2.1 前端基础之Html+CSS+JavaScript

2.1.6 vscode快捷键

快捷键	功能介绍
option	选中多行

2.1.11 格式化标签

语义	标签	说明
加粗	或者	推荐
倾斜	或者	推荐
删除线	或者	推荐
下划线	或者	推荐

2.1.12 图片标签

属性	属性值	说明
src	路径	必须写
alt	文本	图片无法显示时的代替文字
title	文本	鼠标放到图片上显示的文字
width		
height		
border		

2.1.13 有序列表和无序列表

标签	属性	取值
	type	disc，circle，square
	type	A，a，1，i，I
•		

2.1.14 超链接标签

属性	说明
href	
target	_blank：在新窗口中打开；_self：在自己的页面中打开

2.1.16 表格标签

```
<table> //表格标签
  <tr>  //行标签
    <th></th> //表头标签，内容加粗，居中
  </tr>
  <tr>
    <td></td> //表数据标签，内容不加粗，居左
  </tr>
</table>
```

标签	属性	描述
	border: 设置边框； width：宽度； height：高度； align：表格位置设定（left，right，center） cellspacing="0px"：双线变单线 bgcolor：设置背景颜色 background：设置背景图	父标签，相当于整个表格的容器
align： 内容的位置 bgcolor：设置背景颜色	定义行	
		colspan：单元格可横跨的列数 rowspan：单元格可横跨的行数 align：单元格内容的水平对齐方式（left，right，center）
		colspan：单元格可横跨的列数 rowspan：单元格可横跨的行数 align：单元格内容的水平对齐方式（left，right，center）

2.1.20 行合并与列合并练习

2.1.21 表单的介绍

一个完整表单由**表单域，表单控件和提示信息**组成。

2.1.22 表单域介绍

```
<form action="url地址" method="提交方式" name="表单域名城">
  各种表单元素控件
</form>
```

属性	属性值	说明
action	url地址	指定接收并处理表单数据的服务器的url地址
method	get/post	
name	名称	指定表单的名称

2.1.23 表单输入域标签介绍

```
<input type="text" />

<input type="password"/>

<input type="radio"/>

<input type="file"/>

<input type="checkbox"/>

<input type="submit"/>

<input type="reset"/>

<input type="button"/>

<input type="image"/>
```

2.1.24 下拉框和文本域介绍

```
<select>
  <option>北京</option>
</select>

<textarea cols="" rows="">
  ...
</textarea>
```

2.1.26～27 表单控件属性part01+02

输入标签 input

属性	属性值	说明
type	text； password； radio； file； checkbox； submit； reset； button； image； hidden；	文本； 密码； 单选框； 浏览文件； 多选框； 提交按钮，有提交表单的功能；可以通过value来改变现实的内容 重置按钮； 普通按钮； 图片； 隐藏域；
name		只要想把数据传输到后端时，必须要加name属性；radio和checkbox的name必须一致
value		提交到后台的数据；对于submit，reset，button，可以通过value来改变显示的内容；对于radio，checkbox要加value属性
checked		默认选中，对于radio和checkbox；
src		对于image
disable	ture/false	不能选中
readOnly	readOnly="readOnly"	只读，不能写操作

文本域标签textarea

属性	属性值	说明
name		
cols		
rows		

下拉框select/option

属性	属性值	说明
name		
value		给option添加value属性

```
<select name="...">
  <option value="...">北京</option>
</select>
```

2.1.28 get方式提交表单

name属性的属性值会作为key拼接到url后面，用户输出的值会作为value，进而形成键值对。

```
<form action="www.google.com">
  <input type="text" name="username"/>
</form>
```

