



第三章 UNIAPP与VUE (三)

目录

- ◆ 静态聊天页面案例——静态模板
- ◆ 动态聊天页面案例——动态模板
- ◆ 动态聊天页面案例——methods

一、静态聊天页面案例——静态模板

■ chat.vue讲解的相关内容：

1. 设计注意事项
2. upx、rpx、vw、vh、px
3. CSS盒子模型
4. <template>的层级结构与页面显示
5. <template>代码
6. <Script>代码



1.1 设计注意事项

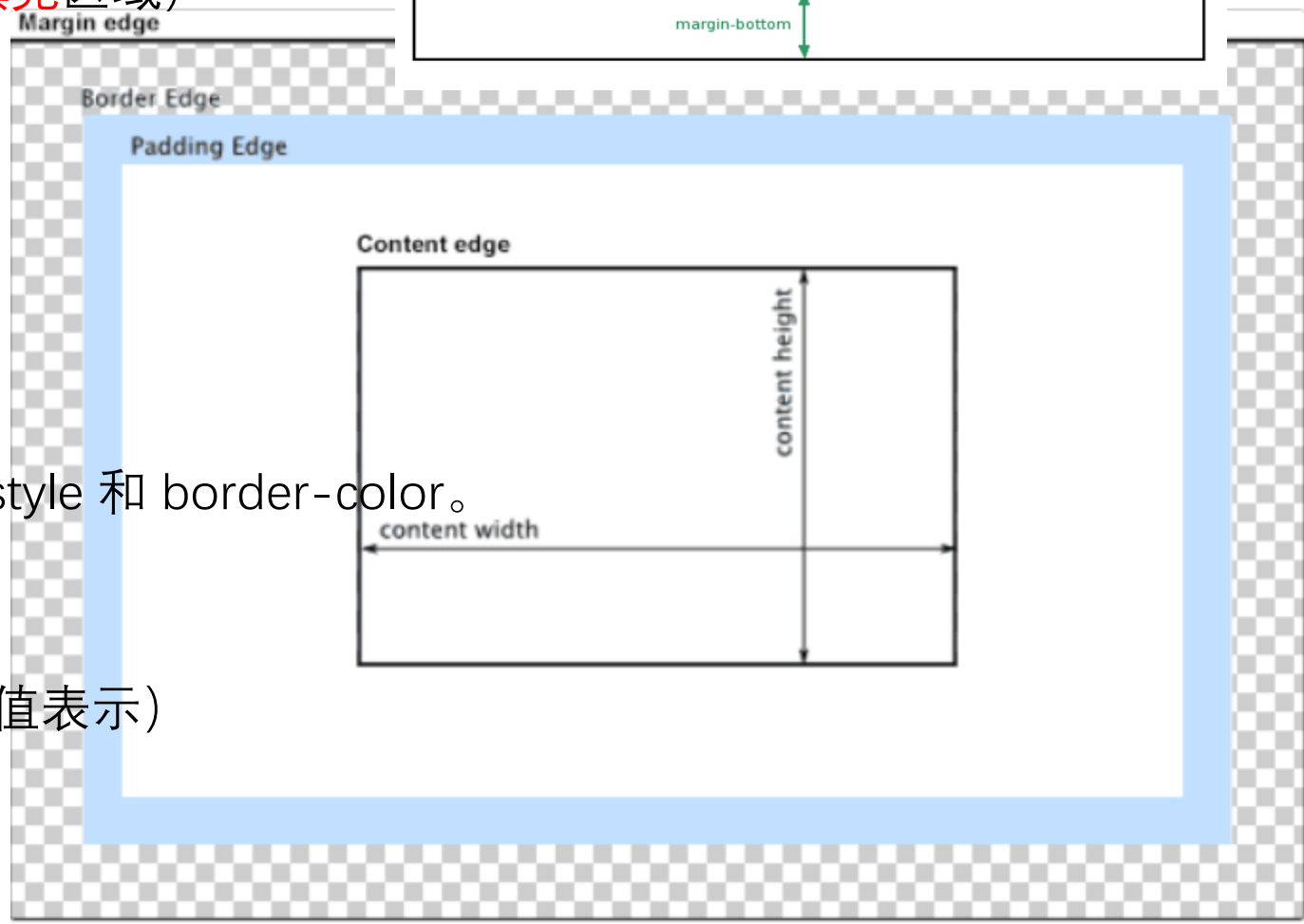
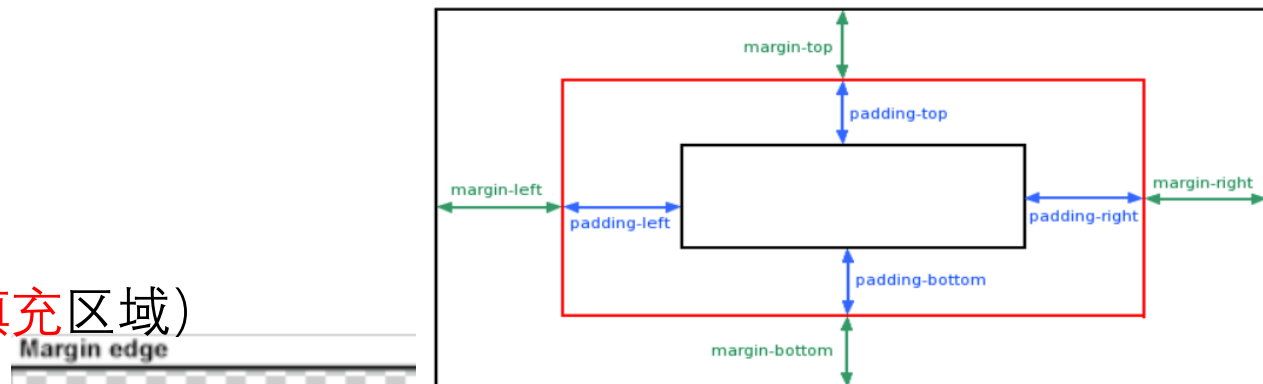
- 页面设计：确定好主要布局、要素，抽象出对象类型和设计风格等
- 建议编码时：先写 style，再写 template
- 注意空格的使用：如减号-前后要空格，“40upx”中间不能有空格

