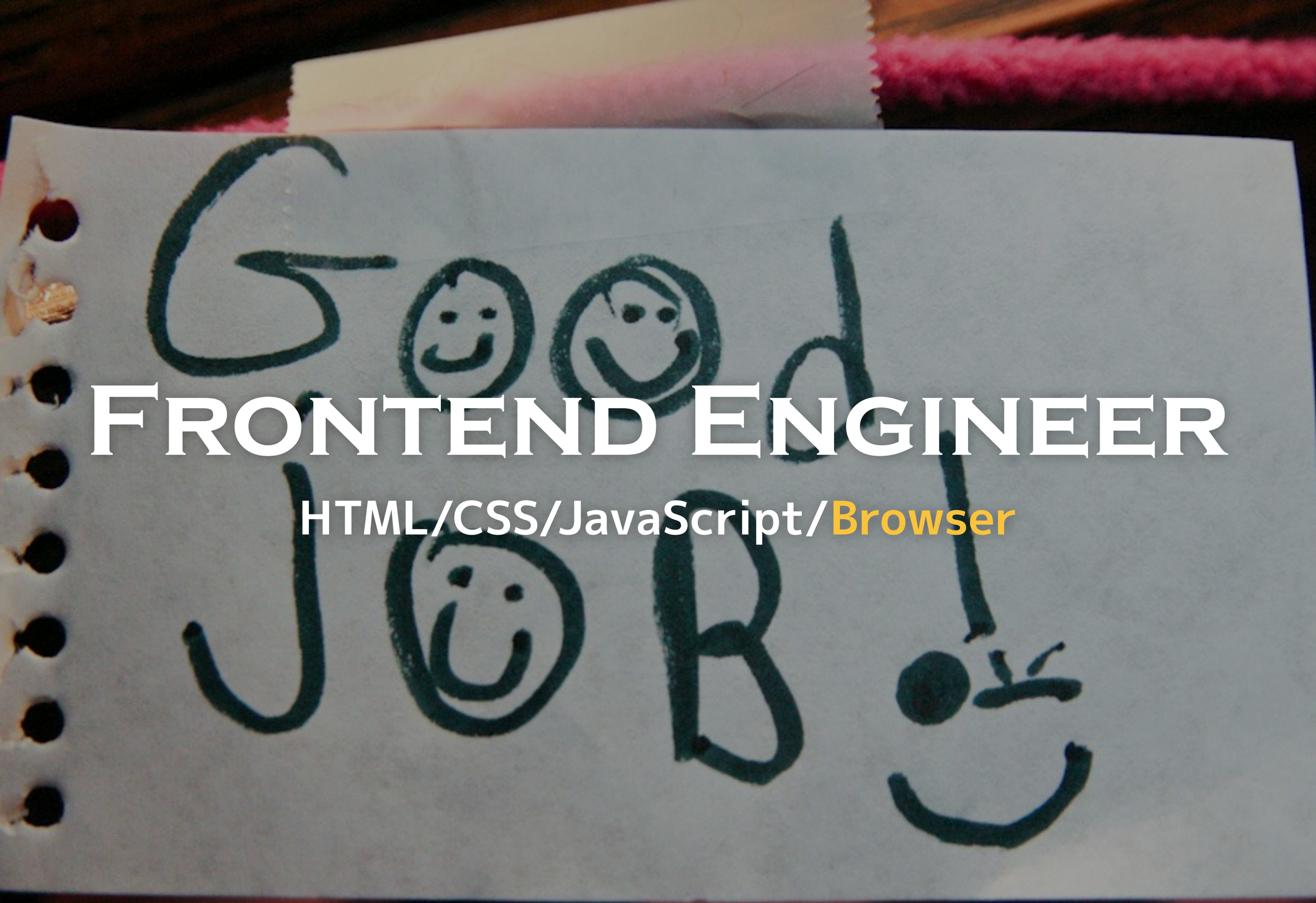
High Performance Web Frontend

Re:Creator’s Kansai - Frontrend in Osaka

@ahomu
CyberAgent, Inc.

佐藤 歩 (๑´ᴗ`๑)

http://aho.mu



Frontend Engineer

HTML/CSS/JavaScript/Browser

2008/04

待ち時間の80%は
フロントエンドである

2010/04

AjaxやHTML5など
技術的トレンドに対応

2013/09

?

2013年現在の
パフォーマンスとは？

フロントのパフォーマンス3要因

Network Compute Render

話の流れ

1. **概論** - #perfmatters
2. **通信** - Network
3. **描画** - Render
4. **計算** - Compute
5. **結論** - Conclusion

今回の内容は
ほぼWebKitベース

Chrome, **Opera**, Android, Safari

開発者のための WebKit (“WebKit for Developers” 日本語訳)
http://myakura.github.io/n/webkit4devs.html


#PERFMATTERS
Performance Matters

昨今のパフォーマンスと
その重要性

zoompf.com/blog/2013/08/web-performance-basics-for-the-marketing-team

コンバージョン率が向上する

Mozillaでは、ページロードの速度が

2.2秒 高速化したことで

ダウンロード数が**15.4**% 向上した

blog.mozilla.org/metrics/2010/04/05/firefox-page-load-speed-%E2%80%93-part-ii/

Googleに嫌われないために

ページパフォーマンスが**低い**サイトは

検索ランキングに**悪い影響**がある

googlewebmastercentral.blogspot.jp/2010/04/using-site-speed-in-web-search-ranking.html

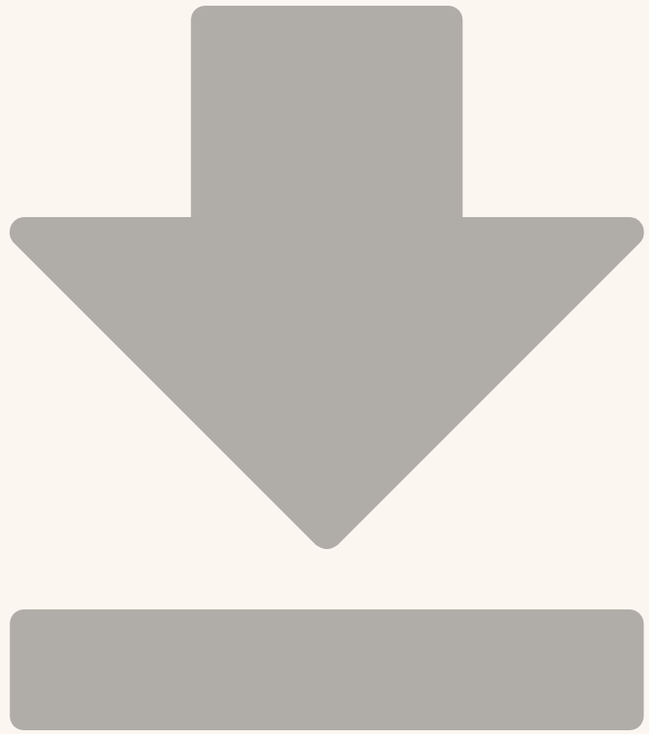
利益に影響を与える

GoogleとMicrosoftは

ページロードが **2秒** 遅くなるたび

サービスにとって **4.3%** の減益に

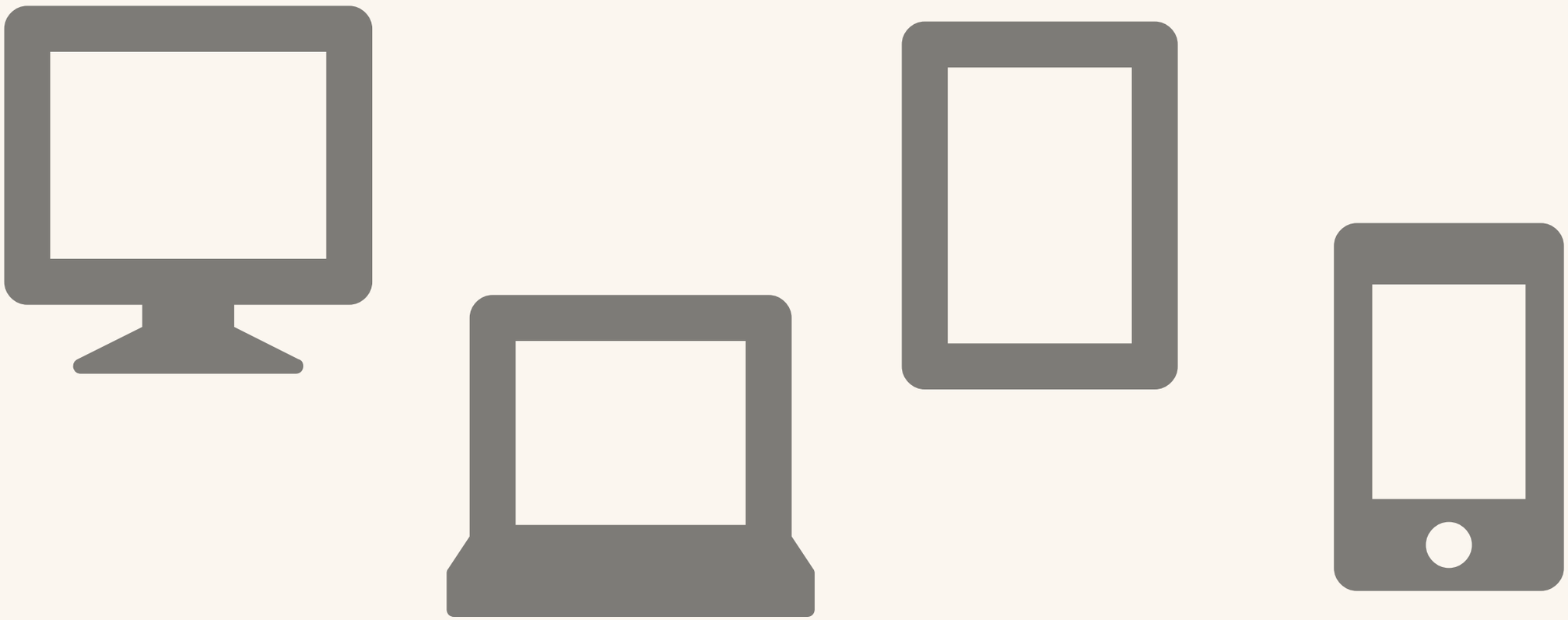
つながる研究を示した



http://radar.oreilly.com/2009/06/bing-and-google-agree-slow-pag.html

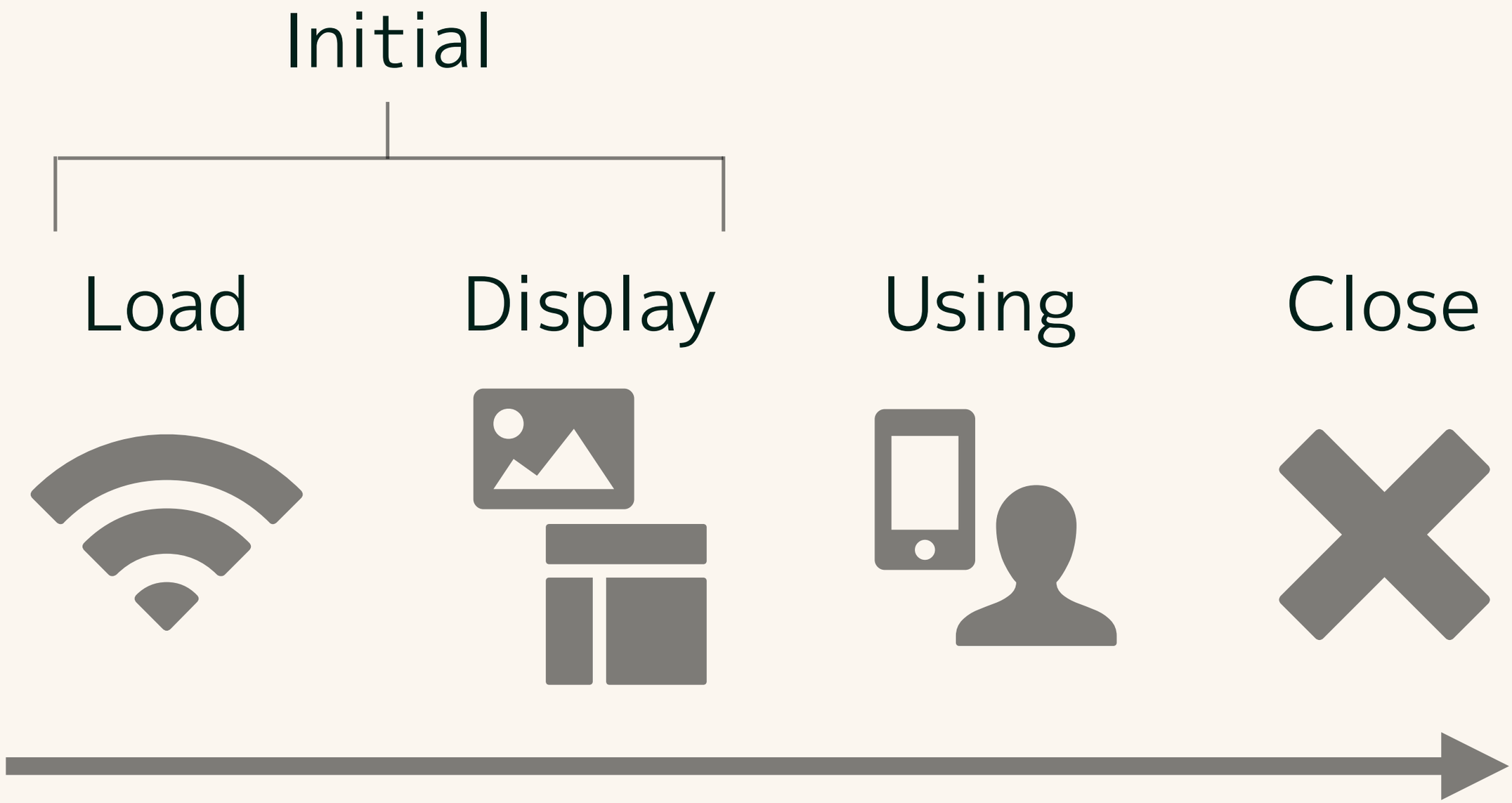
Webの変化によって
重要性が更に増している

実行環境の多様化



要件の複雑化

Page Performance Lifecycle

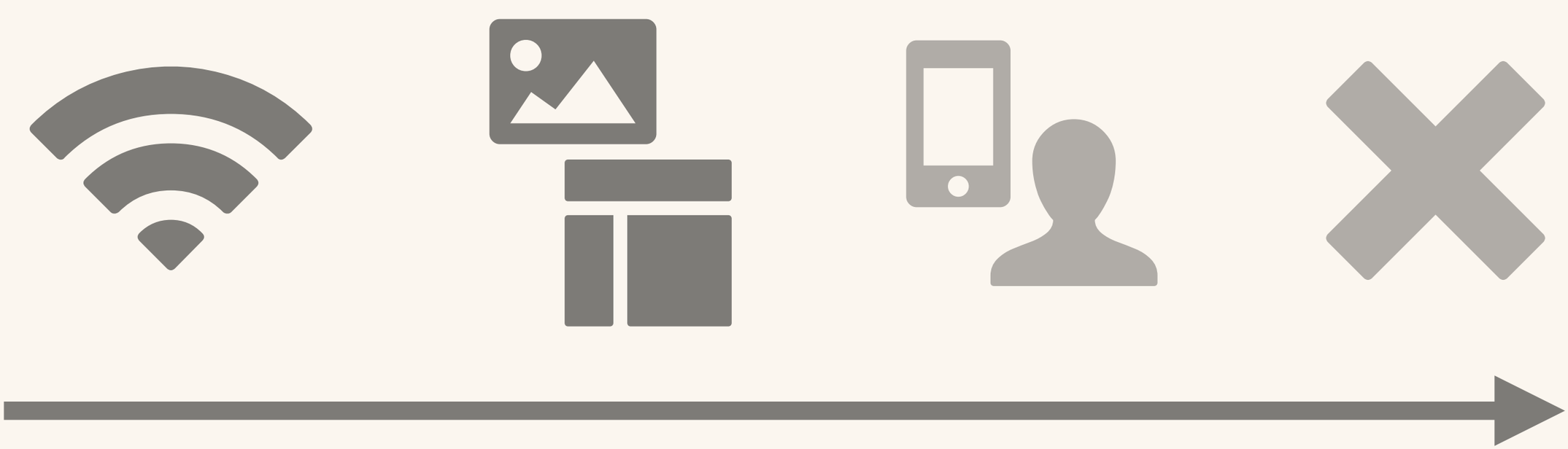


Page Perform ifecycle

画面遷移
のたび繰り返す

Initial

Load Display Using Close



Page Performance Lifecycle

Initial

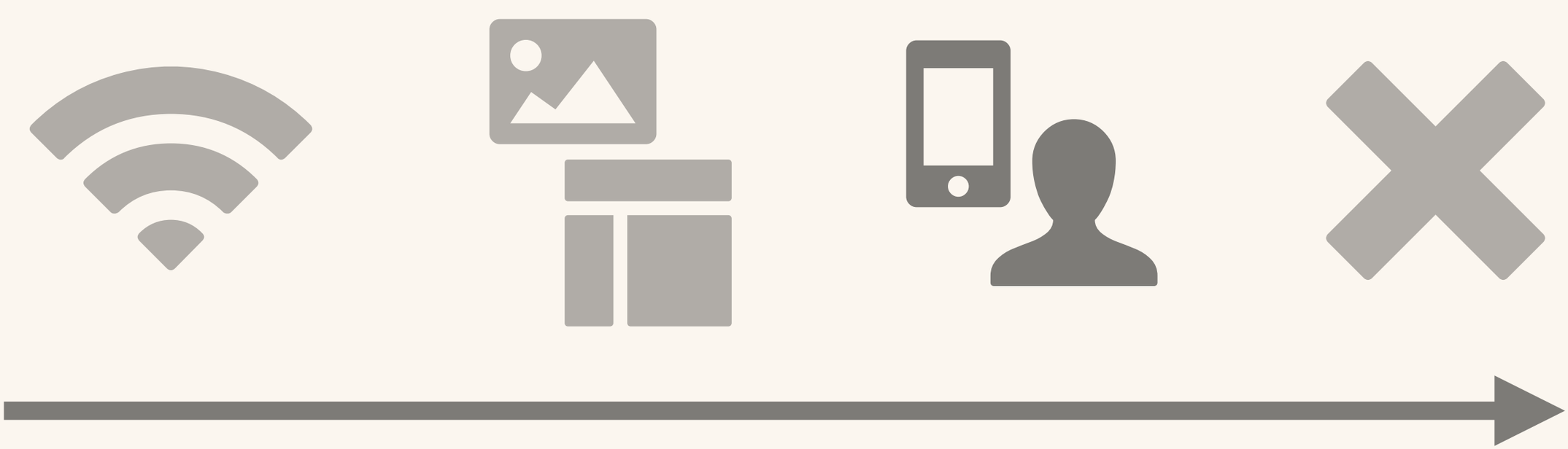
Load

Display

Using

Close

❤使用感の
快適さ・滑らか



Network

ネットワーク・通信

DNS Lookup

TCP Connection

Send HTTP Request

Receive HTTP Reponse

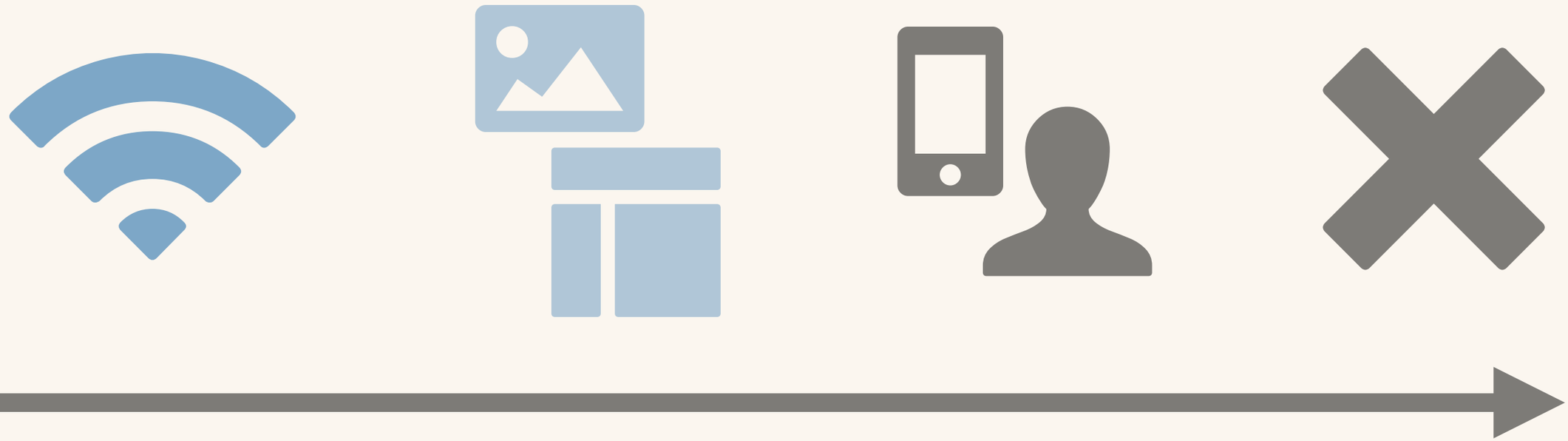
Initial

Load

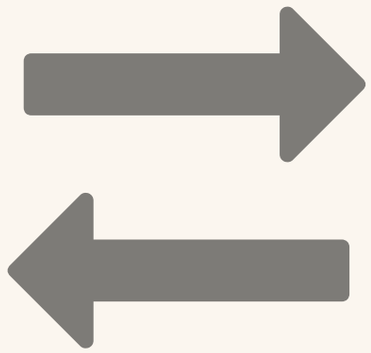
Display

Using

Close



DNS Lookup



TCP Connection

HTTP (Request ~ Response) * n

40KBのリソースを受け取る HTTPリクエストのNetwork

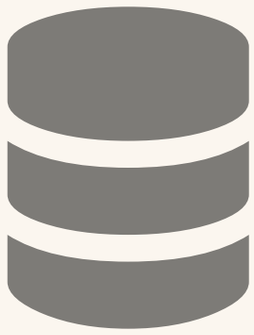
- クライアントがサーバーの
  やり取りイチ往復分にかかる時間を
  Round Trip Timeと呼びます

- 最終的に何往復するでしょうか？

DNS Lookup

TCP Connection

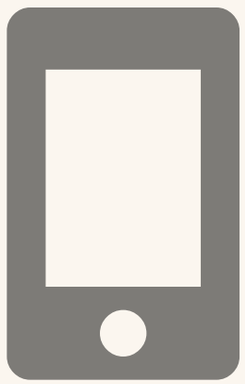
HTTP (Request ~ Response) * n



DNS

「www.example.comって名前の
Serverの住所はどこ？」
(Client)

