# Web性能自动化测试

1. 介绍

从浏览器打开一个网站，需要DNS解析，tcp三次握手，发送请求，dom渲染，js加载等一系列操作，最终在用户面前展示完整的页面。

如果打开页面加载的时间过长，用户会立即放弃浏览。

所以在web项目中，web页面的加载时间测试时必不可少的。

## **全部页面加载时间**

全部页面载入时间指从最初启动浏览开始，直到所有元素都被加载完成后，在 2 秒后仍然没有网络活动的时间.  
0-2 秒：用户体验最好，打分 100  
2-8 秒：用户可以容忍，从第 2 秒开始，每超过 1 秒减 5 分  
8-15 秒：用户不能忍受，从第 2 秒开始，每超过 1 秒减 5 分

## **页面加载时间**

从页面开始加载到页面 onload 事件触发的时间。一般来说 onload 触发代表着直接通过 HTML 引用的 CSS，JS，图片资源已经完全加载完毕.

## **白屏时间**

输入网址到数据返回时间，此时页面还没有进行 dom 渲染

## **首字节时间**

从开始加载到收到服务器返回数据的第一字节的时间

## **DNS 时间**

进行域名解析所需要的时间

## **TCP 时间**

客户端建立连接的时间

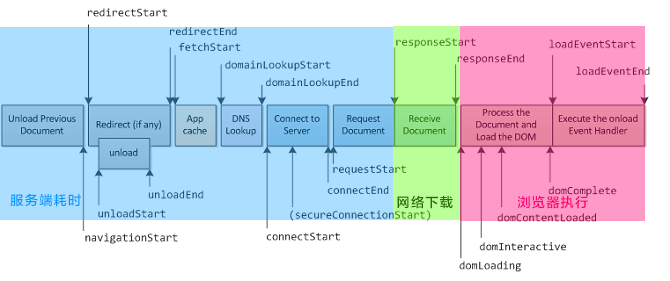
## **DOM 渲染时间**

DOM 节点渲染时间，此时页面还不能执行操作事件

## **js 加载时间**

js、css、图片加载，此时用户可以对页面进行操作事件

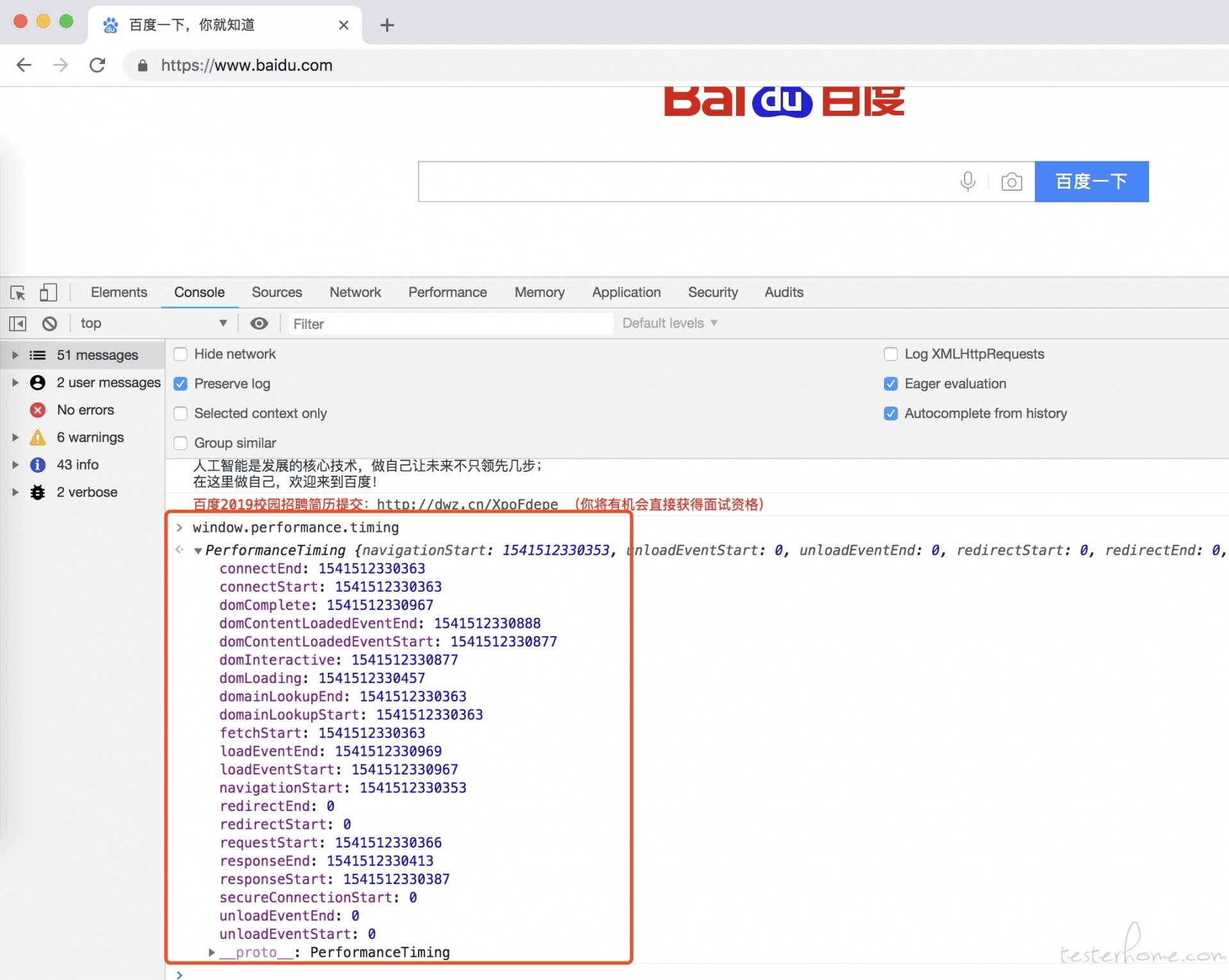
## **访问 web 页面过程**

[](https://img.alicdn.com/tps/TB1WlKAKpXXXXctXVXXXXXXXXXX-670-281.png)

## **查看性能加载数据**

使用 chrome 浏览器中的控制台，使用 window.performance.timing 函数可以查看页面的加载数据.

.

[](https://testerhome.com/uploads/photo/2018/f5f36687-8f68-4bb4-b80c-bb182c296599.jpg!large)

#### **姿势说明**

navigationStart：  
