`

爱创课堂前端培训

# 微信小程序

第4天课堂笔记（本课程共1天）

班级：北京前端训练营25期

讲师：彭帅伟

日期：2019年9月19日

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

目录

[微信小程序](#_Toc28724)

[复习：](#_Toc4251)

[一、 KOA](#_Toc16544)

[1.1 cookie](#_Toc19153)

[1.2 session](#_Toc13232)

[1.3 EJS模板](#_Toc8241)

[1.4 async和await](#_Toc27907)

[二、微信小程序](#_Toc30845)

[2.1小程序简介](#_Toc17776)

[2.1.1注册账号](#_Toc228)

[2.1.2安装开发工具](#_Toc22098)

[2.2 目录部署](#_Toc12947)

[2.2.1 项目配置](#_Toc27743)

[2.2.2 应用配置](#_Toc26305)

[2.3 应用程序](#_Toc1531)

[2.3.1 内置方法](#_Toc11296)

[2.3.2全局方法](#_Toc24935)

[2.3.3页面周期方法](#_Toc28894)

[2.4 渲染页面](#_Toc325)

[2.4.1 数据渲染原理](#_Toc28055)

[2.4.2数据驱动](#_Toc575)

[2.4.3插值语法](#_Toc24803)

[2.4.4更新数据](#_Toc13007)

[2.4.5 数据丢失](#_Toc26738)

[2.5 wxml](#_Toc9860)

[2.5.1 属性](#_Toc27593)

[2.5.2事件](#_Toc2105)

[2.5.3 事件对象](#_Toc28188)

[2.6 组件](#_Toc19212)

[2.6.1 view](#_Toc2018)

[2.6.2 text](#_Toc6177)

[2.6.3 rich-text](#_Toc3599)

[2.6.4 image](#_Toc1911)

[2.6.5 icon](#_Toc23578)

[2.6.6 canvas](#_Toc25681)

[2.6.7 map](#_Toc7993)

[2.6.8 多媒体组件](#_Toc518)

[2.7 指令](#_Toc13436)

[2.7.1 条件指令](#_Toc3126)

[2.7.2 循环指令](#_Toc30885)

# 复习：

类：在ES6中实现了class关键字

语法：

class 类名{}

数据类型：

1 自身数据 通过constructor(arg1, arg2, ...)

2 原型数据 直接在类体中定义方法 如果是属性数据 必须使用get set特性方法

3 静态数据 在数据的前面加上static

继承：

语法： class 子类 extends 父类 {}

全部继承

ES Module:

引入模块文件：

1 引入全部模块文件

import \* as 模块 from 文件

2 从模块文件中解构出几个接口

import { data1, data2 } from 文件

3 引入默认接口

import 模块 from 文件

暴露接口：

直接在模块文件中 通过export暴露接口

通过export default 暴露默认接口

TS: 是一个强类型语法，任何数据都要指明类型

数据类型：

1 js中的基本类型 ts中都支持...

2 还拓展了 any void never

类型猜测：

定义变量的时候，没有设置类型，此时将进行类型猜测

尽量不要让ts进行类型猜测：

1 临时分配空间 消耗性能

2 往往不是我们要的结构

类型推断：

语法：<type>数据 数据 as 类型

数组：

let arr:type[] = []

元组：

let arr:[type1, type2] = []