Client



フォワードルックアップ

DNS解決

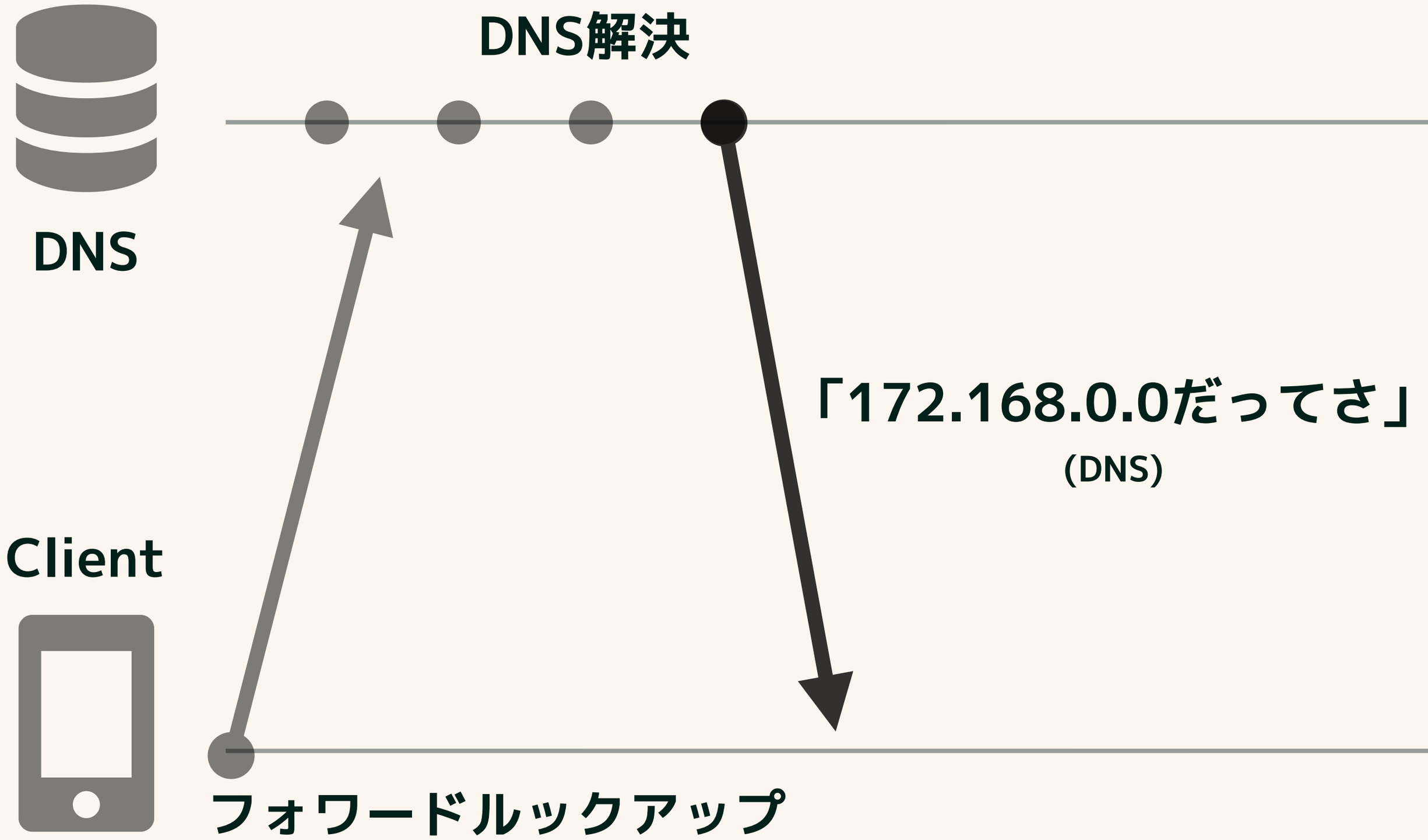
DNS

「あー、おれ知らないけど
知ってそうな他のDNSに聞いてみるわ」

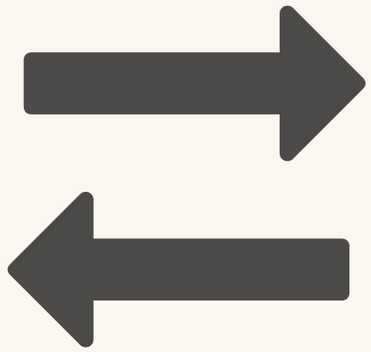
(DNS)

Client

フォワードルックアップ

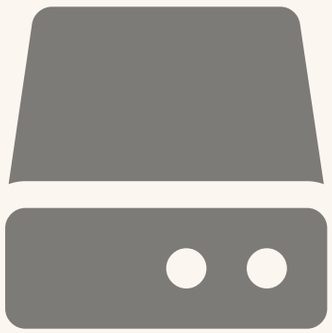

DNS解決
DNS
「172.168.0.0だってさ」
(DNS)
Client
フォワードルックアップ

DNS Lookup



TCP Connection

HTTP (Request ~ Response) * n

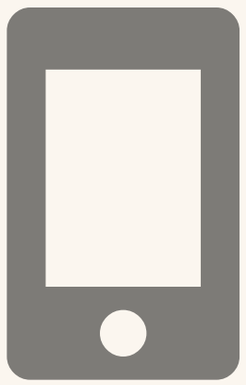


Server

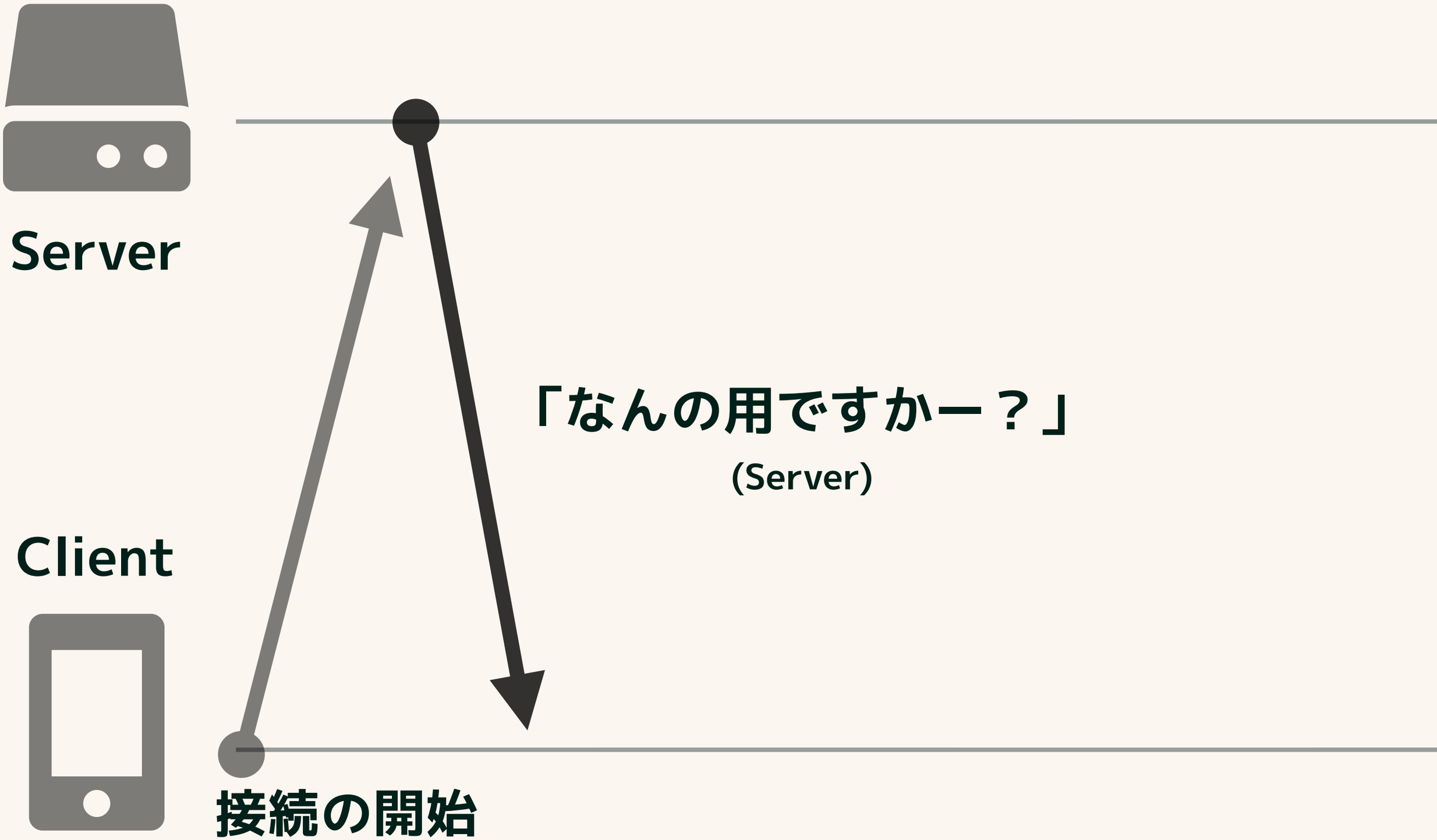
「ごめんくださーい」

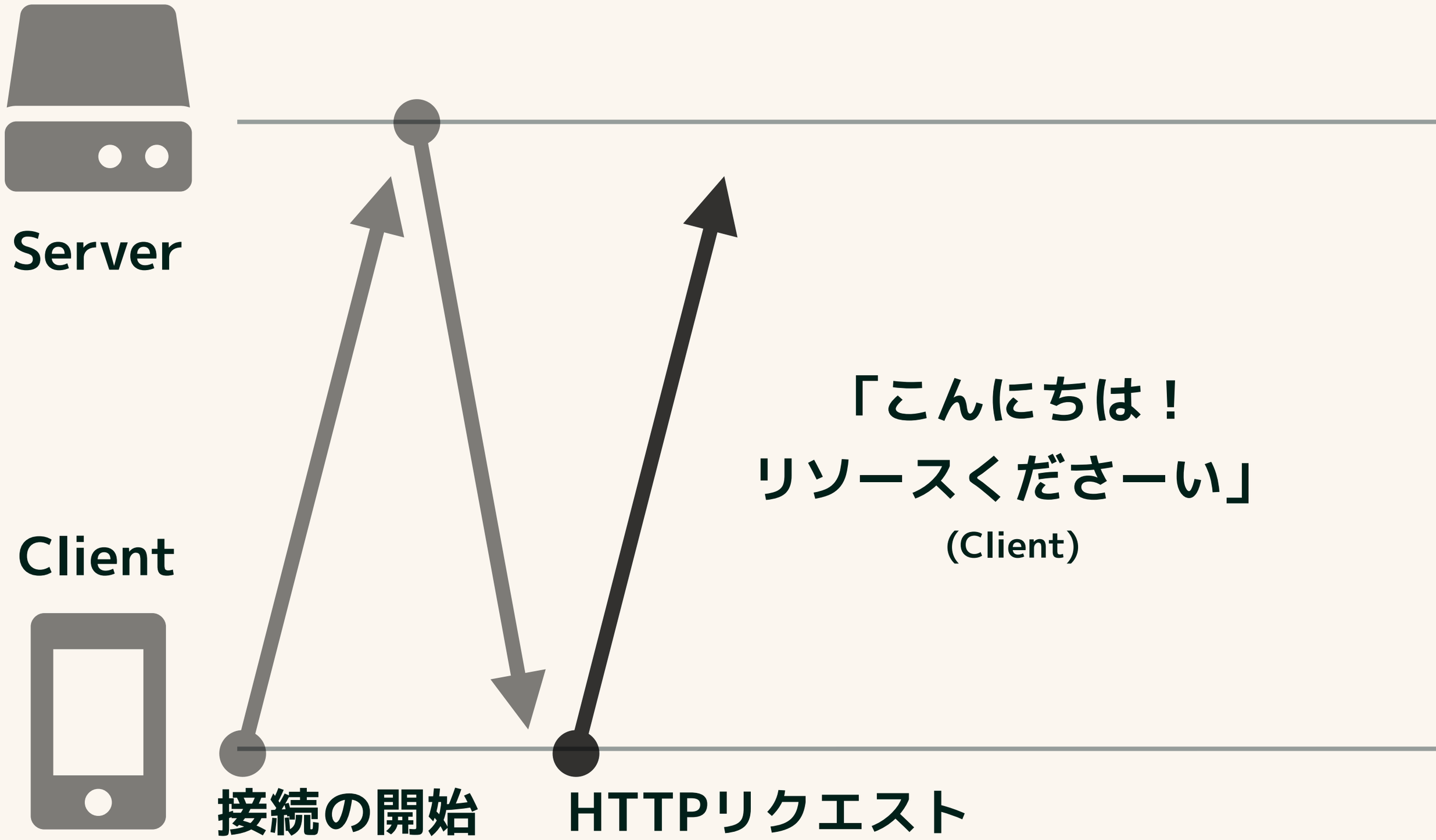
(Client)

Client

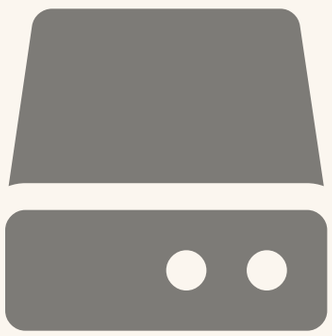


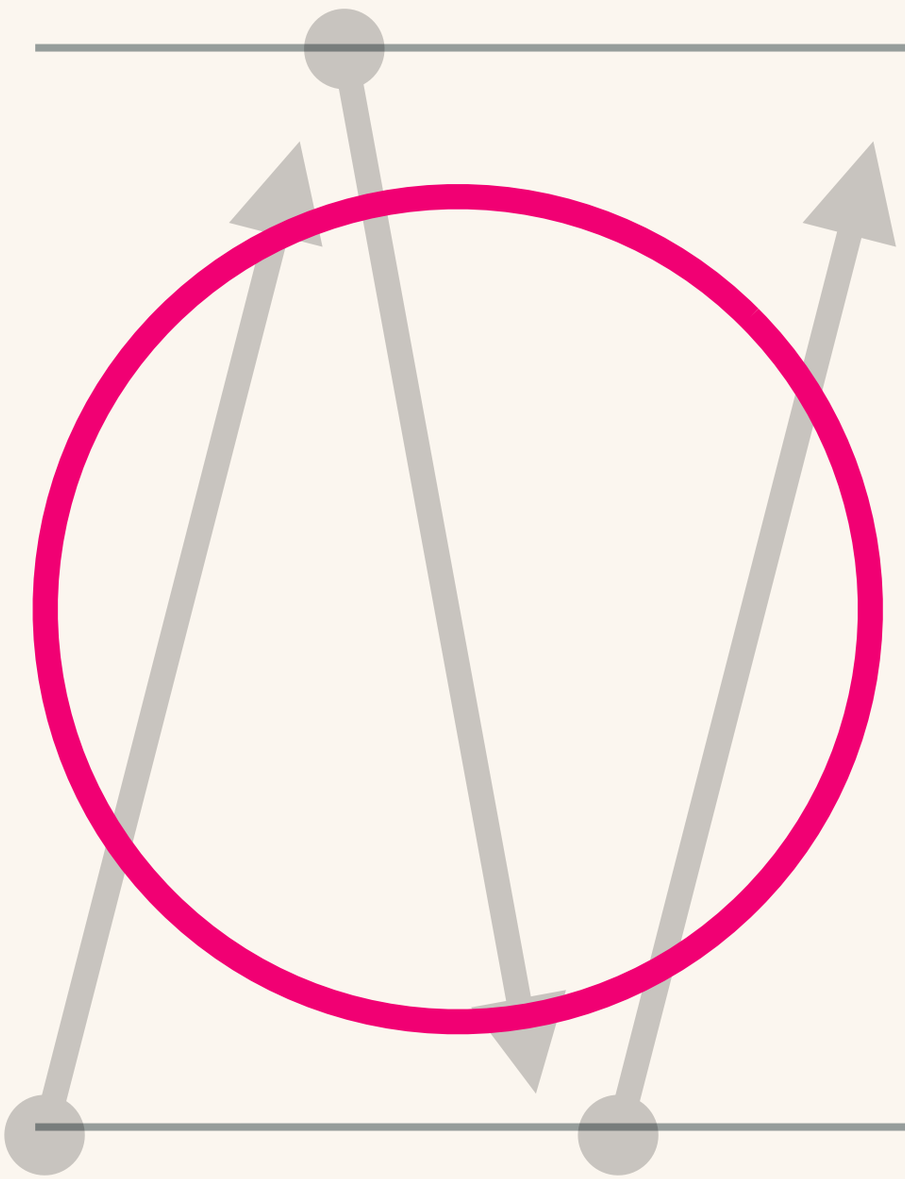
接続の開始


Server
「なんの用ですかー？」
(Server)
Client
接続の開始


Server
Client
「こんにちは！
リソースくださーい」
(Client)
接続の開始
HTTPリクエスト

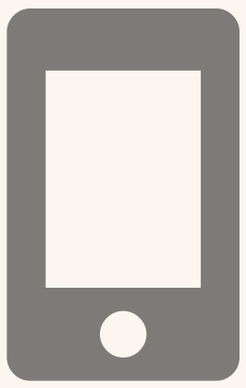
Server





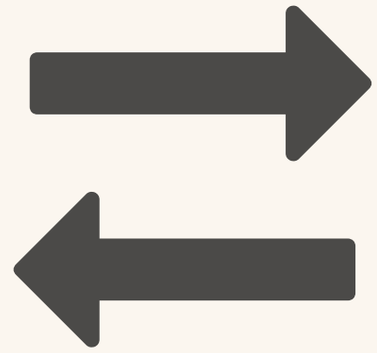
TCP 3-way handshake

Client



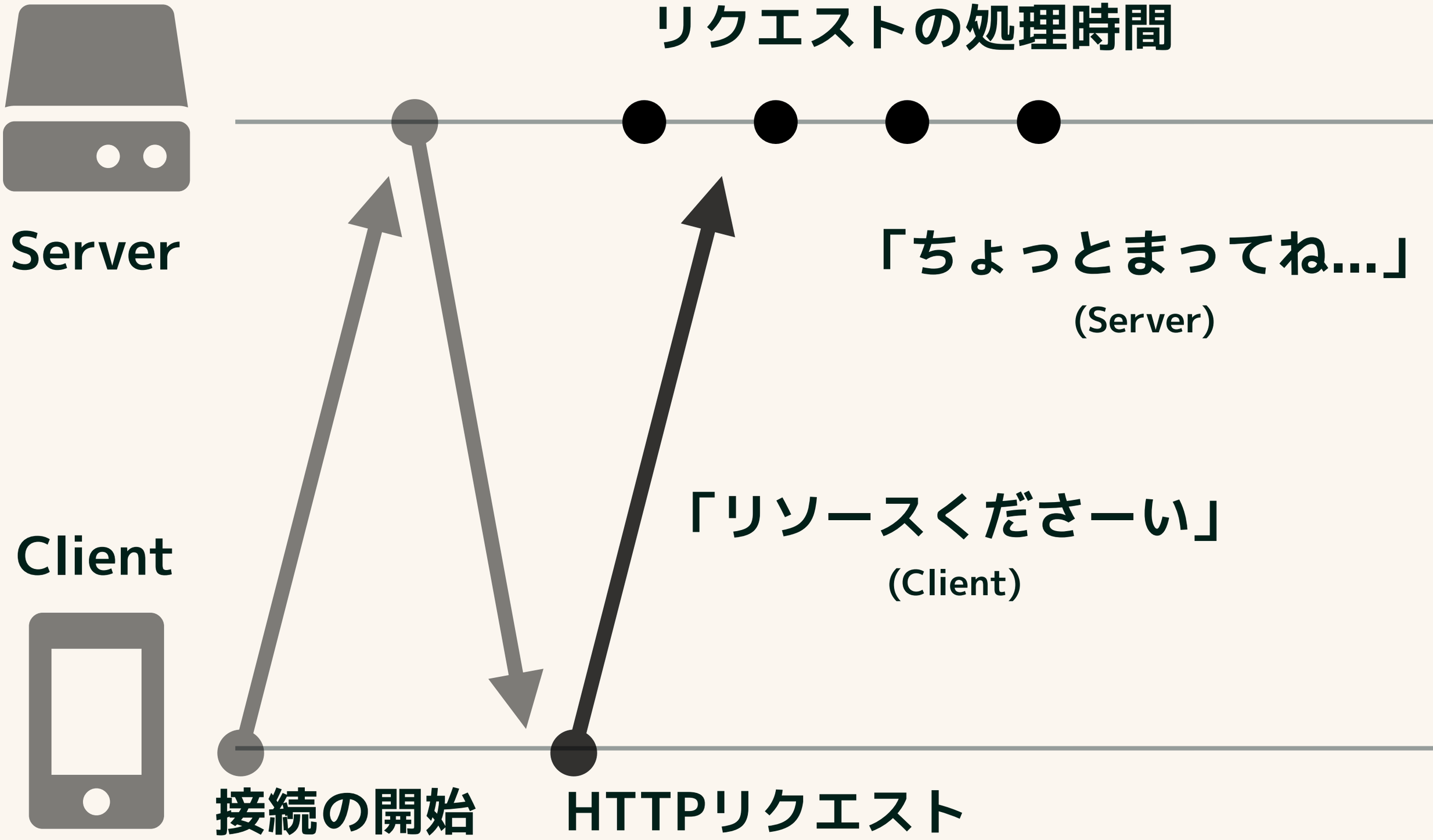
接続の開始　　HTTPリクエスト

DNS Lookup



TCP Connection

HTTP (Request ~ Response) * n


リクエストの処理時間
Server
「ちょっとまってね...」
(Server)
「リソースくださーい」
(Client)
Client
接続の開始
HTTPリクエスト

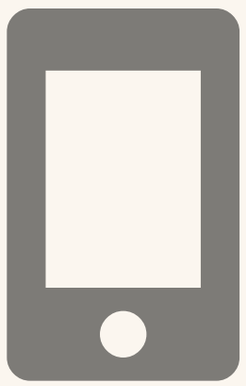
Server

HTTPレスポンス(40KB)

「とりあえず14.6KB送るね」

(Server)

Client

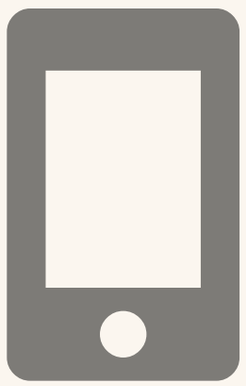


Server

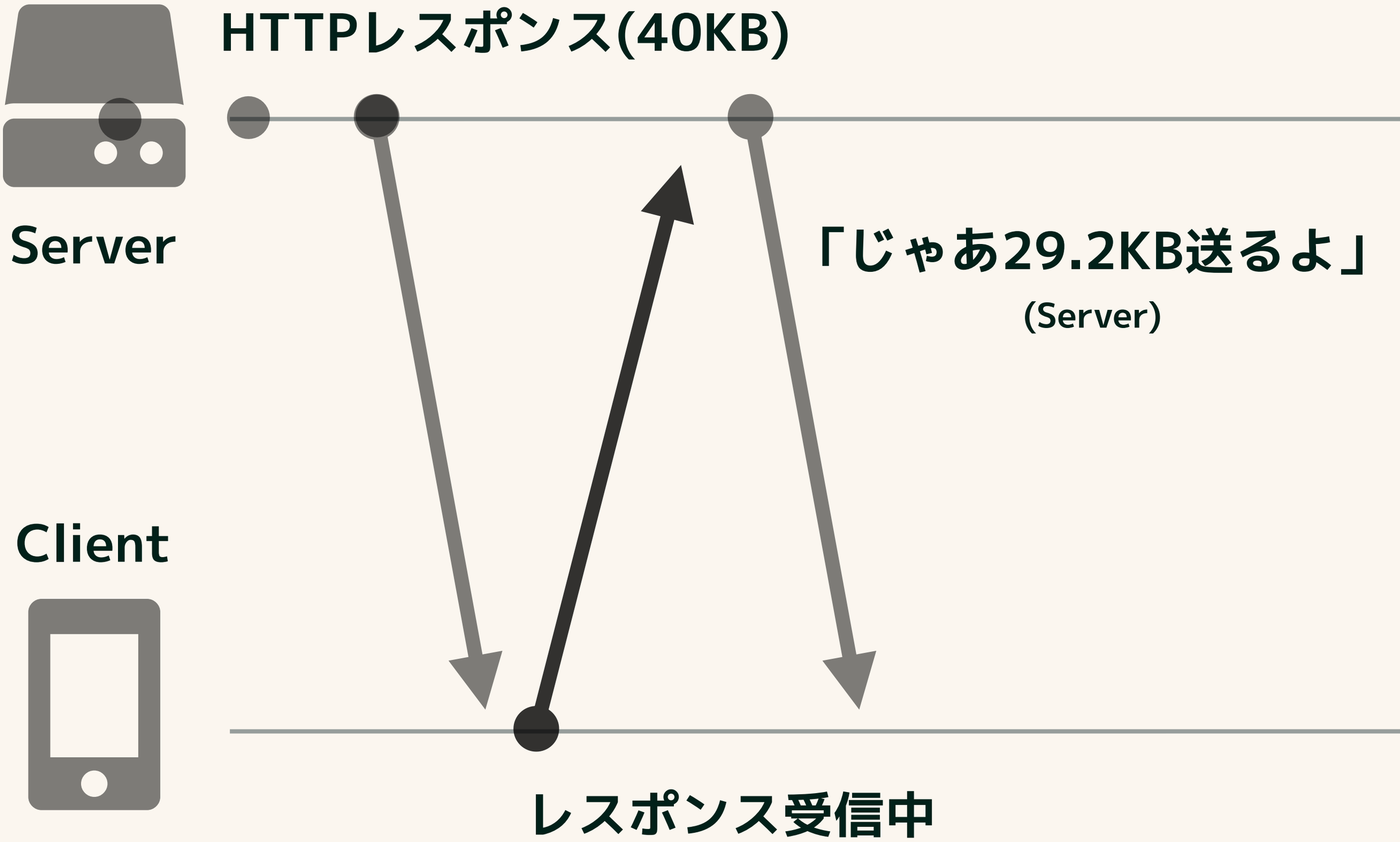
HTTPレスポンス(40KB)

