karmaを使ったSPA向け
E2Eテスト技法

chatwork

自己紹介

- @kyo_ago
- Chatworkの中の人
- 趣味はLocal Proxy製作

- Under30じゃないエンジニア
- 推しAPIはApplication Cache

まずは謝罪

今回通りがいいと思って「E2Eテスト」って言ってますが
Integrationテストって言う方が正確だったりします。

今回伝えたいこと

SPAのテストに

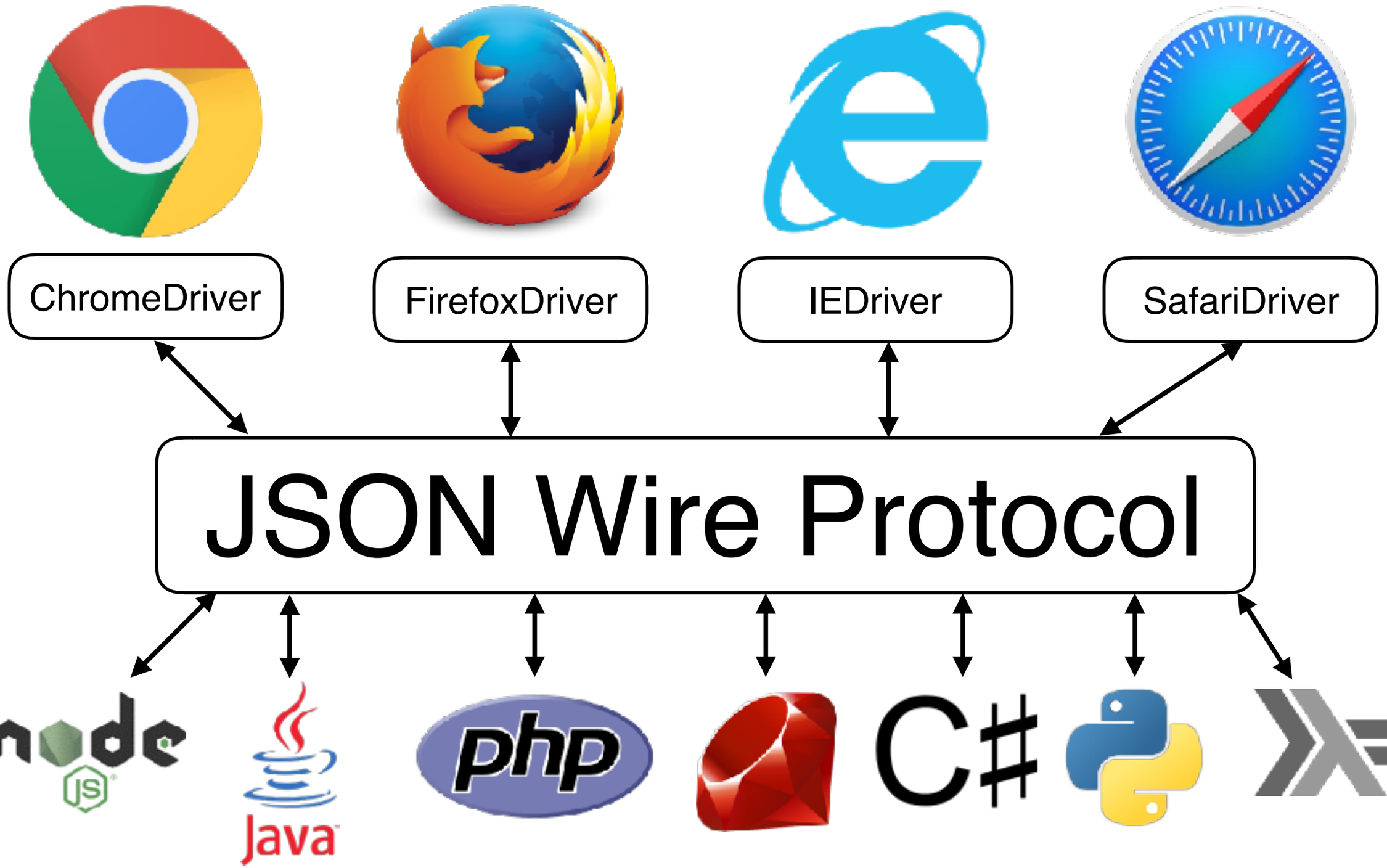
WebDriverは向かない

WebDriverとは

Seleniumから発展したツール

ブラウザごとのドライバと

そのドライバと通信するAPI


ChromeDriver
FirefoxDriver
IEDriver
SafariDriver
JSON Wire Protocol
node
JS
Java
php
C#

