全栈工程师之路
Node.js

by 桑世龙（i5ting）

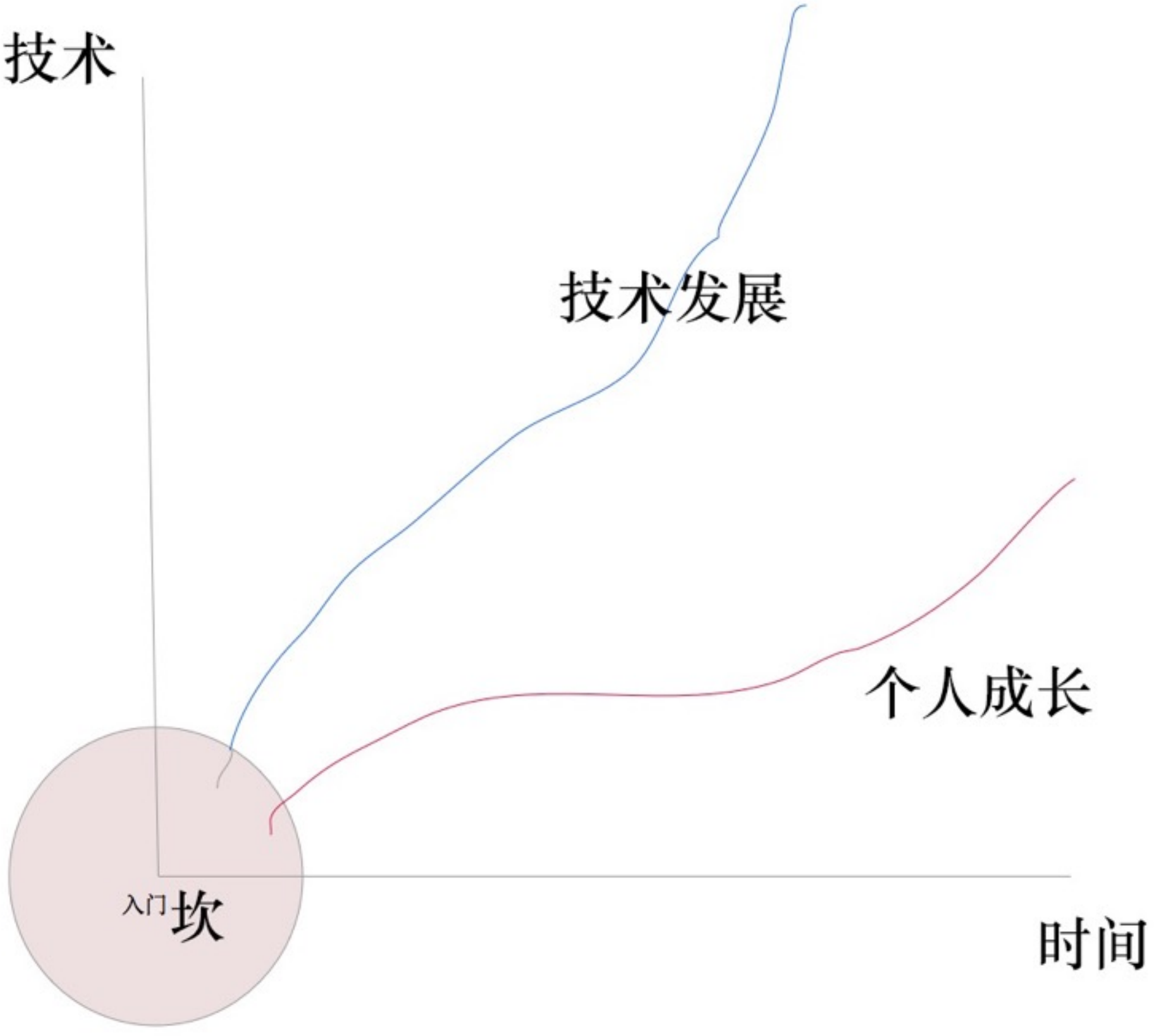
i5ting：一个开源爱好者

StuQ明星讲师

Node.js布道者

空弦科技CTO

Cnodejs管理员


技术
技术发展
个人成长
入门
坎
时间

目录

Part 1：为什么选择 Node.js?

