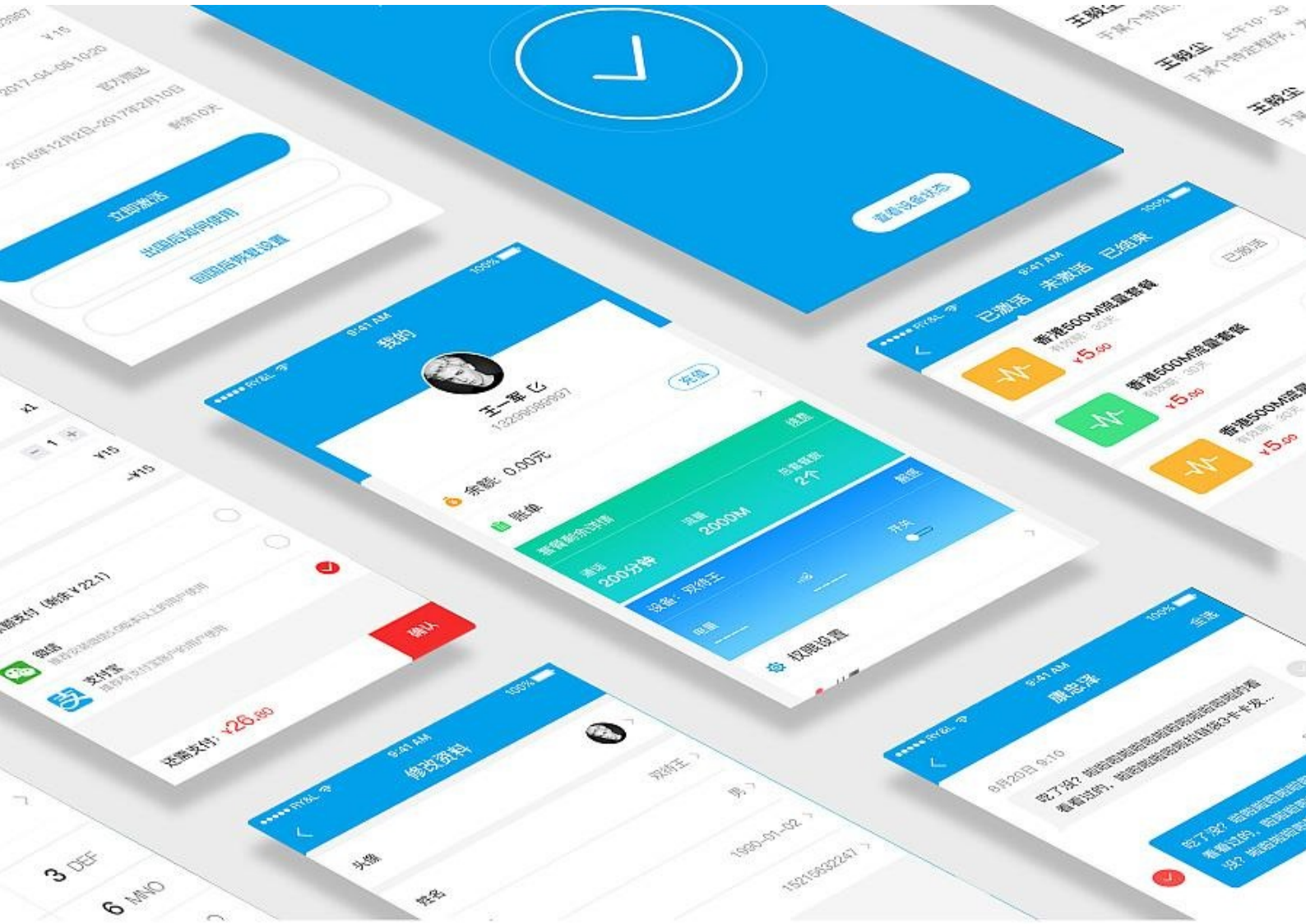


THANK YOU FOR WATCHING

2017-04-20

目　录

第1讲：创建项目、部署 VUE 、入口页面布局

uni-app 介绍

uni-app 是一个使用 Vue.js 开发跨平台应用的前端框架。

开发者通过编写 Vue.js 代码，uni-app 将其编译到iOS、Android、微信小程序等多个平台，保证其正确运行并达到优秀体验。

uni-app 继承自 Vue.js，提供了完整的 Vue.js 开发体验。

uni-app 组件规范和扩展api与微信小程序基本相同。

有一定 Vue.js 和微信小程序开发经验的开发者可快速上手 uni-app ，开发出兼容多端的应用。

uni-app提供了条件编译优化，可以优雅的为某平台写个性化代码、调用专有能力而不影响其他平台。

uni-app打包到App时仍然使用了5+引擎，5+的所有能力都可以在uni-app中可以使用。在App端运行性能和微信小程序基本相同。

对于技术人员而言：不用学那么多的平台开发技术、研究那么多前端框架，学会基于vue的uni-app就够了。

对于公司而言：更低成本，覆盖更多用户，uni-app是高效利器。

uni-app多端演示

为方便开发者体验uni-app的组件、接口、模板，DCloud发布了Hello uni-app演示程序（代码已开源，详见Github），Hello uni-app实现了一套代码，同时发布到iOS、Android、微信小程序。

你可以使用手机扫描下方App码下载iOS、Android原生安装包，也可以使用微信扫描小程序码，体验uni-app的小程序版本：

App码

小程序码

相关链接及教程

1、vue 教程（很全面）：https://ke.qq.com/course/248507?tuin=4f8da6

2、uni-app 官网：http://uniapp.dcloud.io/

第2讲,快速开始第一个项目

搭建环境

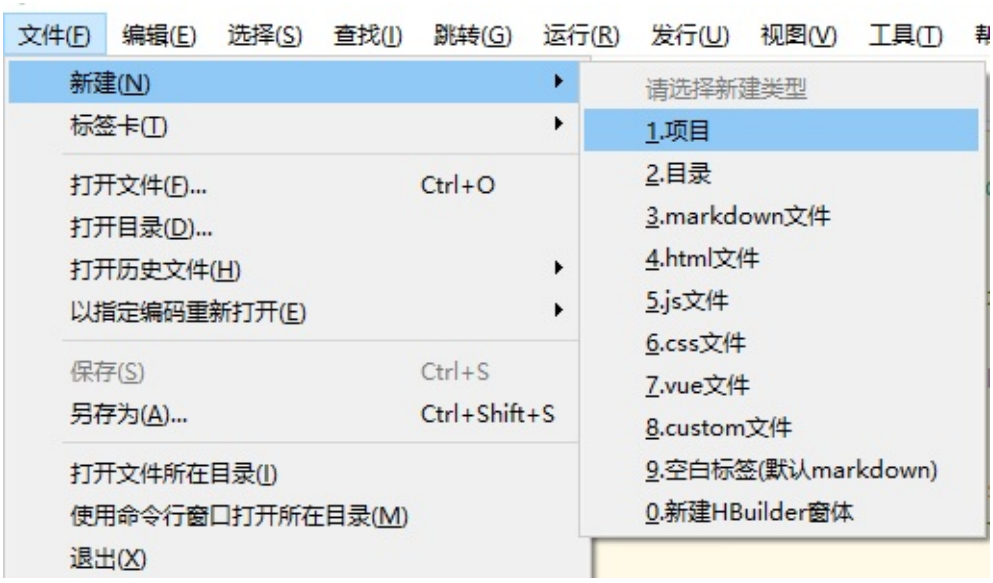
开发者需分别安装如下工具：

```
HBuilderX：内置uni-app编译器及项目模板，http://www.dcloud.io/hbuilderx.html

微信开发者工具：编译调试小程序所用，https://mp.weixin.qq.com/debug/wxadoc/dev/devtools/download.html
```

创建项目

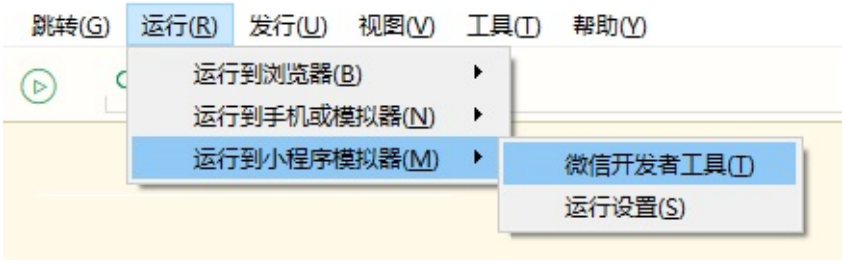
在点击工具栏里的文件 -> 新建 -> 项目：



选择uni-app，输入工程名，如：first-uniapp，点击创建，即可成功创建uni-app。点击从模版创建，选择hello-uniapp即可体验官方示例。

运行uni-app

在微信开发者工具里运行：进入`first-uniapp`项目，点击工具栏的运行 `->` 运行到小程序模拟器 `->` 微信开发者工具，即可在微信开发者工具里面体验`uni-app`。



