“浏览器中的页面循环系统”模块我们已经介绍完了，循环系统是页面的基础，理解了循环系统能让我们从本质上更好地理解页面的工作方式，加深我们对一些前端概念的理解。

接下来我们就要进入新的模块了，也就是“浏览器中的页面”模块，正如专栏简介中所言，页面是浏览器的核心，浏览器中的所有功能点都是服务于页面的，而Chrome开发者工具又是工程师调试页面的核心工具，所以在这个模块的开篇，我想先带你来深入了解下Chrome开发者工具。

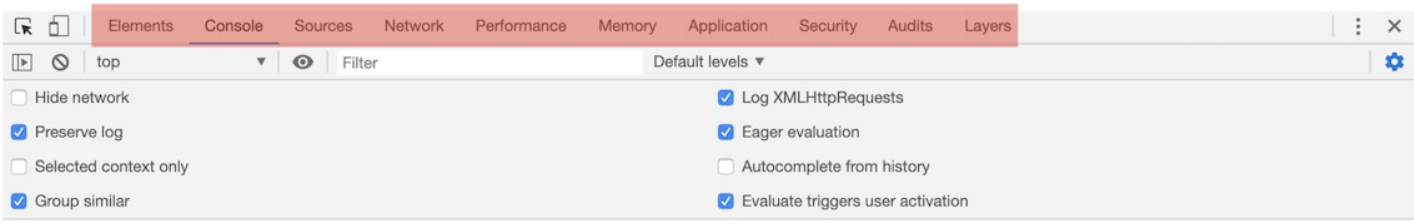
Chrome开发者工具（简称DevTools）是一组网页制作和调试的工具，内嵌于Google Chrome浏览器中。Chrome开发者工具非常重要，所蕴含的内容也是非常多的，熟练使用它能让你更加深入地了解浏览器内部工作原理。（Chrome开发者工具也在不停地迭代改进，如果你想使用最新版本，可以使用Chrome Canary。）

作为这一模块的第一篇文章，我们主要聚焦**页面的源头**和**网络数据的接收**，这些发送和接收的数据都能体现在开发者工具的网络面板上。不过为了你能更好地理解和掌握，我们会先对Chrome开发者工具做一个大致的介绍，然后再深入剖析网络面板。

Chrome开发者工具

Chrome开发者工具有很多重要的面板，比如与性能相关的有网络面板、Performance面板、内存面板等，与调试页面相关的有Elements面板、Sources面板、Console面板等。

你可以在浏览器窗口的右上方选择Chrome菜单，然后选择“更多工具–>开发者工具”来打开Chrome开发者工具。打开的页面如下图所示：



Chrome开发者工具

从图中可以看出，它一共包含了10个功能面板，包括了Elements、Console、Sources、NetWork、Performance、Memory、Application、Security、Audits和Layers。

关于这10个面板的大致功能，我做了一个表格，感兴趣的话，你可以详细看下：

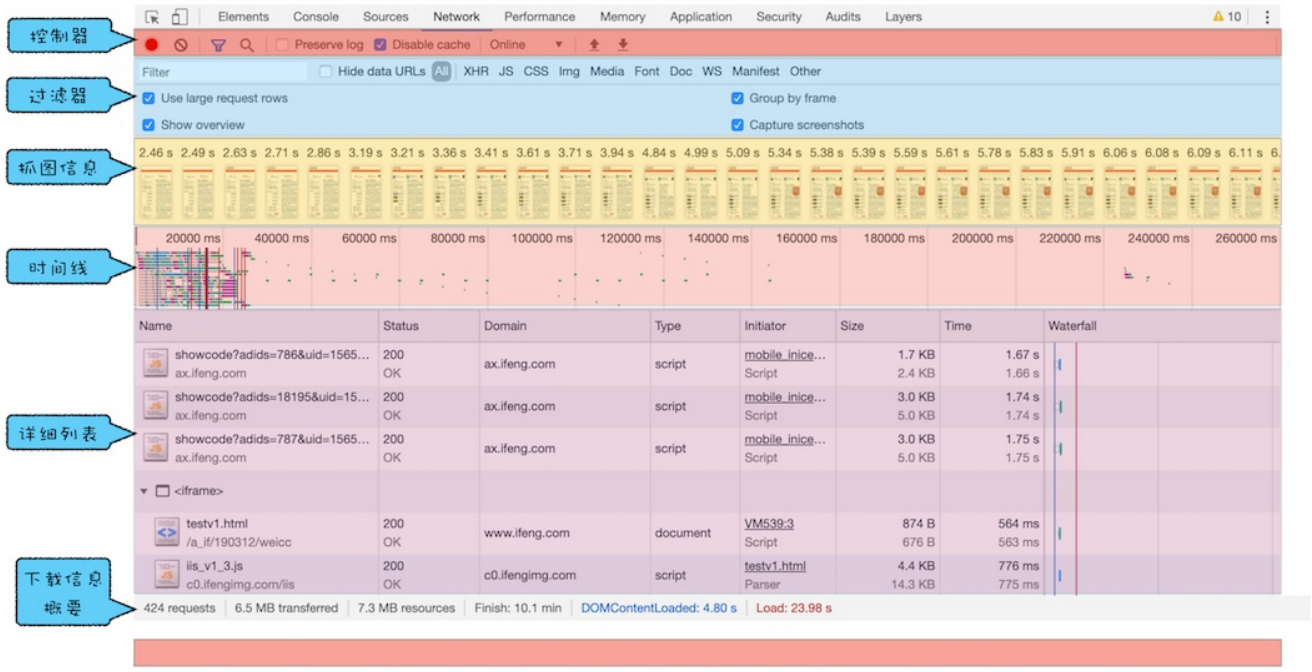
名称	描述
Elements面板	可以查看DOM结构、编辑CSS样式，用于测试页面布局和设计页面。
Console面板	可以看成是JavaScript Shell，能执行JavaScript脚本。通过Console还能和页面中的JavaScript对象交互。
Sources面板	1. 查看Web应用加载的所有文件； 2. 编辑CSS和JavaScript文件内容； 3. 将打乱的CSS文件或者JavaScript文件格式化； 4. 支持JavaScript的调试功能； 5. 设置工作区，将更改的文件保存到本地文件夹中。
NetWork（网络）面板	展示了页面中所有的请求内容列表，能查看每项请求的请求行、请求头、请求体、时间线以及网络请求瀑布图等信息。
Performance面板	记录和查看Web应用生命周期内的各种事件，并用来分析在执行过程中一些影响性能的要点。
Memory面板	用来查看运行过程中的JavaScript占用堆内存情况，追踪是否存在内存泄漏的情况等。
Application（应用）面板	查看Web应用的数据存储情况； PWA的基础数据：IndexedDB；Web SQL；本地和会话存储；Cookie；应用程序缓存；图像；字体和样式表等。
Security（安全）面板	显示当前页面一些基础的安全信息。
Audits面板	会对当前网页进行网络利用情况、网页性能方面的诊断，并给出一些优化建议。
Layers面板	展示一些渲染过程中分层的基础信息。

简单来说，Chrome开发者工具为我们提供了通过界面访问或者编辑DOM和CSSOM的能力，还提供了强大的调试功能和查看性能指标的能力。

OK，接下来我们就要重点看下其中重要的Network面板，即网络面板。

网络面板

网络面板由控制器、过滤器、抓图信息、时间线、详细列表和下载信息概要这6个区域构成（如下图所示）。



网络面板概要图

1. 控制器

其中，控制器有4个比较重要的功能，我们按照下文中的这张图来简单介绍下。

控制器概要图

- 红色圆点的按钮，表示“开始/暂停抓包”，这个功能很常见，很容易理解。
- “全局搜索”按钮，这个功能就非常重要了，可以在所有下载资源中搜索相关内容，还可以快速定位到某几个你想要的文件上。
- Disable cache，即“禁止从Cache中加载资源”的功能，它在调试Web应用的时候非常有用，因为开启了Cache会影响到网络性能测试的结果。
- Online按钮，是“模拟2G/3G”功能，它可以限制带宽，模拟弱网情况下页面的展现情况，然后你就可以根据实际展示情况来动态调整策略，以便让Web应用更加适用于这些弱网。

2. 过滤器

网络面板中的过滤器，主要就是起过滤功能。因为有时候一个页面有太多内容在详细列表区域中展示了，而你可能只想查看JavaScript文件或者CSS文件，这时候就可以通过过滤器模块来筛选你想要的文件类型。

3. 抓图信息

抓图信息区域，可以用来分析用户等待页面加载时间内所看到的内容，分析用户实际的体验情况。比如，如果页面加载1秒多之后屏幕截图还是白屏状态，这时候就需要分析是网络还是代码的问题了。（勾选面板上的“Capture screenshots”即可启用屏幕截图。）

4. 时间线

时间线，主要用来展示HTTP、HTTPS、WebSocket加载的状态和时间的一个关系，用于直观感受页面的加载过程。如果是多条竖线堆叠在一起，那说明这些资源被同时被加载。至于具体到每个文件的加载信息，还需要用到下面要讲的详细列表。

5. 详细列表

这个区域是最重要的，它详细记录了每个资源从发起请求到完成请求这中间所有过程的状态，以及最终请求完成的数据信息。通过该列表，你就能很容易地去诊断一些网络问题。

详细列表是我们本篇文章介绍的重点，不过内容比较多，所以放到最后去专门介绍了。

6. 下载信息概要

下载信息概要中，你要重点关注下DOMContentLoaded和Load两个事件，以及这两个事件的完成时间。

- DOMContentLoaded，这个事件发生后，说明页面已经构建好DOM了，这意味着构建DOM所需要的HTML文件、JavaScript文件、CSS文件都已经下载完成了。
- Load，说明浏览器已经加载了所有的资源（图像、样式表等）。

通过下载信息概要面板，你可以查看触发这两个事件所花费的时间。

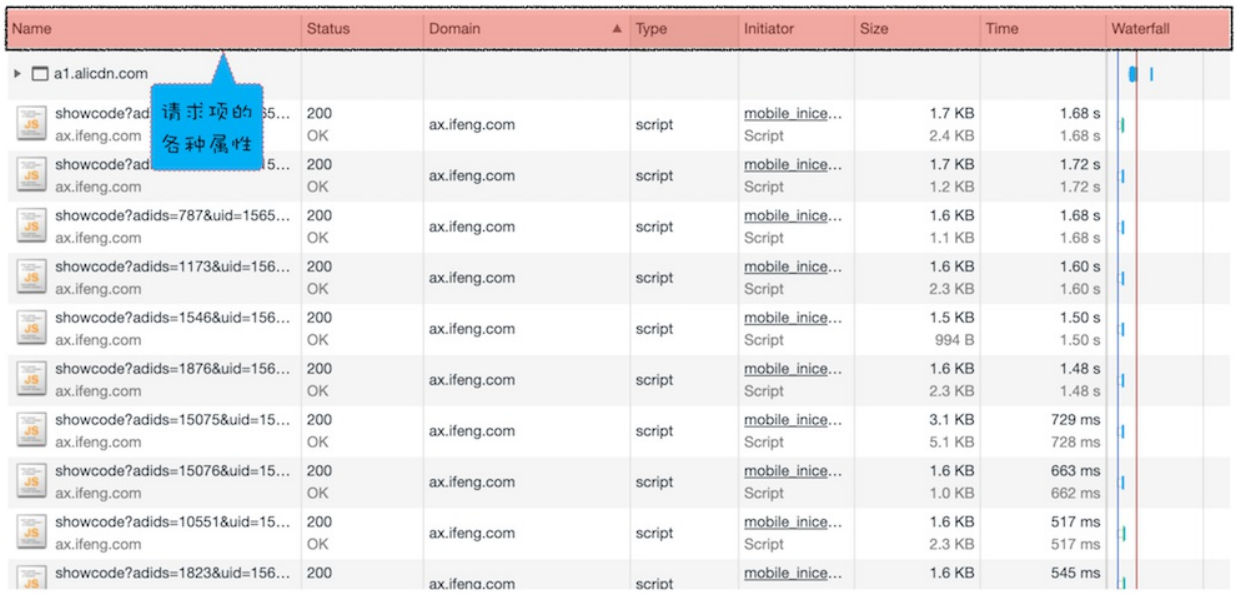
网络面板中的详细列表

下面我们就来重点介绍网络面板中的详细列表，这里面包含了大量有用的信息。

1. 列表的属性

列表的属性比较多，比如Name、Status、Type、Initiator等等，这个不难理解。当然，你还可以通过点击右键的下拉菜单来添加其他属性，这里我就不再赘述了，你可以自己上手实操一下。

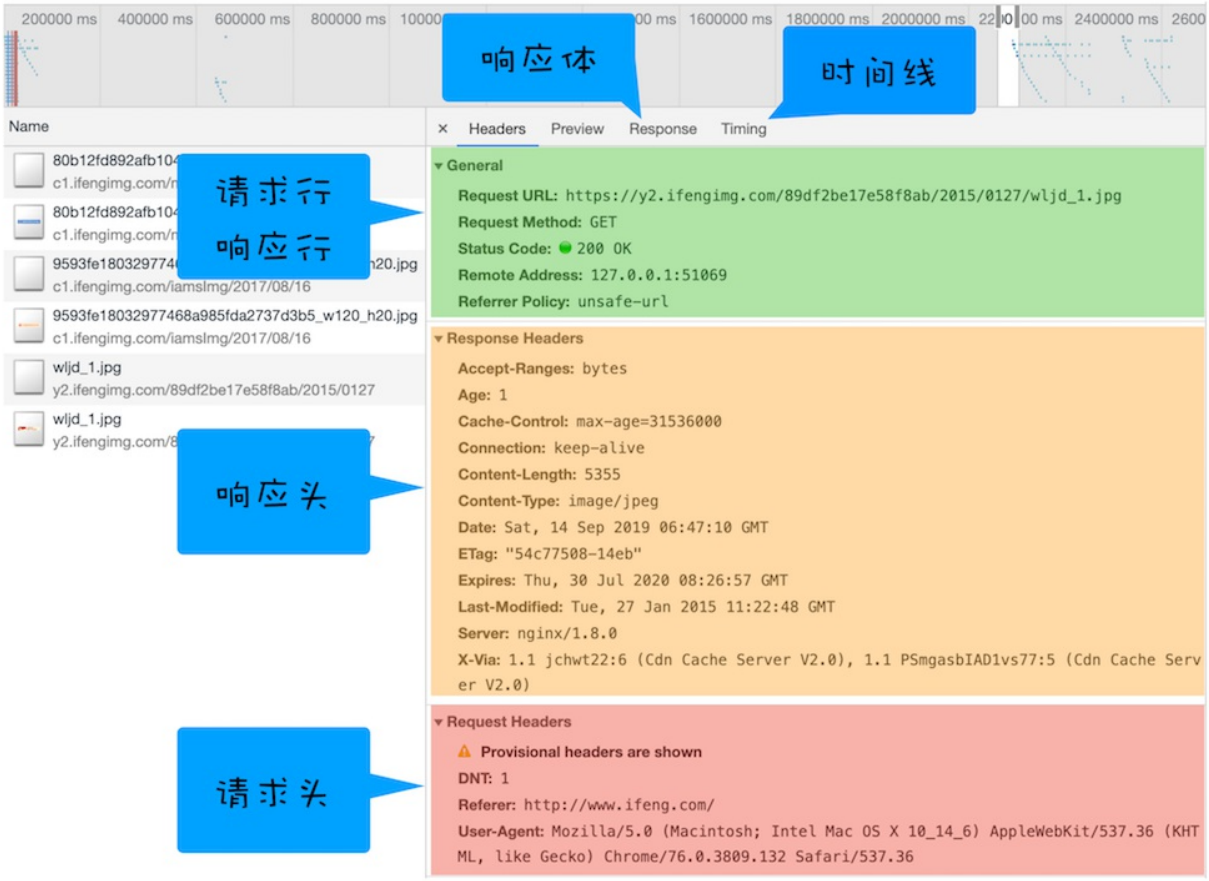
另外，你也可以按照列表的属性来给列表排序，默认情况下，列表是按请求发起的时间来排序的，最早发起请求的资源在顶部。当然也可以按照返回状态码、请求类型、请求时长、内容大小等基础属性排序，只需点击相应属性即可。



根据属性排序

2. 详细信息

如果你选中详细列表中的一项，右边就会出现该项的详细信息，如下所示：

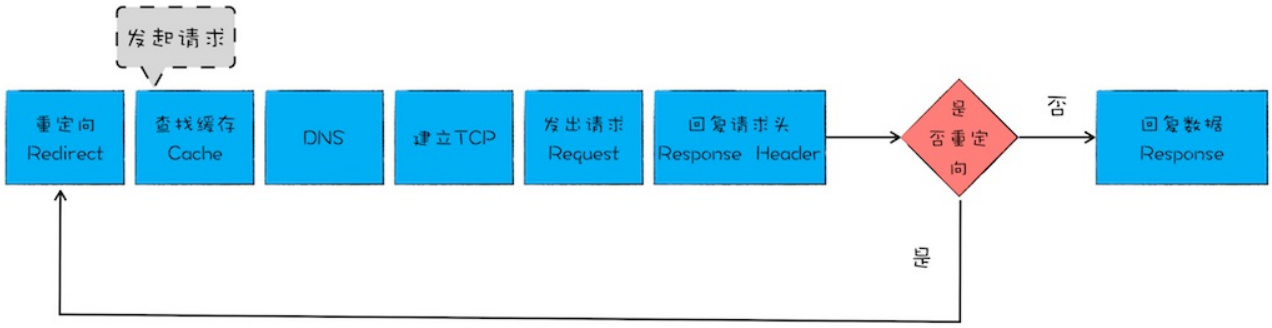


详细请求信息

你可以在此查看请求列表中任意一项的请求行和请求头信息，还可以查看响应行、响应头和响应体。然后你可以根据这些查看的信息来判断你的业务逻辑是否正确，或者有时候也可以用来逆向推导别人网站的业务逻辑。

3. 单个资源的时间线

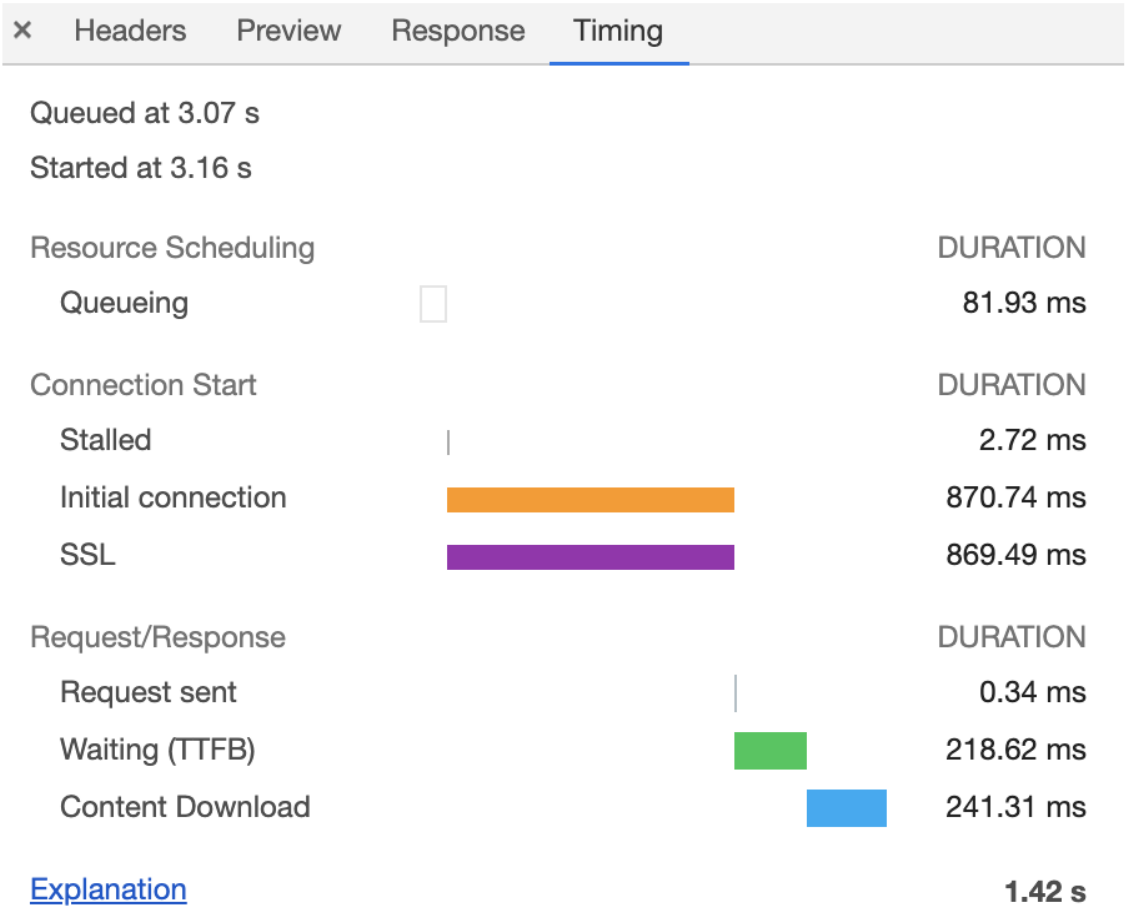
了解了每个资源的详细请求信息之后，我们再来分析单个资源请求时间线，这就涉及具体的HTTP请求流程了。



浏览器中HTTP请求流程

我们再回顾下在《03 | HTTP请求流程：为什么很多站点第二次打开速度会很快？》这篇文章，我们介绍过发起一个HTTP请求之后，浏览器首先查找缓存，如果缓存没有命中，那么继续发起DNS请求获取IP地址，然后利用IP地址和服务器端建立TCP连接，再发送HTTP请求，等待服务器响应；不过，如果服务器响应头中包含了重定向的信息，那么整个流程就需要重新再走一遍。这就是在浏览器中一个HTTP请求的基础流程。

那详细列表中是如何表示出这个流程的呢？这就要重点看下时间线面板了：



单个文件的时间线

那面板中这各项到底是什么含义呢？

第一个是Queuing，也就是排队的意思，当浏览器发起一个请求的时候，会有很多原因导致该请求不能被立即执行，而是需要排队等待。导致请求处于排队状态的原因有很多。

- 首先，页面中的资源是有优先级的，比如CSS、HTML、JavaScript等都是页面中的核心文件，所以优先级最高；而图片、视频、音频这类资源就不是核心资源，优先级就比较低。通常当后者遇到前者时，就需要“让路”，进入待排队状态。
- 其次，我们前面也提到过，浏览器会为每个域名最多维护6个TCP连接，如果发起一个HTTP请求时，这6个TCP连接都处于忙碌状态，那么这个请求就会处于排队状态。
- 最后，网络进程在为数据分配磁盘空间时，新的HTTP请求也需要短暂地等待磁盘分配结束。

等待排队完成之后，就要进入发起连接的状态了。不过在发起连接之前，还有一些原因可能导致连接过程被推迟，这个推迟就表现在面板中的**Stalled**上，它表示停滞的意思。

这里需要额外说明的是，如果你使用了代理服务器，还会增加一个**Proxy Negotiation**阶段，也就是代理协商阶段，它表示代理服务器连接协商所用的时间，不过在上图中没有体现出来，因为这里我们没有使用代理服务器。

接下来，就到了**Initial connection/SSL阶段**了，也就是和服务器建立连接的阶段，这包括了建立TCP连接所花费的时间；不过如果你使用了HTTPS协议，那么还需要一个额外的SSL握手时间，这个过程主要是用来协商一些加密信息的。（关于SSL协商的详细过程，我们会在Web安全模块中介绍。）

和服务器建立好连接之后，网络进程会准备请求数据，并将其发送给网络，这就是**Request sent阶段**。通常这个阶段非常快，因为只需要把浏览器缓冲区的数据发送出去就结束了，并不需要判断服务器是否接收到了，所以这个时间通常不到1毫秒。

数据发送出去了，接下来就是等待接收服务器第一个字节的数据，这个阶段称为Waiting (TTFB)，通常也称为“**第一字节时间**”。 TTFB是反映服务端响应速度的重要指标，对服务器来说，TTFB 时间越短，就说明服务器响应越快。

接收到第一个字节之后，进入陆续接收完整数据的阶段，也就是**Content Download阶段**，这意味着从第一字节时间到接收到全部响应数据所用的时间。

优化时间线上耗时项

了解了时间线面板上的各项含义之后，我们就可以根据这个请求的时间线来实现相关的优化操作了。

1. 排队（Queuing）时间过久

排队时间过久，大概率是由浏览器为每个域名最多维护6个连接导致的。那么基于这个原因，你就可以让1个站点下面的资源放在多个域名下面，比如放到3个域名下面，这样就可以同时支持18个连接了，这种方案称为**域名分片**技术。除了域名分片技术外，我个人还建议你**把站点升级到HTTP2**，因为HTTP2已经没有每个域名最多维护6个TCP连接的限制了。

2. 第一字节时间（TTFB）时间过久

这可能的原因有如下：

- **服务器生成页面数据的时间过久**。对于动态网页来说，服务器收到用户打开一个页面的请求时，首先要从数据库中读取该页面需要的数据，然后把这些数据传入到模板中，模板渲染后，再返回给用户。服务器在处理这个数据的过程中，可能某个环节会出问题。
- **网络的原因**。比如使用了低带宽的服务器，或者本来用的是电信的服务器，可联通的网络用户要来访问你的服务器，这样也会拖慢网速。
- **发送请求头时带上了多余的用户信息**。比如一些不必要的Cookie信息，服务器接收到这些Cookie信息之后可能需要对每一项都做处理，这样就加大了服务器的处理时长。

对于这三种问题，你要有针对性地出一些解决方案。面对第一种服务器的问题，你可以想办法去提高服务器的处理速度，比如通过增加各种缓存的技术；针对第二种网络问题，你可以使用CDN来缓存一些静态文件；至于第三种，你在发送请求时就去尽可能地减少一些不必要的Cookie数据信息。

3. Content Download时间过久

如果单个请求的Content Download花费了大量时间，有可能是字节数太多的原因导致的。这时候你就需要减少文件大小，比如压缩、去掉源码中不必要的注释等方法。

总结

好了，今天就介绍到这里了，下面我来总结下今天的内容。

首先我们简单介绍了Chrome开发者工具10个基础的面板信息；然后重点剖析了网络面板，再结合之前介绍的网络请求流程来重点分析了网络面板中时间线的各个指标的含义；最后我们还简要分析了时间线中各项指标出现异常的可能原因，并给出了一些优化方案。

其实通过今天的分析，我们可以得出这样一个结论：如果你要去做一些实践性的项目优化，理解其背后的理论至关重要。因为理论就是一条“线”，它会把各种实践的内容“串”在一起，然后你可以围绕着这条“线”来排查问题。

思考时间

今天我们介绍了网络面板，还有一个非常重要的Performance面板我们没有介绍，不过你可以去网上查找一些相关的资料。

所以今天留给你的是一道实际操作的题目，你可以结合网络面板和Performance面板来分析一个Web应用的性能瓶颈（比如https://www.12306.cn ）。

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

“浏览器中的页面循环系统”模块我们已经介绍完了，循环系统是页面的基础，理解了循环系统能让我们从本质上更好地理解页面的工作方式，加深我们对一些前端概念的理解。

接下来我们就要进入新的模块了，也就是“浏览器中的页面”模块，正如专栏简介中所言，页面是浏览器的核心，浏览器中的所有功能点都是服务于页面的，而Chrome开发者工具又是工程师调试页面的核心工具，所以在这个模块的开篇，我想先带你来深入了解下Chrome开发者工具。

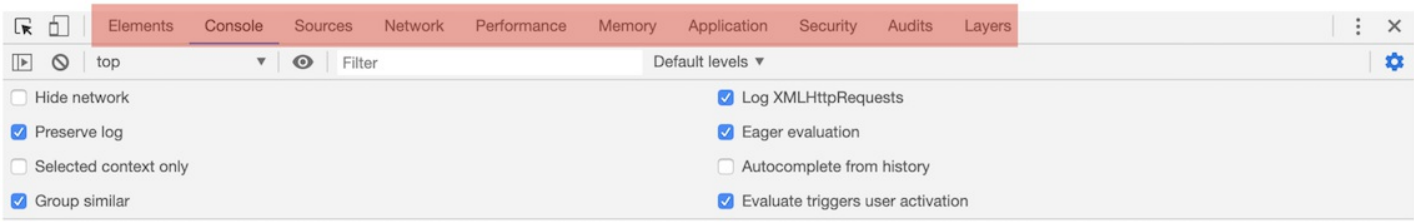
Chrome开发者工具（简称DevTools）是一组网页制作和调试的工具，内嵌于Google Chrome浏览器中。Chrome开发者工具非常重要，所蕴含的内容也是非常多的，熟练使用它能让你更加深入地了解浏览器内部工作原理。（Chrome开发者工具也在不停地迭代改进，如果你想使用最新版本，可以使用Chrome Canary。）

作为这一模块的第一篇文章，我们主要聚焦**页面的源头**和**网络数据的接收**，这些发送和接收的数据都能体现在开发者工具的网络面板上。不过为了你能更好地理解和掌握，我们会先对Chrome开发者工具做一个大致的介绍，然后再深入剖析网络面板。

Chrome开发者工具

Chrome开发者工具有很多重要的面板，比如与性能相关的有网络面板、Performance面板、内存面板等，与调试页面相关的有Elements面板、Sources面板、Console面板等。

