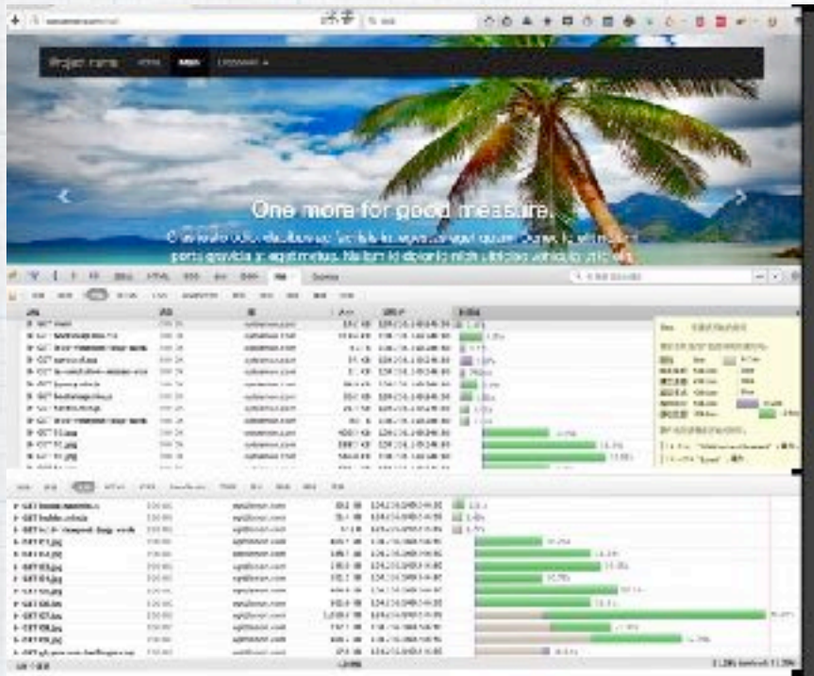
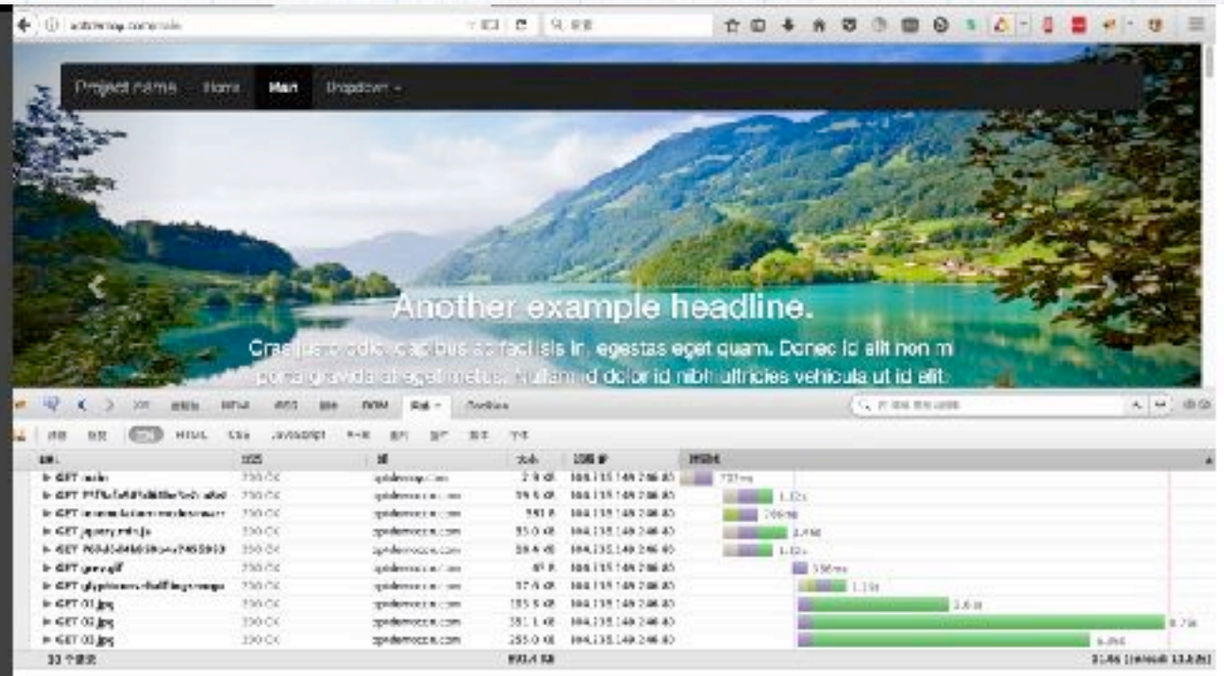
web性能优化