注意：如果是第一次使用，需要先配置小程序ide的相关路径，才能运行成功。如下图，需在输入框输入微信开发者工具的安装路径。 若HBuilderX不能正常启动微信开发者工具，需要开发者手动启动，然后将uni-app生成小程序工程的路径拷贝到微信开发者工具里面，在HBuilderX里面开发，在小微信开发者工具里面就可看到实时的效果。

uni-app默认把项目编译到根目录的unpackage目录。

真机运行：连接手机，开启USB调试，进入first-uniapp项目，点击工具栏的运行 -> 真机运行 -> 选择运行的设

备，即可在该设备里面体验uni-app。

也可直接点击快捷按钮，选择设备运行：

注意：

真机运行需要设置应用表示，登录 hbuilderX 后点击云端获取即可！

第3讲 : uni-app 开发规范及目录结构

为了实现微信小程序、原生App的跨端兼容，综合考虑编译速度、运行性能等因素，uni-app 约定了如下开发规范：

页面规范 - Vue 单文件组件 (SFC) 规范

.vue 文件是一个自定义的文件类型，用类 HTML 语法描述一个 Vue 组件。每个 .vue 文件包含三种类型的顶级语言块 `<template>`、`<script>` 和 `<style>`，还允许添加可选的自定义块：

```
<template>
  <div>{{ msg }}</div>
</template>
<script>
export default {
  data () {
    return {
      msg: 'Hello world!'
    }
  }
}
</script>
<style>
.example {
  color: red;
}
</style>
<custom1>
  This could be e.g. documentation for the component.
</custom1>
```

vue-loader 会解析文件，提取每个语言块，如有必要会通过其它 loader 处理，最后将他们组装成一个 ES Module，它的默认导出是一个 Vue.js 组件选项的对象。

vue-loader 支持使用非默认语言，比如 CSS 预处理器，预编译的 HTML 模版语言，通过设置语言块的 lang 属性。例如，你可以像下面这样使用 Sass 语法编写样式：

```
<style>
  /* write Sass! */
</style>
```

更多细节可以在使用预处理器中找到。

语言块

模板

每个 .vue 文件最多包含一个 `<template>` 块。

内容将被提取并传递给 vue-template-compiler 为字符串，预处理为 JavaScript 渲染函数，并最终注入到从 `<script>` 导出的组件中。

脚本

每个 .vue 文件最多包含一个 `<script>` 块。

这个脚本会作为一个 ES Module 来执行。

它的默认导出应该是一个 Vue.js 的组件选项对象。也可以导出由 Vue.extend() 创建的扩展对象，但是普通对象是更好的选择。

任何匹配 .js 文件 (或通过它的 lang 特性指定的扩展名) 的 webpack 规则都将会运用到这个 `<script>` 块的内容中。

样式

默认匹配：/.css$/。

一个 .vue 文件可以包含多个 `<style>` 标签。

`<style>` 标签可以有 scoped 或者 module 属性 (查看 scoped CSS和 CSS Modules) 以帮助你将样式封装到当前组件。具有不同封装模式的多个 `<style>` 标签可以在同一个组件中混合使用。

任何匹配 .css 文件 (或通过它的 lang 特性指定的扩展名) 的 webpack 规则都将会运用到这个 `<style>` 块的内容中。

自定义块

可以在 .vue 文件中添加额外的自定义块来实现项目的特定需求，例如 `<docs>` 块。vue-loader 将会使用标签名来查找对应的 webpack loader 来应用在对应的块上。webpack loader 需要在 vue-loader 的选项 loaders 中指定。

更多细节，查看自定义块。

Src 导入

如果喜欢把 .vue 文件分隔到多个文件中，你可以通过 src 属性导入外部文件：

```
<template src="./template.html"></template>
<style src="./style.css"></style>
<script src="./script.js"></script>
```

需要注意的是 src 导入遵循和 webpack 模块请求相同的路径解析规则，这意味着：

相对路径需要以 ./ 开始

你可以从 NPM 依赖中导入资源：

```
<!-- import a file from the installed "todomvc-app-css" npm package -->
<style src="todomvc-app-css/index.css">
在自定义块上同样支持 src 导入，例如：
<unit-test src="./unit-test.js">
</unit-test>
```

组件标签靠近微信小程序规范

详见uni-app 组件规范，注意：不能使用标准HTML标签，也不能用js对dom进行操作

接口能力（JS API）靠近微信小程序规范，但需将前缀 wx 替换为 uni，详见uni-app接口规范

数据绑定及事件处理靠近 Vue.js 规范，同时补充了App及页面的生命周期

为兼容多端运行，建议使用flex布局进行开发（flex 布局教程

http://www.hcoder.net/tutorials/info_183.html）。

目录结构

一个uni-app工程，默认包含如下目录及文件：

pages.json：配置页面路由、导航条、选项卡等页面类信息，详见。

manifest.json：配置应用名称、appid、logo、版本等打包信息，详见。

App.vue：应用配置，用来配置App全局样式以及监听应用的生命周期。

main.js：Vue初始化入口文件

static目录：存放应用引用静态资源（如图片、视频等）的地方，注意：静态资源只能存放于此

pages目录：业务页面文件存放目录

components目录：组件文件存放目录

第4讲：uni-app 页面样式与布局

重要说明

请删除 app.vue 中的全局样式：

view{display:flex;}

在需要flex的时候使用flex即可。

尺寸单位

uni-app框架目前仅支持长度单位 px 和 %。与传统 web 页面不同，px 是相对于基准宽度的单位，已经适配了移动端屏幕，其原理类似于 rem。

PS：uni-app 的基准宽度为 750px。

开发者只需按照设计稿确定框架样式中的 px 值即可。设计稿 1px 与框架样式 1px 转换公式如下：

设计稿 1px / 设计稿基准宽度 = 框架样式1px / 750px

换言之，页面元素宽度在uni-app中的宽度计算公式：

750px * 元素在设计稿中的宽度 / 设计稿基准宽度

举例说明：

若设计稿宽度为 640px，元素 A 在设计稿上的宽度为 100px，那么元素 A 在uni-app里面的宽度应该设为：750 * 100 / 640，结果为：117px。

若设计稿宽度为 375px，元素 B 在设计稿上的宽度为 200px，那么元素 B 在uni-app里面的宽度应该设为：750 * 200 / 375，结果为：400px。

样式导入

使用@import语句可以导入外联样式表，@import后跟需要导入的外联样式表的相对路径，用;表示语句结束。

示例代码：

```
<style>
    @import "../../common/uni.css";
    .uni-card {
        box-shadow: none;
    }
</style>
```

内联样式

框架组件上支持使用 style、class 属性来控制组件的样式。

style：静态的样式统一写到 class 中。style 接收动态的样式，在运行时会进行解析，请尽量避免将静态的样式写

进 style 中，以免影响渲染速度。

```
<view style="color:{{color}};" />
```

class：用于指定样式规则，其属性值是样式规则中类选择器名(样式类名)的集合，样式类名不需要带上.，样式类名之间用空格分隔。

```
<view class="normal_view" />
```

选择器

目前支持的选择器有：

.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，仅微信小程序和5+App生效

::before view::before 在 view 组件前边插入内容，仅微信小程序和5+App生效

全局样式与局部样式

定义在 App.vue 中的样式为全局样式，作用于每一个页面。在 pages 目录下 的 vue 文件中定义的样式为局部样式，只作用在对应的页面，并会覆盖 App.vue 中相同的选择器。

注意： App.vue 中通过 @import 语句可以导入外联样式，一样作用于每一个页面。

Flex布局

为支持跨平台，框架建议使用Flex布局，关于Flex布局可以参考外部文档A Complete Guide to Flexbox等。

第5讲 : uni-app 配置文件 - pages.json

pages.json

pages.json 文件用来对 uni-app 进行全局配置，决定页面文件的路径、窗口表现、设置多 tab 等。

pages.json 配置项列表

globalStyle Object 否 设置默认页面的窗口表现

pages Object Array 是 设置页面路径及窗口表现

tabBar Object 否 设置底部 tab 的表现

condition Object 否 启动模式配置

以下是一个包含了所有配置选项的 pages.json ：

```
{
  "pages": [{
    "path": "pages/component/index",
    "style": {
      "navigationBarTitleText": "组件"
    }
  }, {
    "path": "pages/API/index",
    "style": {
      "navigationBarTitleText": "接口"
    }
  }, {
    "path": "pages/component/view/index",
    "style": {
      "navigationBarTitleText": "view"
    }
  }],
  "globalStyle": {
    "navigationBarTextStyle": "black",
    "navigationBarTitleText": "演示",
    "navigationBarBackgroundColor": "#F8F8F8",
    "backgroundColor": "#F8F8F8"
  },
  "tabBar": {
    "color": "#7A7E83",
    "selectedColor": "#3cc51f",
    "borderStyle": "black",
    "backgroundColor": "#ffffff",
    "list": [{
      "pagePath": "pages/component/index",
      "iconPath": "static/image/icon_component.png",
```

```
      "selectedIconPath": "static/image/icon_component_HL.png",
      "text": "组件"
    }, {
      "pagePath": "pages/API/index",
      "iconPath": "static/image/icon_API.png",
      "selectedIconPath": "static/image/icon_API_HL.png",
      "text": "接口"
    }]
  }
}
```

globalStyle

用于设置应用的状态栏、导航条、标题、窗口背景色等。

navigationBarBackgroundColor HexColor #000000 导航栏背景颜色

navigationBarTextStyle String white 导航栏标题颜色，仅支持 black/white

navigationBarTitleText String 导航栏标题文字内容

navigationStyle String default 导航栏样式，仅支持 default/custom。

backgroundColor HexColor #ffffff 窗口的背景色 微信小程序

注意：navigationStyle 只在 pages.json->globalStyle 中设置生效。开启 custom 后，所有窗口均无导航栏。

pages

接收一个数组，来指定应用由哪些页面组成。每一项代表对应页面的信息，应用中新增/减少页面，都需要对 pages 数组进行修改。

文件名不需要写后缀，框架会自动寻找路径下的页面资源。

注意：pages节点的第一项为应用入口页（即首页）。

代码示例：

开发目录为：

```
pages/
pages/index/index.vue
pages/login/login.vue
mainfest.json
pages.json
```

