設計段階から実装まで、今すぐ始める高速化

CSS Nite LP, Disk 23

こもりまさあき

自己紹介を…

こもりまさあき　http://protean.im

1990年代前半に都内のDTP系デザイン会社にてアルバイトをはじめる。大学卒業後そのまま正社員となり、入出力業務、デザイン業務、ネットワーク関連業務に並行して従事。2001年、会社を退職しフリーランスの道へ。
案件ごとに業務内容や立ち位置が異なるため、職域的な肩書きはなし

近著に『レスポンシブ・ウェブデザイン標準ガイド（MdN刊）』
『WordPress 高速化＆スマート運用必携ガイド（共著・MdN刊）』、など

Twitter: @cipher / @proteanbm
Facebook: gaspanik
Instagram: @cipher

Webサイトが遅い原因の80%は、フロントエンドの処理だと言われる

「フロントエンドの処理」って、実装する人だけが考えること？

いやいや、ちょっと待って。
実装前から考える方が良くない？

今日は、そんな設計段階から
実装に関わる部分までの話を

バックエンドの
話もまとめて

今日は、そんな設計段階から
実装に関わる部分までの話を

これからお話しすること

- コンテンツのロードを速くしよう
- キャッシュを使いこなそう
- 外部要因による遅延を改善しよう
- リクエストを分散させてより高速に

キレイなサイト、面白いサイト、完成すればそれで良いのかな？

ブロードバンドの普及、
スマートデバイスの登場

相反する回線環境

ネットワークの仕組みを理解し、より速く配信することを考えよう

表示が速すぎても、誰も文句は言いません

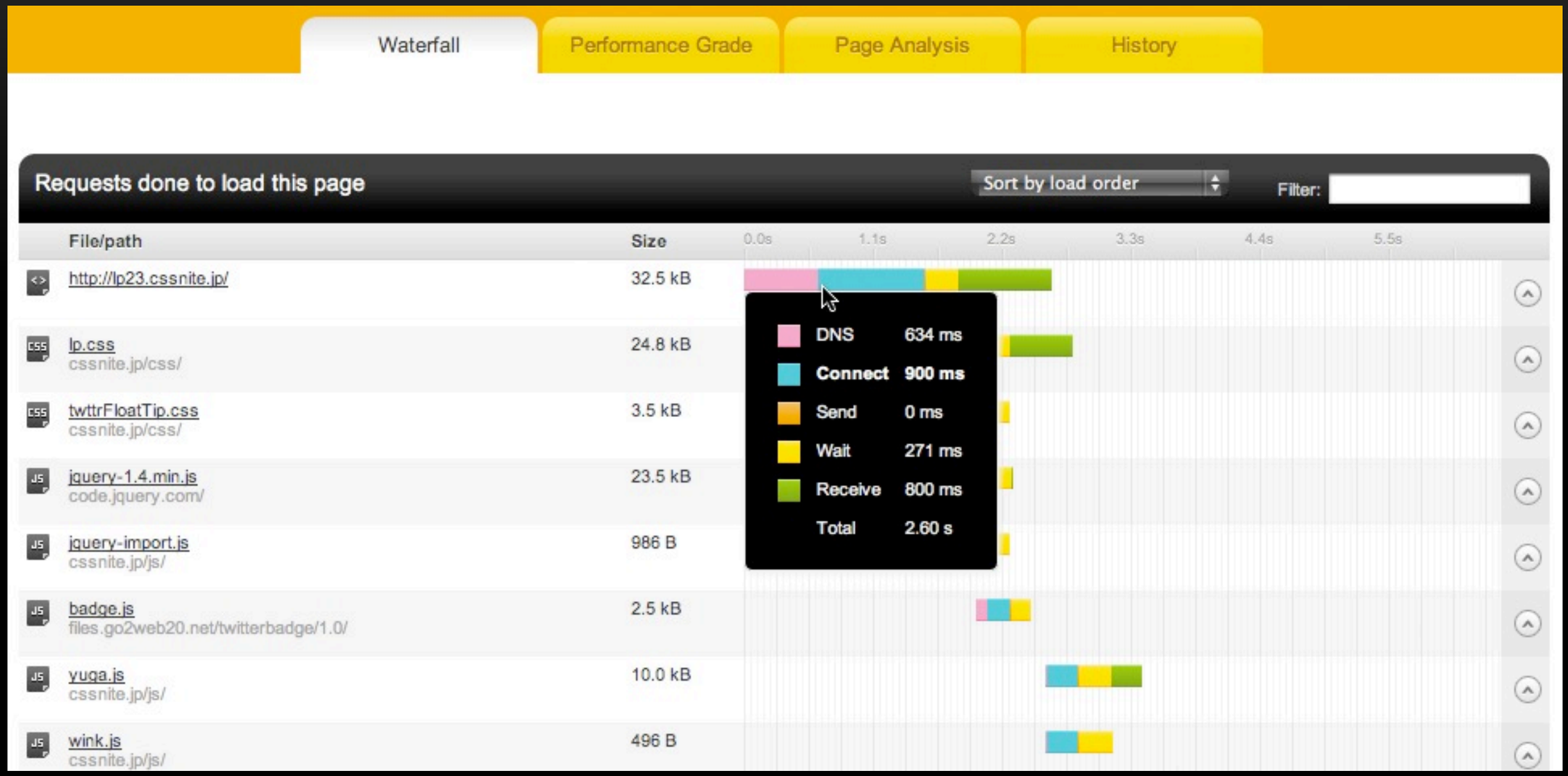
目には見えない部分を改善する、
それもデザインのひとつですから

みんなで考えましょうよ

誰のためのWebサイトなのか

コンテンツのロードを速くするためにできること

コンテンツがロードされる様子、アゲイン



Pingdom Tools: http://tools.pingdom.com/fpt/

