

Jscex

案例、经验、阻碍、展望

赵劼 - 2012.6

关于我

- 赵劼 / 老赵 / Jeffrey Zhao / 赵姐夫
- 日写代码三百行，不辞长作程序员
- 博客：<http://blog.zhaojie.me/>
- 微博：@老赵
- F#, JavaScript, Scala, C#, Python, .NET, Mono...
- 痛恨Java语言

内容提纲

- 简介
- 案例
- 经验
- 阻碍
- 展望

简介

关注人数

- Node.js流程控制模块（80多个）
 - Async: 2500+
 - Step: 900+
 - Jscex: 600+
 - Fiber: 500+
 - ...



jeffz_cn: 有人在用Jscex吗? <http://t.co/lsncDcSp>

12:20pm, Jun 04 from HootSuite

亦未
可知

audreyt: @jeffz_cn 今天剛看到，正在試用，很有意思。AOT 生成的代碼似乎可以考慮用 <https://t.co/gMlCo5Yx> 生成源碼對映表，以便偵錯。另外，也在想照著 <https://t.co/zo5heFEK> 另外寫個 jQuery.Deferred monad 試試看。

2:05am, Jun 10 from Web



jeffz_cn: @audreyt 多谢建议! source map也在考虑，现在用的parser是UglifyJS，信息都丢的差不多了，所以也在考虑在下一个版本里换新的parser <http://t.co/O0DOxpAK> 现在对 Esprima 的感觉不错。

11:39am, Jun 10 from HootSuite

亦未
可知

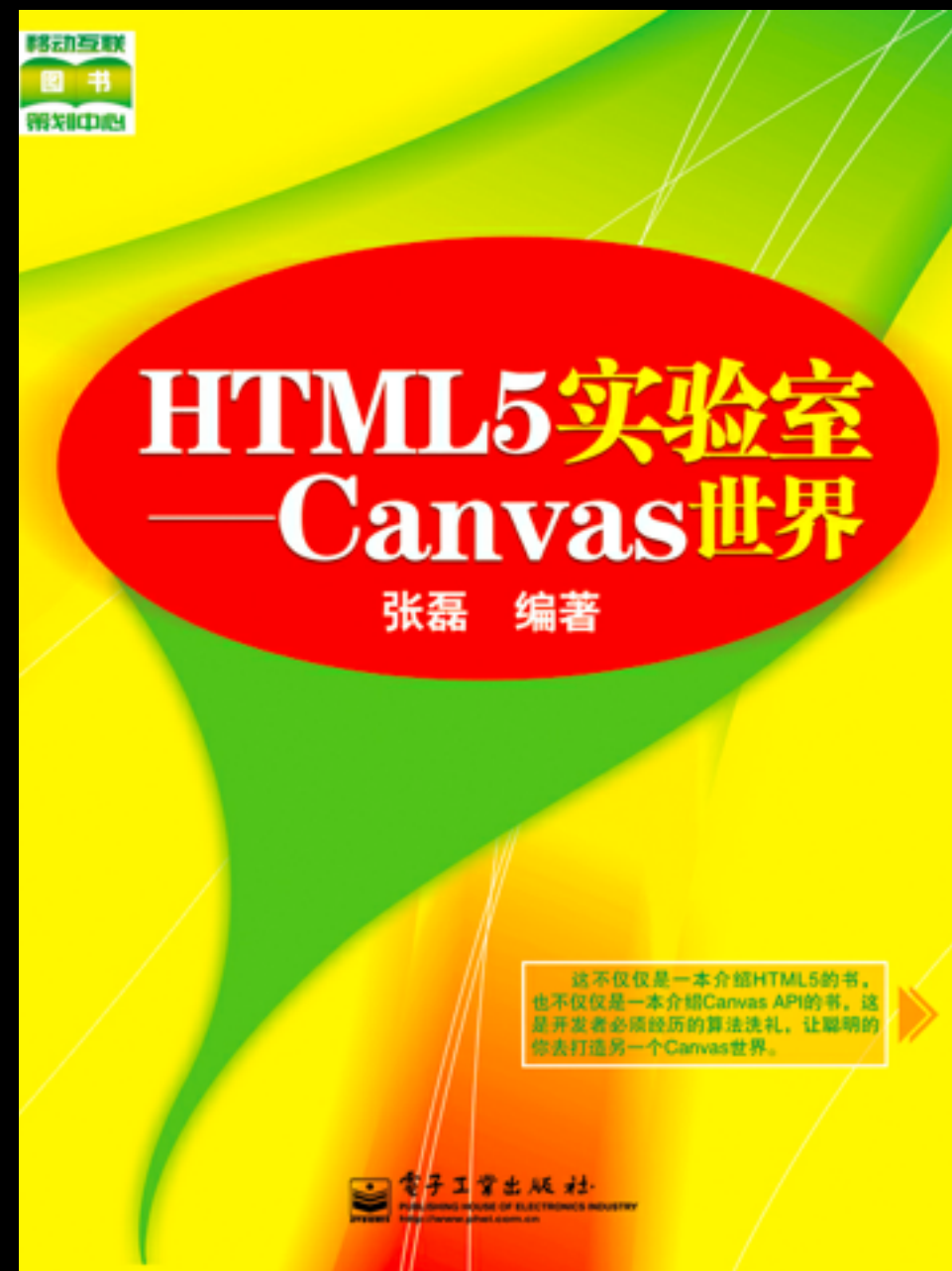
audreyt: @jeffz_cn 看來確實不錯。另外，我剛寫了個很簡單的 jscex-jquery 插件 <https://t.co/g2w9RWLQ>，裡頭有個 Jscex.Async.Binding.fromPromise <https://t.co/CXjsgFTR> 您也許用得上，歡迎指教。

4:09pm, Jun 10 from Web

唐凤大侠好评

HTML 5 实验室

电子工业出版社



回归JavaScript的 异步流程控制

Jscex是什么?