则需要在 pages.json 中填写

```
{
  "pages": [{
      "index": {
```

```
        "path": "pages/index/index"
      }
    },
    {
      "login": {
        "path": "pages/login/login"
      }
    }
  ]
}
```

pages.style

用于设置每个页面的状态栏、导航条、标题、窗口背景色等。

navigationBarBackgroundColor HexColor #000000 导航栏背景颜色，如"#000000"

navigationBarTextStyle String white 导航栏标题颜色，仅支持 black/white

navigationBarTitleText String 导航栏标题文字内容

backgroundColor HexColor #ffffff 窗口的背景色 微信小程序

backgroundTextStyle String dark 下拉 loading 的样式，仅支持 dark/light

enablePullDownRefresh Boolean false 是否开启下拉刷新，详见页面相关事件处理函数

onReachBottomDistance Number 50 页面上拉触底事件触发时距页面底部距离，单位为px

navigationStyle String default 导航栏样式，仅支持 default/custom。custom 模式可自定义导航栏，只保留右上角胶囊状的按钮。 微信小程序

backgroundColorTop String #ffffff 顶部窗口的背景色。 微信小程序且为 iOS

backgroundColorBottom String #ffffff 底部窗口的背景色。 微信小程序且为 iOS

代码示例：

```
{
  "pages": [{
      "index": {
        "path": "pages/index/index",
        "style": {
            "navigationBarTitleText": "首页",//设置页面标题文字
          "enablePullDownRefresh":true//开启下拉刷新
        }
      }
    },
    ...
  ]
}
```

tabBar

如果应用是一个多 tab 应用，可以通过 tabBar 配置项指定 tab 栏的表现，以及 tab 切换时显示的对应页。

Tips

当设置 position 为 top 时，将不会显示 icon

tabBar 中的 list 是一个数组，只能配置最少2个、最多5个 tab，tab 按数组的顺序排序。

属性说明：

color HexColor 是 tab 上的文字默认颜色

selectedColor HexColor 是 tab 上的文字选中时的颜色

backgroundColor HexColor 是 tab 的背景色

borderStyle String 否 black tabbar 上边框的颜色，仅支持 black/white

list Array 是 tab 的列表，详见 list 属性说明，最少2个、最多5个 tab

position String 否 bottom 可选值 bottom、top

其中 list 接收一个数组，数组中的每个项都是一个对象，其属性值如下：

pagePath String 是 页面路径，必须在 pages 中先定义

text String 是 tab 上按钮文字

iconPath String 否 图片路径，icon 大小限制为40kb，建议尺寸为 81px * 81px，当 postion 为 top 时，此参数无效，不支持网络图片

selectedIconPath String 否 选中时的图片路径，icon 大小限制为40kb，建议尺寸为 81px * 81px ，当 postion 为 top 时，此参数无效

condition

启动模式配置，仅开发期间生效，用于模拟直达页面的场景，如：小程序转发后，用户点击所打开的页面。

属性说明：

current Number 是 当前激活的模式，list节点的索引值

list Array 是 启动模式列表

list说明：

name String 是 启动模式名称

path String 是 启动页面路径

query String 否 启动参数，可在页面的 onLoad 函数里获得

注意： 在5+app里真机运行可直接打开配置的页面，微信开发者工具里需要手动改变编译模式：

代码示例：

```
"condition": { //模式配置，仅开发期间生效
    "current": 0, //当前激活的模式（list 的索引项）
    "list": [{
            "name": "swiper", //模式名称
```

```
            "path": "pages/component/swiper/swiper", //启动页面，必选
            "query": "interval=4000&autoplay=false" //启动参数，在页面的onLoad函数
里面得到。
        },
        {
            "name": "test",
            "path": "pages/component/switch/switch"
        }
    ]
}
```

自行测试

1、改变导航背景颜色及标题

2、创建页面

3、创建 tabbar

第6讲：配置文件 - manifest.json

manifest.json 文件是应用的配置文件，用于指定应用的名称、图标、权限等。

属性说明：

name String 应用名称

appid String 新建uni-app项目时，DCloud云端分配。 应用标识

description String 应用描述

versionName String 版本名称，例如：1.0.0

versionCode String 版本号，例如：36

app-plus Object 5+App 特有配置

quickapp Object 快应用特有配置，即将支持

mp-weixin Object 微信小程序特有配置

mp-baidu Object 百度小程序特有配置，即将支持

mp-alipay Object 支付宝小程序特有配置，即将支持

app-plus

modules Object 权限模块

distribute Object 5+App 发布信息

PS：这里只列出了核心部分，更多内容请参考 完整的 manifest.json。

plus->modules 可配置的权限模块：

名称 描述

Contacts 系统通讯录

Fingerprint 指纹识别

Maps 地图

Messaging 短彩邮件消息

OAuth 登录授权

Payment 支付

Push 消息推送

Share 社交分享

Speech 语音识别

Statistic 统计

VideoPlayer 视频播放

LivePusher 直播推流

plus->distribute 配置说明：

属性 类型 默认值 描述

android Object Android 应用配置

ios Object iOS 应用配置

sdkConfigs Object SDK配置

Tips

manifest.json 文件的配置，推荐在 HBuilderX 提供的可视化操作界面完成。

部分配置在打包时的操作界面补全，例如：证书等信息。

Native.js 权限部分会根据配置的模块权限，在打包后自动填充。

部分 modules 是默认的，不需要进行配置。

mp-weixin

属性 类型 说明

appid String 微信小程序的AppID，登录 申请

第7讲 : uni-app 页面生命周期

页面生命周期

不论是app还是小程序，生命周期是非常重要的知识点。

uni-app 支持如下页面生命周期函数：

onLoad 监听页面加载，其参数为上个页面传递的数据，参数类型为object（用于页面传参），参考示例

onShow 监听页面显示

onReady 监听页面初次渲染完成

onHide 监听页面隐藏

onUnload 监听页面卸载

onPullDownRefresh 监听用户下拉动作

onReachBottom 页面上拉触底事件的处理函数

onShareAppMessage 用户点击右上角分享 微信小程序

onPageScroll 监听页面滚动

onTabItemTap 当前是 tab 页时，点击 tab 时触发。

使用演示

```
<script>
export default {
 data: {
  title: 'Hello'
 },
 onLoad:function(options){
  console.log("onLoad");
 },
 onHide:function(){
  console.log("onHide");
 },
 onShow:function(){
  console.log("onShow");
 }
}
</script>
```

第8讲 : uni-app 模板语法 - 数据绑定

基础数据绑定

变量赋值：

```
<script>
export default {
 data: {
  title: 'Hello',
  name : 'hcoder'
 }, .....
```

在视图中使用 {{}} 调用变量：

```
<template>
 <view class="content">
  <text class="title">{{title}}</text>
  <view>
   hi....{{name}}
  </view>
 </view>
</template>
```

数组形式的数据绑定

```
data: {
  students : [
   {name : "张三", age : 18},
   {name : "李四", age : 20}
  ]
 },
```

//调用

```
<view>
{{students[0]['name']}}
{{students[0].name}}
</view>
```

列表渲染

```
<template>
```

```
 <view>
  <view v-for="(item, index) in students">
   <view class="persons">{{index}} - {{item.name}}</view>
  </view>
 </view>
</template>
<script>
export default {
 data: {
  students : [
   {name : "张三", age : 18},
   {name : "李四", age : 20}
  ]
 },
 onLoad:function(options){
  console.log("onLoad");
 },
 onHide:function(){
  console.log("onHide");
 },
 onShow:function(){
  console.log("onShow");
 }
}
</script>
<style>
.persons{width:750px; line-height:2.2em;}
</style>
```

//说明 :

正常展示效果需要删除 app.vue 内的全局样式 (原因见视频教程)

条件渲染

```
<template>
 <view>
  <view v-for="(item, index) in students">
   <view class="persons">{{index}} - {{item.name}}</view>
  </view>
  <view v-if="show">
   这里是条件展示的内容....
  </view>
 </view>
</template>
<script>
export default {
 data: {
  students : [
```

```
    {name : "张三", age : 18},
    {name : "李四", age : 20}
   ],
   show:false
  },
```