空弦科技做的是基于云仓储的 SaaS 服务，给中小卖家提供服务，核心系统是进销存、订单池、WMS。

“JavaScript 是世界上使用最广泛的语言，没有之一，包括后端开发工程师也更爱使用 JavaScript。”

——stackoverflow

以前

我们总是喜欢拿异步说事儿

除了性能，其他都是病？

现在我们拿 Node.js 的强大的生态来炫耀

1、Callback hell 问题

目前已经很好的解决了。promise / generator / async 后面会讲。

2、包管理

npm 已经是开源世界里最大的包管理器了，模块非常丰富（25.6万 ）。

我们的瓶颈

- 人
- 开发速度
- 稳定

Node.js 好处

- 同样不优化，性能比大部分语言好。即使优化，也比其他语言简单，比如Java、go
- 有足够多的选择和架构的平衡
- 如实在不够，Java 补

简单？难？

- 可以采用面向过程
- 可以面向对象
- 可以函数式

快？慢？

- 执行效率，同样不优化，性能比大部分语言好。
- 开发效率，Node.js 本身比较简单，开发效率还是比较高的。完善的生态，比如测试、工具、npm 大量模块。
- 缺少 Rails 一样的大杀器，scaffold 脚手架，ORM 太弱。

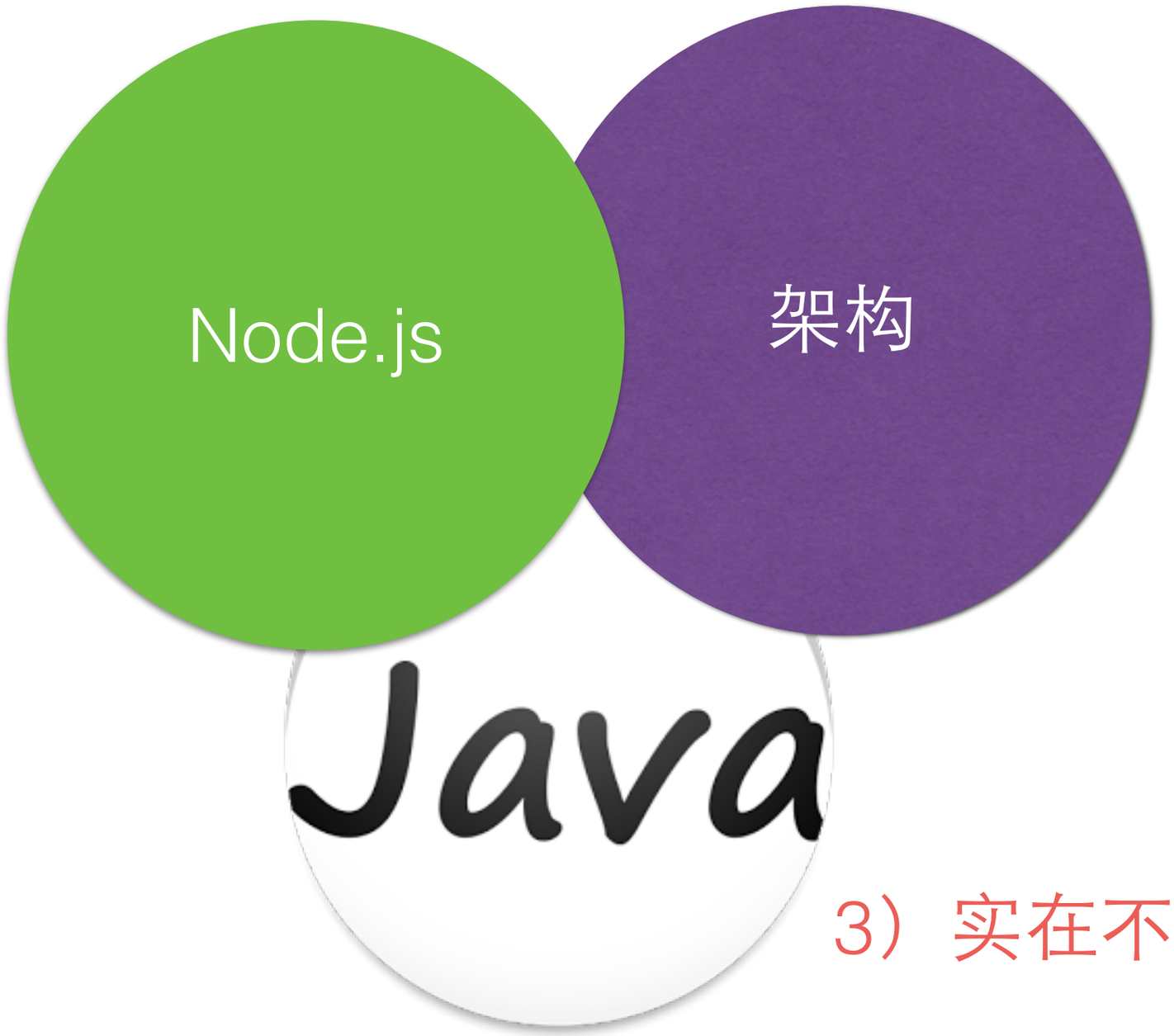
开发大型软件?

