3.1:**微信小程序**

#"微信web开发者工具"

#e:/software/wechat\_devtools\_1.02.1804251\_x64.exe

#下载：<https://developers.weixin.qq.com/miniprogram/dev/devtools/download.html?t=19042222>

3.2:微信公众平台

(1)微信订阅号--(网络媒体平台-文章-图片-音频-视频)

(2)微信小程序--小程序

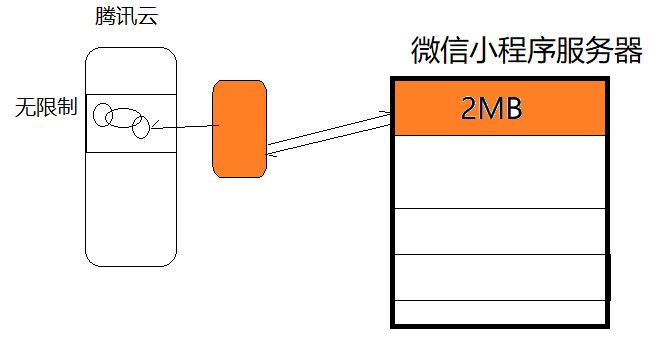
(3)微信小游戏

3.3:微信公众平台-小程序

作用:轻量级APP

小程序特点:微信小程序无需下载即可使用，用完即走

官方要求小程序项目非常小,第一版 1MB 2MB



3.4:微信公众平台-小程序-注册小程序

https://mp.weixin.qq.com/ 微信公众平台

订阅号 提前注册新邮箱

小程序 提前注册新邮箱

3.5:微信公众平台-小程序-工具-目录结构



3.6:微信公众平台-小程序-创建组件

组件=(html+css+js) vue(html+js+css)

3.7:微信公众平台-小程序-文件类型wxml

微信xml文件,严格xml文件不同html

严格(1): 所有标签要求有开始必须有结束

<text></text>

严格(2): 所有属性有单引号或双引号

<text id="btn1"></text>

**常见标签**

<view></view> 相当于 <div></div>

hover-class，string，指定按下去的样式类。当 hover-class="none" 时，没有点击态效果

<text></text> 相当于 <span><p>

属性:

selectable=” false” 文本内容是否可以选中复制

decode =”false” 文本内是否转码 &lt;

<image src="图片路径"></image> 相当于 <img>

-image 默认大小 320\*240

(1)src本地绝对路径 /pages/img/1.jpg

(2)src网络绝对路径 http://127.0.0.1:3000/1.jpg

-图片属性

(1)lazy-load 懒加载 <image lazy-load="true"></image>

(2)mode

aspectFill：等比例显示图片以短边为准

aspectFit：等比例显示图片以长边为准

widthFix：缩放模式，宽度不变，高度自动变化，保持原图宽高比不变

(3)binderror="eventhandle"，当错误发生时触发，event.detail = {errMsg}

bindload="eventhandle"，当图片载入完毕时触发，event.detail = {height, width}--当刷新不触发时在url上加随机数

<form></form> 相当于 <form>

<switch type="checkbox" checked="{{switch2Checked}}" bindchange="switch2Change"/>

checked：boolean，false ，是否选中

disabled：boolean，false ，是否禁用

type：string，样式，有效值：switch, checkbox

color：string，#04BE02，颜色，同 css 的 color

bindchange="eventhandle"，checked 改变时触发 change 事件，event.detail={ value}

<navigator></navigator> 相当于 <a>

target：string，self，在哪个目标上发生跳转，默认当前小程序(miniProgram--其它小程序)

url：string，当前小程序内的跳转链接

open-type：string，navigate，跳转方式，取值如下：

navigate，对应 wx.navigateTo 或 wx.navigateToMiniProgram 的功能

redirect，对应 wx.redirectTo 的功能

switchTab，对应 wx.switchTab 的功能

reLaunch，对应 wx.reLaunch 的功能

navigateBack，对应 wx.navigateBack 的功能

exit 退出小程序，target="miniProgram"时生效

<icon type="success" size="40" color="{{item}}"/>

type：string，icon的类型，有效值：success, success\_no\_circle, info, warn, waiting, cancel, download, search, clear

size：number/string，23，icon的大小

color：string，icon的颜色，同css的color

**轮播图组件：**

<swiper autoplay="{{true}}">

<swiper-item>

<image />

</swiper-item>

</swiper>

swiper属性:

(1)autoplay：[boolean] 默认false，是否自动切换

(2)interval：[number] 5000，自动切换时间间隔(毫秒)