类型级联：

语法： type1 | type2 | type3

枚举类型：

语法： enum 枚举名称 {}

函数：

函数的参数以及返回值要设置类型

语法：

function demo(arg:type, arg2?:type):type {}

返回值类型：

1 返回数据 此时返回值的类型就是数据类型

2 没有返回值 此时返回值的类型是void

3 函数中出现了错误 此时返回值类型是never

泛型：

如果想要参数与返回值一致，就要使用泛型

语法：

function demo<T>(arg:T):T{}

使用函数：

1 demo<type>（数据）

2 demo(数据) 常用

类：

语法： class 类名 类名的首字母要大写

属性： 在类体中要声明属性，声明属性的时候要设置类型，并且还可以赋值

如果没有声明类型的属性，在编译的时候直接删除

声明的属性在构造函数中可以直接赋值

方法：定义方法与ES6中定义的方法是一致的，只不过要定义函数的参数已经返回值的类型

关键字：private protected public static 其中前三个在js中无法实现，在编译的时候直接删除

static用于定义静态数据

在ts中，不能在类的外部定义静态数据，只能修改

实例化：

变量的类型：就是类的类型

接口：

函数接口：函数的参数以及返回值属于结构，所以要为这些数据定义类型

interface 接口名称 {

只定义结构，不要实现

}

函数接口：

interface 函数接口 {

（arg1:type, arg2?: type）:type

}

对象接口：

interface 对象接口 {

key?: type,

method(arg:type, arg1?:type):type

}

类接口：

interface 对象接口 {

key?: type,

method(arg:type, arg1?:type):type

}

实现类接口要通过implements关键字来实现

实例化对象的类型：

如果是类的类型： 可以使用类中所有的数据

如果是接口类型： 可以使用接口中声明的数据

模块：

通过Module 关键字定义

在对象中通过export 暴露接口

Koa: 下载 npm install koa

静态化：通过引入koa-static

路由：引入koa-router 实例化对象中提供了routes方法用于配置路由规则

# KOA

## 1.1 cookie

cookie是在浏览器端存储，一般是伴随请求发送的

cookie只能用于存储非机密性数据

在Koa中的ctx对象中提供了cookies对象，用于操作cookie

set方法用于设置数据

第一个参数是属性名称

第二个参数是数据内容

第三个是配置对象

get 用来获取cookie数据的

参数就是数据名称

demo:

|  |
| --- |
| 1. // 处理post请求 2. router.post('/login', ctx => { 3. // 存储cookie数据 4. let { username, password } = ctx.request.body; 5. // 设置cookie 6. ctx.cookies.set('username', username, { 7. maxAge: 1000 \* 60 \* 60 \* 24 \* 10, 8. httpOnly: false 9. }); 10. }) |

前端获取cookie:

|  |
| --- |
|  |

## 1.2 session

session中的数据存储在后端，用于存储重要的机密性的文件

通过ctx.session 即可设置或者获取session内容

使用方式：

1 引入koa-session模块

2 此时要为app设置keys字段，

3配置session

code:

|  |
| --- |
| 1. // 处理post请求 2. router.post('/login', ctx => { 3. // 存储cookie数据 4. let { username, password } = ctx.request.body; 5. console.log(username, password); 6. // 设置cookie 7. ctx.cookies.set('username', username, { 8. maxAge: 1000 \* 60 \* 60 \* 24 \* 10, 9. httpOnly: false 10. }); 11. // 设置用户密码 12. ctx.session.password = password; 13. }) 14. // 获取cookie和session 15. router.get('/show', ctx => { 16. console.log(59, ctx.cookies.get('username')); 17. console.log(60, ctx.session.password); 18. }) |

## 1.3 EJS模板

express中，默认支持ejs模板引擎。

koa也支持ejs模板引擎，要引入koa-ejs模块

并且还要进行ejs配置

ejs(app, {

viewExt: 配置拓展名

root: 配置目录

layout: 是否让所有的页面使用同一模板

})

|  |
| --- |
| 1. // 处理get请求 2. router.get('/show', ctx => { 3. ctx.render('/login', { 4. title: '爱创课堂', 5. username: '王老五' 6. }); 7. }); |

## 1.4 async和await

在koa程序中可以直接使用async 和 await

|  |
| --- |
| 1. // 定义任务 2. function task1() { 3. return new Promise((resolve, reject) => { 4. // 1s执行成功 5. setTimeout(() => { 6. // 问题解决 7. resolve(); 8. }, 1000) 9. }) 10. } 11. // 处理get请求 12. router.get('/show', async ctx => { 13. await task1(); 14. // 处理接收到的结果 15. }); |
|  |
|  |

