高性能的 Web 页面

Stoyan Stefanov

Yahoo! Exceptional Performance

http://developer.yahoo.com/performance

PHP Quebec, March 13, 2008

迟钝的 Web

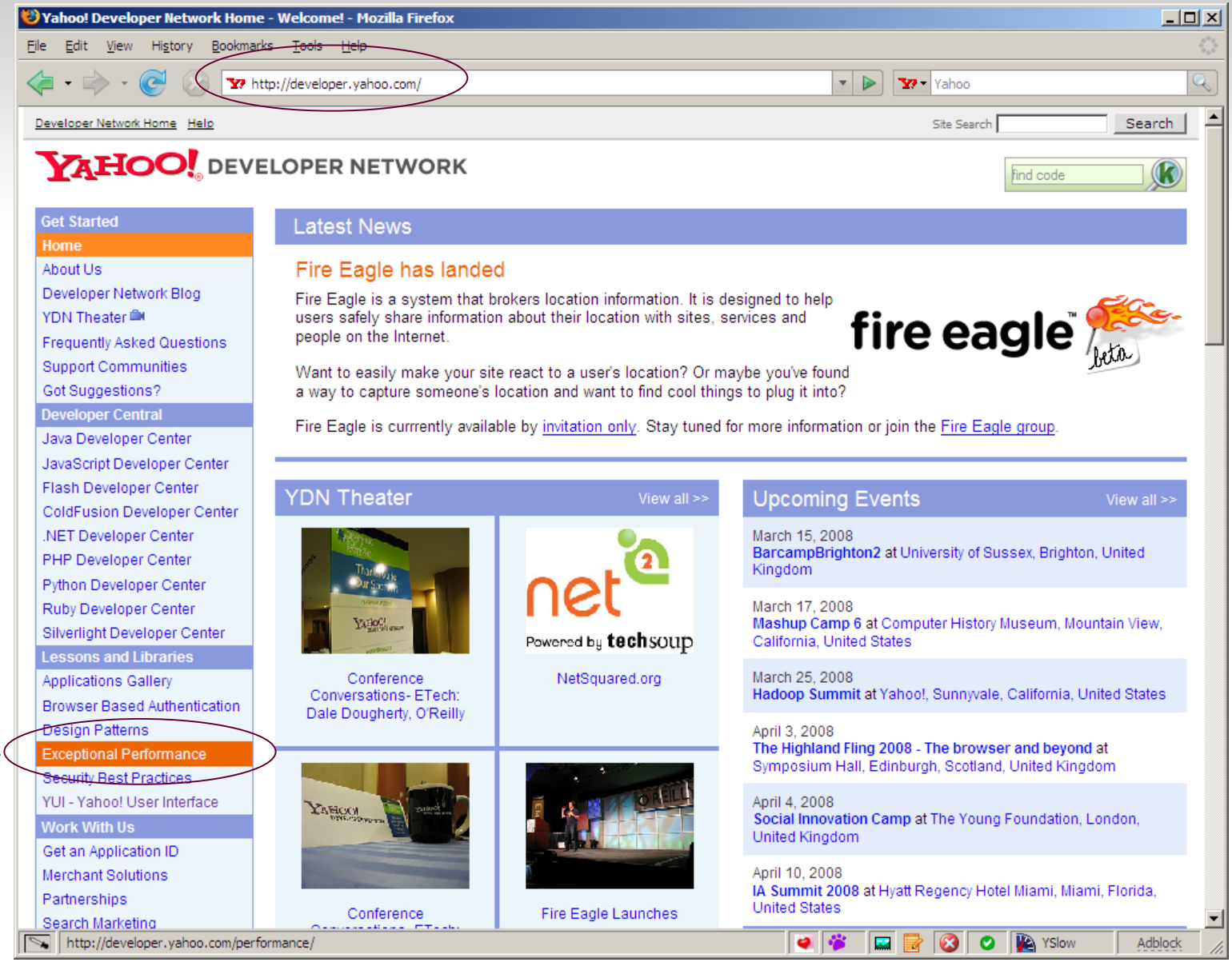
- Web 我们正逐渐习惯将 Web 作为一个日常使用的工具
- 我们都希望有一个好的用户体验
- 我们不能容忍慢的页面（我们有其他的选择）
- 慢　500 ms = 20% 将放弃访问　(Google)
- 慢　100 ms = 1% 将放弃交易　(Amazon)

