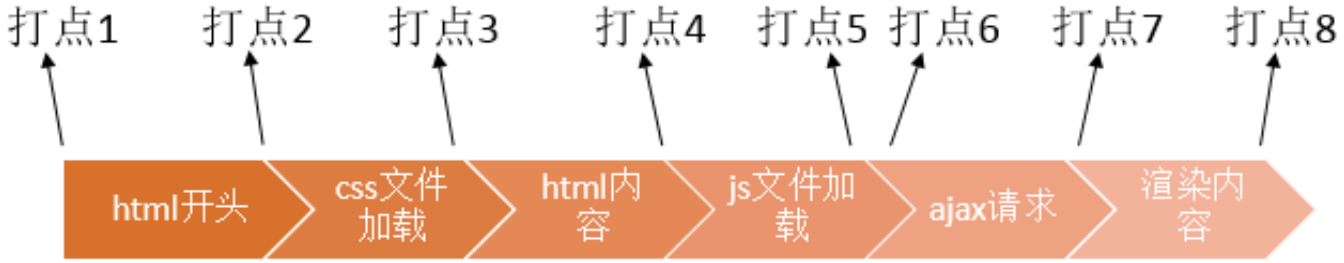
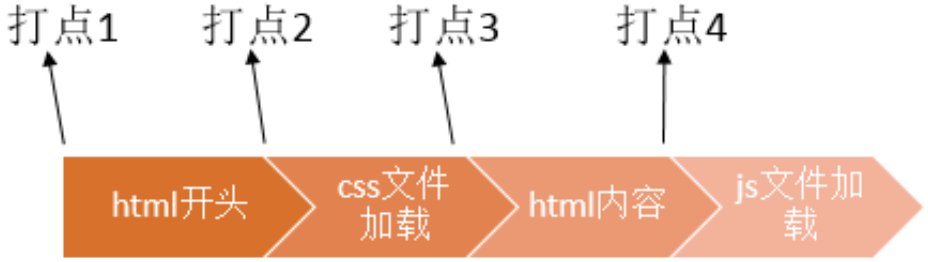
做移动web页面，受移动网络网速和终端性能影响，我们经常要关注首屏内容展示时间（以下简称首屏时间）这个指标，它衡量着我们的页面是否能在用户耐心消磨完之前展示出来，很大程度影响着用户的使用满意度。

怎么获取首屏时间呢？

我们经常要先问自己：页面是怎么加载数据？
A：加载完静态资源后通过ajax请求去后台获取数据，数据回来后渲染内容



在每个点打上一个时间戳，首屏时间 = 点8 – 点1；
B：使用后台直出，返回的html已经带上内容了



此时首屏时间 = 点4 – 点1。

注：1. 打了这么多个点，是因为当我们收集到首屏时间之后，要去分析到底是哪一段是性能瓶颈，哪一段还有优化空间，所以我们需要收集 点2 – 点1、点3 – 点1 ……这些时间以作分析；
2. 打点1我们一般是在html文件head标签的开头打个时间戳；
3. 在css文件加载前一般没有别的加载处理，所以打点1和打点2一般可以合并。

到此我们就收集到首屏相关各种数据，可以做各种针对性优化。Wait！在你大刀阔斧优化前，你要了解一些细节，它们有利于你做更准确的分析和更细致的优化。

细节1：js后面的点 – js前面的点 ≠ js的加载时间

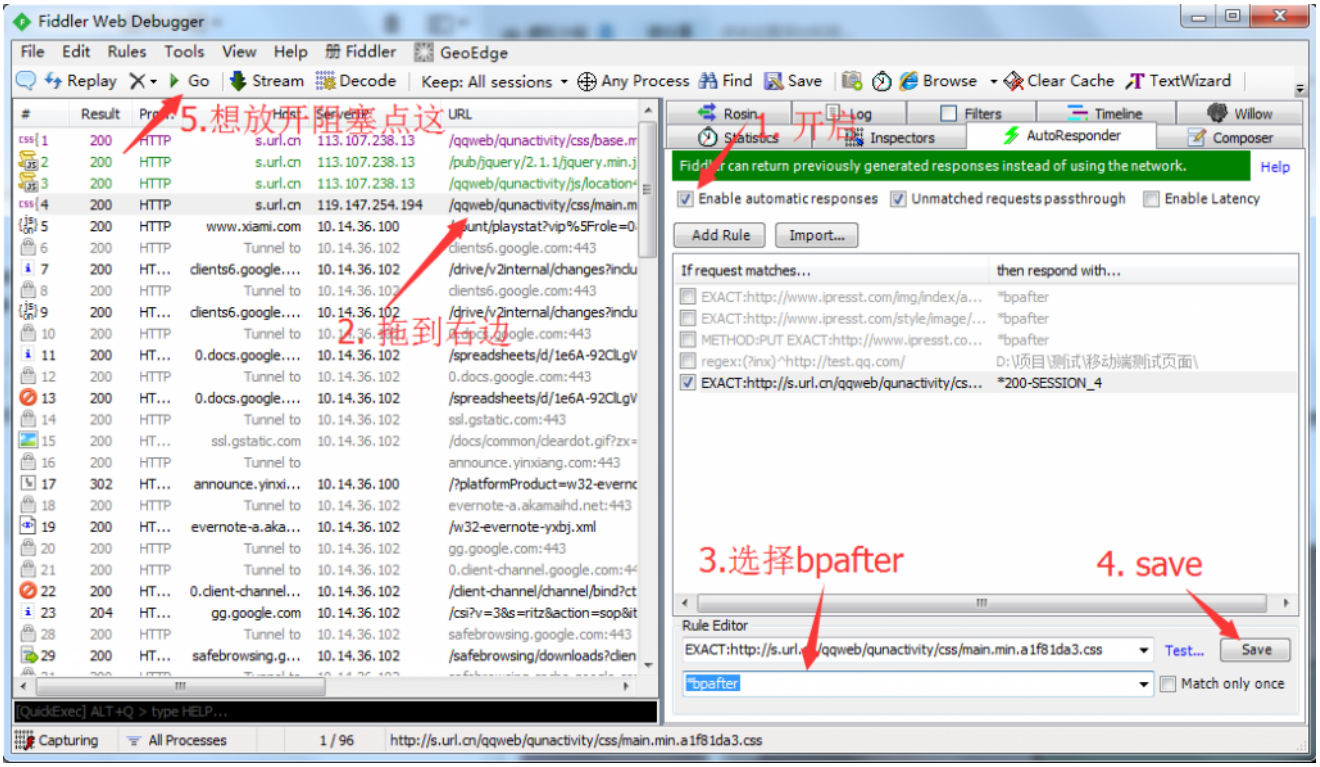
```html
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="utf-8" />

    <title>群活动</title>
    <script type="text/javascript">
        window.pageStartTime = Date.now();
    </script>
    <script type="text/javascript">var CssStartTime = + new Date();</script>
    <link rel="stylesheet" href="http://s.url.cn/qqweb/qunactivity/css/base.min.89d59c0a.css"/>
    <link rel="stylesheet" href="http://s.url.cn/qqweb/qunactivity/css/main.min.a1f81da3.css"/>
    <script type="text/javascript">var CssEndTime = + new Date();</script>
</head>
<body>

    <script type="text/javascript"> var JsStartTime = + new Date(); </script>
    <script src="http://s.url.cn/pub/jquery/2.1.1/jquery.min.js"></script>
    <script src="http://s.url.cn/qqweb/qunactivity/js/location4.min.7ec563e5.js"></script>
    <script type="text/javascript"> var JsEndTime = + new Date(); </script>
    <script type="text/javascript">
    // CSS文件加载时间上报
    speedReport('css', CssEndTime - CssStartTime);
    // JS文件加载时间上报
    speedReport('js', JsEndTime - JsStartTime);
    </script>
<!--Web-->
</body>
</html>
```

JsEndTime – JsStartTime = js文件的加载时间，对吗？
不对！明显地，这个等式忽略了js的执行时间。js执行代码是需要花费时间的，特别是做一些复杂的计算或频繁的dom操作，这个执行时间有时会达到几百毫秒。
那么，JsEndTime – JsStartTime = js文件的加载执行时间？
依然不对！因为CSS**文件的加载执行带来了干扰**。觉得很奇怪对吧，别急，我们来做个试验：我们找一个demo页面，在chrome里面打开，然后启动控制台，模拟低网速，让文件加载时间比较久：

先在正常情况下收集 JsEndTime – JsStartTime 的时间，然后使用fiddler阻塞某一条css请求几秒钟：



然后再恢复请求，拿到此时的 JsEndTime – JsStartTime 结果，会发现第一次的时间是几百毫秒将近1s，而第二次的时间低于100ms甚至接近为0（我的示例，时间视读者具体的js文件决定），两者的差距非常明显。

这是什么原理？这就是我们常说的"加载是并行的，执行是串行的"的结果。html开始加载的时候，浏览器会将页面外联的css文件和js文件并行加载，如果一个文件还没回来，它后面的代码是不会执行的。刚刚我们的demo，我们阻塞了css文件几秒，此时js文件因为并行已经加载回来，但由于css文件阻塞住，所以**后面 JsStartTime 的赋值语句是不执行的！**当我们放开阻塞，此时才会运行到 JsStartTime 的赋值、js文件的解析、JsEndTime的赋值，由于大头时间加载早已完成，所以 JsEndTime 和 JsStartTime 的差值非常小。

知道这个有何用？

1. 别再把 JsEndTime – JsStartTime 的结果成为js文件的加载执行时间（除非你没有外联css文件），不然会被内行人取笑滴；
2. css文件的阻塞会影响后面js代码的执行，自然也包括html代码的执行，即是说此时你的页面就是空白的。所以css文件尽量内联，你可以让构建工具帮你忙；
3. 如果真想要知道js文件的加载时间，最正确的姿势是使用 Resource Timing API，不过这个API移动端只能在Android4.4及以上的版本拿到数据，也就在业务PV大的场景才够我们做分析用

当然，那两个打点留着还是可以做分析用的。

细节2：html里面外联的js文件，前一个文件的加载会阻塞下一个文件的执行；而如果a.js负责渲染并会动态拉取js、拉取cgi并做渲染，会发现它后面的js文件再怎么阻塞也不会影响到它的处理

前半部分的结论在细节1里面已经证明，因为浏览器的执行是串行的。这说明，我们负责渲染内容的js代码要等到它前面所有的js文件加载执行完才会执行，即使那些代码跟渲染无关的代码如数据上报：

```
266  <script type="text/javascript"> var JsStartTime = + new Date(); </script>
267  <script src="lib/config.js"></script>
268  <script src="lib/reportkit.js"></script>
269  <script src="lib/utils.js"></script>
270  <script data-main="js/index" src="lib/require.js"></script>
271  <script type="text/javascript"> var JsEndTime = + new Date(); </script>
```

公共库

渲染相关

而后半部分的结论很好验证，我们在负责渲染的js文件后面外联一个别的js文件并把它阻塞住，你会发现渲染相关的js不管是动态拉取新的js文件、拉取渲染相关内容都一切正常，页面内容顺利渲染出来，它们的执行并不需要等被阻塞的这个文件。

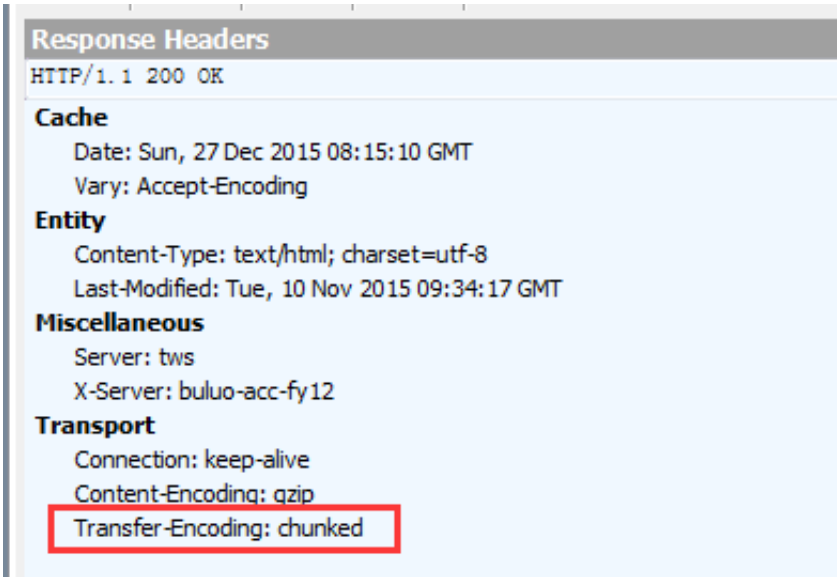
知道这个有何用？

1. **无关紧要"的js不要放在负责渲染的js前面，这里的"无关紧要"是指和首屏渲染无关**，如数据上报组件。我们可以选择将要上报的数据临时存起来，先继续执行渲染的js，等负责渲染的js执行完再加载上报组件再上报。甚至连zepto之类的库我们也可以放后面，把渲染相关的代码抽离出来并用原生js书写，放到最前面；
2. 可以看到，动态加载的js的执行是不会受到html后面外联的js的阻塞的影响，即是说，它的执行和后面js的执行顺序是不确定的。因此我们要小心处理好文件的依赖关系。当然还可以采用最不容易出错的方法：负责动态加载js的文件是html里面外联的最后一个文件

（注：个人觉得这是全文最重要的两点结论，因为我正在做首屏优化^-^）

细节3：如果html的返回头包含chunk，则它是边返回边解析的，不然就是一次性返回再解析。这个是在服务器配置的

打点1一般写在html里head标签的最前面，时常有朋友拿直出时的 点4 – 点1 的时间和非直出时 点8 – 点1 的时候做对比，来说明直出优化了多少多少毫秒，我倒觉得不一定。要知道直出的情况html文件包含渲染后的内容和dom节点，文件大小一般比非直出大，有时甚至大个几十K都有，那我觉得要说明直出优化了多少就要把html的加载时间考虑进去了。那上面的计算方法是否考虑上html的加载时间？那就要看html文件的返回头是否包含chunk：



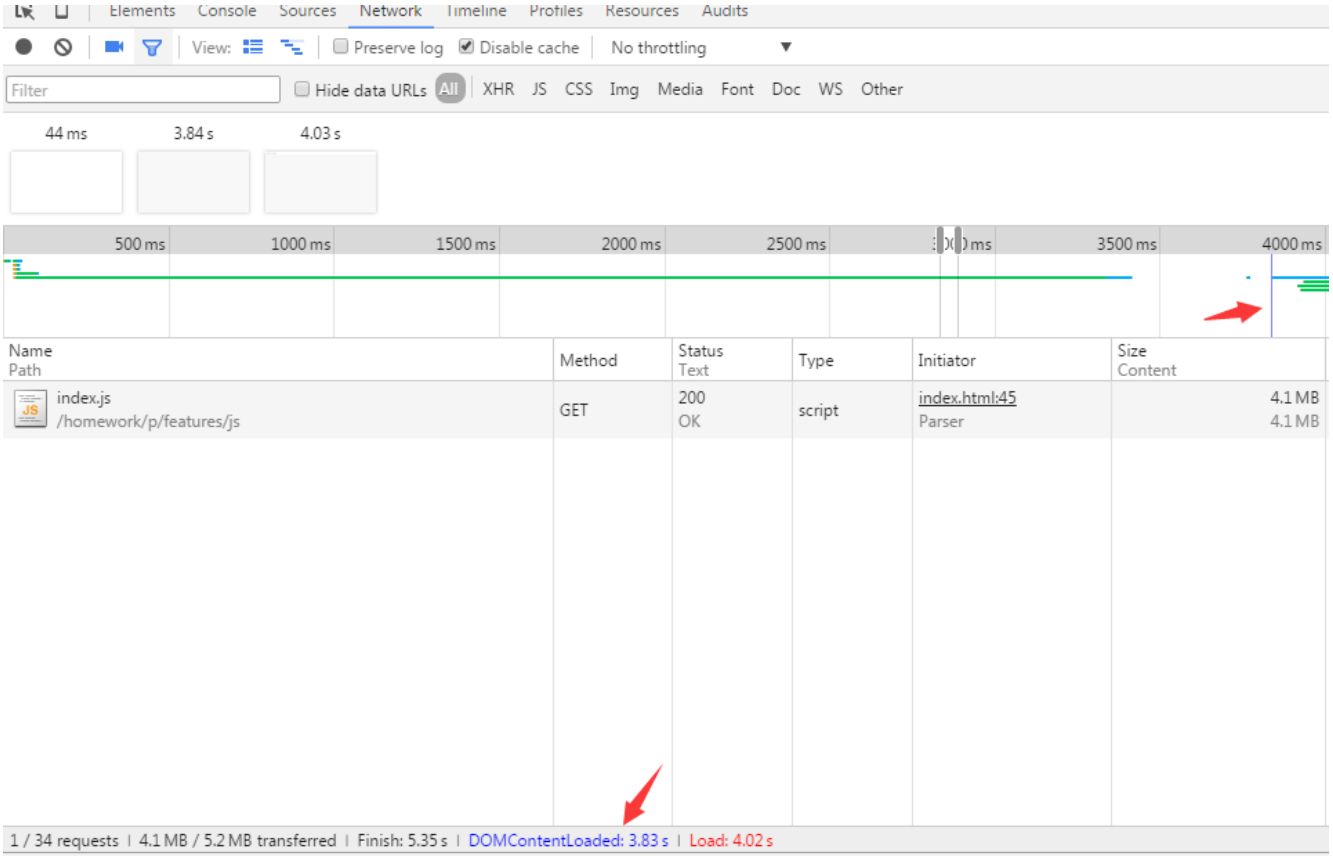
如果包含这个返回头，那html文件是边返回边解析的，此时上面的计算方法是合理的。如果不包含这个头，则html文件是整一个返回来后才开始解析，此时上面的计算方法就少算了html的加载时间，也就不够精准。这个返回头是由后台控制的。

知道这个有何用？

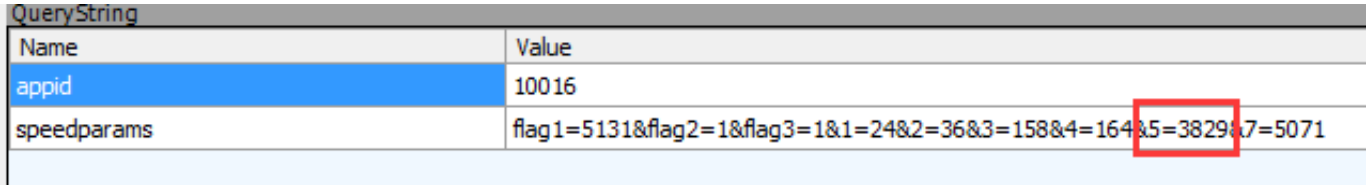
1. 如果我们想说明直出的优化程度，最好先瞧瞧你的html返回头。如果不包含chunk返回头，考虑拿HTML5 performance里面的 navigationStart 作为打点1（这个API也是Android4.4及以上才支持），要不就要评估文件大小变化做点修正了；
2. 对于没有启用chunk的html，建议不要inline太多跟渲染首屏内容无关的js在里面，这样会影响渲染时间

细节4：写在html里面的script节点的加载和解析会影响 domContentLoaded 事件的触发时间

我们有时会用 domContentLoaded 事件代替 onload 事件，在页面准备好的时候做一些处理。然而要知道，domContentLoaded里面的dom不止包含我们常说的普通dom节点，还包括script节点。
试验一下，我们将页面里面外联的一个js文件阻塞住一段时间再放开，我们看下chrome控制台：



很明显，js文件的加载时间会影响这个事件的触发事件。那js代码的解析时间会不会影响？我们在最后一个外联js文件后面打了一个点，它的时间是：



所以js文件加载执行会影响domContentLoaded事件的执行时机。

知道这个有何用？

1. 如果我们打算在domContentLoaded、onLoad 事件里面做一些特殊处理且这些处理比较重要（如跟渲染有关），那我们最好就不要在html里面直接外联一些跟渲染无关的js文件，可以考虑改用动态加载

总结

研究首屏时间和资源加载是一件挺有意思的事情，大家利用好chrome控制台（特别是里面的network标签）以及fiddler可以挖掘出很多有趣的小细节小结论。别以为这是在没事找事，理解好这些对大家做首屏性能优化、定位因为js文件执行顺序错乱导致报错等场景是非常有好处的。所以发现什么记得与我共享哈~