- 测试相关 tdd / bdd 测试覆盖率
- 规范化 standard、各种 lint、hint
- 构建相关 gulp、grunt、webpack，大量插件
- 生成器 yo 等
- 包管理工具 npm 足够简单易用

架构平衡

1）在语言层面可以做，那语言层面做

2）如果语言层面搞不定，那就架构层面做



3）实在不够，java补

我们用Node.js做什么?

- API 服务
- 前端（moa-frontend）
- SDK（OAuth Provider）
- 辅助开发 cli 工具

目前进度

- 使用 0.10.38，开发 Moajs 框架，Express / MongoDB
- pm2 部署, 前后端分离
- 阿里云的 slb 负载，alinode 监控
- moa-api, moa-frontend, moa-h5 (未能用)
- 使用 Redis 缓存，Rabbitmq，senaca 作为 RPC

正在建设的

- 使用 kong 作为 API gateway
- consul 做服务发现和配置
- 上 elk 作为日志分析处理
- 使用 docker compose 作为本地开发环境
- 线上 docker

目前的做法

- 小步快走，一次只上一样新技术；
- 形成梯队，即可准备上新东西；
- 善用 npm，实现 3 化：
    - 模块化、
    - 最小化、
    - 服务化

Part 2：快速开发实践

业务边界优化

创业公司有很多可变性，要做的系统也无数，如何保证业务系统的边界是非常难的，我们其实走了很多弯路

静态api理论

时间去哪儿了?

常规

需求 → 设计 → 开发 → 测试 → 交付

改进前

需求 → 设计/ue → 设计/ui → 后端 API → App / h5 / 前端 → 测试 → 交付

相互依赖、扯皮?

并行开发流程改进

常规

需求 → 设计/UE → 开发 → 测试 → 交付

改进后

设计/UI + 静态API

App
h5
前端
后台

项目

api约定

客户端 API 开发总结 - CNod ×

Alfred

https://cnodejs.org/topi...

技术选项

- express or koa
- rails-api
- grape
- sinatra
- go的类似sinatra的框架beego

开发最佳实践

- api的code和msg可以写一个模块，直接继承到api里
- api测试和mock，最好是可以先mock出静态的，可以api和mobile端同时开发
- 根据测试生成api文档

不过我还没有发现好的实现，还请各位指点

api的最佳实践

还是看open api吧

- http://developer.github.com/v3/
- 微博API

回到顶部

moajs/res.api: res.api is an ×

Alfred

GitHub, Inc. [US] https:/...

README.md

res.api

npm package 1.0.11

res.api is an express middleware for render json api , it convention over api form

```
{
  data: {

  },
  status: {
    code : x,
    msg  : 'some message'
  }
}
```