APIに対応したクライアントを使うことで汎用的なテストができる

2004年開発開始

歴史あるE2Eテストの鉄板ツール

ではなぜ「SPAのテストに
WebDriverは向かない」のか

WebDriverはあくまでも画面を遷移していく「Webサイト」のためのテストツール

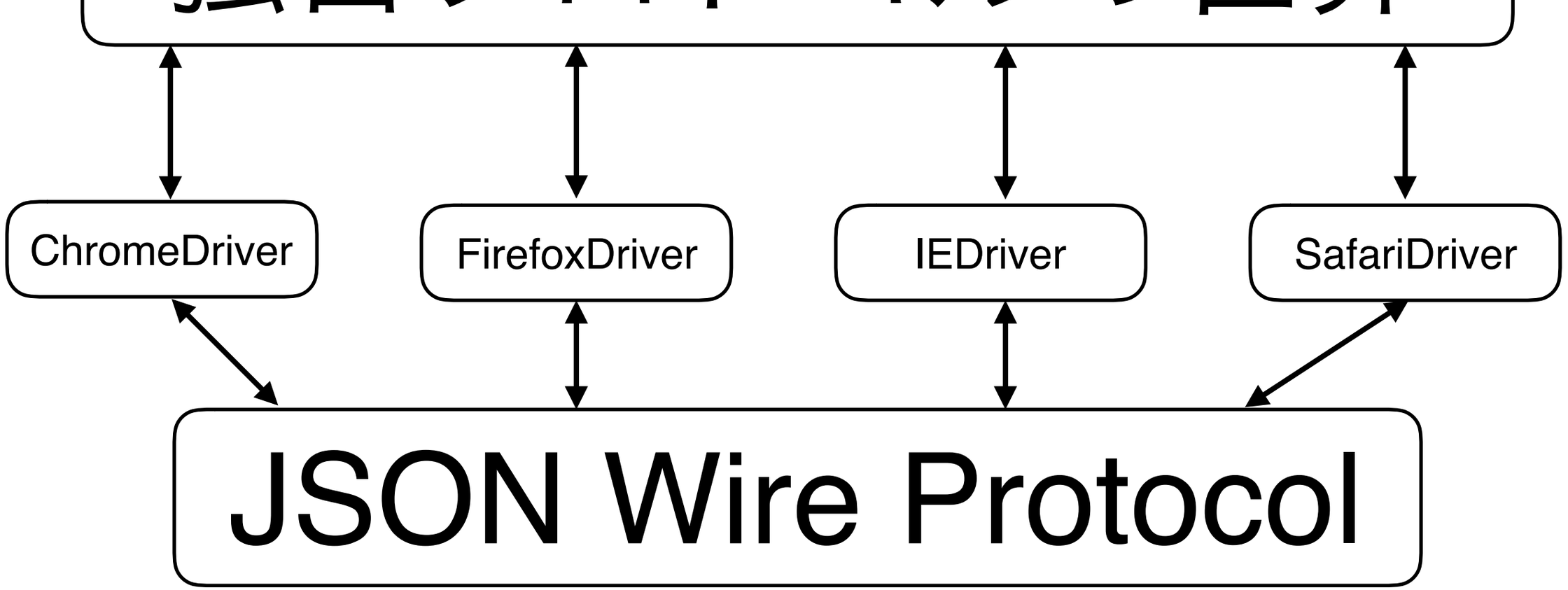
SPA向けのテストツールではない

SPAをWebDriverでテストするのは難しい

WebDriverは通常の開発ツールとは違う概念で動く




独自プロトコルの世界


ChromeDriver
FirefoxDriver
IEDriver
SafariDriver
JSON Wire Protocol

通常の開発フローとの差異が大きい

例えば

Node contextと

Browser contextの分断

Electronが解決したかったやつ

```
let webdriver = require('selenium-webdriver');

let driver = new webdriver.Builder()
    .forBrowser('chrome')
    .build();

driver.executeScript('return 2').then((res) => {
    console.log('returned ', res);
});

driver.get('https://www.google.co.jp/').then(() => {
    driver.quit();
});
```

