1. 微信小程序

1.1. 能否独立开发

能，

1.2. 开发流程

注册小程序

企业注册：企业类型、营业执照注册号、管理员身份证姓名、管理员身份证号码、管理员手机号码、短信验证等（领导提供）

项目成员

最高权限（运营者，开发者，数据分析）

体验成员

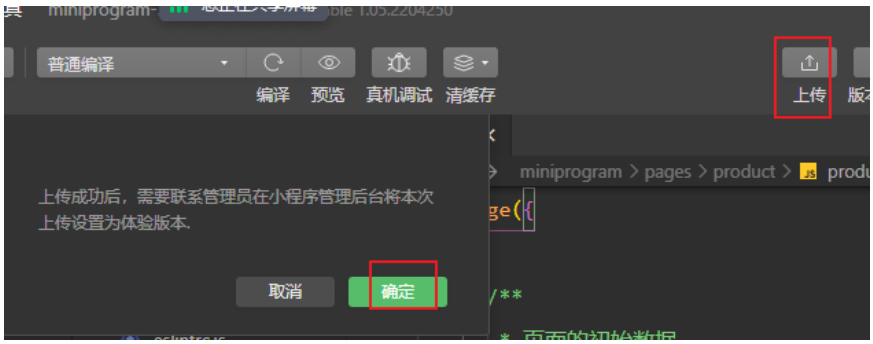
通过设置体验版本，获取体验二维码，体验成员可以扫码体验小程序

appid 的使用：

并不是说任何人都可以使用任何 appid，需要对应小程序的权限才可以

发布

第一步：上传



第二步：体验版本或者提交审核

第三步：提交审核（了解审核规则）

第四步：提交发布小程序

1. 注册微信小程序账号
2. 获取微信小程序的 AppID
3. 登录微信公众平台配置域名
4. 代码上传，提交审核
5. 审核通过即可发布

1.3. 小程序项目和其他项目的区别

小程序是面向微信平台的。部署上线都是直接提交给微信即可。其他项目都需要我们自己设置服务器部署上线。但是小程序包的体积大小有限制。单个 2M，全部 16M

小程序可以借助微信的 api，通过微信，方便的调用手机底层的功能。其他项目不具备。

小程序开发有自己的语法，有自己的开发工具。其他项目没有这些约束。

小程序中没有 bom 对象，dom 对象。类似 mvvm 的开发方式，基本不需要直接操作元素。

小程序只能在微信开发者中预览，其他的可以直接在浏览器上面预览

1.4. 原生开发小程序、wepy、mpvue,uniapp 对比

1.5. 分析微信小程序的优劣势

优势：

小程序部署依赖微信服务器，节省了服务器成本

用户体验好，不用管下载 app，也不用通过浏览器来打开，直接在微信上使用

一次开发，兼容安卓和苹果，开发效率高

劣势：

没有微信，就不能使用微信小程序，跟微信紧密结合
后台调试麻烦，只接受 HTTPS 请求
开发过程中调试必须依靠开发者工具

1.6. 小程序和原生 APP 的区别

小程序无需安装、无需卸载，不占储存空间，用完就走
小程序开发成本低，开发速度快
运行速度比原生的 APP 慢
微信客户群体大，推广容易

1.7. 小程序页面组成

由 4 部分组成 = wxml（结构）+wxss（样式）+json（配置）+js（逻辑）

1.8. 微信小程序主要目录和文件的作用

（1）Project.config.json：项目配置文件，用的最多的就是是否开启 HTTPS 校验；
（2）APP.js：设置全局基础数据等；
（3）APP.json：设置底部 tab、标题栏、路由等；
（4）APP.wxss：设置公共样式；
（5）Pages：放页面的目录
（6）Index.json：配置当前页面标题和引入组件；
（7）Index.wxss：写样式；
（8）Index.wxml：就是 HTML 页面；
（9）Index.js：就是 JS 文件；

