小程序底层设计（原理篇）

@被删

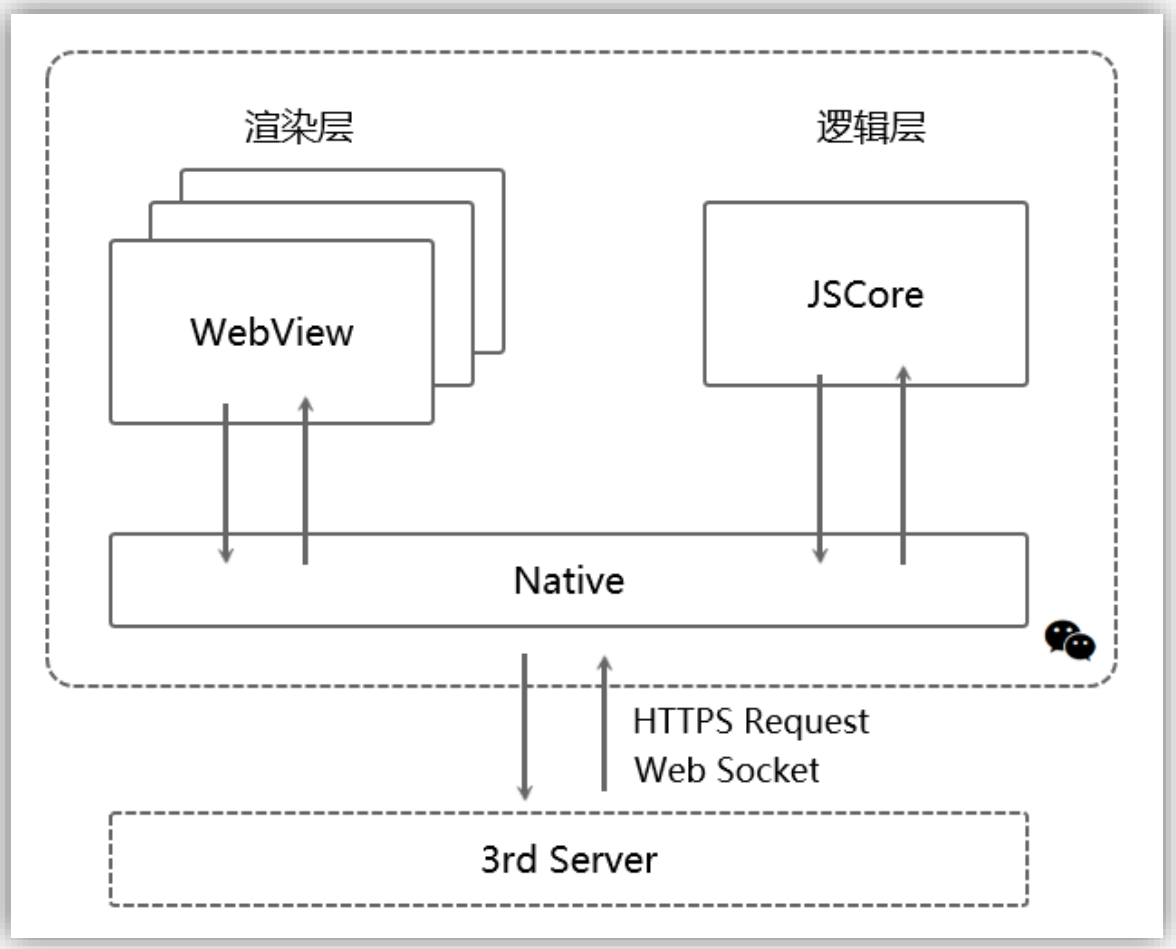
知道原理有什么好处呢?

