那些年，我们追过的

MV*框架😢

IMWeb资深前端开发工程师

负责IMWeb部分技术架构研发工作

互动视频

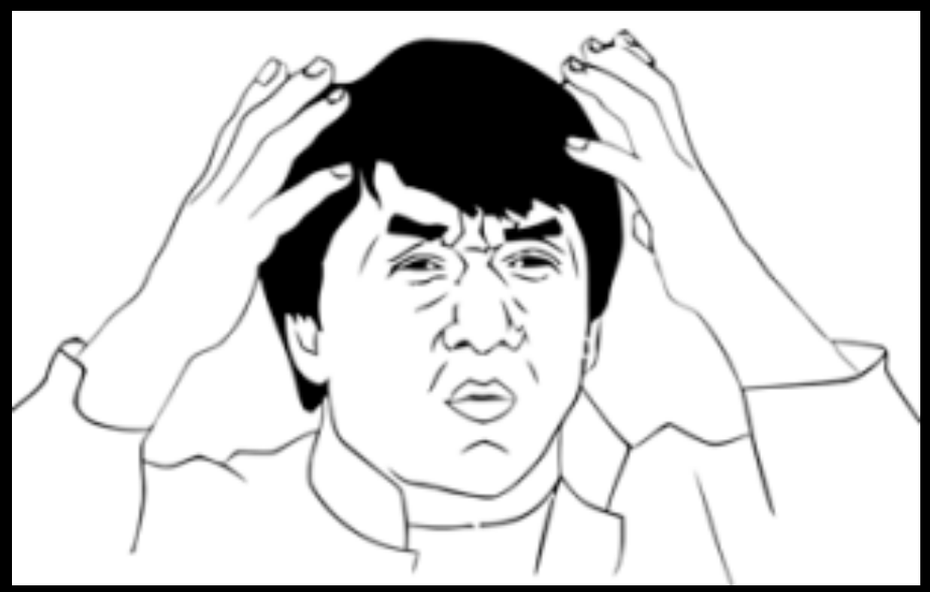
在线教育

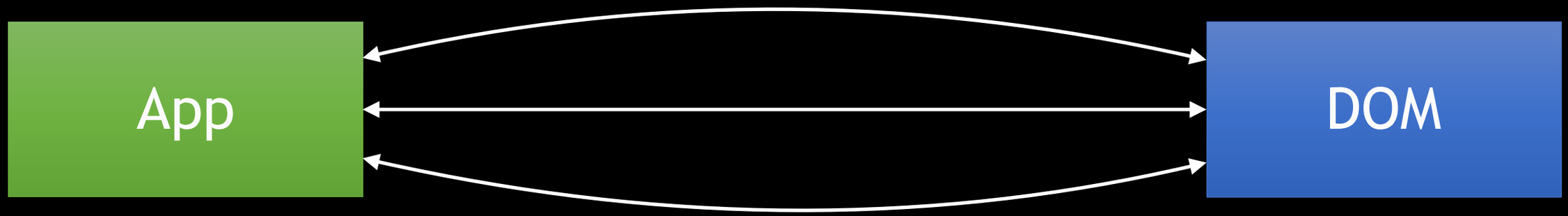
……

Summary

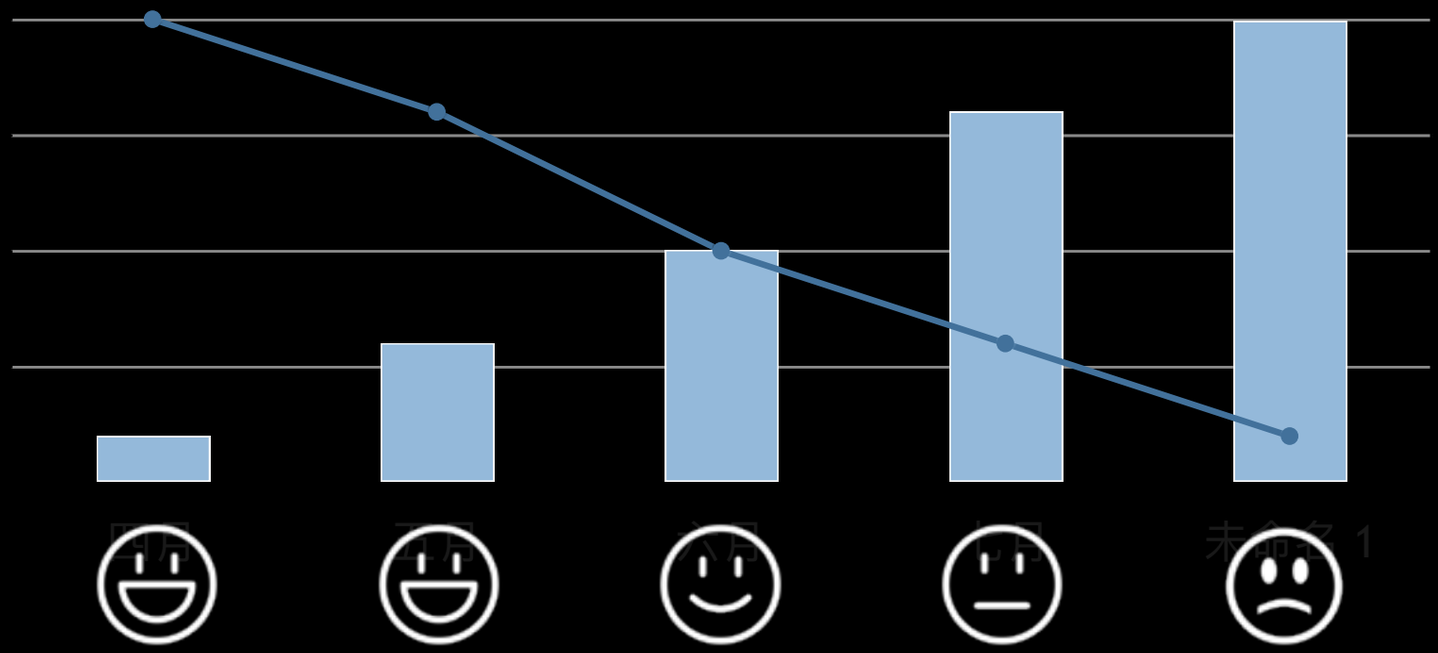
- 那些年让我们崩溃的代码
- Angular & React & VueJS给我们带来了什么?
- 海量业务场景下挑战
- IMWeb的解决方案

