1: 获取用户信息方法使用那个api，获取来的信息保存在哪里 不会丢失，在刷新页面的时候；

globalData

1： 赋值方式 ：globalData.XX == XX

2： 组件不能直接使用在WXML中使用globalData， 需要转为自己的属性

2： 某页面JS更新了globalData里面的某个字段，涉及该字段的JS都能拿到更新后的值

- localstorage同步异步存储的差别

- 同步存储的代码执行早于异步代码执行

- 异步存储支持promise，适用于需要根据当前是否存储成功做一些对应的操作

- globalData 用来保存全局的初始化数据，用于全局数据的状态共享; 几个页面依赖于同一个数据的状态，一个页面发生了操作 其他的页面需要响应这个变化

可以在某个模块更改共享的某个数据

- 所有的数据都必须通过setData（）来更新view层面的变化，此部分比较耗费性能

setData的替代方案

小程序组件之间的通讯？？

# 小程序的插槽语法

小程序的模板语法

wsx语法

Behavior：

类似于mixin 后面引入的同名属性会覆盖前面的同名属性

生命周期总是早于组件的生命周期

需要在组件里引入；在组件的behaviors字段使用数组声明

只有微信小程序有这样的写法吗？ 其他的小程序可以这样写吗？

setData

小程序里面只有调用setData才能触发视图更新

taro对于setData的性能进行了优化？ 从哪方面进行的优化 在原生开发中能否实现 ？？？

小程序的生命周期

小程序中怎样引入第三方库

1： 将第三方包下载到本地 ，查看第三方包里面对我暴露接口是在window上 或者是module.exports,修改第三方包的导出方式

小程序中没有window，不能操作DOM

小程序中怎样封装和注册组件

2：怎样可以退出重新进入当前开发的小程序

3： 路由

4： 全局状态

5： 全局数据

6： 生命周期

7： 全局组件

APP里面可以填写哪些数据

小程序的UI框架

DSL

是一种以React JSX与Vue template的语言特性为基础，更符合UI编排的组件化语言。

VUE的DSL 由其他语言进行编译而来吗？

小程序的实现原理

小程序双线程设计的原因：

为了阻止开发者使用浏览器提供的api来操作DOM,执行脚本，跳转页面从而带来安全问题

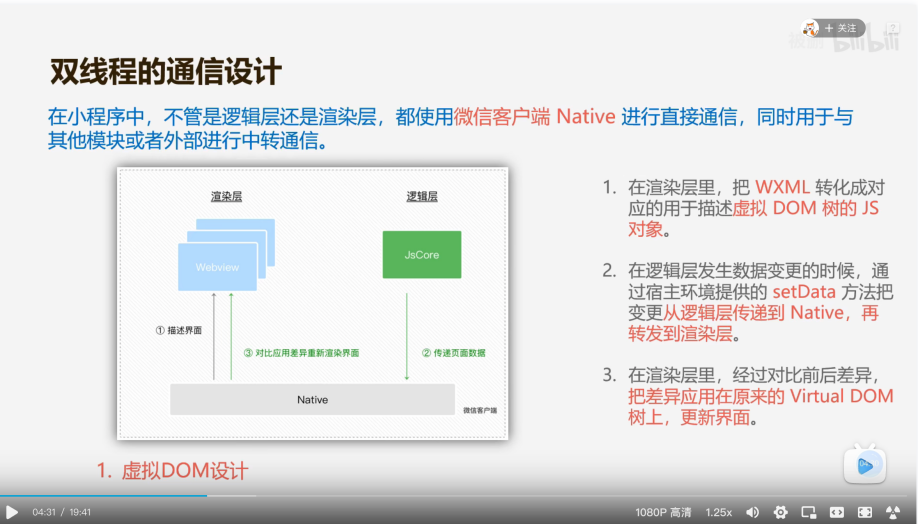
小程序的运行需要依赖相关的宿主环境 比如微信小程序 就会依赖微信客户端提供的一些能力 微信客户端内置JS引擎 没有任何浏览器相关的接口

