SDCC2016

中国软件开发者大会

SOFTWARE DEVELOPER CONFERENCE CHINA

不断重置的前端人生

嘉宾：蒋定宇

@josephj6802

台湾郎

前年四月之前都在台湾做前端

2005.10 YAHOO! 奇摩

2009.09 miiCasa D-Link

2010 WebRebuild

2011 SDCC

2013.06 Faria Systems

2014.02

- 曾来北京演讲过两次
- 建立前端团队、模块化策略

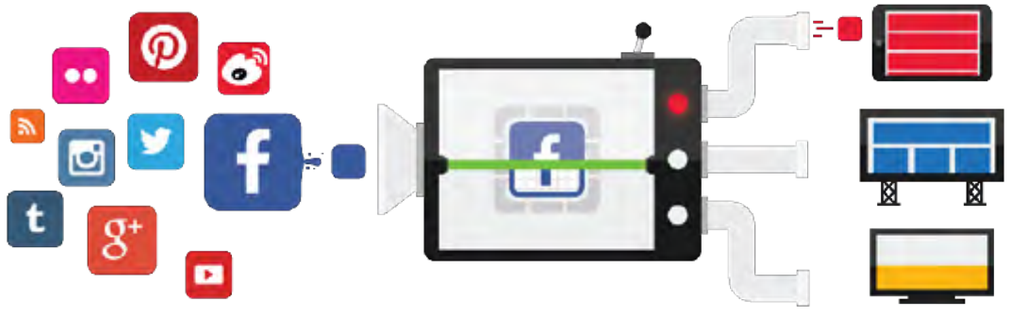
澳客进行式

現在在悉尼的 Stackla

2014.5

Now

Stackla



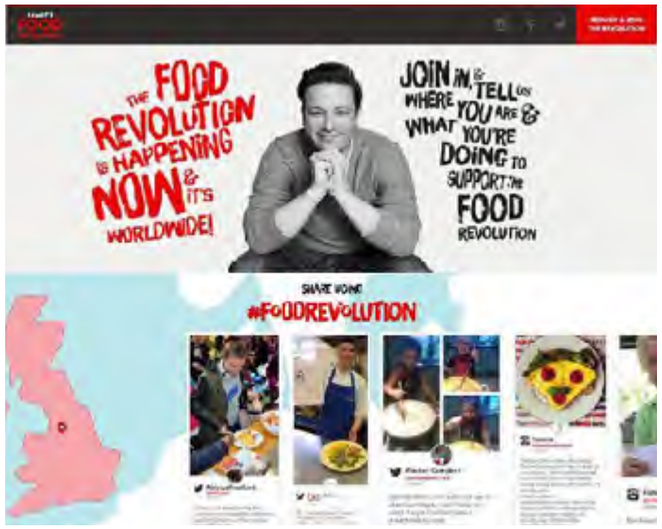
撷取　　过滤　　投射

- 社交媒体**撷取**器
- 撷取后可**管理过滤**资料
- 可用三种方式**投射**
- 多用于**行销**活动、**广告**
- **成功**的 Startup

#不断重置的前端人生#

Stackla

杰米・奥利弗活动



2015 伦敦时装周

2015 悉尼跨年烟火

我自己的婚礼

About 关于重置 

“就跟 CSS Reset 一样，什么东西都得重头来过”

電腦與程式的重置

在电脑系统中，重置通常**以受控制的范围**、清除所有搁置的错误或事件、并且**将系统回复**成**正常**的状况或**初始**值。

无线路由器的重置

```
<style type="text/css">

html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h
a, abbr, acronym, address, big, cite, code, del, dfn, em, font, img,
small, strike, strong, sub, sup, tt, var, dl, dt, dd, ol, ul, li, fi
table, caption, tbody, tfoot, thead, tr, th, td {
    margin: 0;
    padding: 0;
    border: 0;
    outline: 0;
    font-weight: inherit;
    font-style: inherit;
    font-size: 100%;
    font-family: inherit;
    vertical-align: baseline;
}
```

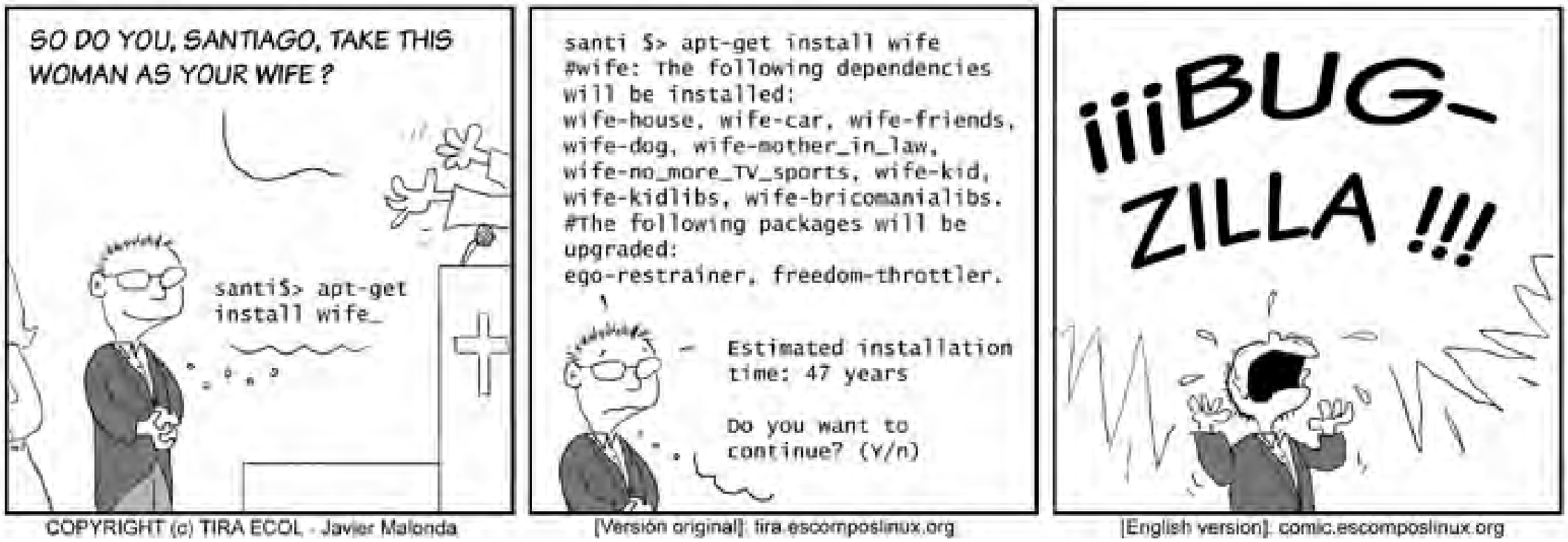
标签样式的重置

重置有**不方便**的地方，但通常可以**解决问题**或**带来好处**

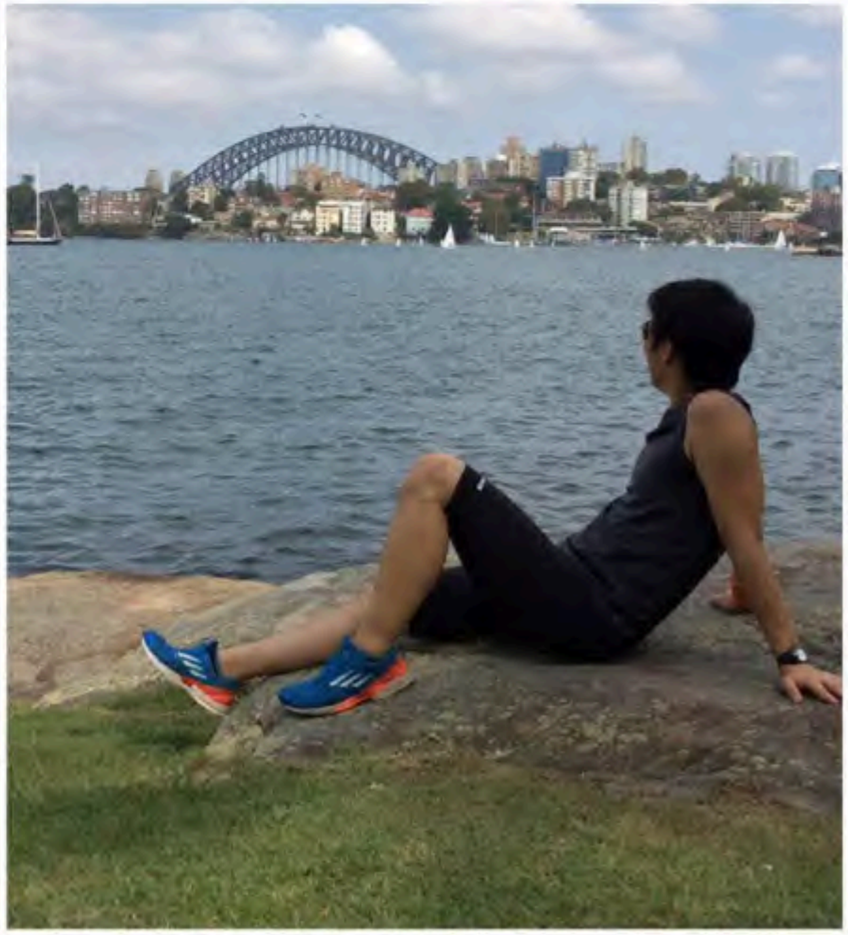
人生的重置

人生也是一样，我们也**不断地增加依赖**

日子久了会有各种奇怪、难解的问题



#1 生活重置：梦想 Life Reset



梦想

43Things

work abroad

到國外工作

想體驗國外的工作經驗、想可以 5:30 下班、想要有更多的收入、想知道黃金聖鬥士 (Google Engineer) 的實力到底有多深不可測。

Aug 03, 2006, 07:56AM PDT | 3 comments

为了梦想，得**重置**许多事情，真的太困难了

- **工作**：不想放弃高薪、好的职位
- **爱情**：在澳大利亚打工度假的女友即将回台
- **语言**：怀疑自己的英文能力
- **其他**：得抛弃房子、车子、宠物、家人、朋友

转变与冲击

不开心的遭遇

- 好处：**高薪**、**成功产品**、**才华洋溢的创办人**、**顶尖的同事**、**优秀的软硬体设备**
- 代价：**F1 方程式般的开发速度**、并满足像**乔布斯**般**要求完美**的创办人

http://www.tudou.com/programs/view/DPp-rRS5Cgo

在跟客户 Demo 时破图，这是完全没办法接受的

那个晚上思考了很久，觉得这样的生活不是我要的

才觉得**需做出改变**：去**寻找我的梦想**

一开始的乐观

在台湾意气风发

来悉尼乏人问津

https://www.linkedin.com/pulse/why-your-resume-landed-my-trash-stacey-alcorn

- **经验**：不相信非西方国家的经历，即使我曾在台湾雅虎
- **语言**：面试、电话常无法完全理解或有效回答对方的问题
- **签证**：没有可以工作的签证 = 没公司或猎头感兴趣
- **金钱**：三个月没收入，压力很大