箱作りのベースは速く、とにかく速く

- HTML
- CSS
- JavaScript

周りを見渡せば、意外と多いこんな状況

- コンテンツ量が多く、
  HTMLのサイズそのものが大きい
- 制作効率、メンテナンス効率、
  そこを重視するあまり、
  分割されすぎたCSSファイル
- あれもこれもと追加されたJavaScript

@import、えっと…

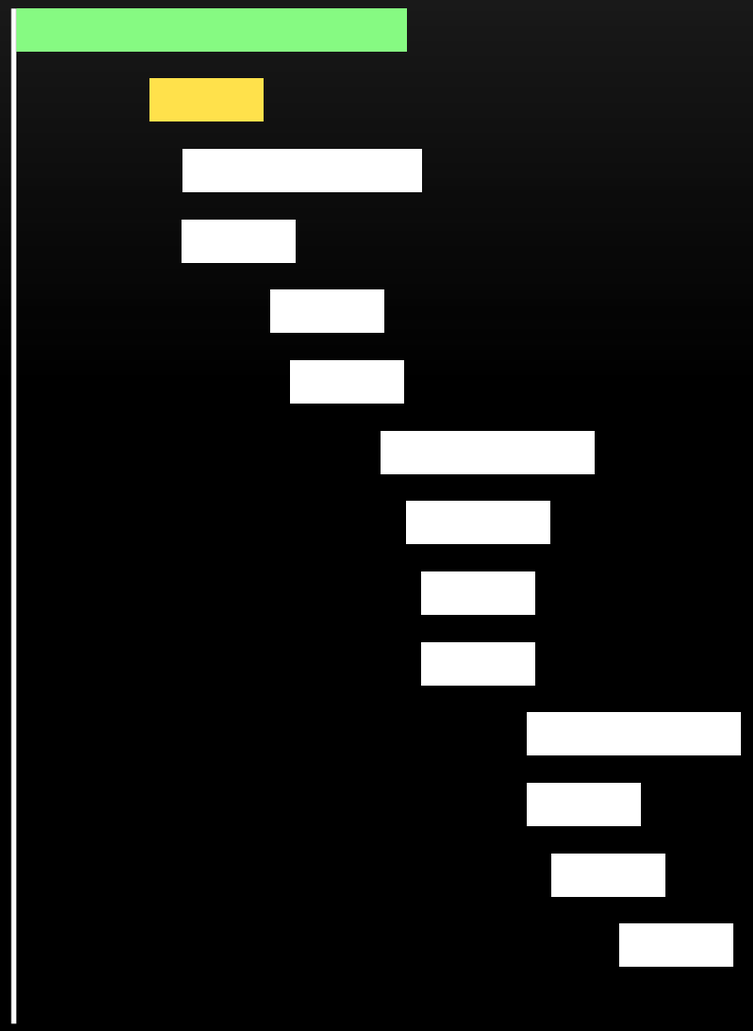
```
<link rel="stylesheet"
    … href="import.css">
```

……➤ import.css

```
@import url(contents_base2.css);
@import url(base.css);
@import url(layout.css);
@import url(common.css);
@import url(menu.css);
@import url(contents_base.css);
@import url(search.css);
@import url(top.css);
@import url(news.css);
@import url(chmn.css);
@import url(map.css);
@import url(faq.css);
@import url(logo.css);
@import url(background.css);
```

これだと、こういう結果になる

example.html
import.css
base.css
layout.css
common.css
menu.css
contents_base.css
search.css
top.css
news.css
chmn.css
map.css
faq.css
logo.css
…



当たり前の話

リクエストとレスポンス、同時接続数

- クライアントとサーバのやりとり
- ブラウザの同時接続数
- 初期段階のロードから
  コンテンツの整形までを速くするには、
  この部分に着目することが大事