1.9. Wxml 与 HTML 异同

wxml 由小程序负责编译，html 由浏览器负责渲染
html 写的是标签，小程序写的是组件
小程序里面可以写判断，循环，绑定变量，但是 html 中不可以。
小程序没有 DOM 树和 window 对象

1.10. Wxss 和 css 的异同：

Wxss 新增了 rpx 单位；
Wxss 不支持个别选择器，比如*号

Wxss 提供全局样式和局部样式

1.11. 微信小程序请求封装

在 utils 目录下创建 config.js 和 http.js 两个文件
Config.js 文件配置项目环境，分生产环境、开发环境
http.js 封装请求方法、设置请求体、带上 token 和异常处理等
在需要发送请求的地方引入 http 就好了

1.12. 开发环境配置

在开发的时候，上线的时候，测试的时候，我们调用的接口路径可能是不同的。那么我们就需要配置不同的开发环境。
通过封装 http.js，设置设置不同的 baseURL。设置不同的接口路径。然后，对应接口路径，需要在管理后台添加才可以。

1.13. 小程序页面间有哪些传递数据的方法

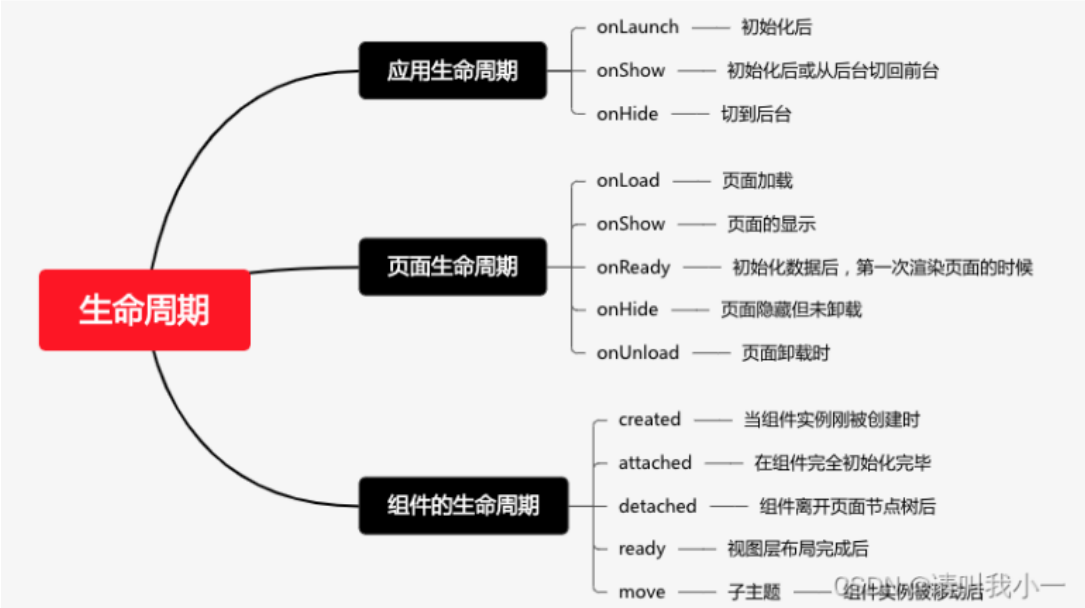
使用全局变量实现数据的传递；（geApp()获取 app 实例）
页面跳转的时候，通过 URL 传参传递数据
使用本地缓存传递数据

1.14. 小程序的双向绑定和 Vue 的区别：

Vue:通过 v-model 来实现
<input v-model=”msg”/>
data(){
return{ msg:”jack”}
}

小程序:通过 model:value 来实现
<input model:value=”{{msg}}”/>
data:{
msg:”lucy”
}

1.15. 小程序的生命周期



页面的主要生命周期函数：

```
Onload（）：加载函数只执行一次；

Onshow（）：页面显示时执行；

Onready（）：页面初次渲染完成时触发，只调用一次；

Onhide（）：页面隐藏时触发，如切换到其他页面；

onUnload（）：页面卸载时触发，如 navigateBack 到其他页面
```

