2024前端高频—小程序&Git万字总结

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，用的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
- **App.wxss**：公共样式，引入iconfont等
- **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
- **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

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
- 多了一些 wx:if 这样的属性以及{{}} 这样的表达式
- WXML仅能在微信小程序开发者工具中预览，而HTML可以在浏览器内预览
- 组件封装不同，WXML对组件进行了重新封装
- 小程序运行在JS Core内，没有DOM树和windiw对象，小程序中无法使用window对象和document对象。

3.请谈谈WXSS和CSS的异同？

都是用来描述页面的样子

- WXSS具有CSS大部分的特性，也做了一些扩充和修改
- WXSS新增了尺寸单位，WXSS在底层支持新的尺寸单位rpx
- WXSS仅支持部分CSS选择器
- WXSS提供全局样式与局部样式

4.你是怎么封装微信小程序的数据请求的？

1. 在根目录下创建util目录及api.js文件和apiConfig.js文件
2. 在apiConfig.js封装基础的get，post和put，upload等请求方法，设置请求体，带上token和异常处理等
3. 在api中引入apiConfig.js封装好的请求方法.根据页面数据请求的urls，设置对应的方法并导出
4. 在具体的页面中导入或将所有的接口放在统一的js文件中并导出
5. 在app.js中创建封装请求数据的方法
6. 在子页面中调用封装的请求数据

5.小程序页面之间有哪些（传值）传递数据的方法？

1. 页面跳转或重定向时，使用url带参数传递数据
2. 使用组件模板 template传递参数

3. 使用缓存传递参数
4. 使用数据库传递参数
5. 给html元素添加data-*属性来传递值，然后通过e.currentTarget.dataset或onload的param参数获取（data- 名称不能有大写字母，不可以存放对象）
6. 设置id 的方法标识来传值，通过e.currentTarget.id获取设置的id值，然后通过设置全局对象的方式来传递数据
7. 在navigator中添加参数数值

6.请谈谈小程序的双向绑定和vue的异同？

大体相同，但小程序之间this.data的属性是不可以同步到视图的，必须调用this.setData()方法

7.请谈谈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/切入前台时候触发，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

8.简述微信小程序原理

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**webview**和**APPService**两个部分：

webview主要用来展示UI，appservice用来处理业务逻辑，数据及接口调用，它们在两个进程中进行，通过系统层JSBridge实现通信，实现UI的渲染，事件处理；appService有来处理业务逻辑、数据及接口调用；

两个部分在两个进程中运行，通过系统层JSBridge实现通信，实现UI的渲染、事件的处理等。 javaScript的代码是运行在微信App中的，因此一些h5技术的应用需要微信APP提供对应的API支持 wxml 微信自己基于xml语法开发的，因此在开发时只能使用微信提供的现有标签，html的标签是无法使用的 wxss具有css的大部分特性，但并不是所有都支持，没有详细文档（wxss的图片引入需要使用外链地址，没有body，样式可以使用import导入）

9.请谈谈原生开发小程序，wepy，mpvue的对比？

个人认为，如果是新项目，且没有旧的 h5 项目迁移，则考虑用小程序原生开发，好处是相比于第三方框架，坑少。 而如果有 老的 h5 项目是 vue 开发 或者 也有 h5 项目也需要小程序开发，则比较适合 wepy 或者 mpvue 来做迁移或者开发，近期看wepy几乎不更新了，所以推荐美团的mpvue。 而如果如果团队前端强大，自己做一套框架也没问题。

10.简单描述下微信小程序的 相关文件类型

1. wxml 模板文件，是框架设计的一套标签预言，结合基础组件，事件系统，可以构建出页面的结构 wxss 样式文件，是一套样式语言，用于描述WXML的组件样式 js脚本逻辑文件。逻辑处理网络请求 json配置文件，小程序设置，如页面注册，页面标题及tabBar
2. app.json是整个小程序的全局配置，包括：
    - pages:所有页面路径
    - 网络设置（网络超时事件）
    - 页面表现（页面注册）
    - window：（背景色，导航样式，默认标题）
    - 底部tab等