いかにして速くブラウザに届けるか

ファイルをまとめる、余分な要素は除去

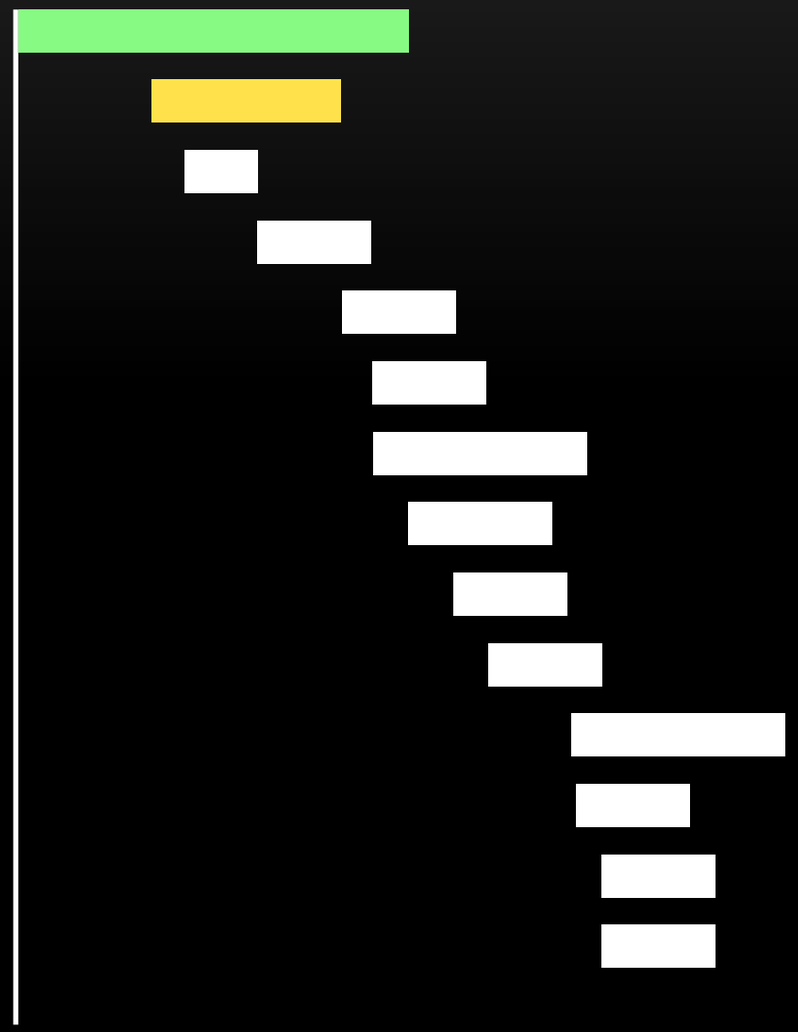
- 分割されすぎたCSS、JavaScriptは、
  可能な限りまとめる
- ファイルに含まれる余分な要素、
  たとえば、改行やコメントなどを除去する

ファイルをまとめる、余分な要素は除去

- 分割されすぎたCSS、JavaScriptは、
  可能な限りまとめる
- ファイルに含まれる余分な要素、
  たとえば、改行やコメントなどを除去する

さっきの@importもまとめてしまえば

example.html
compact.css
jQuery.js
plugin.js
imageA.png
imageB.png
imageC.png
imageD.png
imageE.png
imageF.png
imageG.png
imageH.png
imageI.png
imageJ.png
…