www.google.com?username=xxxx

总结：

1. 提交的数据追加在请求的路径上
2. 因为请求路径长度有限，所以get请求提交的数据有限
3. 敏感数据会在url上显示，不适合密码等数据的提交

2.1.29 post方式提交表单

总结：

1. 提交的数据不追加到url上
2. 请求的数据比get方式大

2.1.30 注册表单案例

表单嵌套表格

2.1.33 引入css样式的三种方式

```
<link rel="stylesheet" type="text/css" href="">
```

2.1.35 css选择器的使用

id选择器 > 类选择器 > 元素选择器

2.1.36 css样式之边框

属性	属性值
border-stlye	dotted；dashed；solid；double
border-width	
border-color	

2.1.41 css样式之浮动

标准文档流/普通文档流：从上向下，从左到右依次布局

浮动：可以给指定的元素设置浮动，该元素就摆脱原来的标准文档流，漂浮在空中，可以进行左右移动

属性	属性值
float	left；right；none

2.1.43 盒子模型

margin：设置元素在页面的位置

padding：设置元素内容和边框的距离；副作用：会导致元素变形！！！

2.1.44 css案例

可以通过text-align来改变div中文字的位置

2.1.45 JavaScript的简介

JavaScript的组成：

1. ECMAScript：描述了该语言的基本语法，语句和基本对象
2. BOM：浏览器对象，处理网页内容的方法和接口
3. DOM：操作文档中的元素和内容

2.1.46 JavaScript的入门案例

```
<script type="text/javascript">
  ...
</script>
```

```
<script src=" "></script>
```

2.1.51 JavaScript中for循环语句

```
for(var i = 1; i <= 10; i++){

}

var arr = [1,2,3,4];
for(index in arr){   //遍历的是index

}

for(element of arr){ //遍历的是数组元素

}
```

2.1.52 JavaScript中函数的定义与调用

```
function method(Parameter p...){

}
methed(P);

var method = function(Patameter P...){

}
methid(P);
```

2.1.53 函数与事件的相关概念

事件源：

事件：

事件绑定：

事件触发：

2.1.54 ～ 56 事件绑定入门

事件名	描述
onclick	
ondblclick	
window.onload	页面加载完毕执行函数，当整个html页面加载完毕之后执行该事件关联的函数；**只执行一次，即使有多个**。
onload	某个页面或图像完成加载后触发这个事件
onsubmit	表单提交会触发这个事件，如果这个事件关联的函数返回true，表单就能正常提交；否则提交失败。
onblur	
onfocus	
onchange	
onmouseenter	
onmousedown	
onmouseup	
onmouseout	

2.1.57 JavaScript事件派发

不会修改原来的html中的元素。获取该元素，给该元素派发一个事件。

```
<input type="button" id="btn"/>
```

```
document.getElementById("btn").onclick = function(){...}
```

2.1.58 BOM浏览器对象模型

BOM对象

1. Screen对象
2. Navigator对象
3. Window对象：浏览器中打开的窗口
4. Histoty对象
5. Location对象

2.1.59 window对象常用方法

方法	描述
alert	提示框；
confirm	确认框；用户点击确认，返回true；否则返回false；
prompt	输入框；返回值是用户输入的值

凡是window的属性和方法，window调用过程中都可以不写window

2.1.60 ～ 61周期执行函数

方法	描述
setInterval(js代码/函数, 毫秒值)	反复执行
clearInterval(周期执行任务id)	
setTimeout(js代码/函数, 毫秒值)	执行一次
clearTimeout(周期执行任务id)	

2.1.62 全局转换方法

将字符串转换成number类型

parseInt()/parseFloat()

2.1.63 location对象使用

属性	描述
herf	完成一个地址的跳转

方法	描述
reload()	刷新

2.1.64 dom文档对象模型介绍

标签元素

属性元素

文本元素

作用：对元素进行增删改查

2.1.65 获取dom元素

方法	描述
getElementById()	
getElementByName()	
getElementByClassName()	
getElementByTagName()	

2.1.66 dom操作标签体内容

innerHtml和innerText

2.1.67 dom操纵属性

setAttribute()	
getAttribute()	
removeAttribute()	

2.1.68 内置对象String

String对象的创建

```
var str = "abc";  //string
var str1 = new String("abc"); //object
```

方法	描述
charAt()	
indexOf()	
lastIndexOf()	
split()	
substring(begin, end)	
substr(begin)	
substr(begin, length)	

