Performance Tuning TIPS *n

Frontrend in Fukuoka 01/25

Ayumu Sato @ahomu

#frontrend_ff1a

さとう　あゆむ

佐藤　歩

ハンドルネーム

@ahomu

興味もってくださったらゼヒ

http://aho.mu

本題

普段、見てる方？

PageSpeed

or

YSlow

ある制作とお客さんと
かっこいいサイト

超かっこいい
サイトつくれたわー

お客さんも大満足だわー

制作者

今、出先なんですけど
サイト開けなくないですか？

お客から電話

あれー
ぼくのPCでは開けますよ？

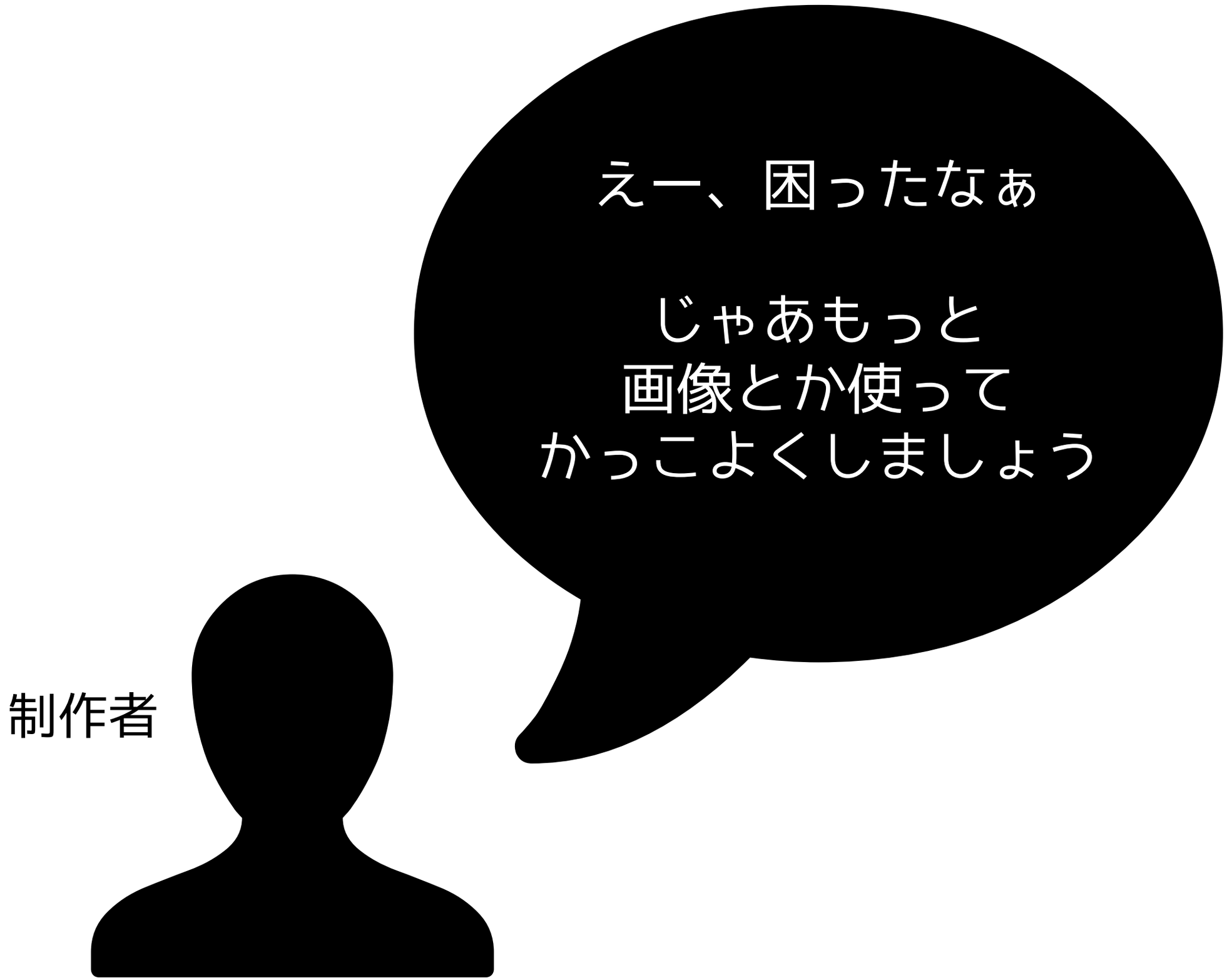
いやぁ、かっこいいなぁ

制作者

そうですか

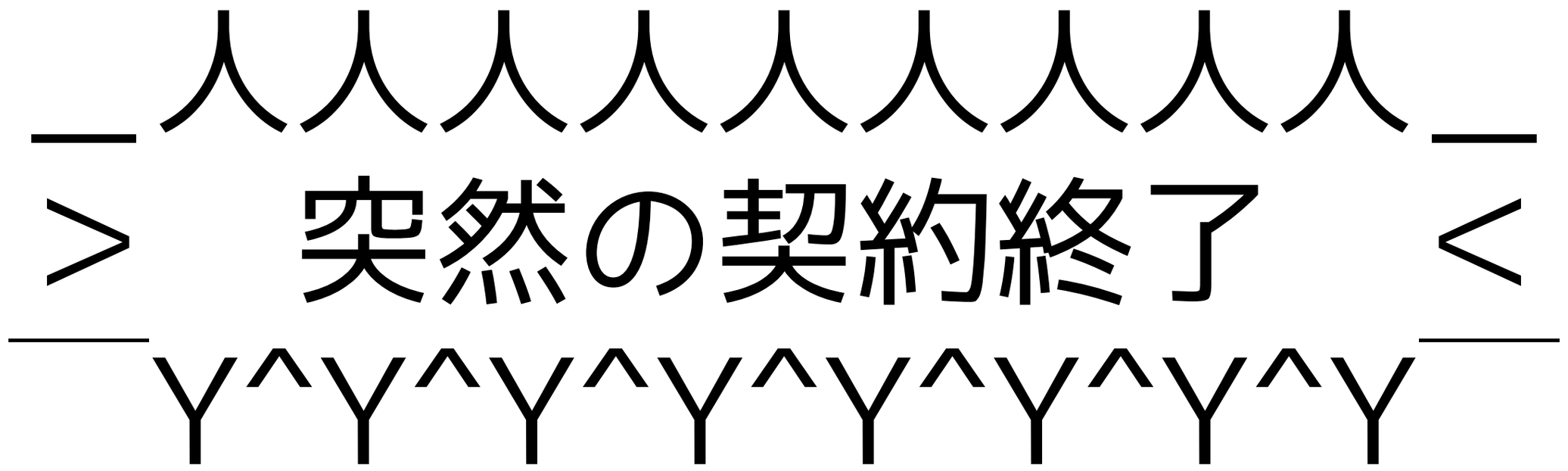
ところでコンバージョン
上がらないんですけど
なんとかなりません？

お客から電話


えー、困ったなぁ
じゃあもっと
画像とか使って
かっこよくしましょう
制作者


わかりました
・・・
お客から電話

モバイルで重くて開けない
開いてもスクロールが重い・カクカク
ユーザーに離脱されてしまう
コンバージョンに至らない


＿人人人人人人人人人＿
＞　突然の契約終了　＜
￣Y^Y^Y^Y^Y^Y^Y^Y^Y￣

そうならないために
知っておきたいTIPSを
紹介します

1. パフォーマンスの動機
2. 考え方と導入
3. Tips *n
    1. ネットワークコスト
    2. レンダリング
    3. スクリプト処理
    4. インタラクション
4. まとめ

考え方と導入

Introduction

フロントエンドとパフォーマンスの関係

Frontend

Performance

Search for:

Search

MOST RECENT POSTS

HTML Imports: scope, security, and suggestions
Performance and Custom Elements
Async Ads with HTML Imports
HTTP Archive 3 Year Anniversary (Thank You Pat Meenan)
Prebrowsing

View Archive

FEEDS

Posts
Comments

the Performance Golden Rule

February 10, 2012 5:37 pm | 26 Comments

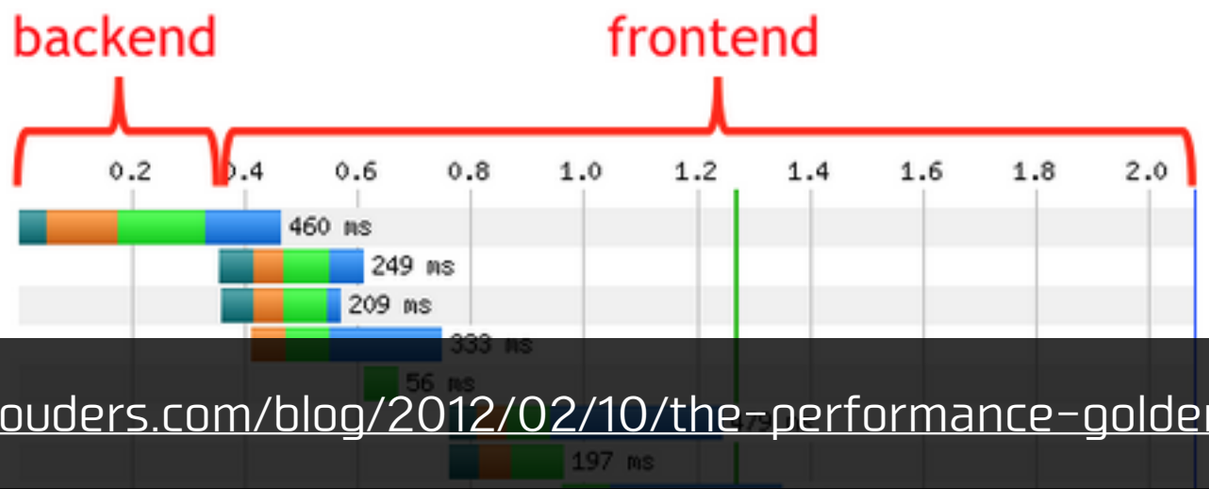
Yesterday I did a workshop at Google Ventures for some of their portfolio companies. I didn't know how much performance background the audience would have, so I did an overview of everything performance-related starting with my first presentations back in 2007. It was very nostalgic. It has been years since I talked about the best practices from High Performance Web Sites. I reviewed some of those early tips, like Make Fewer HTTP Requests, Add an Expires Header, and Gzip Components.

But I needed to go back even further. Thinking back to before Velocity and WPO existed, I thought I might have to clarify why I focus mostly on frontend performance optimizations. I found my slides that explained the Performance Golden Rule:

> *80-90% of the end-user response time is spent on the frontend.*
> *Start there.*

There were some associated slides that showed the backend and frontend times for popular websites, but the data was old and limited, so I decided to update it. Here are the results.

First is an example waterfall showing the backend/frontend split. This waterfall is for LinkedIn. The "backend" time is the time it takes the server to get the first byte back to the client. This typically includes the bulk of backend processing: database lookups, remote web service calls, stitching together HTML, etc. The "frontend" time is everything else. This includes obvious frontend phases like executing JavaScript and rendering the page. It also includes network time for downloading all the resources referenced in the page. I include this in the frontend time because there's a great deal web developers can do to reduce this time, such as async script loading, concatenating scripts and stylesheets, and sharding domains.



http://www.stevesouders.com/blog/2012/02/10/the-performance-golden-rule/

“80-90% of the end-user response time is spent on the frontend.”

