



黑马程序员™  
www.itheima.com

传智播客旗下  
高端IT教育品牌

# Vue.js快速入门

深圳中心-前端与移动开发教研部



# 课程介绍

学前须知

开发工具

课程安排

1

HTML

2

CSS

3

JavaScript

4

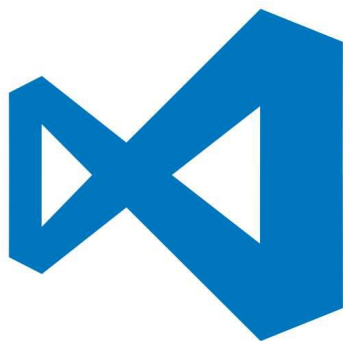
AJAX

# 课程介绍

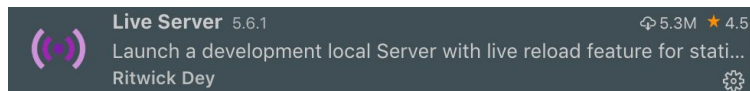
学前须知

开发工具

课程安排



VSCode



插件:浏览器实时预览

# 课程介绍

学前须知

开发工具

课程安排

1

Vue基础

2

本地应用

3

网络应用

4

综合应用

# 课程介绍

学前须知

开发工具

课程安排



# 课程介绍

- ◆ 掌握HTML,CSS,JavaScript,AJAX基础知识
- ◆ 选用VSCode作为课程中的开发工具
- ◆ 课程安排:基础→本地应用→网络应用→综合应用

# 课程安排

1

Vue基础

2

本地应用

3

网络应用

4

综合应用

# 课程安排



1

Vue基础

2

本地应用

3

网络应用

4

综合应用



1

## Vue基础



Vue简介

第一个Vue程序

el:挂载点

data:数据对象

1

## Vue基础



Vue简介

第一个Vue程序

el:挂载点

data:数据对象

## Vue简介



1. JavaScript框架
2. 简化Dom操作
3. 响应式数据驱动

## Vue简介

- ◆ JavaScript框架
- ◆ 简化Dom操作
- ◆ 响应式数据驱动

1

## Vue基础



Vue简介

第一个Vue程序

el:挂载点

data:数据对象



1

## Vue基础



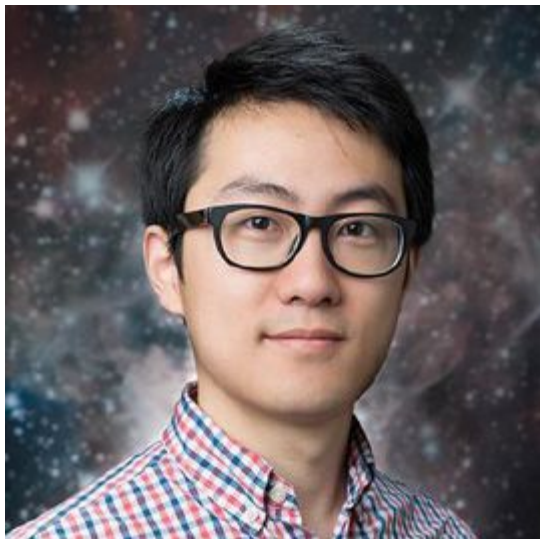
Vue简介

第一个Vue程序

el:挂载点

data:数据对象

## 第一个Vue程序



尤雨溪

文档传送门

<https://cn.vuejs.org>

## 第一个Vue程序

- ◆ 导入开发版本的Vue.js
- ◆ 创建Vue实例对象, 设置el属性和data属性
- ◆ 使用简洁的模板语法把数据渲染到页面上

1

## Vue基础



Vue简介

第一个Vue程序

el:挂载点

data:数据对象

1

## Vue基础



Vue简介

第一个Vue程序

el:挂载点

data:数据对象



el:挂载点

```
<div id="app">
  {{ message }}
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    message:"黑马程序员"
  }
})
```

el:挂载点

```
<div id="app">  
  黑马程序员  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    message:"黑马程序员"  
  }  
})
```

el:挂载点

## Vue实例的作用范围是什么呢？

Vue会管理el选项命中的元素及其内部的后代元素

## 是否可以使用其他的选择器？

可以使用其他的选择器,但是建议使用ID选择器

## 是否可以设置其他的dom元素呢？

可以使用其他的双标签,不能使用HTML和BODY



## el:挂载点

- ◆ el是用来设置Vue实例挂载（管理）的元素
- ◆ Vue会管理el选项命中的元素及其内部的后代元素
- ◆ 可以使用其他的选择器,但是建议使用ID选择器
- ◆ 可以使用其他的双标签,不能使用HTML和BODY

1

## Vue基础



Vue简介

第一个Vue程序

el:挂载点

data:数据对象

1

# Vue基础



Vue简介

第一个Vue程序

el:挂载点

data:数据对象



data:数据对象

```
<div id="app">
  {{ message }}
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    message:"黑马程序员
  },
})
```

## data:数据对象

```
<div id="app">
  {{ message }}
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    message:"黑马程序员",
    array:[],
    obj:{},
  }
})
```

## data:数据对象

- ◆ Vue中用到的数据定义在data中
- ◆ data中可以写复杂类型的数据
- ◆ 渲染复杂类型数据时,遵守js的语法即可

1

# Vue基础



Vue简介

第一个Vue程序

el:挂载点

data:数据对象



# 课程安排



1

Vue基础

2

本地应用

3

网络应用

4

综合应用



# 课程安排

1

Vue基础



2

本地应用

3

网络应用

4

综合应用



2

本地应用

## 用Vue开发网页效果



2

本地应用

# 获取元素,操纵他们



## 2

## 本地应用

v-text

v-html

v-on基础

v-for

v-on补充

v-model

v-show

v-if

v-bind



2

本地应用

# Vue指令

v-text

v-html

v-on基础

v-for

v-on补充

v-model

v-show

v-if

v-bind



2

## 本地应用

v-text

v-html

v-on基础

-

0

+



黑马程序员™ 深圳校区环境  
www.itheima.com



v-show

v-if

v-bind

v-for

v-on补充

v-model

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear completed





2

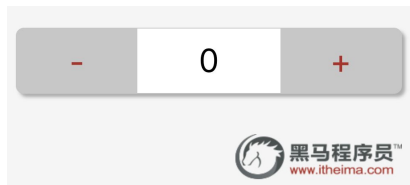
## 本地应用

### 1. 内容绑定,事件绑定

v-text

v-html

v-on基础



v-show

v-if

v-bind

### 2. 显示切换,属性绑定

### 3. 列表循环,表单元素绑定

v-for

v-on补充

v-model

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear completed



黑马程序员™ 深圳校区环境  
www.itheima.com





## 2

## 本地应用

- ◆ 通过Vue实现常见的网页效果
- ◆ 学习Vue指令,以案例巩固知识点
- ◆ Vue指令指的是,以v-开头的一组特殊语法





2

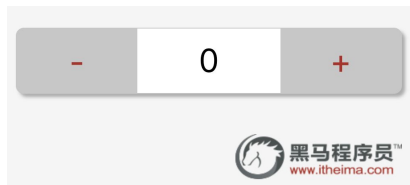
## 本地应用

### 1. 内容绑定,事件绑定

v-text

v-html

v-on基础



v-show

v-if

v-bind

### 2. 显示切换,属性绑定

### 3. 列表循环,表单元素绑定

v-for

v-on补充

v-model

## 小黑记事本

请输入任务

1. 吃饭饭

2. 睡觉觉

3. 写代码

3 items left

Clear completed



2

## 本地应用



v-text

v-html

v-on基础





# v-text

设置标签的文本值(textContent)



# v-text

设置标签的文本值(textContent)

```
<div id="app">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-text

设置标签的文本值(textContent)

```
<div id="app">  
  <h2></h2>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-text

设置标签的文本值(textContent)

```
<div id="app">  
  <h2 v-text=""></h2>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-text

设置标签的文本值(textContent)

```
<div id="app">  
  <h2 v-  
text="message"></h2>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-text

设置标签的文本值(textContent)

```
<div id="app">  
  <h2 v-  
text="message"></h2>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    message:"黑马程  
序员"  
  }  
})
```





# v-text

设置标签的文本值(textContent)

```
<div id="app">
  <h2 v-
text="message"></h2>
  <h2>深圳
  {{ message }}</h2>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    message:"黑马程
    序员"
  }
})
```



# v-text

设置标签的文本值(textContent)

```
<div id="app">  
  <h2 v-  
text="message+'!'"></h2>  
  <h2>深圳{{ message + "  
"}}</h2>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    message:"黑马程  
序员"  
  }  
})
```



## v-text

- ◆ **v-text**指令的作用是:设置标签的内容(textContent)
- ◆ 默认写法会替换全部内容,使用**差值表达式{{}}**可以替换指定内容
- ◆ 内部支持写**表达式**

2

## 本地应用



v-text

v-html

v-on基础



2

## 本地应用



v-text

v-html

v-on基础





# v-html

设置标签的innerHTML

## v-text v-html



# v-html

设置标签的innerHTML

```
<div id="app">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-html

设置标签的innerHTML

```
<div id="app">  
  <p></p>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```





# v-html

设置标签的innerHTML

```
<div id="app">  
  <p></p>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    content:"黑马程序员"  
  }  
})
```



# v-html

设置标签的innerHTML

```
<div id="app">  
  <p v-  
html="content"></p>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    content:"黑马程序员"  
  }  
})
```



# v-html

设置标签的innerHTML

```
<div id="app">  
  <p v-  
html="content"></p>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    // content:"黑马程序员"  
    content:"<a href='#>黑马程  
序员</a>"  
  }  
})
```



## v-html

- ◆ **v-html**指令的作用是:设置元素的**innerHTML**
- ◆ 内容中有**html**结构会被解析为**标签**
- ◆ **v-text**指令无论内容是什么,只会解析为**文本**
- ◆ 解析文本使用**v-text**,需要解析**html**结构使用**v-html**

2

## 本地应用



v-text

v-html

v-on基础



2

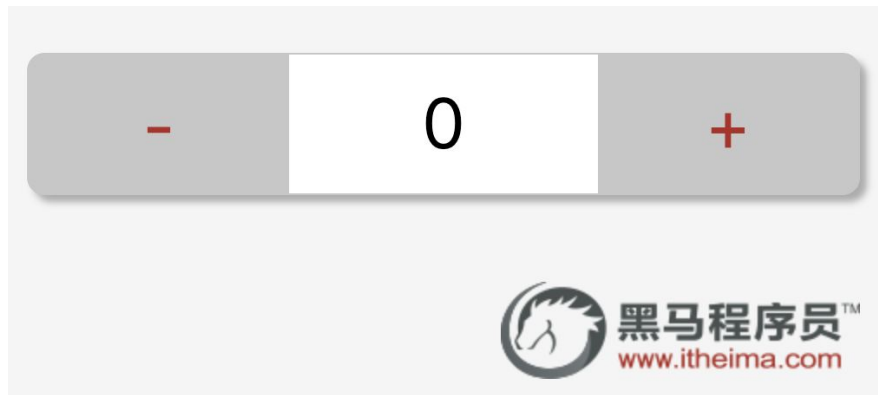
## 本地应用



v-text

v-html

v-on基础





# v-on基础

为元素绑定事件



# v-on基础

为元素绑定事件

```
<div id="app">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
})
```





# v-on基础

为元素绑定事件

```
<div id="app">  
  <input type="button" value="事件绑定" >  
</div>
```

```
var app = new Vue({  
  el:"#app",  
})
```



# v-on基础

为元素绑定事件

```
<div id="app">  
  <input type="button" value="事件绑定" v-on:事件名  
="方法">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
})
```



# v-on基础

为元素绑定事件

```
<div id="app">  
  <input type="button" value="事件绑定" v-  
on:click="方法">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
})
```



# v-on基础

为元素绑定事件

```
<div id="app">  
  <input type="button" value="事件绑定" v-on:click="方法">  
  <input type="button" value="事件绑定" v-on:mouseenter="方法">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
})
```



# v-on基础

为元素绑定事件

```
<div id="app">
  <input type="button" value="事件绑定" v-on:click="方法">
</div>
<input type="button" value="事件绑定" v-on:mouseenter="方法">
<input type="button" value="事件绑定" v-on:dblclick="方法">
</div>
```

```
var app = new Vue({
  el:"#app",
})
```



# v-on基础

为元素绑定事件

```
<div id="app">
  <input type="button" value="事件绑定" v-on:click="方法">
</div>
<input type="button" value="事件绑定" v-on:mouseenter="方法">
<input type="button" value="事件绑定" v-on:dblclick="方法">
</div>
```

```
var app = new Vue({
  el:"#app",
  methods:{
  }
})
```



# v-on基础

为元素绑定事件

```
<div id="app">
  <input type="button" value="事件绑定" v-on:click="方法">
</div>
<input type="button" value="事件绑定" v-on:mouseenter="方法">
<input type="button" value="事件绑定" v-on:dblclick="方法">
</div>
```

```
var app = new Vue({
  el:"#app",
  methods:{
    dolt:function(){
      // 逻辑
    }
  }
})
```



# v-on基础

为元素绑定事件

```
<div id="app">
  <input type="button" value="事件绑定" v-
on:click="dolt">
  <input type="button" value="事件绑定" v-
on:mouseenter="dolt">
  <input type="button" value="事件绑定" v-
on:dblclick="dolt">
</div>
```

```
var app = new Vue({
  el:"#app",
  methods:{
    dolt:function(){
      // 逻辑
    }
  }
})
```





# v-on基础

为元素绑定事件

```
<div id="app">
  <input type="button" value="事件绑定" v-
on:click="dolt">
  <input type="button" value="事件绑定" v-
on:mouseenter="dolt">
  <input type="button" value="事件绑定" v-
on:dblclick="dolt">
  <input type="button" value="事件绑定
" @dblclick="dolt">
</div>
```

```
var app = new Vue({
  el:"#app",
  methods:{
    dolt:function(){
      // 逻辑
    }
  }
})
```



## v-on

- ◆ v-on指令的作用是:为元素绑定事件
- ◆ 事件名不需要写on
- ◆ 指令可以简写为@
- ◆ 绑定的方法定义在methods属性中
- ◆ 方法内部通过this关键字可以访问定义在data中数据

