Acmen、L 学习案例集锦《一》生命周期，组件

一：生命周期

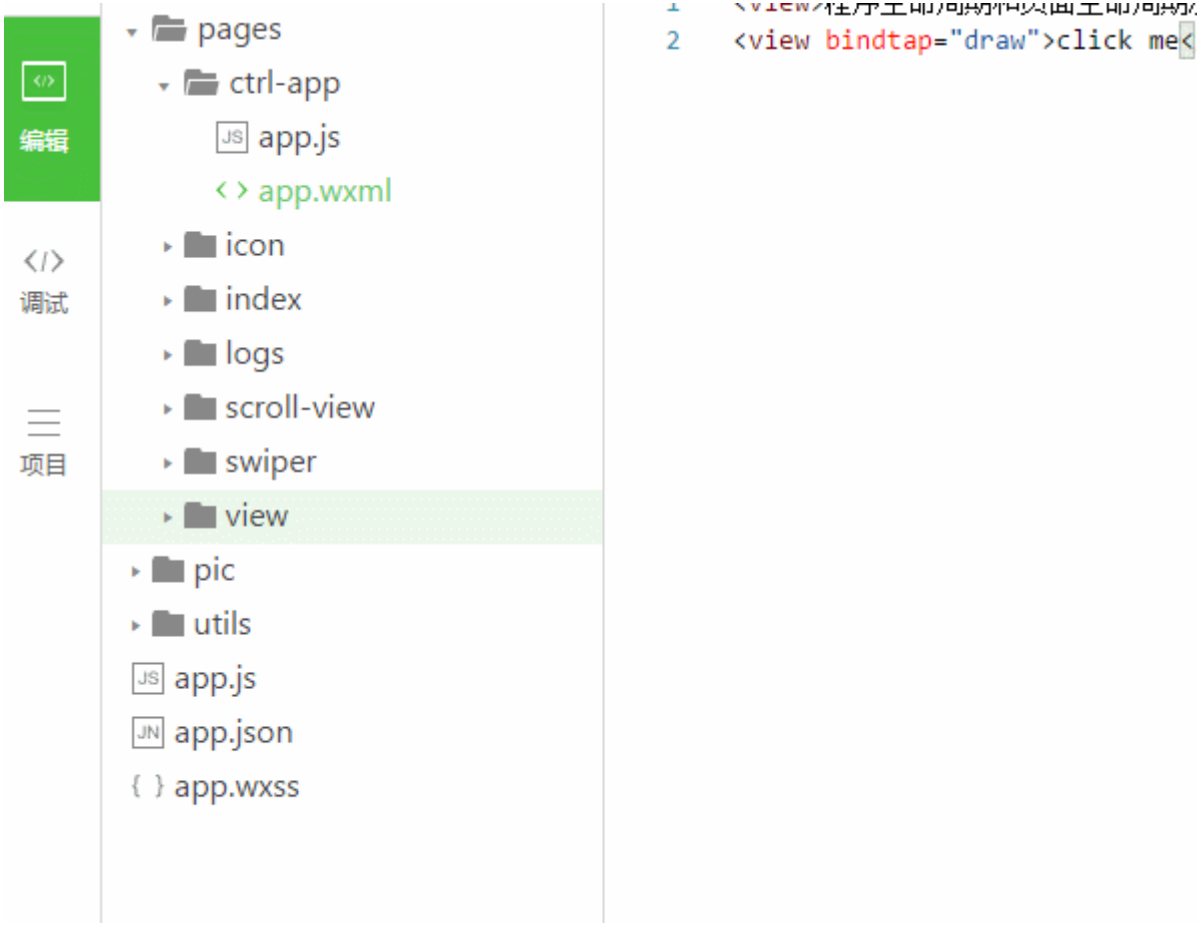
在 app.js 的 app()中注册程序

onLaunch	Function	生命周期函数--监听小程序初始化	当小程序初始化完成时，会触发onLaunch（全局只触发一次）
onShow	Function	生命周期函数--监听小程序显示	当小程序启动，或从后台进入前台显示，会触发onShow
onHide	Function	生命周期函数--监听小程序隐藏	当小程序从前台进入后台，会触发onHide
其他	Any	开发者可以添加任意的函数或数据到Object参数中，用 `this` 可以访问	