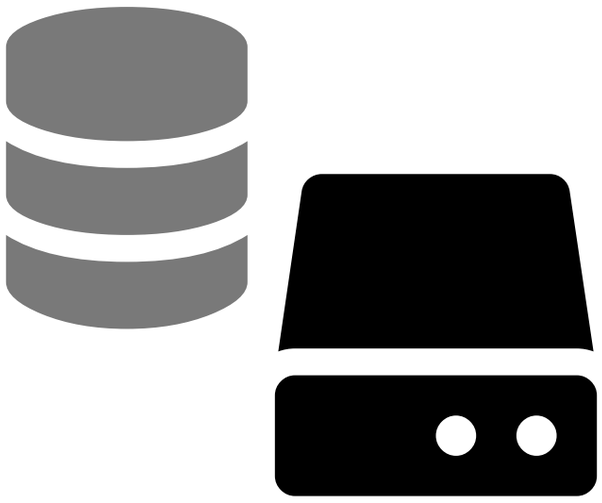
Steve Souders

backend

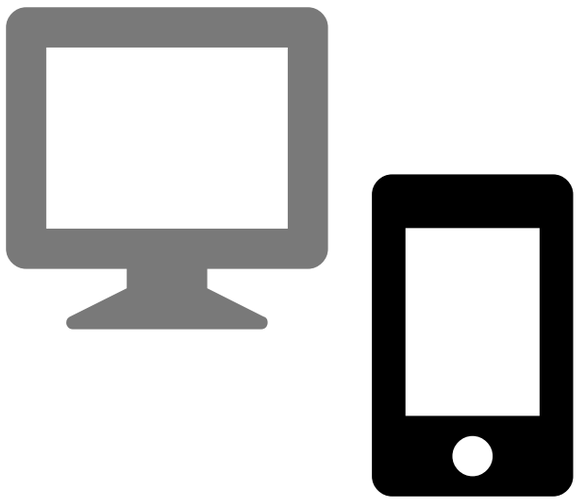
frontend

0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0

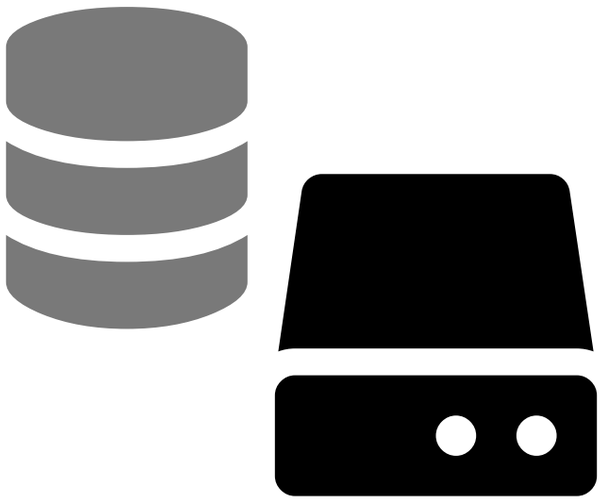
460 ms
249 ms
209 ms
333 ms
56 ms
479 ms
197 ms
386 ms
85 ms
678 ms
546 ms
503 ms
651 ms
75 ms
203 ms
181 ms



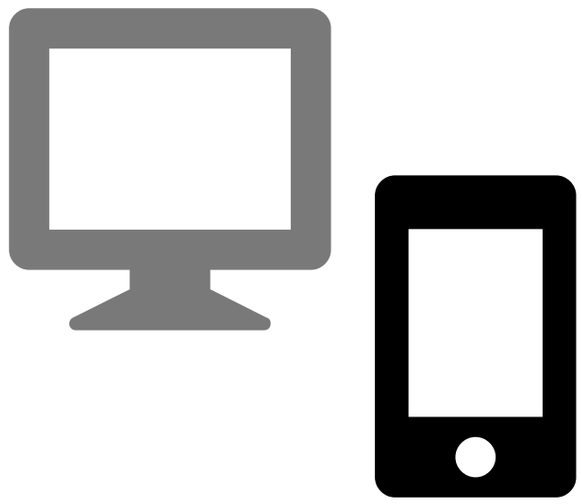
Backend
(DB, WebApp)



Frontend
(Browser)

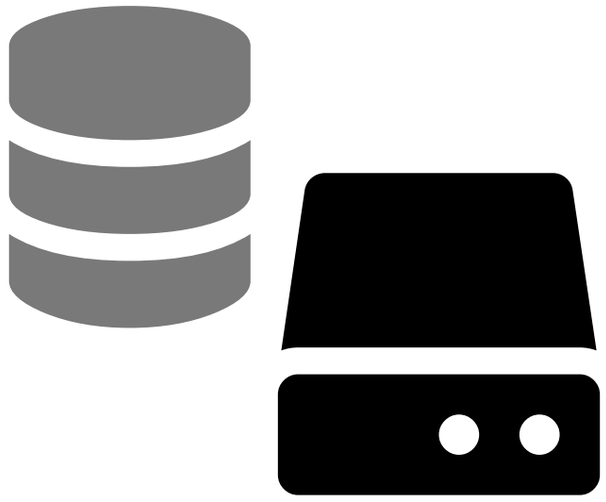


Backend
(DB, WebApp)



Frontend
(Browser)

```
<link href=foo.css…
<script src=bar.js…
<img src=qux.jpg…
```

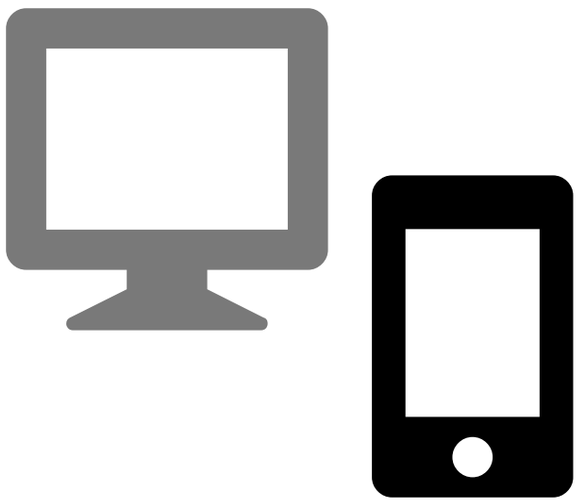


Backend
(DB, WebApp)

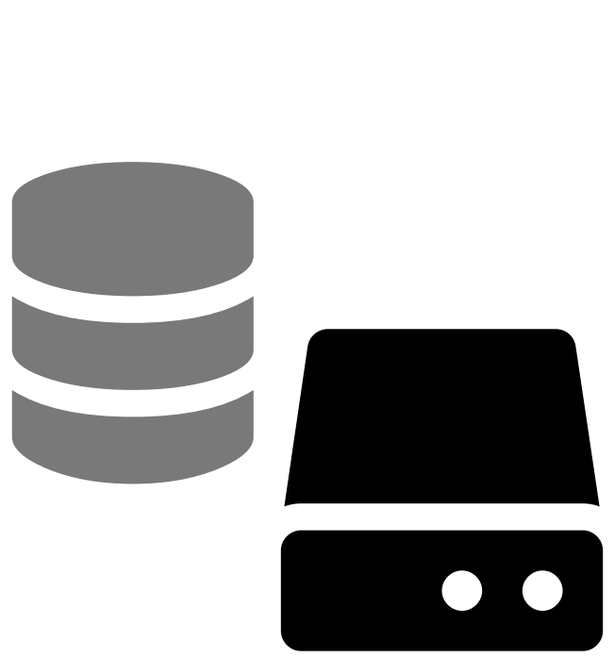
GET /foo.css

GET /bar.js

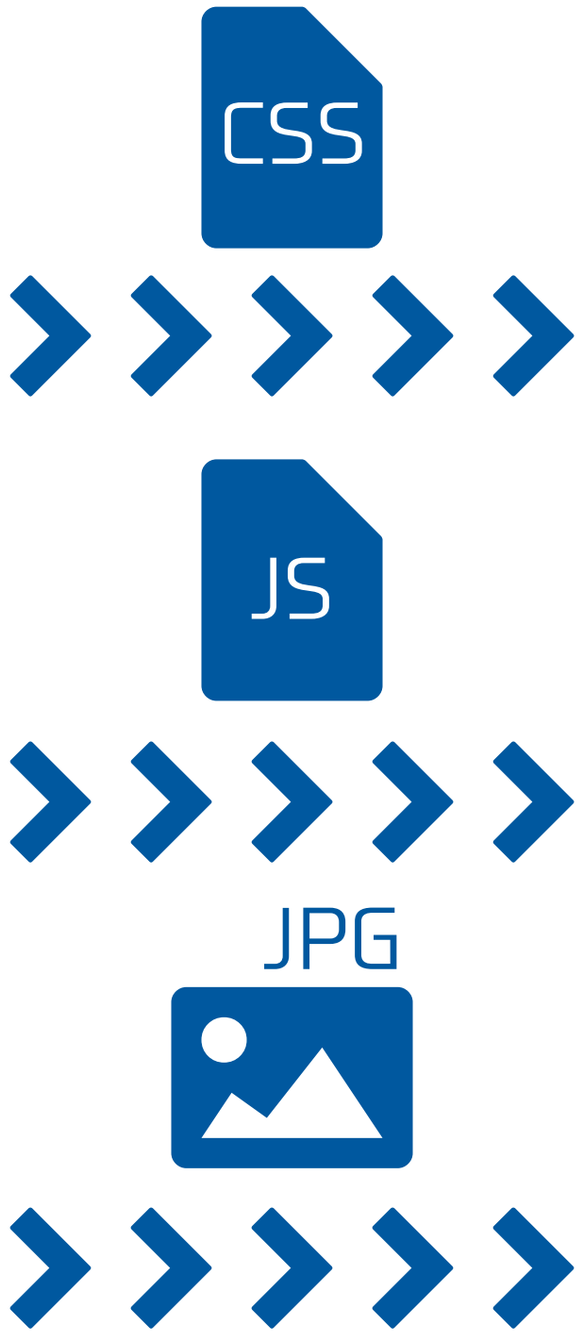
GET /qux.jpg

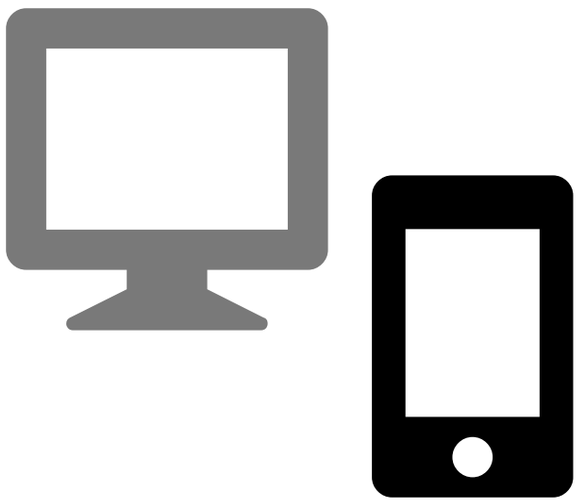


Frontend
(Browser)



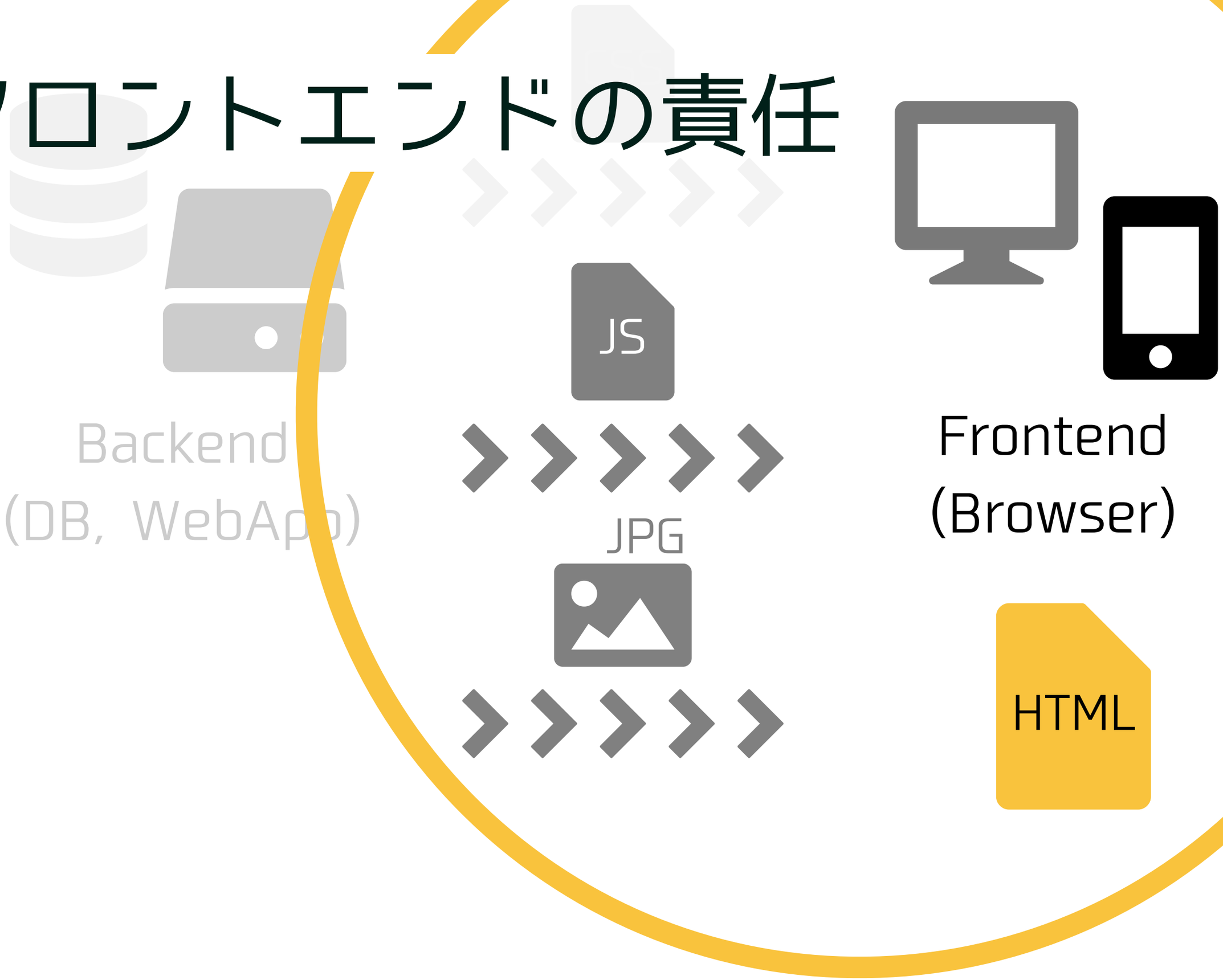
Backend
(DB, WebApp)





Frontend
(Browser)

フロントエンドの責任



もうひとつ･･･

＋α

ユーザーとインターフェースの対話

Interaction & UX

スクロールが遅い

カクカクするサイト

ボタンを押しても
反応がないサイト

✕ 見栄えするけど イライラするサイト

使用感・心地よさも
広義のパフォーマンス

フロントの工夫で
改善できることはする

TIPSを集めた理由

PageSpeed Insights +1 9

PageSpeed Insights Rules

Speed Rules

- Avoid landing page redirects
- Enable compression
- Improve server response time
- Leverage browser caching
- Minify resources
- Optimize images
- Optimize CSS Delivery
- Prioritize visible content
- Remove render-blocking JavaScript
- Use asynchronous scripts

Usability Rules

- Avoid plugins
- Configure the viewport
- Size content to viewport
- Size tap targets appropriately
- Use legible font sizes

https://developers.google.com/speed/docs/insights/rules

Best Practices for Speeding Up Your Web Site

The Exceptional Performance team has identified a number of best practices for making web pages fast. The list includes 35 best practices divided into 7 categories.