ここだけBrowser context

Driverによっては

Developper Toolsと競合

console.logの内容がterminal でしか確認できない (Object展開とか出来ない)

```
$ node index.js
thenableWebDriverProxy {
  flow_:
   ControlFlow {
     propagateUnhandledRejections_: true,
     activeQueue_:
      TaskQueue {
       name_: 'TaskQueue::3',
       flow_: [Circular],
       tasks_: [Object],
       interrupts_: null,
       pending_: null,
       subQ_: null,
```

これが辛い！

画面を遷移していく「Webサイト」であればそれほど問題ない

SPAの場合、これらのツールが使えないのは非常に難しい

また、WebDriverはページの読み込みやクリック後の遷移を自動で待つ機能がある

```
var webdriver = require('selenium-webdriver'),
    By = webdriver.By,
    until = webdriver.until;

var driver = new webdriver.Builder()
    .forBrowser('firefox')
    .build();
driver.get('http://www.google.com/');
driver.findElement(By.name('q')).sendKeys('webdriver');
driver.findElement(By.name('btnG')).click();
driver.wait(until.titleIs('webdriver - Google Search'),
1000);
driver.quit();
```

待ってくれる

が、SPAでは通常これらの機能は使用しない

そこで

SPAのIntegrationテストには

Karma

Karma

・・・しってます？

社内で聞いてみた

「karmaの話するんですよ」

「あれ、AngularJS専用ですよね？」

「まだあったんですか？」

「Protractorになったと思ってました」

もともとAngularJSのテスト用に開発されたテストツール

現在は普通に汎用的なツール

（と言うかむしろAngularとの相性良くないような。。。）

まだまだ開発は活発

（5日前にver 1.5.0公開）

Protractorとは違う

（ProtractorはWebDriver系）

ざっくりいうと

ブラウザ上で動く

Unitテストランナー

これがブラウザ上で動く

```
describe('Assert', () => {
    it('should true', () => {
        expect(true).toEqual(true);
    });
});
```

今回はVer 1.0.0で追加された機能を使ってIntegrationテストの仕方を紹介

customContextFile

テストを指定されたhtml上で行う機能

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <script src="context.js"></script>
    <script type="text/javascript">
        %CLIENT_CONFIG%
        window.__karma__.setupContext(window);
        %MAPPINGS%
    </script>
    %SCRIPTS%
    <title>Inside Frontend</title>
</head>
<body>
<script type="text/javascript">
    window.__karma__.loaded();
</script>
</body>
</html>
```

ここにkarma.conf.jsのFileが展開される

ここから実行を開始

まるでサイト上でUnit testを走らせているかのように書ける

