美团网
meituan.com

多快好省

的前端开发实践

刘敬威
前端高级工程师
ljw.me

刘敬威

Twitter: @th507
URL: ljw.me

Jingwei “John” Liu

以前做：mobile web app, HTML 5，Objective-C

在美团：性能优化、框架开发

多快好省

美团的三高三低

高品质、低折扣，高效率、低成本，高科技、低毛利

多快好省?

对于前端来说：

完成更**多**业务、更**快**的迭代产品

产出质量更**好**的代码，节**省**人力成本（开发、运营）

每天面临各种**挑战**

实现精致的设计

http://sandbox.juan-i.com/longshadows/

代码复用?

没那么容易

这个功能两个页面都用，CSS 放哪儿

这个页面新添的功能没有样式，忘了加 CSS 了

CSS 的副作用

只用某模块的一个方法，就得加载整个 JS

算了，

我另外写一个吧

```
Y.mt.www.Movie = {
    /**
     * 初始化电影详情的展开和折叠
     * @method initMovieDesc
     */
    initMovieDesc: function (isLogin, loginRef, movieId) {
        var ndContainer = Y.one('.J-movie-container'),
            ndShortDesc = Y.one('#J-short-movie-desc'),
            ndLongDesc = Y.one('#J-long-movie-desc'),
            WISH_AND_SCORE_ACTION = '/movie/wishandscore/' + movieId;

        ndContainer.delegate('click', function () {
            2 行: if (ndLongDesc) ndLongDesc.show();
        }, '#J-unfold-movie-desc');

        ndContainer.delegate('click', function () {
            2 行: if (ndLongDesc) ndLongDesc.hide();
        }, '#J-fold-movie-desc');
```

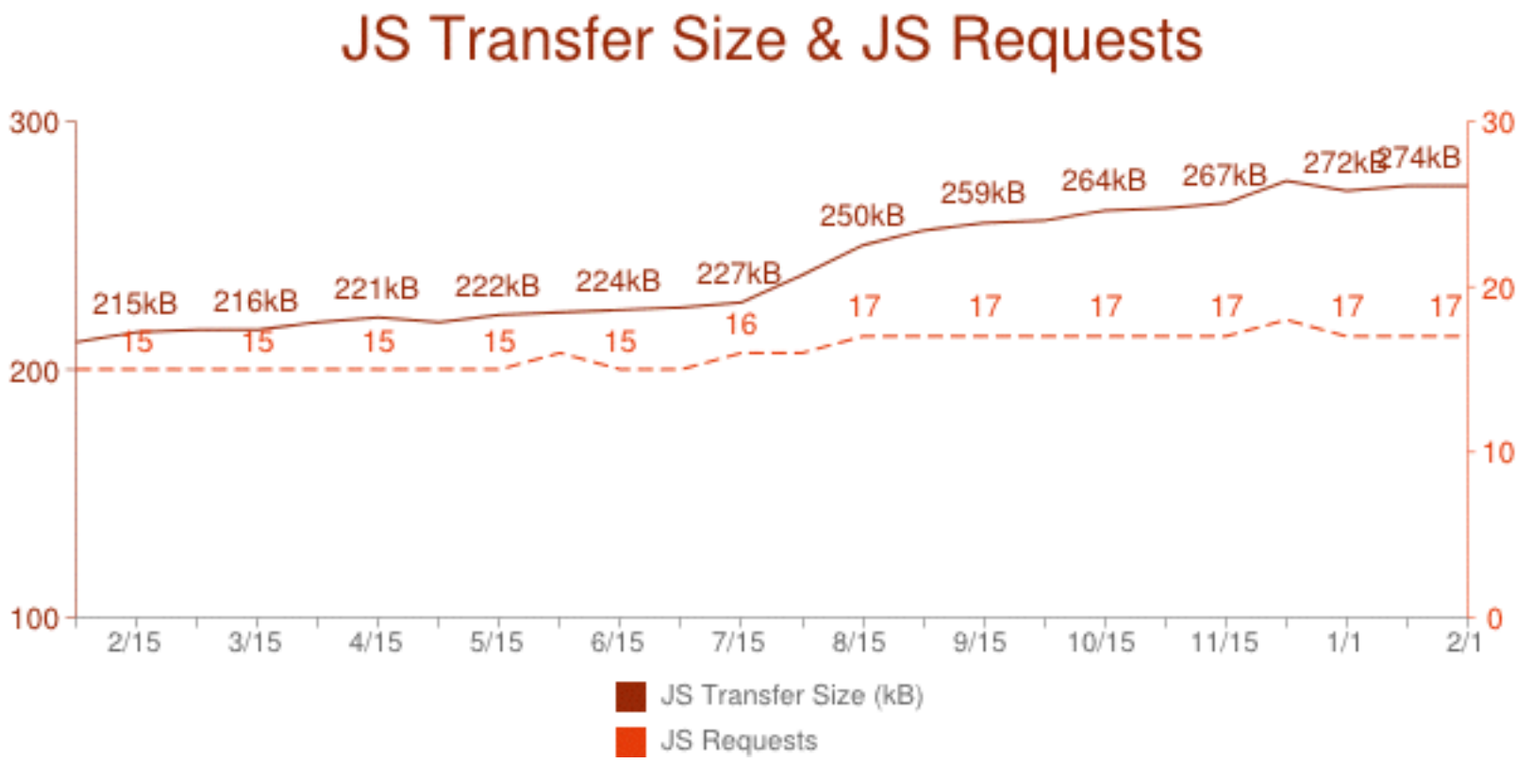
删除一行代码都很可怕

性能监控并不容易

Velocity Conference, Beijing, 2014

美团通用性能监控平台和WEB性能分析框架

代码越来越多



http://httparchive.org/trends.php#bytesTotal&reqTotal

解决方案

实现精致的设计和交互

Sass + Iconfont + UI Guide + COSUI ...

保证开发和迭代速度

Component + Node.js

保证代码质量

unit testing

提高用户访问性能

实时的数据分析与报警 (性能平台)

解决方案

实现精致的设计和交互

Sass + Iconfont + UI Guide + COSUI ...