本次话题

- 如何改善页面的性能
- 集中在前端
- 提高页面速度的 14 种最优方法
- … 20 个更多的

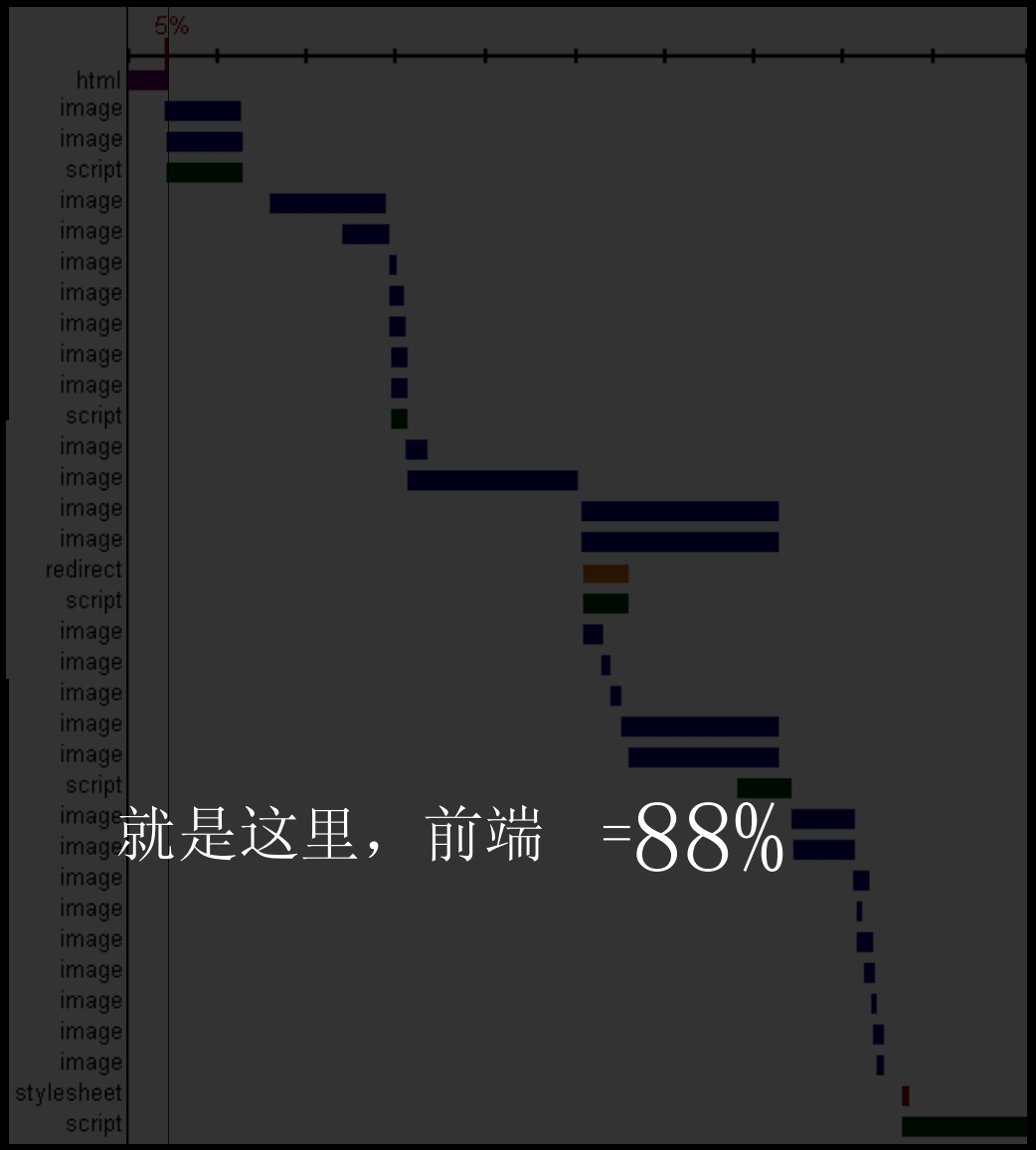
Yahoo 的优异表现！

- 保质的提高所有 Yahoo! 互联网上产品的性能
- 专家研究中心
- 构建工具，分析数据
- 收集，调查研究，对内对外推广最优的方法


Yahoo! Developer Network Home - Welcome! - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://developer.yahoo.com/
Yahoo
Developer Network Home Help
Site Search
Search
YAHOO! DEVELOPER NETWORK
find code
Get Started
Home
About Us
Developer Network Blog
YDN Theater
Frequently Asked Questions
Support Communities
Got Suggestions?
Developer Central
Java Developer Center
JavaScript Developer Center
Flash Developer Center
ColdFusion Developer Center
.NET Developer Center
PHP Developer Center
Python Developer Center
Ruby Developer Center
Silverlight Developer Center
Lessons and Libraries
Applications Gallery
Browser Based Authentication
Design Patterns
Exceptional Performance
Security Best Practices
YUI - Yahoo! User Interface
Work With Us
Get an Application ID
Merchant Solutions
Partnerships
Search Marketing
Latest News
Fire Eagle has landed
Fire Eagle is a system that brokers location information. It is designed to help users safely share information about their location with sites, services and people on the Internet.
fire eagle beta
Want to easily make your site react to a user's location? Or maybe you've found a way to capture someone's location and want to find cool things to plug it into?
Fire Eagle is currrently available by invitation only. Stay tuned for more information or join the Fire Eagle group.
YDN Theater
View all >>
Conference Conversations- ETech: Dale Dougherty, O'Reilly
net2 Powered by techsoup
NetSquared.org
Conference
Fire Eagle Launches
Upcoming Events
View all >>
March 15, 2008
BarcampBrighton2 at University of Sussex, Brighton, United Kingdom
March 17, 2008
Mashup Camp 6 at Computer History Museum, Mountain View, California, United States
March 25, 2008
Hadoop Summit at Yahoo!, Sunnyvale, California, United States
April 3, 2008
The Highland Fling 2008 - The browser and beyond at Symposium Hall, Edinburgh, Scotland, United Kingdom
April 4, 2008
Social Innovation Camp at The Young Foundation, London, United Kingdom
April 10, 2008
IA Summit 2008 at Hyatt Regency Hotel Miami, Miami, Florida, United States
http://developer.yahoo.com/performance/
YSlow
Adblock

前端性能的重要性

后端 =5%



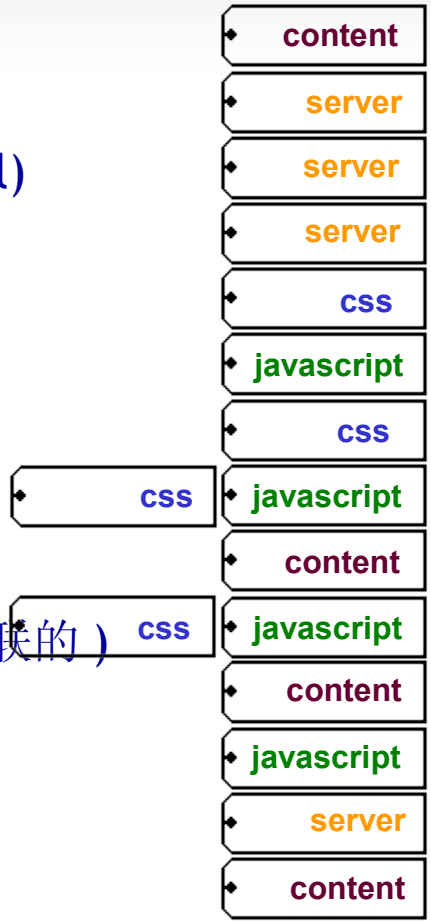
前端 =95%

集中在前端

- 80-90% 的优化空间
- 比改动后台要容易
- 工作成果已证明

14 条最优方法 （更新）

1. 尽可能的减少 HTTP 请求数
2. 使用 CDN
3. 添加 Expires 头（或者 Cache-control）
4. Gzip 组件
5. 把 CSS 样式放在页面的上方。
6. 将脚本放在底部（包括内联的）
7. 避免在 CSS 中使用 Expressions
8. 将 javascript 和 css 独立成外部文件
9. 减少 DNS 查询
10. 压缩 JavaScript 和 CSS 文件 （包括内联的）
11. 避免跳转
12. 移除重复的脚本
13. 配置 ETags
14. 缓存 Ajax 请求