在页面.js 中的 Page({})中注册页面。

data	Object	页面的初始数据
onLoad	Function	生命周期函数--监听页面加载
onReady	Function	生命周期函数--监听页面渲染完成
onShow	Function	生命周期函数--监听页面显示
onHide	Function	生命周期函数--监听页面隐藏
onUnload	Function	生命周期函数--监听页面卸载
其他	Any	开发者可以添加任意的函数或数据到Object参数中，用 `this` 可以访问

执行效果：



二：组件

1.view

把文档分割为独立的、不同的部分。

view 组件类似于 html 中的 div 标签，默认为块级元素，独占一行，可以通过设置 display:inline-block 改为行级元素。

- view.wxml 代码如下：

```
<view class="outerView">
    <view class="innerView1"></view>
    <view class="innerView2"></view>
    <view class="innerView3"></view>
<view>
```

- view.wxss 代码如下

```
.outerView{
    width:100%;height: 100px;margin: 0 auto;background-color: brown;
}
.innerView1{
    width: 40%;height: 40px;background: blue;display: inline-block;
}
.innerView2{
```

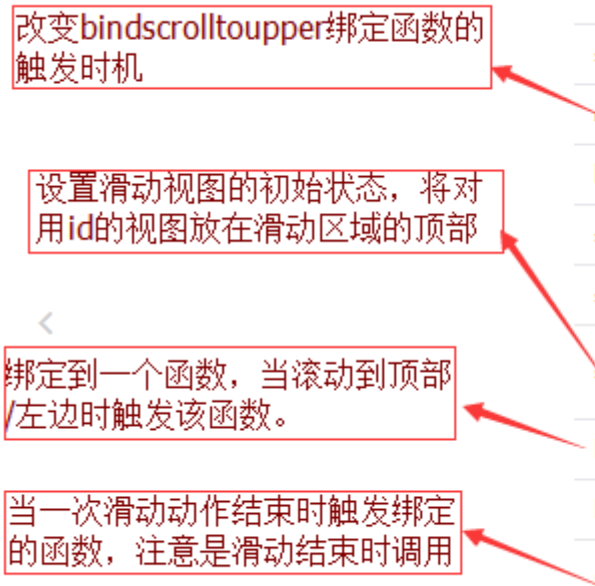
```
    width: 40%;height: 40px;background: yellow;display: inline-block;
}
.innerView3{
    width: 40%;height: 40px;background: peru;
}
```

- 显示效果：

2.scroll-view

可滚动（点击滑动）视图区域。

- 官方给出的属性列：



scroll-x	Boolean	false	允许横向滚动
scroll-y	Boolean	false	允许纵向滚动
upper-threshold	Number	50	距顶部/左边多远时（单位px），触发scrolltoupper事件
lower-threshold	Number	50	距底部/右边多远时（单位px），触发scrolltolower事件
scroll-top	Number		设置竖向滚动条位置
scroll-left	Number		设置横向滚动条位置
scroll-into-view	String		值应为某子元素id，则滚动到该元素，元素顶部对齐滚动区域顶部
bindscrolltoupper	EventHandle		滚动到顶部/左边，会触发scrolltoupper事件
bindscrolltolower	EventHandle		滚动到底部/右边，会触发scrolltolower事件
bindscroll	EventHandle		滚动时触发，event.detail总携带{scrollLeft,scrollTop,scrollHeight,scrollWidth,deltaX,deltaY}