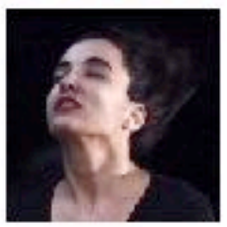
前后端性能优化





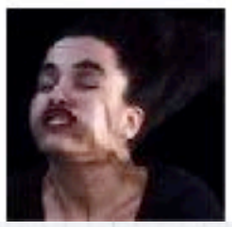
优化前

主html请求(14.2KB)
主html请求响应时间:1.37s
主页面DomReady时间:4.91s
全页面load时间:50.03s
总请求数19个
总内容大小4.9MB



优化后

主html请求(2.9KB)
主html请求响应时间:0.792s
主页面DomReady时间:2.71s
全页面load时间:11.62s
总请求数10个
总内容大小891.4KB



先做广告

看疗效!

优化前后

* 主html请求(14.2KB)
* 主html请求响应时间:1.37s
* 主页面DomReady时间:4.91s
* 全页面load时间:50.03s
* 总请求数19个
* 总内容大小4.9MB

* 主html请求(2.9KB)
* 主html请求响应时间:0.792s
* 主页面DomReady时间:2.71s
* 全页面load时间:11.62s
* 总请求数10个
* 总内容大小891.4KB

优化后约1/4的内容带宽占用

约1倍的响应速度、1/2的请求数

概览

* 广告
* web页面呈现的处理流程
* 服务端优化
* 输出内容优化
* 前端优化
* 架构优化
* Q&A

参考资料

http://www.ruanyifeng.com/blog/2014/02/ssl_tls.html

http://www.cnblogs.com/lovesong/p/5186200.html

http://blog.csdn.net/whuslei/article/details/6667471

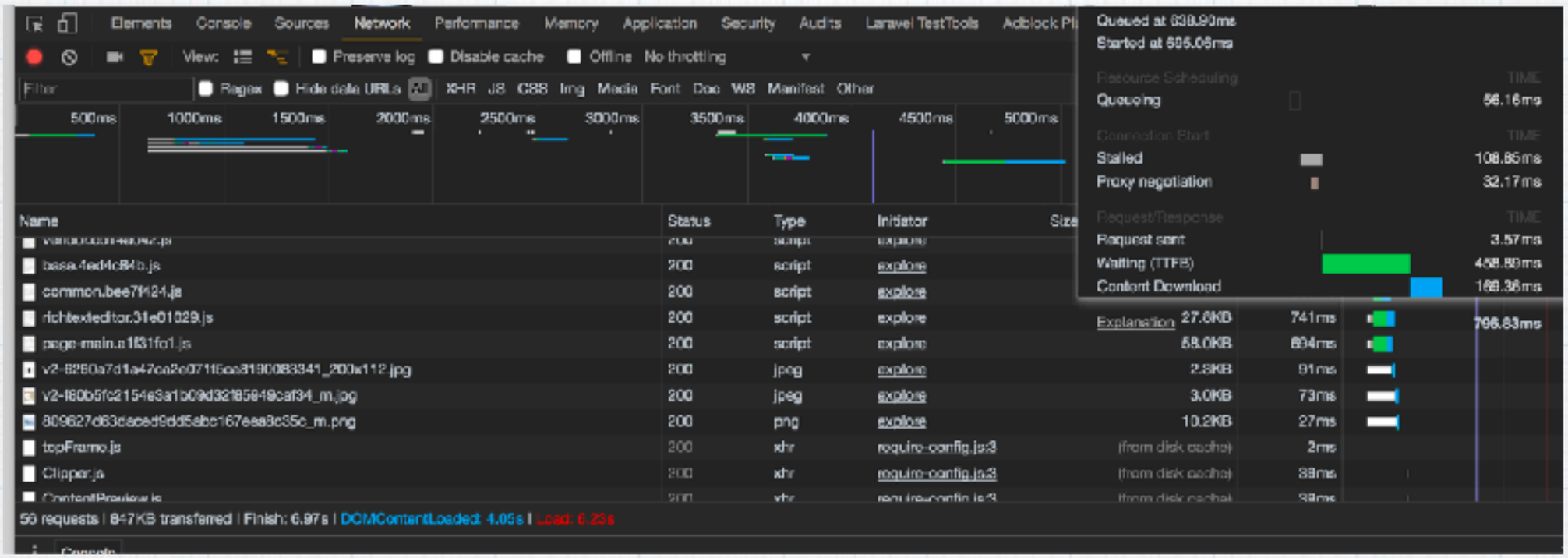
http://www.jianshu.com/p/e305ace24ddf

http://coolshell.cn/articles/9666.html

https://segmentfault.com/a/1190000005169412

https://www.slideshare.net/Fonkie/web-21553879

web服务问题