保证开发和迭代速度

Component + Node.js

保证代码质量

unit testing

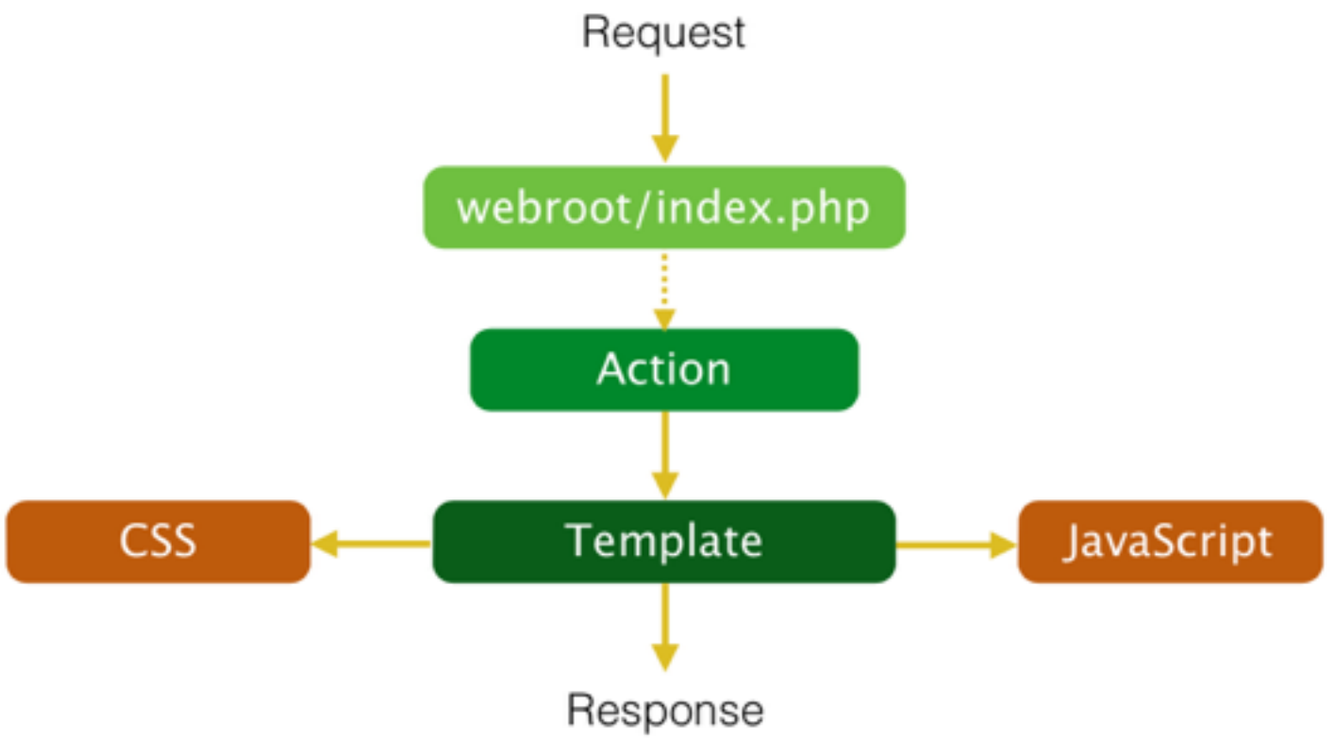
提高用户访问性能

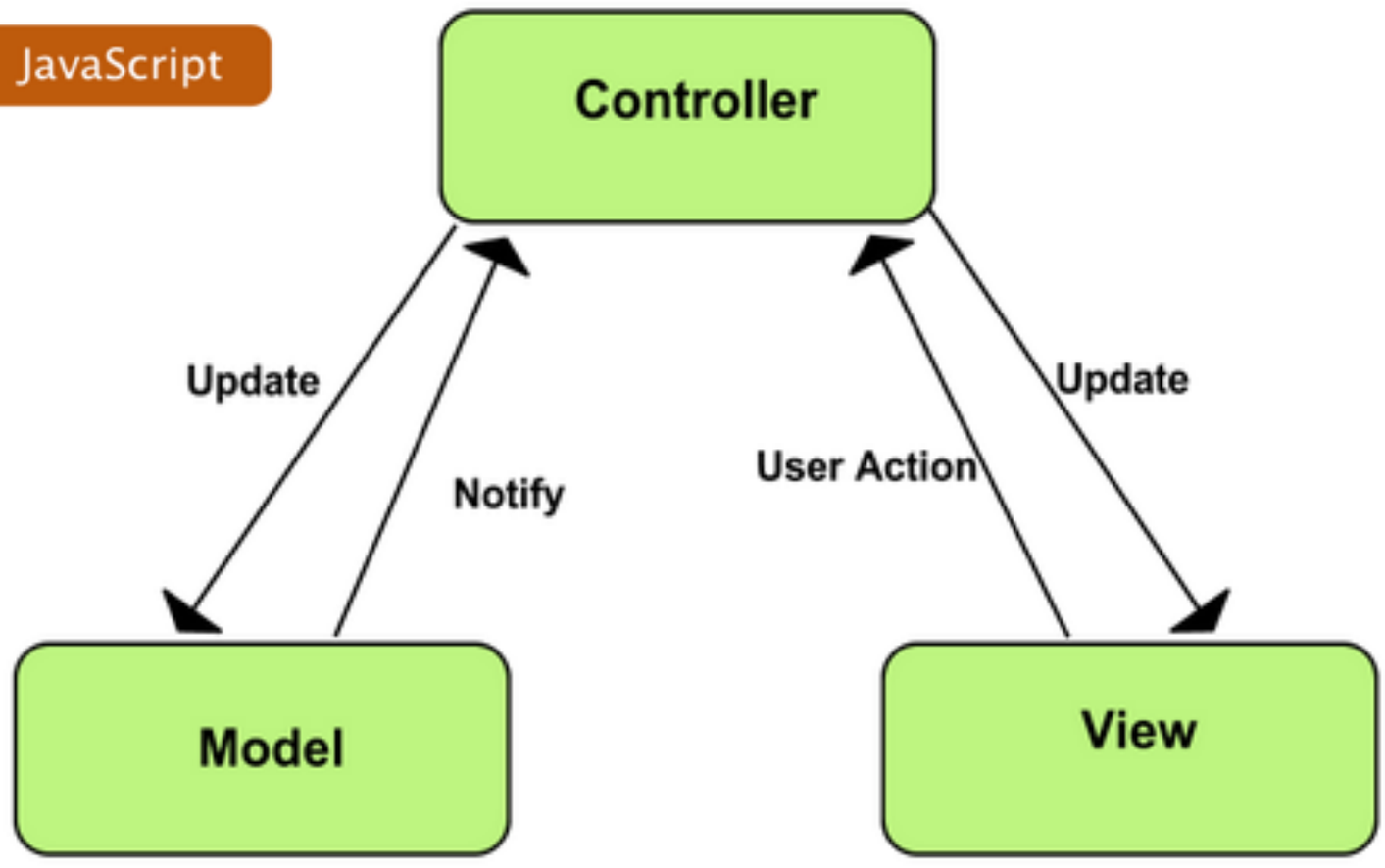
实时的数据分析与报警 (性能平台)

