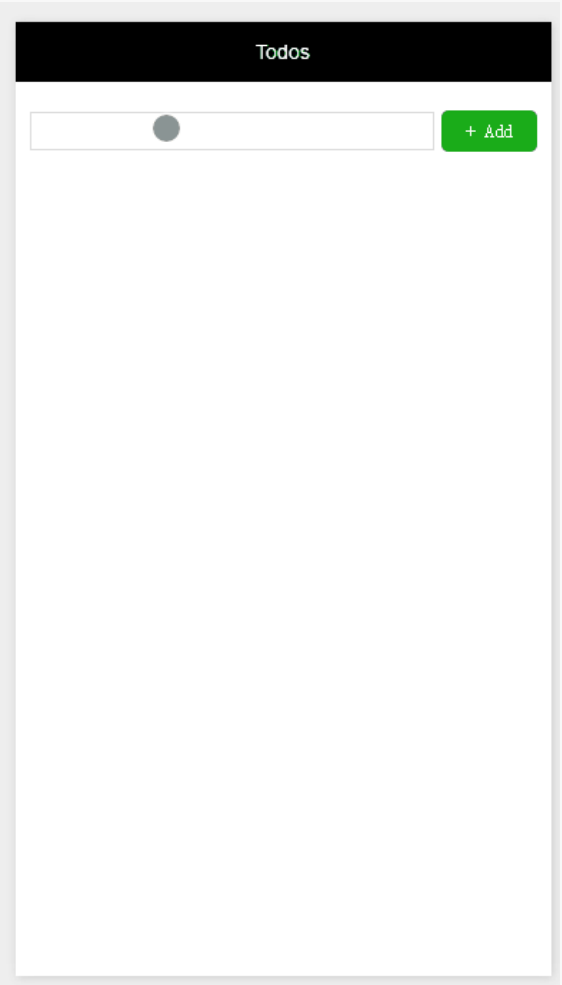
快速了解微信小程序的开发

摘要：微信官方已经开放微信小程序的官方文档和开发者工具。前两天都是在看相关的新闻来了解小程序该如何开发，这两天官方的文档出来之后，赶紧翻看了几眼，重点了解了一下文档中框架与组件这两个部分，然后根据简易教程，做了一个常规的 todo app。这个 app 基于微信小程序的平台，实现了 todo app 的常规功能，同时为了让它更接近实际的工作场景，也用到了 loading 与 toast 这两个组件来完成一些操作的交互与反馈。这个平台给我的直观感受是，技术层面，它跟 vue 有相似性，但是远没有 vue 强大；开发时候的思路，不像 vue，反倒觉得比较像 backbone。所以要是使用过 backbone,vue 等 mvc,mvvm 框

先补充下本文相关的资料：

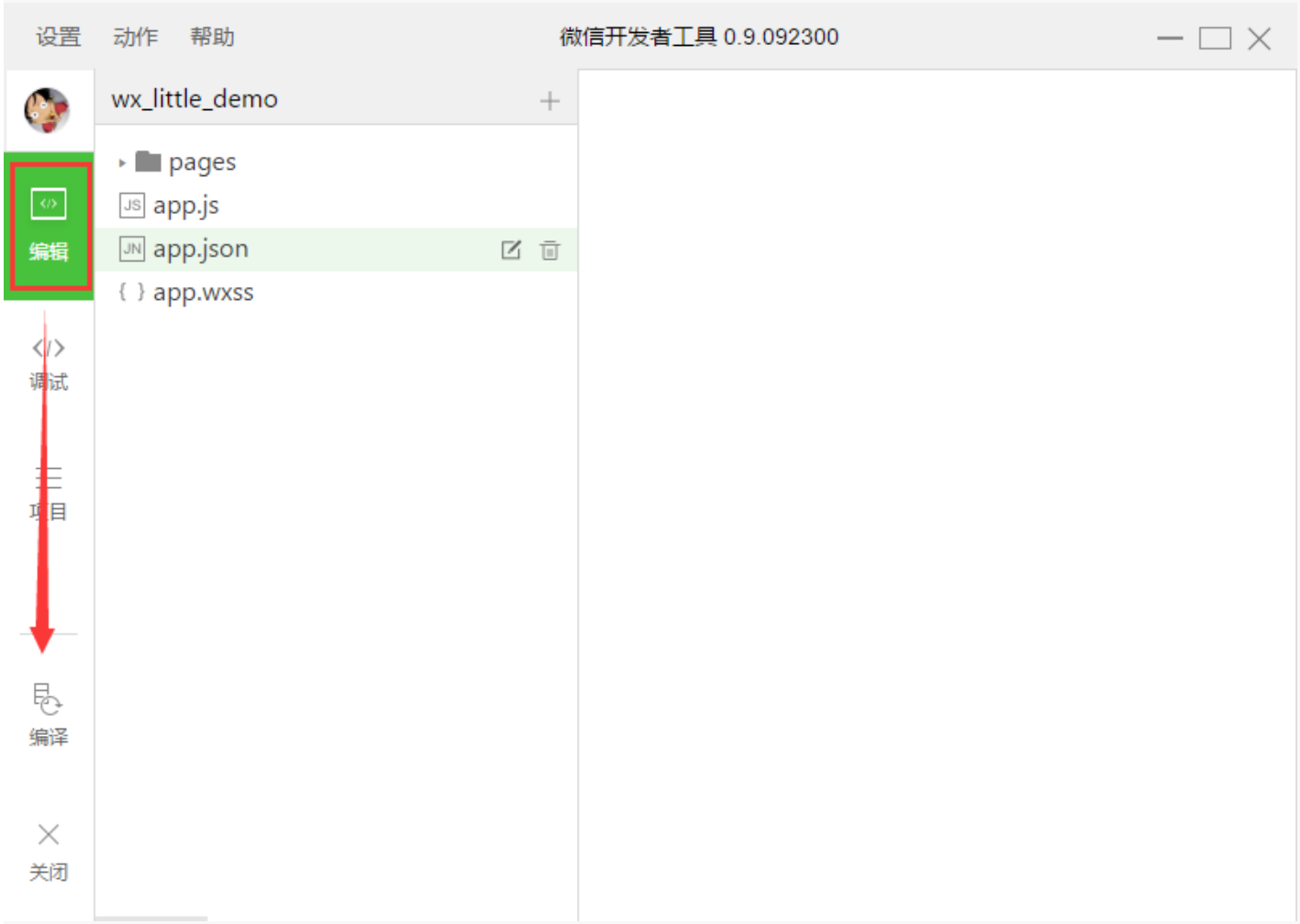
官方文档：https://mp.weixin.qq.com/debug/wxadoc/dev/index.html
官方开发者工具下载：https://mp.weixin.qq.com/debug/wxadoc/dev/devtools/download.html
本文 todo app 的功能演示：