组件的主要生命周期函数：

```
lifetimes:{
  created(): void;
  attached(): void;
  ready(): void;
  moved(): void;
  detached(): void;
```

```
  error(err: Error): void;

 },
```

组件中使用页面的生命周期

```
pageLifetimes:{

  show(){},

  hide(){}

 },
```

1.16. Webview 是什么？怎么用？

Webview 是一个基于 webkit 的引擎，可以解析 DOM 元素，展示 html 页面的控件，它和浏览器展示页面的原理是相同的，所以可以把它当做内置浏览器看待。

用法（需要提前配置业务域名）
<web-view src="地址"></web-view>

使用 web-view 的场景
需要展示 html 页面，而又不使用(或无法使用)浏览器的环境下，就可以使用 webview 了

Webview 中的页面如何返回小程序
通过 wx.miniProgram 接口返回小程序，调用导航相关的 api 如 navigateTo，即可返回小程序的页面

1.17. 小程序调用后台的接口遇到哪些问题

数据的大小限制：超出范围会直接导致小程序崩溃，需要重启小程序
小程序不可以直接渲染文章内容这种类型的 html 文本，显示需要借助组件

1.18. 小程序如何实现下拉刷新

一般情况：设定页面 json 文件的"enablePullDownRefresh"节点为 true
如果使用 scroll-view：需要添加 refresher-enabled 为 true
1. 开启下拉刷新

```
{
```

```
"enablePullDownRefresh": true
}
```

2. 监听下拉刷新事件

```
onPullDownRefresh() {
  //下拉刷新
  //重新请求需要展示的数据，重新赋值即可
},
```

1.19. 如何实现触底加载

```
onReachBottom() {
  //1.请求当前页数据
  //2.请求数据成功，添加到数据列表，当前页+1
},
```

1.20. bindtap 和 catchtap 的区别：

Bind 事件会冒泡
Catch 事件不会冒泡

1.21. image 组件的 mode 属性有哪些值

<image mode="widthFix" src="{{src}}"></image>

```
mode="widthFix"
```

1.22. 说明一下微信的路由跳转

Wx.navigateTo（）：保留当前页面，跳转到其他非 tabBar 页面，
Wx.redirectTo（）：关闭当前页面，跳转到其他非 tabBar 页面；
Wx.swichTab（）：跳转到 Tabbar 页面，关闭其他非 tabbar 页面；
Wx.navigateBack（）：关闭当前页面。返回上一层页面或者多级页面；
Wx.relaunch（）：关闭所有页面，再跳转到目标页面；

1.23. 小程序登录流程：

在小程序调用 wx.login 获取 code；
使用 wx.request 将 code 发送给后台，到后台获取 openID、unionID,进行校验或者保存，生成 token，返回给小程序
小程序将 token 保存到 storage，后续访问可以携带 token（请求封装）

1.24. 支付流程：

1. 调用后台下单接口 （后台调用微信公众平台下单接口，返回预付单标识，返回参数）
2. 通过参数，调用 wx.requestPament 方法
3. 用户进行权限校验（输密码）
4. 返回到小程序页面，小程序根据订单号查询数据库，显示订单详情

1.25. 小程序的打包优化

因为小程序的发布是使用的微信的服务器，不需要我们自己架设服务器。考虑到微信小程序用户体量较大。所以微信给用户设了限制。

限制一：单个分包/主包大小不能超过 2M
限制二：整个小程序所有分包体积不能大于 16M（主包+分包）

小程序打包原则
小程序会按照 subPackages 的配置进行分包，subPackages 之外的目录将会被打包到主包中
主包也可以有自己的 pages
tabBar 页面必须在主包内
分包之间不能相互嵌套