1

双线程的小程序

小程序的双线程设计

小程序中分为渲染层（由 WebView 线程管理）和逻辑层（由客户端 JavaScript 解释引擎线程管理）

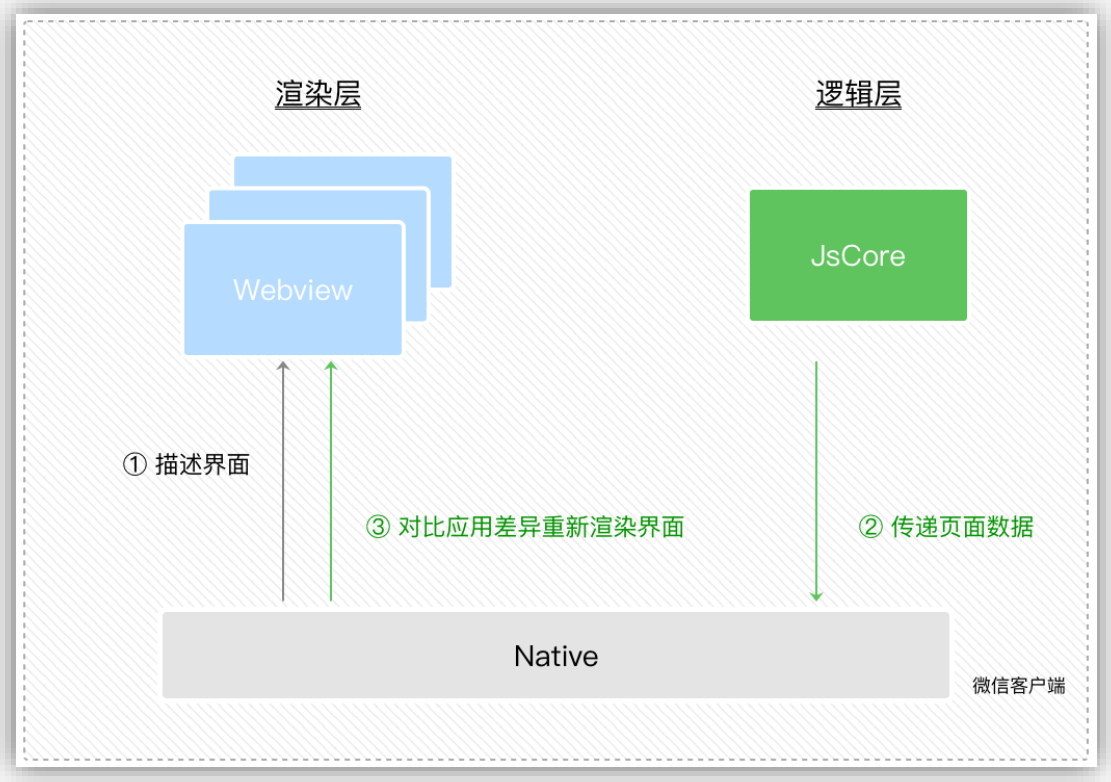


Web 技术是非常开放灵活的，开发者可以利用 JavaScript 脚本随意地操作 DOM，危害用户和网站的安全。

1. 可以防止恶意攻击者的 XSS 攻击；
2. 可以防止开发者恶意盗取用户敏感信息；
3. 提升页面加载性能。

双线程的通信设计

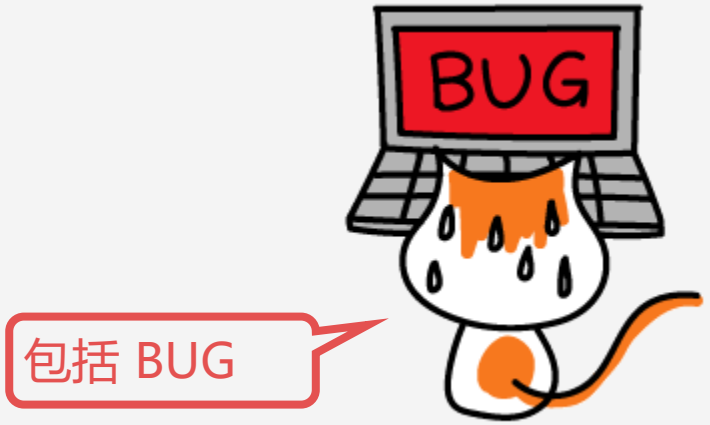
在小程序中，不管是逻辑层还是渲染层，都使用微信客户端 Native 进行直接通信，同时用于与其他模块或者外部进行中转通信。