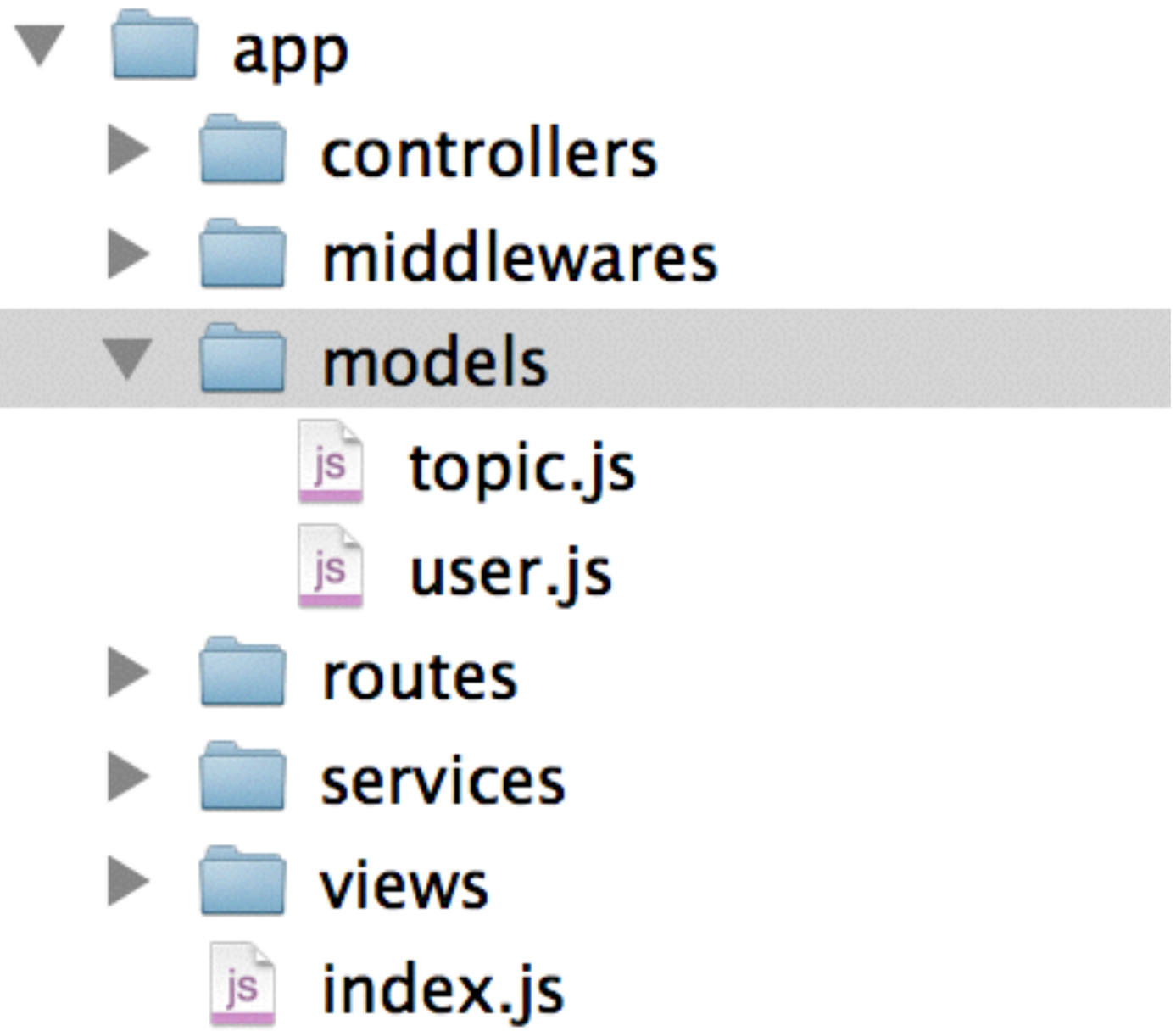
more see at

- 客户端 API 开发总结
- koa.res.api on Koajs

Install

```
npm install --save res.api
```

约定结构


app
controllers
middlewares
models
topic.js
user.js
routes
services
views
index.js

使用npm模块化

- 使用npmjs的private私有模块（目前做法）
- 使用npm的本地模块开发方法（测试和部署都非常快）
- 搭建npm私服（todo）

编写生成器

在web开发里，写了moajs生成器，类似于rails

- moag order name:string password:string

其他开发，如iOS开发里模型校验非常烦,于是写了一个json2objc命令行工具，读取json，生成oc代码，可以节省不少时间

Moajs与前后端分离

- 前端：moa-frontend
    - public下面的采用nginx做反向代理
        - 其他的采用express+jade精简代码（ajax与后端交互）
- 后端：moa-api

- 1）moa生成器
- 2）moa-frontend
    - express
    - jade
    - bootstrap、bootstrap-table
    - jquery
    - gulp
    - nginx