2

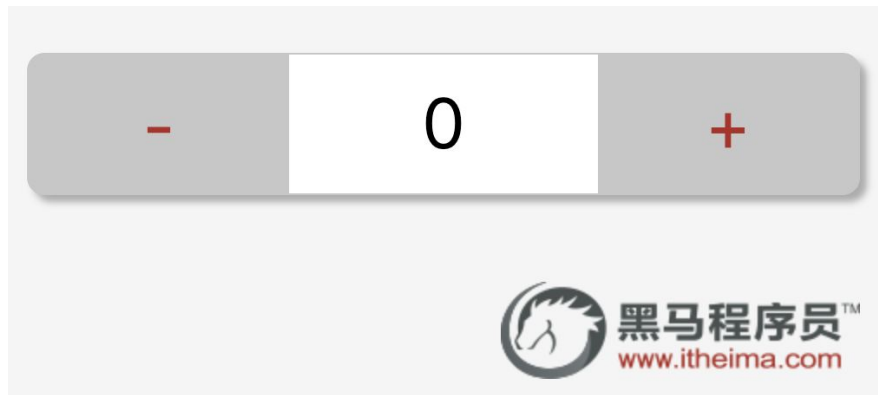
## 本地应用



v-text

v-html

v-on基础



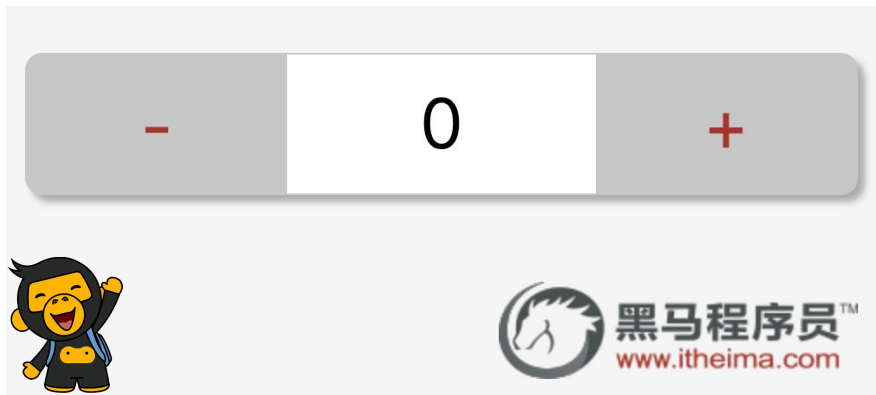
## 2

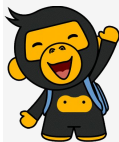
## 本地应用

v-text

v-html

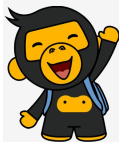
v-on基础





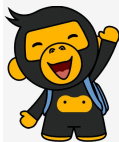
```
<div class="input-num">  
  <button>-</button>  
  <span></span>  
  <button>+</button>  
</div>
```

```
var app = new Vue({  
  el: "#app",  
});
```



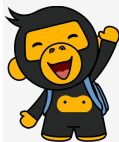
```
<div class="input-num">  
  <button @click>-</button>  
  <span></span>  
  <button @click>+</button>  
</div>
```

```
var app = new Vue({  
  el: "#app",  
});
```



```
<div class="input-num">  
  <button @click>-</button>  
  <span></span>  
  <button @click>+</button>  
</div>
```

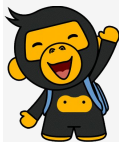
```
var app = new Vue({  
  el: "#app",  
  methods: {  
  }  
});
```



```
<div class="input-num">  
  <button @click>-</button>  
  <span>{{{</span>  
  <button @click>+</button>  
</div>
```

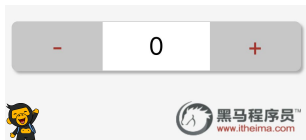
```
var app = new Vue({  
  el: "#app",  
  methods: {  
  }  
});
```





```
<div class="input-num">  
  <button @click>-</button>  
  <span>{{{</span>  
  <button @click>+</button>  
</div>
```

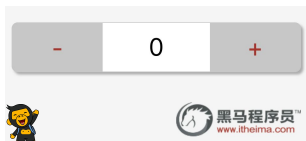
```
var app = new Vue({  
  el: "#app",  
  data: {  
  },  
  methods: {  
  }  
});
```



```
var app = new Vue({  
  el: "#app",  
  data: {  
  },  
  methods: {  
  }  
});
```

1. **data**中定义数据:比如**num**

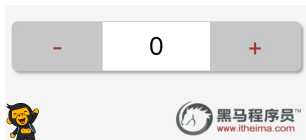
```
<div class="input-num">  
  <button @click>-  
</button>  
  <span>{{{</span>  
  <button @click>+</button>  
>  
</div>
```



```
var app = new Vue({  
  el: "#app",  
  data: {  
    num: 1  
  },  
  methods: {  
  }  
});
```

1. **data**中定义数据:比如**num**

```
<div class="input-num">  
  <button @click>-  
</button>  
  <span>{{{</span>  
  <button @click>+</button>  
>  
</div>
```

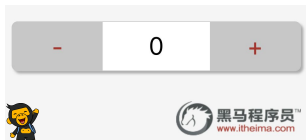


```
var app = new Vue({  
  el: "#app",  
  data: {  
    num: 1  
  },  
  methods: {  
  }  
});
```

1. **data**中定义数据:比如**num**

2. **methods**中添加两个方法:比如**add**(递增),**sub**(递减)

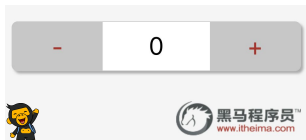
```
<div class="input-num">  
  <button @click>-  
</button>  
  <span>{{{</span>  
  <button @click>+</button>  
>  
</div>
```



```
var app = new Vue({  
  el: "#app",  
  data: {  
    num: 1  
  },  
  methods: {  
    add: function() {},  
    sub: function() {}  
  }  
});
```

```
<div class="input-num">  
  <button @click>-  
</button>  
  <span>{{{</span>  
  <button @click>+</button>  
>  
</div>
```

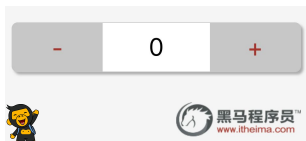
1. **data**中定义数据:比如**num**
2. **methods**中添加两个方法:比如**add**(递增),**sub**(递减)



```
var app = new Vue({  
  el: "#app",  
  data: {  
    num: 1  
  },  
  methods: {  
    add: function() {},  
    sub: function() {}  
  }  
});
```

```
<div class="input-num">  
  <button @click>-  
</button>  
  <span>{{{</span>  
  <button @click>+</button>  
>  
</div>
```

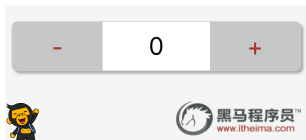
1. **data**中定义数据:比如**num**
2. **methods**中添加两个方法:比如**add**(递增),**sub**(递减)
3. 使用**v-text**将**num**设置给**span**标签



```
var app = new Vue({  
  el: "#app",  
  data: {  
    num: 1  
  },  
  methods: {  
    add: function() {},  
    sub: function() {}  
  }  
});
```

```
<div class="input-num">  
  <button @click>-  
</button>  
  <span>{{ num }}</span>  
  <button @click>+</button>  
>  
</div>
```

1. **data**中定义数据:比如**num**
2. **methods**中添加两个方法:比如**add**(递增),**sub**(递减)
3. 使用**v-text**将**num**设置给**span**标签

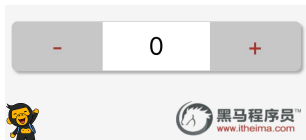


```
var app = new Vue({  
  el: "#app",  
  data: {  
    num: 1  
  },  
  methods: {  
    add: function() {},  
    sub: function() {}  
  }  
});
```

```
<div class="input-num">  
  <button @click>-  
</button>  
  <span>{{ num }}</span>  
  <button @click>+</button>  
>  
</div>
```

1. **data**中定义数据:比如**num**
2. **methods**中添加两个方法:比如**add**(递增),**sub**(递减)
3. 使用**v-text**将**num**设置给**span**标签
4. 使用**v-on**将**add,sub**分别绑定给**+, -**按钮

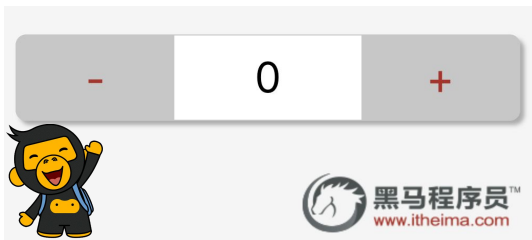




```
var app = new Vue({  
  el: "#app",  
  data: {  
    num: 1  
  },  
  methods: {  
    add: function() {},  
    sub: function() {}  
  }  
});
```

```
<div class="input-num">  
  <button @click="sub">-  
</button>  
  <span>{{ num }}</span>  
  <button @click="add">+</  
button>  
</div>
```

1. **data**中定义数据:比如**num**
2. **methods**中添加两个方法:比如**add**(递增),**sub**(递减)
3. 使用**v-text**将**num**设置给**span**标签
4. 使用**v-on**将**add,sub**分别绑定给**+, -**按钮
5. 累加的逻辑:小于**10**累加,否则提示
6. 递减的逻辑:大于**0**递减,否则提示



- ◆ 创建Vue示例时:**el**(挂载点),**data**(数据),**methods**(方法)
- ◆ **v-on**指令的作用是绑定事件,简写为**@**
- ◆ 方法中通过**this**,关键字获取**data**中的数据
- ◆ **v-text**指令的作用是:设置元素的**文本值**,简写为**{{}}**
- ◆ **v-html**指令的作用是:设置元素的**innerHTML**

## 2

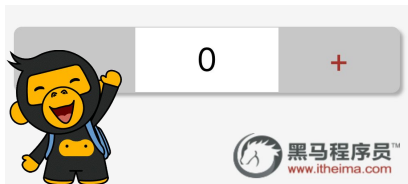
## 本地应用

### 1. 内容绑定,事件绑定

v-text

v-html

v-on基础



v-show

v-if

v-bind

### 2. 显示切换,属性绑定

### 3. 列表循环,表单元素绑定

v-for

v-on补充

v-model



## 小黑记事本

请输入任务

1. 吃饭饭

2. 睡觉觉

3. 写代码

3 items left

Clear completed

## 2

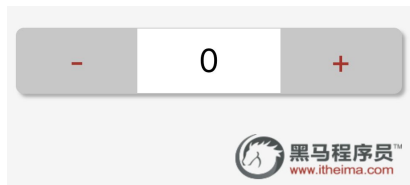
## 本地应用

### 1. 内容绑定,事件绑定

v-text

v-html

v-on基础



v-show

v-if

v-bind

### 2. 显示切换,属性绑定

### 3. 列表循环,表单元素绑定

v-for

v-on补充

v-model



2

## 本地应用



v-show

v-if

v-bind



黑马程序员 深圳校区环境





# v-show

根据表达值的真假,切换元素的显示和隐藏

## 广告



# v-show

根据表达值的真假,切换元素的显示和隐藏

## 遮罩层



# v-show

根据表达值的真假,切换元素的显示和隐藏

```
<div id="app">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```





# v-show

根据表达值的真假,切换元素的显示和隐藏

```
<div id="app">  
    
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-show

根据表达值的真假,切换元素的显示和隐藏

```
<div id="app">  
    
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-show

根据表达值的真假,切换元素的显示和隐藏

```
<div id="app">  
    
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-show

根据表达值的真假,切换元素的显示和隐藏

```
<div id="app">  
    
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    isShow:true  
  }  
})
```



# v-show

根据表达值的真假,切换元素的显示和隐藏

```
<div id="app">  
    
    
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    isShow:true  
  }  
})
```



# v-show

根据表达值的真假,切换元素的显示和隐藏

```
<div id="app">  
    
    
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    isShow:false  
  }  
})
```



# v-show

根据表达值的真假,切换元素的显示和隐藏

```
<div id="app">  
    
    
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    isShow:false,  
    age:16  
  }  
})
```



# v-show

根据表达值的真假,切换元素的显示和隐藏

```
<div id="app">  
    
    
  =18">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    isShow:false,  
    age:16  
  }  
})
```





## v-show

- ◆ **v-show**指令的作用是:根据真假切换元素的显示状态
- ◆ 原理是修改元素的display,实现显示隐藏
- ◆ 指令后面的内容,最终都会解析为**布尔值**
- ◆ 值为**true**元素显示,值为**false**元素隐藏
- ◆ 数据改变之后,对应元素的显示状态会**同步更新**

2

本地应用



v-show

v-if

v-bind



黑马程序员 深圳校区环境



2

## 本地应用



v-show

v-if

v-bind

黑马程序员  
www.itheima.com 深圳校区环境





# v-if

根据表达值的真假,切换元素的显示和隐藏



# v-if

根据表达值的真假,切换元素的显示和隐藏(操纵dom元素)



# v-if

根据表达值的真假,切换元素的显示和隐藏(操纵dom元素)

```
<div id="app">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



## v-if

根据表达值的真假,切换元素的显示和隐藏(操纵dom元素)

```
<div id="app">  
  <p>我是一个p标签</p>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



## v-if

根据表达值的真假,切换元素的显示和隐藏(操纵dom元素)

```
<div id="app">  
  <p v-if="true">我是一个p标签  
</p>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```





## v-if

根据表达值的真假,切换元素的显示和隐藏(操纵dom元素)

```
<div id="app">  
  <p v-if="true">我是一个p标签  
</p>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    isShow:false  
  }  
})
```



## v-if

根据表达值的真假,切换元素的显示和隐藏(操纵dom元素)

```
<div id="app">
  <p v-if="true">我是一个p标签
</p>
  <p v-if="isShow">我是一个p标签
</p>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    isShow:false
  }
})
```



## v-if

根据表达值的真假,切换元素的显示和隐藏(操纵dom元素)