开发和迭代效率的瓶颈

回顾一下以前怎么做的

页面调用


Request
webroot/index.php
Action
CSS
Template
JavaScript
Response


Controller
Update
Notify
User Action
Update
Model
View

复杂度++

现在	以前
页面	网站
≈	
模块	页面

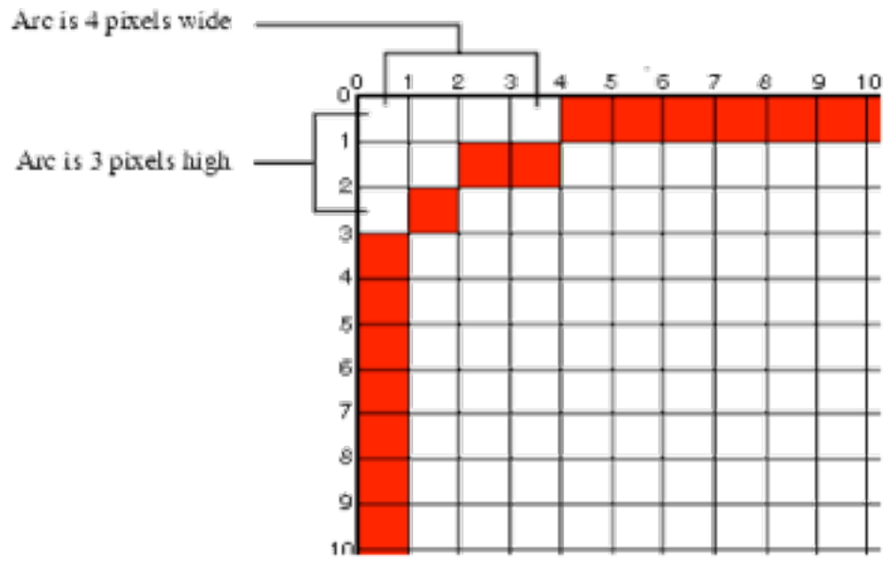
组件化

封装部分细节

为什么要组件化?

回忆一下十年前是怎么做圆角的

- 按圆角半径添加一些 1px 高的元素
- 计算上述元素的 margin



现在是怎么做圆角的

- border-radius: 4px

```
▼<div id="nifty">
  ▼<b class="rtop">
      <b class="r1"></b>
      <b class="r2"></b>
      <b class="r3"></b>
      <b class="r4"></b>
   </b>
```

封装

很多时候你不需要知道那么多细节

组件化就是封装

- 将具备独立展示、交互的页面模块抽离为组件
- 对前端资源做统一管理
- 组件中将生产展示数据过程、渲染内容过程分离

组件化 1.0

组件化

- 每个文件各司其职
- 只需要告诉页面“使用 search-box”组件
- 不用到处找散落在不同页面的代码，提高了维护和开发的效率
- 每个组件有一个专门的 test 页面，可以单独进行开发测试

```
search-box/
├── search-box.scss
├── search-box.js
├── search-box.tpl
└── search-box.php
```

demo

```
// 调用组件
View::useComponent('seo-tag', array('content' => $taghtml));
```

对前端资源做统一管理

多快好省

组件化 1.0

- 需要写的代码还是不少
- 各组件之间加载独立，性能不够好
- 启动时机各不同，怎么统一管理？手工初始化复杂
- 组件之间没有通讯机制，依赖元素 id/className

怎么改进？

图片插入页面时浏览器是怎么处理的?

图片加载的默认行为

1.扫描、收集需要下载的资源

2.按下载优先级排序，等待可用连接

3.下载

4.渲染 data

5.展现

组件化 2.0

组件化 2.0

- **以元素 / 模块为中心设计**

  把配置写在元素 data-* 上

- **合并请求，提高性能**

- **分工明确、低耦合**

  管理器：分发事件

  组件：用事件冒泡上报信息

  组件之间用事件通讯

hubcss	CSS 地址
hubmodule	JS 模块名称
hubnamespace	JS 命名空间
huburl	AJAX 请求地址
hubconfig	JS 需要的额外参数

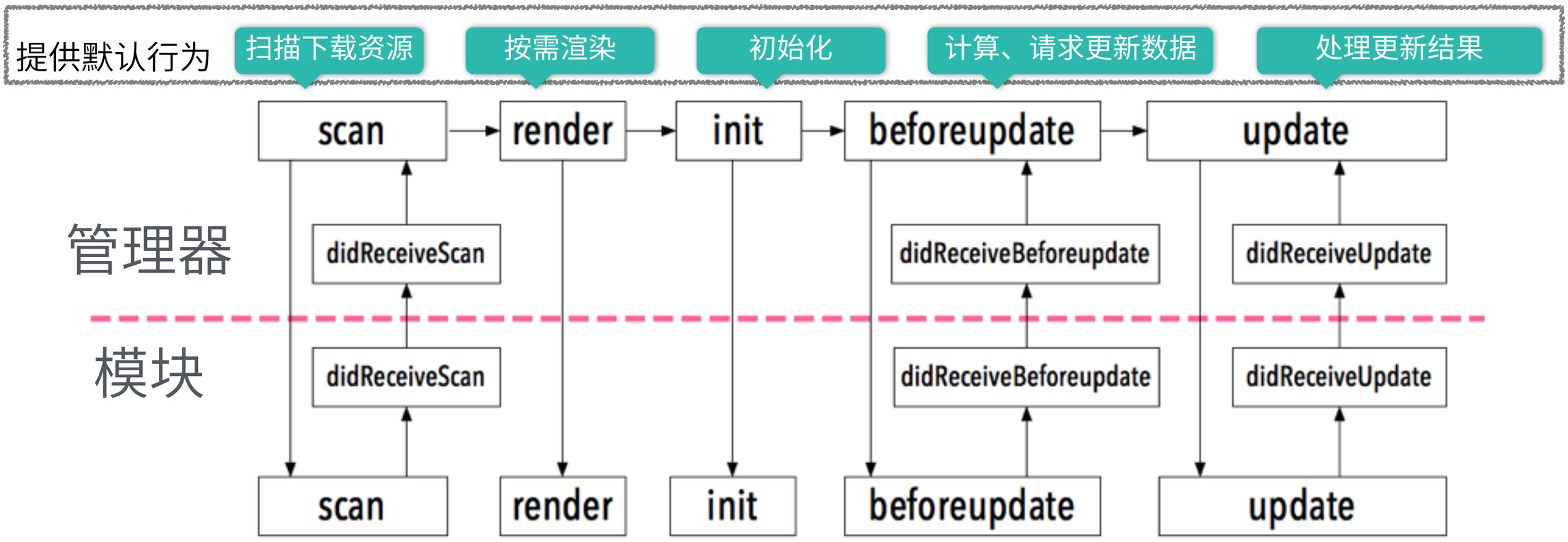
demo

```
<div class="J-hub"
     data-huburl="deal/toprecommend"
     data-hubconfig="3638724"></div>
```