1. 在渲染层里，把 WXML 转化成对应的用于描述虚拟 DOM 树的 JS 对象。
2. 在逻辑层发生数据变更的时候，通过宿主环境提供的 setData 方法把变更从逻辑层传递到 Native，再转发到渲染层。
3. 在渲染层里，经过对比前后差异，把差异应用在原来的 Virtual DOM 树上，更新界面。

1. 虚拟DOM设计
2. Shadow DOM 模型（ Shadow Root、Shadow Tree ）

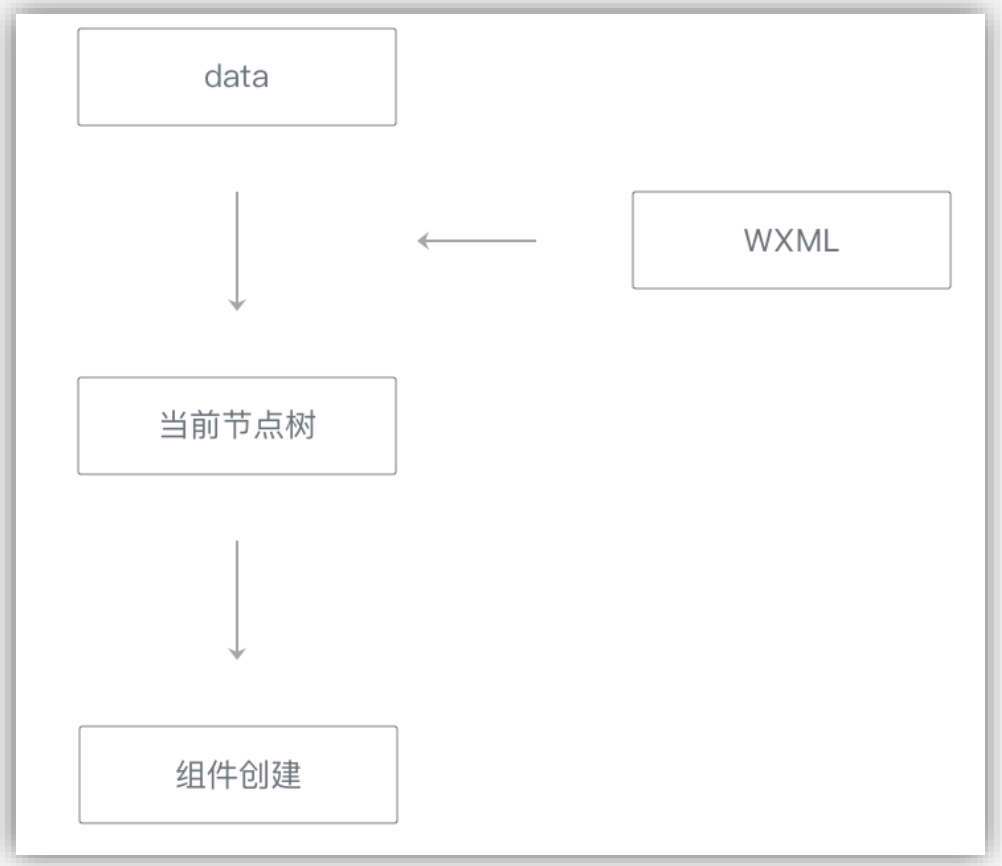
小程序里
一切都离不开双线程



2 小程序的渲染过程

渲染层渲染过程—初始渲染

渲染层在接收到初始数据（data）时，需要进行渲染层渲染。

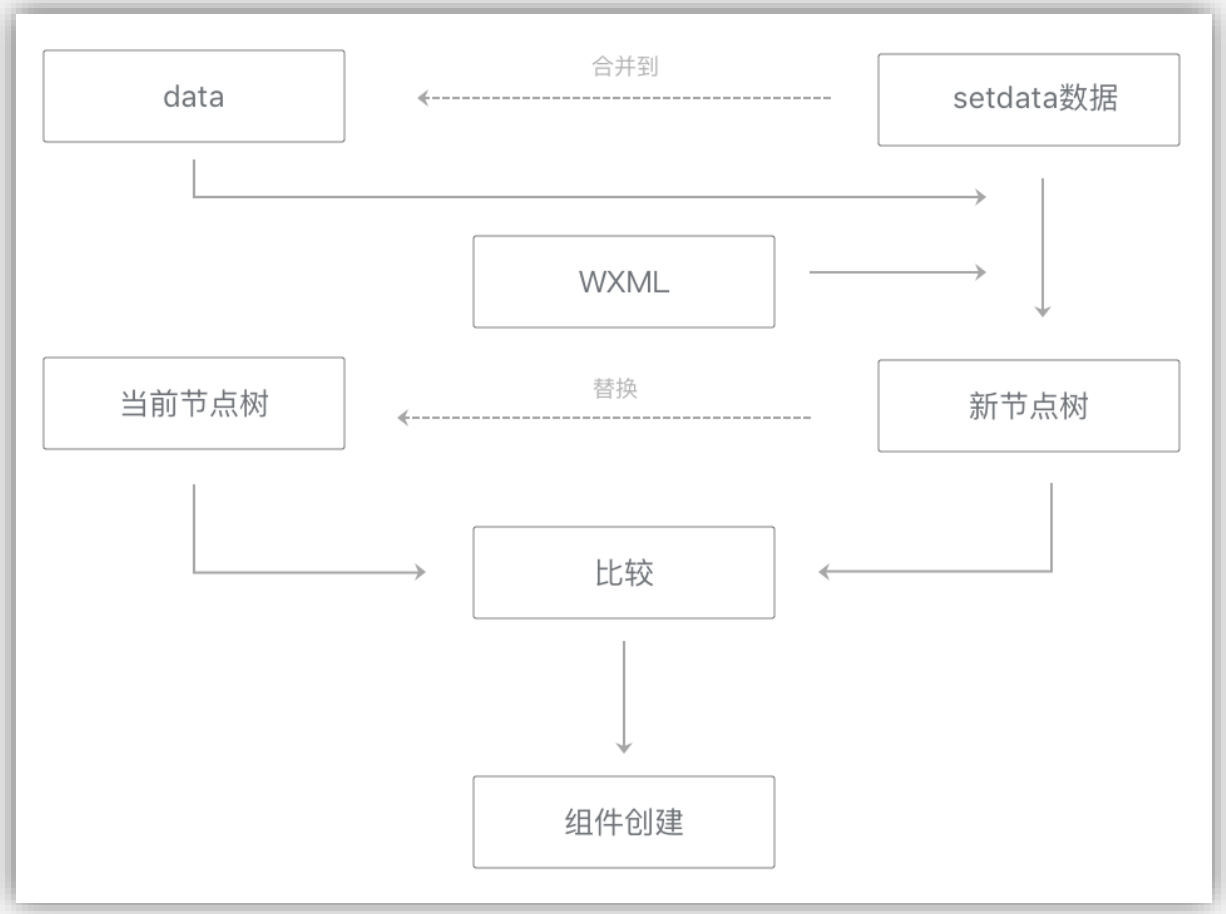


- 初始渲染时，将初始数据套用在对应的 WXML 片段上生成节点树
- 初始渲染中得到的 data 和当前节点树会保留下来用于重渲染

减少 WXML 中节点的数量，可以有效降低初始渲染和重渲染的时间开销，提升渲染性能

渲染层渲染过程—重渲染

渲染层在接收到更新数据（setData数据）时，需要进行渲染层渲染。



虚拟DOM设计

1. 每次重渲染时，将 data 和 setData 数据套用在 WXML 片段上，得到一个新节点树。

2. 将新节点树与当前节点树进行比较，这样可以得到哪些节点的哪些属性需要更新、哪些节点需要添加或移动。

3. 最后，将 setData 数据合并到 data 中，并用新节点树替换旧节点树，用于下一次重渲染。

去掉不必要设置的数据、减少 setData 的数据量也有助于提升这一个步骤的性能

自定义组件的渲染?

