**微信小程序**

小程序是一种**不需要下载安装**即可使用的应用，它实现了应用「触手可及」的梦想，用户**扫一扫或搜一下**即可打开应用。也体现了「**用完即走**」的理念，用户不用关心是否安装太多应用的问题。应用将无处不在，**随时可用**，但又无需安装卸载。

---------------------张小龙

**理念说明例子（餐馆点菜）**

用户排队点餐只需要在餐馆扫一下它的二维码，然后就启动了这个餐馆的小程序，立即在小程序里排队或点餐就行了。无需为了可能一次性的使用行为，下载一个app，或者关注一个公众号，用完又取关。

**一. 简述**

**1.小程序的产生目的**

（1）从服务的查找到支付形成**闭环**；

(2)利用微信的流量入口以及巨大的**流量**，最终形成 wechatOS；



综于上述，微信的限制与倡导使得小程序更像一个依赖于微信的小工具。对于健身团产品的推广（特别是私教），小程序开发的服务到支付形成的闭环是优势所在，而入口是要考虑解决的一个劣势。比如私教在小程序上：由分享进入私教，提供私教模块小视频，私教分套教程说明，用户与教练的互动，每日的课程安排等到支付。微信的强社交，小程序的轻巧，服务支付的闭环等对于私教是比较有利的。

**2.主要理念**

1. 小程序的**目标是线下服务**和商家；

2. 小程序要挖掘和覆盖新的服务场景和提供新的服务能力；

3. **用完即走**（利用**工具**快速有效的完成一项服务）的理念；

4. 去中心化，主要启动来源于二维码；

5. 小程序**分享的是一个活的信息**。

1. **性能**

在1M大小的限制下，使用微信的接口与组件，与原生app的流畅性比较差别不大。小程序运行在微信平台，和其它微信小程序“共享运行内存”，单个小程序的性能极可能遇到瓶颈。

**4.主要功能接口**

上传下载， websocket ，视频音频， 数据缓存， 地图位置 ，设备功能信息（罗盘 电话 扫码）， 支付 ，消息 ，分享， 获取二维码；

**websocket**：实现社区功能，私教里教练与用户的互动；

**模板消息，客服信息**：实现服务支付的闭环；

**视频音频**：提供健身模块小视频吸引客户，扩展电视健身的范围；

**分享（支持分享当前页面信息到好友或群，可以不用打开小程序就可以看到分享页面内容）**： 构造精炼，鲜活的页面内容分享，吸引用户；

**二维码（通过后台接口可以获取小程序任意页面的二维码，扫描该二维码可以直接进入小程序对应的页面 ）**：利于健身团的推广，构建一些模块打通增强电视和手机的联系，构造增强非电视环境的使用场景。

**5.入口**

**搜索，扫二维码，分享，显示在聊天顶部，发送到桌面(Android) 。**



5.1搜索查找弱化，必须**精准匹配**，不支持模糊搜索，主推扫二维码：大众点评查找时只搜大众查找不到，至少要原来app名“大众点评”；

5.2 **小程序不能和公众号重名** ：小程序名："大众点评+"；

5.3**功能简化**，大小限制1MB；比如“携程酒店机票火车票”；

5.4**历史列表**：用户使用过的小程序会被放入列表，方便下次使用；

5.5. **分享小程序页**：支持分享当前页面信息到好友或群，可以不用打开小程序就可以看到分享页面内容。

**6.开发影响**

**对微信有完全严重的依赖**:

1. 封装了一些原生方法为JavaScript app，使得调用手机功能简单，并且提供了离线缓存功能；提供了一些与微信公众平台的相同的接口，可以使用微信一些比较重要功能；

(2)目前许多组件都有各式的bug，开发工具与真机的表现现象不一致，真机调试功能弱---在真机上只有日志框；

（3）小程序运行在微信平台，在发布后可能**和众多小程序“共享运行内存”**，单个小程序的性能极可能遇到瓶颈。当由于小程序的大小限制为1m，而且其不推荐功能复杂化，所以**对开发的影响也不会太大** 。

**7.对比**

**1.兼容性**

（1）Native App要解决繁复的问题；

（2） Hbuilder 页面为html5减轻页面兼容，硬件功能调用由Native.js （不是一个js库，运行环境集成在5+runtime） 原生对象转义，映射为JS对象操作减低难度；

（3）html5实现需要解决页面的兼容；且性能较差，比如视频的播放与移动端硬件的交互等；

（4）小程序使用其提供的组件编写界面，根据其提供接口调用移动端功能，**依赖性是三者中最重**，但实现**简单统一**，当然**目前其存在不少的问题**，但**对于我们公司的需求是足够的。**