余分なリクエストは減り、
その分だけ速く他の要素へ

でも効率が…、手間だし…
嫌がる人が多いのも事実

ツールやサービスを使えば、比較的簡単

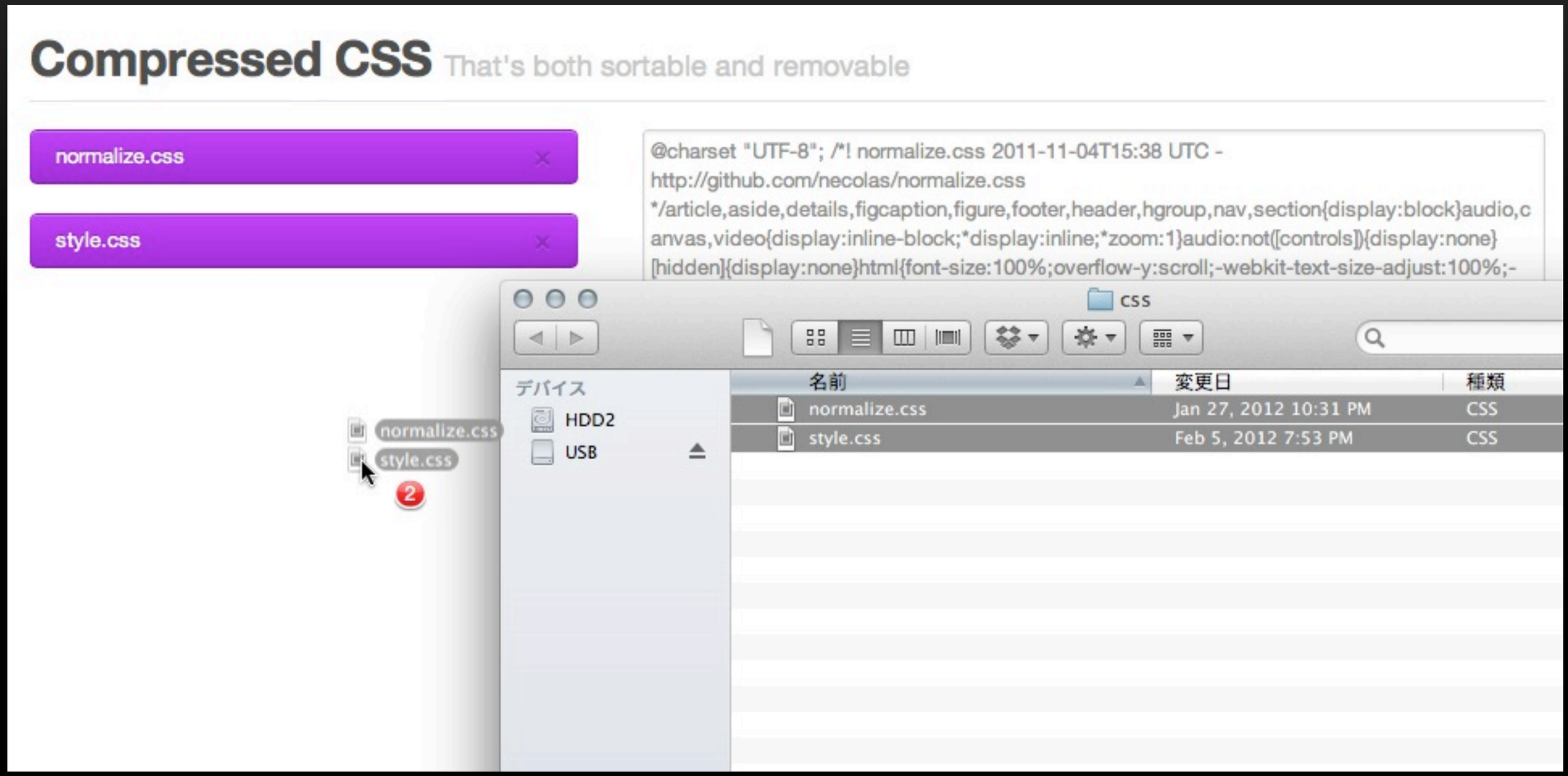
- excssive
  http://www.excssive.com/
- YUI Compressor
  http://developer.yahoo.com/yui/compressor/
- Closure Compiler Service
  http://closure-compiler.appspot.com/home

ツールやサービスを使えば、比較的簡単

- **excssive**
  http://www.excssive.com/
- **YUI Compressor**
  http://developer.yahoo.com/yui/compressor/
- **Closure Compiler Service**
  http://closure-compiler.appspot.com/home

CodeKit: http://incident57.com/codekit/

excssive なら画面内にファイルをドラッグ



運用体制をどうするか？の話でしかない

「それでも面倒」と言うなら、
テキストデータを圧縮しちゃえ

テキストをGzip圧縮して高速化

- バックエンドのサーバ側で
  Gzipを使った符号化をおこなう
- 1/5〜1/3ぐらいにファイルサイズは小さく
- 特に不安定な回線に対して有効

Apacheの場合はこのように

- 2.x系なら、mod_deflateを有効にする

```
<IfModule mod_deflate.c>
 SetOutputFilter DEFLATE
 BrowserMatch ^Mozilla/4 gzip-only-text/html
 BrowserMatch ^Mozilla/4\.0[678] no-gzip
 BrowserMatch \bMSI[E] !no-gzip !gzip-only-text/html
 SetEnvIfNoCase Request_URI \.(?:gif¦jpe?g¦png)$ no-gzip dont-vary
 Header append Vary User-Agent env=!dont-vary
</IfModule>
```

mod_deflateの記述例: https://httpd.apache.org/docs/2.2/mod/mod_deflate.html

転送データサイズはここまで変わる

Home | Grade | Components | Statistics | Rulesets: YSlow(V2)

Components

The page has a total of 17 components and a total weight of 274.8K bytes

↑ TYPE	SIZE (KB)	GZIP (KB)	COOKIE RECEIVED (bytes)	COOKIE SENT (bytes)	HEADERS	URL	EXPIRES (Y/M/D)	RESPONSE TIME (ms)	
− doc (1)	27.5K								
doc	27.5K	8.6K		1089		http://blog.gaspanik.com/	2012/6/29	0	
− js (4)	88.0K								
js	31.1K					http://include.reinvigorate.net/re_.js	no expires	0	
js	50.0K	15.5K				http://static.gaspanik.com/js/sm.js	2013/6/29	0	
js	6.8K	3.1K		34		http://static.chartbeat.com/js/chartbeat.js	no expires	0	"d04742eb97
js	0.02K	0.04K		1089		http://blog.gaspanik.com/mint/?...	1997/7/26	0	
− css (1)	9.0K								
css	9.0K	2.6K		1089		http://blog.gaspanik.com/style.css	2013/6/29	0	
+ cssimage (5)	6.3K								