Filter by category: Content | Server | Cookie | CSS | JavaScript | Images | Mobile | All

Minimize HTTP Requests

tag: content

80% of the end-user response time is spent on the front-end. Most of this time is tied up in downloading all the components in the page: images, stylesheets, scripts, Flash, etc. Reducing the number of components in turn reduces the number of HTTP requests required to render the page. This is the key to faster pages.

One way to reduce the number of components in the page is to simplify the page's design. But is there a way to build pages with richer content while also achieving fast response times? Here are some techniques for reducing the number of HTTP requests, while still supporting rich page designs.

Combined files are a way to reduce the number of HTTP requests by combining all scripts into a single script, and similarly combining all CSS into a single stylesheet. Combining files is more challenging when the scripts and stylesheets vary from page to page, but making this part of your release process improves response times.

CSS Sprites are the preferred method for reducing the number of image requests. Combine your background images into a single image and use the CSS `background-image` and `background-position` properties to display the desired image segment.

Image maps combine multiple images into a single image. The overall size is about the same, but reducing the number of HTTP requests speeds up the page. Image maps only work if the images are contiguous in the page, such as a navigation bar. Defining the coordinates of image maps can be tedious and error prone. Using image maps for navigation is not accessible too, so it's not recommended.

Inline images use the `data:` URL scheme to embed the image data in the actual page. This can increase the size of your HTML document. Combining inline images into your (cached) stylesheets is a way to reduce HTTP requests and avoid increasing the size of your pages. Inline images are not yet supported across all major browsers.

Reducing the number of HTTP requests in your page is the place to start. This is the most important guideline for improving performance for first time visitors. As described in Tenni Theurer's blog post Browser Cache Usage - Exposed!, 40-60% of daily visitors to your site come in with an empty cache. Making your page fast for these first time visitors is key to a better user experience.

top | discuss this rule

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

すぐに使えて

要点をおさえやすい

TIPS → 理由を知る

時と場合によって
判断できるようにする

ネットワークコスト

Networking *n

リクエストの数とファイルサイズを減らす

Reduce

Requests & Size

ファーストビューを速く見せる

Above the fold

画像リソースは必ず
最適化してサイズを抑える

画質を保ちながらデータサイズの圧縮効率を
高めたり、余分な情報を削除したりする

低コスト・高リターンな対策なので
これがやれてないと、かなりモッタイない

テキストリソースには最小化を適用する

いわゆるMinify処理　スペースや改行などを取り除いて、ファイルサイズを小さくする

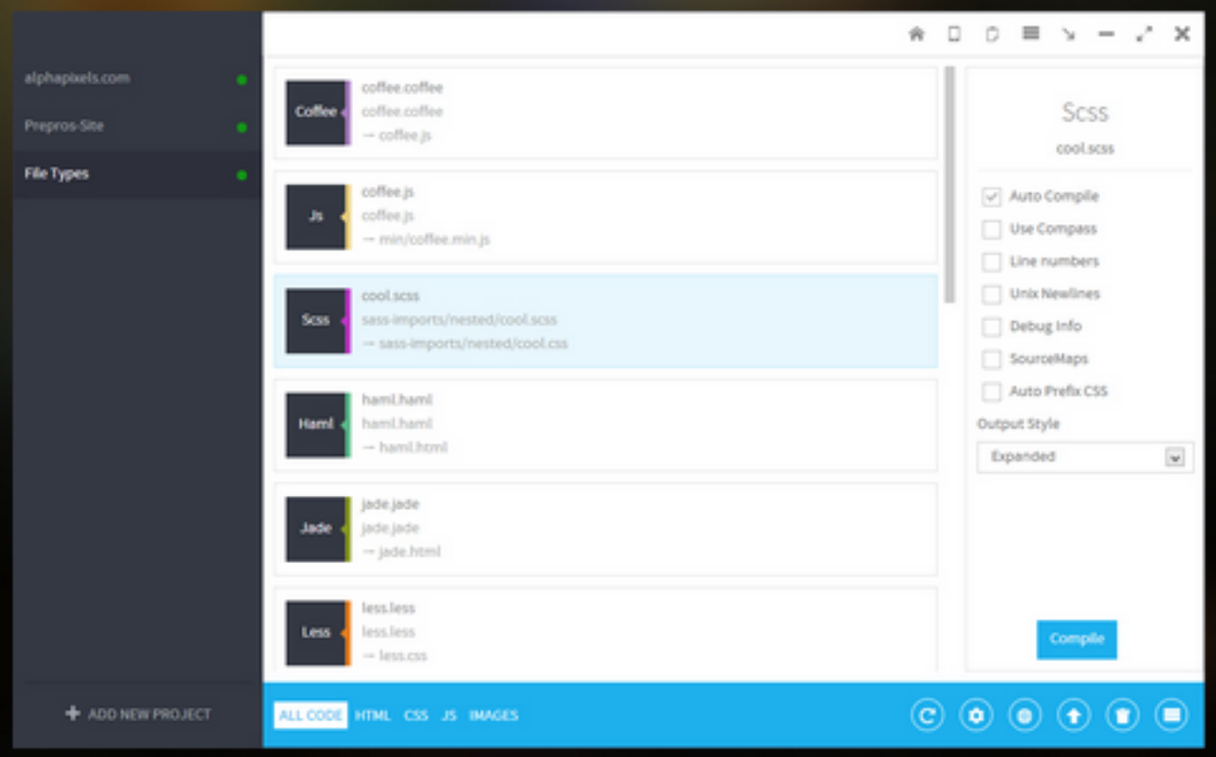
更新するたびに手動でやらない
必ずツールを使って自動化する

Prepros App

Buy Docs FAQ Issues ChangeLog Contact

Prepros App

Preprocessing. Live Browser Refreshing. Multi Device Testing and Much More.

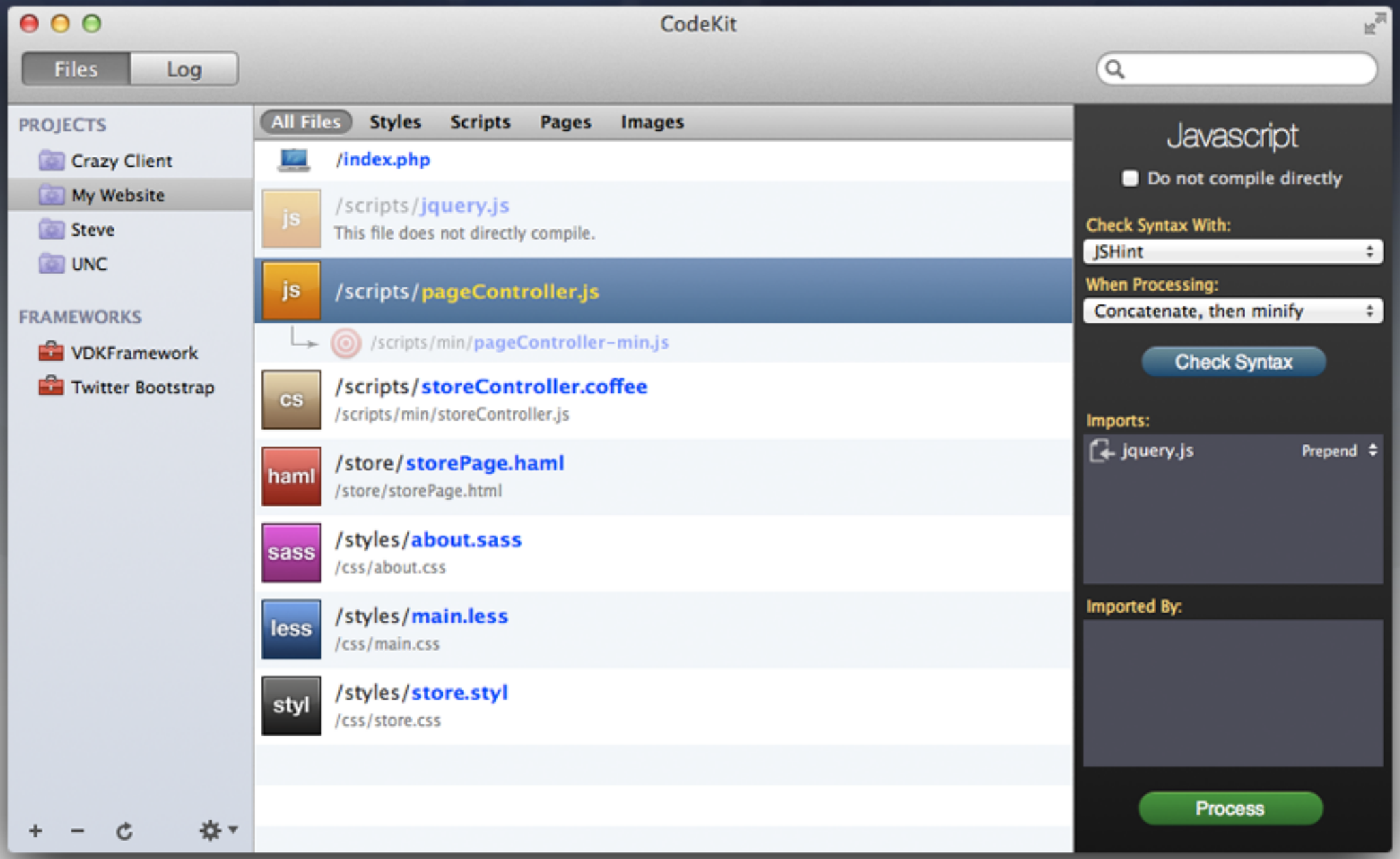


Prepros 4.0.1 Windows XP+

Prepros 4.0.1 OSX 10.8+

http://alphapixels.com/prepros/

CodeKit



It's like steroids for web developers.

http://incident57.com/codekit/

GRUNT

The JavaScript Task Runner

Why use a task runner?

In one word: automation. The less work you have to do when performing repetitive tasks like minification, compilation, unit testing, linting, etc, the easier your job becomes. After you've configured it, a task runner can do most of that mundane work for you—and your team—with basically zero effort.

Why use Grunt?

The Grunt ecosystem is huge and it's growi every day. With literally hundreds of plugins choose from, you can use Grunt to automat just about anything with a minimum of effort someone hasn't already built what you need authoring and publishing your own Grunt pl to npm is a breeze.