注：需长按 todo 的 text，才能直接编辑。因为是在手机端，所以不能使用双击事件来进行编辑，改成了长按事件。小程序的平台也没有提供双击事件的绑定。

相关源码：https://github.com/liuyunzhuge/blog/tree/master/todos/wx
如果你想在本地运行这个项目，需要先安装开发者工具，按照文档中简易教程的描述，先建好一个项目；
建完之后，开发者工具就会打开这个项目；
接着在磁盘上，找到建好的项目的文件夹，把里面的内容都删掉，把上面源码文件夹下的文件都粘贴进去；

然后重新打开开发者工具，先进入到编辑页签，然后点击编译按钮，就会直接进入到调试界面，查看 app 的功能：



下面来介绍下这个 app 开发的要点：

1. 这个 app 的目录结构以及配置等就不详细介绍了，这些在文档-框架部分都有很详细的描述。这个平台里面没有 html 和 css，取而代之的是 wxml 和 wxss。wxss 跟 css 几乎没有区别，缺点就是不如 css 强大，

支持的选择器有限。但是好处是由于只有微信这一个平台，所以几乎没有兼容性问题，能够使用标准的，更新的 css 技术。wxml 里面只能用平台提供的那些组件的标签，html 的标签不能直接用，各个组件的在 wxml 的使用方式，都可以在文档-组件这一部分找到说明的示例。所以实际上 wxml 跟 wxss 编写起来都没有什么难题。

2. wxml 支持以下这些特性：

在 todo app 里面除了模板和引用没有用到之外，其它的都使用到了，不过没有使用到每个特性的各个细节，只根据 app 的需要选用合适的功能。前几天看到有文章说，微信小程序可能是基于 vue 框架来实现的，所以就看了下 vue 的文档。对于数据绑定，条件渲染，列表渲染，事件这几部分都详细看了 vue 的用法。对比下来，wxml 提供的这些特性，跟 vue 的相关特性是还比较像，不过功能并没有那么多，所以也不能轻易地直接拿 vue 框架的特性用到小程序里面。最佳实践，还是基于官方文档中提供的说明来，如果官方文档中没有提到的功能，通过猜测的方式去用，肯定是行不通的。我通过打印的方式，查看一些对象的原型，也并没有发现比官方文档要多的一些实例方法，说明小程序的框架功能确实是有限的。