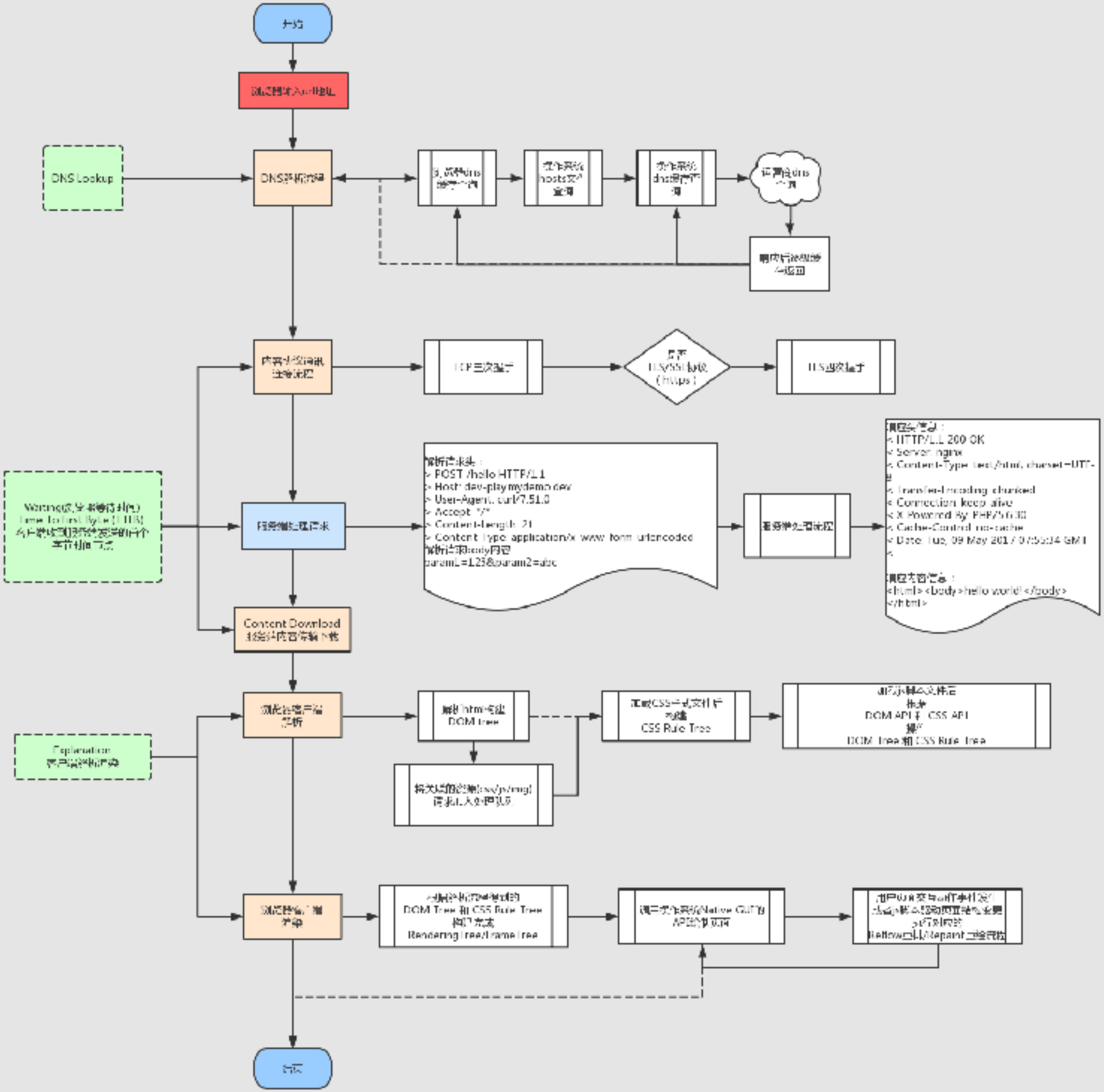


日常的web浏览器访问中：

- HTML主页面的内容传输时间往往只占整个页面响应时间的10~20%
- 80~90%的时间是用来下载页面关联的图片、css样式、js脚本文件

web页面呈现的处理流程

* dns解析
* 建立连接(tcp协议三次握手)
* 服务端处理请求
* 服务端内容输出
* 客户端内容渲染


DNS Lookup
Content Download
POST /hello HTTP/1.1
Host: dev-play.mydemo.dev
User-Agent: curl/7.51.0
Accept: */*
param1=123&param2=abc
HTTP/1.1 200 OK
Server: nginx
Connection: keep-alive
Cache-Control: no-cache
<html><body>hello world!</body></html>
DOM Tree
CSS Rule Tree
Explanation
DOM Tree 和 CSS Rule Tree
Rendering Tree

服务端优化

* 使用Content Delivery Network (CDN)服务
* 添加缓存控制头信息(Cache-Control以及ETags或Last-Modified头信息)
* 压缩传输内容
* 使用CookieFree的域名用于静态资源文件
* 减少Cookie的尺寸
* 尽早输出缓冲区内容
* 可能的情况下使用GET做ajax请求
* 避免空的src的图片请求

缓存控制头信息

* Cache-Control 缓存控制指令(HTTP/1.1)
* Date+Expires 控制缓存的失效日期，优先级比Cache-Control低 (HTTP/1.0)
* Etag和If-None-Match 或 Last-Modified和If-Modified-Since (HTTP/1.1)

输出内容优化

* 减少http请求数量
    * 合并css,js文件
    * 对浏览器兼容性要求不高的情况下可以使用data: URL scheme
* 减少DNS查找
* 避免重定向
* 把ajax请求设计为可缓存的模式
* 延迟加载组件
* 预加载组件