- 3）moa-api

Features

- 自动加载路由
- 支持mongodb配置
- 集成mongoosedao，快速写crud等dao接口
- 自带用户管理
- 使用jsonwebtoken做用户鉴权
- 支持migrate测试
- 支持mocha测试
- 默认集成res.api，便于写接口
- 集成supervisor，代码变动，自动重载
- gulp自动监控文件变动，跑测试
- gulp routes生成路由说明
- 使用log4js记录日志

技术栈

base2(mirco kernel)
mongoose
bluebird
res.api

Part 3：全栈展望

现代Web开发

预编译

模板引擎

CSS预处理器

JavaScript友好语言

平台工具

浏览器

Cordova

Electron

前端发展

4）React组件化、Vuejs（未来趋势）

3）Backbone，Angularjs（当前流行）

2）jQuery、jQuery-ui，Extjs（曾经流行）

1）html/css/js（基础）

构建系统

Grunt

Gulp

Webpack

NPM Scripts

前端开发4阶段

- html/css/js（基础）
- jQuery、jQuery-ui，Extjs（曾经流行）
- Backbone（mvc），Angularjs、Vuejs（当前流行）
- React组件化（未来趋势）、Vuejs

Hybrid开发

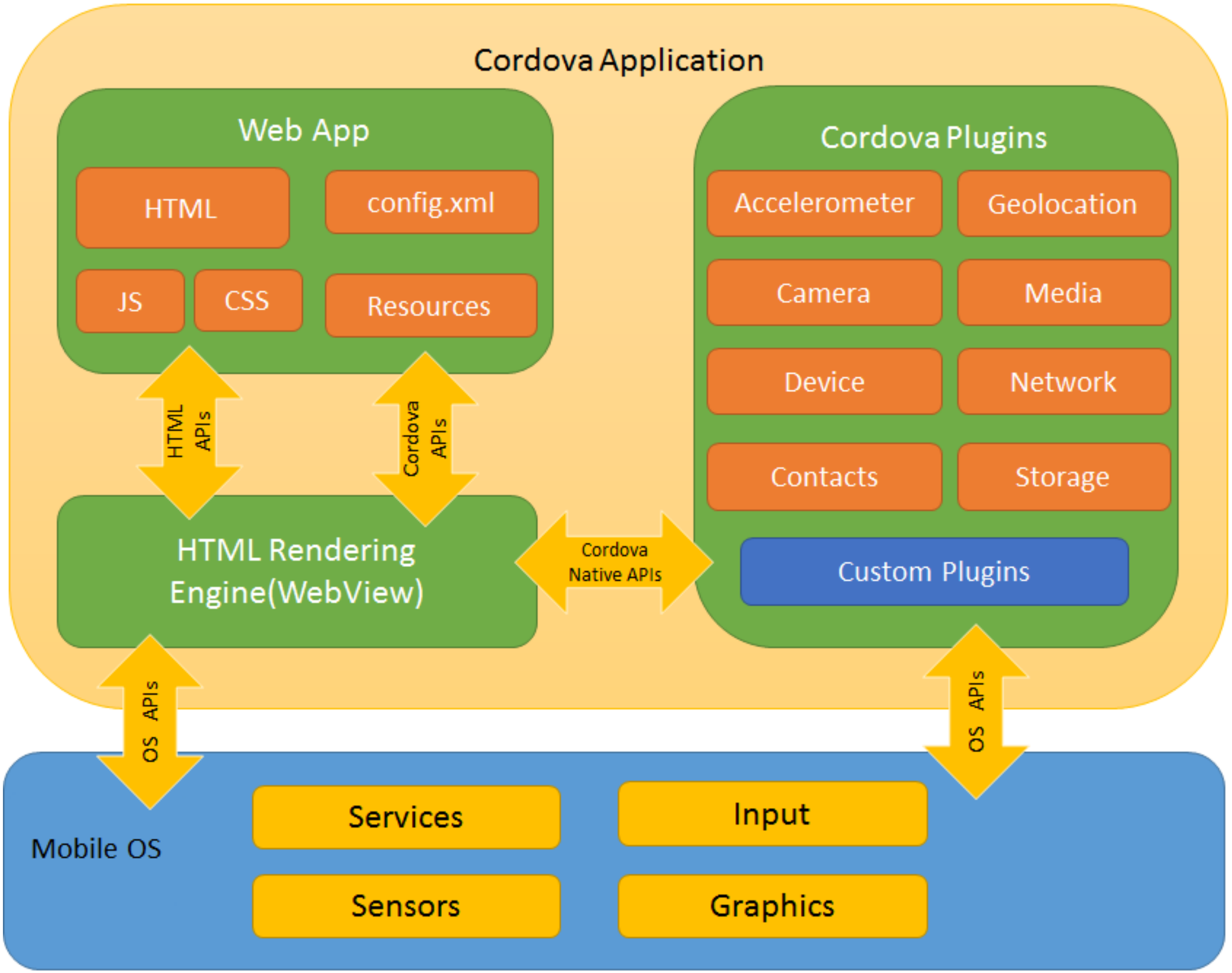
Hybrid混搭开发是指使用html5技术开发的跨浏览器应用，并最终可以将html5.js.css等打包成apk和ipa包的开发方式。它也可以上传到应用商店，提供给移动设备进行安装。