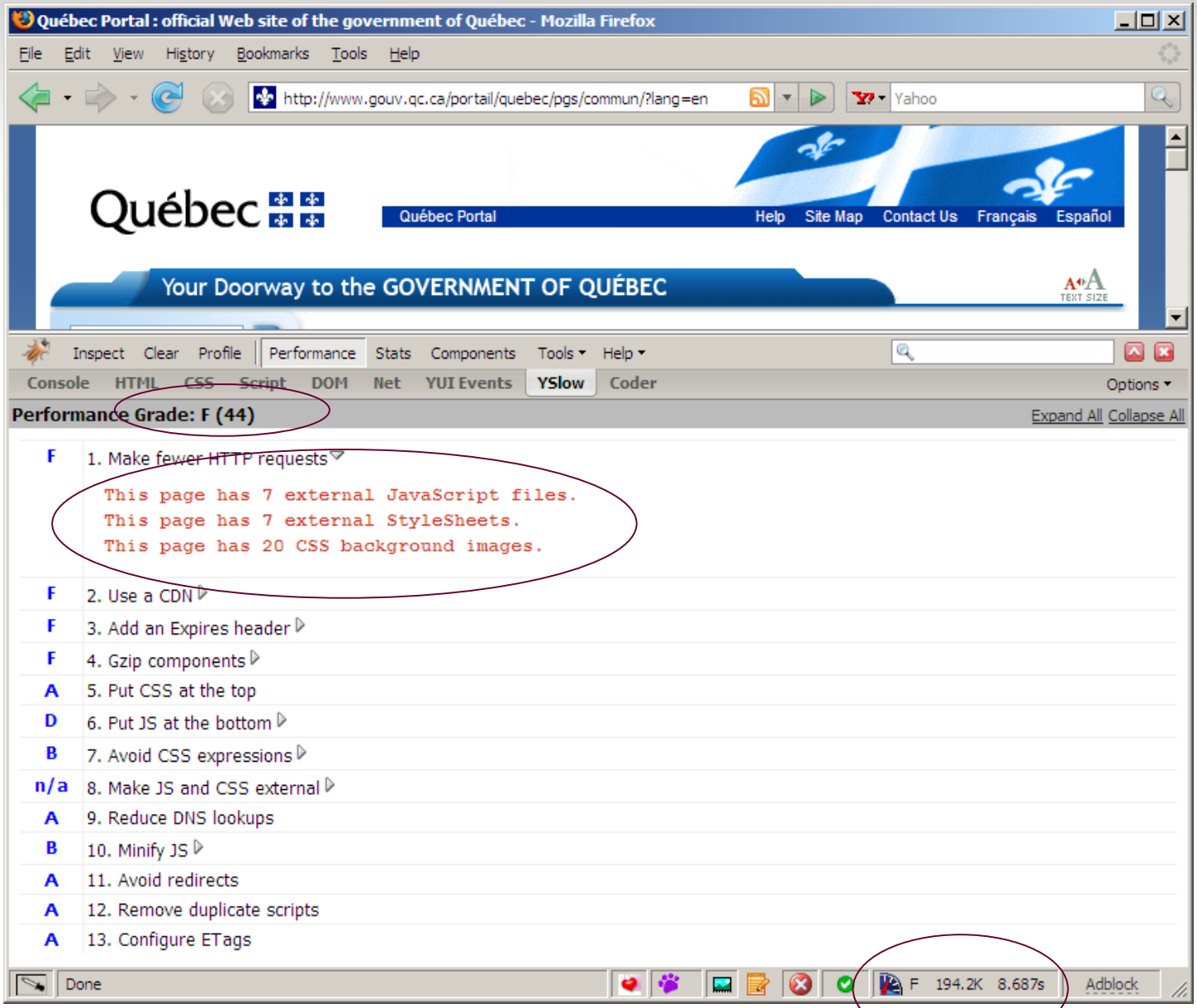


http://developer.yahoo.com/performance/rules.html

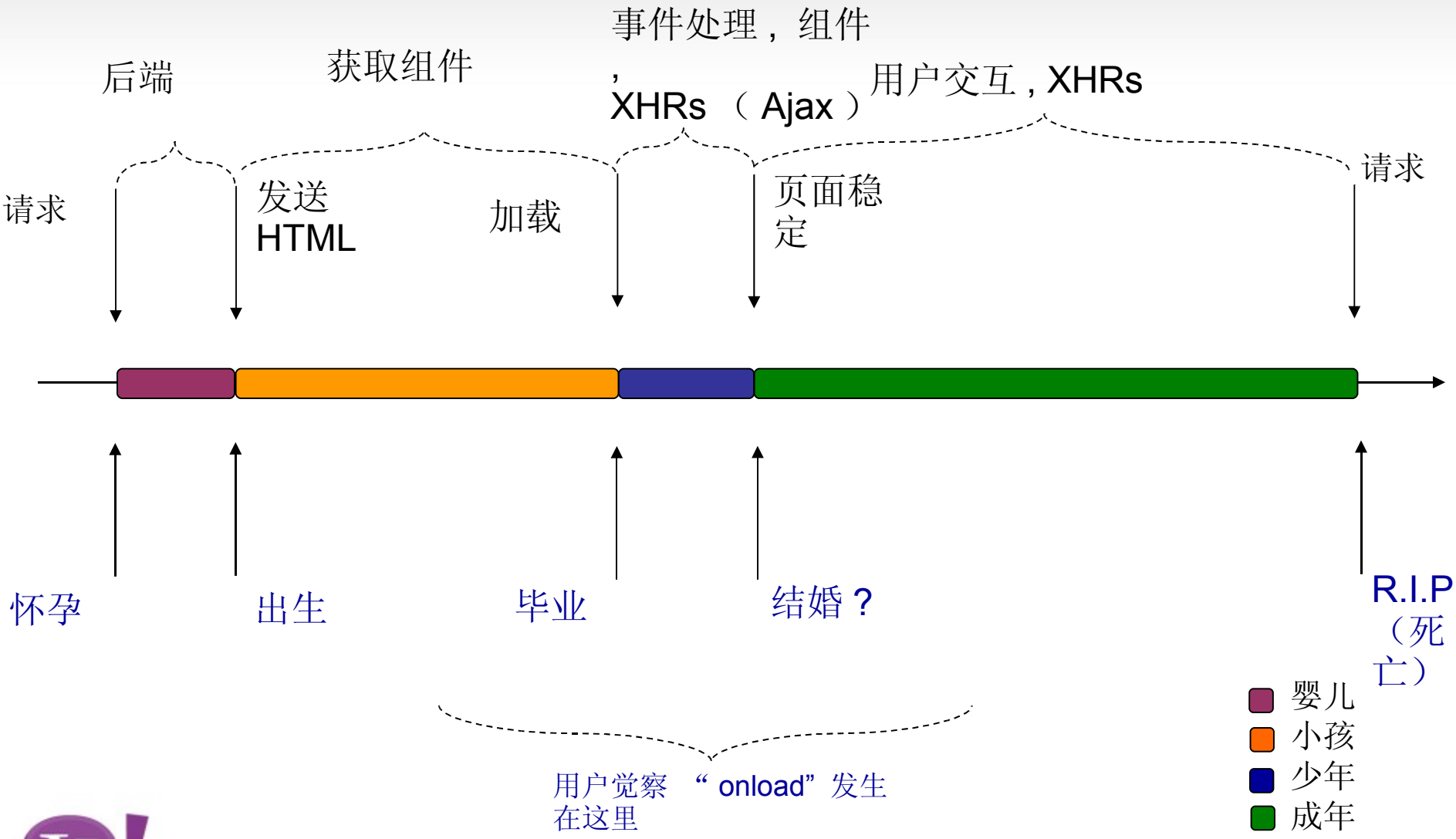
YSlow

- Yahoo! 的性能分析工具
- Firebug 的扩展（ Firefox ）
- 依照最优方法检测
- 评级

http://developer.yahoo.com/yslow/


Québec Portal : official Web site of the government of Québec - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.gouv.qc.ca/portail/quebec/pgs/commun/?lang=en
Yahoo
Québec
Québec Portal
Help
Site Map
Contact Us
Français
Español
Your Doorway to the GOVERNMENT OF QUÉBEC
TEXT SIZE
Inspect Clear Profile Performance Stats Components Tools Help
Console HTML CSS Script DOM Net YUI Events YSlow Coder
Options
Performance Grade: F (44)
Expand All Collapse All
F 1. Make fewer HTTP requests
This page has 7 external JavaScript files.
This page has 7 external StyleSheets.
This page has 20 CSS background images.
F 2. Use a CDN
F 3. Add an Expires header
F 4. Gzip components
A 5. Put CSS at the top
D 6. Put JS at the bottom
B 7. Avoid CSS expressions
n/a 8. Make JS and CSS external
A 9. Reduce DNS lookups
B 10. Minify JS
A 11. Avoid redirects
A 12. Remove duplicate scripts
A 13. Configure ETags
Done
F 194.2K 8.687s
Adblock