```
describe('Document title', () => {
    it('should be', () => {
        expect(document.title).toEqual('Inside Frontend');
    });
});
```

らくい！

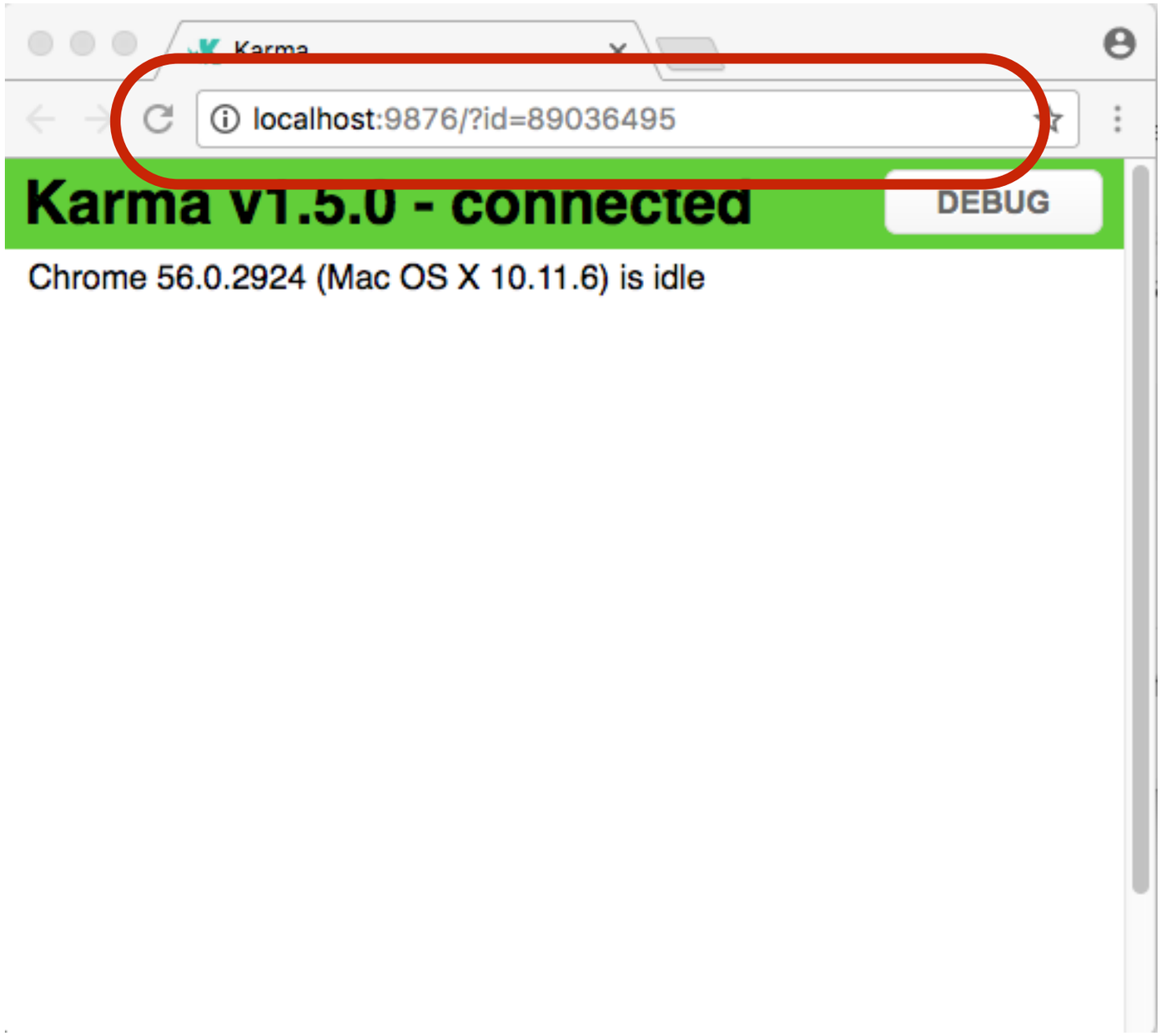
しかし。。。

customContextFileにも物足りなさはある

物足りなさ

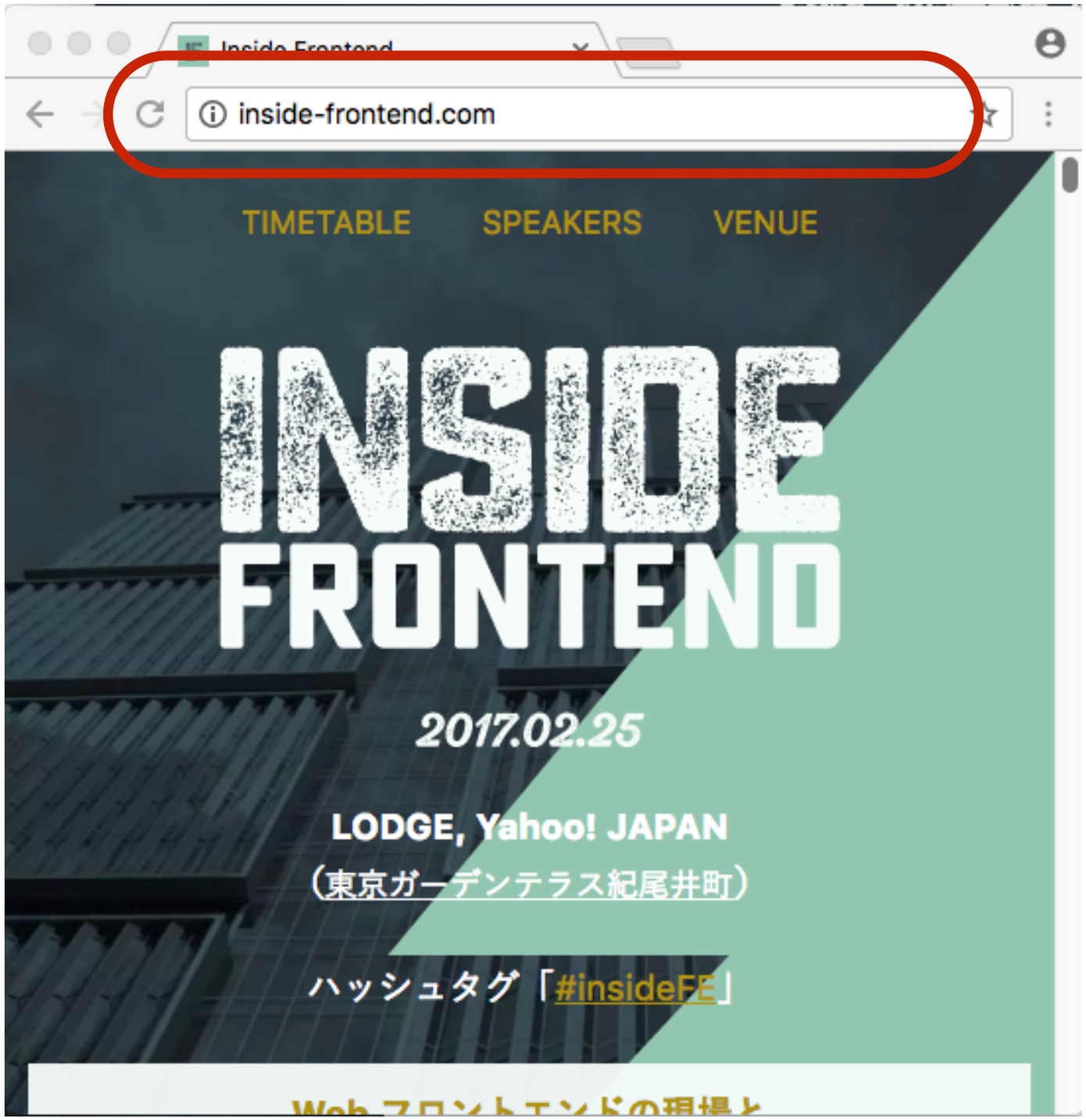
読み込み場所がlocalhost

物足りなさ


