# 1 基础篇

## 1.1 请谈谈微信小程序主要目录和文件的作用

* project.config.json 项目配置文件，用得最多的就是配置是否开启https校验；
* App.js 设置一些全局的基础数据等；
* App.json 底部tab, 标题栏和路由等设置；分包预加载
* App.wxss 公共样式，引入iconfont等；引入外部组件库
* pages 里面包含一个个具体的页面；
* index.json (配置当前页面标题和引入组件等)
* index.wxml (页面结构)；
* index.wxss (页面样式表)；
* index.js (页面的逻辑，请求和数据处理等)；

## 1.2 请谈谈wxml与标准的html的异同

* 都是用来描述页面的结构；
* 都由标签、属性等构成；
* 标签名字不一样，且小程序标签更少，单一标签更多；
* 可以使用wx:if 和wx:for等语句
* WXML在微信小程序中运行，HTML在浏览器中运行
* 小程序没有DOM树和window对象

## 1.3请谈谈WXSS和CSS的异同

* 都是用来描述页面的样式；
* WXSS 具有 CSS 大部分的特性，也做了一些扩充和修改；
* WXSS使用 rpx作为单位；
* WXSS 仅支持部分 CSS 选择器

## 1.4 小程序页面间有哪些传递数据的方法

* 使用全局变量实现数据传递
* 页面跳转或重定向时，使用url带参数传递数据，使用options进行接收
* 使用缓存传递参数

## 1.5 小程序的生命周期函数

* onLoad() 页面加载时触发，只会调用一次，可获取当前页面路径中的参数。
* onShow() 页面显示/切入前台时触发，一般用来发送数据请求；
* onReady() 页面初次渲染完成时触发, 只会调用一次，代表页面已可和视图层进行交互。
* onHide() 页面隐藏/切入后台时触发, 如底部 tab 切换到其他页面或小程序切入后台等。
* onUnload() 页面卸载时触发，如redirectTo或navigateBack到其他页面时。

## 1.6 微信小程序原理

* 小程序本质就是一个单页面应用，所有的页面渲染和事件处理，都在一个页面内进行，但又可以通过微信客户端调用原生的各种接口；
* 它的架构，是数据驱动的架构模式，它的UI和数据是分离的，所有的页面更新，都需要通过对数据的更改来实现；
* 它从技术讲和现有的前端开发差不多，采用JavaScript、WXML、WXSS三种技术进行开发；
* 功能可分为渲染层和逻辑层两个部分；
* 渲染层：使用webview线程展现UI；逻辑层：使用JSCore线程处理业务逻辑、数据及接口调用；两个线程通过微信客户端进行通信

## 1.7 页面跳转函数的区别

* wx.navigateTo()：保留当前页面，跳转到应用内的某个页面。但是不能跳到 tabbar 页面
* wx.redirectTo()：关闭当前页面，跳转到应用内的某个页面。但是不允许跳转到 tabbar 页面
* wx.switchTab()：跳转到 tabBar 页面，并关闭其他所有非 tabBar 页面
* wx.navigateBack()关闭当前页面，返回上一页面或多级页面。可通过 getCurrentPages() 获取当前的页面栈，决定需要返回几层
* wx.reLaunch()：关闭所有页面，打开到应用内的某个页面

## 1.8 数据请求的封装

* 使用promise和wx.request来请求数据，使用success和fail来指定成功或者失败的回调函数
* 在request.js创建封装请求数据的方法
* 在子页面中调用封装的方法请求数据

## 1.9 小程序的优缺点

##### ****小程序的优点****

* 无需下载
* 打开速度快
* 开发成本低
* 为用户提供良好的安全保障。发布有一套严格的审查流程，不能通过审查的程序无法发布上线

##### ****小程序的缺点****

* 大小限制不能超过2M
* 页面栈最多10层

## 1.10 view

类型于html中的div

## 1.11 小程序运行环境

小程序的运行环境分成渲染层和逻辑层，其中 WXML 模板和 WXSS 样式工作在渲染层，JS 脚本工作在逻辑层。

小程序的渲染层和逻辑层分别由2个线程管理：渲染层的界面使用了WebView 进行渲染；逻辑层采用 JsCore 线程运行 JS 脚本。一个小程序存在多个界面，所以渲染层存在多个 WebView 线程，这两个线程的通信会经由微信客户端（下文中也会采用 Native 来代指微信客户端）做中转，逻辑层发送网络请求也经由 Native 转发，小程序的通信模型下图所示。