1.2 upx、rpx、vw、vh、px

- **.vw**: 1vw等于视口(ViewPort)宽度的1%, **100vw**等于窗口的宽度。**vh**: 1vh等于视口(ViewPort)高度的1%, **100vh**等于窗口的高度。**Vmin**、**Vmax**分别表示选取vw、vh中较小、较大值。
- **uni-app 使用 upx** 作为默认尺寸单位, upx 是相对于基准宽度的单位, 可以根据屏幕宽度进行自适应。uni-app 规定屏幕基准宽度750upx。
- 设计稿 1px / 设计稿基准宽度 = 框架样式 1upx / 750upx
- **rpx**单位是**微信小程序**中css的尺寸单位, rpx可以根据屏幕宽度进行自适应。规定屏幕宽为750rpx。
- upx与rpx与具体移动设备px再转换

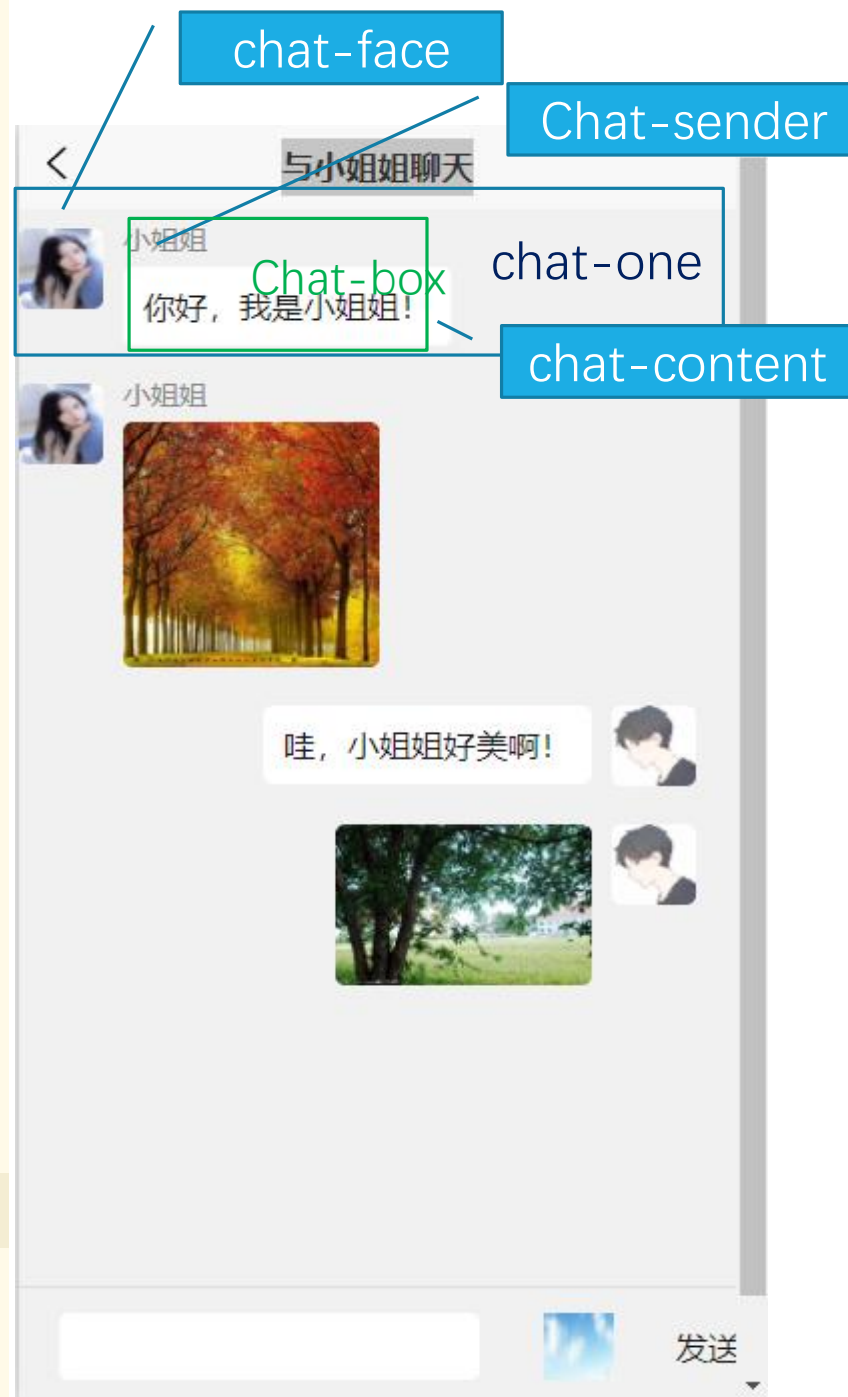
1.3 CSS盒子模型:每个元素一个盒子

- 内容区域content
- 内边距区域padding (元素背景颜色的填充区域)
 - ◆ 四个值:上右下左
 - ◆ 三个值: 上 (左右) 下
 - ◆ 两个值: (上下)(左右);padding:20upx 0;
 - ◆ 一个值:(上下左右)
- 边框区域border: border-width, border-style 和 border-color。
 - ◆ border:5px solid red;
- 外边界区域margin (类似padding的不同值表示)



1.4 template

- template
 - view.container
 - view.chat-body
 - view.chat-one
 - image.chat-face
 - view.chat-box
 - view.chat-sender
 - view.chat-content
 - view.chat-one
 - image.chat-face
 - view.chat-box
 - view.chat-sender
 - image.chat-img
 - view.chat-one.chat-one-mine
 - view.chat-box
 - view.chat-content
 - image.chat-face
 - view.chat-one.chat-one-mine
 - view.chat-box
 - view.chat-content
 - view.chat-footer



1.5 template(1)

```
1 <template>
2   <view class="container">
3     <view class="chat-body">
4       <view class="chat-one">
5         <image class="chat-face" src="/static/faces/1.jpeg" />
6         <view class="chat-box">
7           <view class="chat-sender"> 小姐姐</view>
8           <view class="chat-content">你好，我是小姐姐! </view>
9         </view>
10      </view>
11     <view class="chat-one">
12       <image class="chat-face" src="@/static/faces/1.jpeg" />
13       <view class="chat-box">
14         <view class="chat-sender"> 小姐姐</view>
15         <image class="chat-img" src="/static/imgs/1.jpeg" mode='widthFix' ></image>
16       </view>
17     </view>
```


template(2)

```
18 <view class="chat-one chat-one-mine">
19   <view class="chat-box">
20     <view class="chat-content"> 哇, 小姐姐好美啊! </view>
21   </view>
22   <image class="chat-face" src="/static/faces/2.jpeg" />
23 </view>
24 <view class="chat-one chat-one-mine">
25   <view class="chat-box">
26     <image class="chat-img" src="/static/imgs/2.jpeg" mode='widthFix' ></image>
27   </view>
28   <image class="chat-face" src="/static/faces/2.jpeg" />
29 </view>
30 </view>
31 <!-- 聊天输入 -->
32 <view class="chat-footer">
33   <input class="msg-input" type="text" cursor-spacing="16" />
34   <image class="img-chose" src="@/static/img.jpeg"/>
35   <view class="send-btn">发送</view>
36 </view>
37 </view>
38 </template>
```

1.6 style(1)

```
51 <style lang="scss" scoped>
52   .container{
53     background-color: #f1f1f1;
54     min-height: 100vh;
55   }
56   .chat-body{
57     padding-bottom: 178upx; //给footer
58     .chat-one{
59       display: flex; //弹性布局, 容器
60       flex-direction: row; //行为主
61       flex-wrap: wrap; //当所有item在
62       justify-content: flex-start; /
63       align-items: flex-start; //item
64       padding: 20upx 0;
65     }
66     .chat-one-mine{
67       justify-content: flex-end;
68     }
```

style(2)

```
93 | .chat-content{
94 |     background-color: #fff;
95 |     border-radius: 5px;
96 |     padding: 10px;
97 | }
98 | .chat-img{
99 |     float: left;
100 |     max-width: 60%;
101 |     border-radius: 5px;
102 | }
103 | .chat-one-mine .chat-img{
104 |     float:right;
105 | }
106 | }
```

```
69 | .chat-face{
70 |     width: 84upx;
71 |     height: 84upx;
72 |     border-radius:10upx;
73 |     margin-left: 40upx;
74 | }
75 | .chat-one-mine .chat-face{
76 |     margin-left: 0;
77 |     margin-right: 40upx;
78 | }
79 | .chat-box{
80 |     max-width: calc(100% - 290upx);
81 |     margin-left:20upx;
82 |     font-size: 30upx;
83 | }
84 | .chat-one-mine .chat-box{
85 |     margin-right: 20upx;
86 | }
87 | .chat-sender{
88 |     color: #888;
89 |     font-size: 28upx;
90 |     margin-top: -8upx; //元素内容上移,
91 |     margin-bottom: 10upx;
92 | }
```

style(3)

```
107  .chat-footer{
108      width: 680upx;
109      padding: 0 40upx;
110      height: 120upx;
111      position: fixed;
112      bottom: 0;
113      left: 0;
114      background-color: #f1f1f1;
115      display: flex;
116      flex-direction: row;
117      flex-wrap: wrap;
118      justify-content: space-between;
119      align-items: center;
120      align-content: center;
121      border-top: 1upx solid #ddd;
122  }
123      .msg-input{
124          background-color: #fff;
125          width: calc(100% - 300upx);
126          height: 70upx;
127          line-height: 70upx;
128          font-size: 30upx;
129          border-radius: 10upx;
130          padding: 0 20upx;
131      }
132      .img-chose{
133          height: 70upx;
134          width: 70upx;
135      }
136  </style>
```

二.动态聊天页面案例——动态模板

1. 任务概述
2. 指令及其参数
3. 常用指令：单向绑定 v-bind
4. 常用指令：双向绑定 v-model
5. 常用指令：v-on
6. 常用指令：条件渲染 v-if
7. 常用指令：列表渲染 v-for
8. 代码示例

2.1 任务概述

- 从静态到动态(chat.vue chat2.vue)聊天页面的更新任务
 1. 动态生成template模块的代码
 2. 提交新的聊天信息
- 把直接写在template中chat-body部分的静态组件view换成block，转移到script中data来定义。
- <view> 与 <block>的区别
 - ◆ <view> 是一个组件，会在页面上做渲染；
 - ◆ 小程序中的<block>不是一个组件，仅是一个包装元素，只接受控制属性，不会在页面中做任何渲染。同样地，Vue中的<template>的作用是一种模板占位符，可帮助我们包裹元素，但在循环过程当中，template不会被渲染到页面上。

从常用指令入手

列表渲染

```
<view v-for="(item,index) in chatList" :key="index" >
```

条件渲染

```
<image class="chat-img" v-if="item.type === 'img'" :src="item.content"
```

单向绑定v-bind

双向绑定v-model

```
<input class="msg-input" type="text" v-model="myInput">
```

v-on事件监视

```
<view class="send-btn" @click="sendMsg">发送</view>
```

2.2 Vue的模板语法之一 ——指令

指令

响应式指令

v-text

v-html

v-show

v-class

v-attr

v-style

v-on

v-model

v-if

v-repeat

字面指令

v-transition

v-ref

v-el

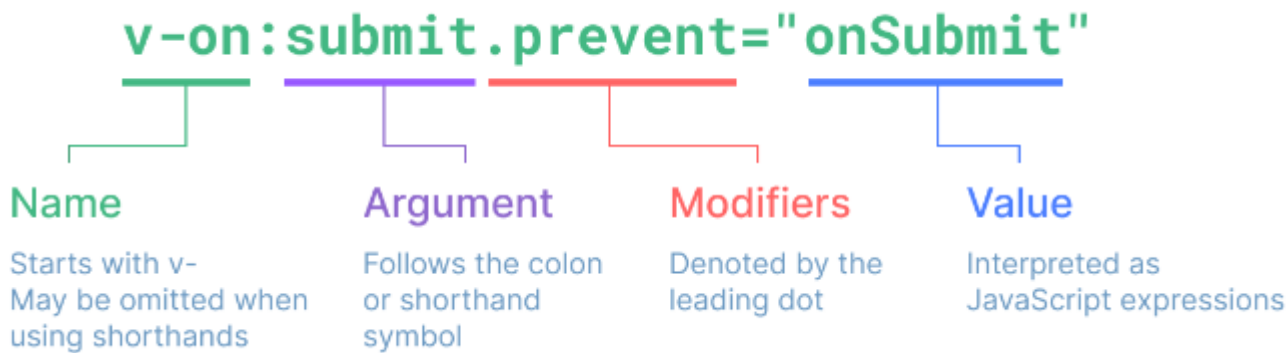
空指令

v-pre

v-cloak

- 指令(Directives)是带有 v- 前缀的特殊属性attribute。Vue 提供了许多内置指令 <https://cn.vuejs.org/guide/essentials/template-syntax.html#directives>
- 指令(除了 v-for、v-on 和 v-slot) 的期望值是一个 JavaScript 表达式。一个指令的任务是在其表达式的值变化时响应式地更新 DOM。
- 响应式指令：将自己与一个 Vue 实例上的属性绑定，或与一个实例作用域中的表达式绑定。当属性或表达式的值发生改变时，指令的 update() 方法会在下一个事件循环中被异步调用。

指令与指令参数



■ 指令包含 名字 参数 修饰符 值四个部分。

■ “参数” **Arguments**: 某些指令会需要一个“参数”，指令名: 参数

■ `<a v-bind:href="url"> ... ` `<a :href="url"> ... `

◆ v-bind: 缩写成:

■ `<a v-on:click="doSomething"> ... ` `<a @click="doSomething"> ... `

◆ v-on指令监听(键盘、鼠标、表单等) 事件, v-on:缩写成@, 参数是要监听的事件名称

◆ `<button v-on="{click:single,dblclick:double}">`单击n+1,双击m-1`</button>`

◆ `@click="demo"` 和 `@click="demo($event)"` 效果一致, 但后者可以传参;

■ 动态参数: 参数是一个 JavaScript 表达式 (写在方括号内), 该表达式必须是个“字符串”或null (null表示显式移除该绑定)

■ `<a @[eventName]="doSomething">`

指令的修饰符 Modifiers

- 修饰符是以点开头的特殊后缀，表明指令需要以一些特殊的方式被绑定。例如 .prevent 修饰符会告知 v-on 指令对触发的事件调用 `event.preventDefault()`：