- JavaScript Computation EXpression
- JavaScript语言扩展，用于改善某些常见场景下的编程（如异步编程）
- F#计算表达式特性的JavaScript移植
 - 受“计算表达式”特性启发
 - 为JavaScript设计，基于JavaScript实现

Jscex不是什么？

- 另一种语言：
 - Jscex是百分百的JavaScript
- 框架
 - Jscex是类库，能够与几乎任何类库/框架一起使用
- JavaScript引擎/运行时
 - 在任何ECMAScript 3上执行

浅尝辄止

冒泡排序

```
var compare = function (x, y) {  
    return x - y;  
}  
  
var swap = function (a, i, j) {  
    var t = a[x]; a[x] = a[y]; a[y] = t;  
}  
  
var bubbleSort = function (array) {  
    for (var x = 0; x < array.length; x++) {  
        for (var y = 0; y < array.length - x; y++) {  
            if (compare(array[y], array[y + 1]) > 0) {  
                swap(array, y, y + 1);  
            }  
        }  
    }  
}
```

做成动画

```
var compare = function (x, y, callback) {
    setTimeout(10, function () {
        callback(x - y);
    });
}

var swap = function (a, i, j, callback) {
    var t = a[x]; a[x] = a[y]; a[y] = t;
    repaint(a);

    setTimeout(20, callback);
}

var outerLoop = function (array, x, callback) {
    if (x < array) {
        innerLoop(array, x, 0, function () {
            outerLoop(array, x + 1, callback);
        });
    } else {
        callback();
    }
}
```

```
var innerLoop = function (array, x, y, callback) {
    if (y < array.length - x) {
        compare(array[y], array[y + 1], function (r) {
            if (r > 0) {
                swap(array, y, y + 1, function () {
                    innerLoop(array, x, y + 1, callback);
                });
            } else {
                innerLoop(array, x, y + 1, callback);
            }
        });
    } else {
        callback();
    }
}

outerLoop(array, 0, function () {
    console.log("done!");
});
```

做成动画

```
var compare = function (x, y, callback) {
  setTimeout(10, function () {
    callback(x - y);
  });
}

var swap = function (a, i, j, callback) {
  var t = a[i]; a[i] = a[j]; a[j] = t;
  repaint(a);

  setTimeout(20, callback);
}

var outerLoop = function (array, x, callback) {
  if (x < array.length) {
    innerLoop(array, x, 0, function () {
      outerLoop(array, x + 1, callback);
    });
  } else {
    callback();
  }
}

var innerLoop = function (array, x, y, callback) {
  if (y < array.length - x) {
    compare(array[y], array[y + 1], function (r) {
      if (r > 0) {
        swap(array, y, y + 1, function () {
          innerLoop(array, x, y + 1, callback);
        });
      } else {
        innerLoop(array, x, y + 1, callback);
      }
    });
  } else {
    callback();
  }
}

outerLoop(array, 0, function () {
  console.log("done!");
});
```

这TMD是什么啊!

冒泡排序动画

```
var compareAsync = eval(Jscex.compile("async", function (x, y) {  
    $await(Jscex.Async.sleep(10)); // each "compare" takes 10 ms.  
    return x - y;  
}));
```

```
var swapAsync = eval(Jscex.compile("async", function (a, x, y) {  
    var t = a[x]; a[x] = a[y]; a[y] = t; // swap  
    repaint(a); // repaint after each swap  
    $await(Jscex.Async.sleep(20)); // each "swap" takes 20 ms.  
}));
```

```
var bubbleSortAsync = eval(Jscex.compile("async", function (array) {  
    for (var x = 0; x < array.length; x++) {  
        for (var y = 0; y < array.length - x; y++) {  
            var r = $await(compareAsync(array[y], array[y + 1]));  
            if (r > 0) $await(swapAsync(array, y, y + 1));  
        }  
    }  
}));
```

冒泡排序动画

```
var compareAsync = eval(Jscex.compile("async", function (x, y) {  
    $await(Jscex.Async.sleep(10)); // each "compare" takes 10 ms.  
    return x - y;  
}));
```

```
var swapAsync = eval(Jscex.compile("async", function (a, x, y) {  
    var t = a[x]; a[x] = a[y]; a[y] = t; // swap  
    repaint(a); // repaint after each swap  
    $await(Jscex.Async.sleep(20)); // each "swap" takes 20 ms.  
}));
```

```
var bubbleSortAsync = eval(Jscex.compile("async", function (array) {  
    for (var x = 0; x < array.length; x++) {  
        for (var y = 0; y < array.length - x; y++) {  
            var r = $await(compareAsync(array[y], array[y + 1]));  
            if (r > 0) $await(swapAsync(array, y, y + 1));  
        }  
    }  
}));
```


异步编程十分困难

破坏代码局部性

- 程序员习惯线性地表达算法
- 异步代码将逻辑拆分地支离破碎
- 难以
 - 异步操作之间的协作及组合
 - 处理异常及取消

异步函数

```
// 使用异步构造器执行编译后的代码
var somethingAsync = eval(Jscex.compile("async",
    function (...) {
        // 实现
    }
));
```

响应

```
function () {  
  var res = $await(<async work>);  
}
```

响应

```
function () {  
    var res = $await(<async work>);  
}
```

HTTP请求
UI事件
时钟回调
查询响应
Web Service响应
代理消息

```
function () {
    var img = $await(readAsync("http://..."));
    console.log("loaded!");
    $await(writeAsync("./files/..."));
    console.log("saved!");
}
```

=

```
(function () {
    var _b_ = Jscex.builders["async"];
    return _b_.Start(this,
        _b_.Delay(function () {
            _b_.Bind(readAsync(...), function (img) {
                console.log("loaded!");
                return _b_.Bind(writeAsync(...), function () {
                    console.log("saved!");
                    return _b_.Normal();
                });
            });
        });
    );
})();
```