```
<div id="app">
  <p v-if="true">我是一个p标签
</p>
  <p v-if="isShow">我是一个p标签
</p>
  <p v-if="表达式">我是一个p标签
</p>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    isShow:false
  }
})
```



## v-if

- ◆ **v-if**指令的作用是:根据表达式的真假切换元素的显示状态
- ◆ 本质是通过操纵**dom**元素来切换显示状态
- ◆ 表达式的值为**true**,元素存在于**dom**树中,为**false**,从**dom**树中移除
- ◆ 频繁的切换**v-show**,反之使用**v-if**,前者的切换消耗小

2

## 本地应用



v-show

v-if

v-bind



黑马程序员 深圳校区环境



2

## 本地应用



v-show

v-if

v-bind



黑马程序员  
www.itheima.com

深圳校区环境





# v-bind

设置元素的属性



黑马程序员  
www.itheima.com

传智播客旗下高端IT教育品牌

# v-bind

设置元素的属性(比如:src,title,class)





# v-bind

设置元素的属性(比如:src,title,class)

```
<div id="app">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-bind

设置元素的属性(比如:src,title,class)

## v-bind:属性名=表达式

```
<div id="app">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-bind

设置元素的属性(比如:src,title,class)

## v-bind:属性名=表达式

```
<div id="app">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-bind

设置元素的属性(比如:src,title,class)

```
<div id="app">  
    
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-bind

设置元素的属性(比如:src,title,class)

```
<div id="app">  
    
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    imgSrc:"图片地址",  
  }  
})
```



# v-bind

设置元素的属性(比如:src,title,class)

```
<div id="app">  
  <img v-bind:src= "imgSrc" >  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    imgSrc:"图片地址",  
  }  
})
```



# v-bind

设置元素的属性(比如:src,title,class)

```
<div id="app">  
  <img v-bind:src= "imgSrc" >  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    imgSrc:"图片地址",  
    imgTitle:"黑马程序员"  
  },  
})
```



# v-bind

设置元素的属性(比如:src,title,class)

```
<div id="app">
  <img v-bind:src= "imgSrc" >
  <img v-
bind:title="imgtitle+'!!!!'">
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgSrc:"图片地址",
    imgTitle:"黑马程序员
  },
})
```





# v-bind

设置元素的属性(比如:src,title,class)

```
<div id="app">
  <img v-bind:src= "imgSrc" >
  <img v-
bind:title="imgtitle+'!!!">
  <img class="active" >
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgSrc:"图片地址",
    imgTitle:"黑马程序员
  }
})
```



# v-bind

设置元素的属性(比如:src,title,class)

```
<div id="app">
  <img v-bind:src= "imgSrc" >
  <img v-
bind:title="imgtitle+'!!!!'">
  <img class="active" >
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgSrc:"图片地址",
    imgTitle:"黑马程序员
",
    isActive:false
  }
})
```



# v-bind

设置元素的属性(比如:src,title,class)

```
<div id="app">
  <img v-bind:src= "imgSrc" >
  <img v-
bind:title="imgtitle+'!!!">
  <img v-
bind:class="isActive?'active':"">
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgSrc:"图片地址",
    imgTitle:"黑马程序员
",
    isActive:false
  }
})
```



# v-bind

设置元素的属性(比如:src,title,class)

```
<div id="app">
  <img v-bind:src= "imgSrc" >
  <img v-
bind:title="imgtitle+'!!!">
  <img v-
bind:class="isActive?'active':"">
  <img v-
bind:class="{active:isActive}">
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgSrc:"图片地址",
    imgTitle:"黑马程序员
",
    isActive:false
  }
})
```



# v-bind

设置元素的属性(比如:src,title,class)

```
<div id="app">
  <img :src= "imgSrc" >
  <img :title="imgtitle+'!!!">
  <img :class="isActive?'active':" ">
  <img :class="{active:isActive}">
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgSrc:"图片地址",
    imgTitle:"黑马程序员
  },
  isActive:false
})
```



# v-bind

- ◆ **v-bind**指令的作用是:为元素绑定属性
- ◆ 完整写法是 **v-bind:属性名**
- ◆ 简写的话可以直接省略**v-bind**,只保留 **:属性名**
- ◆ 需要动态的增删**class**建议使用对象的方式

2

## 本地应用



v-show

v-if

v-bind

黑马程序员  
www.itheima.com

深圳创维校区环境



2

## 本地应用

v-show

v-if

v-bind







# 图片切换



图片数组

索引

v-bind

v-on

v-if

v-show

```
<div id="#app">  
    
  <a href="#">上一张  
</a>  
  <a href="#">下一张  
</a>  
</div>
```



# 图片切换



图片数组

索引

v-bind

v-on

v-if

v-show

```
<div id="#app">  
    
  <a href="#">上一张  
</a>  
  <a href="#">下一张  
</a>  
</div>
```



# 图片切换

图片数组

索引

v-bind

v-on

v-show

```
<div id="#app">
  
  <a href="#">上一张
</a>
  <a href="#">下一张
</a>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
  }
})
```



# 图片切换

## 1. 定义图片数组

索引

v-bind

v-on

v-show

```
<div id="#app">
  
  <a href="#">上一张
</a>
  <a href="#">下一张
</a>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
  }
})
```



# 图片切换

## 1. 定义图片数组

索引

v-bind

v-on

v-show

```
<div id="#app">
  
  <a href="#">上一张
</a>
  <a href="#">下一张
</a>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgArr:[]
  }
})
```



# 图片切换

## 1. 定义图片数组

## 2. 添加图片索引

v-bind

v-on

v-show

```
<div id="#app">
  
  <a href="#">上一张
</a>
  <a href="#">下一张
</a>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgArr:[]
  }
})
```



# 图片切换

## 1. 定义图片数组

## 2. 添加图片索引

v-bind

v-on

v-show

```
<div id="#app">
  
  <a href="#">上一张
</a>
  <a href="#">下一张
</a>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgArr:[],
    index:0
  }
})
```



# 图片切换

1.定义图片数组

2.添加图片索引

3.绑定src属性

v-on

v-show

```
<div id="#app">
  
  <a href="#">上一张
</a>
  <a href="#">下一张
</a>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgArr:[],
    index:0
  }
})
```





# 图片切换

1.定义图片数组

2.添加图片索引

3.绑定src属性

v-on

v-show

```
<div id="#app">
  
  <a href="#">上一张
</a>
  <a href="#">下一张
</a>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgArr:[],
    index:0
  }
})
```



# 图片切换

1.定义图片数组

2.添加图片索引

3.绑定src属性

4.图片切换逻辑

v-show

```
<div id="#app">
  
  <a href="#">上一张
</a>
  <a href="#">下一张
</a>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgArr:[],
    index:0
  }
})
```



# 图片切换

1.定义图片数组

2.添加图片索引

3.绑定src属性

4.图片切换逻辑

v-show

```
<div id="#app">
  
  <a href="#">上一张
</a>
  <a href="#">下一张
</a>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgArr:[],
    index:0
  },
  methods:{
    prev:function(){},
    next:function(){
    }
  }
})
```



# 图片切换

## 1.定义图片数组

## 2.添加图片索引

## 3.绑定src属性

## 4.图片切换逻辑

```
<div id="#app">
  
  <a href="#" @click="prev">上
    一张</a>
  <a href="#" @click="next">下
    一张</a>
</div>
```

v-show

```
var app = new Vue({
  el:"#app",
  data:{
    imgArr:[],
    index:0
  },
  methods:{
    prev:function(){},
    next:function(){
    }
  }
})
```



# 图片切换

## 1.定义图片数组

## 2.添加图片索引

## 3.绑定src属性

## 4.图片切换逻辑

## 5.显示状态切换

```
<div id="#app">
  
  <a href="#" @click="prev" v-show="条件">
    上一张</a>
  <a href="#" @click="next" v-show="条件">
    下一张</a>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    imgArr:[],
    index:0
  },
  methods:{
    prev:function(){},
    next:function(){
    }
  }
})
```



## 图片切换

- ◆ 列表数据使用数组保存
- ◆ `v-bind`指令可以设置元素属性,比如`src`
- ◆ `v-show`和`v-if`都可以切换元素的显示状态,频繁切换用`v-show`

2

## 本地应用

v-show

v-if

v-bind



深圳校区环境



## 2

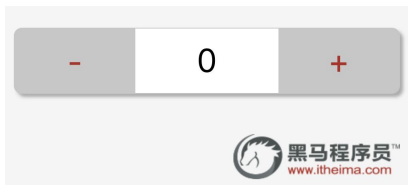
## 本地应用

### 1. 内容绑定,事件绑定

v-text

v-html

v-on基础



v-show

v-if

v-bind

### 2. 显示切换,属性绑定

### 3. 列表循环,表单元素绑定

v-for

v-on补充

v-model





## 2

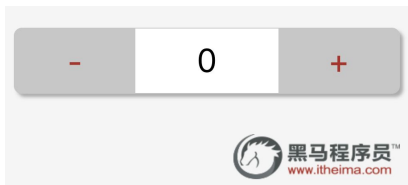
## 本地应用

### 1. 内容绑定,事件绑定

v-text

v-html

v-on基础



v-show

v-if

v-bind

### 2. 显示切换,属性绑定



### 3. 列表循环,表单元素绑定

v-for

v-on补充

v-model

## 小黑记事本

请输入任务

1. 吃饭饭

2. 睡觉觉

3. 写代码

3 items left

Clear completed

2

## 本地应用



v-for

v-on补充

v-model

### 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear completed



黑马程序员™  
www.itheima.com



# v-for

根据数据生成列表结构



# v-for

根据数据生成列表结构

```
<div id="app">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
  }  
})
```



# v-for

根据数据生成列表结构

```
<div id="app">  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    arr:[1,2,3,4,5]  
  }  
})
```



# v-for

根据数据生成列表结构

```
<div id="app">  
  <ul>  
    <li></li>  
  </ul>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    arr:[1,2,3,4,5]  
  }  
})
```



# v-for

根据数据生成列表结构

```
<div id="app">  
  <ul>  
    <li v-for="item in arr"></li>  
  </ul>  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    arr:[1,2,3,4,5]  
  }  
})
```



# v-for

根据数据生成列表结构

```
<div id="app">
  <ul>
    <li v-for="item in arr">
      你好
    </li>
  </ul>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    arr:[1,2,3,4,5]
  }
})
```





# v-for

根据数据生成列表结构

```
<div id="app">
  <ul>
    <li v-
for="item in arr" :title="item">
      {{ item }}
    </li>
  </ul>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    arr:[1,2,3,4,5]
  }
})
```



# v-for

根据数据生成列表结构

```
<div id="app">
  <ul>
    <li v-
for="(item,index) in arr" :title="item">
      {{ item }}
    </li>
  </ul>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    arr:[1,2,3,4,5]
  }
})
```



# v-for

根据数据生成列表结构

```
<div id="app">
  <ul>
    <li v-
for="(item,index) in arr" :title="item">
      {{ index }}{{ item }}
    </li>
  </ul>
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    arr:[1,2,3,4,5]
  }
})
```



# v-for

根据数据生成列表结构

```
<div id="app">
  <ul>
    <li v-
for="(item,index) in arr" :title="item">
      {{ index }}{{ item }}
    </li>
  </ul>
</div>
```

```
var app = new Vue({
  el: "#app",
  data: {
    arr: [1, 2, 3, 4, 5],
    objArr: [
      { name: "jack" },
      { name: "rose" }
    ]
  }
})
```



# v-for

根据数据生成列表结构

```
<div id="app">
  <ul>
    <li v-
for="(item,index) in arr" :title="item">
      {{ index }}{{ item }}
    </li>
    <li v-for="(item,index) in objArr">
      {{ item.name }}
    </li>
  </ul>
</div>
```

```
var app = new Vue({
  el: "#app",
  data: {
    arr: [1, 2, 3, 4, 5],
    objArr: [
      { name: "jack" },
      { name: "rose" }
    ]
  }
})
```



# v-for

- ◆ **v-for**指令的作用是:根据数据生成列表结构
- ◆ 数组经常和**v-for**结合使用
- ◆ 语法是( **item,index** ) in 数据
- ◆ item 和 index 可以结合其他指令一起使用
- ◆ 数组长度的更新会同步到页面上,是响应式的

2

## 本地应用



v-for

v-on补充

v-model

### 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear completed



黑马程序员™  
www.itheima.com

2

本地应用



v-for

v-on补充

v-model

小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear completed



黑马程序员™  
www.itheima.com





# v-on补充

传递自定义参数,事件修饰符



## v-on补充

传递自定义参数,事件修饰符

```
<div id="app">  
  <input type="button" @click="do  
It" />  
</div>
```

```
var app = new Vue({  
  el: "#app",  
  methods: {  
    dolt: function() {},  
  }  
})
```



## v-on补充

传递自定义参数,事件修饰符

```
<div id="app">  
  <input type="button" @click="dolt(p1)" />  
</div>
```

```
var app = new Vue({  
  el: "#app",  
  methods: {  
    dolt: function(p1) {}  
  },  
});
```



## v-on补充

传递自定义参数,事件修饰符

```
<div id="app">  
  <input type="button" @click="dolt(p1,  
p2)" />  
</div>
```

```
var app = new Vue({  
  el: "#app",  
  methods: {  
    dolt: function(p1,p2  
  ) {},  
  }  
})
```



## v-on补充

传递自定义参数,事件修饰符

```
<div id="app">
  <input type="button" @click="dolt(p1,
p2)" />
  <input type="text" @keyup="sayHi">
</div>
```

```
var app = new Vue({
  el: "#app",
  methods: {
    dolt: function(p1,p2
) {},
    sayHi:function(){}
  }
})
```



# v-on补充

传递自定义参数,事件修饰符

```
<div id="app">
  <input type="button" @click="dolt(p1,
p2)" />
  <input type="text" @keyup.enter="say
Hi">
</div>
```

文档传送门

<https://cn.vuejs.org/v2/api/#v-on>