2.3 常用指令：单向绑定 v-bind

`<image class="chat-img" v-if="item.type === 'img'" :src="item.content">`

条件渲染

单向绑定v-bind

- <https://cn.vuejs.org/api/built-in-directives.html#v-bind>
- 动态的绑定一个或多个 attribute，也可以是组件的 prop。缩写：: 或者 . (当使用 .prop 修饰符)
- `` ``
- 后续再关注 **Class 与 Style 绑定** <https://cn.vuejs.org/guide/essentials/class-and-style.html>


2.4 常用指令：双向绑定 v-model

双向绑定v-model

```
<input class="msg-input" type="text" v-model="myInput">
```

- v-model指令来实现表单元素和数据的双向绑定
- 表单提交要求可以在代码逻辑中获取到用户提交的数据，v-model指令可以在表单 input、textarea以及select元素上创建双向数据绑定；
- 它会根据控件类型自动选取正确的方法来更新元素；
- v-model 本质上是语法糖（对语言的功能没有影响，但是更方便程序员使用的语法）它负责监听用户的输入事件来更新数据

2.5 常用指令:v-on监听 DOM 事件



```
<view class="send-btn" @click="sendMsg">发送</view>
```

- **v-on 指令**将监听 DOM 事件，其参数是要监听的事件名称（下例是click事件）。v-on 缩写是 @ 字符

```
<a v-on:click="doSomething"> ... </a>
```

```
<a @click="doSomething"> ... </a>    <!-- 简写 -->
```

关于事件的内容，后续再展开学习

2.6 常用指令： 条件渲染v-if v-show

- **v-if** 指令用于条件性地渲染一块内容。该内容仅在指令表达式返回真值时才被渲染。
- **v-else**： 必须跟在一个 v-if 或者 v-else-if 元素后面， 否则它将不会被识别。
- **v-else-if**： 可以连续多次重复使用， 必须紧跟在一个 v-if 或一个 v-else-if 元素后面。
- **v-show**： 按条件显示一个元素的指令
- **v-if 与v-show的不同与选择**
 - ◆v-show： 元素无论初始条件如何， 始终会被渲染， 仅影响 CSS display 属性的切换。
 - ◆v-if 是惰性的： 若初次渲染时条件值 false， 则不做任何事。只有当条件首次变为 true 时才渲染。切换时， 条件区块内的事件监听器和子组件都会被销毁与重建。
 - ◆v-if 有更高的切换开销， 而 v-show 有更高的初始渲染开销。因此， 如果需要频繁切换， 则使用 v-show 较好； 如果在运行时绑定条件很少改变， 则 v-if 会更合适

条件渲染

■ finder.vue 示例

```
<div v-if="type === 'A'">
  A
</div>
<div v-else-if="type === 'B'">
  B
</div>
<div v-else-if="type === 'C'">
  C
</div>
<div v-else>
  Not A/B/C
</div>
```

双等号==是一种弱类型相等运算符，它比较两个值是否相等，如果两个值类型不同，会自动进行类型转换后再比较。三等号===是一种严格类型相等运算符，它比较两个值的类型和值是否都相等，如果类型不同或值不同，返回 false

2.7 常用指令:列表渲染 v-for

- <https://cn.vuejs.org/guide/essentials/list.html#v-for>
- v-for 指令基于一个数组或对象来渲染一个列表。v-for 支持嵌套
- 对于数组，它需要使用 item in items 形式的特殊语法，其中 items 是源数据的数组，而 item 是迭代项的别名。v-for 块中可以完整地访问父作用域内的属性和变量。可选的第二个参数表示当前项的位置索引（从0开始）。属性之间要添加逗号。
`v-for="(item,index) in items"`（可用 of 作为分隔符来替代 in）
- 渲染对象时，它会遍历对象的属性名。`v-for="(value,key,index) in object"`，其中value是对象的值，key是对象的键，index是索引，key,index 可选。

列表渲染 v-for (2)

```
<template v-for="todo in todos" :key="todo.name" >
  <li v-if="!todo.isComplete">
    {{ todo.name }}
  </li>
</template>
```