离梦想越来越远

理想中来到外国的样子

实际上只能整天待在租屋

鲁蛇(屌丝)无误啊啊啊！

过渡期

不如意，但总是有方法

1. 开始申请**学生签**，至少能继续找
2. **兼差**案子：短暂的两个礼拜、至少有些收入
3. 著急也没用，**乾脆去玩**两个礼拜

从天而降的面试

回来后就收到面试邀约！

- **关系**：SocialStatus 的好朋友
- **技术**：Widget 核心技术我很有经验
- **时间**：新创公司发展到需要导入前端的阶段

转眼就 2 年半

回顾：人生重置

重置前所考量的困难点

- **工作**：不想放弃高薪、好的职位
- **爱情**：在澳大利亚打工度假的女友即将回台
- **语言**：怀疑自己的英文能力
- **其他**：得抛弃房子、车子、宠物、家人、朋友

重置后的实际结果

- **工作**：调薪、升迁、5:00 下班、住公司对面
- **爱情**：已经是老婆、不再是单身男子独自在外
- **语言**：英文能力绝对够用
- **其他**：公司越来越好、朋友越来越多、永久居留申请中

其实**转换跑道**、**创业**都可以算是人生的重置

伴随风险，但即使失败，也必定相当的**收获**

我期待未来继续重置，你呢？

Reset ***Success!***

人生重置成功！

#2 心态重置：产品

做事方法与态度的调整

不習慣

真正的新创公司

- **心态不同**
  - 著重于完成功能，較少程序员间的品质交流
    - 过去有：工程文件制作、Peer programming、Code review、重构、分享会
- **无前端架构**
  - 仍然用 <script/> tag，没有 RequireJS 模块依赖
  - 混乱的全域变数、方法
  - 有很多的复制贴上
  - 没有比较好的制度：例如物件导向
  - 对好东西 Grunt、LiveReload、RequireJS 没兴趣

调整心态还是**要求改变？**

调整心态

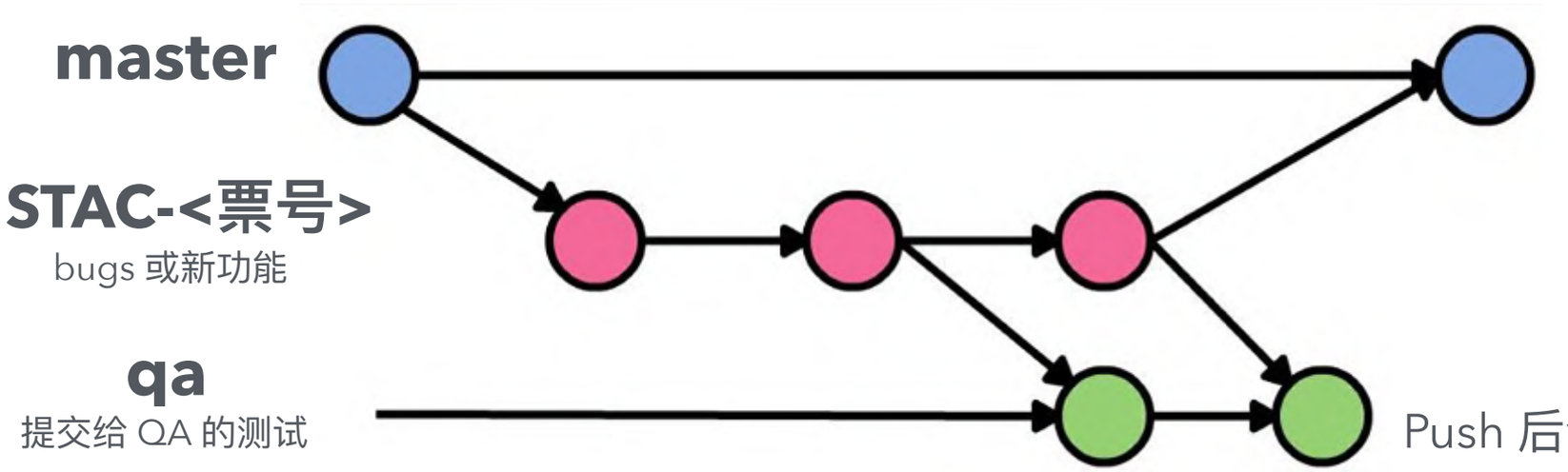
迅速改变代价高

- 缺乏程序员间的**品质交流**：先摆一边，但**尽可能写文档**
- 没有 RequireJS 模块依赖：先用 **grunt-usemin** 解决布署问题
- 混乱的全域变数、方法：先摆一边，未来不再使用
- 有很多的复制贴上：把 DOM 的部份改用 **Mustache**
- 没有比较好的制度：例如物件导向：**新功能再使用**
- 对好东西 Grunt、LiveReload、RequireJS 没兴趣
  - 导入 Grunt 开始处理 SASS 的编译问题
- **开始两个礼拜一次的内部分享会**

离理想很远，但却是大家能够接受的改变，也不会把自己累死

Git 流程

第一次不用 Git Flow



需要发 GitHub Pull Request
先布署到 **Staging** 环境
再到 **Production** 环境

所有人皆可 merge
Push 后会自动布署到 **QA** 测试环境

http://d.pr/y57H

所有**开发、修 Bug** 都在 **STAC-<票号>** 的 branch

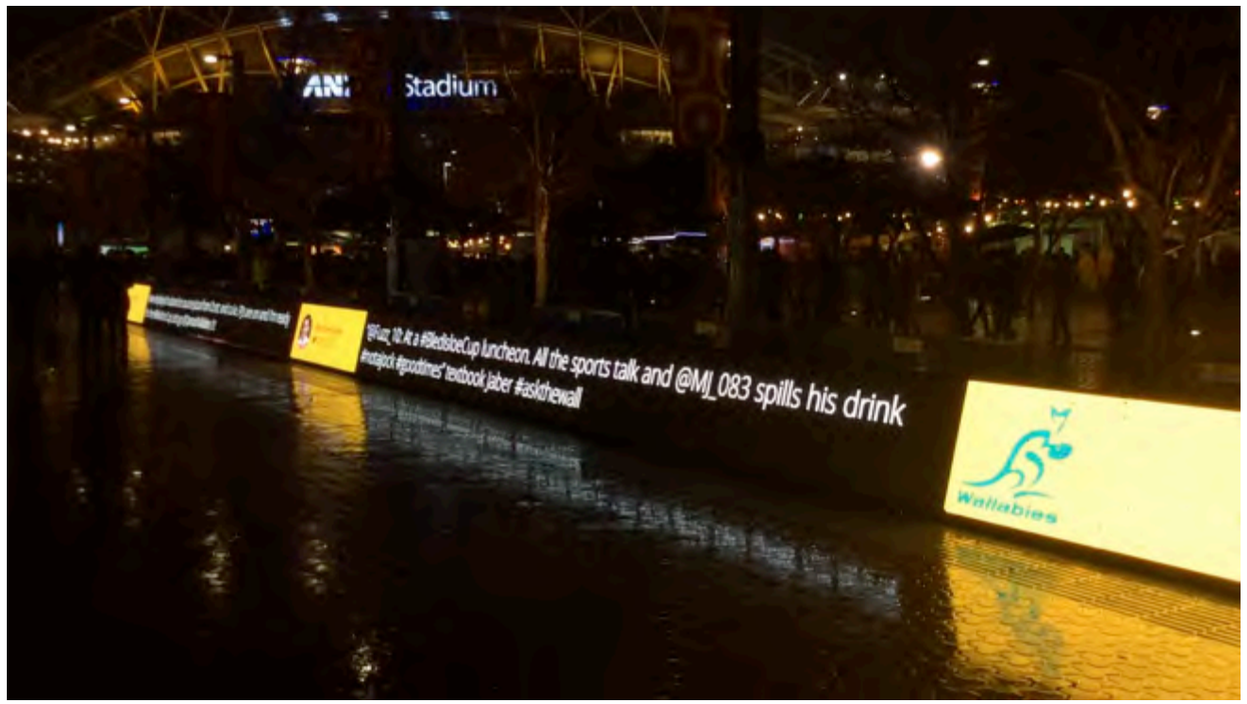
减少了很多不必要的手续、步骤、甚至架构

对小团队的我们其实够用了

V2

放弃既有实作

项目：新的 Event 架构



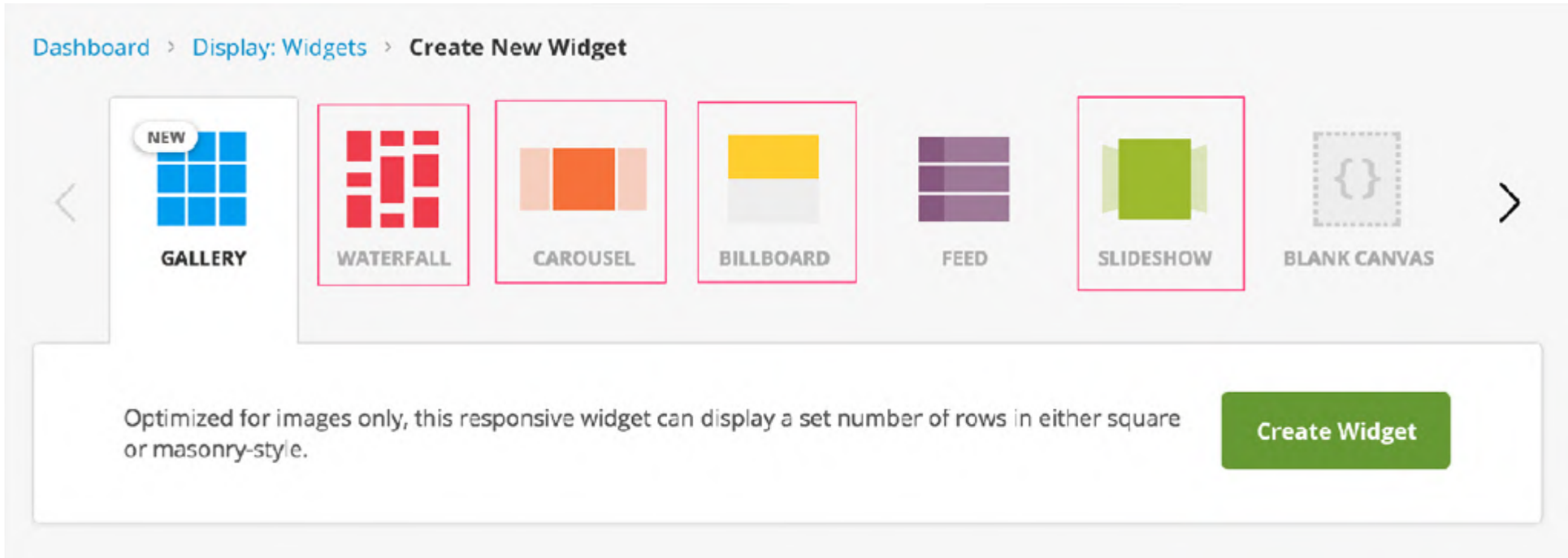
- **背景**：使用客户较少、时间较充裕
- **需求**：放弃既有实作，为了让客制化容易
- **技术**：OOP、Mustache、AlpacaJS