**2.功能**

（1）Native和 Hbuilder都可以实现大而全的**app**功能；

（2）html5只能调用部分手机功能；

（3）小程序目前只能做功能较少的**小应用**，但**对于健身团的需求理念也是足够的。**

**3.平台基础**

（1）Hbuilder依托的Dcloud提供了可以**开发一定能力的app**；

（2）小程序依托的微信**可靠性更强**，且具有**巨大的流量**，且有可能借助利用微信提供的**更有效的用户信息**。

**4.开发**

（1）Hbuilder的页面使用的mui组件是开源的，开发容易发现问题可以对其修改，移动端功能可以使用封装好的html5+，或者Native.js；

(2)小程序组件与接口都由微信提供

**总结**

1.**理念不同**：Native和 Hbuilder是开发功能教大的app，小程序是开发小应用；小程序更适合健身团的需求；

2.**小程序更有优势**：微信的社交功能，有较好的体验，对健身团，特别是私教 有更好的闭环。

**二 . 详述**

**1.性能**

通过**Progressive Web Apps** 理念（下面有大概的说明）来整合。

按照Progressive Web Apps 理念，使用了类似react的虚拟dom来渲染页面结构，加上框架提供的接口调用移动端的一些功能（调用相册或相机等）和获取移动端信息；微信的功能（支付等），微信用户信息。

使用了虚拟dom，每次操作改变页面相关数据，框架就将该部分与之前进行比较，不同再进行局部渲染；部分组件Native实现（video，map，canvas）；WebView的预先加载等使得性能得到一定的提升 。

**2.理念**

**小程序的多处思想都与PWA一致**

**PWA（Progressive Web Apps） 渐进式应用**

2015年，Google 提出，用以提升Web用户体检，把开发者从App Store的打包、部署等流程中解放出来，目标是为Web App提供个性化能力、添加到主屏幕、推送通知、离线工作、触摸体验等。

主要特性（**类APP交互，消息推送，离线缓存**）：

1.渐进增强： 应用运行在尽可能多的环境中。它会使用任何可用的服务，如果没有可用的服务则会优雅降级。

2.响应用户界面： 应用适应多种输入方式（触摸、 语言输入等）和多种输出方式（不同的屏幕尺寸、震动、音频、盲文显示等）。

3.连接性、独立性：应用可在断网以及间歇性断网或低带宽环境下很好地工作。

4.类应用 UI：应用采用原生平台的 UI 元素，包括快速加载用户界面（可以通过 Service Workers 获取重要的缓存资源）。

5.持续更新：Service Workers API 定义一个进程用来将应用自动更新至新版。

6.安全通信：应用使用 HTTPS 通信来提供服务，阻止网络劫持和攻击。

7.应用发现：像 W3C 应用 Manifest 元数据一样，有利于搜索引擎查找 web 应用。

8.推送与互动：推送通知的特性，主动让用户了解最新动态。

9.可本地安装：在一些平台上，你可以安装 web 应用使得它看起来像一个本地应用（将 icon 放在主屏，在应用程序切换器单独列出，chrome 浏览器可选）。所有这些应用不用经过本地应用商店。

10.可连接性：通过 URL 可以轻松分享应用，不用安装即可运行，依赖浏览器，不依赖任何其他平台。

PWA 具体实现是通过三方面来加强用户体验：Web App Manifest，Service Workers，Application Shell

**1）Web App Manifest使Web更像Native**  
 Web App Manifest以JSON的格式定义Web应用的相关配置（应用名称、图标或图像连接、启动URL、自定义特性、启动默认配置、全屏设置等）。**2）Service Workers增强web 能力**