使用数据监听器

有时，在一些数据字段被 setData 设置时，需要执行一些操作

例如， `this.data.sum` 永远是 `this.data.numberA` 与 `this.`

下实现。

```
Component({
  attached: function() {
    this.setData({
      numberA: 1,
      numberB: 2,
    })
  },
  observers: {
    'numberA, numbe
      // 在 numberA
      this.setData(
        sum: number
      })
    }
  }
})
```

behaviors

`behaviors` 是用于组件

每个 `behavior` 可以包

组件中，生命周期函数

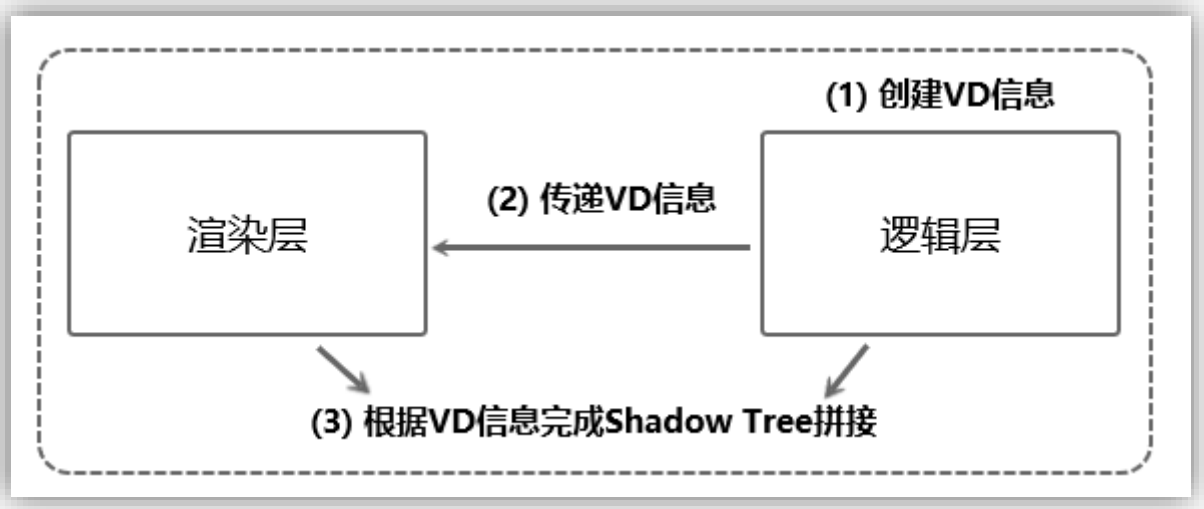
`behavior` 。

详细的参数含义和使用请参考 Behavior 参考文档。

生命周期	参数	描述	最低版本
created	无	在组件实例刚刚被创建时执行	1.6.3
attached	无	在组件实例进入页面节点树时执行	1.6.3
ready	无	在组件在视图层布局完成后执行	1.6.3
moved	无	在组件实例被移动到节点树另一个位置时执行	1.6.3
detached	无	在组件实例被从页面节点树移除时执行	1.6.3
error	`Object Error`	每当组件方法抛出错误时执行	2.4.1

自定义组件的渲染—逻辑层 or 渲染层?

创建方式	同步方式	优点	缺点
在渲染层创建	组件创建（或其他关键事件），通知逻辑层	减少通信数据大小	有很多双向通信和线程间等待
在逻辑层创建	传递创建后的 Shadow Tree	减少通信次数	Shadow Tree数据量很大



