2015.02

Takeshi NOZAWA

PAGE SPEED

Table of Content

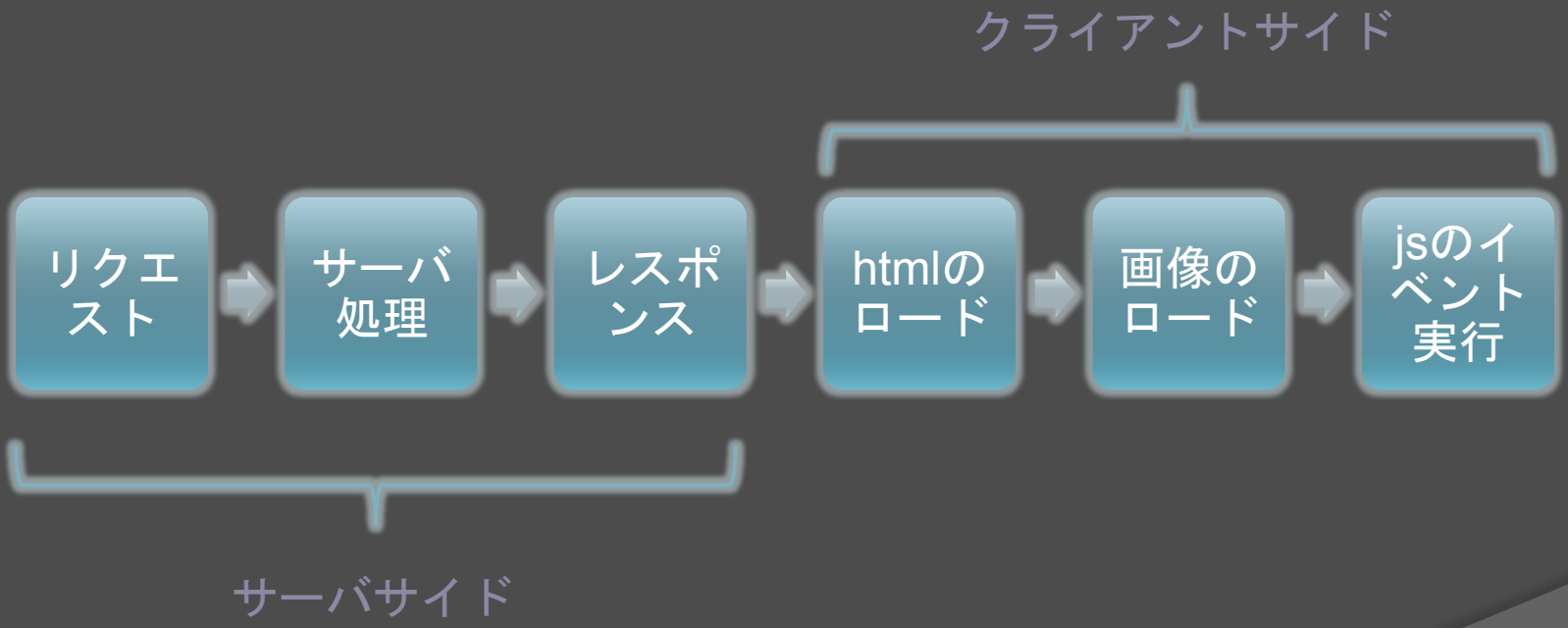
ページスピードが大事な理由

SEO

PV・コンバージョン

ユーザビリティ

ページスピードとは？


クライアントサイド
リクエスト
サーバ処理
レスポンス
htmlのロード
画像のロード
jsのイベント実行
サーバサイド

ページスピード改善のアプローチ

サイズを小さくする

- minify: CSS/JS/HTMLの改行やスペースを取る
- gzip:上記のテキストデータを圧縮する
- 画像のサイズを落とす（ロスレス圧縮など）

接続回数を減らす

- JSやCSSはまとめる
- 外部ファイルはブラウザキャッシュさせる

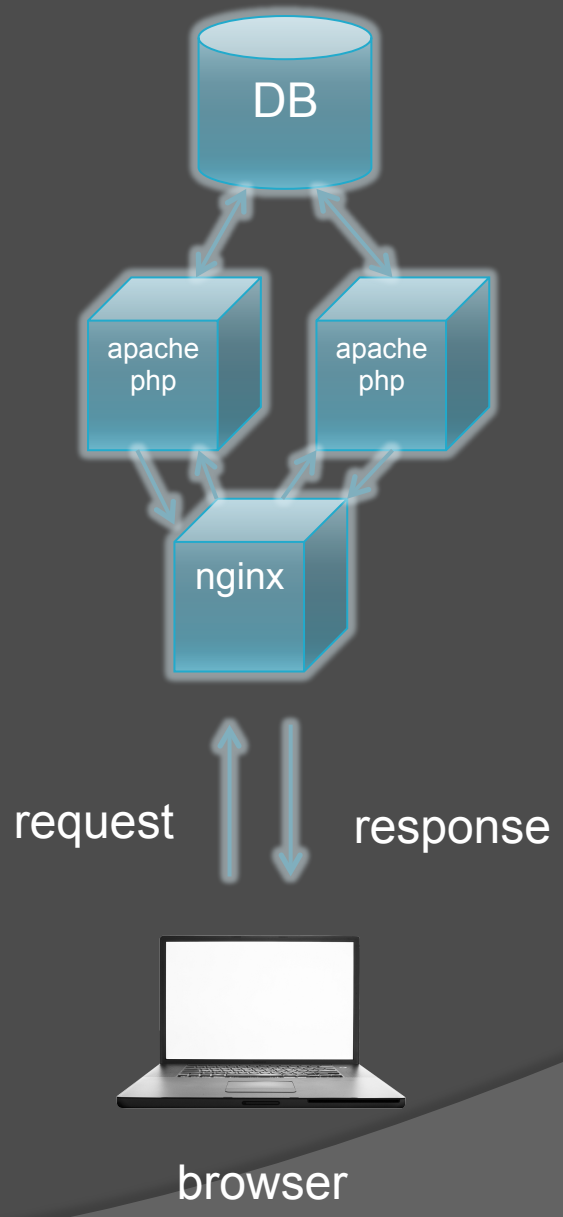
JSを実行する順番を工夫する

- async / defer
- </body>付近
- DOMContentLoaded
- onload

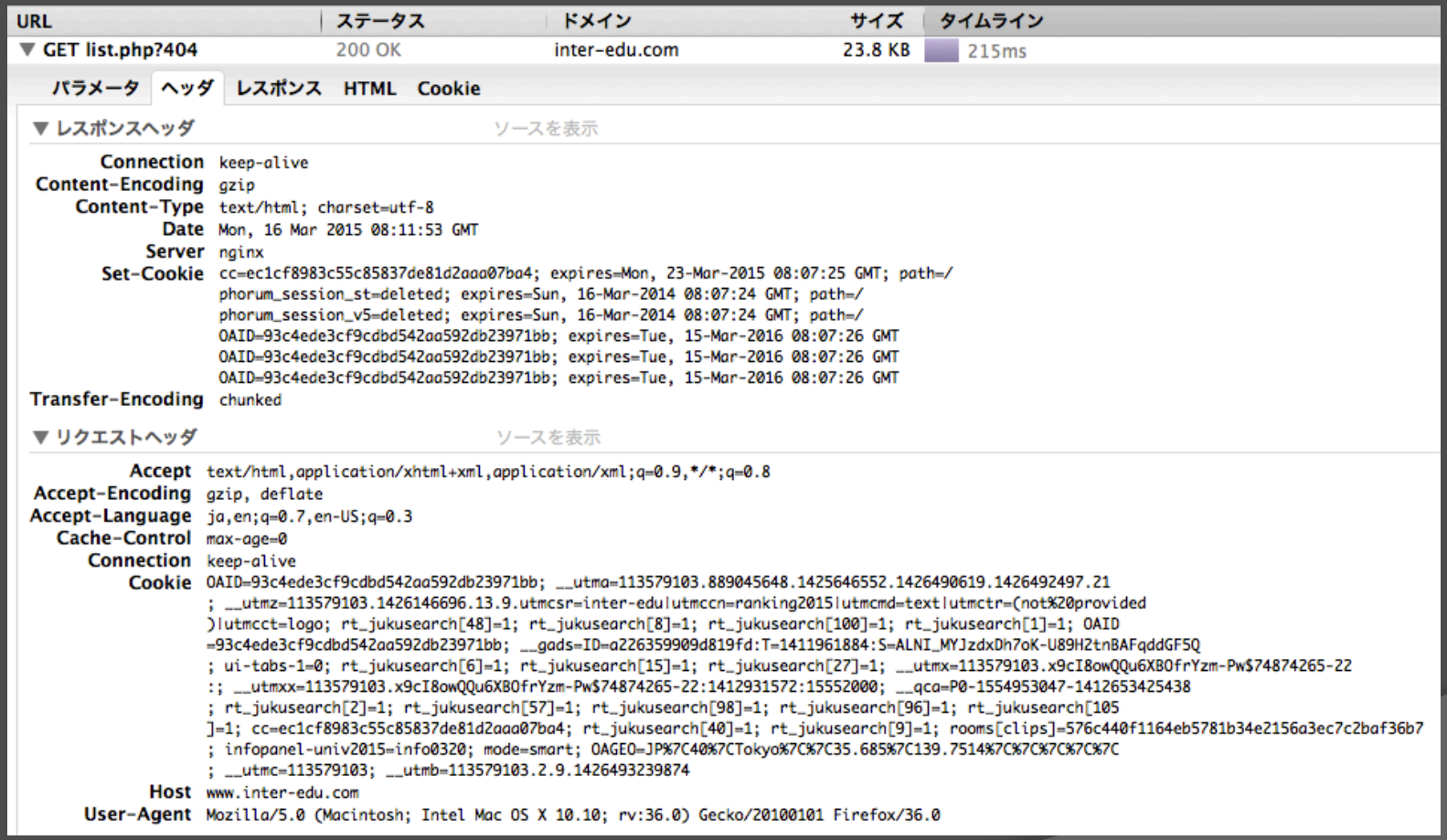
予備知識

http

- http
  - Hyper Text Transfer Protocol = 通信の取り決め
  - 代表的なサーバ
    - Apache
    - Nginx
    - Lighttpd



httpヘッダ



URL	ステータス	ドメイン	サイズ	タイムライン
▼ GET list.php?404	200 OK	inter-edu.com	23.8 KB	215ms