HTMLをピックアップ

27.5k → 8.6k

Gzipしない元のファイル

Gzip後の転送サイズ

HTMLをピックアップ

27.5k

Gzipしない元のファイル

→

8.6k

Gzip後の転送サイズ

小さくなれば、より速く届く

もっと簡単にいろいろできないの？

Google謹製、mod_pagespeedの導入

- Apacheのモジュール「mod_pagespeed」
  https://developers.google.com/speed/pagespeed/mod
- さまざまな高速化の施策をサーバ側で
- サーバ側の負荷、動作確認など、
  事前に徹底的な検証が必要

繰り返します

箱作りのベースは、とにかく速く

方法はいろいろ。
どこまでやるか、
どうやってやるか

必要なファイルだけを配信。キャッシュを使おう

もう一度おさらい、リクエストとレスポンス

- 構成要素の数だけ繰り返される
  リクエストとレスポンス
- サイトにアクセスするたびに
  ファイルをダウンロードさせるのは非効率
- 変更のないファイルは、
  ダウンロードさせたくない

キャッシュといってもいろいろある

- ブラウザが持っているキャッシュ
- アプリケーションキャッシュ
- サーバサイドによるキャッシュ
- その他、DNSのキャッシュなど、いろいろ

キャッシュ対象になるもの

- 変更頻度が極端に少ない
  サイト内で使い回す画像ファイル
- 定期的な改修でしか触らない
  CSS、JavaScript
- jQueryなどのライブラリ

ファイルに有効期限を設定するには

- Apacheなら、mod_expires を有効にする

```
<ifModule mod_expires.c>
ExpiresActive On
ExpiresDefault "access plus 1 seconds"
ExpiresByType image/x-icon "access plus 1 years"
ExpiresByType image/jpeg "access plus 10 years"
ExpiresByType image/png "access plus 10 years"
ExpiresByType text/css "access plus 1 years"
…
</ifModule>
```

ファイルに有効期限を設定すると

SIZE (KB)	GZIP (KB)	COOKIE RECEIVED (bytes)	COOKIE SENT (bytes)	HEADERS	URL	EXPIRES (Y/M/D)	RESPONSE TIME (ms)	
27.5K								
88.0K								
31.1K					http://include.reinvigorate.net/re_.js	no expires	0	
50.0K	15.5K				http://static.gaspanik.com/js/sm.js	2013/6/29	0	
6.8K	3.1K		34		http://static.chartbeat.com/js/chartbeat.js	no expires	0	"d04742eb975012ec5
0.02K	0.04K		1089		http://blog.gaspanik.com/mint/?...	1997/7/26	0	
9.0K								
9.0K	2.6K		1089		http://blog.gaspanik.com/style.css	2013/6/29	0	
6.3K								
0.1K					http://static.gaspanik.com/img/kubrickbgcolor.pn	2022/6/27	0	
0.1K					http://static.gaspanik.com/img/kubrickbgwide.pn	2022/6/27	0	
4.0K					http://static.gaspanik.com/img/kubrickheader.png	2022/6/27	0	

結果として、HTTPリクエストは減る

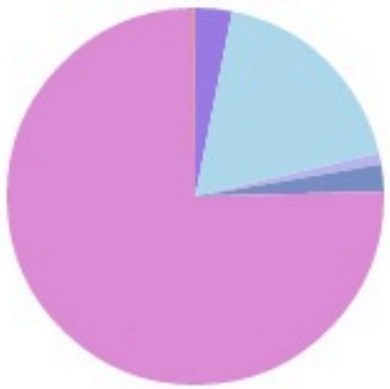
Home | Grade | Components | Statistics | Rulesets YSlow

Statistics

The page has a total of **17** HTTP requests and a total weight of **274.8K** bytes with empty cache