1. 没有找到 JS 模块信息
2. 没有找到命名空间，使用默认元素行为
3. 执行默认的 render 函数
4. 执行默认的 init (空函数)
5. 执行默认的 beforeupdate 得到需要发送AJAX的数据
6. 发送 AJAX 请求 /deal/toprecommend/3638724
7. 执行默认的 update，将返回的数据直接填入该节点

不需要一行 JS

默认行为



- 提供一系列默认行为，并允许按需覆盖，减少代码
- 单个元素可以任意时刻暂停和继续自身的初始化
- 元素之间、元素和管理器之间只通过事件通讯
- 任意时刻插入页面的元素会自动触发通用函数的执行
- 元素和资源、数据绑定，上下线时一体操作
- 资源自动合并请求加载并缓存
- 异步请求合并

组件化 2.0

actor 模型　处理组件之间的通讯

Promise 管理　并行加载静态资源，合并异步请求

默认行为　可以统一加入**模版引擎**，默认惰性加载

暂停和继续　各个组件自己决定加载时机

自动启动　只要放到页面里就可以

多快好省

模版引擎

JS 和 PHP，一个都不能少

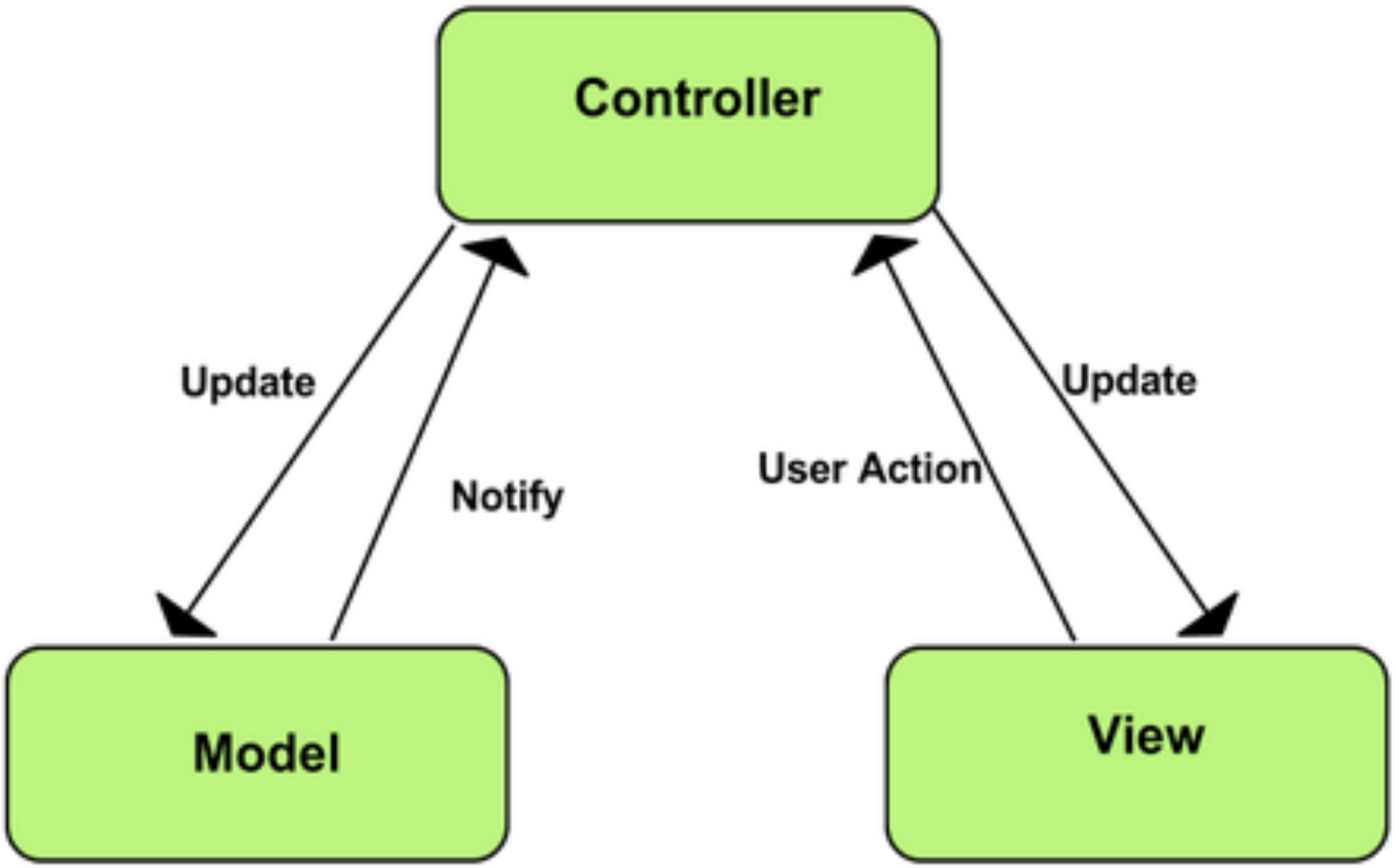
数据获取

前端自己拿业务数据

MVC

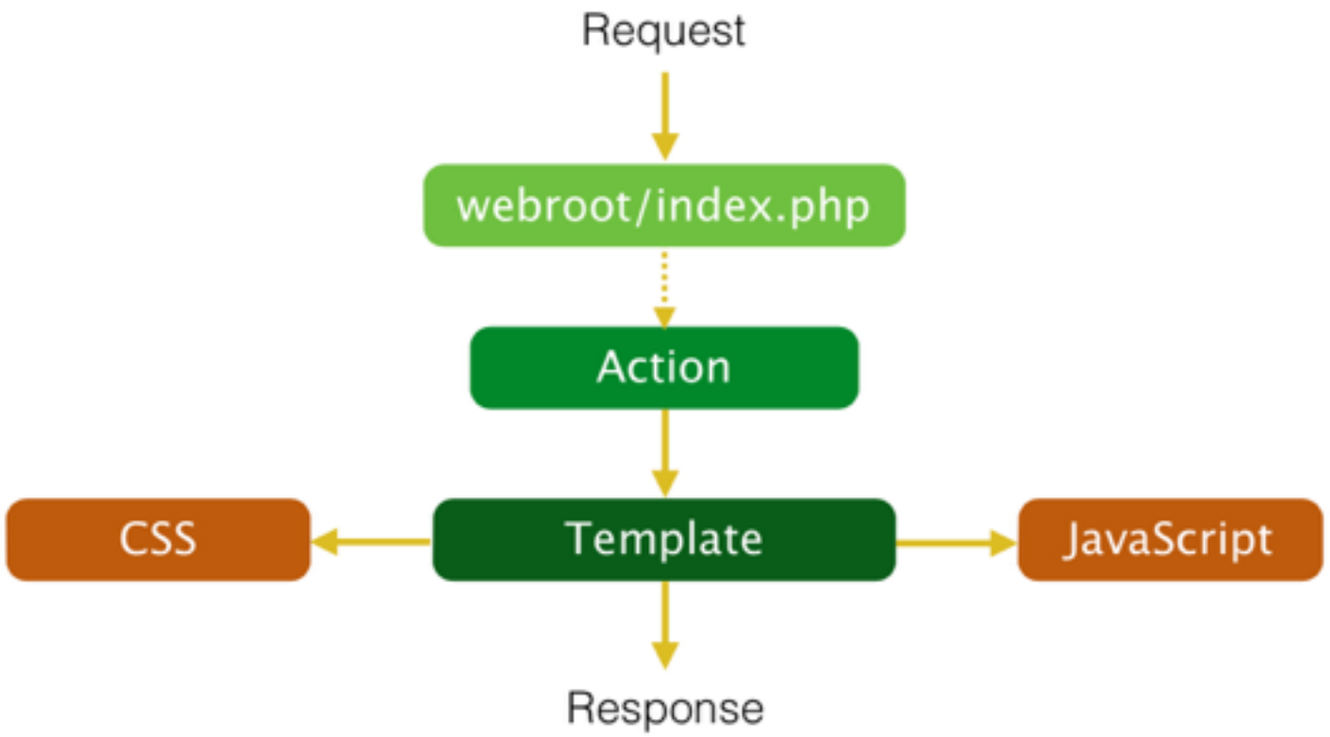
框架结构，尚能饭否?

