ExMobi
企业移动应用平台

跨平台移动应用开发架构实践

烽火星空产品架构师　黄楠

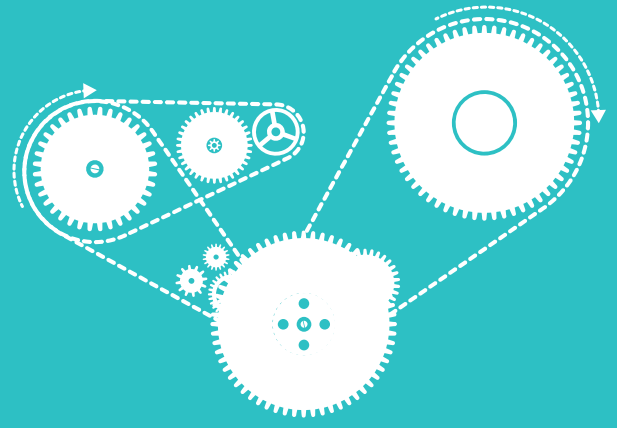
2016-04-17

跨平台移动开发技术大观

ExMobi跨平台架构实践

用互联网思维做产品

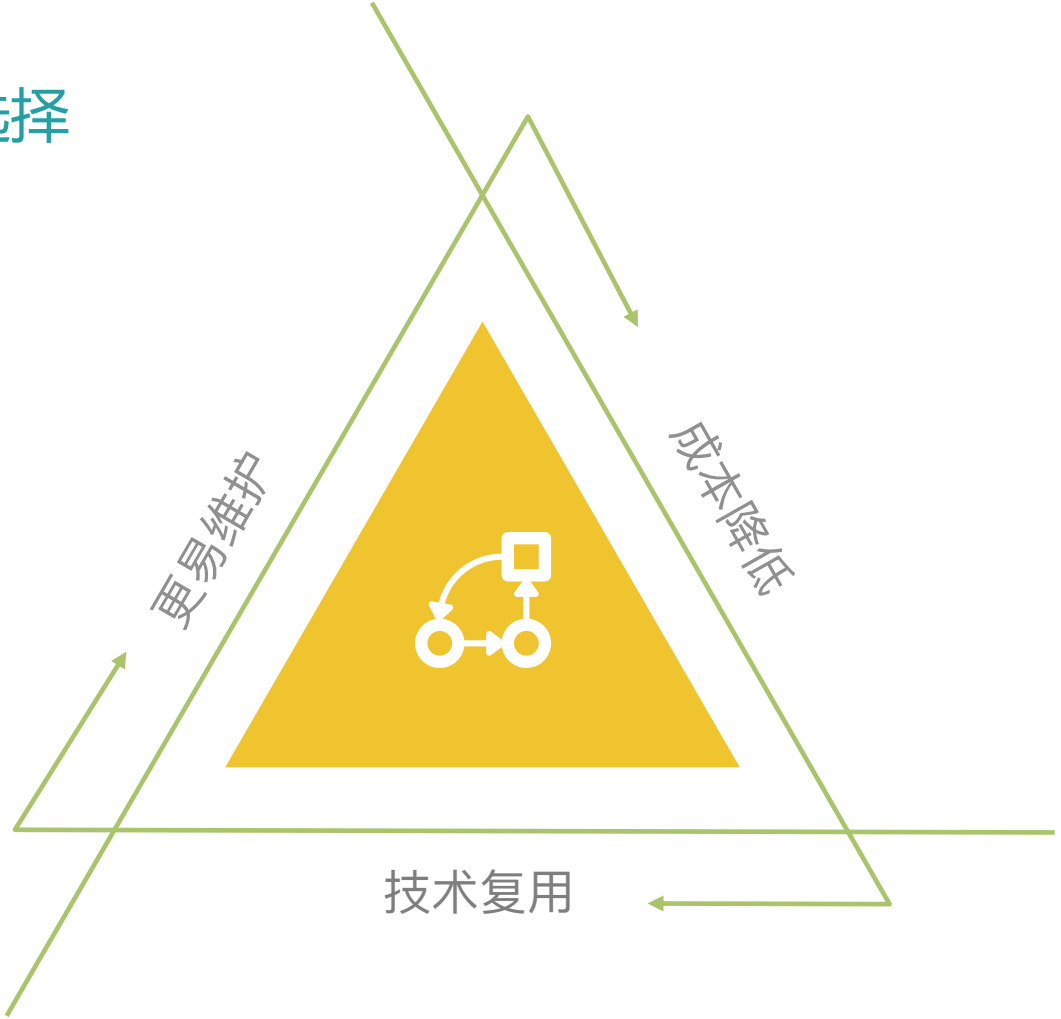
1 跨平台移动开发技术大观



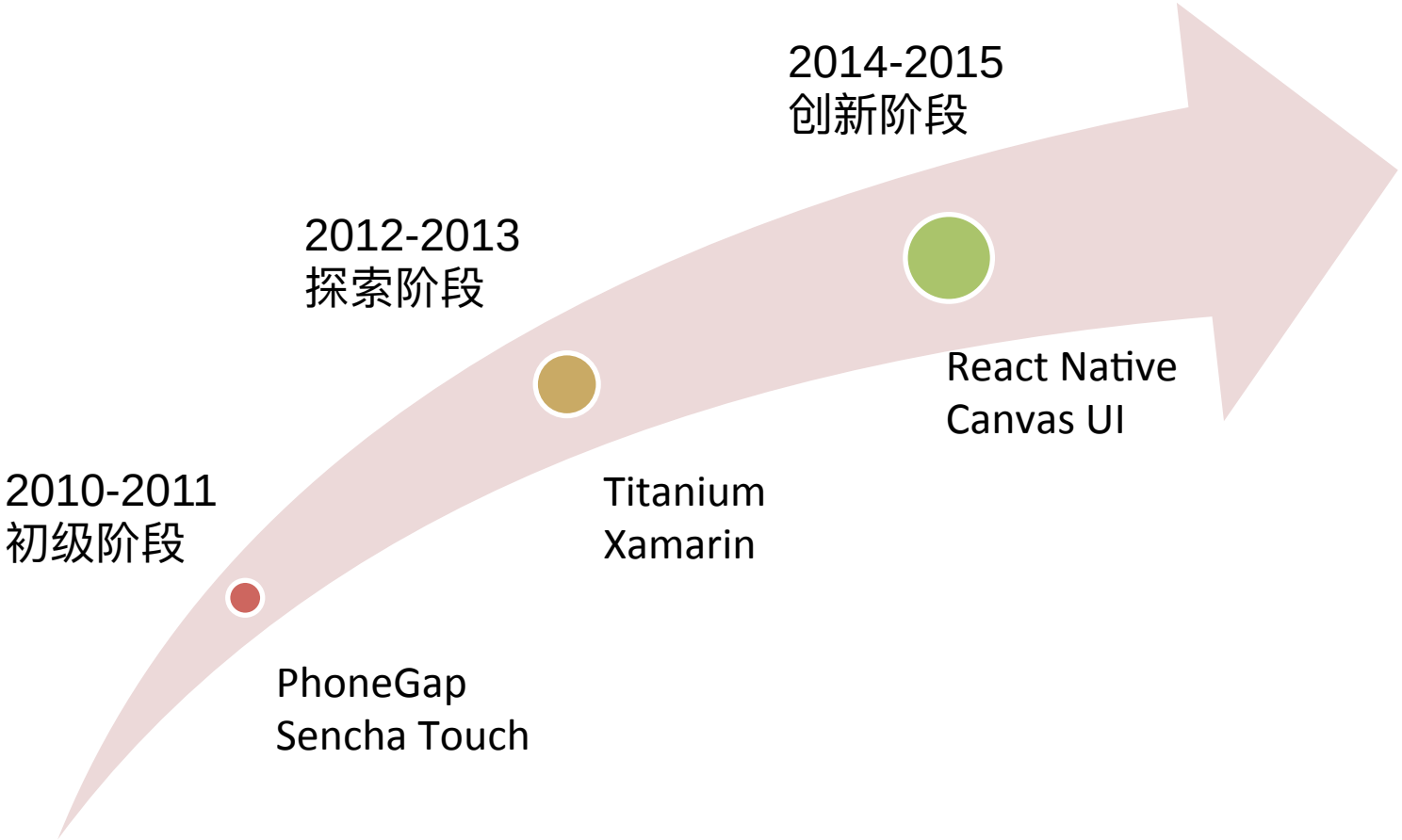
移动应用客户端形态的 必要性

独立入口　　增强体验　　平台运作

跨平台移动开发技术逐步 被选择



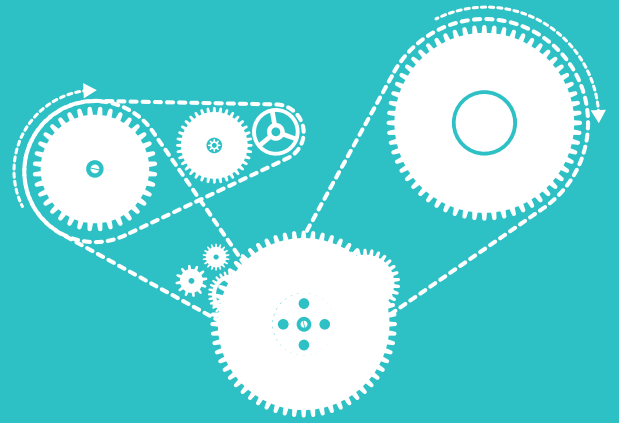
移动应用跨平台开发方案 历程



跨平台移动应用开发技术 原理比较

	Mobile Web	Hybrid	Interpreted App	Cross Compiling	Native App
代表性产品	jQuery Mobile Canvas UI	PhoneGap	Titanium React Native	Xamarin	JAVA Objective-C
跨平台能力	强	强	中	中	低
主要开发语言	HTML5	HTML5	中间语言	微软系语言	原生语言
本地能力	弱	中	高	高	高
用户体验	一般	一般	较好	较好	最好
学习门槛	低	低	低	中	高
升级维护	易	易	易	难	难

