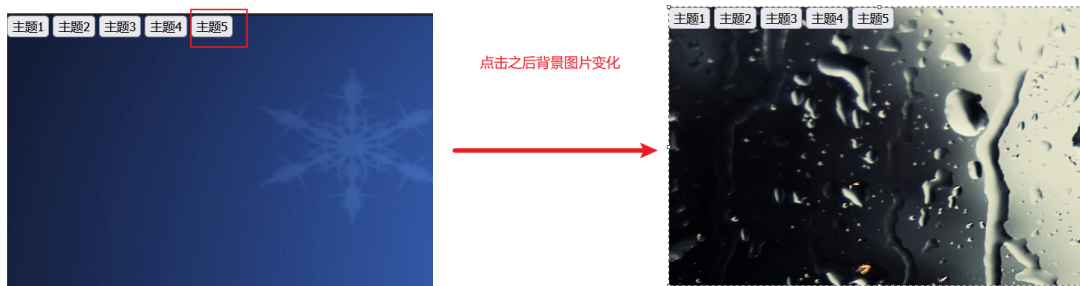


# 1 JavaScript

html完成了架子，css做了美化，但是网页是死的，我们需要给他注入灵魂，所以接下来我们需要学习JavaScript，这门语言会让我们的页面能够和用户进行交互。

## 1.1 介绍

通过**代码/js效果演示**提供资料进行效果演示，通过浏览器打开，我们点击主题5按钮，页面的主题发生了变化，所以js可以让我们的页面更加的智能，让页面和用户进行交互。



## 1.2 引入方式

同样，js代码也是书写在html中的，那么html中如何引入js代码呢？主要通过下面的2种引入方式：

**第一种方式：**内部脚本，将JS代码定义在HTML页面中

- JavaScript代码必须位于<script></script>标签之间
- 在HTML文档中，可以在任意地方，放置任意数量的<script>
- 一般会把脚本置于<body>元素的底部，可改善显示速度

例子：

```
1 <script>
2     alert("Hello JavaScript")
3 </script>
```

**第二种方式：**外部脚本将，JS代码定义在外部JS文件中，然后引入到HTML页面中

- 外部JS文件中，只包含JS代码，不包含<script>标签
- 引入外部js的<script>标签，必须是双标签

例子：

```
1 <script src="js/demo.js"></script>
```

注意：demo.js中只有js代码，没有<script>标签

接下来，我们通过VS Code来编写代码，演示html中2种引入js的方式

第一步：在VS Code中创建名为 10.JS-引入方式.html 的文件

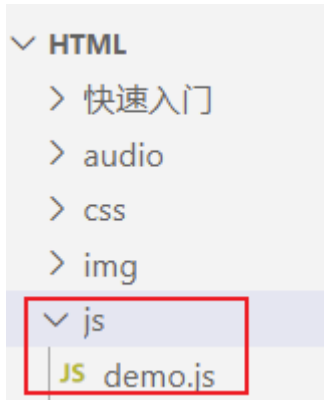
第二步：按照上述第一种内部脚本的方式引入js，编写如下代码：

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7     <title>JS-引入方式</title>
8     <!-- 内部脚本 -->
9     <script>
10         alert('Hello JS');
11     </script>
12 </head>
13 <body>
14 </body>
15 </html>
```

第三步：浏览器打开效果如图所示：



第四步：接下来演示外部脚本，注释掉内部脚本，然后在css目录同级创建js目录，然后创建一个名为demo.js的文件：



第五步：在demo.js中编写如下js内容：

```
1 alert('Hello JS2');
```

第六步：注释掉之前的内部脚本代码，添加<script>标签来引入外部demo.js文件，具体代码如下：

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7     <title>JS-引入方式</title>
8     <!-- 内部脚本 -->
9     <!-- <script>
10         alert('Hello JS');
11     </script> -->
12
13     <!-- 外部脚本 -->
14     <script src="js/demo.js"></script>
15 </head>
16 <body>
17
18 </body>
19 </html>
```

第七步：浏览器刷新效果如图：



## 1.3 基础语法

### 1.3.1 书写语法

掌握了js的引入方式，那么接下来我们需要学习js的书写了，首先需要掌握的是js的书写语法，语法规则如下：

- **区分大小写**：与 Java 一样，变量名、函数名以及其他一切东西都是区分大小写的
- 每行结尾的分号可有可无
- 大括号表示代码块
- 注释：
  - 单行注释：// 注释内容
  - 多行注释：/\* 注释内容 \*/

我们需要借助js中3种输出语句，来演示书写语法

api	描述
<code>window.alert()</code>	警告框
<code>document.write()</code>	在HTML 输出内容
<code>console.log()</code>	写入浏览器控制台

接下来我们选用通过VS Code，接触3种输入语句，来演示js的书写语法

第一步：在VS Code中创建名为 11.JS-基础语法-输出语句.html的文件

第二步：按照基本语法规则，编写3种输出语句的代码，并且添加注释，具体代码如下；

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>JS-基本语法</title>
8  </head>
9  <body>
10
11 </body>
12 <script>
13     /* alert("JS"); */
14     //方式一：弹出警告框
15     window.alert("hello js");
16 </script>
17 </html>
```

浏览器打开如图所示效果：



我们注释掉上述代码，添加代码 `document.write("hello js");` 来输出内容：

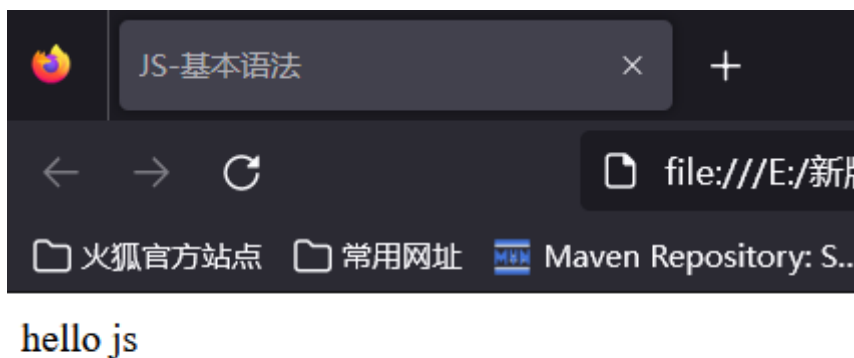
```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>JS-基本语法</title>
8  </head>
9  <body>
10
```

```

11 </body>
12 <script>
13     /* alert("JS"); */
14
15     //方式一：弹出警告框
16     // window.alert("hello js");
17
18     //方式二：写入html页面中
19     document.write("hello js");
20
21 </script>
22 </html>

```

刷新浏览器，效果如图所示：



最后我们使用`console.log("hello js");` 写入到控制台，并且注释掉之前的代码：

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7     <title>JS-基本语法</title>
8 </head>
9 <body>
10
11 </body>
12 <script>
13     /* alert("JS"); */
14
15     //方式一：弹出警告框
16     // window.alert("hello js");
17

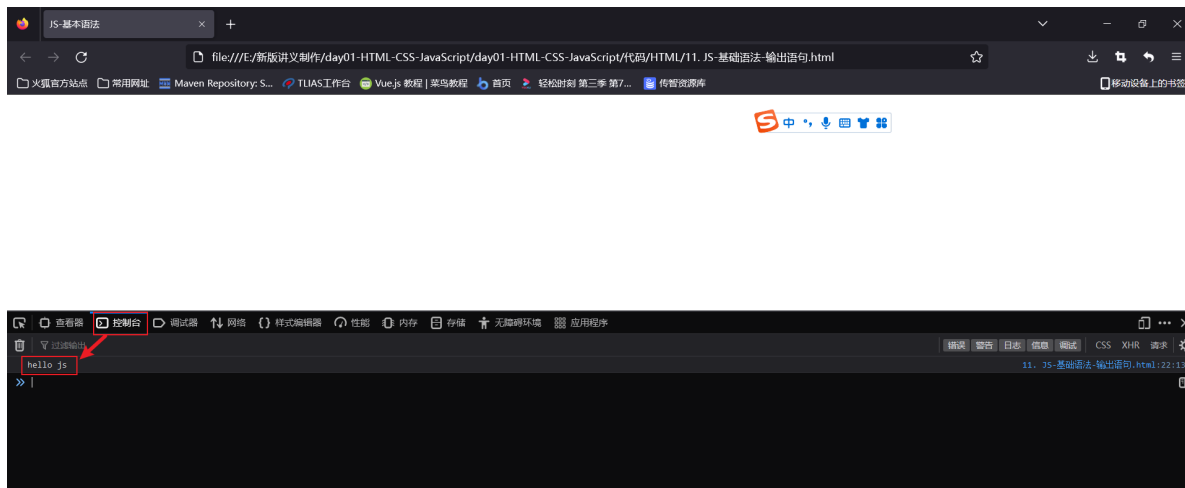
```

```

18      // //方式二：写入html页面中
19      // document.write("hello js");
20
21      //方式三：控制台输出
22      console.log("hello js");
23  </script>
24  </html>


```

浏览器f12抓包，去控制台页面，如图所示：



### 1.3.2 变量

书写语法会了，变量是一门编程语言比不可少的，所以接下来我们需要学习js中变量的声明，在js中，变量的声明和java中还是不同的。首先js中主要通过如下3个关键字来声明变量的：

关键字	解释
var	早期ECMAScript5中用于变量声明的关键字 
let	ECMAScript6中新增的用于变量声明的关键字，相比较var，let只在代码块内生效
const	声明常量的，常量一旦声明，不能修改

在js中声明变量还需要注意如下几点：

- JavaScript 是一门弱类型语言，变量可以存放不同类型的值。
- 变量名需要遵循如下规则：
  - 组成字符可以是任何字母、数字、下划线（\_）或美元符号（\$）
  - 数字不能开头
  - 建议使用驼峰命名

接下来我们需要通过VS Code编写代码来演示js中变量的定义

第一步：在VS Code中创建名为 12.JS-基础语法-变量.html的文件：

第二步：编写代码：

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>JS-基础语法</title>
8  </head>
9  <body>
10
11 </body>
12 <script>
13
14     //var定义变量
15     var a = 10;
16     a = "张三";
17     alert(a);
18
19 </script>
20 </html>
```

可以看到浏览器弹出张三



在js中，我们var声明的变量可以接受任何数据类型的值。并且var声明的变量的作用于是全局的，注释掉之前的代码，添加如下代码：



```
1  <script>
2      //var定义变量
3      // var a = 10;
4      // a = "张三";
5      // alert(a);
6
7      //特点1 ： 作用域比较大，全局变量
8      {
9          var x = 1;
10     }
11     alert(x);
12 </script>
```

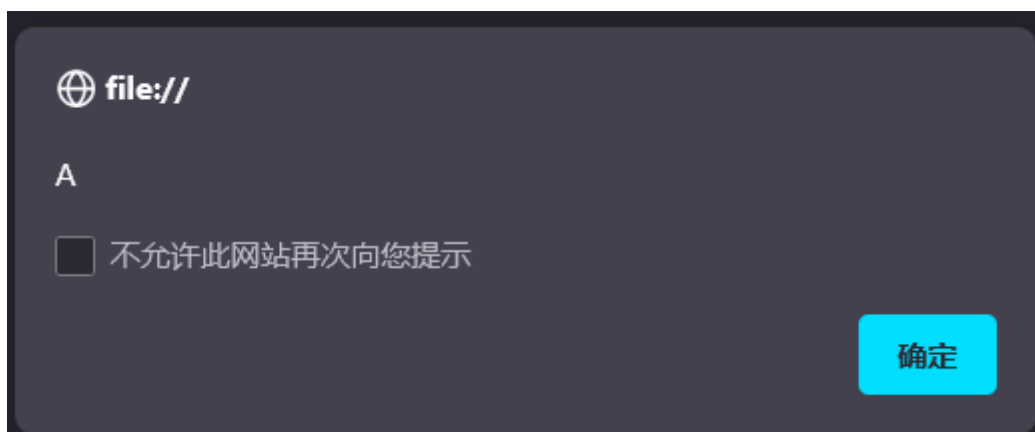
浏览器照样成功弹出：



而且var关键字声明的变量可以重复定义，修改代码如下：

```
1  {
2      var x = 1;
3      var x = "A";
4  }
5  alert(x);
6
```

浏览器弹出内容是A



所以在ECMAScript 6 新增了 **let**关键字来定义变量，它的用法类似于 **var**，但是所声明的变量，只在 **let**关键字所在的代码块内有效，且不允许重复声明。注释掉之前的代码，添加代码如下：

```
1  <script>
2
3      //var定义变量
4      // var a = 10;
5      // a = "张三";
6      // alert(a);
7
8      //特点1 ： 作用域比较大，全局变量
9      //特点2 ： 可以重复定义的
10     // {
11     //     var x = 1;
12     //     var x = "A";
13     // }
14     // alert(x);
15
16     //let ： 局部变量 ； 不能重复定义
17     {
18         let x = 1;
19     }
20     alert(x);
21
22 </script>
```

浏览器打开，f12抓包，来到控制台页面，发现报错，变量没有定义，说明**let**声明的变量在代码块外不生效



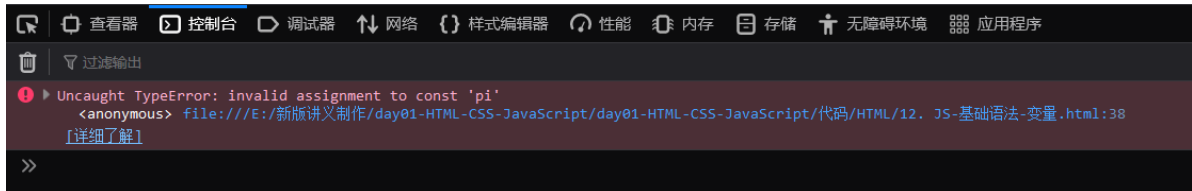
接着我们使用**let**重复定义变量，代码修改如下：发现**idea**直接帮我们报错了，说明**let**声明的变量不能重复定义

```
//let ： 局部变量 ； 不能重复定义
{
    let x = 1;
    let x = 2;
    alert(x);
}
```

在ECMAScript6中，还新增了const关键字用来声明常量，但是一旦声明，常量的值是无法更改的。注释之前的内容，添加如下代码：

```
1      const pi = 3.14;
2      pi = 3.15;
3      alert(pi);
```

浏览器f12抓包，来到控制台页面发现直接报错了，



关于变量的讲解我们就此结束，完整代码如下：

```
1      <!DOCTYPE html>
2      <html lang="en">
3      <head>
4          <meta charset="UTF-8">
5          <meta http-equiv="X-UA-Compatible" content="IE=edge">
6          <meta name="viewport" content="width=device-width, initial-
7              scale=1.0">
8          <title>JS-基础语法</title>
9      </head>
10     <body>
11
12     <script>
13
14         //var定义变量
15         // var a = 10;
16         // a = "张三";
17         // alert(a);
18
19         //特点1 ： 作用域比较大，全局变量
20         //特点2 ： 可以重复定义的
21         // {
22         //     var x = 1;
23         //     var x = "A";
24         // }
25         // alert(x);
26
```

```

27      //let : 局部变量 ; 不能重复定义
28      // {
29      //     let x = 1;
30      //     alert(x);
31      // }
32
33      //const: 常量 , 不能给改变的.
34      const pi = 3.14;
35      pi = 3.15;
36      alert(pi);
37
38  </script>
39  </html>

```

### 1.3.3 数据类型和运算符

虽然js是弱数据类型的语言，但是js中也存在数据类型，js中的数据类型分为：原始类型 和 引用类型，具体有如下类型

数据类型	描述
number	数字（整数、小数、NaN(Not a Number)）
string	字符串， <b>单双引皆可</b>
boolean	布尔。true, false
null	<b>对象</b> 为空
undefined	当声明的变量 <b>未初始化</b> 时，该变量的 <b>默认值是 undefined</b>

使用typeof函数可以返回变量的数据类型，接下来我们需要通过书写代码来演示js中的数据类型

第一步：在VS Code中创建名为13. JS-基础语法-数据类型.html的文件

第二步：编写如下代码，然后直接挨个观察数据类型：

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">

```

```
6     <meta name="viewport" content="width=device-width, initial-  
    scale=1.0">  
7     <title>JS-数据类型</title>  
8 </head>  
9 <body>  
10  
11 </body>  
12 <script>  
13  
14     //原始数据类型  
15     alert(typeof 3); //number  
16     alert(typeof 3.14); //number  
17  
18     alert(typeof "A"); //string  
19     alert(typeof 'Hello');//string  
20  
21     alert(typeof true); //boolean  
22     alert(typeof false); //boolean  
23  
24     alert(typeof null); //object  
25  
26     var a ;  
27     alert(typeof a); //undefined  
28  
29 </script>  
30 </html>
```