# 二、微信小程序

## 2.1小程序简介

微信小程序是嵌入在微信中的一种程序，只有2m的空间，因此称之为小程序，是轻量级的

小程序也属于混合开发

一处开发，可以在多端使用的开发技术

### 2.1.1注册账号

进入https://mp.weixin.qq.com/，点击立即注册按钮，选择小程序



输入相关信息， 在信息登记中，选择个人账户，输入身份信息



注册完成，进入设置页面，选择开发设置，获取appid

开发文档

<https://developers.weixin.qq.com/miniprogram/dev/reference/>

### 2.1.2安装开发工具



下载地址

<https://developers.weixin.qq.com/miniprogram/dev/devtools/devtools.html>

点击安装程序，一直点击下一步安装，安装完成进入首页

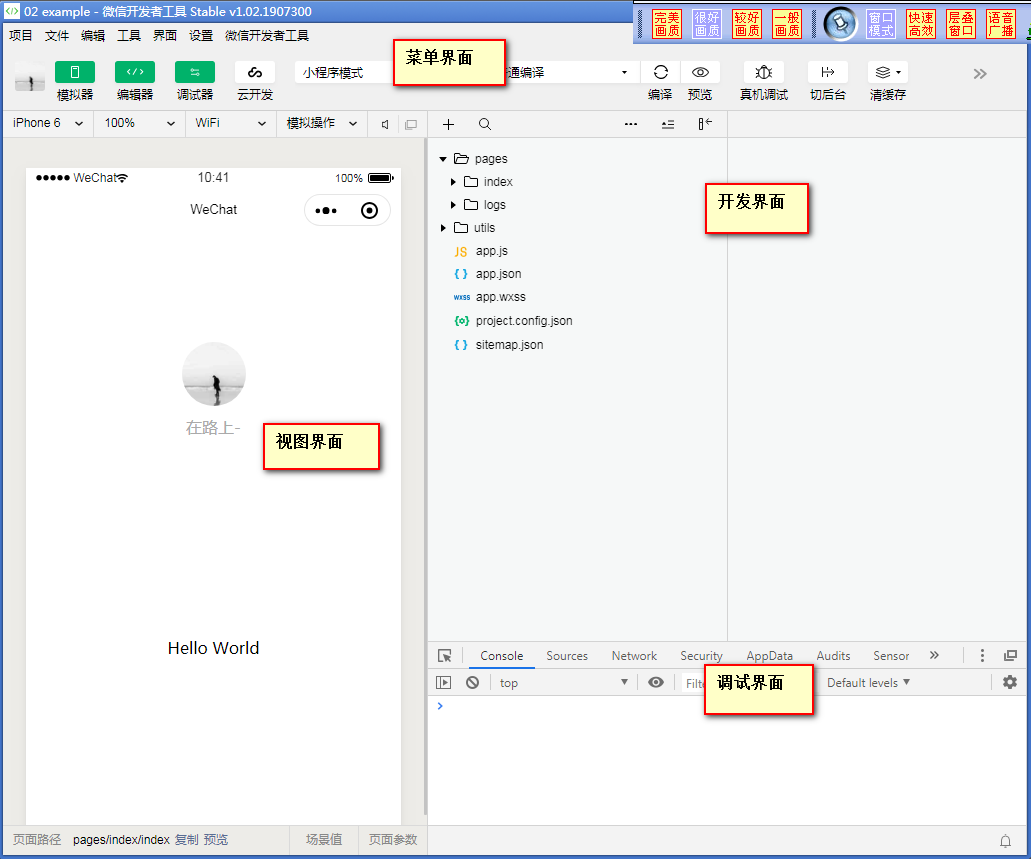
获取APPID:



登录界面：



首页：



## 2.2 目录部署

pages: 用于配置所有页面

index: 是首页

每一个页面文件中 有四个文件

index.js: 脚本文件

index.json: 配置文件

indx.wxml: 模板文件

index.wxss: 样式文件

logs: 日志页面

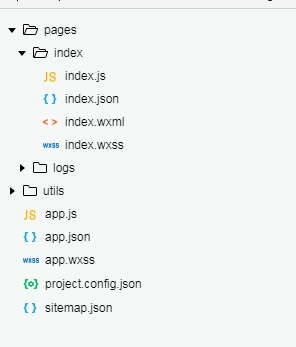
uitils: 插件工具

app.js: 应用程序脚本文件

app.json: 应用程序配置文件

app.wxss: 全局样式文件

project.config.json: 项目配置文件



### 2.2.1 项目配置

app.json要遵守严格json语法（注意，双引号，逗号要注意）

pages： 用于配置页面 值是一个数组，数组中的每一项都是一个页面地址，数组中的第一项表示首页

window: 配置窗口的

backgroundTextStyle：字体样式

navigationBarBackgroundColor： 导航栏的背景颜色

navigationBarTitleText：提示文字

navigationBarTextStyle： 字体样式

tabBar: 用于配置页面下方的icon

backgroundColor: tabBar的背景颜色

list: 代表icon的列表 （通常是2-5个）

pagePath： 对应的页面地址

text：提示文字

iconPath： icon图片地址

selectedIconPath： 选中时候的icon图标地址

networkTimeout： 配置网络时间的

request: 请求时间

connectSocket： socket时间

uploadFile： 上传时间

downloadFile：下载时间

显示：

|  |
| --- |
|  |

### 2.2.2 应用配置

project.config.json: 代表应用配置

里面存储很多信息：

例如 appId， 好比是npm中的package.json文件

|  |
| --- |
|  |

## 2.3 应用程序

app.wxss 用于配置全局样式的，在该文件中配置的样式，会在所有的页面中得到应用

app.js 应用程序配置文件

小程序为应用程序提供了一些周期方法

onLaunch 当页面加载完毕执行的方法

onShow: 后台进入前台执行的方法

onHide: 前台进入后台的时候执行的方法

onError: 出错时候执行的方法

前台与后台：

前台：我们看到的页面

后台：我们看不到的页面

注意：当我们把微信切换到后台的时候，此时微信小程序不会立即销毁，而是等待一段时间之内，没有操作再去销毁

### 2.3.1 内置方法

wx.login 用于登录

参数是一个对象

success 登录成功时候执行的函数

wx.getSetting： 用于获取授权信息

参数是一个对象

success 获取授权信息成功之后执行的回调函数

wx.getUserInfo: 用于获取用户信息

参数是一个对象

success 获取用户信息成功时候执行的回调函数

### 2.3.2全局方法

APP: 创建应用程序

getApp: 获取应用程序

Page: 创建页面

getCurrentPages: 获取当前页面

### 2.3.3页面周期方法

onLoad： 页面加载完毕

onReady: 页面渲染完成

onShow： 页面渲染

onHide： 页面隐藏

onUnload：页面卸载

onPullDownRefresh： 监听用户下拉动作

onReachBottom：监听页面是否到底部

onShareAppMessage： 监听用户分享

|  |
| --- |
| 1. Page({ 2. /\*\* 3. \* 页面的初始数据 4. \*/ 5. data: { 7. }, 8. /\*\* 9. \* 生命周期函数--监听页面加载 10. \*/ 11. onLoad: function (options) { 13. }, 14. /\*\* 15. \* 生命周期函数--监听页面初次渲染完成 16. \*/ 17. onReady: function () { 19. }, 20. /\*\* 21. \* 生命周期函数--监听页面显示 22. \*/ 23. onShow: function () { 25. }, 26. /\*\* 27. \* 生命周期函数--监听页面隐藏 28. \*/ 29. onHide: function () { 31. }, 32. /\*\* 33. \* 生命周期函数--监听页面卸载 34. \*/ 35. onUnload: function () { 37. }, 38. /\*\* 39. \* 页面相关事件处理函数--监听用户下拉动作 40. \*/ 41. onPullDownRefresh: function () { 43. }, 44. /\*\* 45. \* 页面上拉触底事件的处理函数 46. \*/ 47. onReachBottom: function () { 49. }, 50. /\*\* 51. \* 用户点击右上角分享 52. \*/ 53. onShareAppMessage: function () { 55. } 56. }) |