2.0 页面的生命


后端
获取组件
事件处理，组件，XHRs （Ajax）
用户交互，XHRs
请求
发送HTML
加载
页面稳定
请求
怀孕
出生
毕业
结婚？
R.I.P.（死亡）
用户觉察“onload”发生在这里
婴儿
小孩
少年
成年

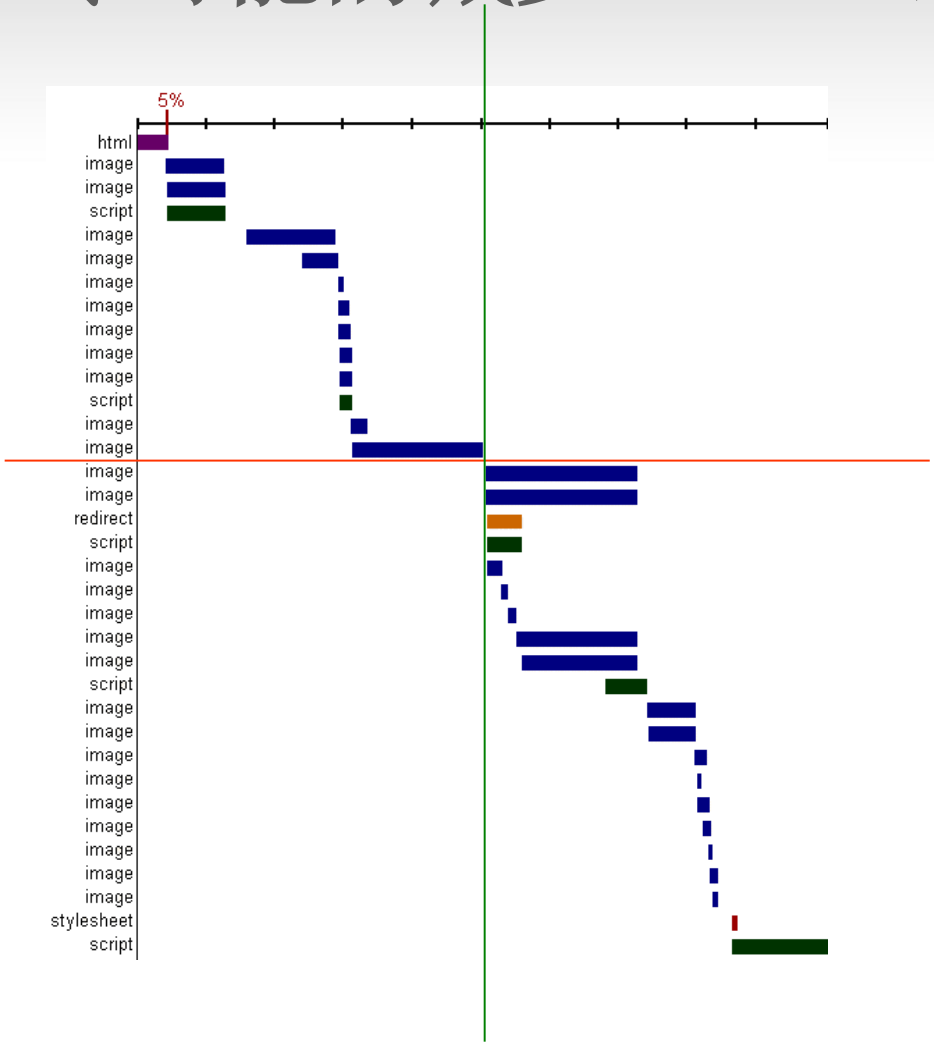
After YSlow "A"?

1.	Flush the buffer early	server
2.	Use GET for AJAX requests	server
3.	Post-load components	content
4.	Preload components	content
5.	Reduce the number of DOM elements	content
6.	Split components across domains	content
7.	Minimize the number of iframes	content
8.	No 404s	content
9.	Reduce cookie size	cookie
10.	Use cookie-free domains for components	cookie
11.	Minimize DOM access	javascript
12.	Develop smart event handlers	javascript
13.	Choose <link> over @import	css
14.	Avoid filters	css
15.	Optimize images	images
16.	Optimize CSS sprites	images
17.	Don't scale images in HTML	images
18.	Make favicon.ico small and cacheable	images
19.	Keep components under 25K	mobile
20.	Pack components into a multipart document	mobile

Part I

回顾 14 条最优方法 (更新)

尽可能的减少 HTTP 请求数



- 组件越少 = 页面越快
- HTTP 请求的开销
- 整合脚本，整合样式，整合图片为 CSS sprites

CSS Sprites

10 个请求变为了一个

```
background-position: -0px -0px;
background-position: -20px -0px;
background-position: -40px -0px;
background-position: -60px -0px;
background-position: -80px -0px;
background-position: -100px -0px;
background-position: -120px -0px;
background-position: -140px -0px;
background-position: -160px -0px;
background-position: -180px -0px;
```

Tools:

http://www.csssprites.com

http://spritegen.website-performance.org/

使用 CDN （ Content Delivery Network ）

- 对于静态的内容
- 让内容更靠近你的用户
- Akamai, Amazon S3

添加 Expire 头 (或者 Cache-control)

- 对于静态的内容
  - “Never expire” 设置， 设置 `Expires` 头到遥远的未来
  - 一旦内容被缓存，浏览器将不再请求他
  - 当你需要更新内容，重命名它
  - Apache 例子：

```
ExpiresActive On
ExpiresDefault "modification plus 10 years"
```

- 对于动态的内容
  - 使用 `Cache-control`
  - 帮助浏览器发送 `If-Modified-Since`
  - 将更新在 YUI blog/YDN， 请关注

启用 Gzip 压缩

- 发送压缩的内容，由浏览器解压
- 主流浏览器会明白压缩的内容
- 搜索引擎蜘蛛也是如此
- 请求头信息

```
Accept-Encoding: gzip,deflate
```

- 响应头信息

```
Content-Encoding: gzip
```

- 所有的文本内容都应该被 gzip 压缩：html (php), js, css, xml, txt…

把 CSS 样式放在页面的上方

- Firefox 和 IE 在等待 CSS 传输完成之前不会渲染任何东西。
- 此时有些 CSS 并不是必要的，例如 @media print
- 尽可能早的放置样式到文档中

```
<head>
   <title>My page</title>
   <link href="styles.css" …/>
</head>
<body>
   <!-- content -->
```

将脚本放在页面底部（包括内联的）

- Scripts 阻塞下载
- 浏览器的逻辑：因为脚本可以在任何时间执行 `location.href` 或者 `document.write`，何必下载可能没有用的内容呢？
- 将脚本放在底部可以去除对页面下载的阻塞
- 内联脚本也该如此

```
<!-- content -->
<script src="script.js" …/>
</body>
</html>
```

避免在 CSS 中使用 Expressions

- CSS expression:

```
#content {
  position: absolute;
  left: expression(document.body.offsetWidth+'px');
}
```