(3)vertical：[boolean]，false，滑动方向是否纵向

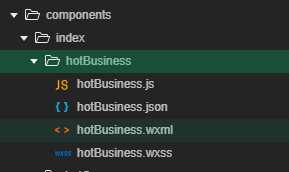
(4)circular：boolean，false，是否采用衔接滑动

(4)indicator-dots：boolean，false，是否显示面板指示点

indicator-color：rgba(0, 0, 0, .3)，指示点颜色

indicator-active-color：#000000，当前选中的指示点颜色

**3.8创建组件**

创建components文件 → 右键 → 创建目录文件 → 右键 → 创建组件

**组件的生命周期**

created：组件实例刚刚被创建好时，created生命周期被触发。此时，组件数据this.data就是在Component构造器中定义的数据data。

注意：此时还不能调用setData。通常情况下，这个生命周期只应该用于给组件this添加一些自定义属性字段。  
 attached：在组件完全初始化完毕、进入页面节点树后，attached生命周期被触发。此时，this.data已被初始化为组件的当前值。这个生命周期很有用，绝大多数初始化工作可以在这个时机进行。  
 detached：在组件离开页面节点树后，detached生命周期被触发。退出一个页面时，如果组件还在页面节点树中，则detached会被触发。  
**组件获取数据：**

Component({

properties: {

navIndex: {

type: Number, //指定接受类型

value: 1, //默认值

observer(newVal, oldVal) { //每次接收传入参数触发

let that=this

if (newVal===1){

wx.request({ }) //获取数据

}

}

}

},

data: {}, //组件初始化的数据

attached() {

console.log("热门案例来了")

}

})

3.1:微信公众平台-小程序-js

小程序每个组件对应一个js文件

Page() 当前组件对象

Page({data:{},onload:function(){...}})

小程序与其它程序js区别：

-ECMAScript

基本类型:number;string;boolean;null;undefined

引入:Object;Date;Math;ExpReg....

自定义:{}

-DOM/BOM 没有

-小程序有顶级对象 wx (类window)

参考文档

https://developers.weixin.qq.com/miniprogram/dev/api/

3.2:微信公众平台-小程序-js-显示数据和控制数据

-在js获取数据

var value = this.data.数据属性名;

示例:

data:{msg:"hello"}.

onload:function(){

console.log(this.data.msg)

}

-在js**修改**数据

this.setData({

数据属性名:新值,

list:res.data.data

});

-在模板中控制数据

(1)创建循环

<view wx:for="{{list}}" wx:key="index">

{{item.id}}：{{item.name}}

</view>

list数组

item数组中对象

index 下标

练习:

1:在data中创建数组 购物车列表

id,uid,count,price

2:在模板中显示数据内容

3:如果数组内容为空

"当前购物车空空，亲 赶快去购物"

练习:更新数组商品价 0.9

|  |
| --- |
| Page({  /\*页面的初始数据\*/  data: {  list:[  {id:1,uid:1,count:1,price:100.00},  {id:2,uid:1,count:2,price:200.00},  {id:3,uid:1,count:3,price:300.99}  ]  },  /\*生命周期函数--监听页面加载\*/  onLoad: function (options) {  var list=this.data.list;  for(var item of list){  item.price\*=0.9;  item.price=item.price.toFixed(2);  console.log(item.price)  }  this.setData({  list  })  }  }) |

2.1:小程序 --**数据与显示**

-如果数据需要在模板中显示 data:{}

-模板中 {{属性名}}

-如果数组循环标签

<view wx:for='{{list}}' wx:for-item='obj' wx:for-index='i' wx:key='i'>

<text>{{obj.lid}}</text>

<text>{{obj.name}}</text>

<text>{{obj.sal}}</text>

</view>

:默认循环中当前对象item当前下标 index

:修改默认值 wx:for-item="当前对象名称"

wx:for-index="当前下标名称"

**-if** 满足显示

<view wx:if="{{判断条件}}"></view>

判断条件:true 元素正常显示

判断条件:false 当前元素删除

**-if else**

<view wx:if="{{判断条件}}"></view>

<view wx:elif="{{判断条件}}"></view>

<view wx:else>..</view>

2.1:小程序 --组件**生命周期**(重点)

组件生命周期:一个组件从创建开始到销毁完整过程

-onLoad 组件创建成功己经加载到内容,触发一次

此次事件中二种操作

(1)发送ajax请求

(2)获取其它组件传递过来数据(id)——参数：options对象

-onReady 组件初始化(数据模板绑定)，渲染完成执行一次

-onShow 显示组件，执行(多次)

-onHide 隐藏组件，前台后切换(多次)

-onUnload 组件卸载(将组件从内存中移出)

跳转到其他组件(tabbar中的组件不能跳转)，会触发onUnload事件

wx.redirectTo({

url:"/pages/index/index"

})

life03(先卸载)-->跳转-->index

2.2:小程序 --**特殊事件**(重点)