パラメータ　ヘッダ　レスポンス　HTML　Cookie

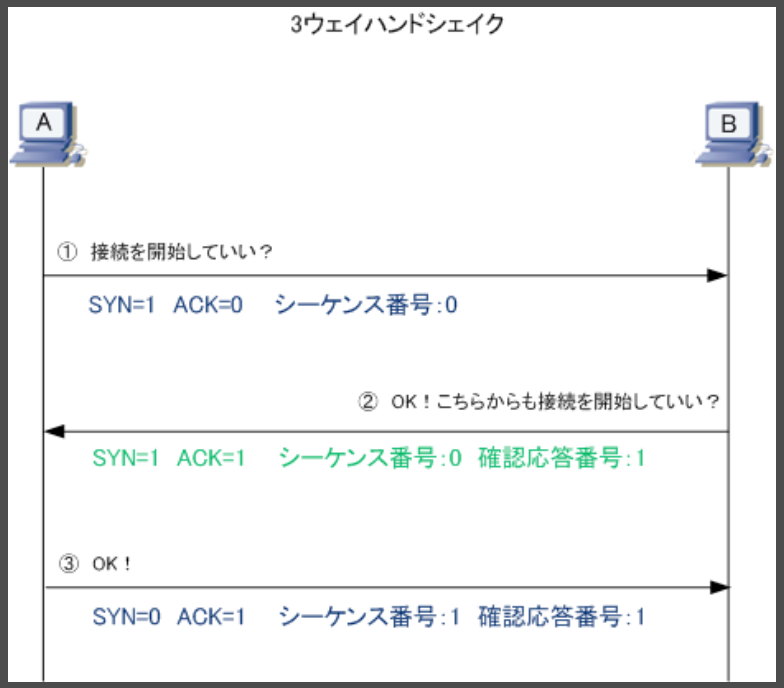
▼ レスポンスヘッダ　　ソースを表示

Connection	keep-alive
Content-Encoding	gzip
Content-Type	text/html; charset=utf-8
Date	Mon, 16 Mar 2015 08:11:53 GMT
Server	nginx
Set-Cookie	cc=ec1cf8983c55c85837de81d2aaa07ba4; expires=Mon, 23-Mar-2015 08:07:25 GMT; path=/ phorum_session_st=deleted; expires=Sun, 16-Mar-2014 08:07:24 GMT; path=/ phorum_session_v5=deleted; expires=Sun, 16-Mar-2014 08:07:24 GMT; path=/ OAID=93c4ede3cf9cdbd542aa592db23971bb; expires=Tue, 15-Mar-2016 08:07:26 GMT OAID=93c4ede3cf9cdbd542aa592db23971bb; expires=Tue, 15-Mar-2016 08:07:26 GMT OAID=93c4ede3cf9cdbd542aa592db23971bb; expires=Tue, 15-Mar-2016 08:07:26 GMT
Transfer-Encoding	chunked