使用 hidden

```
<template>
 <view>
  <view v-for="(item, index) in students">
   <view class="persons">{{index}} - {{item.name}}</view>
  </view>
  <view v-hidden="show">
   这里是条件展示的内容....
  </view>
 </view>
</template>
<script>
export default {
 data: {
  students : [
   {name : "张三", age : 18},
   {name : "李四", age : 20}
  ],
  show:true
 },
```

if 与 hidden

if会根据条件决定是否渲染，hidden 会渲染但根据条件决定是否展示。

第9讲Class 与 Style 绑定（动态菜单激活示例）

为节约性能，我们将 Class 与 Style 的表达式通过 compiler 硬编码到 uni-app 中，支持语法和转换效果如下：

class 支持的语法:

```
<view :class="{ active: isActive }">111</view>
<view class="static" v-bind:class="{ active: isActive, 'text-danger': hasError
}">222</view
<view class="static" :class="[activeClass, errorClass]">333</view>
<view class="static" v-bind:class="[isActive ? activeClass : '', errorClass]">4
44</view>
<view class="static" v-bind:class="[{ active: isActive }, errorClass]">555</view
>
```

style 支持的语法:

```
<view v-bind:style="{ color: activeColor, fontSize: fontSize + 'px' }">666</view
>
<view v-bind:style="[{ color: activeColor, fontSize: fontSize + 'px' }]">777</v
iew>
```

不支持 Vue官方文档：Class 与 Style 绑定 中的 classObject 和 styleObject 语法。

此外还可以用 computed 方法生成 class 或者 style 字符串，插入到页面中，举例说明：

```
<template>
    <!-- 支持 -->
    <view class="container" :class="computedClassStr"></view>
    <view class="container" :class="{active: isActive}"></view>
    <!-- 不支持 -->
    <view class="container" :class="computedClassObject"></view>
</template>
```

动态菜单切换示例

```
<template>
 <scroll-view class="menus">
  <view v-for="(menu, index) in menus" :class="[index == currentIndex ? 'menuAc
tive' : '']">{{menu}}</view>
 </scroll-view>
</template>
```

```
<script>
export default {
 data: {
  currentIndex : 0,
  menus : [
   '新闻', '汽车', '读书'
  ]
 },
 onLoad:function(options){
  console.log("onLoad");
 },
 onHide:function(){
  console.log("onHide");
 },
 onShow:function(){
  console.log("onShow");
 }
}
</script>
<style>
.menus{width:700px; margin:0 auto; padding:20px 0px;}
.menus view{padding:8px; line-height:20px; font:38px; float:left;}
.menuActive{color:#900;}
</style>
```

第10讲 : uni-app 事件处理、事件绑定、事件传参

uni-app 事件

事件映射表，左侧为 WEB 事件，右侧为 `uni-app` 对应事件

{

click: 'tap',

touchstart: 'touchstart',

touchmove: 'touchmove',

touchcancel: 'touchcancel',

touchend: 'touchend',

tap: 'tap',

longtap: 'longtap',

input: 'input',

change: 'change',

submit: 'submit',

blur: 'blur',

focus: 'focus',

reset: 'reset',

confirm: 'confirm',

columnchange: 'columnchange',

linechange: 'linechange',

error: 'error',

scrolltoupper: 'scrolltoupper',

scrolltolower: 'scrolltolower',

scroll: 'scroll'

}

在 input 和 textarea 中 change 事件会被转为 blur 事件。

踩坑注意：

上列表中没有的原生事件也可以使用，例如map组件的regionchange 事件直接在组件上写成 @regionchange, 同时这个事件也非常特殊，它的 event type 有 begin 和 end 两个，导致我们无法在handleProxy 中区分到底是什么事件，所以你在监听此类事件的时候同时监听事件名和事件类型既 <map @regionchange="functionName" @end="functionName" @begin="functionName">

平台差异所致，bind 和 catch 事件同时绑定时候，只会触发 bind ,catch 不会被触发，要避免踩坑。

事件修饰符

stop 的使用会阻止冒泡，但是同时绑定了一个非冒泡事件，会导致该元素上的 catchEventName 失效！

prevent 可以直接干掉，因为uni-app里没有什么默认事件，比如 submit 并不会跳转页面

self 没有可以判断的标识

once 也不能做，因为uni-app没有 removeEventListener, 虽然可以直接在 handleProxy 中处理，但非常的不优雅，违背了原意，暂不考虑

按键修饰符：uni-app运行在手机端，没有键盘事件，所以不支持按键修饰符。

事件绑定 @click

```
<template>
 <view class="demo" @click="clickTest" @longtap="longtap"></view>
</template>
<script>
export default {
 methods:{
  clickTest: function(e){
   console.log(e);
   console.log('click');
  },
  longtap: function(e){
   console.log(e);
   console.log('longtap');
  }
 }
}
</script>
<style>
.demo{width:500px; margin:50px auto; background:#8F8F90; height:500px;}
</style>
```

注意在小程序中观察对应事件对象，可以利用此对象获取更多信息。

事件传参（动态参数演示）

```
<template>
 <view>
   <view class="uni-list">
    <view class="uni-list-cell" v-for="(item,index) in students" :key="index">
        <view class="uni-list-cell-navigate uni-navigate-right" @click="menuCli
ck" v-bind:id="index">
```

```
            {{index}} - {{item.name}}

        </view>
    </view>
   </view>

 </view>
</template>
<script>
export default {
 data: {
  students : [
   {name : "张三", age : 18},
   {name : "李四", age : 20}
  ]
 },
 methods:{
 menuClick : function(e){
  console.log(e);
  console.log(e.target.id);
 }
 },
}
</script>
<style>
.persons{width:750px; line-height:2.2em;}
</style>
```

第11讲 : uni-app 组件 - 基础组件

相关文档

http://uniapp.dcloud.io/component/README

框架为开发者提供了一系列基础组件，开发者可以通过组合这些基础组件进行快速开发。

什么是组件：

**

组件是视图层的基本组成单元**。

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。

```
<template>
    <view>
        <tagname property="value">
            content
        </tagname>
    </view>
</template>
```

Tips

所有组件与属性名都是小写，单词之间以连字符-连接。

根节点为

第12讲 : uni-app 组件 - 表单组件

文档链接

http://uniapp.dcloud.io/component/button

概述

包含表单及常用表单元素的知识点讲解。

包含以下组件的讲解

button、checkbox、form、input、label、picker、radio、slider、switch、textarea

第13讲 : uni-app 组件 - navigator (导航) 及页

navigator

**属性名 **

```
类型 默认值 说明 平台支持

url String  应用内的跳转链接，值为相对路径或绝对路径，如："../first/first", "/pages/first/first"，注意不能加 .vue 后缀

open-type String navigate 跳转方式

delta Number  当 open-type 为 'navigateBack' 时有效，表示回退的层数

hover-class String navigator-hover 指定点击时的样式类，当hover-class="none"时，没有点击态效果

hover-stop-propagation Boolean false 指定是否阻止本节点的祖先节点出现点击态 微信小程序

hover-start-time Number 50 按住后多久出现点击态，单位毫秒

hover-stay-time Number 600 手指松开后点击态保留时间，单位毫秒

open-type 有效值：
```

值 说明 平台支持

```
navigate 对应 uni.navigateTo 的功能

redirect 对应 uni.redirectTo 的功能

switchTab 对应 uni.switchTab 的功能

reLaunch 对应 uni.reLaunch 的功能 微信小程序

navigateBack 对应 uni.navigateBack 的功能 微信小程序
```

注：navigator-hover 默认为 {background-color: rgba(0, 0, 0, 0.1); opacity: 0.7;}, 的子节点背景色应为透明色。

示例：

```
<template>
```

```
    <view>
        <view class="page-body">
            <view class="btn-area">
                <navigator url="navigate/navigate?title=navigate" hover-class="
navigator-hover">
                    <button type="default">跳转到新页面</button>
                </navigator>
                <navigator url="redirect/redirect?title=redirect" redirect hover
-class="other-navigator-hover">
                    <button type="default">在当前页打开</button>
                </navigator>
            </view>
        </view>
    </view>
</template>
export default {
    data() {
        return {
            title: 'navigator'
        }
    },
    methods: {
    }
}
```

页面传值及接收

index/index.vue

test.vue 接收参数

第14讲 : uni-app 组件 - 媒体组件

image 图片

属性名

```
类型 默认值 说明 平台支持
src String 图片资源地址
mode String 'scaleToFill' 图片裁剪、缩放的模式
lazy-load Boolean false 图片懒加载。只针对page与scroll-view下的image有效 微信小程序
@error HandleEvent 当错误发生时，发布到 AppService 的事件名，事件对象event.detail = {
errMsg: 'something wrong'}
@load HandleEvent 当图片载入完毕时，发布到 AppService 的事件名，事件对象event.detail =
{height:'图片高度px', width:'图片宽度px'}
注：image组件默认宽度300px、高度225px
```

mode 有效值：

mode 有 13 种模式，其中 4 种是缩放模式，9 种是裁剪模式。

模式 值 说明

```
缩放 scaleToFill 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
缩放 aspectFit 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。
缩放 aspectFill 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。
缩放 widthFix 宽度不变，高度自动变化，保持原图宽高比不变
裁剪 top 不缩放图片，只显示图片的顶部区域
裁剪 bottom 不缩放图片，只显示图片的底部区域
裁剪 center 不缩放图片，只显示图片的中间区域
裁剪 left 不缩放图片，只显示图片的左边区域
裁剪 right 不缩放图片，只显示图片的右边区域
裁剪 top left 不缩放图片，只显示图片的左上边区域
裁剪 top right 不缩放图片，只显示图片的右上边区域
裁剪 bottom left 不缩放图片，只显示图片的左下边区域
裁剪 bottom right 不缩放图片，只显示图片的右下边区域
```

说明： 图片支持远程路径。

audio 音频

属性名 类型 默认值 说明

```
id String  audio 组件的唯一标识符
```

```
src String  要播放音频的资源地址
loop Boolean false 是否循环播放
controls Boolean false 是否显示默认控件
poster String  默认控件上的音频封面的图片资源地址，如果 controls 属性值为 false 则设置 p
oster 无效
name String 未知音频 默认控件上的音频名字，如果 controls 属性值为 false 则设置 name 无
效
author String 未知作者 默认控件上的作者名字，如果 controls 属性值为 false 则设置 author
 无效
binderror EventHandle  当发生错误时触发 error 事件，detail = {errMsg: MediaError.c
ode}
bindplay EventHandle  当开始/继续播放时触发play事件
bindpause EventHandle  当暂停播放时触发 pause 事件
bindtimeupdate EventHandle  当播放进度改变时触发 timeupdate 事件，detail = {current
Time, duration}
bindended EventHandle  当播放到末尾时触发 ended 事件
MediaError.code
```

返回错误码 描述

```
1 获取资源被用户禁止
2 网络错误
3 解码错误
4 不合适资源
```

示例 :

```
<template>
    <view>
        <view class="page-body">
            <view class="page-section page-section-gap" style="text-align: cent
er;">
                <audio style="text-align: left" :src="current.src" :poster="cur
rent.poster" :name="current.name" :author="current.author" :action="audioAction"
 controls></audio>
            </view>
        </view>
    </view>
</template>
export default {
    data() {
        return {
           title: 'audio',
            current: {
                poster: 'https://img-cdn-qiniu.dcloud.net.cn/uniapp/audio/music
.jpg',
                name: '致爱丽丝',
```

```
                author: '暂无',
                src: 'https://img-cdn-qiniu.dcloud.net.cn/uniapp/audio/music.mp3'
,
            },
            audioAction: {
                method: 'pause'
            }
        }
    }
}
```

第15讲 : uni-app 组件 - 地图组件

map 地图

```
属性名 类型 默认值 说明 平台支持
longitude Number  中心经度
latitude Number  中心纬度
scale Number 16 缩放级别，取值范围为5-18
markers Array  标记点
covers Array  即将移除，请使用 markers
polyline Array  路线
circles Array  圆
controls Array  控件
include-points Array  缩放视野以包含所有给定的坐标点
show-location Boolean  显示带有方向的当前定位点
@markertap EventHandle  点击标记点时触发
@callouttap EventHandle  点击标记点对应的气泡时触发 微信小程序、5+App
@controltap EventHandle  点击控件时触发
@regionchange EventHandle  视野发生变化时触发
@tap EventHandle  点击地图时触发
@updated EventHandle  在地图渲染更新完成时触发 微信小程序
```

注意:

covers 属性即将移除，请使用 markers 替代

uniapp只支持gcj02坐标

markers

标记点用于在地图上显示标记的位置

```
属性 说明 类型 必填 备注 平台支持
id 标记点id Number 否 marker点击事件回调会返回此id。建议为每个marker设置上Number类型id
，保证更新marker时有更好的性能。
latitude 纬度 Number 是 浮点数，范围 -90 ~ 90
longitude 经度 Number 是 浮点数，范围 -180 ~ 180
title 标注点名 String 否
iconPath 显示的图标 String 是 项目目录下的图片路径，支持相对路径写法，以'/'开头则表示相对
小程序根目录；也支持临时路径
rotate 旋转角度 Number 否 顺时针旋转的角度，范围 0 ~ 360，默认为 0
alpha 标注的透明度 Number 否 默认1，无透明，范围 0 ~ 1
width 标注图标宽度 Number 否 默认为图片实际宽度
height 标注图标高度 Number 否 默认为图片实际高度
callout 自定义标记点上方的气泡窗口 Object 否 支持的属性见下表，可识别换行符。 微信小程序、5+
App
label 为标记点旁边增加标签 Object 否 支持的属性见下表，可识别换行符。 微信小程序、5+App
anchor 经纬度在标注图标的锚点，默认底边中点 Object 否 {x, y}，x表示横向(0-1)，y表示竖向(0
```

```
-1)。{x: .5, y: 1} 表示底边中点 微信小程序、5+App
```

marker 上的气泡 callout

```
属性 说明 类型 平台支持
content 文本 String 微信小程序、5+App
color 文本颜色 String 微信小程序、5+App
fontSize 文字大小 Number 微信小程序、5+App
borderRadius callout边框圆角 Number 微信小程序、5+App
bgColor 背景色 String 微信小程序、5+App
padding 文本边缘留白 Number 微信小程序、5+App
display 'BYCLICK':点击显示; 'ALWAYS':常显 String 微信小程序、5+App
textAlign 文本对齐方式。有效值: left, right, center String 微信小程序、5+App
```

marker 上的气泡 label

```
属性 说明 类型 平台支持
content 文本 String 微信小程序、5+App
color 文本颜色 String 微信小程序、5+App
fontSize 文字大小 Number 微信小程序、5+App
x label的坐标，原点是 marker 对应的经纬度 Number 微信小程序、5+App
y label的坐标，原点是 marker 对应的经纬度 Number 微信小程序、5+App
borderWidth 边框宽度 Number 微信小程序、5+App
borderColor 边框颜色 String 微信小程序、5+App
borderRadius 边框圆角 Number 微信小程序、5+App
bgColor 背景色 String 微信小程序、5+App
padding 文本边缘留白 Number 微信小程序、5+App
textAlign 文本对齐方式。有效值: left, right, center String 微信小程序、5+App
```

polyline

指定一系列坐标点，从数组第一项连线至最后一项

```
属性 说明 类型 必填 备注 平台支持
points 经纬度数组 Array 是 [{latitude: 0, longitude: 0}]
color 线的颜色 String 否 8位十六进制表示，后两位表示alpha值，如：#000000AA
width 线的宽度 Number 否
dottedLine 是否虚线 Boolean 否 默认false
arrowLine 带箭头的线 Boolean 否 默认false，开发者工具暂不支持该属性 微信小程序、5+App
arrowIconPath 更换箭头图标 String 否 在arrowLine为true时生效 微信小程序、5+App
borderColor 线的边框颜色 String 否  微信小程序、5+App
borderWidth 线的厚度 Number 否  微信小程序、5+App
```

circles

在地图上显示圆

```
属性 说明 类型 必填 备注
latitude 纬度 Number 是 浮点数，范围 -90 ~ 90
longitude 经度 Number 是 浮点数，范围 -180 ~ 180
color 描边的颜色 String 否 8位十六进制表示，后两位表示alpha值，如：#000000AA
fillColor 填充颜色 String 否 8位十六进制表示，后两位表示alpha值，如：#000000AA
radius 半径 Number 是
strokeWidth 描边的宽度 Number 否
```

controls

在地图上显示控件，控件不随着地图移动

```
属性 说明 类型 必填 备注
id 控件id Number 否 在控件点击事件回调会返回此id
position 控件在地图的位置 Object 是 控件相对地图位置
iconPath 显示的图标 String 是 项目目录下的图片路径，支持相对路径写法，以'/'开头则表示相对小程序根目录；也支持临时路径
clickable 是否可点击 Boolean 否 默认不可点击
```

position

```
属性 说明 类型 必填 备注
left 距离地图的左边界多远 Number 否 默认为0
top 距离地图的上边界多远 Number 否 默认为0
width 控件宽度 Number 否 默认为图片宽度
height 控件高度 Number 否 默认为图片高度
```

地图组件的经纬度必填，如果不填经纬度则默认值是北京的经纬度。

示例：

```
<template>
    <view>
        <view class="page-body">
            <view class="page-section page-section-gap">
                <map style="width: 100%; height: 300px;" :latitude="latitude" :
longitude="longitude" :markers="covers">
                </map>
            </view>
        </view>
    </view>
</template>
export default {
    data() {
        return {
            title: 'map',
```

```
            latitude: 39.909,
            longitude: 116.39742,
            markers: [{
    width : 40,
    height: 40,
                latitude: 39.909,
                longitude: 116.39742,
                iconPath: '../../../static/p.png'
            }]
        }
    },
    methods: {
    }
}
</script>```
```

第16讲 : uni-app 接口 - 网络请求

1、 uni.request(OBJECT) 发起网络请求

```
参数名 类型 必填 默认值 说明

url String 是  开发者服务器接口地址

data Object/String/ArrayBuffer 否  请求的参数

header Object 否  设置请求的 header, header 中不能设置 Referer。

method String 否 GET （需大写）有效值：OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE
, CONNECT

dataType String 否 json 如果设为 json, 会尝试对返回的数据做一次 JSON.parse

responseType String 否 text 设置响应的数据类型。合法值：text、arraybuffer

success Function 否  收到开发者服务成功返回的回调函数

fail Function 否  接口调用失败的回调函数

complete Function 否  接口调用结束的回调函数（调用成功、失败都会执行）
```

success 返回参数说明

```
参数 类型 说明

data Object/String/ArrayBuffer 开发者服务器返回的数据

statusCode Number 开发者服务器返回的 HTTP 状态码

header Object 开发者服务器返回的 HTTP Response Header
```

data 数据说明

最终发送给服务器的数据是 String 类型，如果传入的 data 不是 String 类型，会被转换成 String。转换规则如下：

```
对于 GET 方法, 会将数据转换为 query string。例如 { name: 'name', age: 18 } 转换后的结
果是 name=name&age=18。

对于 POST 方法且 header['content-type'] 为 application/json 的数据, 会进行 JSON 序列
```

化。

对于 POST 方法且 header['content-type'] 为 application/x-www-form-urlencoded 的数据，会将数据转换为 query string。

示例：

```
<template>
    <view></view>
</template>
<script>
export default {
 data:{

 },
    onLoad:function(){
  //get
     const requestTask1 = uni.request({
   url: 'https://demo.hcoder.net',
   success: function (res) {
    console.log(res.data);
   }
  });
  //
  const requestTask2 = uni.request({
   url: 'https://demo.hcoder.net/index.php?m=getJson',
   success: function (res) {
    console.log(res.data);
   }
  });
  //
  const requestTask3 = uni.request({
   url: 'https://demo.hcoder.net/index.php',
   data: {name : 'hcoder...', 'age' : 18},
   method:"POST",
   header : {'content-type':'application/x-www-form-urlencoded'},
   success: function (res) {
    console.log(res.data);
   }
  });
    }
}
</script>
```

第17讲 : uni-app 接口 - 从本地相册选择图片或使

uni.chooseImage(OBJECT) 从本地相册选择图片或使用相机拍照。

count Number 否 最多可以选择的图片张数，默认9

sizeType StringArray 否 original 原图，compressed 压缩图，默认二者都有

sourceType StringArray 否 album 从相册选图，camera 使用相机，默认二者都有

success Function 是 成功则返回图片的本地文件路径列表 tempFilePaths

fail Function 否 接口调用失败的回调函数

complete Function 否 接口调用结束的回调函数（调用成功、失败都会执行）

注：文件的临时路径，在应用本次启动期间可以正常使用，如需持久保存，需在主动调用 uni.saveFile，在应用下次启动时才能访问得到。

success 返回参数说明

参数 类型 说明

tempFilePaths StringArray 图片的本地文件路径列表

tempFiles ObjectArray 图片的本地文件列表，每一项是一个 File 对象

File 对象结构如下

path String 本地文件路径

size Number 本地文件大小，单位：B

示例

```
uni.chooseImage({
count: 6, //默认9
sizeType: ['original', 'compressed'], //可以指定是原图还是压缩图，默认二者都有
sourceType: ['album'], //从相册选择
success: function (res) {
console.log(JSON.stringify(res.tempFilePaths));
}
});
```

uni.previewImage(OBJECT)

预览图片

OBJECT 参数说明

current String 否 当前显示图片的链接，不填则默认为 urls 的第一张

urls StringArray 是 需要预览的图片链接列表

success Function 否 接口调用成功的回调函数

fail Function 否 接口调用失败的回调函数

complete Function 否 接口调用结束的回调函数（调用成功、失败都会执行）

示例

```
uni.chooseImage({
count: 6,
sizeType: ['original', 'compressed'],
sourceType: ['album'],
success: function (res) {
// 预览图片
uni.previewImage({
urls: res.tempFilePaths
});
}
});
```

uni.getImageInfo(OBJECT) 获取图片信息

OBJECT 参数说明

参数名 类型 必填 说明

src String 是 图片的路径，可以是相对路径，临时文件路径，存储文件路径，网络图片路径

success Function 否 接口调用成功的回调函数

fail Function 否 接口调用失败的回调函数

complete Function 否 接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数说明

参数名 类型 说明 最低版本

width Number 图片宽度，单位px *

height Number 图片高度，单位px *

path String 返回图片的本地路径 *

orientation String 返回图片的方向，有效值见下表 *

type String 返回图片的格式 *

orientation 参数说明

枚举值 说明

up 默认

down 180度旋转

left 逆时针旋转90度

right 顺时针旋转90度

up-mirrored 同up，但水平翻转

down-mirrored 同down，但水平翻转

left-mirrored 同left，但垂直翻转

right-mirrored 同right，但垂直翻转

示例

uni.chooseImage({

```
count: 1,
sourceType: ['album'],
success: function (res) {
    uni.getImageInfo({
        src: res.tempFilePaths[0],
        success: function (image) {
            console.log(image.width);
            console.log(image.height);
        }
    });
}
```

});

uni.saveImageToPhotosAlbum(OBJECT)

保存图片到系统相册

参数名 类型 必填 说明

filePath String 是 图片文件路径，可以是临时文件路径也可以是永久文件路径，不支持网络图片路径

success Function 否 接口调用成功的回调函数

fail Function 否 接口调用失败的回调函数

complete Function 否 接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数说明

参数名 类型 说明

errMsg String 调用结果

示例代码：

uni.chooseImage({

```
count: 1,
sourceType: ['camera'],
success: function (res) {
    uni.saveImageToPhotosAlbum({
        filePath: res.tempFilePaths[0],
        success: function () {
            console.log('save success');
        }
    });
}
```

});

第18讲 : uni-app 上传（图片上传实战）

uni.uploadFile(OBJECT)

将本地资源上传到开发者服务器，客户端发起一个 POST 请求，其中 content-type 为 multipart/form-data。如页面通过 uni.chooseImage 等接口获取到一个本地资源的临时文件路径后，可通过此接口将本地资源上传到指定服务器。

OBJECT 参数说明

```
参数名 类型 必填 说明 平台支持

url String 是 开发者服务器 url
files Aarry 否 需要上传的文件列表。使用 files 时，filePath 和 name 不生效。 5+App
filePath String 是 要上传文件资源的路径。
name String 是 文件对应的 key , 开发者在服务器端通过这个 key 可以获取到文件二进制内容
header Object 否 HTTP 请求 Header, header 中不能设置 Referer
formData Object 否 HTTP 请求中其他额外的 form data
success Function 否 接口调用成功的回调函数
fail Function 否 接口调用失败的回调函数
complete Function 否 接口调用结束的回调函数（调用成功、失败都会执行）
```

**files参数说明

**

```
files 参数是一个 file 对象的数组，file 对象的结构如下：

name String 否 multipart 提交时，表单的项目名，默认为 file
uri String 是 文件的本地地址
```

success 返回参数说明

```
参数 类型 说明
data String 开发者服务器返回的数据
statusCode Number 开发者服务器返回的 HTTP 状态码
```

返回值

返回一个 uploadTask 对象，通过 uploadTask，可监听上传进度变化事件，以及取消上传任务。

uploadTask 对象的方法列表

```
onProgressUpdate callback 监听上传进度变化
```

```
abort  中断上传任务

onProgressUpdate 返回参数说明
```

```
参数 类型 说明

progress Number 上传进度百分比

totalBytesSent Number 已经上传的数据长度，单位 Bytes

totalBytesExpectedToSend Number 预期需要上传的数据总长度，单位 Bytes
```

实战：选择一个照片上传（带进度条）

```
<template>
    <view>
  <view>
   <progress :percent="percent" stroke-width="10"></progress>
  </view>
  <view>
   <button type="primary" :loading="loading" :disabled="disabled" @click="uploa
d">选择照片</button>
  </view>
 </view>

</template>
<script>
var _self;
export default {
 data:{
  percent:0,
  loading:false,
  disabled:false
 },
 methods : {
  upload : function(){
   _self = this;
   uni.chooseImage({
    count: 1,
    sizeType: ['original', 'compressed'], //可以指定是原图还是压缩图，默认二者都有
    sourceType: ['album'], //从相册选择
    success: function (res) {
     const tempFilePaths = res.tempFilePaths;
     const uploadTask = uni.uploadFile({
      url : 'https://demo.hcoder.net/index.php?c=uperTest',
      filePath: tempFilePaths[0],
      name: 'file',
```

```
        formData: {
         'user': 'test'
        },
        success: function (uploadFileRes) {
         console.log(uploadFileRes.data);
        }
       });

       uploadTask.onProgressUpdate(function (res) {
        _self.percent = res.progress;
        console.log('上传进度' + res.progress);
        console.log('已经上传的数据长度' + res.totalBytesSent);
        console.log('预期需要上传的数据总长度' + res.totalBytesExpectedToSend);
       });
      },
      error : function(e){
       console.log(e);
      }
     });
    }
  },
     onLoad:function(){

     }
}
</script>
```

后端文件接收代码 (php 版)

```
<?php
class uperTestController extends witController{
    public function index(){
        if(!empty($_FILES['file'])){
            //获取扩展名
            $exename  = $this->getExeName($_FILES['file']['name']);
            if($exename != 'png' && $exename != 'jpg' && $exename != 'gif'){
                exit('不允许的扩展名');
            }
            $imageSavePath = uniqid().'.'.$exename;
            if(move_uploaded_file($_FILES['file']['tmp_name'], $imageSavePath)){

                echo $imageSavePath;
            }
        }
    }

    public function getExeName($fileName){
        $pathinfo      = pathinfo($fileName);
        return strtolower($pathinfo['extension']);
```

```
    }
}
```

第19讲 : uni-app 接口 - 数据缓存

uni.setStorage(OBJECT)

将数据存储在本地缓存中指定的 key 中，会覆盖掉原来该 key 对应的内容，这是一个异步接口。

OBJECT 参数说明

```
参数名 类型 必填 说明
key String 是 本地缓存中的指定的 key
data Object/String 是 需要存储的内容
success Function 否 接口调用成功的回调函数
fail Function 否 接口调用失败的回调函数
complete Function 否 接口调用结束的回调函数（调用成功、失败都会执行）
示例
```

```
uni.setStorage({
    key: 'storage_key',
    data: 'hello',
    success: function () {
        console.log('success');
    }
});
```

uni.setStorageSync(KEY,DATA)

将 data 存储在本地缓存中指定的 key 中，会覆盖掉原来该 key 对应的内容，这是一个同步接口。

参数说明

```
参数 类型 必填 说明
key String 是 本地缓存中的指定的 key
data Object/String 是 需要存储的内容
```

```
try {
    uni.setStorageSync('storage_key', 'hello');
} catch (e) {
    // error
}
```

uni.getStorage(OBJECT)

从本地缓存中异步获取指定 key 对应的内容。

OBJECT 参数说明

```
参数名 类型 必填 说明
key String 是 本地缓存中的指定的 key
success Function 是 接口调用的回调函数, res = {data: key对应的内容}
fail Function 否 接口调用失败的回调函数
complete Function 否 接口调用结束的回调函数（调用成功、失败都会执行）
```

success 返回参数说明

```
参数 类型 说明
data String key 对应的内容
```

示例

```
uni.getStorage({
    key: 'storage_key',
    success: function (res) {
        console.log(res.data);
    }
});
```

uni.getStorageSync(KEY)

从本地缓存中同步获取指定 key 对应的内容。

参数说明

`参数 类型 必填 说明 key String 是 本地缓存中的指定的 key`

示例

```
try {
    const value = uni.getStorageSync('storage_key');
    if (value) {
        console.log(value);
    }
} catch (e) {
    // error
}
```

uni.getStorageInfo(OBJECT)

异步获取当前 storage 的相关信息。

OBJECT 参数说明

```
参数名 类型 必填 说明
success Function 是 接口调用的回调函数，详见返回参数说明
fail Function 否 接口调用失败的回调函数
complete Function 否 接口调用结束的回调函数（调用成功、失败都会执行）
```

success 返回参数说明

参数 类型 说明

keys String Array 当前 storage 中所有的 key

currentSize Number 当前占用的空间大小, 单位：kb

limitSize Number 限制的空间大小, 单位：kb

示例

```
uni.getStorageInfo({
    success: function (res) {
        console.log(res.keys);
        console.log(res.currentSize);
        console.log(res.limitSize);
    }
});
```

uni.getStorageInfoSync()**

同步获取当前 storage 的相关信息。

示例

```
try {
    const res = uni.getStorageInfoSync();
    console.log(res.keys);
    console.log(res.currentSize);
    console.log(res.limitSize);
} catch (e) {
    // error
}
```

uni.removeStorage(OBJECT)

从本地缓存中异步移除指定 key。

OBJECT 参数说明

```
参数名 类型 必填 说明
```

```
key String 是 本地缓存中的指定的 key
success Function 是 接口调用的回调函数
fail Function 否 接口调用失败的回调函数
complete Function 否 接口调用结束的回调函数（调用成功、失败都会执行）
```

示例

```
uni.removeStorage({
    key: 'storage_key',
    success: function (res) {
        console.log('success');
    }
});
```

uni.removeStorageSync(KEY)

从本地缓存中同步移除指定 key。

参数说明

```
参数名 类型 必填 说明
key String 是 本地缓存中的指定的 key
```

示例

```
try {
    uni.removeStorageSync('storage_key');
} catch (e) {
    // error
}
```

uni.clearStorage()

清理本地数据缓存。

示例

```
uni.clearStorage();
```

uni.clearStorageSync()

同步清空本地数据缓存。

示例

```
try {
    uni.clearStorageSync();
} catch (e) {
    // error
}
```

第20讲 : uni-app 设备相关

本节课程包含以下信息：

1、系统信息

2、网络状态

3、加速度计

4、拨打电话

5、剪贴板

6、屏幕亮度

7、振动

api 手册地址

http://uniapp.dcloud.io/api/system/info

第21讲 : uni-app 交互反馈

包含以下内容

```
uni.showToast
uni.showLoading
uni.hideToast
uni.hideLoading
uni.showModal
uni.showActionSheet
```

手册地址

http://uniapp.dcloud.io/api/system/info

第22讲 : uni-app 设置导航条

设置导航条

```
uni.setNavigationBarTitle
uni.showNavigationBarLoading
uni.hideNavigationBarLoading
uni.setNavigationBarColor
```

手册地址

http://uniapp.dcloud.io/api/ui/navigationbar

第23讲 : uni-app 导航（页面流转）

导航

```
uni.navigateTo

uni.redirectTo

uni.reLaunch

uni.switchTab

uni.navigateBack
```

手册地址

http://uniapp.dcloud.io/api/ui/navigate?id=navigateback

第24讲 : uni-app 下拉刷新

onPullDownRefresh

在 js 中定义 onPullDownRefresh 处理函数（和onLoad等生命周期函数同级），监听该页面用户下拉刷新事件。

需要在 pages.json 里，找到的当前页面的pages节点，并在 style 选项中开启 enablePullDownRefresh

当处理完数据刷新后，uni.stopPullDownRefresh 可以停止当前页面的下拉刷新。

uni.startPullDownRefresh(OBJECT)

开始下拉刷新，调用后触发下拉刷新动画，效果与用户手动下拉刷新一致。

```
参数名 类型 必填 说明
success Function 否 接口调用成功的回调
fail Function 否 接口调用失败的回调函数
complete Function 否 接口调用结束的回调函数（调用成功、失败都会执行）
```

uni.stopPullDownRefresh()

停止当前页面下拉刷新。

完整演示

page.json 开启下拉刷新

```
{
 "pages": [
  {
   "path": "pages/index/index",
   "style": {
    "navigationBarTitleText": "hi uni",
    "enablePullDownRefresh": true
   }
  },
  .......
```

index.vue

```
<template>
    <view>
  <view v-for="(item, index) in newsList" class="newslist">{{item}}</view>
 </view>
</template>
```

```
<script>
var _self;
export default {
 data:{
  newsList:[]
 },
    onLoad:function(){
  _self = this;
    },
 onPullDownRefresh:function(){
  this.getnewsList();
 },
 methods:{
  getnewsList : function(){
   uni.showNavigationBarLoading();
   uni.request({
    url: 'https://demo.hcoder.net/index.php?user=hcoder&pwd=hcoder&m=list1&page
=1',
    method: 'GET',
    success: function(res){
     console.log(res);
     _self.newsList = res.data.split('--hcSplitor--');
     uni.hideNavigationBarLoading();
     uni.stopPullDownRefresh();
    }
   });
  }
 },
}
</script>
<style>
.newslist{padding:10px; font-size:28px;}
</style>
```

第25讲 : uni-app 上拉加载更多

完整代码

```
<template>
    <view>
  <view v-for="(item, index) in newsList" class="newslist">{{item}}</view>
  <view class="loading">{{loadingText}}</view>
 </view>
</template>
<script>
var _self, page = 1, timer = null;
export default {
 data:{
  newsList:[],
  loadingText:'加载中...'
 },
    onLoad:function(){
  _self = this;
  this.getnewsList();
    },
 onPullDownRefresh:function(){
  this.getnewsList();
 },
 onReachBottom:function(){
  if(timer != null){
   clearTimeout(timer);
  }
  timer = setTimeout(function(){
   _self.getmorenews();
  }, 1000);
 },
 methods:{
  getmorenews : function(){
   if(_self.loadingText != '' && _self.loadingText != '加载更多'){
    return false;
   }
   _self.loadingText = '加载中...';
   uni.showNavigationBarLoading();
   uni.request({
    url: 'https://demo.hcoder.net/index.php?user=hcoder&pwd=hcoder&m=list1&page
='+page,
    method: 'GET',
    success: function(res){
     _self.loadingText = '';
     if(res.data == null){
      uni.hideNavigationBarLoading();
```

```
        _self.loadingText = '已加载全部';
        return false;
       }
       page++;
       console.log(res);
       _self.newsList = _self.newsList.concat(res.data.split('--hcSplitor--'));
       _self.loadingText = '加载更多';
       uni.hideNavigationBarLoading();
      }
     });
    },
    getnewsList : function(){
     page = 1;
     uni.showNavigationBarLoading();
     uni.request({
      url: 'https://demo.hcoder.net/index.php?user=hcoder&pwd=hcoder&m=list1&page
=1',
      method: 'GET',
      success: function(res){
       page++;
       _self.newsList = res.data.split('--hcSplitor--');
       uni.hideNavigationBarLoading();
       uni.stopPullDownRefresh();
       _self.loadingText = '加载更多';
      }
     });
    }
   }
}
</script>
<style>
.newslist{padding:10px; line-height:60px; font-size:28px;}
.loading{text-align:center; line-height:80px;}
</style>
```