熟悉了js的数据类型了，那么我们需要学习js中的运算法，js中的运算规则绝大多数还是和java中一致的，具体运算符如下：

运算规则	运算符
算术运算符	+ , - , * , / , % , ++ , --
赋值运算符	= , += , -= , *= , /= , %=
比较运算符	> , < , >= , <= , != , == , = 注意 == 会进行类型转换, = 不会进行类型转换
逻辑运算符	&& ,    , !
三元运算符	条件表达式 ? true_value: false_value

接下来我们通过代码来演示js中的运算法，主要记忆js中和java中不一致的地方

第一步：在VS Code中创建名为14. JS-基础语法-运算符.html的文件

第二步：编写代码

在js中，绝大多数的运算规则和java中是保持一致的，但是js中的==和===是有区别的。

- ==：只比较值是否相等，不区分数据类型，哪怕类型不一致，==也会自动转换类型进行值得比较
- ===：不光比较值，还要比较类型，如果类型不一致，直接返回false

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>JS-运算符</title>
8  </head>
9  <body>
10
11 </body>
12 <script>
13     var age = 20;
14     var _age = "20";
15     var $age = 20;

```

```

16
17     alert(age == _age); //true , 只比较值
18     alert(age === _age); //false , 类型不一样
19     alert(age === $age); //true , 类型一样, 值一样
20
21 </script>
22 </html>

```

在js中, 虽然不区分数据类型, 但是有时候涉及到数值计算, 还是需要进行类型转换的, js中可以通过 `parseInt()` 函数来进行将其他类型转换成数值类型。注释之前的代码, 添加代码如下:

```

1 // 类型转换 - 其他类型转为数字
2 alert(parseInt("12")); //12
3 alert(parseInt("12A45")); //12
4 alert(parseInt("A45")); //NaN (not a number)

```

除此之外, 在js中, 还有非常重要的一点是: `0, null, undefined, "", NaN` 理解成 `false`, 反之理解成 `true`。注释掉之前的代码, 添加如下代码:

```

1 if(0){ //false
2     alert("0 转换为false");
3 }

```

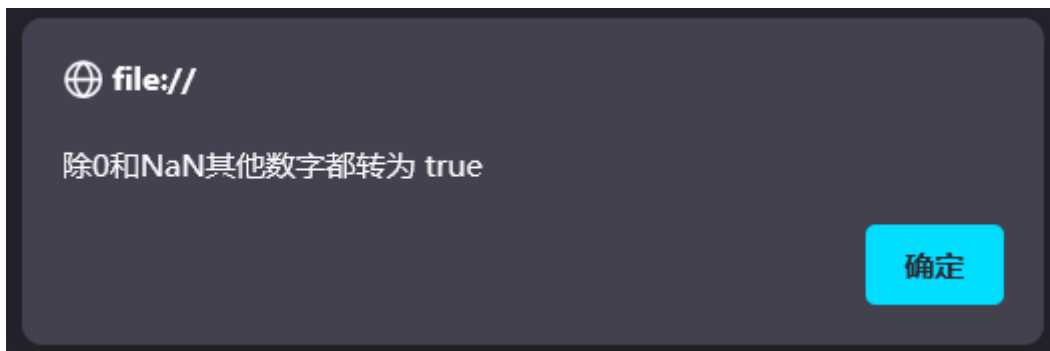
浏览器刷新页面, 发现没有任何弹框, 因为0理解成false, 所以条件不成立。注释掉上述代码, 添加如下代码:

```

1 if(1){ //true
2     alert("除0和NaN其他数字都转为 true");
3 }

```

浏览器刷新, 因为1理解成true, 条件成立, 所以浏览器效果如下;



其他情况可以一一演示, 完整演示代码如下:

```

1 // if(0){ //false
2 //     alert("0 转换为false");

```

```

3      // }
4      // if(NaN){ //false
5      //      alert("NaN 转换为false");
6      // }
7      if(1){ //true
8          alert("除0和NaN其他数字都转为 true");
9      }
10
11      // if(""){ //false
12      //      alert("空字符串为 false, 其他都是true");
13      // }
14
15      // if(null){ //false
16      //      alert("null 转化为false");
17      // }
18      // if(undefined){ //false
19      //      alert("undefined 转化为false");
20      // }

```

流程控制语句if, switch, for等和java保持一致, 此处不再演示

**需要注意的是:** 在js中, 0,null,undefined,"",NaN理解成false,反之理解成true

## 1.4 函数

在java中我们为了提高代码的复用性, 可以使用方法。同样, 在JavaScript中可以使用函数来完成相同的事情。JavaScript中的函数被设计为执行特定任务的代码块, 通过关键字function来定义。接下来我们学习一下JavaScript中定义函数的2种语法

### 1.4.1 第一种定义格式

第一种定义格式如下:

```

1      function 函数名(参数1, 参数2...){
2          要执行的代码
3      }

```



因为JavaScript是弱数据类型的语言，所以有如下几点需要注意：

- 形式参数不需要声明类型，并且JavaScript中不管什么类型都是let或者var去声明，加上也没有意义。
- 返回值也不需要声明类型，直接return即可

如下示例：

```
1  function add(a, b){  
2      return a + b;  
3  }
```

接下来我们需要在VS Code中编写代码来演示

第一步：新建名为js的文件夹，创建名为01. JS-函数的html文件，然后在<script>中定义上述示例的函数：

```
1  <script>  
2      function add(a,b){  
3          return a + b;  
4      }  
5  </script>
```

但是上述只是定义函数，**函数需要被调用才能执行！** 所以接下来我们需要调用函数

第二步：因为定义的add函数有返回值，所以我们可以接受返回值，并且输出到浏览器上，添加如下代码：

```
1  let result = add(10,20);  
2  alert(result);
```

查看浏览器运行结果：浏览器弹框内容如下图所示：



## 1.4.2 第二种定义格式

第二种可以通过`var`去定义函数的名字，具体格式如下：

```
1  var functionName = function (参数1, 参数2...){
2      //要执行的代码
3  }
```

接下来我们按照上述的格式，修改代码如下：只需要将第一种定义方式注释掉，替换成第二种定义方式即可，函数的调用不变

```
1  <script>
2
3      //定义函数-1
4      // function add(a,b){
5      //     return a + b;
6      // }
7
8      //定义函数-2
9      var add = function(a,b){
10         return a + b;
11     }
12
13
14     //函数调用
15     var result = add(10,20);
16     alert(result);
17
18 </script>
```

浏览器弹框效果和上述一致



我们在调用`add`函数时，再添加2个参数，修改代码如下：

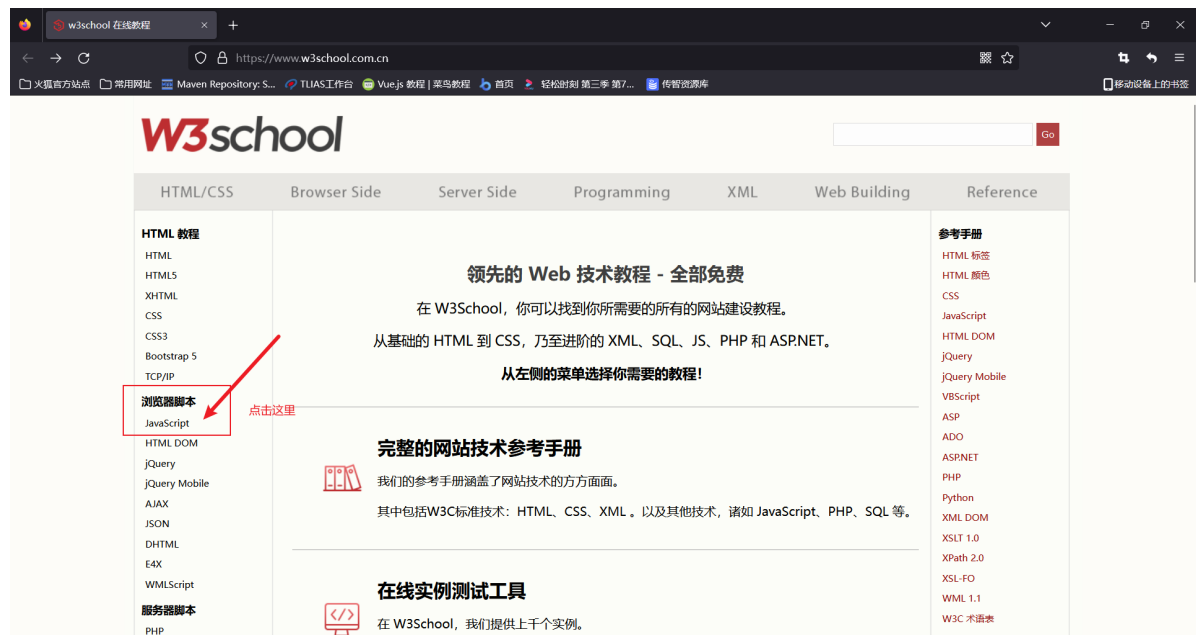
```
1  var result = add(10,20,30,40);
```

浏览器打开，发现没有错误，并且依然弹出30，这是为什么呢？

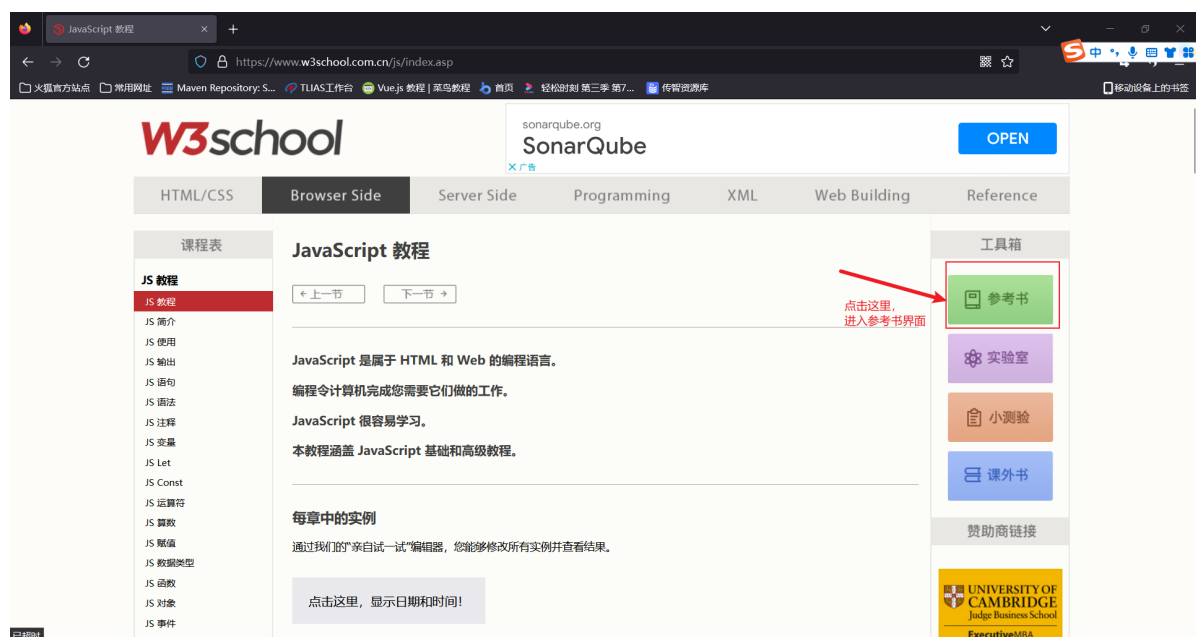
因为在JavaScript中，函数的调用只需要名称正确即可，参数列表不管的。如上述案例，10传递给了变量a，20传递给了变量b，而30和40没有变量接受，但是不影响函数的正常调用。

## 1.5 JavaScript对象

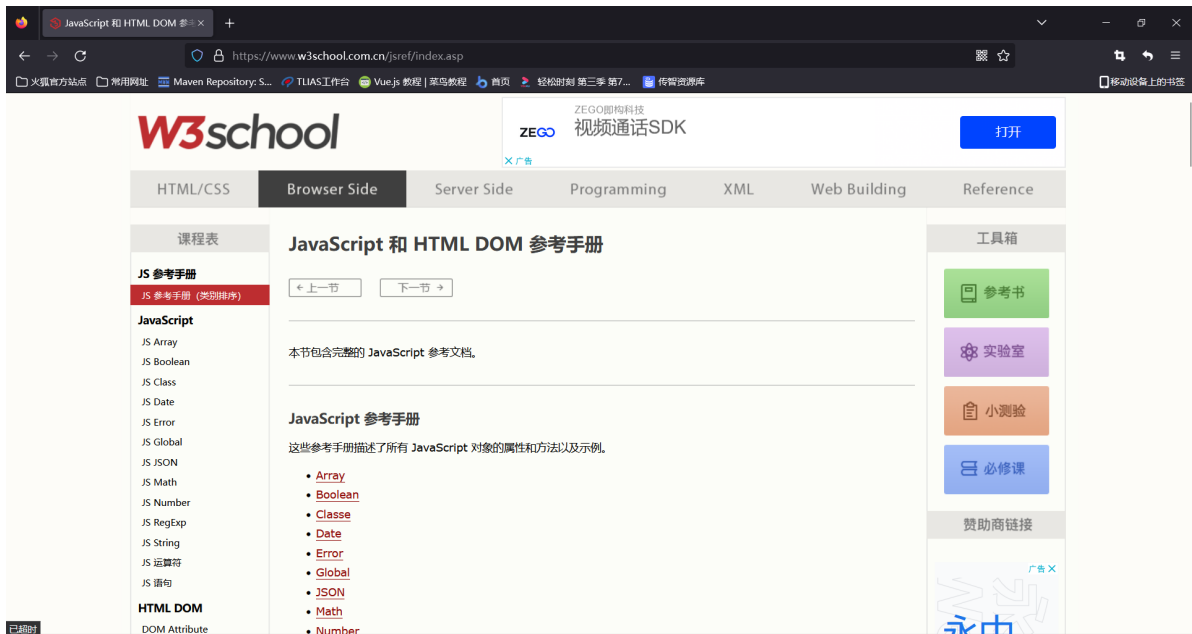
JavaScript中的对象有很多，主要可以分为如下3大类，我们可以打开w3school在线学习文档，来到首页，在左侧栏找到浏览器脚本下的JavaScript，如下图所示：



然后进入到如下界面，点击右侧的参考书



然后进入到如下页面，此页面列举出了JavaScript中的所有对象



可以大体分页3大类：

第一类：**基本对象**，我们主要学习Array和JSON和String



第二类：**BOM对象**，主要是和浏览器相关的几个对象



第三类：**DOM对象**，JavaScript中将html的每一个标签都封装成一个对象

## HTML 对象

<a> 此处只截取部分  
<abbr>  
<address>  
<area>  
<article>  
<aside>

我们先来学习基本对象种的Array对象

### 1.5.1 基本对象

#### 1.5.1.1 Array对象

##### 语法格式

Array对象时用来定义数组的。常用语法格式有如下2种：

方式1：

```
1 var 变量名 = new Array(元素列表);
```

例如：

```
1 var arr = new Array(1,2,3,4); //1,2,3,4 是存储在数组中的数据（元素）
```

方式2：

```
1 var 变量名 = [ 元素列表 ];
```

例如：

```
1 var arr = [1,2,3,4]; //1,2,3,4 是存储在数组中的数据（元素）
```

数组定义好了，那么我们该如何获取数组中的值呢？和java中一样，需要通过索引来获取数组中的值。

语法如下：

```
1 arr[索引] = 值;
```

接下来，我们在VS Code中创建名为02. JS-对象-Array.html的文件，按照上述的语法定义数组，并且通过索引来获取数组中的值。

```

1  <script>
2      //定义数组
3      var arr = new Array(1,2,3,4);
4      var arr = [1,2,3,4];
5      //获取数组中的值，索引从0开始计数
6      console.log(arr[0]);
7      console.log(arr[1]);
8  </script>

```

浏览器控制台观察的效果如下：输出1和2



## 特点

与java中不一样的是，JavaScript中数组相当于java中的集合，数组的长度是可以变化的。而且JavaScript是弱数据类型的语言，所以数组中可以存储任意数据类型的值。接下来我们通过代码来演示上述特点。

注释掉之前的代码，添加如下代码：

```

1  //特点：长度可变 类型可变
2  var arr = [1,2,3,4];
3  arr[10] = 50;
4
5  console.log(arr[10]);
6  console.log(arr[9]);
7  console.log(arr[8]);

```

上述代码定义的arr变量中，数组的长度是4，但是我们直接再索引10的位置直接添加了数据10，并且输出索引为10的位置的元素，浏览器控制台数据结果如下：



因为索引8和9的位置没有值，所以输出内容undefined，当然，我们也可以给数组添加其他类型的值，添加代码如下：注释掉之前控制台输出的代码