你可以在浏览器窗口的右上方选择Chrome菜单，然后选择“更多工具–>开发者工具”来打开Chrome开发者工具。打开的页面如下图所示：



Chrome开发者工具

从图中可以看出，它一共包含了10个功能面板，包括了Elements、Console、Sources、NetWork、Performance、Memory、Application、Security、Audits和Layers。

关于这10个面板的大致功能，我做了一个表格，感兴趣的话，你可以详细看下：

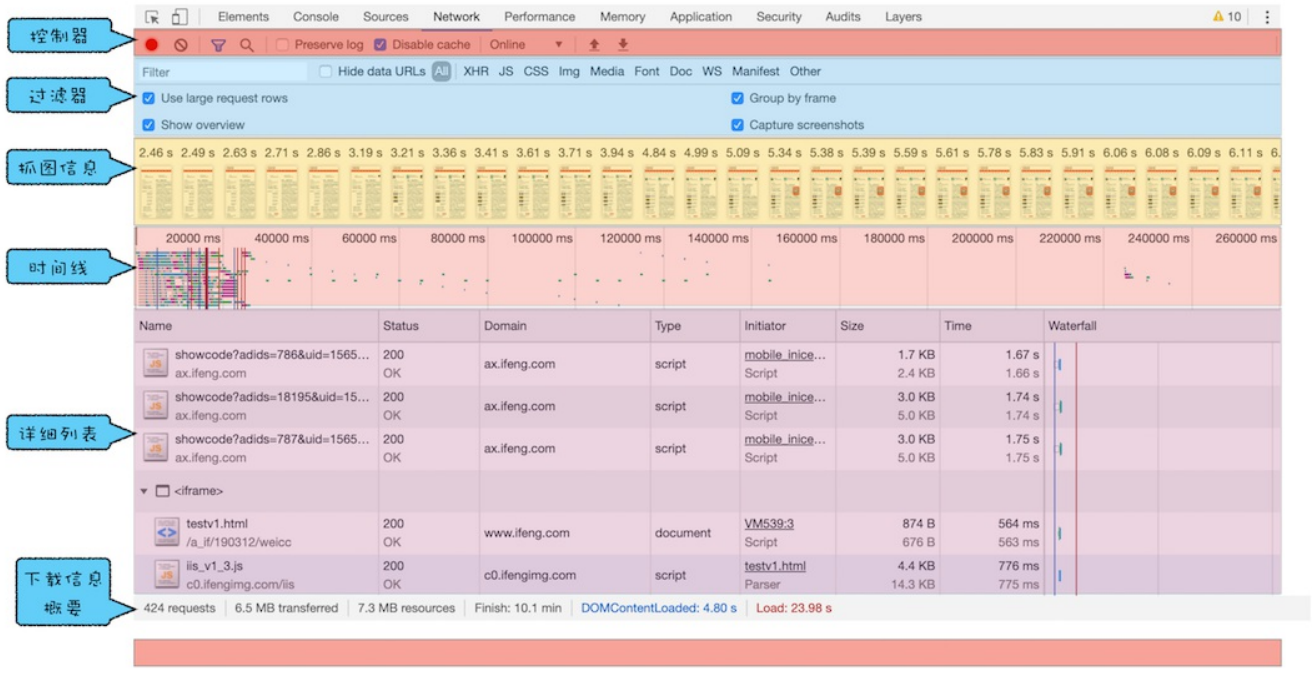
名称	描述
Elements面板	可以查看DOM结构、编辑CSS样式，用于测试页面布局和设计页面。
Console面板	可以看成是JavaScript Shell，能执行JavaScript脚本。通过Console还能和页面中的JavaScript对象交互。
Sources面板	1. 查看Web应用加载的所有文件； 2. 编辑CSS和JavaScript文件内容； 3. 将打乱的CSS文件或者JavaScript文件格式化； 4. 支持JavaScript的调试功能； 5. 设置工作区，将更改的文件保存到本地文件夹中。
NetWork（网络）面板	展示了页面中所有的请求内容列表，能查看每项请求的请求行、请求头、请求体、时间线以及网络请求瀑布图等信息。
Performance面板	记录和查看Web应用生命周期内的各种事件，并用来分析在执行过程中一些影响性能的要点。
Memory面板	用来查看运行过程中的JavaScript占用堆内存情况，追踪是否存在内存泄漏的情况等。
Application（应用）面板	查看Web应用的数据存储情况； PWA的基础数据：IndexedDB；Web SQL；本地和会话存储；Cookie；应用程序缓存；图像；字体和样式表等。
Security（安全）面板	显示当前页面一些基础的安全信息。
Audits面板	会对当前网页进行网络利用情况、网页性能方面的诊断，并给出一些优化建议。
Layers面板	展示一些渲染过程中分层的基础信息。

简单来说，Chrome开发者工具为我们提供了通过界面访问或者编辑DOM和CSSOM的能力，还提供了强大的调试功能和查看性能指标的能力。

OK，接下来我们就要重点看下其中重要的Network面板，即网络面板。

网络面板

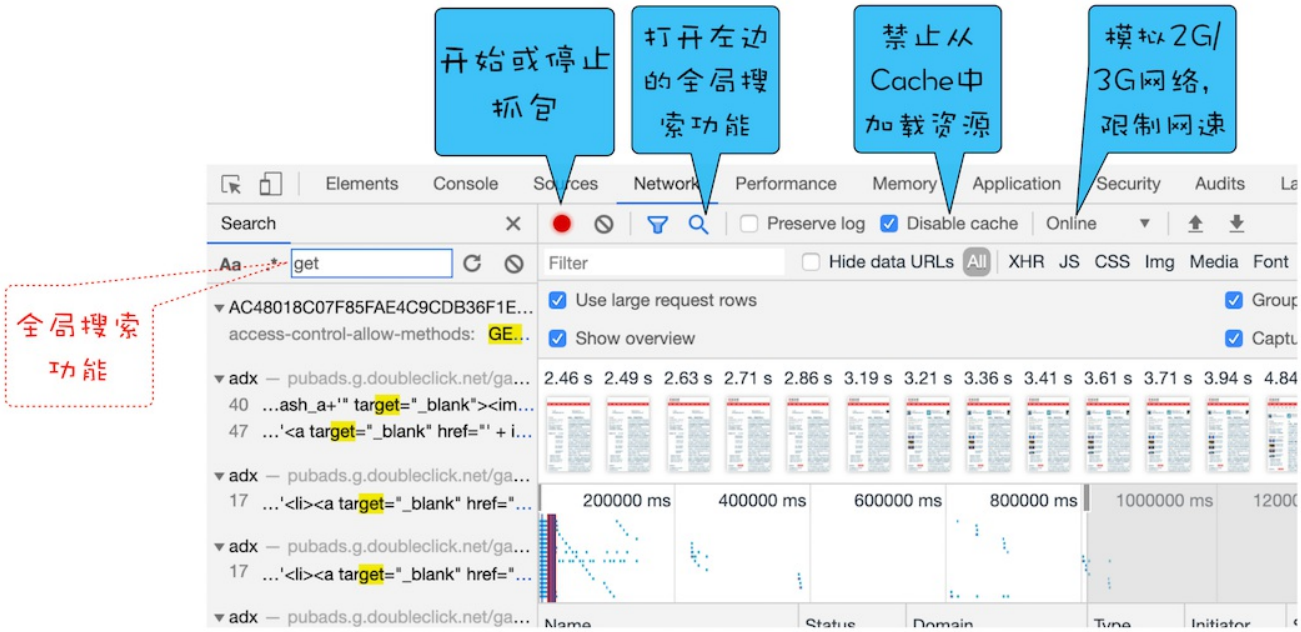
网络面板由控制器、过滤器、抓图信息、时间线、详细列表和下载信息概要这6个区域构成（如下图所示）。



网络面板概要图

1. 控制器

其中，控制器有4个比较重要的功能，我们按照下文中的这张图来简单介绍下。



控制器概要图

- 红色圆点的按钮，表示“开始/暂停抓包”，这个功能很常见，很容易理解。
- “全局搜索”按钮，这个功能就非常重要了，可以在所有下载资源中搜索相关内容，还可以快速定位到某几个你想要的文件上。
- Disable cache，即“禁止从Cache中加载资源”的功能，它在调试Web应用的时候非常有用，因为开启了Cache会影响到网络性能测试的结果。
- Online按钮，是“模拟2G/3G”功能，它可以限制带宽，模拟弱网情况下页面的展现情况，然后你就可以根据实际展示情况来动态调整策略，以便让Web应用更加适用于这些弱网。

2. 过滤器

网络面板中的过滤器，主要就是起过滤功能。因为有时候一个页面有太多内容在详细列表区域中展示了，而你可能只想查看JavaScript文件或者CSS文件，这时候就可以通过过滤器模块来筛选你想要的文件类型。

3. 抓图信息

抓图信息区域，可以用来分析用户等待页面加载时间内所看到的内容，分析用户实际的体验情况。比如，如果页面加载1秒多之后屏幕截图还是白屏状态，这时候就需要分析是网络还是代码的问题了。（勾选面板上的“Capture screenshots”即可启用屏幕截图。）

4. 时间线

时间线，主要用来展示HTTP、HTTPS、WebSocket加载的状态和时间的一个关系，用于直观感受页面的加载过程。如果是多条竖线堆叠在一起，那说明这些资源被同时被加载。至于具体到每个文件的加载信息，还需要用到下面要讲的详细列表。

5. 详细列表

这个区域是最重要的，它详细记录了每个资源从发起请求到完成请求这中间所有过程的状态，以及最终请求完成的数据信息。通过该列表，你就能很容易地去诊断一些网络问题。

详细列表是我们本篇文章介绍的重点，不过内容比较多，所以放到最后去专门介绍了。

6. 下载信息概要

下载信息概要中，你要重点关注下DOMContentLoaded和Load两个事件，以及这两个事件的完成时间。

- DOMContentLoaded，这个事件发生后，说明页面已经构建好DOM了，这意味着构建DOM所需要的HTML文件、JavaScript文件、CSS文件都已经下载完成了。
- Load，说明浏览器已经加载了所有的资源（图像、样式表等）。

通过下载信息概要面板，你可以查看触发这两个事件所花费的时间。

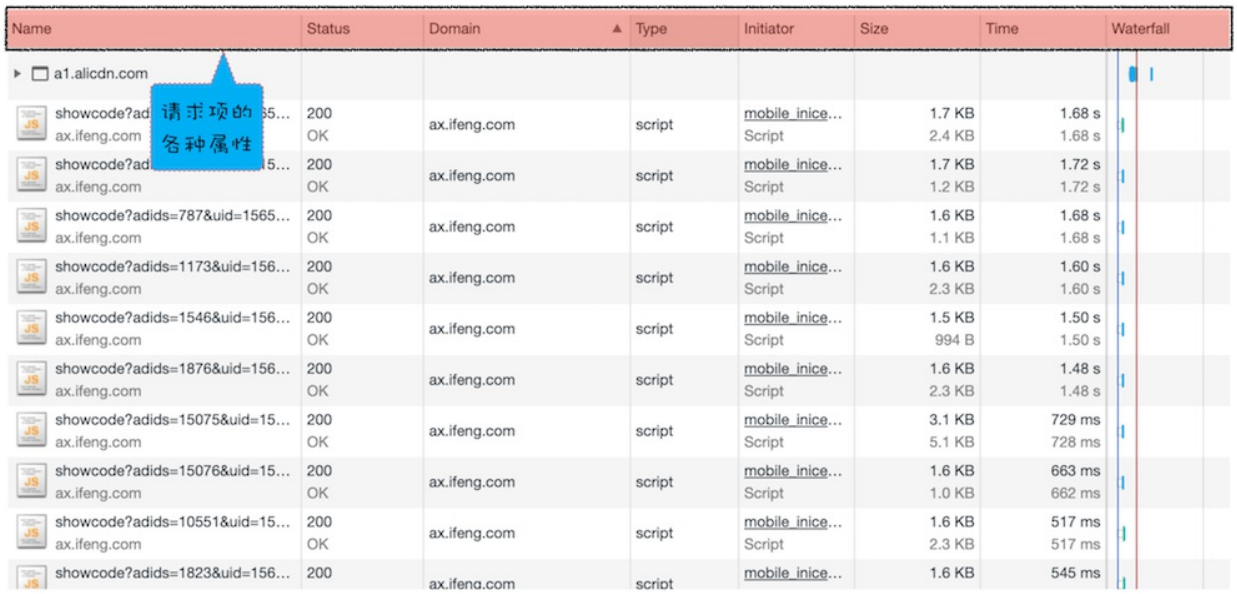
网络面板中的详细列表

下面我们就来重点介绍网络面板中的详细列表，这里面包含了大量有用的信息。

1. 列表的属性

列表的属性比较多，比如Name、Status、Type、Initiator等等，这个不难理解。当然，你还可以通过点击右键的下拉菜单来添加其他属性，这里我就不再赘述了，你可以自己上手实操一下。

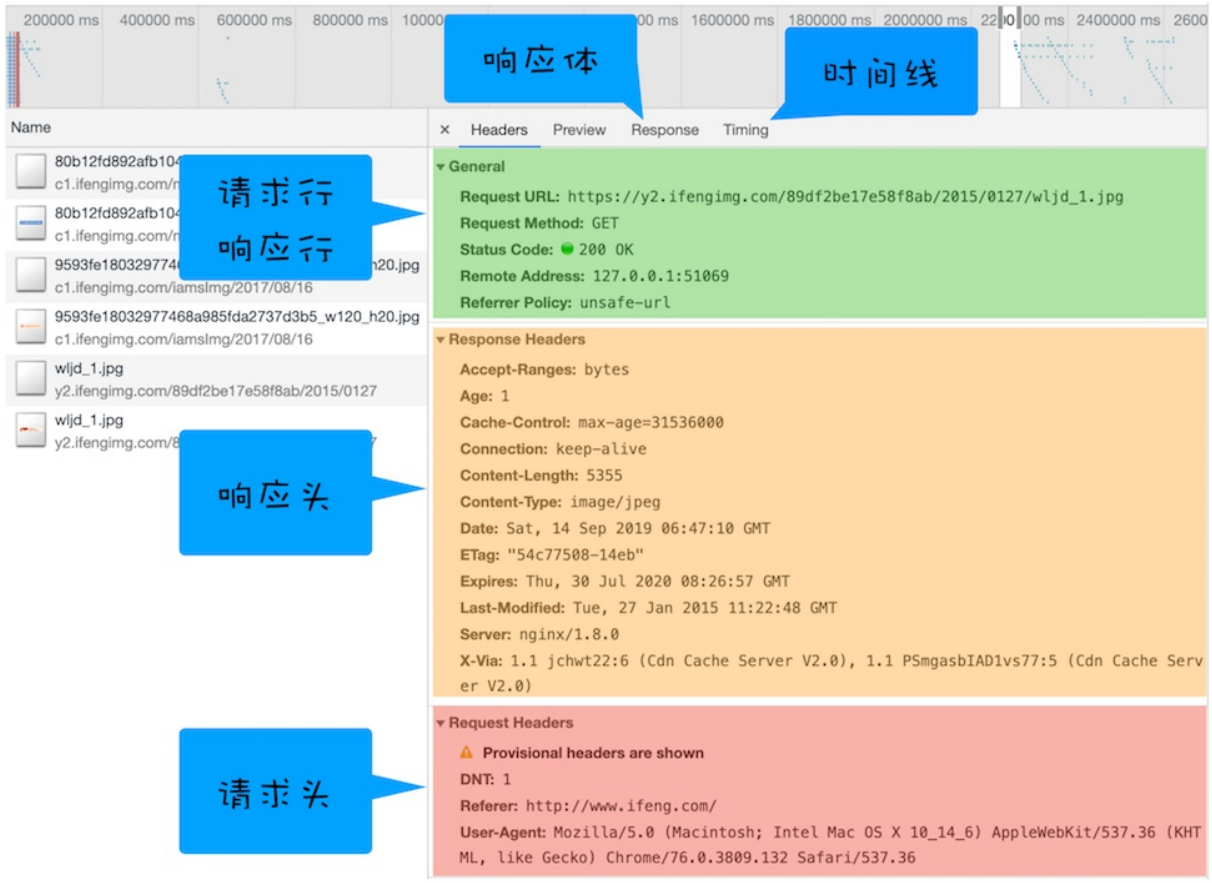
另外，你也可以按照列表的属性来给列表排序，默认情况下，列表是按请求发起的时间来排序的，最早发起请求的资源在顶部。当然也可以按照返回状态码、请求类型、请求时长、内容大小等基础属性排序，只需点击相应属性即可。



根据属性排序

2. 详细信息

如果你选中详细列表中的一项，右边就会出现该项的详细信息，如下所示：

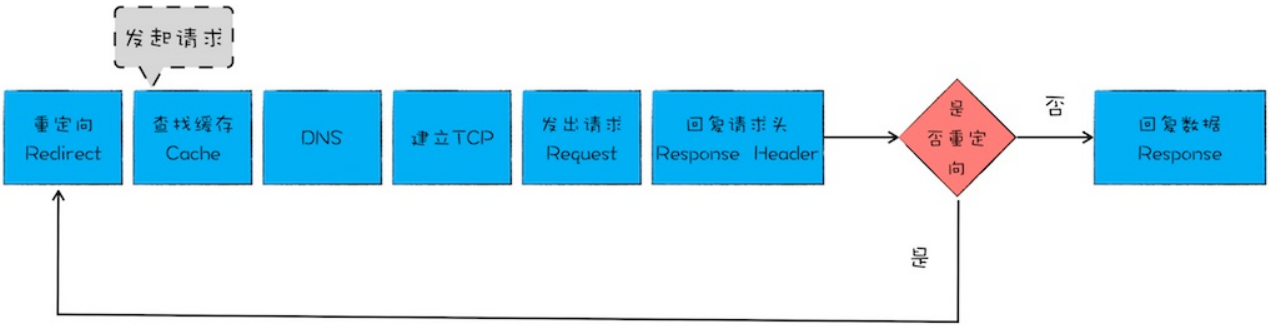


详细请求信息

你可以在此查看请求列表中任意一项的请求行和请求头信息，还可以查看响应行、响应头和响应体。然后你可以根据这些查看的信息来判断你的业务逻辑是否正确，或者有时候也可以用来逆向推导别人网站的业务逻辑。

3. 单个资源的时间线

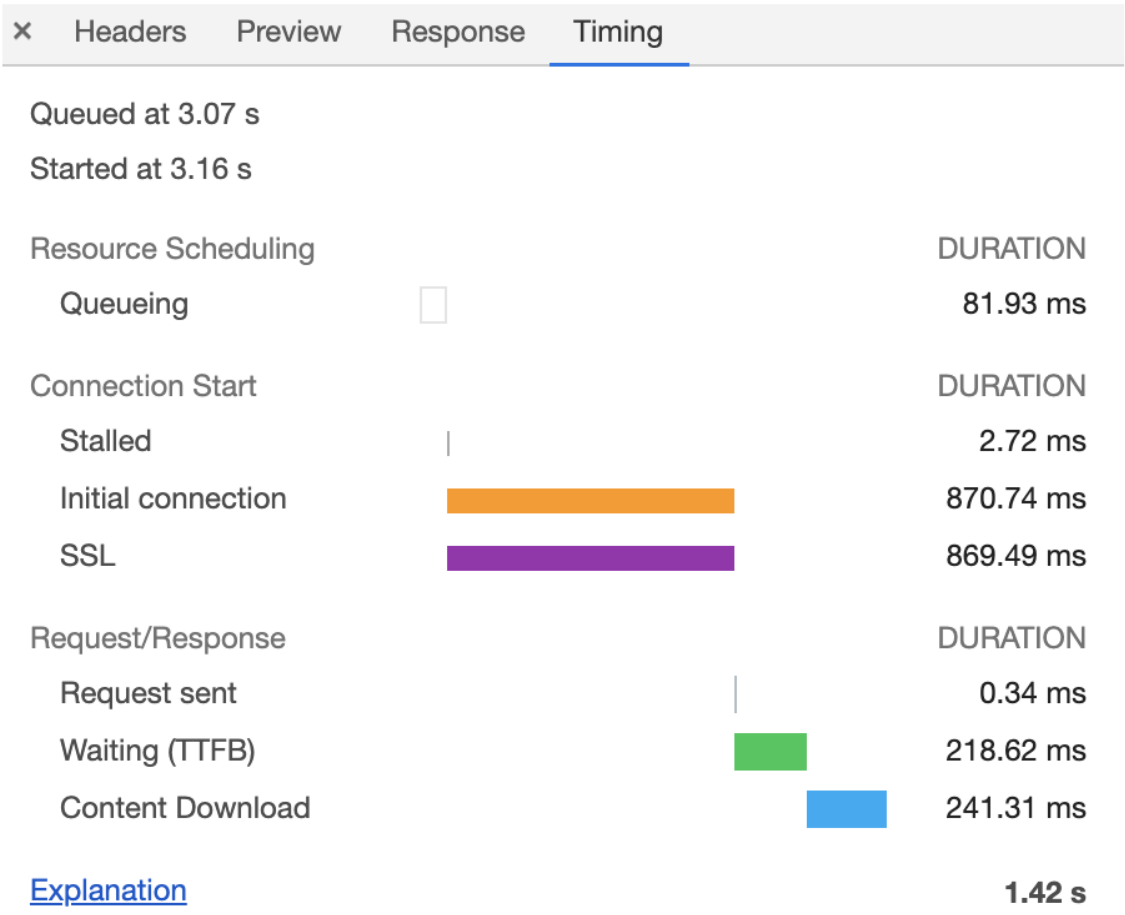
了解了每个资源的详细请求信息之后，我们再来分析单个资源请求时间线，这就涉及具体的HTTP请求流程了。



浏览器中HTTP请求流程

我们再回顾下在《03 | HTTP请求流程：为什么很多站点第二次打开速度会很快？》这篇文章，我们介绍过发起一个HTTP请求之后，浏览器首先查找缓存，如果缓存没有命中，那么继续发起DNS请求获取IP地址，然后利用IP地址和服务器端建立TCP连接，再发送HTTP请求，等待服务器响应；不过，如果服务器响应头中包含了重定向的信息，那么整个流程就需要重新再走一遍。这就是在浏览器中一个HTTP请求的基础流程。

那详细列表中是如何表示出这个流程的呢？这就要重点看下时间线面板了：



单个文件的时间线

那面板中这各项到底是什么含义呢？

第一个是Queuing，也就是排队的意思，当浏览器发起一个请求的时候，会有很多原因导致该请求不能被立即执行，而是需要排队等待。导致请求处于排队状态的原因有很多。

- 首先，页面中的资源是有优先级的，比如CSS、HTML、JavaScript等都是页面中的核心文件，所以优先级最高；而图片、视频、音频这类资源就不是核心资源，优先级就比较低。通常当后者遇到前者时，就需要“让路”，进入待排队状态。
- 其次，我们前面也提到过，浏览器会为每个域名最多维护6个TCP连接，如果发起一个HTTP请求时，这6个TCP连接都处于忙碌状态，那么这个请求就会处于排队状态。
- 最后，网络进程在为数据分配磁盘空间时，新的HTTP请求也需要短暂地等待磁盘分配结束。

等待排队完成之后，就要进入发起连接的状态了。不过在发起连接之前，还有一些原因可能导致连接过程被推迟，这个推迟就表现在面板中的**Stalled**上，它表示停滞的意思。

这里需要额外说明的是，如果你使用了代理服务器，还会增加一个**Proxy Negotiation**阶段，也就是代理协商阶段，它表示代理服务器连接协商所用的时间，不过在上图中没有体现出来，因为这里我们没有使用代理服务器。

接下来，就到了**Initial connection/SSL阶段**了，也就是和服务器建立连接的阶段，这包括了建立TCP连接所花费的时间；不过如果你使用了HTTPS协议，那么还需要一个额外的SSL握手时间，这个过程主要是用来协商一些加密信息的。（关于SSL协商的详细过程，我们会在Web安全模块中介绍。）

和服务器建立好连接之后，网络进程会准备请求数据，并将其发送给网络，这就是**Request sent阶段**。通常这个阶段非常快，因为只需要把浏览器缓冲区的数据发送出去就结束了，并不需要判断服务器是否接收到了，所以这个时间通常不到1毫秒。

数据发送出去了，接下来就是等待接收服务器第一个字节的数据，这个阶段称为Waiting (TTFB)，通常也称为“**第一字节时间**”。 TTFB是反映服务端响应速度的重要指标，对服务器来说，TTFB 时间越短，就说明服务器响应越快。

接收到第一个字节之后，进入陆续接收完整数据的阶段，也就是**Content Download阶段**，这意味着从第一字节时间到接收到全部响应数据所用的时间。

优化时间线上耗时项

了解了时间线面板上的各项含义之后，我们就可以根据这个请求的时间线来实现相关的优化操作了。

1. 排队（Queuing）时间过久

排队时间过久，大概率是由浏览器为每个域名最多维护6个连接导致的。那么基于这个原因，你就可以让1个站点下面的资源放在多个域名下面，比如放到3个域名下面，这样就可以同时支持18个连接了，这种方案称为**域名分片**技术。除了域名分片技术外，我个人还建议你**把站点升级到HTTP2**，因为HTTP2已经没有每个域名最多维护6个TCP连接的限制了。

2. 第一字节时间（TTFB）时间过久

这可能的原因有如下：

- **服务器生成页面数据的时间过久**。对于动态网页来说，服务器收到用户打开一个页面的请求时，首先要从数据库中读取该页面需要的数据，然后把这些数据传入到模板中，模板渲染后，再返回给用户。服务器在处理这个数据的过程中，可能某个环节会出问题。
- **网络的原因**。比如使用了低带宽的服务器，或者本来用的是电信的服务器，可联通的网络用户要来访问你的服务器，这样也会拖慢网速。
- **发送请求头时带上了多余的用户信息**。比如一些不必要的Cookie信息，服务器接收到这些Cookie信息之后可能需要对每一项都做处理，这样就加大了服务器的处理时长。

对于这三种问题，你要有针对性地出一些解决方案。面对第一种服务器的问题，你可以想办法去提高服务器的处理速度，比如通过增加各种缓存的技术；针对第二种网络问题，你可以使用CDN来缓存一些静态文件；至于第三种，你在发送请求时就去尽可能地减少一些不必要的Cookie数据信息。

3. Content Download时间过久

如果单个请求的Content Download花费了大量时间，有可能是字节数太多的原因导致的。这时候你就需要减少文件大小，比如压缩、去掉源码中不必要的注释等方法。

总结

好了，今天就介绍到这里了，下面我来总结下今天的内容。

首先我们简单介绍了Chrome开发者工具10个基础的面板信息；然后重点剖析了网络面板，再结合之前介绍的网络请求流程来重点分析了网络面板中时间线的各个指标的含义；最后我们还简要分析了时间线中各项指标出现异常的可能原因，并给出了一些优化方案。

其实通过今天的分析，我们可以得出这样一个结论：如果你要去做一些实践性的项目优化，理解其背后的理论至关重要。因为理论就是一条“线”，它会把各种实践的内容“串”在一起，然后你可以围绕着这条“线”来排查问题。

思考时间

今天我们介绍了网络面板，还有一个非常重要的Performance面板我们没有介绍，不过你可以去网上查找一些相关的资料。

所以今天留给你的是一道实际操作的题目，你可以结合网络面板和Performance面板来分析一个Web应用的性能瓶颈（比如https://www.12306.cn ）。

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

“浏览器中的页面循环系统”模块我们已经介绍完了，循环系统是页面的基础，理解了循环系统能让我们从本质上更好地理解页面的工作方式，加深我们对一些前端概念的理解。

接下来我们就要进入新的模块了，也就是“浏览器中的页面”模块，正如专栏简介中所言，页面是浏览器的核心，浏览器中的所有功能点都是服务于页面的，而Chrome开发者工具又是工程师调试页面的核心工具，所以在这个模块的开篇，我想先带你来深入了解下Chrome开发者工具。

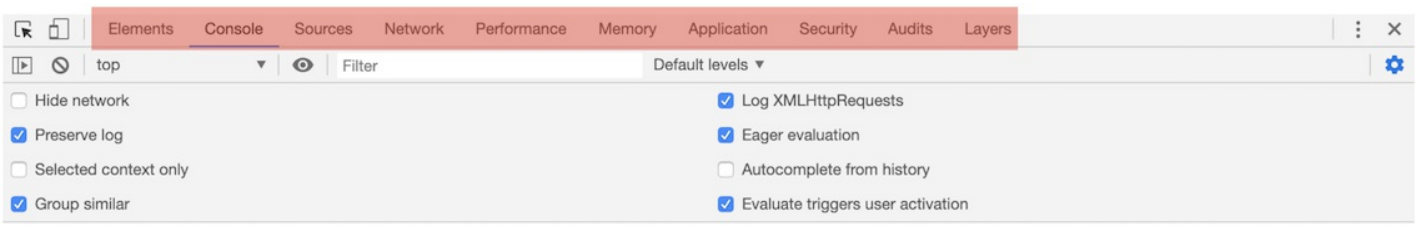
Chrome开发者工具（简称DevTools）是一组网页制作和调试的工具，内嵌于Google Chrome 浏览器中。Chrome开发者工具非常重要，所蕴含的内容也是非常多的，熟练使用它能让你更加深入地了解浏览器内部工作原理。（Chrome开发者工具也在不停地迭代改进，如果你想使用最新版本，可以使用Chrome Canary。）