## 2.4 渲染页面

### 2.4.1 数据渲染原理

在小程序中，视图与脚本文件的渲染是在不同的线程中

视图，在webView中渲染

js脚本，是另起一个线程

### 2.4.2数据驱动

小程序实现了数据驱动，当我们定义什么数据，在页面中就渲染什么数据

定义数据的方式：

1 在js脚本文件中定

通过data属性定义数据

2 在视图模板中通过插值语法 渲染数据

### 2.4.3插值语法

插值语法： {{数据}}

小程序提供了一个伪js环境，只能使用一些简单的表达式

例如：加减乘除运算，三元运算符，对象的点语法获取属性，数组的中括号语法获取成员等

对于一些复杂的表达式，小程序不支持，例如使用方法等。

demo:

|  |
| --- |
| 1. <view> 2. <view>{{num}}</view> 3. <view>{{num + 100 \* 2}}</view> 4. <view>{{obj.msg}}</view> 5. <view>{{obj.msg.toUpperCase() + 'hello'}}</view> 6. </view> |

### 2.4.4更新数据

在组件中提供了setData方法用于更新数据

参数是一个对象

key: 要更新的属性名称

属性名称可以是直接属性，也可以是间接属性

例如： let obj = { a: { b: { c: 123 } } }

此时a是obj的直接属性， obj.

而b、c 是obj的间接属性： obj.a.b obj.a.b.c

更新a 此时key是a

更新b 此时key就是a.b

更新c 此时key就是a.b.c

value: 表示新的数据值

工作中，尽量只更新要更新的数据

demo:

|  |
| --- |
| 1. setTimeout(() => { 2. this.setData({ 3. num: 200 4. }) 5. }, 2000) |

### 2.4.5 数据丢失

data属性中存储的都是定义的数据，如果直接更改data中的数据

此时，数据更新了，但是视图没有更新，就称之为数据丢失

为了避免数据丢失，要通过this.setData更新数据

demo:

|  |
| --- |
| 1. this.data.num = 500; 不要通过这种方式修改数据 |

## 2.5 wxml

wxml是模仿html创建的一种文件格式

html是可以被浏览器识别的

wxml是小程序封装之后的文件，内部称之为组件

所以，wxml不适用浏览器环境，例如一些web端的框架 ： jquery这些框架不适用与操作wxml中的组件

### 2.5.1 属性

html中的元素可以设置属性

在wxml中也可以设置属性，一共分为两类

第一类，可以给所有的组件设置的属性，称为共有属性

第二类，可以给特有的组件设置的属性，称为特有属性

共有属性有：

id: 设置组件的id

class: 用于设置类名

style： 设置延时

hidden: 隐藏组件

data-name: 自定义数据

bind\*|catch： 绑定事件

|  |
| --- |
| 1. <view class="demo">hello world</view> 2. <view id="demo">爱创课堂</view> 3. <view style="font-size: 30px; color: orange;">你好，世界</view> 4. <view hidden>你好，世界</view> 5. <view data-ickt="aaa">你好，明天</view> |

### 2.5.2事件

在小程序中事件类型可以分为两类：

1 可以为所有的组件绑定的事件， 称为共有事件

共有事件大约有： touchstart、touchend、touchmove、touchcancel、tap、longtap(350ms)