2 ExMobi跨平台架构实践



烽火星空
STARRYSKY

ExMobi
企业移动应用平台

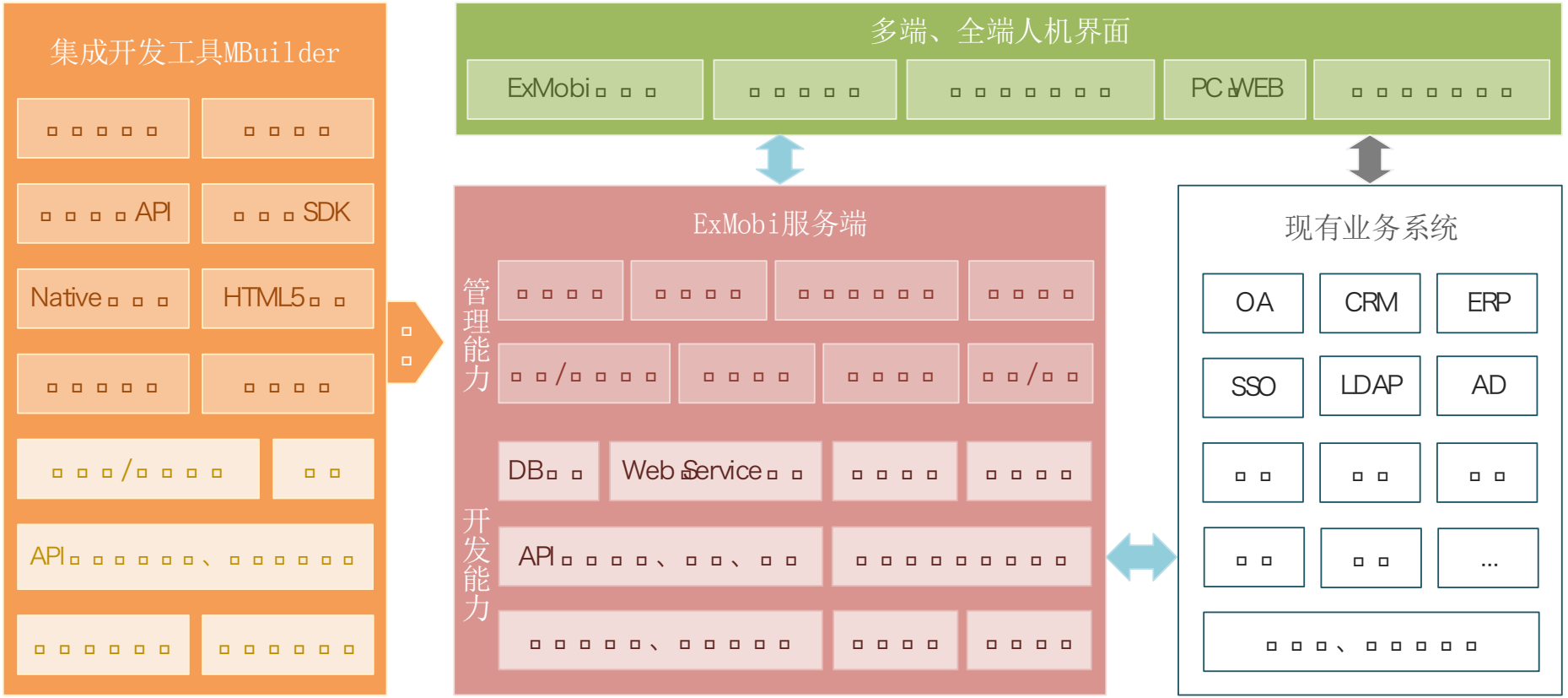
8
年历程

8

5

5
代产品

ExMobi移动应用平台 生态体系

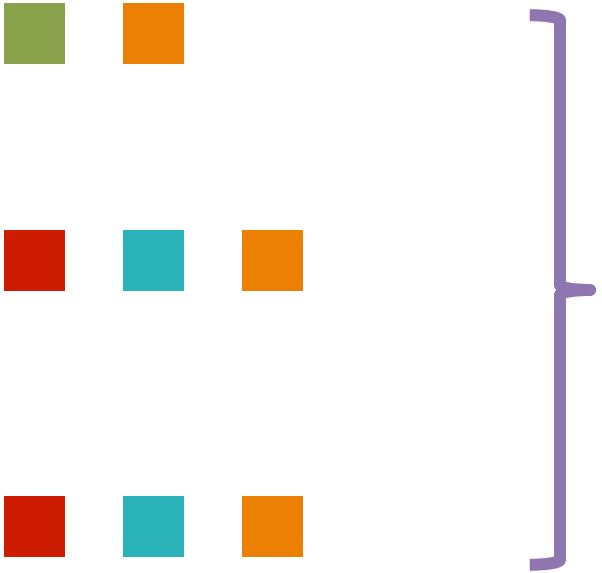


ExMobi的选择之 架构

保持原生性能和效率

使用主流开发技术

方便进行升级维护



解释型架构不仅可以保持原生的效果还能自由选择中间语言向主流靠拢，而且能够像html一样发布代码，避免应用市场审核耗时

Mobile Web　Hybrid　Cross Compiling　Interpreted App

ExMobi的选择之 中间语言

UI组件表示——自定义

降低学习门槛——标准化

技能复用——主流技术

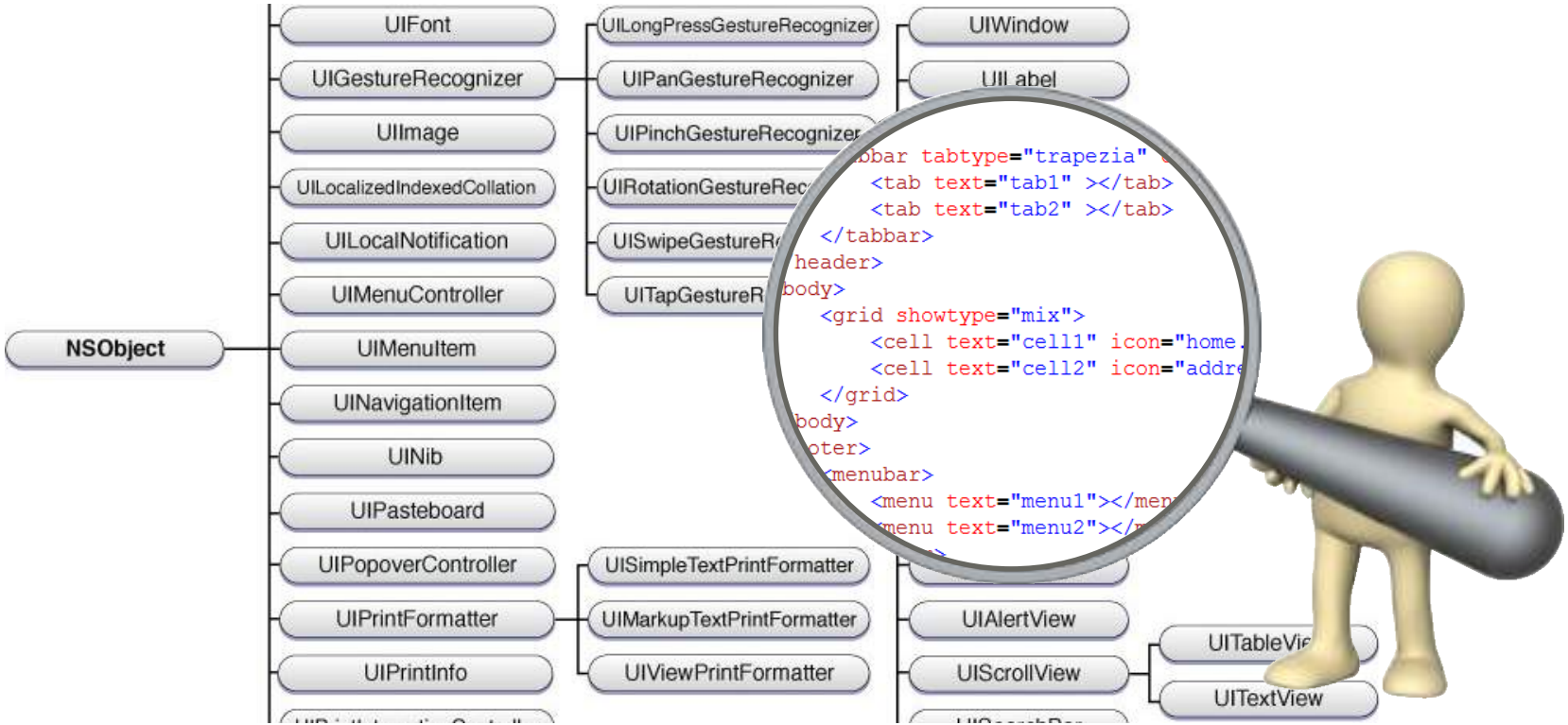
VS

UIXML自定义标记语言

CSS样式表UI组件渲染

JS原生脚本引擎扩展

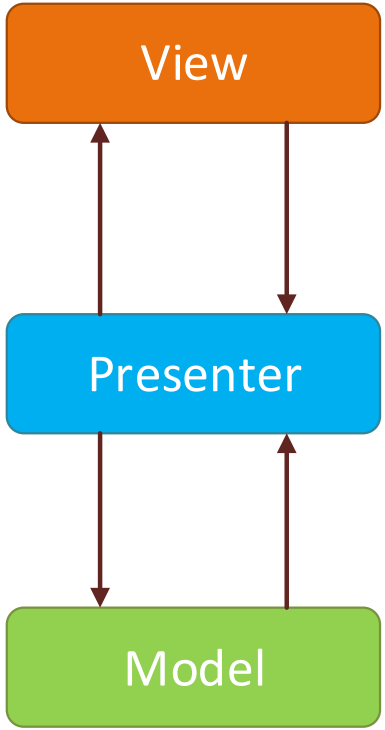
ExMobi的选择之 中间语言



ExMobi的选择之 中间语言

```
<!-- ExMobi UIXML(XHTML)文件 -->
<html>
    <head>
        <meta content="charset=utf-8"/>
        <title>Hello World</title>
        <script>
        <![CDATA[
            function doLoad(){
                document.getElementById('content').innerHTML = '欢迎参加南京开发者大会';
            }
        ]]>
        </script>
    </head>
    <body onload="doLoad()">
        <div id="content" style="text-align: center;">正在加载，请稍后</div>
    </body>
</html>
```