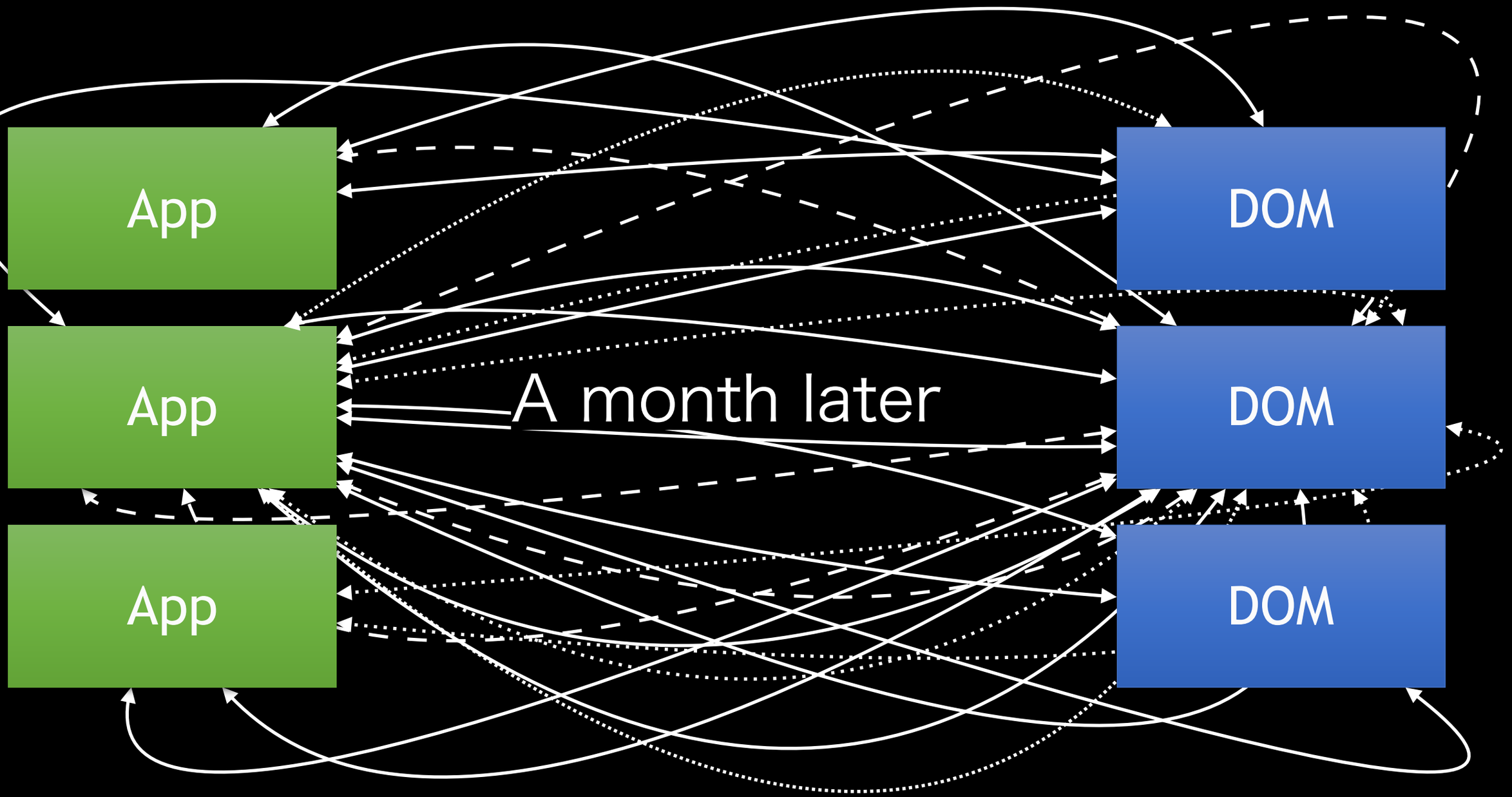
那些年，让我们崩溃的代码.....