-onPullDownRefresh 用户下拉刷新事件

#默认小程序禁止用户下拉操作

#修改默认行为

(1)全局范围允许所有组件用户都可以下拉操作

app.json

"window":{

"enablePullDownRefresh":true

}

(2)只允许当前组件用户可以下拉操作，在life03.json中配置：

{"enablePullDownRefresh":true}

-onReachBottom 用户上拉触底

-onPageScroll(opt) 用户滑动操作

#opt.scrollTop 页面垂直方向己经滚动距离

实例：onPageScroll:function(opt){

console.log(opt.scrollTop)

}

2.3:小程序 --**全局配置文件 app.json**

pages:[ ... ] 组件列表,默认显示第一个位置组件

windows:

"navigationBarBackgroundColor": "#fff",——顶部导航背景

以上颜色必须使用16进制格式,不能使用单词

red/blue 错误

#f00/#000 正确

"navigationBarTitleText": "商城", ——顶部导航文本

"navigationBarTextStyle": "black", ——顶部导航文本颜色

注意:文本颜色 white/black （两选一）

#以上配置项在当前组件中配置文件中也可使用

exam03.json

{

"navigationBarTitleText": "商城"

}

#配置文件有原则"就近原则"

#如果全局与局部的配置项有重复,局部优先

tabBar:

"list": [

{ "pagePath": "pages/home/home", //跳转到的网页

"text": "这里是首页", //导航栏显示的文字

"iconPath": "pages/img/home.png", //未点击页面显示的图片

"selectedIconPath": "pages/img/home-active.png" //已点击页面显示的图片

},

...

]

2.2:小程序--微信样式文件 wxss(重点)

小程序为css添加二个新功能

-新单位: **rpx** (r：响应式像素)

前端开发流程

(1)设计师ps创建手机小程序效果图-切片(图片) 物理像素

(2)前端--创建静态网页(html/css/js) 逻辑像素

物理像素=逻辑像素\*dpr

dpr iphone6 2

iphone6 plus 3

示例:物理像素 750=2\*375

解决方案:小程序rpx

物理像素 750 =逻辑像素750rpx

-@import “引入其它的样式文件”

2.3.2:小程序--功能:**九宫格**(弹性布局，宽度33.333%)

|  |
| --- |
| wxml中：  <view class="container">  <!-- 数据循环写轮播图-->  <view class='slider'>  <swiper autoplay='{{autoplay}}' interval='{{interval}}' vertical='{{vertical}}'>  <swiper-item wx:for='{{list}}' wx:key='index'>  <image src='{{item.img\_url}}' lazy-load='true' mode='aspectfill'></image>  </swiper-item>  </swiper>  </view>  <!-- 九宫格 -->  <view class='grid'>  <view class='grid-item' wx:for='{{navlist}}' wx:key='index'>  <image src='{{item.img\_url}}' lazy-load='true' mode='aspectfill'></image>  <text>{{item.title}}</text>  </view>  </view>  <!-- tabbar -->  </view> |
| wxss中：  /\* 轮播图 \*/  .slider,.slider swiper,.slider image{  height:390rpx;  width:100%;  }  /\* 九宫格 \*/  .grid{  display: flex;  flex-wrap: wrap;  }  .grid-item{  height:200rpx;  width:33.33333%;  box-sizing: border-box;  display: flex;  flex-direction: column;  justify-content: center;  align-items: center;  border-right: 1rpx solid #eee;  border-bottom: 1rpx solid #eee;  font-size: 14px;  }  .grid-item:nth-child(3n){  border-right:0;  }  .grid-item image{  width:80rpx;  height:80rpx;  margin-bottom:5px;  } |

2.4:小程序--功能:**事件**--浏览器原生

屏幕宽度 操作方式 事件

pc端项目: w>970px 键盘鼠标 点击/双击

移动端项目: w<970px; 手指 触碰

-开发使用组件库

vueui mint-ui mui 组件库针对移动端软件项目

vueui elementui 组件库针对pc端软件项目

-事件分类--触摸事件--通过手机原生BOM事件

(1)touchstart 当手指触碰屏幕时发事件不管几个手指

(2)touchmove 当手指在屏幕上滑动时连续触发

(3)touchend 当手指离开屏幕时触发

-原生移动事件使用灵活稍一些,可以第三方js库，简单完成以上操作：

zepto.js --相当于pc端jquery

tap 触碰一次

longtap 长按一次 300ms 以上

swipe 滑动

swipeLeft 左滑

swipeRight 右滑

swipeUp 上滑

swipeDown 下滑...