ExMobi的选择之 开发模式



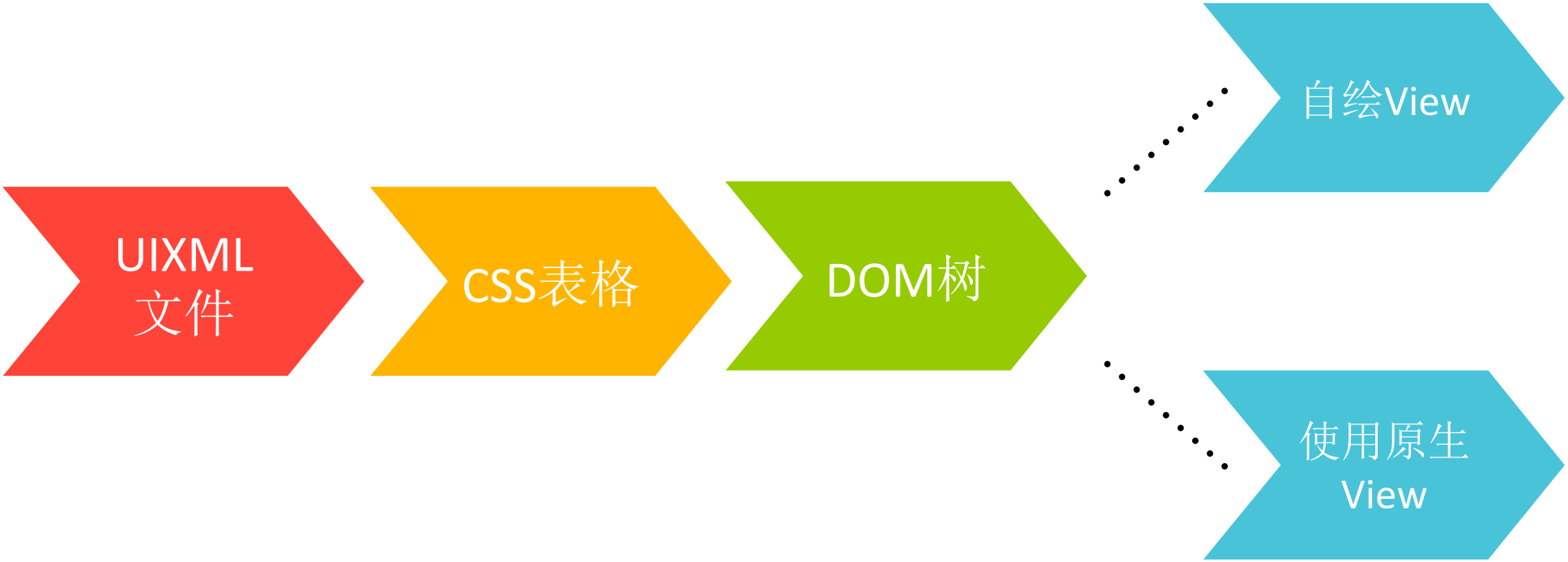
MVP模式特点：

View和Model不直接交互，而是通过与Presenter的双向通信交互（数据注入）

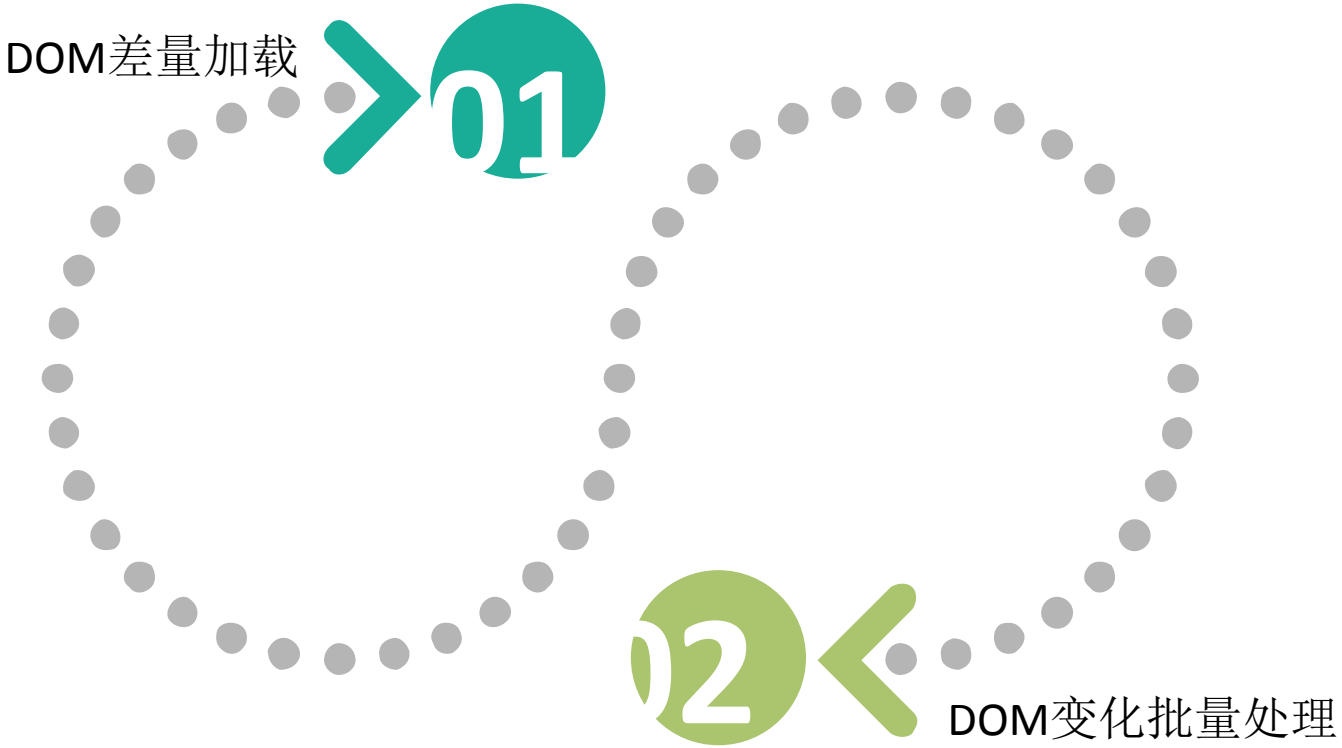
View为被动视图，所有逻辑操作均在Presenter中完成（事件驱动）

```
$('#content').renderAfter('res:page/template/
my.template', 'http://${domain}/server.jsp',
function(h, t, o){
    //alert($.JSON.stringify(o));
});
```

