**小程序：**

微信小程序主要目录和文件的作用：

Behaviors的使用：

是用于组件间代码共享的特性，类似于一些编程语言中的“mixins”或“traits”

使用数据监听器：

observers

小程序的生命周期函数:

路由方式：

打开新页面：调用 API [wx.navigateTo](https://link.juejin.im?target=https%3A%2F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Fapi%2Fwx.navigateTo.html" \t "_blank) 或使用组件 [< navigator open-type="navigateTo"/>](https://link.juejin.im?target=https%3A%2F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Fcomponent%2Fnavigator.html" \t "_blank)

页面重定向：调用 API [wx.redirectTo](https://link.juejin.im?target=https%3A%2F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Fapi%2Fwx.redirectTo.html" \t "_blank) 或使用组件 [<navigator open-type="redirectTo"/>](https://link.juejin.im?target=https%3A%2F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Fcomponent%2Fnavigator.html" \t "_blank)

页面返回：调用 API [wx.navigateBack](https://link.juejin.im?target=https%3A%2F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Fapi%2Fwx.navigateBack.html" \t "_blank) 或使用组件[<navigator open-type="navigateBack">](https://link.juejin.im?target=https%3A%2F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Fcomponent%2Fnavigator.html" \t "_blank)或用户按左上角返回按钮

Tab 切换：调用 API [wx.switchTab](https://link.juejin.im?target=https%3A%2F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Fapi%2Fwx.switchTab.html" \t "_blank) 或使用组件 [<navigator open-type="switchTab"/>](https://link.juejin.im?target=https%3A%2F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Fcomponent%2Fnavigator.html" \t "_blank) 或用户切换 Tab

重启动：调用 API [wx.reLaunch](https://link.juejin.im?target=https%3A%2F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Fapi%2Fwx.reLaunch.html" \t "_blank) 或使用组件 [<navigator open-type="reLaunch"/>](https://link.juejin.im?target=https%3A%2F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Fcomponent%2Fnavigator.html" \t "_blank)

小程序实现原理:

* 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用来展现UI，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。

**Vue：**

组件之间的传值：

MVVM模式和MVC模式：

MVVM模式是Model-View-ViewModel的缩写。MVVM是一种设计思想。Model层代码数据模型，也是可以在Model中定义数据修改和操作的业务逻辑；View层代表UI组件，它负责将数据模型转化为UI展示出来，ViewModel是一个同步View和Model的对象。

    在MVVM架构中，View和Model之间没有直接联系，通过ViewModel进行交互，Model和ViewModel之间的交互是双向的，因此View数据变化会同步到Model中，而Model的数据变化也会立即反应到View上。

    ViewModel通过双向数据绑定把View层和Model层连接起来，而View和Model之间的同步工作完全是自动的，无需人为干涉，因开发者只需要关注业务逻辑，不需要手动操作DOM，不需要关注数据状态的同步问题，复杂的数据状态维护完全由MVVM来统一管理。

    主要是MVC中的Controller演变成为MVVM中的ViewModel。MVVM主要解决了MVC中大量的DOM操作使页面渲染性能下降，加载速度变慢，影响用户体验。和当Model频繁变化的时候，开发者需要主动更新到View。

生命周期：

vuex实现原理：

路由实现原理：

阅读源码：

vue双向数据绑定原理：

采用数据劫持结合发布者-订阅者模式的方式，通过Object.defineProperty()来劫持各个属性的setter/getter，在数据变动时发布消息给订阅者，触发相应的监听回调。

    具体步骤如下：

        A.需要observe的数据对象进行递归遍历

        B.compile解析模板指令

        C.Watcher订阅者是Observer和Compile之间通信桥梁

        D.MVVM作为数据绑定的入口，整合Observer、Compile和Watcher三者

**ES6：**

Promise的实现和异步函数的应用：

解构赋值：

数组扩展：

Array.from()、Array.of()、Array.Map()、Array.reduce()、Arr.fill()、Arr.copyWithin()、Arr.find() Arr,findIndex()、遍历相关(for in / let of)

class, extends, super：

set数据结构：（数组去重）

**展开运算符(...)：**

将字符串转成数组 var str="abcd"; console.log([...str]) // ["a", "b", "c", "d"]

将集合转成数组 var sets=new Set([1,2,3,4,5]) console.log([...sets]) // [1, 2, 3, 4, 5]

两个数组的合并 var a1=[1,2,3]; var a2=[4,5,6]; console.log([...a1,...a2]); //[1, 2, 3, 4, 5, 6]

rest参数  …变量名称

rest 参数是一个数组 ，它的后面不能再有参数，不然会报错

**js底层：**

数据类型分类和判断：

基本类型：Undefuned / Null / Boolean / Number / String / Symbol

引用类型：Object对象：对象 / 数组 / 函数

判断：tyoeof / instanceof / constructor / Object.prototype.toString.call()

浅拷贝和深拷贝：

浅拷贝：Object.assign() / Array.prototype.concat() / Array.prototype.slice()

深拷贝：遍历属性

函数库lodash / jquery( $.extend ) / JSON.parse(JSON.stringify) / 手写递归

作用域和作用域链:

全局作用域、函数作用域、块级作用域

闭包:（函数中的函数）

作用：

1. 可以读取函数内部的变量

2. 让这些变量的值始终保持在内存中

工作中的实际应用：（具体的应用 ）

1. 函数作为参数传递
2. 函数作为返回值

缺点：内存泄露，影响性能

This的指向问题:

异步和同步：

JS需要异步的原因是JS是单线程运行的

异步场景：定时任务( setTimeout、setInterval )、网络请求( ajax请求、动态加载 )、事件绑定

原型和原型链：

继承：（常见的六种）

原型链+借用构造函数的组合继承、寄生组合继承、ES6中的class继承.....

DOM操作和BOM操作：

DOM事件模型：捕获和冒泡 （事件流） 阻止冒泡

事件代理（事件委托）

BOM操作：是浏览器本身的一些信息的设置和获取

Ajax和跨域：

手写XMLHttpRequest

跨域：JSONP、CORS（后端）、Node中间件代理或nginx反向代理

存储：

sessionStorage、localStorage和coookie之间的区别

模块化：

**算法：**

冒泡排序:

数组去重：

递归：