在同一个浏览器上下文中，当网页 load/unload 动作被触发时，也可能是提示关闭当前文档时（即回车键在 url 地址栏中按下，页面被再次刷新，submit 按钮被点击）。如果无前一个网页 unload，那么 navigationStart 的值就是 fetchStart。

redirectStart：  
页面重定向时的开始时间（如果存在重定向），无页面重定向时值是 0。

unloadEventStart：  
前一个网页（与当前页面同域）unload 的开始的时间戳，如果无前一个网页 unload 或者前一个网页与当前页面不同域，则值为 0

unloadEventEnd：  
和 unloadEventStart 相对应，返回前一个网页 unload 事件回调函数执行完毕的时间戳，如果无前一个网页 unload 或者前一个网页与当前页面不同域，则值为 0

redirectEnd：  
如果存在重定向的话，redirectEnd 表示最后一次重定向后服务器端 response 的数据被接收完毕的时间。无重定向的话值为 0。

fetchStart：  
在 fetchStart 和 domainLookupStart 之间，浏览器会检查当前文档的缓存，浏览器发起任何请求之前的时间值。

domainLookupStart：  
浏览器开始检查当前域名的 DNS 之前的那一时刻。如果因为任何原因没有去检查 DNS（即浏览器使用了缓存，持久连接，或者本地资源），那么它的值等同于 fetchStart。

domainLookupEnd：  
指浏览器完成 DNS 检查时的时间。如果 DNS 没有被检查，那么它的值等同于 fetchStart。

connectStart：  
当浏览器开始于服务器连接时的时间。如果资源取自缓存（或者服务器由于其他任何原因没有建立连接，例如持久连接），那么它的值等同于 domainLookupEnd。

connectEnd：  
当浏览器端完成与服务器端建立连接的时刻。如果没有建立连接它的值等同于 domainLookupEnd。

secureConnectionStart：  
如果页面使用 HTTPS，它的值是安全连接握手之前的时刻。如果该属性不可用，则返回 undefined。如果该属性可用，但没有使用 HTTPS，则返回 0。