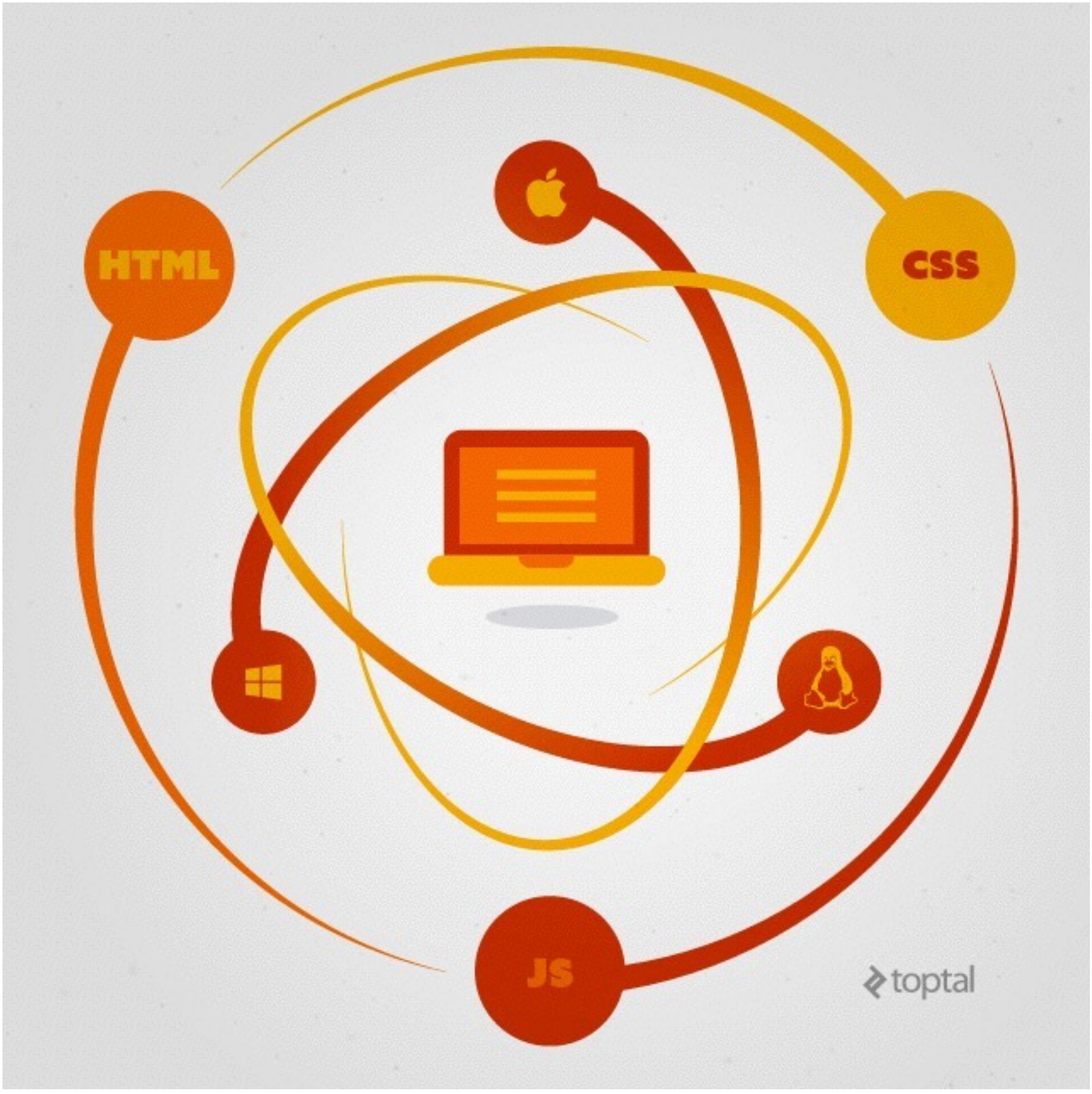
它最大的好处是通过h5开发一次，就可以在多个平台上安装。

- 未来的2点
    - js一统天下（nodejs做后端，传统web和h5使用javasctipt，更智能的工具如gulp，更简单的写法如coffeescript等）
    - h5大行其道（网速变快，硬件内存增长）

跨平台

- C/S 架构到 B/S 架构
- 移动端加壳
- PC 端加壳
- 组件化：统一用法


Cordova Application
Web App
HTML
config.xml
JS
CSS
Resources
HTML APIs
Cordova APIs
HTML Rendering Engine(WebView)