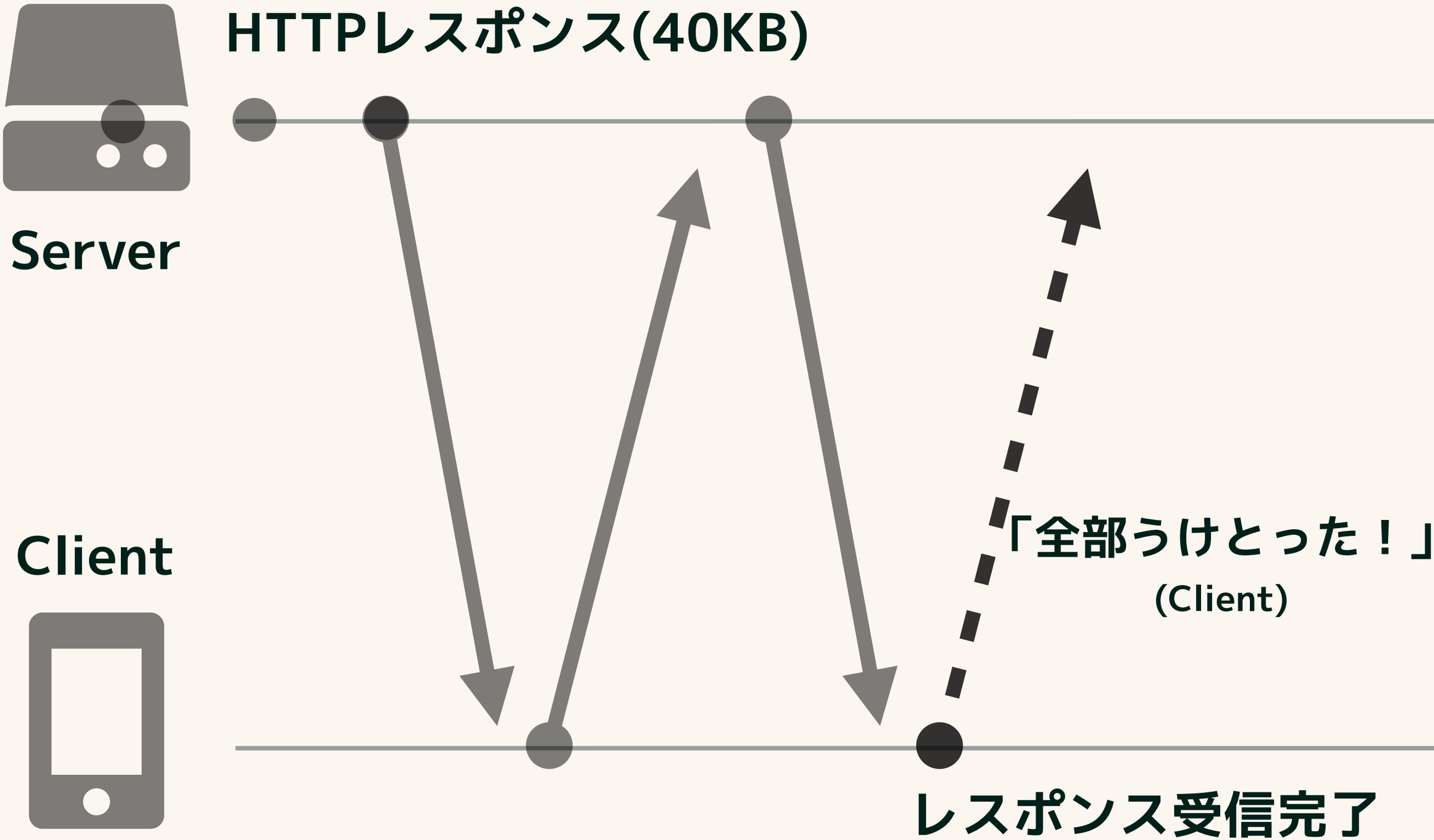
「ありがとう！
もっと沢山送っていいよ！」
(Client)

Client



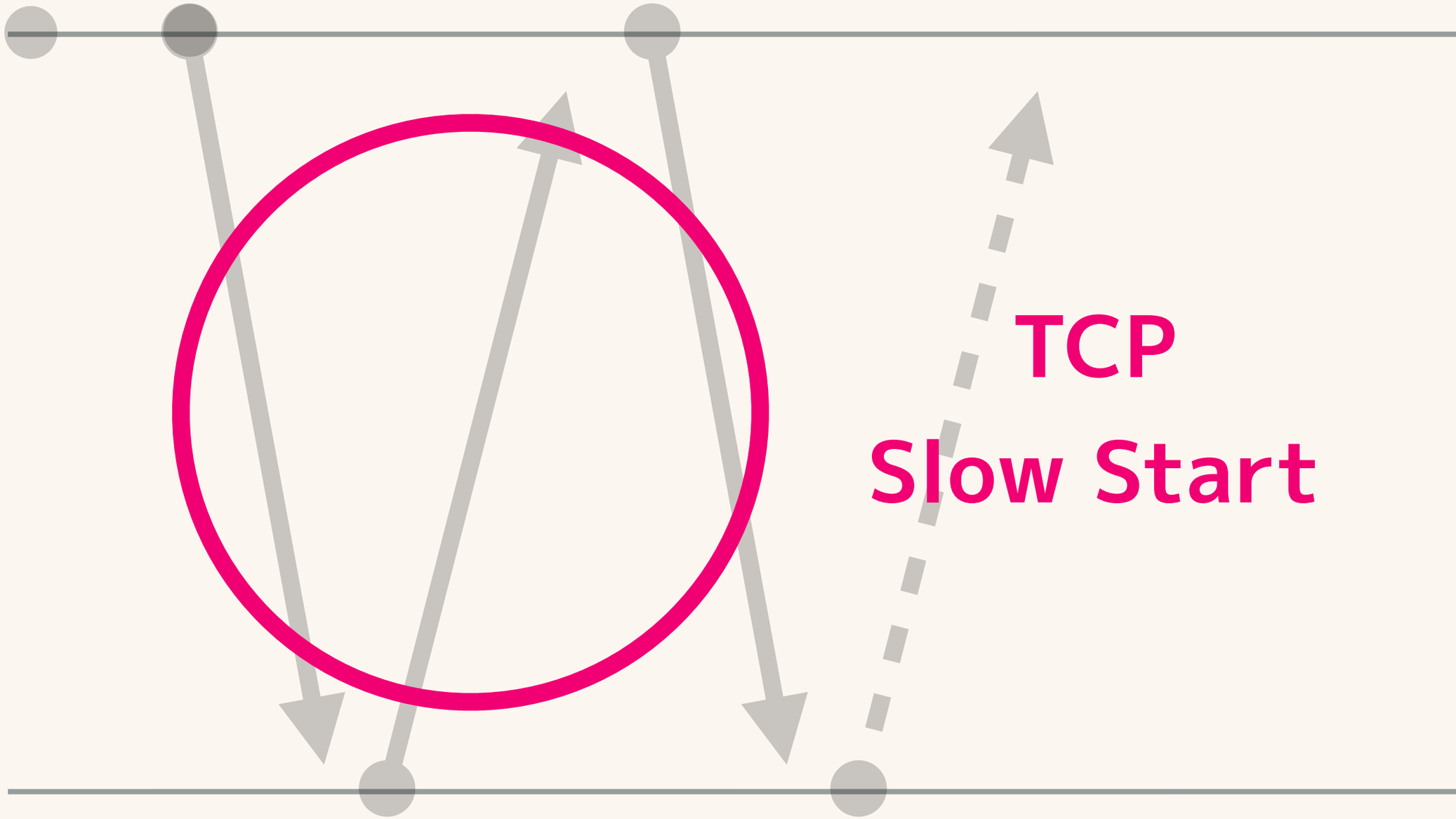
レスポンス受信中


HTTPレスポンス(40KB)
Server
「じゃあ29.2KB送るよ」
(Server)
Client
レスポンス受信中


HTTPレスポンス(40KB)
Server
Client
「全部うけとった！」
(Client)
レスポンス受信完了

HTTPレスポンス(40KB)

Server



TCP
Slow Start

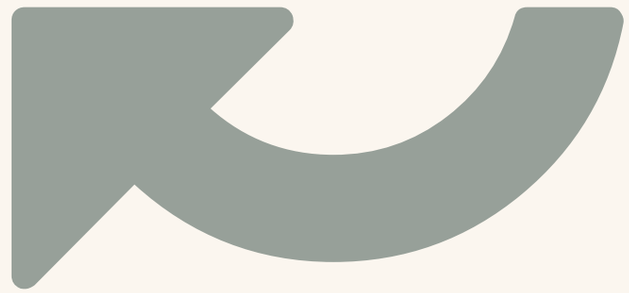
Client

レスポンス受信完了

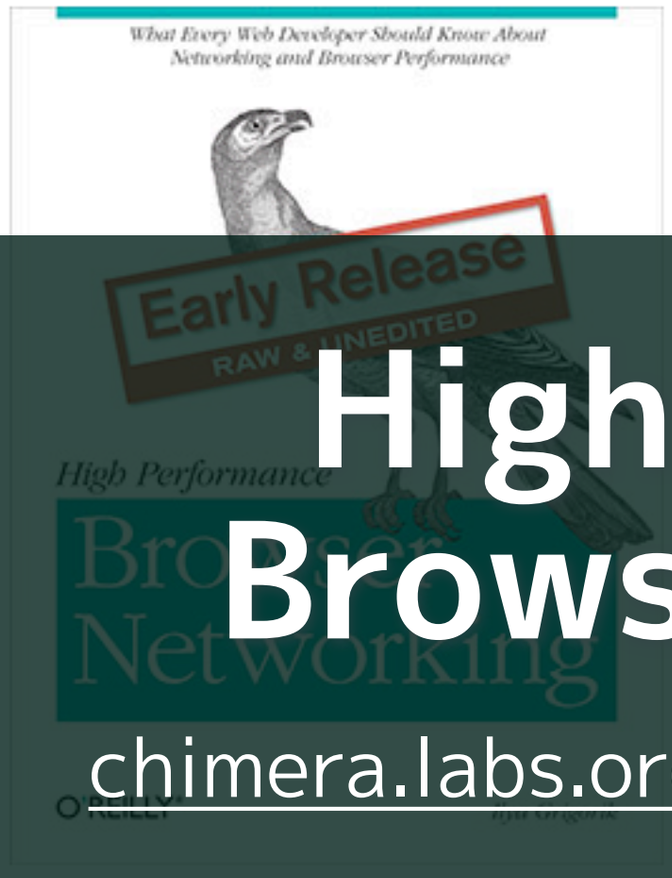
オーバーヘッドが出やすい

- 計4回のRound Tripがありました

- 1リクエストと言えども
  数回のRound Tripを繰り返している



- 通信が不安定な環境だとRound Tripの
  回数に比例してオーバーヘッドが特にかさむ



High Performance Browser Networking

by Ilya Grigorik

How prepared are you when it comes to building high-performance web applications? This book provides what every developer must know about networking in the browser to build faster, more responsive applications—from fundamental protocol limitations that affect performance to architecture and constraints of mobile networks and devices. By understanding how the underlying protocols work, you'll be able to make better design decisions and deliver the best experience for your users.

Brought to you by Velocity Conference

High-Performance Browser Networking

chimera.labs.oreilly.com/books/1230000000545/

About the Author

Ilya Grigorik is a web performance engineer and developer advocate on the Make The Web Fast team at Google, where he spends his days and nights on making the web fast and driving adoption of performance best practices. You can find Ilya online on his blog at igvita.com and on Twitter and Google+.

Buy the Ebook on orelly.com

Author Ilya Grigorik—a developer advocate and web performance engineer at Google—starts with the building blocks of TCP, UDP and TLS, and then dives into newer technologies such as HTTP 2.0, WebSockets, and WebRTC. This book explains how these technologies work under the hood, their benefits, and provides hands-on optimization tips and best practices for

ちょっと難しいですよね...

はじめのアクションは
パフォーマンスの指標化

標準レポート
リアルタイム
ユーザー
トラフィック
コンテンツ
サマリー
サイト コンテンツ
サイトの速度
サマリー
ページ速度
ユーザー設定速度
速度の提案
サイト内検索
イベント
AdSense
ウェブテスト
ページ解析
コンバージョン

例えば... コンテンツ > サイトの速度

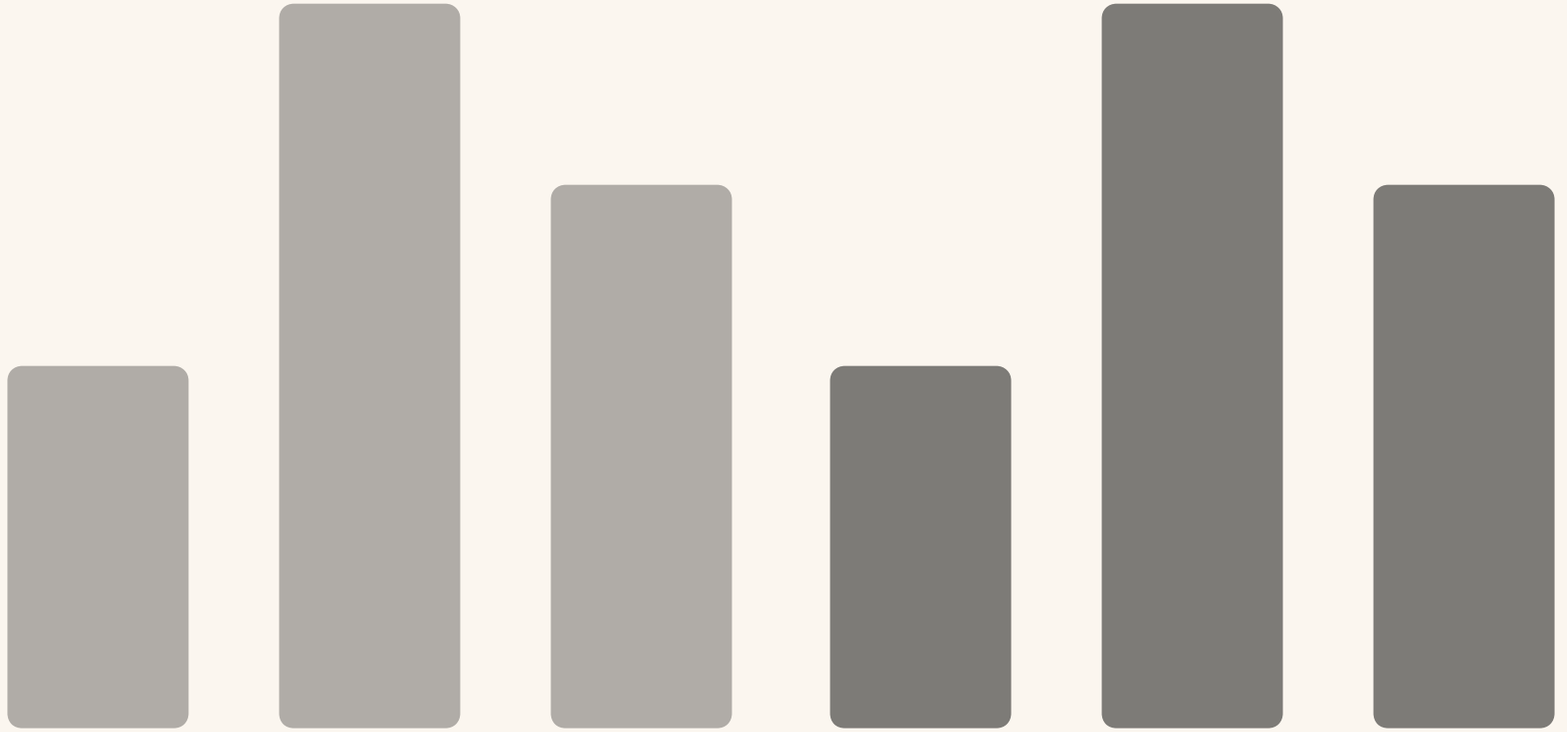
www.google.com/analytics/

	ページ	ページビュー	平均表示時間	PageSpeed	PageSpeed スコア
1.	/develop/javascript/e171-ajax-file-upload.html	847	1.92	合計 9 個	91
2.	/	829	0.79	合計 9 個	78
3.	/co[illegible]/e[illegible]-[illegible].html	712	11.99	合計 9 個	93
4.	/develop/php/e188-php_file_get_contents_tips.html	658	4.76	合計 8 個	93
5.	/food/e479-beef_tendon_curry.html	634	3.13	合計 8 個	96
6.	/develop/git/e513-git_branch_model.html	624	1.99	合計 9 個	90
7.	/develop/javascript/e189-jquery-plugin-placeholder.html	575	9.28	合計 10 個	92
8.	/develop/performance/e554-paint_gpu_acceleration_problems.html	572	1.28	合計 9 個	89
9.	/develop/php/e167-php-apc-install.html	509	1.42	合計 9 個	87
10.	/develop/ruby/e555-rails_on_heroku_app.html	455	24.49	合計 7 個	94

1 - 10/515

このレポートは 2013/06/22 23:46:32 に作成されました - レポートを更新

１指標として目標値を定め
継続して追跡する



標準レポート
リアルタイム
ユーザー
トラフィック
コンテンツ
サマリー
サイト コンテンツ
サイトの速度
サマリー
ページ速度
ユーザー設定速度
速度の提案
サイト内検索
イベント
AdSense
ウェブテスト
ページ解析
コンバージョン

コンテンツ > サイトの速度 > 速度の提案

www.google.com/analytics/

	ページ	ページビュー数	平均表示時間	PageSpeed	PageSpeed スコア
1.	/develop/javascript/e171-ajax-file-upload.html	847	1.92	合計 9 個	91
2.	/	829	0.79	合計 9 個	78
3.	/co[illegible]/e[illegible]-[illegible]_[illegible].html	712	11.99	合計 9 個	93
4.	/develop/php/e188-php_file_get_contents_tips.html	658	4.76	合計 8 個	93
5.	/food/e479-beef_tendon_curry.html	634	3.13	合計 8 個	96
6.	/develop/git/e513-git_branch_model.html	624	1.99	合計 9 個	90
7.	/develop/javascript/e189-jquery-plugin-placeholder.html	575	9.28	合計 10 個	92
8.	/develop/performance/e554-paint_gpu_acceleration_problems.html	572	1.28	合計 9 個	89
9.	/develop/php/e167-php-apc-install.html	509	1.42	合計 9 個	87
10.	/develop/ruby/e555-rails_on_heroku_app.html	455	24.49	合計 7 個	94

1 - 10/515

このレポートは 2013/06/22 23:46:32 に作成されました - レポートを更新

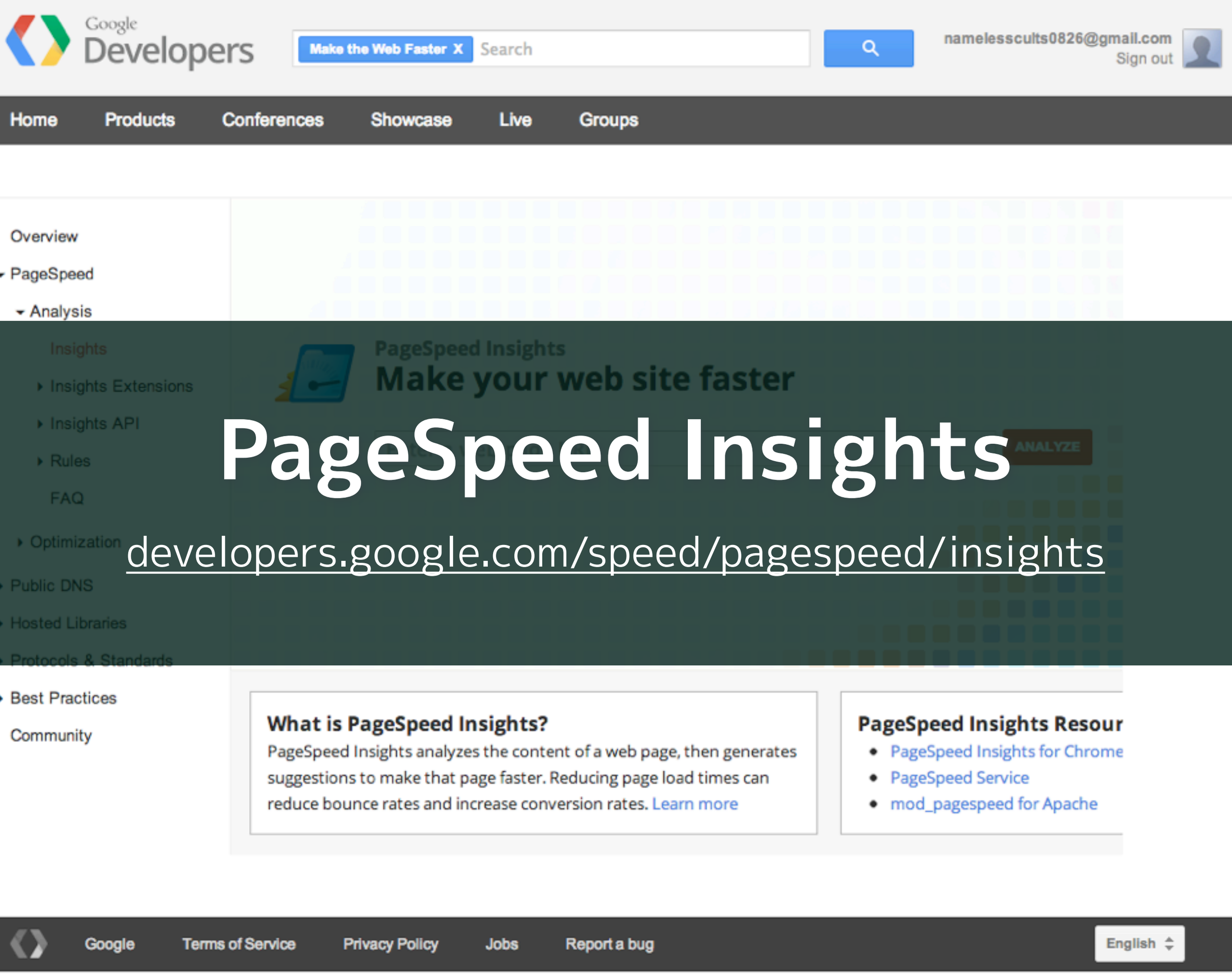
速度の提案（一部）

- ブラウザキャッシュを活用する
- リダイレクトの回数を減らす
- リソースを圧縮する
- CSSスプライトを利用する
- CSSとJSの読み込み順を最適化する

etc...

PageSpeed Insights

developers.google.com/speed/pagespeed/insights



Mobile 70 / 100

Desktop 66 / 100

Suggestions Summary

- Leverage browser caching

  Setting an expiry date or a maximum age in the HTTP headers for static resources instructs the browser to load previously downloaded resources from local disk rather than over the network.

- Reduce server response time

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

  Your page has 7 blocking CSS resources. This causes a delay in rendering your page.

- Optimize images

  Properly formatting and compressing images can save many bytes of data.

- Enable compression

  Compressing resources with gzip or deflate can reduce the number of bytes sent over the network.