导入较好的作法后：客制化与新增不同 Type 都变简单许多

复制、贴上

这样做不对吧！？

项目：Widget 新样式



- **背景**：使用客户很多、功能复杂、时间不足
- **需求**：样式与部分行为变更、但不能影响客户既有的 Widget
- **技术**：无（复制、贴上、修改）

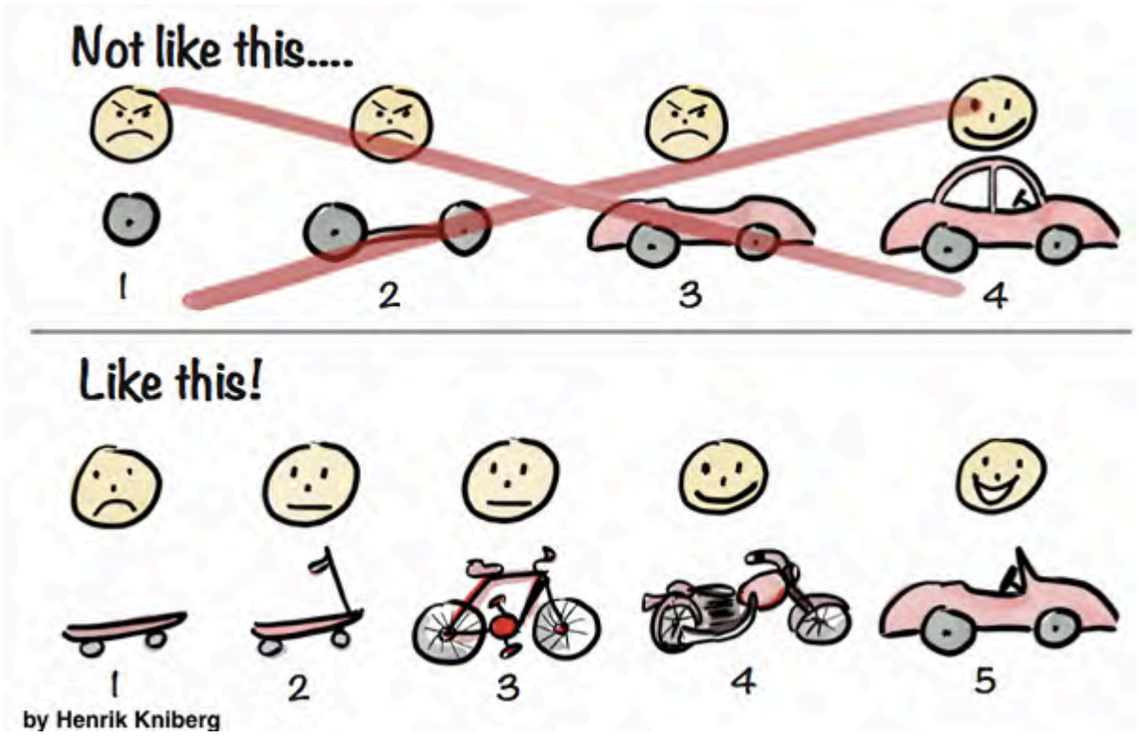
结果：在时程内完工、外观完胜竞争者

MVP

最简可行产品

仅实作「**够用**」的功能：以取得客户回馈供未来改进

而非一开始就「**要求完美**」



这是很多**产品与视觉设计**人的死穴

总是希望完美，但往往花了**数倍**的时间在**微调**上，但整体效益不高

知易行难，但我们的产品与视觉都有这样的弹性

砍生财工具

Professional Service

用内部开发者资源

帮客户做我们产品的客制化

例如：微网站、Widget、Event 的客制化

能够**快速收益**、让**客户满意**

但对内部开发者的成长、或平台本身成长都是负面的

去年大刀一砍，我们不用再分心做客制化服务

当经营者有这样的勇气，不得不佩服

心态重置：小结

1997 WWDC，一位听众质疑乔布斯砍掉 OpenDocs 的决定

“Working Backwards to Technology”

“需**从客户体验开始**，再**回头去找适合技术**
而不能先从科技开始、去想要卖什么”

不应被技术或方法牵著鼻子走

心态重置：小结

1997 WWDC，一位听众质疑乔布斯砍掉 OpenDocs 的决定

Stackla 的产品经验

让我从**技术导向**转为**解决问题导向**

让**产品**、**资源**来决定使用什么方法或技术

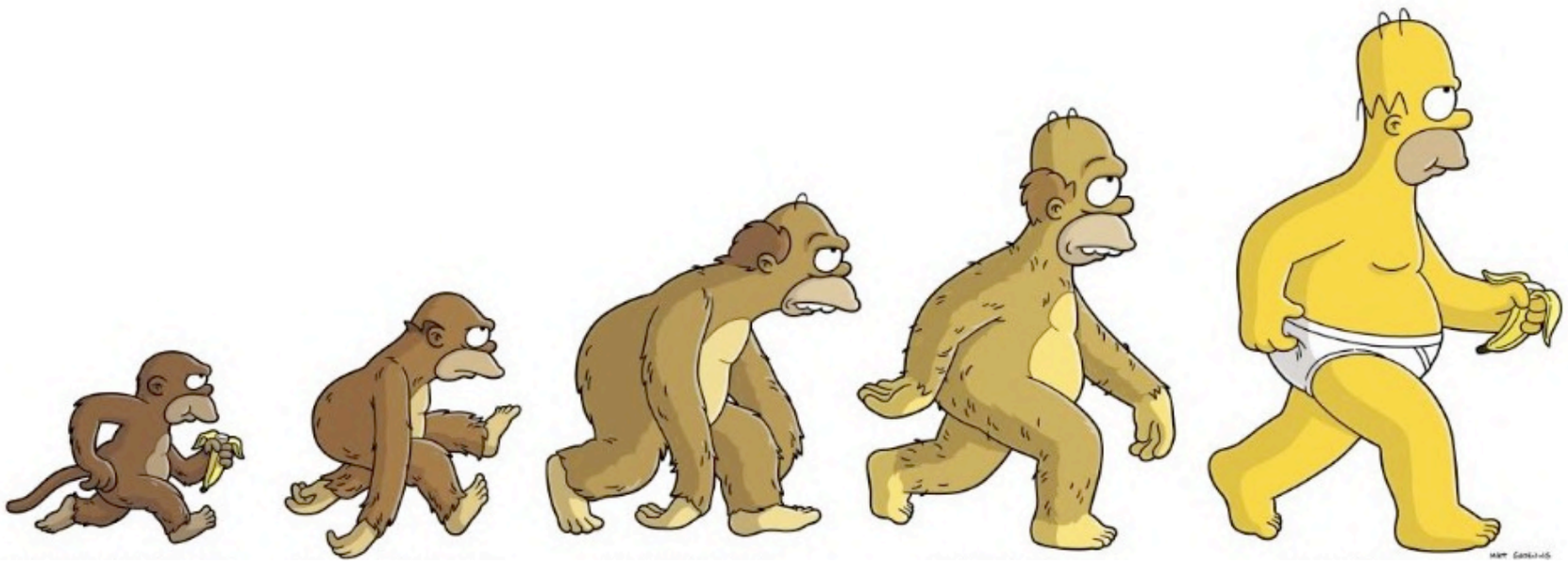
> “Working Backwards: Start with Technology”
>
> 而不能先从科技开始、去想要卖什么”