Cordova Native APIs
Cordova Plugins
Accelerometer
Geolocation
Camera
Media
Device
Network
Contacts
Storage
Custom Plugins
OS APIs
OS APIs
Mobile OS
Services
Input
Sensors
Graphics


HTML
CSS
JS
toptal

组件化：统一用法

React 的出现影响最大的是 JSX 的出现，解决了长久以来组件化的问题：

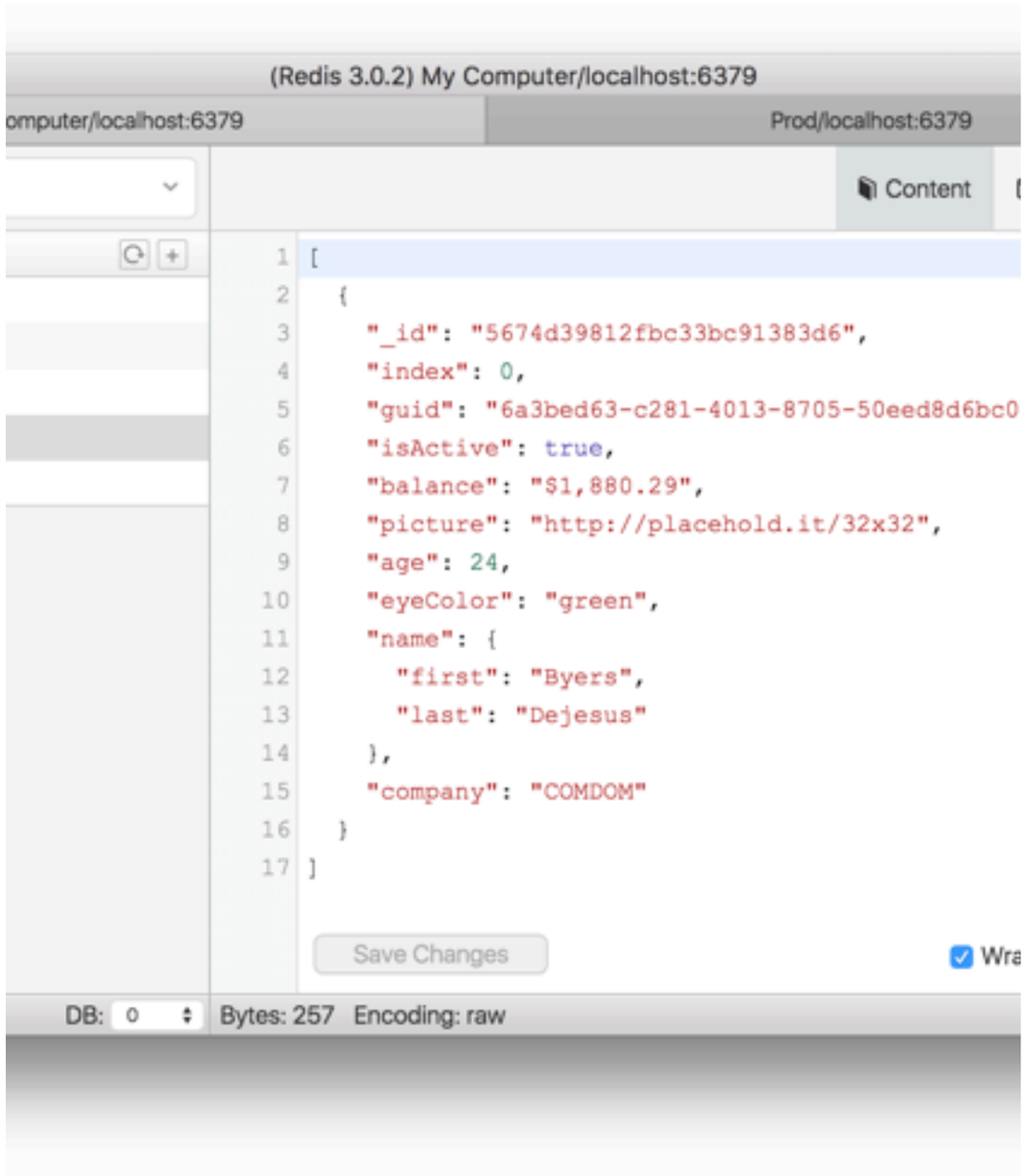
- 我们反复的折腾 JavaScript ，依然无法搞定
- 我们尝试 OO，比如 extjs
- 我们最终还是找个中间格式 JSX