```
var app = new Vue({
  el: "#app",
  methods: {
    dolt: function(p1,p2
) {},
    sayHi:function(){}
  }
})
```



## v-on补充

- ◆ 事件绑定的方法写成函数调用的形式，可以传入自定义参数
- ◆ 定义方法时需要定义形参来接收传入的实参
- ◆ 事件的后面跟上.修饰符 可以对事件进行限制
- ◆ .enter 可以限制触发的按键为回车
- ◆ 事件修饰符有多种

2

本地应用



v-for

v-on补充

v-model

小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear completed



黑马程序员™  
www.itheima.com



2

## 本地应用



v-for

v-on补充

v-model

### 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear completed



黑马程序员™  
www.itheima.com



# v-model

获取和设置表单元素的值



# v-model

获取和设置表单元素的值(双向数据绑定)



# v-model

获取和设置表单元素的值(双向数据绑定)

```
<div id="app">  
</div>
```

```
var app = new Vue({  
  el: "#app",  
  data: {  
  }  
})
```



# v-model

获取和设置表单元素的值(双向数据绑定)

```
<div id="app">  
  <input type="text" />  
</div>
```

```
var app = new Vue({  
  el: "#app",  
  data: {  
  }  
})
```



# v-model

获取和设置表单元素的值(双向数据绑定)

```
<div id="app">  
  <input type="text" />  
</div>
```

```
var app = new Vue({  
  el: "#app",  
  data: {  
    message: "黑马程序员"  
  }  
})
```

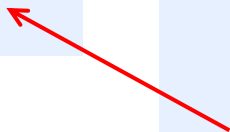


# v-model

获取和设置表单元素的值(双向数据绑定)

```
<div id="app">  
  <input type="text" v-  
model="message" />  
</div>
```

```
var app = new Vue({  
  el: "#app",  
  data: {  
    message: "黑马程序员"  
  }  
})
```



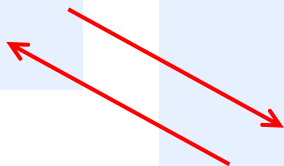


# v-model

获取和设置表单元素的值(双向数据绑定)

```
<div id="app">  
  <input type="text" v-  
model="message" />  
</div>
```

```
var app = new Vue({  
  el: "#app",  
  data: {  
    message: "黑马程序员"  
  }  
})
```







# v-model

获取和设置表单元素的值(双向数据绑定)

```
<div id="app">
  <input type="text" v-
model="message" />
</div>
```

```
var app = new Vue({
  el: "#app",
  data: {
    message: "黑马程序员"
  }
})
```



# v-model

- ◆ **v-model**指令的作用是便捷的设置和获取表单元素的值
- ◆ 绑定的数据会和表单元素**值**相关联
- ◆ 绑定的数据 $\longleftrightarrow$ 表单元素的值

2

## 本地应用



v-for

v-on补充

v-model

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



2

本地应用

v-for

v-on补充

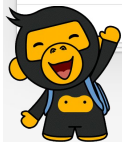
v-model

小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

Clear





# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear

1 新增

2 删除

3 统计

4 清空

5 隐藏

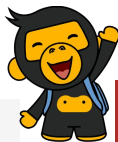
# 记事本

## 小黑记事本

请输入任务

- 吃饭饭
- 睡觉觉
- 写代码

3 items left Clear



1 新增

2 删除

3 统计

4 清空

5 隐藏

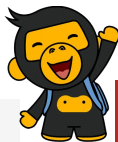
# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left Clear



1 新增

1. 生成列表结构
2. 获取用户输入
3. 回车,新增数据

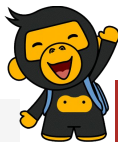
# 记事本

## 小黑记事本

请输入任务

- 吃饭饭
- 睡觉觉
- 写代码

3 items left Clear



1 新增

1. 生成列表结构(v-for 数组)
2. 获取用户输入
3. 回车,新增数据



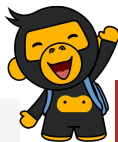
# 记事本

## 小黑记事本

请输入任务

- 吃饭饭
- 睡觉觉
- 写代码

3 items left Clear



1 新增

1. 生成列表结构(v-for 数组)
2. 获取用户输入(v-model)
3. 回车,新增数据

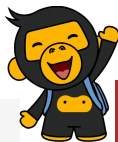
# 记事本

## 小黑记事本

请输入任务

- 吃饭饭
- 睡觉觉
- 写代码

3 items left Clear



1

新增

1. 生成列表结构(v-for 数组)
2. 获取用户输入(v-model)
3. 回车,新增数据(v-on .enter 添加数据)

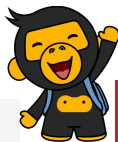
# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left Clear



1

新增

1. **v-for**指令的作用
2. **v-model**指令的作用
3. **v-on**指令,事件修饰符
4. 通过**审查元素**快速定位

# 记事本

## 小黑记事本

请输入任务

- 吃饭饭
- 睡觉觉
- 写代码

3 items left Clear



1 新增

2 删除

3 统计

4 清空

5 隐藏

# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



1 新增

2 删除

3 统计

4 清空

5 隐藏

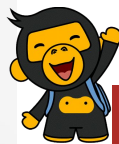
# 记事本

## 小黑记事本

请输入任务

- 吃饭饭
- 睡觉觉
- 写代码

3 items left Clear



2

删除

1. 点击删除指定内容

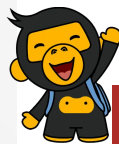
# 记事本

## 小黑记事本

请输入任务

- 吃饭饭
- 睡觉觉
- 写代码

3 items left Clear



2 删除

1. 点击删除指定内容(v-on)

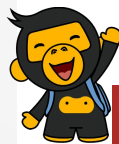
# 记事本

## 小黑记事本

请输入任务

- 吃饭饭
- 睡觉觉
- 写代码

3 items left Clear



2 删除

1. 点击删除指定内容(v-on splice)



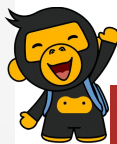
# 记事本

## 小黑记事本

请输入任务

- 吃饭饭
- 睡觉觉
- 写代码

3 items left Clear



2 删除

1. 点击删除指定内容(v-on splice 索引)

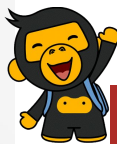
# 记事本

## 小黑记事本

请输入任务

- 吃饭饭
- 睡觉觉
- 写代码

3 items left Clear



### 2 删除

1. 数据改变,和数据绑定的元素同步改变
2. 事件的自定义参数
3. splice方法的作用

# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



1 新增

2 删除

3 统计

4 清空

5 隐藏

# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



1 新增

2 删除

3 统计

4 清空

5 隐藏

# 记事本

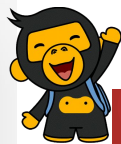
## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



3

统计

1. 统计信息个数

# 记事本

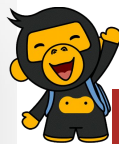
## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



3

统计

1. 统计信息个数(v-text)

# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



3

统计

1. 统计信息个数(v-text length)

# 记事本

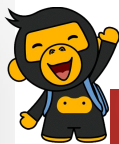
## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



3

统计

1. 基于数据开发方式
2. v-text指令的作用



# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



1 新增

2 删除

3 统计

4 清空

5 隐藏

# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



1 新增

2 删除

3 统计

4 清空

5 隐藏



# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



4

清空

1. 点击清除所有信息

# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



4

清空

1. 点击清除所有信息(v-on)

# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left Clear



4

清空

1. 点击清除所有信息(v-on 清空数组)

# 记事本

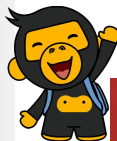
## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



4

清空

### 1. 基于数据的开发方式

# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



1 新增

2 删除

3 统计

4 清空

5 隐藏



# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



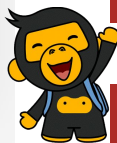
1 新增

2 删除

3 统计

4 清空

5 隐藏





# 记事本

## 小黑记事本

请输入任务

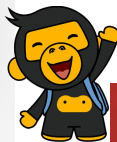
1. 吃饭饭

2. 睡觉觉

3. 写代码

3 items left

Clear



5

隐藏

1. 没有数据时,隐藏元素

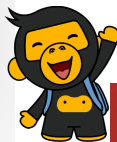
# 记事本

## 小黑记事本

请输入任务

- 吃饭饭
- 睡觉觉
- 写代码

3 items left Clear



5

隐藏

1. 没有数据时,隐藏元素(v-show v-if)

# 记事本

## 小黑记事本

请输入任务

- 1. 吃饭饭
- 2. 睡觉觉
- 3. 写代码

3 items left Clear



5

隐藏

1. 没有数据时,隐藏元素(v-show v-if 数组非空)

# 记事本

## 小黑记事本

请输入任务

1. 吃饭饭
2. 睡觉觉
3. 写代码

3 items left

Clear



1

新增

2

删除

3

统计

4

清空

5

隐藏





# 记事本

- ◆ 列表结构可以通过v-for指令结合数据生成
- ◆ v-on结合事件修饰符可以对事件进行限制,比如.enter
- ◆ v-on在绑定事件时可以传递自定义参数
- ◆ 通过v-model可以快速的设置和获取表单元素的值
- ◆ 基于数据的开发方式

# 课程安排

1

Vue基础



2

本地应用

3

网络应用

4

综合应用

# 课程安排

1

Vue基础

2

本地应用

3

网络应用



4

综合应用



3

## 网络应用

Vue结合网络数据开发应用

axios

网络请求库

axios+vue

结合Vue一起

天知道

漠河

搜索

北京 上海 广州 深圳

多云

低温 -17℃ ~ 高温 0℃

11日星期五

多云

低温 -15℃ ~ 高温 0℃

12日星期六

多云

低温 -11℃ ~ 高温 0℃

13日星期天

多云

低温 -4℃ ~ 高温 5℃

14日星期一

多云

低温 -2℃ ~ 高温 6℃

15日星期二

天气预报案例





3

## 网络应用

Vue结合网络数据开发应用

axios

网络请求库

axios+vue

结合Vue一起

天知道

漠河

搜索

北京 上海 广州 深圳

多云

低温 -17℃ ~ 高温 0℃

11日星期五

多云

低温 -15℃ ~ 高温 0℃

12日星期六

多云

低温 -11℃ ~ 高温 0℃

13日星期天

多云

低温 -4℃ ~ 高温 5℃

14日星期一

多云

低温 -2℃ ~ 高温 6℃

15日星期二

天气预报案例



3

## 网络应用

Vue结合网络数据开发应用

axios

网络请求库

axios+vue

结合Vue一起

天知道

漠河

搜索

北京 上海 广州 深圳

多云

低温 -17℃ ~ 高温 0℃

11日星期五

多云

低温 -15℃ ~ 高温 0℃

12日星期六

多云

低温 -11℃ ~ 高温 0℃

13日星期天

多云

低温 -4℃ ~ 高温 5℃

14日星期一

多云

低温 -2℃ ~ 高温 6℃

15日星期二

天气预报案例



# axios

功能强大的网络请求库

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```



# axios

功能强大的网络请求库

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

```
axios.get(地址)
```



# axios

功能强大的网络请求库

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

```
axios.get(地址).then(function(response){})
```



# axios

功能强大的网络请求库

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

```
axios.get(地址).then(function(response){},function(err){})
```



# axios

功能强大的网络请求库

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>  
axios.get(地址?查询字符串).then(function(response){},function(err){})
```



# axios

功能强大的网络请求库

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

```
axios.get(地址  
?key=value&key2=value2).then(function(response){},function(err){})
```





# axios

功能强大的网络请求库

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

```
axios.get(地址  
?key=value&key2=value2).then(function(response){},function(err){})  
axios.post(地址).then(function(response){},function(err){})
```



# axios

功能强大的网络请求库

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

```
axios.get(地址  
?key=value&key2=value2).then(function(response){},function(err){})  
axios.post(地址,参数对象).then(function(response){},function(err){})
```



# axios

功能强大的网络请求库

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

```
axios.get(地址  
?key=value&key2=values).then(function(response){},function(err){})  
axios.post(地址  
,{key:value,key2:value2}).then(function(response){},function(err){})
```



# axios

功能强大的网络请求库

## 随机获取笑话的接口

- 请求地址: <https://autumnfish.cn/api/joke/list>
- 请求方法: get
- 请求参数: num

参数名	参数说明	备注
num	笑话条数	类型为数字

- 响应内容: 随机笑话

## 用户注册接口1

- 请求地址: <https://autumnfish.cn/api/user/reg>
- 请求方法: post
- 请求参数: username

参数名	参数说明	备注
username	用户名	不能为空

- 响应内容: 注册成功或失败



# axios

- ◆ **axios**必须先导入才可以使用
- ◆ 使用**get**或**post**方法即可发送对应的请求
- ◆ **then**方法中的回调函数会在请求成功或失败时触发
- ◆ 通过回调函数的形参可以获取响应内容,或错误信息

## 文档传送门

<https://github.com/axios/axios>

# 3

## 网络应用

Vue结合网络数据开发应用



axios

axios+vue

网络请求库

结合Vue一起

天气预报案例



# 3

## 网络应用

Vue结合网络数据开发应用



axios

axios+vue

网络请求库

结合Vue一起

天气预报案例





# axios+vue

axios如何结合vue开发网络应用

```
var app = new Vue({  
  el: "#app",  
  data: {  
    joke: "搞笑的笑话"  
  },  
  methods: {  
    getJokes: function() {  
      // this.joke  
      axios.get("地址"  
    ).then(function(response) {  
      // this.joke ?  
    }, function(err) {});  
    }  
  }  
})
```





# axios+vue

axios如何结合vue开发网络应用

随机获取笑话的接口

- 请求地址: <https://autumnfish.cn/api/joke>
- 请求方法: get
- 请求参数: 无
- 响应内容: 一条随机笑话



## axios+vue

- ◆ axios回调函数中的this已经改变,无法访问到data中数据
- ◆ 把this保存起来,回调函数中直接使用保存的this即可
- ◆ 和本地应用的最大区别就是改变了数据来源

# 3

## 网络应用

Vue结合网络数据开发应用



axios

axios+vue

网络请求库

结合Vue一起

天气预报案例



# 3

## 网络应用

Vue结合网络数据开发应用

axios

网络请求库

axios+vue

结合Vue一起

天知道

漠河

搜索

北京 上海 广州 深圳

多云

低温 -15℃ ~ 高温 0℃

期五

多云

低温 -15℃ ~ 高温 0℃

12日星期六

多云

低温 -11℃ ~ 高温 0℃

13日星期天

多云

低温 -4℃ ~ 高温 5℃

14日星期一

多云

低温 -2℃ ~ 高温 6℃

15日星期二



天气预报案例



# 天知道

查询天气的应用

## 天知道

搜索

[北京](#) [上海](#) [广州](#) [深圳](#)

多云

低温 -17℃ ~ 高温 0℃

11日星期五

多云

低温 -15℃ ~ 高温 0℃

12日星期六

多云

低温 -11℃ ~ 高温 0℃

13日星期天

多云

低温 -4℃ ~ 高温 5℃

14日星期一

多云

低温 -2℃ ~ 高温 6℃

15日星期二

1 回车查询

2 点击查询



# 天知道

查询天气的应用

## 天知道

搜索

北京 上海 广州 深圳

多云	多云	多云	多云	多云
低温 -17℃ ~ 高温 0℃	低温 -15℃ ~ 高温 0℃	低温 -11℃ ~ 高温 0℃	低温 -4℃ ~ 高温 5℃	低温 -2℃ ~ 高温 6℃
11日星期五	12日星期六	13日星期天	14日星期一	15日星期二

1

回车查询

2

点击查询

# 天知道

查询天气的应用



1 回车查询

2 点击查询

## 天知道

搜索

北京 上海 广州 深圳

多云

低温 -17℃ ~ 高温 0℃

11日星期五

多云

低温 -15℃ ~ 高温 0℃

12日星期六

多云

低温 -11℃ ~ 高温 0℃

13日星期天

多云

低温 -4℃ ~ 高温 5℃

14日星期一

多云

低温 -2℃ ~ 高温 6℃

15日星期二

# 天知道

查询天气的应用



1 回车查询

1. 按下回车
2. 查询数据
3. 渲染数据

## 天知道

搜索

北京 上海 广州 深圳

多云	多云	多云	多云	多云
低温 -17℃ ~ 高温 0℃	低温 -15℃ ~ 高温 0℃	低温 -11℃ ~ 高温 0℃	低温 -4℃ ~ 高温 5℃	低温 -2℃ ~ 高温 6℃
11日星期五	12日星期六	13日星期天	14日星期一	15日星期二



# 天知道

查询天气的应用



1 回车查询

1. 按下回车(v-on .enter)
2. 查询数据
3. 渲染数据

## 天知道

搜索

北京 上海 广州 深圳

多云

低温 -17℃ ~ 高温 0℃

11日星期五

多云

低温 -15℃ ~ 高温 0℃

12日星期六

多云

低温 -11℃ ~ 高温 0℃

13日星期天

多云

低温 -4℃ ~ 高温 5℃

14日星期一

多云

低温 -2℃ ~ 高温 6℃

15日星期二

# 天知道

查询天气的应用



1

回车查询

1. 按下回车(v-on .enter)
2. 查询数据(axios 接口 v-model )
3. 渲染数据

## 天知道

搜索

北京 上海 广州 深圳

多云	多云	多云	多云	多云
低温 -17℃ ~ 高温 0℃	低温 -15℃ ~ 高温 0℃	低温 -11℃ ~ 高温 0℃	低温 -4℃ ~ 高温 5℃	低温 -2℃ ~ 高温 6℃
11日星期五	12日星期六	13日星期天	14日星期一	15日星期二

# 天知道

查询天气的应用



## 1 回车查询

1. 按下回车(v-on .enter)
2. 查询数据(axios 接口 v-model )
3. 渲染数据(v-for 数组 that)

### 天知道

搜索

北京 上海 广州 深圳

多云

低温 -17℃ ~ 高温 0℃

11日星期五

多云

低温 -15℃ ~ 高温 0℃

12日星期六

多云

低温 -11℃ ~ 高温 0℃

13日星期天

多云

低温 -4℃ ~ 高温 5℃

14日星期一

多云

低温 -2℃ ~ 高温 6℃

15日星期二

# 天知道

查询天气的应用

## 天知道

搜索

北京 上海 广州 深圳

多云	多云	多云	多云	多云
低温 -17℃ ~ 高温 0℃	低温 -15℃ ~ 高温 0℃	低温 -11℃ ~ 高温 0℃	低温 -4℃ ~ 高温 5℃	低温 -2℃ ~ 高温 6℃
11日星期五	12日星期六	13日星期天	14日星期一	15日星期二



1

## 回车查询

1. 按下回车(v-on .enter)
2. 查询数据(axios 接口 v-model )
3. 渲染数据(v-for 数组 that)

### 天气接口

- 请求地址：  
[http://wthrcdn.etouch.cn/weather\\_mini](http://wthrcdn.etouch.cn/weather_mini)
- 请求方法：get
- 请求参数：city（查询的城市名）
- 响应内容：天气信息



1

## 回车查询

- ◆ 应用的逻辑代码建议和页面分离，使用单独的js文件编写
- ◆ axios回调函数中this指向改变了，需要额外的保存一份
- ◆ 服务器返回的数据比较复杂时，获取的时候需要注意层级结构

# 天知道

查询天气的应用

## 天知道

搜索

北京 上海 广州 深圳

多云	多云	多云	多云	多云
低温 -17℃ ~ 高温 0℃	低温 -15℃ ~ 高温 0℃	低温 -11℃ ~ 高温 0℃	低温 -4℃ ~ 高温 5℃	低温 -2℃ ~ 高温 6℃
11日星期五	12日星期六	13日星期天	14日星期一	15日星期二



1

回车查询

2

点击查询

# 天知道

查询天气的应用

## 天知道

搜索

北京 上海 广州 深圳

多云	多云	多云	多云	多云
低温 -17℃ ~ 高温 0℃	低温 -15℃ ~ 高温 0℃	低温 -11℃ ~ 高温 0℃	低温 -4℃ ~ 高温 5℃	低温 -2℃ ~ 高温 6℃
11日星期五	12日星期六	13日星期天	14日星期一	15日星期二



1

回车查询

2

点击查询

# 天知道

查询天气的应用

## 天知道

搜索

北京 上海 广州 深圳

多云

低温 -17℃ ~ 高温 0℃

11日星期五

多云

低温 -15℃ ~ 高温 0℃

12日星期六

多云

低温 -11℃ ~ 高温 0℃

13日星期天

多云

低温 -4℃ ~ 高温 5℃

14日星期一

多云

低温 -2℃ ~ 高温 6℃

15日星期二



2

点击查询

1. 点击城市
2. 查询数据
3. 渲染数据



# 天知道

查询天气的应用

## 天知道

搜索

北京 上海 广州 深圳

多云

低温 -17℃ ~ 高温 0℃

11日星期五

多云

低温 -15℃ ~ 高温 0℃

12日星期六

多云

低温 -11℃ ~ 高温 0℃

13日星期天

多云

低温 -4℃ ~ 高温 5℃

14日星期一

多云

低温 -2℃ ~ 高温 6℃

15日星期二



2

点击查询

1. 点击城市(v-on 自定义参数)
2. 查询数据
3. 渲染数据

# 天知道

查询天气的应用

## 天知道

搜索

北京 上海 广州 深圳

多云

低温 -17℃ ~ 高温 0℃

11日星期五

多云

低温 -15℃ ~ 高温 0℃

12日星期六

多云

低温 -11℃ ~ 高温 0℃

13日星期天

多云

低温 -4℃ ~ 高温 5℃

14日星期一

多云

低温 -2℃ ~ 高温 6℃

15日星期二



2

点击查询

1. 点击城市(v-on 自定义参数)
2. 查询数据 (this.方法())
3. 渲染数据



2

点击查询

- ◆ 自定义参数可以让代码的复用性更高
- ◆ `methods`中定义的方法内部，可以通过`this`关键字点出其他的方法

# 课程安排

1

Vue基础

2

本地应用

3

网络应用



4

综合应用

# 课程安排

1

Vue基础

2

本地应用

3

网络应用

4

综合应用





4

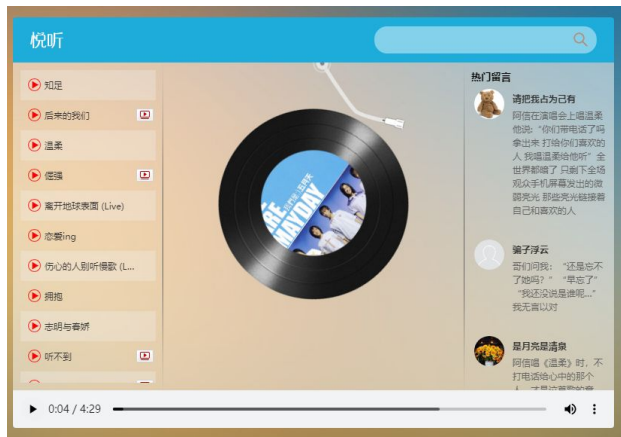
## 综合应用





## 4

## 综合应用



### 1

## 歌曲搜索

### 2

## 歌曲播放

### 3

## 歌曲封面

### 4

## 歌曲评论

### 5

## 播放动画

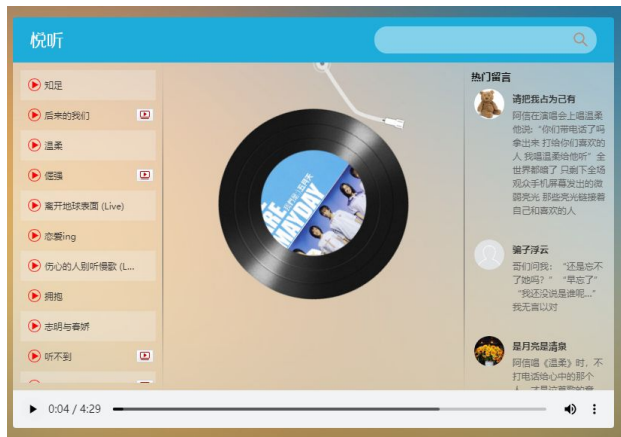
### 6

## mv播放



## 4

## 综合应用



### 1

## 歌曲搜索

### 2

## 歌曲播放

### 3

## 歌曲封面

### 4

## 歌曲评论

### 5

## 播放动画

### 6

## mv播放



## 4

## 综合应用



### 1

## 歌曲搜索

### 2

## 歌曲播放

### 3

## 歌曲封面

### 4

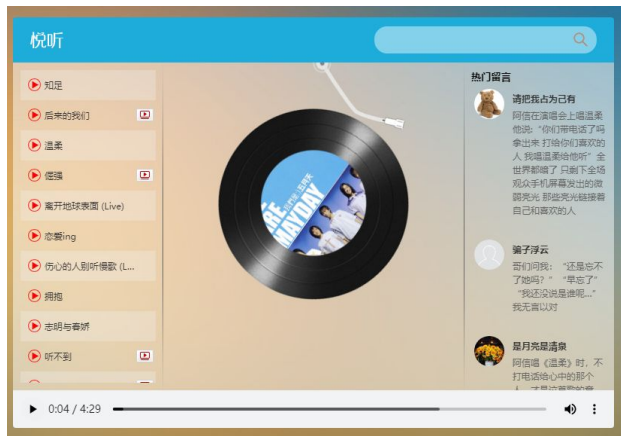
## 歌曲评论

### 5

## 播放动画

### 6

## mv播放



## 4

## 综合应用



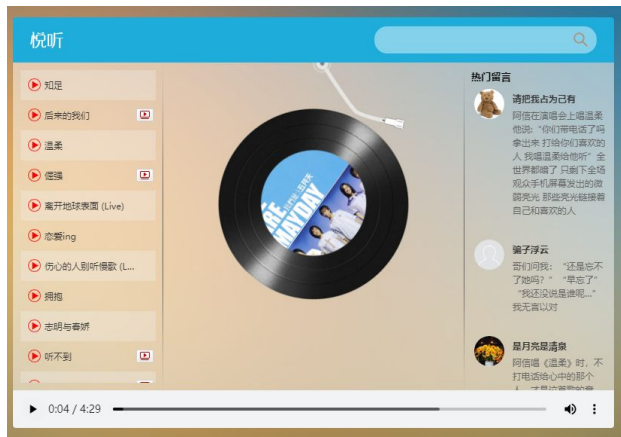
### 1

## 歌曲搜索

1. 按下回车

2. 查询数据

3. 渲染数据



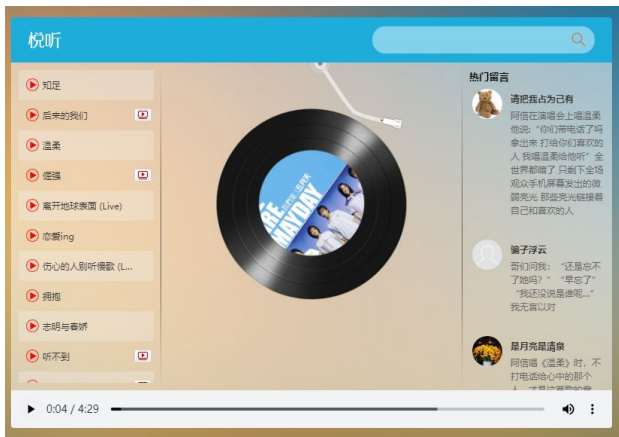
## 4

## 综合应用



## 1

## 歌曲搜索



1. 按下回车(v-on .enter)

