一、vue基础

2021年6月27日 13:42

vue基础

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>vue基础</title>
</head>
<body>
    <div id="app">
        {{message}}
    </div>
<!-- 开发环境版本，包含了有帮助的命令行警告 -->
<script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
<script>
    var app = new Vue({
        el:"#app",
        data:{
            message:"Hello Vue!"
        }
    })
</script>
</body>
</html>
```

二、el:挂载点

2021年6月27日 13:40

2. el:挂载点

2.1 vue实例的作用范围是什么?

vue会管理el选项命中的元素及其内部的后代元素。

2.2 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。

el:"#id"

el:".class"

2.3 是否可以设置dom元素呢?

可以使用其他的双标签，但不能使用HTML和BODY。

三、data 数据对象

2021年6月27日 13:41

3. data 数据对象

vue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

```
<body>
    <div id="app">
        {{message }}
        <h2>{{school.name }} {{school.mobile}}</h2>
        <ul>
            <li>{{campus[0]}}</li>
            <li>{{campus[3]}}</li>
        </ul>
    </div>
    <!-- 开发环境版本，包含了有帮助的命令行警告 -->
    <script src="https://cdn.jsdelivr.net/npm/vue@
2/dist/vue.js"></script>
    <script>
        var app = new Vue({
            el:"#app",
            data:{
                message:"你好 小黑！",
                school:{
                    name:"张三",
                    mobile:"119"
                },
                campus:["吉林","北京","上海","松原"]
            }
        })
    </script>
</body>
```

四、vue指令

2021年6月27日 13:41

4. vue指令

4.1 内容绑定，事件绑定

v-text	v-html	v-on基础

v-text:

设置标签的文本值(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

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

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

v-html:

设置元素的innerHTML

内容中的有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

```
<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-on基础:

为元素绑定事件

事件名不需要写on

指令可以简写@

绑定的方法在methods属性中

```
<div id="app">
    <input type="button"  value="事件绑定" v-
    on:monseenter="doIt">
    <input type="button" value="v-on指令" v-on:click="doIt">
    <input type="button" value="v-on简写" @click="doIt">
    <input type="button" value="双击事件" @dblclick="doIt">
</div>

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

计数器:

创建Vue事例时：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

4.2 显示切换，属性绑定

v-show	v-if	v-bind

v-show:

根据表达值的真假，切换元素的显示和隐藏。

作用：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。

```
<div id="app">
    <img src="地址" v-show="true">
    <img src="地址" v-show="isShow">
    <img src="地址" v-show="age>=18">
</div>

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

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）

本质是通过操纵dom元素来切换显示状态

表达式的值为true时，元素存在于dom树中，为false 时，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。

```
<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-bind:

设置元素的属性（比如：src,title,class）

作用：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

```
<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:"程序员",
        isActive:false
    }
})
```

列表数据可以使用数组保存

v-bind指令可以设置元素属性，比如src

v-if和v-show都可以切换元素的显示状态，频繁的切换使用v-show

4.3 列表循环，表单元素绑定

v-for	v-on补充	v-model

v-for：

根据数据生产列表结构

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。

```
<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-on补充:

传递自定义参数，事件修饰符。

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

时间修饰符有很多种

文档传送门（常见的修饰符）https://cn.vuejs.org/v2/api/#v-on

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

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

```
    }
})
```

v-model:

获取和设置表单元素的值（双向数据绑定）

绑定的数据会和表单元素值相关联

绑定的数据<---->表单元素的值

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

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

demo-记事本

2021年6月27日 13:54

记事本功能：

1. 新增
    a. 生成列表结构(v-for 数组)
    b. 获取用户输入(v-model)
    c. 回车，新增数据(v-on .enter 添加数据)
2. 删除
    a. 点击删除指定内容(v-on splice 索引)
    b. 数据改变和数据绑定的元素会同步改变
    c. 事件的自定义参数
3. 统计
    a. 统计信息的个数(v-text length)
    b. 基于数据的开发方式
4. 清空
    a. 点击清空所有的信息(v-on)
5. 隐藏
    a. 没有数据时，隐藏元素(v-show v-if 数组非空)