单纯的 React 只是 view 层面的，还不足以应用，于是又有 Redux。核心概念：Actions、Reducers 和 Store，简单点说就是状态控制，然后再结合打包加壳，变成 app 或可执行文件。iOS、Android 上用 Cordova，PC 上使用 Electron。

当下流行玩法 medis

技术点

- 使用 Node.js 模块
- 使用 Webpack 构建
- 使用 React（视图）
  Redux（控制逻辑）
- 使用 Electron 加壳打包



亲，你看到未来了么?

讲了node工具，前端4阶段，hybrid，各种跨平台，目前就是为了介绍Node全栈的各种可能，下面讲一下如何能做到Node全栈?

Part 4: 如何全栈?

全栈核心

- 后端不会的 UI（界面相关）
- 前端不会的 DB（业务相关）

从后端转

做后端的人对数据库是比较熟悉，无论 MongoDB，还是 Mysql、Postgres，对前端理解比较弱，会基本的 Html，Css，模板引擎等比较熟悉

4 阶段循序渐进，build 与工具齐飞

从前端转

- 玩转 npm、gulp 这样的前端工具类（此时还是前端）
- 使用 Node 做前后端分离（此时还是前端）
- Express、Koa 这类框架
- Jade、ejs 等模板引擎
- Nginx
- 玩转【后端】异步流程处理 promise / es6 的 (generator | yield) / es7 (async|await)
- 玩转【后端】MongoDB、Mysql 对应的 Node 模块

一般的前端都非常容易学会，基本 2 周就已经非常熟练了，我的计划是半年后，让他们接触【异步流程处理】和【数据库】相关内容，学习后端代码，就可以全栈了

从移动端转

从**cordova（以前叫phonegap）开始做hybrid开发**

- 只要关注www目录里的h5即可，比较简单
- 如果h5不足以完成的情况下，可以编写cordova插件，即通过插件让js调用原生sdk里功能
- cordova的cli可以通过npm安装，学习npm的好方法
- 学习gulp构建工具

只要入了**h5的坑，其实就非常好办了**

- 然后h5、zeptojs、iscroll、fastclick等
- 然后微信常用的，如weui、vux（vue+weui）、jmui（react+weui）
- 然后可以玩点框架，比如jquery mobile，sencha touch
- 然后可以玩点高级货，ionicframework（基于angularjs、cordova）
- 然后前端4阶段，依次打怪升级

Feature maybe future

春梦?

变革机遇?

总结

- 闲时要有吃紧的心思，忙里要有偷闲的乐趣
- 人生不只有代码，虽然它能让我快乐
- 每日精进，重家庭，重社交
- 少抱怨，多思考，未来更美好

招聘

- sang@aircos.com 坐标天津空港

Q&A

少抱怨，多思考，未来更美好。有的时候我看的不是你一时的能力，而是你面对世界的态度。

console.log('The End, Thanks~')