2.1.69 内置对象Array

Array对象的创建

```
var arr1 = new Array();
var arr2 = new Array(3);
var arr3 = new Array(1,true,"true");
```

注意：

1. 没有数组下标异常
2. 数组的长度是不固定的，如果超出范围会自动扩充
3. 数组可以包含多种数据类型

2.1.70 轮播图案例

2.2 前段进阶之jQuery+Ajax+Vue

2.2.2 jQuery基本概念

jQuery是一个javascipt库。jQuery对dom进行了封装。

2.2.5 jQuery对象和dom对象相互转换

jQuery到dom

```
<input type="text" name="username" id="uid" value="jack">

<script>
  var $user = $("#uid");
  var user = $user[0];
  var user = $user.get(0);
</script>
```

dom到jQuery

```
<input type="text" name="username" id="uid" value="jack">

<script>
  var user = document.getElementById("uid");
  var $user = $(user);
</script>
```

2.2.6 jQuery页面加载函数

jQuery页面加载函数可以执行对次，和原生态的windom.onload不同

方式一

$(document).ready(function(){

})

方式二

$(function(){

})

2.2.7 jQuery事件绑定与事件派发

事件派发需要写在页面加载函数中

2.2.8 jQuery选择器-基本选择器

2.2.9 jQuery选择器-层级选择器

选择器名称	语法	说明
后代选择器	$("A B")	选择A元素内部的所有B元素
子选择器	$("A>B")	选择A元素内部的所有为直接子元素的B元素

2.2.11 jQuery选择器-基本过滤选择器

选择器名称	语法	说明
首元素选择器	:first	获得选择的元素中的第一个元素
尾元素选择器	:last	获得选择的元素中的最后一个元素
偶数选择器	:even	索引值为偶数；从0开始
奇数选择区	:odd	
等于索引选择器	:eq(index)	
大于索引选择器	:gt(index)	
小于索引选择器	:lt(index)	

2.2.12 jQuery选择器-表单选择器

2.2.13 ~ 17 jQuery的dom操作-文本和value (待看)

```
<input type="button" value="btn" onclick=fnc(this)>
```

如果在事件绑定的方法中传入this，this指的是当前input标签的**dom对象**

2.2.18 jQuery的第一种遍历方式

```
<ul>
  <li>北京</li>
  <li>上海</li>
  <li>杭州</li>
</ul>
```

```
var $liArr = $("li");
for(var i = 0; i < $liArr.length; i++){
  //每次遍历的结果是dom对象
  alert($liArr[i]);

  //把dom对象转换成jQuery对象
  var city = $($liArr[i]).html();
  alert(city);
}
```

2.2.19 jQuery的第二种遍历方式

```
$("li").each(function(index, element){
  var city = $(element).html();
  alert(city);
})
```

2.2.20 jQuery的第三种和第四种遍历方式

```
$.each($("li"),function(index,element){
  var city = $(element).html();
  alert(city);
})
```

```
for(element of $("li")){
  var city = $(element).html();
  alert(city);
}
```

2.2.21 jQuery的动画-显示与隐藏

方法名称	解释
show(speed,[easing],[fn])	
hide(speed,[easing],[fn])	
toggle(speed,[easing],[fn])	

参数名称	解释
speed	三种预定速度之一的字符串("slow","normal","fast")或表示动画时长的毫秒值
easing	切换效果
fn	在动画完成后执行的函数，每个元素执行一次

2.2.22 jQuery的动画-滑入与滑出

方法名称	解释
slideDown(speed,[easing],[fn])	
slideUp(speed,[easing],[fn])	
slideToggle(speed,[easing],[fn])	

2.2.23 jQuery的链式编程

2.2.24 jQuery的animate自定义动画

$(selector).animate({params}, speed, easing, callback)

2.2.25 弹幕效果

2.2.26 ajax和json内容介绍

2.2.27 ajax的概念

Ajax是一种在无需要重新加载整个网页的情况下，能够更新部分网页的技术。通过在后台与服务器少量数据交换，ajax可以使网页实现异步更新。

应用：

1. 数据验证
2. 按需求获取数据
3. 自动更新页面内容

异步：

客户端发送一个异步请求，请求发送给服务器。服务器处理的这段时间内，客户端看到的内容不变，直到服务器处理完毕，通过一个**XMLHttpRequest**对象将服务器响应的结果带回客户端，进行一个局部更新。

2.2.28 服务器Servlet简单实用

tomcat应用服务器

Servlet: 后台技术，可以接受用户请求，给用户提供响应

2.2.29 原生js放送ajax请求

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function send(){
            //1. 创建ajax引擎对象
            const xhr = new XMLHttpRequest();

            //2. 为ajax引擎对象绑定监听事件
            //onreadystatechange 监听核心对象XMLHttpRequest对象的事件
            xhr.onreadystatechange = function (){
                //readystate 执行的状态 1 2 3 4（响应结束）
                //state 响应码 代表服务器响应成功
                if(xhr.readyState == 4 && xhr.status == 200){
                    //接受响应数据
                    alert(xhr.responseText);
                }
            }

            //3. 绑定提交地址
            xhr.open("get","/demo/DemoServlet",true);

            //4. 发送请求
            xhr.send();
        }
    </script>
</head>

<body>
     <input type="button" value="button" onclick="send()">
</body>
</html>
```

2.2.30 jQuery发送get方式ajax请求

方法	描述
get请求	$.get(url,[data],[callback],[type])
post请求	
ajax请求	

参数	描述
url	必须填写，请求服务器的地址
data	发送给服务器的参数 key=value，服务器可以根据key值获取value
callback	回调函数，形式参数data代表服务器返回的数值
type	服务器返回的数据类型，text/json等

2.2.31 jQuery发送post方式ajax请求

2.2.32 jQuery发送ajax请求

$.ajax([settings])

属性	描述
url	
async	
type	
dataType	
success	
error	

```
$.ajax({
  url: "Demo/DemoServlet",
  async: true,
  data: "username=tom&age=18",
  type: "post",
  dataType: "text",
  success: function(data){

  },
  error: function(){

  }
})
```

2.2.33 json的基本概念

2.2.35 jackson转换工具

java对象转成json对象

ObjectMapper对象：可以将java对象转换成json对象

writeValueAsString(java对象)：返回一个json对象

@JsonFormat(pattern = "YYYY-MM-dd")

@JsonIgnore

2.2.36 jackson转换不同的数据类型

2.2.37~39

注意：

1. jar包放到web/WEB-INFO下的lib包
2. javascript文件放到web下的js包

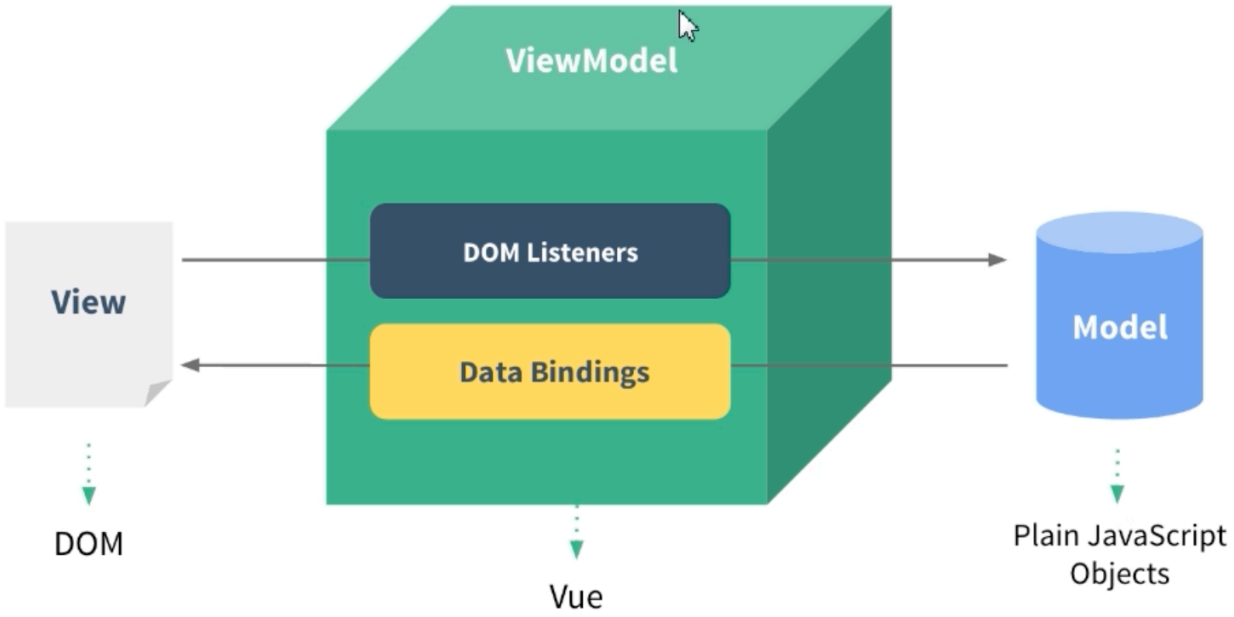
2.2.40 vue课程介绍

2.2.41 vue简介

1. 构建用户界面的渐进式框架
2. 核心库只关注视图层

MVVM：Model-View-ViewModel

1. 本质上是MVC的进阶版
2. Model：负责数据存储
3. View：负责页面展示
4. ViewModel：负责业务逻辑处理（比如ajax请求等），对数据进行加工后交给视图处理



2.2.42 vue入门案例

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
</head>

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

<!--创建vue对象-->
<script>
  var app = new Vue({
    el: "#app",
```

```
    data:{
      message:"Hello Vue!"
    }
   })
</script>
</html>
```

2.2.43 插值表达式

作用：使用vue中的数据替换html的内容

能写哪些内容：

1. 除了能写data中的数据，还可以编写一些数学运算
2. 不能写语句，能够写三元运算符

范围：一定是vue对象接管的范围内，超出范围不会生效，不会被解析

2.2.44 EL挂载点

注意：

1. 可以使用其他选择器
2. html和body元素不能被vue对象接管

2.2.45 data数据对象

data中的数据可以是哪些：

1. 字符串
2. 数值型内容
3. 对象类型
4. 布尔类型
5. 数组类型

2.2.46 vue中指令的介绍

指令	作用
v-if	
v-text	用的很少
v-html	用的很少

2.2.47 v-text指令

作用和差值表达式很像，区别：

1. v-text替换所有内容，而差值表达式只替换部分内容
2. v-text拼接时加单引号，而差值表达式可以是双引号

2.2.48 v-html指令

v-text和v-html的区别：

1. v-text只设置元素之间的文本
2. v-html可以设置元素之间的文本和html元素

2.2.49 v-if 和 v-show指令

v-if：从dom中移除

v-show：等于display:none

2.2.50 v-on指令

格式1： v-on:事件 = "函数名"

格式2：@事件="函数名"

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
</head>
<body>
    <div id="app">
      <button v-on:click="add">count增加</button>
      <button @click="minus">count减少</button>
      <hr>
      <h2>{{count}}</h2>
      <hr>
      <button @click="fun1(count)">获取count</button>
      <hr>
      <!--获取事件-->
      <input @keydown="fun2($event)">
    </div>
</body>
```

```
<!--创建vue对象-->
<script>
  var app = new Vue({
      el: "#app",
      data:{
          count:1
      }
      methods:{
        add:function(){
          this.count += 1;
        }
        minus:function(){
          this.count -= 1;
        }
        fun1:function(c){
          alert(c)
        }
        fun2:function(e){
          if(e.keyCode > 57 || e.keyCode < 48){
            //阻止事件发生
            e.preventDefault()
          }
        }
      }
  })
</script>
</html>
```

2.2.52 v-on指令修饰符

格式1： v-on:事件.修饰符 = "函数名"

格式2：@事件.修饰符="函数名"

修饰符	
once	方法只被调用一次
enter	只有回车键才能触发

2.2.53 v-for指令

格式：v-for="(元素, 索引) in 数组名/对象"

2.2.54 v-bind指令

绑定元素的属性值

格式1：v-bind:属性名 = "data中的值"

格式2：:属性名 = "data中的值"

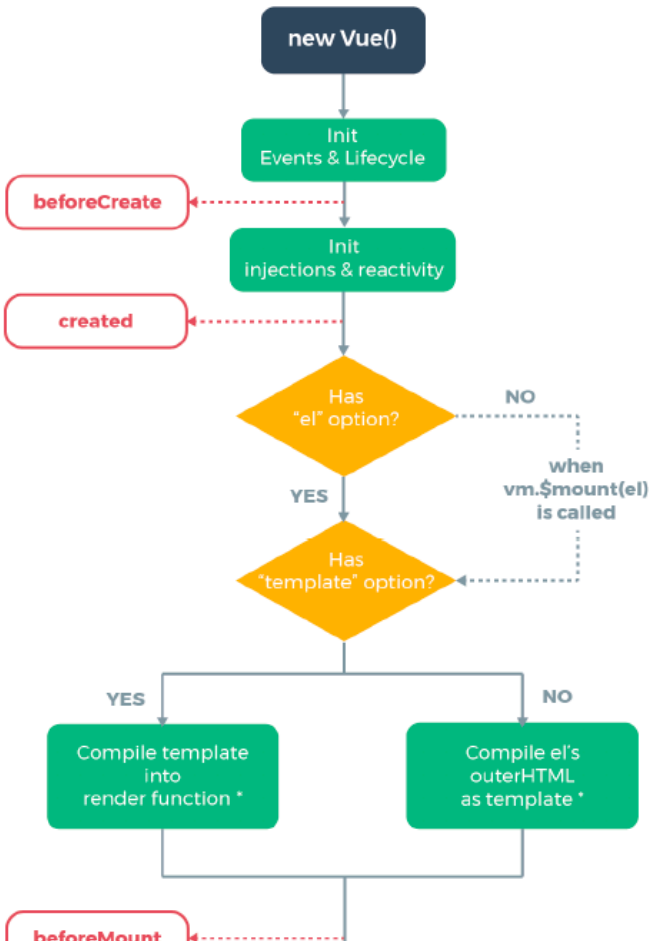
格式3：v-bind:属性名 = "{属性名: data中的值}"

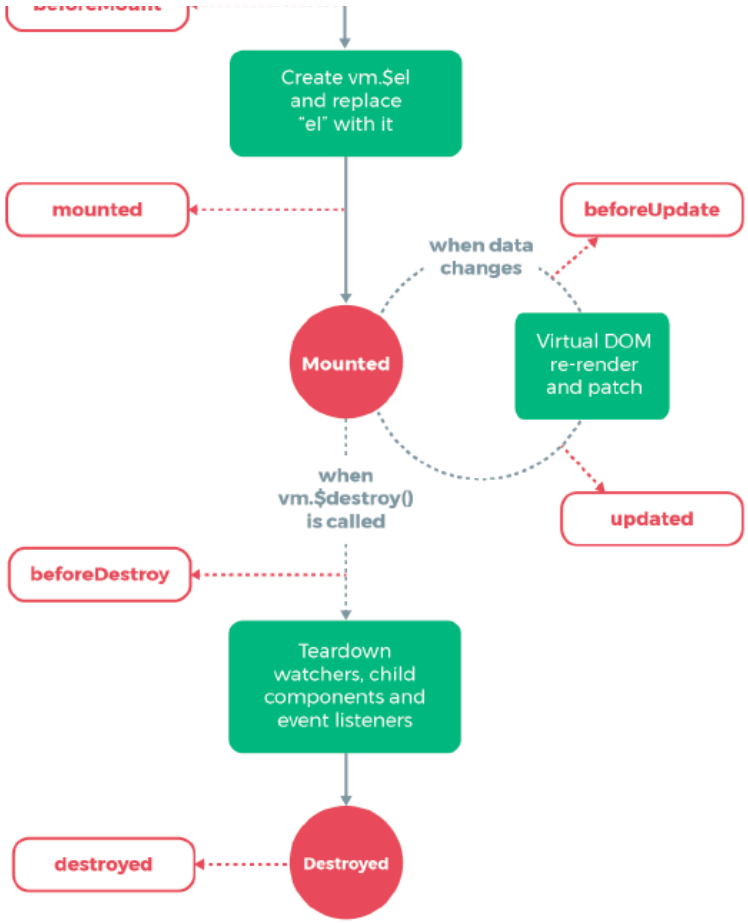
格式4：v-bind:属性名 = "[data中的值1, data中的值2...]"

2.2.55 v-model指令

获取和设置表单元素的值（双像绑定）

2.2.59 vue的声明周期





生命周期	说明
beforeCreate	该函数执行时，data中的数据和methods中的函数还没有初始化
created	此方法可以获取data中的数据和调用methods中的函数
beforeMount	可以获取内存中的值，但不能获取到页面上渲染的值。也就是说明值还没有被渲染到页面上。
mounted	实现内存和页面的数据同步
beforeUpdate	当数据发生变化时，该函数触发。内存中的值已经改变，但页面还没有更新
updated	内存中的数据更新，页面也已经渲染为最新的数据
beforeDestroy	
destroyed	

2.2.60 vue中axios异步请求方式

Get方法

格式1：axios.get(地址?key1=value1&key2=value2).then(function(response){}, function(error){});

格式2：axios.get(地址, {params:{key1:value1,key2:value2}}).then(function(response){}, function(error){});

```
<script>
  new Vue({
    el:"#app",
    data:{
      jokes: ""
    }
    methods:{
      createJokes(){
        var temp = this; //必须用变量记录this
        axios.get("https://autumnfish.cn/api/joke/list?
num=3").then(function(response){
          temp.jokes = response.data.jokes;
        }, function(error){})
      }
    }
  });
</script>
```

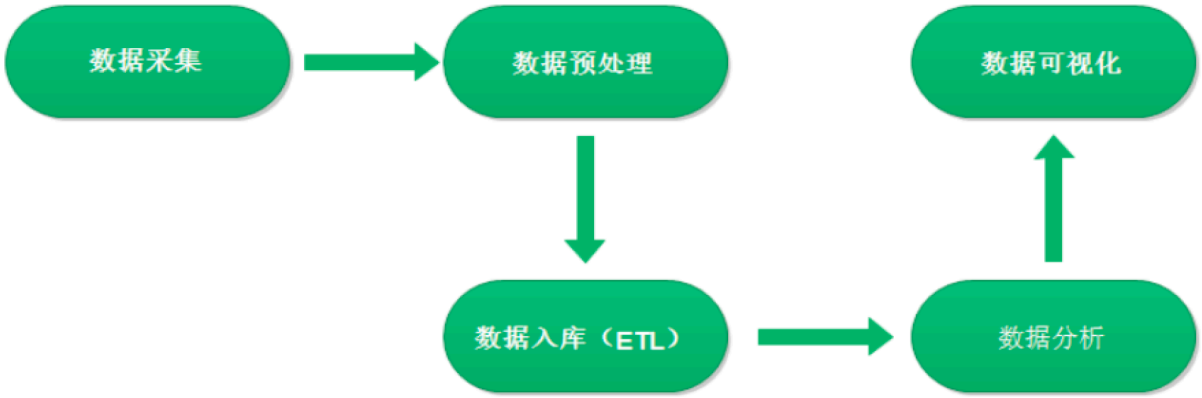
Post方法

axios.post(地址, {key1:value1,key2:value2}).then(function(response){}, function(error){});

注意⚠️：axios请求中不能直接使用this！！！

2.3 Highcharts+ECharts数据可视化

2.3.1 数据可视化简介



2.3.4 系统的架构

2.3.5 Highcharts介绍

highcharts， hightstock，highmaps

2.3.15 Echarts介绍及入门

End