3. app.js 监听并处理小程序的生命周期函数，声明全局变量
4. app.wxss 全局配置的样式文件

11.那些方法来提高微信小程序的应用速度？

- 提高页面的加载速度
- 用户行为预测
- 减少默认的data的大小
- 组件化方案

12.分析微信小程序的优劣势？

优势：

- 容易上手，基础组件库比较全，基本不需要考虑兼容问题
- 开发文档比较完善，开发社区比较活跃，支持插件式开发
- 良好的用户体验，无需下载，通过搜索和扫一扫就可以打开，打开速度快，安卓上可以添加到桌面，与原生APP差不多
- 开发成本比APP要低
- 为用户提供良好的保障（小程序发布，严格是审查流程）

劣势：

- 限制较多，页面大小不能超过1M，不能打开超过5个层级的页面
- 样式单一，部分组件已经是成型的，样式不可修改，例如：幻灯片，导航
- 推广面窄，不能分享朋友圈，只能通过分享给朋友，附加小程序推广
- 依托与微信，无法开发后台管理功能
- 后台调试麻烦，因为api接口必须https请求且公网地址
- 真机测试，个别安卓和苹果表现迥异，例如安卓的定位功能加载很慢

13.微信小程序和H5的区别？

- 运行环境不同（小程序在微信运行，h5在浏览器运行）
- 开发成本不同（h5需要兼容不同的浏览器）
- 获取系统权限不同（系统级权限可以和小程序无缝衔接）
- 应用在生成环境的运行速度流程（h5需不断对项目优化来提高用户体验）

14.怎么解决微信小程序的异步请求问题？

在回调函数中调用下一个组件的函数

```
*/app.js*/

success:function(info){

  that.apirtnCallback(info)

}
*/index.js*/

onLoad:function(){
```

```
    app.apirtnCallback = res =>{

        console.log(res)

    }

}
```

15.小程序关联微信公众号如何确定用户的唯一性？

使用wx.getUserInfo方法 withCredentials为true时，可获取encryptedData，里面有union_id，后端需要进行对称解密

16.使用webview直接加载要注意那些事项？

- 必须要在小程序后台使用管理员添加业务域名
- h5页面跳转至小程序的脚步必须是1.3.1以上
- 微信分享只可以是小程序的主名称，如要自定义分享内容，需小程序版本在1.7.1以上
- h5的支付不可以是微信公众号的appid，必须是小程序的appid，而且用户的openid也必须是用户和小程序的

17.小程序调用后台接口遇到那些问题？

- 数据的大小限制，超过范围会直接导致整个小程序崩溃，除非重启小程序
- 小程序不可以直接渲染文章内容这类型的html文本，显示需要借助插件

  注：插件渲染会导致页面加载变慢，建议在后台对文章内容的html进行过滤，后台直接处理批量替换p标签div标签为view标签，然后其他的标签让插件来做

18.微信小程序如何实现下拉刷新？

用view代替scroll-view，设置onPullDownRefresh函数实现

19.webview中的页面怎么跳转回小程序?

```
wx.miniProgram.navigateTo({
    url:'pages/login/login'+'$params'
})

wx.miniProgram.switchTab({
    url:'/pages/index/index'
})
```

20.bindtap和catchtap的区别？

bind事件绑定不会阻止冒泡事件向上冒泡 catch事件绑定可以阻止冒泡事件向上冒泡

21.简述 wx.navigateTo(),wx.redirectTo(),wx.switchTab(),wx.navigateBack(),wx.reLaunch() 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。但是不能跳转 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