▼ リクエストヘッダ　　ソースを表示

Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding	gzip, deflate
Accept-Language	ja,en;q=0.7,en-US;q=0.3
Cache-Control	max-age=0
Connection	keep-alive
Cookie	OAID=93c4ede3cf9cdbd542aa592db23971bb; __utma=113579103.889045648.1425646552.1426490619.1426492497.21 ; __utmz=113579103.1426146696.13.9.utmcsr=inter-edu|utmccn=ranking2015|utmcmd=text|utmctr=(not%20provided)|utmcct=logo; rt_jukusearch[48]=1; rt_jukusearch[8]=1; rt_jukusearch[100]=1; rt_jukusearch[1]=1; OAID =93c4ede3cf9cdbd542aa592db23971bb; __gads=ID=a226359909d819fd:T=1411961884:S=ALNI_MYJzdxDh7oK-U89H2tnBAFqddGF5Q ; ui-tabs-1=0; rt_jukusearch[6]=1; rt_jukusearch[15]=1; rt_jukusearch[27]=1; __utmx=113579103.x9cI8owQQu6XBOfrYzm-Pw$74874265-22 :; __utmxx=113579103.x9cI8owQQu6XBOfrYzm-Pw$74874265-22:1412931572:15552000; __qca=P0-1554953047-1412653425438 ; rt_jukusearch[2]=1; rt_jukusearch[57]=1; rt_jukusearch[98]=1; rt_jukusearch[96]=1; rt_jukusearch[105]=1; cc=ec1cf8983c55c85837de81d2aaa07ba4; rt_jukusearch[40]=1; rt_jukusearch[9]=1; rooms[clips]=576c440f1164eb5781b34e2156a3ec7c2baf36b7 ; infopanel-univ2015=info0320; mode=smart; OAGEO=JP%7C40%7CTokyo%7C%7C35.685%7C139.7514%7C%7C%7C%7C%7C ; __utmc=113579103; __utmb=113579103.2.9.1426493239874
Host	www.inter-edu.com
User-Agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:36.0) Gecko/20100101 Firefox/36.0

接続(connection)



http://www.infraexpert.com/study/tcpip9.html

- Transmission Control Protocol (伝送制御プロトコル:データを通信する際の取り決め)
- 繋ぐのも切るのも、結構複雑な処理をしています。
- 接続回数を極力少なくすることが大事です

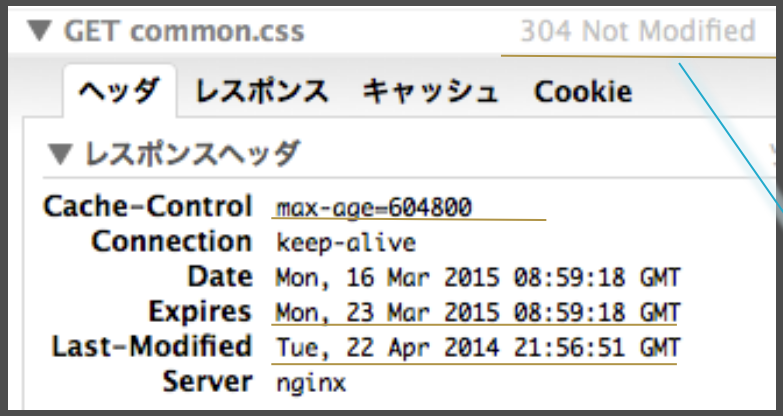
※サーバ側でKeepAlive機能が有効になっていると省略できる。

ここからはGoolgeのページスピード向上の
テクニックを分かりやすく紹介していきます
https://developers.google.com/speed/docs/insights/rules

テクニック

ブラウザキャッシュ

- css, js, 画像など一度ダウンロードしたファイルは、ブラウザに保存される。
- ブラウザキャッシュが有効だと、再度利用するときには、ダウンロードが発生しないので大幅に転送量を削減できる(モバイル環境では必須)
- その気になればHTMLソースもキャッシュ可
- ただし、内容に変更があっても、最初のロード時にはキャッシュファイルを使うことがあるのでリロードしないと変わらないことがあります。
- なのでhoge.js?20150316のように呼び出し元のファイル名にバージョンを入れるのがベストです。



変更がないのでローカルのキャッシュから使いましたの意味

【翻訳】
このファイルは最大604,800秒(7日間)、キャッシュが有効だよ！

最後に変更されたのは2014/04/22で、ダウンロードしたのは2015/4/16です。キャッシュが切れるのは7日後の4/23ですよ。

※インターエデュではNginxで自動的に上記の設定を付与しています。

ブラウザキャッシュ

URL	ステータス	ドメイン	サイズ	タイムライン
GET list.php?404	200 OK	inter-edu.com	23.7 KB	898ms
GET common.css	304 Not Modified	inter-edu.com	2.5 KB	10ms
GET frame.css	304 Not Modified	inter-edu.com	2.9 KB	10ms
GET forum.css	304 Not Modified	inter-edu.com	11.1 KB	10ms
GET fixed-menu.css	304 Not Modified	inter-edu.com	1.4 KB	9ms
GET banner.css	304 Not Modified	inter-edu.com	2.6 KB	10ms
GET colorbox_enq.css	304 Not Modified	inter-edu.com	1.4 KB	12ms
GET compliance.css	304 Not Modified	inter-edu.com	128 B	12ms
GET ydn.js	304 Not Modified	inter-edu.com	1.4 KB	11ms
GET yads.js	304 Not Modified	yads.yahoo.co.jp	11.7 KB	50ms
GET jquery.js	304 Not Modified	inter-edu.com	29.0 KB	11ms
GET urchin.js	304 Not Modified	inter-edu.com	5.3 KB	11ms
GET openx.js	304 Not Modified	inter-edu.com	768 B	14ms
GET forum.js	304 Not Modified	inter-edu.com	101 B	13ms
GET jquery.min.js	304 Not Modified	ajax.googleapis.com	32.7 KB	11ms
GET compliance.js?14243370	304 Not Modified	inter-edu.com	582 B	11ms
GET jsAreaSelect.js	304 Not Modified	inter-edu.com	366 B	10ms
GET freakout_map.js	304 Not Modified	inter-edu.com	947 B	11ms
GET iconvault-preview.css	304 Not Modified	inter-edu.com	1.6 KB	10ms
GET yads_vimps-1.0.0.js?201	304 Not Modified	i.yimg.jp	10.7 KB	16ms
GET tag?s=81192_2061&t=j&	200 OK	yads.yahoo.co.jp	10.3 KB	128ms
GET __utm.gif?utmwv=1&utm	200 OK	inter-edu.com	35 B	9ms
GET dc.js	304 Not Modified	stats.g.doubleclick.net	15.5 KB	40ms
GET spc.php?zones=super%3	200 OK	ad.inter-edu.com	2.8 KB	157ms
GET fl.js	304 Not Modified	ad.inter-edu.com	2.1 KB	20ms
GET __utm.gif?utmwv=5.6.3d	200 OK	stats.g.doubleclick.net	35 B	41ms
GET jquery_ui_tab.js	304 Not Modified	inter-edu.com	5.7 KB	12ms
GET fixed-menu.js	304 Not Modified	inter-edu.com	921 B	10ms
GET jquery.vticker.js	304 Not Modified	inter-edu.com	900 B	10ms
GET jquery.bxslider.js	304 Not Modified	inter-edu.com	4.4 KB	9ms
GET juku_sli.js	304 Not Modified	inter-edu.com	138 B	13ms