作为这一模块的第一篇文章，我们主要聚焦**页面的源头**和**网络数据的接收**，这些发送和接收的数据都能体现在开发者工具的网络面板上。不过为了你能更好地理解和掌握，我们会先对Chrome开发者工具做一个大致的介绍，然后再深入剖析网络面板。

Chrome开发者工具

Chrome开发者工具有很多重要的面板，比如与性能相关的有网络面板、Performance面板、内存面板等，与调试页面相关的有Elements面板、Sources面板、Console面板等。

你可以在浏览器窗口的右上方选择Chrome菜单，然后选择“更多工具–>开发者工具”来打开Chrome开发者工具。打开的页面如下图所示：



从图中可以看出，它一共包含了10个功能面板，包括了Elements、Console、Sources、NetWork、Performance、Memory、Application、Security、Audits和Layers。

关于这10个面板的大致功能，我做了一个表格，感兴趣的话，你可以详细看下：

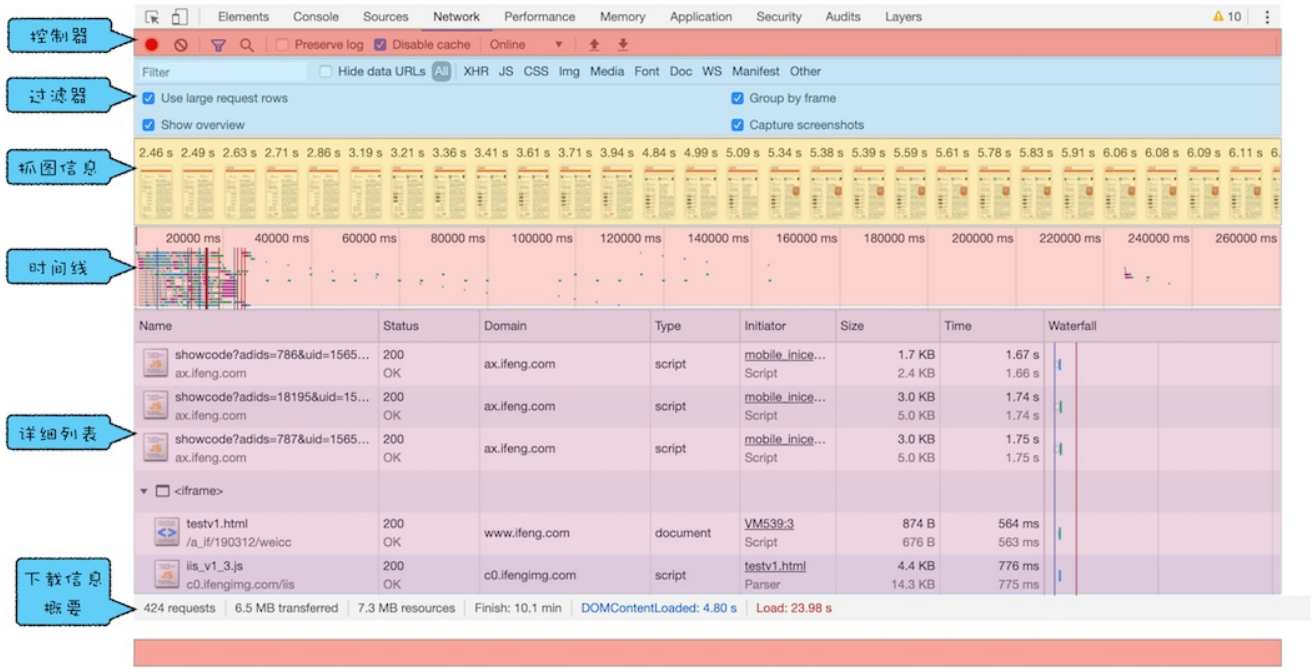
名称	描述
Elements面板	可以查看DOM结构、编辑CSS样式，用于测试页面布局和设计页面。
Console面板	可以看成是JavaScript Shell，能执行JavaScript脚本。通过Console还能和页面中的JavaScript对象交互。
Sources面板	1. 查看Web应用加载的所有文件； 2. 编辑CSS和JavaScript文件内容； 3. 将打乱的CSS文件或者JavaScript文件格式化； 4. 支持JavaScript的调试功能； 5. 设置工作区，将更改的文件保存到本地文件夹中。
NetWork（网络）面板	展示了页面中所有的请求内容列表，能查看每项请求的请求行、请求头、请求体、时间线以及网络请求瀑布图等信息。
Performance面板	记录和查看Web应用生命周期内的各种事件，并用来分析在执行过程中一些影响性能的要点。
Memory面板	用来查看运行过程中的JavaScript占用堆内存情况，追踪是否存在内存泄漏的情况等。
Application（应用）面板	查看Web应用的数据存储情况； PWA的基础数据：IndexedDB；Web SQL；本地和会话存储；Cookie；应用程序缓存；图像；字体和样式表等。
Security（安全）面板	显示当前页面一些基础的安全信息。
Audits面板	会对当前网页进行网络利用情况、网页性能方面的诊断，并给出一些优化建议。
Layers面板	展示一些渲染过程中分层的基础信息。

简单来说，Chrome开发者工具为我们提供了通过界面访问或者编辑DOM和CSSOM的能力，还提供了强大的调试功能和查看性能指标的能力。

OK，接下来我们就要重点看下其中重要的Network面板，即网络面板。

网络面板

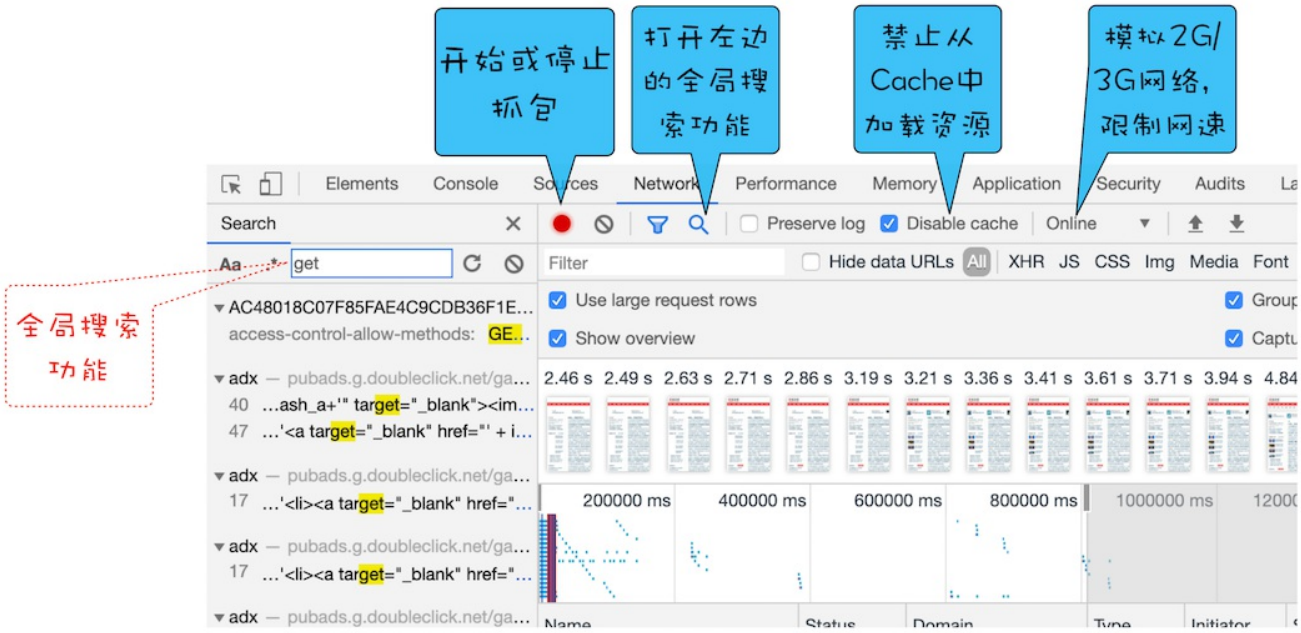
网络面板由控制器、过滤器、抓图信息、时间线、详细列表和下载信息概要这6个区域构成（如下图所示）。



网络面板概要图

1. 控制器

其中，控制器有4个比较重要的功能，我们按照下文中的这张图来简单介绍下。



控制器概要图

- 红色圆点的按钮，表示“开始/暂停抓包”，这个功能很常见，很容易理解。
- “全局搜索”按钮，这个功能就非常重要了，可以在所有下载资源中搜索相关内容，还可以快速定位到某几个你想要的文件上。
- Disable cache，即“禁止从Cache中加载资源”的功能，它在调试Web应用的时候非常有用，因为开启了Cache会影响到网络性能测试的结果。
- Online按钮，是“模拟2G/3G”功能，它可以限制带宽，模拟弱网情况下页面的展现情况，然后你就可以根据实际展示情况来动态调整策略，以便让Web应用更加适用于这些弱网。

2. 过滤器

网络面板中的过滤器，主要就是起过滤功能。因为有时候一个页面有太多内容在详细列表区域中展示了，而你可能只想查看JavaScript文件或者CSS文件，这时候就可以通过过滤器模块来筛选你想要的文件类型。

3. 抓图信息

抓图信息区域，可以用来分析用户等待页面加载时间内所看到的内容，分析用户实际的体验情况。比如，如果页面加载1秒多之后屏幕截图还是白屏状态，这时候就需要分析是网络还是代码的问题了。（勾选面板上的“Capture screenshots”即可启用屏幕截图。）

4. 时间线

时间线，主要用来展示HTTP、HTTPS、WebSocket加载的状态和时间的一个关系，用于直观感受页面的加载过程。如果是多条竖线堆叠在一起，那说明这些资源被同时被加载。至于具体到每个文件的加载信息，还需要用到下面要讲的详细列表。

5. 详细列表

这个区域是最重要的，它详细记录了每个资源从发起请求到完成请求这中间所有过程的状态，以及最终请求完成的数据信息。通过该列表，你就能很容易地去诊断一些网络问题。

详细列表是我们本篇文章介绍的重点，不过内容比较多，所以放到最后去专门介绍了。

6. 下载信息概要

下载信息概要中，你要重点关注下DOMContentLoaded和Load两个事件，以及这两个事件的完成时间。

- DOMContentLoaded，这个事件发生后，说明页面已经构建好DOM了，这意味着构建DOM所需要的HTML文件、JavaScript文件、CSS文件都已经下载完成了。
- Load，说明浏览器已经加载了所有的资源（图像、样式表等）。

通过下载信息概要面板，你可以查看触发这两个事件所花费的时间。

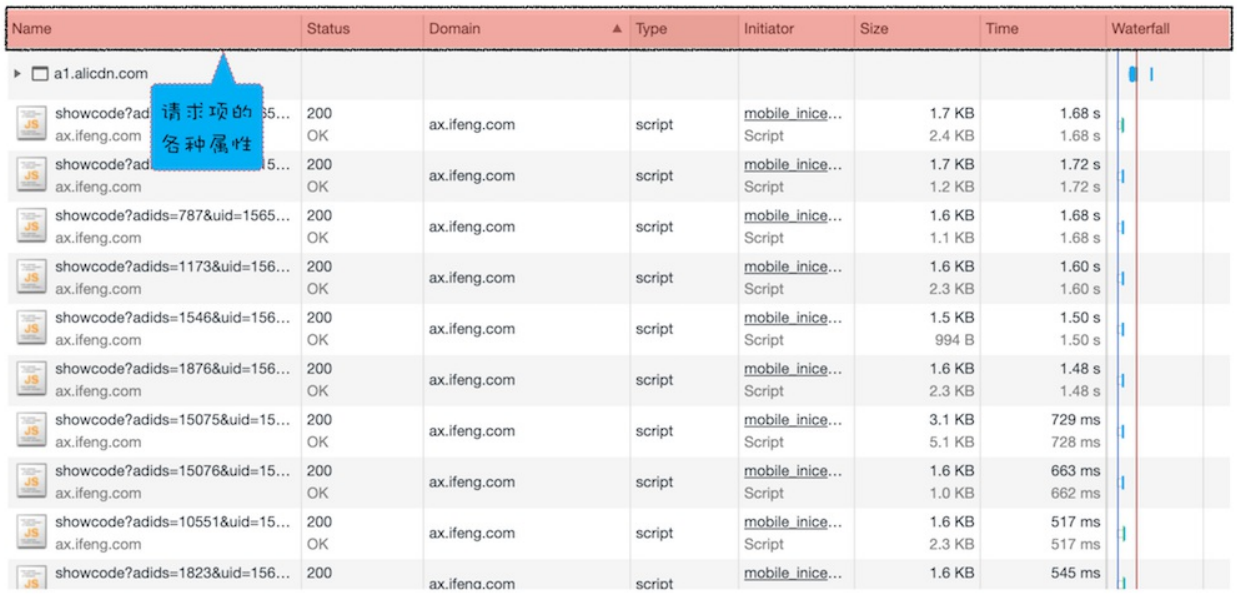
网络面板中的详细列表

下面我们就来重点介绍网络面板中的详细列表，这里面包含了大量有用的信息。

1. 列表的属性

列表的属性比较多，比如Name、Status、Type、Initiator等等，这个不难理解。当然，你还可以通过点击右键的下拉菜单来添加其他属性，这里我就不再赘述了，你可以自己上手实操一下。

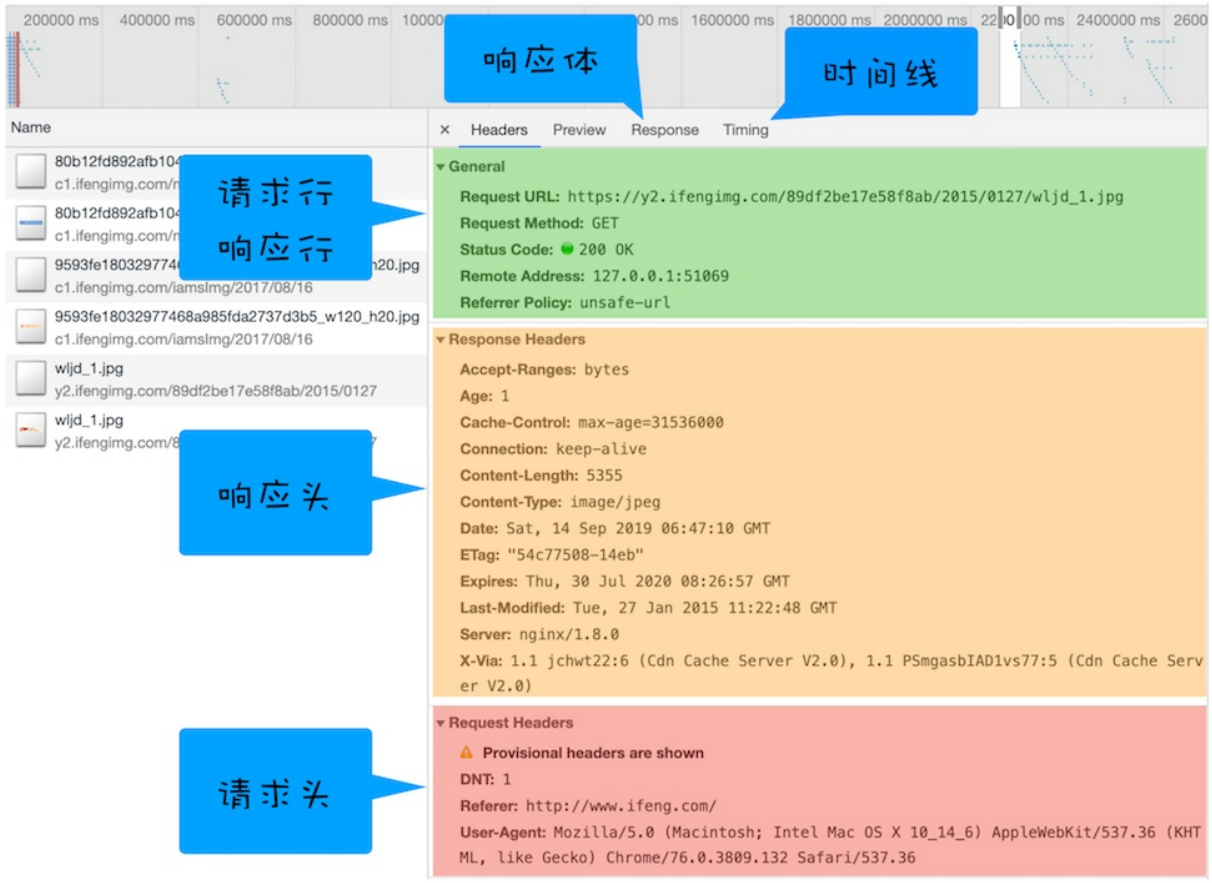
另外，你也可以按照列表的属性来给列表排序，默认情况下，列表是按请求发起的时间来排序的，最早发起请求的资源在顶部。当然也可以按照返回状态码、请求类型、请求时长、内容大小等基础属性排序，只需点击相应属性即可。



根据属性排序

2. 详细信息

如果你选中详细列表中的一项，右边就会出现该项的详细信息，如下所示：

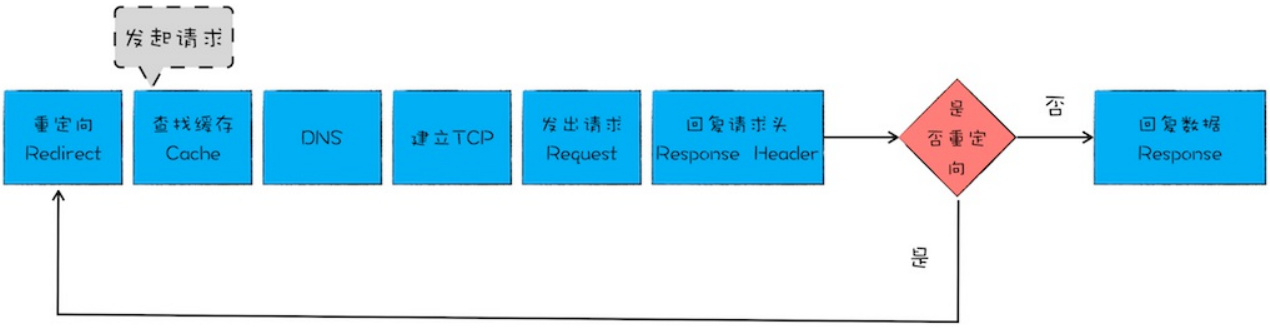


详细请求信息

你可以在此查看请求列表中任意一项的请求行和请求头信息，还可以查看响应行、响应头和响应体。然后你可以根据这些查看的信息来判断你的业务逻辑是否正确，或者有时候也可以用来逆向推导别人网站的业务逻辑。

3. 单个资源的时间线

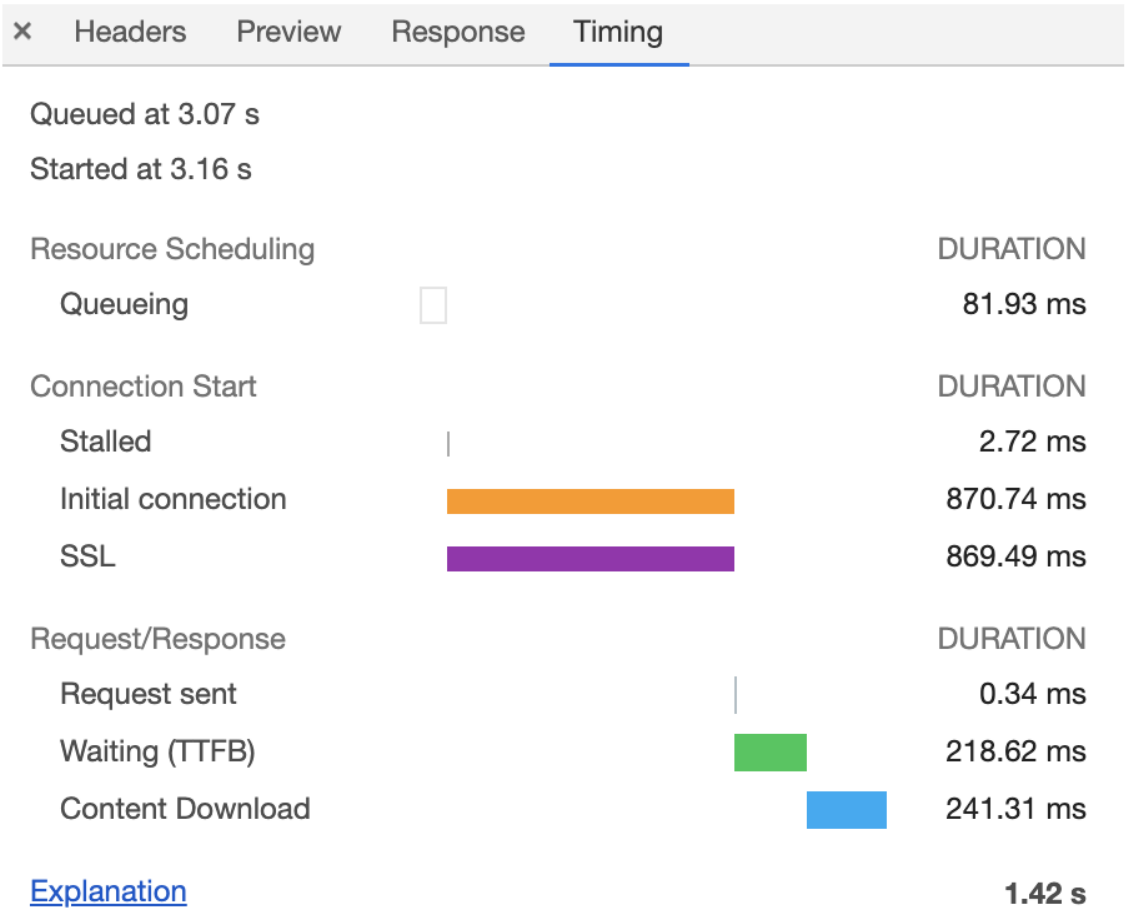
了解了每个资源的详细请求信息之后，我们再来分析单个资源请求时间线，这就涉及具体的HTTP请求流程了。



浏览器中HTTP请求流程

我们再回顾下在《03 | HTTP请求流程：为什么很多站点第二次打开速度会很快？》这篇文章，我们介绍过发起一个HTTP请求之后，浏览器首先查找缓存，如果缓存没有命中，那么继续发起DNS请求获取IP地址，然后利用IP地址和服务器端建立TCP连接，再发送HTTP请求，等待服务器响应；不过，如果服务器响应头中包含了重定向的信息，那么整个流程就需要重新再走一遍。这就是在浏览器中一个HTTP请求的基础流程。

那详细列表中是如何表示出这个流程的呢？这就要重点看下时间线面板了：



单个文件的时间线

那面板中这各项到底是什么含义呢？

第一个是Queuing，也就是排队的意思，当浏览器发起一个请求的时候，会有很多原因导致该请求不能被立即执行，而是需要排队等待。导致请求处于排队状态的原因有很多。

- 首先，页面中的资源是有优先级的，比如CSS、HTML、JavaScript等都是页面中的核心文件，所以优先级最高；而图片、视频、音频这类资源就不是核心资源，优先级就比较低。通常当后者遇到前者时，就需要“让路”，进入待排队状态。
- 其次，我们前面也提到过，浏览器会为每个域名最多维护6个TCP连接，如果发起一个HTTP请求时，这6个TCP连接都处于忙碌状态，那么这个请求就会处于排队状态。
- 最后，网络进程在为数据分配磁盘空间时，新的HTTP请求也需要短暂地等待磁盘分配结束。

等待排队完成之后，就要进入发起连接的状态了。不过在发起连接之前，还有一些原因可能导致连接过程被推迟，这个推迟就表现在面板中的**Stalled**上，它表示停滞的意思。

这里需要额外说明的是，如果你使用了代理服务器，还会增加一个**Proxy Negotiation**阶段，也就是代理协商阶段，它表示代理服务器连接协商所用的时间，不过在上图中没有体现出来，因为这里我们没有使用代理服务器。

接下来，就到了**Initial connection/SSL阶段**了，也就是和服务器建立连接的阶段，这包括了建立TCP连接所花费的时间；不过如果你使用了HTTPS协议，那么还需要一个额外的SSL握手时间，这个过程主要是用来协商一些加密信息的。（关于SSL协商的详细过程，我们会在Web安全模块中介绍。）

和服务器建立好连接之后，网络进程会准备请求数据，并将其发送给网络，这就是**Request sent阶段**。通常这个阶段非常快，因为只需要把浏览器缓冲区的数据发送出去就结束了，并不需要判断服务器是否接收到了，所以这个时间通常不到1毫秒。

数据发送出去了，接下来就是等待接收服务器第一个字节的数据，这个阶段称为Waiting (TTFB)，通常也称为“**第一字节时间**”。 TTFB是反映服务端响应速度的重要指标，对服务器来说，TTFB 时间越短，就说明服务器响应越快。

接收到第一个字节之后，进入陆续接收完整数据的阶段，也就是**Content Download阶段**，这意味着从第一字节时间到接收到全部响应数据所用的时间。

优化时间线上耗时项

了解了时间线面板上的各项含义之后，我们就可以根据这个请求的时间线来实现相关的优化操作了。

1. 排队（Queuing）时间过久

排队时间过久，大概率是由浏览器为每个域名最多维护6个连接导致的。那么基于这个原因，你就可以让1个站点下面的资源放在多个域名下面，比如放到3个域名下面，这样就可以同时支持18个连接了，这种方案称为**域名分片**技术。除了域名分片技术外，我个人还建议你**把站点升级到HTTP2**，因为HTTP2已经没有每个域名最多维护6个TCP连接的限制了。

2. 第一字节时间（TTFB）时间过久

这可能的原因有如下：

- **服务器生成页面数据的时间过久**。对于动态网页来说，服务器收到用户打开一个页面的请求时，首先要从数据库中读取该页面需要的数据，然后把这些数据传入到模板中，模板渲染后，再返回给用户。服务器在处理这个数据的过程中，可能某个环节会出问题。
- **网络的原因**。比如使用了低带宽的服务器，或者本来用的是电信的服务器，可联通的网络用户要来访问你的服务器，这样也会拖慢网速。
- **发送请求头时带上了多余的用户信息**。比如一些不必要的Cookie信息，服务器接收到这些Cookie信息之后可能需要对每一项都做处理，这样就加大了服务器的处理时长。

对于这三种问题，你要有针对性地出一些解决方案。面对第一种服务器的问题，你可以想办法去提高服务器的处理速度，比如通过增加各种缓存的技术；针对第二种网络问题，你可以使用CDN来缓存一些静态文件；至于第三种，你在发送请求时就去尽可能地减少一些不必要的Cookie数据信息。

3. Content Download时间过久

如果单个请求的Content Download花费了大量时间，有可能是字节数太多的原因导致的。这时候你就需要减少文件大小，比如压缩、去掉源码中不必要的注释等方法。

总结

好了，今天就介绍到这里了，下面我来总结下今天的内容。

首先我们简单介绍了Chrome开发者工具10个基础的面板信息；然后重点剖析了网络面板，再结合之前介绍的网络请求流程来重点分析了网络面板中时间线的各个指标的含义；最后我们还简要分析了时间线中各项指标出现异常的可能原因，并给出了一些优化方案。

其实通过今天的分析，我们可以得出这样一个结论：如果你要去做一些实践性的项目优化，理解其背后的理论至关重要。因为理论就是一条“线”，它会把各种实践的内容“串”在一起，然后你可以围绕着这条“线”来排查问题。

思考时间

今天我们介绍了网络面板，还有一个非常重要的Performance面板我们没有介绍，不过你可以去网上查找一些相关的资料。

所以今天留给你的是一道实际操作的题目，你可以结合网络面板和Performance面板来分析一个Web应用的性能瓶颈（比如https://www.12306.cn ）。

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

“浏览器中的页面循环系统”模块我们已经介绍完了，循环系统是页面的基础，理解了循环系统能让我们从本质上更好地理解页面的工作方式，加深我们对一些前端概念的理解。

接下来我们就要进入新的模块了，也就是“浏览器中的页面”模块，正如专栏简介中所言，页面是浏览器的核心，浏览器中的所有功能点都是服务于页面的，而Chrome开发者工具又是工程师调试页面的核心工具，所以在这个模块的开篇，我想先带你来深入了解下Chrome开发者工具。

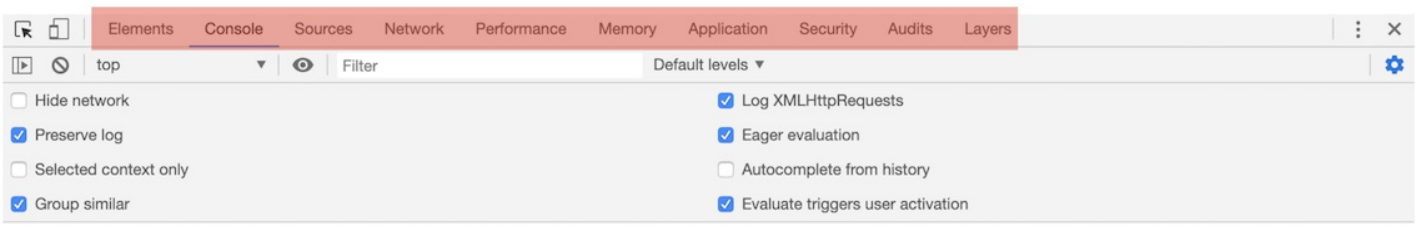
Chrome开发者工具（简称DevTools）是一组网页制作和调试的工具，内嵌于Google Chrome 浏览器中。Chrome开发者工具非常重要，所蕴含的内容也是非常多的，熟练使用它能让你更加深入地了解浏览器内部工作原理。（Chrome开发者工具也在不停地迭代改进，如果你想使用最新版本，可以使用Chrome Canary。）