22.小程序和Vue写法的区别？

1. 遍历的时候：小程序wx:for="list", 而Vue是v-for="item in list"
2. 调用data模型（赋值）的时候：

   小程序：this.data.item // 调用，this.setDate({item:1})//赋值

   Vue：this.item //调用，this.item=1 //赋值

23.小程序与原生App那个好？

各有各自的优点，都又有缺点

小程序的优点：

- 基于微信平台开发，享受微信自带的流量，这个优点最大
- 无需安装，只要打开微信就能用，不占手机内存，体验好
- 开发周期段，一般最多一个月就可以上线完成
- 开发所需的资金少，所需资金是开发原生APP的一半不到
- 小程序名称是唯一的，在微信的搜索里权重很高
- 容易上手，只要之前有HTML+CSS+JS基础知识，写小程序基本没有大问题
- 基本不需要考虑兼容性问题，只要微信可以正常运行的机器，就可以运行小程序
- 发布，审核高效，基本上午发布审核，下午就审核通过，升级简单，支持灰度发布
- 开发文档完善，社区活跃
- 支持插件式开发，一些基本功能可以开发成插件，供多个小程序使用

  缺点：

- 局限性很强（比如页面大小不能超过1M，不能打开超过5个层级的页面，样式单一，小程序的部分组件已经是成型的了，样式不能修改，比如幻灯片，导航）只能依赖于微信依托与微信，无法开发后台管理功能
- 不利于推广，推广面窄，不能分享朋友圈，只能分享给朋友，附近小程序推广，其中附加小程序也收到微信限制
- 后台调试麻烦，因为API接口必须https请求，且公网地址，也就是说后台代码必须发布到远程服务器上；当然我们可以修改host进行dns映射把远程服务器转到本地，或者开启tomcat远程调试；不管怎么说终归调试比较麻烦
- 前台测试有诸多坑，最头疼莫过于模拟器与真机显示不一致
- js引用只能使用绝对路径，不能操作DOM

原生App优点：

- 原生的相应速度快
- 对于有无网络操作时，譬如离线操作基本选用原生开发
- 需要调用系统硬件的功能（摄像头，拨号，短信蓝牙..）
- 在无网络或者弱网情况下体验好

  原生App缺点：

- 开发周期长，开发成本高，需要下载

24.小程序的发布流程（开发流程）

1. 注册微信小程序账号
2. 获取微信小程序的AppID
3. 下载微信小程序开发者工具
4. 创建demo项目
5. 去微信公众号配置域名
6. 手机浏览
7. 代码上传
8. 提交审核
9. 小程序发布

25.webview中的页面怎么跳回小程序中？

```
<script type="text/javascript" src="<https://res.wx.qq.com/open/js/jweixin-1.3.0.js>"></script
>

$(".kaiqi").click(function(){
        wx.miniProgram.redirectTo({url: '/pages/indexTwo/indexTwo'})
});
```

26.小程序授权登录流程

授权，微信登录获取code，微信登录，获取 iv , encryptedData 传到服务器后台，如果没有注册就需要注册。

27.小程序支付如何实现？

```
    'timeStamp': '',
    'nonceStr': '',
    'package': '',
    'signType': 'MD5',
    'paySign': '',
    'success':function(res){},
    'fail':function(res){},
    'complete':function(res){}
})
```

28.小程序还有那些功能？

客服功能，录音，视频，音频，地图，定位，拍照，动画，canvas

29\. 小程序的常见问题：

- rpx：小程序的尺寸单位，规定屏幕为750rpx，可适配不同分辨率屏幕 本地资源无法通过wxss获取：background-image：可以使用网络图片，或者base64，或者使用标签
- wx.navigateTo无法打开页面：一个应用同时只能打开5个页面，请避免多层级的交互方式，或使用wx.redirectTo
- tabBar设置不显示：

  1.tabBar的数量少于2项或超过5项都不会显示。

  2.tabBar写法错误导致不会显示。

  3.tabBar没有写pagePath字段（程序启动后显示的第一个页面）

30\. 什么是uni-app

`uni-app` 是一个使用 Vue.js (opens new window)开发所有前端应用的框架，开发者编写一套代码，可发布到iOS、Android、Web（响应式）、以及各种小程序（微信/支付宝/百度/头条/飞书/QQ/快手/钉钉/淘宝）、快应用等多个平台。