- Minify CSS

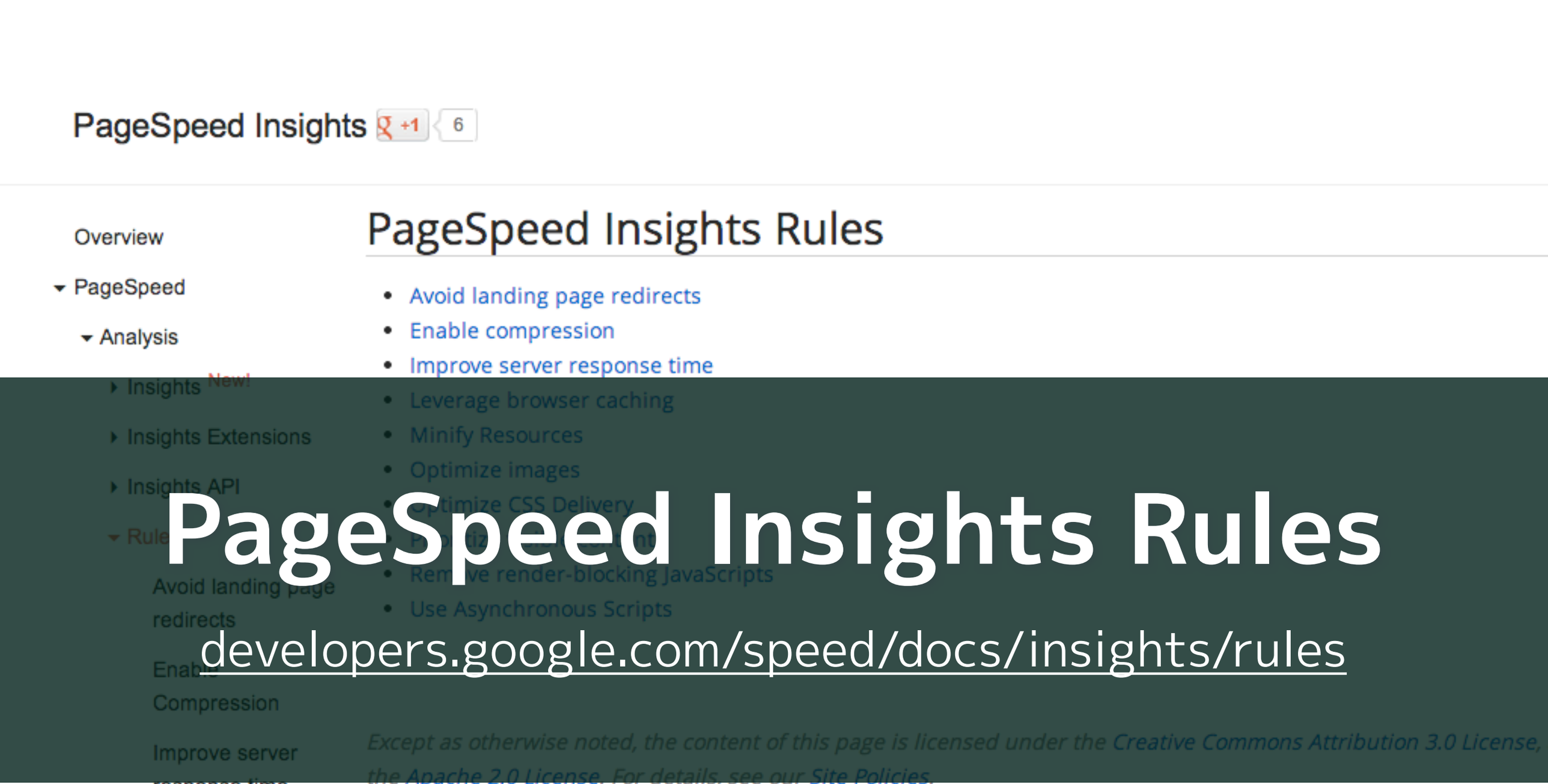
  Compacting CSS code can save many bytes of data and speed up download and parse times.

- Minify HTML

  Compacting HTML code, including any inline JavaScript and CSS contained in it, can save many bytes of data and speed up download and parse times.

3 Passed Rules

**The results are cached for 30s. If you have made changes to your page, please wait for 30s before re-running the test.*



PageSpeed Insights Rules

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

Webパフォーマンス ベストプラクティス

WebページをPage Speedで調べるとルールに準拠していないものが提示される。このルールというのは、一般的にあなたが開発段階において取り入れるべきフロントエンドのベストプラクティスだ。あなたがPage Speedを使用しようとしまいと、私たちはこの各ルールについてのドキュメントを提供する（たぶんちょうど新しいサイトを開発中でテストする準備が整ってないだろう）。もちろん、これらのページはいつでも参照することができる。私たちはあなたの開発プロセスに取り入れてもらうために、このベストプラクティスを実装するための明確なティップスと提案を提供する。

パフォーマンス ベストプラクティスについて

Web パフォーマンスベストプラクティス

Page Speedはクライアント側からの観点でパフォーマンスを評価し、一般的にページの読み込み時間を計測する。これはユーザーがページをリクエストした瞬間からブラウザによって完全にレンダリングされるまでの経過時間のことを指している。

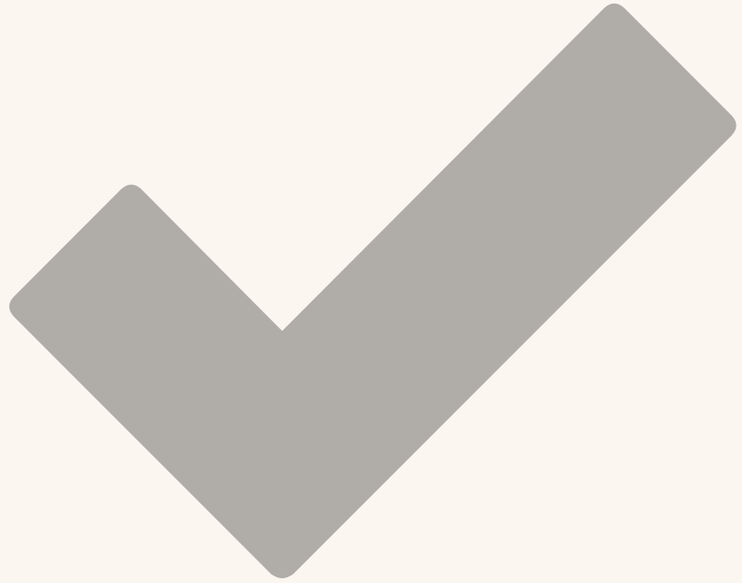
github.com/t32k/speed/blob/master/docs/best-practicesrules_intro.md

決、TCPコネクションの確立、HTTPリクエストの伝達、リソースのダウンロード、キャッシュからのリソース読み込み、スクリプトのパースと実行、ページ上のオブジェクトのレンダリングに至るまでだ。基本的にPage SpeedをWebページで使用することで、それらのステップをちゃんとしているかどうか評価し、読み込みが完了するまでの時間を短縮させることができる。

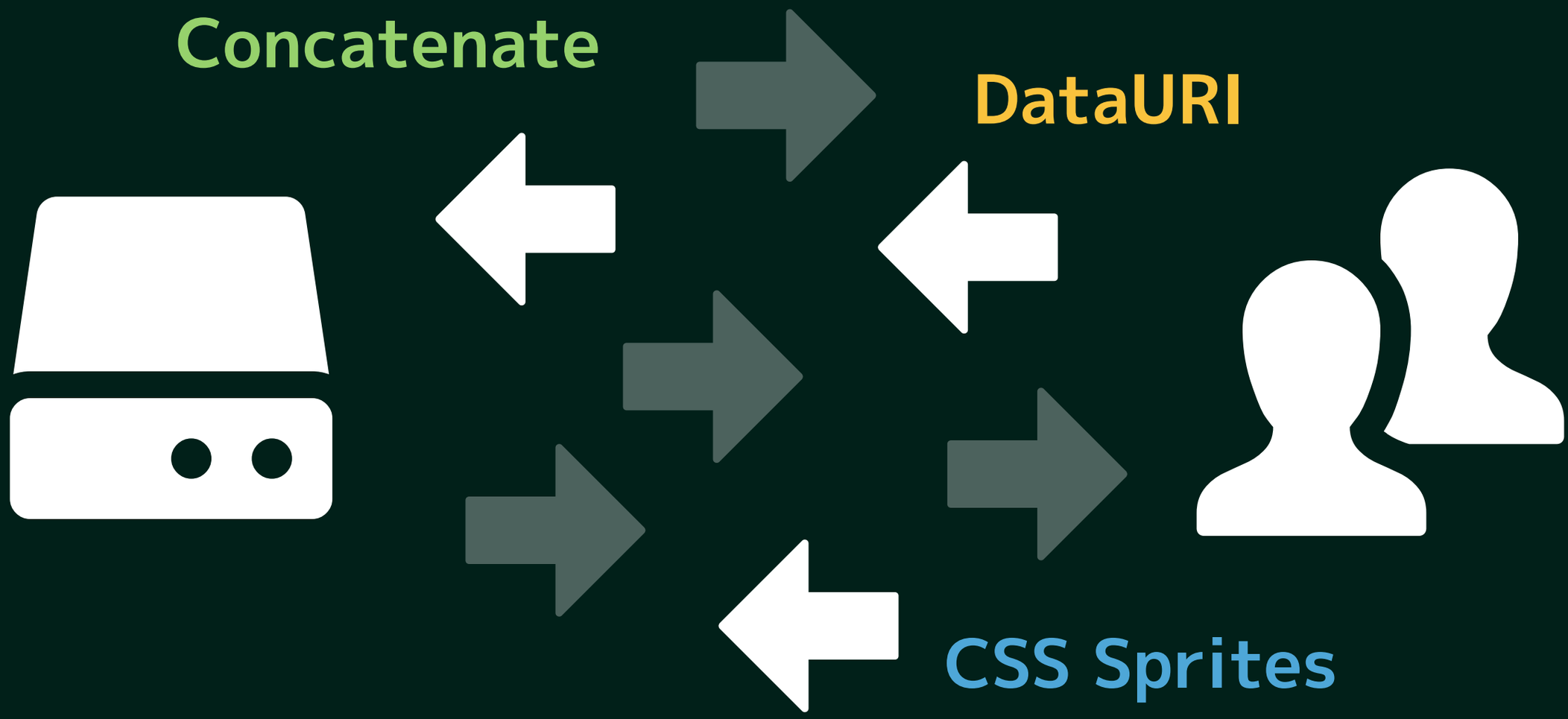
ベストプラクティスは異なる面からページ読み込み最適化カバーする6つのカテゴリに分類されている。

- キャッシュの最適化 - アプリケーションのデータとロジックをネットワークから完全に分離させる
- ラウンドトリップ回数を最小にする - リクエスト-レスポンスの一連のサイクルの回数を減らす
- リクエストオーバーヘッドの縮小化 - アップロードサイズを減らす
- 読み込みサイズの減量 - レスポンス、ダウンロード、キャッシュ化されたページのサイズを減らす
- ブラウザレンダリングの最適化 - ブラウザのページレイアウト改善
- モバイルのための最適化 - モバイル端末・ネットワーク特有のチューニング

つぎのアクションは
Network節約の基本対策



リクエストの数を減らす



Reduce number of HTTP requests

リソースのサイズを減らす

まずはコスパの良い
対策を解決していく

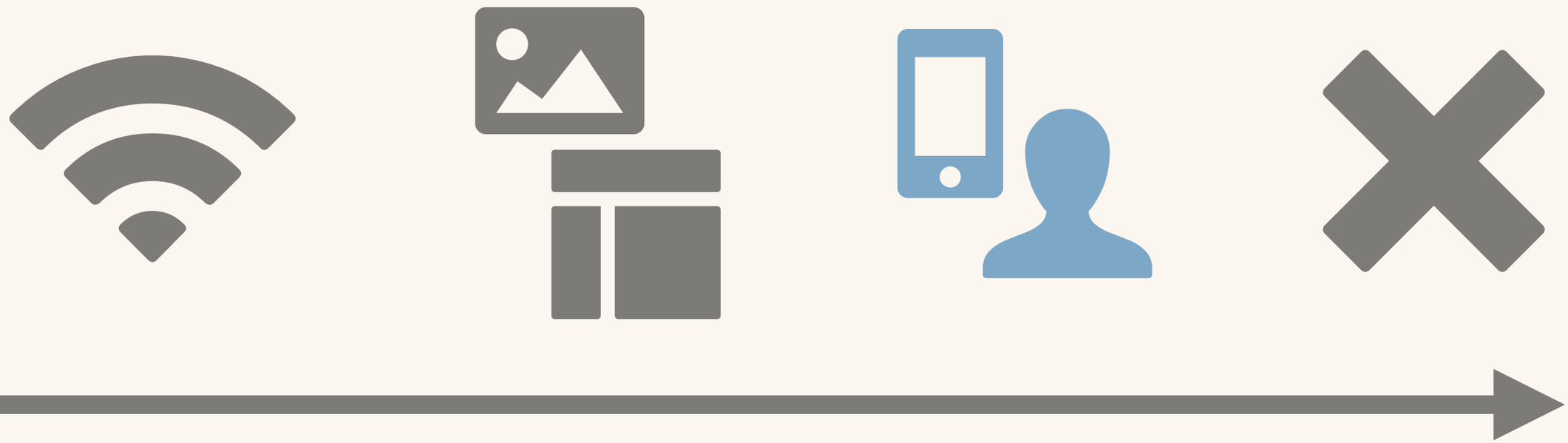
Initial

Load

Display

Using

Close



16.666...

= 1000ms/60FPS

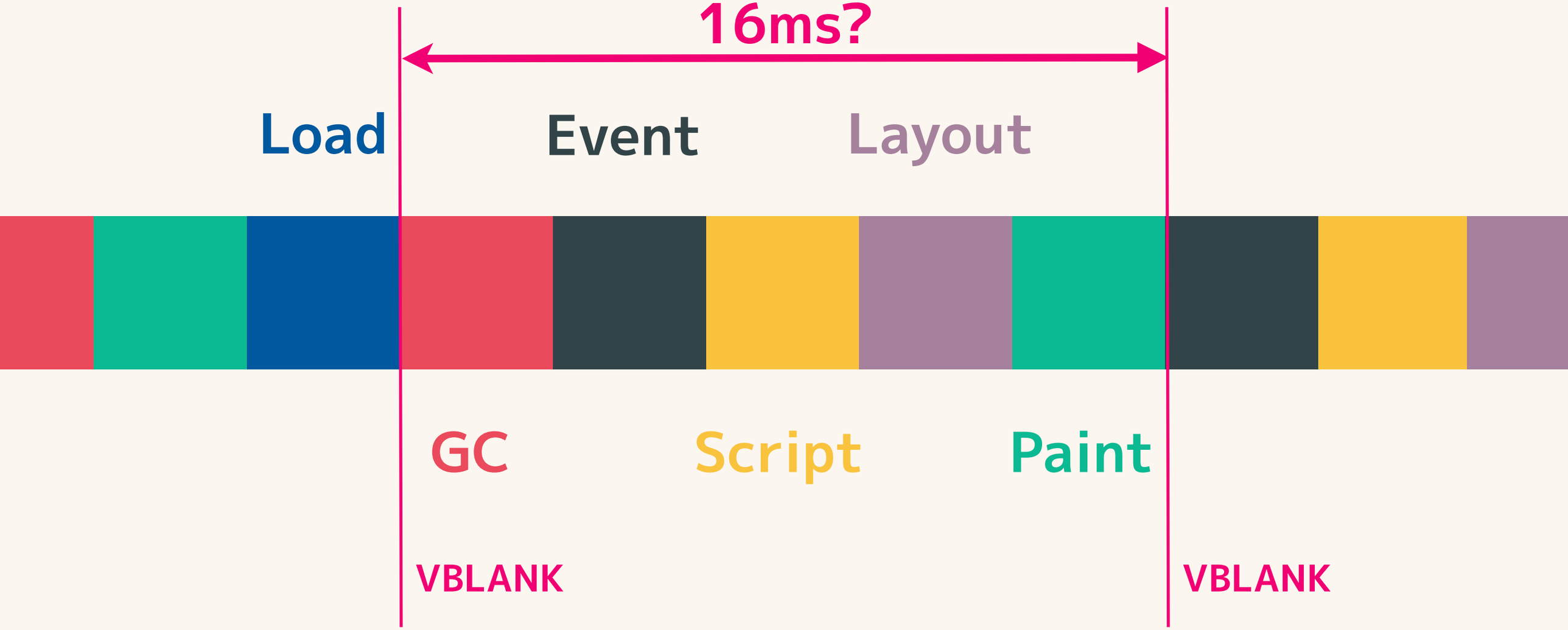
FPSとリフレッシュレート

FPS (frame per second)

モバイル含めて多くの画面は
1秒に60回リフレッシュする(60Hz)

60FPSを保つためには
1フレームの処理を16msで済ませる

ブラウザの1フレームには
様々な処理が詰まっている



フレーム処理の状態をTimelineで確認する

Timeline > Frames

Shadow Radius Dotted

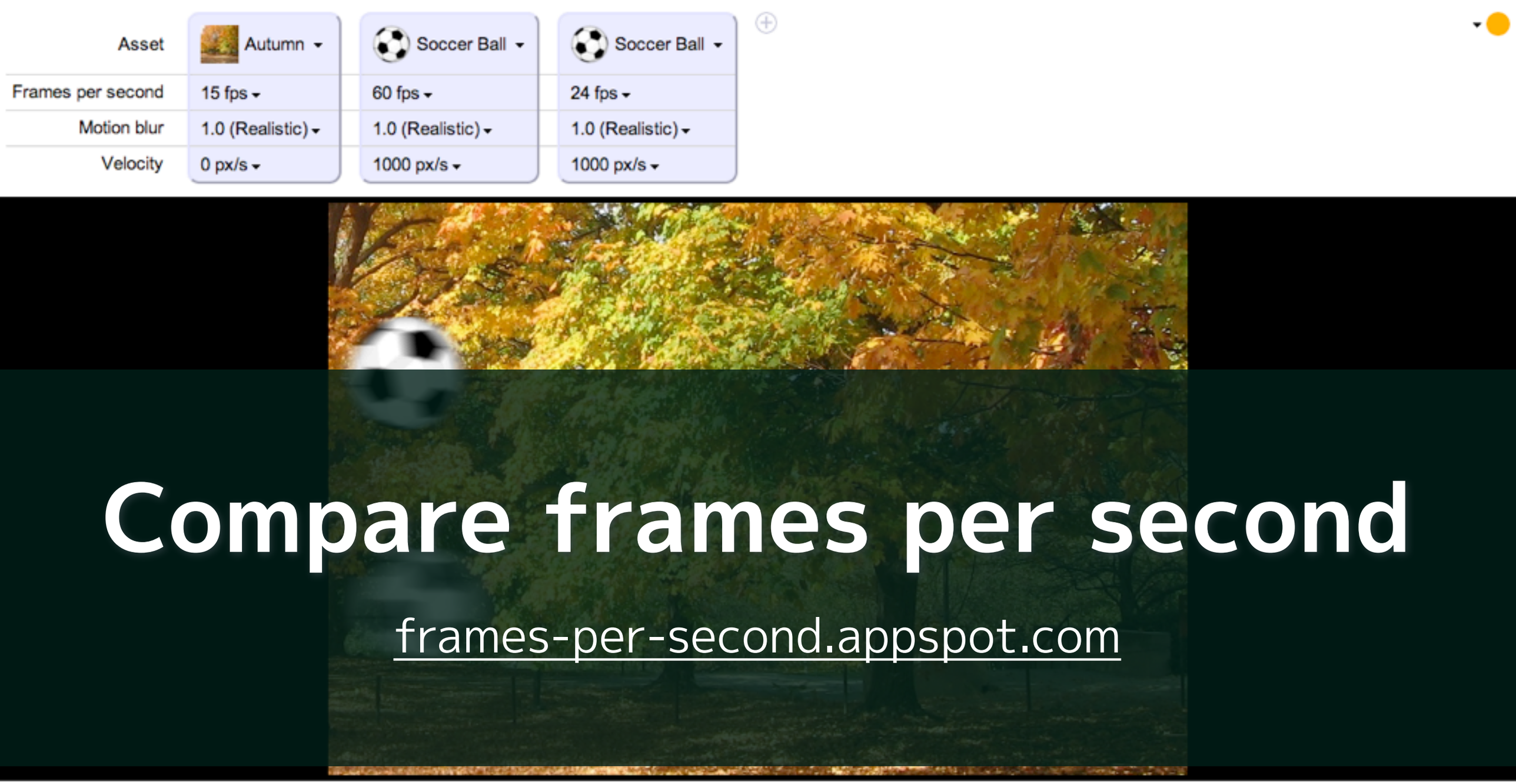
Elements Resources Network Sources Timeline Profiles Audits Console PageSpeed

Events
Frames
Memory

30 FPS
60 FPS

RECORDS

All Loading Scripting Rendering Painting 0 of 0 frames shown



Frame rate and motion blur are important aspects of video quality. This demo helps to show the visual differences between various frame rates and motion blur.

A few presets to try out:

- 15 vs. 25 vs. 48 vs. 60 frames per second (with motion blur exaggeration)
- 25 frames per second with and without motion blur
- 60 frames per second with and without motion blur

Motion blur is a natural effect when you film the world in discrete time intervals. When a film is recorded at 25 frames per second, each frame has an exposure time of up to 40 milliseconds (1/25 seconds). All the changes in the scene over that entire 40 milliseconds will blend into the final frame. Without motion blur, animation will appear to jump and will not look fluid.

When the frame rate of a movie is too low, your mind will no longer be convinced that the contents of the movie are continuous, and the movie will appear to jump (also called strobing).