作为这一模块的第一篇文章，我们主要聚焦**页面的源头**和**网络数据的接收**，这些发送和接收的数据都能体现在开发者工具的网络面板上。不过为了你能更好地理解和掌握，我们会先对Chrome开发者工具做一个大致的介绍，然后再深入剖析网络面板。

Chrome开发者工具

Chrome开发者工具有很多重要的面板，比如与性能相关的有网络面板、Performance面板、内存面板等，与调试页面相关的有Elements面板、Sources面板、Console面板等。

你可以在浏览器窗口的右上方选择Chrome菜单，然后选择“更多工具–>开发者工具”来打开Chrome开发者工具。打开的页面如下图所示：



从图中可以看出，它一共包含了10个功能面板，包括了Elements、Console、Sources、NetWork、Performance、Memory、Application、Security、Audits和Layers。

关于这10个面板的大致功能，我做了一个表格，感兴趣的话，你可以详细看下：

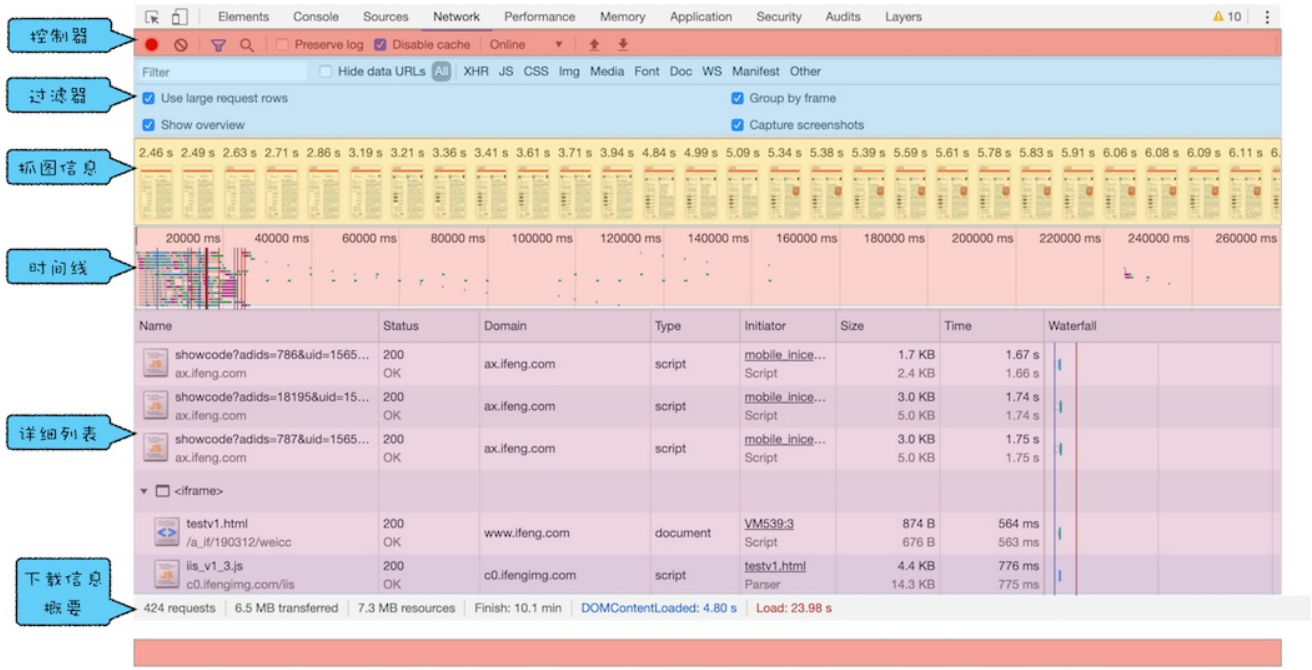
名称	描述
Elements面板	可以查看DOM结构、编辑CSS样式，用于测试页面布局和设计页面。
Console面板	可以看成是JavaScript Shell，能执行JavaScript脚本。通过Console还能和页面中的JavaScript对象交互。
Sources面板	1. 查看Web应用加载的所有文件； 2. 编辑CSS和JavaScript文件内容； 3. 将打乱的CSS文件或者JavaScript文件格式化； 4. 支持JavaScript的调试功能； 5. 设置工作区，将更改的文件保存到本地文件夹中。
NetWork（网络）面板	展示了页面中所有的请求内容列表，能查看每项请求的请求行、请求头、请求体、时间线以及网络请求瀑布图等信息。
Performance面板	记录和查看Web应用生命周期内的各种事件，并用来分析在执行过程中一些影响性能的要点。
Memory面板	用来查看运行过程中的JavaScript占用堆内存情况，追踪是否存在内存泄漏的情况等。
Application（应用）面板	查看Web应用的数据存储情况； PWA的基础数据：IndexedDB；Web SQL；本地和会话存储；Cookie；应用程序缓存；图像；字体和样式表等。
Security（安全）面板	显示当前页面一些基础的安全信息。
Audits面板	会对当前网页进行网络利用情况、网页性能方面的诊断，并给出一些优化建议。
Layers面板	展示一些渲染过程中分层的基础信息。

简单来说，Chrome开发者工具为我们提供了通过界面访问或者编辑DOM和CSSOM的能力，还提供了强大的调试功能和查看性能指标的能力。

OK，接下来我们就要重点看下其中重要的Network面板，即网络面板。

网络面板

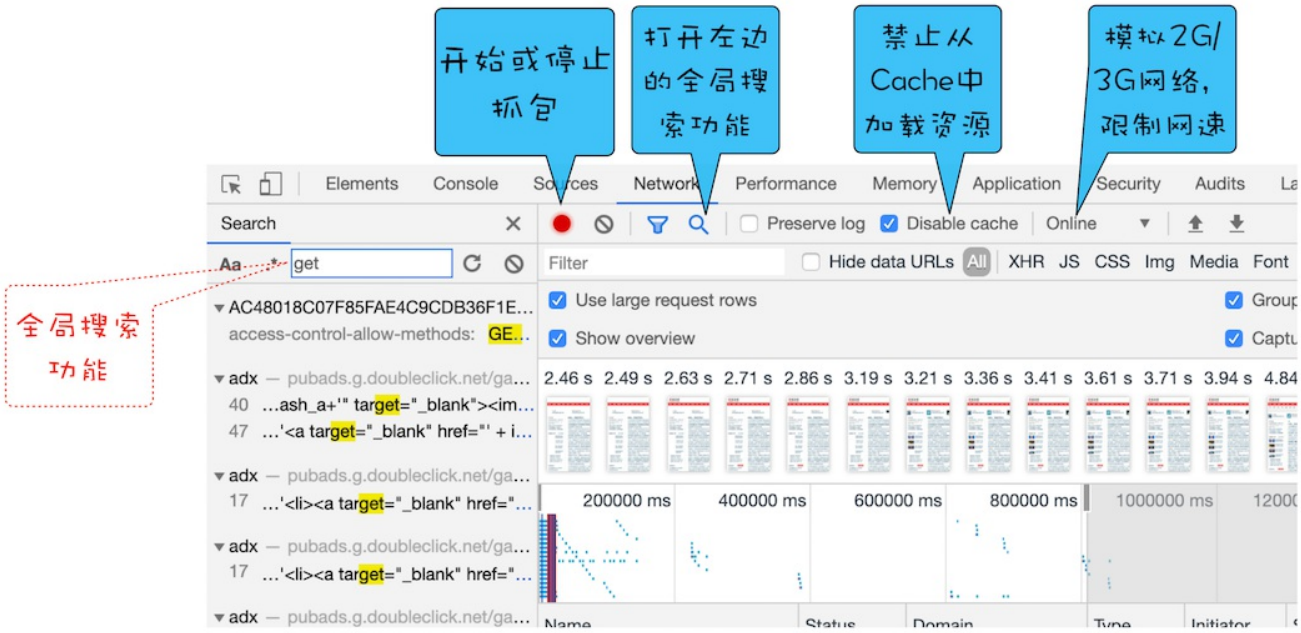
网络面板由控制器、过滤器、抓图信息、时间线、详细列表和下载信息概要这6个区域构成（如下图所示）。



网络面板概要图

1. 控制器

其中，控制器有4个比较重要的功能，我们按照下文中的这张图来简单介绍下。



控制器概要图

- 红色圆点的按钮，表示“开始/暂停抓包”，这个功能很常见，很容易理解。
- “全局搜索”按钮，这个功能就非常重要了，可以在所有下载资源中搜索相关内容，还可以快速定位到某几个你想要的文件上。
- Disable cache，即“禁止从Cache中加载资源”的功能，它在调试Web应用的时候非常有用，因为开启了Cache会影响到网络性能测试的结果。
- Online按钮，是“模拟2G/3G”功能，它可以限制带宽，模拟弱网情况下页面的展现情况，然后你就可以根据实际展示情况来动态调整策略，以便让Web应用更加适用于这些弱网。

2. 过滤器

网络面板中的过滤器，主要就是起过滤功能。因为有时候一个页面有太多内容在详细列表区域中展示了，而你可能只想查看JavaScript文件或者CSS文件，这时候就可以通过过滤器模块来筛选你想要的文件类型。

3. 抓图信息

抓图信息区域，可以用来分析用户等待页面加载时间内所看到的内容，分析用户实际的体验情况。比如，如果页面加载1秒多之后屏幕截图还是白屏状态，这时候就需要分析是网络还是代码的问题了。（勾选面板上的“Capture screenshots”即可启用屏幕截图。）

4. 时间线

时间线，主要用来展示HTTP、HTTPS、WebSocket加载的状态和时间的一个关系，用于直观感受页面的加载过程。如果是多条竖线堆叠在一起，那说明这些资源被同时被加载。至于具体到每个文件的加载信息，还需要用到下面要讲的详细列表。

5. 详细列表

这个区域是最重要的，它详细记录了每个资源从发起请求到完成请求这中间所有过程的状态，以及最终请求完成的数据信息。通过该列表，你就能很容易地去诊断一些网络问题。

详细列表是我们本篇文章介绍的重点，不过内容比较多，所以放到最后去专门介绍了。

6. 下载信息概要

下载信息概要中，你要重点关注下DOMContentLoaded和Load两个事件，以及这两个事件的完成时间。

- DOMContentLoaded，这个事件发生后，说明页面已经构建好DOM了，这意味着构建DOM所需要的HTML文件、JavaScript文件、CSS文件都已经下载完成了。
- Load，说明浏览器已经加载了所有的资源（图像、样式表等）。

通过下载信息概要面板，你可以查看触发这两个事件所花费的时间。

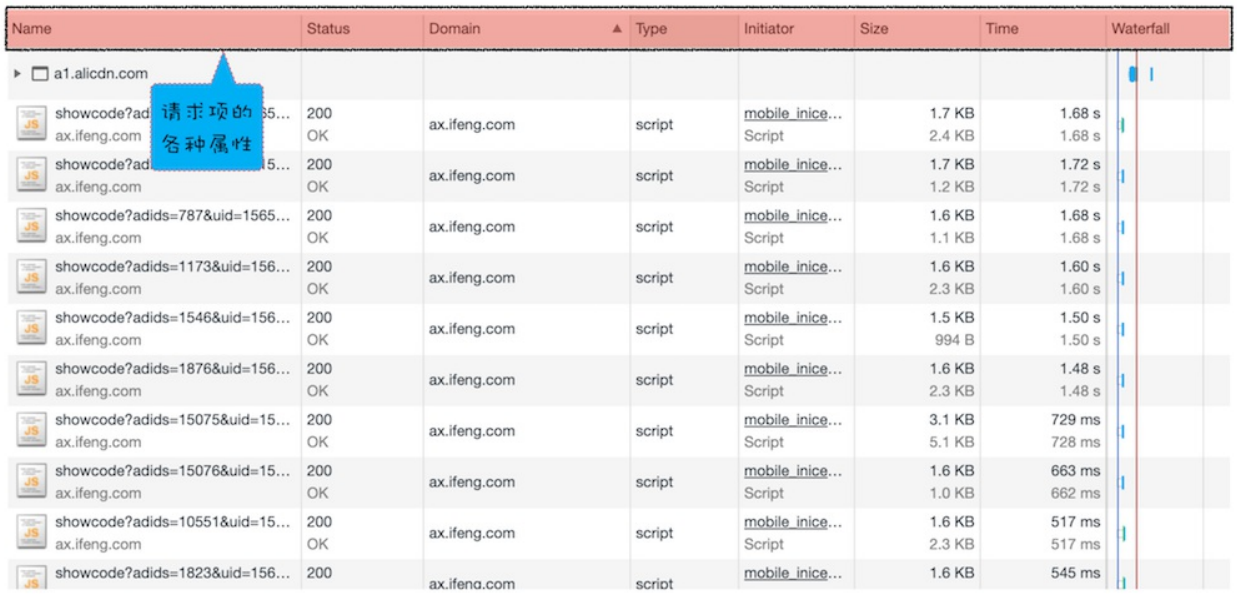
网络面板中的详细列表

下面我们就来重点介绍网络面板中的详细列表，这里面包含了大量有用的信息。

1. 列表的属性

列表的属性比较多，比如Name、Status、Type、Initiator等等，这个不难理解。当然，你还可以通过点击右键的下拉菜单来添加其他属性，这里我就不再赘述了，你可以自己上手实操一下。

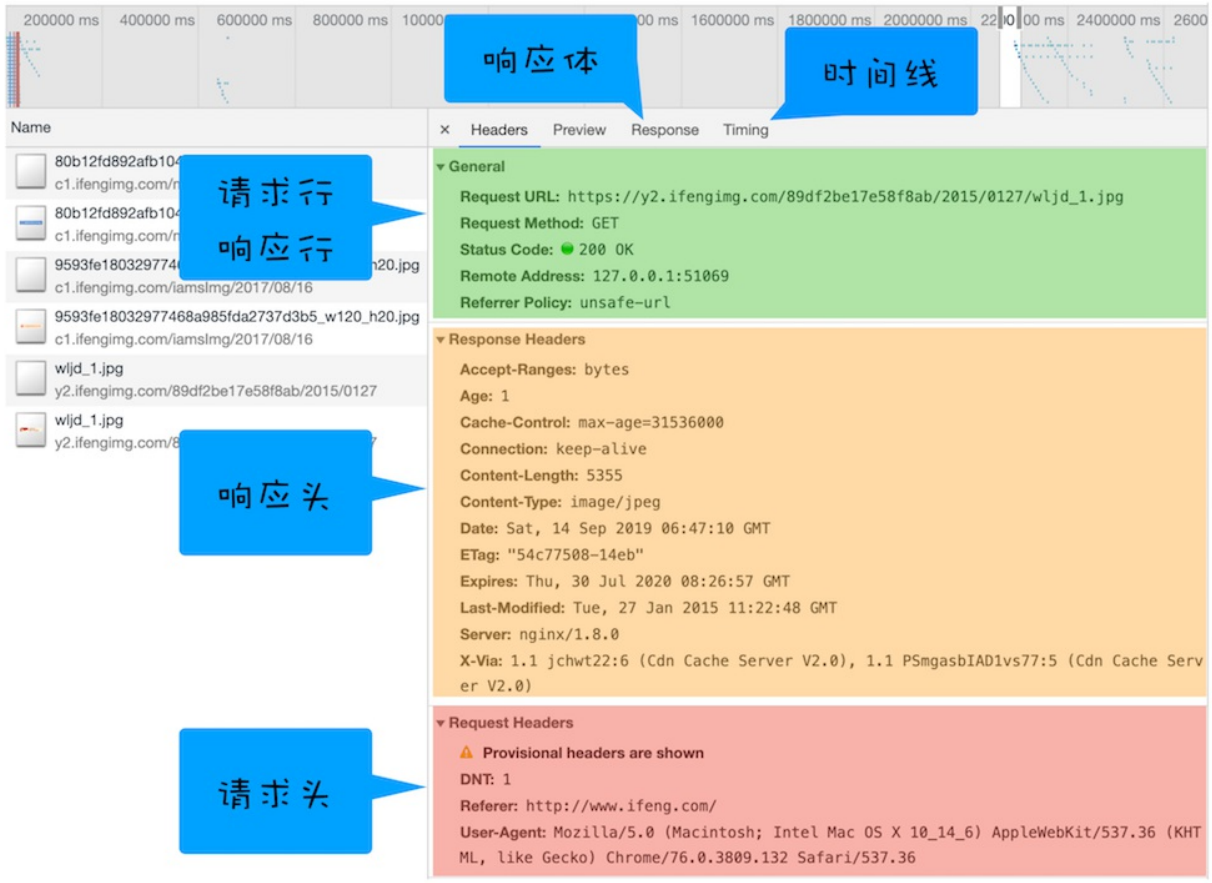
另外，你也可以按照列表的属性来给列表排序，默认情况下，列表是按请求发起的时间来排序的，最早发起请求的资源在顶部。当然也可以按照返回状态码、请求类型、请求时长、内容大小等基础属性排序，只需点击相应属性即可。



根据属性排序

2. 详细信息

如果你选中详细列表中的一项，右边就会出现该项的详细信息，如下所示：

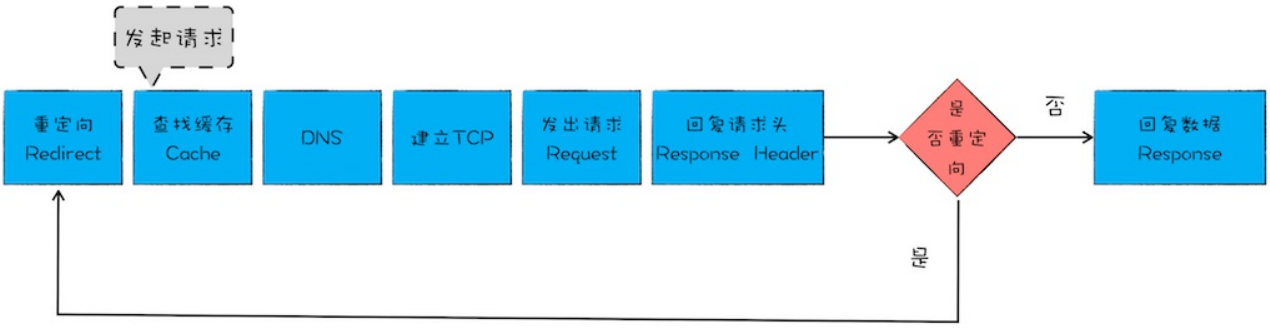


详细请求信息

你可以在此查看请求列表中任意一项的请求行和请求头信息，还可以查看响应行、响应头和响应体。然后你可以根据这些查看的信息来判断你的业务逻辑是否正确，或者有时候也可以用来逆向推导别人网站的业务逻辑。

3. 单个资源的时间线

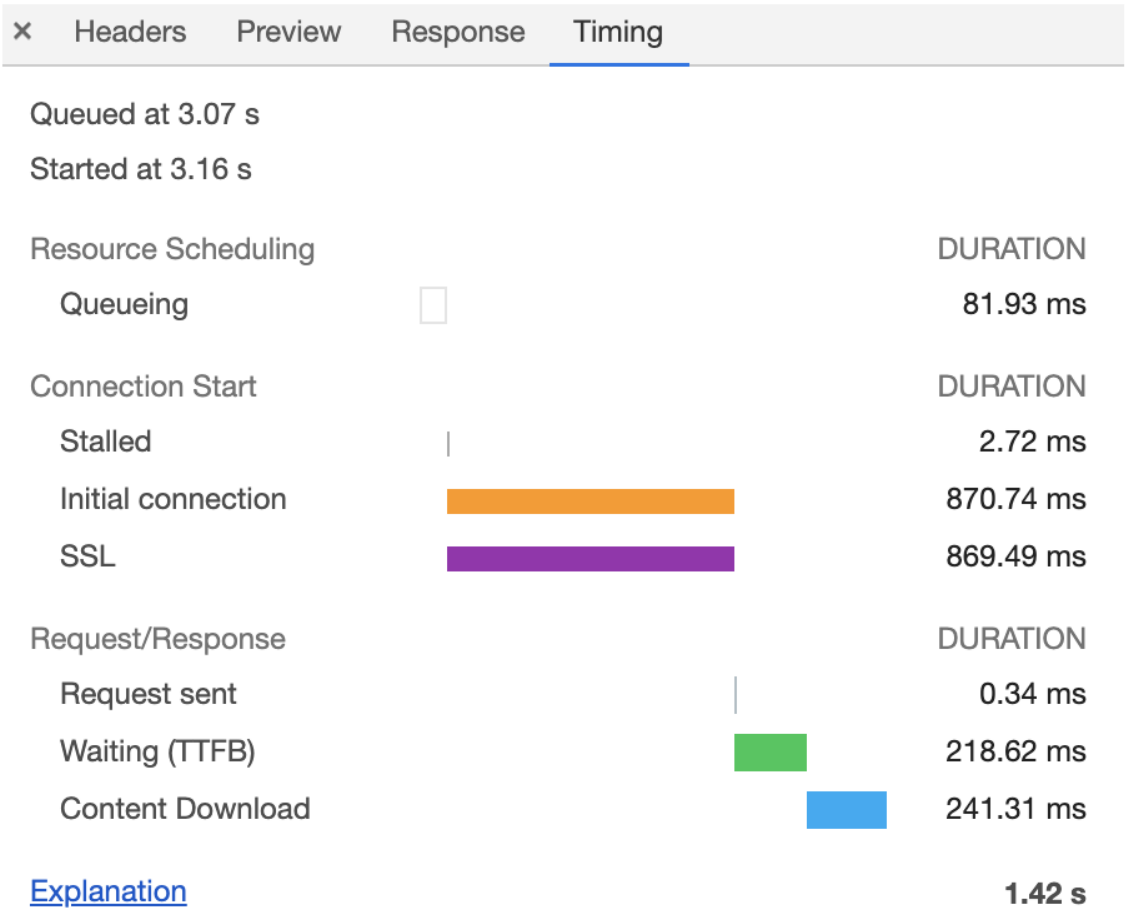
了解了每个资源的详细请求信息之后，我们再来分析单个资源请求时间线，这就涉及具体的HTTP请求流程了。



浏览器中HTTP请求流程

我们再回顾下在《03 | HTTP请求流程：为什么很多站点第二次打开速度会很快？》这篇文章，我们介绍过发起一个HTTP请求之后，浏览器首先查找缓存，如果缓存没有命中，那么继续发起DNS请求获取IP地址，然后利用IP地址和服务器端建立TCP连接，再发送HTTP请求，等待服务器响应；不过，如果服务器响应头中包含了重定向的信息，那么整个流程就需要重新再走一遍。这就是在浏览器中一个HTTP请求的基础流程。

那详细列表中是如何表示出这个流程的呢？这就要重点看下时间线面板了：



单个文件的时间线

那面板中这各项到底是什么含义呢？

第一个是Queuing，也就是排队的意思，当浏览器发起一个请求的时候，会有很多原因导致该请求不能被立即执行，而是需要排队等待。导致请求处于排队状态的原因有很多。

- 首先，页面中的资源是有优先级的，比如CSS、HTML、JavaScript等都是页面中的核心文件，所以优先级最高；而图片、视频、音频这类资源就不是核心资源，优先级就比较低。通常当后者遇到前者时，就需要“让路”，进入待排队状态。
- 其次，我们前面也提到过，浏览器会为每个域名最多维护6个TCP连接，如果发起一个HTTP请求时，这6个TCP连接都处于忙碌状态，那么这个请求就会处于排队状态。
- 最后，网络进程在为数据分配磁盘空间时，新的HTTP请求也需要短暂地等待磁盘分配结束。

等待排队完成之后，就要进入发起连接的状态了。不过在发起连接之前，还有一些原因可能导致连接过程被推迟，这个推迟就表现在面板中的**Stalled**上，它表示停滞的意思。

这里需要额外说明的是，如果你使用了代理服务器，还会增加一个**Proxy Negotiation**阶段，也就是代理协商阶段，它表示代理服务器连接协商所用的时间，不过在上图中没有体现出来，因为这里我们没有使用代理服务器。

接下来，就到了**Initial connection/SSL阶段**了，也就是和服务器建立连接的阶段，这包括了建立TCP连接所花费的时间；不过如果你使用了HTTPS协议，那么还需要一个额外的SSL握手时间，这个过程主要是用来协商一些加密信息的。（关于SSL协商的详细过程，我们会在Web安全模块中介绍。）

和服务器建立好连接之后，网络进程会准备请求数据，并将其发送给网络，这就是**Request sent阶段**。通常这个阶段非常快，因为只需要把浏览器缓冲区的数据发送出去就结束了，并不需要判断服务器是否接收到了，所以这个时间通常不到1毫秒。

数据发送出去了，接下来就是等待接收服务器第一个字节的数据，这个阶段称为Waiting (TTFB)，通常也称为“**第一字节时间**”。 TTFB是反映服务端响应速度的重要指标，对服务器来说，TTFB 时间越短，就说明服务器响应越快。

接收到第一个字节之后，进入陆续接收完整数据的阶段，也就是**Content Download阶段**，这意味着从第一字节时间到接收到全部响应数据所用的时间。

优化时间线上耗时项

了解了时间线面板上的各项含义之后，我们就可以根据这个请求的时间线来实现相关的优化操作了。

1. 排队（Queuing）时间过久

排队时间过久，大概率是由浏览器为每个域名最多维护6个连接导致的。那么基于这个原因，你就可以让1个站点下面的资源放在多个域名下面，比如放到3个域名下面，这样就可以同时支持18个连接了，这种方案称为**域名分片**技术。除了域名分片技术外，我个人还建议你**把站点升级到HTTP2**，因为HTTP2已经没有每个域名最多维护6个TCP连接的限制了。

2. 第一字节时间（TTFB）时间过久

这可能的原因有如下：

- **服务器生成页面数据的时间过久**。对于动态网页来说，服务器收到用户打开一个页面的请求时，首先要从数据库中读取该页面需要的数据，然后把这些数据传入到模板中，模板渲染后，再返回给用户。服务器在处理这个数据的过程中，可能某个环节会出问题。
- **网络的原因**。比如使用了低带宽的服务器，或者本来用的是电信的服务器，可联通的网络用户要来访问你的服务器，这样也会拖慢网速。
- **发送请求头时带上了多余的用户信息**。比如一些不必要的Cookie信息，服务器接收到这些Cookie信息之后可能需要对每一项都做处理，这样就加大了服务器的处理时长。

对于这三种问题，你要有针对性地出一些解决方案。面对第一种服务器的问题，你可以想办法去提高服务器的处理速度，比如通过增加各种缓存的技术；针对第二种网络问题，你可以使用CDN来缓存一些静态文件；至于第三种，你在发送请求时就去尽可能地减少一些不必要的Cookie数据信息。

3. Content Download时间过久

如果单个请求的Content Download花费了大量时间，有可能是字节数太多的原因导致的。这时候你就需要减少文件大小，比如压缩、去掉源码中不必要的注释等方法。

总结

好了，今天就介绍到这里了，下面我来总结下今天的内容。

首先我们简单介绍了Chrome开发者工具10个基础的面板信息；然后重点剖析了网络面板，再结合之前介绍的网络请求流程来重点分析了网络面板中时间线的各个指标的含义；最后我们还简要分析了时间线中各项指标出现异常的可能原因，并给出了一些优化方案。

其实通过今天的分析，我们可以得出这样一个结论：如果你要去做一些实践性的项目优化，理解其背后的理论至关重要。因为理论就是一条“线”，它会把各种实践的内容“串”在一起，然后你可以围绕着这条“线”来排查问题。

思考时间

今天我们介绍了网络面板，还有一个非常重要的Performance面板我们没有介绍，不过你可以去网上查找一些相关的资料。

所以今天留给你的是一道实际操作的题目，你可以结合网络面板和Performance面板来分析一个Web应用的性能瓶颈（比如https://www.12306.cn ）。

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

“浏览器中的页面循环系统”模块我们已经介绍完了，循环系统是页面的基础，理解了循环系统能让我们从本质上更好地理解页面的工作方式，加深我们对一些前端概念的理解。

接下来我们就要进入新的模块了，也就是“浏览器中的页面”模块，正如专栏简介中所言，页面是浏览器的核心，浏览器中的所有功能点都是服务于页面的，而Chrome开发者工具又是工程师调试页面的核心工具，所以在这个模块的开篇，我想先带你来深入了解下Chrome开发者工具。

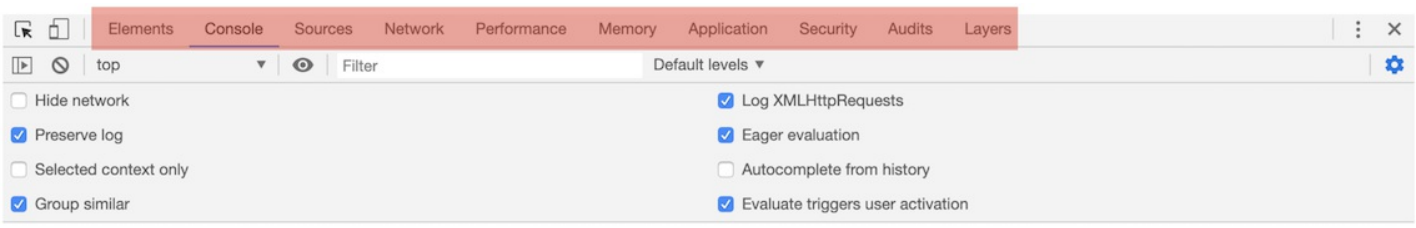
Chrome开发者工具（简称DevTools）是一组网页制作和调试的工具，内嵌于Google Chrome 浏览器中。Chrome开发者工具非常重要，所蕴含的内容也是非常多的，熟练使用它能让你更加深入地了解浏览器内部工作原理。（Chrome开发者工具也在不停地迭代改进，如果你想使用最新版本，可以使用Chrome Canary。）