I/O并行

- 许多程序是I/O密集型应用
 - 使用Web服务
 - 使用磁盘上的数据
- 网络和磁盘速度发展很慢
- I/O资源天然可以并行
 - 提高性能的关键方式

首要设计原则

Jscex即JavaScript

- 语言特性
- 语言语义
- 编程体验

语言特性

- 支持几乎所有JavaScript语言功能
 - 循环： `while` / `for` / `for...in` / `do`
 - 判断： `if` / `switch`
 - 错误处理： `try...catch...finally`
 - 其他： `return` / `break` / `continue` / `throw`
- 嵌套函数
- 不支持的语言特性
 - `with`块
 - 带标签的`break`和`continue`
 - `switch`内带条件的`break`

语义

- 保持完整的JavaScript语义
- 独立的“bind”操作（例如\$await）
 - 语言里的唯一扩展
 - 表现形式为“方法调用”
 - 清楚表明“特殊”的操作

编程体验

- 如JavaScript一般编写、执行、调试
- 修改后立即生效
- 没有额外的编译步骤
 - 代码执行过程中由JIT编译器生成代码

案例

有JS的地方就有JsceX

- 浏览器
- Node.js
- PhoneGap
- Windows 8 Metro
-

工作台

worktile.com 让工作更简单。



组织 事务 协同



工作台

组织

把人汇聚起来，做应该做的事。

 **worktile** (worktile) 

<http://worktile.com>

简介


设置


 成员



 添加成员

 **worktile**  [查看组织简介](#)

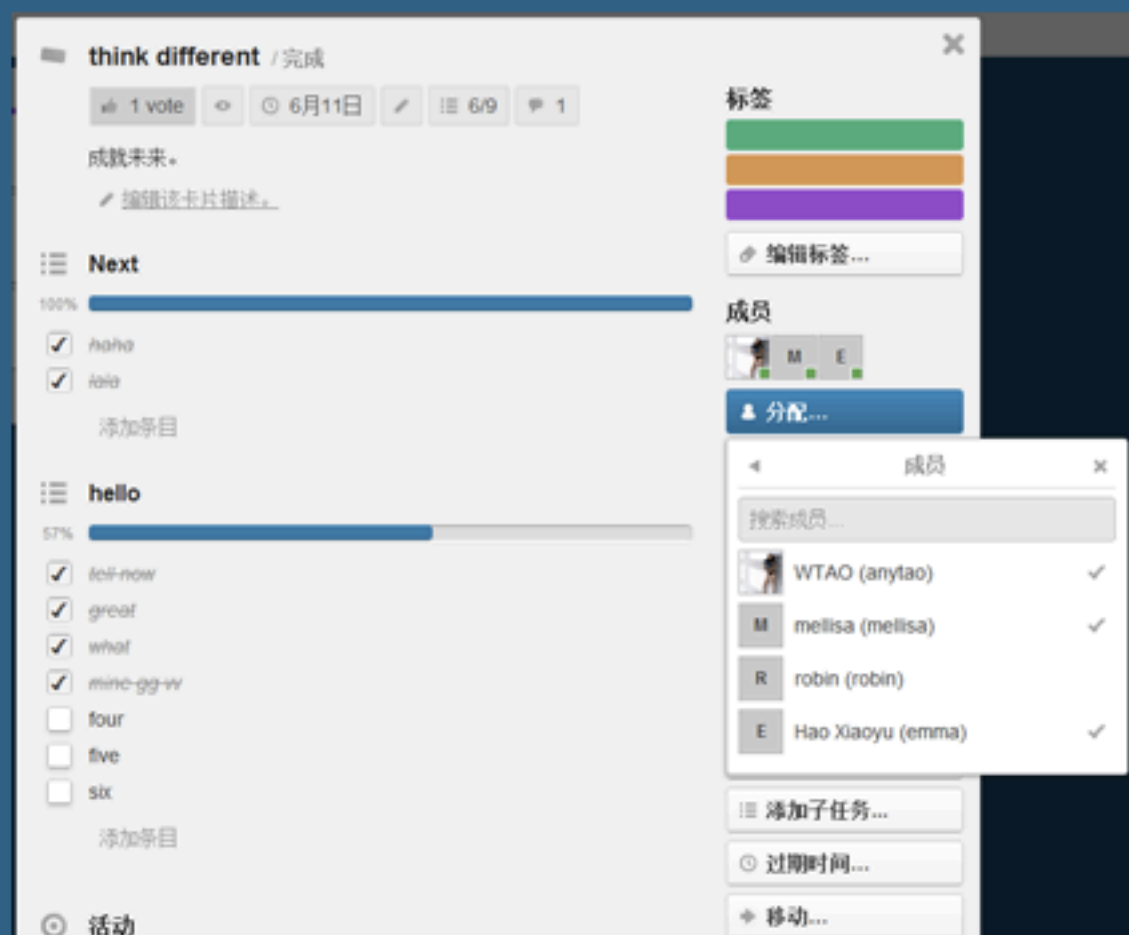
Meme 

doner 

Welldone 

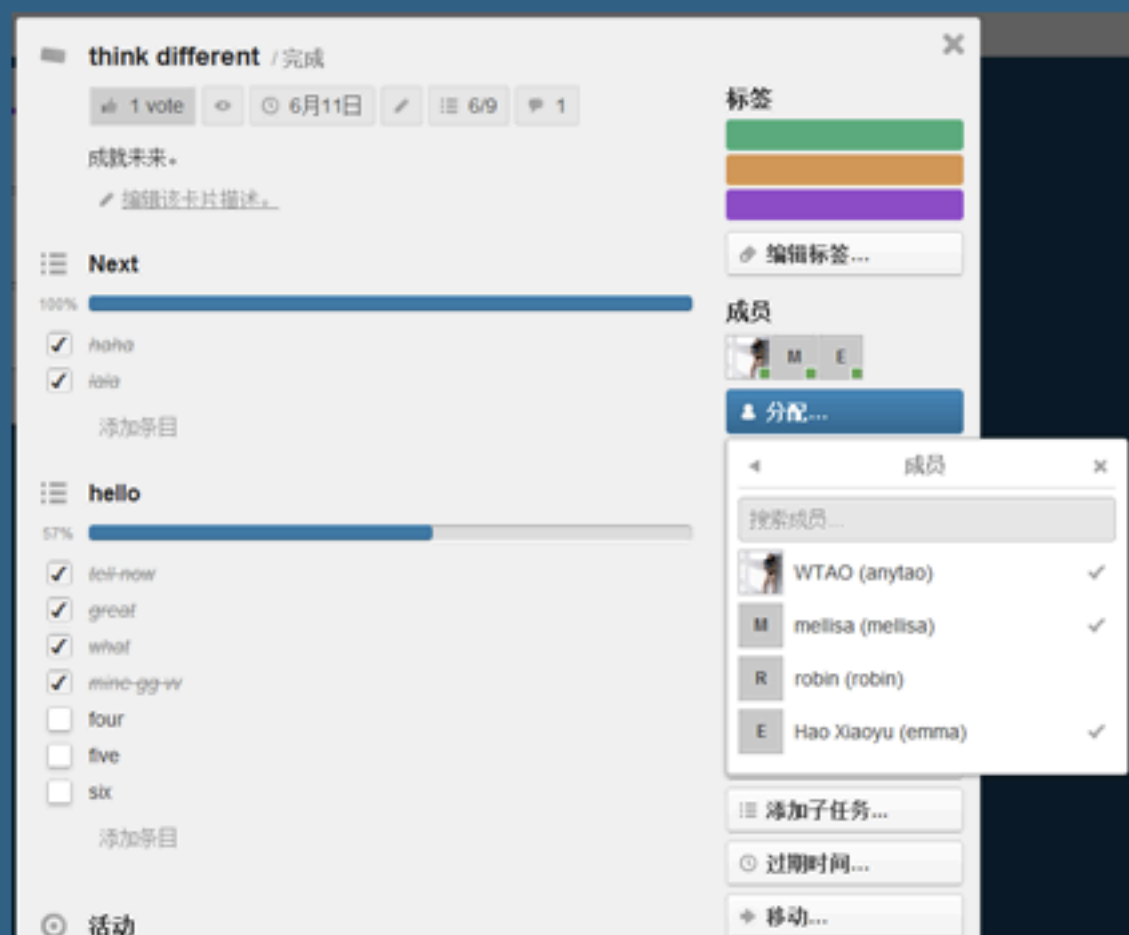
事务

以最方便的方式，处理每天发生的故事。



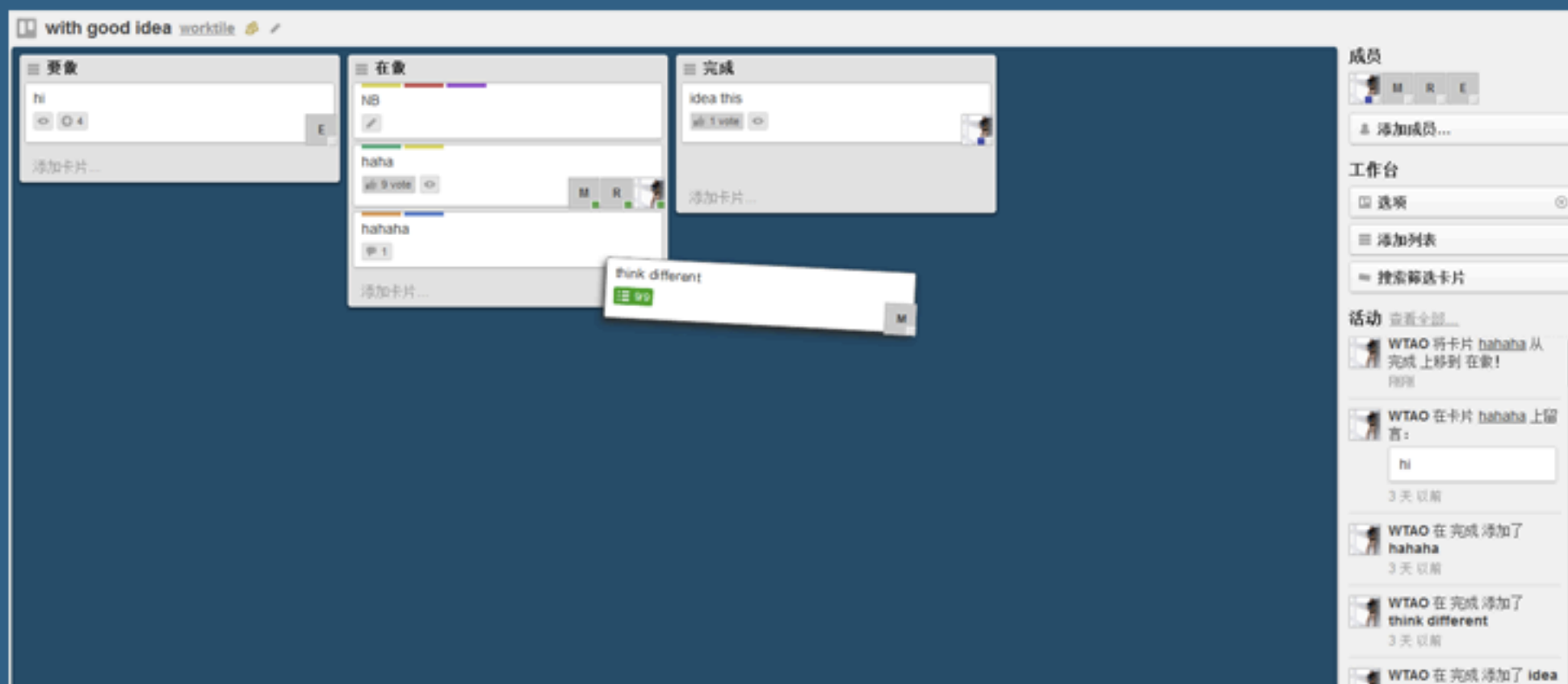
事务

以最方便的方式，处理每天发生的故事。



协同

实时、便捷、交互，拖拖就完成。



“工作台”简介

- 为敏捷团队和个人打造的任务协同和事物管理的生产力工具
- 前后端均使用JavaScript
 - Node.js, Socket.io, Mongoose,
- 挑战：完全异步编程在某些同步需求下变得非常难于实现

Show Me the Code

```
exports.getUserActivity = function (uid, me_uid, count, page, cb) {  
  var query = models.Activity.find(...).where(...).desc(...);  