3. wxss 其实是可以用 less 或者 sass 来写的，只要选择器满足框架的要求即可。由于时间原因，就没有在这个 app 里面去尝试了。

4. 没有双向绑定。在 vue 里面，一个 vue 实例就是一个 view-model；view 层对数据的更新，会实时反馈到 model；model 的更新，也会实时反馈的到 view。在小程序里面，没有双向绑定，view 的更新不会直接同步到 model；需要在相关事件回调里面，直接从 view 层拿到数据，然后通过 setData 的方式，更新 model，小程序内部会在 setData 之后重新渲染 page。比如单个 todo 项，toggle 的操作：

```
toggleTodo: function( e ) {

    var id = this.getTodoId( e, 'todo-item-chk-' );
    var value = e.detail.value[ 0 ];
    var complete = !!value;
    var todo = this.getTodo( id );

    todo.complete = complete;
    this.updateData( true );
    this.updateStorage();
},
```

以上代码中，通过 e.detail.value[0]拿到单个 todo 项里面 checkbox 的值，通过该值来判断 todo 的 complete 状态。最后在 updateData 的内部，还会通过 setData 方法，刷新 model 的内容。只有这样，在 toggle 操作之后，app 底部的统计信息才会更新。

5. 事件绑定的时候，无法传递参数，只能传递一个 event。比如上面那个 toggle 的操作，我其实很想在回调里面把当前 todo 的 id 传到这个回调里面，但是想尽办法都做不到，最后只能通过 id 的方式来处理：就是在 wxml 中绑定事件的组件上面，加一个 id，这个 id 全 page 也不能重复，所以 id 得加前缀，然后在 id 最后加上 todo 的 id 值；当事件触发的时候，通过 e.currentTarget.id 就能拿到该组件的 id，去掉相应的 id 前缀，就得到 todo 的 id 值了。这是目前用到的一个方法，我认为不是很优雅，希望后面能发现更好的办法来实现。

```
<checkbox-group id="todo-item-chk-{{todo.id}}" bindchange="toggleTodo">
    <label class="checkbox">
        <checkbox value="1" checked="{{todo.complete}}"/>
    </label>
</checkbox-group>
```

6. app 中考虑到了 loading 的效果，要利用 button 组件的 loading 属性来实现。但是 loading 仅仅是一个样式的控制，它不会控制这个按钮是否能重复点击。所以还要利用 buttong 的 disabled 属性，防止重复点击。

剩下的实现细节，都在下面两个文件的源码中，欢迎大家指出其中的问题。

index.wxml 的源码：

```
<!--list.wxml-->
<view class="container">
    <view class="app-hd">
        <view class="fx1">
            <input class="new-todo-input" value="{{newTodoText}}" auto-focus
bindinput="newTodoTextInput"/>
        </view>
        <button type="primary" size="mini" bindtap="addOne"
loading="{{addOneLoading}}" disabled="{{addOneLoading}}">
        + Add
        </button>
    </view>
    <view class="todos-list" >
        <view class="todo-item {{index == 0 ? '' : 'todo-item-not-first'}}
{{todo.complete ? 'todo-item-complete' : ''}}" wx:for="{{todos}}" wx:for-item="todo">
            <view wx-if="{{!todo.editing}}">
                <checkbox-group id="todo-item-chk-{{todo.id}}"
```

```
bindchange="toggleTodo">
                    <label class="checkbox">
                        <checkbox value="1" checked="{{todo.complete}}"/>
                    </label>
                </checkbox-group>
            </view>
            <view id="todo-item-txt-{{todo.id}}" class="todo-text"
wx-if="{{!todo.editing}}" bindlongtap="startEdit">
                <text>{{todo.text}}</text>
            </view>
            <view wx-if="{{!todo.editing}}">
                <button id="btn-del-item-{{todo.id}}" bindtap="clearSingle"
type="warn" size="mini" loading="{{todo.loading}}"  disabled="{{todo.loading}}">
                    Clear
                </button>
            </view>
            <input id="todo-item-edit-{{todo.id}}" class="todo-text-input"
value="{{todo.text}}" auto-focus bindblur="endEditTodo" wx-if="{{todo.editing}}"/>
        </view>
    </view>
    <view class="app-ft" wx:if="{{todos.length > 0}}">
        <view class="fx1">
            <checkbox-group bindchange="toggleAll">
                <label class="checkbox">
                    <checkbox value="1" checked="{{todosOfUncomplted.length == 0}}"/>
                </label>
            </checkbox-group>
            <text>{{todosOfUncomplted.length}} left.</text>
        </view>
        <view wx:if="{{todosOfComplted.length > 0}}">
            <button type="warn" size="mini" bindtap="clearAll"
loading="{{clearAllLoading}}" disabled="{{clearAllLoading}}">
                Clear {{todosOfComplted.length}} of done.
```

```
            </button>
        </view>
    </view>
    <loading hidden="{{loadingHidden}}" bindchange="loadingChange">
        {{loadingText}}
    </loading>
    <toast hidden="{{toastHidden}}" bindchange="toastChange">
        {{toastText}}
    </toast>
</view>
```