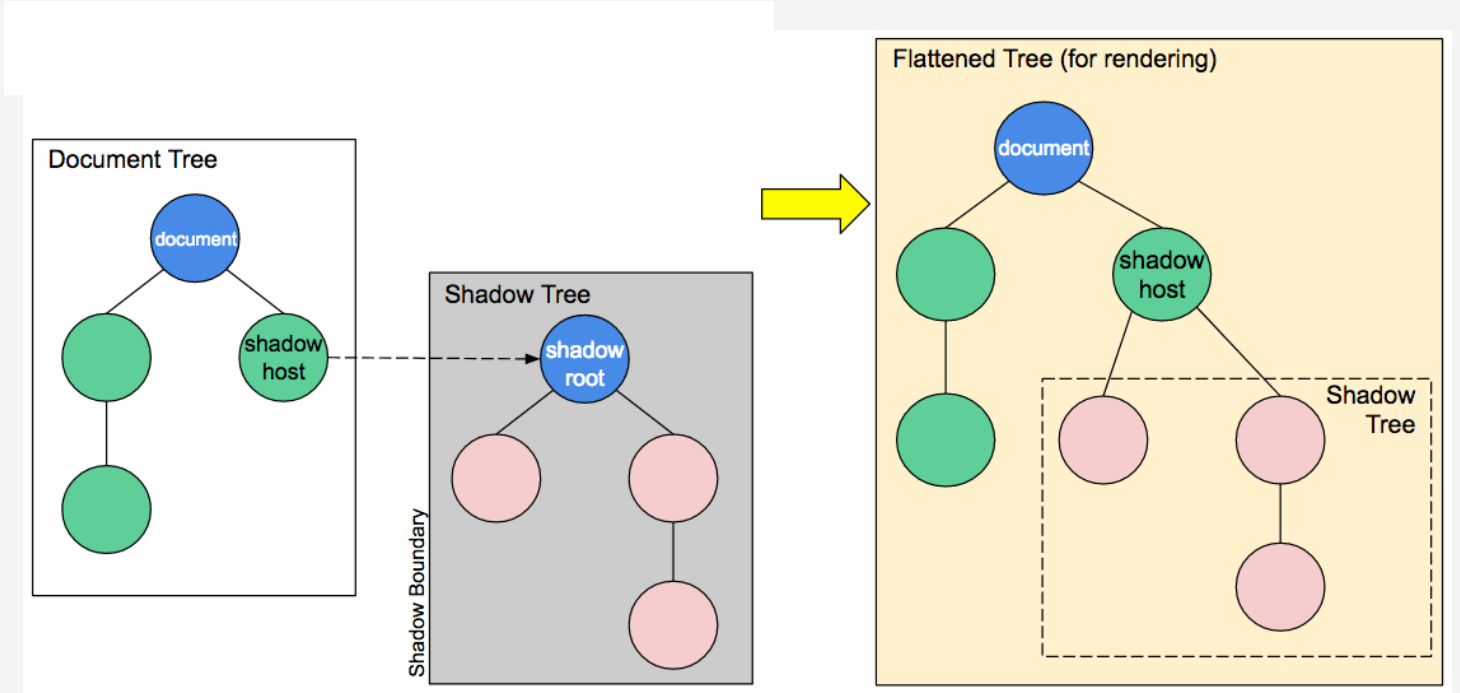
逻辑层和渲染层都维护 Virtual DOM

什么是 Shadow Tree

Web components 的一个重要属性是封装：可以将标记结构、样式和行为隐藏起来，并与页面上的其他代码相隔离，保证不同的部分不会混在一起，可使代码更加干净、整洁。

Web Components 中的三项技术：

- Custom elements（自定义元素）
- Shadow DOM（影子DOM）
- HTML templates（HTML模板）



其中，Shadow DOM 接口是关键所在，它可以将一个隐藏的、独立的 DOM 附加到一个元素上。

自定义组件的渲染—Shadow Tree

1. 逻辑层新建组件，并通知渲染层。

逻辑层
1. 首先 WXML 和 JS 需要生成一个JS 对象。
2. 然后 JS 的节点部分生成 Virtual DOM 信息。
3. 最后通过底层通信通知到渲染层。