打开小程序的过程其实是动态下载小程序运行包的过程 动态加载资源包 允许开发者可以在线版本更新？？？和bug修复。

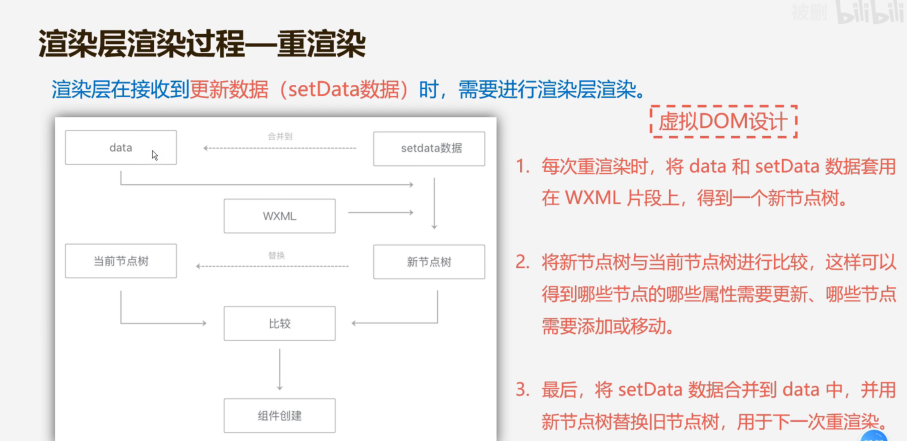
定义在组件data里面的变量、自定义component是会随着页面的注销而销毁但是 未定义在data中的变量会一直在内存中 直到小程序被删除或者缓存被清理后才会重新加载环境

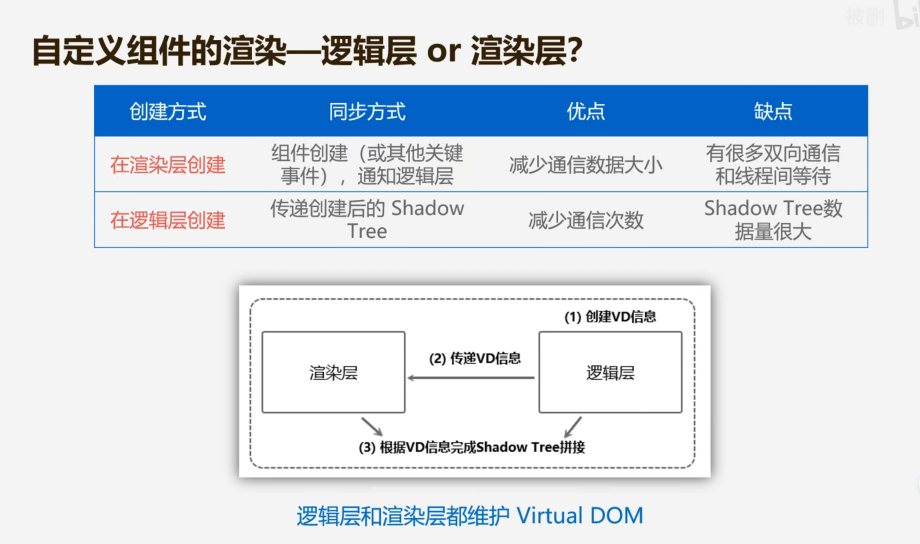
一个小程序只有一个逻辑线程，一个页面是一个单独的webview线程。对于JS每个页面有自己独立的作用域。

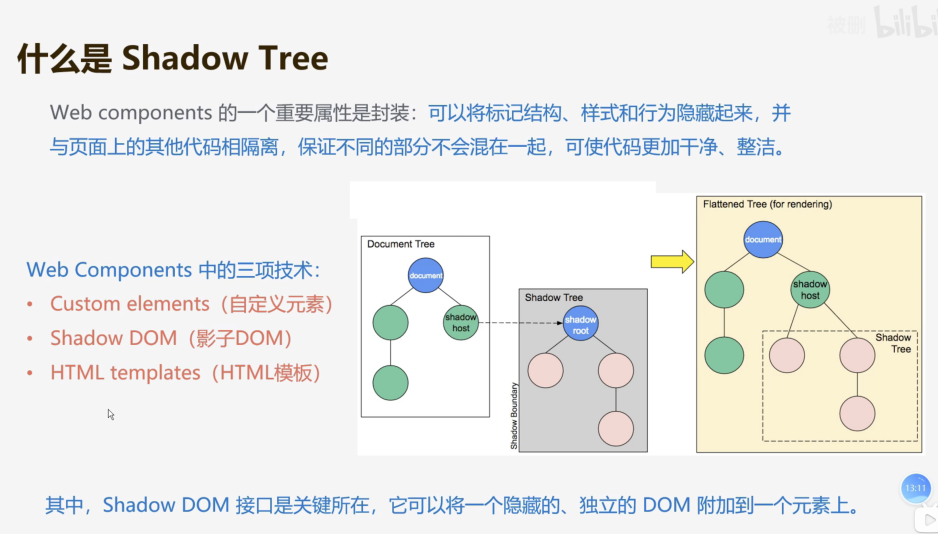
小程序所有的页面被管理在一个栈里面，通过逻辑层的代码控制渲染哪个页面。所以小程序的渲染层存在这多个webview