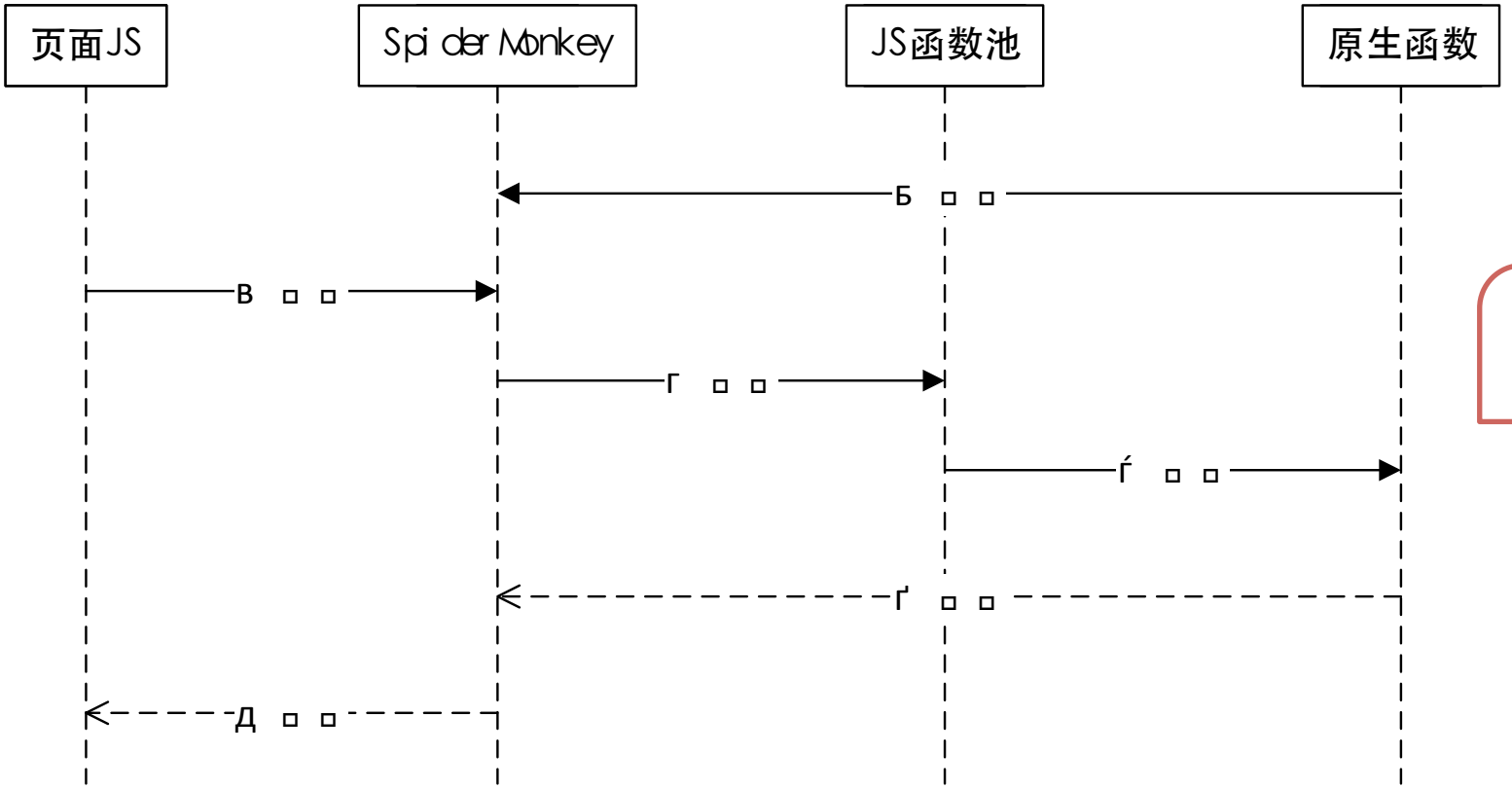
关键性问题的解决之 自定义组件和CSS渲染



关键性问题的解决之 解析效率



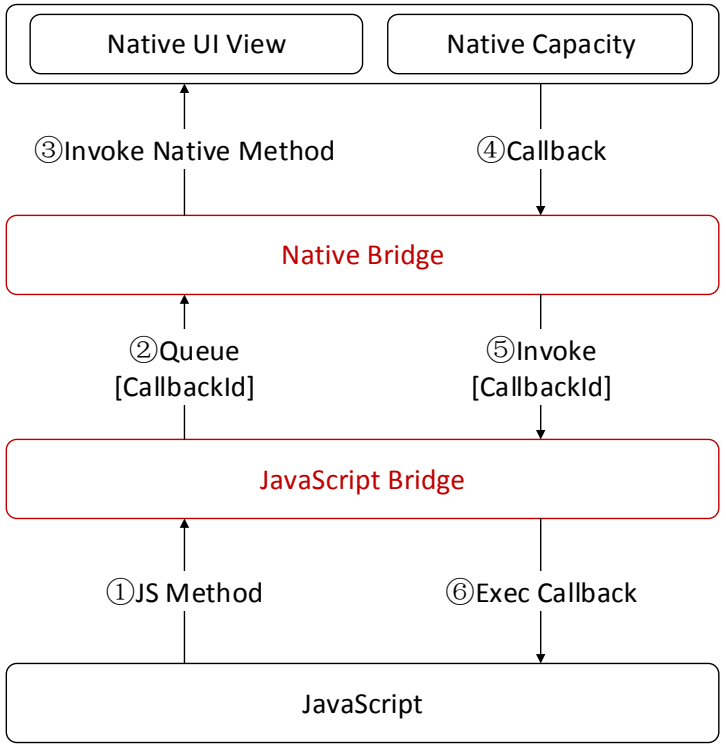
关键性问题的解决之 JS绑定



跨平台JS引擎
JS函数分发

关键性问题的解决之 开发调试

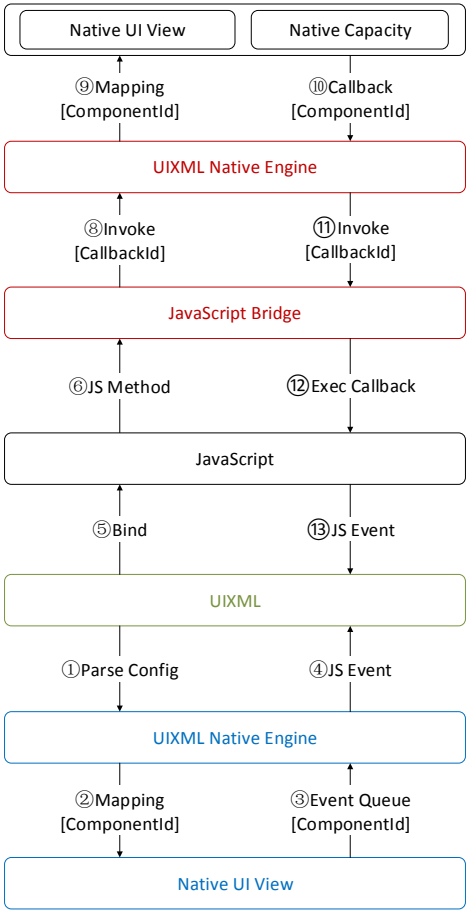
ExMobi与React Native的 组件通信原理比较




```
class HelloWorld extends React.Component {
  render() {
            return  React.createElement(React.Text,  {style:
styles.text}, "Hello World!");