渲染层
1. 拿到 Virtual DOM 节点信息。
2. 创建 Shadow DOM，拼接 Shadow Tree。
3. 注入初始数据渲染。

2. 逻辑层调用setData，更新数据到渲染层。

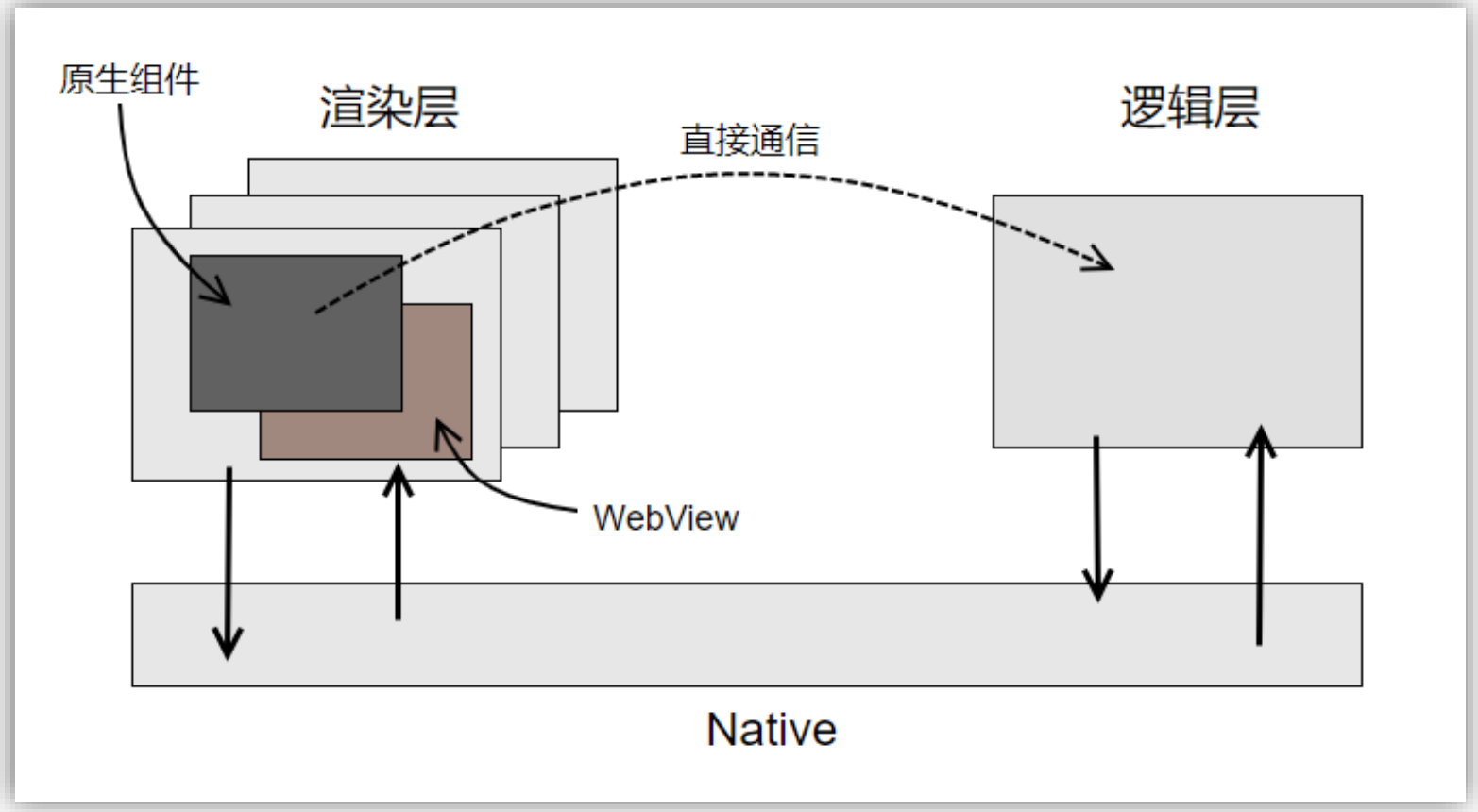
逻辑层执行逻辑，调用 setData 之后，会在逻辑层进行 DOM Diff，然后将 Diff 结果传到渲染层
（注意，此处与之前的渲染流程不一致）