不应被技术或方法牵著鼻子走

心态重置改善！

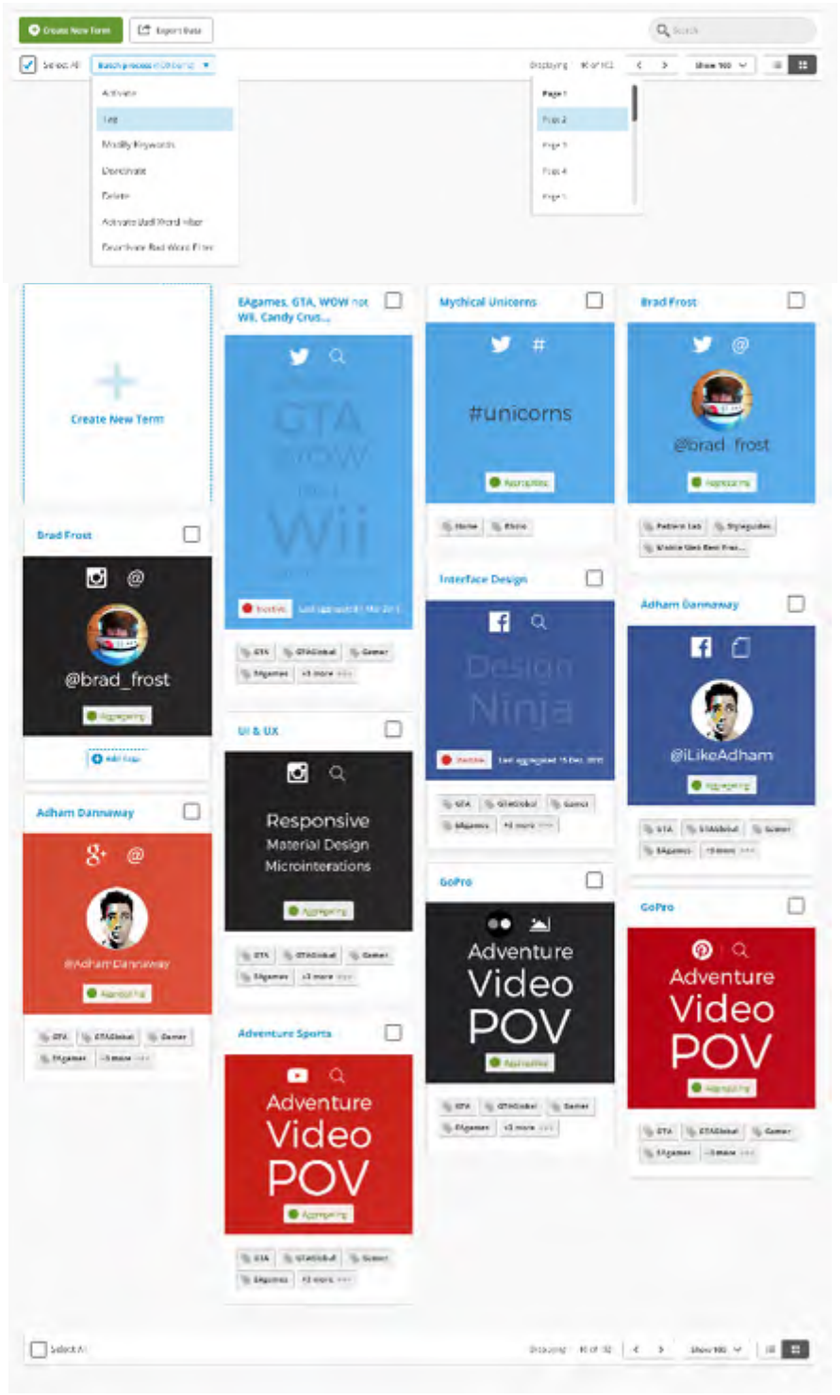
#3 团队重置：成长 *Team Reset*

技术改变、团队成长的喜悦与痛苦



技术抉择

UI 开始复杂化、一定得改

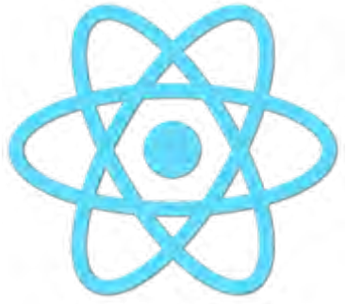


如何决定

学习**成本高**

完整框架**弹性小**

稳定、较多人使用



学习**成本低**

函式库**弹性高**

新、较少人使用

导入成功秘诀

全栈工程师们眼睛发亮

足够的教学

上手时间短

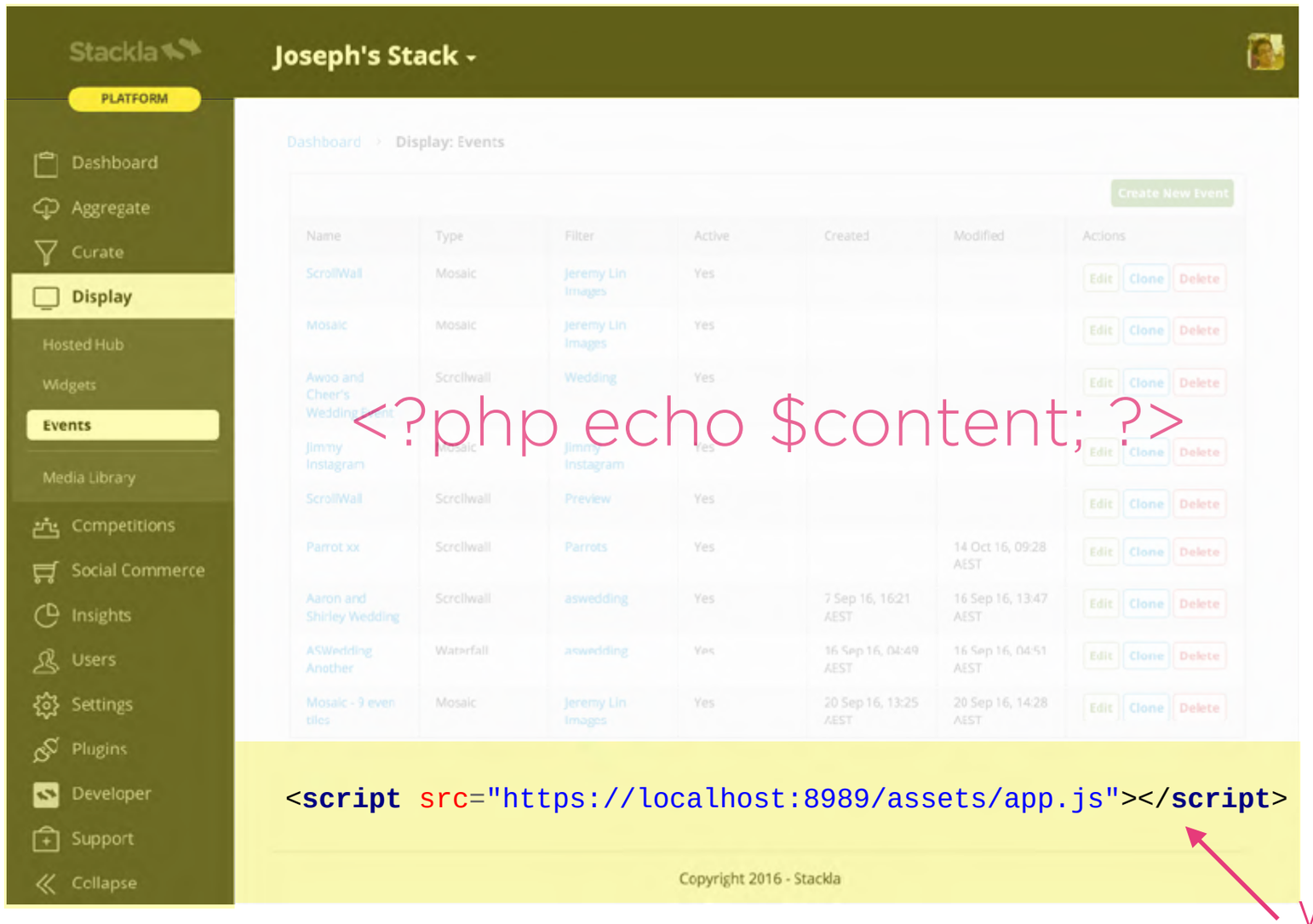
团队优先！

如何导入

SPA 的概念太耗时、不可行

维持现有架构，能开始用比较重要

PHP View



虽有 react-router，但仍是靠 PHP 路由整页刷新

大有益处

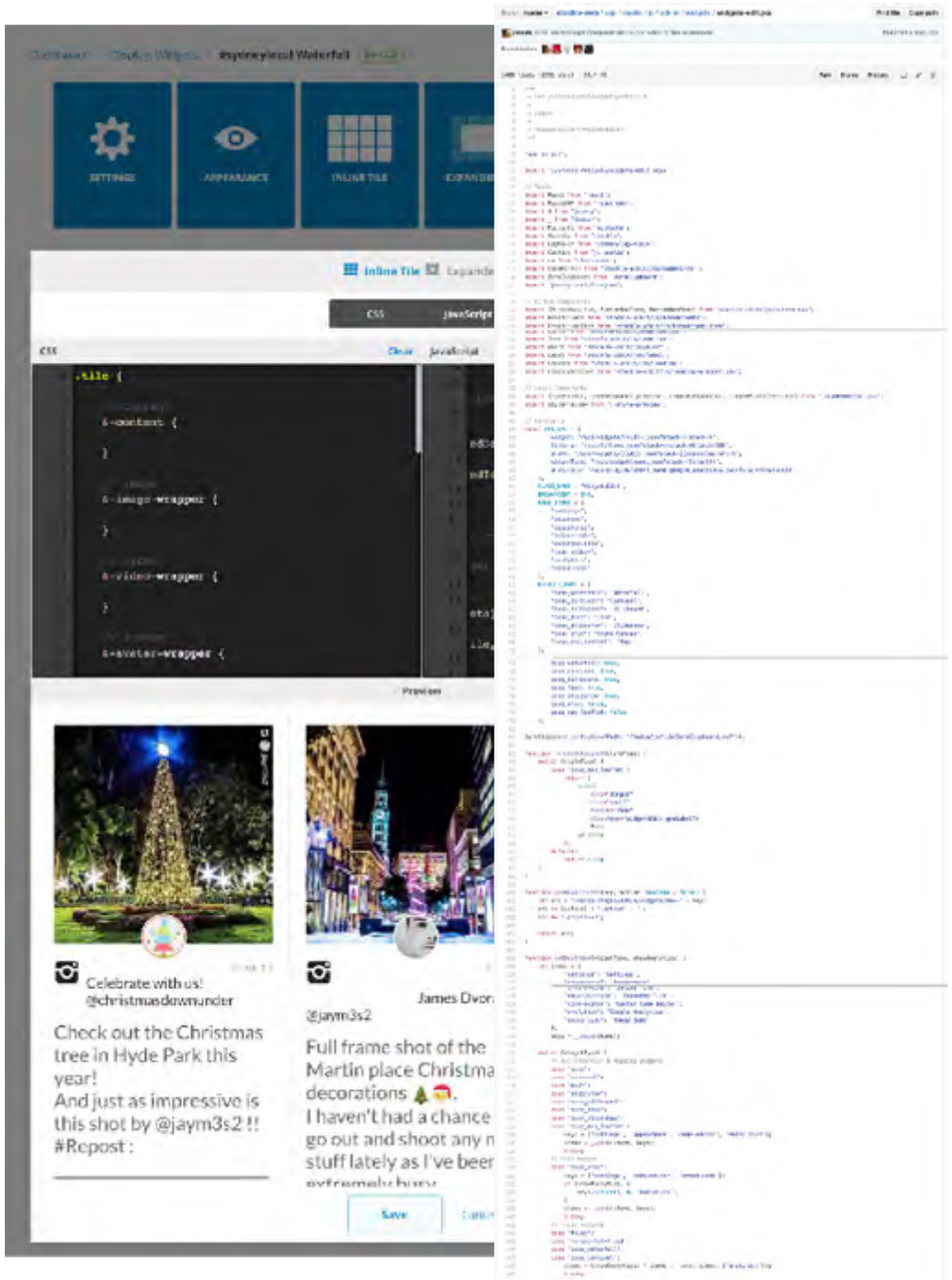
前端终于有适当的**开发模式**

- 终于有比较好的**套件管理** npm
- 终于有 JS/CSS 的**依赖**
- 终于能使用先进技术、**面向未来** BABEL
- 终于能用简单的方法制作 **UI**

提供基础的 **Webpack** 是**最重要**的工具

导入 Redux

一样也是碰到问题再来处理，**避免太早抽象化**



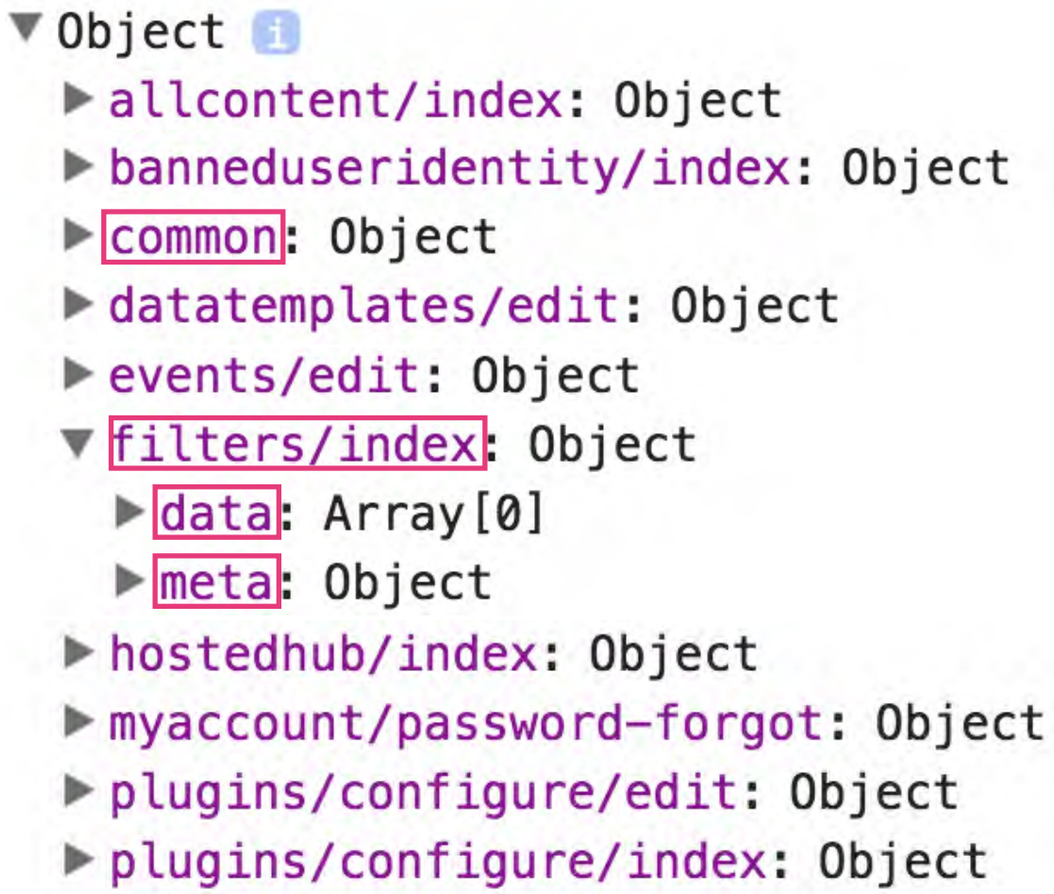
- **Delegation 困难：**
  - 由上而下要带太多 props
- **太多概念在同 JSX 中：**
  - View 逻辑
  - API 资料载入
  - 资料 Decoration
- **State 管理出问题：**
  - 无法在元件之外共用
  - 随意增加 State