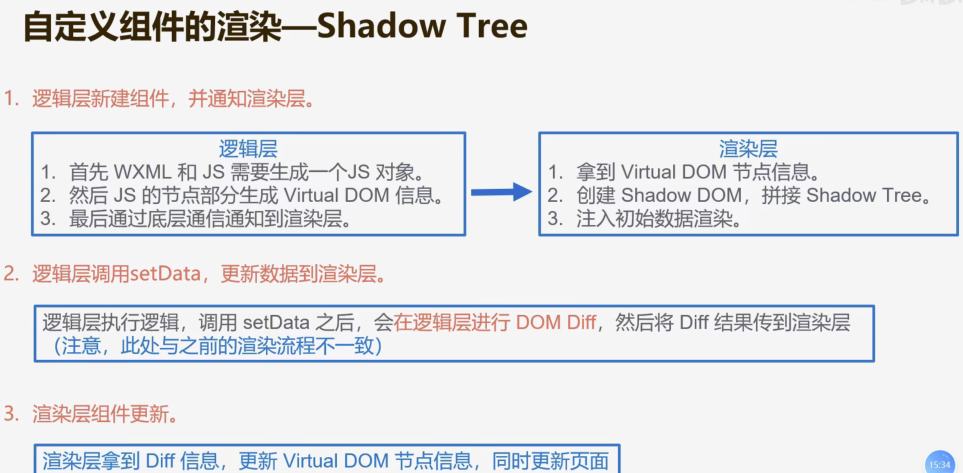




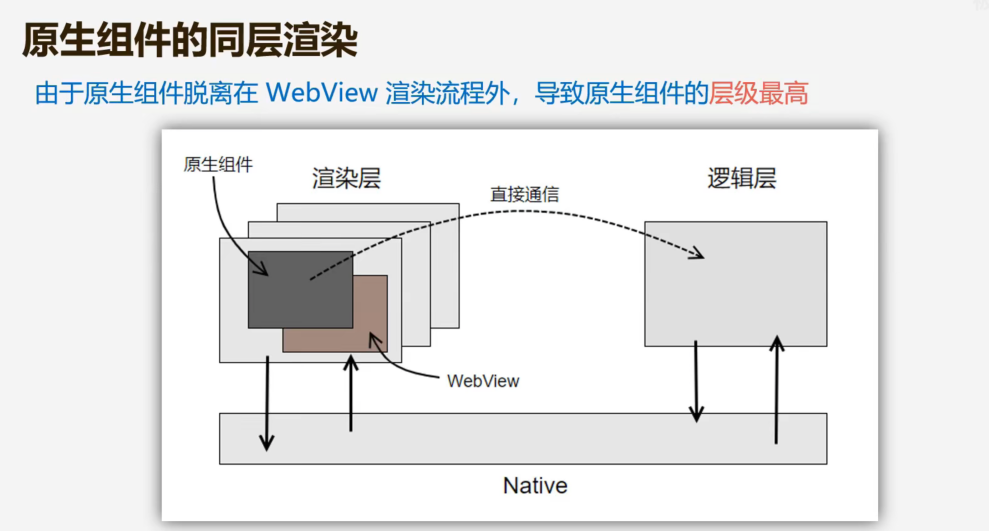






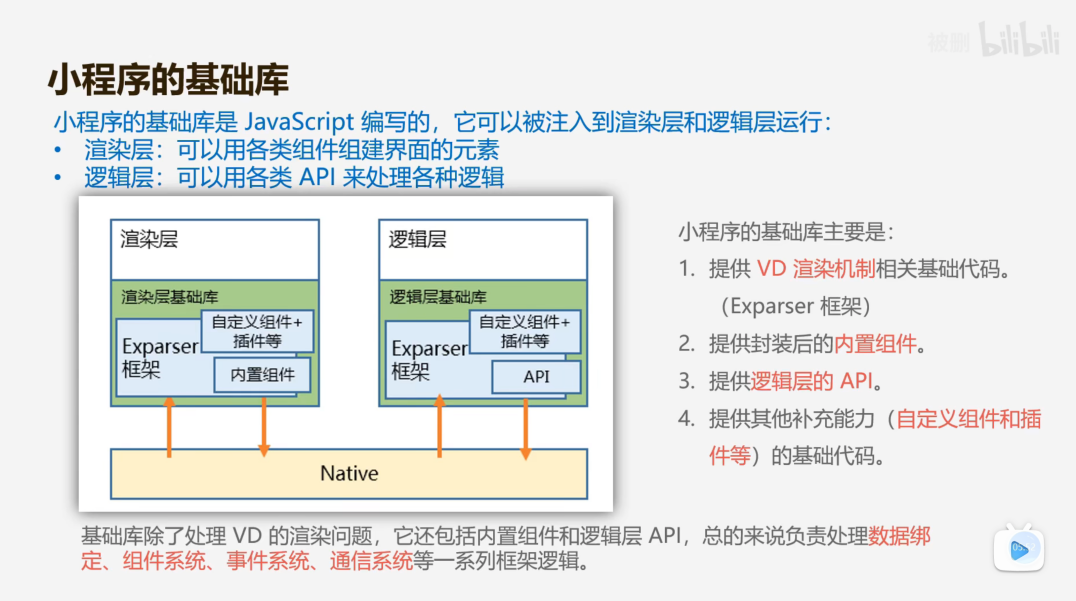