URL	ステータス	ドメイン	サイズ	タイムライン
GET list.php?404	200 OK	inter-edu.com	23.8 KB	442ms
GET common.css	200 OK	inter-edu.com	2.5 KB	15ms
GET frame.css	200 OK	inter-edu.com	2.9 KB	14ms
GET forum.css	200 OK	inter-edu.com	11.1 KB	23ms
GET fixed-menu.css	200 OK	inter-edu.com	1.4 KB	21ms
GET banner.css	200 OK	inter-edu.com	2.6 KB	20ms
GET colorbox_enq.css	200 OK	inter-edu.com	1.4 KB	13ms
GET compliance.css	200 OK	inter-edu.com	128 B	13ms
GET ydn.js	200 OK	inter-edu.com	1.4 KB	18ms
GET yads.js	200 OK	yads.yahoo.co.jp	11.7 KB	50ms
GET jquery.js	200 OK	inter-edu.com	29.0 KB	27ms
GET urchin.js	200 OK	inter-edu.com	5.3 KB	13ms
GET openx.js	200 OK	inter-edu.com	768 B	21ms
GET forum.js	200 OK	inter-edu.com	101 B	19ms
GET jquery.min.js	200 OK	ajax.googleapis.com	32.7 KB	13ms
GET compliance.js?14243370	200 OK	inter-edu.com	582 B	15ms
GET jsAreaSelect.js	200 OK	inter-edu.com	366 B	12ms
GET freakout_map.js	200 OK	inter-edu.com	947 B	21ms
GET iconvault-preview.css	200 OK	inter-edu.com	1.6 KB	9ms
GET yads_vimps-1.0.0.js?201	200 OK	i.yimg.jp	10.7 KB	31ms
GET tag?s=81192_2061&t=j&	200 OK	yads.yahoo.co.jp	10.3 KB	138ms
GET __utm.gif?utmwv=1&utm	200 OK	inter-edu.com	35 B	16ms
GET dc.js	200 OK	stats.g.doubleclick.net	15.5 KB	42ms
GET spc.php?zones=super%3	200 OK	ad.inter-edu.com	2.8 KB	150ms
GET fl.js	200 OK	ad.inter-edu.com	2.1 KB	25ms
GET __utm.gif?utmwv=5.6.3d	200 OK	stats.g.doubleclick.net	35 B	42ms
GET jquery_ui_tab.js	200 OK	com	5.7 KB	10ms
GET fixed-menu.js	200 OK	com	921 B	12ms
GET jquery.vticker.js	200 OK	com	900 B	9ms
GET jquery.bxslider.js	200 OK	com	4.4 KB	11ms
GET juku_sli.js	200 OK	com	138 B	14ms

ブラウザキャッシュ有効
“304 Not Modified”
470KB ← 55% down

ブラウザキャッシュ無効
“200 OK”
1MB

Minify

Minifyとは？

- 改行やタブ、スペースを排除し、データサイズを下げる技術(?)

変換ツール

- CSS
  - http://cssminifier.com/
- JS
  - http://closure-compiler.appspot.com/home
  - googleが使っているツール

運用上の注意

- 人が読みにくいのでLESSや元ファイルをgit commitする。
- *.min.js, *.min.cssなどの名前で識別出来るようにすると便利。

- 例：jquery-1.11.2.js

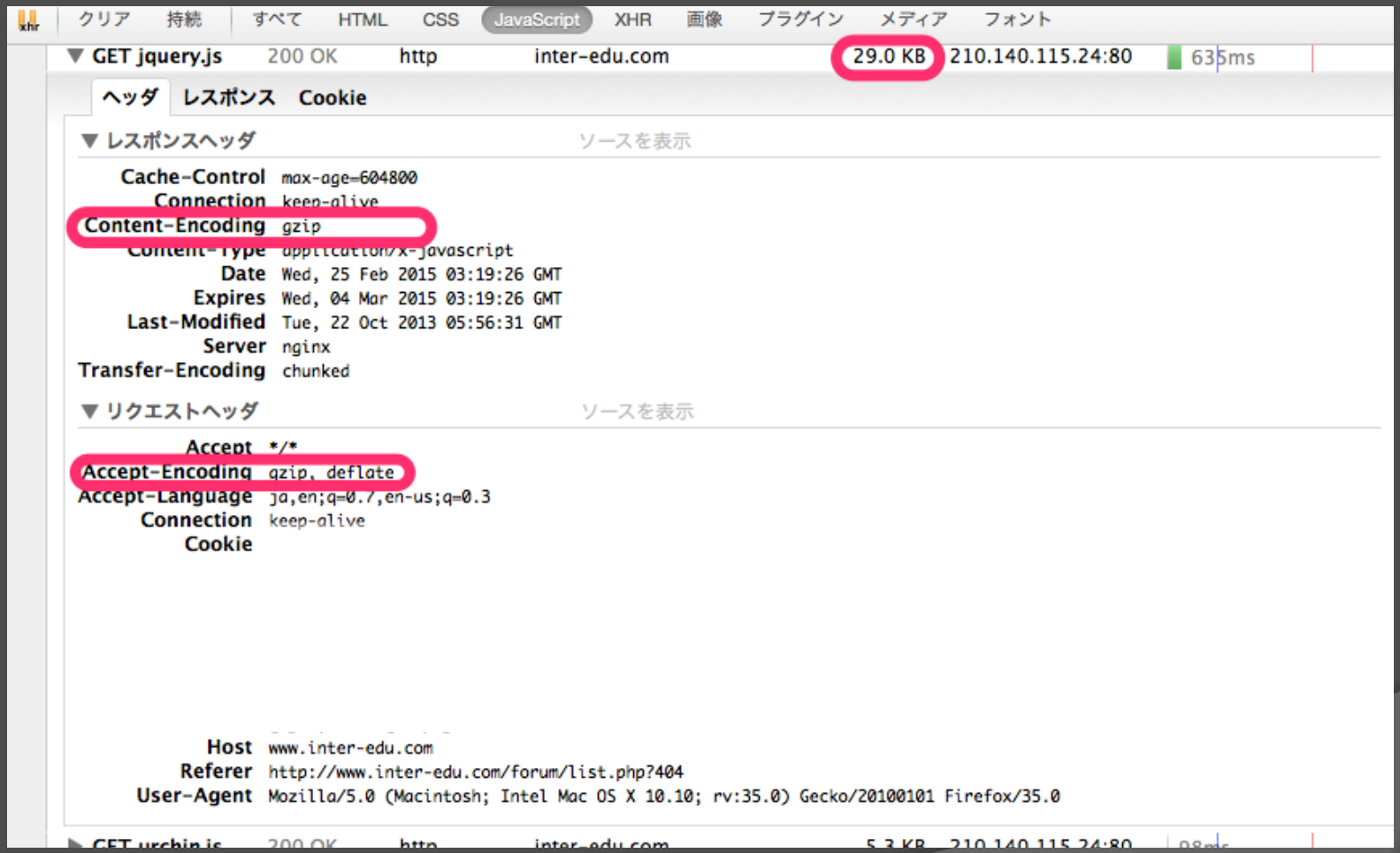
URL	ステータス	ドメイン	サイズ
▶ GET jquery-1.11.2.js	200 OK	code.jquery.com	101.0 KB