WEIGHT GRAPHS

Empty Cache

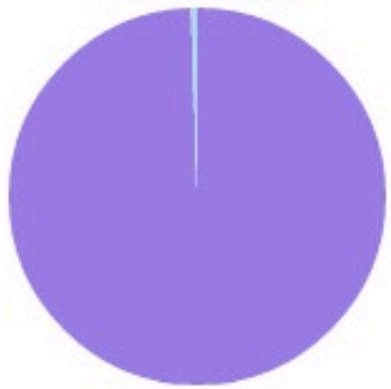


HTTP Requests – 17

Total Weight – 274.8K

1	HTML/Text	8.6K
4	JavaScript File	49.9K
1	Stylesheet File	2.6K
5	CSS Image	6.3K
5	Image	207.0K
1	Favicon	0.2K

Primed Cache



HTTP Requests – 4

Total Weight – 8.7K

1	HTML/Text	8.6K
3	JavaScript File	0.04K

もう一度、全文掲載

```
<ifModule mod_expires.c>
 ExpiresActive On
 ExpiresDefault "access plus 1 seconds"
 ExpiresByType image/x-icon "access plus 1 years"
 ExpiresByType image/vnd.microsoft.icon "access plus 1 years"
 ExpiresByType image/jpeg "access plus 10 years"
 ExpiresByType image/png "access plus 10 years"
 ExpiresByType image/gif "access plus 10 years"
 ExpiresByType text/css "access plus 1 years"
 ExpiresByType text/javascript "access plus 1 years"
 ExpiresByType application/x-javascript "access plus 1 years"
 ExpiresByType text/html "access plus 600 seconds"
 ExpiresByType application/xhtml+xml "access plus 600 seconds"
</ifModule>
```

コピペするだけ、時間にして2分

もうひとつのキャッシュ、.appcache

- 通常のキャッシュとは別枠。
  指定したファイルをキープしてくれる
- サイズの大きい画像、
  jQueryなどのライブラリなど
- 強制的にネットワークを参照させることも可

.appcacheを有効にして、ファイルを列挙

- キャッシュ・マニフェストを用意

```
CACHE MANIFEST
# version 2
CACHE:
index_Resources/PIE.htc
index_Resources/TopImage_Bg.jpg
…
NETWORK:
index.html
index_Resources/HYPE.js
```

.appcacheが有効になっている状態

Manifest: http://gaspanik.com/[redacted]

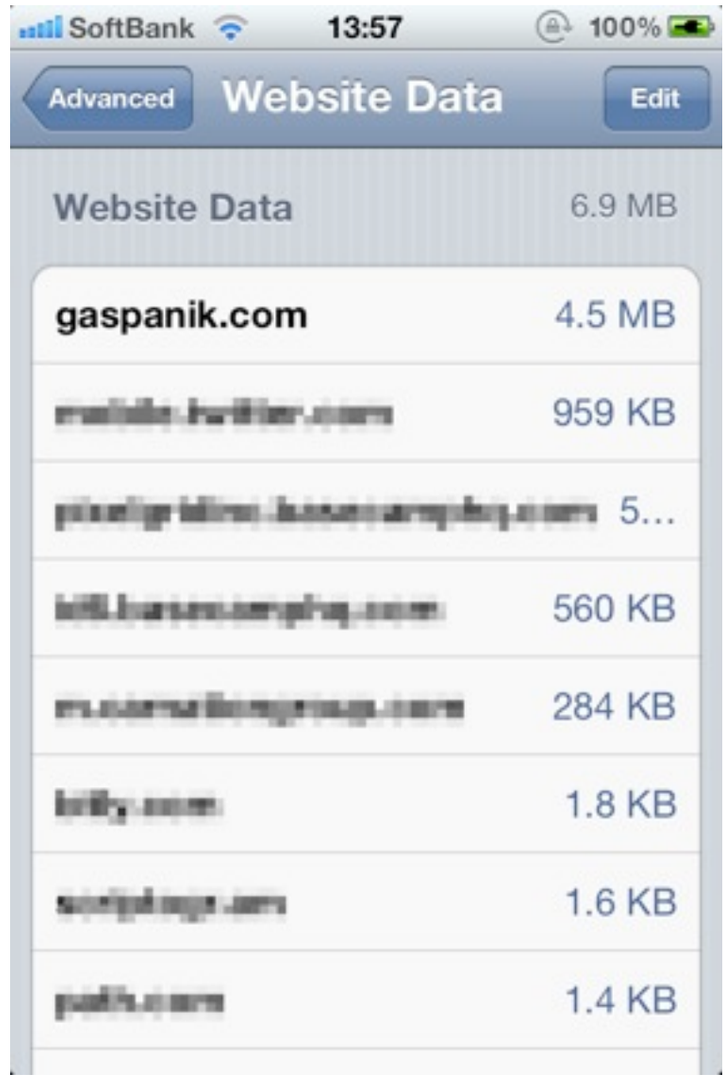
Remove View Entries

- Size: 1.3 MB
- Creation Time: 2012年3月7日水曜日10:00:58
- Last Update Time: 2012年3月7日水曜日10:00:58
- Last Access Time: 2012年4月3日火曜日18:59:30

Manifest: http://gaspanik.com/[redacted]

Remove View Entries

- Size: 264 kB
- Creation Time: 2012年3月7日水曜日9:47:21



chrome://appcache-internals/

変更の少ないファイルは、できるだけ配信しない工夫

さらに？ キャッシングサーバの導入

- 大量のアクセスを処理するなら、
  キャッシングサーバを追加して
  静的なファイルだけはそこ経由で配信する
- Apache + Varnish
  https://www.varnish-cache.org/
- Nginx によるリバースプロキシ
  http://wiki.nginx.org/NginxJa