一个档案超过 2000 行！

Store #1

Redux Store 的初期规划

目标：**搬移**原本 Container Component 的 setState



- 由**路径**决定命名空间
- **data** - API 取得的资料
- **meta** - UI State
- **common** - 共用组件
  - ex. Tag 选单
- 优点：**直觉**
- 缺点：**没共用资料**

Ducks

不做无谓的抽象化

ducks-modular-redux

“开发时有 95% 的机会
只需要一对 reducer/actions”

为什么要拆分？

```
// Actions
const LOAD   = 'my-app/widgets/LOAD';
const CREATE = 'my-app/widgets/CREATE';
const UPDATE = 'my-app/widgets/UPDATE';
const REMOVE = 'my-app/widgets/REMOVE';

// Action Creators
export function loadWidgets() {
  return { type: LOAD };
}

export function createWidget(widget) {
  return { type: CREATE, widget };
}

export function updateWidget(widget) {
  return { type: UPDATE, widget };
}

export function removeWidget(widget) {
  return { type: REMOVE, widget };
}

// Reducer
export default function reducer(state = {}, action = {}) {
  switch (action.type) {
    // do reducer stuff
    default: return state;
  }
}
```

高手加入

在台湾每个公司都想要的前端高手

Jonathan

Art Pai

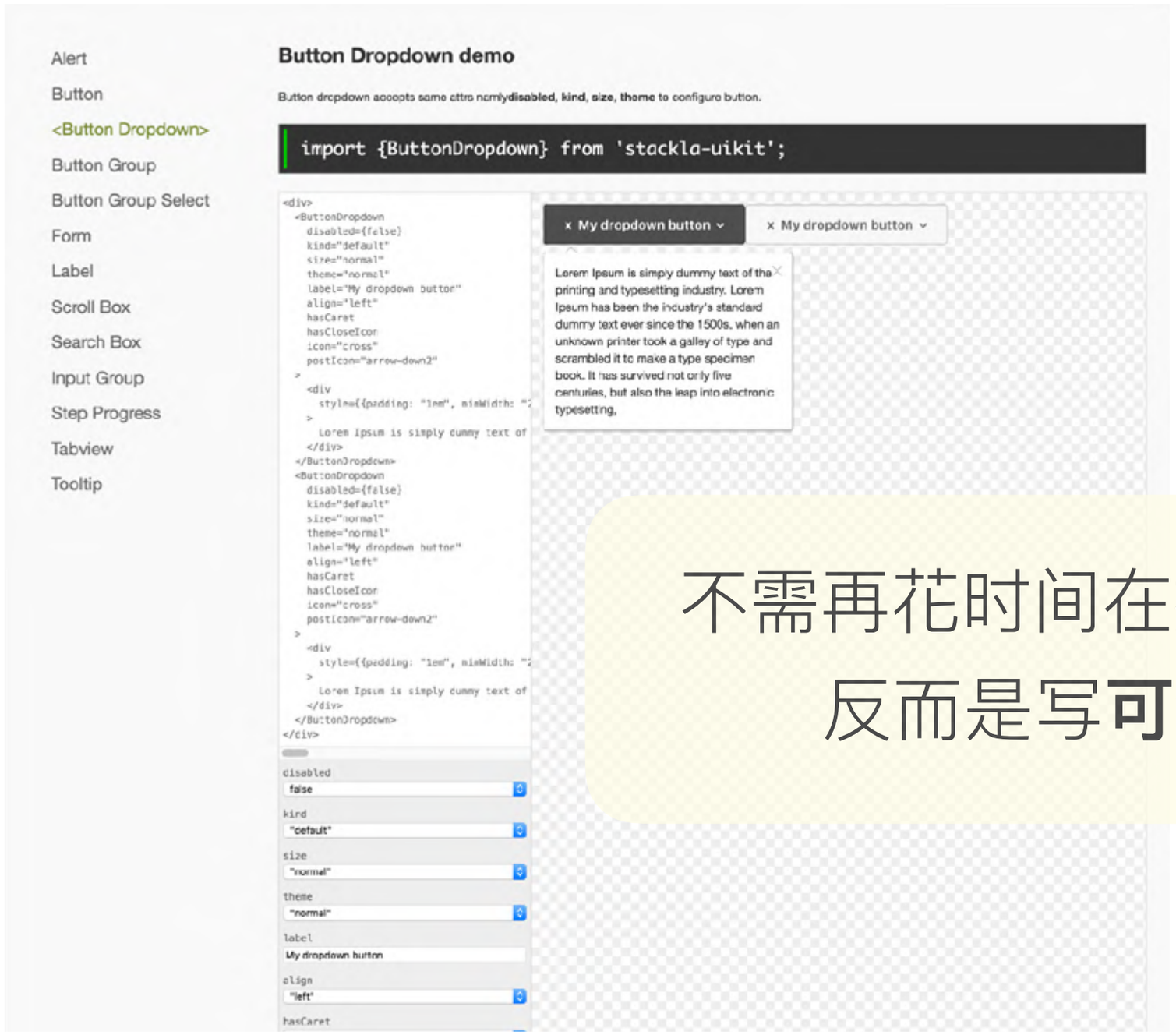
刚好因为他们想出国工作

把他们**抢来当同事**！

很幸福：不斷能學新東西、前端也一直再改进

提升 UI 组件

react-demo



不需再花时间在撰写 UI 说明文档上

反而是写**可测试的 Demo**!

RxJS

RxJS + redux-observable

RxJS **对异步处理极为优异、组合性高**

唯一缺点是**起步学习曲线陡峭**

克服之后回不去 Promise 了！

Redux 简化

redux-actions

过去 - Redux 代码有重复性高

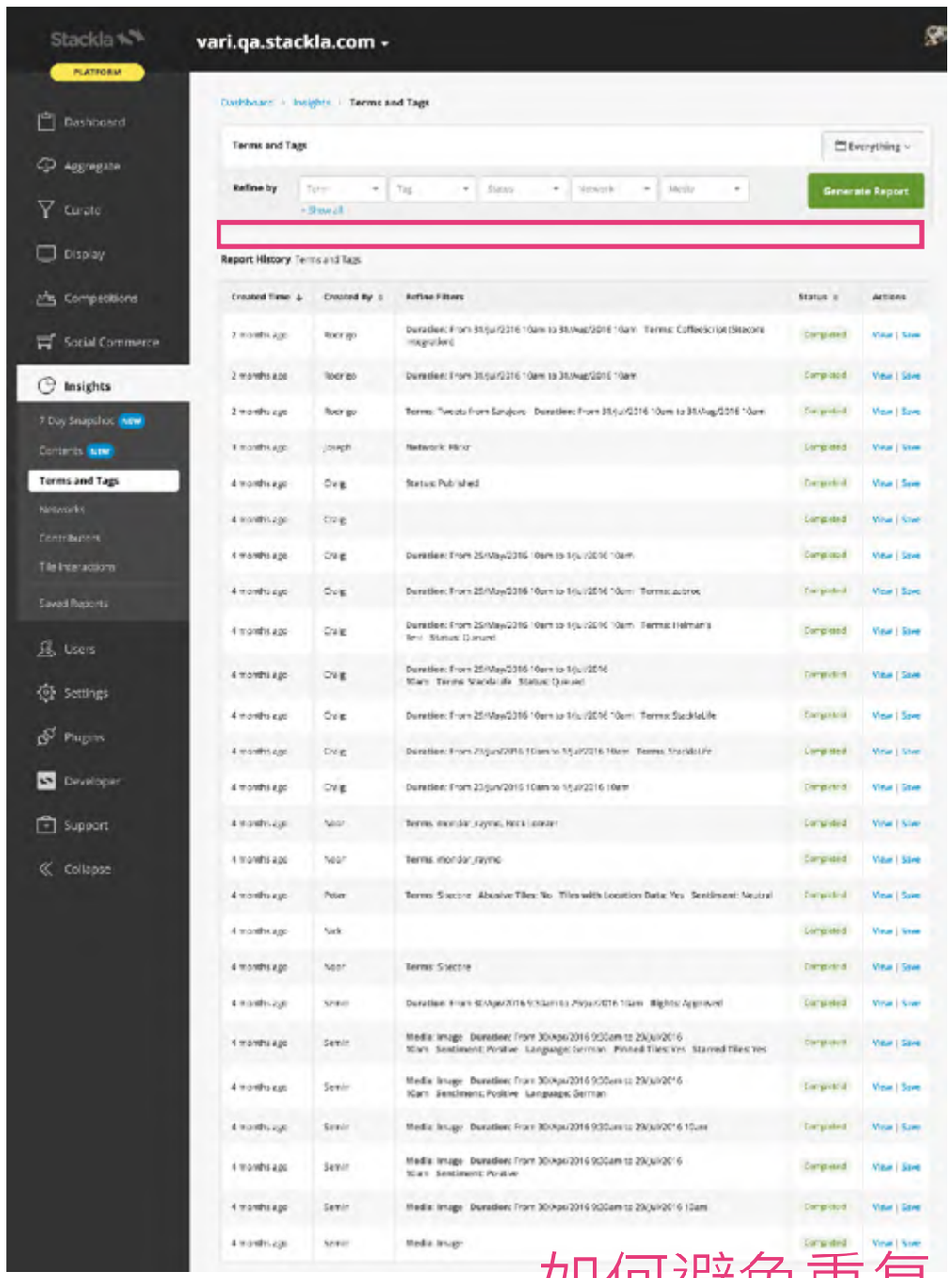
```
// Actions
const LOAD   = 'my-app/widgets/LOAD';
const CREATE = 'my-app/widgets/CREATE';
const UPDATE = 'my-app/widgets/UPDATE';
const REMOVE = 'my-app/widgets/REMOVE';

// Action Creators
export function loadWidgets() {
  return { type: LOAD };
}

export function createWidget(widget) {
  return { type: CREATE, widget };
}

export function updateWidget(widget) {
  return { type: UPDATE, widget };
}

export function removeWidget(widget) {
  return { type: REMOVE, widget };
}

// Reducer
export default function reducer(state = {}, action = {}) {
  switch (action.type) {
    // do reducer stuff
    default: return state;
  }
}
```

现在 - 藉由工具减少很多

```
import {createAction} from 'redux-actions';

const PREFIX = 'my-app/widgets';

// Action Creators
export const loadWidgets = createAction(`${PREFIX}/LOAD`)
export const createWidget = createAction(`${PREFIX}/CREATE`);
export const updateWidget = createAction(`${PREFIX}/UPDATE`);
export const removeWidget= createAction(`${PREFIX}/REMOVE`);

// Reducer
export default const reducer = handleActions({
    [loadWidgets]: (state) => {/* do load widget */}
});
```