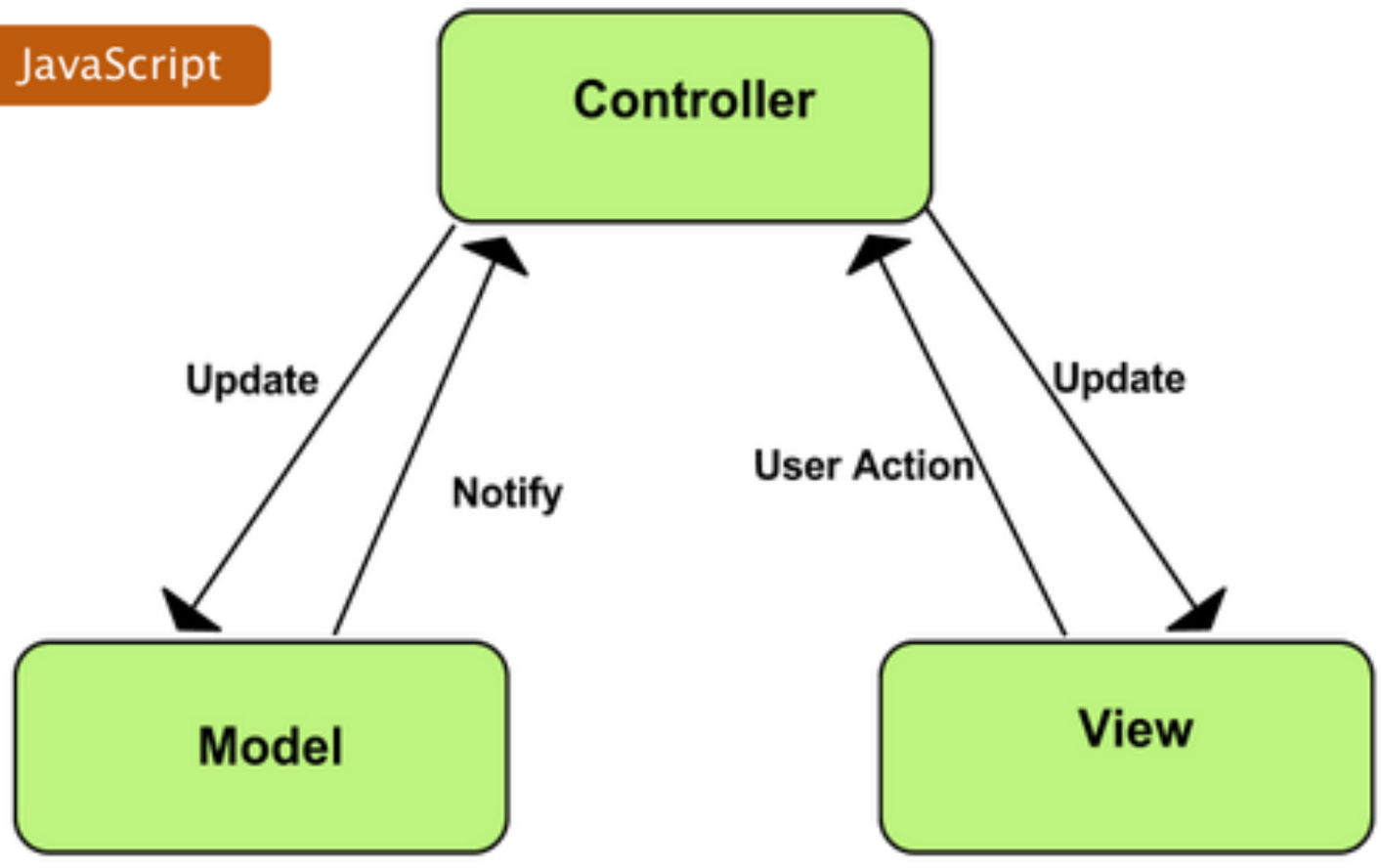
页面级别的 MVC



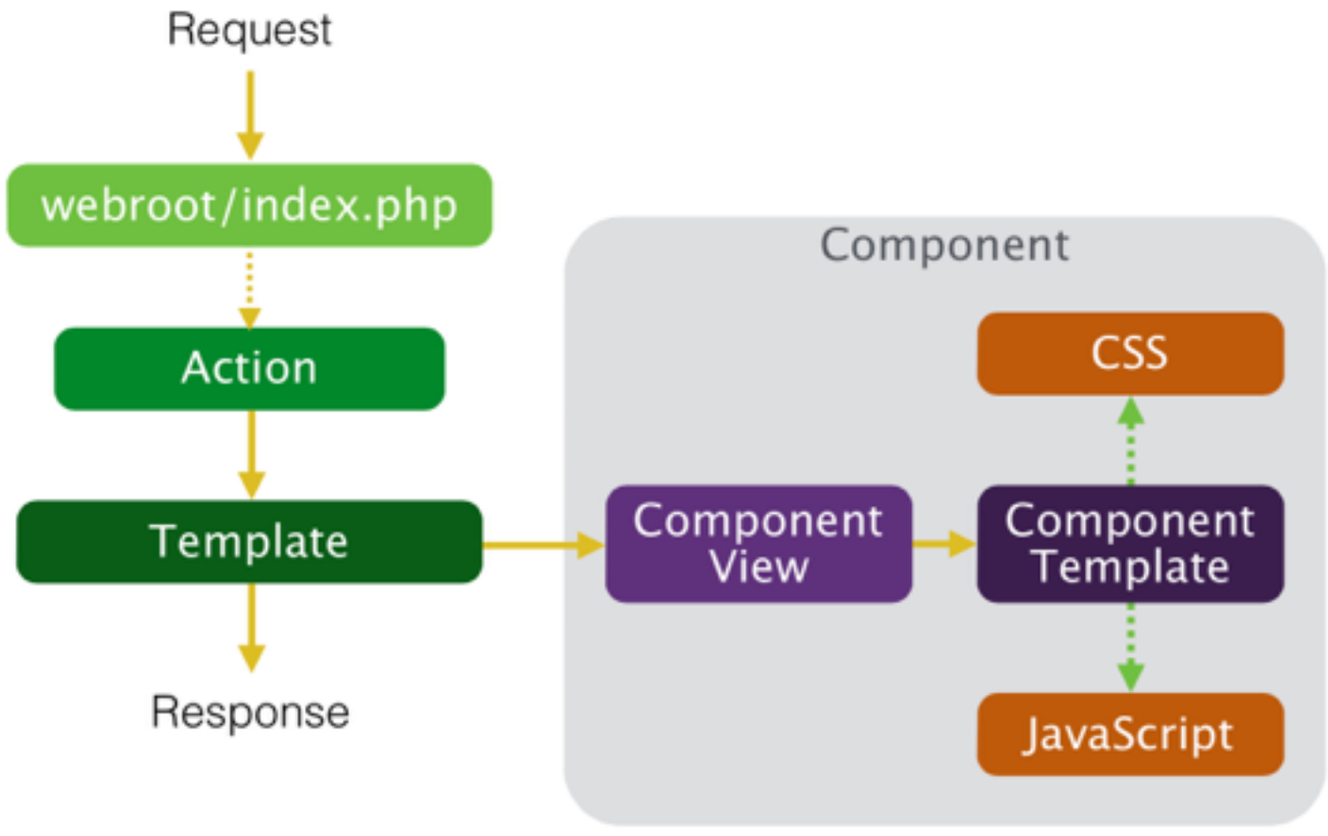
https://developer.chrome.com/apps/app_frameworks