App
DOM

A year later

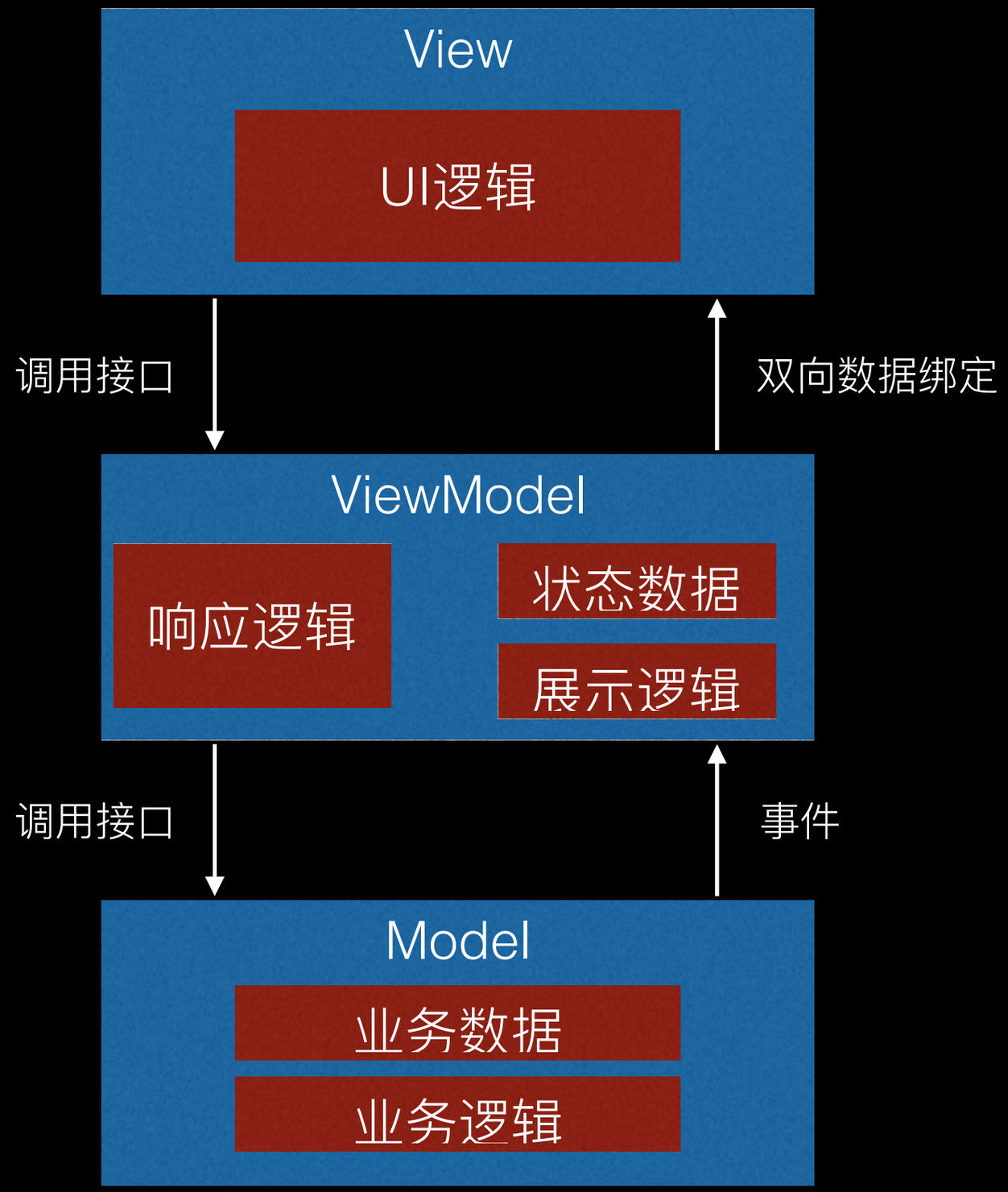


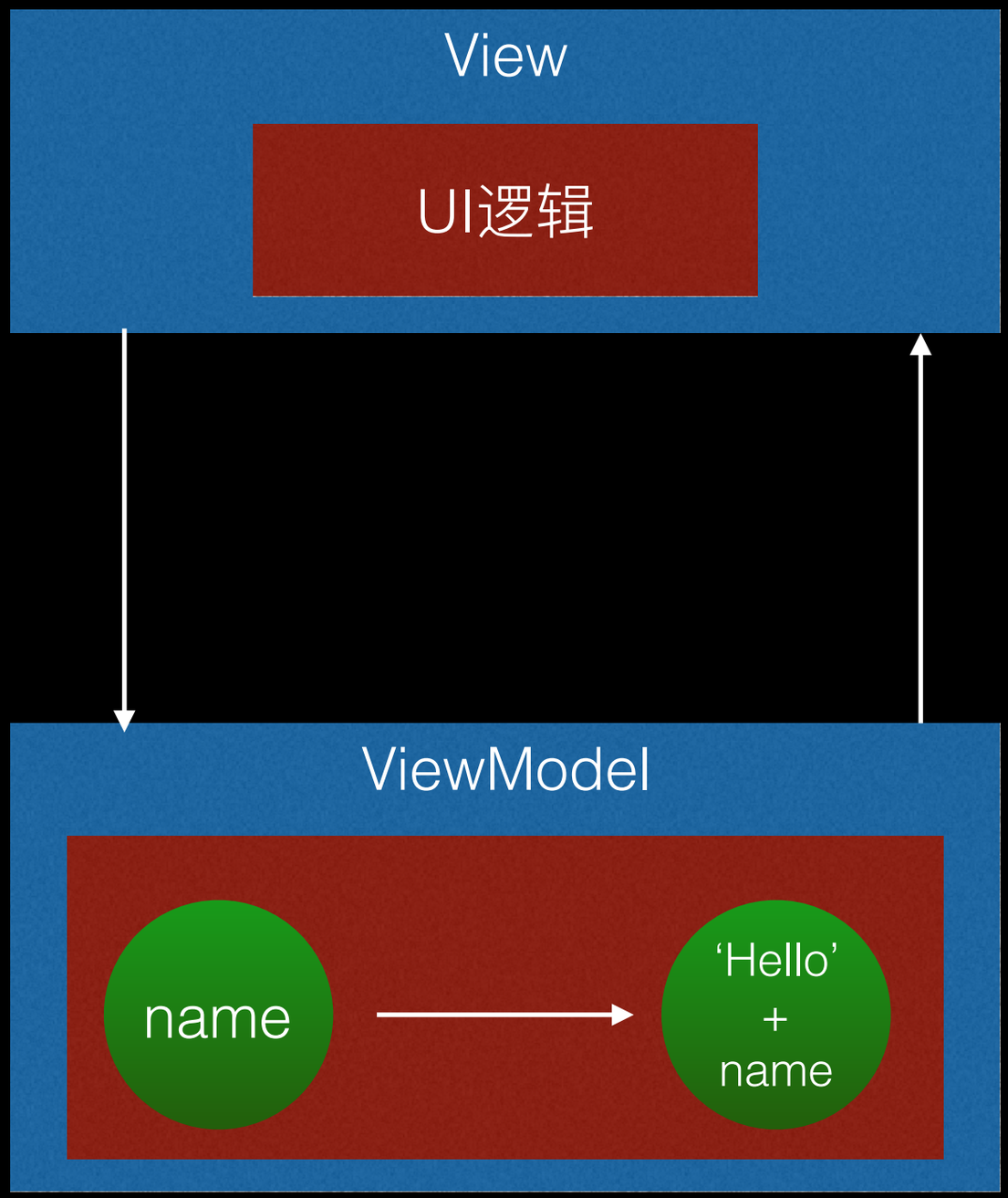

App
App
App
DOM
DOM
DOM
A month later

Answer: f(state) = UI


ANGULAR

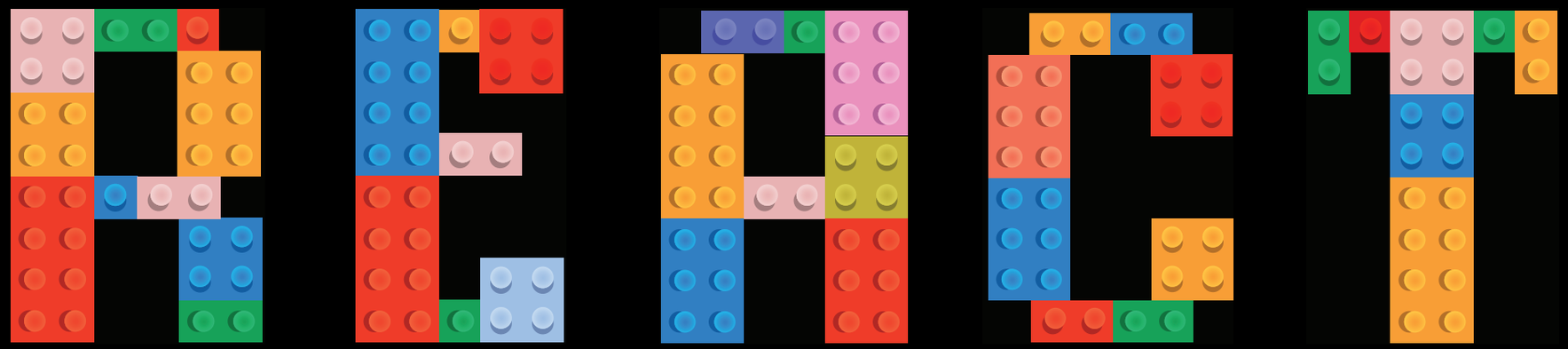
MVVM


View
UI逻辑
调用接口
双向数据绑定
ViewModel
响应逻辑
状态数据
展示逻辑
调用接口
事件
Model
业务数据
业务逻辑

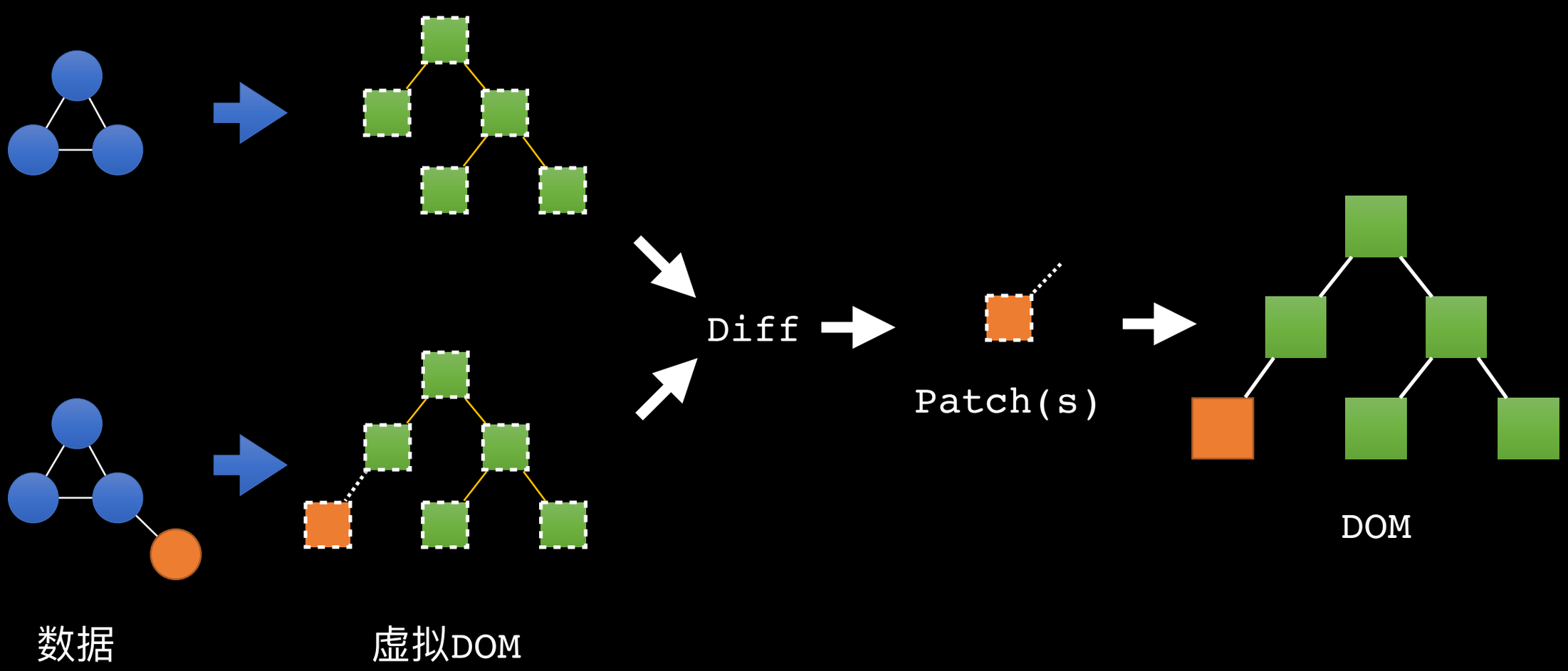

View
UI逻辑
ViewModel
name
‘Hello’
+
name

DI (Dependence injection)

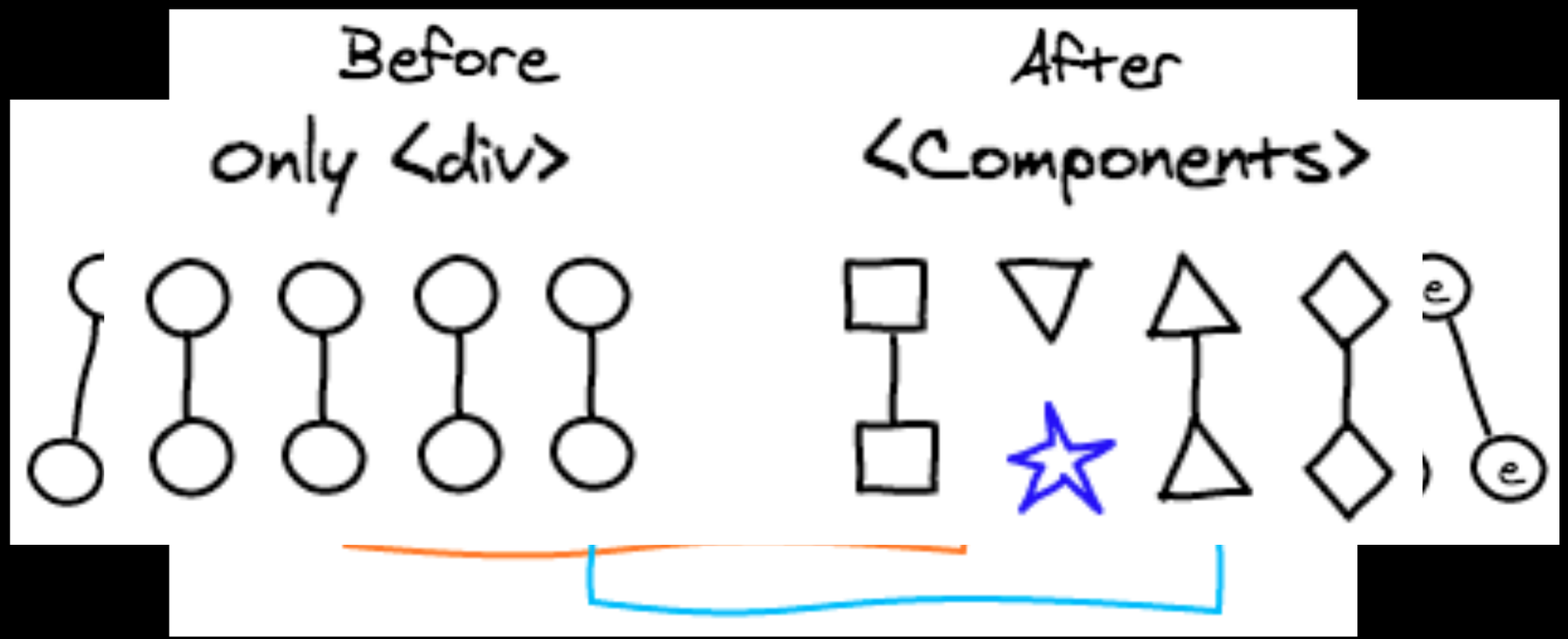
```
1 var pop = require('pop-it');
2
3 pop('aPoppy', 'This is it!');
4
5 pop(function (aPoppy) {
6   // aPoppy = 'This is it!'
7 })();
```



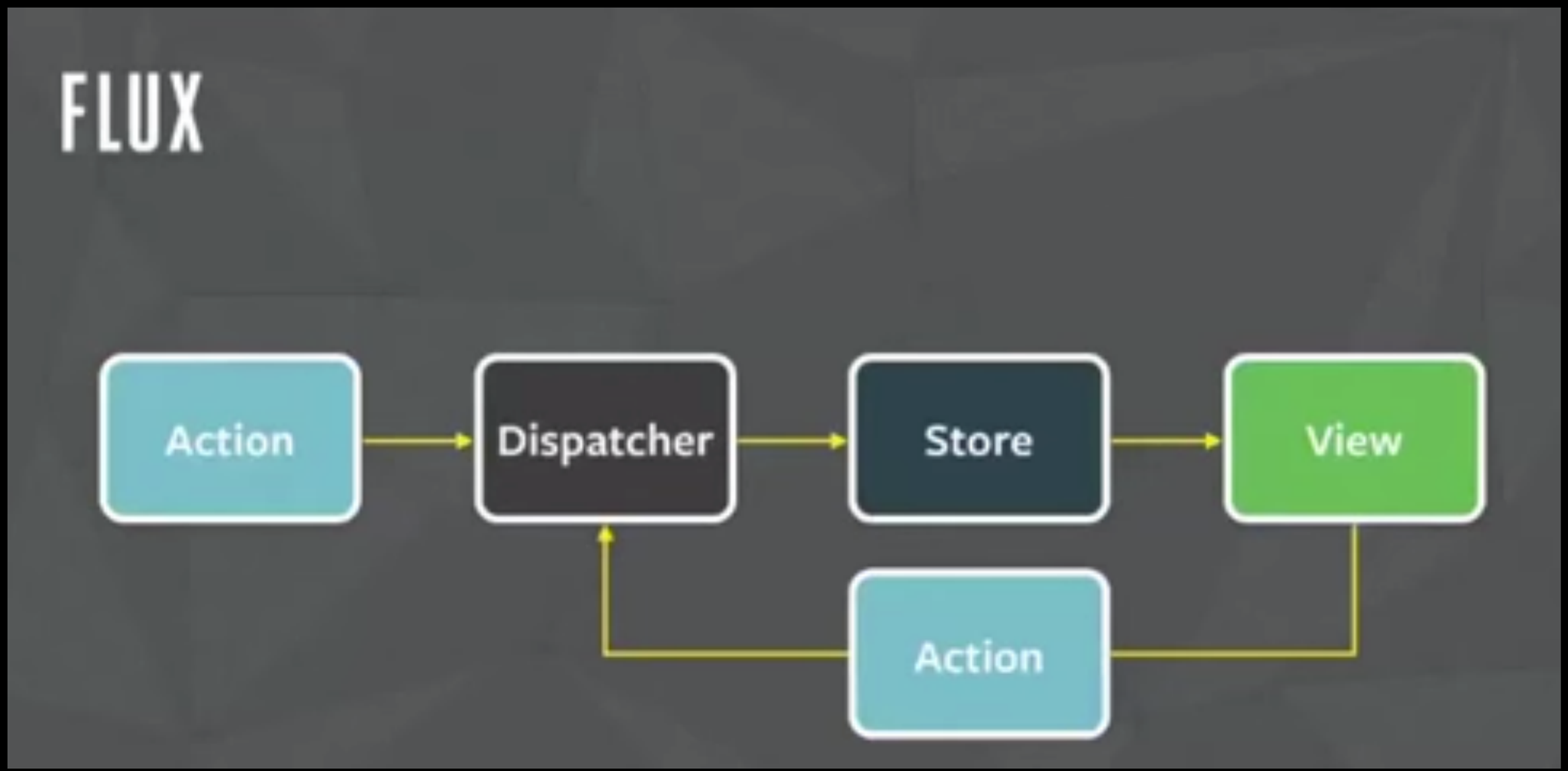
Virtual DOM



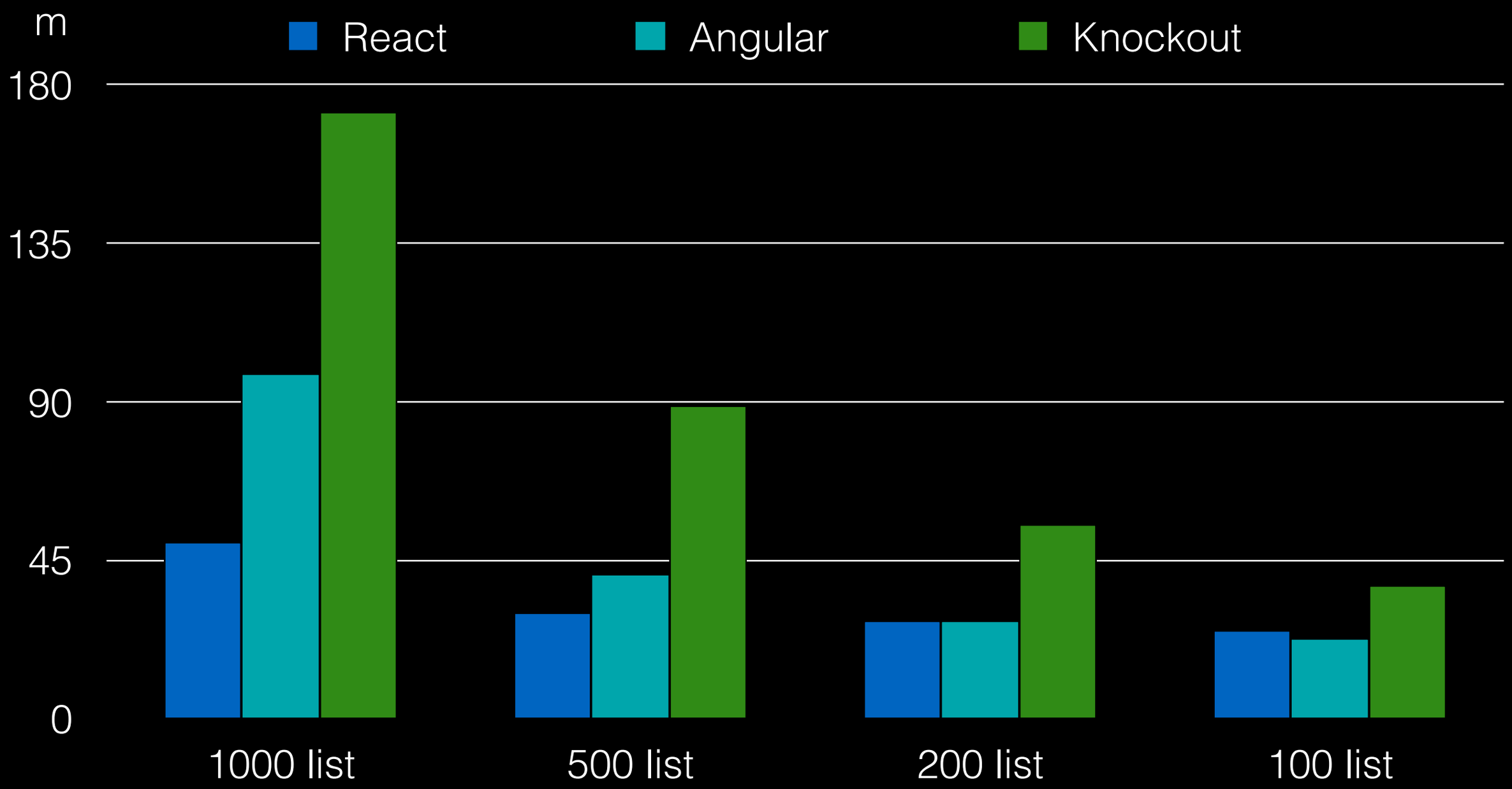
取巧的对比算法



规范的数据流动



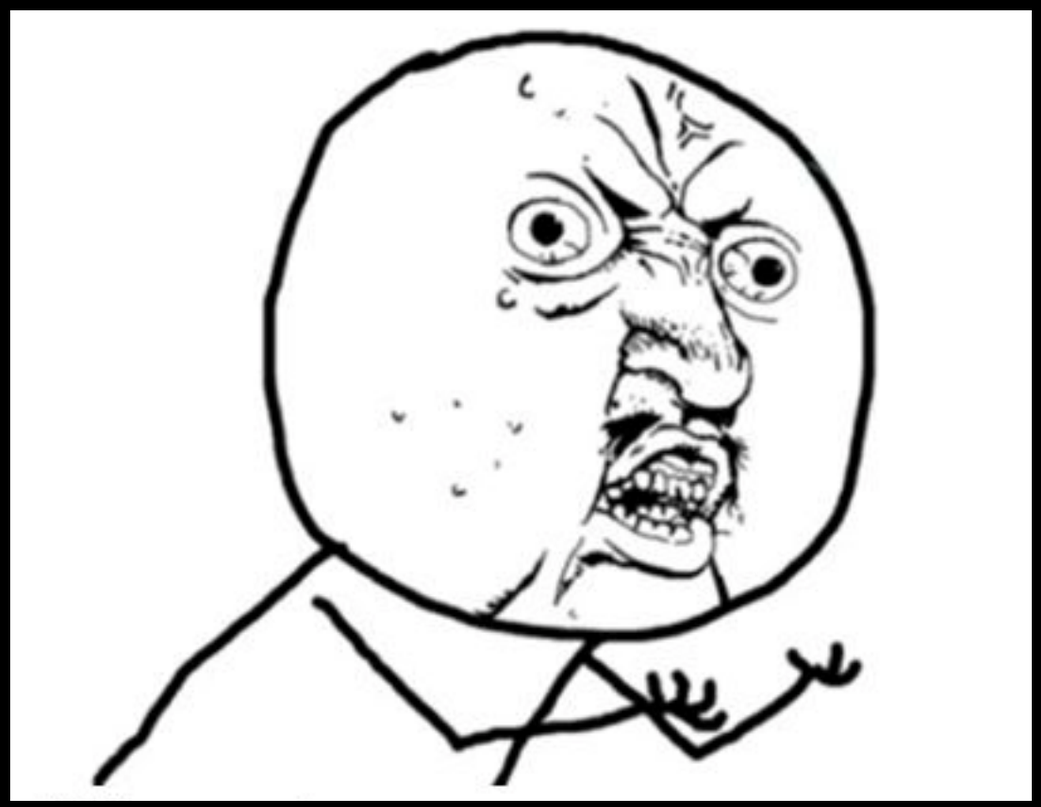
Benchmark


m
React
Angular
Knockout
180
135
90
45
0
1000 list
500 list
200 list
100 list

Vue.js

接口简单 & 易学易用

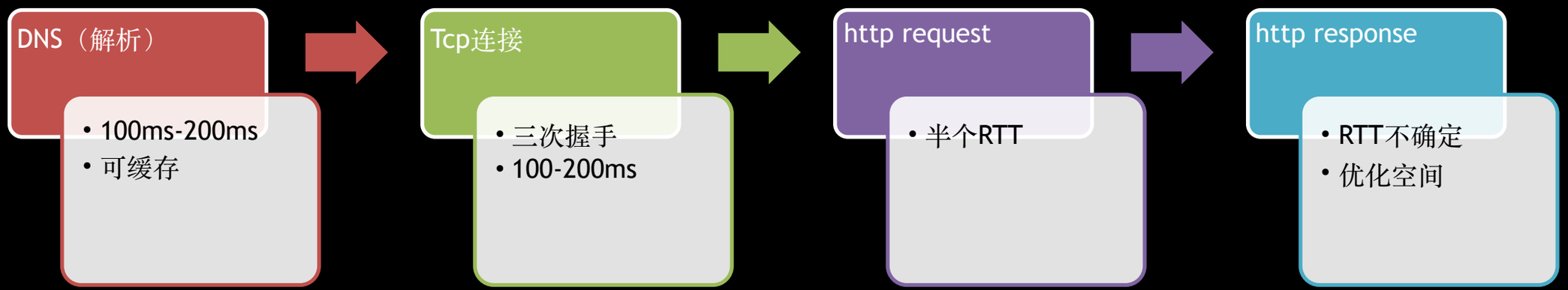
and then?



抛弃IE8！！！

海量业务场景下的挑战

访问一个页面的网络过程



?

多普勒测速

t1 – t2 = DNS

t2 – t3 = TCP/IP

T3 = RTT

多普勒测速方案：

- http://a-doppler.facebook.com/test_pixel?HTTP1.0&t=1&size=0k

  t1= DNS + TCP/IP + RTT

- http://a-doppler.facebook.com/test_pixel?HTTP1.1&t=2&size=0k

  t2= TCP/IP + RTT

- http://a-doppler.facebook.com/test_pixel?HTTP1.1&t=3&size=0k

  t3= RTT

- http://a-doppler.facebook.com/test_pixel?HTTP1.1&t=4&size=10k

  10k/(t4- t3) ≈ bandwidth

我们的方案

t2 = RTT

t1 – t3 = DNS

t1 – RTT – DNS = TCP/IP

我们自己的方案：

- http://1.url.cn?t=1&size=0k

t1= DNS + TCP/IP + RTT

- http://1.url.cn?t=2&size=0k

t2= RTT

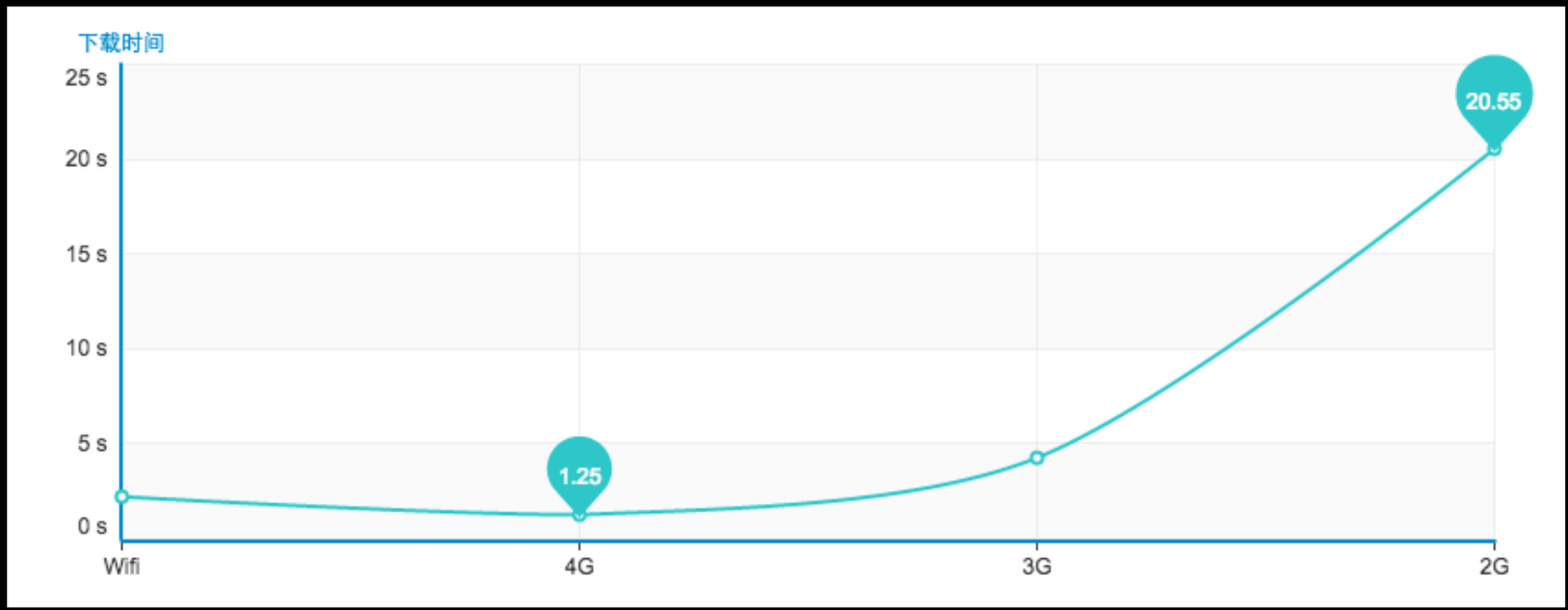
- <link rel=“dns-prefetch” href=“//2.url.cn” />

dns 被浏览器缓存

- http://2.url.cn?t=3&size=0k

t3 = tcp/ip + RTT

多普勒测速



市面上的MV*库要不太大，要不不支持IE8.....

我们的MVVM库——Q.js

- 足够小 - gzip 6k

```
Q.js 54.79 kB → 17.34 kB → 6.24 kB (gzip)
```

- 兼容性 - IE6+ (with jQuery and es5-shim) & All Mobile Browser (with Zepto)

仅仅使用MV*还不够，

还需要组件化

通常的组件化代码

```
<link href="./bootstrap.min.css" rel="stylesheet">
<script src="./jquery.js"></script>
<script src="./bootstrap.min.js"></script>
```

```
<div class="modal fade" id="my-dialog">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
        </span></button>
        <h4 class="modal-title">我是Dialog</h4>
      </div>
      <div class="modal-body">
        <p>hello world</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">确定</button>
        <button type="button" class="btn btn-primary">取消</button>
      </div>
    </div>
  </div>
</div>
```

```
$('#my-dialog').dialog(myOpts);
```

如果使用组件能跟使用原生DOM元素一样，该多好？

```
<dialog>
    <title>我是Dialog</title>
    <article>hello world</article>
</dialog>
```

Yes, Web Component!

- 全平台？全场景？(IE8？安卓2.3？Web侧？PC客户端？手机客户端？)
- 重用性与易扩展性如何解决?
- 如何降低调试成本?
- 迁移成本

基于动态编译和MVVM的类Web Component组件化方案——Ques

· 线下编译解决兼容性：IE8(甚至IE6)兼容

```
<!DOCTYPE html>
<html>
<head>
<title>Ques</title>
<style>
/* just a reset */
body {
    margin: 0;
}
.q-mark {
    display: none;
}
</style>
</head>
<body>
<enterlive></enterlive>
<qtree id="anchor_list">
    <title>我关注的主播</title>
</qtree>
<qtree id="room_list">
    <title>我收藏的房间</title>
</qtree>
<script src="./pages/client/main"></script>
</body>
</html>
```

开发阶段

```
<!DOCTYPE html><html><head><title>Ques</title><style>body{margin:0}.q-mark{display:none}</style><script>
    var _T=[+new Date];</script><link rel="stylesheet" href="css/client.a2a6e.css"></head><body><div
class="enterlive__container component-1" role="button" q-on="click: enterLive">&#x8FDB;&#x5165;&#x76F4;&
#x64AD;&#x5927;&#x5385;</div><div class="qtree__container component-2" id="anchor_list"><div class="
qtree__title" q-on="click: toggle"><i class="qtree__triangle"></i> &#x6211;&#x5173;&#x6CE8;&#x7684;&
#x4E3B;&#x64AD; <span class="qtree__num" q-text="num"></span></div><ul class="qtree__list qtree__list-
hide" q-height="list | calHeight" q-class="qtree__list-hide: hide"><li class="ui-card" q-repeat="list" q-
on="dblclick: enterRoom(this)"><img q-class="ui-card__avatar-offline: offline" src="" q-attr="src: avatar
" class="ui-card__avatar"><div class="ui-card__info"><div class="ui-card__name"><span name="" q-text="
name"></span></div><div class="ui-card__intro"><span intro="" q-text="offline | showIntro"></span></div>
</div><div class="extend" q-on="click: del(this)"><i></i>&#x4ECE;&#x5217;&#x8868;&#x5220;&#x9664;&#x6B64;
&#x76F4;&#x64AD;&#x95F4;</div></li></ul></div><div class="qtree__container component-2" id="room_list"><
div class="qtree__title" q-on="click: toggle"><i class="qtree__triangle"></i> &#x6211;&#x6536;&#x85CF;&
#x7684;&#x623F;&#x95F4; <span class="qtree__num" q-text="num"></span></div><ul class="qtree__list
qtree__list-hide" q-height="list | calHeight" q-class="qtree__list-hide: hide"><li class="ui-card" q-
repeat="list" q-on="dblclick: enterRoom(this)"><img q-class="ui-card__avatar-offline: offline" src="" q-
attr="src: avatar" class="ui-card__avatar"><div class="ui-card__info"><div class="ui-card__name"><span
name="" q-text="name"></span></div><div class="ui-card__intro"><span intro="" q-text="offline | showIntro
"></span></div></div><div class="extend" q-on="click: del(this)"><i></i>&#x4ECE;&#x5217;&#x8868;&#x5220;&
#x9664;&#x6B64;&#x76F4;&#x64AD;&#x95F4;</div></li></ul></div><script>_T.push(+new Date);</script><script
src="http://7.url.cn/edu/jslib/requirejs/2.1.6/require.min.js"></script><script>require.config({paths:{
    jquery:"http://pub.idqqimg.com/guagua/qiqiclient/js/lib/jquery-1.11.0.min",main:"./js/client.1fea2"},
    shim:{}}),require(["jquery","main"],function(){});</script><script>_T.push(+new Date);</script></body
></html>
```

上线阶段

· 可组合：复杂UI可拆分成多个简单UI

```
<div>
    <clkchange q-ref="a"></clkchange>
    <ttext q-ref="b"></ttext>
</div>
```

```
/**
 * ## Clickchange Component
 * Click a button and go to change a text node
 *
 * @component clickchange
 */
module.exports = {
    data: {},
    // when vm init complied bind the data
    compiled: function () {
        var a = this.$.a,
            b = this.$.b;
        a.$on('change', function (value) {
            b.$set('text', value);
        });
    }
};
```

· 可继承：相似组件可通过基类扩展

```
<dialog extend>
    <header>
        <h2>欢迎使用Ques</h2>
    </header>
    <article>
        <p>请输入要设置的值</p>
        <ui-input value="" q-model="curVal" q-on="keyup: submit | key enter" q-focus="focus"></ui-input>
    </article>
</dialog>
```

新组件继承自dialog，将继承dialog的数据结构，方法，事件等

· 可重用：可在不同场景使用

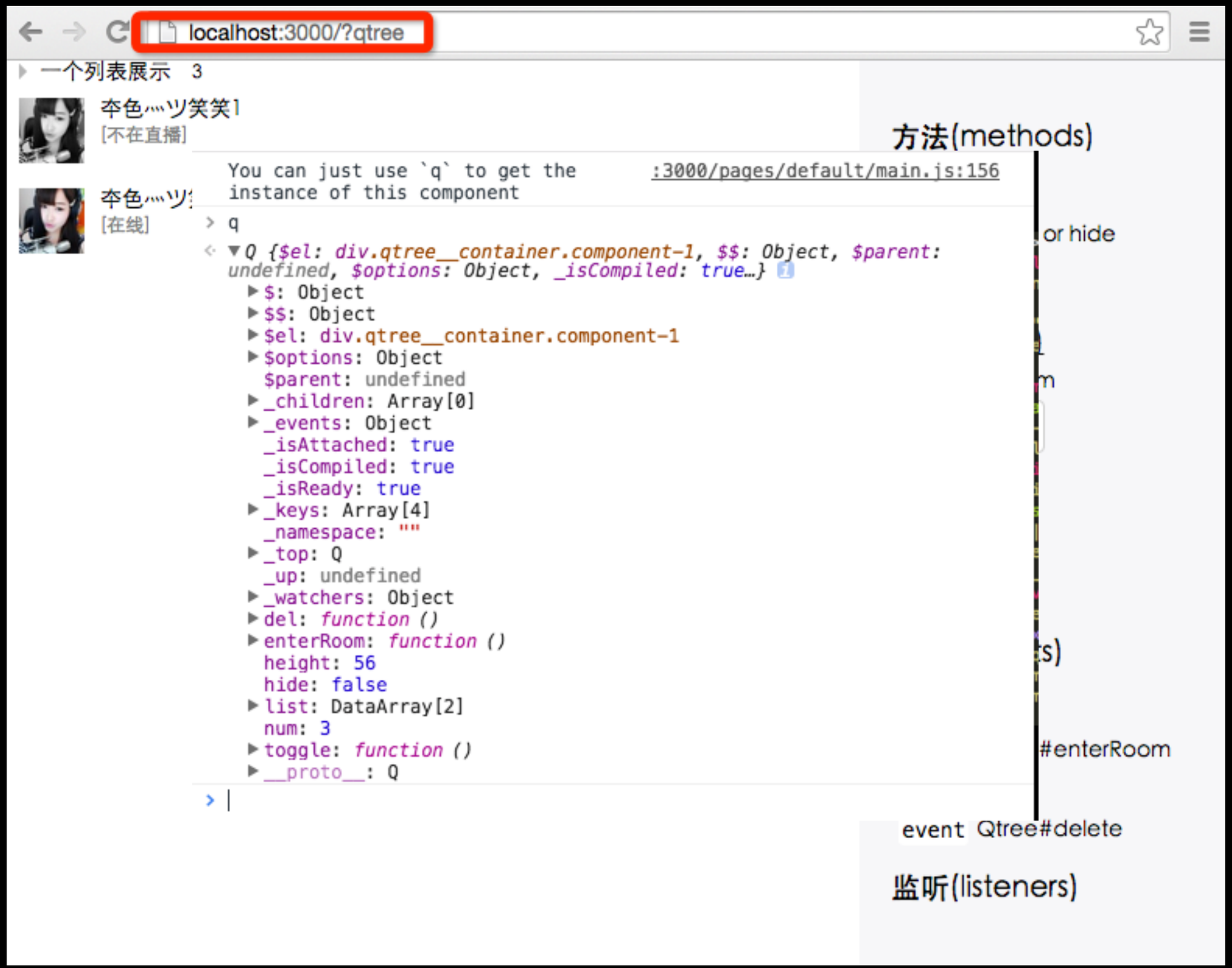
```
<third-code>
    &lt;diy-preload&gt;&lt;/diy-preload&gt;
</third-code>

<p>在页面目录如对应的db配置文件加上preload属性：/src/pages/diy/db.diy.js</p>
<third-code>
    var DB = require('db');

    DB.extend({
        ke: DB.httpMethod({
            url: 'http://ke.qq.com/cgi-bin/index_json',
            type: 'JSONP',
            preload: true
        })
    })

    module.exports = DB;
</third-code>
```

· 单例调试 + 自动文档化


localhost:3000/?qtree
一个列表展示 3
李色灬ツ笑笑1
[不在直播]
李色灬ツ
[在线]
You can just use `q` to get the instance of this component
:3000/pages/default/main.js:156
q
Q {$el: div.qtree__container.component-1, $$: Object, $parent: undefined, $options: Object, _isCompiled: true…}
$: Object
$$: Object
$el: div.qtree__container.component-1
$options: Object
$parent: undefined
_children: Array[0]
_events: Object
_isAttached: true
_isCompiled: true
_isReady: true
_keys: Array[4]
_namespace: ""
_top: Q
_up: undefined
_watchers: Object
del: function ()
enterRoom: function ()
height: 56
hide: false
list: DataArray[2]
num: 3
toggle: function ()
__proto__: Q
方法(methods)
or hide
s)
#enterRoom
event Qtree#delete
监听(listeners)

· 第三方组件 = less组件开发成本 + less迁移成本

```
<third-code>
    &lt;third-button&gt;&lt;/third-button&gt;
</third-code>
```

高亮代码组件，通过第三方库highlight.js来实现非常快

```
1  <pre class="$__pre">
2      <code>
3          <content></content>
4      </code>
5  </pre>
```

```
1  var _ = require('Q')._;
2
3  m
4      bind: function () {
5          var el = this.el,
6              script = document.createElement('script');
7          script.onload = function () {
8              /* globals hljs: false */
9              hljs.highlightBlock(el);
10             _.addClass(el, 'show');
11         };
12         script.src = '//cdnjs.cloudflare.com/ajax/libs/highlight.js/8.6/highlight.min.js';
13         document.body.appendChild(script);
14     },
15     unbind: function () {}
16 };
```

只需封装一个接口，便可在任意地方使用，
唯一的弱点是内部不可嵌套

最终效果：结构有迹可循，代码可预测

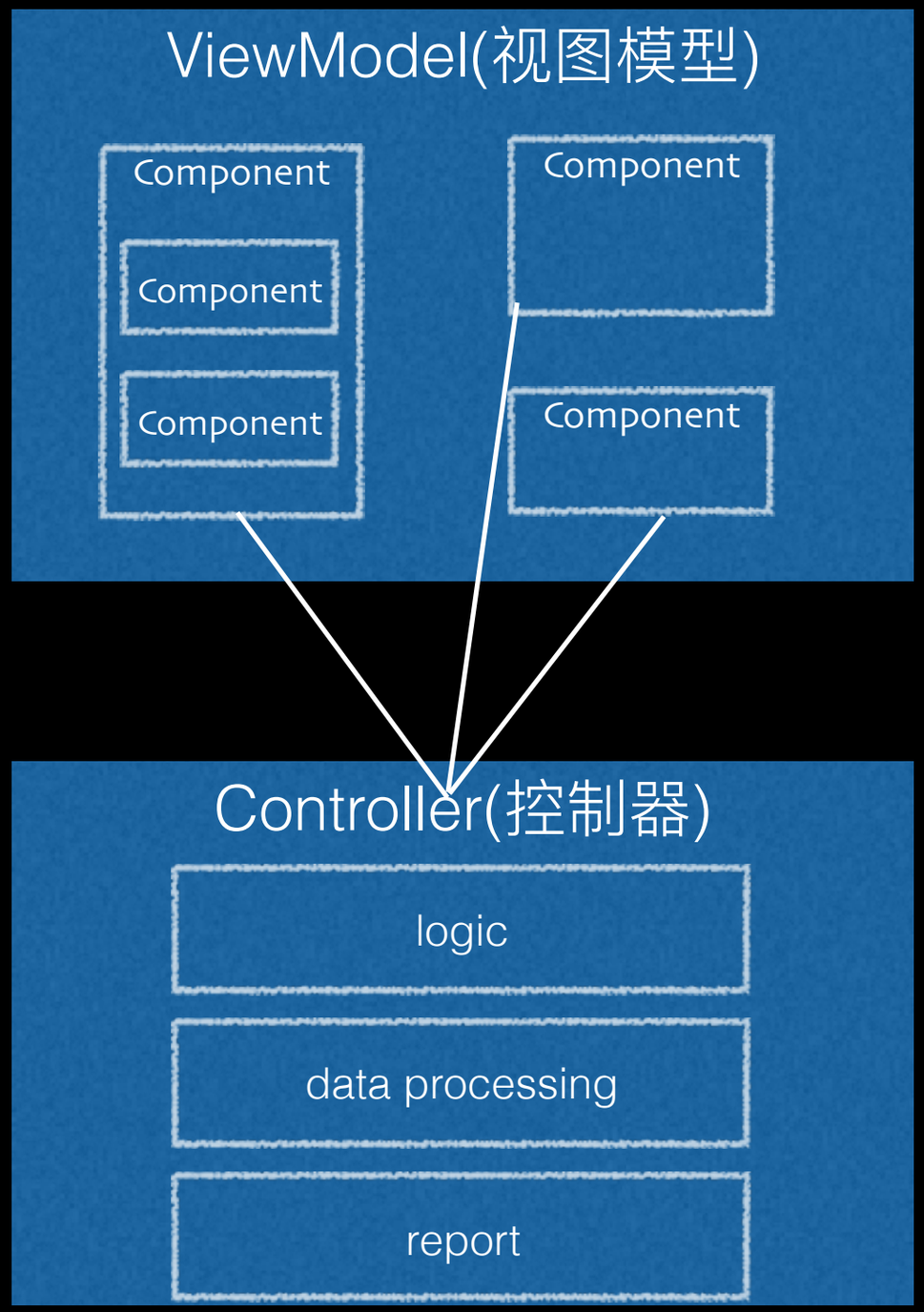
CSS(样式)

HTML(结构)

Javascript(逻辑)

以前我们的代码结构是这样的

符合人类通常思维模式：
将事物拆分成有机个体，
再各个击破



现在的代码结构是这样的