  query.exec(function (err, activities) {  
    if (err) return cb(err);  
  
    for (var i = 0; i < activities.length; i++) {  
      var activity = activities[i];  
      var board_id = activity.data.board.board_id;  
  
      db.ActivityData.getBoard(board_id, function (err, board) {  
        // how to execute async operations in a loop?  
      });  
    }  
  });  
};
```

Jscex化

```
exports.getUserActivityAsync = eval(..., function (...) {  
    var query = models.Activity.find(...).where(...).desc(...);  
    var activities = $await(query.execAsync());  
  
    for (var i = 0; i < activities.length; i++) {  
        var activity = activities[i];  
        var board_id = activity.data.board.board_id;  
  
        var board = $await(db.ActivityData.getBoardAsync(board_id));  
        ...  
    }  
}));
```

并行化

```
exports.getUserActivityAsync = eval(..., function (...) {  
  var query = models.Activity.find(...).where(...).desc(...);  
  var activities = $await(query.execAsync());  
  
  var boardsTasks = _.map(activities, function (a) {  
    var board_id = a.data.board.board_id;  
    // no $await!  
    return db.ActivityData.getBoardAsync(board_id);  
  });  
  
  // execute in parallel  
  var boards = $await(Task.whenAll(boardsTasks));  
}));
```