原生组件： 微信小程序提供的类似于调动手机摄像头之类的原生组件



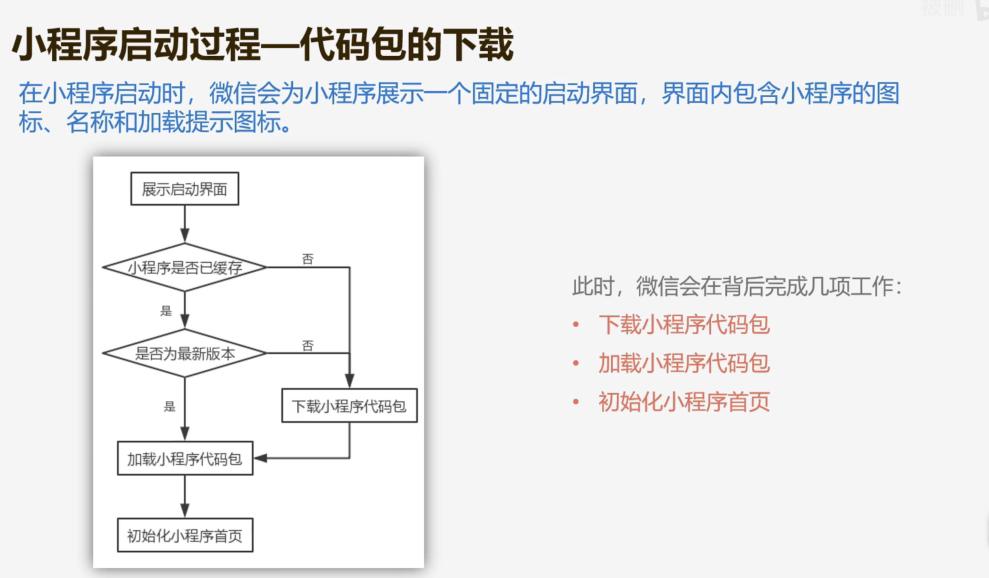


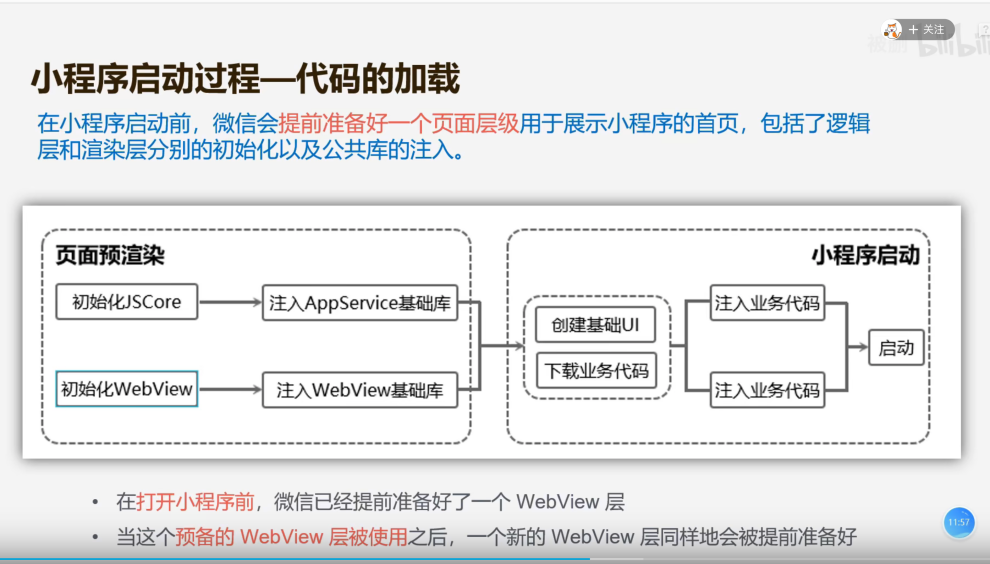




我们编写的代码是运行在小程序基础之上的