示例：$('#box2').longTap(function(){

$(this).html('我长按了一下')

})

2.5:小程序--功能:事件--常见事件(vscode-浏览器)

tap 手指触碰屏幕马上离开

longtap 手指触碰屏幕300ms离开(旧)

longpress 手指触碰屏幕300ms离开(新)

2.6:小程序--功能:事件--事件绑定

(1)支持事件冒泡

<view bindtap="事件处理函数">tap me=</view>

(2)不支持事件冒泡

<view catchtap="事件处理函数">tap me</view>

2.7:**事件--事件对象-获取元素数据/参数**

事件对象:当用户完成一个操作，浏览器创建事件对象(e)——事件坐标,触发事件元素,..

type:"tap" 触发事件名称 tap

target: 触发事件元素

timeStamp: 当组件打开后到触发事经过毫秒

<view bindtap="handle4" data-id="{{item.id}}"><view>

handle4:function(e){

e.target.dataset.id;

//将参数传递到另一个组件tap02

wx.redirectTo({

url: '/pages/tap02/tap02?id='+id,

})

}

//tap02接收参数，并且放在当前组件data中

onLoad: function (options) {

this.setData({

lid:options.id

})

},

双向绑定数据

|  |
| --- |
| ①不用点击按钮实现  //wxml  <input placeholder="输入关键字" bindinput="bindKeyInput"></input>  <text>{{ sugData }}</text>  //js  Page({  data: {  sugData:''  },  bindKeyInput: function(e){  this.setData({  sugData:e.detail.value;  })  },  })  ②form表单中通过点击按钮实现数据绑定  //wxml  <view class='section'>  <form bindsubmit='searchBox'>  <input type='text' name='keyword'></input>  <button form-type='submit'>提交</button>  </form>  <text>{{ sugData }}</text>  </view>  //js  Page({  data: {  sugData:''  },  searchBox:function(e){  this.setData({  sugData:e.detail.value.keyword  })  }  }) |

2.1:小程序--**组件之间跳转**--两类(标签/程序)

跳转:关闭或者保留当前组件，并且打开新组件

方式一:标签

<navigator url="组件路径">文字</navigator>

#建议使用组件绝对路径

示例：<navigator url="/pages/index/index">..</navigtor>

#如果组件己经定义在tabbar中不能通过标签方式跳转

#标签跳转方式:保留跳转

方式二:通过js编程方式

(1) wx.redirectTo({url:"组件路径"})

#关闭(卸载)当前组件，并且打开新组件

#不能跳转tabbar中定义组件

(2) wx.navigateTo({url:"组件路径"})--(最常见)

#保留当前组件，并且打开新组件

#不能跳转tabbar中定义组件

wx.navigateBack({

delta: 2 //返回的页面数，如果delta大于现有页面数，则返回到首页

})

(3) wx.reLaunch({url:"组件路径"})

#关闭当前组件，并且打开新组件

#能跳转tabBar 中定义组件

(4)wx.reLaunch({

url: 'test?id=1'

})

#关闭所有页面，打开到应用内的某个页面

小结:在使跳转方式依据顺序

-如果有标签，首先标签 <navigator></navigator>

-如果没有标签，使用保留跳转 wx.navigateTo({ })

2.2:小程序--**ajax**

wx.request({

url:请求服务器程序地址,

method:"POST", //请求方式 GET POST(默认GET)

data:发送请求数据给服务器端程序,

success:(res=>{ 请求成功回调函数,res服务器返回结果

...

})

fail:(res=>{ 请求失败 })

complete:(res=>{ 无论请求成功或者失败都会回调函数 })

});

#注意事项:

-小程序对发送ajax有要求,服务器域名合法(互联中心注册/交费),而且服务器协议使用https (加密隧道)不支持http协议。开发阶段：禁止以上验证。

点击小程序开发工具->设置菜单->项目设置

[\* ]不校验合法域名，不验证HTTPS 证书

|  |  |  |
| --- | --- | --- |
| 请求方式 | GET | POST |
| 语义 | 客户端获取服务器上数据  商品列表,用户列表.. | 客户端向服务器发送数据  注册用户  添加商品 |
| 发送数据量 | 最低标准 1kb | 没限制 |
| 发送数据方式 | 请求url地址添加  http://127.0.0.1:3000/index?id=1&pwd=213 | 请求主体中 |
| 安全 | 非常不安全 | 不安全 |

2.3:小程序--**美食列表**(重点)

(1)分析表创建表(xz\_shoplist)

id INT

img\_url VARCHAR(255)

dname VARCHAR(255)

tel VARCHAR(25)

addr VARCHAR(255)

ctime VARCHAR(255)

point INT

(2)分析服务器端程序 app.js

#服务器端 session丢失

#(1)如果你添加路由session丢失,服务器端node.js造成

server.use(session({

secret:"128位字符串", //配置密钥

resave:false, //每次请求是否更新数据

saveUninitialized:true //保存初始化数据

}));

解决问题:resave:true

#(2)名称冲突

登录保存session: req.session.id = 3;