URL	ステータス	ドメイン	サイズ
▶ GET jquery-1.11.2.min.js	200 OK	code.jquery.com	38.0 KB

62% size down

gzip

- 転送量を下げる方法として、html,js,css,jsonなどの「文字情報」に対して有効。
- エデュではApache(Nginx)で自動的にgzip圧縮をしています。
- ほとんどのモダンブラウザで対応しています



gzip: どれだけ軽くなるか？

```
278K  3 16 17:39 jquery-1.11.2.js
 94K  3 16 17:39 jquery-1.11.2.min.js
 83K  3 16 17:40 jquery-1.11.2.js.gz
 33K  3 16 17:40 jquery-1.11.2.min.js.gz
```

- : default
34%: minify
30%: gzip
11%: minify + gzip

画像

サイズ指定

- 適切なサイズの画像を使いましょう。
- 50px × 50pxの枠に300px × 300pxの画像
- width, heightは必ず指定 (altもね・・・)

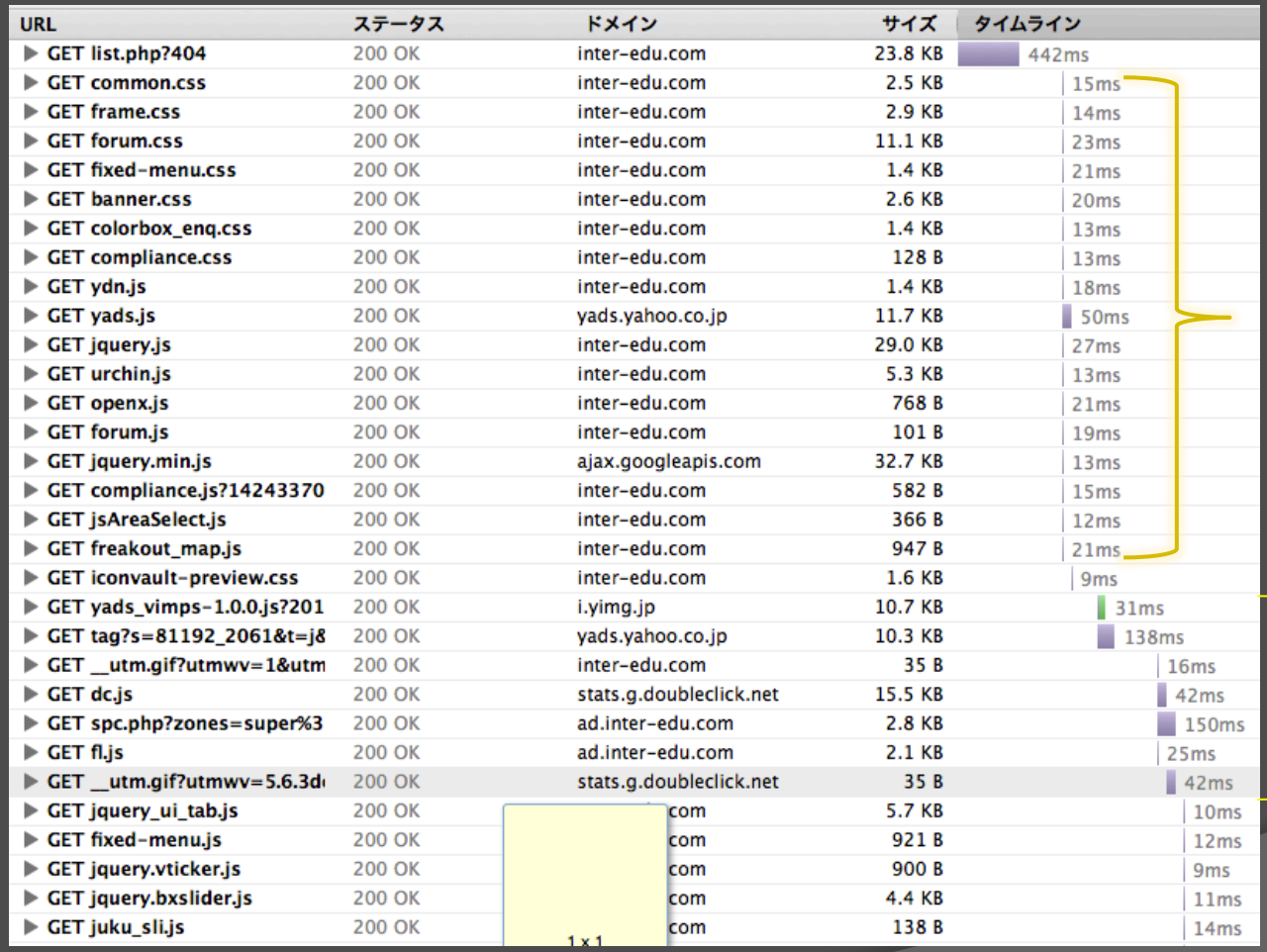
ロスレス圧縮(不可逆圧縮)

- 画像データとは別にexifなどの付帯情報を削除すること。
- 画質は維持しつつ、ファイルサイズを減らす

CSSとJSの順番

- できるだけファイル化する。そうすることでブラウザキャッシュが効くようになる。インラインはキャッシュできない。
- ただし数行のJS/CSSファイルのために通信を開くのは逆に遅くなるので、そういったものはインラインでも良い。
- 当然ファイル数は少ないほうが良い。
- 読み込む順番(原則)
  1. 内部CSS
  2. 外部CSS
  3. 内部JS
  4. 外部JS
  5. インラインJS
- まとめておくとブラウザはいっぺんに接続を開いてまとめてダウンロードしてくれるので早い。（特に同ホストからはまとめてダウンロードする）