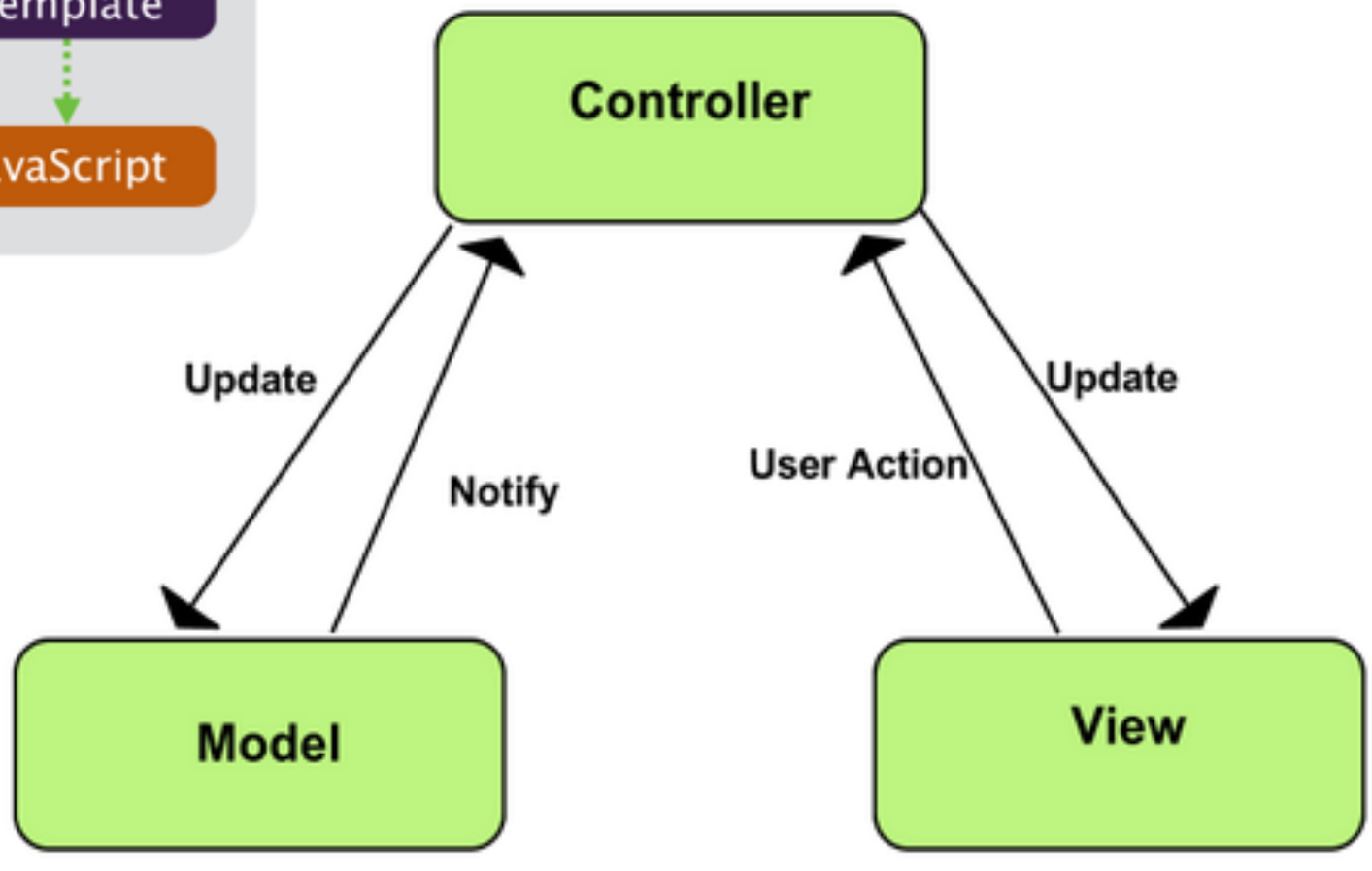
组件化之前的 MVC


Request
webroot/index.php
Action
CSS
Template
JavaScript
Response

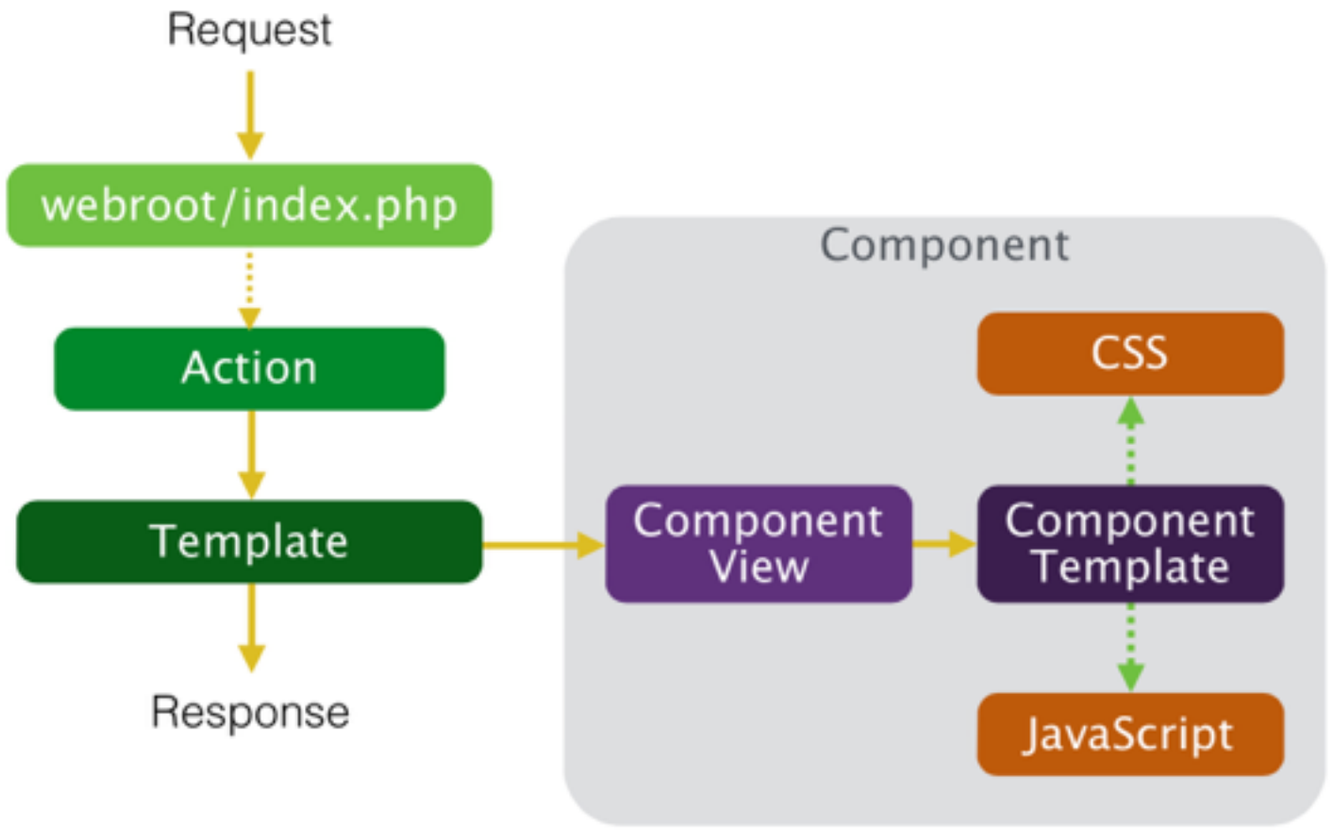

Controller
Update
Notify
User Action
Update
Model
View

组件化 + MVC