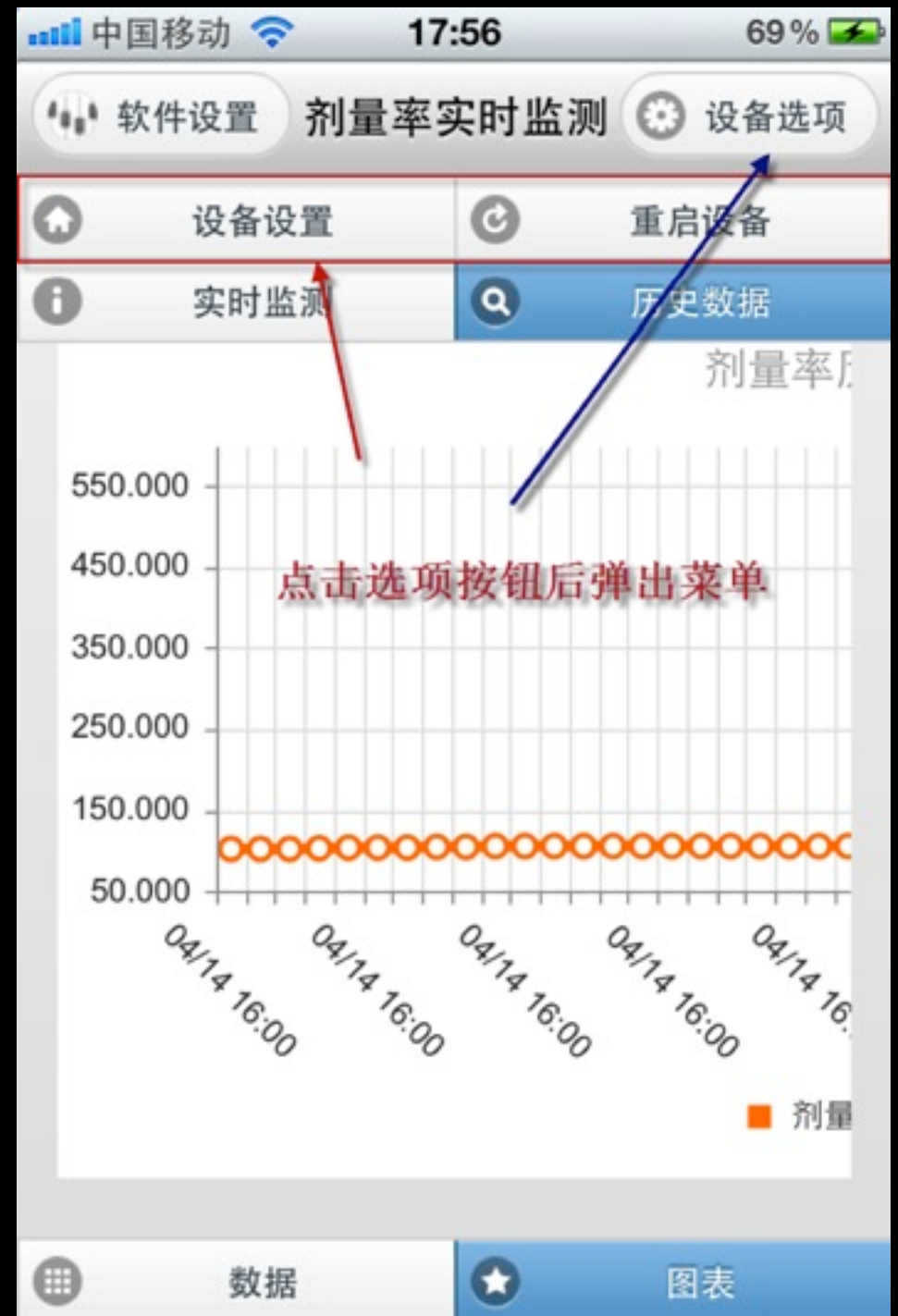


剂量率监测软件

正在启动，请稍候

剂量率监测报警系统





检测声光报警系统

- 使用移动终端监测 RSI3I 剂量率检测设备，方便用户随时查看设备检测值
- 使用Cordova制作的iPhone软件（企业内部部署）
- 场景
 - 闪光灯会循环一明一暗，并循环播放报警声音
 - 弹出一个警告对话框提醒用户当前报警值，用户点击后即停止

```
var isAlerting = false;

function startAlert() {
    isAlerting = true; //全局报警开
    setTimeout(flashOn, 500);
}

function stopAlert() { isAlerting = false; }