responseStart：  
指客户端收到从服务器端（或缓存、本地资源）响应回的第一个字节的数据的时刻。

responseEnd：  
指客户端收到从服务器端（或缓存、本地资源）响应回的最后一个字节的数据的时刻。

domLoading：  
指开始解析渲染 DOM 树的时间，document 对象创建完成，Document.readyState 变为 loading，并将抛出 readystatechange 相关事件的时刻。

domInteractive：  
指完成解析 DOM 树的时间，包括在 “传统模式” 下被阻塞的通过 script 标签加载的内容（除了使用 defer 或者 async 属性异步加载的情况），Document.readyState 变为 interactive，并将抛出 readystatechange 相关事件，这时候并没有开始加载网页内的资源。

domContentLoadedEventStart：  
当 DOMContentLoaded 事件触发之前，DOM 解析完成后，网页内资源加载开始的时间，浏览器完成所有 script（包括设置了 defer 属性但未设置 async 属性的 script）的下载和解析之后的时刻。

domContentLoadedEventEnd：  
当 DOMContentLoaded 事件完成之后的时刻，DOM 解析完成后，网页内资源加载完成的时间，它也是 javascript 类库中 DOMready 事件触发的时刻。

domComplete：  
DOM 树解析完成，且资源也准备就绪的时间，如果已经没有任何延迟加载的事件（所有图片的加载）阻止 load 事件发生，那么该时刻将会将 document.readyState 属性设置为 "complete"，并将抛出 readystatechange 相关事件，此时刻就是 domComplete。

loadEventStart：  
load 事件发送给文档，也即 load 回调函数开始执行的时间，该属性返回的是 load 事件刚刚发生的时刻，如果 load 事件还没有发生，则返回 0。

loadEventEnd：  
load 事件的回调函数执行完毕的时间，该属性返回 load 事件完成之后的时刻。如果 load 事件未发生，则返回 0。

#### **网页性能常见指标**

白屏时间（domLoading - navigationStart）  
等待页面开始展示的时间

首屏时间（首屏内容渲染结束时间点 - navigationStart）  
用户能够看到区域内所有元素加载完的时间，这个首屏渲染结束时间点取得有点蛋疼，还没方案

可交互时间（domInteractive - navigationStart）  
完成 Dom 解析，可以绑定用户事件响应用户操作的时间

页面加载完成的时间（ loadEventEnd - navigationStart）  
反映用户等待页面可用的时间

DOM 解析时间（domComplete - responseEnd）  
值太大时，请检查一下是否你的 DOM 树嵌套是不是太多了太复杂，是否可以精简优化

重定向的时间（redirectEnd - redirectStart）  
尽量不使用重定向

DNS 解析时间（domainLookupEnd - domainLookupStart）  
是否需要进行 DNS 预加载，是否可以尽量减少不同的域名

首一个字节时间（responseStart - navigationStart）  
反映用户拿到资源占用的时间，是否需要增加加异地机房了么，是否需要进行加 CDN ，是否需要加带宽等

内容加载完成的时间（responseEnd - requestStart）  
页面返回的内容是否进行压缩优化（页面内容 gzip 压缩，css/js 等压缩）

onload 执行时间（loadEventEnd - loadEventStart）  
画面 onload 回调函数是否执行了过多耗时操作，是否需要考虑延迟加载、按需加载等策略

DNS 缓存时间（domainLookupStart - fetchStart）  
是否需要进行 DNS 缓存

卸载页面的时间（unloadEventEnd - unloadEventStart）  
前一个网页卸载开始到完成的时间

TCP/IP 建连时间（connectEnd - connectStart）  
HTTP（TCP） 开始到完成建立连接的时间

#### 当出现目标服务器拒绝时，关掉代理，当前有效的解决方式如下：

#### 第一步启动 chrome 浏览器

启动浏览器时要确保:

* 所有 chrome 浏览器已经关闭
* chrome 的安装目录已经被加入环境变量 Path 中

这样你在命令行里输入:chrome.exe --remote-debugging-port=9222, 这样就能启动浏览器，并且开启了调试端口 9222。

重新打开一个浏览器，输入 http://127.0.0.1:9222, 打开这个网址就应该看到出现一个页面，并且有几个选项，但是我们并不关心这个页面，只是用来证明 chrome 的 server 已经正确的启动了。