2 可以为特定的组件绑定的事件，称为特有事件

事件按照冒泡与非冒泡事件，又分为两种

冒泡：事件执行的时候，从子组件传递到父组件之间的过程

第一类 冒泡事件

touchstart： 触摸开始

touchend： 触摸结束

touchmove： 触摸移动

touchcancel： 停止触摸

tap： 轻拍

longtap： 长按

longpress: 代替longtap

animationstart: 动画开始

animationend: 动画结构

transitionend: 过度完成

animationiteration: 循环事件中，每一次循环完毕执行的事件  
 touchforcechange： 强制touch

第二类是非冒泡事件

在小程序中可以通过bind和catch为组件绑定事件

bind绑定的事件是冒泡的

catch绑定的事件是不会冒泡

在视图文件中定义的事件，要在脚本文件中定义出来

### 2.5.3 事件对象

changedTouches: 手指离开时候的相关信息

currentTarget： 绑定事件的元素

detail: 标记手指的位置

target: 触发事件的对象

dataset 可以获取自定义的数据

touches: 指示手指列表相关信息

clinetX: 手指位于视口中的x位置

clientY: 手指位于视口中的y位置

type: 事件类型

|  |
| --- |
|  |

## 2.6 组件

### 2.6.1 view

view组件好比是html中的div元素， 容器组件，可以独占一行

有四个特有属性：

hover-class: 用于添加类

hover-start-time: 延迟添加类

hover-stay-tiem: 定义类停留的时间

hover-stop-propation: 组织冒泡

|  |
| --- |
| 1. <view data-root="root" bindtap="rootTap"> 2. root 3. <view data-root="parent" bindtap="parentTap" hover-class="demo" hover-start-time="2000" hover-stay-time="2000"> 4. parent 5. <view data-root="child" bindtap="childTap" hover-stop-propagation="true">child</view> 6. </view> 7. </view> |

### 2.6.2 text

text组件，用于定义文本，不会独占一行

space 定义空格模块 如果没有定义 默认是空白折叠

emsp: 渲染全角空格

ensp: 渲染中文字符大小的一般

nbsp: 渲染英文半角空格

selectable： 长按时候文本是否选中

decode: 是否解码

|  |
| --- |
| 1. <text space="nbsp">hello 你好，明天</text> 2. <view> 3. <text space="ensp">hello 你好，明天</text> 4. </view> 5. <view> 6. <text space="emsp" selectable>hello 你好，明天</text> 7. </view> 8. <view> 9. <text space="emsp" decode>hello &lt; 你好，明天</text> 10. </view> |

### 2.6.3 rich-text

rich-text对text组件的拓展

用于定义富文本组件，用于渲染html（有限的）

nodes属性定义渲染内容，有三种方式：

第一种方式： 可以渲染html标签(受到限制的)

第二种方式： 可以渲染node节点

name: 节点的名称

arrts: 定义节点的属性

children: 定义子节点

type: 定义类型

text: 定义内容

第三种方式：可以渲染字符实体

|  |
| --- |
| 1. <rich-text nodes="{{html}}"></rich-text> 2. <rich-text nodes="{{node}}"></rich-text> 3. <rich-text nodes="{{str}}"></rich-text> |

js:

|  |
| --- |
| 1. data: { 2. html: '<div style="color: red; font-size: 20px;">hello</div>', 3. node: [ 4. { 5. name: 'div', 6. attrs: { 7. id: 'box', 8. style: 'color: blue; font-size: 25px;' 9. }, 10. children: [ 11. { 12. type: 'text', 13. text: 'hello world' 14. } 15. ] 16. } 17. ], 18. str: '爱创课堂 &gt;' 19. }, |

### 2.6.4 image

image组件类似于html中的img标签

通过src引入图片，可以是本地的可以是线上（支持https）

mode属性： 定义裁剪模式

top: 渲染图片顶部

bottom: 渲染图片的底部

left: 渲染图片的左边部分

right: 渲染图片的右边部分

center: 渲染中间部分

这些属性可以互相组合： 例如 let top

scaleToFill: 缩放模式（可能会拉伸），填满容器

aspectFill: 等比缩放，会显示整体图片，其余部分剪切

aspectFit: 等比缩放，会显示整张图片，其余部分留白

widthFix: 等比缩放，保证宽度填满，剪切的是容器

|  |
| --- |
| 1. <image class="demo" src="https://www.icketang.com/static/img/banner\_news.jpg"></image> 2. <image class="demo" mode="top" src="https://www.icketang.com/static/img/banner\_news.jpg"></image> 3. <image class="demo" mode="right" src="https://www.icketang.com/static/img/banner\_news.jpg"></image> 4. <image class="demo" mode="left" src="https://www.icketang.com/static/img/banner\_news.jpg"></image> 5. <image class="demo" mode="bottom" src="https://www.icketang.com/static/img/banner\_news.jpg"></image> 6. <image class="demo" mode="center" src="https://www.icketang.com/static/img/banner\_news.jpg"></image> --> 7. <image class="demo" mode="scaleToFill" src="https://www.icketang.com/static/img/banner\_news.jpg"></image> 8. <image class="demo" mode="aspectFill" src="https://www.icketang.com/static/img/banner\_news.jpg"></image> 9. <image class="demo" mode="aspectFit" src="https://www.icketang.com/static/img/banner\_news.jpg"></image> 10. <image class="demo" mode="widthFix" src="https://www.icketang.com/static/img/banner\_news.jpg"></image> |

