7.3 数据通信

在每个小程序页面的生命周期中，存在着若干次页面数据通信。逻辑层向视图层发送页面数据（data 和 setData 的内容），视图层向逻辑层反馈用户事件。

7.3.1 页面初始数据通信

在小程序启动或一个新的页面被打开时，页面的初始数据（data）和路径等相关信息会从逻辑层发送给视图层，用于视图层的初始渲染。Native 层会将这些数据直接传递给视图层，同时向用户展示一个新的页面层级，视图层在这个页面层级上进行界面绘制。视图层接收到相关数据后，根据页面路径来选择合适的 WXML 结构，WXML 结构与初始数据相结合，得到页面的第一次渲染结果。

图 7-4 初始数据通信时序图

分析这个流程不难得知：页面初始化的时间大致由页面初始数据通信时间和初始渲染时间两部分构成。其中，数据通信的时间指数据从逻辑层开始组织数据到视图层完全接收完毕的时间，数据量小于 64KB 时总时长可以控制在 30ms 内。传输时间与数据量大体上呈现正相关关系，传输过大的数据将使这一时间显著增加。因而减少传输数据量是降低数据传输时间的有效方式。

图 7-5 数据传输时间与数据量关系图

7.3.2 更新数据通信

初始渲染完毕后，视图层可以在开发者调用 setData 后执行界面更新。在数据传输时，逻辑层会执行一次 JSON.stringify 来去除掉 setData 数据中不可传输的

部分，之后将数据发送给视图层。同时，逻辑层还会将 setData 所设置的数据字段与 data 合并，使开发者可以用 this.data 读取到变更后的数据。

因此，为了提升数据更新的性能，开发者在执行 setData 调用时，最好遵循以下原则：

1.不要过于频繁调用 setData,应考虑将多次 setData 合并成一次 setData 调用；

2.数据通信的性能与数据量正相关,因而如果有一些数据字段不在界面中展示且数据结构比较复杂或包含长字符串，则不应使用 setData 来设置这些数据；

3.与界面渲染无关的数据最好不要设置在 data 中,可以考虑设置在 page 对象的其他字段下。

代码清单 7-3 提升数据更新性能方式的代码示例

```
Page({
  onShow: function() {

    // 不要频繁调用 setData
    this.setData({ a: 1 })
    this.setData({ b: 2 })
    // 绝大多数时候可优化为
    this.setData({ a: 1, b: 2 })

    // 不要设置不在界面渲染时使用的数据，并将界面无关的数据放在 data 外
    this.setData({
      myData: {
        a: '这个字符串在 WXML 中用到了',
        b: '这个字符串未在 WXML 中用到，而且它很长.....'
      }
    })
    // 可以优化为
    this.setData({
      'myData.a': '这个字符串在 WXML 中用到了'
    })
    this._myData = {
      b: '这个字符串未在 WXML 中用到，而且它很长.....'
    }
```

```
  }
})
```

7.3.3 用户事件通信

视图层会接受用户事件，如点击事件、触摸事件等。用户事件的通信比较简单：当一个用户事件被触发且有相关的事件监听器需要被触发时，视图层会将信息反馈给逻辑层。如果一个事件没有绑定事件回调函数，则这个事件不会被反馈给逻辑层。视图层中有一套高效的事件处理体系，可以快速完成事件生成、冒泡、捕获等过程。

视图层将事件反馈给逻辑层时，同样需要一个通信过程，通信的方向是从视图层到逻辑层。因为这个通信过程是异步的，会产生一定的延迟，延迟时间同样与传输的数据量正相关，数据量小于 64KB 时在 30ms 内。降低延迟时间的方法主要有两个。

1.去掉不必要的事件绑定（WXML 中的 bind 和 catch），从而减少通信的数据量和次数；

2.事件绑定时需要传输 target 和 currentTarget 的 dataset，因而不要在节点的 data 前缀属性中放置过大的数据。

图 7-6 事件通信时间与数据量关系图