将商品添加购物车 var id = req.session.id

原因:session对象自有属性id 128位加密码

解决:req.session.uid; var uid = req.session.uid

#(3)拼写错误

|  |
| --- |
| 数据库乱码:  原因:编码不统一  (1)创建库一定要指定编码  CREATE DATABASE xz CHARSET=UTF8;  (2)快速导入大量sql语句  mysql -uroot -p < db.sql |

(3)创建组件shoplist 美食组件

-当组件创成功获取美食列表并且显示 第一页

-用户向上滑动显示更多的美食 下一页

**day05**

2:美食列表

2.1:小程序--加载数据时如果数据还未显示,则显示加载动画效果

加载动画作用:提升用户体验

-创建加载动画组件

wx.showLoading({title:"正在加载数据..."});

-卸载加载动画组件

wx.hideLoading();

#学习阶段模拟网络繁忙效果

wx.showLoading({title:"正在加载数据..."});

setTimeout(function(){

wx.hideLoading();

},1000);

#实际项目不能模拟

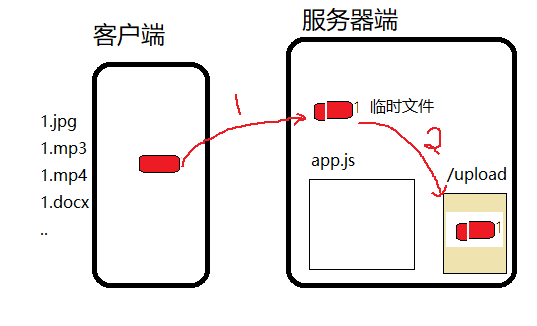
|  |
| --- |
| 项目中引用  onLoad: function (options) {  // 加载页面获取数据  this.loadMore();    },  loadMore:function(){  var pn = this.data.pno+1;  var ps = this.data.pageSize;  // 1.创建加载动画组件（请求数据前）  wx.showLoading({  title: '正在加载...',  })  // ajax请求数据  wx.request({  url: 'http://127.0.0.1:3000/shoplist?pno=' + pn + '&pageSize=' + ps,  method: 'GET',  success: (res) => {  console.log(res.data.data);  var rows = this.data.list.concat(res.data.data)  this.setData({  list: rows,  // 修改当前的页码  pno: pn  })  // 2.隐藏加载动画组件（数据请求成功后）  wx.hideLoading();  },  fail: function (res) { },  complete: function (res) { },  })  } |

3:上传文件(重点)

在软件项目中将客户端某一些文件上传到服务器指定文件夹下,

这是一种常见操作.

示例:上传头像;微信发文件;发送邮件(附件);docx;....



3.2:小程序-上传文件:注意事项

(1)用户可以上传任意类型文件 1.jpg 1.mp3 1.docx 1.mp4

可以限制用户上传文件类型:示例:只允许上传图片

(2)用户默认情况可以上传任意大小文件

可以限制用户上传文件大小:2MB

(3)用户上传文件名一定要重新修改后再保存到服务器中

防止名称冲突

用户a[11.jpg]-->上传服务器--[/upload]/12928779192.jpg

用户b[11.jpg] -->上传服务器--[/upload]/23823823839.jpg

新名字:时间戳+随机数+文件后缀

(4)上传文件操作=将临时文件 移动 upload

3.3:小程序注意事项-工作中常见错误

(1)目录不正确

1.1:node.js 创建目录只能帮创建一层

upload/1.jpg 程序自动创建upload

upload/2019/1.jpg 出错

1.2:相对目录容易出错

var desFile = "upload/1.jpg";

var desFile = \_\_dirname+"/upload/1.jpg" #\_\_dirname 绝对路径

(2)目录中创建文件权限不足

公司服务器不是window用的linux 服务器(90%>),

2.1:严格区大小写字母:[表小写字母+大写]

-程序文件名/表名/库名/列名/ 严格区分大小写

2.2:目录不允许 403报错——权限不足

-找公司运维工程师 /upload 权限创建文件权限

3.4:小程序-上传文件 --服务器程序

-下载并且安装第三方模块 multer

npm i multer //node\_modules/multer/..

-加载第三方模块并且创建成功

const multer = require("multer"); 加载模块

var upload = multer({dest:"upload/"});

创建对象并且指定临时文件目录 upload

-处理上传请求

server.post("/uploadFile",upload.single("mypic"),(req,res)=>{

})

#post 用户上传文件服务器一定要用post接收

#/uploadFile 处理上传文件程序路径

# upload.single("mypic"); 一次上传一个文件

# mypic 上传文件参数名称 "mypic":1.jpg

# req 请求对象

#req.file 保存上传文件信息

req.file.size; 上传文件大小 1000 字节byte[b]

req.file.originalname 上传文件原文件名 11.jpg

req.file.mimetype 上传文件类型

image/jpg image/png text/css application/javascript

req.file.path 临时文件路径

3.5:小程序-上传文件 --客户端html表单[pc/vue/...]