CSSとJSの順番



URL	ステータス	ドメイン	サイズ	タイムライン
GET list.php?404	200 OK	inter-edu.com	23.8 KB	442ms
GET common.css	200 OK	inter-edu.com	2.5 KB	15ms
GET frame.css	200 OK	inter-edu.com	2.9 KB	14ms
GET forum.css	200 OK	inter-edu.com	11.1 KB	23ms
GET fixed-menu.css	200 OK	inter-edu.com	1.4 KB	21ms
GET banner.css	200 OK	inter-edu.com	2.6 KB	20ms
GET colorbox_enq.css	200 OK	inter-edu.com	1.4 KB	13ms
GET compliance.css	200 OK	inter-edu.com	128 B	13ms
GET ydn.js	200 OK	inter-edu.com	1.4 KB	18ms
GET yads.js	200 OK	yads.yahoo.co.jp	11.7 KB	50ms
GET jquery.js	200 OK	inter-edu.com	29.0 KB	27ms
GET urchin.js	200 OK	inter-edu.com	5.3 KB	13ms
GET openx.js	200 OK	inter-edu.com	768 B	21ms
GET forum.js	200 OK	inter-edu.com	101 B	19ms
GET jquery.min.js	200 OK	ajax.googleapis.com	32.7 KB	13ms
GET compliance.js?14243370	200 OK	inter-edu.com	582 B	15ms
GET jsAreaSelect.js	200 OK	inter-edu.com	366 B	12ms
GET freakout_map.js	200 OK	inter-edu.com	947 B	21ms
GET iconvault-preview.css	200 OK	inter-edu.com	1.6 KB	9ms
GET yads_vimps-1.0.0.js?201	200 OK	i.yimg.jp	10.7 KB	31ms
GET tag?s=81192_2061&t=j&	200 OK	yads.yahoo.co.jp	10.3 KB	138ms
GET __utm.gif?utmwv=1&utm	200 OK	inter-edu.com	35 B	16ms
GET dc.js	200 OK	stats.g.doubleclick.net	15.5 KB	42ms
GET spc.php?zones=super%3	200 OK	ad.inter-edu.com	2.8 KB	150ms
GET fl.js	200 OK	ad.inter-edu.com	2.1 KB	25ms
GET __utm.gif?utmwv=5.6.3d	200 OK	stats.g.doubleclick.net	35 B	42ms
GET jquery_ui_tab.js	200 OK	com	5.7 KB	10ms
GET fixed-menu.js	200 OK	com	921 B	12ms
GET jquery.vticker.js	200 OK	com	900 B	9ms
GET jquery.bxslider.js	200 OK	com	4.4 KB	11ms
GET juku_sli.js	200 OK	com	138 B	14ms

1 x 1

まとめているのでダウンロードのタイミングが揃っている

ネットワーク広告はカオスなのでガタガタ

レンダリングブロック

- ブラウザはJSの実行をする際、それが終わるまで後続する処理を始めない。＝レンダリングブロック（表示処理のブロック）これがページスピードの遅れに繋がる。
- インラインで実行スクリプトがあるとそこで止まる。(OpenXのOA_showなど)

掲示板CPCテキスト下(掲示板右テキスト)
OA_show('text_R');
で止まっている状態

ファーストビュー

- Googleはファーストビューの表示にこだわるし(PageSpeed Insightのスコア)、ユーザーもそのほうが嬉しい（直帰率も下がる）
- だからファーストビューに関係ないJSはasync属性をつけるか、</body>直前に入れる。（前者の方がメンテナンスしやすい？）
  - async（＝非同期）。async属性のついたJSは前の処理が終わるのを待たずに、次の処理を始める（レンダリングブロックが発生しない）。
  - しかし、非同期だと上から順番に処理をしないので、jQueryの関数が登録される前に実行しようするエラーが出たりするので困る。
  - そういうときは
    - deferを使う(</body>前に入れるのと近しい)。
    - jsでscriptエレメントを作成し、順番にappendするような非同期スクリプトを書く（ムズイ）

asyncとレンダリングブロック

```
<script type="text/javascript">
    urchinTracker();
</script>
<script type="text/javascript">
 var _gaq = _gaq || [];
 _gaq.push(['_setAccount',                    ]);
 _gaq.push(['_setSiteSpeedSampleRate', 30]);
 _gaq.push(['_trackPageview']);
            _gaq.push(['_setCustomVar', 4, 'theme', 'sp', 3]);
        setTimeout('_gaq.push([\'_trackEvent\', \'NoBounce\', \'Over 10 seconds\'])', 10000);

 (function() {
   var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
   ga.src = ('https:' == document.location.protocol ? 'https://' : 'http://') + 'stats.g.doubleclick.net/dc.js';
   var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
 })();
</script>
```

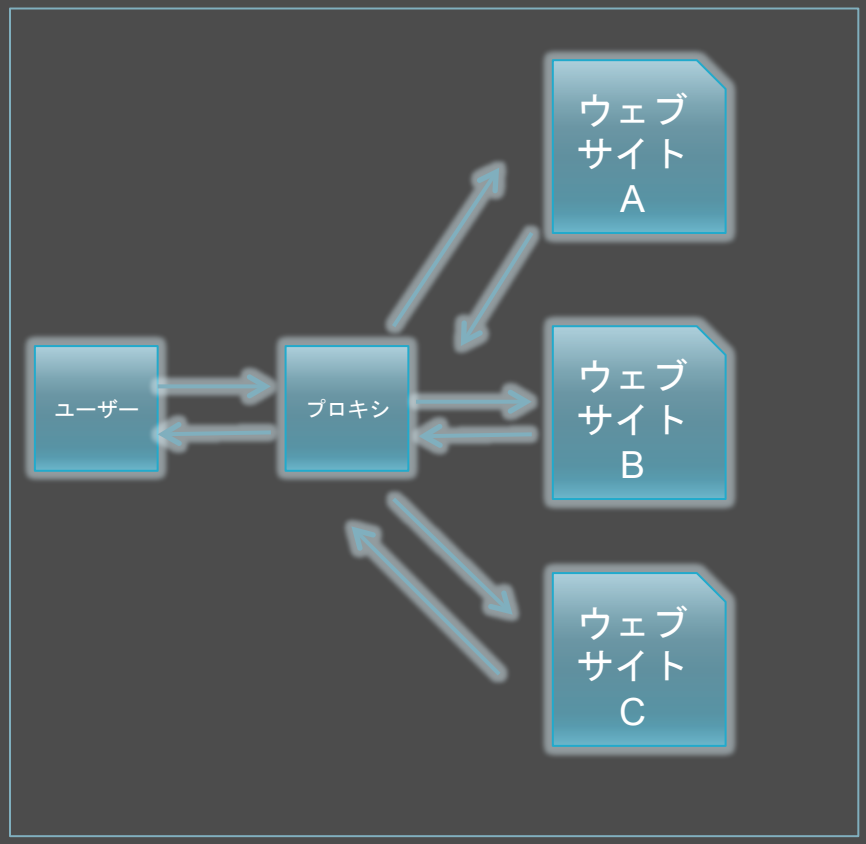
```
<script type="text/javascript" async="true" src="/js/sp/common.js"></script>
<script type="text/javascript" async="true" src="/js/lib/jquery.bxslider.min.js"></script>
<script type="text/javascript" async="true" src="/js/lib/jquery.cookie-1.4.0.js"></script>
<script type="text/javascript" src="/common2/js/im/ydn.js" charset="utf-8"></script>
```