- 只有 IE 才允许 JavaScript 存在于 CSS 中
- 它们的执行次数常常比你预期的要多，想下 *onmousemove* 的执行
- 聪明的 expressions 会自我重写

将 javascript 和 css 独立成外部文件

- 有助于缓存，永不过期的策略
- 与其他页面共享
- 但会增加 2 个或更多的 HTTP 请求
- 首页可以考虑直接写在页面内（内联）

减少 DNS 查询

- 浏览器需要将域名映射到一个 IP 地址
- DNS 查询需要时间
- 每个页面 2-4 个域名

压缩 JavaScript 和 CSS 文件 (包括内联的)

- 压缩，但依然用 gzip
- JSMin (用 JavaScript 编写，有一个 php 封装)
- YUI compressor – 也可以压缩 CSS
- 内联样式和脚本也应该压缩

压缩之前

```
/**
 * The dom module provides helper methods for
 *    manipulating Dom elements.
 * @module dom
 *
 */

(function() {
    var Y = YAHOO.util,     // internal shorthand
        getStyle,           // for load time browser branching
        setStyle,           // ditto
        propertyCache = {}, // for faster hyphen converts
        reClassNameCache = {},          // cache regexes for className
        document = window.document;     // cache for faster lookups

    YAHOO.env._id_counter = YAHOO.env._id_counter || 0;
```

Minify: after

```
(function(){var
B=YAHOO.util,K,I,J={},F={},M=window.document;YAHOO.env._i
d_counter=YAHOO.env._id_counter||0;
```

避免重定向

- 增加 web 请求
- 原因：重新启动一个请求

删除重复的脚本引用

- 咄……
- IE 可能会再次加载

配置 ETags

- ETags 是用来帮助缓存的
- 服务器 A 上的一个组件与服务器 B 上的相同组件拥有不同的 ETag
- 配置 ETags 时不要包括文件索引 (inode)
- … 或者索性移除它们并贯彻 “永不过期” 策略

Apache default `FileETag INode MTime Size`

Change to `FileETag None`

缓存 AJAX 请求

- 通过 `XMLHttpRequest` 返回的内容和其他组件是相似的
- 应该使用压缩 gzipped
- 可以被缓存
- `Cache-control: max-age=?`

Part II

After YSlow “A”:
20 more best practices

Part II

tag: server
tag: content
tag: cookie
tag: javascript
tag: css
tag: images
tag: mobile

尽早清除缓冲区

- 让浏览器开始预取组件，当后台服务器繁忙的时候
- PHP 可以使用此方法 `flush()`
- 繁忙的后台 / 轻量的前端时最好的选择

```
    ... <!-- css, js -->
</head>
<?php flush(); ?>
<body>
    ... <!-- content -->
```

- 学习案例：Yahoo! Search

在 AJAX 请求中使用 GET 方法

- GET 可以获取数据
- POST 方法有两个过程 (发送请求 headers，发送请求数据)
- GET 请求在一个 TCP 数据包中 (除非你有很多的 cookies)
- URL 最大长度为 2K (IE 的限制)
- POST 传递数据的方式和 GET 有很大不同
- 学习案例： Yahoo! Mail Research

Part II

tag: server
tag: content
tag: cookie
tag: javascript
tag: css
tag: images
tag: mobile

延迟加载组件

- 问问自己：那些是页面显示时绝对需要的组件
- 余下的组件可以稍后再加载 (比如：拖动，动画，不可见的内容，折叠起来的图片)
- 可以通过 JavaScript 实现分块加载
- 常用工具： YUI Image Loader
- 常用工具： YUI Get Utility

延迟加载组件

- 学习案例：yahoo.com
- onload.js and onload.css
- Progressive enhancement 逐步增强

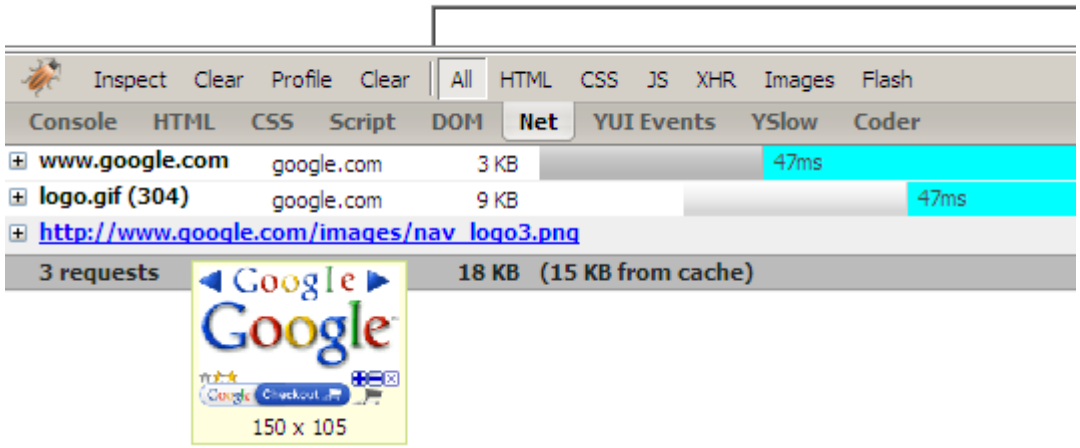
预加载组件

预加载

- 预加载将来会使用到的组件
- 无条件的预加载 (google.com 在页面装载完毕时会下载一个 sprite)
- 有条件的预加载 (search.yahoo.com 当你在输入框中输入时)
- 预先考虑的预加载，在启动重设计前预先加载

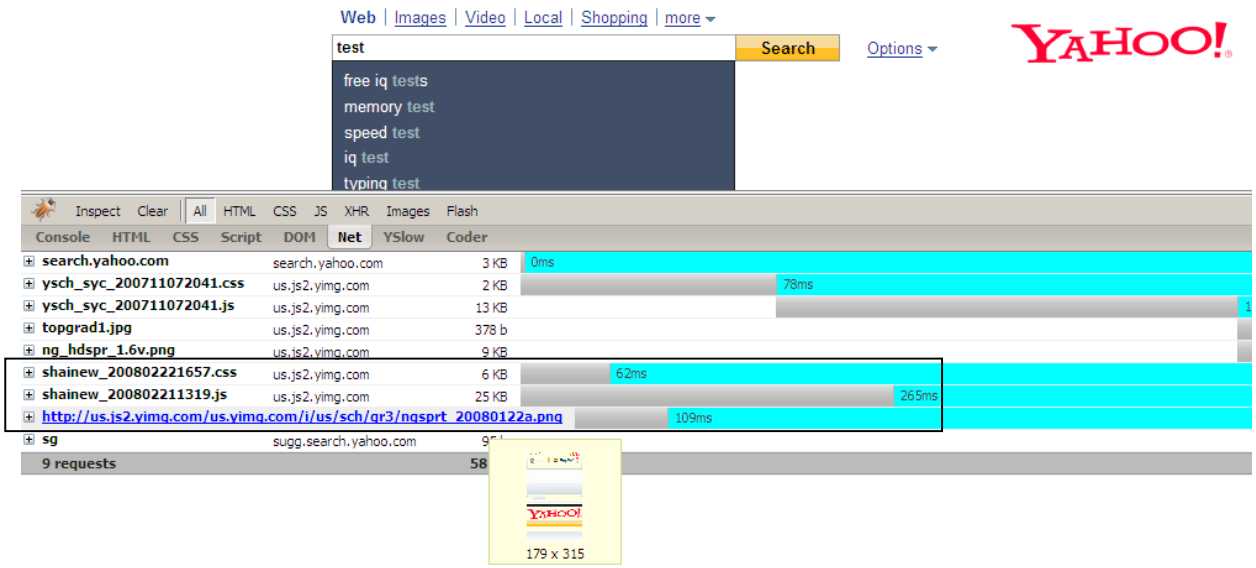
预加载组件 (续上页)