Karma
localhost:9876/?id=89036495
Karma v1.5.0 - connected
DEBUG
Chrome 56.0.2924 (Mac OS X 10.11.6) is idle

実際テストしたいドメインで読み込まれるわけではない

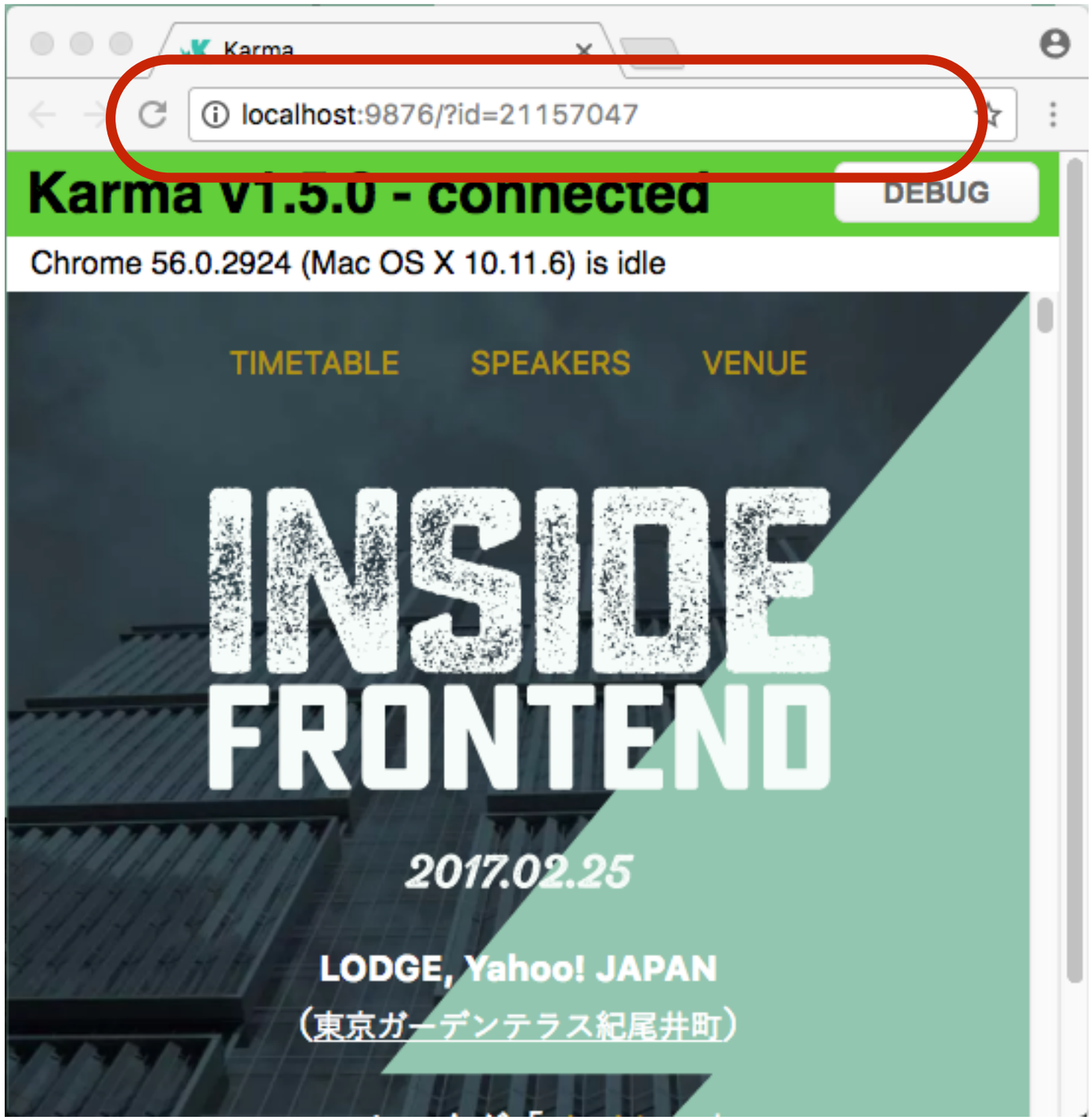
こうしたい


Inside Frontend
inside-frontend.com
TIMETABLE
SPEAKERS
VENUE
INSIDE
FRONTEND
2017.02.25
LODGE, Yahoo! JAPAN
（東京ガーデンテラス紀尾井町）
ハッシュタグ「#insideFE」

調べてみた

。。。ちょっと無理っぽい

こうする


Karma
localhost:9876/?id=21157047
Karma v1.5.0 - connected
DEBUG
Chrome 56.0.2924 (Mac OS X 10.11.6) is idle
TIMETABLE
SPEAKERS
VENUE
INSIDE
FRONTEND
2017.02.25
LODGE, Yahoo! JAPAN
（東京ガーデンテラス紀尾井町）

問題

別ドメイン通信

ブラウザをだます

手順

1. 通信の書き換え

Local proxyを使う