index.js 的源码：

```
var app = getApp();

Page( {
    data: {
        todos: [],
        todosOfUncomplted: [],
        todosOfComplted: [],
        newTodoText: '',
        addOneLoading: false,
        loadingHidden: true,
        loadingText: '',
        toastHidden: true,
        toastText: '',
        clearAllLoading: false
    },
    updateData: function( resetTodos ) {
        var data = {};
        if( resetTodos ) {
            data.todos = this.data.todos;
        }
```

```
        data.todosOfUncomplted = this.data.todos.filter( function( t ) {
            return !t.complete;
        });

        data.todosOfComplted = this.data.todos.filter( function( t ) {
            return t.complete;
        });

        this.setData( data );
    },
    updateStorage: function() {
        var storage = [];
        this.data.todos.forEach( function( t ) {
            storage.push( {
                id: t.id,
                text: t.text,
                complete: t.complete
            })
        });

        wx.setStorageSync( 'todos', storage );
    },
    onLoad: function() {
        this.setData( {
            todos: wx.getStorageSync( 'todos' ) || []
        });
        this.updateData( false );
    },
    getTodo: function( id ) {
        return this.data.todos.filter( function( t ) {
            return id == t.id;
        })[ 0 ];
    },
```

```
    getTodoId: function( e, prefix ) {
        return e.currentTarget.id.substring( prefix.length );
    },
    toggleTodo: function( e ) {

        var id = this.getTodoId( e, 'todo-item-chk-' );
        var value = e.detail.value[ 0 ];
        var complete = !!value;
        var todo = this.getTodo( id );

        todo.complete = complete;
        this.updateData( true );
        this.updateStorage();
    },
    toggleAll: function( e ) {
        var value = e.detail.value[ 0 ];
        var complete = !!value;

        this.data.todos.forEach( function( t ) {
            t.complete = complete;
        });

        this.updateData( true );
        this.updateStorage();

    },
    clearTodo: function( id ) {
        var targetIndex;
        this.data.todos.forEach( function( t, i ) {
            if( targetIndex !== undefined ) return;

            if( t.id == id ) {
                targetIndex = i;
```

```
            }
        });

        this.data.todos.splice( targetIndex, 1 );
    },
    clearSingle: function( e ) {
        var id = this.getTodoId( e, 'btn-del-item-' );
        var todo = this.getTodo( id );

        todo.loading = true;
        this.updateData( true );

        var that = this;
        setTimeout( function() {
            that.clearTodo( id );
            that.updateData( true );
            that.updateStorage();
        }, 500 );
    },
    clearAll: function() {
        this.setData( {
            clearAllLoading: true
        });

        var that = this;
        setTimeout( function() {
            that.data.todosOfComplted.forEach( function( t ) {
                that.clearTodo( t.id );
            });
            that.setData( {
                clearAllLoading: false
            });
            that.updateData( true );
```

```
            that.updateStorage();

            that.setData( {
                toastHidden: false,
                toastText: 'Success'
            });
        }, 500 );

    },
    startEdit: function( e ) {
        var id = this.getTodoId( e, 'todo-item-txt-' );
        var todo = this.getTodo( id );
        todo.editing = true;

        this.updateData( true );
        this.updateStorage();
    },
    newTodoTextInput: function( e ) {
        this.setData( {
            newTodoText: e.detail.value
        });
    },
    endEditTodo: function( e ) {
        var id = this.getTodoId( e, 'todo-item-edit-' );
        var todo = this.getTodo( id );

        todo.editing = false;
        todo.text = e.detail.value;

        this.updateData( true );
        this.updateStorage();
    },
    addOne: function( e ) {
```

```
    if( !this.data.newTodoText ) return;

    this.setData( {
        addOneLoading: true
    });

    //open loading
    this.setData( {
        loadingHidden: false,
        loadingText: 'Waiting...'
    });

    var that = this;
    setTimeout( function() {
        //close loading and toggle button loading status
        that.setData( {
            loadingHidden: true,
            addOneLoading: false,
            loadingText: ''
        });

        that.data.todos.push( {
            id: app.getId(),
            text: that.data.newTodoText,
            compelte: false
        });

        that.setData( {
            newTodoText: ''
        });

        that.updateData( true );
        that.updateStorage();
```

```
        }, 500 );
    },
    loadingChange: function() {
        this.setData( {
            loadingHidden: true,
            loadingText: ''
        });
    },
    toastChange: function() {
        this.setData( {
            toastHidden: true,
            toastText: ''
        });
    }
});
```

最后需要补充的是，这个 app 在有限的时间内依据微信的官方文档进行开发，所以这里面的实现方式到底是不是合理的，我也不清楚。我也仅仅是通过这个 app 来了解小程序这个平台的用法。希望微信官方能够推出一些更全面、最好是项目性的 demo，在代码层面，给我们这些开发者提供一个最佳实践规范。欢迎有其它的开发思路的朋友，帮我指出我以上实现中的问题。