<form action="http://127.0.0.1:3000/uploadFile" method="POST" enctype="multipart/form-data">

<input type="file" name="mypic" />

<input type="submit" value="上传文件" />

</form>

#action 请求服务器程序地址

#method 请求方式POST

#enctype multipart/form-data

默认表单提交:将表单中数据转换字符串发送服务器程序，

但是如果上传文件[图片/mp3/mp4]不能转换字符串，发送失败。

解决方案将:将表单数据转换二进制格式，任何数据文件都可以传输服务器端。

如果将默认数据格式转换二进制在表单中添加一个属性：

enctype="multipart/form-data"

#file:mypic file表单type选择一个文件mypic文件参数名称.

3.5:小程序-上传文件 --客户端 小程序-图片

-在当前手机中选中图片(1)

\*手机拍照

\*相册中选择

wx.chooseImage({

#count:1 选择图片张图:最多9张

#sourceType:["camera","album"] 指定图片来源

#success:fn(res)=>{ 选择图片成功

res.tempFilePaths[0] 数组:保存选中图片列表

}

})

-将选中图片上传服务器

\*将选中图片上传服务器

wx.uploadFile({

url: 服务器程序路径

filePath:选中图片路径res.tempFilePaths[0]

name: 上传图片参数名称 mypic

header:"Content-Type":"multipart/form-data"

success:fn 上传成功

})

|  |
| --- |
| app.js中：  // 九.小程序上传图片  // 9.1 加载模块multer  const multer=require('multer')  const fs=require('fs');  // 创建upload对象并且指上传临时文件目录  var upload=multer({dest:'upload/'})  // 接收post请求 /uploadFile  app.post('/uploadFile',upload.single('mypic'),(req,res)=>{  // 判断文件大小，不能超过2MB  var size=req.file.size/1000/1000;  if(size>2){  res.send({code:-1,msg:'上传文件过大，请重新选择'});  return;  }  // 判断文件必须图片  var type=req.file.mimetype;  var idx=type.indexOf('image')  if(idx==-1){  res.send({code:-2,msg:'上传文件类型不正确'});  return;  }  // 获取原文件  var src=req.file.originalname;  // 创建文件名[?]  var ft=new Date().getTime();  var fr=Math.floor(Math.random()\*9999);  var i3=src.lastIndexOf('.');  var suff=src.slice(i3)  var des=\_\_dirname+'/upload/'+ft+fr+suff;  // 将临时文件移动到upload目录下  fs.renameSync(req.file.path,des);  // 返回成功信息  res.send({code:1,msg:'上传成功'})  }) |
| index.js中：  myUpload:function(){  // 从相册选中图片  wx.chooseImage({  count:1, //选中图片的数量：最多9张  sizeType: [],  sourceType: ['camera','album'], //图片来源  success: function(res) {  // 获取选中图片  var src=res.tempFilePaths[0] //数组  console.log(src)  // 将选中图片上传服务器程序  wx.uploadFile({  url: 'http://127.0.0.1:3000/uploadFile',  filePath:src,  name: 'mypic',  header:{  'Content-Type':'multipart/form-data'  },  success:(res)=>{  console.log(res)  }  })  },  fail: function(res) {},  complete: function(res) {},  })  } |

3.6:小程序-上传文件--学子商城

app.js

小程序创建组件 avatar 上传头像

1:最大图片不能超过2MB

req.file.size 上传文件大小[单位:字节 byte]

1024byte = 1kb

1024kb = 1Mb

1024Mb = 1Gb

1024Gb = 1Tb

示例:不超过2Mb

req.file.size/1000/1000 > 2

2:必须图片

req.file.mimetype.indexOf("image")==-1 不是图片

3:在小程序创建组件 avatar

**day06**

2.1:小程序--全局存储

**存储方式一:局部存储**

data:{} data中存储数据在当前组件任意方法调用

但其它组件不能共享

获取局部存储: this.data.属性;

修改局部存储: this.setData({属性:值})

**存储方式二:全局存储**

问题:哪些数据都要全局存储:ajax请求服务器路径

http://127.0.0.1:3000/

解决问题:全局存储

2.2:小程序--全局存储-操作

(1)创建全局共享数据

app.js

App({

globalData:{属性名:属性值,属性名1:属性值,...}

})

(2)获取全局共享数据

\*加载全局对象

const app = getApp();

\*获取全局共享数据

var 数据 = app.globalData.属性名;

(3)修改全局共享数据

const app = getApp();

app.globalData.属性名 = 新值;

示例:

1:创建全局共享数据

app.js

App({

globalData:{baseUrl:"http://127.0.0.1:3000/"}

})

2:其它组件中获取

组件.js

const app = getApp();

var url = app.globalData.baseUrl;

this.setData({baseUrl:url})

组件.wxml

<image src="{{baseUrl}}img/banner1.png" />

2.3:小程序--**背景音乐**

小程序后台音乐播放器(背景音乐)

小程序要求同一时间段只能播放一个背景音乐

当其它小程序占用后台音乐播放器,当前音乐停止退出

-播放背景音乐

wx.playBackgroundAudio({})

dataUrl: 音乐列表

title: 音乐标题

#小程序支持音频格式 mp3/wav/aac/m4a/..