```
1 //特点：长度可变 类型可变
2 var arr = [1,2,3,4];
3 arr[10] = 50;
4
5 // console.log(arr[10]);
6 // console.log(arr[9]);
7 // console.log(arr[8]);
8
9 arr[9] = "A";
10 arr[8] = true;
11
12 console.log(arr);
```

浏览器控制台输出结果如下：



## 属性和方法

Array作为一个对象，那么对象是有属性和方法的，所以接下来我们介绍一下Array对象的属性和方法

官方文档中提供了Array的很多属性和方法，但是我们只学习常用的属性和方法，如下图所示：

属性：

属性	描述
length	设置或返回数组中元素的数量。

方法：

方法方法	描述
forEach()	遍历数组中的每个有值得元素，并调用一次传入的函数
push()	将新元素添加到数组的末尾，并返回新的长度
splice()	从数组中删除元素

- length属性：

length属性可以用来获取数组的长度，所以我们可以借助这个属性，来遍历数组中的元素，添加如下代码：

```
1  var arr = [1,2,3,4];
2  arr[10] = 50;
3      for (let i = 0; i < arr.length; i++) {
4      console.log(arr[i]);
5  }
```

浏览器控制台输出结果如图所示：



- forEach() 函数

首先我们学习forEach()方法，顾名思义，这是用来遍历的，那么遍历做什么事呢？所以这个方法的参数，需要传递一个函数，而且这个函数接受一个参数，就是遍历时的数组的值。修改之前的遍历代码如下：

```
1  //e是形参，接受的是数组遍历时的值
2  arr.forEach(function(e) {
3      console.log(e);
4  })
```

当然了，在ES6中，引入箭头函数的写法，语法类似java中lambda表达式，修改上述代码如下：

```
1  arr.forEach((e) => {
2      console.log(e);
3  })
```

浏览器输出结果如下：注意的是，没有元素的内容是不会输出的，因为forEach只会遍历有值的元素



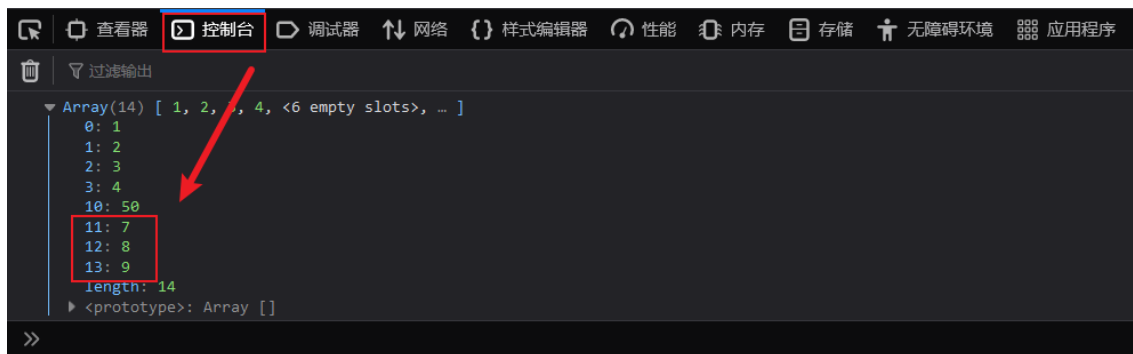


- `push()` 函数

`push()` 函数是用于向数组的末尾添加元素的，其中函数的参数就是需要添加的元素，编写如下代码：**向数组的末尾添加3个元素**

```
1 //push: 添加元素到数组末尾
2 arr.push(7,8,9);
3 console.log(arr);
```

浏览器输出结果如下：



- `splice()` 函数

`splice()` 函数用来数组中的元素，函数中填入2个参数。

**参数1：表示从哪个索引位置删除**

**参数2：表示删除元素的个数**

如下代码表示：从索引2的位置开始删，删除2个元素

```
1 //splice: 删除元素
2 arr.splice(2,2);
3 console.log(arr);
```

浏览器执行效果如下：元素3和4被删除了，元素3是索引2



Array数组的完整代码如下：

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
scale=1.0">
7      <title>JS-对象-Array</title>
8  </head>
9  <body>
10
11 </body>
12 <script>
13     //定义数组
14     // var arr = new Array(1,2,3,4);
15     // var arr = [1,2,3,4];
16
17     // console.log(arr[0]);
18     // console.log(arr[1]);
19
20     //特点：长度可变 类型可变
21     // var arr = [1,2,3,4];
22     // arr[10] = 50;
23
24     // console.log(arr[10]);
25     // console.log(arr[9]);
26     // console.log(arr[8]);
27
28     // arr[9] = "A";
29     // arr[8] = true;
30
31     // console.log(arr);
32
33
34
```

```

35     var arr = [1,2,3,4];
36     arr[10] = 50;
37     // for (let i = 0; i < arr.length; i++) {
38     //     console.log(arr[i]);
39     // }
40
41     // console.log("=====");
42
43     //forEach: 遍历数组中有值的元素
44     // arr.forEach(function(e) {
45     //     console.log(e);
46     // })
47
48     // //ES6 箭头函数: (...) => {...} -- 简化函数定义
49     // arr.forEach((e) => {
50     //     console.log(e);
51     // })
52
53     //push: 添加元素到数组末尾
54     // arr.push(7,8,9);
55     // console.log(arr);
56
57     //splice: 删除元素
58     arr.splice(2,2);
59     console.log(arr);
60
61 </script>
62 </html>

```

### 1.5.1.2 String对象

#### 语法格式

String对象的创建方式有2种:

方式1:

```
1  var 变量名 = new String("...") ; //方式一
```

例如:

```
1  var str = new String("Hello String");
```

方式2:

```
1  var 变量名 = "..."; //方式二
```

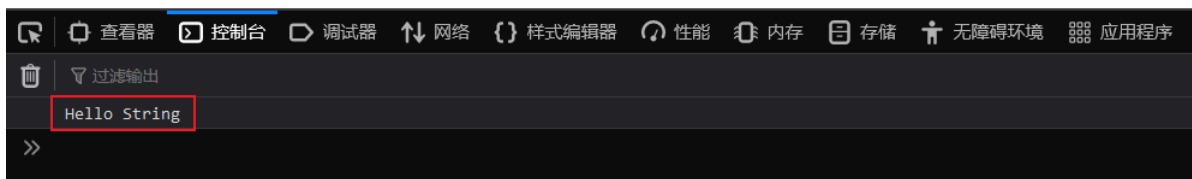
例如:

```
1  var str = 'Hello String';
```

按照上述的格式, 在VS Code中创建为名03. JS-对象-String.html的文件, 编写代码如下:

```
1  <script>
2      //创建字符串对象
3      //var str = new String("Hello String");
4      var str = "  Hello String  ";
5
6      console.log(str);
7  </script>
```

浏览器控制台输出结果如下:



## 属性和方法

String对象也提供了一些常用的属性和方法, 如下表格所示:

属性:

属性	描述
length	字符串的长度。

方法:

方法	描述
<code>charAt()</code>	返回在指定位置的字符。
<code>indexOf()</code>	检索字符串。
<code>trim()</code>	去除字符串两边的空格
<code>substring()</code>	提取字符串中两个指定的索引号之间的字符。

- `length`属性:

`length`属性可以用于返回字符串的长度，添加如下代码：

```
1 //length
2 console.log(str.length);
```

- `charAt()` 函数:

`charAt()` 函数用于返回在指定索引位置的字符，函数的参数就是索引。添加如下代码：

```
1 console.log(str.charAt(4));
```

- `indexOf()` 函数

`indexOf()` 函数用于检索指定内容在字符串中的索引位置的，返回值是索引，参数是指定的内容。添加如下代码：

```
1 console.log(str.indexOf("lo"));
```

- `trim()` 函数

`trim()` 函数用于去除字符串两边的空格的。添加如下代码：

```
1 var s = str.trim();
2 console.log(s.length);
```

- `substring()` 函数

`substring()` 函数用于截取字符串的，函数有2个参数。

参数1：表示从那个索引位置开始截取。包含

参数2：表示到那个索引位置结束。不包含

```
1 console.log(s.substring(0,5));
```

整体代码如下：

```
1 <!DOCTYPE html>
```

```
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>JS-对象-String</title>
8  </head>
9  <body>
10
11 </body>
12 <script>
13     //创建字符串对象
14     //var str = new String("Hello String");
15     var str = "  Hello String  ";
16
17     console.log(str);
18
19     //length
20     console.log(str.length);
21
22     //charAt
23     console.log(str.charAt(4));
24
25     //indexOf
26     console.log(str.indexOf("lo"));
27
28     //trim
29     var s = str.trim();
30     console.log(s.length);
31
32     //substring(start,end) --- 开始索引, 结束索引 (含头不含尾)
33     console.log(s.substring(0,5));
34
35 </script>
36 </html>
```

浏览器执行效果如图所示:



### 1.5.1.3 JSON对象

#### 自定义对象

在 JavaScript 中自定义对象特别简单，其语法格式如下：

```
1  var 对象名 = {  
2      属性名1: 属性值1,  
3      属性名2: 属性值2,  
4      属性名3: 属性值3,  
5      函数名称: function(形参列表) {}  
6  };  
7
```

我们可以通过如下语法调用属性：

```
1  对象名.属性名
```

通过如下语法调用函数：

```
1  对象名.函数名()
```

接下来，我们再VS Code中创建名为04. JS-对象-JSON.html的文件演示自定义对象

```
1  <script>  
2      //自定义对象  
3      var user = {  
4          name: "Tom",  
5          age: 10,  
6          gender: "male",  
7          eat: function() {  
8              console.log("用膳~");  
9          }  
10     }  
11  
12     console.log(user.name);  
13     user.eat();
```

```
14 <script>
```

浏览器控制台结果如下：



其中上述函数定义的语法可以简化成如下格式：

```
1     var user = {
2         name: "Tom",
3         age: 10,
4         gender: "male",
5         // eat: function(){
6         //     console.log("用膳~");
7         // }
8         eat() {
9             console.log("用膳~");
10        }
11    }
```

## json对象

**JSON对象**：JavaScript Object Notation, JavaScript对象标记法。是通过JavaScript标记法书写的文本。其格式如下：

```
1  {
2      "key":value,
3      "key":value,
4      "key":value
5  }
```

其中，**key必须使用引号并且是双引号标记**，**value可以是任意数据类型**。

例如我们可以直接百度搜索“json在线解析”，随便挑一个进入，然后编写内容如下：

```
1  {
2      "name": "李传播"
3  }
```



JSON解析JSON在线解析JSON压缩转义JSON视图JSON着色JSON/XML互转JSON生成实体JSON对比JSON压缩成一个

1 {  
2   " name": "李传播"  
3 }

1.编写内容，key需要加双引号

② 这里可以收缩编辑框

2.点击校验格式

① 鼠标左键按下拖动，调整大小 >> << 下次打开本工具，大小是记忆的

校验 / 格式化

压缩

转义

去转义

Unicode转中文

中文转Unicode

复制结果

清空

保存本地

老铁，这个JSON格式，没毛病。

3.校验结果

但是当我们将双引号切换成单引号，再次校验，则报错，如下图所示：

JSON解析JSON在线解析JSON压缩转义JSON视图JSON着色JSON/XML互转JSON生成实体JSON对比JSON压缩成一个

1 {  
2   ' name': "李传播"  
3 }

1.换成单引号

② 这里可以收缩编辑框

① 鼠标左键按下拖动，调整大小 >> << 下次打开本工具，大小是记忆的

校验 / 格式化

压缩

转义

去转义

Unicode转中文

中文转Unicode

复制结果

清空

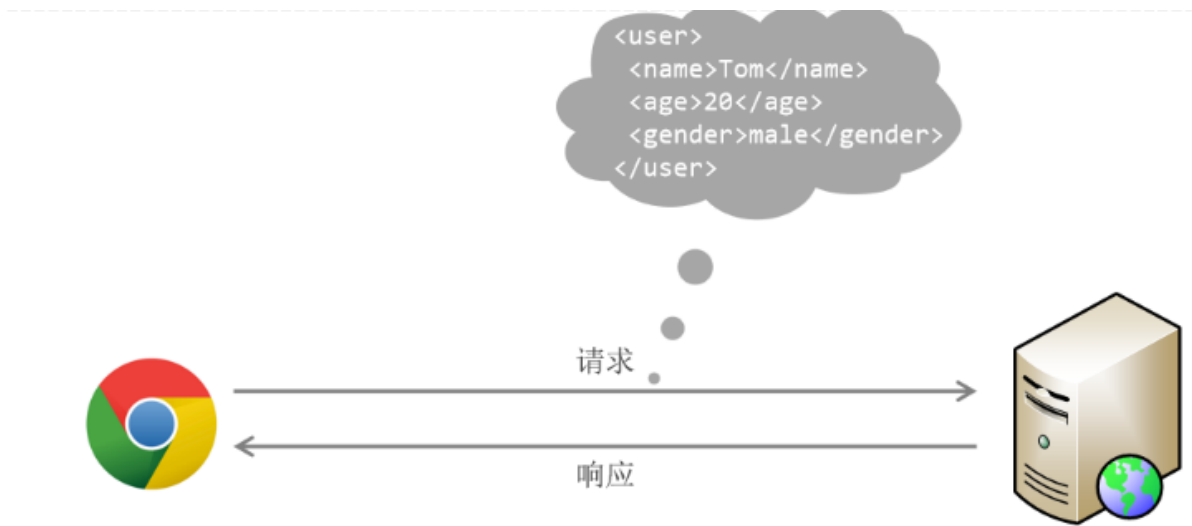
保存本地

第 1 行解析错误：  
{ 'name': "李传播"}  
--> (标准的 JSON 格式 请使用英文双引号 '"')

2.校验格式失败

那么json这种数据格式的文本到底应用在企业开发的什么地方呢？ -- 经常用来作为前后台交互的数据载体

如下图所示：前后台交互时，我们需要传输数据，但是java中的对象我们该怎么去描述呢？我们可以使用如图所示的xml格式，可以清晰的描述java中需要传递给前端的java对象。



但是xml格式存在如下问题:

- 标签需要编写双份, 占用带宽, 浪费资源
- 解析繁琐

所以我们可以使用json来替代, 如下图所示:



接下来我们通过代码来演示json对象: 注释掉之前的代码, 编写代码如下:

```
1  var jsonstr = '{"name": "Tom", "age": 18, "addr": ["北京", "上海", "西安"]}';
2  alert(jsonstr.name);
```

浏览器输出结果如下:



为什么呢? 因为上述是一个json字符串, 不是json对象, 所以我们需要借助如下函数来进行json字符串和json对象的转换。添加代码如下:

```
1 var obj = JSON.parse(jsonstr);
2 alert(obj.name);
```

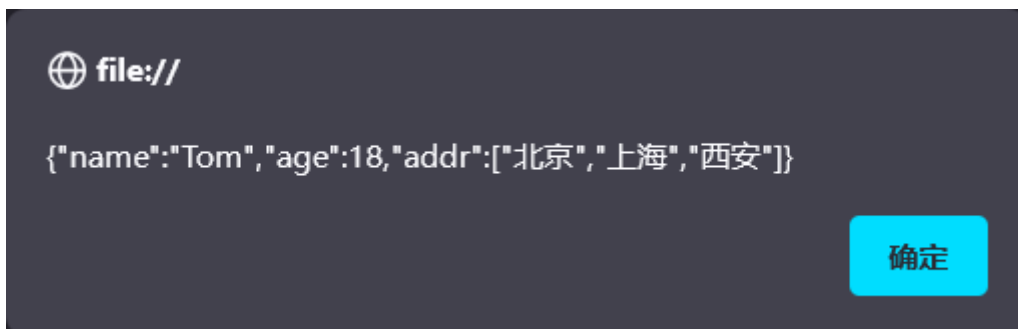
此时浏览器输出结果如下：



当然了，我们也可以通过如下函数将json对象再次转换成json字符串。添加如下代码：

```
1 alert(JSON.stringify(obj));
```


浏览器输出结果如图所示：



整体全部代码如下：

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7     <title>JS-对象-JSON</title>
8 </head>
9 <body>
10
11 </body>
12 <script>
13     //自定义对象
14     // var user = {
15     //     name: "Tom",
16     //     age: 10,
```