### 2.6.5 icon

icon组件用于定义字体图片

图片的资源可以占用空间比较大，所以小程序提供了字体图标，代理引入图片

type: 定义类型

success, success\_no\_circle, info, warn, waiting, cancel, download, search, clear

size: 定义图片大小

color: 定义图片的颜色

|  |
| --- |
| 1. <icon type="warn" size="40"></icon> 2. <icon type="success" size="40" color="gold"></icon> 3. <icon type="info" size="40" color="gold"></icon> 4. <icon type="waiting" size="40" color="gold"></icon> 5. <icon type="cancel" size="40" color="gold"></icon> |

### 2.6.6 canvas

在小程序中提供了canvas组件用于定义canvas画布，通过canvas-id定义canvas的属性名称

canvas画布默认的宽高350px 和 150px

我们可以自定义宽高

在js中，通过wx.createCanvasContenxt来获取ctx对象

获取之后该对象中提供了绘制画布的方法

|  |
| --- |
| 1. <canvas canvas-id="myCanvas" style="width: 320px; height: 300px; border: 1px solid #ccc;" ></canvas> |

### 2.6.7 map

map组件用于定义地图组件

longitude: 设置经度

latitude: 设置纬度

|  |
| --- |
| 1. <map longitude="116.376568" latitude="40.10133"></map> |

### 2.6.8 多媒体组件

video组件 用于定义多媒体组件

通过src 引入视频

camera 开启摄像头

divice-position: 定义前置或者后置摄像头

live-player: 直播组件

live-pusher: 推送组件

|  |
| --- |
| 1. <video src="https://www.icketang.com/static/video/shipin2.mp4" controls></video> 2. <camera style="width: 200px; height: 200px;" divice-postion="front"></camera> 3. <live-player></live-player> 4. <live-pusher></live-pusher> |

## 2.7 指令

指令是一种特殊的属性，用于实现某种功能

在小程序中通过wx: 定义指令

### 2.7.1 条件指令

wx:if=”statements”

如果表达式中出现了变量，要使用插值语法

是真正的创建与删除，不是控制元素的样式

demo:

|  |
| --- |
| 1. <view bindtap="toggle">切换显隐</view> 2. <text wx:if="{{isShow}}">pages/direct/direct.wxml</text> |

### 2.7.2 循环指令

语法： wx:for=”data”

data表示js中的数据，例如数组、数字

一定要设置wx:key

保证属性值具有唯一性，例如： id ,或者设置this

自定义成员变量名称与索引值：

wx:for-item 自定义成员变量名称

wx:for-index 自定义成员索引值名称

|  |
| --- |
| 1. <!-- 默认成员的名称是item 默认成员的索引值是index --> 2. <view wx:for="{{colors}}" wx:key="this">{{item}}---{{index}}</view> 3. <!-- 自定义成员名称与索引值 --> 4. <view wx:for="{{colors}}" wx:key="this" wx:for-item="ickt" wx:for-index="i">{{ickt}}---{{i}}</view> |

block可以作为组件使用（添加样式，添加属性，添加指令），但是就是不会渲染在页面中（模板）

|  |
| --- |
| 1. <view wx:for="{{num}}" wx:key="this" wx:for-item="i"> 2. <block wx:for="{{num}}" wx:key="this" wx:for-item="j" wx:if="{{i >= j}}">|{{i + 1}} \* {{j + 1}} = {{(i + 1) \* (j + 1)}}|</block> 3. </view> |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |