# 笔记03

## 回顾

### 页面配置

只能配置window选项

### 场景值

如何进入到小程序的

### 小程序的逻辑层

小程序注册：

App.js 生命周期函数， any ， 在app中获取内部的数据使用的this

页面注册：

Page， 生命周其函数，事件监听函数，data ，any

getApp() 获取全局的应用实例

### wxml视图层

基本的数据绑定 {{}}

列表渲染 wx:for

条件渲染 wx:if

模板（template）/引入 import include

### 样式

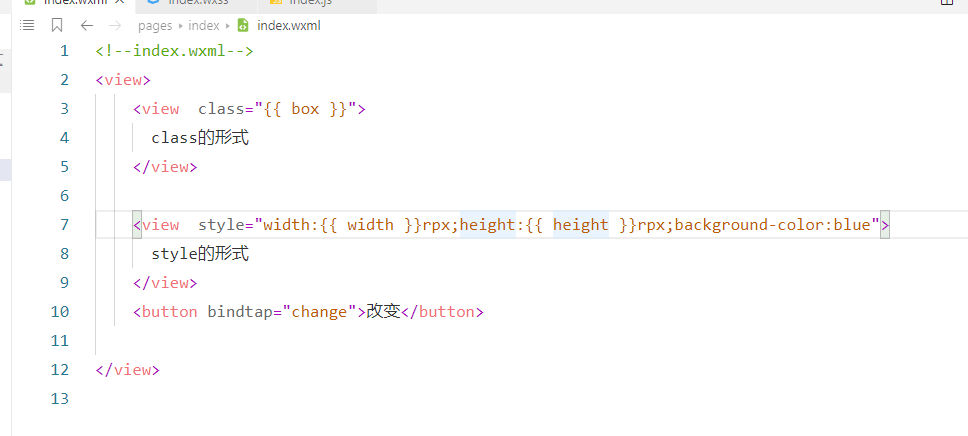
单位rpx， 样式导入 @import “相对路径”;

## 样式

### 动态样式/静态样式

Style ： 做动态样式

选择器（class）：静态样式





## 事件系统

### 什么事件？

事件就是视图层wxml到逻辑层js的通讯方式。

### 事件如何去绑定？

Bind+vlalue = “事件函数”

Catch+value = “事件函数”

注意：事件函数调用不能加括号

Value： 事件类型

| **类型** | **触发条件** | **最低版本** |
| --- | --- | --- |
| touchstart | 手指触摸动作开始 |  |
| touchmove | 手指触摸后移动 |  |
| touchcancel | 手指触摸动作被打断，如来电提醒，弹窗 |  |
| touchend | 手指触摸动作结束 |  |
| tap | 手指触摸后马上离开 |  |
| longpress | 手指触摸后，超过350ms再离开，如果指定了事件回调函数并触发了这个事件，tap事件将不被触发 | [1.5.0](https://developers.weixin.qq.com/miniprogram/dev/framework/compatibility.html) |
| longtap | 手指触摸后，超过350ms再离开（推荐使用longpress事件代替） |  |
| transitionend | 会在 WXSS transition 或 wx.createAnimation 动画结束后触发 |  |
| animationstart | 会在一个 WXSS animation 动画开始时触发 |  |
| animationiteration | 会在一个 WXSS animation 一次迭代结束时触发 |  |
| animationend | 会在一个 WXSS animation 动画完成时触发 |  |
| touchforcechange | 在支持 3D Touch 的 iPhone 设备，重按时会触发 | [1.9.90](https://developers.weixin.qq.com/miniprogram/dev/framework/compatibility.html) |

### 事件函数如何去写？

 // 自定义函数--  不需要使用methods

  fun:function(){

    console.log('fun传统形式')

  },

  // fun1:function(){

  //   console.log('fun1')

  // }

  fun1:()=>{

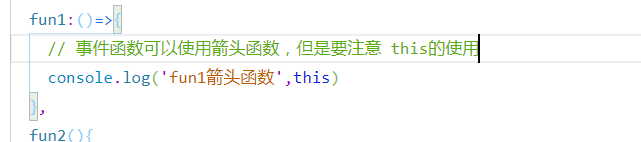
    console.log('fun1箭头函数')

  },

  fun2(){

    console.log('fun2简写形式')

  }



### 传参/接参 \*\*\*\*

#### 传参



Id： 属于公共属性，直接在组件上定义即可

Data-\* ： 用来设置自定义属性的传参值

#### 接参

事件函数中，事件事件对象的存在

Id： 直接是 E.target/currentTarget .id

Data-\*: e.target/currentTarget.dataset.属性

### 事件分类

冒泡事件/非冒泡事件

事件分为冒泡事件和非冒泡事件：

1. 冒泡事件：当一个组件上的事件被触发后，该事件会向父节点传递。
2. 非冒泡事件：当一个组件上的事件被触发后，该事件不会向父节点传递。

Tips：

冒泡事件如何触发：

Bind 允许事件冒泡

Catch 阻止事件冒泡

### 事件对象



注意事项：

1. Bind 与catch的区别是什么？

Bind 允许事件冒泡

Catch 阻止事件冒泡

1. Target/currentTarget 有什么区别？

Target ：事件源上的参数，点击谁就是谁上面的数据

currentTarget： 事件绑定组件上的数据

## 猜拳

参考代码

## 组件

### 基础组件的概念

什么是组件：

* 组件是视图层的基本组成单元。
* 组件自带一些功能与微信风格一致的样式。
* 一个组件通常包括 开始标签 和 结束标签，属性 用来修饰这个组件，内容 在两个标签之内。

<tagname property="value">

Content goes here ...