第26讲：uni-app 第三方登录（小程序篇）

重要说明

因小程序和app登录接口不同，需要在前端进行跨端兼容处理！

小程序端必须的配置

小程序端必须配置 app id（申请小程序开发者并获取 appid 及相关秘钥，支持个人开发者）。获取appid后编辑 manifest.json：

```
"mp-weixin" : {
    "appid" : "您的app id"
}
```

接口地址

https://developers.weixin.qq.com/miniprogram/dev/api/open.html#wxgetuserinfoobject

app 端必须的配置

app 端支持微信、qq、微博等多种登录方式，都需要申请对应的开发者并获取对应的 appid。获取对应的appid 后打开 manifest 可视化操作填写即可：

是否登录判断（App.vue）

```
global.isLogin = function(){
 try{
  var suid  = uni.getStorageSync('suid');
  var srand = uni.getStorageSync('srand');
 }catch(e){
  //TODO handle the exception
 }
 if(suid == '' || srand == ''){
  return false;
 }else{
  return [suid, srand];
 }
};
```

需要登录的页面判断

```
var res = global.isLogin();
  if(!res){
```

```
    uni.showModal({
     title:'请登录',
     content:"请登录",
     success:function(){
      uni.navigateTo({
       url:"/pages/login"
      });
     }
    })
   }
```

登录页面开发

```
<template>
    <view style="padding:35px;">
  <!-- #ifdef MP-WEIXIN -->
  <button type="primary" open-type="getUserInfo" @getuserinfo="getuserinfo" wit
hCredentials="true">微信登录</button>
  <!-- #endif -->
  <!-- #ifdef APP-PLUS -->
  <button type="primary" open-type="getUserInfo" @click="getuserinfo" withCrede
ntials="true">微信登录</button>
  <!-- #endif -->
  <button style="margin-top:50px;">手机号码登录</button>
 </view>
</template>
<script>
var _self;
export default {
 data:{

 },
    onLoad:function(){
  _self = this;
    },
 methods:{
  getuserinfo : function(res1){
   console.log(res1);
   //如果只需要opendid 和非加密数据至此登录完成
   //此处连接数据库利用openid 就可以进行登录环节
   //免费的视频教程 http://www.hcoder.net/tutorials/info_141.html
   wx.login({
    success:function(res2){
     //获取 sessionKey
     wx.request({
      url : 'https:///hoa.hcoder.net/xcxencode/?c=sk&appid=wxbb7f9f1f2c6f4f33&s
ecret=739b970b832f0df158f54c494a08e440&code='+res2.code,
      success:function(res3){
       console.log(res3);
```

```
        //记录到本地
        try{
         uni.setStorageSync('sk', res3.data.session_key);
         uni.setStorageSync('openid', res3.data.openid);
        }catch(e){
         //TODO handle the exception
        }
        uni.hideLoading();
        //以下步骤可以获取加密信息，需要授权
        //获取加密信息
        if(!res1.detail.iv){
         uni.showToast({
          title:"您取消了授权,登录失败",
          icon:"none"
         });
         return false;
        }
        try{
         var sessionKey = uni.getStorageSync('sk');
         console.log(sessionKey);
        }catch(e){
         //TODO handle the exception
        }
        uni.request({
         /**
         * $appid         = $_POST['appid'];
          $sessionKey    = $_POST['sessionKey'];
          $encryptedData = $_POST['encryptedData'];
          $iv            = $_POST['iv'];
         */
         method : "POST",
         url : 'https:///hoa.hcoder.net/xcxencode/',
         header : {'content-type':'application/x-www-form-urlencoded'},
         data : {
          appid : "wxbb7f9f1f2c6f4f33",
          sessionKey : sessionKey,
          iv : res1.detail.iv,
          encryptedData : res1.detail.encryptedData
         },
         success:function(res4){
          //"{"openId":"oS6of0V0rdp9nY_BuvCnQUasOHYc","nickName":"深海",
          //"gender":1,"language":"zh_CN","city":"Xi'an","province":"Shaanxi",
          //"country":"China","avatarUrl":"https://wx.qlogo.cn/mmopen/vi_32/7iag
s6YD4enyU"
          console.log(res4);
          //至此登录完成
         }
        });
       }
      })
```

```
      }
    });
   }
  }
}
</script>
<style>

</style>
```

第27讲 : uni-app 登录 (h5+ app 篇)

完整代码

```
<template>
    <view style="padding:35px;">
  <!-- #ifdef MP-WEIXIN -->
  <button type="primary" open-type="getUserInfo" @getuserinfo="getuserinfo" wit
hCredentials="true">微信登录</button>
  <!-- #endif -->
  <!-- #ifdef APP-PLUS -->
  <button type="primary" open-type="getUserInfo" @click="getuserinfoh5appwx" wi
thCredentials="true">微信登录</button>
  <!-- #endif -->
  <button style="margin-top:50px;">手机号码登录</button>
 </view>
</template>
<script>
var _self;
export default {
 data:{

 },
    onLoad:function(){
  _self = this;
    },
 methods:{
  getuserinfoh5appwx: function(){
   uni.login({
    success:function(res2){
     console.log(JSON.stringify(res2) + '2');
     uni.getUserInfo({
      success:function(res3){
       console.log(JSON.stringify(res3) + '3');
      }
     })
    },
   });
  },
  getuserinfo : function(res1){
   console.log(JSON.stringify(res1) + '1');
   //如果只需要opendid 和非加密数据至此登录完成
   //此处连接数据库利用openid 就可以进行登录环节
   //免费的视频教程 http://www.hcoder.net/tutorials/info_141.html
   uni.login({
    success:function(res2){
     console.log(JSON.stringify(res2) + '2');
```

```
      //获取 sessionKey
      uni.request({
       url : 'https:///hoa.hcoder.net/xcxencode/?c=sk&appid=wxbb7f9f1f2c6f4f33&s
ecret=739b970b832f0df158f54c494a08e440&code='+res2.code,
      success:function(res3){
       console.log(JSON.stringify(res3) + '3');
       //记录到本地
       try{
        uni.setStorageSync('sk', res3.data.session_key);
        uni.setStorageSync('openid', res3.data.openid);
       }catch(e){
        //TODO handle the exception
       }
       uni.hideLoading();
       //以下步骤可以获取加密信息, 需要授权
       //获取加密信息
       if(!res1.detail.iv){
        uni.showToast({
         title:"您取消了授权,登录失败",
         icon:"none"
        });
        return false;
       }
       try{
        var sessionKey = uni.getStorageSync('sk');
        console.log(sessionKey);
       }catch(e){
        //TODO handle the exception
       }
       uni.request({
        /**
        * $appid         = $_POST['appid'];
         $sessionKey    = $_POST['sessionKey'];
         $encryptedData = $_POST['encryptedData'];
         $iv            = $_POST['iv'];
        */
        method : "POST",
        url : 'https:///hoa.hcoder.net/xcxencode/',
        header : {'content-type':'application/x-www-form-urlencoded'},
        data : {
         appid : "wxbb7f9f1f2c6f4f33",
         sessionKey : sessionKey,
         iv : res1.detail.iv,
         encryptedData : res1.detail.encryptedData
        },
        success:function(res4){
         //"{"openId":"oS6of0V0rdp9nY_BuvCnQUasOHYc","nickName":"深海",
         //"gender":1,"language":"zh_CN","city":"Xi'an","province":"Shaanxi",
         //"country":"China","avatarUrl":"https://wx.qlogo.cn/mmopen/vi_32/7iag
s6YD4enyU"
```

```
          console.log(JSON.stringify(res4) + '4');
          //至此登录完成
         }
        });
       }
      })
     }
    });
   }
  }
}
</script>
<style>

</style>
```

第28讲：自定义组件创建及使用

创建自定义组件

1、新建 组件.vue 文件

2、组件文档结构

```
<template name="组件名称">
    <view>
        ......
    </view>
</template>
<script>
export default {
    name: "组件名称",
    //属性
    props: {
        属性名称: {
            type: String,//属性类型
            value: "值"
        },
        ......
    },
    //组件生命周期
    created:function(e){

    },
    methods: {
        函数名称:function(obj){

        },
    }
}
</script>
<style>
**组件样式**
`</style>`
```

使用组件

1、引用组件

import 组件名称 from "../../components/组件名.vue";

2、注册组件

```
export default{
    components:{
        组件名称
    },
}
```

3、在试图模板中使用组件

<组件名称 组件属性="对应的值"> </组件名称>