大量に配信するものは、
より高速なサーバで処理

自身の環境に合わせて、
キャッシュをうまく使って

ご利用は計画的に

外部要因による遅延を極力抑えよう

自サイトのコンテンツ以外の要素が問題

- 効果測定のためなど、
  外部サービスのタグ（JavaScriptのコード）
- サイト内にあれこれ貼られる広告
- あれもこれもと追加した
  ソーシャルメディア系のリソース群

大人の事情で減らせない…

外部の要素を取り込むことで起こる問題

- 外部のサービスが止まった場合、
  最悪コンテンツのロードが途中で止まる
- JavaScriptの
  document.write()
- 外部ドメインに接続する時、
  起こっているのは、DNSルックアップ

document.writeでSCRIPTを書き出すなやで！: http://t32k.me/mol/log/dont-docwrite-scripts/

DNSルックアップは、極力減らすが吉

- 接続先がわからなければ、
  初回はこのDNSルックアップが起きる
- できる限り減らしておいた方がいいもの
- 理想をいえば、4つぐらい

DNSプリフェッチの導入という手も

- 接続する予定のドメインは、
  あらかじめDNSに問い合わせさせておこう
- 有効になる、ならないブラウザがある
- link要素を使って、参照するドメインを記述
  https://github.com/h5bp/html5-boilerplate/wiki/DNS-Prefetching

無駄を省いて、できる限り外部ドメイン参照を減らす

外部要因といえば、もうひとつ

貼り付けた後、放置してない？

- Google Analyticsのコード
- Twitterのボタン、Facebookの all.js
- その他、
  ソーシャル系サービスのJSの読み込み

放置はよくない、定期的にチェック

- 最近のコードは、
  非同期でロードするように改良
- 微妙な変更なども入れると、
  貼り込み用のコードは頻繁に変わっている
- 定期的に見直しを

コンテンツロードの阻害要因は、
できるだけ解消しないと止まる

減らせないリクエスト。配信サーバを分けてしまう

HTTPリクエストを減らせと言われても…

- コンテンツはこれ以上少なくできない
- 制作効率とかコストを考えた場合、
  ファイルの結合などもできれば避けたい
- 画像も使わざるを得ない

コンテンツは減らせない、ならどうする？

- 同じホスト、
  同じネットワーク、
  それでまかなうから限界が出てくる
- それを何とかするには…

狭い道に車が押し寄せたら
当然のように渋滞が起きる

ならば、バイパスを作るか、もっと速い高速道路を作るか

サーバを分割し、配信を効率化する

- サイトで頻繁に使う画像だけ別サーバに
- サブドメインでも分けないよりはマシ
- コンテンツ内の画像は、
  Flickrなどの外部サービスを使ってみる

たとえば、みんな大好きjQuery

- Googleなどから読み込むように変更
- これで自サイトの接続数は、1本減ることに
- 当たり前だけど、やっぱり速い

Google Libraries API

Make the Web Faster +1 1.7k

Overview
- PageSpeed
- Public DNS
- Hosted Libraries
  - Overview
  - Developer's Guide
  - Terms of Service
- Protocols
- Standards
- Performance Best Practices
- Articles

Tools

Community

What is the Google Libraries API?

The Libraries API is a content distribution network for the most popular, open-source JavaScript libraries. The Libraries API provides access to a growing list of the most popular, open-source JavaScript libraries, including:

- Chrome Frame
- Dojo
- Ext Core
- jQuery
- jQuery UI
- MooTools
- Prototype
- script.aculo.us
- SWFObject
- WebFont Loader

Google works directly with the key stake holders for each library effort and accepts the latest stable versions as they are released. Once we host a release of a given library, we are committed to hosting that release indefinitely.

How do I start?

- Check out the developer's guide to get t on your website or application.

Want to talk to other developers using this A discussion group. You can also interact with channel.