  }
}

class Main extends React.Component {
  render() {
    return (<React.NavigatorIOS
      style={styles.container}
            initialRoute={{
       title: 'Property Finder',
       component: HelloWorld,
      }}/>);
  }
}
```

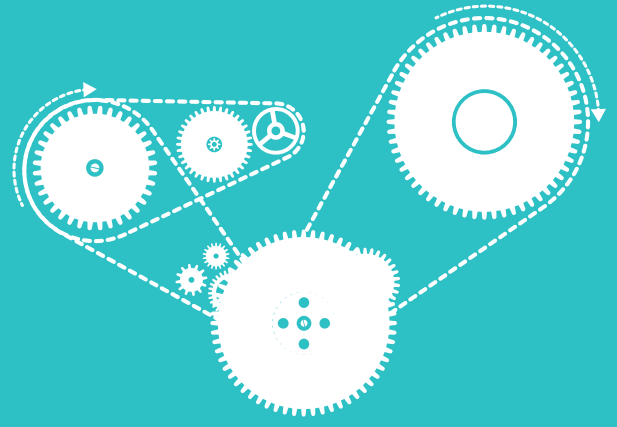
ExMobi与React Native的 组件通信原理比较



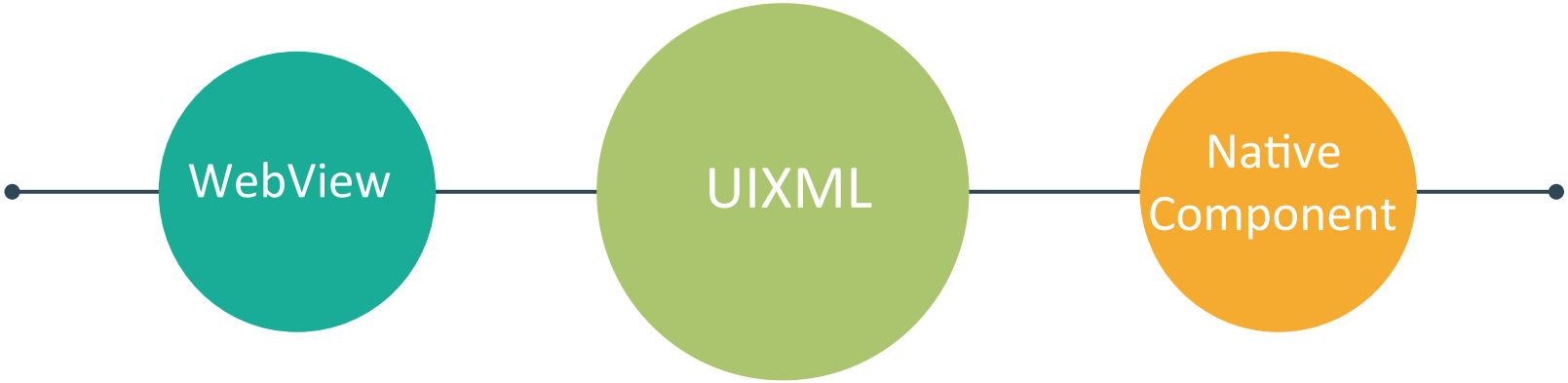
采用解释型架构，使用JS、标签和CSS等主流中间语言进行开发，UI大部分是自绘，灵活有余动画不足