通过在浏览器中增加可编写脚本的网络代理层，监听浏览器与服务器的交互。通过Service Works实现资源离线缓存和更新

**3）App Shell 提升显示效率**

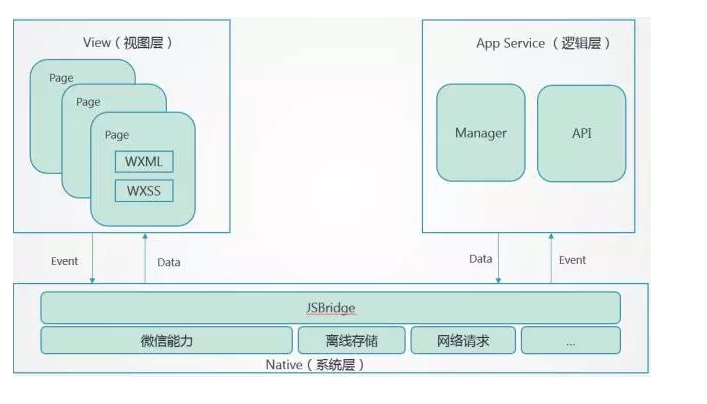
App Shell是应用的用户界面所需的最基本的 HTML、CSS 和 JavaScript，也是一个用来确保应用性能的组件。简单来说就是应用的外壳，它的首次加载将会稍微慢点，加载后立刻被缓存下来。这意味着应用的外壳不需要每次使用时都被下载，而是只异步加载需要的数据，以达到UI保持本地化。通过动态API，也可以达到实时加载和定期更新内容，所以使web能拥有如下特性：负载快、缓存、动态显示内容。

**2.框架**

微信小程序的框架包含两部分：**View**视图层、**App Service**逻辑层。

（1） 它们在两个线程里**独立**运行。

（2） 通过**JSBridage**进行通信，逻辑层把数据变化通知到视图层，触发视图层页面更新；视图层把触发的事件通知到逻辑层进行业务处理。



2.1.View视图层

View层由 WXML 与 WXSS 编写，用来渲染页面结构。

1. 一个 view 模块对应一个 webview 组件（基于WebView渲染，并**非原生渲染**）,
2. 小程序支持同时**多个 view** 存在。
3. WXSS中无法使用本地资源（所以在wxss中使用background-image属性开发工具可以显示，但是真机无法显示，解决：1.使用base64转码 2.使用image标签代替背景图片）。
4. 返回显示历史View，退出小程序，View状态不会进行销毁

2.2.App Service逻辑层

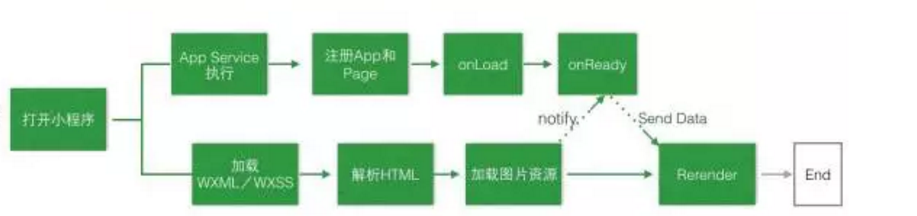
AppService层由JavaScript编写,用来逻辑处理、数据请求、接口调用。

1. App Service运行环境：IOS----------JSCore ; Android------- X5 JS解析器 ; DevTool ------ nwjs Chrome 内核
2. 页面的脚本逻辑是在JsCore中运行，JsCore是一个没有窗口对象的环境，所以不能在脚本中使用window，也无法在脚本中操作组件。
3. 开发者写的所有代码最终将会打包成一份 JavaScript，并在小程序启动的时候运行，直到小程序销毁。
4. 编写理念由操作dom转为操作数据。
5. 每个页面有独立的作用域，并提供模块化能力。

2.3.线程运行

Appservice Thread和View Thread会同时进行、并行加载，甚至Appservice Thread会更早执行，当View Thread加载完，通知Appservice Thread，Appservice Thread 会把准备好的数据用setData的方法返回给View Thread。

小程序的这种优化策略，可以减少用户的等待时间、加快小程序的响应速度。



1） View thread

初始化；向Appservice Thread 发送初始化完毕信号，等待传回初始化数据状态；接受到Appservice Thread 的数据后首次渲染；发送首次渲染完毕信号到Appservice Thread；进入持续渲染状态，一直等待Appservice Thread 通过this.setdata（）函数发送来的数据，只要收到就重新局部渲染。

2）Appservice Thread

初始化（onload只会首次渲染的时候执行一次，onshow是每次界面切换都会执行），接收到 View Thread 发送的初始化完毕信号后，将初始化数据发送给View thread，收到View thread发送来的首次渲染完成信号后进入激活状态（onReady()），在此状态下可以通过 this.setData 函数发送数据给View Thread ， 如果界面进入后台，服务线程就进入后台运行状态。

**3.文件结构**

小程序包含一个描述整体程序的 app 和多个描述各自页面的 page。

**(1)App()**

用来注册一个小程序。在小程序启动的时候调用，并创建小程序，直到销毁。在整个小程序的生命周期过程中，它都是存在的，单例，全局。

1)只能在app.js中注册一次。

2)在代码的任何地方都可以通过 getApp() 获取这个唯一的小程序单例

**(2) Page()**

通过App()注册完成小程序之后，框架就开始注册页面。所以不要在App()的 onLaunch 中调用 getCurrentPage() 方法，因为此时页面还没有注册完成。当页面注册完成之后，可以在 page.js 文件中调用 getCurrentPage() 方法，获取当前页面对象。