- scroll-view.wxml 代码如下：

```
<view class="section">
    <scroll-view scroll-y="true" style="height: 200px;" bindscrolltoupper="upper"
     bindscrolltolower="lower" bindscroll="scroll" lower-threshold="50"    scroll-into-view="{{toView}}"
      scroll-top="1000px">
        <view id="green" class="scroll-view-item bc_green" style="width:100px;height:100px;background:red"></view>
        <view id="red"    class="scroll-view-item bc_red" style="width:100px;height:100px;background:blue"></view>
```

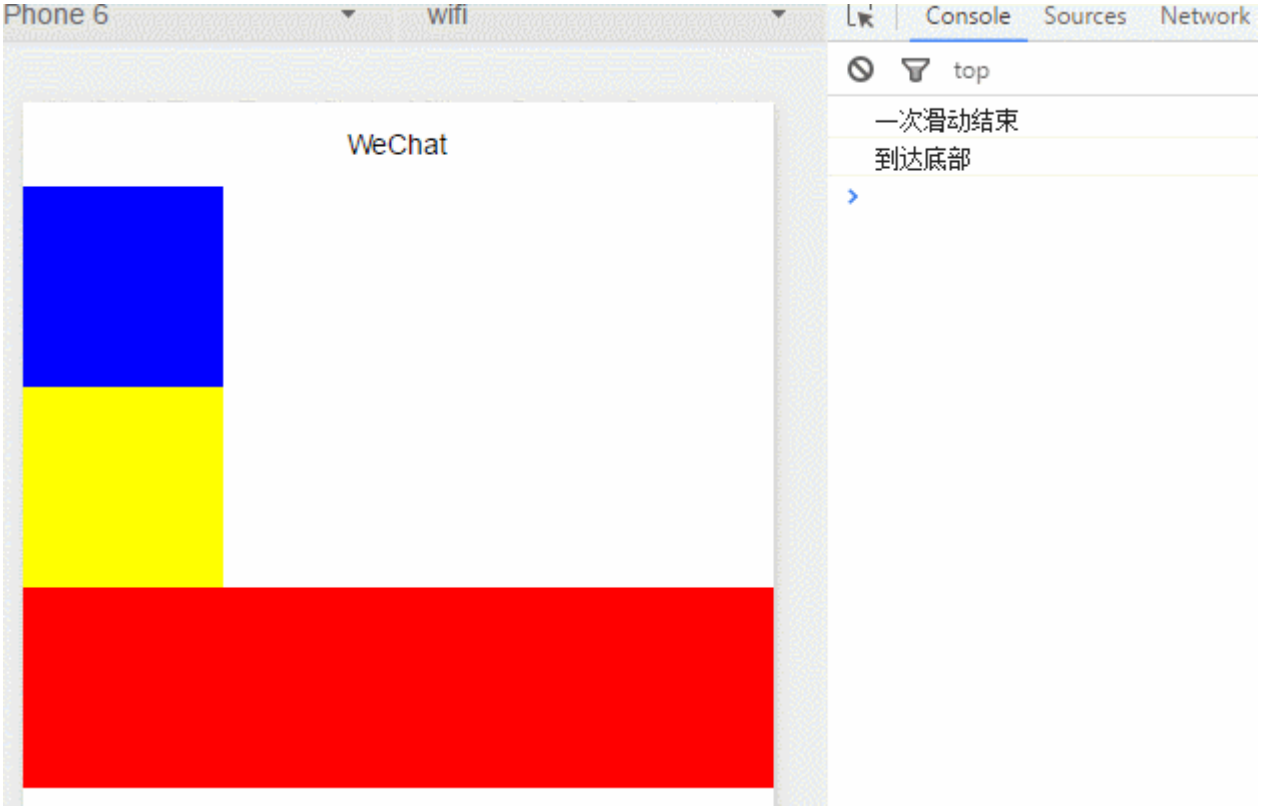
```
        <view id="yellow" class="scroll-view-item bc_yellow" style="width:100px;height:100px;background:yellow"></view>
    </scroll-view>
</view>
<view class="section section_gap">
    <scroll-view class="scroll-view_H" scroll-x="true" style=" white-space: nowrap" >
        <view id="green" style="width:400px;height:100px;background:red;display: inline-block"></view>
        <view id="red"   style="width:400px;height:100px;background:blue;display: inline-block"></view>
    </scroll-view>
</view>
```