</tagname>

**注意：所有组件与属性都是小写，以连字符**-**连接**

### 视图容器

#### View



#### swiper

滑块视图容器

Swiper-item 组件必须放到swiper内，

其中只可放置[swiper-item](https://developers.weixin.qq.com/miniprogram/dev/component/swiper-item.html)组件，否则会导致未定义的行为。

Swiper 固定高度 150px 宽度 100%；

Swiper-item仅可放置在[swiper](https://developers.weixin.qq.com/miniprogram/dev/component/swiper.html)组件中，宽高自动设置为100%。

<!--

  swiper

  swiper-item

    indicator-dots="{{ true }}"

  indicator-color="#000"

  indicator-active-color="#fff"

 -->

<swiper  class="swiper"

  circular="{{ true }}"

  autoplay="{{ true }}"

  interval="1000"

  vertical="{{ false }}"

  display-multiple-items="1"

  bindchange="swiperchange"

>

   <swiper-item  wx:for="{{ imgs }}">

      <image src="{{ item }}" ></image>

   </swiper-item>

</swiper>

<view class="customdots">

  <view wx:for="{{ [1,2,3] }}"   class="customdots\_view {{ activeindex == index ? 'activedots' :'' }}">

  </view>

</view>

   \*/

  data: {

      imgs:[

        "/imgs/0.jpg",

        "/imgs/1.jpg",

        "/imgs/2.jpg",

      ],

      activeindex:0  // 默认显示第一个

  },

  swiperchange(e){

    // 组件的特殊属性/私有的事件，获取值的时候，是在detail里面获取

    // console.log(e.detail.current)

    let  activeindex = e.detail.current;

    this.setData({

      activeindex

    })

  }

})

#### scroll-view

可滚动视图区域

Y轴：

1. 必须给scroll-view设置sroll-y 为真
2. 给scroll-view加上一个固定的高度

<!-- 竖向滚动 y -->

<!-- <scroll-view class="scrollview" scroll-y="{{ true }}">

    <view class="red">red</view>

    <view class="green">green</view>

    <view class="blue">blue</view>

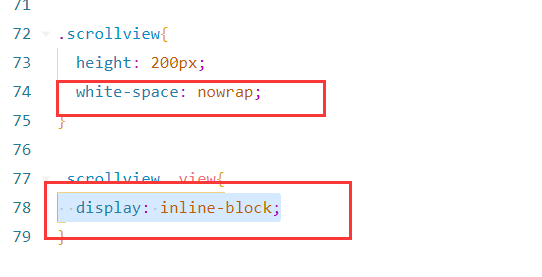
</scroll-view> -->

X轴：

1)必须给scroll-view设置sroll-x 为真

2) 子元素设置为 行内的块状

3） 父元素不换行



<scroll-view class="scrollview" scroll-x="{{ true }}">

    <view class="red">red</view>

    <view class="green">green</view>

    <view class="blue">blue</view>

</scroll-view>

### 基础组件

#### TEXT 文本

<!--pages/component2/component2.wxml-->

<!--

  selectable  可不可以被长按复制剪切( 在模拟器上演示用鼠标左键长按)

  space  连续显示空格

  decode  是否解码

   decode可以解析的有 &nbsp; &lt; &gt; &amp; &apos; &ensp; &emsp;

 -->

<text

  selectable="{{ true }}"

>text组件</text>

<view></view>

<text space="nbsp">你好           世界！</text>

<view></view>

<text decode="{{ true }}">我爱 &gt; 打游戏</text>

#### rich-text 富文本

1. tip: nodes 不推荐使用 String 类型，性能会有所下降。
2. tip: rich-text 组件内屏蔽所有节点的事件。
3. tip: attrs 属性不支持 id ，支持 class 。
4. tip: name 属性大小写不敏感。
5. tip: 如果使用了不受信任的HTML节点，该节点及其所有子节点将会被移除。
6. tip: img 标签仅支持网络图片。
7. tip: 如果在自定义组件中使用 rich-text 组件，那么仅自定义组件的 wxss 样式对 rich-text 中的 class 生效。

<!--

  rich-text  富文本   （v-html）

 -->

 <!-- <h2>我是h2标签</h2>  html string 形式-->

 <rich-text nodes="<h2>我是h2标签</h2>"></rich-text>

 <!-- array 数组形式 -->

 <rich-text nodes="{{ nodesArray }}"></rich-text>

 <rich-text nodes="{{ nodesBrray }}"></rich-text>

 data: {

    //   <div class="mydiv">我是div</div>

    nodesArray: [{

      type: "node",

      name: "div",

      attrs: {

        class: "mydiv"

      },

      children: [{

        type: "text",

        text: "我是div"

      }]

    }],

    /\*

        <div class="div1">

            <div class="red">我是red</div>

            <div class="blue">我是blue</div>

        <div>

        <div class="green">我是green</div>

    \*/

    nodesBrray: [{

        type: "node",

        name: "div",

        attrs: {

          class: "div1"

        },

        children: [{

            type: "node",

            name: "div",

            attrs: {

              class: "red"

            },

            children: [{

              type: "text",

              text: "我是red"

            }]

          },

          {

            type: "node",

            name: "div",

            attrs: {

              class: "blue"

            },

            children: [{

              type: "text",

              text: "我是blue"

            }]

          }

        ]

      },

      {

        type: "node",

        name: "div",

        attrs: {

          class: "green"

        },

        children: [{

          type: "text",

          text: "我是green"

        }]

      }

    ]

  },

### 媒体组件

#### Image

图片

Src 图片地址

  show-menu-by-longpress="{{ true }}" 开启长按图片显示识别小程序码菜单

bindload="imgload" 图片加载完毕调用

 lazy-load="{{ true }}" 图片懒加载

作业：

1. 课上代码 敲一遍
2. 继续完成 猜拳
3. 预习 api