<head>付近

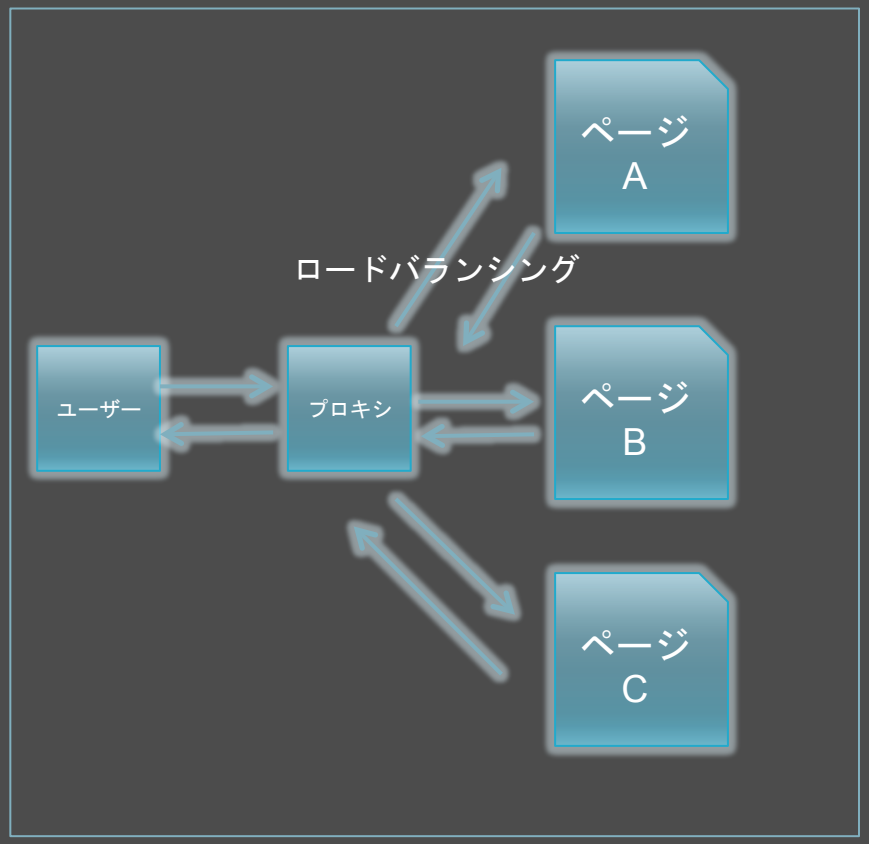
Nginx

- ロシア製のプロキシサーバ/httpサーバ
- プロキシ＝代理(人)
  - フォワードプロキシ
  - リバースプロキシ（ウェブアクセラレータ）
    - Squid
    - Varnish
    - ページ(HTMLソース)を丸ごとキャッシュするので背後にあるPHPのプログラムに処理をさせないため、PHPサーバの負荷が激減する。
    - 単純なHTMLのサービングになるため、非常に高速かつ多くのリクエストを同時に捌ける。

プロキシサーバ



フォワードプロキシ
ユーザーはプロキシを通して、ページにアクセスする。プロキシはユーザーの代わりにページ情報をリクエストし、キャッシュする。もしキャッシュがあれば、情キャッシュ内容をユーザーに提供する。ナローバンド時代、通信コスト削減ために採用された技術。



リバースプロキシ
フォワードプロキシと理屈は同じだが、サイト管理者が動的なコンテンツを生成させるPHPなどの重たいプログラムを動かすサーバの負荷をさげるために使われている。