初始化一个小程序 === 用脚手架搭建了一个vue项目





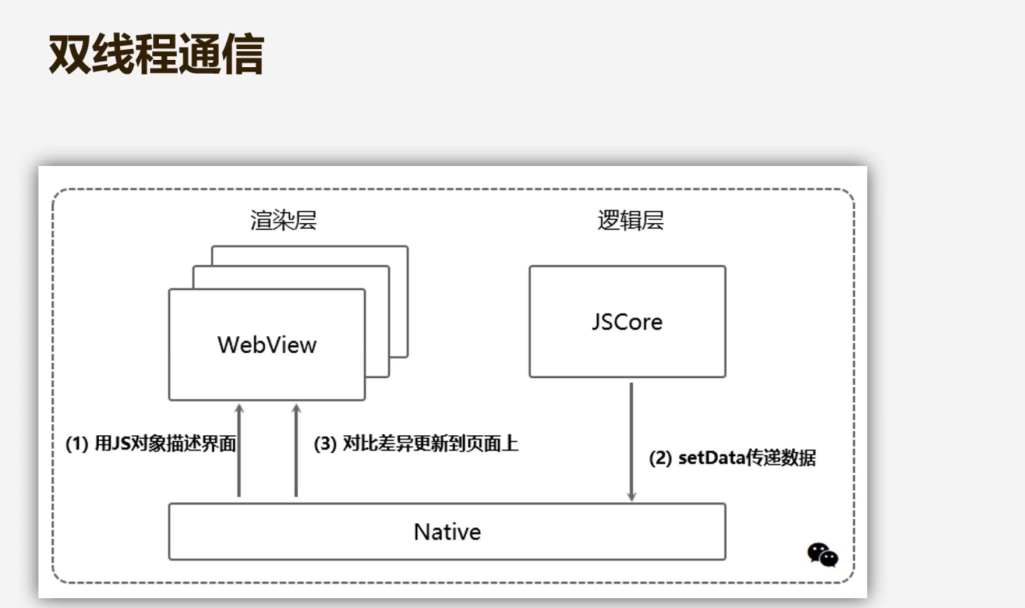


小程序的基础库可以升级 后台可以配置 让用户的小程序版本强制升级

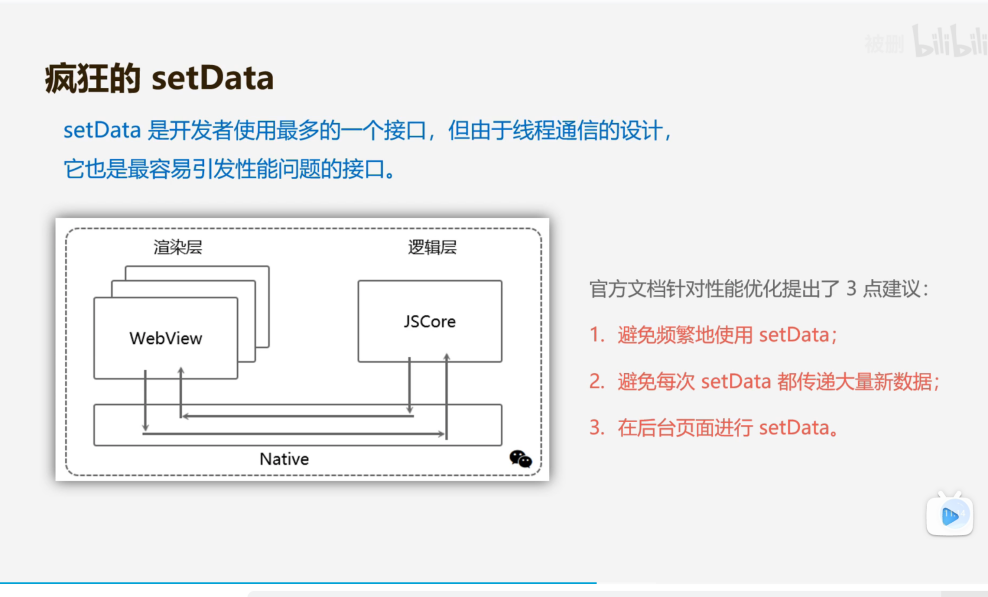