- scroll-view.js 代码如下：

```
Page({
  data: {
    toView: "red",//设置初始滑动区顶部显示的 view，可通过 this.setdata({toView:"blue"})来改变
    scrollTop: 10,
    threshold:0
```

```
  },
  upper: function(e) {
    console.log("到达顶部")
  },
  lower: function(e) {
    console.log("到达底部")
  },
  scroll: function(e) {
    console.log("一次滑动结束")
  }
})
```

- 运行效果：



3.swiper

滑动面板，定时滑动切换图片或手动滑动切换图片。

- swiper.wxml

```
<!--indicator-dots 是否显示圆点 ,autoplay 自动播放 ,current 初始显示的 item( 0 代表第一个 item ), duration 滑动速度，bindchange
监听滚动和点击事件，interval 滑动间隔时间-->
<swiper indicator-dots="true" autoplay="true" current="1" duration="1000" bindchange="listenSwiper" interval="2000">
```

```
<!--swiper-item 只能包含一个节点再多会自动删除-->


    <swiper-item>
        <view style="height: 150px"><image src="../../pic/pic2.png" style="width:100%;height:100%"/></view>
    </swiper-item>
    <swiper-item>
        <view style="height: 150px"><image src="../../pic/pic1.png" style="width:100%;height:100%;"/>
        </view>
    </swiper-item>
     <swiper-item>
         <view style="height: 150px"><image src="../../pic/pic3.png"    style="width:100%;height:100%;"/>
         </view>
    </swiper-item>
</swiper>
```

- 运行效果：

4.icon，text，progress，checkbox，input，readio-group，slider，switch

```
<!-- type=[ 'success', 'info', 'warn', 'waiting', 'safe_success', 'safe_warn',

'success_circle', 'success_no_circle', 'waiting_circle', 'circle', 'download',

'info_circle', 'cancel', 'search', 'clear'] -->

<icon type="success" size="23" color="red"/>

<view><text>可以长按选中</text></view>

<!-- percent:百分比，active:是否显示从左往右的动画（但是经过测试无论设置为 true 还是 false 动画都会显示，去掉该属性则不显示动

画），showInfo：是否显示百分比，strokeWidth：进度条宽度 -->

<progress percent="100" active="false" showInfo="true" strokeWidth="7" />

<!-- type：[primary, default, warn],size:[default,mini],loading:按钮前是否带 loading 图标,plain:按钮是否镂空，disabled：是否生效，

formType:[submit,reset],hover-class:按下时的样式类名 -->

<button type="warn" size="mini" loading="true" plain="true" disabled="true" bindtap="default" formType="reset"
```

```
hover-class="none"> default </button>

<!-- disabled:不可选中，value：当 value 改变时触发 bindchange 绑定的函数  -->

<checkbox-group bindchange="checkboxChange">

    <label class="checkbox" wx:for-items="{{[1,2,3,4,5]}}">

        <checkbox value="{{item}}" checked='false' disabled="true"/>{{item}}

    </label>

</checkbox-group>


<input placeholder="禁用了的 input" value="禁用了的 input" type="text" auto-focus/>


<radio-group bindchange="test">

    <label class="radio" wx:for-items="{{[1,2,3,4]}}">

        <radio value="{{item}}" checked="true"/>{{item}}

    </label>

</radio-group>
```

```
<!-- disabled:禁用无法滑动，step:步长如果设置为 2 则显示 value 为 50 52 54...

      show-value:是否显示当前值

 -->

<slider bindchange="test" min="50" max="200" show-value step="2"/>


<!-- type:[checkbox,switch]两种样式，disabled,checked -->

<switch checked="true" bindchange="test" type="checkbox"/>

<switch checked="true" bindchange="test" type="switch"/>
```