- **v-for里使用范围值：**该模板基于 1...n 的取值范围重复多次。

`<li v-for="n in 10">{{ n }}` (此处 n 的初值是从 1 开始而非 0)

- 同时使用 v-if 和 v-for 是不推荐的，因为这样二者的优先级不明显。若有需要，在外新包装一层 `<template>` 再在其上使用 v-for。
- 为了给 Vue 一个提示，以便它可以跟踪每个节点的标识，从而重用和重新排序现有的元素，需要为每个元素对应的块提供一个唯一的 key 属性，这里的 key 是通过 v-bind 绑定的特殊属性，不要和在 v-for 中使用对象里的对象属性名相混淆。

2.8 代码示例

- 添加chatList变量

```
<script>
export default {
  data() {
    return {
      chatList:[
        {
          isMe:false,
          type:'txt',
          content:'你好，我是小姐姐！'
        },
        {
          isMe:false,
          type:'img',
          content:'/static/imgs/1.jpg'
        },
        {
          isMe:true,
          type:'txt',
          content:'哇，小姐姐好美啊！'
        },
        {
          isMe:true,
          type:'img',
          content:'/static/imgs/2.jpg'
        }
      ]
    }
  }
}
```

```
<view class="chat-body">
<view v-for="(item,index) in chatList" :key="index" >
  <view class="chat-one" v-if="!item.isMe">
    <image class="chat-face" src="@/static/faces/1.jpeg" />
    <view class="chat-box">
      <view class="chat-sender"> 小姐姐</view>
      <view class="chat-content" v-if="item.type === 'txt'">{{item.content}}</view>
      <image class="chat-img" v-if="item.type === 'img'" :src="item.content" mode='widthFix' />
    </view>
  </view>
  <view v-else class="chat-one chat-one-mine">
    <view class="chat-box">
      <view class="chat-content" v-if="item.type === 'txt'"> {{item.content}}</view>
      <image class="chat-img" v-if="item.type === 'img'" :src="item.content" mode='widthFix'></image>
    </view>
    <image class="chat-face" src="@/static/faces/2.jpeg" />
  </view>
</view>
</view>
```

三、动态聊天页面案例——methods

```
<!-- 聊天输入 -->
<view class="chat-footer">
  <input class="msg-input" type="text" v-model="myInput" cursor-spacing="16" />
  <image class="img-chose" src="@/static/img.jpeg" @click="chooseImgAndSend"/>
  <view class="send-btn" @click="sendMsg">发送</view>
</view>
<!-- <input type="text" placeholder="请输入聊天内容 " /> -->
</view>
```

myInput: ""

// 页面周期函数

onShow() {

 if (!!uni.getStorageSync('chatList')){

 this.chatList = JSON.parse(uni.getStorageSync('chatList'))

 }

},

```
methods: {
  sendMsg(){
    if(!this.myInput) return ;
    let txtMsg = {
      isMe: true,
      type: "txt",
      content: this.myInput
    }
    this.chatList.push(txtMsg)
    uni.setStorageSync('chatList', JSON.stringify(this.chatList)) // API 数据缓存 setStorageSync
  },
  chooseImgAndSend(){
```

```

chooseImgAndSend(){
  uni.chooseImage({ //from API 媒体 图片
    count: 1, //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', 'camera'], //从相册选择
    // success: function (res) {
    //   console.log(JSON.stringify(res.tempFilePaths));
    // }
    success: res => {
      console.log(res.tempFilePaths[0])
      let sendMsg = {
        isMe: true,
        type: "img",
        content: res.tempFilePaths[0]
      }
      this.chatList.push(sendMsg)
      let resMsg = {
        isMe: false,
        type: "img",
        content: res.tempFilePaths[0]
      }
      this.chatList.push(resMsg)
      uni.pageScrollTo({ //from API >> 界面 >> 滚动
        scrollTop: 999999,
        duration: 0
      })
      uni.setStorageSync('chatList', JSON.stringify(this.chatList)) //API 数据缓存 setStorageSync
    }
  });
},

```

第三章作业1

■ 在完成 tarBar、subPackage与路由跳转功能基础上，完成动态聊天页面；要求：

1. 界面布局合理
2. 聊天功能不少于课程案例（至少含文本与图片）
3. 作业DL：10月30日前