Latest Version

- Stable: v0.4.2
- Development: N/A

Latest News

Grunt 0.4.2 released
November 21, 2013

Grunt 0.4.1 released
March 13, 2013

Grunt 0.4.0 released
February 18, 2013

Available Grunt plugins

Many of the tasks you need are already available as Grunt Plugins, and new plugins are published every day. While the plugin listing is more complete, here's a few you may have heard of.

http://gruntjs.com/

CSS Spritesと
Data URIを有効に使う

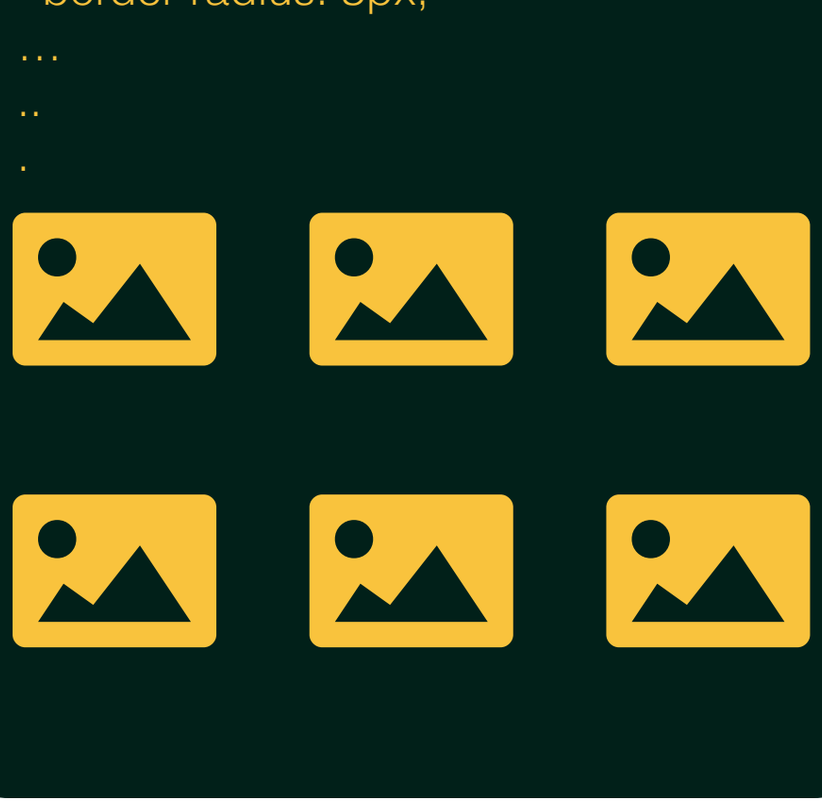
画像を1枚にまとめたり、CSSの中に文字列で
埋め込むことで、リクエスト数を節約できる

目安 30KB 以内でおさまるようにして
ひとつのファイルを大きくしすぎないよう注意

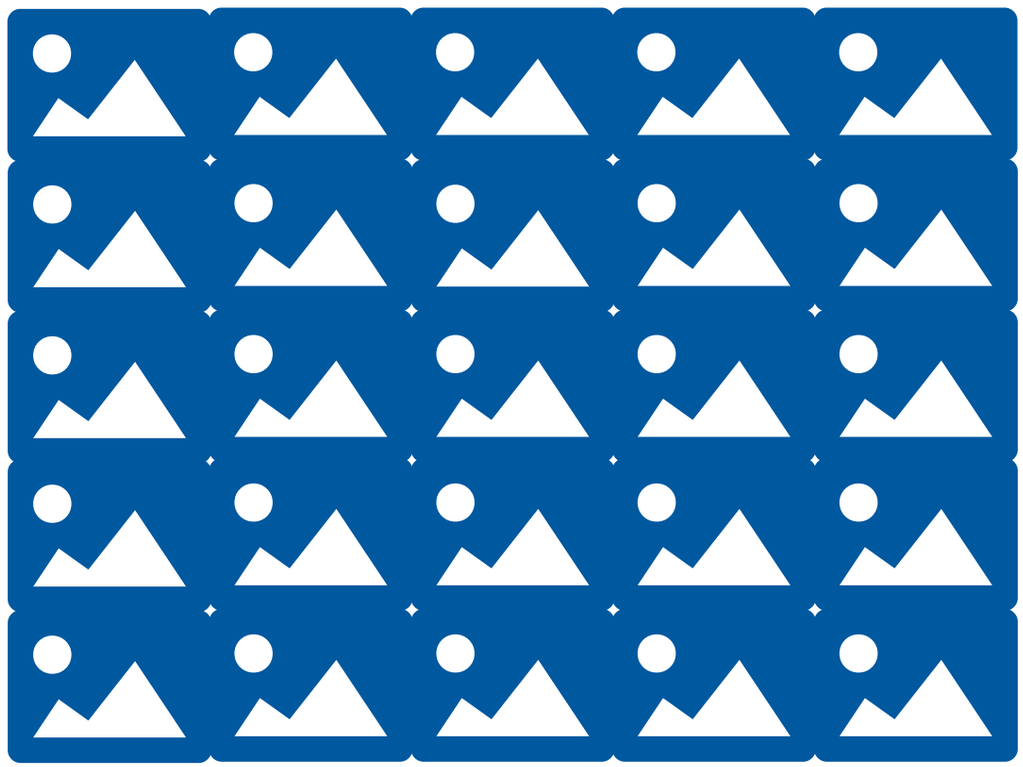
Data URI

CSS

```
div {
  width: 100px;
  height: 50px;
  border-radius: 3px;
...
..
.
```



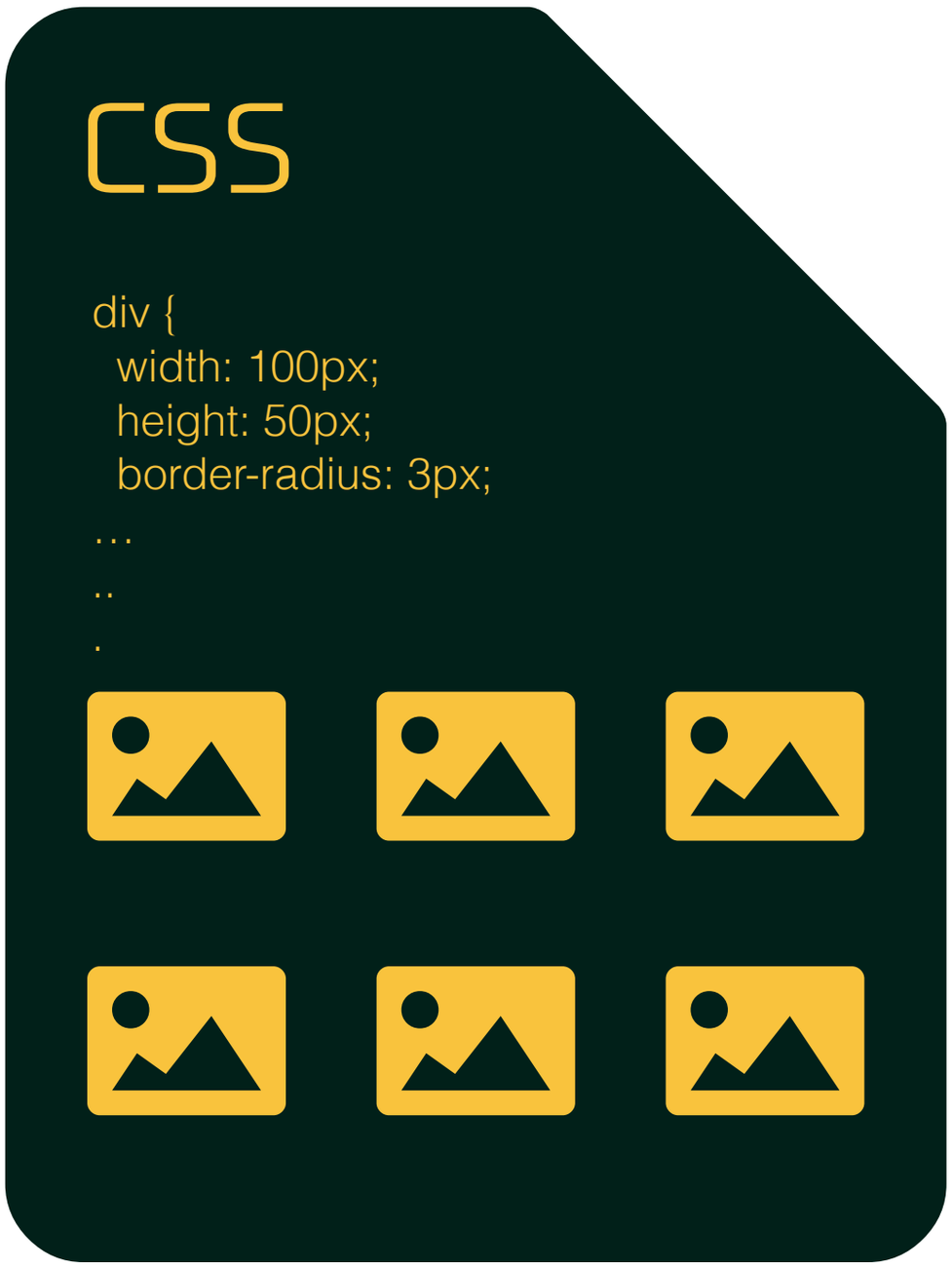
CSS Sprites



CSS

```
background-image: url(“
AQABAAD2wBDAAMCAggICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAoICAgICQoKCAg
MDAoIDAgICQj2wBDAQMEBAYFBgo
GBgoNDAwNDQ0NDQwMDQ0MDA0
MDQ0MDAwMDQwMDAwMDAwMDA
wMDAwMDAwMDAwMDAwMDAwMD
AwMDAzwAARCADIAMgDAREAAhEB
AxEB8QAHQAAAgEFAQEAAAAAAAA
AAAAAAQIAAwQFBgcICfEAEUQAAIB
AgMEBwMJBgUDBQAAAAECAAMRB
BIhBQYxQQcIE1FhcZEiMoEjQlJicqGx
wfAUM4KissIVNFPR8WOSoyRDc4Ph8
QAGwEAAgIDAQAAAAAAAAAAAAAA
AAECAwQFBgf/xAAvEQACAgEEAg…
```

バイナリデータを
文字列として埋め込む



もし1MBもあると…

1MB読み込まれるまで

スタイルが適用されない

Data URIの画像も表示されない

\(^o^)/

すべての画像をRetina向けに x2で用意しなくてもよい

大きく見せなくていいサムネイル画像は
そもそも x1 で十分なことが多い

テキスト画像やキービジュアルは
さすがに x2 じゃないと辛い

Expiresなどのヘッダーと gzip圧縮を必ず適用する

Expiresヘッダーでキャッシュを活用して
gzipの設定で転送量をおさえる

非常に初歩的だが、見逃してしまうと
ダメージが大きいので必ずチェックしたい

PageSpeed Insights +1 1.3k

http://havelog.ayumusato.com/ ANALYZE

83 / 100 Mobile | 93 / 100 Desktop

Speed

Should Fix:

Eliminate render-blocking JavaScript and CSS in above-the-fold content

▸ Show how to fix

Consider Fixing:

Leverage browser caching

▸ Show how to fix

Optimize images

▸ Show how to fix

Minify JavaScript

▸ Show how to fix

Minify CSS

▸ Show how to fix

5 Passed Rules

▸ Show details

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

Google Analytics

Settings My Account Sign out

http://havelog.ayumusato.com - http://havelog.ayumusato.c...
havelog.ayumusato.com

Reporting | Customization | Admin | Help

Find reports & more

MY STUFF

- Dashboards
- Shortcuts
- Intelligence Events

STANDARD REPORTS

- Real-Time
- Audience
  - Overview
  - Demographics
  - Interests
  - Geo
  - Behavior
  - Technology
  - Mobile
  - Custom
  - Visitors Flow
- Acquisition
  - Overview NEW

Site Speed Overview

Dec 13, 2013 - Jan 12, 2014

Email Export Add to Dashboard Shortcut

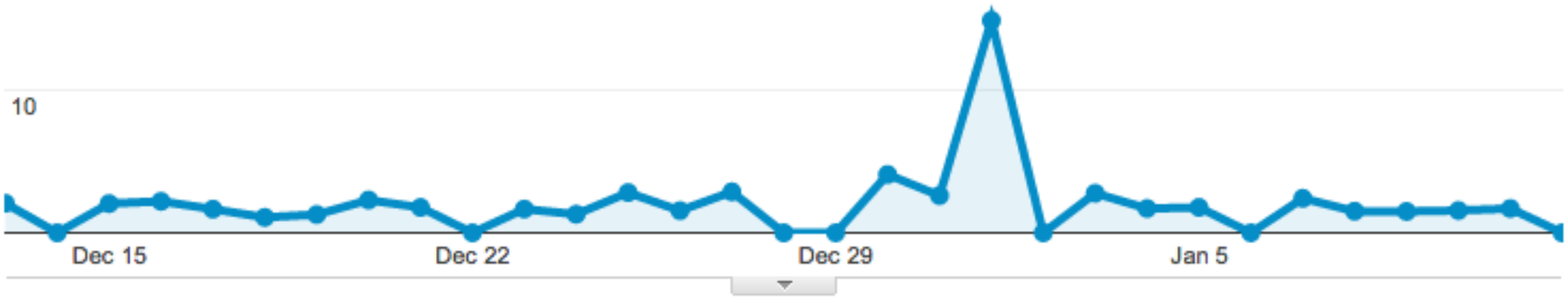
All Visits
100.00%

Overview

Avg. Page Load Time (sec) vs. Select a metric

Hourly Day Week Month

Avg. Page Load Time (sec)



112 of pageviews sent page load sample

Avg. Page Load Time (sec)	Avg. Redirection Time (sec)	Avg. Domain Lookup Time (sec)
2.11	0.03	0.05

Avg. Server Connection Time (sec)	Avg. Server Response Time (sec)	Avg. Page Download Time (sec)
0.01	0.19	0.13

www.google.com/analytics/

Web Page Performance Test for

www.cyberagent.co.jp

From: Tokyo, Japan - **Chrome - 3G**
1/13/2014 12:23:08 PM

Need help improving?

A	A	C	C	D	F	X
First Byte Time	Keep-alive Enabled	Compress Transfer	Compress Images	Progressive JPEGs	Cache static content	Effective use of CDN

Summary | **Details** | **Performance Review** | **Content Breakdown** | **Domains** | **Screen Shot**

Export HTTP Archive (.har)

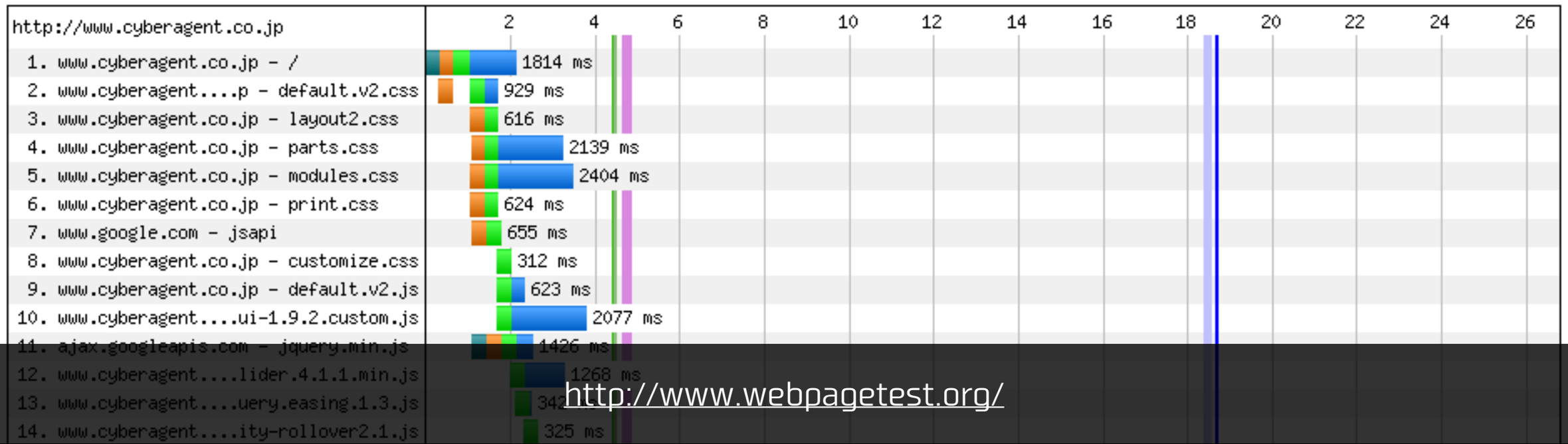
							Document Complete			Fully Loaded		
Load Time	First Byte	Start Render	Visually Complete	Speed Index	DOM Elements	Result (error code)	Time	Requests	Bytes In	Time	Requests	Bytes In
18.646s	0.993s	4.402s	28.800s	10911	842	0	18.646s	103	2,940 KB	26.745s	130	3,390 KB

msFirstPaint	domContentLoaded	loadEvent
4.458s	4.651s - 4.850s (0.199s)	18.376s - 18.526s (0.150s)

Waterfall View

DNS Lookup | Initial Connection | SSL Negotiation | Time to First Byte | Content Download | 3xx response | 4xx+ response

Start Render | msFirstPaint | DOM Content Loaded | On Load | Document Complete



http://www.webpagetest.org/

レンダリング

Rendering *n

複雑にしない

Avoid Complexities

レイアウト・再配置が
遅れて発生しないようにする

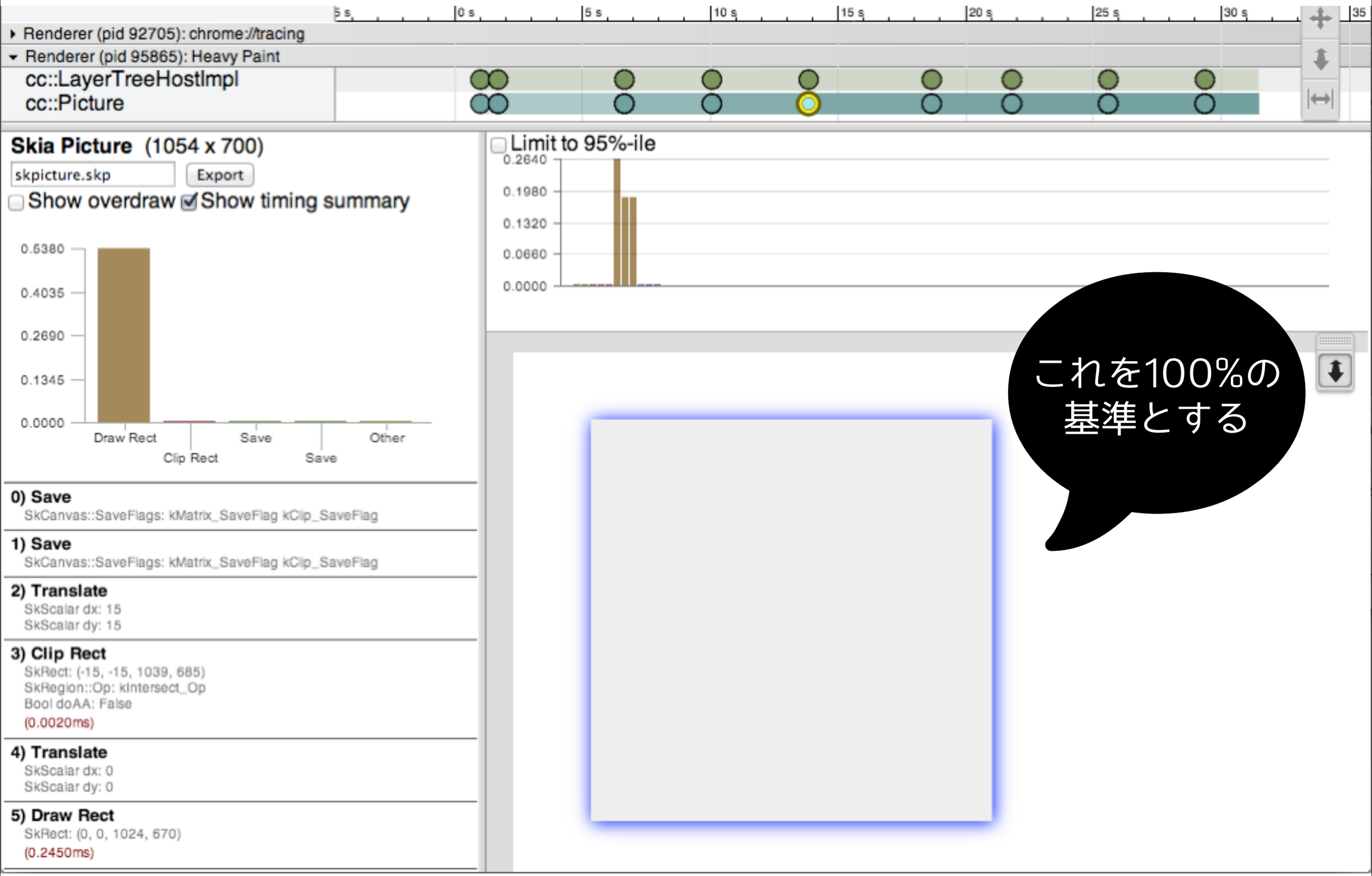
レスポンシブな画像も 解像度ごとに
height だけでも決めれば ガコってならない

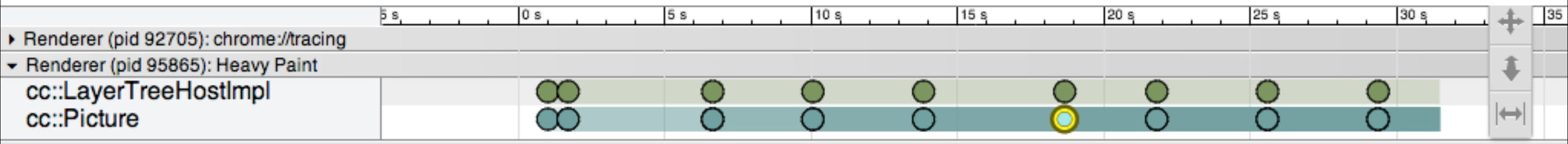
例えば <img> の width, height がないと
画像読み込み時にガクっとなってしまう

高コストなCSS3は慎重に取り扱う

box-shadow や border-radius などの
いわゆる CSS3 は描画コストが低くない

組み合わせると、描画パスが複雑になるので
特に描画パフォーマンスが劣化しやすい

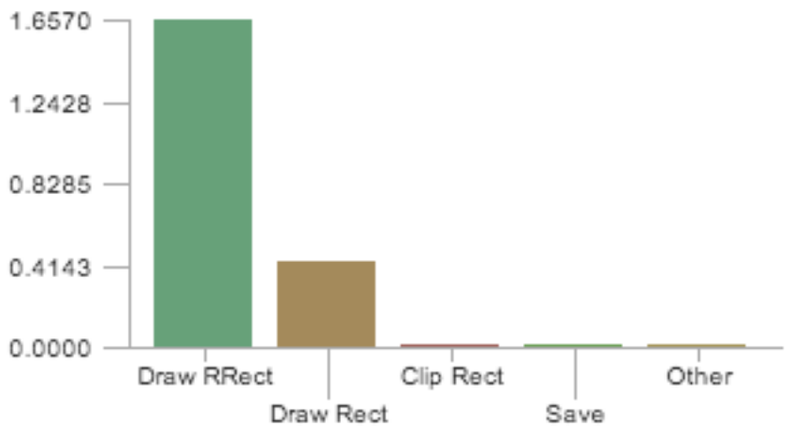

5 s
0 s
5 s
10 s
15 s
20 s
25 s
30 s
35
Renderer (pid 92705): chrome://tracing
Renderer (pid 95865): Heavy Paint
cc::LayerTreeHostImpl
cc::Picture
Skia Picture (1054 x 700)
skpicture.skp
Export
Show overdraw
Show timing summary
0.6380
0.4035
0.2690
0.1345
0.0000
Draw Rect
Clip Rect
Save
Save
Other
Limit to 95%-ile
0.2640
0.1980
0.1320
0.0660
0.0000
0) Save
SkCanvas::SaveFlags: kMatrix_SaveFlag kClip_SaveFlag
1) Save
SkCanvas::SaveFlags: kMatrix_SaveFlag kClip_SaveFlag
2) Translate
SkScalar dx: 15
SkScalar dy: 15
3) Clip Rect
SkRect: (-15, -15, 1039, 685)
SkRegion::Op: kIntersect_Op
Bool doAA: False
(0.0020ms)
4) Translate
SkScalar dx: 0
SkScalar dy: 0
5) Draw Rect
SkRect: (0, 0, 1024, 670)
(0.2450ms)
これを100%の
基準とする



Skia Picture (1054 x 700)

skpicture.skp Export

Show overdraw Show timing summary



0) Save
SkCanvas::SaveFlags: kMatrix_SaveFlag kClip_SaveFlag

1) Save
SkCanvas::SaveFlags: kMatrix_SaveFlag kClip_SaveFlag

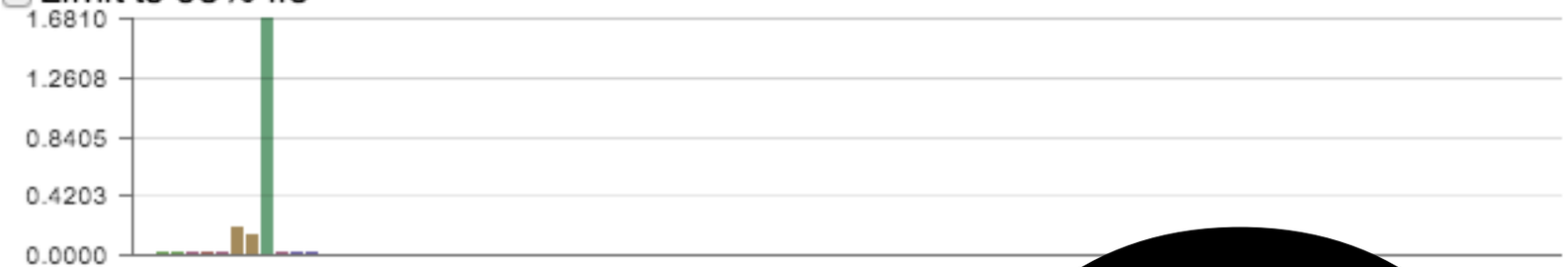
2) Translate
SkScalar dx: 15
SkScalar dy: 15

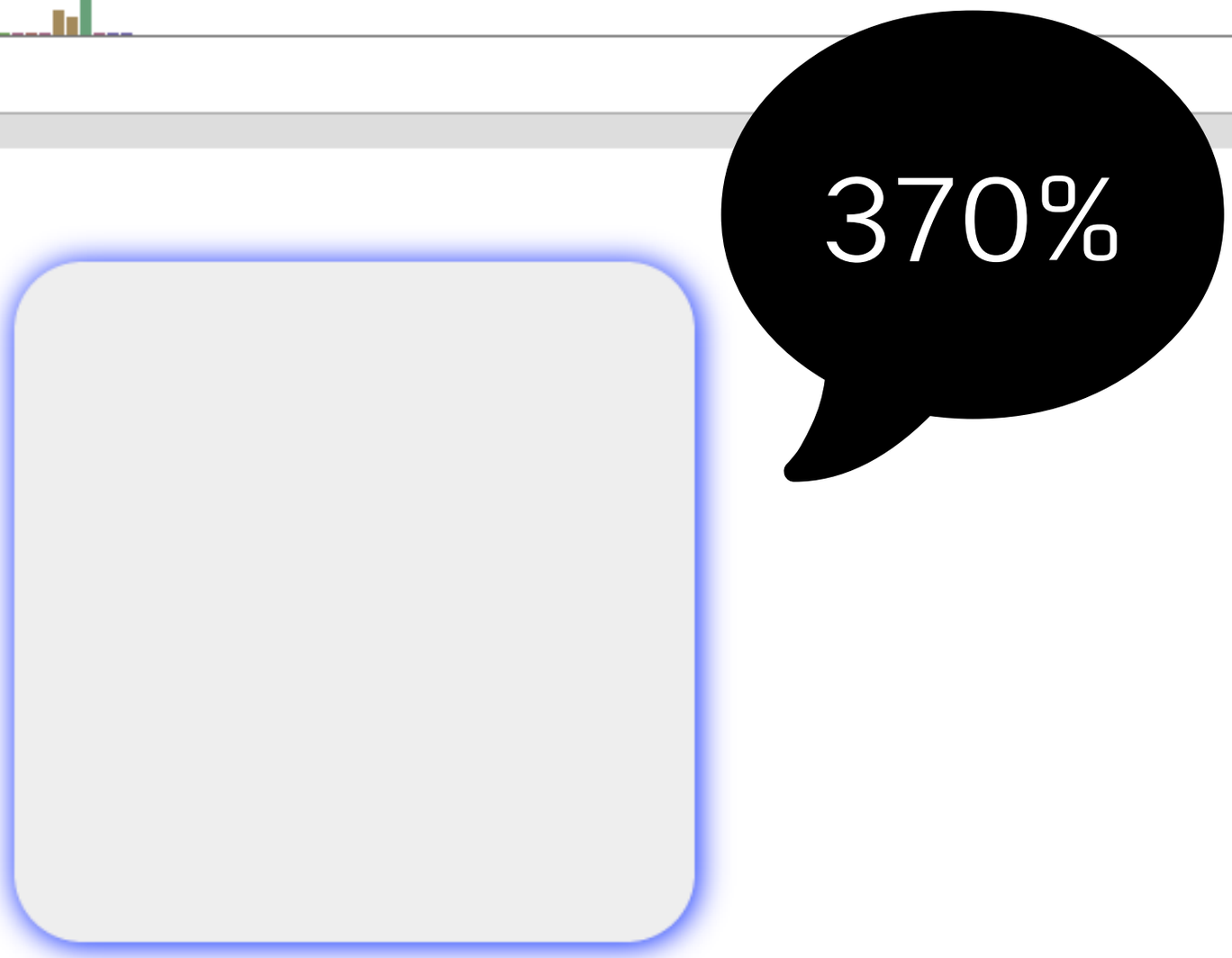
3) Clip Rect
SkRect: (-15, -15, 1039, 685)
SkRegion::Op: kIntersect_Op
Bool doAA: False
(0.0020ms)

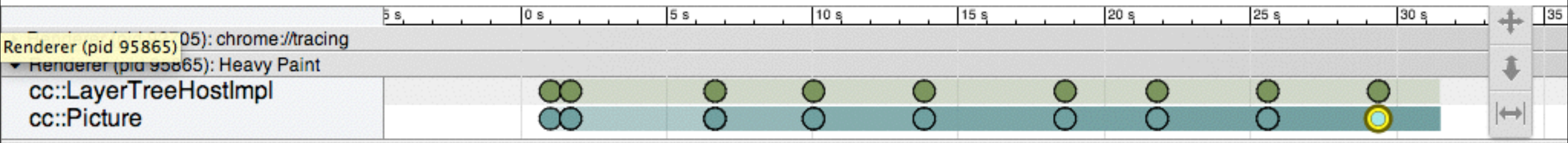
4) Translate
SkScalar dx: 0
SkScalar dy: 0

5) Draw Rect
SkRect: (0, 0, 1024, 670)
(0.2500ms)

Limit to 95%-ile



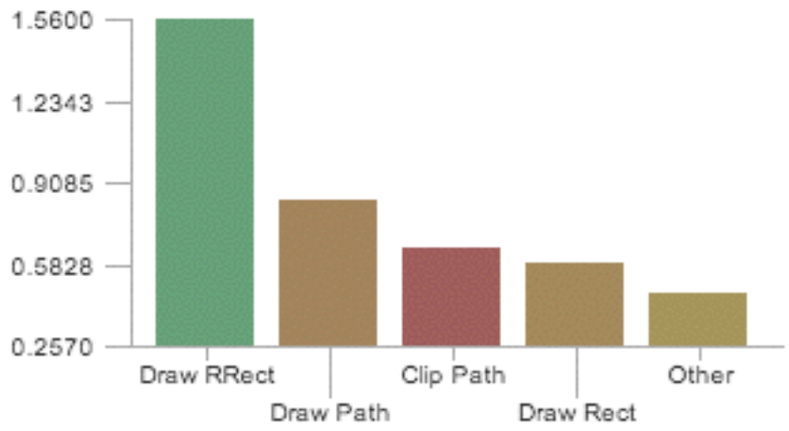




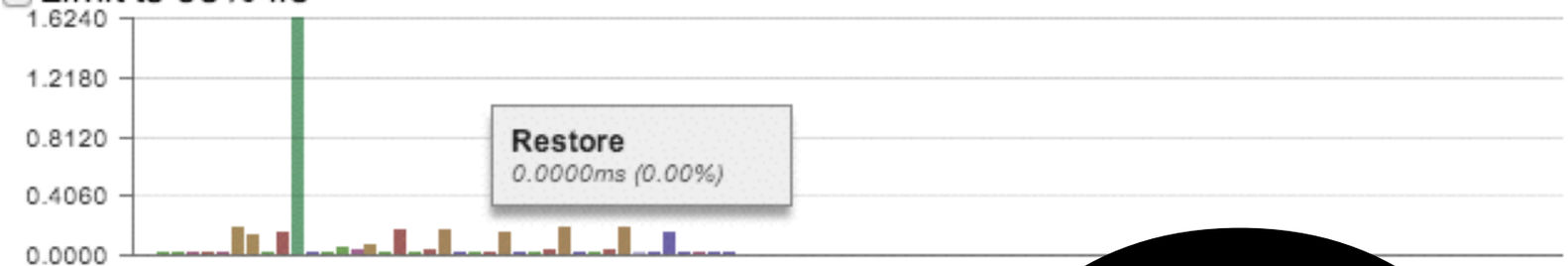
Skia Picture (1054 x 700)

skpicture.skp Export

Show overdraw Show timing summary



Limit to 95%-ile



660%

0) Save
SkCanvas::SaveFlags: kMatrix_SaveFlag kClip_SaveFlag

1) Save
SkCanvas::SaveFlags: kMatrix_SaveFlag kClip_SaveFlag

2) Translate
SkScalar dx: 15
SkScalar dy: 15

3) Clip Rect
SkRect: (-15, -15, 1039, 685)
SkRegion::Op: kIntersect_Op
Bool doAA: False
(0.0010ms)

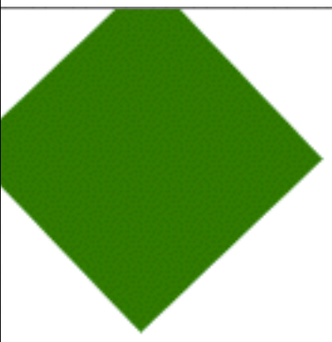
4) Translate
SkScalar dx: 0
SkScalar dy: 0

5) Draw Rect
SkRect: (0, 0, 1024, 670)
(0.1830ms)

GPUアクセラレーションが適用されるCSSを利用する

translate3d などGPUアクセラレーションが
適用されるCSSを使うと滑らかに動く

おまじないとして使いすぎると
キツいしっぺ返しがある



Accidental layer creation

Click the box above, it spins, woo! Click somewhere else, it stops, boo!

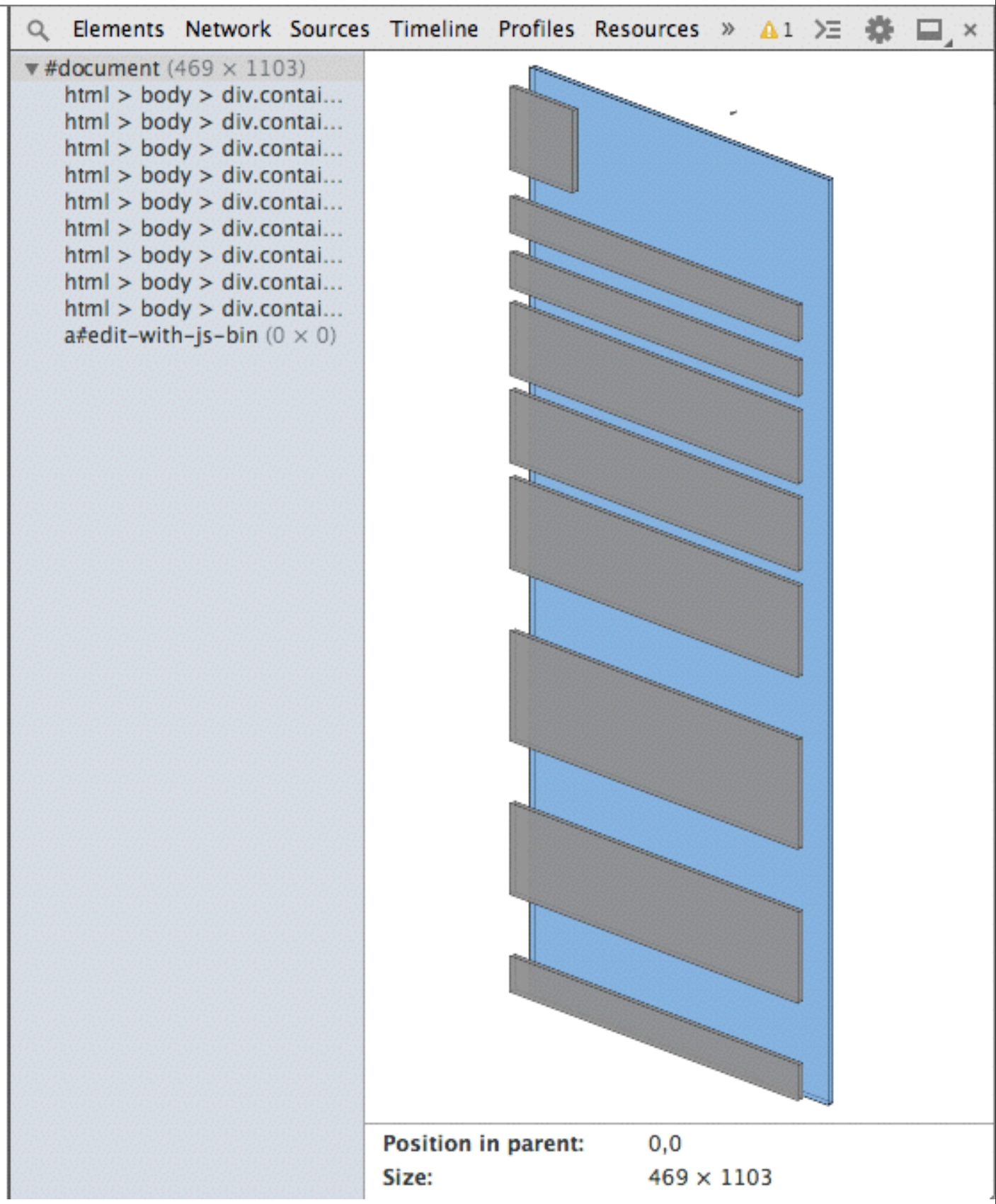
Take a close look at this text when you start and stop the box spinning, it changes ever so slightly, you may have to zoom in to see it. It's losing subpixel antialiasing. This is a symptom of text getting its own texture layer on the GPU.

Fire up this page in Chrome Canary, and turn on "Show composited layer borders" in devtools settings. You'll see the text getting an orange border when the box is spinning, confirming its getting its own texture-backed layer.

Aside from the minor text rendering change, this isn't a big deal on desktop, creating textures and uploading to the GPU is pretty cheap. However, mobile devices often don't have such a friendly relationship with the GPU, and texture creation isn't as cheap.

Why is this happening?

The box has `position: relative` but so do the headings and



CODEPEN | Fork | Share | Editor | Login | Sign up

CSS

```
@keyframes move {
  50% {
    top: 200px;
    left: 200px;
  }
}


.macbook {
    background-color: #082746;
    background-image: linear-gradient(70deg,
transparent 67%, hsla(0,0%,100%,.05) 67%,
```

add 10 more macbooks | clear

with Top/Left

CodePen PRO is pretty sweet. Just saying. | Embed This Pen | A Pen by Paul Irish | ⌘ Shortcuts

Elements Network Sources Timeline Profiles Resources Audits Console Layers 1

0 of 0 frames shown

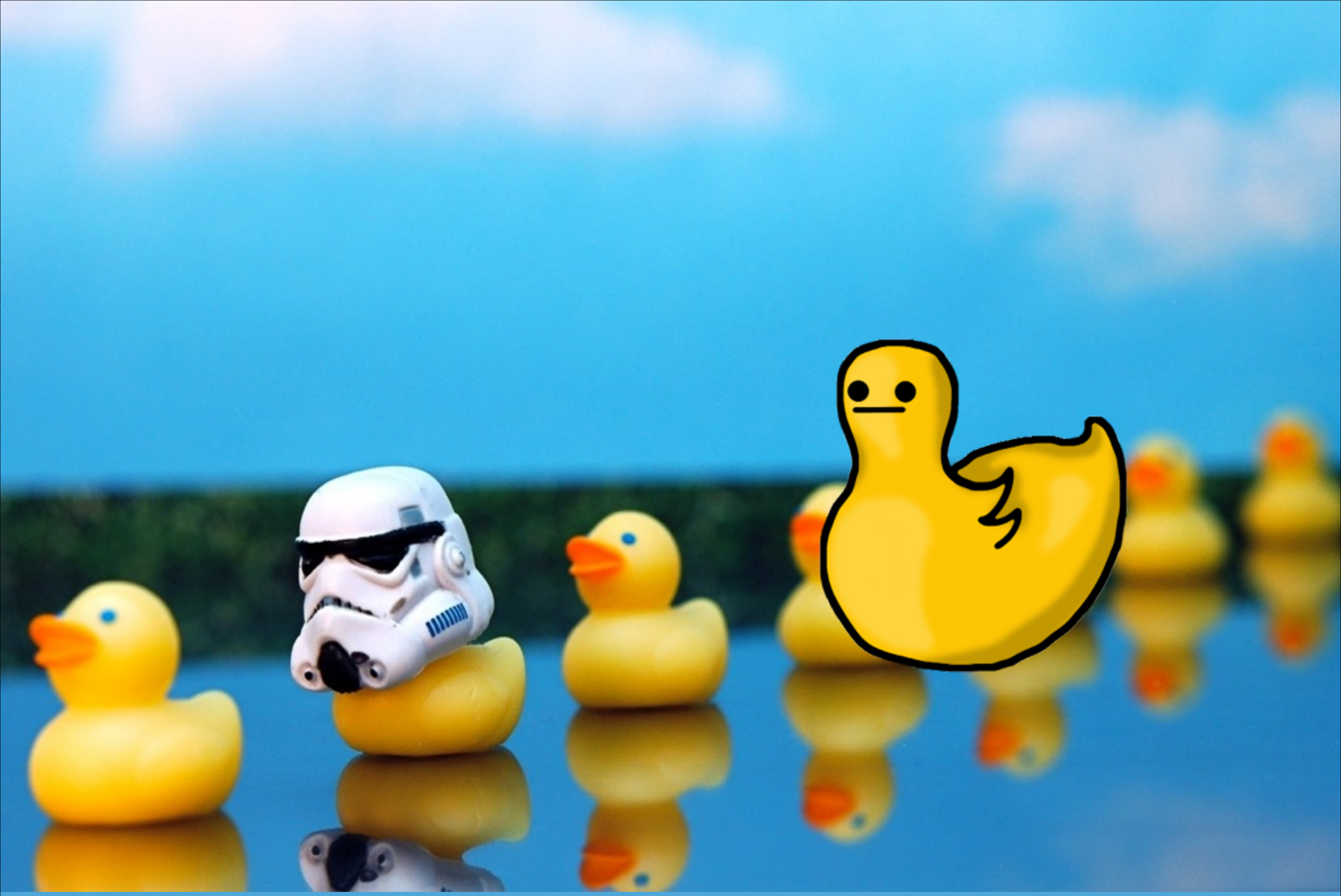
Events

Frames

Memory

RECORDS

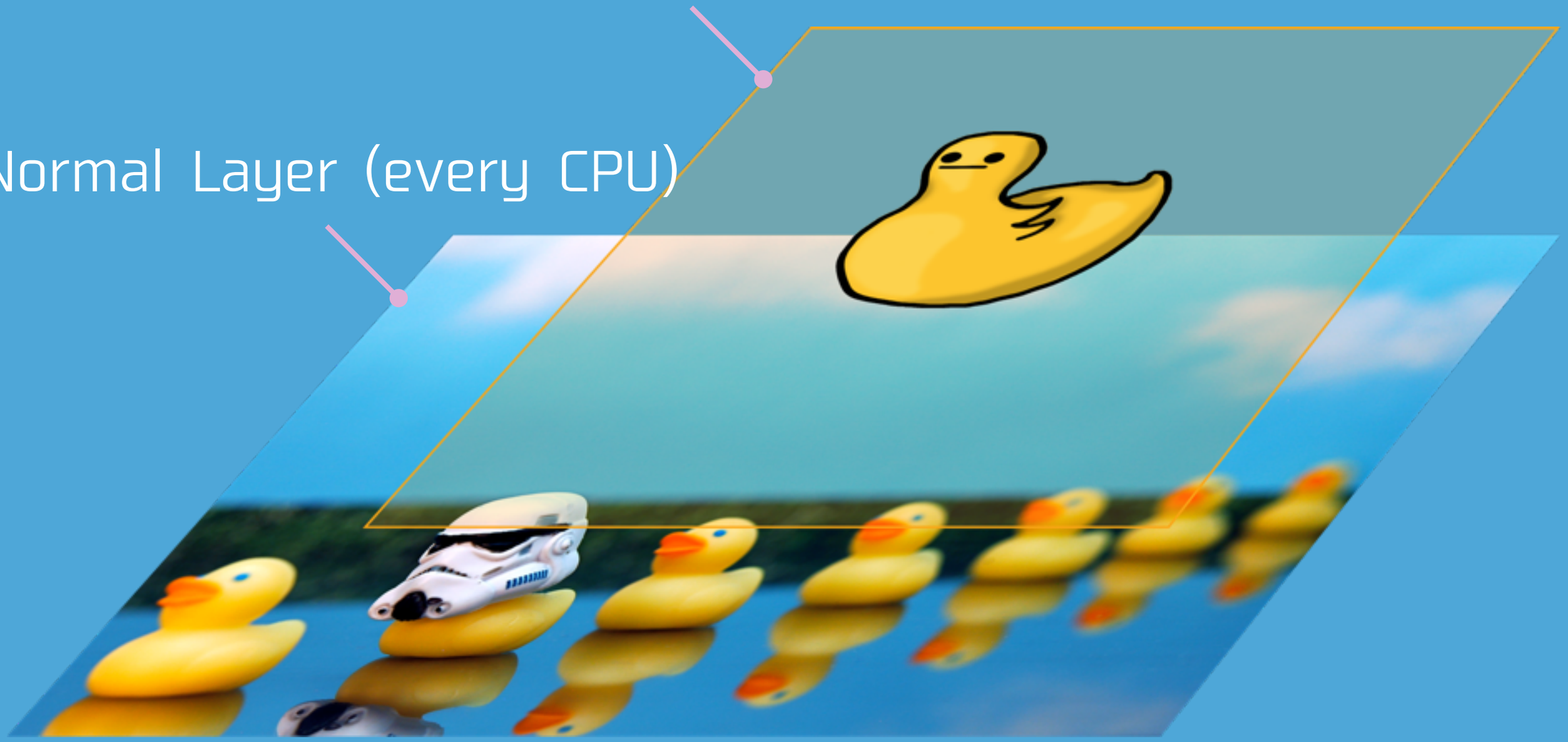
DETAILS



https://developers.google.com/events/io/sessions/325091862

Compositing Layer (on GPU)

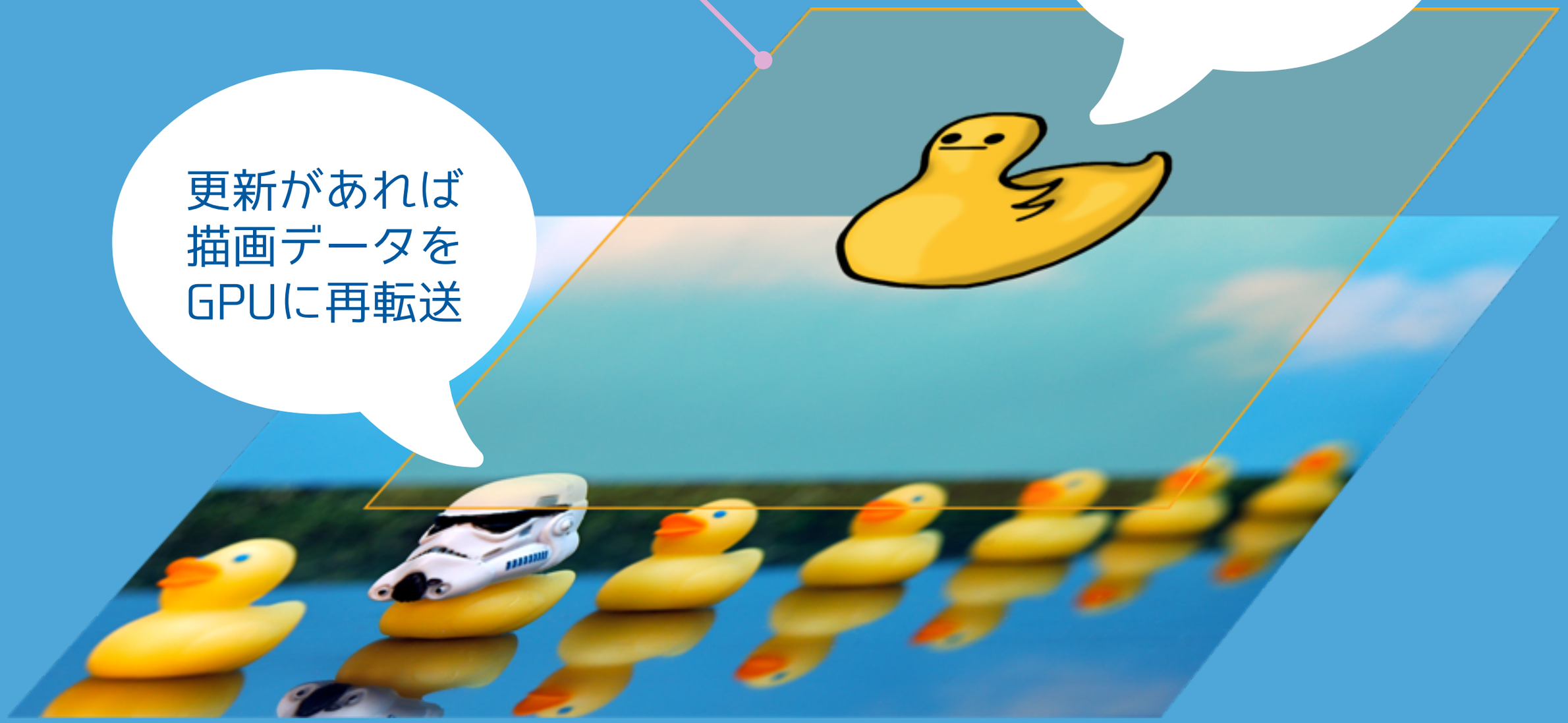
Normal Layer (every CPU)



GPU Texture

描画データを
キープしといて
GPU上で合成

更新があれば
描画データを
GPUに再転送



レイヤー丸ごとを
操作させるだけで
サクサク移動

広い範囲を
再描画しないから
スムーズ

スクリプト処理

Scripting *n

ブラウザの気持ちになって書く

Think Computing

jQueryのセレクタを
きちんとキャッシュする

jQueryにセレクタを渡して要素を選択するとき
探索結果を返すので、それをキャッシュする

キャッシュしないと、毎回セレクタを走らせて
要素を探索していることになってしまう

```
$(‘#list’).append(‘<li>Item1</li>’);
$(‘#list’).append(‘<li>Item2</li>’);
$(‘#list’).append(‘<li>Item3</li>’);
$(‘#list’).append(‘<li>Item4</li>’);
$(‘#list’).append(‘<li>Item5</li>’);

<ul id=“list”></ul>
```

```
var $list = $(‘#list’);
$list.append(‘<li>Item1</li>’);
$list.append(‘<li>Item2</li>’);
$list.append(‘<li>Item3</li>’);
$list.append(‘<li>Item4</li>’);
$list.append(‘<li>Item5</li>’);

<ul id=“list”></ul>
```

```
var $list = $(‘#list’);
$list.append(‘<li>Item1</li>’);
$list.append(‘<li>Item2</li>’);
$list.append(‘<li>Item3</li>’);
$list.append(‘<li>Item4</li>’);
$list.append(‘<li>Item5</li>’);

<ul id=“list”></ul>
```

```
var $list = $(‘#list’);
// ちょっと極端ですが…
$list.append(‘<li>Item1</li>’ +
             ’<li>Item2</li>’ +
             ‘<li>Item3</li>’ +
             ‘<li>Item4</li>’ +
             ‘<li>Item5</li>’);
```

```
<ul id=“list”></ul>
```

イベントは Delegate を使ってまとめる

沢山ある・動的に増える可能性がある要素へのイベントは、Delegate(委譲)を使う

数百セルあるエクセル風なフロント実装を想像…

```
// たくさんの TD へ、個別にイベントを貼る
$(‘table td’).on(‘click’, function() {
  // do something
});

// ひとつの table にイベントを貼る
$(‘table’).on(‘click’, ‘td’, function() {
  // do something
});
```

```
// たくさんの TD へ、個別にイベントを貼る
$(‘table td’).on(‘click’, function() {
  // do something
});


<table>
<tr>
  <td>セル1</td>
  <td>セル2</td>
  <td>セル3</td>
</tr>
</table>
```

```
// ひとつの TABLE にイベントを貼る
$(‘table’).on(‘click’, ‘td’, function() {
  // do something
});


<table>
<tr>
  <td>セル1</td>
  <td>セル2</td>
  <td>セル3</td>
<tr>
</table>
```

TABLE がイベントを
受け取って
イベント発生源が
TD か確認する

scroll, resizeのイベントを間引いて処理する

throttle

連続呼び出しをnミリ秒の周期に間引いて実行

debounce

nミリ秒以内の連続呼び出しを無視して
呼び出しがおさまったら実行

```
/* Underscore.JS */

// scroll に合わせて位置を更新する何か
var throttled = _.throttle(updatePos, 100);
$(window).scroll(throttled);

// resize に反応してレイアウトし直す何か
var lazyLayout = _.debounce(calcLayout,300);
$(window).resize(lazyLayout);
```

100msごとに発火するイベント

throttle 200ms

debounce 200ms

意味の薄いベンチマークに惑わされない・気にしない

些細すぎるパフォーマンスよりは
記法の統一性・可読性が優先されるべき

速い・遅いのベンチマーキングは非常に大事
けど、その処理をコールする頻度を考える

メモリリークの回避
JSエンジンの最適化考慮

[E-3] 45MIN. 16:00 - 16:45
Browser Computing Structure

詳しくは、この次の次のセッションで！

でもちょっとだけ…

Use forensics and detective work to solve JavaScript performance mysteries

By John McCutchan

Published: June 13th, 2013
Comments: 10

Table of Contents

Introduction

In recent years, web applications have been sped up considerably. Many applications now run fast enough that I've heard some developers wonder aloud "is the web fast enough?". For some applications it may be, but, for the developers working on high performance applications, we know it is not fast enough. Despite the amazing advances in JavaScript virtual machine technology, a recent study showed that Google applications spend between 50% and 70%

http://www.html5rocks.com/en/tutorials/performance/mystery/

```
function updateSprites(dt) {
  for (var sprite in sprites) {
    sprite.position.x += 0.5 * dt;
    // 20 more lines of arithmetic computation.
  }
}
```

```
function updateSprites(dt) {
  for (var sprite in sprites) {
    updateSprite(sprite, dt);
  }
}

function updateSprite(sprite, dt) {
  sprite.position.x += 0.5 * dt;
  // 20 more lines of arithmetic computation.
}
```

ループ内の処理を別関数に分離することで
最適化が適用されるようになった (らしい)

インタラクション

Interaction *n

主にモバイル系の話

in Mobile

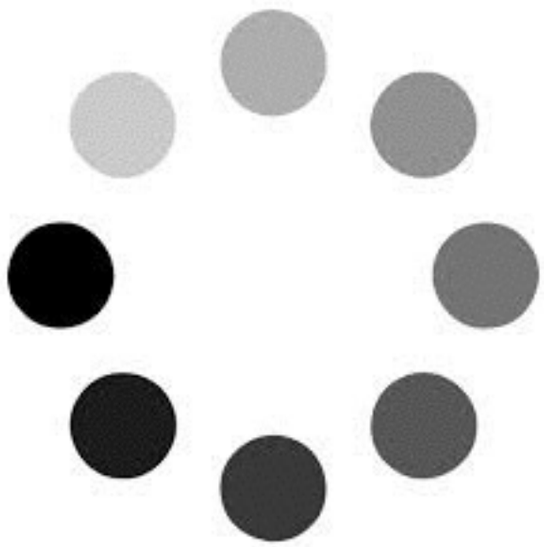
遅延ロード時の
低解像度プレースホルダー

デフォルトでサイズ逆算2-3KBの低解像度
スクロール遅延で本来の高解像度にさしかえ

ネットワーク的にリクエスト数はもちろん増える
けど、体感を上げられるとこでは使うべき

ローディング表示は
200-300ms待って出す

それ以下であれば、ローディング表示が明滅してしまうだけで逆に見栄えが良くない



こういうスピナー系のやつ

```
var timer;

$(document).on('ajaxSend', function(){
  timer = setTimeout(function() {
    $('#loading').show();
  }, 300);
});
```

```
$(document).on('ajaxComplete', function(){
  clearTimeout(timer);
  $('#loading').hide();
});
```

タップ時の状態表現を徹底的につける

ブラウザ標準の tap-highlight-color でもよいし
touchstart ~ end で class を着脱してもよい

tap-highlight-color を rgba(0,0,0,0) に
設定したまま無反応な UI にするのは最悪

```
// タッチ開始時に、is-activeクラスを付与する
$(document).on('touchstart','a',function() {
  $(this).addClass('is-active');
});

// タッチ終了時に、is-activeクラスを解除する
$(document).on('touchend', 'a', function() {
  $(this).removeClass('is-active');
});

// 厳密には、UIの種類によって、クラス着脱のときに
// ディレイをかけてチューニングしないと自然にならない
```

画面全体のトランジションは コスパ悪いので避ける

やればできるが、広い範囲をサポートするには
検証コスト・不具合対応コストが高い

jQuery Mobile ですら 1.0 -> 1.1 で地味になり
1.4のデモは更に消極的な感じになっている

モバイルにおけるclickの300msディレイを知る

モバイルブラウザはダブルタップの判定のため
タップしてから click の発生が 300ms 遅延する

ライブラリを使って改善することはできる
単なる touchstart, touchend では過敏すぎる

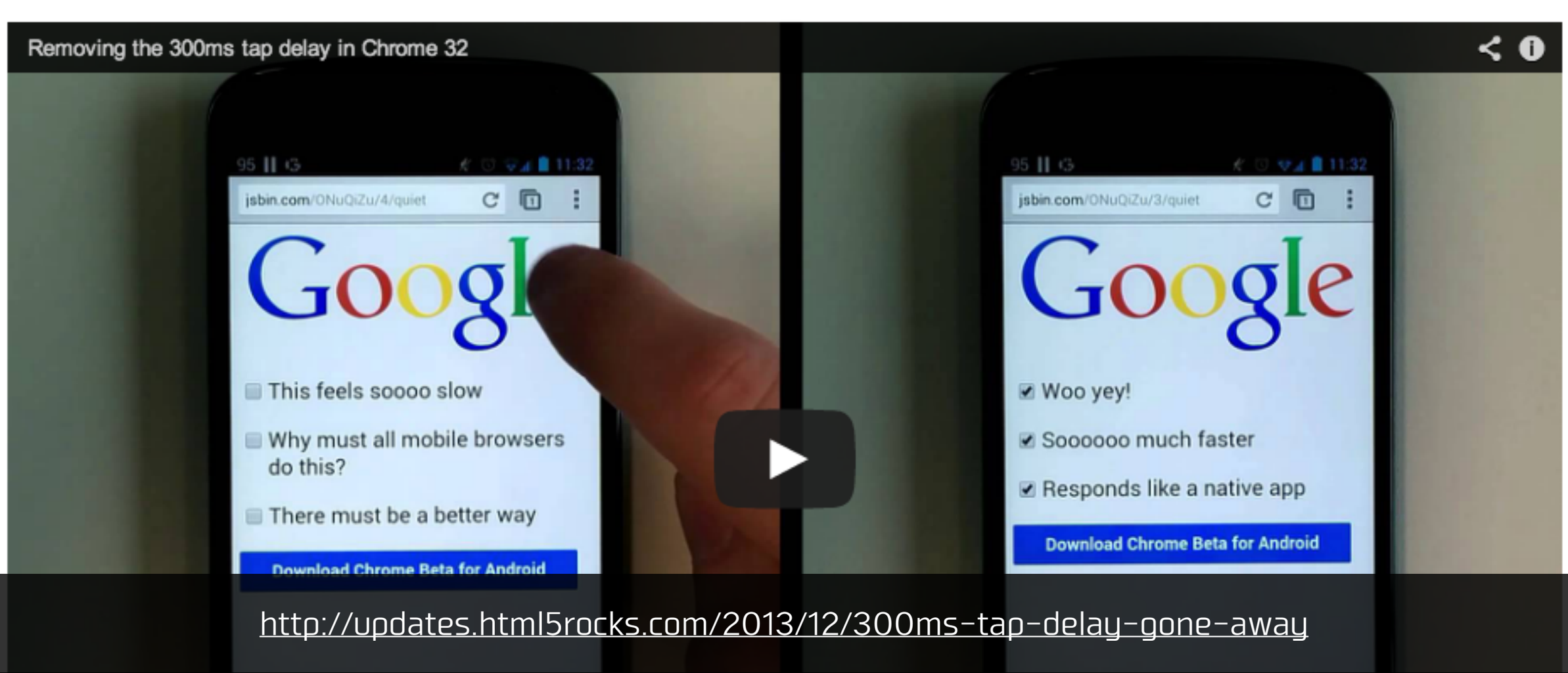
300ms tap delay, gone away

By Jake Archibald at 13 December, 2013

mobile performance events touch

You'll find large articles throughout this site dedicated to shaving 10ms here and 90ms there in order to deliver a fast and fluid user experience. Unfortunately every touch-based mobile browser, across platforms, has an artificial ~300ms delay between you tapping a thing on the screen and the browser considering it a "click". When people think of the web as being sluggish compared to native apps on mobile, this is this one of the main contributors.

However, as of Chrome 32 for Android, which is currently in beta, this **delay is gone** for **mobile-optimised** sites, **without removing pinch-zooming!**



Mobile

5 Ways to Prevent the 300ms Click Delay on Mobile Devices

Craig Buckler
Contributing Editor
Published January 8, 2014

Most touch-based mobile browsers wait 300ms between your tap on the screen and the browser firing the appropriate handler for that event. It was implemented because you could be double-tapping to zoom the page to full width. Therefore, the browser waits for a third of a second — if you don't tap again, the "click" is activated.

The delay was a sensible precaution in the days before Responsive Web Design and multi-touch pinch zooming. Unfortunately, it's now one of the primary reasons users consider web applications to be slower and less capable than native alternatives. So

User Experience BETA

Not currently a part of the overall score

Consider Fixing:

Size tap targets appropriately

Some of the links/buttons on your webpage may be too small for a user to easily tap on a touchscreen. Consider making these tap targets larger to provide a better user experience.
The following tap targets are small. Consider increasing their size.

The tap target `<a href="" class="bx-pager-link">1</a>` and 4 others are only 0.94mm tall on a typical device (6 CSS pixels) (see screenshot).

The tap target `<a href="" class="bx-pager-link active">2</a>` and 1 others are only 0.94mm tall on a typical device (6 CSS pixels) (see screenshot).

Hide details

4 Passed Rules

Hide details

Avoid plugins

Your page does not appear to use plugins, which would prevent content from being usable on many platforms. Learn more about the importance of avoiding plugins.

Configure the viewport

Your page specifies a viewport matching the device's size, which allows it to render properly on all devices. Learn more about configuring viewports.

Size content to viewport

The contents of your page fit within the viewport. Learn more about sizing content to the viewport.

Use legible font sizes

The text on your page is legible. Learn more about using legible font sizes.

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

User Experience BETA

Not currently a part of the overall score

Consider Fixing:

Size tap targets appropriately

Some of the links/buttons on your webpage may be too small for a user to easily tap on a touchscreen. Consider making these tap targets larger to provide a better user experience.
The following tap targets are small. Consider increasing their size.

The tap target `<a href="" class="bx-pager-link">1</a>` and 4 others are only 0.94mm tall on a typical device (6 CSS pixels) (see screenshot).

The tap target `<a href="" class="bx-pager-link active">2</a>` and 1 others are only 0.94mm tall on a typical device (6 CSS pixels) (see screenshot).

Hide details

タップサイズが小さいという警告

4 Passed Rules

Hide details

Avoid plugins

Your page does not appear to use plugins, which would prevent content from being usable on many platforms. Learn more about the importance of avoiding plugins.

Configure the viewport

Your page specifies a viewport matching the device's size, which allows it to render properly on all devices. Learn more about configuring viewports.

Size content to viewport

The contents of your page fit within the viewport. Learn more about sizing content to the viewport.

Use legible font sizes

The text on your page is legible. Learn more about using legible font sizes.

まとめ

Conclusion

エンジニアリング
X
インタラクション

エンジニアリングの
すべてを知ることは
難しい

インタラクションの
すべてを検証することも
難しい

両者の良いバランスを探るフロントエンド

Thanks!

http://aho.mu

@ahomu

github.com/ahomu

Photos by…

クレヨン http://www.flickr.com/photos/laffy4k/404298099

Introduction http://www.flickr.com/photos/jinterwas/3906565085/

Networking http://www.flickr.com/photos/kewl/8475764430/

Rendering http://www.flickr.com/photos/jdhancock/6151250051/

Scripting http://www.flickr.com/photos/nyuhuhuu/3367743012/

Intaraction http://www.flickr.com/photos/stokedmediaphotography/8176660602/

Conclusion http://www.flickr.com/photos/saad/1968774/