**(3)app和page生命周期**

**1. 应用生命周期**

1)用户首次打开小程序，触发 onLaunch（全局只触发一次）。

2)小程序初始化完成后，触发onShow方法，监听小程序显示。

3)小程序从前台进入后台，触发 onHide方法。

4)小程序从后台进入前台显示，触发 onShow方法。

5)小程序后台运行一定时间，或系统资源占用过高，会被销毁。

**2. Page生命周期**

1)小程序注册完成后，加载页面，触发onLoad方法。

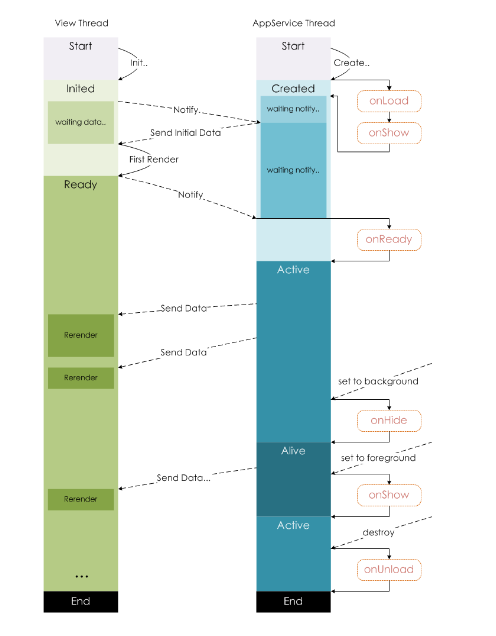
2)页面载入后触发onShow方法，显示页面。

3)首次显示页面，会触发onReady方法，渲染页面元素和样式，一个页面只会调用一次。

4)当小程序后台运行或跳转到其他页面时，触发onHide方法。

5)当小程序有后台进入到前台运行或重新进入页面时，触发onShow方法。

6)当使用重定向方法wx.redirectTo(OBJECT)或关闭当前页返回上一页 wx.navigateBack()，触发onUnload



**4.文件语言说明**



**4.1.wxml组件**

与html中 **<article〉** 、 **<section>** 这样的语义标签不同，小程序中的标签定义为的组件，每个组件都有自己背后的职能和使用方式。

**4.2. rpx单位**

rpx单位是微信小程序中css的尺寸单位，rpx可以根据屏幕宽度进行自适应 。每 750rpx 等价于当前设备的屏幕宽度.

**4.3. js**

小程序支持 ES6 标准的 JavaScript，但窗口级的 JavaScript 却完全被废弃掉了，开发者无法用 JavaScript 去调用 window、document 对象来修改界面元素完成逻辑。小程序中的 JavaScript 其实直接对应 Node.js 的用法，用来完成后台业务逻辑，而不是直接控制交互。无法操作DOM，通过改变page data（类似React的state）来改变视图展现。小程序的这个设计，使其可以用到 Virtual Dom 的方式来渲染界面，让界面数据更新时的性能优化成为可能，但付出的代价就是少了窗口级 JavaScript 的那层胶水黏合，使得很多功能的开发变得呆板和繁复。

**4.4. App.js**

因为在它内部注册的变量或方法，都是可以被所有页面获取到，利用它也可以处理一些跨页面事件触发问题。而且可以注册globalData供所有页面取用，例如可以把systemInfo直接注册到globalData中，这样就不用在每个页面都获取一遍.

**4.5 app.json**

app.json文件来对微信小程序进行全局配置，决定页面文件的路径、窗口表现、设置网络超时时间、设置多 tab 等

三. 小程序推荐

**1.rpx**

rpx单位是微信小程序中css的尺寸单位，rpx可以根据屏幕宽度进行自适应 。

**2.flex**

flex布局可用来为盒状模型提供最大的灵活性。

四. 工具

**1.WEPT（微信小程序实时开发工具）**

<https://github.com/chemzqm/wept>

**2.ngrok（内网映射工具）**

<http://www.wxdev.org/>

**3.阿里巴巴矢量图标库官网**

<http://iconfont.cn/>

**4.小程序API**

<https://mp.weixin.qq.com/debug/wxadoc/dev/api/?t=2017112>