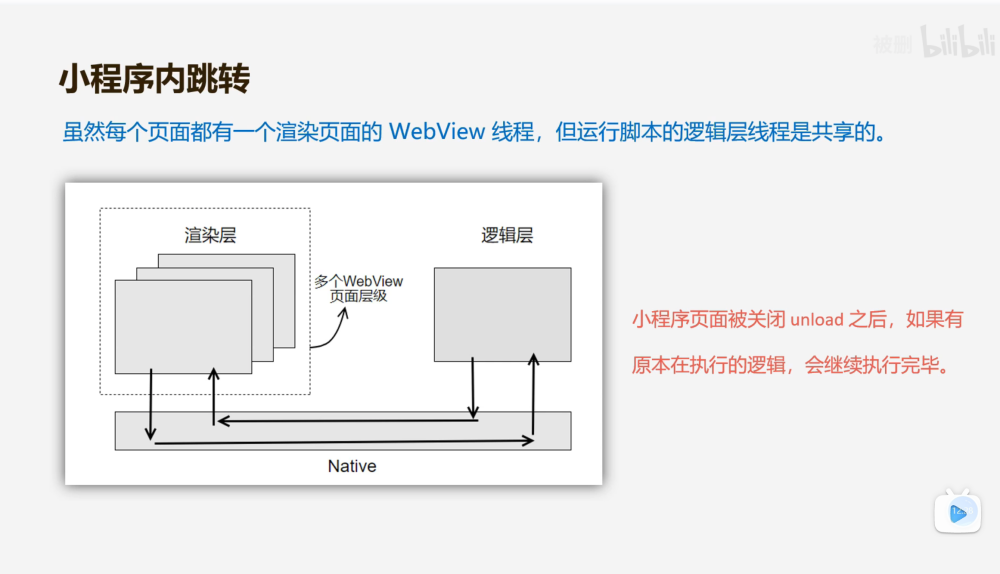
加入本次开发的小程序有BUG，需要马上更新，应该怎样操作

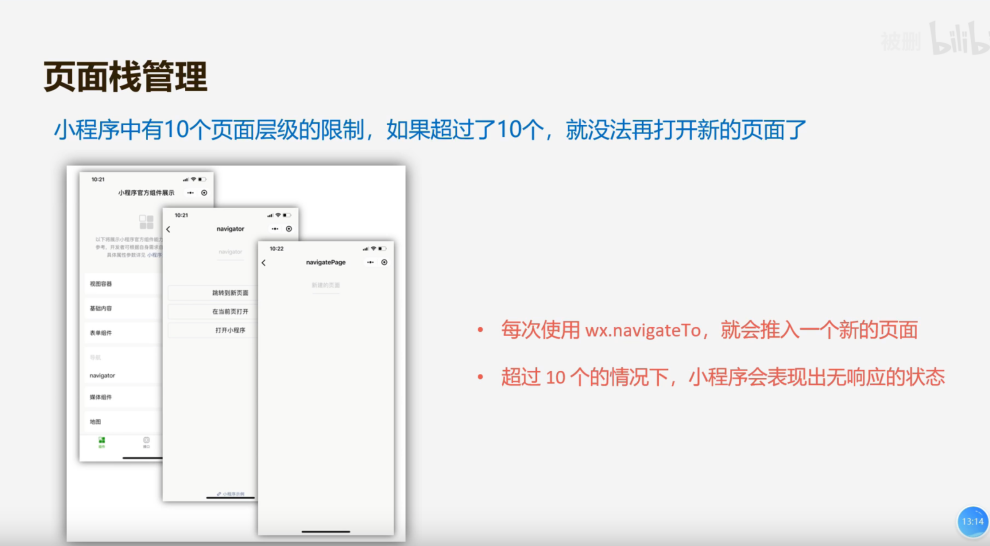




Kbone









小程序的WXS类似于VUE里面的filter