function flashOn() {
    if (isAlerting) {
        window.plugins.torch.turnOn(); //开启闪光灯
        setTimeout(flashOff, 500);
    }
}

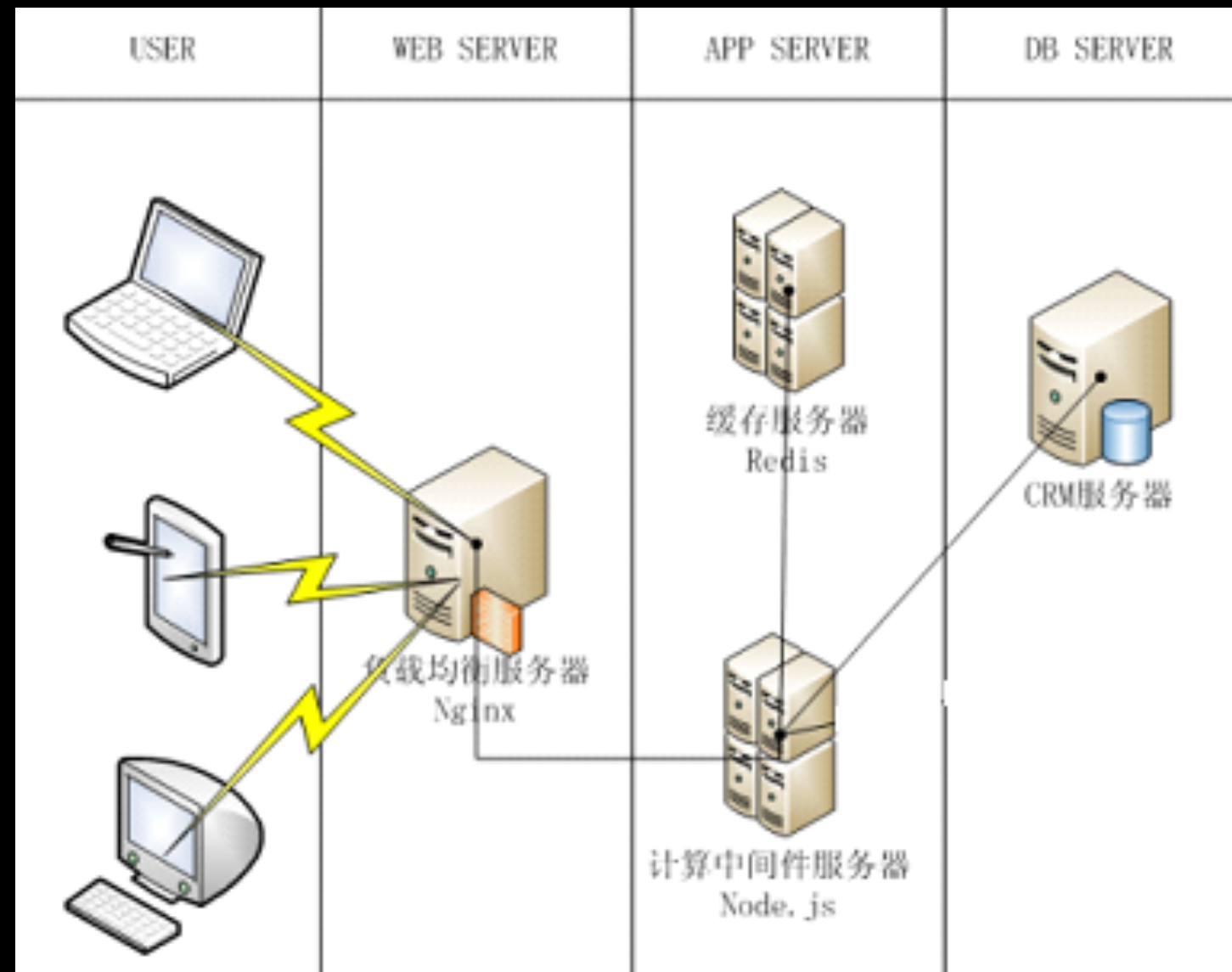
function flashOff() {
    window.plugins.torch.turnOff(); //关闭闪光灯
    if (isAlerting) { setTimeout(flashOn, 500); }
}
```

“闪光灯的一明一暗应该是一个整体效果，但我们这里不得不把它拆成2个函数，还必须在每个函数里嵌入判断全局开关的逻辑。”

“这种思路是非常反人类的。”