1\. git常用命令

1.1、git init

初始化一个 Git 仓库，它将创建一个 .git 文件夹，后续的操作记录都会在此文件夹里，相当于 Git 的数据库。

1.2、git remote add origin 远程仓库地址

将本地仓库和远程仓库关联，origin 是远程仓库的名字，是 Git 的默认叫法。关联之后，我们就可以将本地的提交历史推送到远程仓库，完成和其他人的协同工作了。

1.3、git remote -v

查看关联的远程仓库列表，返回远程仓库名和 URL：

```
$ git remote -v
origin  <https://github.com/schacon/ticgit> (fetch)
```

1.4、git status

显示当前工作目录和暂存区的状态，例如创建了一个文件，此时 git status 就会在 Untracked files 里显示该未追踪的文件，如果将该文件 add 了之后，就会在 Changes to be committed 看到，即已经加到缓存区，等待提交。最后，当我们 commit，就会发现没有任何修改和未提交的文件了。

1.5、git add \[file\] 、git add .

用于将已修改或未跟踪的文件添加到暂存区

1.6、git commit -m "提交日志"

将暂存区的文件提交到本地仓库

1.7、git log --oneline

查看提交的日志信息

1.8、git diff

查看工作区的文件和暂存区的不同之处

1.9、git push origin <本地分支名>

将本地的分支推送到 origin 远程仓库的上，第一次推送远程仓库将会在服务器上创建对应的分支，当第一次推送完后，后续可以直接使用 git push 这种简介用法了

1.10、git pull

将远程仓库的最新内容合并到本地仓库里

1.11、git merge <其他分支名>

将其他分支里的提交内容合并到当前分支里

1.12、git merge --abort

合并是有可能有冲突的，如果冲突后想放弃合并，可以使用这个命令

1.13、git branch 分支名称

创建新的分支

1.14、git checkout 分支名称

切换到其他分支上

1.15、git checkout -b 新分支名称

相当于上面连个命令的合体功能，即创建新分支，然后切换到新分支上。

1.16、git branch、git branch -r、git branch -a

分别是查看本地分支、查看远程分支、查看所有分支

1.17、git branch -d 分支名称

删除分支，如果该分支没有合并过，则会提示相应错误，如果想要强制删除，可使用 `git branch -D` 分支名称

1.18、git clean -f

删除本地仓库中未跟踪的文件，如果想删除的是目录，使用 `-d`

1.19、git reset

用于重置暂存区的文件与上一次的提交(commit)保持一致，但不会重置工作区的修改，需使用 `git checkout <文件名>` 命令才能重置工作区的改动。或者使用比较危险的 `git reset --hard HEAD` 命令，会将工作区和暂存区都重置到上一次版本，包括 commit 信息。

1.20、git rm 文件、git rm --cached 文件名

如果只是简单的在工作目录里手动删除文件，则还需要自己将修改添加到暂存区，然后再提交到本地仓库里才完成一次改动版本的记录。`git rm 文件` 则帮我们在删除的同时，也将修改添加到了暂存区，少了一步的操作。

但有时候我们想保留该文件，以便后续使用，但又想把删除的改动添加到暂存区，此时就可以使用 `git rm --cached 文件名` 命令来达到此效果了。

1.21、git stash 和 git stash pop

如果我们开发到一半，需要重新创建一个新分支去解决线上问题，但此时又不想将当前的分支改动提交到对应分支上，则可以使用 git stash 将修改（包括工作区和暂存区）保存到堆栈中，等新分支处理完毕后，就可以切换到之前的分支，然后使用 `git stash pop` 恢复缓存的堆栈内容了。