#音乐列表中文件一定保存服务器上

-暂停播放背景音乐

wx.pauseBackgroundAudio()

-停止播放背景音乐

wx.stopBackgroundAudio();

* 常见错误

(1)闪退:音频文件不存在

练习:

\*-创建新组件playAudio

\*-在模板中创建图片标签

public/play.png pause.png

-点击图片完成背景播放和暂停

-点击图片完成图片之间切换

|  |
| --- |
| backAudio.js中：  data: {  baseUrl:'',  isPlaying:false, //当前音乐播放状态  imgUrl:''  },  /\*\*  \* 生命周期函数--监听页面加载  \*/  onLoad: function (options) {  var url = app.globalData.baseUrl;  this.setData({  baseUrl:url,  imgUrl: url+'/img/play.png'  })  },  myplay:function(){  // 获取背景音乐的播放状态  var isp=this.data.isPlaying;  // 播放背景音乐  if(!isp){ //当前正暂停，就需要播放  wx.playBackgroundAudio({  dataUrl: this.data.baseUrl + 'bg.mp3',  })  // 修改播放的状态  this.setData({  isPlaying:true,  imgUrl: this.data.baseUrl + '/img/pause.png'  })  }else{  // 暂停背景音乐  wx.pauseBackgroundAudio();  this.setData({  isPlaying: false,  imgUrl: this.data.baseUrl + '/img/play.png'  })  }    }, |
| backAudio.wexml中：  <view class='container'>  <!-- ①创建变量imgUrl改变图片-->  <!-- <image bindtap='myplay' src='{{imgUrl}}' lazy-load='true' mode='aspectFill' class='bplay'></image> -->  <!-- ②三目运算符显示图片 -->  <image bindtap='myplay' src="{{isPlaying ? 'http://127.0.0.1:3000/img/pause.png' : 'http://127.0.0.1:3000/img/play.png'}}" lazy-load='true' mode='aspectFill' class='bplay'></image>  </view> |
| app.js中：  app({  globalData: {  userInfo: null,  // 基础的请求路径  baseUrl:'http://127.0.0.1:3000/',  }  }) |

2.4:小程序--**搜索商品**

分析:当用户输入某种商品部分名称,我们服务器

返回包含此部分名称商品列表

-数据库哪张表,查询哪几列

xz\_laptop:[id;title;price]

-app.js

(1)参数 pno=1 pageSize=4

(2)sql SELECT lid,title,price FROM xz\_laptop WHERE title LIKE concat('%',?,'%');

(3)json

#concat 拼接字符串 concat('%',?,'%') -> '%?%'

#%通配符表示0个或多个字符

2.6:微信公众号--订阅号--聊天机器人

-建议在微信公众平台,注册订阅号

-提前注册新邮箱(不要使用小程序邮箱)

微信:腾讯推出交流工具

公众平台:[订阅号;小程序;小游戏]

订阅号:2013推出媒体平台[网络报纸,网络杂志,电台,电视台]

2017左右: 100 top [优质广告]

胡说:[历史;明星;国际...]

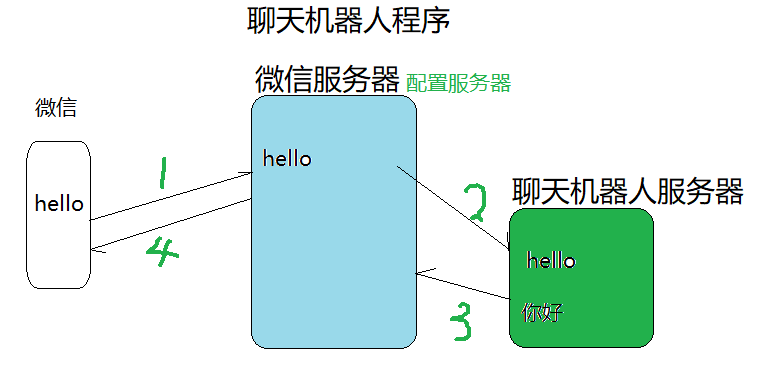
精致:[情感;知识;旅游;美食..]

专业:[前端大全;医生;..]