React Native	ExMobi
JS为入口	标签为入口
JS环境为全局	JS环境为窗口级
数据共享简单直接	依赖原生生命周期
UI具有系统特性	UI使用统一特性

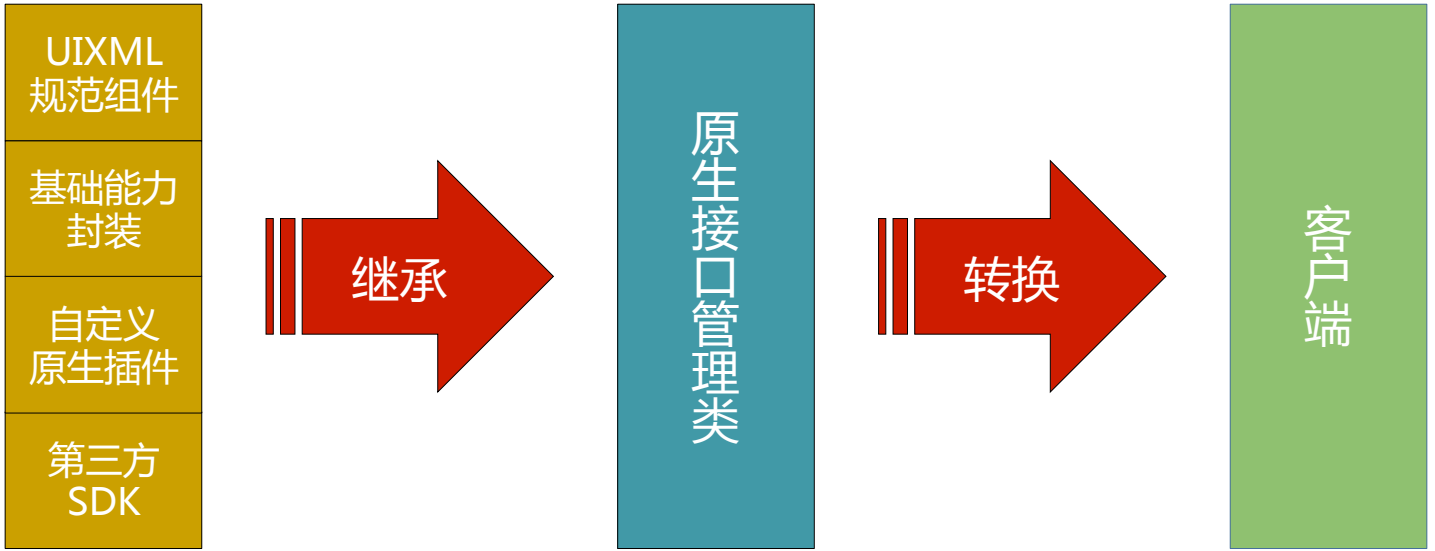
3 用互联网思维做产品



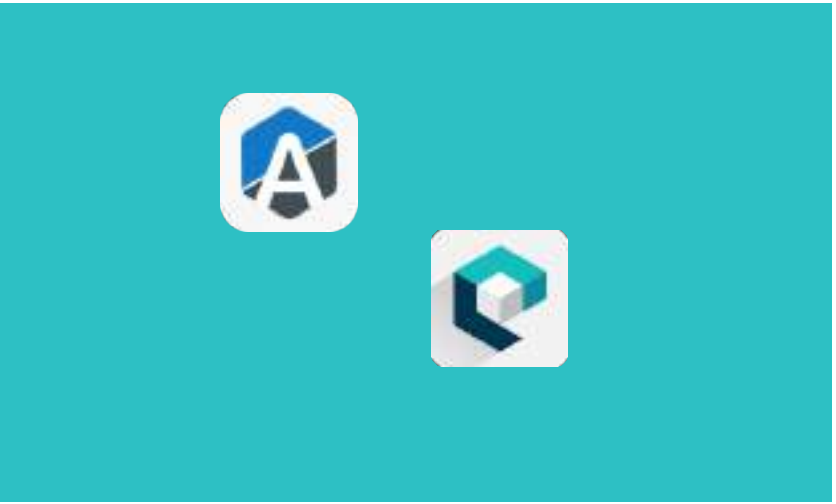
ExMobi架构上的 改变



ExMobi架构上的 改变



以开放之本拥抱 开源世界



跟我们一起探讨 跨平台移动应用开发

是否需要设计工具

自绘组件带来的动画问题

面向的开发者技能要求

是否可以与React Native结合

真机JS远程调试

代码的安全性问题

NJSD 2016

感谢聆听

ExMobi开发者交流群二群

扫一扫二维码，加入该群。

扫一扫上面的二维码图案，加我微信

ExMobi示例应用下载

扫二维码下载示例应用体验