避免重复

类似的页面、资料来源一致、但各有自己 UI State



如何避免重复、又让客制化最大化？

方法

把共用的 Ducks 用 Function 包起来

report/common/redux.js

```
export default function(PREFIX) {
    return {
        // Action Creators
        changeFilters: createAction(`${PREFIX}/CHANGE_FILTERS`),
        resetFilters: createAction(`${PREFIX}/RESET_FILTERS`),
        saveReport: createAction(`${PREFIX}/SAVE_REPORT`),
        // Reducer
        reducer: combineReducers({
            reports: handleActions({
                [saveReport]: () => {},
            }),
            options: combineReducers({
                filters: handleActions({
                    [changeFilters]: () => {},
                    [resetFilters]: () => {},
                })
            })
        })
    }
};
```

report/user/redux.js

```
import commonRedux from './common/redux';
const PREFIX = 'reports/user';
const {
    changeFilters,
    resetFilters,
    savedReport,
    reducer,
} = commonRedux(PREFIX);


export default combineReducers({
    common: reducer,
    visibleResultsCount: handleAction()
});
```

好奇大家避免 Redux 重复的方法为何呢？

产生的 Redux Store

```
▼ Object
  ▶ reports/network: Object
  ▶ reports/aggregated: Object
  ▶ reports/tile: Object
  ▼ reports/user: Object
    ▶ common: Object
      visibleResultsCount: 3
```

代碼猴子時間

少用脑的重构

目标简单

全员出动以短时间完成

任务分派明确、互相检视

短时间内让团队学习、建立一致性

#1 ESLint

代码的基本品质工具

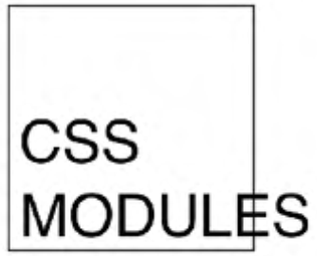
问题：早就在用，但警告太多，没人理会

```
[12:28:09] 'source:lint' errored after 102 ms
[12:28:09] ESLintError in plugin 'gulp-eslint'
Message:
    'require' is not defined.
Details:
    fileName: C:\Users\Adrian\Source\Repos\bubblesearch-vnext\vNextApps\src\BubbleSearch\src\js\init.js
    lineNumber: 1
[12:28:09]
C:\Users\Adrian\Source\Repos\bubblesearch-vnext\vNextApps\src\BubbleSearch\src\js\init.js
   1:0   error  'require' is not defined                    no-undef
   2:13  error  Strings must use doublequote                quotes
   4:8   error  Strings must use doublequote                quotes
   4:18  error  Strings must use doublequote                quotes
   5:8   error  Strings must use doublequote                quotes
   5:21  error  Strings must use doublequote                quotes
   6:8   error  Strings must use doublequote                quotes
   6:23  error  Strings must use doublequote                quotes
   9:8   error  Strings must use doublequote                quotes
   9:23  error  Trailing spaces not allowed                 no-trailing-spaces
  11:16  error  Strings must use doublequote                quotes
  12:14  error  Trailing spaces not allowed                 no-trailing-spaces
  17:0   error  'require' is not defined                    no-undef
  18:4   error  Strings must use doublequote                quotes
  19:4   error  Strings must use doublequote                quotes
  20:4   error  Strings must use doublequote                quotes
  21:4   error  Strings must use doublequote                quotes
  22:3   error  Missing "use strict" statement              strict
  24:6   error  'document' is not defined                   no-undef
  25:10  error  Strings must use doublequote                quotes
  28:1   error  Newline required at end of file but not found  eol-last

C:\Users\Adrian\Source\Repos\bubblesearch-vnext\vNextApps\src\BubbleSearch\src\js\classes\MySearchBox.js
  1:1  error  Illegal import declaration

✖ 22 problems (22 errors, 0 warnings)

[12:28:10] Finished 'source:compile-js' after 808 ms
```

- 只**启用部分**规则 (不然修不完)
- 将警告、错误**全部修复**
- 将**部分规则改为 Error**，让所有人一定得遵守

#2 CSS Module

避免全域 CSS 覆写问题



问题：全域 CSS 覆写问题日渐严重

全栈工程师对 CSS 策略没兴趣

```scss
wrapper {
    background: red;
}
.tag-box {
    border: solid 1px #ccc;
}
```

```jsx
import css from './style.scss';

export default (props) => (
    <div className={css.wrapper}>
        <div className={css.tagBox}>...</div>
    </div>
);
```

- 只针对 ***.scss** 处理，把 *.css 留给第三方库
- 启用 **camelSpace**，在 JSX 中好写些
- 比预估困难上许多：因过去元件样式表中**混杂使用全域**

#3 组件结构

目录及语法调整

```
jsx
├── scroll-box
│   ├── demo.jsx
│   ├── index.jsx
│   └── style.scss
├── search-box
│   ├── demo.jsx
│   ├── index.jsx
│   └── style.scss
└── step-progress
    ├── demo.jsx
    ├── index.jsx
    └── style.scss
```

问题：过去 JSX 与 jQuery 插件混放

语法不一致（改为 import, ES6 class）

好处：让大家知道目录结构调整

也邀了全栈工程师一起帮忙

#4 单元测试

替 API 层写测试

```
import {Observable} from 'rxjs';

export const TagsAPI =  {
    URL: '/api/tags',
    create$() {
        return Observable.ajax({method: 'POST', ...});
    }
    retrieve$() {
        return Observable.ajax({method: 'GET', ...});
    }
    modify$() {
        return Observable.ajax({method: 'PUT', ...});
    }
    destroy$() {
        return Observable.ajax({method: 'DELETE', ...});
    }
};
```

问题：大家**介面名称不统一**、**实作方法也不一致**

益处：互相写测试、學習，提出不少**改进**

代码猴子

CodeMonkey Session 是个很好的**团队默契培养**时间

藉由**简单可确定**的任务，所有人往同一方向迈进

应是**定期都要举办**的活动

变化带来问题

WHY? 人手变多、技术成长都很好啊！

人力变多、效率反而变慢？

越大的团队效率越不好

技术成长、但其它团队成员跟上了吗？

以团队思考、而非个人

人月神话

The Mythical Man Month

“在一個臨界點上，顧用一個新的开发者、反而會讓開發速度降低，因為**軟體開發的複雜性**會需要更多的**溝通及管理成本**”

协作成本增加：例如开会、讨论、Code Review

前端草创

导入新技术速度较慢

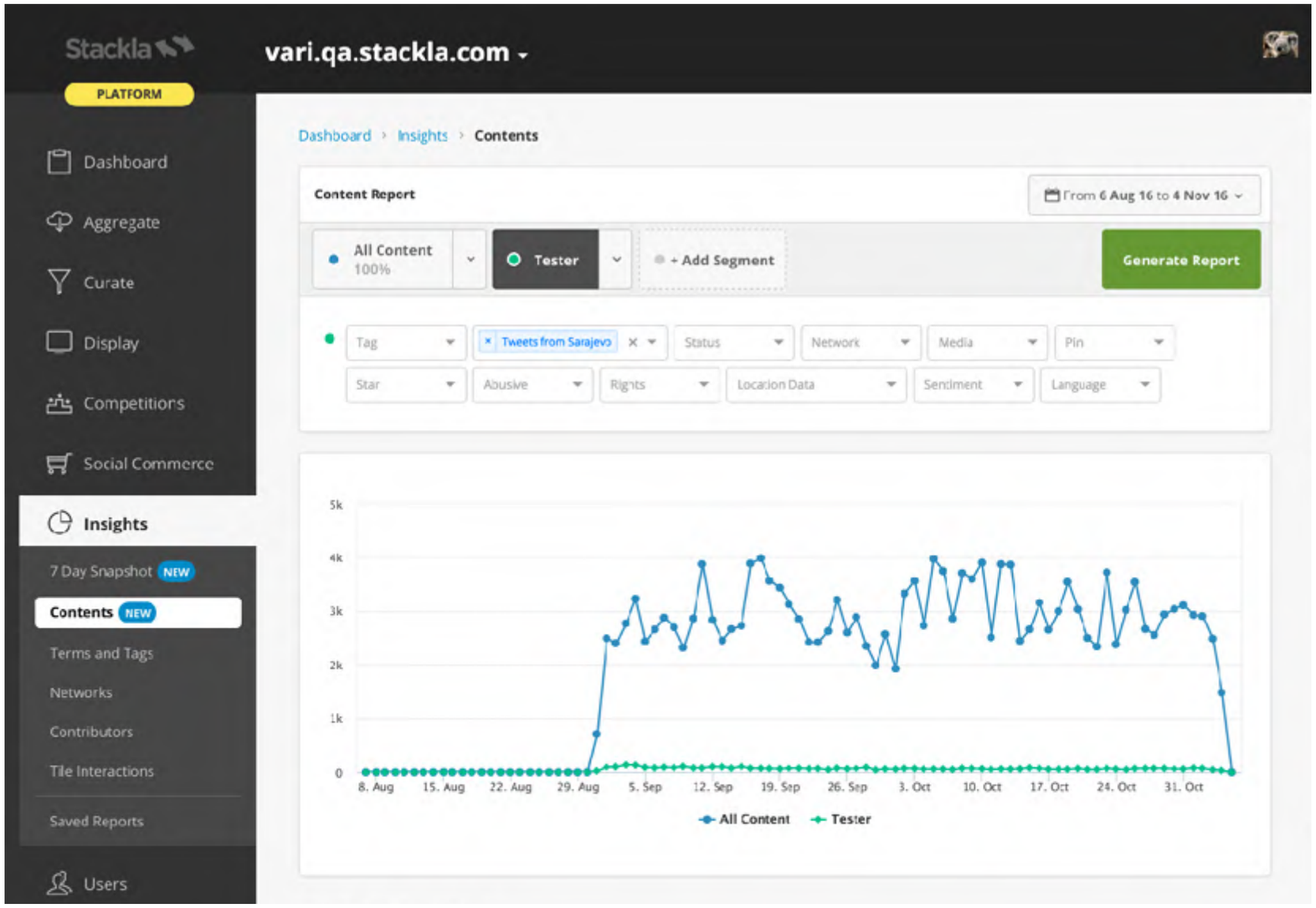
大多数 Startup 不会有明显的前后端分工

但前端开发者应**领导前端工具**的使用

选择能**实际解决问题**、**不增加负担**的工具、其它人会很乐意跟随

新项目

带入太多新技术、抽象



过度工程化：想要解决目前 Redux 重复的问题

Store #3

阶段 #1：不考虑共用

```
▼ filters/index: Object
  ▶ data: Array[0]
  ▶ meta: Object
```

阶段 #2：导入共用层

```
▼ reports/user: Object
  ▶ common: Object
    visibleResultsCount: 3
```