3. 渲染层组件更新。

渲染层拿到 Diff 信息，更新 Virtual DOM 节点信息，同时更新页面

原生组件的同层渲染

由于原生组件脱离在 WebView 渲染流程外，导致原生组件的层级最高



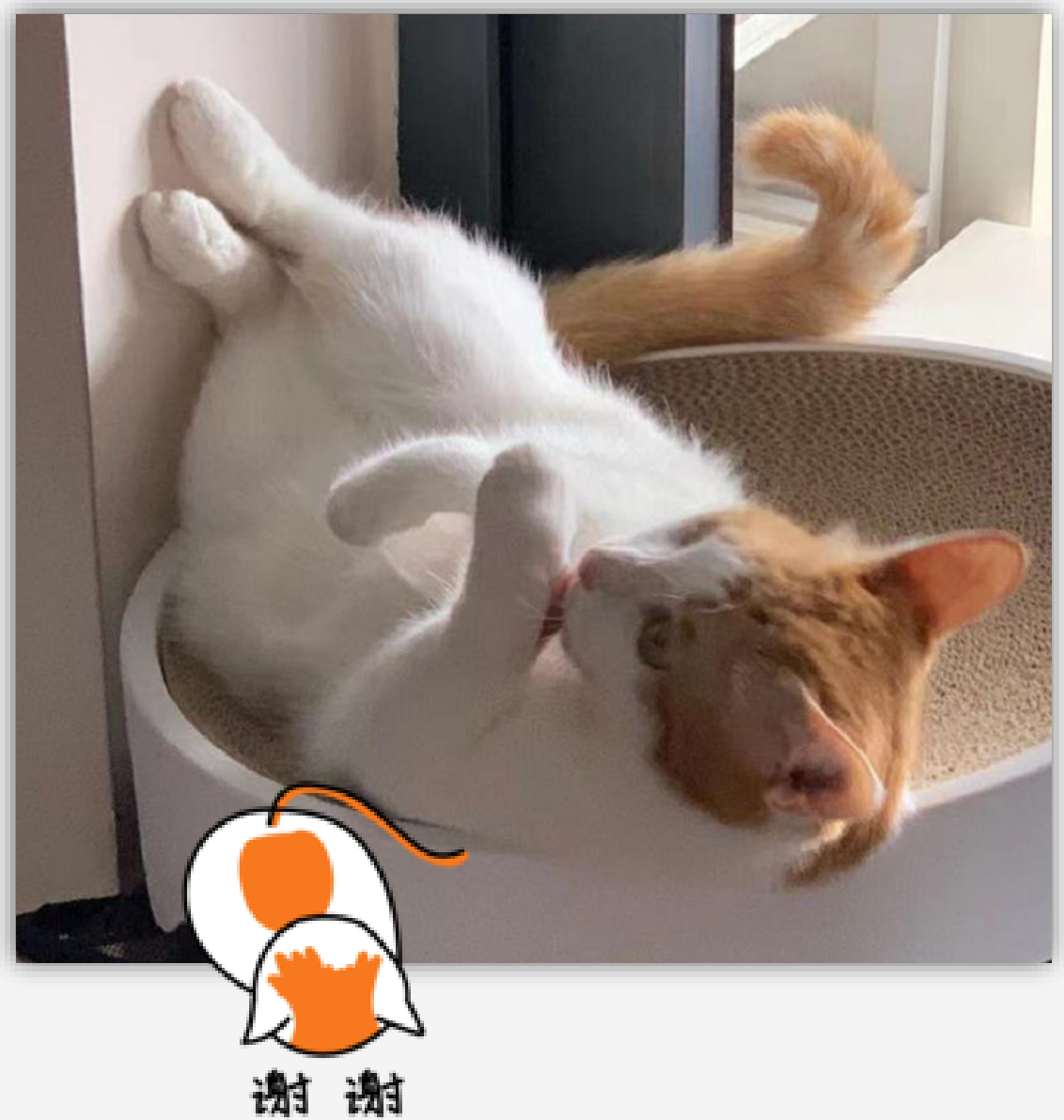
同层渲染：小程序通过某种 hack 方式将原生组件插入可控层级的方式，使得原生组件的层级和非原生组件一样可控，从而解决了遮挡的问题。

小程序的运行机制
会在下一次介绍哦

推广时间

本书包括三部分内容：

1. 小程序快速入门与实战
2. **小程序原理分析与避坑指南【我写哒】**
3. 云开发案例与项目实战



Github: godbasin
@被删