2\. git提交时发生冲突，你能解释冲突时如何产生的吗？你是如何解决的？

冲突怎么产生的

开发过程中，我们都有自己的特性分支，所以冲突发生的并不多，但也碰到过。诸如公共类的公共方法，我和别人同时修改同一个文件，他提交后我再提交就会报冲突的错误。

如何解决冲突

1、发生冲突，在IDE里面一般都是对比本地文件和远程分支的文件，然后把远程分支上文件的内容手工修改到本地文件，然后再提交冲突的文件使其保证与远程分支的文件一致，这样才会消除冲突，然后再提交自己修改的部分。特别要注意下，修改本地冲突文件使其与远程仓库的文件保持一致后，需要提交后才能消除冲突，否则无法继续提交。必要时可与同事交流，消除冲突。

2、发生冲突，也可以使用命令。通过git stash命令，把工作区的修改提交到栈区，目的是保存工作区的修改；通过git pull命令，拉取远程分支上的代码并合并到本地分支，目的是消除冲突；通过git stash pop命令，把保存在栈区的修改部分合并到最新的工作空间中；

3\. 如果本次提交失误，如何撤销

如果想撤销提交到索引区的文件，可以通过gitresetHEADfile。如果想撤销提交到本地仓库的文件，可以通过**gitreset–softHEAD^n**，恢复当前分支的版本库至上一次提交的状态，索引区和工作空间不变更。

通过**gitreset–mixedHEAD^n**恢复当前分支的版本库和索引区至上一次提交的状态，工作区不变更。

通过**gitreset–hardHEAD^n**恢复当前分支的版本库、索引区和工作空间至上一次提交的状态。

4\. git和svn有什么区别

- git是分布式版本控制，svn是集中式版本控制（核心区别）
- git相对于svn的优势就是不需要网络即可版本控制
- git把内容按数据方式存储，而svn是按文件
- git可以是公用的，可以分享，svn基本是公司内部才能访问，网外不方便访问
- git不依赖中央服务器，即使服务器有问题也不受影响，svn依赖服务器，一旦服务器有问题就会受影响
- git没有一个全局的版本号，svn有

分布式和集中式的区别：

每个节点的地位都是平等，拥有自己的版本库，在没有网络的情况下，对工作空间内代码的修改可以提交到本地仓库，此时的本地仓库相当于集中式的远程仓库，可以基于本地仓库进行提交、撤销等常规操作，从而方便日常开发

5\. git fetch、git merge、git pull的区别

git pull相当于git fetch和git merge，即更新远程仓库的代码到本地仓库，然后将内容合并到当前分支。 git merge:将内容合并到当前分支 git fetch相当于是从远程获取最新版本到本地，不会自动merge 方便记忆: git pull=git fetch+git merge

6\. Git的rebase和merge的区别是什么？

`git rebase` 和 `git merge` 两个命令都用于从一个分支获取内容并合并到当前分支。

以一个 `feature/todo` 分支合并到 `master` 主分支为例，我们来看一下分别用 `rebase` 和 `merge` 会有什么不同。

使用 Merge

`merge` 会自动创建一个新的 `commit`， 如果合并时遇到冲突的话，只需要修改后重新 `commit`。

- 优点：能记录真实的 `commit` 情况，包括每个分支的详情
- 缺点：由于每次 `merge` 会自动产生一个 `merge commit`，

  因此在使用一些可视化的 git 工具时会看到这些自动产生的 `commit`，

  这些 `commit` 对于程序员来说没有什么特别的意义，多了反而会影响阅读

使用 Rebase

`rebase` 会合并之前的 `commit` 历史。

- 优点：可以得到更简洁的提交历史，去掉了 `merge commit`
- 缺点：因为合并而产生的代码问题，就不容易定位，因为会重写提交历史信息

建议

- 当需要保留详细的合并信息，建议使用 `git merge`, 尤其是要合并到 master 上
- 当发现自己修改某个功能时提交比较频繁，并觉得过多的合并记录信息对自己来说没有必要，那么可尝试使用 `git rebase`