2. 查询数据

3. 渲染数据

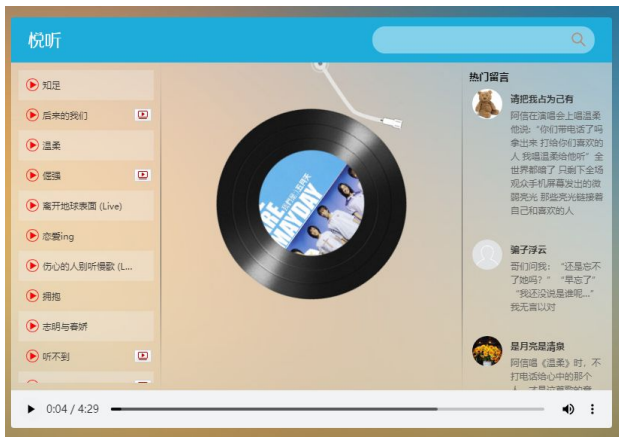
## 4

## 综合应用



## 1

## 歌曲搜索



1. 按下回车(v-on .enter)
2. 查询数据(axios 接口 v-model )
3. 渲染数据

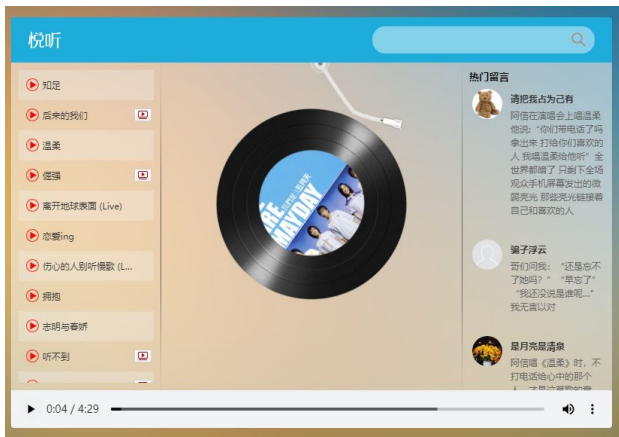
## 4

## 综合应用



## 1

## 歌曲搜索



1. 按下回车(v-on .enter)
2. 查询数据(axios 接口 v-model )
3. 渲染数据(v-for 数组 that)

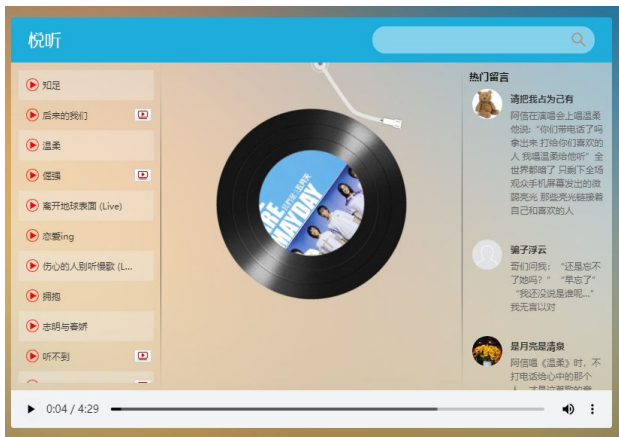
## 4

## 综合应用



### 1

## 歌曲搜索



1. 按下回车(v-on .enter)
2. 查询数据(axios 接口 v-model )
3. 渲染数据(v-for 数组 that)

### 歌曲搜索接口

- 请求地址:

<https://autumnfish.cn/search>

- 请求方法: get
- 请求参数: keywords (查询的关键字)
- 响应内容: 歌曲搜索结果



1

## 歌曲搜索

- ◆ 服务器返回的数据比较复杂时，获取的时候需要注意**层级**结构
- ◆ 通过**审查元素**快速定位到需要操纵的元素

## 4

## 综合应用



### 1

## 歌曲搜索

### 2

## 歌曲播放

### 3

## 歌曲封面

### 4

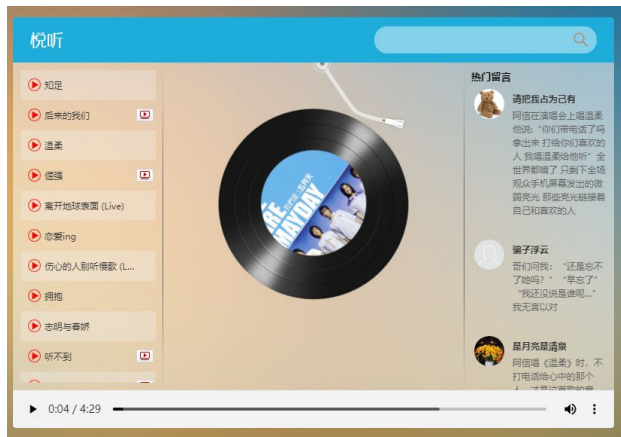
## 歌曲评论

### 5

## 播放动画

### 6

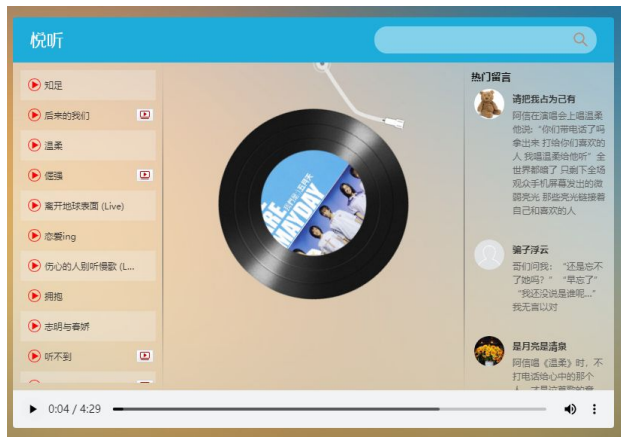
## mv播放





## 4

## 综合应用



### 1

歌曲搜索

### 2

歌曲播放

### 3

歌曲封面

### 4

歌曲评论

### 5

播放动画

### 6

mv播放



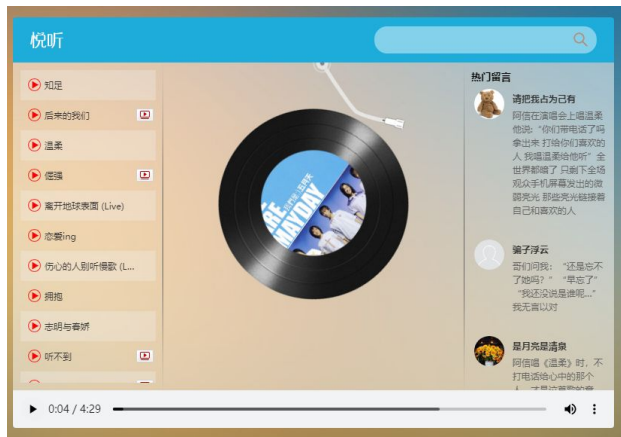
## 4

## 综合应用



## 2

## 歌曲播放



1. 点击播放
2. 歌曲地址获取
3. 歌曲地址设置

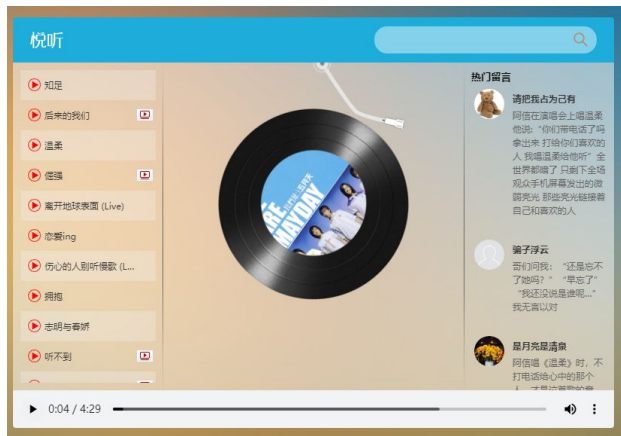
## 4

## 综合应用



## 2

## 歌曲播放



1. 点击播放(v-on)

2. 歌曲地址获取

3. 歌曲地址设置

## 4

## 综合应用



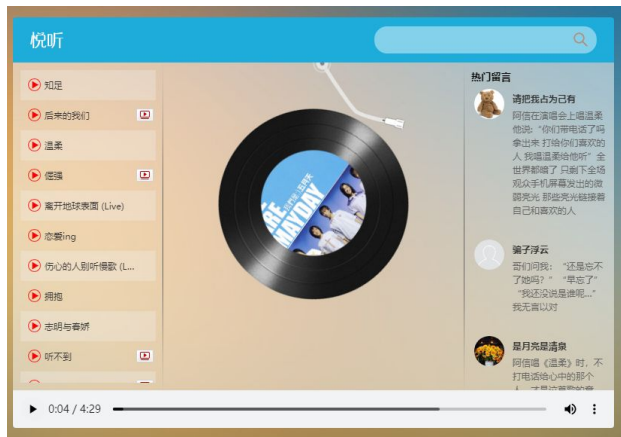
## 2

## 歌曲播放

1. 点击播放(v-on)

2. 歌曲地址获取

3. 歌曲地址设置



### 歌曲url获取

- 请求地址:

<https://autumnfish.cn/song/url>

- 请求方法: get
- 请求参数: id (歌曲id)
- 响应内容: 歌曲的url地址

## 4

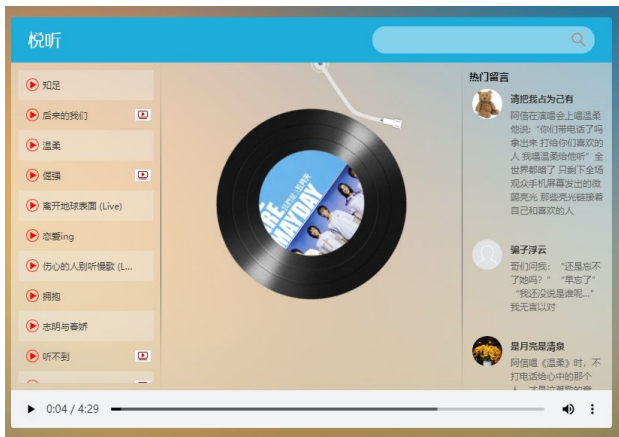
## 综合应用



## 2

## 歌曲播放

1. 点击播放(v-on 自定义参数)
2. 歌曲地址获取
3. 歌曲地址设置



### 歌曲url获取

- 请求地址：  
<https://autumnfish.cn/song/url>
- 请求方法：get
- 请求参数：id (歌曲id)
- 响应内容：歌曲的url地址

## 4

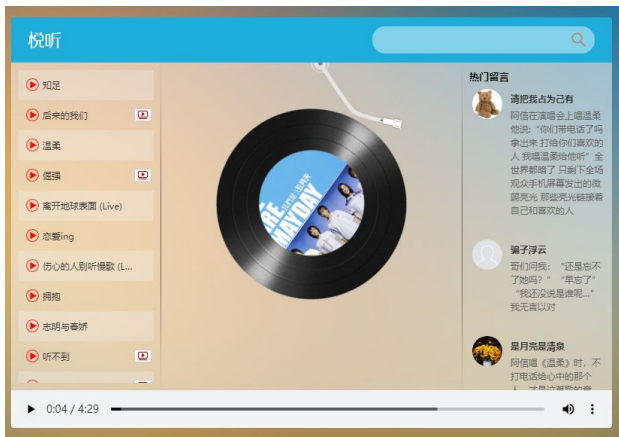
## 综合应用



## 2

## 歌曲播放

1. 点击播放(v-on 自定义参数)
2. 歌曲地址获取(接口 歌曲id)
3. 歌曲地址设置



### 歌曲url获取

- 请求地址：  
<https://autumnfish.cn/song/url>
- 请求方法：get
- 请求参数：id (歌曲id)
- 响应内容：歌曲的url地址

## 4

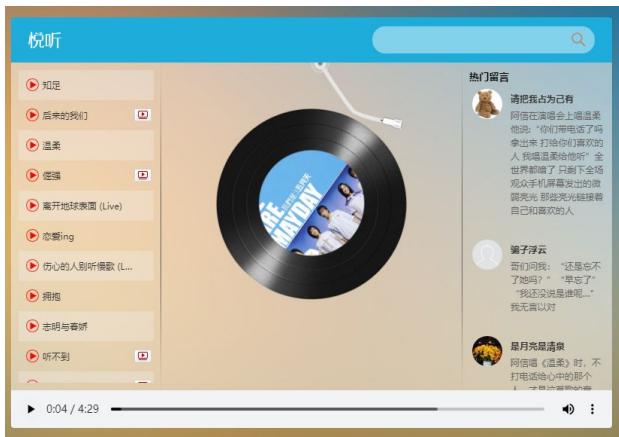
## 综合应用



## 2

## 歌曲播放

1. 点击播放(v-on 自定义参数)
2. 歌曲地址获取(接口 歌曲id)
3. 歌曲地址设置(v-bind)



### 歌曲url获取

- 请求地址：  
<https://autumnfish.cn/song/url>
- 请求方法：get
- 请求参数：id (歌曲id)
- 响应内容：歌曲的url地址



