WXSS-WXML-WXS语法

王红元 coderwhy

目录
content

小程序的样式写法

- **页面样式的三种写法：**
  - 行内样式、页面样式、全局样式
  - 三种样式都可以作用于页面的组件
- **如果有相同的样式：**
  - 优先级依次是：行内样式 > 页面样式 > 全局样式

home.wxss

```
.box {
  color: blue;
  font-size: 25px;
}

.page {
  font-size: 35px;
  background: green;
}

```

home.wxml | app.wxss | home.wxss

```
<!-- 1.行内样式 -->
<view style='color: red; font-size:20px;'>行内样式</view>

<!-- 2.页面样式 -->
<view class='box'>页面样式</view>

<!-- 3.全局样式 -->
<view class='container'>全局样式</view>

<!-- 4.三种样式同时作用 -->
<view style='color: orange; background: blue;' class='page app'>三种样式的作用</view>
```

app.wxss

```
.container {
  color: green;
  font-size: 30px;
}

.app {
  text-decoration: underline;
  background: purple;
}
```

WXSS支持的选择器

目前支持的选择器有：

选择器	样例	样例描述
.class	`.intro`	选择所有拥有 class="intro" 的组件
#id	`#firstname`	选择拥有 id="firstname" 的组件
element	`view`	选择所有 view 组件
element, element	`view, checkbox`	选择所有文档的 view 组件和所有的 checkbox 组件
::after	`view::after`	在 view 组件后边插入内容
::before	`view::before`	在 view 组件前边插入内容

coderwhy

WXSS优先级与CSS类似，权重如图

! important	style=""	#id	.class	element
∞	1000	100	10	1

wxss的扩展 – 尺寸单位

- **尺寸单位**
  - rpx（responsive pixel）：可以根据屏幕宽度进行自适应，规定屏幕宽为750rpx。
  - 如在 iPhone6 上，屏幕宽度为375px，共有750个物理像素，则750rpx = 375px = 750物理像素，1rpx = 0.5px = 1物理像素。

设备	rpx换算px (屏幕宽度/750)	px换算rpx (750/屏幕宽度)
iPhone5	1rpx = 0.42px	1px = 2.34rpx
iPhone6	1rpx = 0.5px	1px = 2rpx
iPhone6 Plus	1rpx = 0.552px	1px = 1.81rpx

- **建议：** 开发微信小程序时设计师可以用 iPhone6 作为视觉稿的标准。

Mustache语法

- **WXML基本格式：**
  - 类似于HTML代码：比如可以写成单标签，也可以写成双标签；
  - 必须有严格的闭合：没有闭合会导致编译错误
  - 大小写敏感：class和Class是不同的属性

- **开发中，界面上展示的数据并不是写死的，而是会根据服务器返回的数据，或者用户的操作来进行改变.**
  - 如果使用原生JS或者jQuery的话，我们需要通过操作DOM来进行界面的更新.
  - 小程序和Vue一样，提供了插值语法：Mustache语法(双大括号)

```
<view class="info">
  <view>{{ message }}</view>
  <view>{{ firstName + " " + lastName }}</view>
  <view>{{ date }}</view>
</view>
```

```
message: "Hello World",
firstName: "kobe",
lastName: "bryant",
date: new Date().toLocaleDateString()
```

逻辑判断 wx:if – wx:elif – wx:else

- **某些时候，我们需要根据条件来决定一些内容是否渲染：**
  - 当条件为true时，view组件会渲染出来
  - 当条件为false时，view组件不会渲染出来

```
<view class="score">
  <block wx:if="{{ score > 90 }}">
    <view>优秀</view>
  </block>
  <block wx:elif="{{ score > 80 }}">
    <view>良好</view>
  </block>
  <block wx:elif="{{ score >= 60 }}">
    <view>及格</view>
  </block>
  <block wx:else>
    <view>不及格</view>
  </block>
</view>
```

hidden属性

- **hidden属性:**
  - hidden是所有的组件都默认拥有的属性;
  - 当hidden属性为true时, 组件会被隐藏;
  - 当hidden属性为false时, 组件会显示出来;

```
<view hidden="{{false}}">哈哈哈</view>
```

- **hidden和wx:if的区别**
  - hidden控制隐藏和显示是控制是否添加hidden属性
  - wx:if是控制组件是否渲染的

列表渲染 – wx:for基础

- **为什么使用wx:for?**
  - 我们知道，在实际开发中，服务器经常返回各种列表数据，我们不可能一一从列表中取出数据进行展示；
  - 需要通过for循环的方式，遍历所有的数据，一次性进行展示；
- **在组件中，我们可以使用wx:for来遍历一个数组 （字符串 - 数字）**
  - 默认情况下，遍历后在wxml中可以使用一个变量index，保存的是当前遍历数据的下标值。
  - 数组中对应某项的数据，使用变量名item获取。

```
<view>
  <!-- 遍历一个数组 -->
  <view wx:for="{{['abc', 'cba', 'nba']}}">{{item}}</view>
  <!-- 遍历一个字符串 -->
  <view wx:for="{{'hello coderwhy'}}">{{item}}</view>
  <!-- 遍历一个数字 -->
  <view wx:for="{{10}}">{{item}}</view>
</view>
```

block标签

- **什么是block标签?**
  - 某些情况下，我们需要使用 wx:if 或 wx:for时，可能需要包裹一组组件标签
  - 我们希望对这一组组件标签进行整体的操作，这个时候怎么办呢?

- **注意:**
  - <block/> 并不是一个组件，它仅仅是一个包装元素，不会在页面中做任何渲染，只接受控制属性。
- **使用block有两个好处:**
  - 1）将需要进行遍历或者判断的内容进行包裹。
  - 2）将遍历和判断的属性放在block便签中，不影响普通属性的阅读，提高代码的可读性。

列表渲染 - item/index名称

- **默认情况下，item – index的名字是固定的**
  - 但是某些情况下，我们可能想使用其他名称
  - 或者当出现多层遍历时，名字会重复
- **这个时候，我们可以指定item和index的名称：**

```
<view class="books">
  <block
    wx:for="{{books}}"
    wx:key="id"
    wx:for-item="book"
    wx:for-index="i"
  >
    <view>{{book.name}}-{{i}}</view>
  </block>
</view>
```

列表渲染 – key作用

- **我们看到，使用wx:for时，会报一个警告：**
  - 这个提示告诉我们，可以添加一个key来提供性能。
- **为什么需要这个key属性呢?**
  - 这个其实和小程序内部也使用了虚拟DOM有关系（和Vue、React很相似）。
  - 当某一层有很多相同的节点时，也就是列表节点时，我们希望插入、删除一个新的节点，可以更好的复用节点;
- **wx:key 的值以两种形式提供**
  - 字符串，代表在 for 循环的 array 中 item 的某个 property，该 property 的值需要是列表中唯一的字符串或数字，且不能动态改变。
  - 保留关键字 *this 代表在 for 循环中的 item 本身，这种表示需要 item 本身是一个唯一的字符串或者数字。

```
<view class="books">
  <block wx:for="{{books}}" wx:key="id">
    <view>{{item.name}}</view>
  </block>
</view>
```

```
books: [
  { name: "算法导论", id: 111 },
  { name: "Vue3+TS", id: 112 },
]
```

什么是WXS?

- **WXS（WeiXin Script）是小程序的一套脚本语言，结合 WXML，可以构建出页面的结构。**
  - 官方：WXS 与 JavaScript 是不同的语言，有自己的语法，并不和 JavaScript 一致。（不过基本一致）
- **为什么要设计WXS语言呢?**
  - 在WXML中是不能直接调用Page/Component中定义的函数的.
  - 但是某些情况，我们可以希望使用函数来处理WXML中的数据(类似于Vue中的过滤器)，这个时候就使用WXS了
- **WXS使用的限制和特点：**
  - WXS 不依赖于运行时的基础库版本，可以在所有版本的小程序中运行；
  - WXS 的运行环境和其他 JavaScript 代码是隔离的，WXS 中不能调用其他 JavaScript 文件中定义的函数，也不能调用小程序提供的API；
  - 由于运行环境的差异，在 iOS 设备上小程序内的 WXS 会比 JavaScript 代码快 2 ~ 20 倍。在 android 设备 上二者运行效率无差异；

WXS的写法

- **WXS有两种写法：**
  - - 写在<wxs>标签中
  - - 写在以.wxs结尾的文件中
- **<wxs>标签的属性：**

属性名	类型	默认值	说明
module	String		当前 `<wxs>` 标签的模块名。必填字段。
src	String		引用 .wxs 文件的相对路径。仅当本标签为**单闭合标签**或**标签的内容为空**时有效。

- **每一个 .wxs 文件和 <wxs> 标签都是一个单独的模块。**
  - 每个模块都有自己独立的作用域。即在一个模块里面定义的变量与函数，默认为私有的，对其他模块不可见；
  - 一个模块要想对外暴露其内部的私有变量与函数，只能通过 module.exports 实现；

WXS的练习（一）

- 使用两种方式来计算一个数组的和：

```
<wxs module="total">
  function calcTotal(nums) {
    return nums.reduce(function(prevValue, item) {
      return prevValue + item
    }, 0)
  }
  module.exports = {
    calcTotal: calcTotal
  }
</wxs>

<view>{{total.calcTotal(nums)}}</view>
```

```
function calcTotal(nums) {
  return nums.reduce(function(prevValue, item) {
    return prevValue + item
  }, 0)
}
module.exports = {
  calcTotal: calcTotal
}
```

WXS的练习（二）

- **案例练习题目：**
  - 题目一：传入一个数字，格式化后进行展示（例如36456，展示结果3.6万）；
  - 题目二：传入一个事件，格式化后进行展示（例如100秒，展示结果为01:40）；

```
function formatCount(count) {
  count = Number(count)
  if (count > 100000000) {
    return (count / 100000000).toFixed(1) + "亿"
  } else if (count > 10000) {
    return (count / 10000).toFixed(1) + "万"
  } else {
    return count + ""
  }
}
```

```
function padLeft(time) {
  time = time + ""
  return ("00" + time).slice(time.length)
}

function formatDuration(duration) {
  var minute = Math.floor(duration / 60)
  var second = Math.floor(duration) % 60

  return padLeft(minute) + ":" + padLeft(second)
}
```