阶段 #3：再导入 5 种以上抽象

```
▼ entities: Object
  ▼ reports: Object
    ▼ models: Object
      ▶ 2f48a8794bae89906c6f30e1e7d8c239: Object
      ▶ 4c5ed1d5a990dab607b88d2748120113: Object
      ▶ 4fc165f5dec8a7fb667d9c36eae07ba7: Object
      ▶ 5eb92930950eb8cf6d70f49f4cc5ab43: Object
      ▶ 7a095e5d2ce0c1095d05413719de19e1: Object
      ▶ 7b3058acffe9ef13aaac1529c6d60275: Object
      ▶ 9dfb01ab0255c40fd85daaef5df4f8b8: Object
      ▶ 25b4f3f5dcbe31d9fd96f547cfd6e21c: Object
      ▶ 74db77449ec399ad1c8ad8bc7364600e: Object
      ▶ 86b31e3813b07e2514da3cc457bc4bf2: Object
      ▶ 295ac935ab66e64632ce7f6d3cac5fb3: Object
  ▼ reports/user: Object
    ▶ common: Object
      visibleResultsCount: 3
```

- **类似只有 controller 的概念**
  - Store 不管重复资料
  - API 返回阵列，不会处理
- **类似有 model + controller 的概念**
  - entities 中不应该有重复资料
  - 可直接用 ID 查找、速度快

档案架构演化

阶段 #1：不考虑共用

- reports/user/**index.jsx**
- reports/user/**redux.js**
  - Action Creators
  - Reducer
- reports/user/**style.scss**

阶段 #2：导入共用层

- reports/**common**/**index.jsx**
- reports/**common**/**redux.jsx**
  - Action Creators
  - Reducer
- reports/user/index.jsx
- reports/user/redux.js
  - Action Creators
  - Reducer
- reports/user/style.scss

阶段 #3：再导入 5 种以上抽象

- **common/syncing/index.js**
- **common/entities/reports/epic.js**
- **common/entities/reports/redux.js**
  - Action Creators
  - Reducer
  - **Selector**
  - **Schema**
- reports/common/index.jsx
- reports/common/redux.js
- reports/content/index.jsx
- reports/content/redux.js
  - Action Creators
  - Reducer
  - **Selector**
  - **Schema**
- reports/content/style.scss

是进化还是退化

团队成长也是个挑战

现况：仍需要全栈工程师的贡献，但**过多技术**制造了**隔阂**

是进化还是退化

团队成长也是个挑战

永远不要忘了问自己这个问题

「我 (我们团队) 真的需要他吗？」

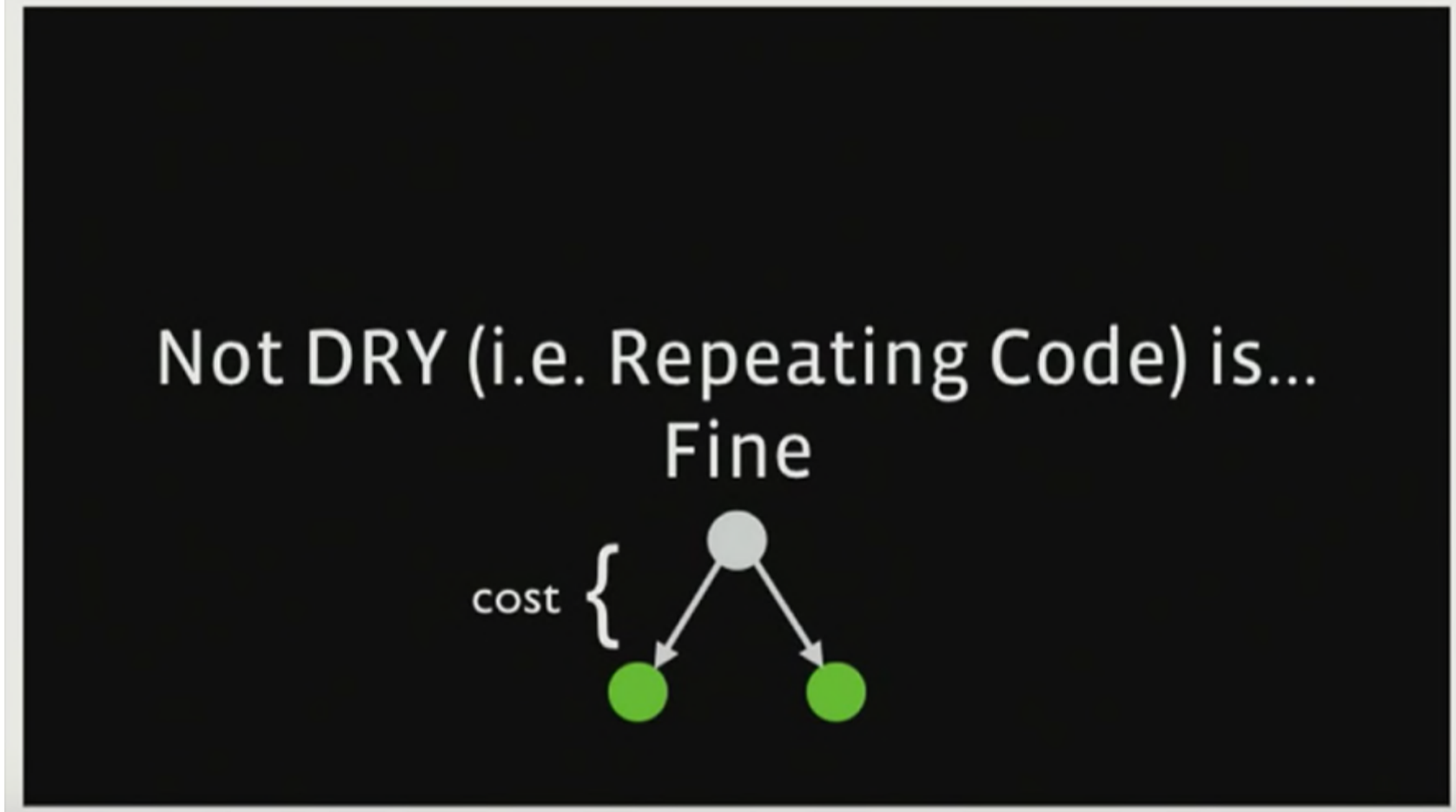
即使没有这些技术：问题一样有解法、速度或许更快

当团队都很享受、气氛和谐时，效率是最好的

现况：仍需要全栈工程师的贡献，但**过多技术**制造了**隔阂**

重复有时是好的

On the Spectrum of Abstraction



RANGLE.IO
REWRITING THE WEB

任何的抽象都会增加复杂度

Copy & Paste 因为没有抽象，复杂度是低的

https://www.youtube.com/watch?v=mVVNJKv9esE

专注＝牺牲

每买个东西、每安装个 Lib，都有成本，累积起来很巨大

把电视卖掉？把车子卖掉？把 reselect 移除？

https://www.youtube.com/watch?v=hlYiWznhhzw

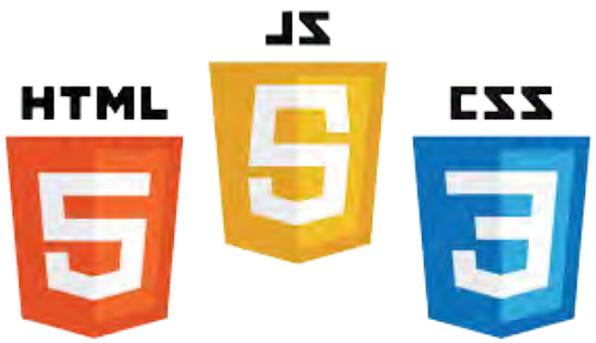
团队重置：小结

学习**牺牲**，评估移除部分你很喜欢的工具或函式库

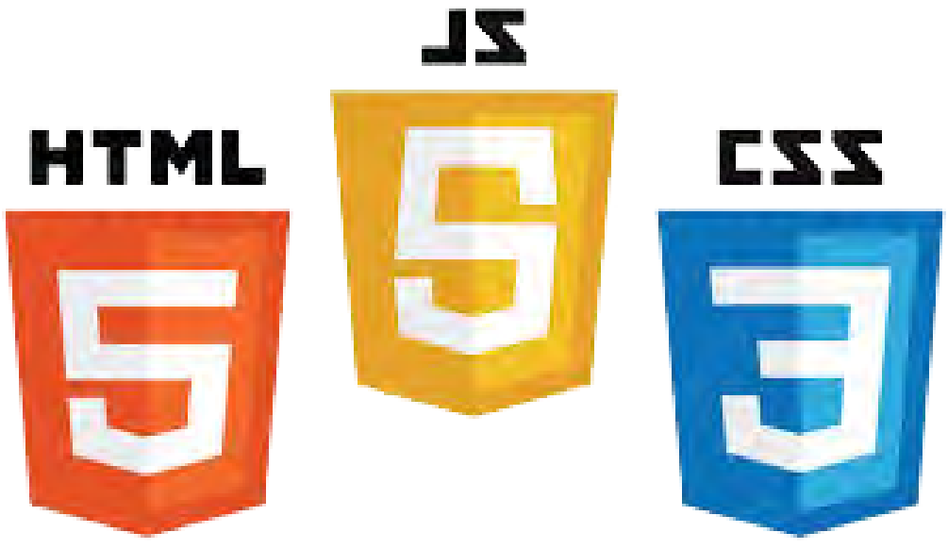
- 怎么样的**复杂度**是团队可以接受的？
- **是否真的需要**：没了这个工具，好处与坏处？
- 怎么样才**对团队有益**？

团队重置：**努力中**

#4 技术重置：前端 ⟳ *Front-end Reset*



最佳实践



HTML 文档

CSS 选择器命名

JavaScript 编程

HTML 文件

1995 结构、样式、行为全部写在 HTML 中

最佳实践：HTML、样式、JS 合并一起写

1998 WaSP 网页标准化小组

最佳实践：HTML、CSS、JS 应分离

2015 React 暴红

最佳实践：HTML、CSS、JS 应合并

CSS 命名方式

1998

CSS2

最佳实践：无 (随意命名)

2005

2005 以 ID、模块为命名空间、禁止以外观命名

最佳实践：命名空间、结构化

2011

Bootstrap 暴红、带动了 OOCSS

最佳实践：元件化、用各种方式赋予选取器意义

2015

2015 CSS Module - 随意以外观命名

最佳实践：无 (随意命名)