- 无条件的预加载的例子



预加载组件 (续上页)

- 有条件的预加载的例子 search.yahoo.com
- 当你开始输入时，页面可以假定你将进入搜索结果页面
- Time to preload



减少文档中元素的数量

- 世界上最快的页面是什么？`about:blank`!
- 复杂的页面意味着需要下载很多数据
- 这也意味着 JavaScript 操纵 DOM 时会更慢
- 这也意味着可能使用了错误的语义标签 (譬如嵌套的 table，泛滥的 `<div>`)
- 使用有语义的标签
- 使用 YUI 中的 reset.css, fonts.css, grids.css
- 得到文档中元素的数量的简单方法，在 Firebug 的 console 中输入：

```
document.getElementsByTagName('*').length
```

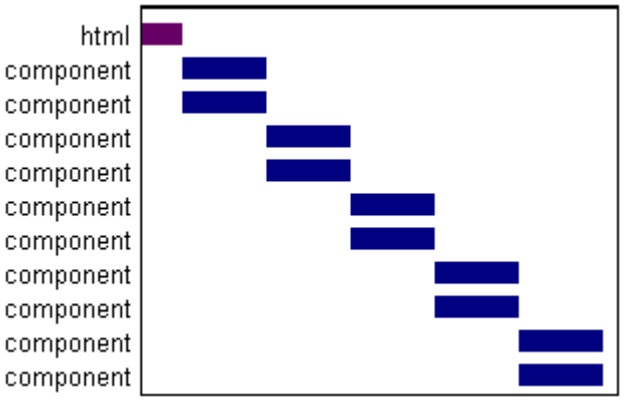
- yahoo.com 是个很繁忙的页面，但是其仍然只有不到 700 个元素 (HTML tags)

跨域分离组件

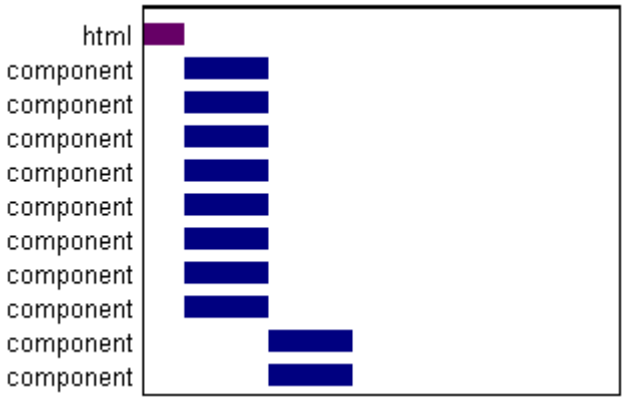
- 最大化的平行下载
- 但是不要超过 2-4 个域，因为 DNS 查找需要开销
- `www.example.org` - HTML 内容
- `static.example.org` - 静态的组件

- 展望：IE8 将允许每个域 6 个请求

跨域分离组件 (续上页)



2 个组件平行加载



8 个组件平行加载

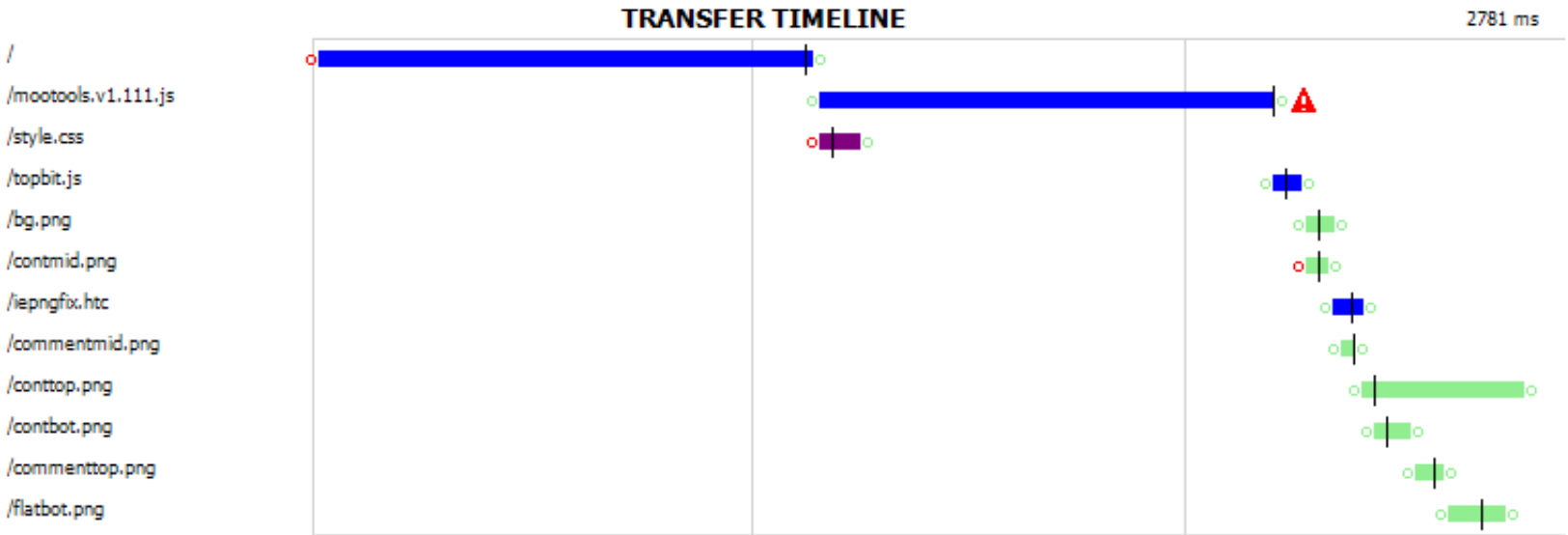
尽量少使用 iframes

- <iframe> 的优点：
  - 能帮助你延后加载第三方的内容。譬如标志和广告。
  - 沙箱安全
  - 可以平行的加载脚本
- <iframe> 的缺点：
  - 它们会造成一些花费，即使它们是空白
  - 阻塞页面的 onload 事件
  - 没有语义

No 404s

- 404 表示没有发现
- 无用的加载
- 一些站点会显示帮助性的 404s 信息“ Did you mean X？ ”
- … 这些会用到些服务器资源 (DB, etc)
- 当一个外部的脚本遭遇 404， 浏览器仍然尝试去解析它以发现一些可用的内容