```

17      //      gender: "male",
18      //      // eat: function() {
19      //          //      console.log("用膳~");
20      //          //      }
21      //      eat() {
22      //          console.log("用膳~");
23      //      }
24      //  }
25
26      // console.log(user.name);
27      // user.eat();
28
29
30      // //定义json
31      var jsonstr = '{"name":"Tom", "age":18, "addr":["北京","上海","西
    安"]}';
32      //alert(jsonstr.name); 
33
34      // //json字符串--js对象
35      var obj = JSON.parse(jsonstr);
36      //alert(obj.name);
37
38      // //js对象--json字符串
39      alert(JSON.stringify(obj));
40
41
42  </script>
43  </html>

```

### 1.5.2 BOM对象

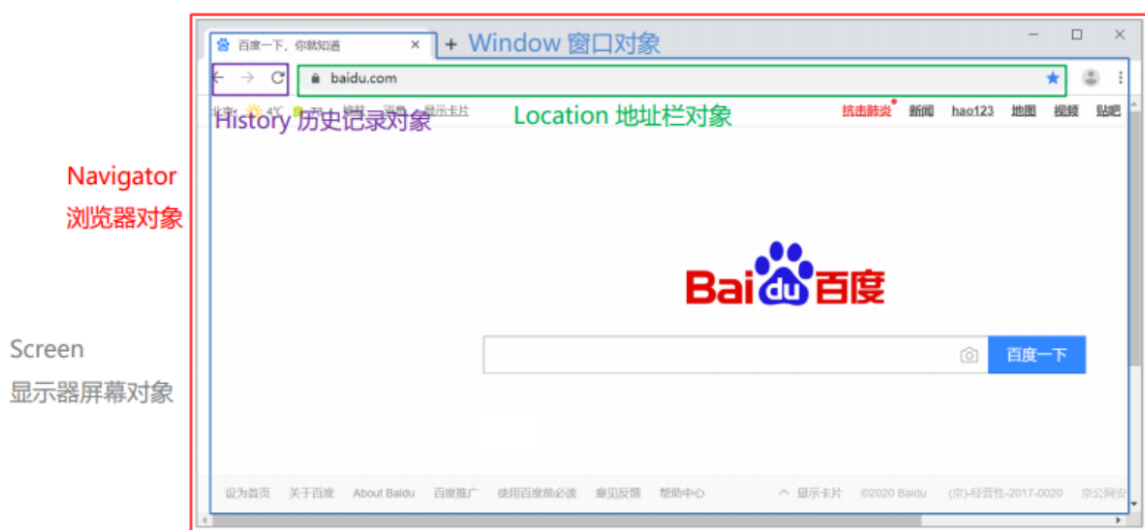
接下来我们学习BOM对象，BOM的全称是Browser Object Model，翻译过来是浏览器对象模型。也就是JavaScript将浏览器的各个组成部分封装成了对象。我们要操作浏览器的部分功能，可以通过操作BOM对象的相关属性或者函数来完成。例如：我们想要将浏览器的地址改为 <http://www.baidu.com>，我们就可以通过BOM中提供的location对象的href属性来完成，代码如下：

```
location.href='http://www.baidu.com'
```

BOM中提供了如下5个对象：

对象名称	描述
Window	浏览器窗口对象
Navigator	浏览器对象
Screen	屏幕对象
History	历史记录对象
Location	d地址栏对象

上述5个对象与浏览器各组成对应的关系如下图所示：



对于上述5个对象，我们重点学习的是Window对象、Location对象这两个。

### 1.5.2.1 Window对象

window对象指的是浏览器窗口对象，是JavaScript的全部对象，所以对于window对象，我们可以直接使用，并且对于window对象的方法和属性，我们可以省略window.例如：我们之前学习的alert()函数其实是属于window对象的，其完整的代码如下：

```
1 window.alert('hello');
```

其可以省略window. 所以可以简写成

```
1 alert('hello')
```

所以对于window对象的属性和方法，我们都是采用简写的方式。window提供了很多属性和方法，下表列出了常用属性和方法

window对象提供了获取其他BOM对象的属性：

属性	描述
history	用于获取history对象
location	用于获取location对象
Navigator	用于获取Navigator对象
Screen	用于获取Screen对象

也就是说我们要使用location对象，只需要通过代码 `window.location` 或者简写 `location` 即可使用

window也提供了一些常用的函数，如下表格所示：

函数	描述
alert()	显示带有一段消息和一个确认按钮的警告框。
confirm()	显示带有一段消息以及确认按钮和取消按钮的对话框。
setInterval()	按照指定的周期（以毫秒计）来调用函数或计算表达式。
setTimeout()	在指定的毫秒数后调用函数或计算表达式。

接下来，我们通过VS Code中创建名为05. JS-对象-BOM.html文件来编写代码来演示上述函数：

- alert() 函数：弹出警告框，函数的内容就是警告框的内容

```
1  <script>
2      //window对象是全局对象，window对象的属性和方法在调用时可以省略
   window.
3      window.alert("Hello BOM");
4      alert("Hello BOM Window");
5  </script>
```

浏览器打开，依次弹框，此处只截图一张

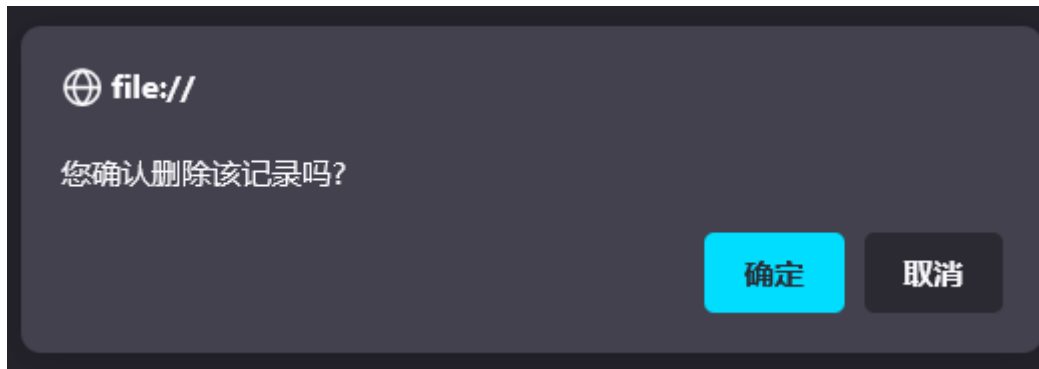


- `confirm()` 函数：弹出确认框，并且提供用户2个按钮，分别是确认和取消。

添加如下代码：

```
1 confirm("您确认删除该记录吗?");
```

浏览器打开效果如图所示：



但是我们怎么知道用户点击了确认还是取消呢？所以这个函数有一个返回值，当用户点击确认时，返回`true`，点击取消时，返回`false`。我们根据返回值来决定是否执行后续操作。修改代码如下：再次运行，可以查看返回值`true`或者`false`

```
1 var flag = confirm("您确认删除该记录吗?");
2 alert(flag);
```

- `setInterval(fn, 毫秒值)`：定时器，用于周期性的执行某个功能，并且是**循环执行**。该函数需要传递2个参数：

**fn**:函数，需要周期性执行的功能代码

**毫秒值**：间隔时间

注释掉之前的代码，添加代码如下：

```
1 //定时器 - setInterval -- 周期性的执行某一个函数
2 var i = 0;
3 setInterval(function() {
4     i++;
5     console.log("定时器执行了"+i+"次");
6 }, 2000);
```

刷新页面，浏览器每个一段时间都会在控制台输出，结果如下：



- `setTimeout(fn, 毫秒值)`：定时器，只会在一段时间后**执行一次功能**。参数和上述`setInterval`一致

注释掉之前的代码，添加代码如下：

```
1 //定时器 - setTimeout -- 延迟指定时间执行一次
2 setTimeout(function(){
3     alert("JS");
4 },3000);
```

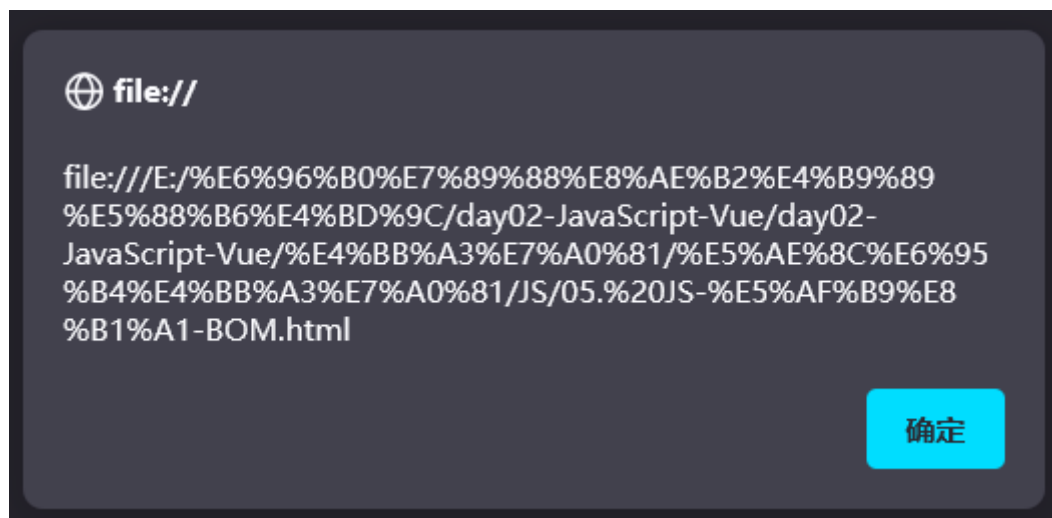
浏览器打开，3s后弹框，关闭弹框，发现再也不会弹框了。

### 1.5.2.2 Location对象

location是指代浏览器的地址栏对象，对于这个对象，我们常用的是href属性，用于获取或者设置浏览器的地址信息，添加如下代码：

```
1 //获取浏览器地址栏信息
2 alert(location.href);
3 //设置浏览器地址栏信息
4 location.href = "https://www.itcast.cn";
```

浏览器效果如下：首先弹框展示浏览器地址栏信息，



然后点击确定后，因为我们设置了地址栏信息，所以浏览器跳转到传智首页

完整代码如下：

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-
7     scale=1.0">
8     <title>JS-对象-BOM</title>
```



```
8   </head>
9   <body>
10
11  </body>
12  <script>
13      //获取
14      // window.alert("Hello BOM");
15      // alert("Hello BOM Window");
16
17      //方法
18      //confirm - 对话框 -- 确认: true , 取消: false
19      // var flag = confirm("您确认删除该记录吗?");
20      // alert(flag);
21
22      //定时器 - setInterval -- 周期性的执行某一个函数
23      // var i = 0;
24      // setInterval(function(){
25      //     i++;
26      //     console.log("定时器执行了"+i+"次");
27      // },2000);
28
29      //定时器 - setTimeout -- 延迟指定时间执行一次
30      // setTimeout(function(){
31      //     alert("JS");
32      // },3000);
33
34      //location
35      alert(location.href);
36
37      location.href = "https://www.itcast.cn";
38
39  </script>
40 </html>
```

### 1.5.3 DOM对象

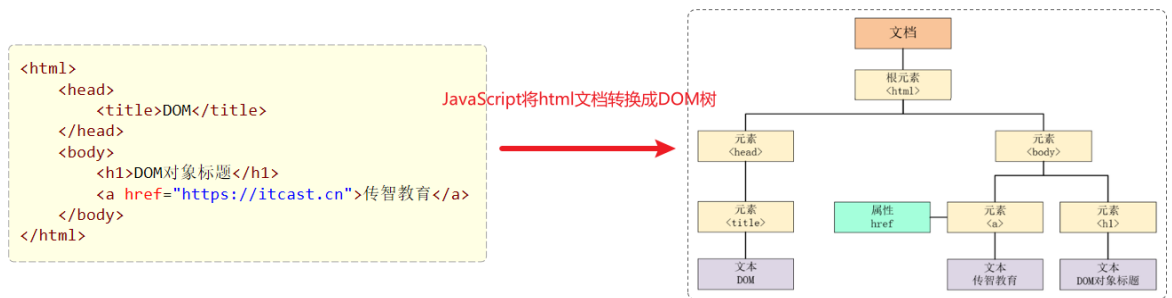
#### 1.5.3.1 DOM介绍

DOM: Document Object Model 文档对象模型。也就是 JavaScript 将 HTML 文档的各个组成部分封装为对象。

DOM 其实我们并不陌生，之前在学习 XML 就接触过，只不过 XML 文档中的标签需要我们写代码解析，而 HTML 文档是浏览器解析。封装的对象分为

- Document：整个文档对象
- Element：元素对象
- Attribute：属性对象
- Text：文本对象
- Comment：注释对象

如下图，左边是 HTML 文档内容，右边是 DOM 树



那么我们学习DOM技术有什么用呢？主要作用如下：

- 改变 HTML 元素的内容
- 改变 HTML 元素的样式 (CSS)
- 对 HTML DOM 事件作出反应
- 添加和删除 HTML 元素

从而达到动态改变页面效果目的，具体我们可以查看代码中提供的06. JS-对象-DOM-演示.html来体会DOM的效果。

### 1.5.3.2 获取DOM对象

我们知道DOM的作用是通过修改HTML元素的内容和样式等来实现页面的各种动态效果，但是我们要操作DOM对象的前提是先获取元素对象，然后才能操作。所以学习DOM, 主要的核心就是学习如下2点：

- 如何获取DOM中的元素对象 (Element对象，也就是标签)
- 如何操作Element对象的属性, 也就是标签的属性。

接下来我们先来学习如何获取DOM中的元素对象。

HTML中的Element对象可以通过Document对象获取，而Document对象是通过window对象获取的。

document对象提供的用于获取Element元素对象的api如下表所示：

函数	描述
<code>document.getElementById()</code>	根据id属性值获取，返回单个Element对象
<code>document.getElementsByTagName()</code>	根据标签名称获取，返回Element对象数组
<code>document.getElementsByName()</code>	根据name属性值获取，返回Element对象数组
<code>document.getElementsByClassName()</code>	根据class属性值获取，返回Element对象数组

接下来我们通过VS Code中创建名为07. JS-对象-DOM-获取元素.html的文件来演示上述api，首先在准备如下页面代码：

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>JS-对象-DOM</title>
8  </head>
9
10 <body>
11      <br><br>
12
13     <div class="cls">传智教育</div> <br>
14     <div class="cls">黑马程序员</div> <br>
15
16     <input type="checkbox" name="hobby"> 电影
17     <input type="checkbox" name="hobby"> 旅游
18     <input type="checkbox" name="hobby"> 游戏
19 </body>
20
21 </html>
```

- `document.getElementById()`：根据标签的id属性获取标签对象，id是唯一的，所以获取到是单个标签对象。

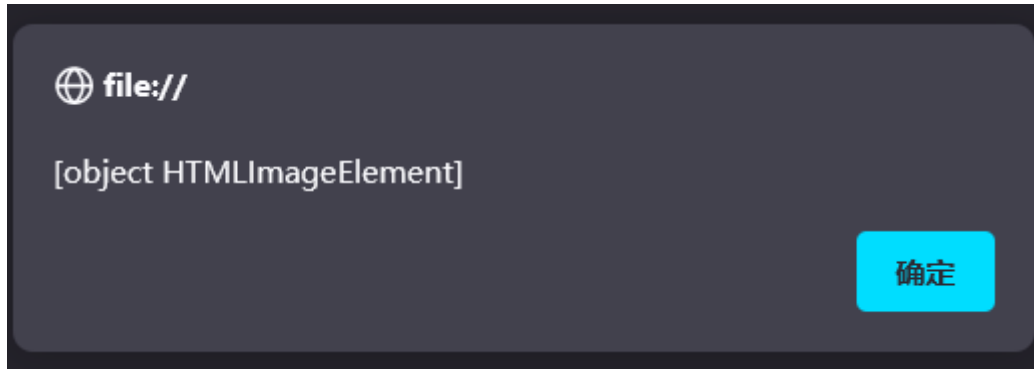
添加如下代码：

```

1  <script>
2  //1. 获取Element元素
3
4  //1.1 获取元素-根据ID获取
5  var img = document.getElementById('h1');
6  alert(img);
7  </script>

```

浏览器打开，效果如图所示：从弹出的结果能够看出，这是一个图片标签对象



- `document.getElementsByTagName()` : 根据标签的名字获取标签对象，同名的标签有很多，所以返回值是数组。

添加如下代码：

```

1  //1.2 获取元素-根据标签获取 - div
2  var divs = document.getElementsByTagName('div');
3  for (let i = 0; i < divs.length; i++) {
4      alert(divs[i]);
5  }

```

浏览器输出2次如下所示的弹框

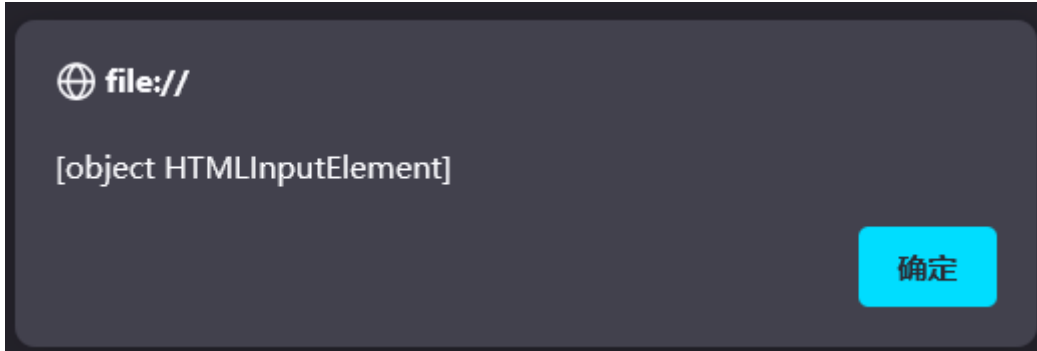


- `document.getElementsByName()` : 根据标签的name的属性值获取标签对象，name属性值可以重复，所以返回值是一个数组。

添加如下代码：

```
1 //1.3 获取元素-根据name属性获取
2 var ins = document.getElementsByName('hobby');
3 for (let i = 0; i < ins.length; i++) {
4     alert(ins[i]);
5 }
```

浏览器会有3次如下图所示的弹框：



- `document.getElementsByClassName()` : 根据标签的class属性值获取标签对象，class属性值也可以重复，返回值是数组。

添加如下图所示的代码：

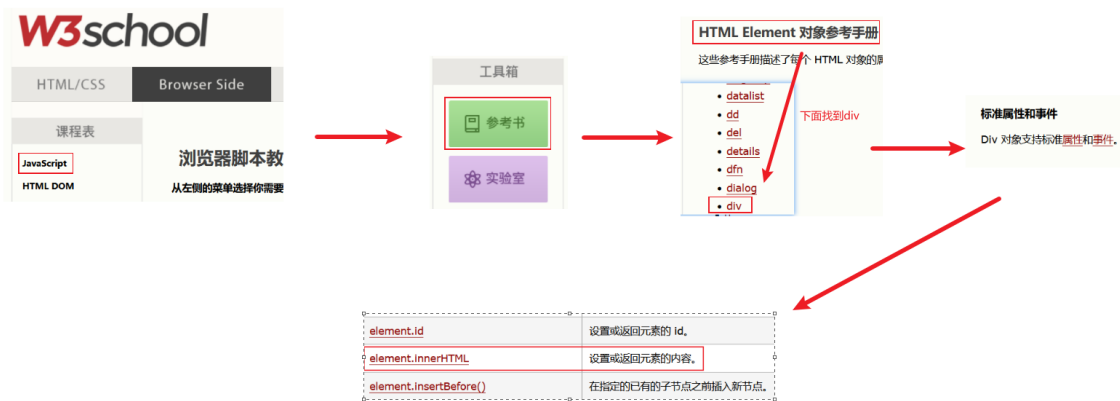
```
1 //1.4 获取元素-根据class属性获取
2 var divs = document.getElementsByClassName('cls');
3 for (let i = 0; i < divs.length; i++) {
4     alert(divs[i]);
5 }
```

浏览器会弹框2次，都是div标签对象



- 操作属性

那么获取到标签了，我们如何操作标签的属性呢？通过查询文档资料，如下图所示：



得出我们可以通过div标签对象的innerHTML属性来修改标签的内容。此时我们想把页面中的**传智教育**替换成**传智教育666**，所以要获取2个div中的第一个，所以可以通过下标0获取数组中的第一个div，注释之前的代码，编写如下代码：

```
1  var divs = document.getElementsByClassName('cls');
2  var div1 = divs[0];
3
4  div1.innerHTML = "传智教育666";
```

浏览器刷新页面，展示效果如下图所示：

**传智教育666**

**黑马程序员**

发现页面内容变成了传智教育666

完整代码如下：

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
7          scale=1.0">
8      <title>JS-对象-DOM</title>
9  </head>
10 <body>
11      <br><br>
12
13     <div class="cls">传智教育</div> <br>
14     <div class="cls">黑马程序员</div> <br>
15
```

```
16     <input type="checkbox" name="hobby"> 电影
17     <input type="checkbox" name="hobby"> 旅游
18     <input type="checkbox" name="hobby"> 游戏
19 </body>
20
21 <script>
22     //1. 获取Element元素
23
24     //1.1 获取元素-根据ID获取
25     // var img = document.getElementById('h1');
26     // alert(img);
27
28     //1.2 获取元素-根据标签获取 - div
29     // var divs = document.getElementsByTagName('div');
30     // for (let i = 0; i < divs.length; i++) {
31     //     alert(divs[i]);
32     // }
33
34
35     //1.3 获取元素-根据name属性获取
36     // var ins = document.getElementsByName('hobby');
37     // for (let i = 0; i < ins.length; i++) {
38     //     alert(ins[i]);
39     // }
40
41
42     //1.4 获取元素-根据class属性获取
43     // var divs = document.getElementsByClassName('cls');
44     // for (let i = 0; i < divs.length; i++) {
45     //     alert(divs[i]);
46     // }
47
48
49
50     //2. 查询参考手册，属性、方法
51     var divs = document.getElementsByClassName('cls');
52     var div1 = divs[0];
53
54     div1.innerHTML = "传智教育666";
55
56 </script>
57 </html>
```

## 1.5.4 案例

### 1.5.4.1 需求说明

鲁迅说的好，光说不练假把式，光练不说傻把式。所以接下来我们需要通过案例来加强对于上述DOM知识的掌握。需求如下3个：

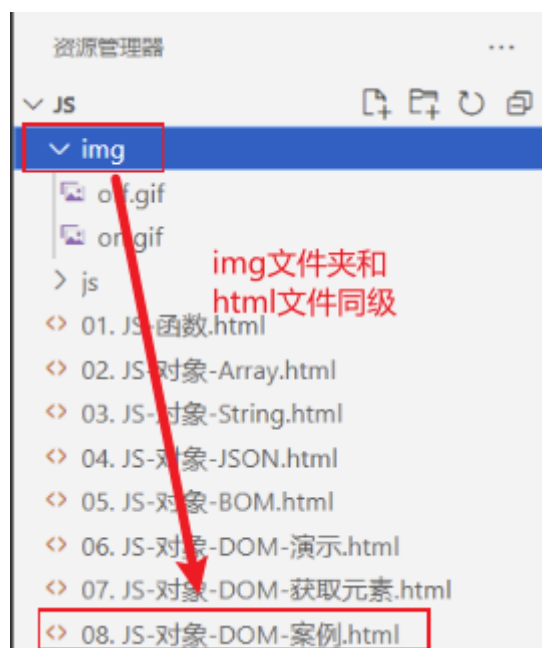
- 点亮灯泡
- 将所有的div标签的标签体内容后面加上：very good
- 使所有的复选框呈现被选中的状态

效果如下所示：



### 1.5.4.2 资料准备

在JS目录下，也就是用于存放html文件的同级创建img文件夹，然后将 [资料/图片素材](#) 中提供的2张图片拷贝到img文件夹中，最终整体结果如下图所示：





在VS Code中创建名为08. JS-对象-DOM-案例.html的文件，然后准备如下代码：

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>JS-对象-DOM-案例</title>
8  </head>
9
10 <body>
11      <br><br>
12
13     <div class="cls">传智教育</div> <br>
14     <div class="cls">黑马程序员</div> <br>
15
16     <input type="checkbox" name="hobby"> 电影
17     <input type="checkbox" name="hobby"> 旅游
18     <input type="checkbox" name="hobby"> 游戏
19 </body>
20
21 <script>
22
23 </script>
24 </html>
```

浏览器打开此时效果如图所示：



传智教育

黑马程序员

☐ 电影 ☐ 旅游 ☐ 游戏

#### 1.5.4.3 需求1

- 需求

点亮灯泡

- 分析

此时我们需要把灯泡点亮，其实就是换一张图片。那么我们需要切换图片，就需要操作图片的src属性。要操作图片的src属性，就需要先来获取img标签对象。

- 步骤

- 首先获取img标签对象
- 然后修改img标签对象的src属性值，进行图片的切换

- 代码实现

```
1 //1. 点亮灯泡 : src 属性值
2 //首先获取img标签对象
3 var img = document.getElementById('h1');
4 //然后修改img标签对象的src属性值，进行图片的切换
5 img.src = "img/on.gif";
```

浏览器打开，效果如图所示：



传智教育

黑马程序员

☐ 电影 ☐ 旅游 ☐ 游戏

#### 1.5.4.4 需求2

- 需求

将所有的div标签的标签体内容后面加上：very good

并且very good是红色字体

- 分析

我们需要在原有内容后面追加红色的very good.所以我们首先需要获取原有内容,然后再进行内容的追加。但是如何保证very good是红色的呢?所以我们可以通过之前html中学过的<font>标签和属性来完整。如何进行内容的替换呢?之前我们学习过innerHTML属性。需要替换2个div的内容,所以我们需要获取2个div,并且遍历进行替换。

- 步骤
  - 通过标签的名字div获取所有的div标签
  - 遍历所有的div标签
  - 获取div标签的原有内容,然后追加<font color='red'>very good</font>,并且替原内容
- 代码实现

```
1 //2. 将所有div标签的内容后面加上: very good (红色字体) -- <font
   color='red'></font>
2 var divs = document.getElementsByTagName('div');
3 for (let i = 0; i < divs.length; i++) {
4     const div = divs[i];
5     div.innerHTML += "<font color='red'>very good</font>";
6 }
```

浏览器打开效果如图所示:



传智教育very good

黑马程序员very good

☐ 电影 ☐ 旅游 ☐ 游戏

#### 1.5.4.5 需求3

- 需求

使所有的复选框呈现被选中的状态
- 分析

要让复选框处于选中状态，那么什么属性或者方法可以使复选框选中？可以查询资料得出checkbox标签对象的checked属性设置为true，可以改变checkbox为选中状态。那么需要设置所有的checkbox，那么我们需要获取所有的checkbox并且遍历

- 步骤
  - 可以通过name属性值获取所有的checkbox标签
  - 遍历所有的checkbox标签，
  - 设置每个checkbox标签的
- 代码实现

```
1  // //3. 使所有的复选框呈现选中状态
2  var ins = document.getElementsByName('hobby');
3  for (let i = 0; i < ins.length; i++) {
4    const check = ins[i];
5    check.checked = true; //选中
6  }
```

浏览器刷新，效果如图所示：



传智教育very good

黑马程序员very good

☒ 电影 ☒ 旅游 ☒ 游戏

#### 1.5.4.6 完整代码

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
7      scale=1.0">
8      <title>JS-对象-DOM-案例</title>
```

```

8   </head>
9
10  <body>
11        <br><br>
12
13      <div class="cls">传智教育</div>  <br>
14      <div class="cls">黑马程序员</div>  <br>
15
16      <input type="checkbox" name="hobby"> 电影
17      <input type="checkbox" name="hobby"> 旅游
18      <input type="checkbox" name="hobby"> 游戏
19  </body>
20
21  <script>
22      //1. 点亮灯泡 : src 属性值
23      var img = document.getElementById('h1');
24      img.src = "img/on.gif";
25
26
27      //2. 将所有div标签的内容后面加上: very good (红色字体) -- <font
color='red'></font>
28      var divs = document.getElementsByTagName('div');
29      for (let i = 0; i < divs.length; i++) {
30          const div = divs[i];
31          div.innerHTML += "<font color='red'>very good</font>";
32      }
33
34
35      // //3. 使所有的复选框呈现选中状态
36      var ins = document.getElementsByName('hobby');
37      for (let i = 0; i < ins.length; i++) {
38          const check = ins[i];
39          check.checked = true;//选中
40      }
41
42  </script>
43  </html>

```

## 1.6 JavaScript事件

### 1.6.1 事件介绍

如下图所示的百度注册页面，当我们用户输入完内容，百度可以自动的提示我们用户名已经存在还是可以使用。那么百度是怎么知道我们用户名输入完了呢？这就需要用到JavaScript中的事件了。



欢迎注册

已有帐号? [登录](#)

用户名

此用户名太受欢迎,请更换一个

手机号

密 码

验证码

☐ 阅读并接受《[百度用户协议](#)》、《[儿童个人信息保护声明](#)》及《[百度隐私权保护声明](#)》

什么是事件呢？HTML事件是发生在HTML元素上的“事情”，例如：

- 按钮被点击
- 鼠标移到元素上
- 输入框失去焦点
- .....

而我们可以给这些事件绑定函数，当事件触发时，可以自动的完成对应的功能。这就是事件监听。例如：对于我们所说的百度注册页面，我们给用户名输入框的失去焦点事件绑定函数，当我们用户输入完内容，在标签外点击了鼠标，对于用户名输入框来说，失去焦点，然后执行绑定的函数，函数进行用户名内容的校验等操作。JavaScript事件是js非常重要的一部分，接下来我们进行事件的学习。那么我们对于JavaScript事件需要学习哪些内容呢？我们得知道有哪些常用事件，然后我们得学会如何给事件绑定函数。所以主要围绕2点来学习：

- 事件绑定
- 常用事件

## 1.6.2 事件绑定

JavaScript对于事件的绑定提供了2种方式：

- 方式1：通过html标签中的事件属性进行绑定

例如一个按钮，我们对于按钮可以绑定单击事件，可以借助标签的onclick属性，属性值指向一个函数。

在VS Code中创建名为09. JS-事件-事件绑定.html，添加如下代码：

```
1 <input type="button" id="btn1" value="事件绑定1" onclick="on()">
```

很明显没有on函数，所以我们需要创建该函数，代码如下：

```
1 <script>
2     function on(){
3         alert("按钮1被点击了...");
4     }
5 </script>
```

浏览器打开，然后点击按钮，弹框如下：



- 方式2：通过DOM中Element元素的事件属性进行绑定

依据我们学习过得DOM的知识点，我们知道html中的标签被加载成element对象，所以我们也可以通过element对象的属性来操作标签的属性。此时我们再次添加一个按钮，代码如下：

```
1 <input type="button" id="btn2" value="事件绑定2">
```

我们可以先通过id属性获取按钮对象，然后操作对象的onclick属性来绑定事件，代码如下：

```
1 document.getElementById('btn2').onclick = function(){
2     alert("按钮2被点击了...");
3 }
```

浏览器刷新页面，点击第二个按钮，弹框如下：



需要注意的是：事件绑定的函数，只有在事件被触发时，函数才会被调用。

整体代码如下：

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>JS-事件-事件绑定</title>
8  </head>
9
10 <body>
11     <input type="button" id="btn1" value="事件绑定1"
    onclick="on()">
12     <input type="button" id="btn2" value="事件绑定2">
13 </body>
14
15 <script>
16     function on(){
17         alert("按钮1被点击了...");
18     }
19
20     document.getElementById('btn2').onclick = function(){
21         alert("按钮2被点击了...");
22     }
23
24 </script>
25 </html>
```



### 1.6.3 常见事件

上面案例中使用到了 `onclick` 事件属性，那都有哪些事件属性供我们使用呢？下面就给大家列举一些比较常用的事件属性

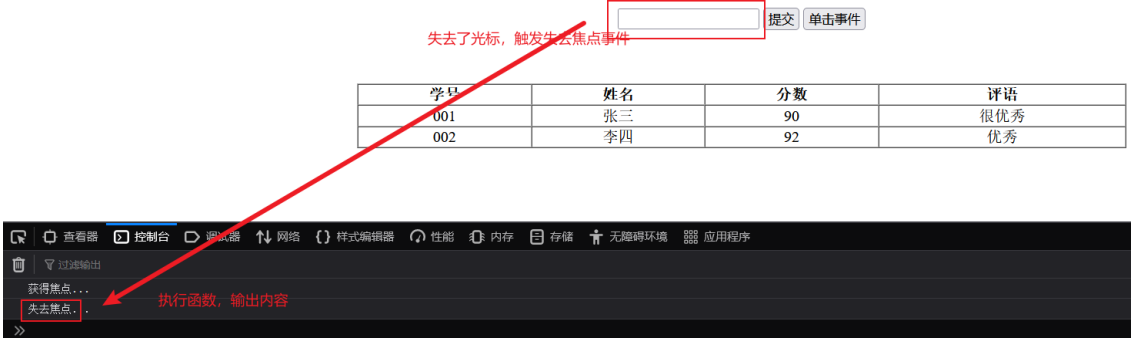
事件属性名	说明
onclick	鼠标单击事件
onblur	元素失去焦点
onfocus	元素获得焦点
onload	某个页面或图像被完成加载
onsubmit	当表单提交时触发该事件
onmouseover	鼠标被移到某元素之上
onmouseout	鼠标从某元素移开

在代码中提供了10. JS-事件-常见事件.html的文件，我们可以通过浏览器打开来观察几个常用事件的具体效果：

- onfocus:获取焦点事件，鼠标点击输入框，输入框中光标一闪一闪的，就是输入框获取焦点了



- onblur:失去焦点事件，前提是输入框获取焦点的状态下，在输入框之外的地方点击，光标从输入框中消失了，这就是失去焦点。



其他事件的效果，同学们可以通过提供好的代码去演示，亲身体会事件在什么时候触发。

## 1.6.4 案例

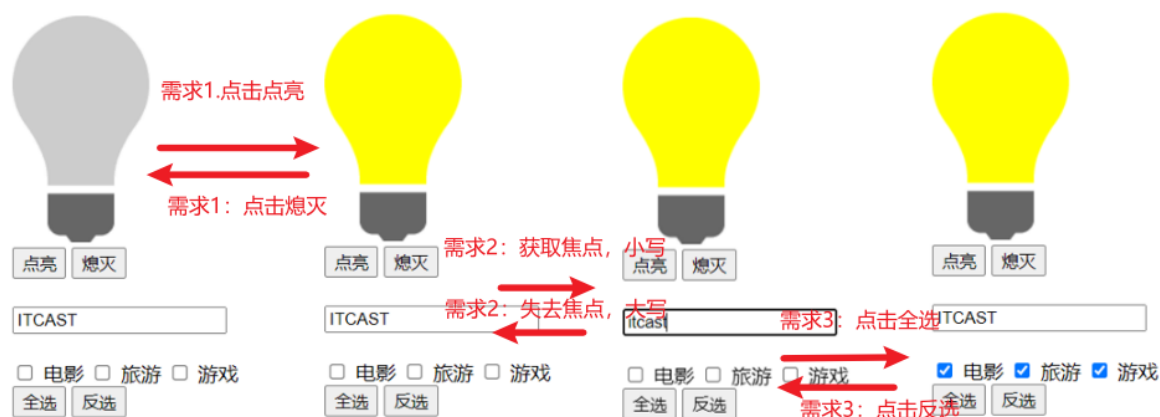
### 1.6.4.1 需求说明

接下来我们通过案例来加强所学js知识点的掌握。

需求如下3个：

1. 点击“点亮”按钮 点亮灯泡，点击“熄灭”按钮 熄灭灯泡
2. 输入框鼠标聚焦后，展示小写；鼠标离焦后，展示大写。
3. 点击“全选”按钮使所有的复选框呈现被选中的状态，点击“反选”按钮使所有的复选框呈现取消勾选的状态。

效果如图所示：



### 1.6.4.2 资料准备

在VS Code中创建名为11. JS-事件-案例.html的文件，提前准备如下代码：

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-
  scale=1.0">
7   <title>JS-事件-案例</title>
8 </head>
9 <body>
10
11    <br>
12
```

```

13     <input type="button" value="点亮" >
14     <input type="button" value="熄灭" >
15
16     <br> <br>
17
18     <input type="text" id="name" value="ITCAST" >
19     <br> <br>
20
21     <input type="checkbox" name="hobby"> 电影
22     <input type="checkbox" name="hobby"> 旅游
23     <input type="checkbox" name="hobby"> 游戏
24     <br>
25
26     <input type="button" value="全选" >
27     <input type="button" value="反选" >
28
29 </body>
30
31 </html>

```

浏览器打开如图所示：



#### 1.6.4.3 需求1

- 需求：

点击“点亮”按钮 点亮灯泡，点击“熄灭”按钮 熄灭灯泡

- 分析：

点击按钮的时候触发，所以我们需要绑定单击事件。不管是点亮还是熄灭，都是图片的变化，所以我们需要修改图片。但是修改图片我们还需要先获取标签图片标签对象。

- 步骤：

- 首先给点亮按钮和熄灭按钮都绑定单击事件。分别绑定函数on()和off()
- 然后在js中定义on()和off()函数
- on()函数中，通过id获取img标签对象，然后通过img标签对象的src属性切换点亮的图片
- off()函数中，通过id获取img标签对象，然后通过img标签对象的src属性切换熄灭的图片

- 代码实现：

#### 事件绑定

```
1 <input type="button" value="点亮" onclick="on()">
2 <input type="button" value="熄灭" onclick="off()">
```

#### on()和off()函数

```
1 //1. 点击 "点亮" 按钮，点亮灯泡；点击 "熄灭" 按钮，熄灭灯泡； --
  onclick
2 function on(){
3     //a. 获取img元素对象
4     var img = document.getElementById("light");
5     //b. 设置src属性
6     img.src = "img/on.gif";
7 }
8
9 function off(){
10    //a. 获取img元素对象
11    var img = document.getElementById("light");
12    //b. 设置src属性
13    img.src = "img/off.gif";
14 }
```

### 1.6.4.4 需求2

- 需求：

输入框鼠标聚焦后，展示小写；鼠标离焦后，展示大写。

- 分析：

聚焦和失焦的时候完成功能，所以我们需要给input标签绑定onfocus和onblur事件；我们要切换大小写，那么我们肯定要获取原本输入框的内容，通过查询资料，需要使用input标签对象的value属性，然后进行大小写切换；切换完成我们需要重新填入，所以还是通过value属性来设置input标签输入框的内容

- 步骤：
  - 给input标签的onfocus和onblur事件分别绑定lower()和upper()函数
  - 然后在js中定义lower()和upper()函数
  - 对于lower()函数，先通过id获取输入框对象，然后通过输入框的value属性来设置内容，内容的话可以通过字符串的toLowerCase()函数来进行小写转换
  - 对于upper()函数，先通过id获取输入框对象，然后通过输入框的value属性来设置内容，内容的话可以通过字符串的toUpperCase()函数来进行大写转换
- 代码实现：、

事件绑定：

```
1 <input type="text" id="name" value="ITCAST" onfocus="lower()"
  onblur="upper()">
```

lower()和upper()函数

```
1 //2. 输入框聚焦后，展示小写；输入框离焦后，展示大写；-- onfocus ,
  onblur
2 function lower() { //小写
3     //a. 获取输入框元素对象
4     var input = document.getElementById("name");
5
6     //b. 将值转为小写
7     input.value = input.value.toLowerCase();
8 }
9
10 function upper() { //大写
11     //a. 获取输入框元素对象
12     var input = document.getElementById("name");
13
14     //b. 将值转为大写
15     input.value = input.value.toUpperCase();
16 }
```

#### 1.6.4.5 需求3

- 需求：

点击“全选”按钮使所有的复选框呈现被选中的状态，点击“反选”按钮使所有的复选框呈现取消勾选的状态。
- 分析：

点击按钮完成功能，所以我们需要给2个按钮绑定单击事件；我们需要设置所有复选框的状态，通过我们之前的案例，我们知道，我们需要获取所有的复选框，然后遍历，可以通过设置checked属性为true，来设置复选框为选中；那么反之，设置checked属性为false，来设置复选框为未选中。

- 步骤：

- 给全选和反选按钮绑定单击事件，分别绑定函数checkAll()和reverse()
- 在js中定义checkAll()和reverse()函数
- 对于checkAll()函数，首先通过name属性值为hobby来获取所有的复选框，然后遍历复选框，设置每个复选框的checked属性为true即可
- 对于reverse()函数，首先通过name属性值为hobby来获取所有的复选框，然后遍历复选框，设置每个复选框的checked属性为false即可

- 代码实现：

事件绑定：

```
1 <input type="button" value="全选" onclick="checkAll()">
2 <input type="button" value="反选" onclick="reverse()">
```

checkAll()和reverse()函数

```
1 //3. 点击 "全选" 按钮使所有的复选框呈现选中状态；点击 "反选" 按钮使
  所有的复选框呈现取消勾选的状态；
2 function checkAll() {
3     //a. 获取所有复选框元素对象
4     var hobbies = document.getElementsByName("hobby");
5
6     //b. 设置选中状态
7     for (let i = 0; i < hobbies.length; i++) {
8         const element = hobbies[i];
9         element.checked = true;
10    }
11
12 }
13
14 function reverse() {
15     //a. 获取所有复选框元素对象
16     var hobbies = document.getElementsByName("hobby");
17
18     //b. 设置未选中状态
19     for (let i = 0; i < hobbies.length; i++) {
20         const element = hobbies[i];
21         element.checked = false;
22    }
```

#### 1.6.4.6 完整代码

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>JS-事件-案例</title>
8  </head>
9  <body>
10
11       <br>
12
13      <input type="button" value="点亮" onclick="on()">
14      <input type="button" value="熄灭" onclick="off()">
15
16      <br> <br>
17      <input type="text" id="name" value="ITCAST" onfocus="lower()"
    onblur="upper()">
18      <br> <br>
19
20      <input type="checkbox" name="hobby"> 电影
21      <input type="checkbox" name="hobby"> 旅游
22      <input type="checkbox" name="hobby"> 游戏
23      <br>
24      <input type="button" value="全选" onclick="checkAll()">
25      <input type="button" value="反选" onclick="reverse()">
26  </body>
27  <script>
28      //1. 点击 "点亮" 按钮, 点亮灯泡; 点击 "熄灭" 按钮, 熄灭灯泡; --
    onclick
29      function on() {
30          //a. 获取img元素对象
31          var img = document.getElementById("light");
32
33          //b. 设置src属性
34          img.src = "img/on.gif";
35      }
36

```

```
37     function off() {
38         //a. 获取img元素对象
39         var img = document.getElementById("light");
40
41         //b. 设置src属性
42         img.src = "img/off.gif";
43     }
44
45     //2. 输入框聚焦后，展示小写；输入框离焦后，展示大写； -- onFocus ,
onblur
46     function lower() { //小写
47         //a. 获取输入框元素对象
48         var input = document.getElementById("name");
49
50         //b. 将值转为小写
51         input.value = input.value.toLowerCase();
52     }
53
54     function upper() { //大写
55         //a. 获取输入框元素对象
56         var input = document.getElementById("name");
57
58         //b. 将值转为大写
59         input.value = input.value.toUpperCase();
60     }
61
62     //3. 点击 "全选" 按钮使所有的复选框呈现选中状态 ; 点击 "反选" 按钮使所
有的复选框呈现取消勾选的状态 ; -- onclick
63     function checkAll() {
64         //a. 获取所有复选框元素对象
65         var hobbies = document.getElementsByName("hobby");
66
67         //b. 设置选中状态
68         for (let i = 0; i < hobbies.length; i++) {
69             const element = hobbies[i];
70             element.checked = true;
71         }
72     }
73
74     function reverse() {
75         //a. 获取所有复选框元素对象
76         var hobbies = document.getElementsByName("hobby");
77
78         //b. 设置未选中状态
```



```

79         for (let i = 0; i < hobbies.length; i++) {
80             const element = hobbies[i];
81             element.checked = false;
82         }
83     }
84
85 </script>
86 </html>

```

## 2 Vue

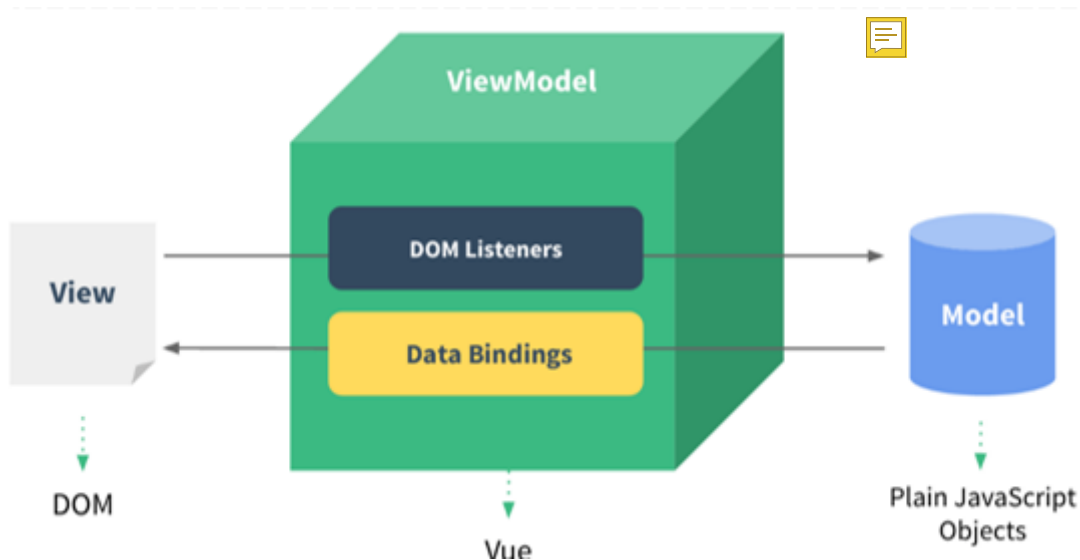
### 2.1 Vue概述

通过我们学习的html+css+js已经能够开发美观的页面了，但是开发的效率还有待提高，那么如何提高呢？我们先来分析下页面的组成。一个完整的html页面包括了视图和数据，数据是通过请求从后台获取的，那么意味着我们需要将后台获取到的数据呈现到页面上，很明显，这就需要我们使用DOM操作。正因为这种开发流程，所以我们引入了一种叫做MVVM(Model-View-ViewModel)的前端开发思想，即让我们开发者更加关注数据，而非数据绑定到视图这种机械化的操作。那么具体什么是MVVM思想呢？

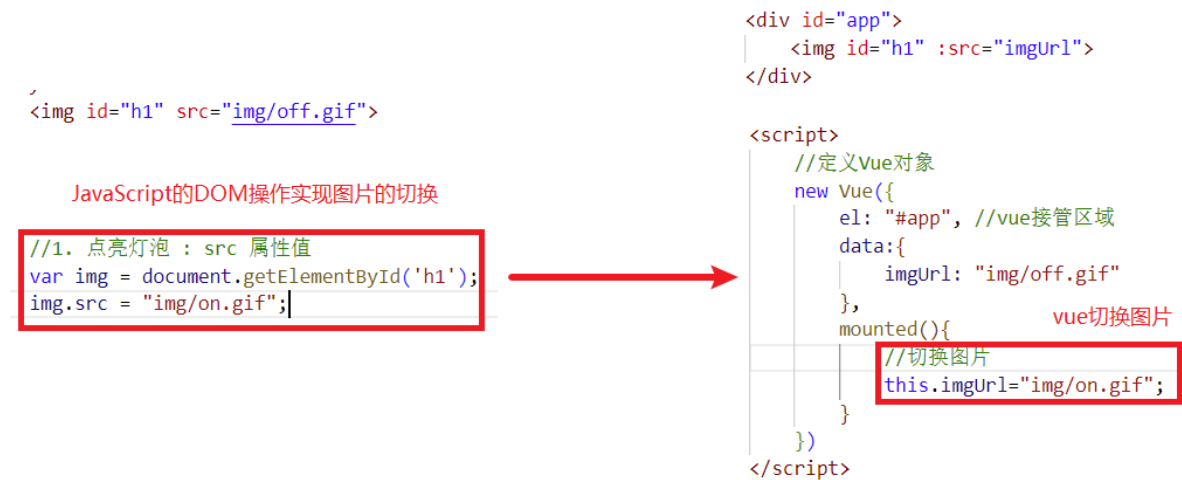
MVVM:其实是Model-View-ViewModel的缩写，有3个单词，具体释义如下：

- Model：数据模型，特指前端中通过请求从后台获取的数据
- View：视图，用于展示数据的页面，可以理解成我们的html+css搭建的页面，但是没有数据
- ViewModel：数据绑定到视图，负责将数据（Model）通过JavaScript的DOM技术，将数据展示到视图（View）上

如图所示就是MVVM开发思想的含义：



基于上述的MVVM思想，其中的Model我们可以通过Ajax来发起请求从后台获取；对于View部分，我们将来会学习一款ElementUI框架来替代HTML+CSS来更加方便的搭建View；而今天我们要学习的就是侧重于ViewModel部分开发的vue前端框架，用来替代JavaScript的DOM操作，让数据展示到视图的代码开发变得更加的简单。可以简单到什么程度呢？可以参考下图对比：



在更加复杂的dom操作中，vue只会变得更加的简单！在上述的代码中，我们看不到之前的DOM操作，因为vue全部帮我们封装好了。

接下来我们来介绍一下vue。

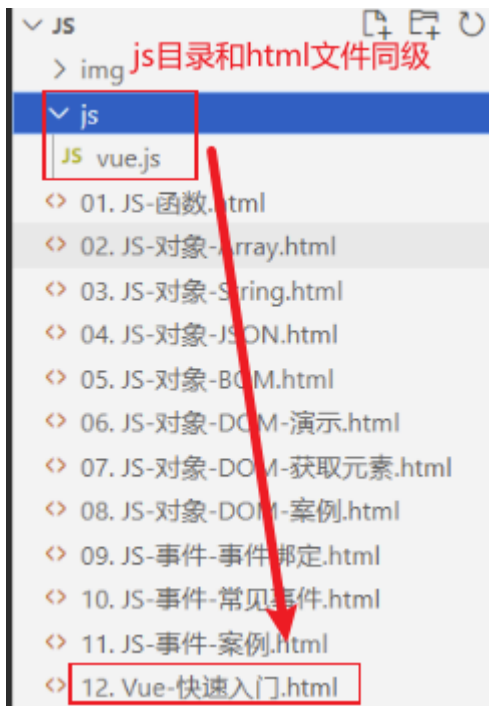
Vue.js (读音 /vju:/, 类似于 **view**) 是一套构建用户界面的 **渐进式框架**。与其他重量级框架不同的是，Vue 采用自底向上增量开发的设计。Vue 的核心库只关注视图层，并且非常容易学习，非常容易与其它库或已有项目整合。Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

框架即是一个半成品软件，是一套可重用的、通用的、软件基础代码模型。基于框架进行开发，更加快捷、更加高效。

## 2.2 快速入门

接下来我们通过一个vue的快速入门案例，来体验一下vue。

第一步：在VS Code中创建名为12. Vue-快速入门.html的文件，并且在html文件同级创建js目录，将资料/vue.js文件目录下得vue.js拷贝到js目录，如下图所示：



第二步：然后编写<script>标签来引入vue.js文件，代码如下：

```
1 <script src="js/vue.js"></script>
```

第三步：在js代码区域定义vue对象,代码如下：

```
1 <script>
2     //定义Vue对象
3     new Vue ({
4         el: "#app", //vue接管区域
5         data:{
6             message: "Hello Vue"
7         }
8     })
9 </script>
```

在创建vue对象时，有几个常用的属性：

- el： 用来指定哪儿些标签受 Vue 管理。 该属性取值 #app 中的 app 需要是受管理的标签的id属性值
- data： 用来定义数据模型
- methods： 用来定义函数。这个我们在后面就会用到

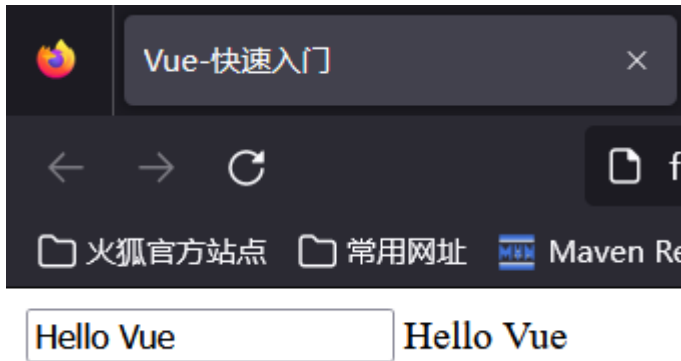
第四步：在html区域编写视图，其中{{}}是插值表达式，用来将vue对象中定义的model展示到页面上的

```

1   <body>
2       <div id="app">
3           <input type="text" v-model="message">
4               {{message}}
5       </div>
6   </body>

```

浏览器打开效果如图所示：



整体代码如下：

```

1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-
7           scale=1.0">
8       <title>Vue-快速入门</title>
9       <script src="js/vue.js"></script>
10  </head>
11  <body>
12      <div id="app">
13          <input type="text" v-model="message">
14              {{message}}
15      </div>
16
17  </body>
18  <script>
19      //定义Vue对象
20      new Vue({
21          el: "#app", //vue接管区域
22          data:{
23              message: "Hello Vue"

```

```
24      }
25    })
26  </script>
27  </html>
```

## 2.3 Vue指令

在上述的快速入门中，我们发现了html中输入了一个没有学过的属性 `v-model`，这个就是vue的指令。

**指令：**HTML 标签上带有 `v-` 前缀的特殊属性，不同指令具有不同含义。例如：`v-if`，`v-for`...

在vue中，通过大量的指令来实现数据绑定到视图的，所以接下来我们需要学习vue的常用指令，如下表所示：

指令	作用
<code>v-bind</code>	为HTML标签绑定属性值，如设置 <code>href</code> ， <code>css</code> 样式等
<code>v-model</code>	在表单元素上创建双向数据绑定
<code>v-on</code>	为HTML标签绑定事件
<code>v-if</code>	条件性的渲染某元素，判定为 <code>true</code> 时渲染，否则不渲染
<code>v-else</code>	
<code>v-else-if</code>	
<code>v-show</code>	根据条件展示某元素，区别在于切换的是 <code>display</code> 属性的值
<code>v-for</code>	列表渲染，遍历容器的元素或者对象的属性

### 2.3.1 v-bind和v-model

我们首先来学习`v-bind`指令和`v-model`指令。

指令	作用
v-bind	为HTML标签绑定属性值，如设置 href ， css样式等
v-model	在表单元素上创建双向数据绑定

- v-bind: 为HTML标签绑定属性值，如设置 href ， css样式等。当vue对象中的数据模型发生变化时，标签的属性值会随之发生变化。

接下来我们通过代码来演示。

首先我们在VS Code中创建名为13. Vue-指令-v-bind和v-model.html的文件，然后准备好如下代码：

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>Vue-指令-v-bind</title>
8      <script src="js/vue.js"></script>
9  </head>
10 <body>
11     <div id="app">
12
13         <a >链接1</a>
14         <a >链接2</a>
15
16         <input type="text" >
17
18     </div>
19 </body>
20 <script>
21     //定义Vue对象
22     new Vue({
23         el: "#app", //vue接管区域
24         data:{
25             url: "https://www.baidu.com"
26         }
27     })
28 </script>
29 </html>
```

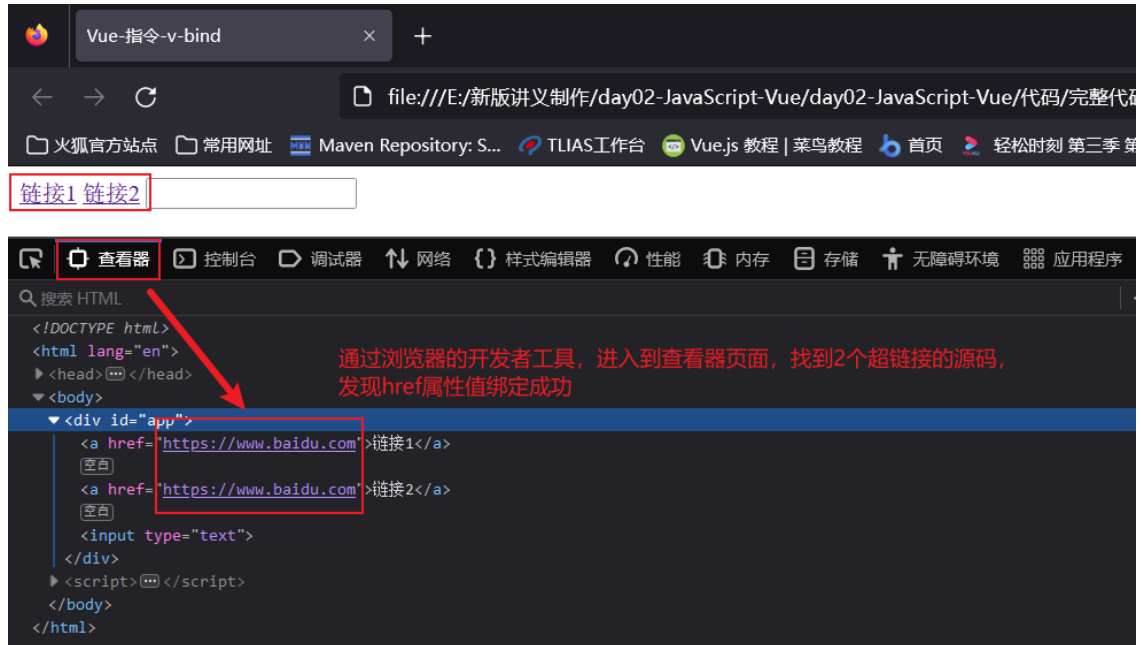
在上述的代码中，我们需要给标签的href属性赋值，并且值应该来自于vue对象的数据模型中的url变量。所以编写如下代码：

```
1 <a v-bind:href="url">链接1</a>
```

在上述的代码中，v-bind指令是可以省略的，但是:不能省略，所以第二个超链接的代码编写如下：

```
1 <a :href="url">链接2</a>
```

浏览器打开，2个超链接都可以点击，然后跳转到百度去！效果如图所示：



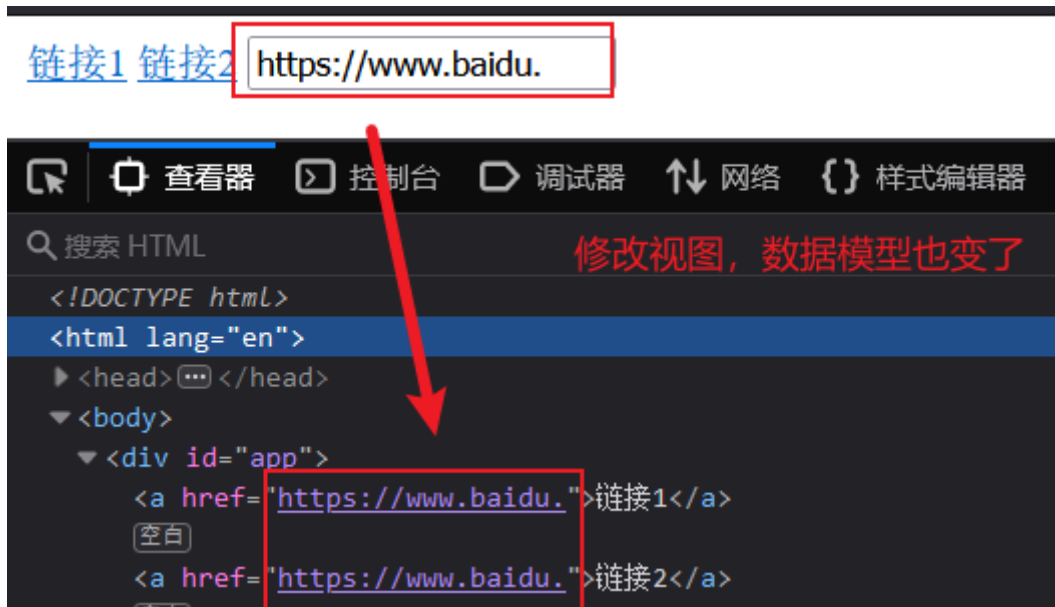
**注意：**html属性前面有:表示采用的vue的属性绑定！

- v-model： 在表单元素上创建双向数据绑定。什么是双向？
  - vue对象的data属性中的数据变化，视图展示会一起变化
  - 视图数据发生变化，vue对象的data属性中的数据也会随着变化。

data属性中数据变化，我们知道可以通过赋值来改变，但是视图数据为什么会发生变化呢？**只有表单项标签！所以双向绑定一定是使用在表单项标签上的。**编写如下代码：

```
1 <input type="text" v-model="url">
```

打开浏览器，我们修改表单项标签，发现vue对象data中的数据也发生了变化，如下图所示：



通过上图我们发现，我们只是改变了表单数据，那么我们之前超链接的绑定的数据值也发生了变化，为什么？

就是因为我们双向绑定，在视图发生变化时，同时vue的data中的数据模型也会随着变化。那么这个在企业开发的应用场景是什么？

**双向绑定的作用：可以获取表单的数据的值，然后提交给服务器**

整体代码如下：

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-
  scale=1.0">
7   <title>Vue-指令-v-bind</title>
8   <script src="js/vue.js"></script>
9 </head>
10 <body>
11   <div id="app">
12
13     <a v-bind:href="url">链接1</a>
14     <a :href="url">链接2</a>
15
16     <input type="text" v-model="url">
17
18   </div>
19 </body>
20 <script>
```



```

21      //定义vue对象
22      new Vue({
23          el: "#app", //vue接管区域
24          data:{
25              url: "https://www.baidu.com"
26          }
27      })
28  </script>
29  </html>

```

### 2.3.2 v-on

接下来我们学习一下v-on指令。

v-on：用来给html标签绑定事件的。**需要注意的是如下2点：**

- v-on语法给标签的事件绑定的函数，必须是vue对象中声明的函数
- v-on语法绑定事件时，事件名相比较js中的事件名，没有on

例如：在js中，事件绑定demo函数

```

1  <input onclick="demo()">

```

vue中，事件绑定demo函数

```

1  <input v-on:click="demo()">

```

接下来我们通过代码演示。

首先在VS Code中创建名为14. Vue-指令-v-on.html的文件，提前准备如下代码：

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>Vue-指令-v-on</title>
8      <script src="js/vue.js"></script>
9  </head>
10 <body>
11     <div id="app">
12

```

```

13         <input type="button" value="点我一下">
14         <input type="button" value="点我一下">
15
16     </div>
17 </body>
18 <script>
19     //定义Vue对象
20     new Vue({
21         el: "#app", //vue接管区域
22         data:{
23
24         },
25         methods: {
26
27         }
28     })
29 </script>
30 </html>

```

然后我们需要在vue对象的methods属性中定义事件绑定时需要的handle()函数，代码如下：

```

1     methods: {
2         handle: function() {
3             alert("你点我了一下...");
4         }
5     }

```

然后我们给第一个按钮，通过v-on指令绑定单击事件，代码如下：

```

1     <input type="button" value="点我一下" v-on:click="handle()">

```

同样，v-on也存在简写方式，即v-on: 可以替换成@，所以第二个按钮绑定单击事件的代码如下：

```

1     <input type="button" value="点我一下" @click="handle()">

```

完整代码如下：

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7     <title>Vue-指令-v-on</title>

```

```

8      <script src="js/vue.js"></script>
9  </head>
10 <body>
11     <div id="app">
12
13         <input type="button" value="点我一下" v-on:click="handle()">
14
15         <input type="button" value="点我一下" @click="handle()">
16
17     </div>
18 </body>
19 <script>
20     //定义Vue对象
21     new Vue({
22         el: "#app", //vue接管区域
23         data:{
24
25         },
26         methods: {
27             handle: function() {
28                 alert("你点我了一下...");
29             }
30         }
31     })
32 </script>
33 </html>

```

### 2.3.3 v-if和v-show

指令	描述
v-if	条件性的渲染某元素，判定为true时渲染，否则不渲染
v-if-else	
v-else	
v-show	根据条件展示某元素，区别在于切换的是display属性的值

我们直接通过代码来演示效果。在VS Code中创建名为15. Vue-指令-v-if和v-show.html的文件，提前准备好如下代码：

```

1  <!DOCTYPE html>

```

```

2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7       <title>Vue-指令-v-if与v-show</title>
8       <script src="js/vue.js"></script>
9   </head>
10  <body>
11      <div id="app">
12
13          年龄<input type="text" v-model="age">经判定,为:
14          <span>年轻人 (35及以下)</span>
15          <span>中年人 (35-60)</span>
16          <span>老年人 (60及以上)</span>
17
18          <br><br>
19      </div>
20  </body>
21  <script>
22      //定义Vue对象
23      new Vue({
24          el: "#app", //vue接管区域
25          data:{
26              age: 20
27          },
28          methods: {
29
30          }
31      })
32  </script>
33  </html>

```

其中采用了双向绑定到age属性，意味着我们可以通过表单输入框来改变age的值。

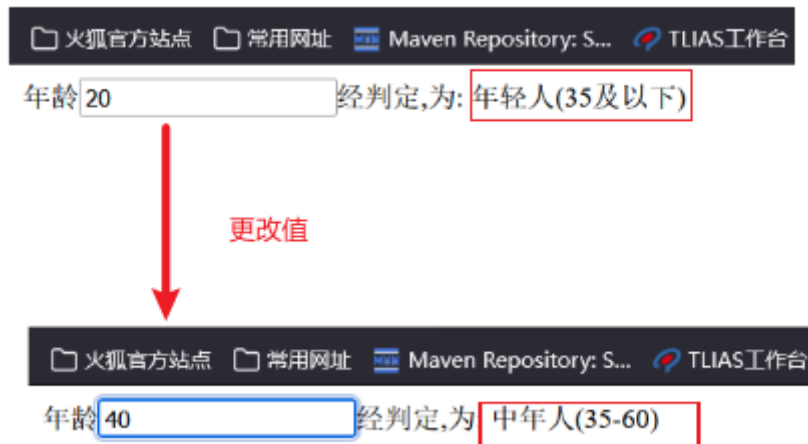
需求是当我们改变年龄时，需要动态判断年龄的值，呈现对应的年龄的文字描述。年轻人，我们需要使用条件判断 `age<=35`，中年人我们需要使用条件判断 `age>35 && age<60`，其他情况是老年人。所以通过v-if指令编写如下代码：

```

1   年龄<input type="text" v-model="age">经判定,为:
2   <span v-if="age <= 35">年轻人 (35及以下)</span>
3   <span v-else-if="age > 35 && age < 60">中年人 (35-60)</span>
4   <span v-else>老年人 (60及以上)</span>

```

浏览器打开测试效果如下图：



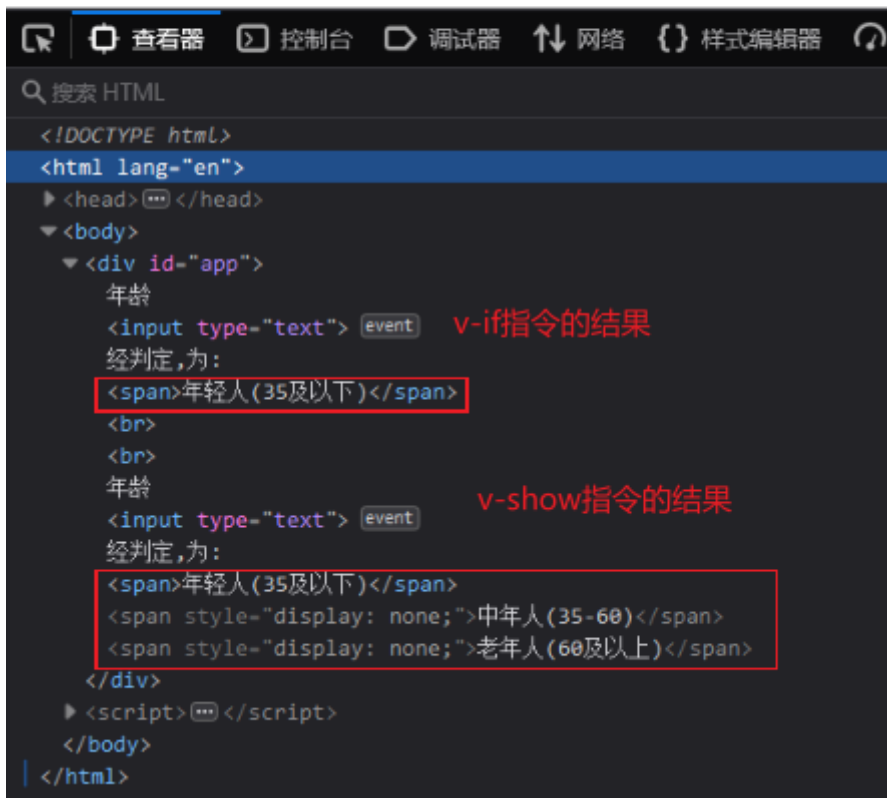
`v-show`和`v-if`的作用效果是一样的，只是原理不一样。复制上述html代码，修改`v-if`指令为`v-show`指令，代码如下：

```
1  年龄<input type="text" v-model="age">经判定,为:
2  <span v-show="age <= 35">年轻人 (35及以下) </span>
3  <span v-show="age > 35 && age < 60">中年人 (35-60) </span>
4  <span v-show="age >= 60">老年人 (60及以上) </span>
```

打开浏览器，展示效果如下所示：

年龄  经判定,为: 年轻人(35及以下)

年龄  经判定,为: 年轻人(35及以下)



可以发现,浏览器呈现的效果是一样的,但是浏览器中html源码不一样。v-if指令,不满足条件的标签代码直接没了,而v-show指令中,不满足条件的代码依然存在,只是添加了css样式来控制标签不去显示。

完整代码如下:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
7  scale=1.0">
8      <title>Vue-指令-v-if与v-show</title>
9      <script src="js/vue.js"></script>
10 </head>
11 <body>
12     <div id="app">
```

```

13         年龄<input type="text" v-model="age">经判定,为:
14         <span v-if="age <= 35">年轻人 (35及以下)</span>
15         <span v-else-if="age > 35 && age < 60">中年人 (35-60)</span>
16         <span v-else>老年人 (60及以上)</span>
17
18         <br><br>
19
20         年龄<input type="text" v-model="age">经判定,为:
21         <span v-show="age <= 35">年轻人 (35及以下)</span>
22         <span v-show="age > 35 && age < 60">中年人 (35-60)</span>
23         <span v-show="age >= 60">老年人 (60及以上)</span>
24
25     </div>
26 </body>
27 <script>
28     //定义Vue对象
29     new Vue({
30         el: "#app", //vue接管区域
31         data:{
32             age: 20
33         },
34         methods: {
35
36         }
37     })
38 </script>
39 </html>

```

### 2.3.4 v-for

v-for: 从名字我们就能看出, 这个指令是**用来遍历**的。其语法格式如下:

```

1  <标签 v-for="变量名 in 集合模型数据">
2      {{变量名}}
3  </标签>

```

需要注意的是: **需要循环那个标签, v-for 指令就写在那个标签上。**

有时我们**遍历时需要使用索引**, 那么v-for指令遍历的语法格式如下:

```

1  <标签 v-for="(变量名,索引变量) in 集合模型数据">
2      <!--索引变量是从0开始, 所以要表示序号的话, 需要手动的加1-->
3      {{索引变量 + 1}} {{变量名}}
4  </标签>

```

接下来, 我们再VS Code中创建名为16. Vue-指令-v-for.html的文件编写代码演示, 提前准备如下代码:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>Vue-指令-v-for</title>
8      <script src="js/vue.js"></script>
9  </head>
10 <body>
11     <div id="app">
12
13     </div>
14 </body>
15 <script>
16     //定义Vue对象
17     new Vue({
18         el: "#app", //vue接管区域
19         data:{
20             addrs:["北京", "上海", "西安", "成都", "深圳"]
21         },
22         methods: {
23
24         }
25     })
26 </script>
27 </html>
```

然后分别编写2种遍历语法, 来遍历数组, 展示数据, 代码如下:

```
1  <div id="app">
2      <div v-for="addr in addrs">{{addr}}</div>
3      <hr>
4      <div v-for="(addr,index) in addrs">{{index + 1}} : {{addr}}</div>
5  </div>
```

浏览器打开, 呈现如下效果:

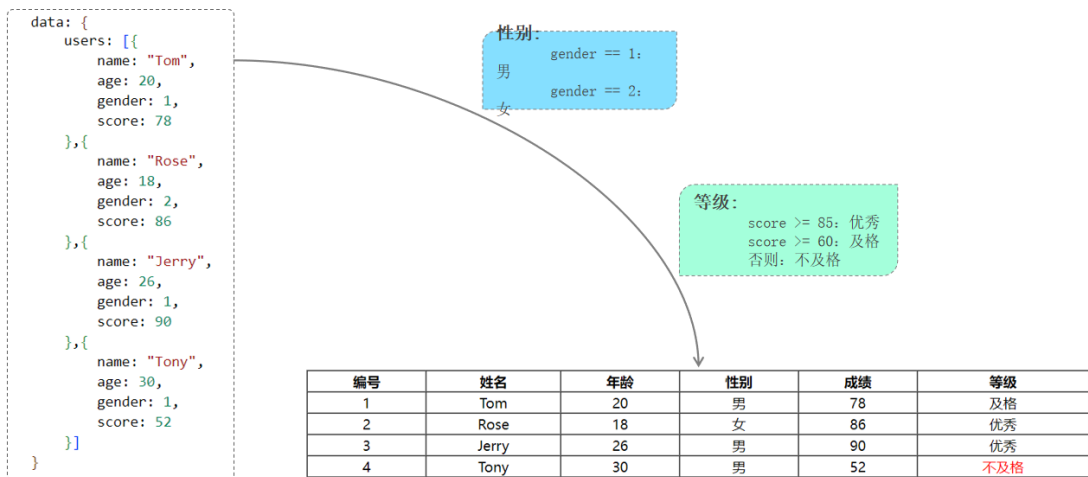


北京  
上海  
西安  
成都  
深圳

1: 北京  
2: 上海  
3: 西安  
4: 成都  
5: 深圳

### 2.3.5 案例

- 需求：



如上图所示，我们提供好了数据模型，users是数组集合，提供了多个用户信息。然后我们需要将数据以表格的形式，展示到页面上，其中，性别需要转换成中文男女，等级需要将分数数值转换成对应的等级。

- 分析：

首先我们肯定需要遍历数组的，所以需要使用v-for标签；然后我们每一条数据对应一行，所以v-for需要添加在tr标签上；其次我们需要将编号，所以需要使用索引的遍历语法；然后我们要将数据展示到表格的单元格中，所以我们需要使用{{}}插值表达式；最后，我们需要转换内容，所以我们需要使用v-if指令，进行条件判断和内容的转换

- 步骤：

- 使用v-for的带索引方式添加到表格的<tr>标签上
- 使用{{}}插值表达式展示内容到单元格
- 使用索引+1来作为编号

- 使用v-if来判断，改变性别和等级这2列的值
- 代码实现：

首先创建名为17. Vue-指令-案例.html的文件，提前准备如下代码：

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>Vue-指令-案例</title>
8      <script src="js/vue.js"></script>
9  </head>
10 <body>
11
12     <div id="app">
13
14         <table border="1" cellspacing="0" width="60%">
15             <tr>
16                 <th>编号</th>
17                 <th>姓名</th>
18                 <th>年龄</th>
19                 <th>性别</th>
20                 <th>成绩</th>
21                 <th>等级</th>
22             </tr>
23         </table>
24
25     </div>
26
27 </body>
28
29 <script>
30     new Vue({
31         el: "#app",
32         data: {
33             users: [{
34                 name: "Tom",
35                 age: 20,
36                 gender: 1,
37                 score: 78
38             }, {
39                 name: "Rose",
```

```

40         age: 18,
41         gender: 2,
42         score: 86
43     }, {
44         name: "Jerry",
45         age: 26,
46         gender: 1,
47         score: 90
48     }, {
49         name: "Tony",
50         age: 30,
51         gender: 1,
52         score: 52
53     }]
54 },
55 methods: {
56
57 },
58 })
59 </script>
60 </html>

```

然后在<tr>上添加v-for进行遍历，以及通过插值表达式{{}}和v-if指令来填充内容和改变内容，其代码如下：

```

1  <tr align="center" v-for="(user,index) in users">
2      <td>{{index + 1}}</td>
3      <td>{{user.name}}</td>
4      <td>{{user.age}}</td>
5      <td>
6          <span v-if="user.gender == 1">男</span>
7          <span v-if="user.gender == 2">女</span>
8      </td>
9      <td>{{user.score}}</td>
10     <td>
11         <span v-if="user.score >= 85">优秀</span>
12         <span v-else-if="user.score >= 60">及格</span>
13         <span style="color: red;" v-else>不及格</span>
14     </td>
15 </tr>

```

其完整代码如下：

```

1  <!DOCTYPE html>

```

```
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-
  scale=1.0">
7   <title>Vue-指令-案例</title>
8   <script src="js/vue.js"></script>
9 </head>
10 <body>
11
12   <div id="app">
13
14     <table border="1" cellspacing="0" width="60%">
15       <tr>
16         <th>编号</th>
17         <th>姓名</th>
18         <th>年龄</th>
19         <th>性别</th>
20         <th>成绩</th>
21         <th>等级</th>
22       </tr>
23
24       <tr align="center" v-for="(user,index) in users">
25         <td>{{index + 1}}</td>
26         <td>{{user.name}}</td>
27         <td>{{user.age}}</td>
28         <td>
29           <span v-if="user.gender == 1">男</span>
30           <span v-if="user.gender == 2">女</span>
31         </td>
32         <td>{{user.score}}</td>
33         <td>
34           <span v-if="user.score >= 85">优秀</span>
35           <span v-else-if="user.score >= 60">及格</span>
36           <span style="color: red;" v-else>不及格</span>
37         </td>
38       </tr>
39     </table>
40
41   </div>
42
43 </body>
44
```

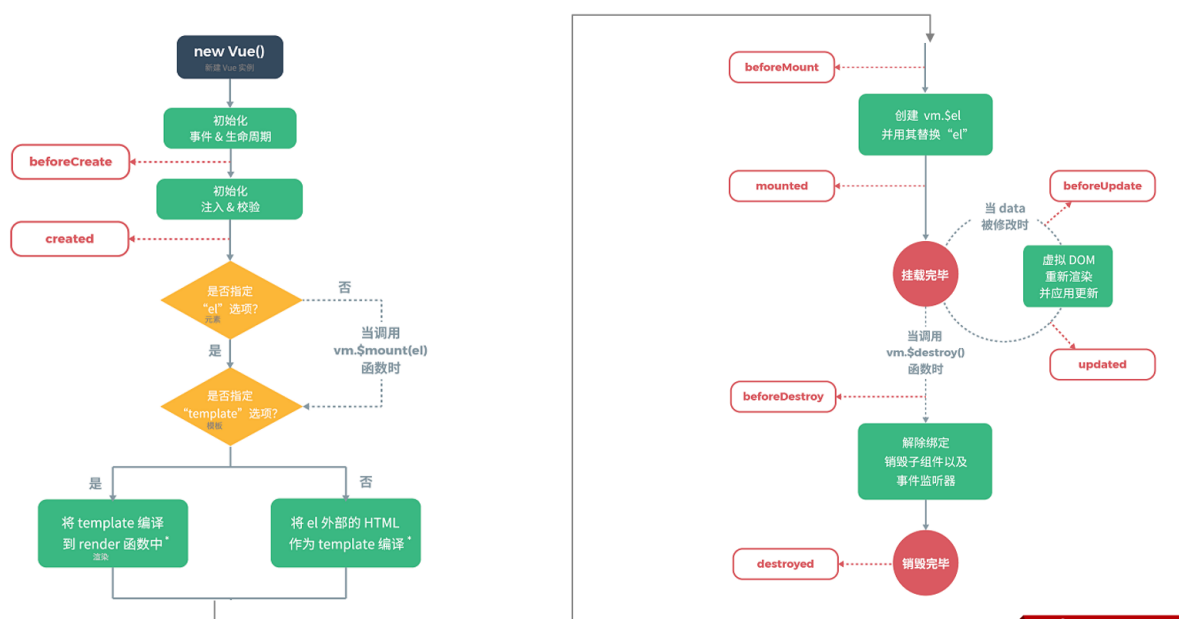
```
45 <script>
46     new Vue({
47         el: "#app",
48         data: {
49             users: [{
50                 name: "Tom",
51                 age: 20,
52                 gender: 1,
53                 score: 78
54             }, {
55                 name: "Rose",
56                 age: 18,
57                 gender: 2,
58                 score: 86
59             }, {
60                 name: "Jerry",
61                 age: 26,
62                 gender: 1,
63                 score: 90
64             }, {
65                 name: "Tony",
66                 age: 30,
67                 gender: 1,
68                 score: 52
69             }]
70         },
71         methods: {
72
73         },
74     })
75 </script>
76 </html>
```

## 2.4 生命周期

vue的生命周期：指的是vue对象从创建到销毁的过程。vue的生命周期包含8个阶段：每触发一个生命周期事件，会自动执行一个生命周期方法，这些生命周期方法也被称为钩子方法。其完整的生命周期如下图所示：

状态	阶段周期
beforeCreate	创建前
created	创建后
beforeMount	挂载前
mounted	挂载完成
beforeUpdate	更新前
updated	更新后
beforeDestroy	销毁前
destroyed	销毁后

下图是 Vue 官网提供的从创建 Vue 到效果 Vue 对象的整个过程及各个阶段对应的钩子函数：



其中我们需要重点关注的是**mounted**，其他的我们了解即可。

**mounted**：挂载完成，Vue初始化成功，HTML页面渲染成功。以后我们一般用于页面初始化自动的ajax请求后台数据



我们在VS Code中创建名为18. Vue-生命周期.html的文件编写代码来演示效果，提前准备如下代码：

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">

```

```
6     <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7     <title>Vue-指令-v-for</title>
8     <script src="js/vue.js"></script>
9 </head>
10 <body>
11     <div id="app">
12
13     </div>
14 </body>
15 <script>
16     //定义Vue对象
17     new Vue({
18         el: "#app", //vue接管区域
19         data:{
20
21         },
22         methods: {
23
24         }
25     })
26 </script>
27 </html>
```

然后我们编写mounted声明周期的钩子函数，与methods同级，代码如下：

```
1 <script>
2     //定义Vue对象
3     new Vue({
4         el: "#app", //vue接管区域
5         data:{
6
7         },
8         methods: {
9
10        },
11        mounted () {
12            alert("vue挂载完成,发送请求到服务端")
13        }
14    })
15 </script>
```

浏览器打开，运行结果如下：我们发现，自动打印了这句话，因为页面加载完成，vue对象创建并且完成了挂在，此时自动触发mounted所绑定的钩子函数，然后自动执行，弹框。