2

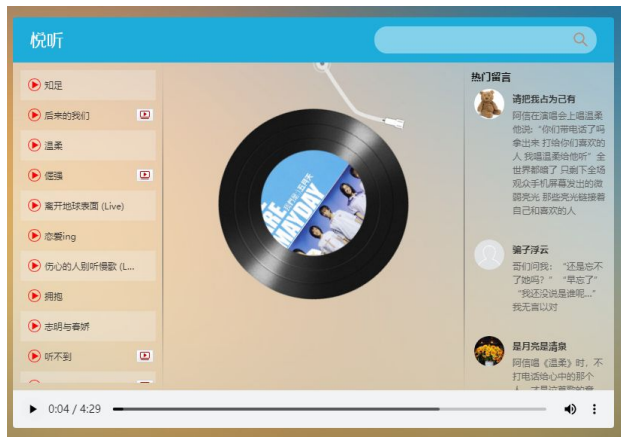
## 歌曲播放

◆歌曲id依赖歌曲搜索的结果，对于不用的数据也需要关注



## 4

## 综合应用



### 1

歌曲搜索

### 2

歌曲播放

### 3

歌曲封面

### 4

歌曲评论

### 5

播放动画

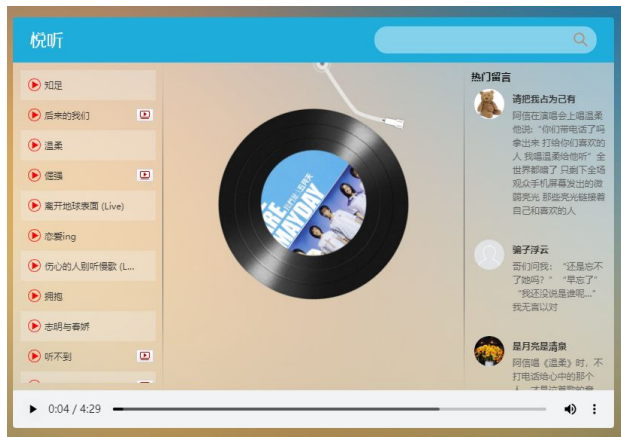
### 6

mv播放



## 4

## 综合应用



### 1

### 歌曲搜索

### 2

### 歌曲播放

### 3

### 歌曲封面

### 4

### 歌曲评论

### 5

### 播放动画

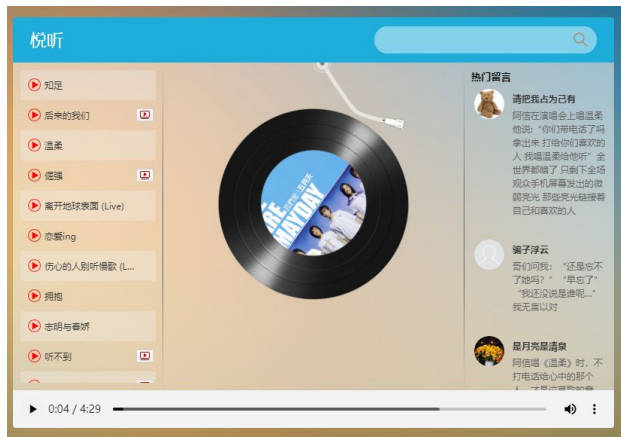
### 6

### mv播放



## 4

## 综合应用



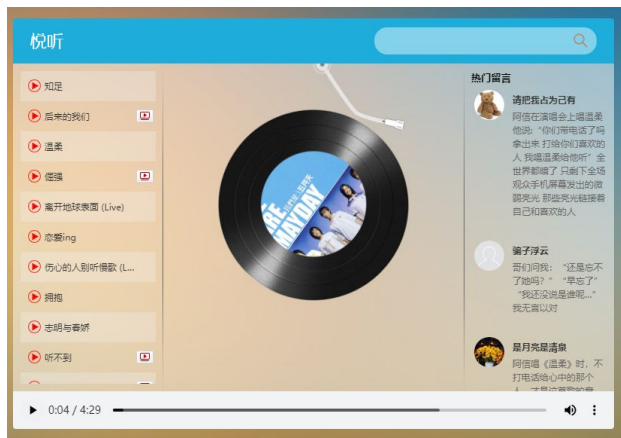
## 3

## 歌曲封面

1. 点击播放
2. 歌曲封面获取
3. 歌曲封面设置

## 4

## 综合应用



## 3

## 歌曲封面

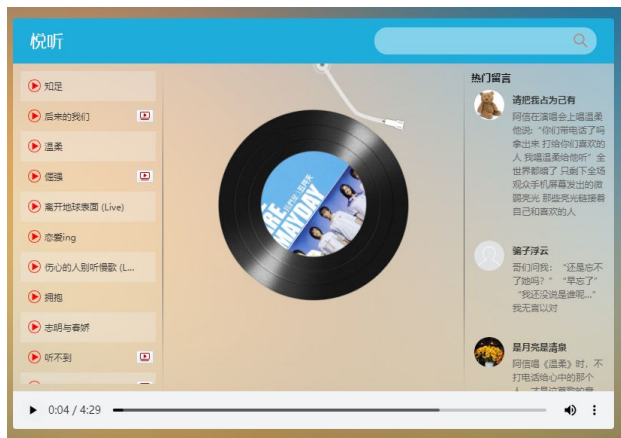
1. 点击播放(增加逻辑)
2. 歌曲封面获取
3. 歌曲封面设置

### 歌曲详情获取

- 请求地址：  
<https://autumnfish.cn/song/detail>
- 请求方法：get
- 请求参数：ids (歌曲id)
- 响应内容：歌曲详情，包含封面信息

## 4

## 综合应用



## 3

## 歌曲封面

1. 点击播放(增加逻辑)
2. 歌曲封面获取(接口 歌曲id)
3. 歌曲封面设置

### 歌曲详情获取

- 请求地址：  
<https://autumnfish.cn/song/detail>
- 请求方法：get
- 请求参数：ids (歌曲id)
- 响应内容：歌曲详情，包含封面信息

## 4

## 综合应用



## 3

## 歌曲封面

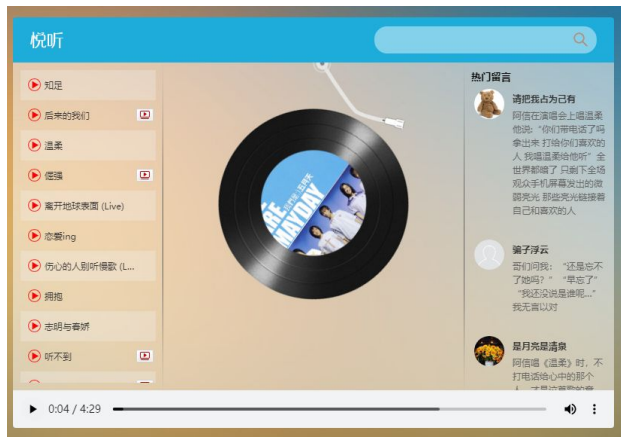
1. 点击播放(增加逻辑)
2. 歌曲封面获取(接口 歌曲id)
3. 歌曲封面设置(v-bind)

### 歌曲详情获取

- 请求地址：  
<https://autumnfish.cn/song/detail>
- 请求方法：get
- 请求参数：ids (歌曲id)
- 响应内容：歌曲详情，包含封面信息

## 4

## 综合应用



## 3

## 歌曲封面

1. 点击播放(增加逻辑)
2. 歌曲封面获取(接口 歌曲id)
3. 歌曲封面设置(v-bind)

### 歌曲详情获取

- 请求地址：  
<https://autumnfish.cn/song/detail>
- 请求方法：get
- 请求参数：ids (歌曲id)
- 响应内容：歌曲详情，包含封面信息



3

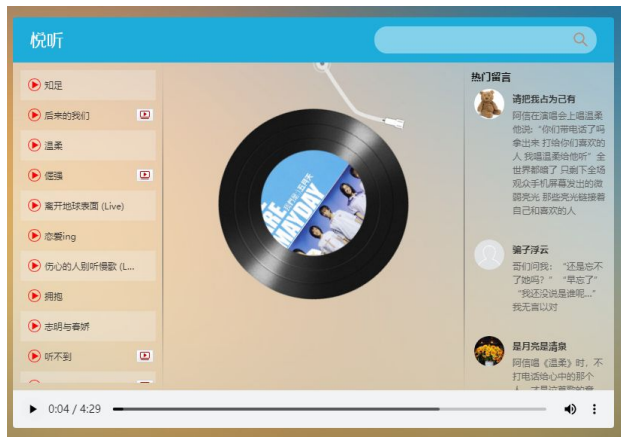
### 歌曲封面

- ◆ 在vue中通过v-bind操纵属性
- ◆ 本地无法获取的数据，基本都会有对应的接口



## 4

## 综合应用



### 1

### 歌曲搜索

### 2

### 歌曲播放

### 3

### 歌曲封面

### 4

### 歌曲评论

### 5

### 播放动画

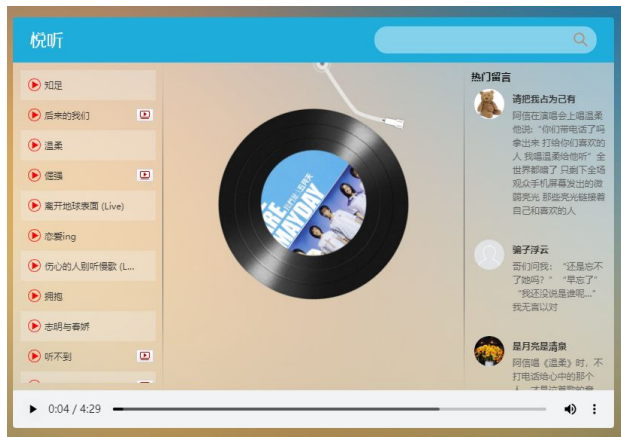
### 6

### mv播放



## 4

## 综合应用



### 1

### 歌曲搜索

### 2

### 歌曲播放

### 3

### 歌曲封面

### 4

### 歌曲评论

### 5

### 播放动画

### 6

### mv播放



## 4

## 综合应用



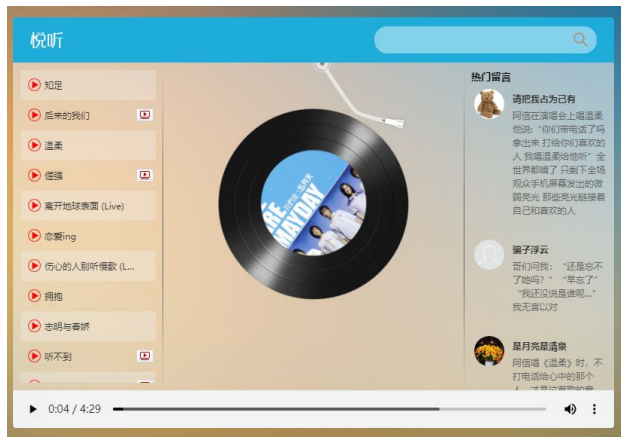
## 4

## 歌曲评论

1. 点击播放
2. 歌曲评论获取
3. 歌曲评论渲染

## 4

## 综合应用



## 4

## 歌曲评论

1. 点击播放(增加逻辑)

2. 歌曲评论获取

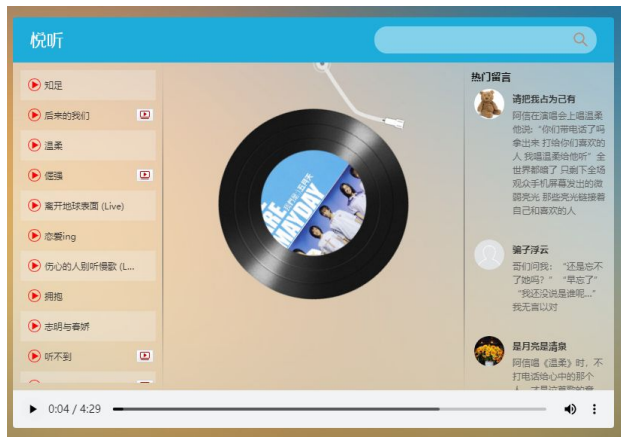
3. 歌曲评论渲染

### 热门评论获取

- 请求地址：  
<https://autumnfish.cn/comment/hot?type=0>
- 请求方法: get
- 请求参数: id (歌曲id,type固定为0)
- 响应内容: 歌曲的热门评论

## 4

## 综合应用



## 4

## 歌曲评论

1. 点击播放(增加逻辑)
2. 歌曲评论获取(接口 歌曲id)
3. 歌曲评论渲染

### 热门评论获取

- 请求地址:  
<https://autumnfish.cn/comment/hot?type=0>
- 请求方法: get
- 请求参数: id (歌曲id,type固定为0)
- 响应内容: 歌曲的热门评论

## 4

## 综合应用



## 4

## 歌曲评论

1. 点击播放(增加逻辑)
2. 歌曲评论获取(接口 歌曲id)
3. 歌曲评论渲染(v-for)

### 热门评论获取

- 请求地址:  
<https://autumnfish.cn/comment/hot?type=0>
- 请求方法: get
- 请求参数: id (歌曲id,type固定为0)
- 响应内容: 歌曲的热门评论



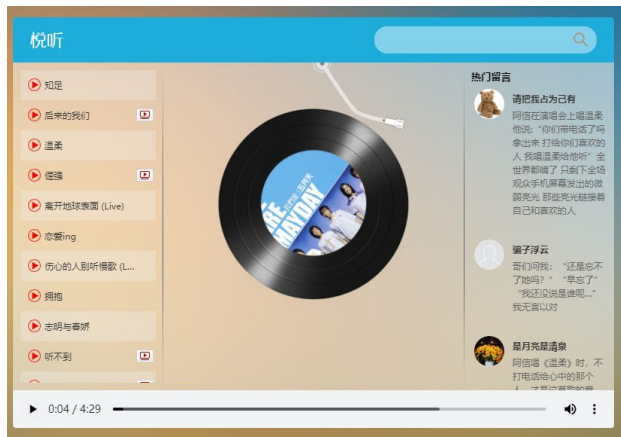
4

## 歌曲评论

◆ 在vue中通过v-for生成列表

## 4

## 综合应用



### 1

### 歌曲搜索

### 2

### 歌曲播放

### 3

### 歌曲封面

### 4

### 歌曲评论

### 5

### 播放动画

### 6

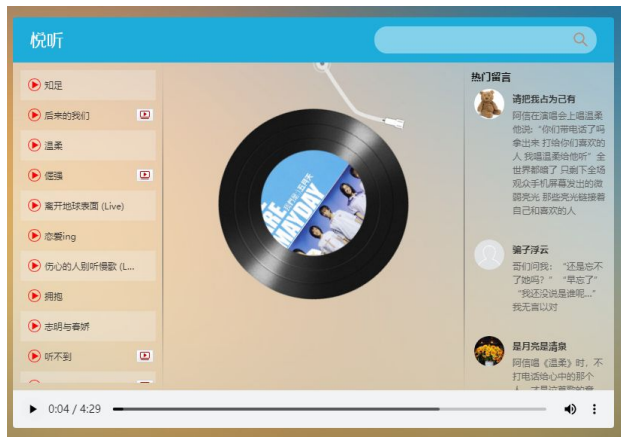
### mv播放





## 4

## 综合应用



### 1

### 歌曲搜索

### 2

### 歌曲播放

### 3

### 歌曲封面

### 4

### 歌曲评论

### 5

### 播放动画

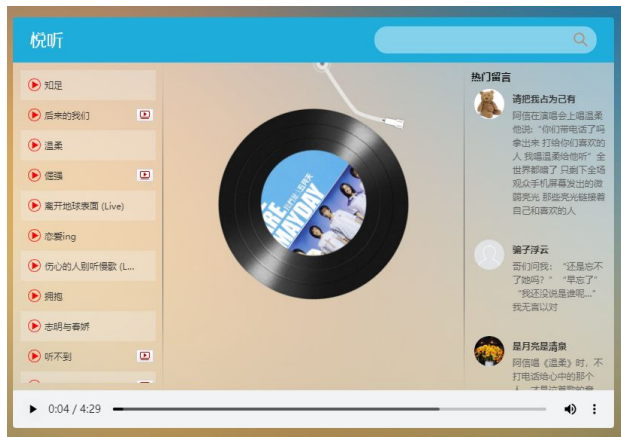
### 6

### mv播放



## 4

## 综合应用



1. 监听音乐播放

2. 监听音乐暂停

3. 操纵类名

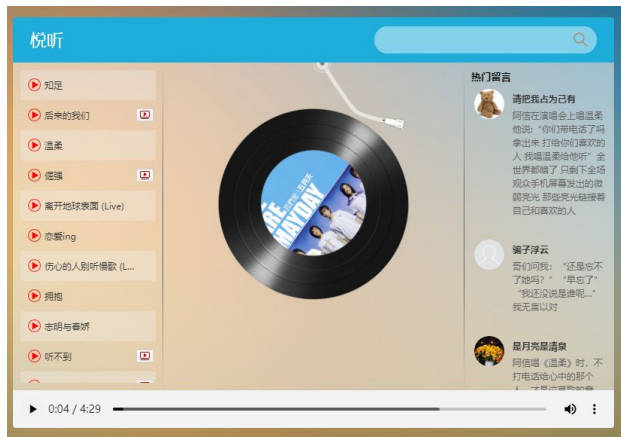


## 5

## 播放动画

## 4

## 综合应用



1. 监听音乐播放(v-on play)