引用原则

主包无法引用分包的私有资源

分包之间不能相互引用私有资源

分包可以引用主包内的公共资源

分包预下载

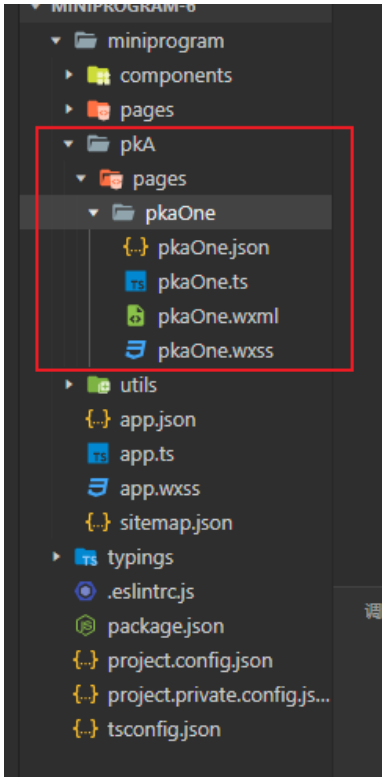
比如我们有 1 个主包，5 个分包，

刚进入程序的时候，只加载了一个主包，当主包加载完毕，不管他要不要继续访问，我们可以设置预先加载其他的包

跳转到分包

```
<navigator url="/pkA/pages/pkaOne/pkaOne">跳转</navigator>
```

页面结构



分包的配置

方案
分包是最直接能够减少代码包体积，提升渲染效率的一个方案
分离静态资源，比如图片，css 等内容，部署到 cdn 上最好
删除无用的代码
通过组件和逻辑的复用，减少重复代码
开发者工具上开发代码压缩
第三方组件库配置按需加载
本地图片压缩

1.26. 小程序加载优化

使用 redirectTo 或者 switchTabBar，及时关闭多余的页面

通过分包控制小程序包的大小，提高初次加载速度

通过分包预加载，提前加载下个页面的数据，可以后续的加载速度
图片压缩，提高图片的加载速度
优化接口的响应速度
使用使用缓存，优化加载速度

1.27. 小程序还有哪些功能

客服功能，录音，视频，音频，地图，定位，拍照，动画，canvas 等

1.28. 小程序性能和体验优化方法

1.小程序应避免出现任何 JavaScript 异常

出现 JavaScript 异常可能导致小程序的交互无法进行下去，我们应当追求零异常，保证小程序的高鲁棒性和高可用性

2.小程序所有请求应响应正常

请求失败可能导致小程序的交互无法进行下去，应当保证所有请求都能成功

3.所有请求的耗时不应太久

请求的耗时太长会让用户一直等待甚至离开，应当优化好服务器处理时间、减小回包大小，让请求快速响应

4.避免短时间内发起太多的图片请求

短时间内发起太多图片请求会触发浏览器并行加载的限制，可能导致图片加载慢，用户一直处理等待。应该合理控制数量，可考虑使用雪碧图技术或在屏幕外的图片使用懒加载

5.避免短时间内发起太多的请求

短时间内发起太多请求会触发小程序并行请求数量的限制，同时太多请求也可能导致加载慢等问题，应合理控制请求数量，甚至做请求的合并等

6.避免 setData 的数据过大

setData 工作原理

小程序的视图层目前使用 WebView 作为渲染载体，而逻辑层是由独立的 JavascriptCore 作为运行环境。在架构上，WebView 和 JavascriptCore 都是独立的模块，并不具备数据直接共享的通道。当前，视图层和逻辑层的数据传输，实际上通过两边提供的 evaluateJavascript 所实现。即用户传输的数据，需要将其转换为字符串形式传递，同时把转换后的数据内容拼接成一份 JS 脚本，再通过执行 JS 脚本的形式传递到两边独立环境。

而 evaluateJavascript 的执行会受很多方面的影响，数据到达视图层并不是实时的。

由于小程序运行逻辑线程与渲染线程之上，setData 的调用会把数据从逻辑层传到渲染层，数据太大会增加通信时间

常见的 setData 操作错误

频繁的去 setData

Android 下用户在滑动时会感觉到卡顿，操作反馈延迟严重，因为 JS 线程一直在编译执行渲染，未能及时将用户操作事件传递到逻辑层，逻辑层亦无法及时将操作处理结果及时传递到视图层

染有出现延时，由于 WebView 的 JS 线程一直处于忙碌状态，逻辑层到页面层的通信耗时上升，视图层收到的数据消息时距离发出时间已经过去了几百毫秒，渲染的结果并不实时

每次 setData 都传递大量新数据

由 setData 的底层实现可知，数据传输实际是一次 evaluateJavascript 脚本过程，当数据量过大时会增加脚本的编译执行时间，占用 WebView JS 线程

后台态页面进行 setData

当页面进入后台态（用户不可见），不应该继续去进行 setData，后台态页面的渲染用户是无法感受的，另外后台态页面去 setData 也会抢占前台页面的执行

避免 setData 的调用过于频繁

setData 接口的调用涉及逻辑层与渲染层间的线程通过，通信过于频繁可能导致处理队列阻塞，界面渲染不及时而导致卡顿，应避免无用的频繁调用

7.避免将未绑定在 WXML 的变量传入 setData

setData 操作会引起框架处理一些渲染界面相关的工作，一个未绑定的变量意味着与界面渲染无关，传入 setData 会造成不必要的性能消耗

8.合理设置可点击元素的响应区域大小

我们应该合理地设置好可点击元素的响应区域大小，如果过小会导致用户很难点中，体验很差

9.避免渲染界面的耗时过长

渲染界面的耗时过长会让用户觉得卡顿，体验较差，出现这一情况时，需要校验下是否同时渲染的区域太大

10.避免执行脚本的耗时过长

执行脚本的耗时过长会让用户觉得卡顿，体验较差，出现这一情况时，需要确认并优化脚本的逻辑

11.对网络请求做必要的缓存以避免多余的请求

发起网络请求总会让用户等待，可能造成不好的体验，应尽量避免多余的请求，比如对同样的请求进行缓存

12.wxss 覆盖率较高，较少或没有引入未被使用的样式

按需引入 wxss 资源，如果小程序中存在大量未使用的样式，会增加小程序包体积大小，从而在一定程度上影响加载速度

13.文字颜色与背景色搭配较好，适宜的颜色对比度更方便用户阅读

文字颜色与背景色需要搭配得当，适宜的颜色对比度可以让用户更好地阅读，提升小程序的用户体验

14.所有资源请求都建议使用 HTTPS

使用 HTTPS，可以让你的小程序更加安全，而 HTTP 是明文传输的，存在可能被篡改内容的风险

15.不使用废弃接口

使用即将废弃或已废弃接口，可能导致小程序运行不正常。一般而言，接口不会立即去掉，但保险起见，建议不要使用，避免后续小程序突然运行异常

16.避免过大的 WXML 节点数目

建议一个页面使用少于 1000 个 WXML 节点，节点树深度少于 30 层，子节点数不大于 60 个。一个太大的 WXML 节点树会增加内存的使用，样式重排时间也会更长

17.避免将不可能被访问到的页面打包在小程序包里

小程序的包大小会影响加载时间，应该尽量控制包体积大小，避免将不会被使用的文件打包进去

18.及时回收定时器

定时器是全局的，并不是跟页面绑定的，当页面因后退被销毁时，定时器应注意手动回收

19.避免使用 css‘:active’伪类来实现点击态

使用 css‘:active’伪类来实现点击态，很容易触发，并且滚动或滑动时点击态不会消失，体验较差

建议使用小程序内置组件的‘hover-*’属性来实现

滚动区域可开启惯性滚动以增强体验

惯性滚动会使滚动比较顺畅，在安卓下默认有惯性滚动，而在 iOS 下需要额外设置

-webkit-overflow-scrolling: touch 的样式

1.29. rpx、px、em、rem、%、vh、vw 的区别是什么?

rpx 相当于把屏幕宽度分为 750 份，1 份就是 1rpx

px 绝对单位，页面按精确像素展示

em 相对单位，相对于它的父节点字体进行计算

rem 相对单位，相对根节点 html 的字体大小来计算

% 一般来说就是相对于父元素

vh 视窗高度，1vh 等于视窗高度的 1%

vw 视窗宽度，1vw 等于视窗宽度的 1%