```
$ npm install --save-dev karma-proxy-plugin
```

karma.conf.js

SSL Proxy時の証明書エラーを無視

テスト実行ごとに環境をクリアしない

```
customContextFile: "custom-context-file.html",
proxyValidateSSL: false,
client: { clearContext: false },
middleware: ['proxy'],
proxy:
  '/inside-frontend/': {
      'target': 'http://inside-frontend.com/',
      'changeOrigin': true
  }
},
proxyReq: (proxyReq, req, res, options) => {
},
proxyRes: (proxyRes) => {
},
```

karma-proxy-plugin使用

karma標準のproxiesでないことに注意

送信イベント

受信イベント

/inside-frontend/への通信を http://inside-frontend.com/ へ転送

2. 通信の向き先の変更

htmlやJS内の通信の向き先を/inside-frontend/に向けたい

1ではProxyプロセスが立ち上がっただけで、

ブラウザのProxy設定を行ったわけではない

custom-context-file.html

```
let base = document.createElement('base');
base.href = '/example/';
document.head.insertAdjacentElement('afterbegin', base);
```

custom-context-file.html

```
XMLHttpRequest = () => {/* … */};
fetch = () => {/* … */};
```

簡単

他にいろいろしたい場合

Refererを書き換えたい

karma.conf.js

```
proxyReq: (proxyReq, req, res, options) => {
    proxyReq.setHeader('Referer', 'https://example.com/');
},
```

Cookieを書き換えたい

karma.conf.js

```
proxyRes: (proxyRes) => {
    if (proxyRes.headers['set-cookie']) {
        proxyRes.headers['set-cookie'] =
proxyRes.headers['set-cookie'].map((cookie) => {
            // karma内はhttpなのでsecureを消す
            return cookie.replace(/\s*secure;?/i, '');
        })
    }
},
```

リダイレクト先を変更したい

karma.conf.js

```
proxyRes: (proxyRes) => {
    if (proxyRes.headers['location']) {
        // リダイレクト先を変更
        proxyRes.headers['location'] =
proxyRes.headers['location'].replace(/^https:\/\/
example.com/i, '/example/');
    }
},
```

ログインしたい

custom-context-file.html

```
fetch('/example/login', {
    method: 'POST',
    body: new FormData(document.getElementById('#login'))
}).then((response) => response.text()).then((text) => {
    // レスポンスにTokenがあれば変数に保存
    // Cookieは保持される
    window.__karma__.loaded();
});
```

こういった方法を使うことで別ドメインで読み込まれる問題を解消

proxyReq、proxyResの引数は以下参照

https://github.com/nodejitsu/node-http-proxy

ここまで来るとテストを書くのは非常に簡単

普通にサイト上のDevTools上で検証するのと変わらないレベルでテストが書ける

```
const assert = require("power-assert");

describe('index page', () => {
    it('Google map', () => {
        assert(google.maps);
    });
    it('copyright', () => {
        assert(document.querySelector('.copyright'));
    });
});
```

非常におすすめ

実際やってみてどうだったか

テストは非常に書きやすい

ただ、ブラウザに起因する不安定さはどうしようもなかった

DOMという広域変数やばい

状態のクリアが自分の責任になるので辛い
(楽になる部分もあるけど)

自分で制御できる範囲が広がるのは嬉しい

(WebDriverのブラックボックス感がない)

そもそも

WebDriverは受け入れテストのツールであって、開発者向けツールではない

開発しながらテストするのであれば別の方法を検討する方がいい

ただ、受け入れテストなら WebDriverのほうが良いかもしれない

WebDriverは高性能だけど、不要な機能も多い

最後に

E2Eテストは

書かないに越したことはない

ご静聴ありがとうございました

ご質問は？

ご質問は？

例

- 実行時間は？
- 実際どういうふうに動かしてる？
- 他のメンバーの反応は？
- やってよかった？
- どういうときにおすすめ？
- CIとの連携は？