* 减少DOM元素数量
* 把组件分散到不同的域名
* 减少iframe的数量
* 尽量避免无效的404请求

前端优化

* 把样式文件放在页面头部
* 把脚本文件放在页面底部
* 避免CSS表达式
* 能用<link>外链css的就尽量不要用@import
* 尽可能避免使用CSS滤镜Filter
* 外置js和css文件
* 最小化(Minify)JS和CSS文件内容
* 移除重复的脚本
* 最小化DOM访问
* 优化事件处理方法

前端优化

* 优化图像尺寸
* css Sprites 技术整合小图像文件
* 尽可能不要拉伸图像在HTML中
* 优化favicon.ico图标文件并让其可缓存
* 尽可能让资源文件小于25k
* 打包组件到一个Multipart文档

集群/多节点优化 - 架构优化

* 负载均衡（Load Balance）
* 分布式存储(Distributed Storage)
* 分布式缓存(Distributed Cache)
* 分布式计算(Distributed Computing)
* 软件定义网络SDN(Software-Defined Networking)

Demo

optdemon.com

optdemoy.com

服务端优化：
1. 无必要的页面去除session/cookie信息
2. 页面尽可能做http-cache、etag
3. 启用gzip压缩内容

内容优化
1. cdn ugc数据和静态资源域名
2. css 内容压缩,js 内容压缩
3. img 图片编辑,img 图片优化
4. 资源请求请求合并
5. 延迟加载不必要的内容
6. 预加载后续页面的资源

Demo回顾

* 无必要的页面去除session/cookie信息
* 页面尽可能做http-cache、etag
* 启用gzip压缩内容
* cdn ugc数据和静态资源域名
* css 内容压缩,js 内容压缩
* img 图片编辑,img 图片优化
* 资源请求请求合并
* 延迟加载不必要的内容
* 预加载后续页面的资源

Q&A

相关资源

* https://github.com/nickfan/laraveloptdemo

谢谢

阿熊
<nickfan81@gmail.com>
https://github.com/nickfan