- 开发者

```
var showAlert = eval(..., function () {  
    // 取消功能  
    var ct = new CancellationToken();  
    showFlash(ct).start(); // 启动showFlash任务，但不等待  
  
    // 等待用户点击确认按钮  
    $await(Async.onEvent(btnOK, "click"));  
    ct.cancel(); // 取消showFlash任务  
}));  
  
var showFlash = eval(..., function (ct) {  
    try {  
        while (true) {  
            window.plugins.torch.turnOn();  
            $await(Async.sleep(500, ct)); // 等待500毫秒  
  
            window.plugins.torch.turnOff();  
            $await(Async.sleep(500, ct)); // 等待500毫秒  
        }  
    } catch (cex) {  
        window.plugins.torch.turnOff();  
    }  
}));
```



Oracle旧系统改造

系统改造

- 旧方案
 - 使用一个几乎透明的中间件连通客户端和Oracle
 - 几乎所有业务和计算都使用Oracle存储过程实现
- 新方案
 - 使用Node.js承担业务逻辑和计算职责
 - 使用redis作为缓存

挑战与应对

- 挑战

- 将Oracle编写的业务逻辑移植到Node.js代码
- 在异步环境中编写复杂业务逻辑

- 应对

- 将Oracle存储过程自动转化为JS代码
- 稍作修改，配合Jscex即可使用

软件人阿锋：我们在二个星期内就完成了十几个复杂的Oracle包和存储过程的改造。案例改造得益于老赵的JsceX框架，很给力！

@老赵V：刚收到一个很漂亮的JsceX实际案例：一个系统改造过程中需要把原本很多写在Oracle存储过程中的逻辑移植到Node.js代码上，但原本存储过程逻辑是线性编写的，而移植到Node.js时就需要各种回调了，程序员很不适应。现在有了JsceX，他们将大量的逻辑自动转化到JavaScript，再稍作整理就能用上了。很好！

6月23日19:33 来自新浪微博

转发(41) | 评论(19)

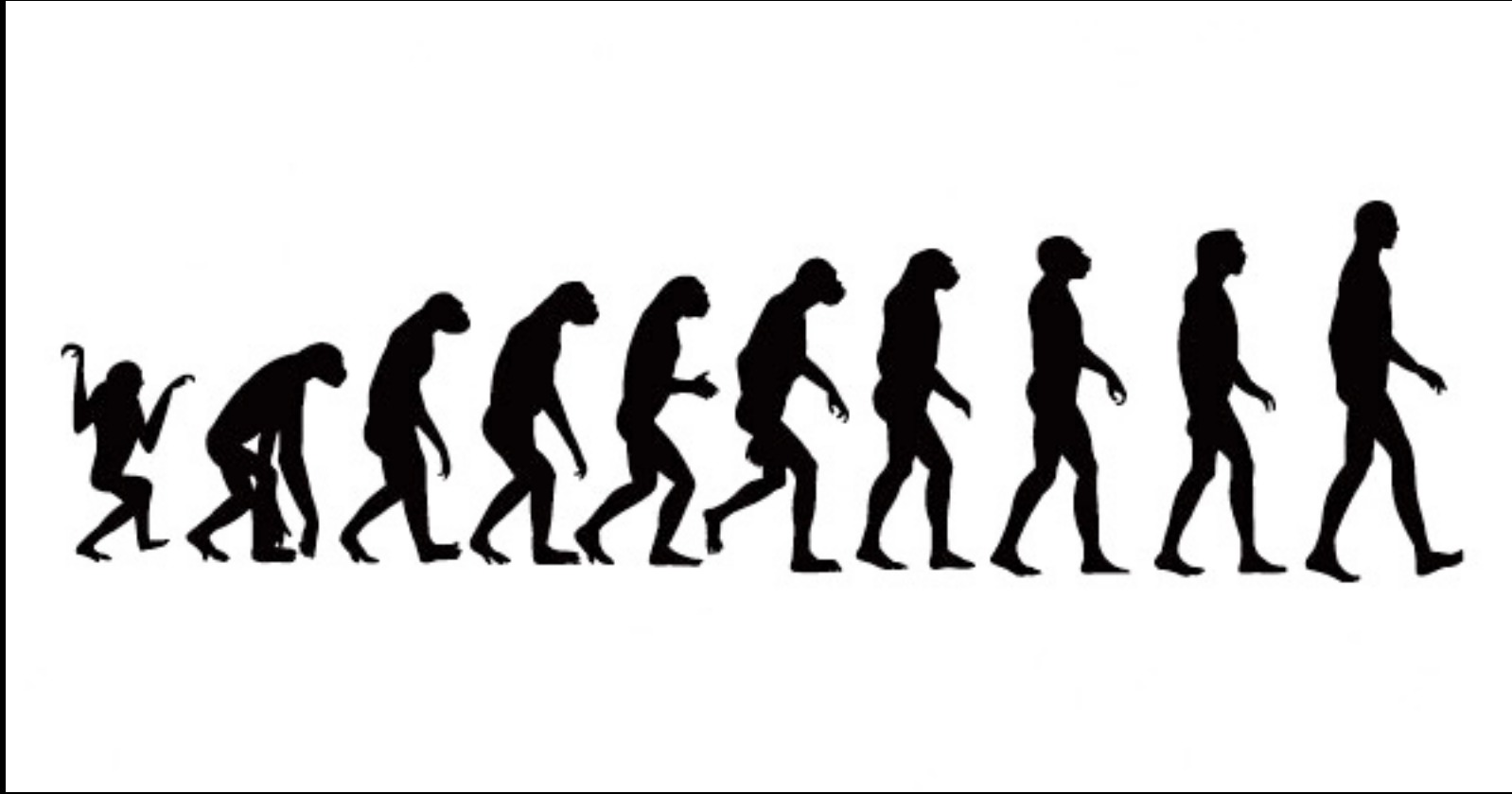
用户评价

更多JsceX使用经验：<http://blog.sina.com.cn/u/1867000922>

阻碍

思维的怪诞

人类学会了行走



冰天雪地

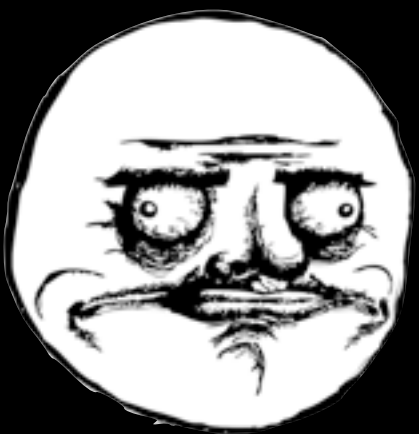


雪橇?

