小程序推荐文章

***来源：[有用！关于微信小程序，那些开发文档没有告诉你的](http://link.zhihu.com/?target=http://www.wxapp-union.com/portal.php?mod=view&aid=327" \t "https://zhuanlan.zhihu.com/p/_blank)***

***作者：王婷婷***

本文由广研微信小程序的开发团队所做，作者为UI开发工程师王婷婷。本文从UI开发的角度，结合OM小程序的案例，剖析小程序的组件用法与传统HTML5标签的区别。

## **小程序与传统HTML5的区别**

小程序刚开放公测，互联网圈内开始了各种解读和猜测。其中有观点认为小程序和HTML5有着紧密关联，甚至小程序就是基于HTML5开发。

经过仔细研究文档和代码开发，从视图层的角度来说，小程序与传统HTML5还是有明显的区别，主要区别在于：

1、开发工具不同；

区别于H5的开发工具+浏览器Device Mode预览的模式，小程序的开发基于自己的开发者工具，可以实现同步本地文件+开发调试+编译+预览+上传+发布等一整套流程。

2、开发语言不同；

小程序自己开发了一套WXML标签语言和WXSS样式语言，并非直接使用标准的HTML5+CSS3。

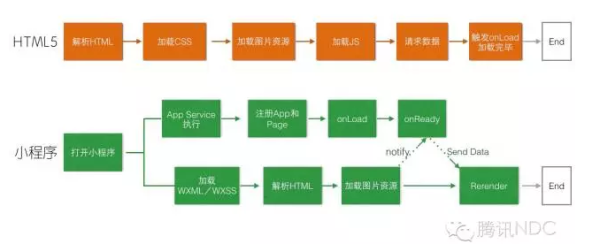
3、组件封装不同；

小程序独立出来了很多原生APP的组件，在HTML5需要模拟才能实现的功能，小程序里可以直接调用组件。

**小程序开发者工具**

微信小程序的开发工具，基于MINA框架（现已取消该名称），现在官方公布的工具名为**微信web开发者工具**。小程序开发工具是一种基于Native System系统层的框架，由于并非运行在浏览器中，所以JavaScript在web中的一些诸如Document、Window等方法无法使用。

从执行的速度方面，普通HTML5和小程序有哪些不同呢，用一张图表简单表示下：



* 传统HTML5在加载的时候受限于网络环境，需要顺序加载HTML、CSS、JS，然后返回数据，最后渲染页面显示在浏览器中。用户经常需要等待很长时间，体验会受到影响。
* 相比之下，小程序的两个线程：Appservice Thread和View Thread会同时进行、并行加载，甚至Appservice Thread会更早执行，当视图线程加载完，通知Appservice，Appservice 会把准备好的数据用setData的方法返回给视图线程。

小程序的这种优化策略，可以减少用户的等待时间、加快小程序的响应速度。

### **WXML**

**1、标签**

WXML在语法上更接近XML语言，遵循SGML规范，区别于HTML语言随意的标签闭合方式，WXML语言必须包括开始标签和结束标签，以image标签为例，以下2种写法都支持：

<image src="" />

or<image src="" ></image>

这里需要注意的是：

**所有组件与属性都是小写，以连字符-连接。**

**2、文件引入**

WXML提供两种文件引入方式，import和include。区别在于：import可以引入定义好的template模板，模板是有作用域的；而include就是拷贝一个公用的代码片段到目标文件中，适合做公共页面片的拆分。

*<!--import-->*<import src="../template/a.wxml"/>*<!--include-->*<include src="../include/footer.wxml"/>

文件引入在小程序做模块化拆分的过程中非常重要。

### **WXSS**

**1、尺寸单位**

WXSS支持的单位有px、rem和rpx，其中rem和rpx可以针对屏幕容器进行适配，px则为固定尺寸。 其中1rpx=0.5px，在WXSS和WXML中定义的rpx单位最终会转换为在手机端可以识别的rem单位。

**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。   
所以工程师拿到750的设计稿，在PS中量取的容器大小，可以直接定义为rpx，不需要进行2倍尺寸的换算。

view{

font-size:26rpx;

width:400rpx;

height:400rpx;

}

备注：rpx的单位不光在样式中会自适应，写在WXML的style里也会根据屏幕自适应。

**2、样式引入**

看到很多文章说小程序不支持样式的@import，其实官方公布的第一个正式开发者工具就已经支持了。

import "../../wxss/common.wxss";

**3、选择器**

小程序支持的选择器在官方公布的文档中包括.class、#id、 element、element,element、::after(注意是双冒号)、::before这6种选择器。

经过测试，小程序对于:first-child、:last-child、.class-a .class-b{}，甚至更多层级的嵌套都是支持的。 不过官方并不推荐级联的这种写法，因为考虑到后面切Native的扩展可能，会没办法支持级联选择。

所以保险起见，不建议.class-a .class-b{}这种级联的写法，以免后期工具过滤导致页面错乱。

### **组件**

小程序在0.10.102800的版本中加入了 textarea，并即将废弃操作反馈的系列组件。   
我们简单通过一个表格来对比下HTML5和小程序的组件标签的区别；



下面一一来分析下：

**1、view**

div和view都是盒模型，默认display:block。 盒模型在布局过程中，一般推荐display:flex的写法，配合justify-content:center;align-items:center;的定义实现盒模型在横向和纵向的居中。

**2、text**

除了text文本节点以外的其他节点都无法长按选中。截止到0.10.102800的开发者工具text支持嵌套text，不过有class的text会被面板过滤，样式不影响。

<text class="text-tips">

<text class="text-light">\* </text>友情提示!</text> *<!--支持text嵌套text-->*<text>

<view></view></text> *<!--不支持text嵌套其他标签-->*

**3、 icon**

icon可以直接用微信组件默认的图标，默认是iconfont格式的，从WeUI那边沿袭过来的一种做法。 自定义的icon推荐svg sprite格式或者iconfont。

目前来看，市面上还没有很好的自动合并单个svg为svg sprite的工具，需要手动拼图。

**4、input**

input 的类型，有效值：text, number, idcard, digit, time, date 。 input不需要设置line-height或padding来纵向居中，默认placeholder的文字是居中的。 小程序把checkbox和radio都单独做成了组件，默认的input只支持输入文本。 上传文件在小程序里需要调用chooseImage事件完成；

<input type="file" class="upload-input" accept="image/\*">*<!--HTML5的上传文件-->*<view class=" upload-block" bindtap="chooseImage"></view><view class="upload-block" wx:for="{{imageList}}" wx:for-item="image">

<image src="{{image}}" class="pic" data-src="{{image}}" bindtap="previewImage"></image></view>*<!--小程序的上传图片，可以调用app原生的拍照和上传图片接口-->*

小程序CSS里的 :focus 不生效，需要修改placehoder的样式，通过placeholder-class=”class”来定义。

.login .input-group input::-webkit-input-placeholder {

color: #c0c0c0;

}

.login .input-group input:focus::-webkit-input-placeholder {

color: transparent;

}*<!--HTML5通过focus修改placeholder默认和点击时候的样式-->*<input type="text" placeholder="邮箱" placeholder-style="color:#c0c0c0" /><input password type="number" placeholder="密码" placeholder-class="placeholder" />*<!--小程序里通过placeholder-style和placeholder-class修改样式，不过并不能修改点击时候input的边框颜色-->*

**5、picker**

picker默认支持普通、日期和时间三种选择器。 picker通过bindchange事件来调取range中自定义的数据数据，并展示到页面中，调用的是系统原生的select。

这里小程序废弃了select组件，考虑到的是这个组件的交互不适合移动场景，最终用picker代替了。

<select class="select-block">

<option value="0">选择</option>

<option value="1">北京</option>

<option value="2">上海</option></select>*<!--HTML5的下拉框-->*<picker bindchange="bindPickerChange" value="{{index}}" range="{{area}}">

<view class="picker">

{{area[index]}}

</view></picker>

Page({

data: {

area: ['中国', '美国', '巴西', '日本'],

}

})*<!--js里的area数组--><!--picker下拉框-->*

**6、 navigator**

navigator支持相对路径和绝对路径的跳转，默认是打开新页面，当前页面打开需要加redirect；

navigator仅支持5级页面的跳转；

navigator不可跳转到小程序外的链接地址；

**<**navigator **class=**"navigator" redirect url**=**"../login/index" **>**登录页**<**/navigator>

在小程序开发工具里，默认打开新页面，工具左上角有返回按钮。加上redirect，当前页打开，不出现返回按钮。

**7、image**

小程序的image与HTML5的img最大的区别在于：小程序的image是按照background-image来实现的。 默认image的高宽是320\*240。必须通过样式定义去覆盖这个默认高宽，auto在这里不生效。

（**开发者说这样设置的原因是：如果设置 auto ，页面布局会因为图片加载的过程有一个闪的现象（例如高度从 0 到 height ），所以要求一定要设置一个宽度和高度。**）

最新的api支持获取图片的高宽。不过这里返回的高宽是px单位，不支持屏幕自适应；

图片包括三种缩放模式scaleToFill、aspectFit、aspectFill和9种裁剪模式，三种缩放模式的实现原理对应如下：

scaleToFill{

background-size:100% 100%;//不保持纵横比缩放图片，使图片的宽高完全拉伸至填满 image 元素

}

aspectFit{

background-size:contain;//保持纵横比缩放图片，使图片的长边能完全显示出来。也就是说，可以完整地将图片显示出来。

}

aspectFill{

background-size:cover;//保持纵横比缩放图片，只保证图片的短边能完全显示出来。也就是说，图片通常只在水平或垂直方向是完整的，另一个方向将会发生截取。

}

**8、button**

额外补充下button的实现方式，button的边框是用:after方式实现的，用户如果在button上定义边框会出现两条线，需用:after的方式去覆盖默认值。不过这个应该会在最近的开发者工具中修复。

button::after {

content:" ";

width:200%;

height:200%;

border:1px solid rgba(0, 0, 0, 0.2);

}

小程序不支持button:active这种样式的写法，button的点击态通过.button-hover{}的样式覆盖，也可修改hover-class为自定义的样式名。

**9、css3动画**

最新版的开发工具已经支持transition和keyframes动画，不用js苦哈哈的写动画队列了。

除了官方公布的小程序的组件之外，有一些标签比如，span、em、strong、b也是支持的，只是官方并不推荐使用。

### **浏览器内核**

在iOS平台上，微信的浏览器渲染内核是wkwebview；

而在Android平台上，微信则采用了腾讯QQ浏览器2016年4月份发布的X5内核（blink内核）作为渲染引擎。

在小程序的开发工具上，小程序的JavaScript是运行在chrome内核（nwjs）中。

### **autoprefixer**

小程序会在接下来的版本中加入自动补全css前缀，使用的插件是postcss的autoprefixer，设置的兼容级别是> ios 8及> android 4.1。

const browserOptions = {

browsers: [

'last 3 versions',

'ios >= 8',

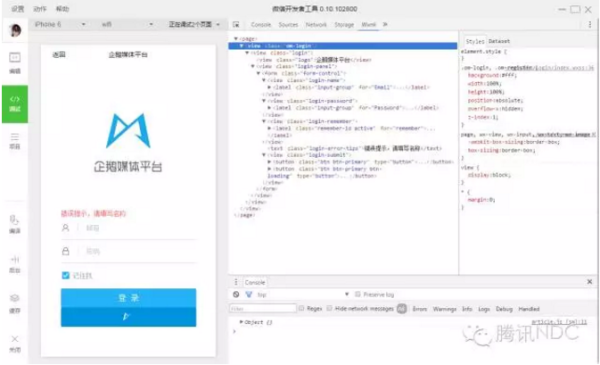
'android >= 4.1',

]

}

也就是说，我们在写css的时候只需要写没有前缀的写法。比如：display:flex，工具自动编译为display:flex;display:-webkit-flex。

### **OM小程序实战**

上图为OM小程序的开发界面。下面我们从布局、智能裁图和loading动画几个方面简单说下OM小程序具体的UI开发经验。

**1、flex布局**



以上图om的文章列表为例，文章的形态包括纯文字的和图文混合的。图文混合的文字不管是1行还是2行都需要相对于图片纵向居中。用flex的布局，就可以实现这3种状态不修改CSS文件，只按需隐藏DOM结构就搞定。

.media {

display: flex;

justify-content:center;//设置或检索弹性盒子元素在主轴（横轴）方向上的对齐方式。

align-items:center;//定义flex子项在flex容器的当前行的侧轴（纵轴）方向上的对齐方式。

}

.media .content {

flex: 1;

}

在做传统H5的时候，为了兼容各种低端设备的机型，通常不太敢轻易尝试flex，但在小程序里就可以大胆的使用了。

**2、智能裁图**

正是由于小程序把图片处理成背景图片，OM的素材管理页面图片的实现方式在这里遇到了一个挑战。 简单列举下om各种不同尺寸图片在平台上的展示方案。

1、高<50px：

（1）宽<175px，原图居中显示；   
（2）宽>175px，定宽等比上下居中显示；

2、50px<高<400px：

（1）宽<175px，原图居中显示；   
（2）宽>175px，定宽等比显示；

3、高>400px：

（1）宽<175px，原图居中显示、超出400px高度隐藏；   
（2）宽>175px，定宽等比显示、超出400px高度隐藏；

这种方案，用css和img实现起来，只需要设置外层盒子最大高宽，图片自适应缩放即可。

.pic-list .pic-item .pic-body .pic {

width: 100%;

text-align: center;

overflow: hidden;

max-height: 4rem;

min-height: 0.5rem;

display: -webkit-box;

-webkit-box-orient: horizontal;

-webkit-box-align: center;

-webkit-box-pack: center;

}

.pic img {

-webkit-box-flex: 1;

}

然而因为小程序里是用背景图片的方式，简单的css设置并不能实现智能裁图的方案。需要配合js计算出不同尺寸图片对应的适配尺寸。

.pic-list .pic-item .pic-body .pic {

width: 100%;

text-align: center;

overflow: hidden;

max-height: 800rpx;

min-height: 100rpx;

display:flex;

align-items:center;

justify-content:center;

}

这里需要后台接口提供数据列表的图片高宽，js对拿到图片的不同尺寸进行算法计算，重新赋值再返回给数据。

**3、css3动画改变默认loading**

小程序默认提供的loading是普通的菊花loading，这里OM使用自定义的keyframes序列帧动画。



.icon-loading {

animation: loadingWhite 1.2s infinite linear;

animation-timing-function: steps(10);

}

@keyframes loadingWhite {

0% {

background-position:0 0;

}

100%{

background-position:-500rpx 0;

}

}

### **写在最后**

微信小程序集成了很多原生APP的组件，从体验和页面流畅度来说，都会比HTML5要优秀很多。

微信小程序相对于HTML5开发来说，除了熟悉API需要学习成本之外，开发难度指数3颗星，还是很容易上手的。

开发者工具、组件和API目前刚刚对外公测，还不算太成熟，里面还有很多坑需要开发者去填。

文章篇幅有限，在这里只能简单从UI开发角度介绍下小程序的开发经验。有关小程序的更多API，可以查阅小程序开发文档 。

由于开发工具一直在持续更新，文章中可能会有不准确或者更新不及时的地方，还请谅解。

——————————————————————————————————————————————————————————————————————————————————

获取更多小程序尖货，请访问：[微信小程序联盟](http://link.zhihu.com/?target=http://www.wxapp-union.com" \t "https://zhuanlan.zhihu.com/p/_blank)，或关注微信小程序联盟微信号。

[http://weixin.qq.com/r/Nzjq8mzEtjHJrSiR920l](http://link.zhihu.com/?target=http://weixin.qq.com/r/Nzjq8mzEtjHJrSiR920l" \t "https://zhuanlan.zhihu.com/p/_blank) (二维码自动识别)

<https://zhuanlan.zhihu.com/p/23536784> 源地址

# 条件渲染

### wx:if

在框架中，使用 wx:if="{{condition}}" 来判断是否需要渲染该代码块：

### block wx:if

因为 wx:if 是一个控制属性，需要将它添加到一个标签上。如果要一次性判断多个组件标签，可以使用一个 <block/> 标签将多个组件包装起来，并在上边使用 wx:if 控制属性。

****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。

### wx:if vs hidden

因为 wx:if 之中的模板也可能包含数据绑定，所有当 wx:if 的条件值切换时，框架有一个局部渲染的过程，因为它会确保条件块在切换时销毁或重新渲染。

同时 wx:if 也是****惰性的****，如果在初始渲染条件为 false，框架什么也不做，在条件第一次变成真的时候才开始局部渲染。

相比之下，hidden 就简单的多，组件始终会被渲染，只是简单的控制显示与隐藏。

一般来说，wx:if 有更高的切换消耗而 hidden 有更高的初始渲染消耗。因此，如果需要频繁切换的情景下，用 hidden 更好，如果在运行时条件不大可能改变则 wx:if 较好。

# 模板

WXML提供模板（template），可以在模板中定义代码片段，然后在不同的地方调用。

### 定义模板

使用 name 属性，作为模板的名字。然后在<template/>内定义代码片段，如：

<template name="msgItem">

<view>

<text> {{index}}: {{msg}} </text>

<text> Time: {{time}} </text>

</view></template>

### 使用模板

使用 is 属性，声明需要的使用的模板，然后将模板所需要的 data 传入，如：

<template is="msgItem" data="{{...item}}"/>

is 属性可以使用 Mustache 语法，来动态决定具体需要渲染哪个模板：

<block wx:for="{{[1, 2, 3, 4, 5]}}">

<template is="{{item % 2 == 0 ? 'even' : 'odd'}}"/></block>

### 模板的作用域

模板拥有自己的作用域，只能使用data传入的数据。

# 事件

## 什么是事件

事件是视图层到逻辑层的通讯方式。

事件可以将用户的行为反馈到逻辑层进行处理。

事件可以绑定在组件上，当达到触发事件，就会执行逻辑层中对应的事件处理函数。

事件对象可以携带额外信息，如 id, dataset, touches。

## 事件的使用方式

在组件中绑定一个事件处理函数。

如bindtap，当用户点击该组件的时候会在该页面对应的Page中找到相应的事件处理函数。在相应的Page定义中写上相应的事件处理函数，参数是event。

## 事件详解

### 事件分类

事件分为冒泡事件和非冒泡事件：

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。

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

WXML的冒泡事件列表：

| 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://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html" \o "基础库 1.5.0 开始支持，低版本需做兼容处理。) |
| longtap | 手指触摸后，超过350ms再离开（推荐使用longpress事件代替） |  |

****注：除上表之外的其他组件自定义事件如无特殊申明都是非冒泡事件，如**[<form/>](https://mp.weixin.qq.com/debug/wxadoc/dev/component/form.html)**的**submit**事件，**[<input/>](https://mp.weixin.qq.com/debug/wxadoc/dev/component/input.html)**的**input**事件，**[<scroll-view/>](https://mp.weixin.qq.com/debug/wxadoc/dev/component/scroll-view.html)**的**scroll**事件，(详见各个**[组件](https://mp.weixin.qq.com/debug/wxadoc/dev/component/)**)****

### 事件的捕获阶段

自基础库版本 [1.5.0](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/compatibility.html" \o "基础库 1.5.0 开始支持，低版本需做兼容处理。) 起，触摸类事件支持捕获阶段。捕获阶段位于冒泡阶段之前，且在捕获阶段中，事件到达节点的顺序与冒泡阶段恰好相反。需要在捕获阶段监听事件时，可以采用capture-bind、capture-catch关键字，后者将中断捕获阶段和取消冒泡阶段。

在下面的代码中，点击 inner view 会先后调用handleTap2、handleTap4、handleTap3、handleTap1。

<view id="outer" bind:touchstart="handleTap1" capture-bind:touchstart="handleTap2">

outer view

<view id="inner" bind:touchstart="handleTap3" capture-bind:touchstart="handleTap4">

inner view

</view></view>

如果将上面代码中的第一个capture-bind改为capture-catch，将只触发handleTap2。

<view id="outer" bind:touchstart="handleTap1" capture-catch:touchstart="handleTap2">

outer view

<view id="inner" bind:touchstart="handleTap3" capture-bind:touchstart="handleTap4">

inner view

</view></view>

https://mp.weixin.qq.com/debug/wxadoc/dev/framework/view/wxml/event.html