Google Page Speed Insight

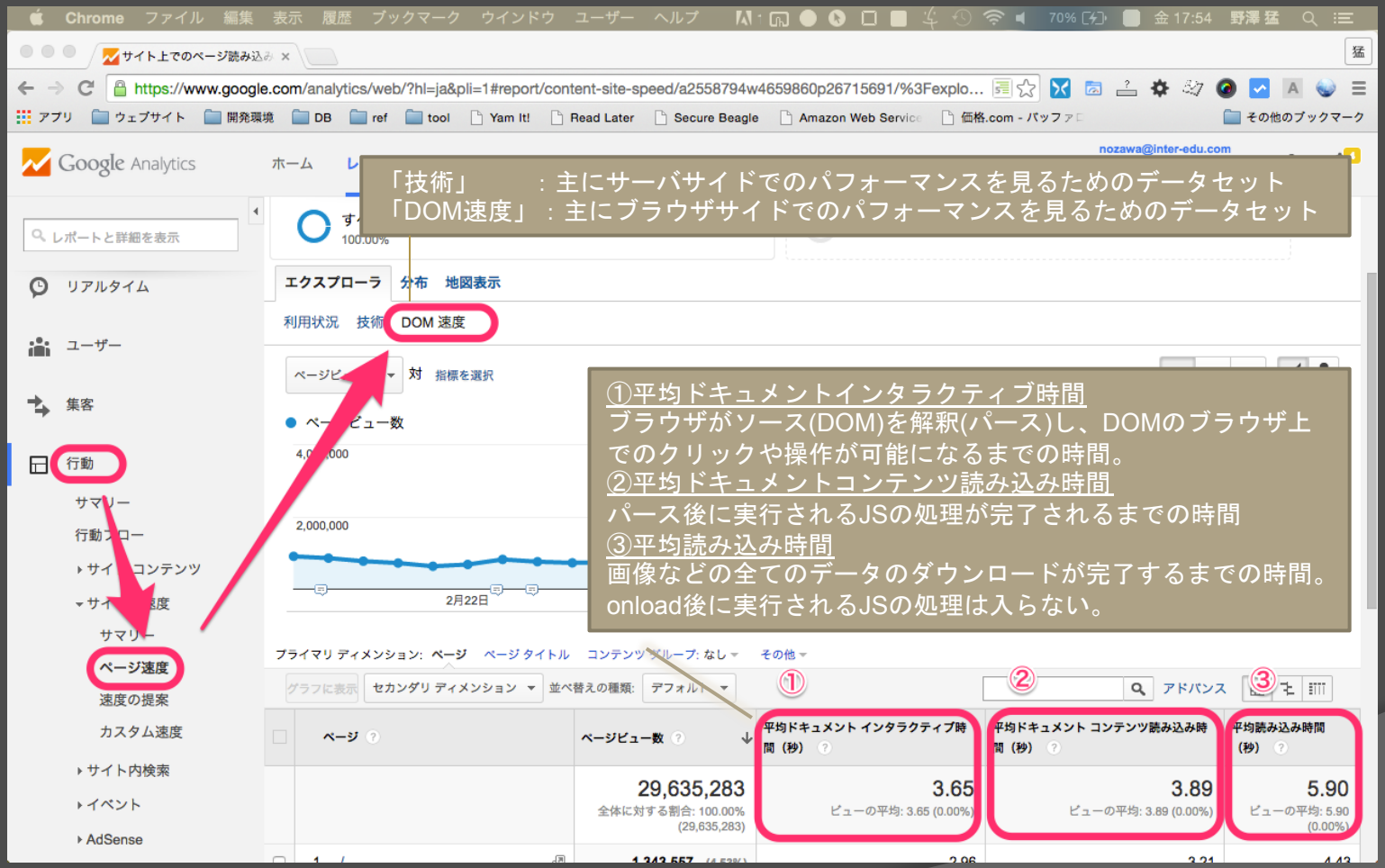
- 最も基本的で分かりやすいツール
- 秒数 ＜ スコア

https://developers.google.com/speed/pagespeed/insights/

Chrome addon（廃止）

- Web版PageSpeed Insightとほぼ同じ機能
- スコアは出ない
- ローカルのURLでも検証可能
- Firefox版は利用不可

GAのページスピードの見方



performance.timing

ブラウザの開発ツールのコンソールなどで

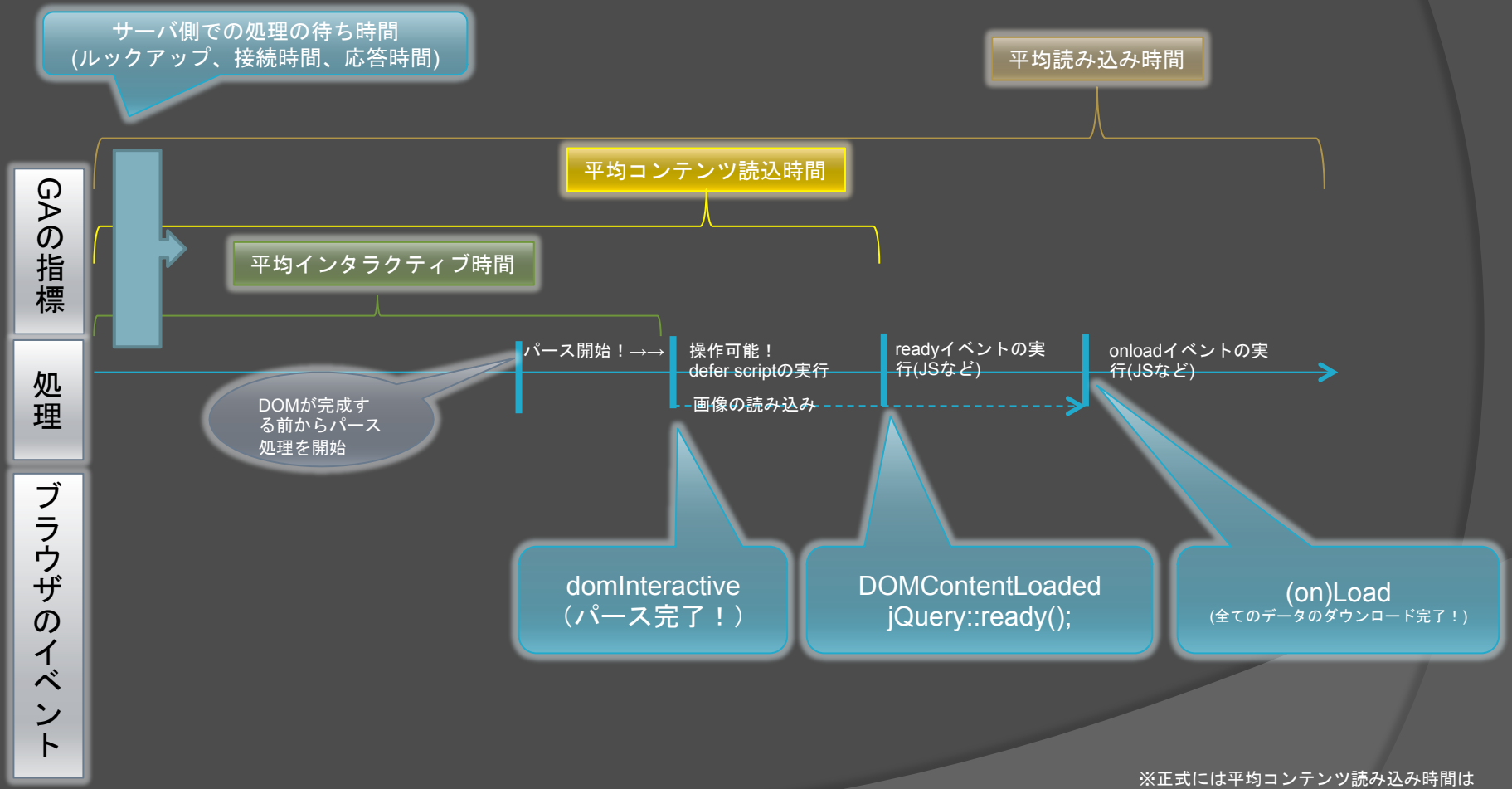
performance.timig

と実行するとJSの”Navigation Timing API”という機構によって、JSが計測している各イベントにかかった時間を呼び出せる。

Firebugではグラフにして分かりやすく表示してくれる。

- 青いライン
  - DOMのパースが完了し、パース後のjs(deferなど)を処理し始めたタイミング(deferのjsなどに357msかかっている)。インタラクティブになるのはこの直前くらい。
- 赤いライン
  - 画像など全てデータのダウンロードが終わりonloadイベントが始まったタイミング(onload後のjsに48msかかっている)

ブラウザイベントとGAの指標まとめ



※正式には平均コンテンツ読み込み時間はDOMContentLoadedEventEndで実行されているみたいです。