总结：

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

扩展：

vue中的变异方法之splice，可以实现增删改功能。

增加：

```
function add ()  {
    this.list.splice(index,0,newItem)
}
```

修改：

```
function update () {
    this.list.splice(index,1,newItem)
}
```

删除：

```
function update ()  {
    this.list.splice(index,1)
}
```

```
<!-- 代码-->
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
    <title>记事本</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
    <meta name="robots" content="noindex, nofollow" />
    <meta name="googlebot" content="noindex, nofollow" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" type="text/css" href="./css/note.css" />
  </head>
  <body>
    <!-- 主体区域 -->
    <section id="todoapp">
      <!-- 输入框 -->
      <header class="header">
        <h1>记事本</h1>
        <input v-model = "inputValue" @keyup.enter="add" autofocus="autofocus"
         autocomplete="off" placeholder="请输入任务" class="new-todo" />
      </header>
      <!-- 列表区域 -->
      <section class="main">
        <ul class="todo-list">
          <li class="todo" v-for="(item,index) in list">
            <div class="view">
              <span class="index">{{index+1}}.</span>
              <label>{{item}}</label>
              <button class="destroy" @click="remove(index)"></button>
            </div>
          </li>
        </ul>
      </section>
      <!-- 统计和清空 -->
      <footer class="footer">
```

```
            <span class="todo-count" v-if="list.length!=0">
                <strong>{{list.length}}</strong> items left
            </span>
            <button class="clear-completed" @click="clear" v-show="list.length!=0">
              Clear
            </button>
          </footer>
        </section>
        <!-- 底部 -->
        <footer class="info">
          <p>
            <a href="http://www.itheima.com/">
                <img src="./images/02.jpg" alt="" />
            </a>
          </p>
        </footer>
        <!-- 开发环境版本，包含了有帮助的命令行警告 -->
        <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
        <script>
            var app = new Vue({
                el:"#todoapp",
                data:{
                    list:["青岛的大海","成都的街道","重庆的火锅","天津的摩天轮"],
                    inputValue:"长春的家"
                },
                methods:{
                    add:function(){
                        this.list.push(this.inputValue);
                    },
                    remove:function(index){
                        // console.log("删除");
                        // console.log(index);
                        this.list.splice(index,1);
                    },
                    clear:function(){
                        this.list = [];
                    }
                }
            })
        </script>
    </body>
</html>
```

五、网络应用

2021年6月28日 9:17

Vue结合网络数据开发应用

1. axios

   功能强大的网络请求库

   axios必须先导入才可以使用

   使用get和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

   ```
   <!-- 官网提供的 axios 在线地址-->
   <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,{key:value,key2:value2}).then(function(response){},function(err){})
   ```

2. axios+vue --结合vue一起

   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。

   和本地应用最大的区别就是改变了数据的来源。

   ```
   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){});
           }
       }
   })
   ```

3. demo天知道

功能：

1. 回车查询
    a. 按下回车(v-on .enter)
    b. 查询数据(axios 接口 v-model)
    c. 渲染数据(v-for 数组 that)

    注意：

    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

2. 点击查询
    a. 点击城市(v-on 自定义参数)
    b. 查询数据(this.方法())
    c. 渲染数据

    注意：

    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

天气接口：

请求地址：

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

六、综合应用

2021年6月28日 14:41

1. 歌曲搜索
   a. 按下回车(v-on .enter)
   b. 查询数据(axios 接口 v-model)
   c. 渲染数据(v-for 数组 that)

   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

   歌曲搜索接口

   请求地址:https://autumnfish.cn/search

   请求方法:get

   请求参数:keywords(查询关键字)

   响应内容:歌曲搜索结果
2. 歌曲播放
   a. 点击播放(v-on)
   b. 歌曲地址获取(接口 歌曲id)
   c. 歌曲地址设置(v-bind)

   歌曲id依赖歌曲搜索的结果，对于不用的数据也需要关注。

   歌曲url获取接口

   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url地址
3. 歌曲封面
   a. 点击播放(增加逻辑)
   b. 歌曲封面获取(接口 歌曲id)
   c. 歌曲封面设置(v-bind)

   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

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(包括封面信息)

4. 歌曲评论
   a. 点击播放(增加逻辑)
   b. 歌曲评论获取(接口 歌曲id)
   c. 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,地址中的type固定为0)

响应内容:歌曲的热门评论

5. 播放动画
   a. 监听音乐播放(v-on play)
   b. 监听音乐暂停(v-on pause)
   c. 操纵类名(v-bind 对象)

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

6. MV播放
   a. mv图标显示(v-if)
   b. mv地址获取(接口 mvid)
   c. 遮罩层(v-show v-on)
   d. 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表示没有mv)

响应内容:mv的地址

注意:

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在data中定义