2. 监听音乐暂停

3. 操纵类名

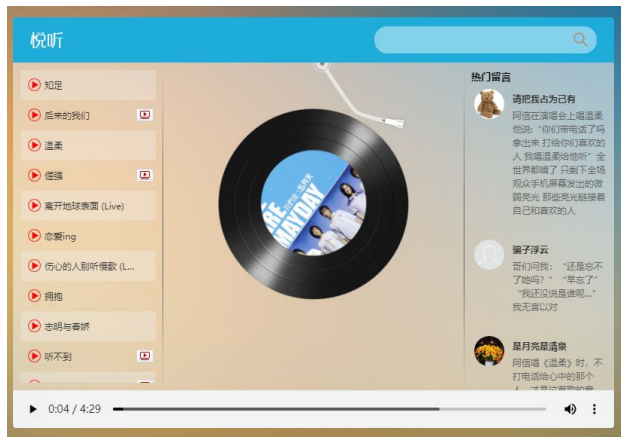


## 5

## 播放动画

## 4

## 综合应用



1. 监听音乐播放(v-on play)
2. 监听音乐暂停(v-on pause)
3. 操纵类名

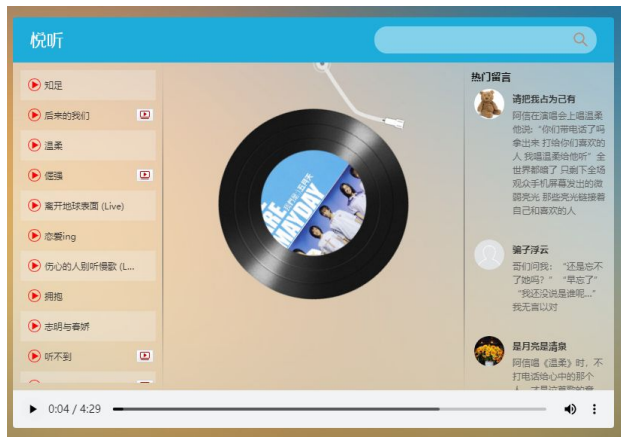


## 5

## 播放动画

## 4

## 综合应用



1. 监听音乐播放(v-on play)
2. 监听音乐暂停(v-on pause)
3. 操纵类名(v-bind 对象)



## 5

## 播放动画



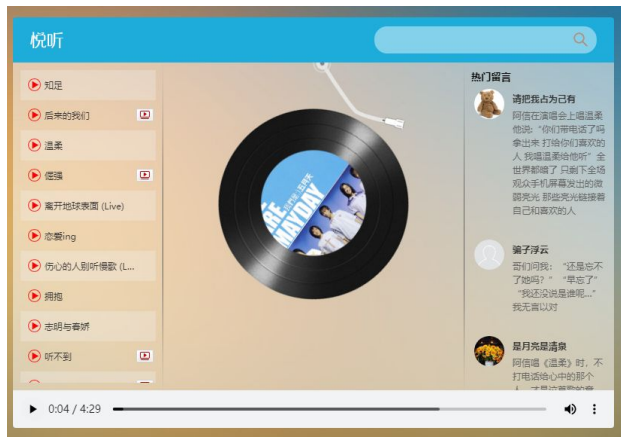
5

## 播放动画

- ◆ **audio**标签的**play**事件会在音频播放的时候触发
- ◆ **audio**标签的**pause**事件会在音频暂停的时候触发
- ◆ 通过**对象**的方式设置类名，类名生效与否取决于后面值的**真假**

## 4

## 综合应用



### 1

### 歌曲搜索

### 2

### 歌曲播放

### 3

### 歌曲封面

### 4

### 歌曲评论

### 5

### 播放动画

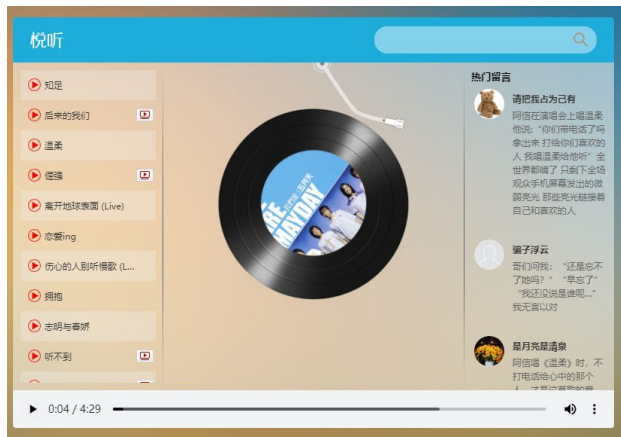
### 6

### mv播放



## 4

## 综合应用



### 1

### 歌曲搜索

### 2

### 歌曲播放

### 3

### 歌曲封面

### 4

### 歌曲评论

### 5

### 播放动画

### 6

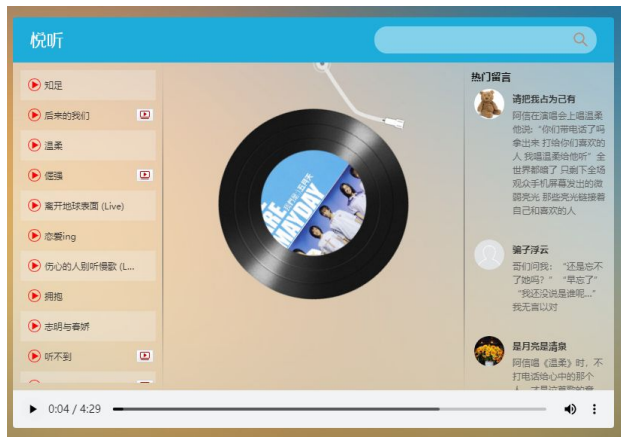
### mv播放





## 4

## 综合应用



## 6

## mv播放

1. mv图标显示

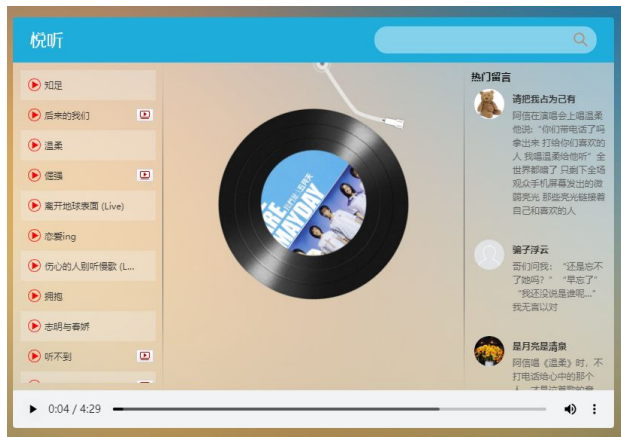
2. mv地址获取

3. 遮罩层

4. mv地址设置

## 4

## 综合应用



## 6

## mv播放

1. mv图标显示(v-if)

2. mv地址获取

3. 遮罩层

4. mv地址设置

### mv地址获取

• 请求地址：

<https://autumnfish.cn/mv/url>

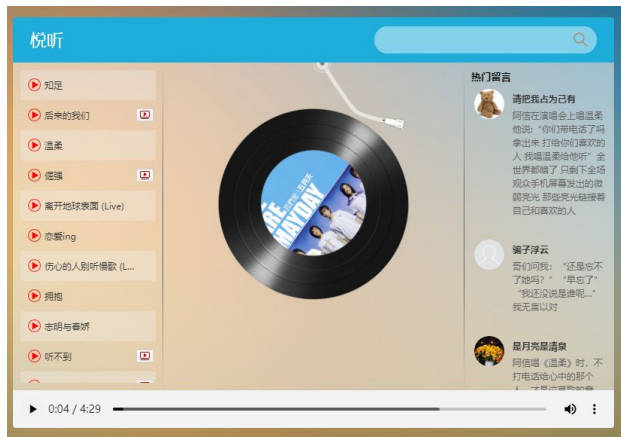
• 请求方法：get

• 请求参数：id (mvid, 为0说明没有mv)

• 响应内容：mv的地址

## 4

## 综合应用



## 6

## mv播放

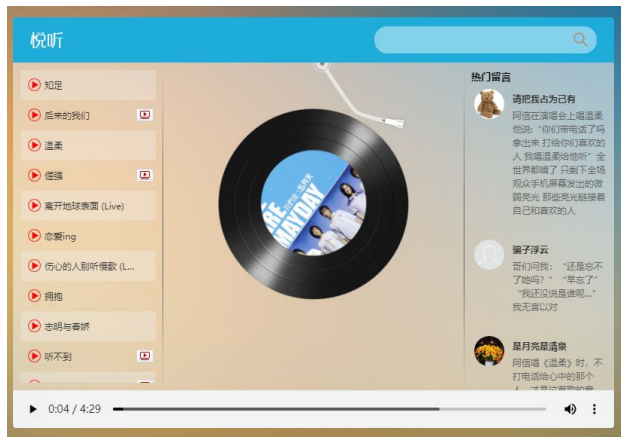
1. mv图标显示(v-if)
2. mv地址获取(接口 mvid)
3. 遮罩层
4. mv地址设置

### mv地址获取

- 请求地址：  
<https://autumnfish.cn/mv/url>
- 请求方法：get
- 请求参数：id (mvid, 为0说明没有mv)
- 响应内容：mv的地址

## 4

## 综合应用



## 6

## mv播放

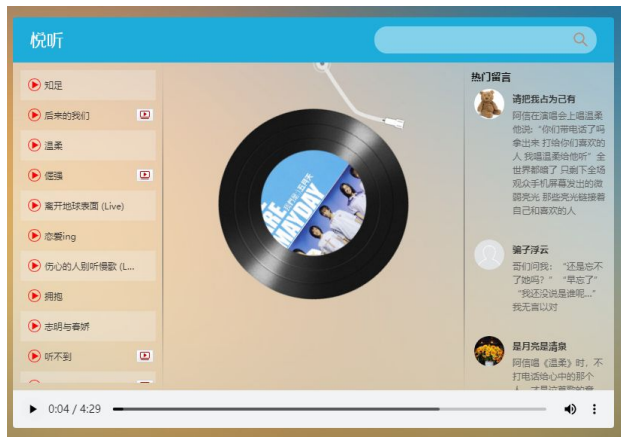
1. mv图标显示(v-if)
2. mv地址获取(接口 mvid)
3. 遮罩层(v-show v-on)
4. mv地址设置

### mv地址获取

- 请求地址：  
<https://autumnfish.cn/mv/url>
- 请求方法：get
- 请求参数：id (mvid, 为0说明没有mv)
- 响应内容：mv的地址

## 4

## 综合应用



## 6

## mv播放

1. mv图标显示(v-if)
2. mv地址获取(接口 mvid)
3. 遮罩层(v-show v-on)
4. mv地址设置(v-bind)

### mv地址获取

- 请求地址：  
<https://autumnfish.cn/mv/url>
- 请求方法：get
- 请求参数：id (mvid, 为0说明没有mv)
- 响应内容：mv的地址

## 4

## 综合应用



### 1

### 歌曲搜索

### 2

### 歌曲播放

### 3

### 歌曲封面

### 4

### 歌曲评论

### 5

### 播放动画

### 6

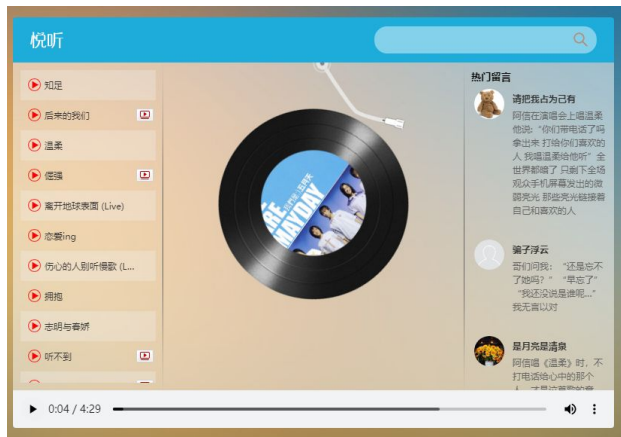
### mv播放





## 4

## 综合应用



### 1

## 歌曲搜索

### 2

## 歌曲播放

### 3

## 歌曲封面

### 4

## 歌曲评论

### 5

## 播放动画

### 6

## mv播放



## 4

## 综合应用

- ◆不同的接口需要的数据是**不同**的，文档的阅读需要**仔细**
- ◆页面结构复杂之后，通过**审查元素**的方式去**快速定位**相关元素
- ◆响应式的数据都需要定义在**data**中定义





1

Vue基础

2

本地应用

3

网络应用

4

综合应用

1

Vue基础



2

本地应用

3

网络应用

4

综合应用

1

Vue基础

2

本地应用

3

网络应用



4

综合应用

1

Vue基础

2

本地应用

3

网络应用

4

综合应用