作为这一模块的第一篇文章，我们主要聚焦**页面的源头**和**网络数据的接收**，这些发送和接收的数据都能体现在开发者工具的网络面板上。不过为了你能更好地理解和掌握，我们会先对Chrome开发者工具做一个大致的介绍，然后再深入剖析网络面板。

Chrome开发者工具

Chrome开发者工具有很多重要的面板，比如与性能相关的有网络面板、Performance面板、内存面板等，与调试页面相关的有Elements面板、Sources面板、Console面板等。

你可以在浏览器窗口的右上方选择Chrome菜单，然后选择“更多工具–>开发者工具”来打开Chrome开发者工具。打开的页面如下图所示：



从图中可以看出，它一共包含了10个功能面板，包括了Elements、Console、Sources、NetWork、Performance、Memory、Application、Security、Audits和Layers。

关于这10个面板的大致功能，我做了一个表格，感兴趣的话，你可以详细看下：

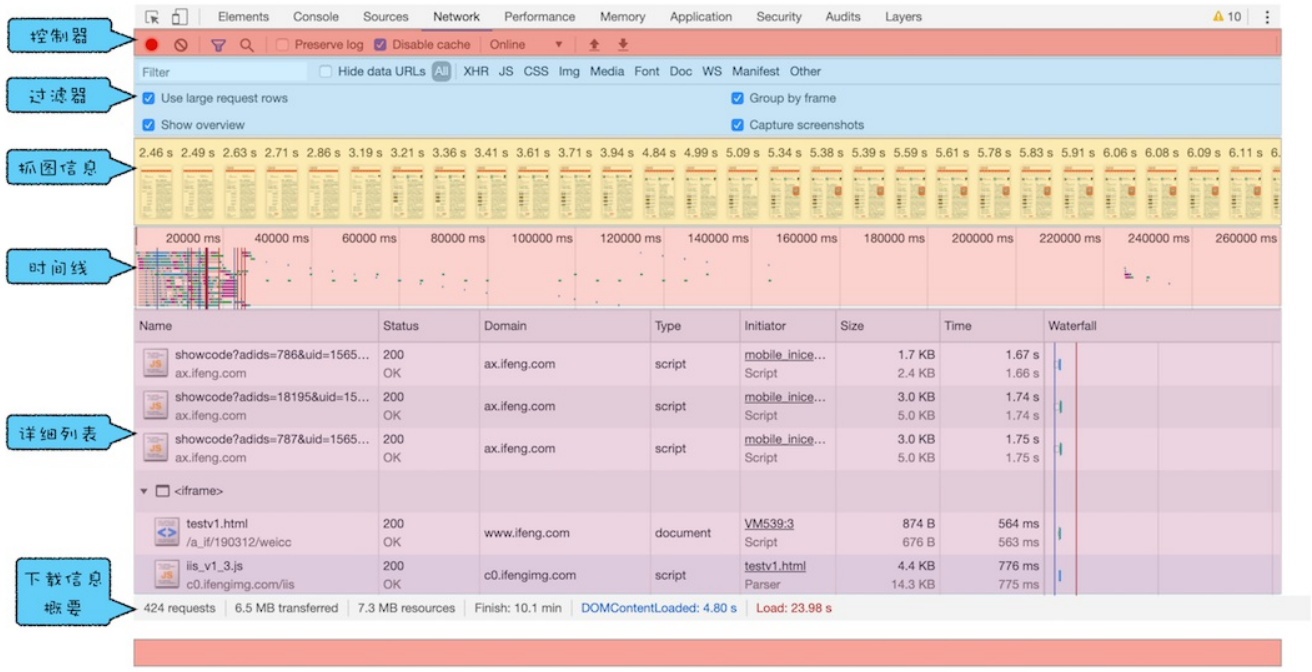
名称	描述
Elements面板	可以查看DOM结构、编辑CSS样式，用于测试页面布局和设计页面。
Console面板	可以看成是JavaScript Shell，能执行JavaScript脚本。通过Console还能和页面中的JavaScript对象交互。
Sources面板	1. 查看Web应用加载的所有文件； 2. 编辑CSS和JavaScript文件内容； 3. 将打乱的CSS文件或者JavaScript文件格式化； 4. 支持JavaScript的调试功能； 5. 设置工作区，将更改的文件保存到本地文件夹中。
NetWork（网络）面板	展示了页面中所有的请求内容列表，能查看每项请求的请求行、请求头、请求体、时间线以及网络请求瀑布图等信息。
Performance面板	记录和查看Web应用生命周期内的各种事件，并用来分析在执行过程中一些影响性能的要点。
Memory面板	用来查看运行过程中的JavaScript占用堆内存情况，追踪是否存在内存泄漏的情况等。
Application（应用）面板	查看Web应用的数据存储情况； PWA的基础数据：IndexedDB；Web SQL；本地和会话存储；Cookie；应用程序缓存；图像；字体和样式表等。
Security（安全）面板	显示当前页面一些基础的安全信息。
Audits面板	会对当前网页进行网络利用情况、网页性能方面的诊断，并给出一些优化建议。
Layers面板	展示一些渲染过程中分层的基础信息。

简单来说，Chrome开发者工具为我们提供了通过界面访问或者编辑DOM和CSSOM的能力，还提供了强大的调试功能和查看性能指标的能力。

OK，接下来我们就要重点看下其中重要的Network面板，即网络面板。

网络面板

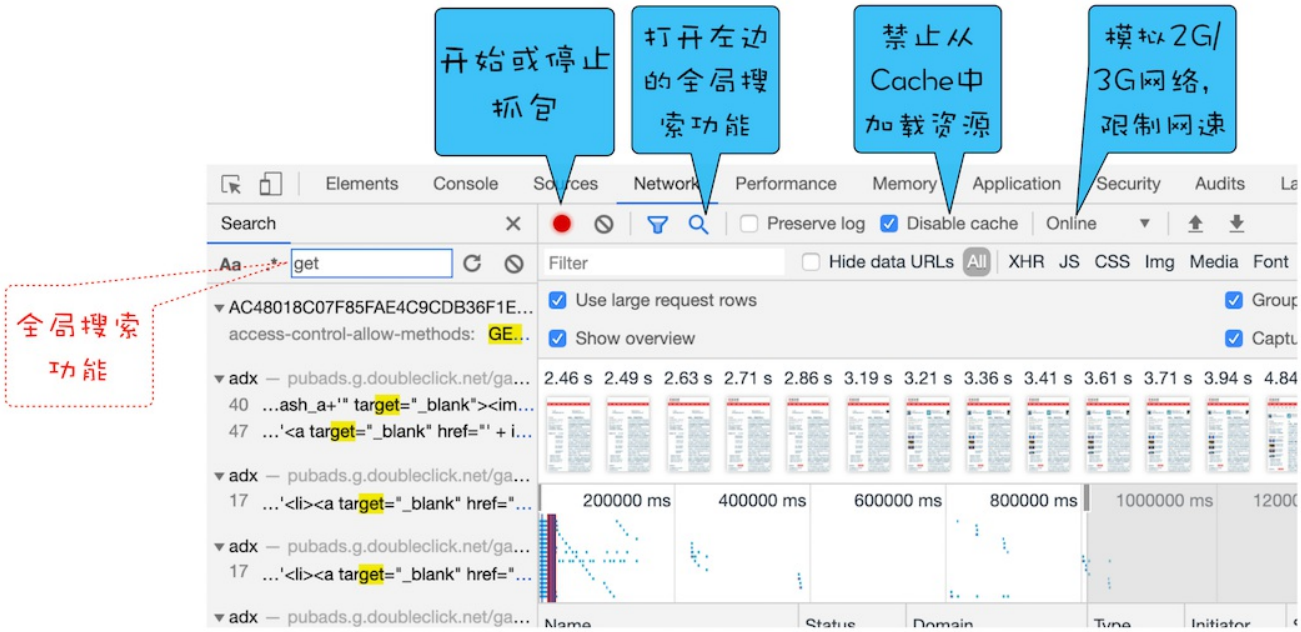
网络面板由控制器、过滤器、抓图信息、时间线、详细列表和下载信息概要这6个区域构成（如下图所示）。



网络面板概要图

1. 控制器

其中，控制器有4个比较重要的功能，我们按照下文中的这张图来简单介绍下。



控制器概要图

- 红色圆点的按钮，表示“开始/暂停抓包”，这个功能很常见，很容易理解。
- “全局搜索”按钮，这个功能就非常重要了，可以在所有下载资源中搜索相关内容，还可以快速定位到某几个你想要的文件上。
- Disable cache，即“禁止从Cache中加载资源”的功能，它在调试Web应用的时候非常有用，因为开启了Cache会影响到网络性能测试的结果。
- Online按钮，是“模拟2G/3G”功能，它可以限制带宽，模拟弱网情况下页面的展现情况，然后你就可以根据实际展示情况来动态调整策略，以便让Web应用更加适用于这些弱网。

2. 过滤器

网络面板中的过滤器，主要就是起过滤功能。因为有时候一个页面有太多内容在详细列表区域中展示了，而你可能只想查看JavaScript文件或者CSS文件，这时候就可以通过过滤器模块来筛选你想要的文件类型。

3. 抓图信息

抓图信息区域，可以用来分析用户等待页面加载时间内所看到的内容，分析用户实际的体验情况。比如，如果页面加载1秒多之后屏幕截图还是白屏状态，这时候就需要分析是网络还是代码的问题了。（勾选面板上的“Capture screenshots”即可启用屏幕截图。）

4. 时间线

时间线，主要用来展示HTTP、HTTPS、WebSocket加载的状态和时间的一个关系，用于直观感受页面的加载过程。如果是多条竖线堆叠在一起，那说明这些资源被同时被加载。至于具体到每个文件的加载信息，还需要用到下面要讲的详细列表。

5. 详细列表

这个区域是最重要的，它详细记录了每个资源从发起请求到完成请求这中间所有过程的状态，以及最终请求完成的数据信息。通过该列表，你就能很容易地去诊断一些网络问题。

详细列表是我们本篇文章介绍的重点，不过内容比较多，所以放到最后去专门介绍了。

6. 下载信息概要

下载信息概要中，你要重点关注下DOMContentLoaded和Load两个事件，以及这两个事件的完成时间。

- DOMContentLoaded，这个事件发生后，说明页面已经构建好DOM了，这意味着构建DOM所需要的HTML文件、JavaScript文件、CSS文件都已经下载完成了。
- Load，说明浏览器已经加载了所有的资源（图像、样式表等）。

通过下载信息概要面板，你可以查看触发这两个事件所花费的时间。

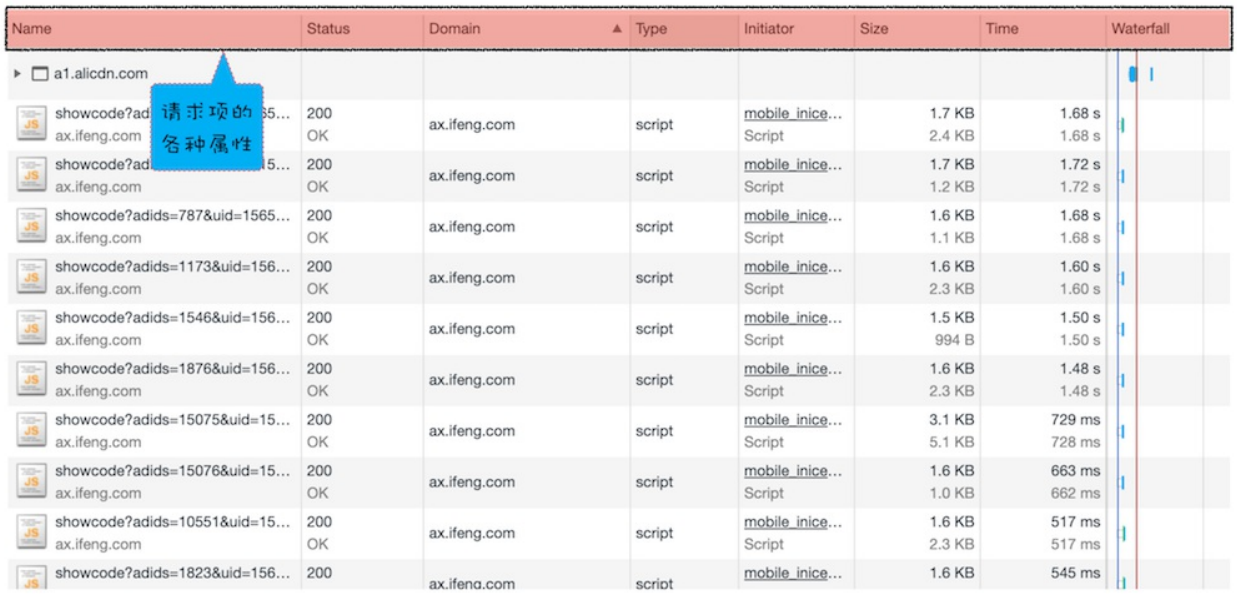
网络面板中的详细列表

下面我们就来重点介绍网络面板中的详细列表，这里面包含了大量有用的信息。

1. 列表的属性

列表的属性比较多，比如Name、Status、Type、Initiator等等，这个不难理解。当然，你还可以通过点击右键的下拉菜单来添加其他属性，这里我就不再赘述了，你可以自己上手实操一下。

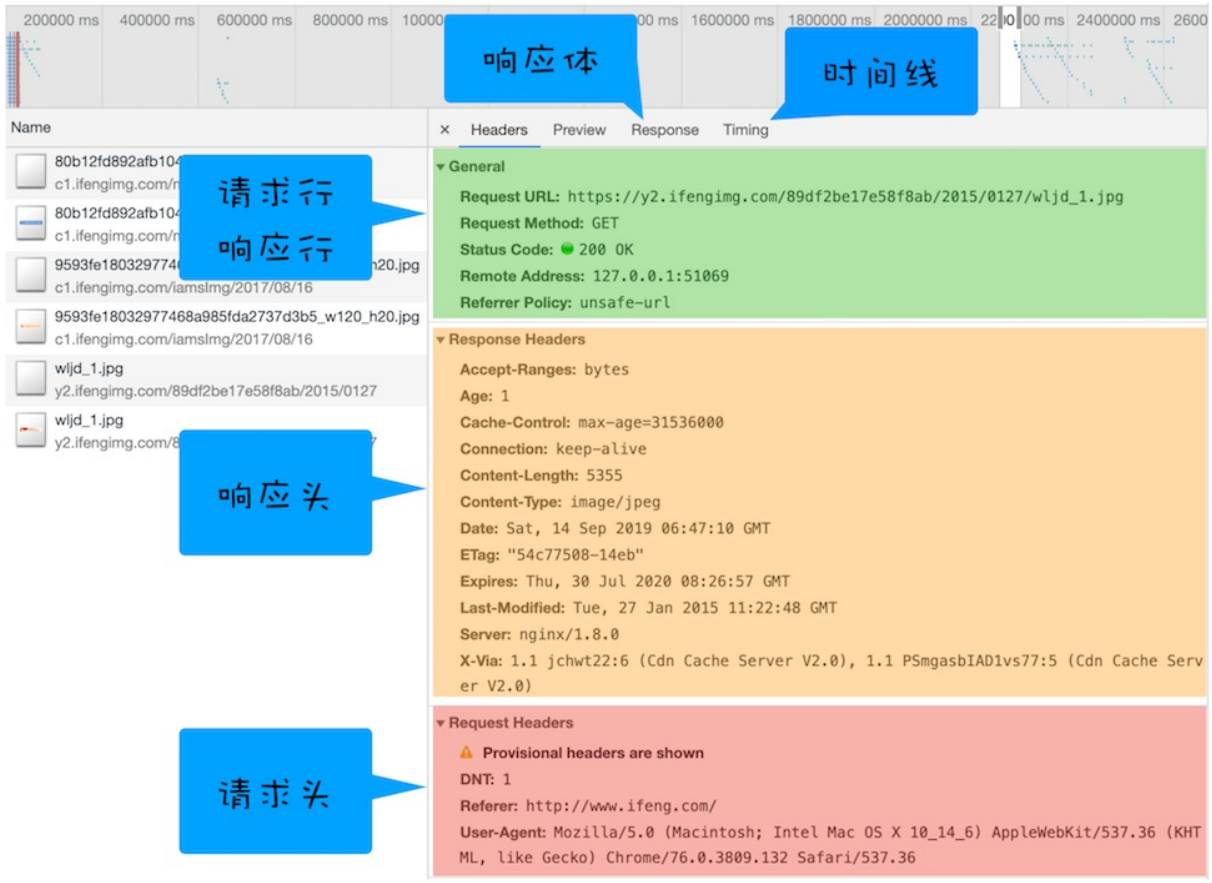
另外，你也可以按照列表的属性来给列表排序，默认情况下，列表是按请求发起的时间来排序的，最早发起请求的资源在顶部。当然也可以按照返回状态码、请求类型、请求时长、内容大小等基础属性排序，只需点击相应属性即可。



根据属性排序

2. 详细信息

如果你选中详细列表中的一项，右边就会出现该项的详细信息，如下所示：

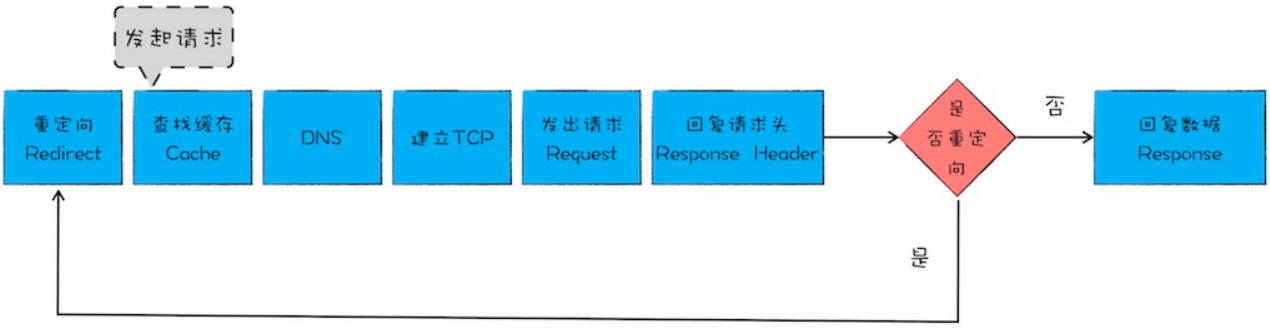


详细请求信息

你可以在此查看请求列表中任意一项的请求行和请求头信息，还可以查看响应行、响应头和响应体。然后你可以根据这些查看的信息来判断你的业务逻辑是否正确，或者有时候也可以用来逆向推导别人网站的业务逻辑。

3. 单个资源的时间线

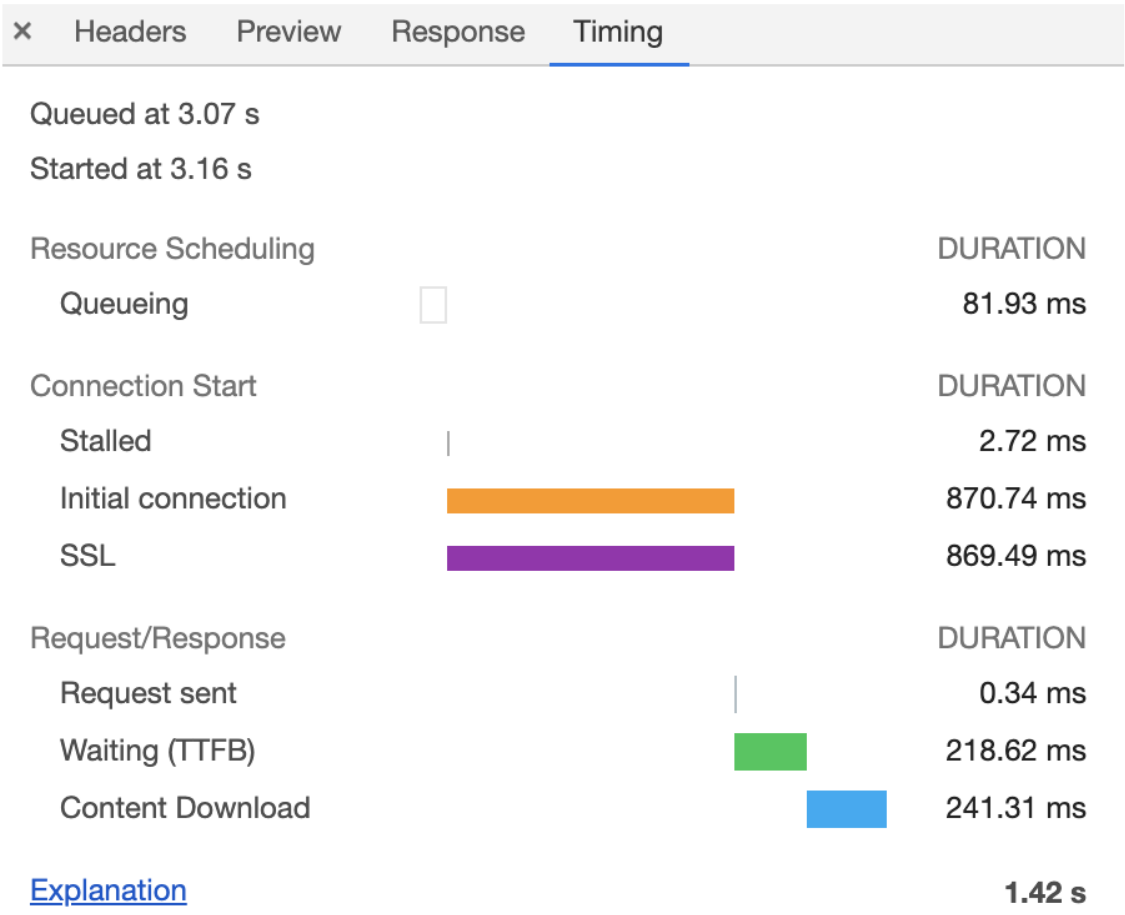
了解了每个资源的详细请求信息之后，我们再来分析单个资源请求时间线，这就涉及具体的HTTP请求流程了。



浏览器中HTTP请求流程

我们再回顾下在《03 | HTTP请求流程：为什么很多站点第二次打开速度会很快？》这篇文章，我们介绍过发起一个HTTP请求之后，浏览器首先查找缓存，如果缓存没有命中，那么继续发起DNS请求获取IP地址，然后利用IP地址和服务器端建立TCP连接，再发送HTTP请求，等待服务器响应；不过，如果服务器响应头中包含了重定向的信息，那么整个流程就需要重新再走一遍。这就是在浏览器中一个HTTP请求的基础流程。

那详细列表中是如何表示出这个流程的呢？这就要重点看下时间线面板了：



单个文件的时间线

那面板中这各项到底是什么含义呢？

第一个是Queuing，也就是排队的意思，当浏览器发起一个请求的时候，会有很多原因导致该请求不能被立即执行，而是需要排队等待。导致请求处于排队状态的原因有很多。

- 首先，页面中的资源是有优先级的，比如CSS、HTML、JavaScript等都是页面中的核心文件，所以优先级最高；而图片、视频、音频这类资源就不是核心资源，优先级就比较低。通常当后者遇到前者时，就需要“让路”，进入待排队状态。
- 其次，我们前面也提到过，浏览器会为每个域名最多维护6个TCP连接，如果发起一个HTTP请求时，这6个TCP连接都处于忙碌状态，那么这个请求就会处于排队状态。
- 最后，网络进程在为数据分配磁盘空间时，新的HTTP请求也需要短暂地等待磁盘分配结束。

等待排队完成之后，就要进入发起连接的状态了。不过在发起连接之前，还有一些原因可能导致连接过程被推迟，这个推迟就表现在面板中的**Stalled**上，它表示停滞的意思。

这里需要额外说明的是，如果你使用了代理服务器，还会增加一个**Proxy Negotiation**阶段，也就是代理协商阶段，它表示代理服务器连接协商所用的时间，不过在上图中没有体现出来，因为这里我们没有使用代理服务器。

接下来，就到了**Initial connection/SSL阶段**了，也就是和服务器建立连接的阶段，这包括了建立TCP连接所花费的时间；不过如果你使用了HTTPS协议，那么还需要一个额外的SSL握手时间，这个过程主要是用来协商一些加密信息的。（关于SSL协商的详细过程，我们会在Web安全模块中介绍。）

和服务器建立好连接之后，网络进程会准备请求数据，并将其发送给网络，这就是**Request sent阶段**。通常这个阶段非常快，因为只需要把浏览器缓冲区的数据发送出去就结束了，并不需要判断服务器是否接收到了，所以这个时间通常不到1毫秒。

数据发送出去了，接下来就是等待接收服务器第一个字节的数据，这个阶段称为Waiting (TTFB)，通常也称为“**第一字节时间**”。 TTFB是反映服务端响应速度的重要指标，对服务器来说，TTFB 时间越短，就说明服务器响应越快。

接收到第一个字节之后，进入陆续接收完整数据的阶段，也就是**Content Download阶段**，这意味着从第一字节时间到接收到全部响应数据所用的时间。

优化时间线上耗时项

了解了时间线面板上的各项含义之后，我们就可以根据这个请求的时间线来实现相关的优化操作了。

1. 排队（Queuing）时间过久

排队时间过久，大概率是由浏览器为每个域名最多维护6个连接导致的。那么基于这个原因，你就可以让1个站点下面的资源放在多个域名下面，比如放到3个域名下面，这样就可以同时支持18个连接了，这种方案称为**域名分片**技术。除了域名分片技术外，我个人还建议你**把站点升级到HTTP2**，因为HTTP2已经没有每个域名最多维护6个TCP连接的限制了。

2. 第一字节时间（TTFB）时间过久

这可能的原因有如下：

- **服务器生成页面数据的时间过久**。对于动态网页来说，服务器收到用户打开一个页面的请求时，首先要从数据库中读取该页面需要的数据，然后把这些数据传入到模板中，模板渲染后，再返回给用户。服务器在处理这个数据的过程中，可能某个环节会出问题。
- **网络的原因**。比如使用了低带宽的服务器，或者本来用的是电信的服务器，可联通的网络用户要来访问你的服务器，这样也会拖慢网速。
- **发送请求头时带上了多余的用户信息**。比如一些不必要的Cookie信息，服务器接收到这些Cookie信息之后可能需要对每一项都做处理，这样就加大了服务器的处理时长。

对于这三种问题，你要有针对性地出一些解决方案。面对第一种服务器的问题，你可以想办法去提高服务器的处理速度，比如通过增加各种缓存的技术；针对第二种网络问题，你可以使用CDN来缓存一些静态文件；至于第三种，你在发送请求时就去尽可能地减少一些不必要的Cookie数据信息。

3. Content Download时间过久

如果单个请求的Content Download花费了大量时间，有可能是字节数太多的原因导致的。这时候你就需要减少文件大小，比如压缩、去掉源码中不必要的注释等方法。

总结

好了，今天就介绍到这里了，下面我来总结下今天的内容。

首先我们简单介绍了Chrome开发者工具10个基础的面板信息；然后重点剖析了网络面板，再结合之前介绍的网络请求流程来重点分析了网络面板中时间线的各个指标的含义；最后我们还简要分析了时间线中各项指标出现异常的可能原因，并给出了一些优化方案。

其实通过今天的分析，我们可以得出这样一个结论：如果你要去做一些实践性的项目优化，理解其背后的理论至关重要。因为理论就是一条“线”，它会把各种实践的内容“串”在一起，然后你可以围绕着这条“线”来排查问题。

思考时间

今天我们介绍了网络面板，还有一个非常重要的Performance面板我们没有介绍，不过你可以去网上查找一些相关的资料。

所以今天留给你的是一道实际操作的题目，你可以结合网络面板和Performance面板来分析一个Web应用的性能瓶颈（比如https://www.12306.cn ）。

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

“浏览器中的页面循环系统”模块我们已经介绍完了，循环系统是页面的基础，理解了循环系统能让我们从本质上更好地理解页面的工作方式，加深我们对一些前端概念的理解。

接下来我们就要进入新的模块了，也就是“浏览器中的页面”模块，正如专栏简介中所言，页面是浏览器的核心，浏览器中的所有功能点都是服务于页面的，而Chrome开发者工具又是工程师调试页面的核心工具，所以在这个模块的开篇，我想先带你来深入了解下Chrome开发者工具。

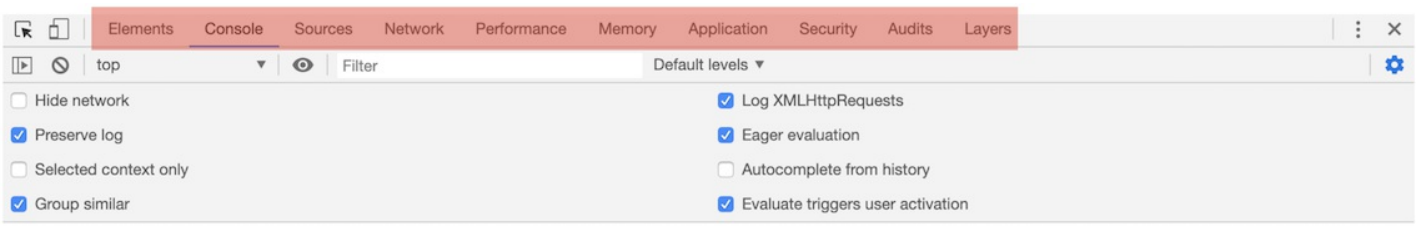
Chrome开发者工具（简称DevTools）是一组网页制作和调试的工具，内嵌于Google Chrome 浏览器中。Chrome开发者工具非常重要，所蕴含的内容也是非常多的，熟练使用它能让你更加深入地了解浏览器内部工作原理。（Chrome开发者工具也在不停地迭代改进，如果你想使用最新版本，可以使用Chrome Canary。）

作为这一模块的第一篇文章，我们主要聚焦**页面的源头**和**网络数据的接收**，这些发送和接收的数据都能体现在开发者工具的网络面板上。不过为了你能更好地理解和掌握，我们会先对Chrome开发者工具做一个大致的介绍，然后再深入剖析网络面板。

Chrome开发者工具

Chrome开发者工具有很多重要的面板，比如与性能相关的有网络面板、Performance面板、内存面板等，与调试页面相关的有Elements面板、Sources面板、Console面板等。

你可以在浏览器窗口的右上方选择Chrome菜单，然后选择“更多工具–>开发者工具”来打开Chrome开发者工具。打开的页面如下图所示：



从图中可以看出，它一共包含了10个功能面板，包括了Elements、Console、Sources、NetWork、Performance、Memory、Application、Security、Audits和Layers。

关于这10个面板的大致功能，我做了一个表格，感兴趣的话，你可以详细看下：

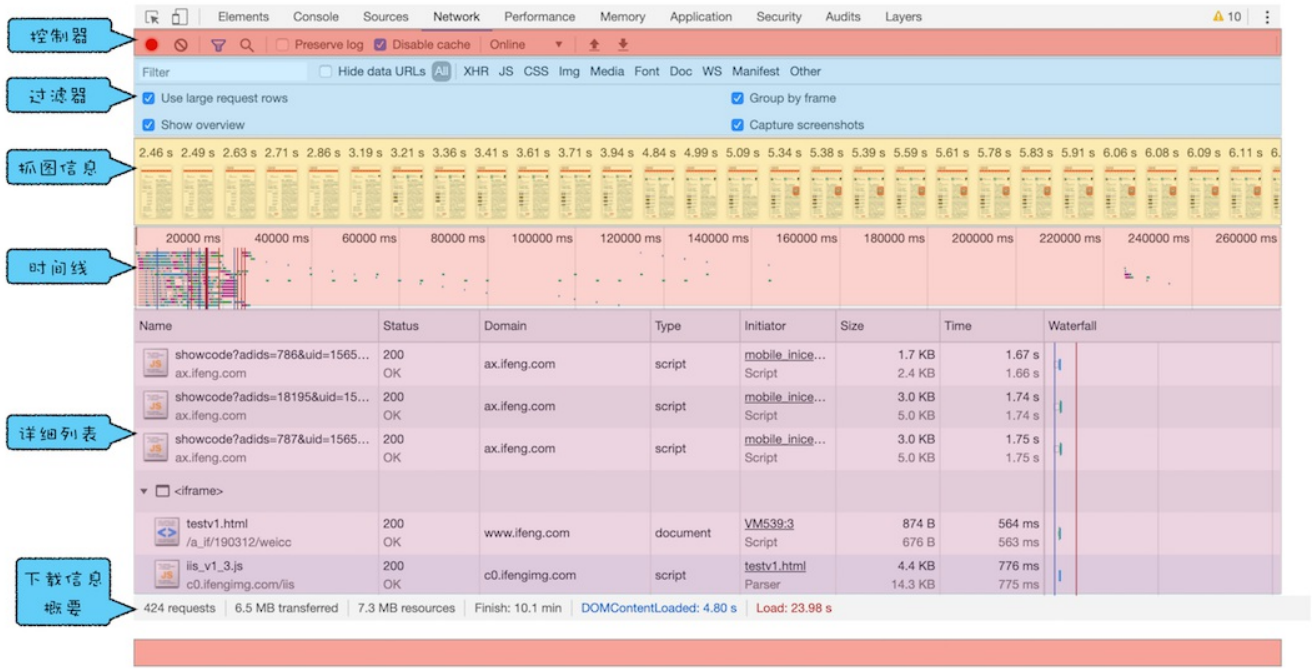
名称	描述
Elements面板	可以查看DOM结构、编辑CSS样式，用于测试页面布局和设计页面。
Console面板	可以看成是JavaScript Shell，能执行JavaScript脚本。通过Console还能和页面中的JavaScript对象交互。
Sources面板	1. 查看Web应用加载的所有文件； 2. 编辑CSS和JavaScript文件内容； 3. 将打乱的CSS文件或者JavaScript文件格式化； 4. 支持JavaScript的调试功能； 5. 设置工作区，将更改的文件保存到本地文件夹中。
NetWork（网络）面板	展示了页面中所有的请求内容列表，能查看每项请求的请求行、请求头、请求体、时间线以及网络请求瀑布图等信息。
Performance面板	记录和查看Web应用生命周期内的各种事件，并用来分析在执行过程中一些影响性能的要点。
Memory面板	用来查看运行过程中的JavaScript占用堆内存情况，追踪是否存在内存泄漏的情况等。
Application（应用）面板	查看Web应用的数据存储情况； PWA的基础数据：IndexedDB；Web SQL；本地和会话存储；Cookie；应用程序缓存；图像；字体和样式表等。
Security（安全）面板	显示当前页面一些基础的安全信息。
Audits面板	会对当前网页进行网络利用情况、网页性能方面的诊断，并给出一些优化建议。
Layers面板	展示一些渲染过程中分层的基础信息。

简单来说，Chrome开发者工具为我们提供了通过界面访问或者编辑DOM和CSSOM的能力，还提供了强大的调试功能和查看性能指标的能力。

OK，接下来我们就要重点看下其中重要的Network面板，即网络面板。

网络面板

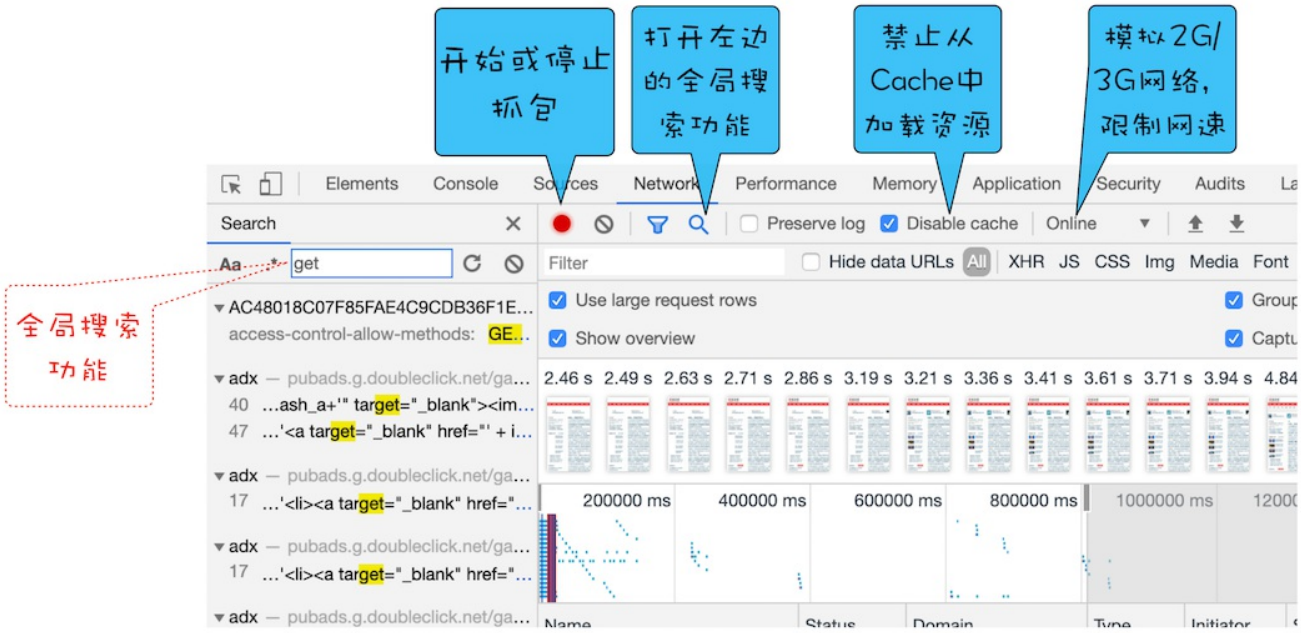
网络面板由控制器、过滤器、抓图信息、时间线、详细列表和下载信息概要这6个区域构成（如下图所示）。



网络面板概要图

1. 控制器

其中，控制器有4个比较重要的功能，我们按照下文中的这张图来简单介绍下。



控制器概要图

- 红色圆点的按钮，表示“开始/暂停抓包”，这个功能很常见，很容易理解。
- “全局搜索”按钮，这个功能就非常重要了，可以在所有下载资源中搜索相关内容，还可以快速定位到某几个你想要的文件上。
- Disable cache，即“禁止从Cache中加载资源”的功能，它在调试Web应用的时候非常有用，因为开启了Cache会影响到网络性能测试的结果。
- Online按钮，是“模拟2G/3G”功能，它可以限制带宽，模拟弱网情况下页面的展现情况，然后你就可以根据实际展示情况来动态调整策略，以便让Web应用更加适用于这些弱网。

2. 过滤器

网络面板中的过滤器，主要就是起过滤功能。因为有时候一个页面有太多内容在详细列表区域中展示了，而你可能只想查看JavaScript文件或者CSS文件，这时候就可以通过过滤器模块来筛选你想要的文件类型。

3. 抓图信息

抓图信息区域，可以用来分析用户等待页面加载时间内所看到的内容，分析用户实际的体验情况。比如，如果页面加载1秒多之后屏幕截图还是白屏状态，这时候就需要分析是网络还是代码的问题了。（勾选面板上的“Capture screenshots”即可启用屏幕截图。）

4. 时间线

时间线，主要用来展示HTTP、HTTPS、WebSocket加载的状态和时间的一个关系，用于直观感受页面的加载过程。如果是多条竖线堆叠在一起，那说明这些资源被同时被加载。至于具体到每个文件的加载信息，还需要用到下面要讲的详细列表。

5. 详细列表

这个区域是最重要的，它详细记录了每个资源从发起请求到完成请求这中间所有过程的状态，以及最终请求完成的数据信息。通过该列表，你就能很容易地去诊断一些网络问题。

详细列表是我们本篇文章介绍的重点，不过内容比较多，所以放到最后去专门介绍了。

6. 下载信息概要

下载信息概要中，你要重点关注下DOMContentLoaded和Load两个事件，以及这两个事件的完成时间。

- DOMContentLoaded，这个事件发生后，说明页面已经构建好DOM了，这意味着构建DOM所需要的HTML文件、JavaScript文件、CSS文件都已经下载完成了。
- Load，说明浏览器已经加载了所有的资源（图像、样式表等）。

通过下载信息概要面板，你可以查看触发这两个事件所花费的时间。

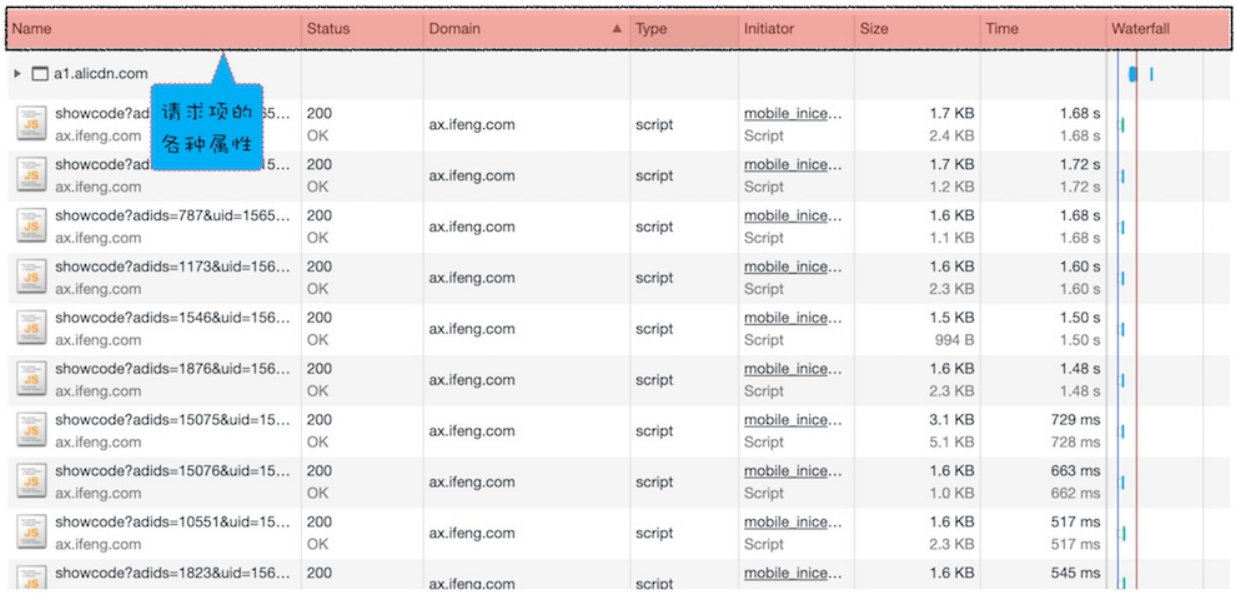
网络面板中的详细列表

下面我们就来重点介绍网络面板中的详细列表，这里面包含了大量有用的信息。

1. 列表的属性

列表的属性比较多，比如Name、Status、Type、Initiator等等，这个不难理解。当然，你还可以通过点击右键的下拉菜单来添加其他属性，这里我就不再赘述了，你可以自己上手实操一下。

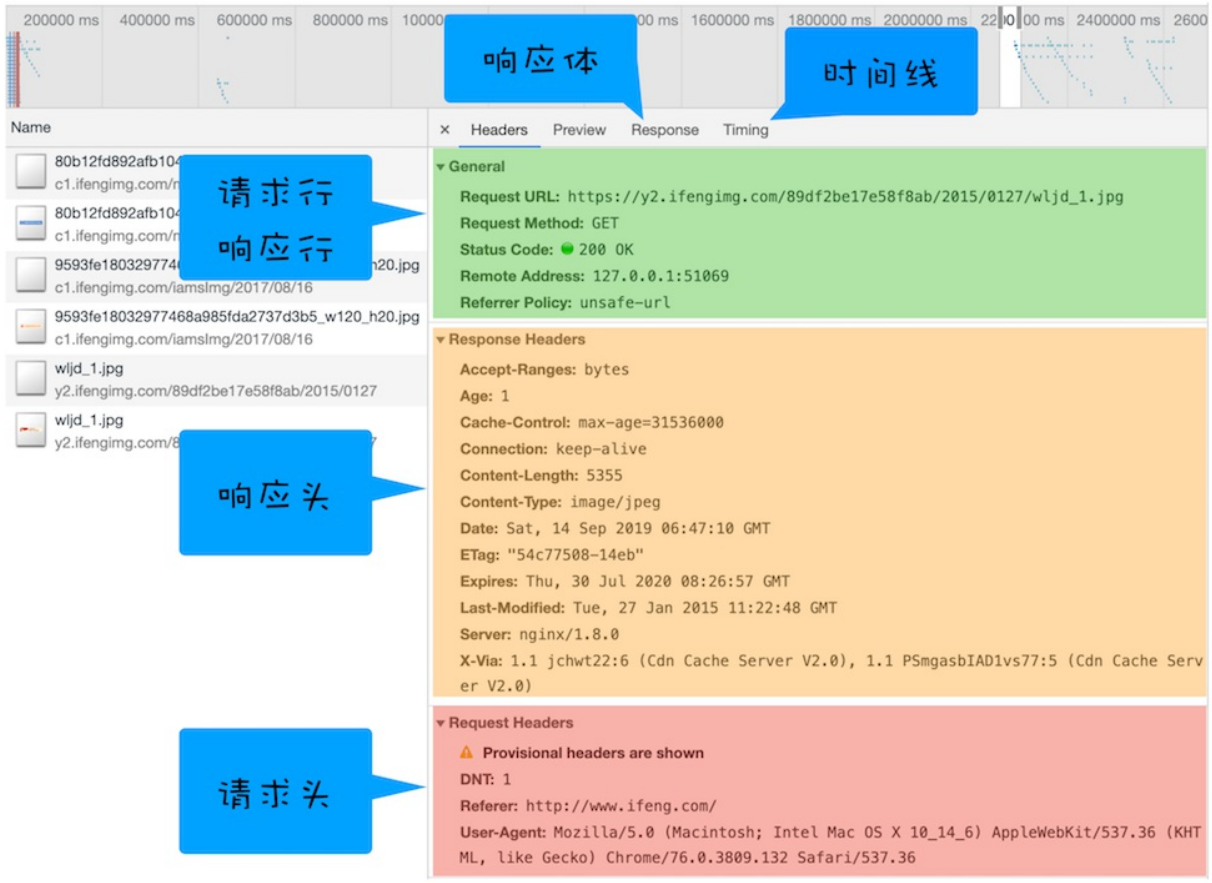
另外，你也可以按照列表的属性来给列表排序，默认情况下，列表是按请求发起的时间来排序的，最早发起请求的资源在顶部。当然也可以按照返回状态码、请求类型、请求时长、内容大小等基础属性排序，只需点击相应属性即可。



根据属性排序

2. 详细信息

如果你选中详细列表中的一项，右边就会出现该项的详细信息，如下所示：

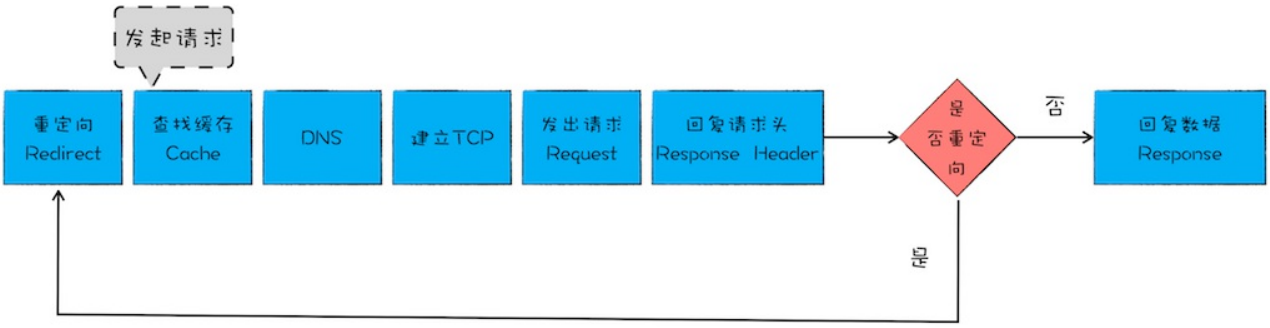


详细请求信息

你可以在此查看请求列表中任意一项的请求行和请求头信息，还可以查看响应行、响应头和响应体。然后你可以根据这些查看的信息来判断你的业务逻辑是否正确，或者有时候也可以用来逆向推导别人网站的业务逻辑。

3. 单个资源的时间线

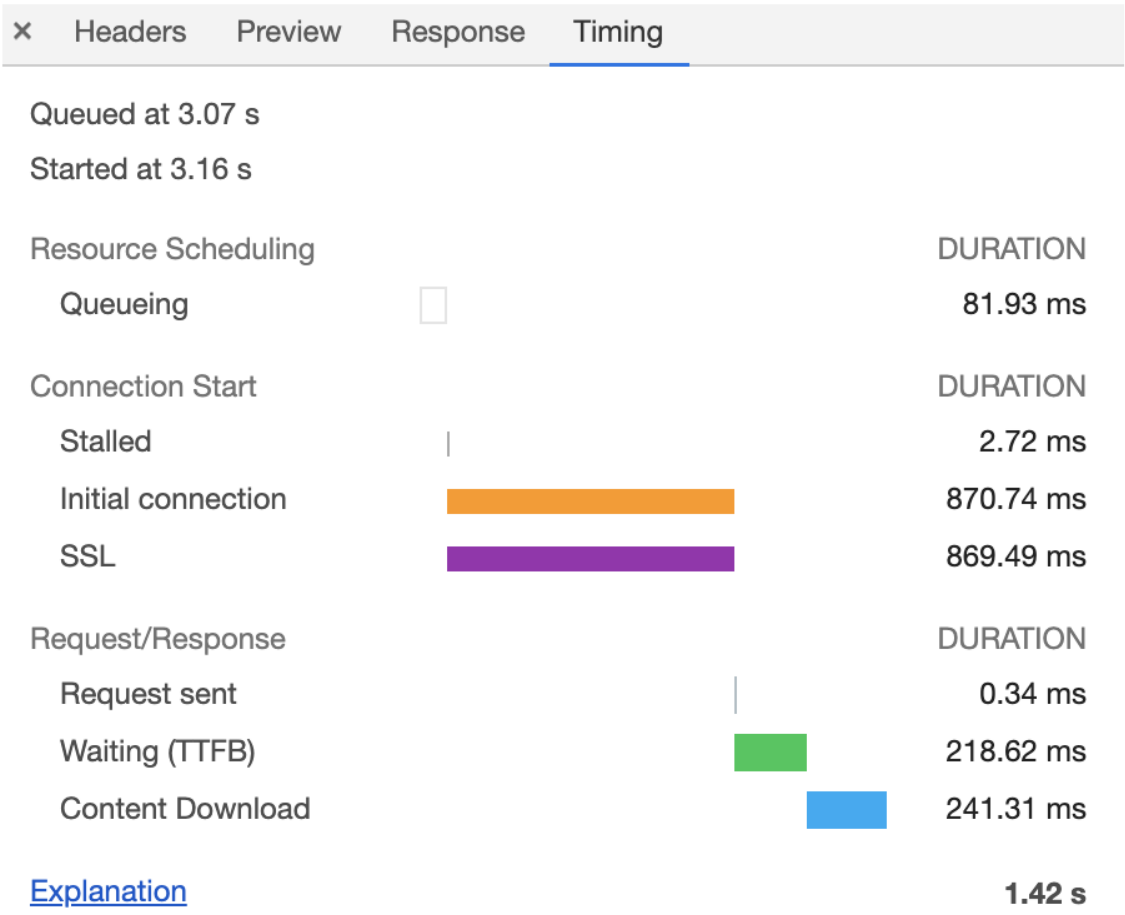
了解了每个资源的详细请求信息之后，我们再来分析单个资源请求时间线，这就涉及具体的HTTP请求流程了。



浏览器中HTTP请求流程

我们再回顾下在《03 | HTTP请求流程：为什么很多站点第二次打开速度会很快？》这篇文章，我们介绍过发起一个HTTP请求之后，浏览器首先查找缓存，如果缓存没有命中，那么继续发起DNS请求获取IP地址，然后利用IP地址和服务器端建立TCP连接，再发送HTTP请求，等待服务器响应；不过，如果服务器响应头中包含了重定向的信息，那么整个流程就需要重新再走一遍。这就是在浏览器中一个HTTP请求的基础流程。

那详细列表中是如何表示出这个流程的呢？这就要重点看下时间线面板了：



单个文件的时间线

那面板中这各项到底是什么含义呢？

第一个是Queuing，也就是排队的意思，当浏览器发起一个请求的时候，会有很多原因导致该请求不能被立即执行，而是需要排队等待。导致请求处于排队状态的原因有很多。

- 首先，页面中的资源是有优先级的，比如CSS、HTML、JavaScript等都是页面中的核心文件，所以优先级最高；而图片、视频、音频这类资源就不是核心资源，优先级就比较低。通常当后者遇到前者时，就需要“让路”，进入待排队状态。
- 其次，我们前面也提到过，浏览器会为每个域名最多维护6个TCP连接，如果发起一个HTTP请求时，这6个TCP连接都处于忙碌状态，那么这个请求就会处于排队状态。
- 最后，网络进程在为数据分配磁盘空间时，新的HTTP请求也需要短暂地等待磁盘分配结束。

等待排队完成之后，就要进入发起连接的状态了。不过在发起连接之前，还有一些原因可能导致连接过程被推迟，这个推迟就表现在面板中的**Stalled**上，它表示停滞的意思。

这里需要额外说明的是，如果你使用了代理服务器，还会增加一个**Proxy Negotiation**阶段，也就是代理协商阶段，它表示代理服务器连接协商所用的时间，不过在上图中没有体现出来，因为这里我们没有使用代理服务器。

接下来，就到了**Initial connection/SSL阶段**了，也就是和服务器建立连接的阶段，这包括了建立TCP连接所花费的时间；不过如果你使用了HTTPS协议，那么还需要一个额外的SSL握手时间，这个过程主要是用来协商一些加密信息的。（关于SSL协商的详细过程，我们会在Web安全模块中介绍。）

和服务器建立好连接之后，网络进程会准备请求数据，并将其发送给网络，这就是**Request sent阶段**。通常这个阶段非常快，因为只需要把浏览器缓冲区的数据发送出去就结束了，并不需要判断服务器是否接收到了，所以这个时间通常不到1毫秒。

数据发送出去了，接下来就是等待接收服务器第一个字节的数据，这个阶段称为Waiting (TTFB)，通常也称为“**第一字节时间**”。 TTFB是反映服务端响应速度的重要指标，对服务器来说，TTFB 时间越短，就说明服务器响应越快。

接收到第一个字节之后，进入陆续接收完整数据的阶段，也就是**Content Download阶段**，这意味着从第一字节时间到接收到全部响应数据所用的时间。

优化时间线上耗时项

了解了时间线面板上的各项含义之后，我们就可以根据这个请求的时间线来实现相关的优化操作了。

1. 排队（Queuing）时间过久

排队时间过久，大概率是由浏览器为每个域名最多维护6个连接导致的。那么基于这个原因，你就可以让1个站点下面的资源放在多个域名下面，比如放到3个域名下面，这样就可以同时支持18个连接了，这种方案称为**域名分片**技术。除了域名分片技术外，我个人还建议你**把站点升级到HTTP2**，因为HTTP2已经没有每个域名最多维护6个TCP连接的限制了。

2. 第一字节时间（TTFB）时间过久

这可能的原因有如下：

- **服务器生成页面数据的时间过久**。对于动态网页来说，服务器收到用户打开一个页面的请求时，首先要从数据库中读取该页面需要的数据，然后把这些数据传入到模板中，模板渲染后，再返回给用户。服务器在处理这个数据的过程中，可能某个环节会出问题。
- **网络的原因**。比如使用了低带宽的服务器，或者本来用的是电信的服务器，可联通的网络用户要来访问你的服务器，这样也会拖慢网速。
- **发送请求头时带上了多余的用户信息**。比如一些不必要的Cookie信息，服务器接收到这些Cookie信息之后可能需要对每一项都做处理，这样就加大了服务器的处理时长。

对于这三种问题，你要有针对性地出一些解决方案。面对第一种服务器的问题，你可以想办法去提高服务器的处理速度，比如通过增加各种缓存的技术；针对第二种网络问题，你可以使用CDN来缓存一些静态文件；至于第三种，你在发送请求时就去尽可能地减少一些不必要的Cookie数据信息。

3. Content Download时间过久

如果单个请求的Content Download花费了大量时间，有可能是字节数太多的原因导致的。这时候你就需要减少文件大小，比如压缩、去掉源码中不必要的注释等方法。

总结

好了，今天就介绍到这里了，下面我来总结下今天的内容。

首先我们简单介绍了Chrome开发者工具10个基础的面板信息；然后重点剖析了网络面板，再结合之前介绍的网络请求流程来重点分析了网络面板中时间线的各个指标的含义；最后我们还简要分析了时间线中各项指标出现异常的可能原因，并给出了一些优化方案。

其实通过今天的分析，我们可以得出这样一个结论：如果你要去做一些实践性的项目优化，理解其背后的理论至关重要。因为理论就是一条“线”，它会把各种实践的内容“串”在一起，然后你可以围绕着这条“线”来排查问题。

思考时间

今天我们介绍了网络面板，还有一个非常重要的Performance面板我们没有介绍，不过你可以去网上查找一些相关的资料。

所以今天留给你的是一道实际操作的题目，你可以结合网络面板和Performance面板来分析一个Web应用的性能瓶颈（比如https://www.12306.cn ）。

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

“浏览器中的页面循环系统”模块我们已经介绍完了，循环系统是页面的基础，理解了循环系统能让我们从本质上更好地理解页面的工作方式，加深我们对一些前端概念的理解。

接下来我们就要进入新的模块了，也就是“浏览器中的页面”模块，正如专栏简介中所言，页面是浏览器的核心，浏览器中的所有功能点都是服务于页面的，而Chrome开发者工具又是工程师调试页面的核心工具，所以在这个模块的开篇，我想先带你来深入了解下Chrome开发者工具。

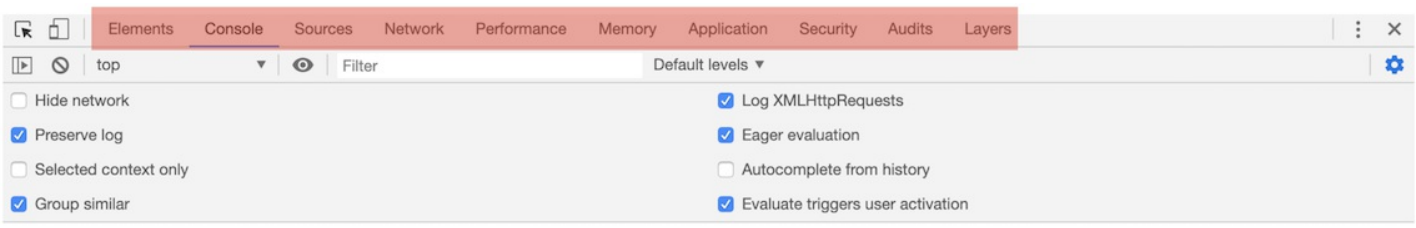
Chrome开发者工具（简称DevTools）是一组网页制作和调试的工具，内嵌于Google Chrome 浏览器中。Chrome开发者工具非常重要，所蕴含的内容也是非常多的，熟练使用它能让你更加深入地了解浏览器内部工作原理。（Chrome开发者工具也在不停地迭代改进，如果你想使用最新版本，可以使用Chrome Canary。）

作为这一模块的第一篇文章，我们主要聚焦**页面的源头**和**网络数据的接收**，这些发送和接收的数据都能体现在开发者工具的网络面板上。不过为了你能更好地理解和掌握，我们会先对Chrome开发者工具做一个大致的介绍，然后再深入剖析网络面板。

Chrome开发者工具

Chrome开发者工具有很多重要的面板，比如与性能相关的有网络面板、Performance面板、内存面板等，与调试页面相关的有Elements面板、Sources面板、Console面板等。

你可以在浏览器窗口的右上方选择Chrome菜单，然后选择“更多工具–>开发者工具”来打开Chrome开发者工具。打开的页面如下图所示：



从图中可以看出，它一共包含了10个功能面板，包括了Elements、Console、Sources、NetWork、Performance、Memory、Application、Security、Audits和Layers。

关于这10个面板的大致功能，我做了一个表格，感兴趣的话，你可以详细看下：

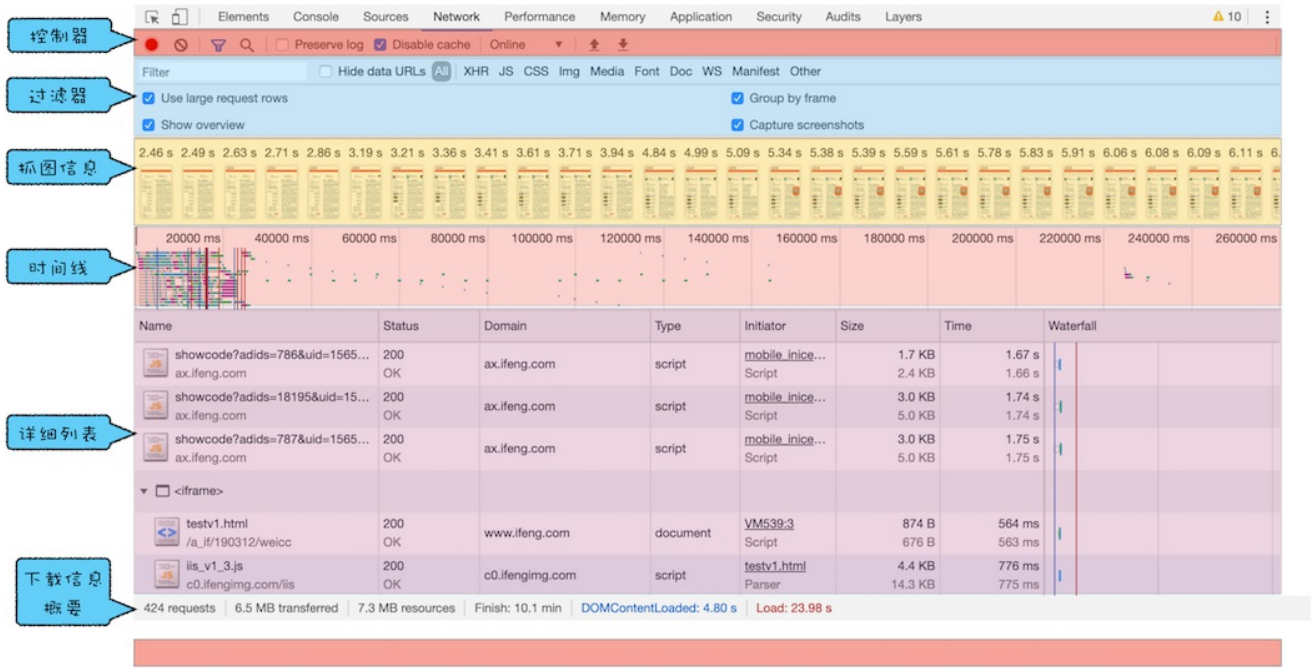
名称	描述
Elements面板	可以查看DOM结构、编辑CSS样式，用于测试页面布局和设计页面。
Console面板	可以看成是JavaScript Shell，能执行JavaScript脚本。通过Console还能和页面中的JavaScript对象交互。
Sources面板	1. 查看Web应用加载的所有文件； 2. 编辑CSS和JavaScript文件内容； 3. 将打乱的CSS文件或者JavaScript文件格式化； 4. 支持JavaScript的调试功能； 5. 设置工作区，将更改的文件保存到本地文件夹中。
NetWork（网络）面板	展示了页面中所有的请求内容列表，能查看每项请求的请求行、请求头、请求体、时间线以及网络请求瀑布图等信息。
Performance面板	记录和查看Web应用生命周期内的各种事件，并用来分析在执行过程中一些影响性能的要点。
Memory面板	用来查看运行过程中的JavaScript占用堆内存情况，追踪是否存在内存泄漏的情况等。
Application（应用）面板	查看Web应用的数据存储情况； PWA的基础数据：IndexedDB；Web SQL；本地和会话存储；Cookie；应用程序缓存；图像；字体和样式表等。
Security（安全）面板	显示当前页面一些基础的安全信息。
Audits面板	会对当前网页进行网络利用情况、网页性能方面的诊断，并给出一些优化建议。
Layers面板	展示一些渲染过程中分层的基础信息。

简单来说，Chrome开发者工具为我们提供了通过界面访问或者编辑DOM和CSSOM的能力，还提供了强大的调试功能和查看性能指标的能力。

OK，接下来我们就要重点看下其中重要的Network面板，即网络面板。

网络面板

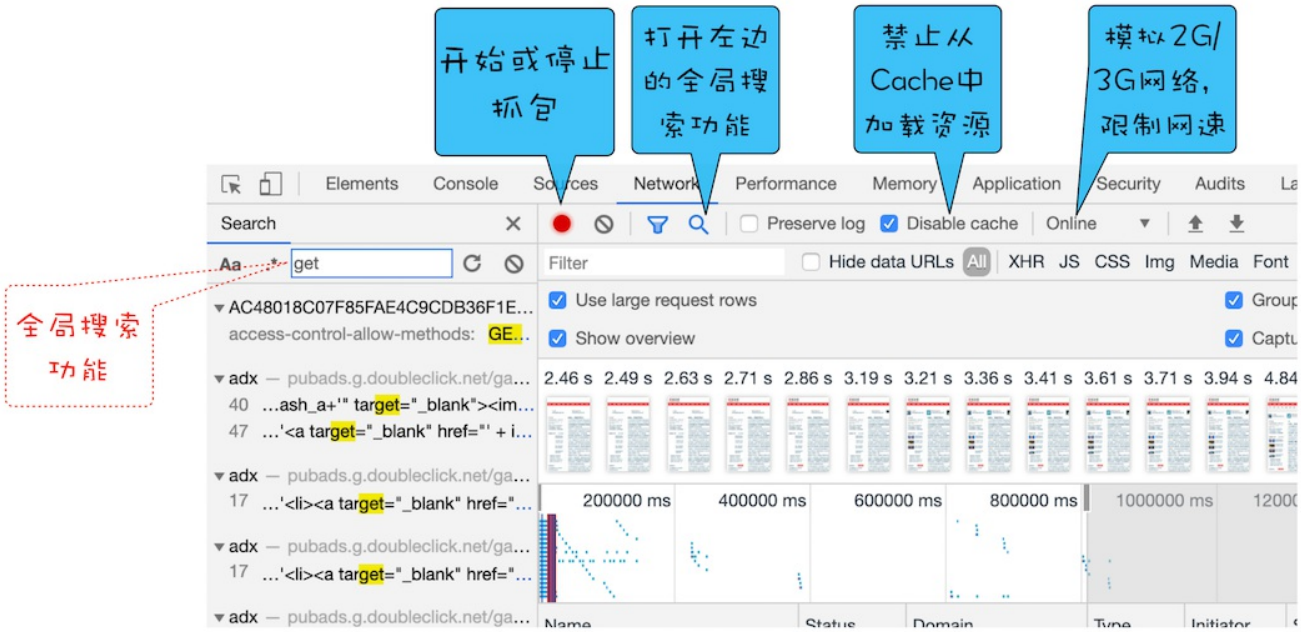
网络面板由控制器、过滤器、抓图信息、时间线、详细列表和下载信息概要这6个区域构成（如下图所示）。



网络面板概要图

1. 控制器

其中，控制器有4个比较重要的功能，我们按照下文中的这张图来简单介绍下。



控制器概要图

- 红色圆点的按钮，表示“开始/暂停抓包”，这个功能很常见，很容易理解。
- “全局搜索”按钮，这个功能就非常重要了，可以在所有下载资源中搜索相关内容，还可以快速定位到某几个你想要的文件上。
- Disable cache，即“禁止从Cache中加载资源”的功能，它在调试Web应用的时候非常有用，因为开启了Cache会影响到网络性能测试的结果。
- Online按钮，是“模拟2G/3G”功能，它可以限制带宽，模拟弱网情况下页面的展现情况，然后你就可以根据实际展示情况来动态调整策略，以便让Web应用更加适用于这些弱网。

2. 过滤器

网络面板中的过滤器，主要就是起过滤功能。因为有时候一个页面有太多内容在详细列表区域中展示了，而你可能只想查看JavaScript文件或者CSS文件，这时候就可以通过过滤器模块来筛选你想要的文件类型。

3. 抓图信息

抓图信息区域，可以用来分析用户等待页面加载时间内所看到的内容，分析用户实际的体验情况。比如，如果页面加载1秒多之后屏幕截图还是白屏状态，这时候就需要分析是网络还是代码的问题了。（勾选面板上的“Capture screenshots”即可启用屏幕截图。）

4. 时间线

时间线，主要用来展示HTTP、HTTPS、WebSocket加载的状态和时间的一个关系，用于直观感受页面的加载过程。如果是多条竖线堆叠在一起，那说明这些资源被同时被加载。至于具体到每个文件的加载信息，还需要用到下面要讲的详细列表。

5. 详细列表

这个区域是最重要的，它详细记录了每个资源从发起请求到完成请求这中间所有过程的状态，以及最终请求完成的数据信息。通过该列表，你就能很容易地去诊断一些网络问题。

详细列表是我们本篇文章介绍的重点，不过内容比较多，所以放到最后去专门介绍了。

6. 下载信息概要

下载信息概要中，你要重点关注下DOMContentLoaded和Load两个事件，以及这两个事件的完成时间。

- DOMContentLoaded，这个事件发生后，说明页面已经构建好DOM了，这意味着构建DOM所需要的HTML文件、JavaScript文件、CSS文件都已经下载完成了。
- Load，说明浏览器已经加载了所有的资源（图像、样式表等）。

通过下载信息概要面板，你可以查看触发这两个事件所花费的时间。

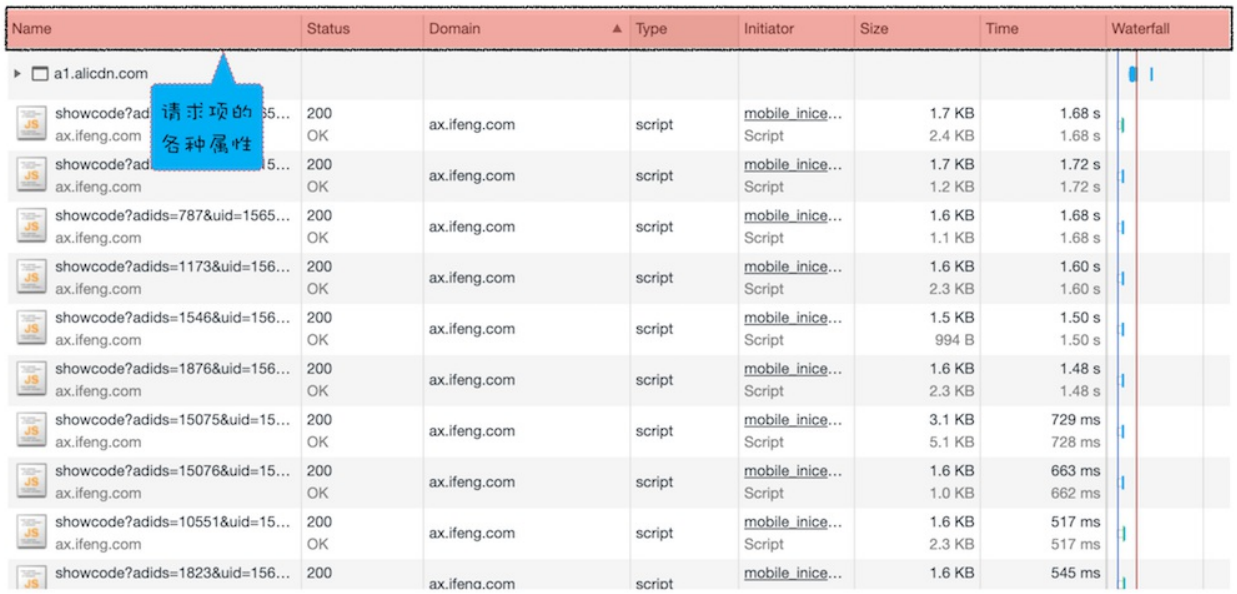
网络面板中的详细列表

下面我们就来重点介绍网络面板中的详细列表，这里面包含了大量有用的信息。

1. 列表的属性

列表的属性比较多，比如Name、Status、Type、Initiator等等，这个不难理解。当然，你还可以通过点击右键的下拉菜单来添加其他属性，这里我就不再赘述了，你可以自己上手实操一下。

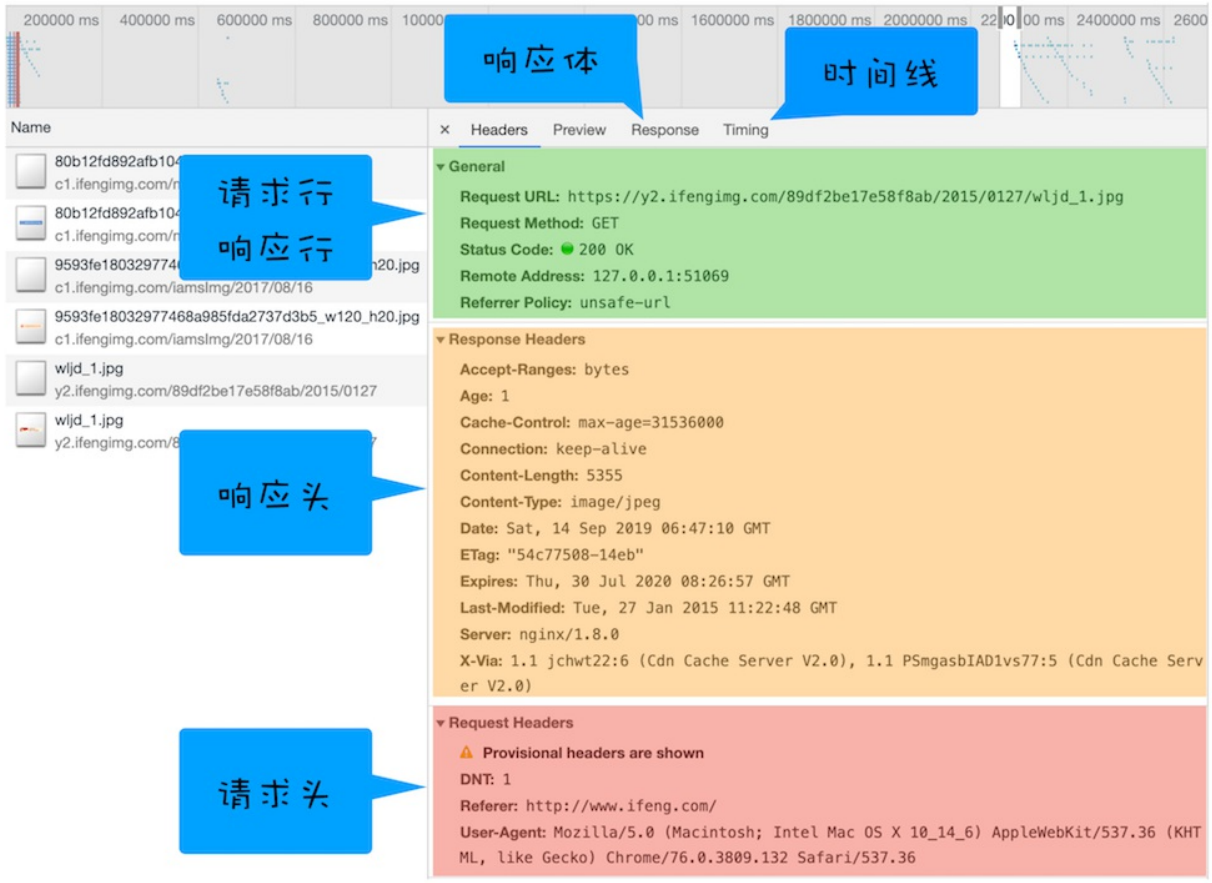
另外，你也可以按照列表的属性来给列表排序，默认情况下，列表是按请求发起的时间来排序的，最早发起请求的资源在顶部。当然也可以按照返回状态码、请求类型、请求时长、内容大小等基础属性排序，只需点击相应属性即可。



根据属性排序

2. 详细信息

如果你选中详细列表中的一项，右边就会出现该项的详细信息，如下所示：

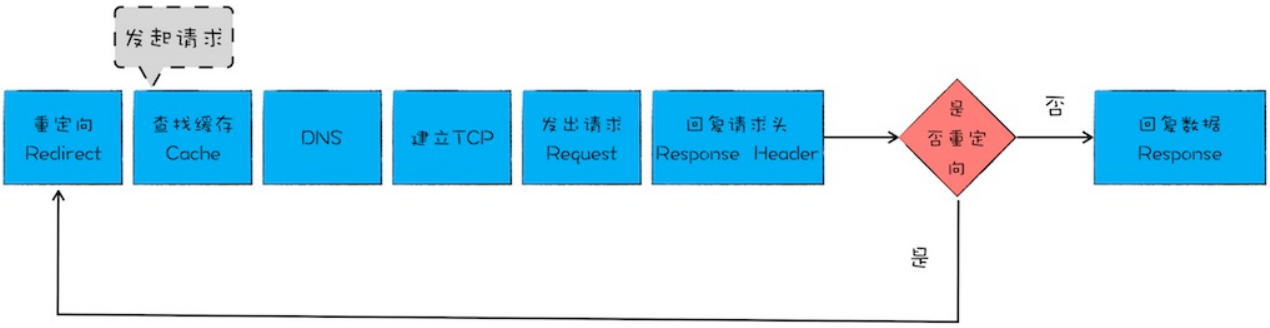


详细请求信息

你可以在此查看请求列表中任意一项的请求行和请求头信息，还可以查看响应行、响应头和响应体。然后你可以根据这些查看的信息来判断你的业务逻辑是否正确，或者有时候也可以用来逆向推导别人网站的业务逻辑。

3. 单个资源的时间线

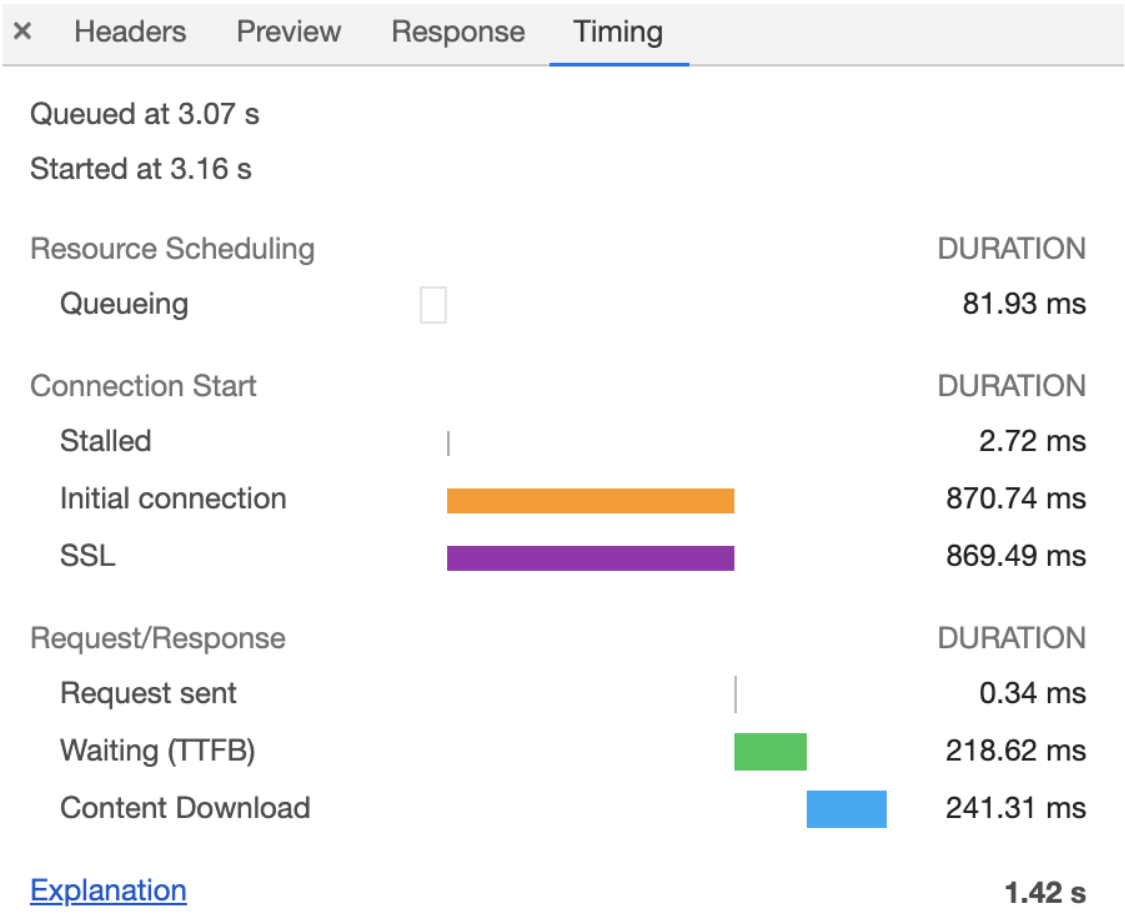
了解了每个资源的详细请求信息之后，我们再来分析单个资源请求时间线，这就涉及具体的HTTP请求流程了。



浏览器中HTTP请求流程

我们再回顾下在《03 | HTTP请求流程：为什么很多站点第二次打开速度会很快？》这篇文章，我们介绍过发起一个HTTP请求之后，浏览器首先查找缓存，如果缓存没有命中，那么继续发起DNS请求获取IP地址，然后利用IP地址和服务器端建立TCP连接，再发送HTTP请求，等待服务器响应；不过，如果服务器响应头中包含了重定向的信息，那么整个流程就需要重新再走一遍。这就是在浏览器中一个HTTP请求的基础流程。

那详细列表中是如何表示出这个流程的呢？这就要重点看下时间线面板了：



单个文件的时间线

那面板中这各项到底是什么含义呢？

第一个是Queuing，也就是排队的意思，当浏览器发起一个请求的时候，会有很多原因导致该请求不能被立即执行，而是需要排队等待。导致请求处于排队状态的原因有很多。

- 首先，页面中的资源是有优先级的，比如CSS、HTML、JavaScript等都是页面中的核心文件，所以优先级最高；而图片、视频、音频这类资源就不是核心资源，优先级就比较低。通常当后者遇到前者时，就需要“让路”，进入待排队状态。
- 其次，我们前面也提到过，浏览器会为每个域名最多维护6个TCP连接，如果发起一个HTTP请求时，这6个TCP连接都处于忙碌状态，那么这个请求就会处于排队状态。
- 最后，网络进程在为数据分配磁盘空间时，新的HTTP请求也需要短暂地等待磁盘分配结束。

等待排队完成之后，就要进入发起连接的状态了。不过在发起连接之前，还有一些原因可能导致连接过程被推迟，这个推迟就表现在面板中的**Stalled**上，它表示停滞的意思。

这里需要额外说明的是，如果你使用了代理服务器，还会增加一个**Proxy Negotiation**阶段，也就是代理协商阶段，它表示代理服务器连接协商所用的时间，不过在上图中没有体现出来，因为这里我们没有使用代理服务器。

接下来，就到了**Initial connection/SSL阶段**了，也就是和服务器建立连接的阶段，这包括了建立TCP连接所花费的时间；不过如果你使用了HTTPS协议，那么还需要一个额外的SSL握手时间，这个过程主要是用来协商一些加密信息的。（关于SSL协商的详细过程，我们会在Web安全模块中介绍。）

和服务器建立好连接之后，网络进程会准备请求数据，并将其发送给网络，这就是**Request sent阶段**。通常这个阶段非常快，因为只需要把浏览器缓冲区的数据发送出去就结束了，并不需要判断服务器是否接收到了，所以这个时间通常不到1毫秒。

数据发送出去了，接下来就是等待接收服务器第一个字节的数据，这个阶段称为Waiting (TTFB)，通常也称为“**第一字节时间**”。 TTFB是反映服务端响应速度的重要指标，对服务器来说，TTFB 时间越短，就说明服务器响应越快。

接收到第一个字节之后，进入陆续接收完整数据的阶段，也就是**Content Download阶段**，这意味着从第一字节时间到接收到全部响应数据所用的时间。

优化时间线上耗时项

了解了时间线面板上的各项含义之后，我们就可以根据这个请求的时间线来实现相关的优化操作了。

1. 排队（Queuing）时间过久

排队时间过久，大概率是由浏览器为每个域名最多维护6个连接导致的。那么基于这个原因，你就可以让1个站点下面的资源放在多个域名下面，比如放到3个域名下面，这样就可以同时支持18个连接了，这种方案称为**域名分片**技术。除了域名分片技术外，我个人还建议你**把站点升级到HTTP2**，因为HTTP2已经没有每个域名最多维护6个TCP连接的限制了。

2. 第一字节时间（TTFB）时间过久

这可能的原因有如下：

- **服务器生成页面数据的时间过久**。对于动态网页来说，服务器收到用户打开一个页面的请求时，首先要从数据库中读取该页面需要的数据，然后把这些数据传入到模板中，模板渲染后，再返回给用户。服务器在处理这个数据的过程中，可能某个环节会出问题。
- **网络的原因**。比如使用了低带宽的服务器，或者本来用的是电信的服务器，可联通的网络用户要来访问你的服务器，这样也会拖慢网速。
- **发送请求头时带上了多余的用户信息**。比如一些不必要的Cookie信息，服务器接收到这些Cookie信息之后可能需要对每一项都做处理，这样就加大了服务器的处理时长。

对于这三种问题，你要有针对性地出一些解决方案。面对第一种服务器的问题，你可以想办法去提高服务器的处理速度，比如通过增加各种缓存的技术；针对第二种网络问题，你可以使用CDN来缓存一些静态文件；至于第三种，你在发送请求时就去尽可能地减少一些不必要的Cookie数据信息。

3. Content Download时间过久

如果单个请求的Content Download花费了大量时间，有可能是字节数太多的原因导致的。这时候你就需要减少文件大小，比如压缩、去掉源码中不必要的注释等方法。

总结

好了，今天就介绍到这里了，下面我来总结下今天的内容。

首先我们简单介绍了Chrome开发者工具10个基础的面板信息；然后重点剖析了网络面板，再结合之前介绍的网络请求流程来重点分析了网络面板中时间线的各个指标的含义；最后我们还简要分析了时间线中各项指标出现异常的可能原因，并给出了一些优化方案。

其实通过今天的分析，我们可以得出这样一个结论：如果你要去做一些实践性的项目优化，理解其背后的理论至关重要。因为理论就是一条“线”，它会把各种实践的内容“串”在一起，然后你可以围绕着这条“线”来排查问题。

思考时间

今天我们介绍了网络面板，还有一个非常重要的Performance面板我们没有介绍，不过你可以去网上查找一些相关的资料。

所以今天留给你的是一道实际操作的题目，你可以结合网络面板和Performance面板来分析一个Web应用的性能瓶颈（比如https://www.12306.cn ）。

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

“浏览器中的页面循环系统”模块我们已经介绍完了，循环系统是页面的基础，理解了循环系统能让我们从本质上更好地理解页面的工作方式，加深我们对一些前端概念的理解。

接下来我们就要进入新的模块了，也就是“浏览器中的页面”模块，正如专栏简介中所言，页面是浏览器的核心，浏览器中的所有功能点都是服务于页面的，而Chrome开发者工具又是工程师调试页面的核心工具，所以在这个模块的开篇，我想先带你来深入了解下Chrome开发者工具。

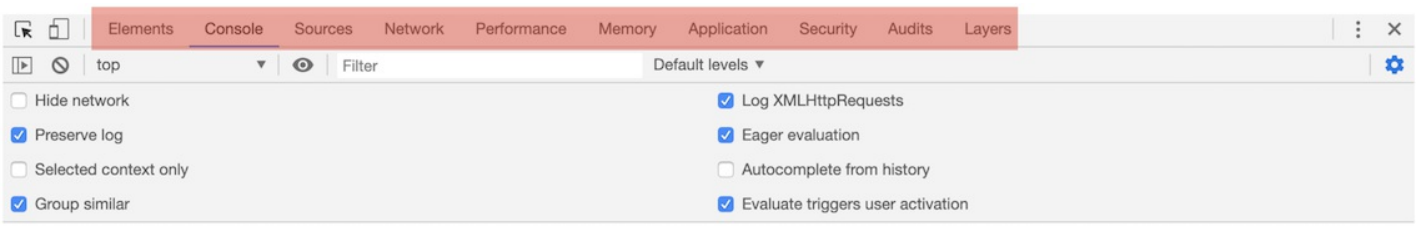
Chrome开发者工具（简称DevTools）是一组网页制作和调试的工具，内嵌于Google Chrome 浏览器中。Chrome开发者工具非常重要，所蕴含的内容也是非常多的，熟练使用它能让你更加深入地了解浏览器内部工作原理。（Chrome开发者工具也在不停地迭代改进，如果你想使用最新版本，可以使用Chrome Canary。）

作为这一模块的第一篇文章，我们主要聚焦**页面的源头**和**网络数据的接收**，这些发送和接收的数据都能体现在开发者工具的网络面板上。不过为了你能更好地理解和掌握，我们会先对Chrome开发者工具做一个大致的介绍，然后再深入剖析网络面板。

Chrome开发者工具

Chrome开发者工具有很多重要的面板，比如与性能相关的有网络面板、Performance面板、内存面板等，与调试页面相关的有Elements面板、Sources面板、Console面板等。

你可以在浏览器窗口的右上方选择Chrome菜单，然后选择“更多工具–>开发者工具”来打开Chrome开发者工具。打开的页面如下图所示：



从图中可以看出，它一共包含了10个功能面板，包括了Elements、Console、Sources、NetWork、Performance、Memory、Application、Security、Audits和Layers。

关于这10个面板的大致功能，我做了一个表格，感兴趣的话，你可以详细看下：

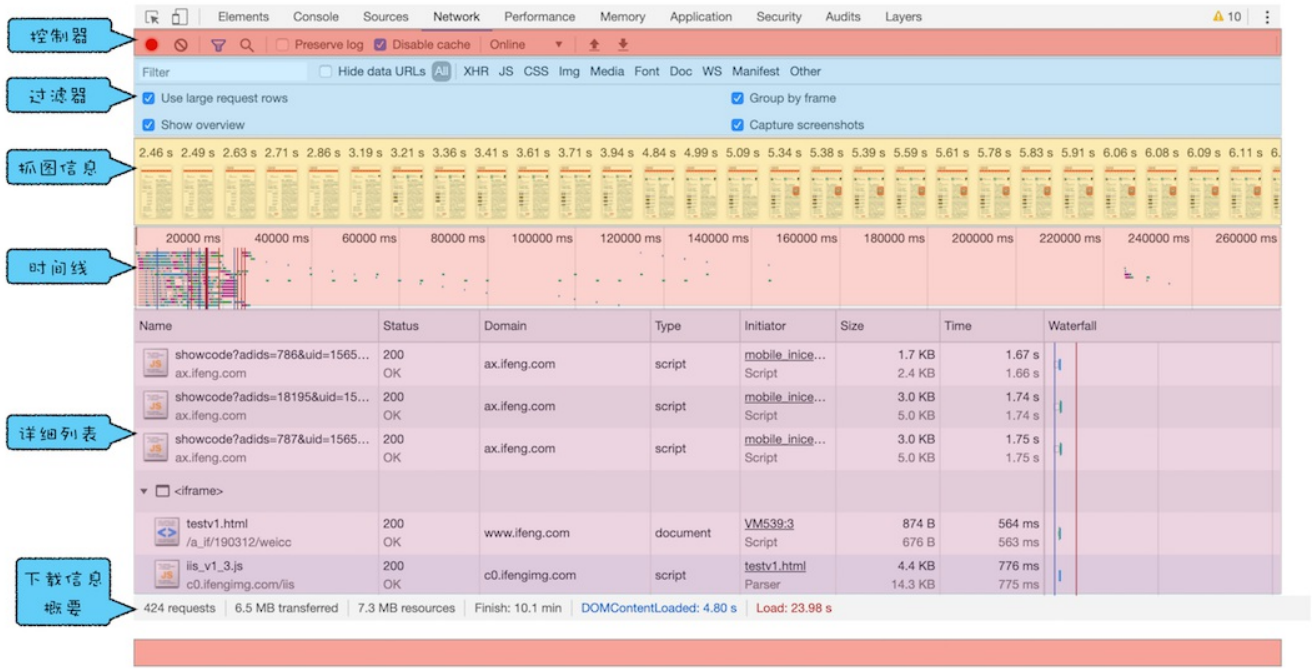
名称	描述
Elements面板	可以查看DOM结构、编辑CSS样式，用于测试页面布局和设计页面。
Console面板	可以看成是JavaScript Shell，能执行JavaScript脚本。通过Console还能和页面中的JavaScript对象交互。
Sources面板	1. 查看Web应用加载的所有文件； 2. 编辑CSS和JavaScript文件内容； 3. 将打乱的CSS文件或者JavaScript文件格式化； 4. 支持JavaScript的调试功能； 5. 设置工作区，将更改的文件保存到本地文件夹中。
NetWork（网络）面板	展示了页面中所有的请求内容列表，能查看每项请求的请求行、请求头、请求体、时间线以及网络请求瀑布图等信息。
Performance面板	记录和查看Web应用生命周期内的各种事件，并用来分析在执行过程中一些影响性能的要点。
Memory面板	用来查看运行过程中的JavaScript占用堆内存情况，追踪是否存在内存泄漏的情况等。
Application（应用）面板	查看Web应用的数据存储情况； PWA的基础数据：IndexedDB；Web SQL；本地和会话存储；Cookie；应用程序缓存；图像；字体和样式表等。
Security（安全）面板	显示当前页面一些基础的安全信息。
Audits面板	会对当前网页进行网络利用情况、网页性能方面的诊断，并给出一些优化建议。
Layers面板	展示一些渲染过程中分层的基础信息。

简单来说，Chrome开发者工具为我们提供了通过界面访问或者编辑DOM和CSSOM的能力，还提供了强大的调试功能和查看性能指标的能力。

OK，接下来我们就要重点看下其中重要的Network面板，即网络面板。

网络面板

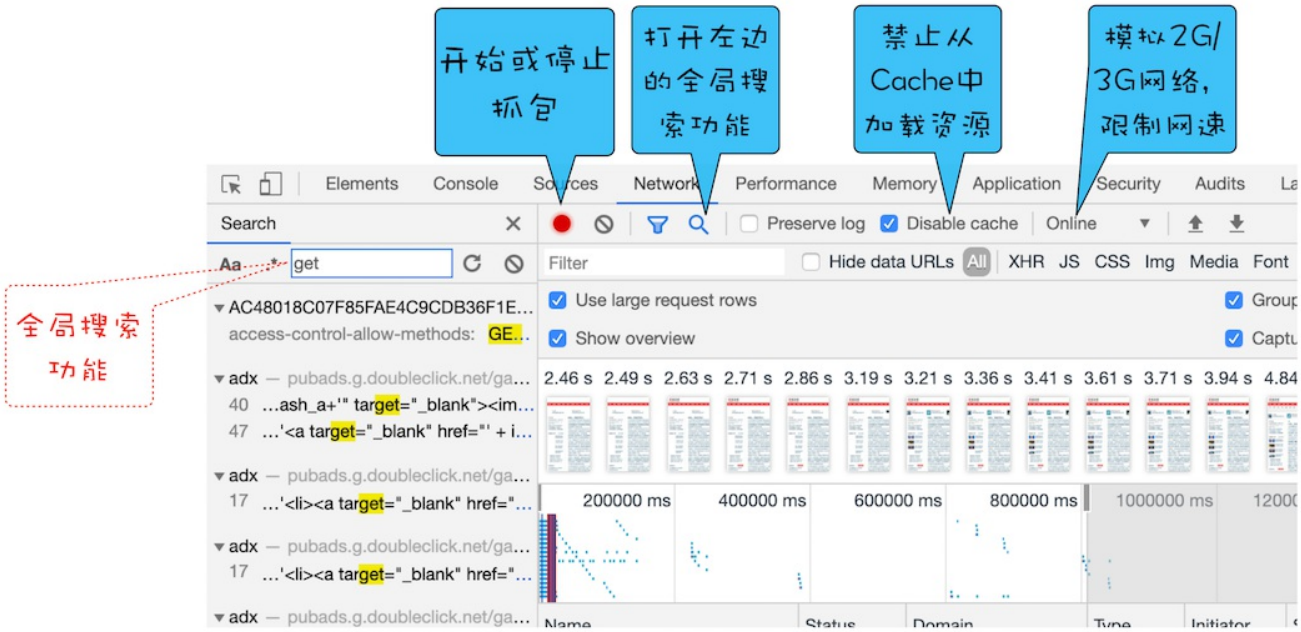
网络面板由控制器、过滤器、抓图信息、时间线、详细列表和下载信息概要这6个区域构成（如下图所示）。



网络面板概要图

1. 控制器

其中，控制器有4个比较重要的功能，我们按照下文中的这张图来简单介绍下。



控制器概要图

- 红色圆点的按钮，表示“开始/暂停抓包”，这个功能很常见，很容易理解。
- “全局搜索”按钮，这个功能就非常重要了，可以在所有下载资源中搜索相关内容，还可以快速定位到某几个你想要的文件上。
- Disable cache，即“禁止从Cache中加载资源”的功能，它在调试Web应用的时候非常有用，因为开启了Cache会影响到网络性能测试的结果。
- Online按钮，是“模拟2G/3G”功能，它可以限制带宽，模拟弱网情况下页面的展现情况，然后你就可以根据实际展示情况来动态调整策略，以便让Web应用更加适用于这些弱网。

2. 过滤器

网络面板中的过滤器，主要就是起过滤功能。因为有时候一个页面有太多内容在详细列表区域中展示了，而你可能只想查看JavaScript文件或者CSS文件，这时候就可以通过过滤器模块来筛选你想要的文件类型。

3. 抓图信息

抓图信息区域，可以用来分析用户等待页面加载时间内所看到的内容，分析用户实际的体验情况。比如，如果页面加载1秒多之后屏幕截图还是白屏状态，这时候就需要分析是网络还是代码的问题了。（勾选面板上的“Capture screenshots”即可启用屏幕截图。）

4. 时间线

时间线，主要用来展示HTTP、HTTPS、WebSocket加载的状态和时间的一个关系，用于直观感受页面的加载过程。如果是多条竖线堆叠在一起，那说明这些资源被同时被加载。至于具体到每个文件的加载信息，还需要用到下面要讲的详细列表。

5. 详细列表

这个区域是最重要的，它详细记录了每个资源从发起请求到完成请求这中间所有过程的状态，以及最终请求完成的数据信息。通过该列表，你就能很容易地去诊断一些网络问题。

详细列表是我们本篇文章介绍的重点，不过内容比较多，所以放到最后去专门介绍了。

6. 下载信息概要

下载信息概要中，你要重点关注下DOMContentLoaded和Load两个事件，以及这两个事件的完成时间。

- DOMContentLoaded，这个事件发生后，说明页面已经构建好DOM了，这意味着构建DOM所需要的HTML文件、JavaScript文件、CSS文件都已经下载完成了。
- Load，说明浏览器已经加载了所有的资源（图像、样式表等）。

通过下载信息概要面板，你可以查看触发这两个事件所花费的时间。

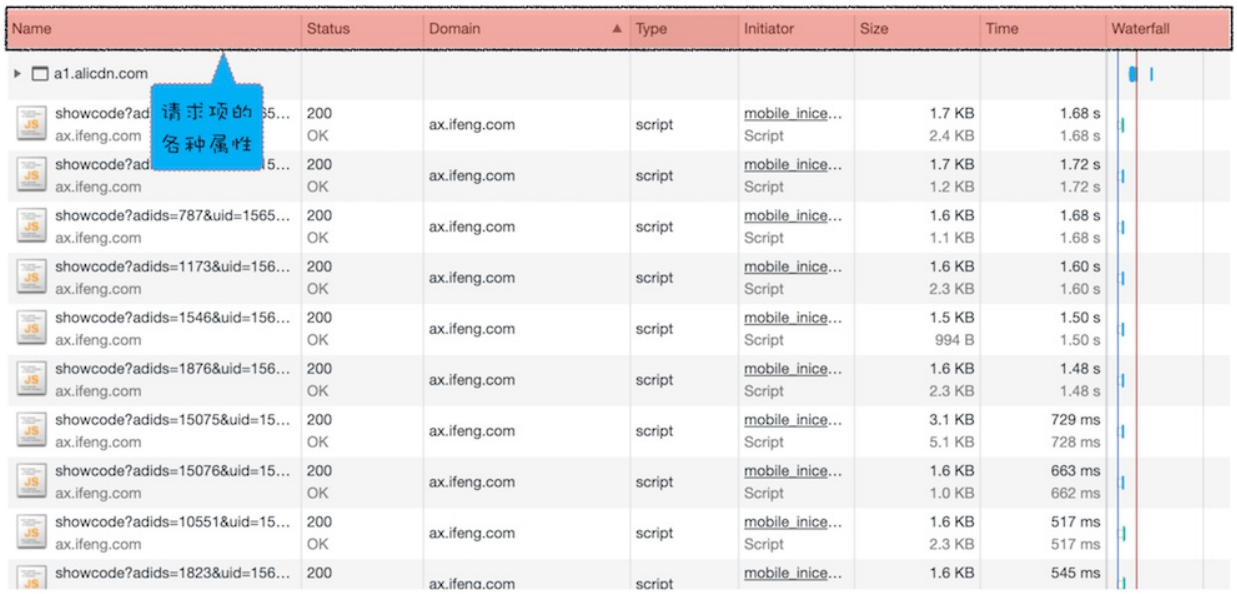
网络面板中的详细列表

下面我们就来重点介绍网络面板中的详细列表，这里面包含了大量有用的信息。

1. 列表的属性

列表的属性比较多，比如Name、Status、Type、Initiator等等，这个不难理解。当然，你还可以通过点击右键的下拉菜单来添加其他属性，这里我就不再赘述了，你可以自己上手实操一下。

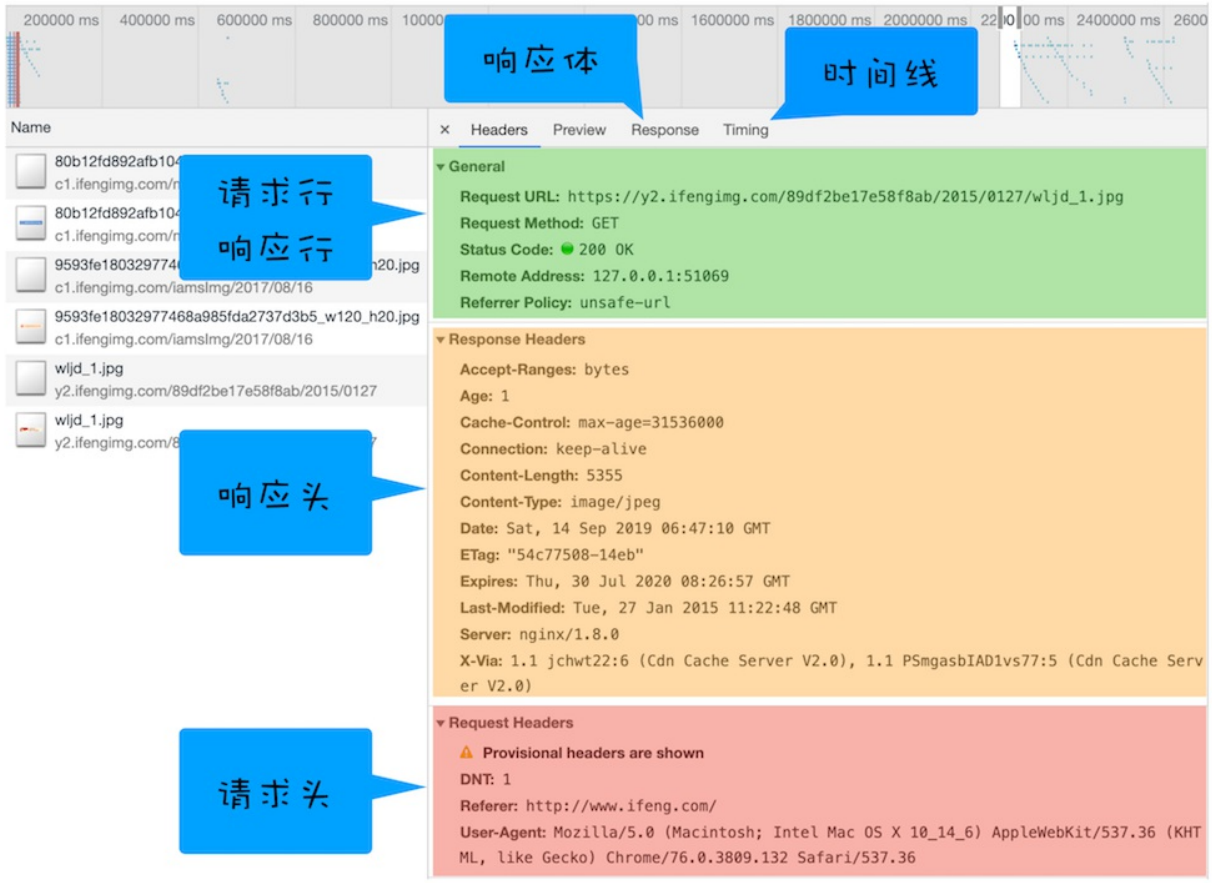
另外，你也可以按照列表的属性来给列表排序，默认情况下，列表是按请求发起的时间来排序的，最早发起请求的资源在顶部。当然也可以按照返回状态码、请求类型、请求时长、内容大小等基础属性排序，只需点击相应属性即可。



根据属性排序

2. 详细信息

如果你选中详细列表中的一项，右边就会出现该项的详细信息，如下所示：

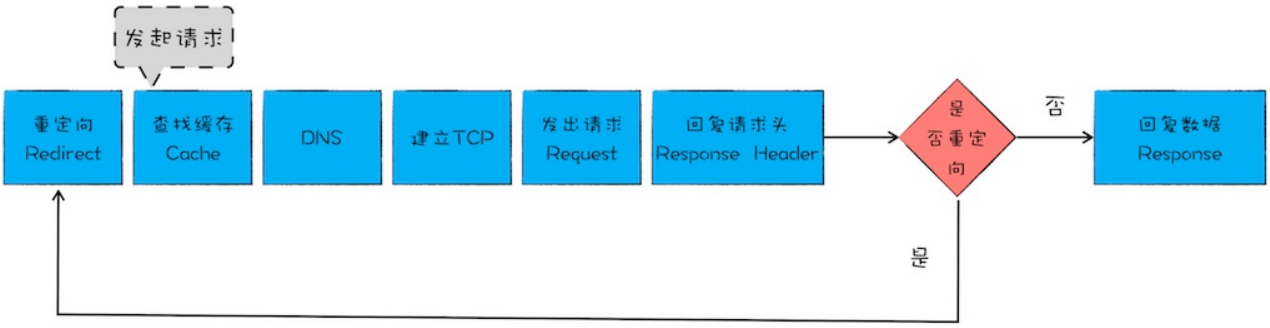


详细请求信息

你可以在此查看请求列表中任意一项的请求行和请求头信息，还可以查看响应行、响应头和响应体。然后你可以根据这些查看的信息来判断你的业务逻辑是否正确，或者有时候也可以用来逆向推导别人网站的业务逻辑。

3. 单个资源的时间线

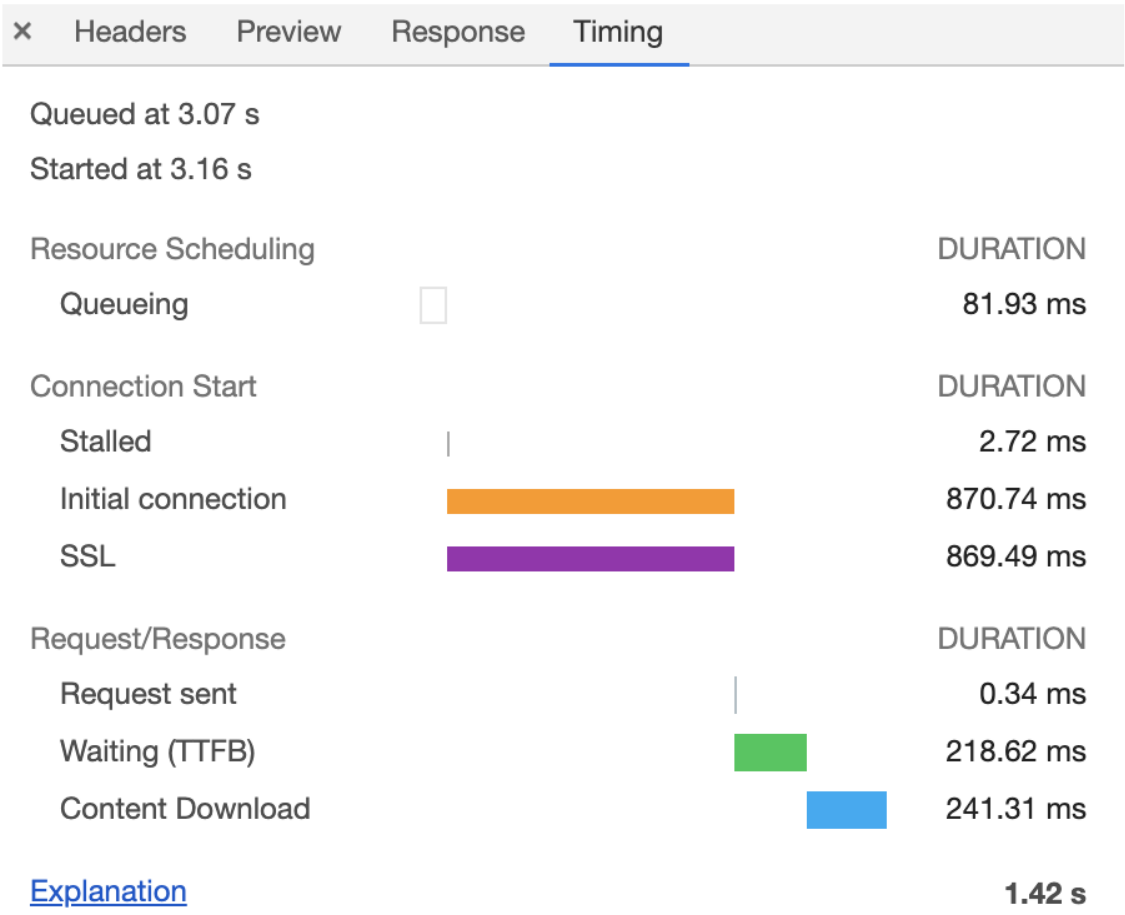
了解了每个资源的详细请求信息之后，我们再来分析单个资源请求时间线，这就涉及具体的HTTP请求流程了。



浏览器中HTTP请求流程

我们再回顾下在《03 | HTTP请求流程：为什么很多站点第二次打开速度会很快？》这篇文章，我们介绍过发起一个HTTP请求之后，浏览器首先查找缓存，如果缓存没有命中，那么继续发起DNS请求获取IP地址，然后利用IP地址和服务器端建立TCP连接，再发送HTTP请求，等待服务器响应；不过，如果服务器响应头中包含了重定向的信息，那么整个流程就需要重新再走一遍。这就是在浏览器中一个HTTP请求的基础流程。

那详细列表中是如何表示出这个流程的呢？这就要重点看下时间线面板了：



单个文件的时间线

那面板中这各项到底是什么含义呢？

第一个是Queuing，也就是排队的意思，当浏览器发起一个请求的时候，会有很多原因导致该请求不能被立即执行，而是需要排队等待。导致请求处于排队状态的原因有很多。

- 首先，页面中的资源是有优先级的，比如CSS、HTML、JavaScript等都是页面中的核心文件，所以优先级最高；而图片、视频、音频这类资源就不是核心资源，优先级就比较低。通常当后者遇到前者时，就需要“让路”，进入待排队状态。
- 其次，我们前面也提到过，浏览器会为每个域名最多维护6个TCP连接，如果发起一个HTTP请求时，这6个TCP连接都处于忙碌状态，那么这个请求就会处于排队状态。
- 最后，网络进程在为数据分配磁盘空间时，新的HTTP请求也需要短暂地等待磁盘分配结束。

等待排队完成之后，就要进入发起连接的状态了。不过在发起连接之前，还有一些原因可能导致连接过程被推迟，这个推迟就表现在面板中的**Stalled**上，它表示停滞的意思。

这里需要额外说明的是，如果你使用了代理服务器，还会增加一个**Proxy Negotiation**阶段，也就是代理协商阶段，它表示代理服务器连接协商所用的时间，不过在上图中没有体现出来，因为这里我们没有使用代理服务器。

接下来，就到了**Initial connection/SSL阶段**了，也就是和服务器建立连接的阶段，这包括了建立TCP连接所花费的时间；不过如果你使用了HTTPS协议，那么还需要一个额外的SSL握手时间，这个过程主要是用来协商一些加密信息的。（关于SSL协商的详细过程，我们会在Web安全模块中介绍。）

和服务器建立好连接之后，网络进程会准备请求数据，并将其发送给网络，这就是**Request sent阶段**。通常这个阶段非常快，因为只需要把浏览器缓冲区的数据发送出去就结束了，并不需要判断服务器是否接收到了，所以这个时间通常不到1毫秒。

数据发送出去了，接下来就是等待接收服务器第一个字节的数据，这个阶段称为Waiting (TTFB)，通常也称为“**第一字节时间**”。 TTFB是反映服务端响应速度的重要指标，对服务器来说，TTFB 时间越短，就说明服务器响应越快。

接收到第一个字节之后，进入陆续接收完整数据的阶段，也就是**Content Download阶段**，这意味着从第一字节时间到接收到全部响应数据所用的时间。

优化时间线上耗时项

了解了时间线面板上的各项含义之后，我们就可以根据这个请求的时间线来实现相关的优化操作了。

1. 排队（Queuing）时间过久

排队时间过久，大概率是由浏览器为每个域名最多维护6个连接导致的。那么基于这个原因，你就可以让1个站点下面的资源放在多个域名下面，比如放到3个域名下面，这样就可以同时支持18个连接了，这种方案称为**域名分片**技术。除了域名分片技术外，我个人还建议你**把站点升级到HTTP2**，因为HTTP2已经没有每个域名最多维护6个TCP连接的限制了。

2. 第一字节时间（TTFB）时间过久

这可能的原因有如下：

- **服务器生成页面数据的时间过久**。对于动态网页来说，服务器收到用户打开一个页面的请求时，首先要从数据库中读取该页面需要的数据，然后把这些数据传入到模板中，模板渲染后，再返回给用户。服务器在处理这个数据的过程中，可能某个环节会出问题。
- **网络的原因**。比如使用了低带宽的服务器，或者本来用的是电信的服务器，可联通的网络用户要来访问你的服务器，这样也会拖慢网速。
- **发送请求头时带上了多余的用户信息**。比如一些不必要的Cookie信息，服务器接收到这些Cookie信息之后可能需要对每一项都做处理，这样就加大了服务器的处理时长。

对于这三种问题，你要有针对性地出一些解决方案。面对第一种服务器的问题，你可以想办法去提高服务器的处理速度，比如通过增加各种缓存的技术；针对第二种网络问题，你可以使用CDN来缓存一些静态文件；至于第三种，你在发送请求时就去尽可能地减少一些不必要的Cookie数据信息。

3. Content Download时间过久

如果单个请求的Content Download花费了大量时间，有可能是字节数太多的原因导致的。这时候你就需要减少文件大小，比如压缩、去掉源码中不必要的注释等方法。

总结

好了，今天就介绍到这里了，下面我来总结下今天的内容。

首先我们简单介绍了Chrome开发者工具10个基础的面板信息；然后重点剖析了网络面板，再结合之前介绍的网络请求流程来重点分析了网络面板中时间线的各个指标的含义；最后我们还简要分析了时间线中各项指标出现异常的可能原因，并给出了一些优化方案。

其实通过今天的分析，我们可以得出这样一个结论：如果你要去做一些实践性的项目优化，理解其背后的理论至关重要。因为理论就是一条“线”，它会把各种实践的内容“串”在一起，然后你可以围绕着这条“线”来排查问题。

思考时间

今天我们介绍了网络面板，还有一个非常重要的Performance面板我们没有介绍，不过你可以去网上查找一些相关的资料。

所以今天留给你的是一道实际操作的题目，你可以结合网络面板和Performance面板来分析一个Web应用的性能瓶颈（比如https://www.12306.cn ）。

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。