7.4 视图层渲染

视图层在接收到初始数据（data）和更新数据（setData 数据）时，需要进行视图层渲染。在一个页面的生命周期中，视图层会收到一份初始数据和多份更新数据。收到初始数据时需要执行初始渲染，每次收到更新数据时需要执行重渲染。

7.4.1 初始渲染

初始渲染发生在页面刚刚创建时。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。节点树也就是在开发者工具 WXML 面板中看到的页面树结构，它包含页面内所有组件节点的名称、属性值和事件回调函数等信息。最后根据节点树包含的各个节点，在界面上依次创建出各个组件。

图 7-7 视图层初始渲染流程图

在这整个流程中，时间开销大体上与节点树中节点的总量成正比例关系。因而减少 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。

代码清单 7-4 简化 WXML 代码的例子

```
<view data-my-data="{{myData}}"> <!-- 这个 view 和下一行的 view 可以合并 -->
  <view class="my-class" data-my-data="{{myData}}" bindtap="onTap">
    <text> <!-- 这个 text 通常是没必要的 -->
      {{myText}}
    </text>
  </view>
</view>

<!-- 可以简化为 -->

<view class="my-class" data-my-data="{{myData}}" bindtap="onTap">
  {{myText}}
```

```
</view>
```

7.4.2 重渲染

初始渲染完毕后，视图层可以多次应用 setData 的数据。每次应用 setData 数据时，都会执行重渲染来更新界面。

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。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。然后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除。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。

图 7-8 视图层重渲染流程图

在进行当前节点树与新节点树的比较时，会着重比较 setData 数据影响到的节点属性。因而，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。