JS 编程方式

1998

仅表单验证，只需写 Function

最佳实践：函数式编程 $f(x)$

2004

JavaScript 开始变得复杂（Web 2.0）

最佳实践：物件导向编程 OOP

2015

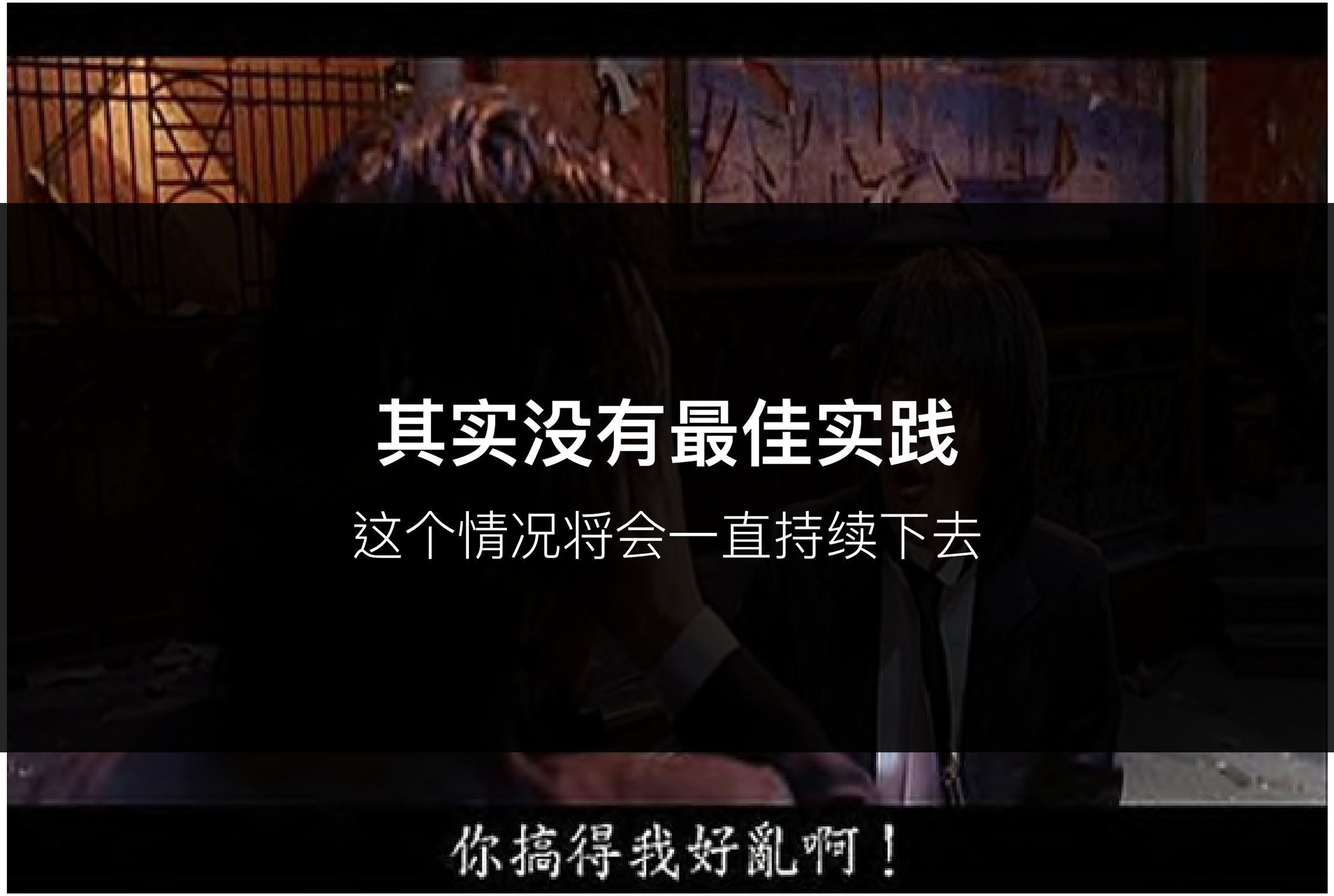
函数式编程火红、只需写 Function

最佳实践：函数式编程 $f(x)$


你搞得我好亂啊！

其实没有最佳实践

这个情况将会一直持续下去



技术不断迭代

在2016 年学JavaScript 是一种什么样的体验？

用讽刺法来表达 JS 生态圈让人困扰的快速变化与混乱

技术不断迭代

在2016 年学JavaScript 是一种什么样的体验？

Jose Aguinaga Follow
Web Engineer, UX & Interactions Designer, Full-Stack Javascript Developer. Previously @numbrs, @...
Oct 4 · 12 min read

How it feels to learn JavaScript in 2016

我们几乎一直在重置

背后的意义是什么？

用讽刺法来表达 JS 生态圈让人困扰的快速变化与混乱

架構的演化

套件管理

CSS 预处理器

JS 模块载入

JS 语法编译器

构建系统

架構的演化

套件管理

CSS 预处理器

sass-loader

less-loader

JS 模块载入

JS 语法编译器

coffee-loader

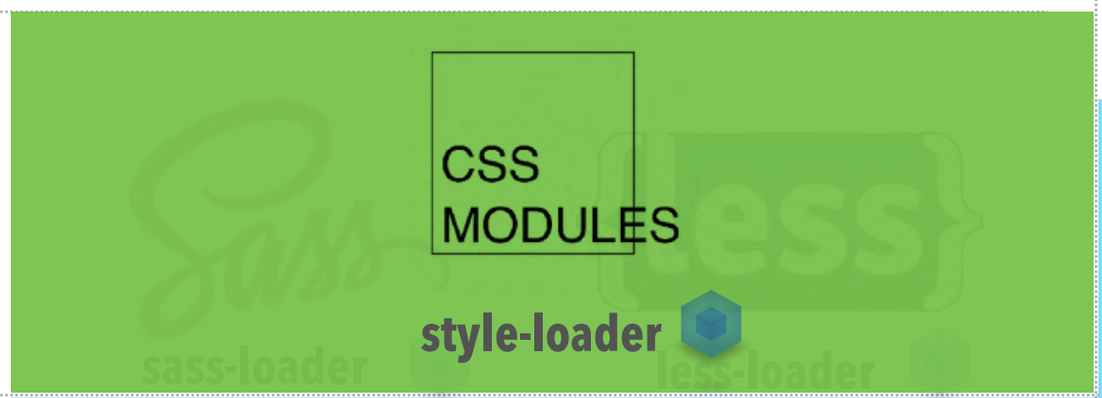
ts-loader

构建系统

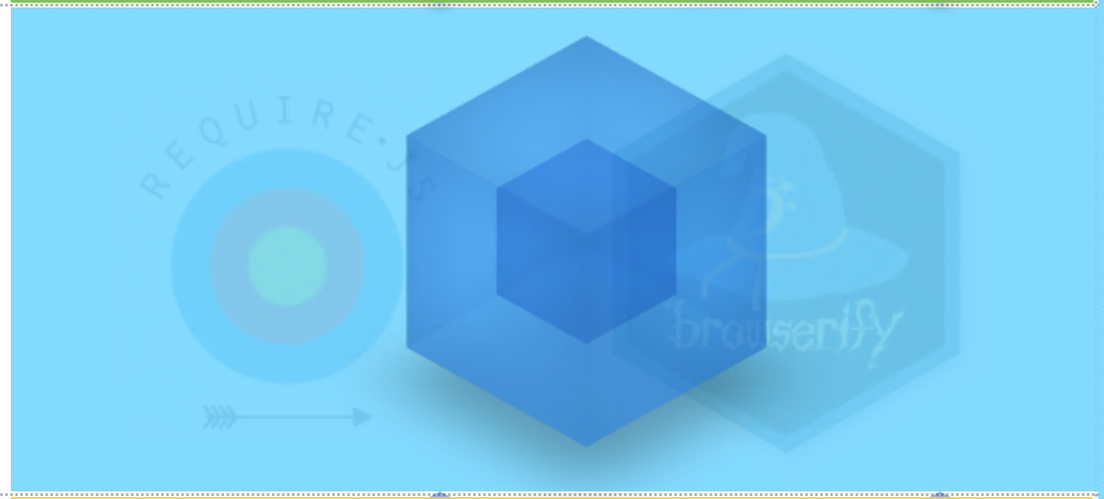
架構的演化

套件管理

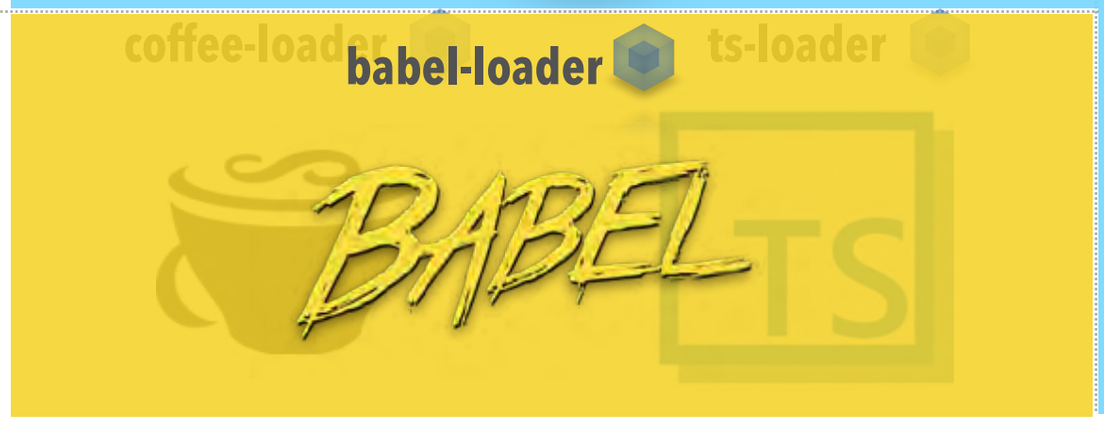
CSS 预处理器



JS 模块载入



JS 语法编译器



构建系统

架構的演化

套件管理

构建系统

CSS 预处理器

CSS MODULES

style-loader

技术重置 = 持续地汰换、往好方向发展

不断分裂又集成、让前端接近真正的软体开发

JS 模块载入

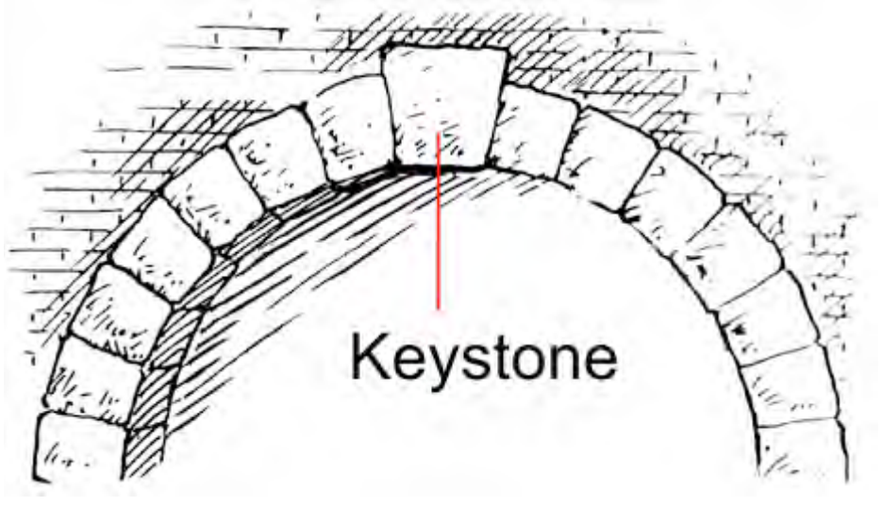
JS 语法编译器

coffee-loader

babel-loader

ts-loader

前端重置：小结



前端工程：软体工业的碶石

- 前端**年轻**、仍然有很多问题待解决
- 前端**极其重要**，才会吸引这么多人不断思考改变
- 工具的迭代带来了正确的方向、更**接近真正的软体开发**
- 前端工程师是**时代的桥梁**、只有我们才有能力**朝更好的方向迈进**

Q & A

- 生活重置
- 心态重置
- 团队重置
- 技术重置


Cheers,
mate!