Request
webroot/index.php
Action
Template
Response
Component
CSS
Component View
Component Template
JavaScript

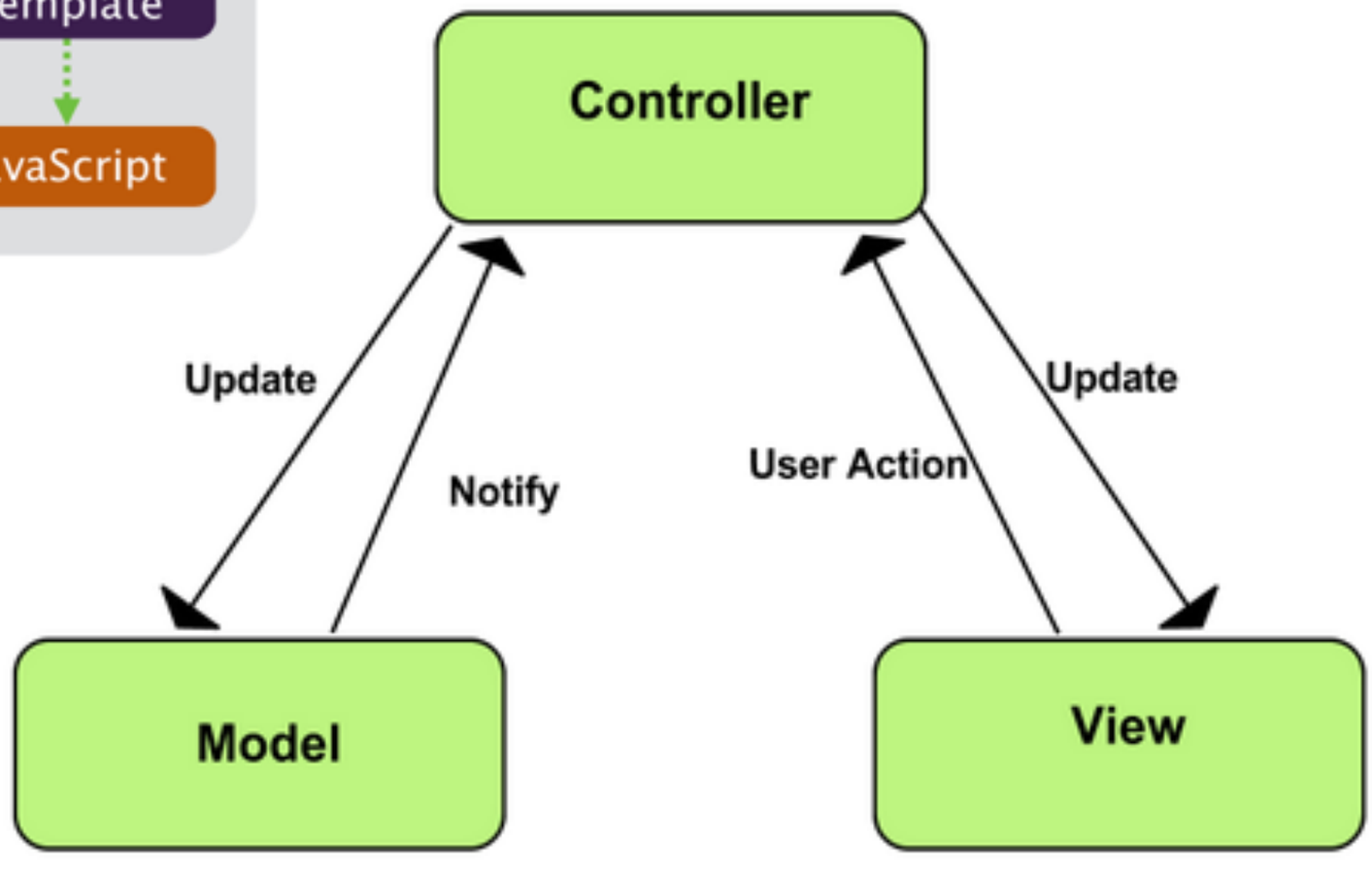

Controller
Update
Notify
User Action
Update
Model
View

后组件化的 MVC?



数据获取还没有封装

做模块的MVC



页面级别的 MVC

逐渐会成为敏捷开发的下一个瓶颈

全栈开发

http://dobee.deviantart.com/art/Island-Road-117063469

彻底的组件化

无需依赖页面数据，实现组件级别 MVC

聚合方式

CSS 脱离 HTML 是没有意义的

JS 脱离 HTML 是没有意义的

现在的组件化

这是开始，不是结束

重新定义聚合方式

CSS 脱离 HTML 是没有意义的

JS 脱离 HTML 是没有意义的

HTML 脱离数据也是没有意义的

No strings attached

封装

无需依赖页面数据，实现组件级别 MVC

封装数据源

```
// datasource for Foo
function* Foo(name) {
    var result = yield request("http://path/to/foo?name=" + name);

    this.rawData = JSON.parse(result.body);

    return this;
}

Foo.prototype = {
    getA: function() {
        return this.rawData.A;
    },
    isBar: function* () {
        var bar = yield request("http://path/to/bar");

        bar = bar.body;

        return this.rawData.bar === bar;
    }
};
```

提供封装，降低学习成本

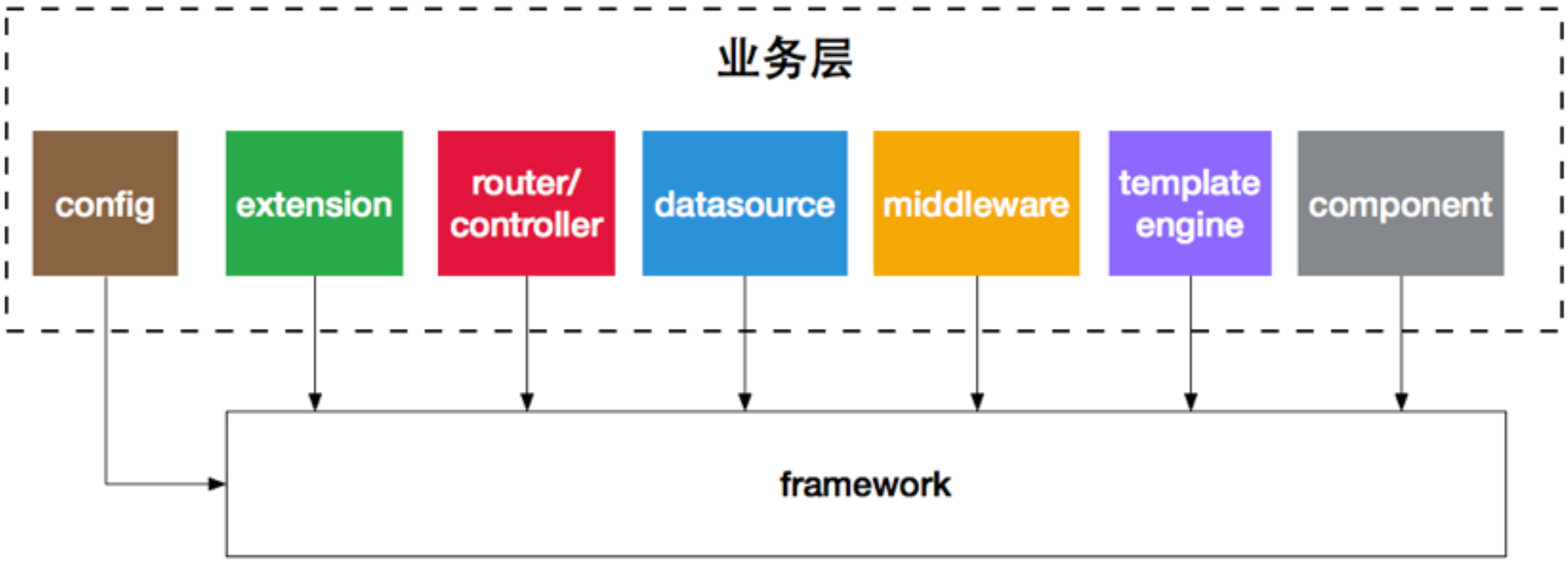
使用数据源

```
// using it from the framework
var $new = require("data-layer");
var foo = yield $new.Foo("Alice");
// this instance of Foo is automatically cached

var isBar = yield foo.isBar();

if (isBar) {
    …
}

…

// returning cached value
var anotherElementUsingFoo;
// catch async error with a traditional try/catch
try {
    anotherElementUsingFoo = yield $new.Foo("Bob");
} catch (e) {}
```

使用 yield “消灭”异步回调函数

全栈组成

- 基于 JavaScript (ES6) / Node.js
- 框架本身和业务隔离，方便单独测试
- 业务层子系统职责单一，可以单独替换
- 前后端开始有一个隔离，减少耦合



全栈在美团

- **前后端适当分离**：提高效率；方便测试
- **业务计算部署规范**：固定业务逻辑后移，临时patch和活动在前端层进行
- **以元素为中心设计**：自动缓存数据，减少重复计算的开销，促进功能模块封装
- **框架轻量化**：合理分层，各司其职
- **保留扩展性**：方便未来技术演进

多快好省

今天你可以带走什么？

http://www.kitsilano.ca/2014/01/31/5-tasty-takeout-options-in-kitsilano/

http://commons.wikimedia.org/wiki/File:Lego_Technic_8837_-_Pneumatic_Excavator.jpg

为满足**可扩展性**而设计

而不是为了满足功能

下线和上线效率一样重要

敏捷开发更核心的是快速迭代，而不是快速原型

减少技术负债

日常开发不是持续 hackathon

http://pixabay.com/p-30148

为什么比怎么做更重要

盲目跟风不一定适合你和你的团队

web worker

#perfmatters

service worker

#nodejs

web socket

boilerplate

#fullstack

NoSQL

#responsive

业务

前端

体验

SEO

animation

remote debugging

offline first

material design

hybrid app

#polyfill

fast prototyping

CSS3

web worker
#perfmatters
service worker
#nodejs
web socket
boilerplate
#fullstack
NoSQL
#responsive

业务

前端

体验

SEO
animation
remote debugging
offline first
material design
hybrid app
#polyfill
fast prototyping
CSS3

用技术支撑可持续发展

保障代码质量

We are hiring.

#perfmatters

web worke

service worker

#nodejs

web socket

boilerplate

#fullstack

NoSQL

#responsive

Thank you.

业务

体验

SEO

animation

remote debugging

offline first

material design

hybrid app

#polyfill

fast prototyping

CSS3