一天可以发一篇文章[文字;图片;音频;视频..]

2.7:微信公众号--订阅号使用

2.8:微信公众号--订阅号-**测试帐户开发**-聊天机器人



-进入公众平台测试帐户-开发-开发者工具

-测试帐户

-记录以下信息

appid[wxa56801014e697901]

接口配置信息

[你程序运行服务器域名]URL

[weixin]Token #相当于微信与程序之间暗号

#weixin

JS接口安全域名

[你程序运行服务器域名]域名 #域名

测试号二维码:

用你手机或者其它手机扫描二维码

2.9:微信公众号--订阅号-测试帐户开发-聊天机器人-操作流程

(1)合法域名

解决:内网穿透工具——ngrok.exe

工具:生成合法域名让我们使用 8小时与本地程序绑定

3000端口

在命令行下

ngrok.exe 本地服务器协议 端口号

示例: .\ngrok.exe http 3000

#将本地3000端口程序与合法域名绑定

http://64ebeba8.ngrok.io

#以上窗口不要关闭,关闭窗口域名失效

(2)开发

-下载二个模块:express;wechat[第三方模块]

#wechat 基于node.js 微信聊天第三方模块

#文字聊天

-准备基本数据

(1)appid[wxa56801014e697901]

(2)令牌[weixin]

-服务器代码

-加载模块

-创建配置对象

var config = {

appid:"wxa56801014e697901",

token:"weixin"

}

-创建express服务器对象并且监听3000

-接收post请求 /

微信用户将聊天内容发送给服务器,请求方式post

请求地址 /

server.post("/",wechat(config,(req,res)=>{

var msg = req.weixin.Content;//msg微信用户发送消息

console.log(msg);

res.reply(stringMsg);//你将消息返回微信用户

}))

-接收 get 请求 /

微信服务器测试你服务器发送测试

server.get("/",wechat(config,(req,res)=>{

var msg = req.weixin.Content;

res.reply(stringMsg)

}));

#启动服务器 node 05\_wechar.js

(3)进入微信测试帐户配置服务器上，开发者工具-->微信平台测试账号

-->打开测试帐户填url(http://64ebeba8.ngrok.io) 令牌(weixin) 域名

(4)扫描二维码聊天

练习:

分析:将用户聊天内容保存数据库表

用户发送消息,去数据查找是否有匹配记录

将数据库回复消息返回微信用户

-xz/xz\_chat

id form replay

1 hi 你好

2 88 再见

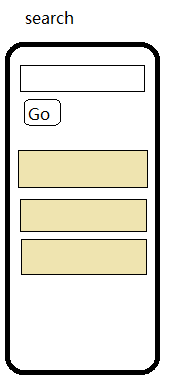
3 hihi 你想干啥

..

作业2:聊天机器使用数据库版本

|  |
| --- |
| app.js  // 接收post  server.post('/',wechat(config,(req,res)=>{  // 接收微信用户发送信息  var msg=req.weixin.Content;  console.log(msg);  // 创建sql查询数据库是否匹配记录  var sql='SELECT id,repley FROM xz\_chart WHERE form LIKE concat("%",?,"%")'  pool.query(sql,[msg],(err,result)=>{  if(err) throw err;  // 没有匹配记录 没听懂  if(result.length==0){  res.reply('你娃娃说啥？')  }else{ // 有匹配记录 返回字符串给微信用户  res.reply(result[0].repley)  }  })  })) |

作业1:商品搜索组件



|  |
| --- |
| app.js中：  // 十. 商品搜索  app.get('/search',(req,res)=>{  var pno=req.query.pno;  var pageSize=req.query.pageSize;  // 查询的关键字  var key=req.query.key;  if(!pno){  pno=1  }  if(!pageSize){  pageSize=6  }  var ps=parseInt(pageSize);  var offset=(pno-1)\*ps;  // concat为字符串拼接 -->'%?%'  var sql='SELECT lid,title,price FROM xz\_laptop WHERE title LIKE concat("%",?,"%") LIMIT ?,?';  pool.query(sql,[key,offset,ps],(err,result)=>{  if(err) throw err;  res.send({code:1,data:result});  })  }) |
| search.wxjs中：  data: {  list:[]  },  // 商品搜索功能  searchKey:function(e){  var key = e.detail.value.key;  var url=app.globalData.baseUrl+'search';  wx.request({  url,  data:{  key  },  success: (res) =>{  console.log(res.data.data)  this.setData({  list: res.data.data  })  }  })  } |
| search.wxml中：  <view class='card'>  <form bindsubmit='searchKey'>  <view class='cell-item'>  <input type='text' placeholder='请输入商品名称' name='key'></input>  </view>  <view class='cell-item'>  <button form-type='submit'>GO</button>  </view>  </form>  </view> |