More information:

Render

描画処理について

Paint

Layout

GPU

Google Chrome Canary

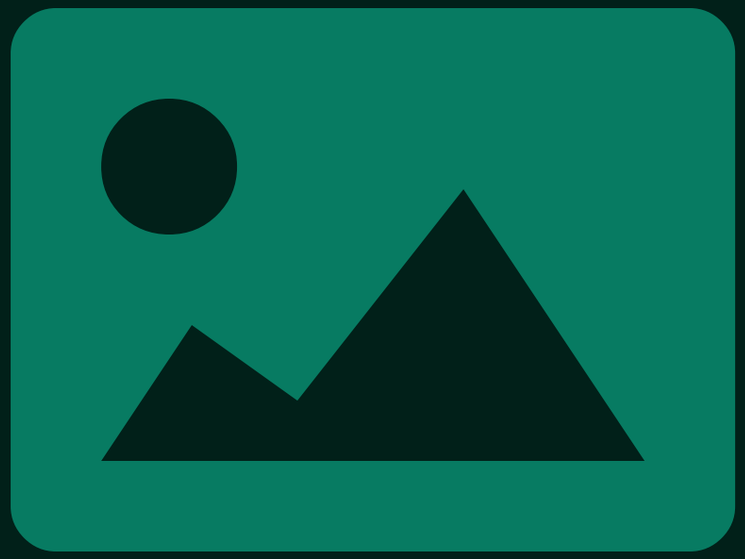
最新の機能がイチ速く搭載される
開発者ツールの進化も速い

いわゆるNightly Build（毎晩更新）
動かない日があっても泣かない

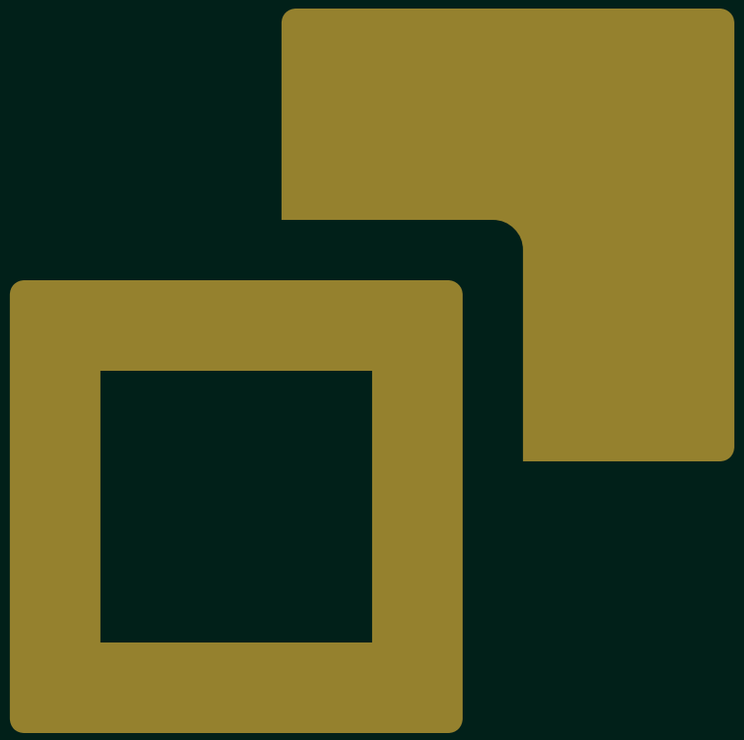
Heavy Painting

重いペイント

ペイントコストの2要因



Big Image

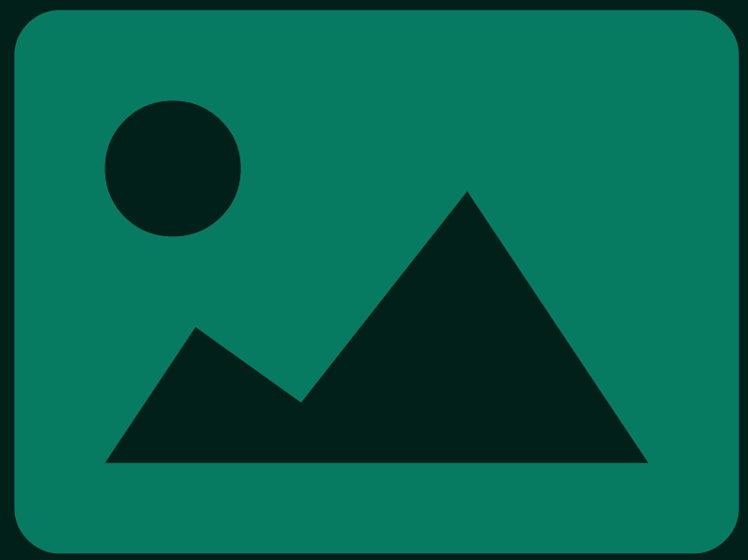


Heavy CSS

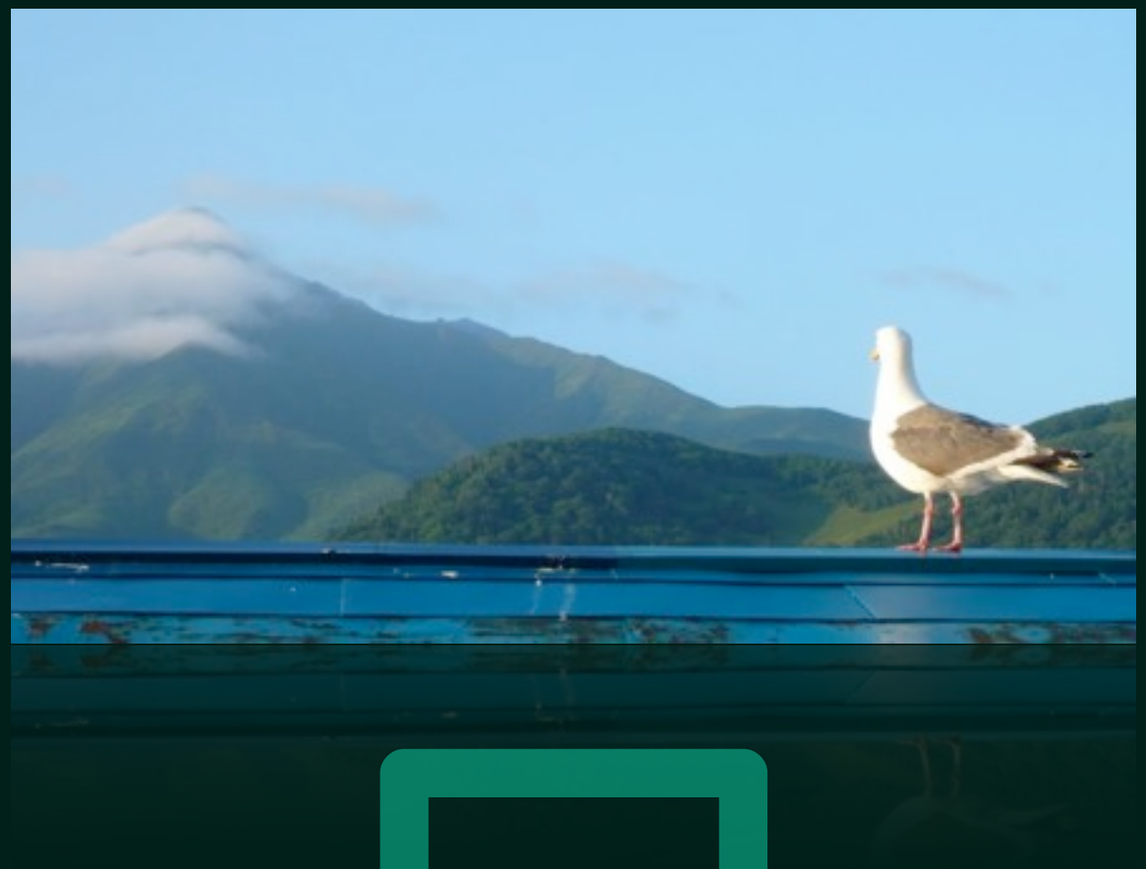
The CPU and GPU Cheatsheet

Performance Checklist for the Mobile Web

大きい画像とリサイズ

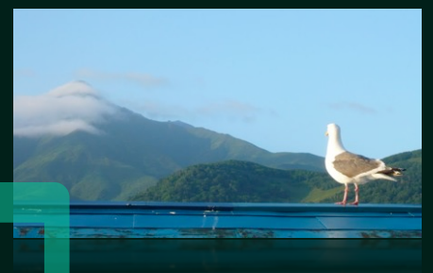


RWDにみる画像リサイズのコスト



max-width: 100%

力任せよりはサーバーサイドで賢く

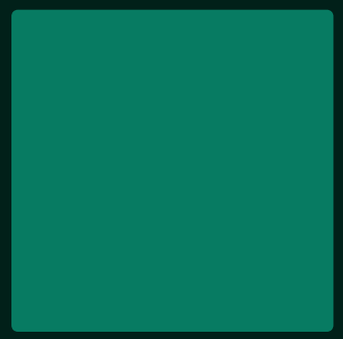


高解像度化によるサイズ増

縦横2倍で、面積は4倍になる

本当に精細さが必要なリソース？

x1



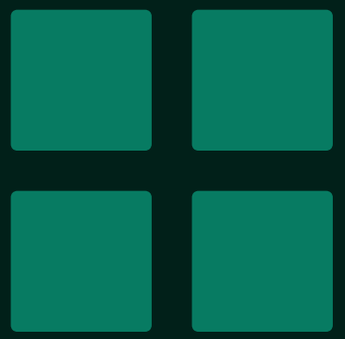
サムネイル

影などの装飾

x1.5

UIパーツ?

x2



テキスト系画像

キービジュアル

重いCSSの過剰利用

どれが重いプロパティか確認しながら直す

THINGS WE LEFT ON THE MOON

News Things Stuff Junk About

We landed on the moon and we left stuff there. A lot of it.

5 9 2 3 7

things left so far.

THINGS WE LEFT ON THE MOON

css3exp.com/moon

LATEST THING NEWS

New things discovered this week

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure **dolor in reprehenderit** in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

TOP CATEGORIES

Furniture	185 items
Fashion	4,729 items
Pets	4 items

NEW THING ALERTS

Your Name

Your Email

THINGS WE LEFT ON THE MOON

News Things Stuff Junk About

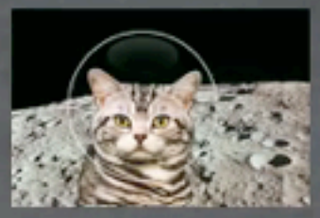
We landed on the moon and we left stuff there. A lot of it.

5 9 2 3 7

things left so far.

1 BIG DOUGHNUT

1 LAWNMOWER



1 ASTRO CAT

1 RECLINER

1 MAGIC GNOME

LATEST THING NEWS

New things discovered this week

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure **dolor in reprehenderit** in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate

TOP CATEGORIES

Furniture	185 items
Fashion	4,729 items
Pets	4 items

NEW THING ALERTS

Your Name

Your Email

THINGS WE LEFT ON THE M ×

css3exp.com/moon/

Page paint time (ms)
16.5 15.3-25.5
Junk About

THINGS WE LEFT ON THE MOON

News Things Stuff

We landed on the moon and we left stuff there. A lot of it.

5 9 2 3 7
things left so far.


1 BIG DOUGHNUT


1 LAWNMOWER


1 MAGIC GNOME

✓ Force accelerated compositing

✓ Enable continuous page repainting

Settings

General
Overrides
Workspace
Experiments

Shortcuts

General

Rendering

- [ ] Show paint rectangles
- [x] Force accelerated compositing
- [ ] Show composited layer borders
- [ ] Show FPS meter
- [x] Enable continuous page repainting
- [ ] Show potential scroll bottlenecks

THINGS WE LEFT ON THE M

css3exp.com/moon/

stuff there. A lot of it.

things left so far.

Page paint time (ms)
8.6 8.4-27.5

1 BIG DOUGHNUT

1 LAWNMOWER

1 AST

C GNOME

LATEST THING NEWS

New things discovered this week

Lorem ipsum dolor sit amet, consectetur adipisic
tempor incididunt ut labore et dolore magna aliqua.
veniam, quis nostrud exercitation ullamco laboris nisi
commodo consequat. Duis aute irure dolor in reprehende

185 items

4,729 items

4 items

box-shadowを外すことで
Page paint time が激減

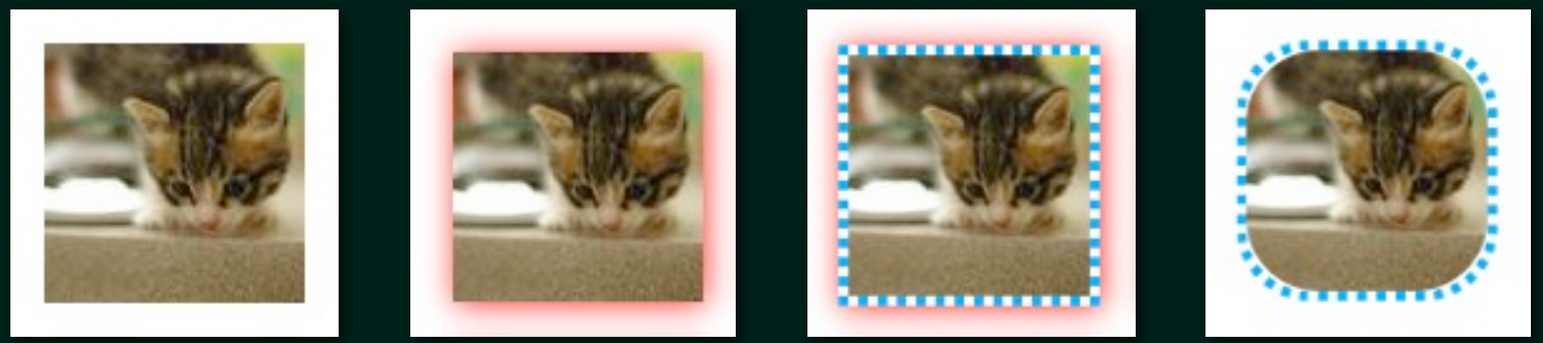
Elements Resources Network Sources Timeline Profiles Audits Console

```
<!DOCTYPE html>
<html lang="en">
  <head>…</head>
  <body>
    <div id="header" class="group">…</div>
    <!-- /header -->
    <div id="wrap">
      <div id="page" class>
        <div class="group">…</div>
        <!-- /.group -->
        <div class="full group">
          <a href="#" id="things-prev">
            <img src="img/icon-prev.png" alt="previous" class>
          </a>
          <a href="#" id="things-next">…</a>
          <ul id="things">…</ul>
        </div>
```

```
  -moz-border-radius: 10px;
  border-radius: 10px;
  -webkit-box-shadow: 0 0 40px
    rgba(0,0,0,.7);
  -moz-box-shadow: 0 0 40px rgba(0,0,0,.7);
  box-shadow: 0 0 100px rgba(0,0,0,.7);
}

media="screen, projection"                    css3exp.com/
html, body, div, span, applet,                reset.css:6
object, iframe, h1, h2, h3, h4, h5, h6, p,
blockquote, pre, a, abbr, acronym, address,
big, cite, code, del, dfn, em, font, img,
ins, kbd, q, s, samp, small, strike, strong,
sub, sup, tt, var, b, u, i, center, dl, dt,
dd, ol, ul, li, fieldset, form, label,
legend, table, caption, tbody, tfoot, thead,
tr, th, td {
  margin: 0;
```

CSSコストの検証 with ねこ36匹



Skia Debugger

描画履歴の閲覧や、解析（プロファイル）が行える

(none)
31.33 ms

box-shadow
102.66 ms

border-radius
146.07 ms

dotted-border
39.29 ms

shadow+dotted
126.35 ms

shadow+radius
649.71 ms

radius+dotted
3099.11 ms

all mix!!
3651.06 ms

※あくまでDebugger上の数字であり、実際のブラウザではもっと早く処理されます

(none)
100 %

shadow+dotted
403 %

box-shadow
328 %

shadow+radius
2,074 %

border-radius
466%

radius+dotted
9,892 %

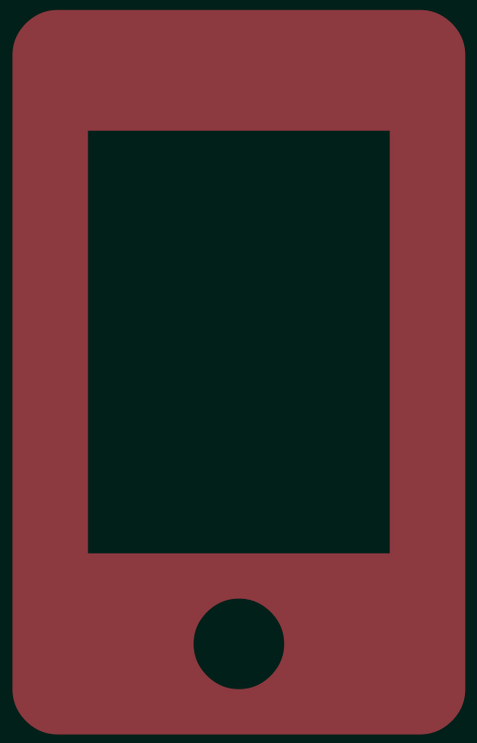
dotted-border
125 %

all mix!!
11,654 %

havelog.ayumusato.com/develop/performance/e560-css_rendering_with_skia_debugger.html

モバイルは特に危ない

Mobile devices are low spec...



シャドウ＋角丸とか頻出しがち

“重い”デザインは確かに存在する

影は可能であれば、いっそ画像にする

当然、Networkコストと天秤

CSSのご利用は計画的に

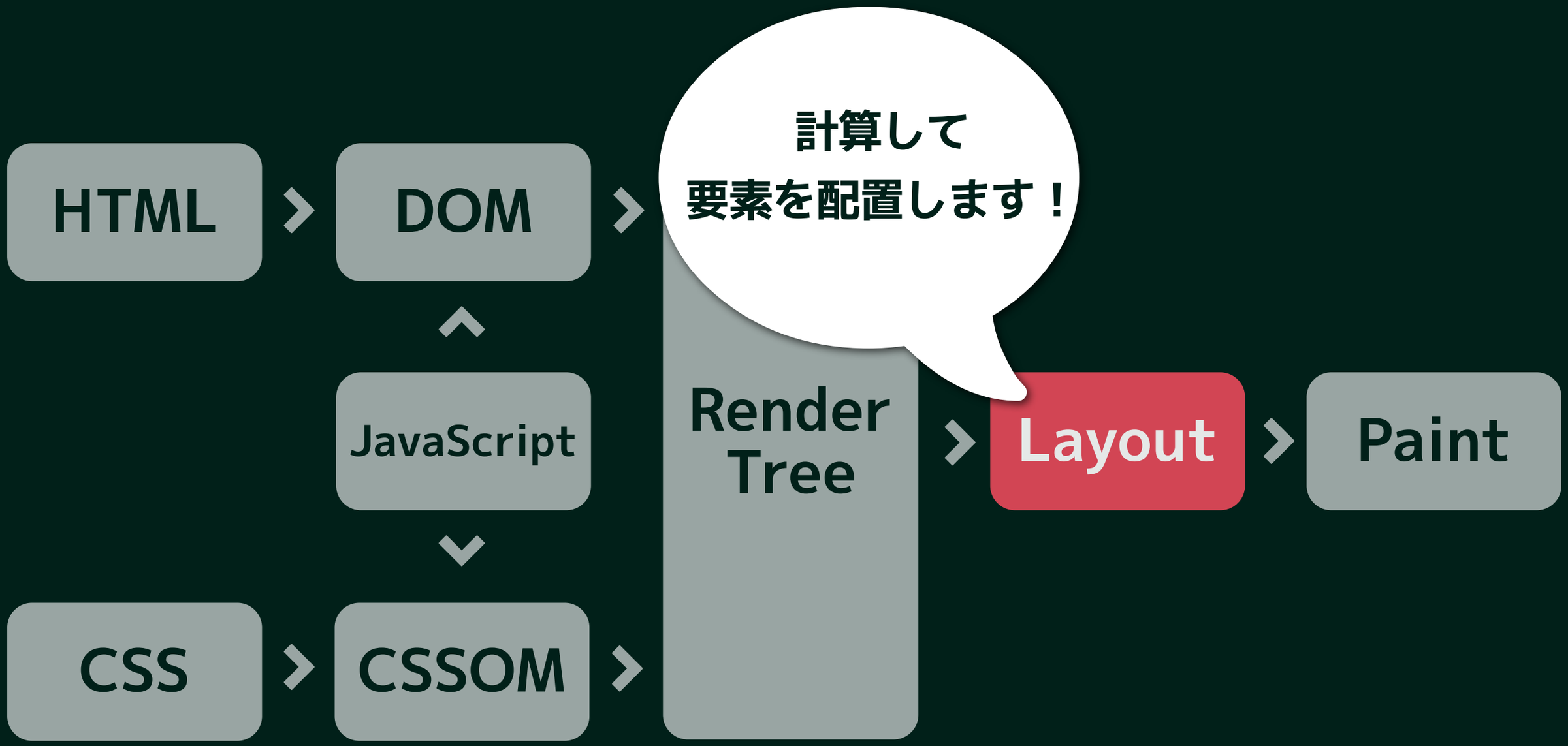
Layout Thrashing

意図しないレイアウト処理の繰り返し

パララックスでありがちな
レイアウト処理の問題

Don’t you talk about it !!

要素のレイアウト（再配置・リフローとも）



- Authoring & Development Workflow
- Editing Styles & DOM
- Managing Application Storage
- Evaluating network performance
- Debugging JavaScript
- Performance Profiling With The Timeline
  - Demo: Finding forced synchronous layouts
- Profiling JavaScript performance
- Profiling Memory Performance
- Mobile Emulation
- Device Emulation
- Keyboard Shortcuts
- Tips And Tricks
- Settings
- Remote Debugging on Android
- Additional Resources
- Contributing

Timeline demo: Diagnosing forced synchronous layouts

This demo shows how you can use the Timeline to identify a kind of performance bottle-neck called 'forced synchronous layouts'. The demo application animates several images back and forth using `requestAnimationFrame()`, the recommended approach for performing frame-based animation. But there's a considerable amount of stuttering and jank as the animation runs We'll use the Timeline to diagnose what's going on.

For more information about using Frames mode and forced asynchronous layouts see Locating forced synchronous layouts in Using the Timeline.

Make a recording

First you'll make a recording of the animation.

1. Click **Start** to start the animation.
2. Open the Timeline panel on the page, and go to the Frames view.
3. Click the Record button in the Timeline.
4. After after a second or two (10-12 frames recorded) stop the recording and click **Stop** to stop the animation.

Stop

Diagnosing forced synchronous layouts

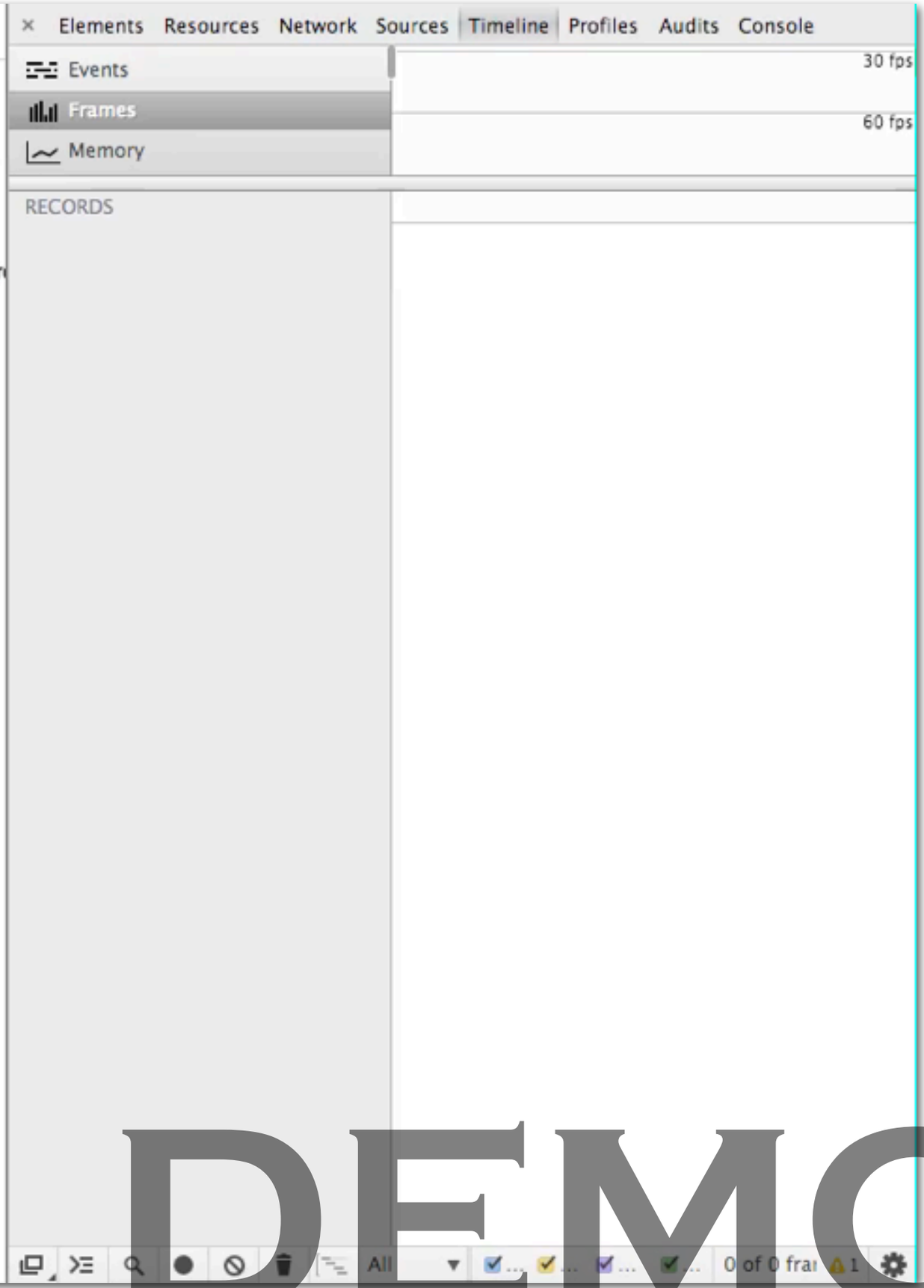
developers.google.com/chrome-developer-tools/docs/demos/too-much-layout/

Make a recording

First you'll make a recording of the animation.

1. Click **Start** to start the animation.
2. Open the Timeline panel on this page, and go the Frames view.
3. Click the Record button in the Timeline.
4. After after a second or two (10-12 frames recorded) stop the recor
   animation.

Start



Start

が連続して
危険な状態

Analyze the recording

Looking at the recording of the first few frames it's clear that each
hover your mouse over one of the frames a pop-up appears show

Elements Resources Network Sources Timeline Profiles Audits Console PageSpeed

Events
Frames
Memory

30 FPS
60 FPS

78.189 ms 153.170 ms 213.638 ms

Layout (source.js:48)
Recalculate Style (source.js:48)
Request Animation Frame (sou…
Paint (377 × 600)
Function Call (× 39)
Animation Frame Fired (source…
Paint (374 × 560)

Layout – Details

Duration 0.266 ms (at 826 ms)
Note Forced synchronous layout is a possible performance bottleneck.

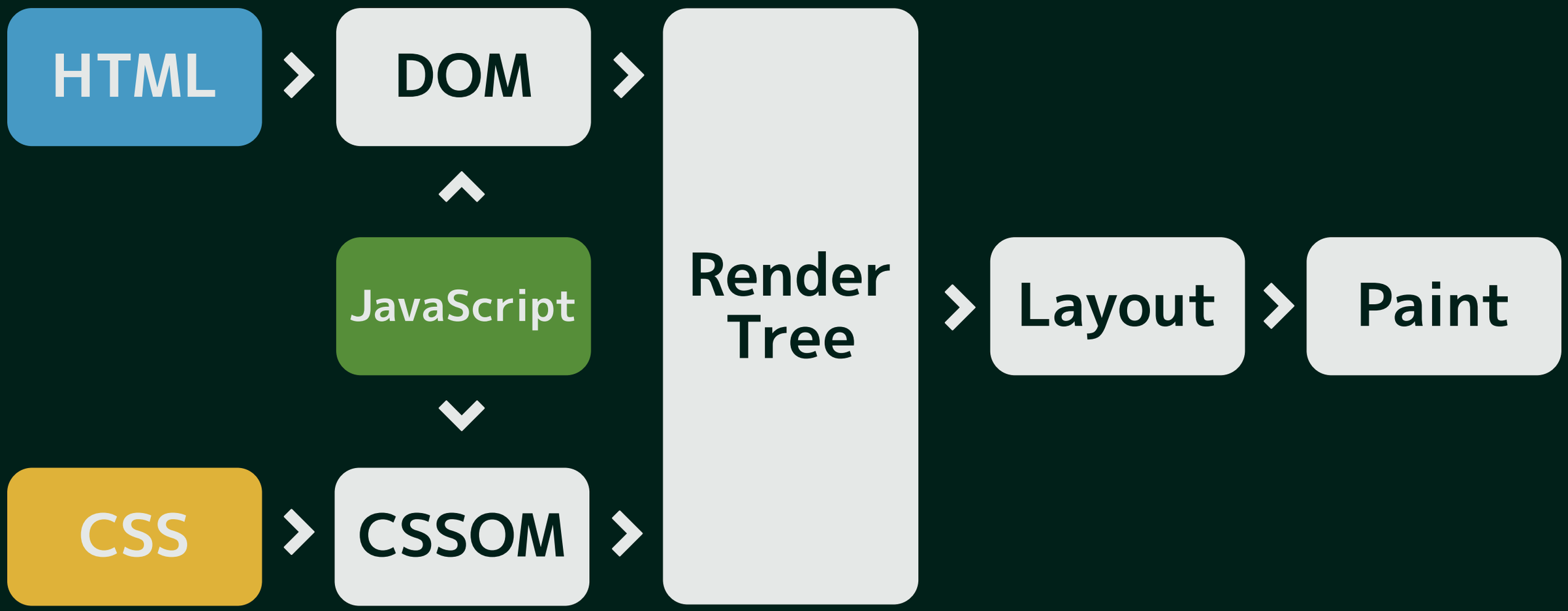
Layout invalidated update @ source.js:48

Layout forced update @ source.js:48

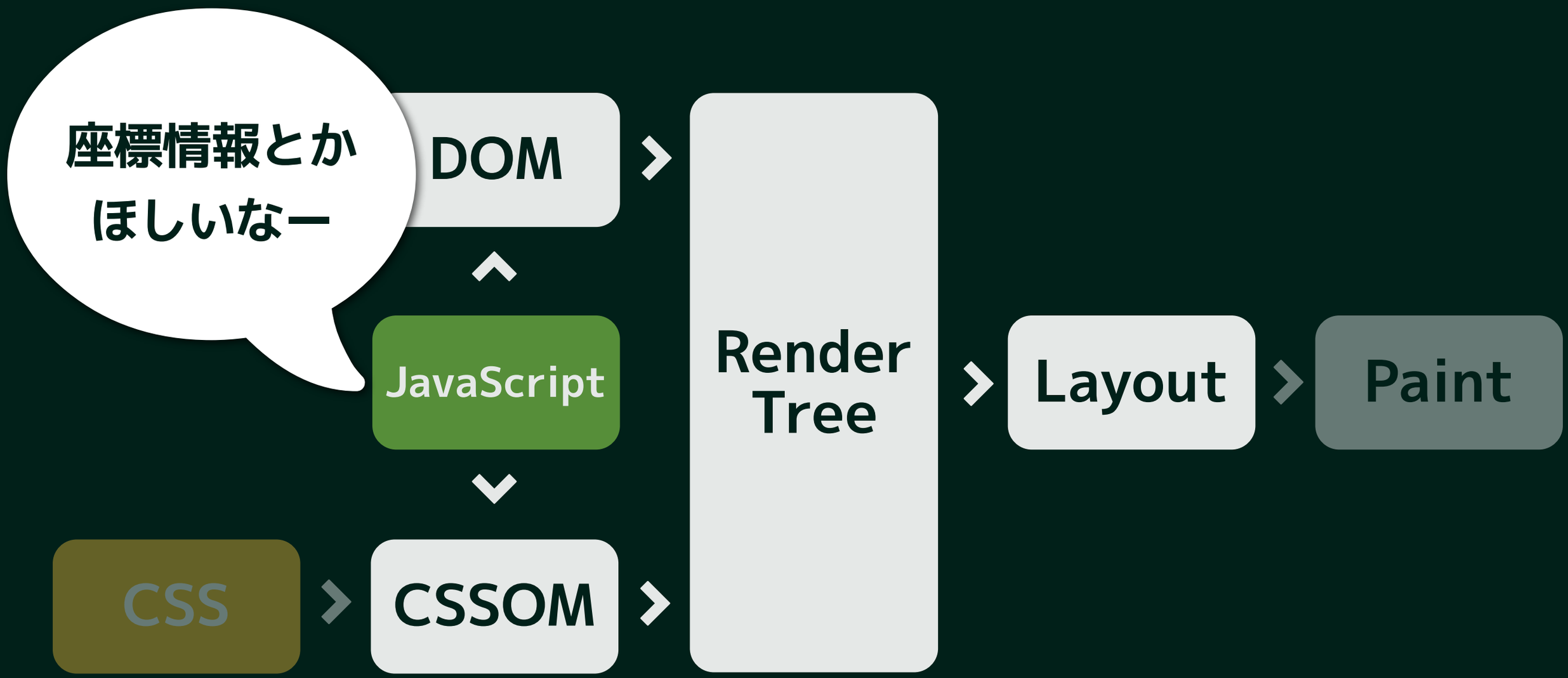
問題のループの中で起こっていること

Forced Synchronous layout

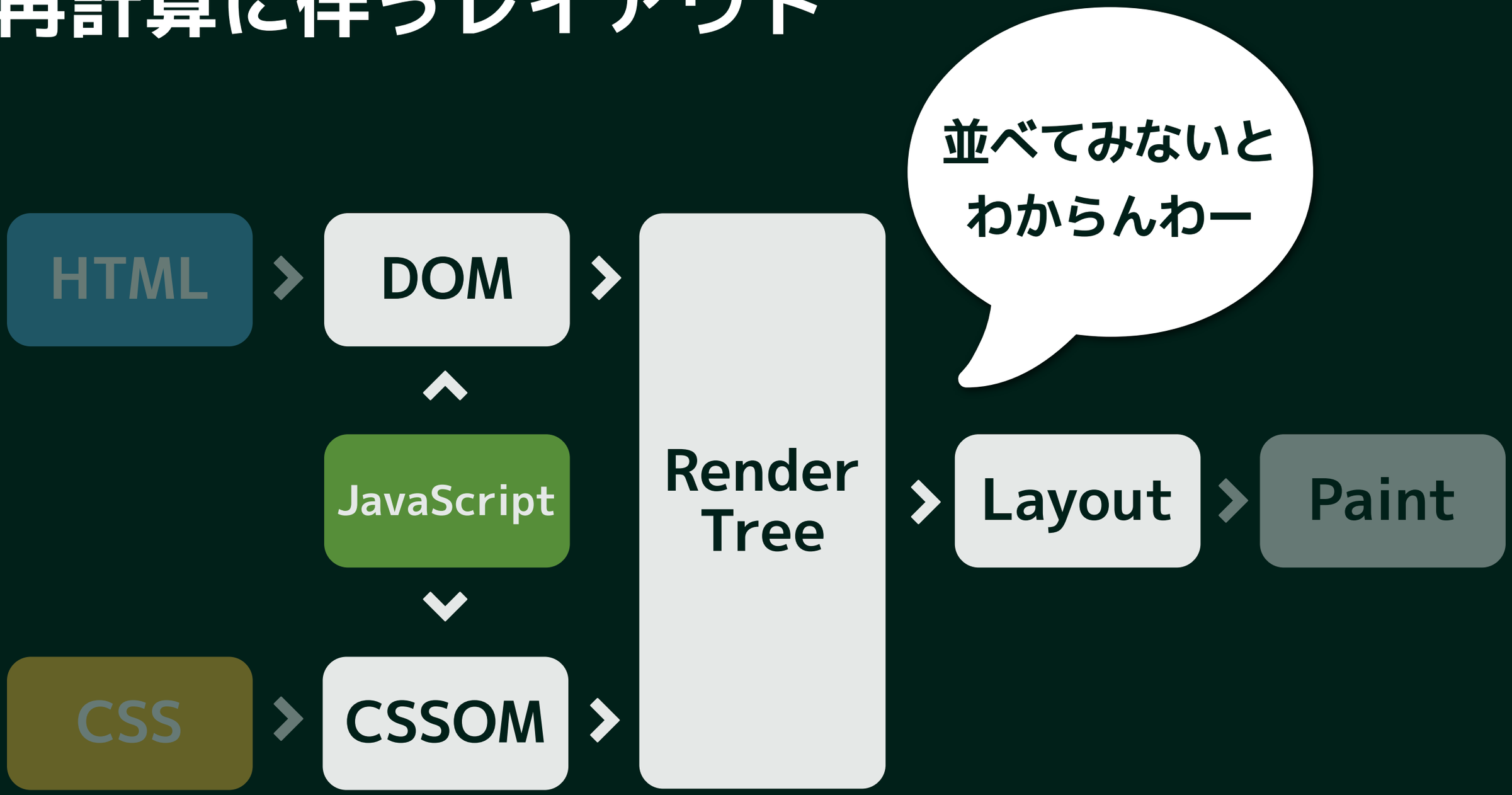
レンダリングのフロー



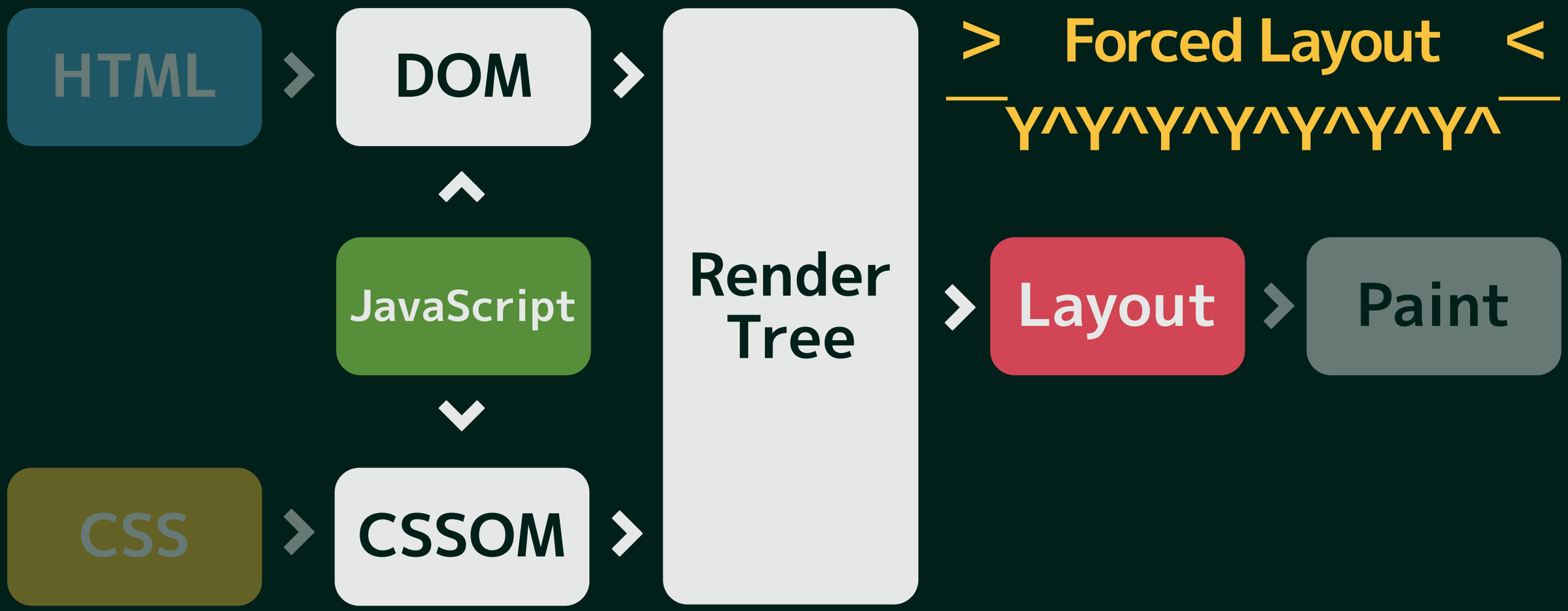
座標情報の要求


座標情報とか
ほしいなー
DOM
JavaScript
CSS
CSSOM
Render
Tree
Layout
Paint

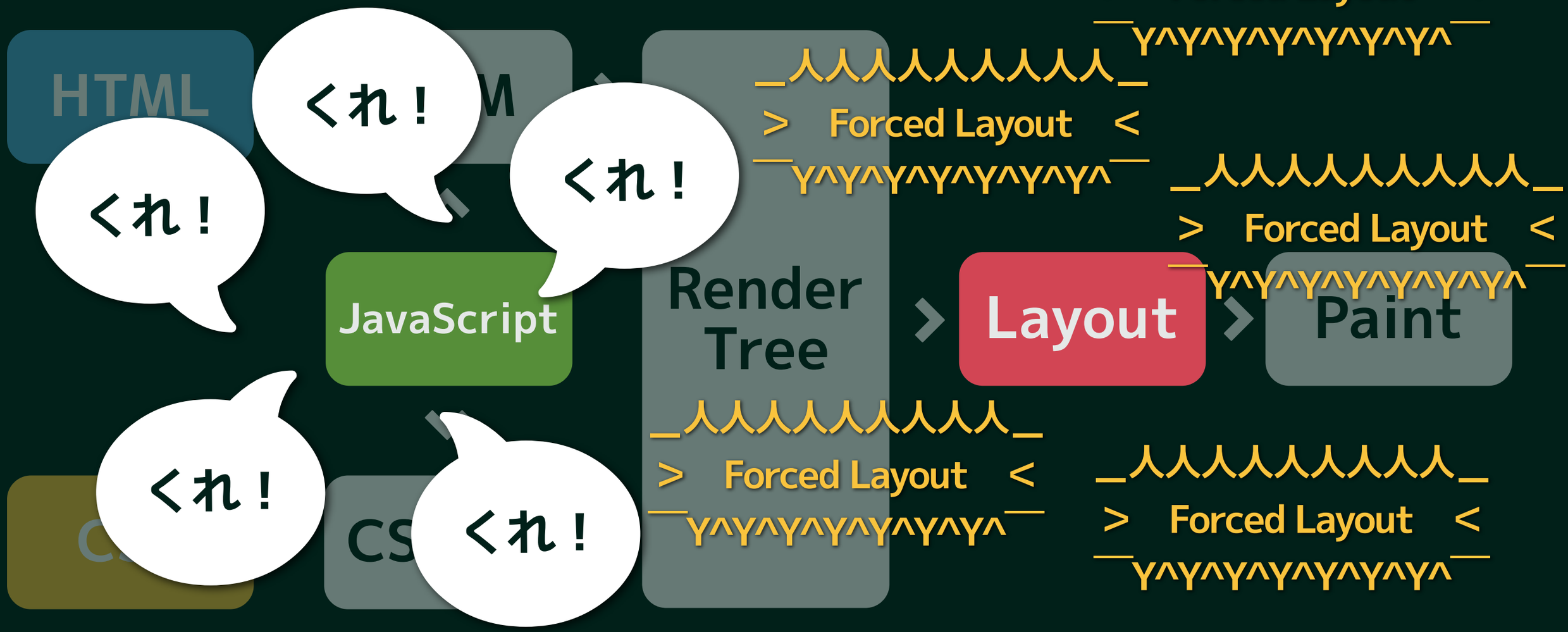
再計算に伴うレイアウト



強制レイアウト（この間、スクリプトはブロック）

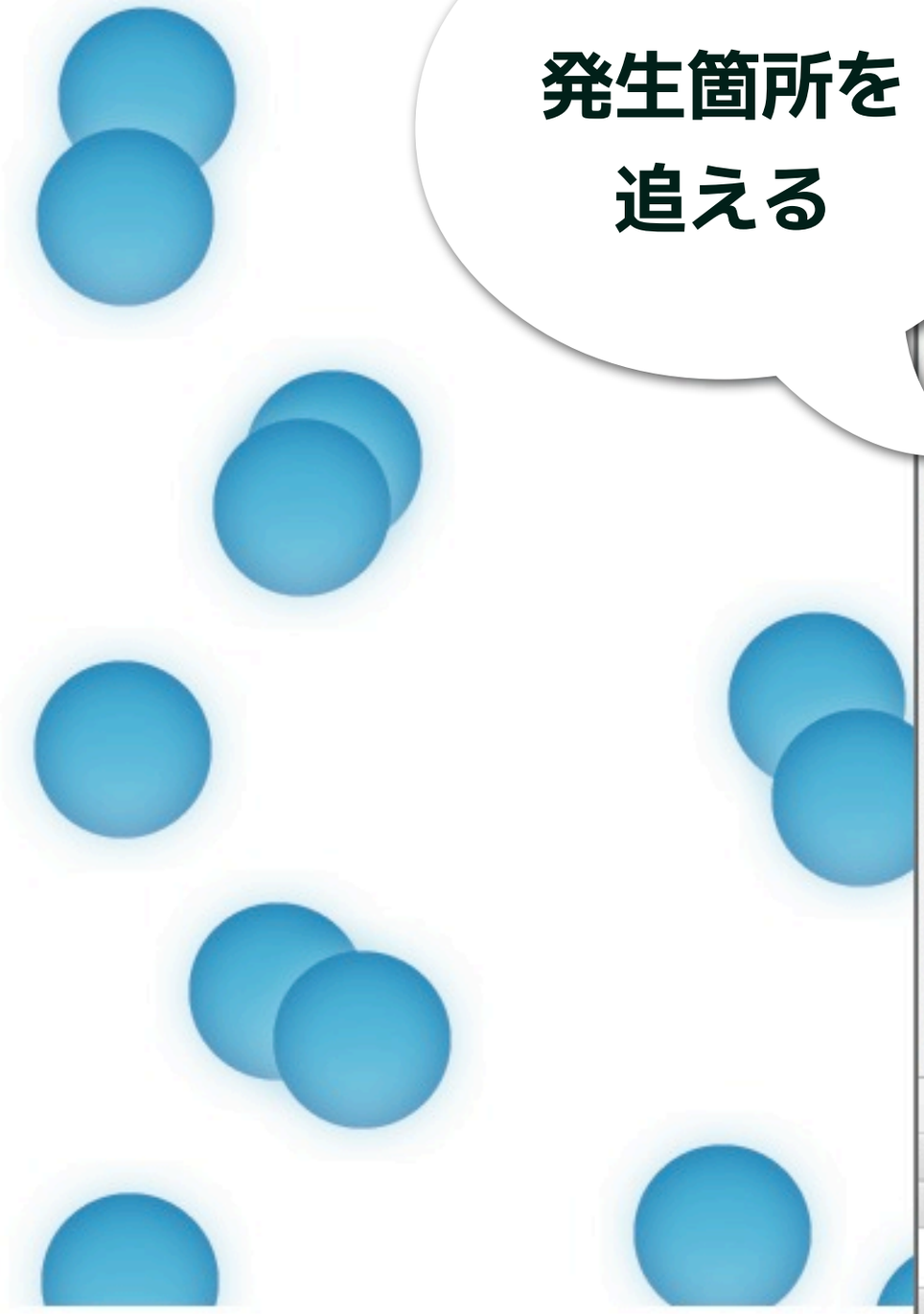


スラッシングレイアウトの発生



特定のプロパティへのRead/Writeで発生

Start



発生箇所を
追える

Analyze the recording

Looking at the recording of the first few frames it's clear that each
hover your mouse over one of the frames a pop-up appears show

Elements Resources Network Sources Timeline Profiles Audits Console PageSpeed

source.js

```
30 }());
31
    var raf;
    ar isAnimating = false;
    r btn = document.querySelector('button');
    r movers = document.querySelectorAll('.mover');

        Set the tops of each DOM element
    nction init() {
            movers[0].style.top = '50px';
            for (var m = 1; m < movers.length; m++) {
                movers[m].style.top = (m * 20) + 'px';
            }
        })();
44
45  // animation loop
46  function update(timestamp) {
        for (var m = 0; m < movers.length; m++) {
48          movers[m].style.left = ((Math.sin(movers[m].offsetTop +
49              timestamp / 1000) + 1) * 500) +
50           'px';
51          // movers[m].style.left = ((Math.sin(m + timestamp/1000)+1) * 500) + 'px';
52          }
53      raf = window.requestAnimationFrame(update);
54  }
55
56  function toggleAnim(e) {
57      if (isAnimating) {
58          window.cancelAnimationFrame(raf);
59          isAnimating = false;
60          e.currentTarget.innerHTML = 'Start';
61      } else {
62          raf = window.requestAnimationFrame(update);
63          isAnimating = true;
64          e.currentTarget.innerHTML = 'Stop';
65      }
66  }
```

Scope Variables | Watch Expressions

▼ Call Stack

▼ Breakpoints

No Breakpoints

▶ DOM Breakpoints

▶ XHR Breakpoints

▶ Event Listener Breakpoints

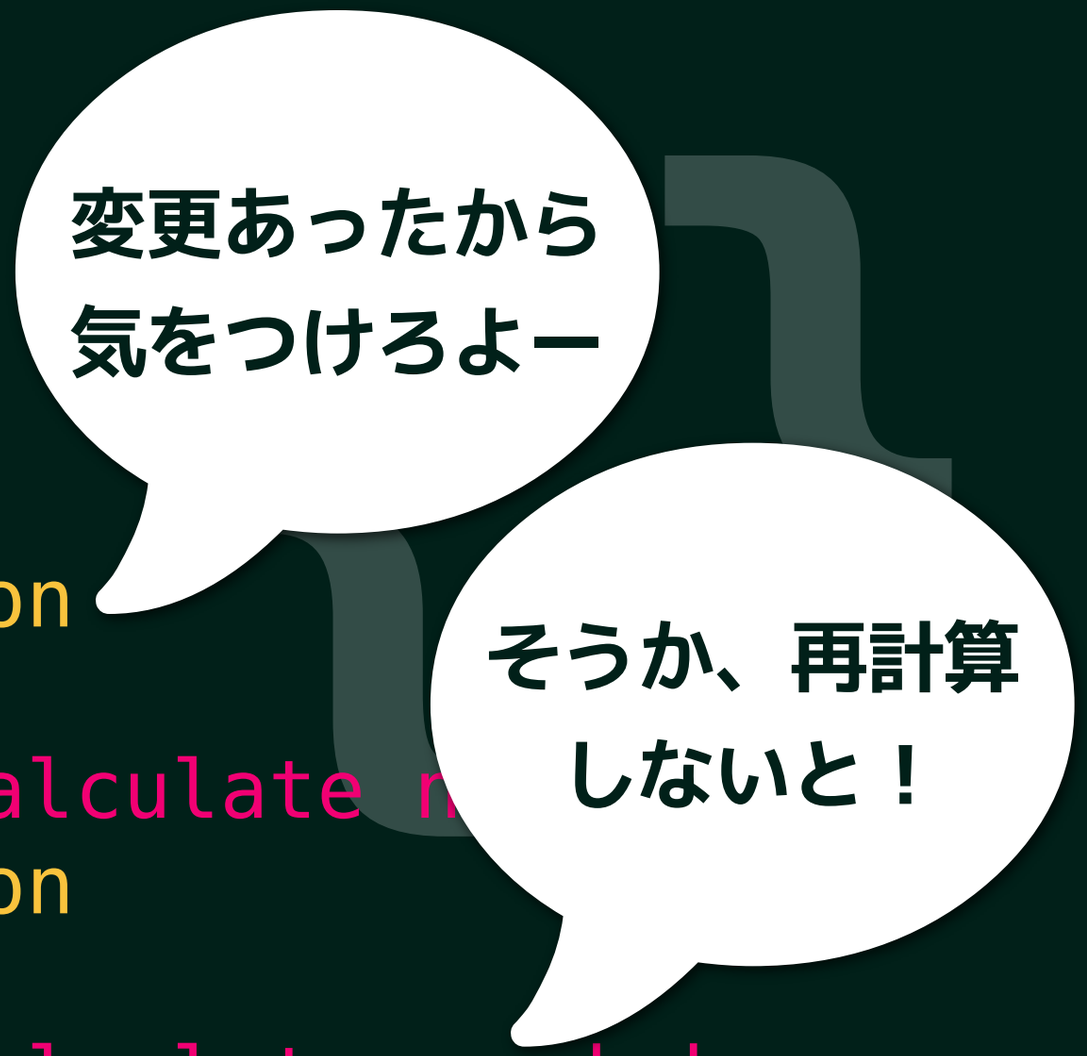
▶ Workers

問題のループ

```
// animation loop
function update(timestamp) {
  for (var m = 0; m < movers.length; m++) {
    // Layout invalidated
    movers[m].style.left = (
      (Math.sin(
        // Layout forced
        movers[m].offsetTop + timestamp / 1000
      ) + 1) * 500
    ) +'px';
  }
  raf = window.requestAnimationFrame(update);
}
```

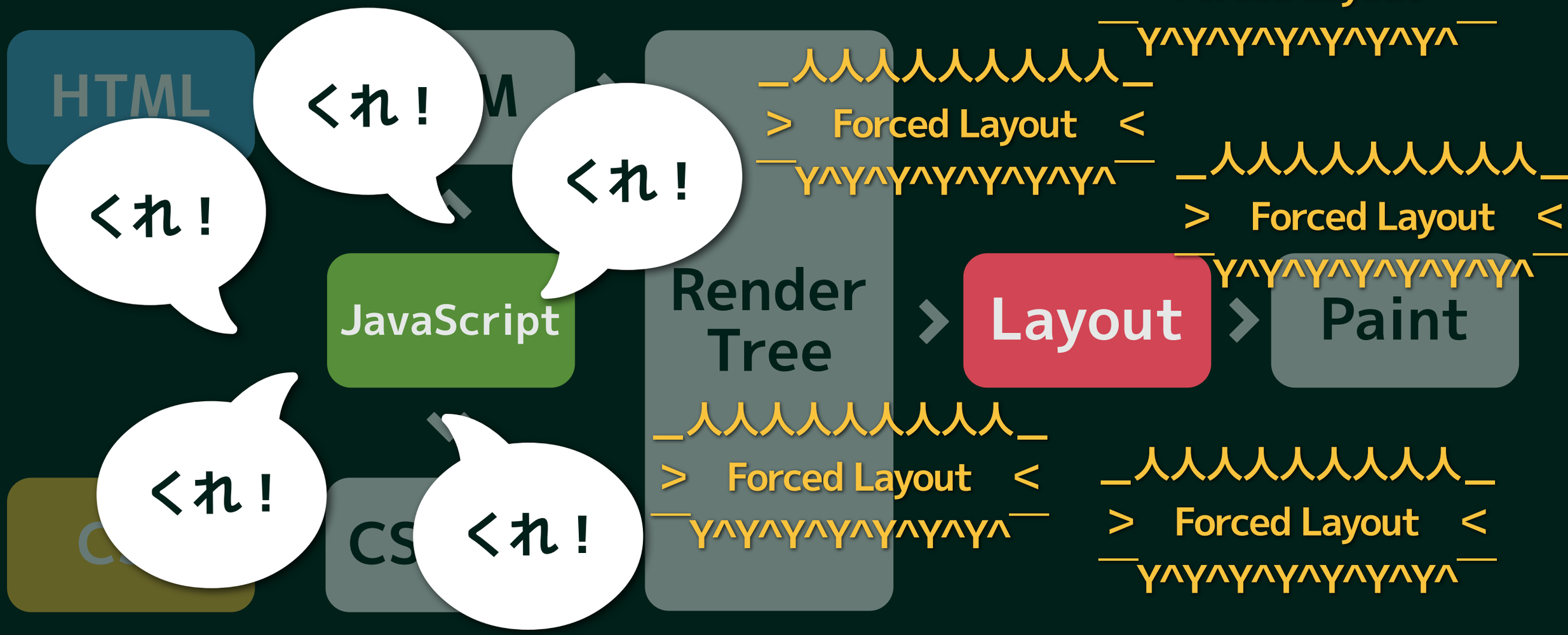
存外にマメな処理

```
--- 1st loop ---
R offsetTop          up to date
W style.left         dirty flag on
--- 2nd loop ---
R offsetTop          dirty...recalculate n
W style.left         dirty flag on
--- 3rd loop ---
R offsetTop          dirty...recalculate needed
W style.left         dirty flag on
--- 4rd loop ---
R offsetTop          dirty...recalculate needed
W style.left         dirty flag on
--- 5th loop ---
...
..
```



www.youtube.com/watch?v=0UNWi7FA36M

スラッシングレイアウト (再)



問題になるプロパティ・メソッド

```
Element
  clientHeight clientLeft clientTop clientWidth
  offsetHeight offsetLeft offsetTop offsetWidth
  scrollHeight scrollLeft scrollTop scrollWidth
  innertText   outerText  getBoundingClientRects
                                              etc...
MouseEvent
  layerX       layerY     offsetX   offsetY

Window
  scrollBy     scrollTo   scrollX   scrollY
  getComputedStyle

Frame, Document & Image
  height       widh
```

kellegous.com/j/2013/01/26/layout-performance/

GPU

魔法のコスト

アカウント作成

ページ　ノート　　閲覧　編集　履歴表示　検索

ペディア
百科事典

ージ
ティ・ポータ

来事
ージ
新
表示
ージ
ード (ウィキメ
コモンズ)

せ
報告

ペディアに関
問い合わせ

ボックス

版

оски
ski

ch

WIKIMEDIA

あなたの意見を伝えましょう！
新しいプライバシーポリシーの草案へのご意見をお願いします。

[翻訳にご協力く

Graphics Processing Unit

Graphics Processing Unit（グラフィックス プロセッシング ユニット、略して**GPU**）とは、パーソナルコンピュータやワークステーション等の画像処理を担当する主要な部品のひとつ。**Visual Processing Unit**（ビジュアル プロセッシング ユニット、**VPU**）という名称もある。

を行う集積回路をさす。現在の高機能GPUは高速のVRAMと接続され、グラフィックスシェー

目次 [非表示]

1 歴史
　1.1 1970年代～1980年代
　1.2 1990年代
　1.3 2000年代
　1.4 2010年代
2 ゲーム機
3 組み込みシステム
4 GPU 開発企業
　4.1 チップセットまたはCPU統合GPUのみ手がけている企業
　4.2 過去にGPUまたはビデオチップを手がけていた企業
5 その他
　5.1 「GPU」という名前

NVIDIA製のGPU - GeFo
6600 GT

Graphics Processing Unit

ja.wikipedia.org/wiki/Graphics_Processing_Unit

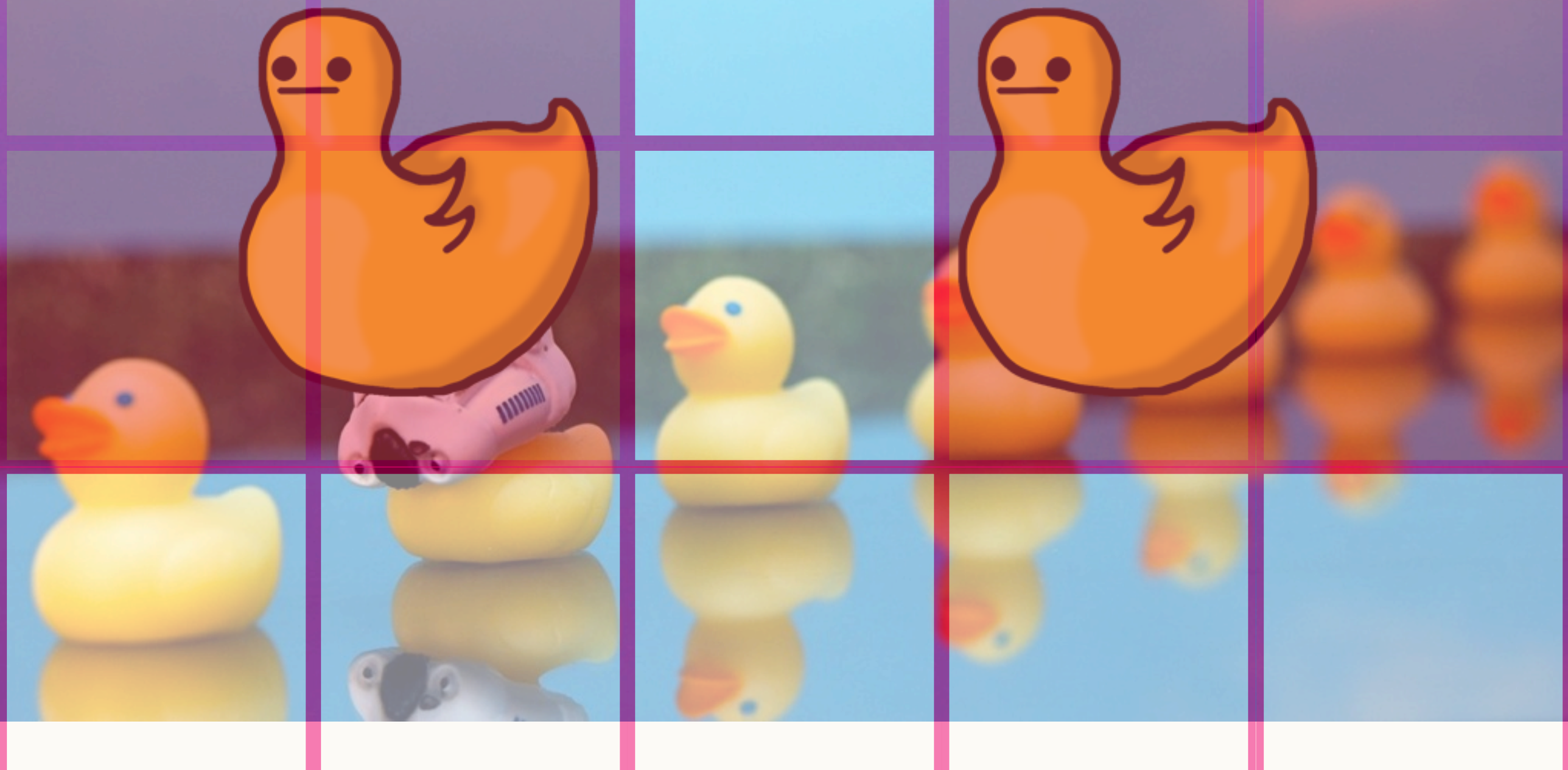
Window size: 1024 x

ここは特にWebKit Chromiumポートの話

アニメーションと
ポイントにみるGPU

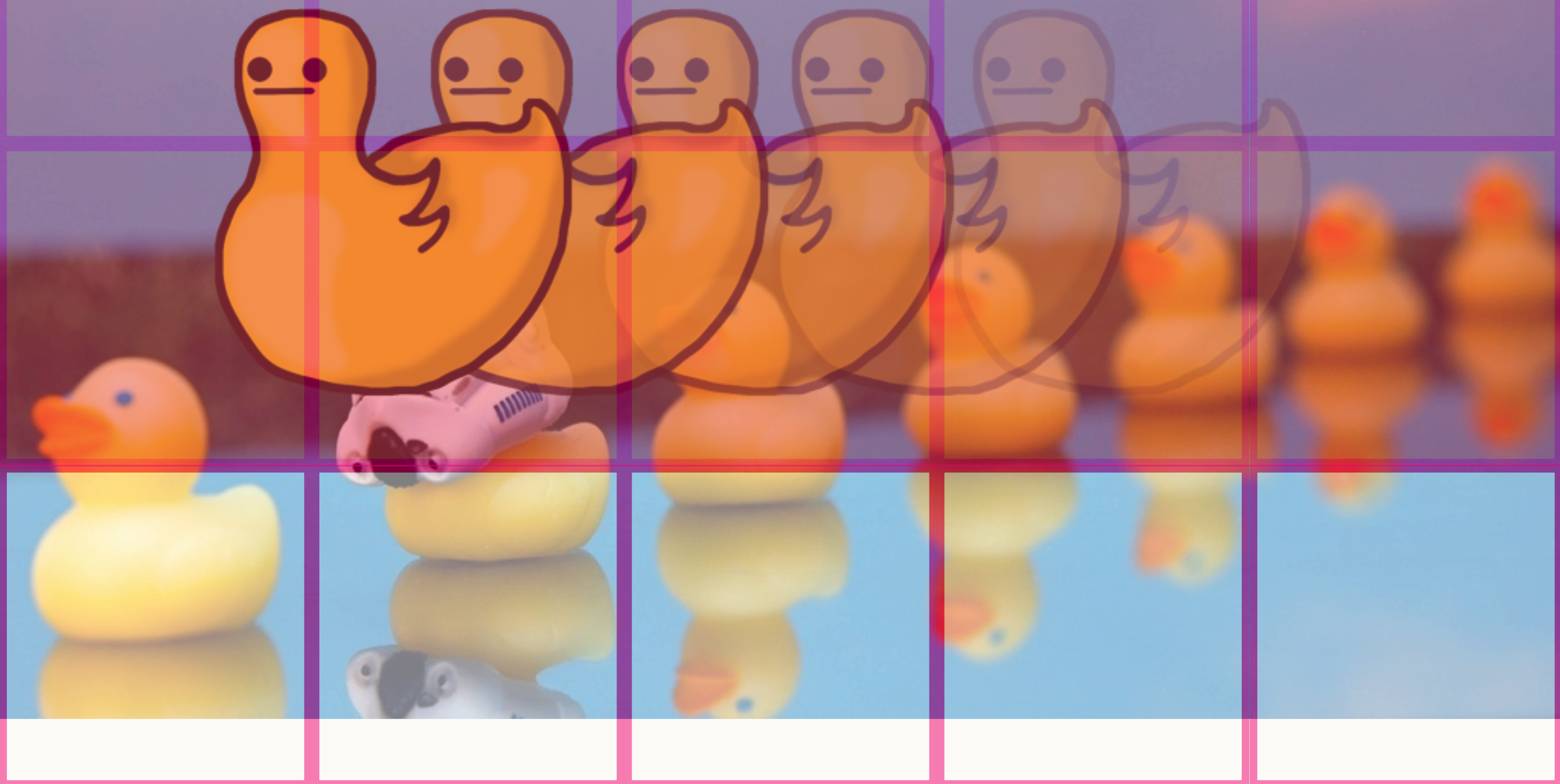
ポジションの移動

2箇所分のペイント



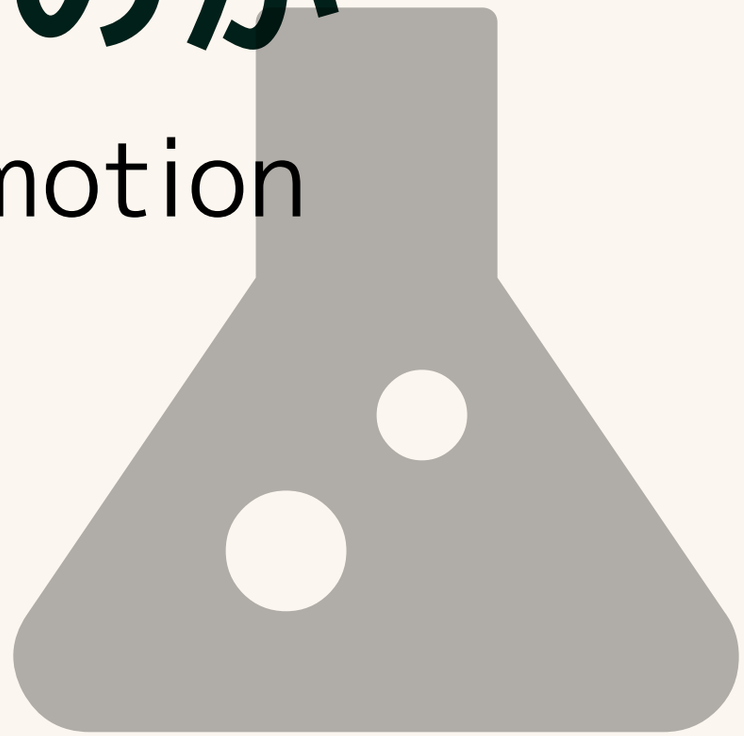
タイマーアニメーションの場合

毎フレームごとにアバババ

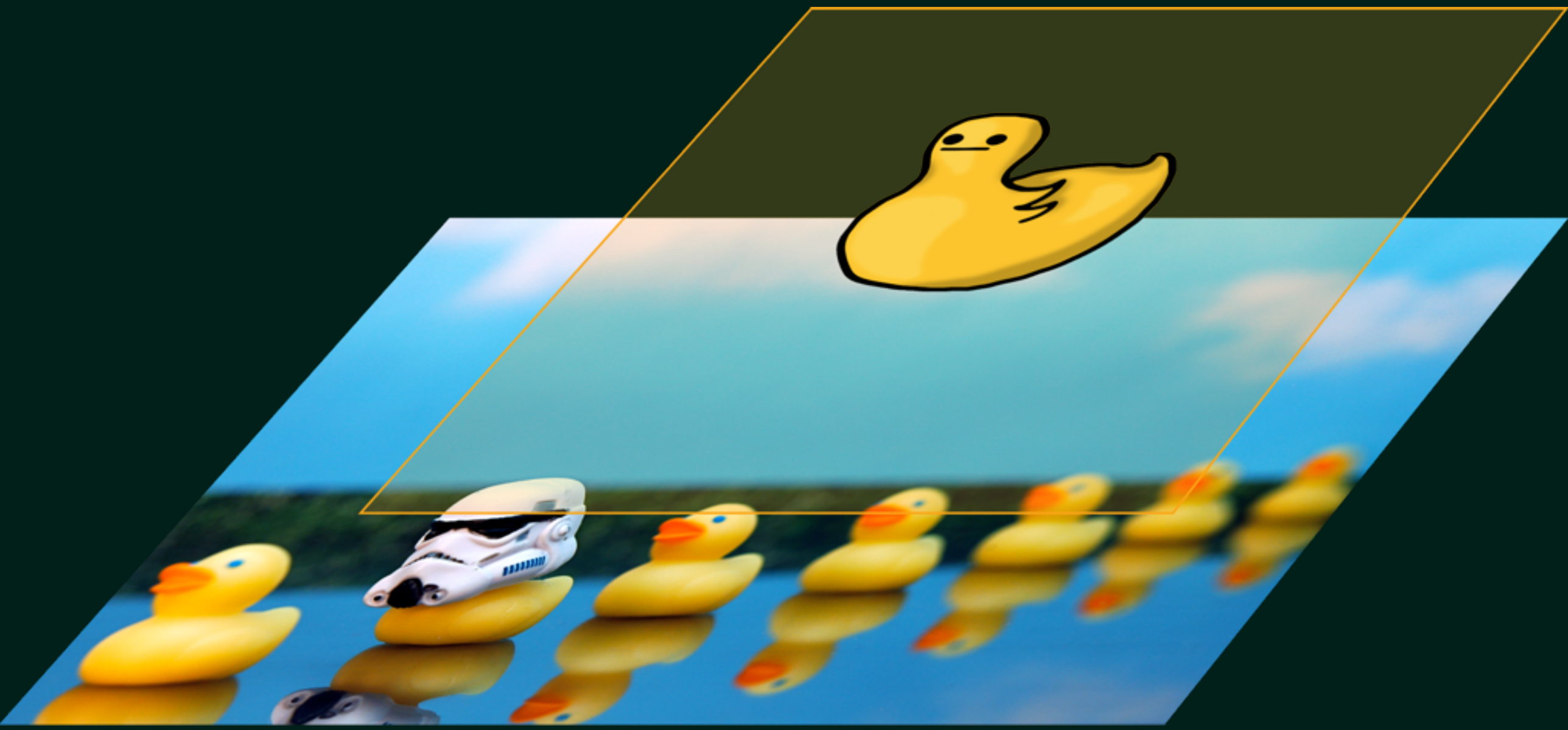


CSS Animations はなぜ滑らかに動くのか

Animation based layer promotion



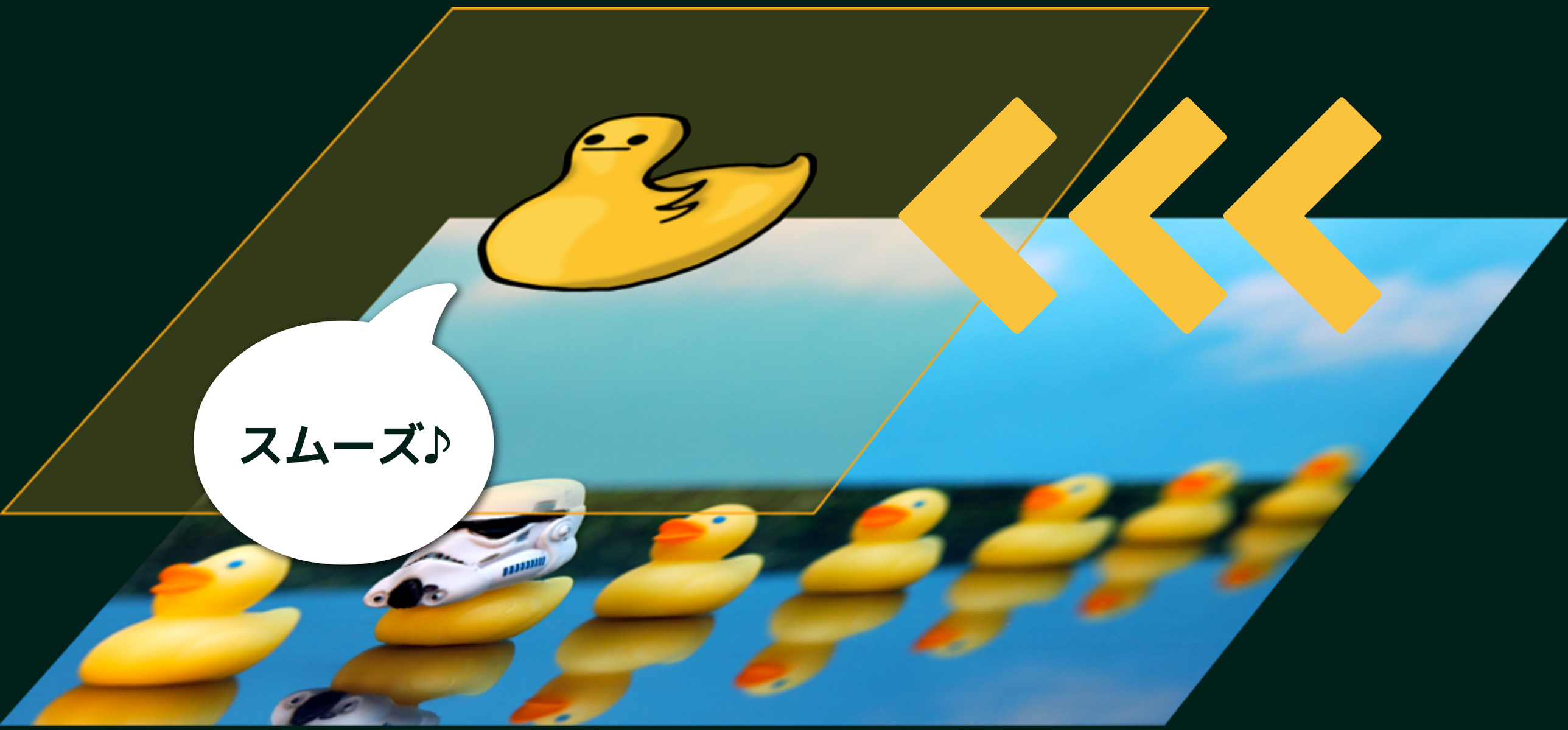
合成レイヤーにテクスチャをのせる



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

velocityconf.com/velocity2013/public/schedule/detail/31377

合成レイヤー上でテクスチャを動かす



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

velocityconf.com/velocity2013/public/schedule/detail/31377

GPUにテクスチャを転送して処理

ポジション
大きさ　これらの変更はGPU内で
角度　高速に処理できる！
透明度

やりすぎると辛い

Assign time layer promotion

トリガーになるプロパティを適用すると
描画系の不具合が副次的に改善することがある

が、あまり広い範囲に適用すると
GPUへの過転送が起こり
パフォーマンス低下につながる

合成レイヤーが意図せず生成される事故と対策

Accidental layer creation

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 paragraphs. This means the headings and paragraphs are layered above the box. When the box get its own texture-backed layer, by starting a transform-based animation, anything that could appear on top of it must also get a texture-backed layer, as layers without one cannot be rendered on top.

How do I fix it?

Result

Edit in JSFiddle

44 × 44 Lorem Ipsum is simply dummy text of the printing and typesetting industry. Like

44 × 44 Lorem Ipsum is simply dummy text of the printing and typesetting industry. Like

44 × 44 Lorem Ipsum is simply dummy text of the printing and typesetting industry. Like

44 × 44 Lorem Ipsum is simply dummy text of the printing and typesetting industry. Like

Elements Resources Network Sources Timeline Profiles Audits Console

```
<!DOCTYPE html>
<html slick-uniqueid="3">
  <head>…</head>
  <body>
    <div id="wrapper">
      <div id="head">…</div>
      <div id="tabs" style="height: 404px;">
        <div class="tCont result active" id=
        "result">
          <iframe src="http://fiddle.jshell.net/
          ahomu/wqXDR/2/show/light/" style="height:
          406px;">
            #document
              <!DOCTYPE html>
```

Computed Style Show inherited

Styles

```
element.style {
}
```

Matched CSS Rules

```
.container div {  fiddle.jshell.n…show/light/:31
  position: relative;
  text-align: right;
}
```

```
div {                          user agent stylesheet
  display: block;
}
```

Metrics

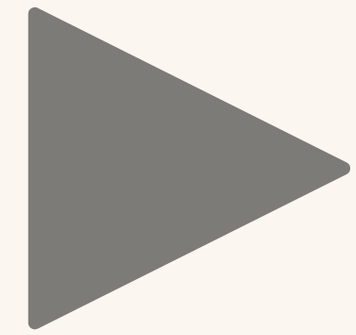
html body #wrapper #tabs #result iframe html body section div

DEMO

Compute

演算・リソース

JavaScript
Garbage Collection
Memory Leak



JavaScriptの実行

ガベージコレクション

メモリリーク

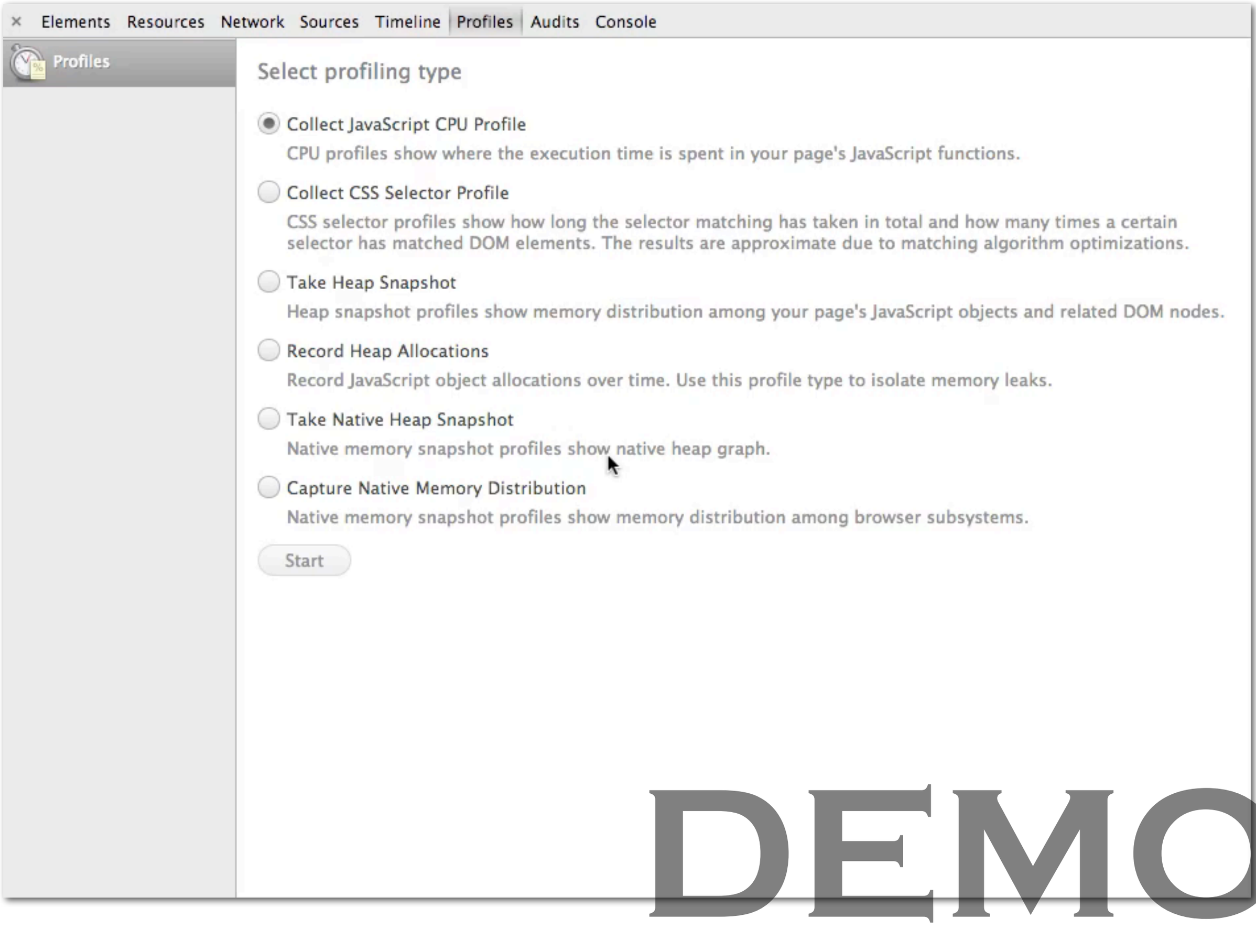
JavaScriptの実行

ガベージコレクション

メモリリーク

JavaScript CPU Profiler

処理の実行にかかった時間


Elements
Resources
Network
Sources
Timeline
Profiles
Audits
Console
Profiles
Select profiling type
Collect JavaScript CPU Profile
CPU profiles show where the execution time is spent in your page's JavaScript functions.
Collect CSS Selector Profile
CSS selector profiles show how long the selector matching has taken in total and how many times a certain selector has matched DOM elements. The results are approximate due to matching algorithm optimizations.
Take Heap Snapshot
Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.
Record Heap Allocations
Record JavaScript object allocations over time. Use this profile type to isolate memory leaks.
Take Native Heap Snapshot
Native memory snapshot profiles show native heap graph.
Capture Native Memory Distribution
Native memory snapshot profiles show memory distribution among browser subsystems.
Start

DEMO

JavaScriptの実行

ガベージコレクション

メモリリーク

Timeline > Memory Object Counters

メモリやオブジェクト数の推移

Elements Resources Network Sources Timeline Profiles Audits Console

Events

Frames

Memory

RECORDS

COUNTERS

Document Count [2 – 2]

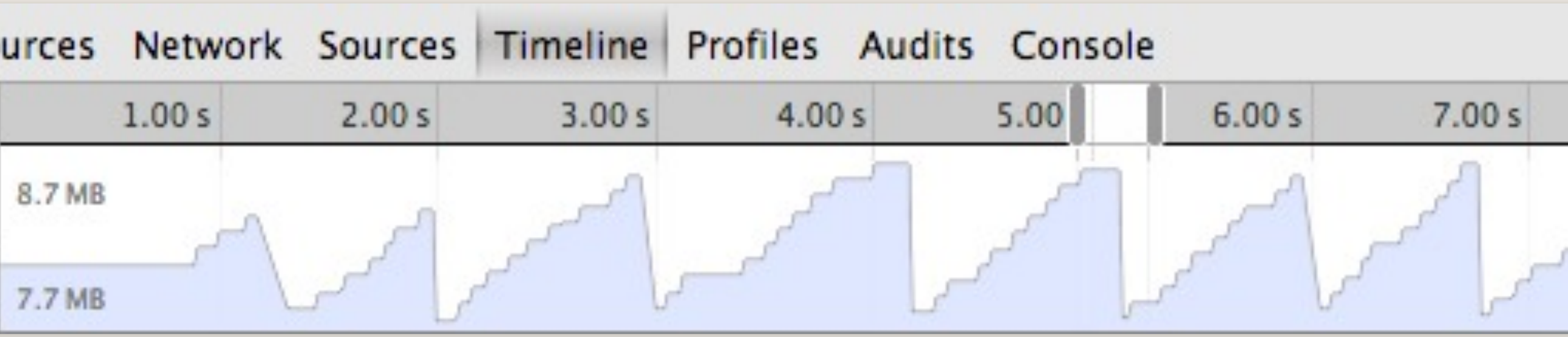
DOM Node Count [2976 – 3214]

Event Listener Count [360 – 401]

All Loading Scripting Rendering Painting 0 of 0 records shown 99

典型的に問題のあるパターンの紹介

Timelineの波形を注意深く見る



のこぎり歯パターン

GCの頻度が高すぎる

inception-explained.com

GC（ガベージコレクション）とは

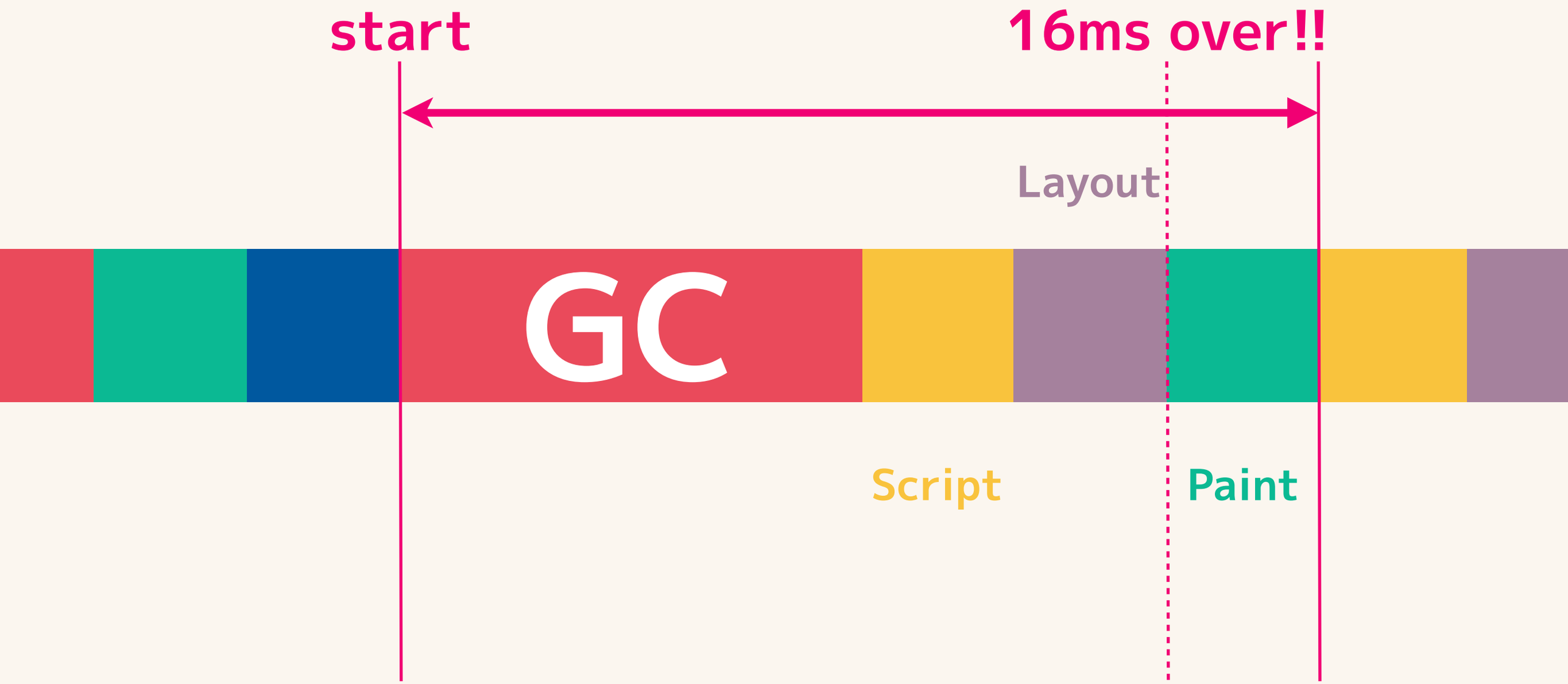
Garbage Collection

オブジェクトの不要な参照から
メモリ領域を解放する機構

→ ルンバのように室内のゴミを
自動で収集してくれるヤツ

GCが実行されている間
他の処理は停止する

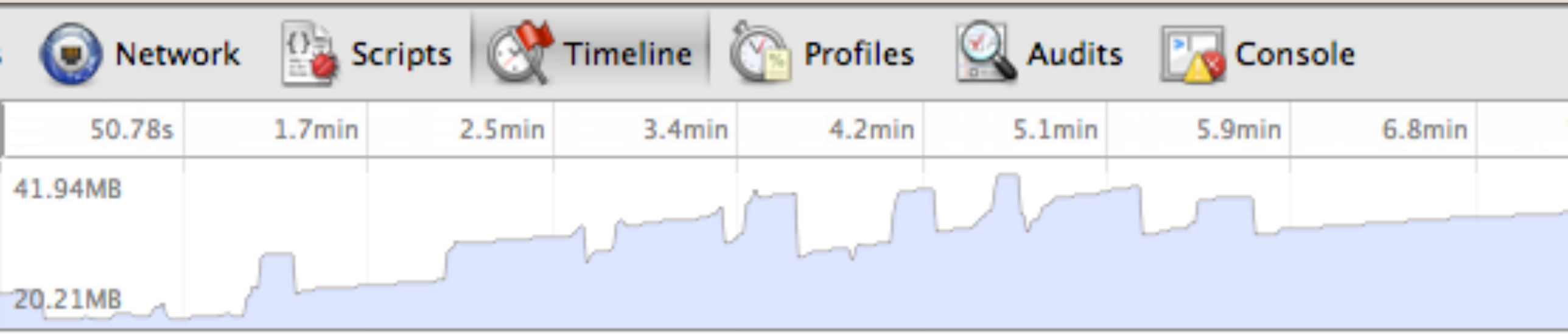
GCが16.666...msの中の数割を持っていってしまう



Pre-Allocate または Object Pool などの対策

buildnewgames.com/garbage-collector-friendly-code/

www.html5rocks.com/en/tutorials/speed/static-mem-pools/



階段パターン

ゴミが回収されない

blog.newrelic.com/2012/07/17/using-chrome-developer-tools-to-find-memory-leaks/

メモリリークとは

Memory Leaks

解放されないメモリ領域が溜まり
徐々に空き領域が減る現象

→ JSにおいてはある種の自然現象
　ゴミは大なり小なり溜まるもの

✓ 変数スコープの複雑な絡み合い

✓ 削除済みDOMのイベントリスナー

✓ オブジェクト間の循環参照

✓ console.logなどによる出力

etc...

coding.smashingmagazine.com/2012/11/05/writing-fast-memory-efficient-javascript

Record Heap Allocations

便利ツールを簡単にご紹介

Ameba ホームへ

Overrides: User-Agent

妊娠中に元カノと飲みに行く夫

ホーム　マイページ　お知らせ 101　IN

新着☆みんなのチェックイン！

マイエリア　すべて

気軽にみんなにいいね！しよう♪

しおりん
53秒前

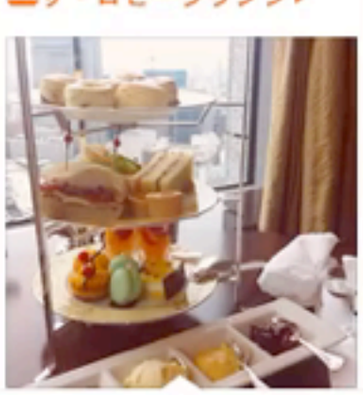
サワディー 渋谷道…

タイカレー豚肉炒め。ピリ辛で美味しかった～

コメント 0　いいね！ 0

maldoror
4分前

ザ・ロビーラウンジ



アフタヌーンティー！

コメント 0　いいね！ 0

あーちゃん@真…
5分前

とも
6分前

Elements Resources Network Sources Timeline Profiles Audits Console

Profiles

Select profiling type

Collect JavaScript CPU Profile

CPU profiles show where the execution time is spent in your page's JavaScript functions.

Collect CSS Selector Profile

CSS selector profiles show how long the selector matching has taken in total and how many times a certain selector has matched DOM elements. The results are approximate due to matching algorithm optimizations.

Take Heap Snapshot

Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.

Record Heap Allocations

Record JavaScript object allocations over time. Use this profile type to isolate memory leaks.

Start

1 98

Elements Resources Network Sources Timeline Profiles Audits Console

Profiles

HEAP TIMELINES

Snapshot 1
5.6 MB

5.00 s 10.00 s 15.00 s 20.00 s 25.00 s

100 KB

Class filter

Constructor	Distance	Shallow Size	Retained Size
▶(compiled code)	0	244 39%	3 072 828 52%
▶(closure)	0	6%	2 195 784 37%
▶system / Context	3	3%	1 791 584 30%
▶(array)		%	1 645 820 28%
▶Object			1 004 596 17%
▶(system)			915 964 16%
▶Array			417 820 7%
▶(string)			393 112 7%
▶t.hasOwnProperty.r			326 172 6%
▶Window / http://www.g...		%	313 468 5%
▶HTMLImageElement		%	235 440 4%
▶T	6	0%	143 504 2%

ブルー メモリ割り当て中

グレー 割り当て解放済み

Object's retaining tree

Object	Shallow Size	Retained Size

Summary All objects ? Selected size: 5.6 MB 98

Conclusion

まとめ

紹介した各要素について大まかに振り返り

Webパフォーマンスを司る3要素

Network

Compute

Render

- 継続的に計測すべし
- 基本の対策を徹底する

Network

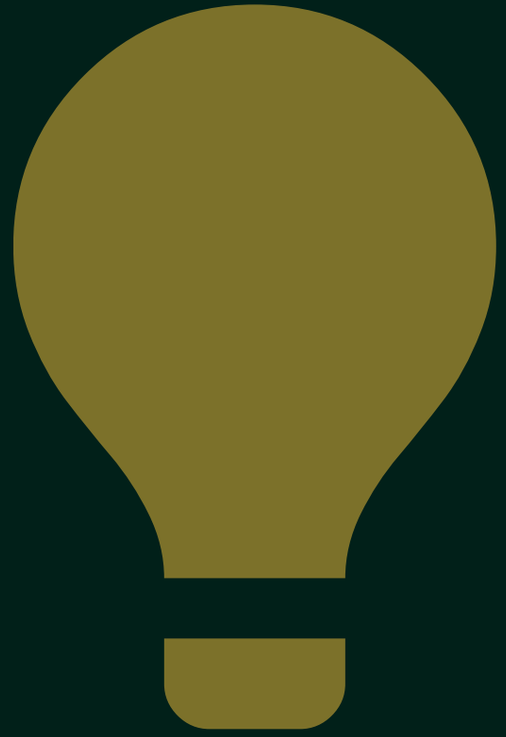
Render

- CSSの過装飾に注意
- レイアウトの発生も注意
- GPU処理は慎重に扱う

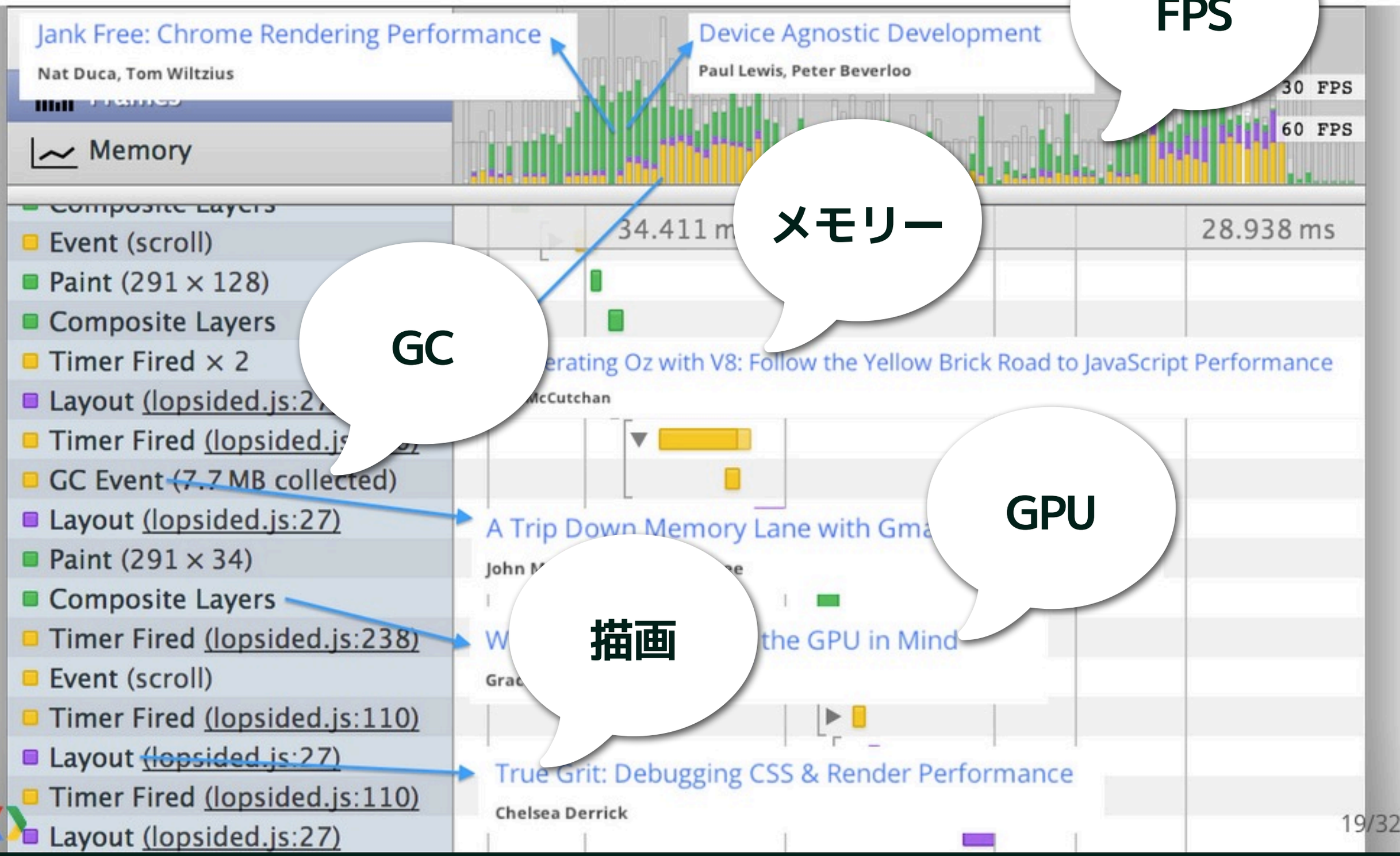
Compute

- 気にしすぎなくてOK
- ツールを使えば見て取れる
- ゲームとかはシビアな問題

Timelineは神ツール
色々な問題を発見できる



Dig deeper into perf tooling at #io13



dl.dropboxusercontent.com/u/39519/talks/devtools2013/index.html

- Web技術の転用が広がる
- 実行環境の多様化
- 制限とパフォーマンス

Env

Web技術で戦い続けるなら
すぐに取り組むべき！

Questions?

http://aho.mu

@ahomu

github.com/ahomu

Photo Credits♡

1. http://www.flickr.com/photos/tbisaacs/2407175368/
2. http://www.flickr.com/photos/stevendepolo/4294686346/
3. http://www.flickr.com/photos/epsos/8122951216/
4. http://www.flickr.com/photos/johnniewalker/8740192710/
5. http://www.flickr.com/photos/alesk/345519308/
6. http://www.flickr.com/photos/9301165@N08/2528387081/
7. http://www.flickr.com/photos/lachlanhardy/5174018127/
8. http://www.flickr.com/photos/msvg/5544223319/
9. http://www.flickr.com/photos/us-mission/6934693637/
10. http://www.flickr.com/photos/jdhancock/6151250051/
11. http://www.flickr.com/photos/thskyt/4929745746/
12. http://www.flickr.com/photos/hansel5569/7687221498/