No 404s（续上页）



第二个组件是一个 404 JavaScript 资源，它阻塞了页面其余组件的加载

Part II

tag: server
tag: content
tag: cookie
tag: javascript
tag: css
tag: images
tag: mobile

减少 cookie 体积

- 除去不必要的 cookies
- 保持 cookies 体积尽可能小，因为会用户影响相应的时间
- 留意 cookies 设置在恰当的域级，这样其他的子域是不受影响的
- 设置合适的有效时间，一个以前的时间或 none 会更快的删除 cookies ，提高用户的响应时间

为组件使用 Cookie-free 的主机

- Option 1: 子域分隔 (static.example.org)
- Option 2: 新的顶级域名 (譬如 yimg.com, ytimg.com, images-amazon.com)
- 代理可能会拒绝缓存
- www.www-yes.org vs www-no.org?
- no-www 让你没有选择，但是可以操作 *.example.org 的 cookies

Part II

tag: server
tag: content
tag: cookie
tag: javascript
tag: css
tag: images
tag: mobile

尽可能减少访问 DOM 的次数

- 访问 DOM 是很影响速度的事情
- 缓存
- “离线”更新节点，然后再把它们添加到 DOM 树中
- 避免使用 JavaScript 去修正布局

开发聪明的事件处理程序

- 不要等待 `onload`，使用 `DOMContentLoaded`
- 事件会往上冒泡，所以可以使用委托 (附加 listeners 到外层容器上)
- 正确清楚以防止 IE 内存泄露
- onresize 缓冲
- 使用 YUI Event utility

Part II

tag: server
tag: content
tag: cookie
tag: javascript
tag: css
tag: images
tag: mobile

选择 <link> 而不是 @import

- CSS 应该放在页面头部
- 在 IE 里， @import 相当于将 <link> 放在页面尾部

避免使用 filters

- IE 私有
- `AlphaImageLoader`
- 修正 IE6 不支持半透明 PNG 图片的问题，IE7 已修复
- 堵塞渲染，造成浏览器假死
- 增加内存消耗
- 每个元素，不要每张图片！

最佳方式：完全避免，使用可兼容的 PNG8

万非得以：使用下划线 Hack(`_filter`)，不要殃及 IE7+ 用户

Part II

tag: server
tag: content
tag: cookie
tag: javascript
tag: css
tag: images
tag: mobile

优化图片

- GIF - 不要使用用不着的调色板
- 使用 PNG 替代 GIF
- “我们反复呼吁： 请给 PiNG 一个机会!"
- pngcrush tool (or optipng, or pngoptimizer)
- 移除 gamma chunks 也有益于跨浏览器的颜色
- 除去注释
- jpegtran - 无损 JPEG 处理，可用来优化和移除注释

优化图片 (contd.)

你可以编写一个简单的工具，在站点发布前遍历你的图片目录并按下面的顺序做：

1. 转换所有的 GIF 成 PNG (并检查是否节约了空间)

```
> convert image.gif image.png
```

2. 压缩所有 PNG

```
> pngcrush image.png -rem alla -reduce
result.png
```

3. 从 JPEG 中剥离注释

```
> jpegtran -copy none -optimize -perfect
src.jpg dest.jpg
```

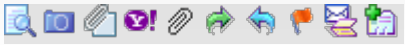
优化图片 (contd.)

- 你会感到惊讶：有多少站点从小变大，却能优化下载大小
- 一个页面的传输包含 200K 无用的图片信息！？

优化 CSS sprites

- 尽可能选择 horizontal 而不是 vertical
- 合并相似的颜色
- 保证颜色数量 (<256) 以适合 PNG8
- “手持设备友好” – 不要留下大的鸿沟
  - 文件没有增大很多，但图片需要被解压成像素分布表
  - 100x100 是 10000 像素
  - 1000x1000 是 1 百万 像素
  - 个例研究：Yahoo! Mail Classic

优化 Sprites



不要在 HTML 中缩放图片

- 下载无谓的字节
- 如果你需要

```
<img width="100" height="100"
  src="mycat.jpg" />
```

那么就让 mycat.jpg 100x100 而不是 500x500

减小体积并缓存 favicon.ico

- www.example.org/favicon.ico
- 必要之恶：
  - 浏览器总会请求它
  - 还是不要返回 404
  - 会发送 Cookie
  - 不能放到 CDN 上
  - 干涉到下载序列
- 让它变小 (<= 1K)
- 动画 favicons 一点也不酷
- 设置 Expires header
- 工具：imagemagick, png2ico
- 个案研究：Yahoo! Search - favicon.ico 占到所有 PV 的 9%

Bonus: crossdomain.xml

- Flash/Flex 的跨域策略
- 放在根路径下：example.org/crossdomain.xml

```
<cross-domain-policy>
  <allow-access-from
  domain="*.yahoo.com" secure="false"/>
</cross-domain-policy>
```

- 设置 Expires 邮件头
- gzip
- … 出于安全考虑，别这么做：

~~<allow-access-from domain="*" />~~

Part II

tag: server
tag: content
tag: cookie
tag: javascript
tag: css
tag: images
tag: mobile

保证组件在 25k 以下

- iPhone 不会 Cache 它们
- 是 *未压缩大小* 小于 25kb
- 除了 JS 和 CSS 外，也尽量减小 HTML

将组件拆包分到多个文档中

- 如果 UserAgent 中宣布支持的话 (iPhone doesn't)
- 附件通过 email 发送

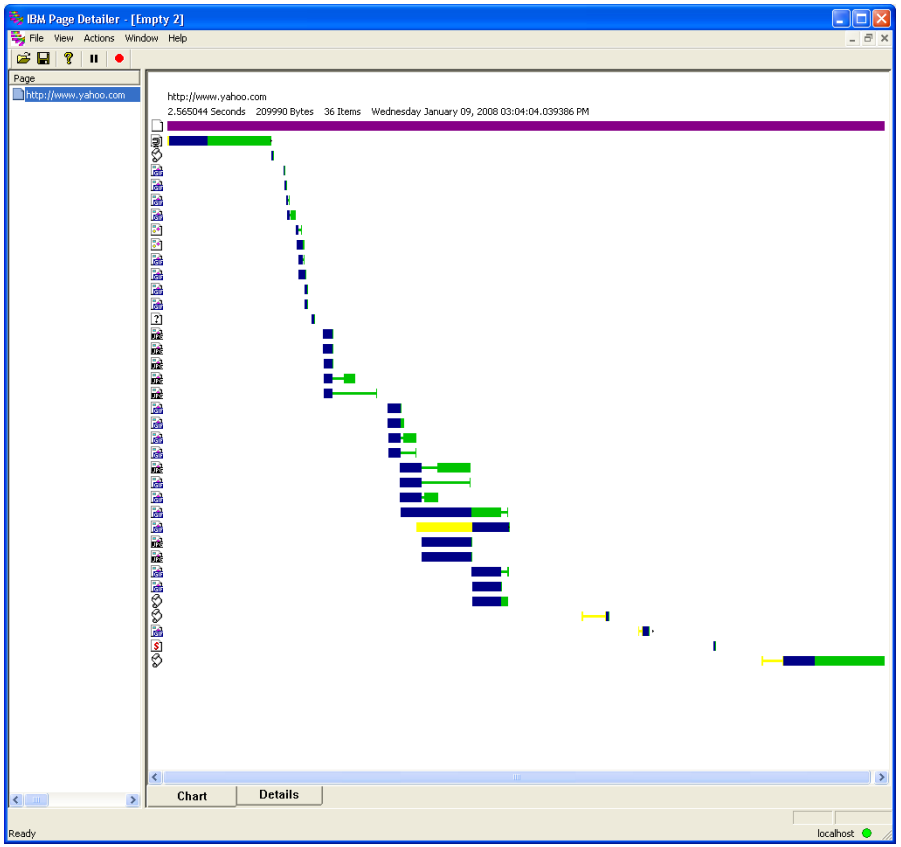
Part II

tag: server
tag: content
tag: cookie
tag: javascript
tag: css
tag: images
tag: mobile

工具

- YSlow (http://developer.yahoo.com/yslow/)
- Fiddler (http://www.fiddlertool.com/fiddler/)
- IBM Page Detailer (http://www.alphaworks.ibm.com/tech/pagedetailer)
- HTTPWatch (http://www.httpwatch.com/)
- AOL Pagetest (http://pagetest.wiki.sourceforge.net/)
- Firebug Net Panel (http://www.getfirebug.com/)

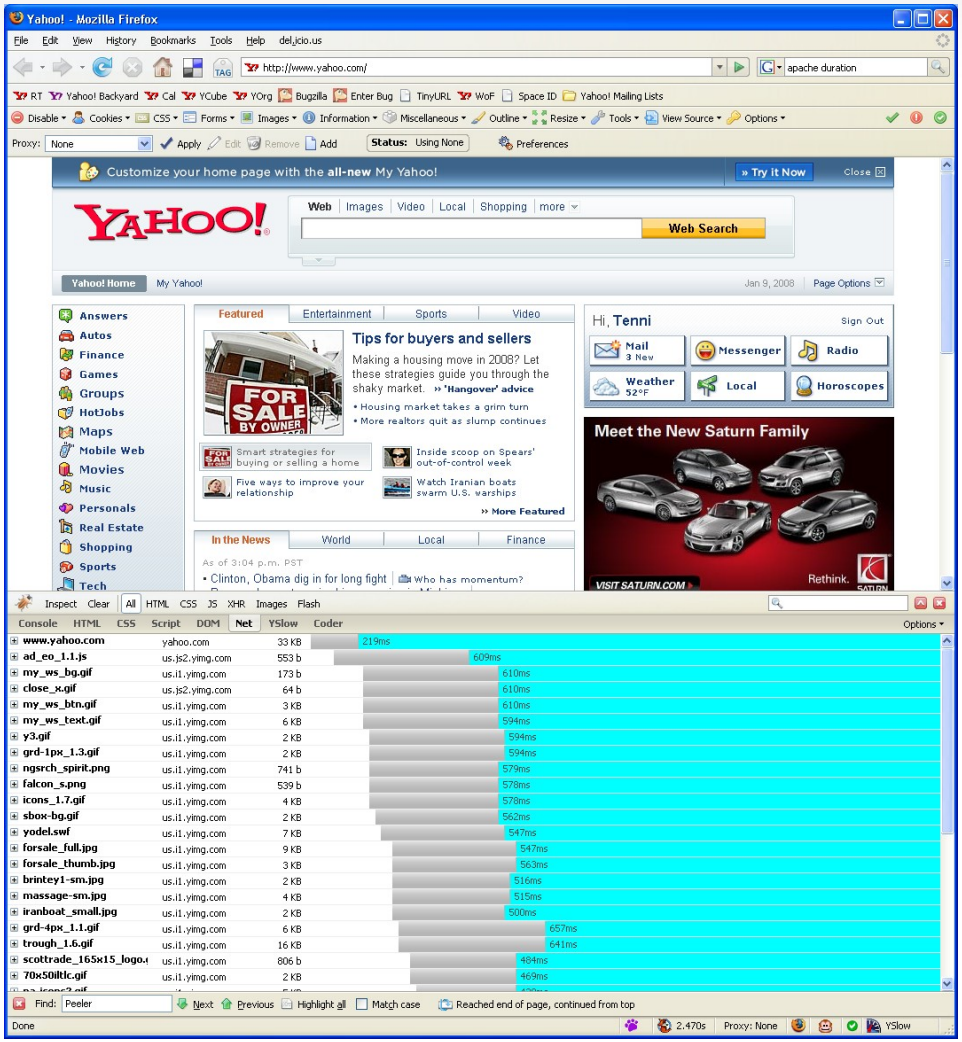
IBM Page Detailer



- Methodology
  - 封包监听器 (Packet Sniffer)
- 优势
  - 非常精确
  - 提供详细数据
  - 所有 browser 下都可以工作
  - 最佳的瀑布试图
- 缺点
  - 需要下载
  - 90 天免费试用
  - 只能在 Windows 下运行
  - 缺少被 Cache 的组件

Firebug NET Panel

- Methodology
  - 封包监听器 (Packet Sniffer)
- 优势
  - 集成在 Firebug 中
  - 瀑布视图
  - 提供 HTTP 头信息
- 缺点
  - 只能运行在 Firefox 中
  - 不精确的瀑布视图
    - 无渲染时间
    - 无解析时间
    - 无转向区别标记
    - 无 DNS 查找记录
  - 不显示已被 Cache 的组件（1.1 提供）



URLs – Exceptional Performance

YUI blog

http://yuiblog.com/blog/category/performance/

YDN (Yahoo Developer Network)

http://developer.yahoo.com/performance/

YDN blog

http://developer.yahoo.net/blog/archives/performance/

Mailing list (Yahoo! Group)

http://tech.groups.yahoo.com/group/exceptional-performan

Feedback

http://developer.yahoo.com/yslow/feedback.html

URLs (contd.)

- "When the Cookie Crumbles" Tenni Theurer, Steve Souders
  http://yuiblog.com/blog/2007/03/01/performance-research-part-3/
- "Maximizing Parallel Downloads in the Carpool Lane", Tenni Theurer, Patty Chi
  http://yuiblog.com/blog/2007/04/11/performance-research-part-4/
- YUI Image Loader (http://developer.yahoo.com/yui/imageloader/)
- YUI Get (http://developer.yahoo.com/yui/get/)
- YUI Compressor (http://developer.yahoo.com/yui/compressor/ contains a Java port of an internal PHP CSS minifier tool written by Isaac Schlueter, http://foohack.com/)
- JSMin (http://www.crockford.com/javascript/jsmin.html)
- "High-performance AJAX applications" Julien Lecompte
  http://yuiblog.com/blog/2007/12/20/video-lecomte/
- Yahoo! engineer Michael J. Radwin talk back in 2004
  http://www.radwin.org/michael/talks/

致谢 – thank you!

请牢记

- 关注前端（ Focus on the front-end ）
- 由浅入深，循序渐进（ Harvest the low hanging fruit ）
- 为用户着想（ Be an advocate for your users ）
- 马上作，作精彩（ Start early ）

Thank you!

Merci beaucoup!