Google Developers: https://developers.google.com/speed/libraries/

外部からjQueryを読み込んでみる

- フォールバックも忘れずに

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/バージョン/
jquery.min.js"></script>
<script type="text/javascript">
window.jQuery ¦¦ document.write(unescape('%3Cscript
src="js/jquery/jquery-バージョン.min.js"%3E%3C/script%3E'));
</script>
```

速いネットワークから配信できればなぁ…

そこで登場するのが、CDN

- コンテンツ・デリバリー・ネットワークの略
- データのコピーを世界中の拠点におき、
  ネットワークの距離的に近い場所から配信する
- いまは、無償のサービスや
  有償でも手に届くレベルの存在に

代表的なCDNサービス

- Akamai
  http://www.akamai.co.jp/enja/
- CloudFlare
  http://jp.cloudflare.com/
- Amazon Web Service（S3* / CloudFront)
  http://aws.amazon.com/jp/cloudfront/

＊ S3はストレージサービスでCDNというわけではない

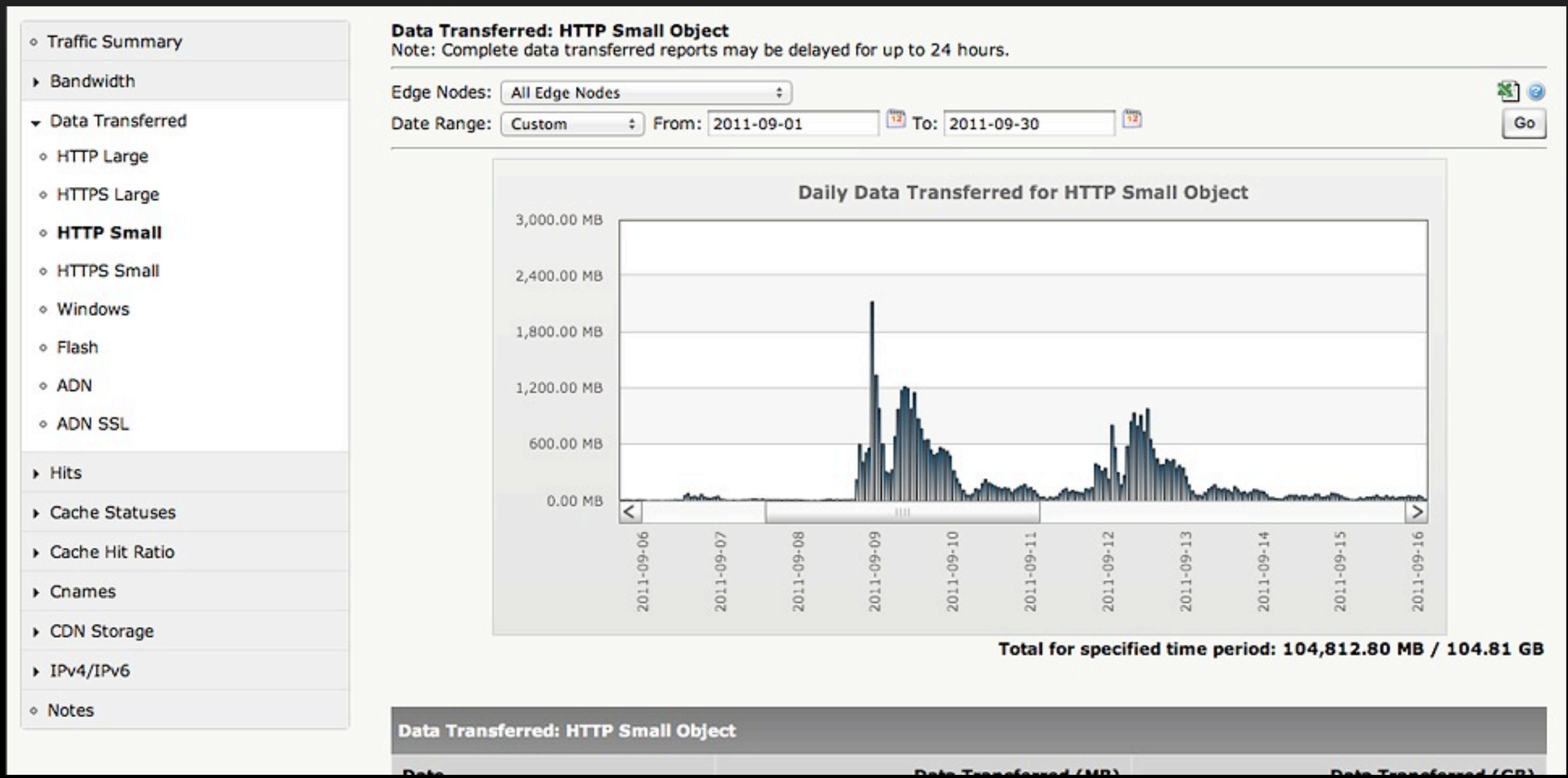
無償から始められるCloud Flare

AmazonのインフラをCloudFront

サービスによって異なる仕組み

- Cloud Flareは、ネームサーバを切り替える
- オリジナルデータを、
  自分で指定されたサーバにアップする
- リクエストがあったデータを、
  オリジナルから拾ってキャッシュする

従量制の場合は、コストチェックを



構成要素が多い場合など、
使いどころ、利用頻度で分割

それを、より高速な環境から配信する

多くのサイトでネックなのは、
そこに配置された画像ファイル

インフラそのものが弱い、
使用画像が多いときなど、
その効果は絶大なはず

本体のサーバに極力仕事をさせない

このあたりの実際は、ライブデモで

今日のまとめ

- 箱作りのファイルは、とにかく速く
- キャッシュをうまく使えばリクエスト減
- 外部要因による遅延を解消する
- ファイルが多すぎるなら、サーバ分割
- 設計段階からちゃんと考えること

これからのコンテンツ配信は、実装する人間だけじゃなく、いろんな立場が意識をしないと

本日はありがとうございました