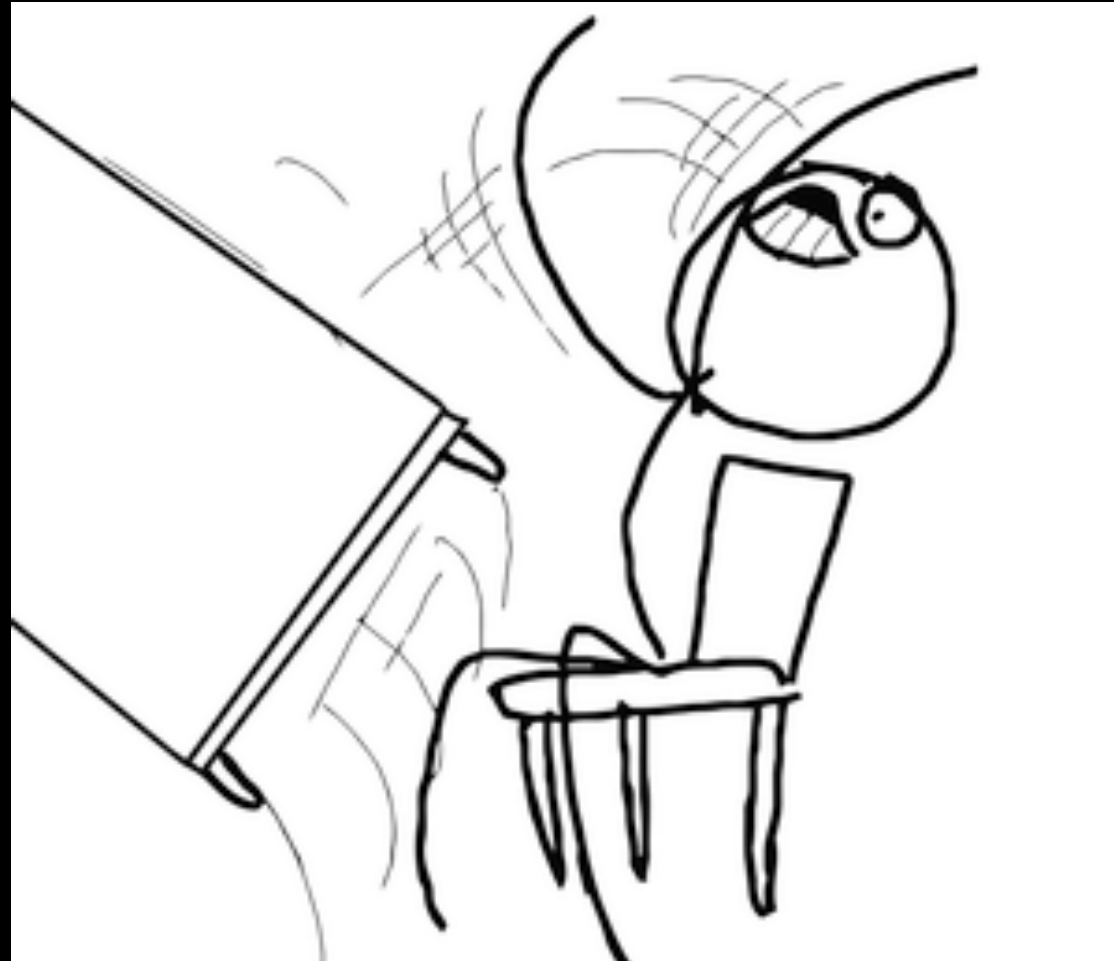


NO.

趴着走是趋势，要迎合



趴着走是文化，要遵循



• • • • •

异步编程

- 异步是一种性质，不是一种编程模型
- 异步有多种编程模型，不同场景适合不同模型
- 迎合语言限制并无不可，但因此开始享受是可悲的

固有的错误思维

- eval is (always) evil
- API太丑，eval为什么不封装？
- 看不懂，不用
- 编译很重很慢，不适合前端
- 生成的代码看不懂，难以调试
-

经验

用户是需要教育的

- 用户并不知道自己想要什么
- 先进的理念会遇到很大阻碍
- 从容易接受的群众下手，威逼利诱，不择手段

开源软件推广

- 做一个开源软件很容易，推广一个开源软件很难
- 技术外的东西十分重要，会耗费大量精力（文档，网站，社区，...）
- 一定要争取共同开发者
- 坚持，坚持，再坚持

接受无奈

- 授权
- 企业文化
- 屌丝身份
 - 高帅富：Sea.js

展望

专业化

- 更多的单元测试
 - 目前测试覆盖率低于30%
 - 下一版本尽可能做到80%
- 更好的调试及错误定位支持
 - Source Map
 - 调试工具

现代化

- 基于更成熟的解析器 (Esprima)
- ECMAScript 5支持
- 更丰富的基础类库
- 更多第三方组件的支持

社区化

- 更多的社区参与度
- 吸引贡献者
 - 代码
 - 示例
 - 文档

国际化

- 英文资源（网站，文档，...）
- 英文媒体
- 英文社区
 - Node.js
 - HTML 5
 - Windows 8

立即使用

- 基于BSD协议发布
- 站点
 - 主站: <http://jscex.info/>
 - 源码: <https://github.com/JeffreyZhao/jscex>

Q & A

谢谢