

# 04-JavaScript

HTML完成了架子，CSS做了美化，但是网页是死的，我们需要为其注入灵魂，接下来我们学习JavaScript，这门语言会让我们的页面能够和用户进行交互。

**JavaScript（简称JS）是一种跨平台、功能强大、广泛应用的面向对象脚本语言。**它具有动态性、强大的客户端交互能力、广泛的应用领域、丰富的生态系统等特点。JavaScript的发展使得网页和应用程序更加交互和动态，**是现代Web开发的核心技术之一。**

**JavaScript 和 Java 是完全不同的语言**，其在1995年由Netscape(网景)的Brendan Eich设计，最初命名为LiveScript，后来Netscape在与Sun合作之后，**为了营销考虑将其改名为JavaScript。**

JavaScript最初受Java启发而开始设计的，目的之一就是“**看上去像Java**”，因此语法上有类似之处，一些名称和命名规范也借自Java，但JavaScript的主要设计原则源自Self和Scheme。

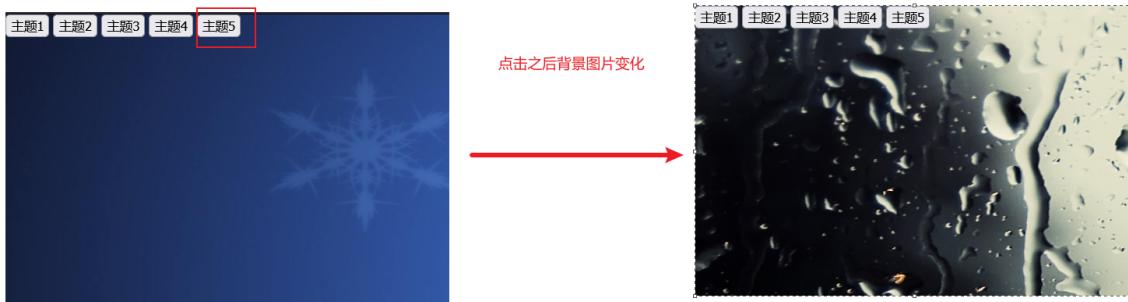
JavaScript的标准是**ECMAScript**。截至2012年，所有浏览器都完整的支持ECMAScript 5.1，旧版本的浏览器至少支持ECMAScript 3标准。2015年6月17日，ECMA国际组织发布了ECMAScript的第六版，该版本**正式名称**为ECMAScript 2015，但通常被称为**ECMAScript 6**或者ES2015。

ECMA：ECMA国际（前身为欧洲计算机制造商协会），制定了标准化的脚本程序设计语言ECMAScript，使得这种语言得到广泛应用。

ECMAScript6 (ES6) 是最主流的 JavaScript 版本（发布于 2015 年）。

## 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。



## 2 引入

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：

```
<script>
    alert("Hello JavaScript")
</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，不可以自闭合

案例：

html文件

```
<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 **01-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：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JS-引入方式</title>
    <!-- 内部脚本 -->
    <script>
        alert('Hello JS');
    </script>
</head>
<body>
</body>
</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中编写内容如下：

```
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：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JS-引入方式</title>
```

```
<!-- 内部脚本 -->
<!-- <script>
    alert('Hello JS');
</script> -->

<!-- 外部脚本 -->
<script src="js/demo.js"></script>
</head>
<body>

</body>
</html>
```

第七步：浏览器刷新效果如图：



## 3 基础语法

### 3.1 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，建议加上
- 大括号表示代码块

```
//判断
if(count == 3){
    alert(count);
}
```

- 注释

单行注释：// 注释内容

多行注释：/\* 注释内容 \*/

**输出形式：** JS中存在3种常见输出语句

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

**案例展示：**



The screenshot displays three examples of JavaScript output statements:

- Alert Box:** Shows a dialog box with the text "Hello JavaScript".
- Browser Output:** Shows the text "Hello JavaScript" displayed directly in the browser window.
- Console Output:** Shows the text "Hello JavaScript" in the browser's developer tools console.

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 **02-JS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；

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

</body>
<script>
    /* alert("JS"); */
    //方式一：弹出警告框
    window.alert("hello js");
</script>
</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：

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>JS-基本语法</title>
</head>
<body>

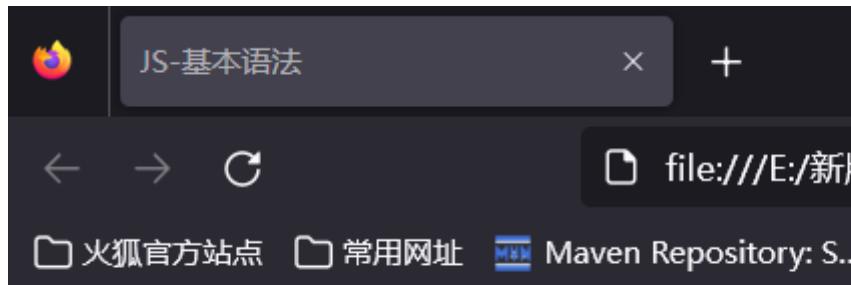
</body>
<script>
/* alert("JS"); */

//方式一：弹出警告框
// window.alert("hello js");

//方式二：写入html页面中
document.write("hello js");

</script>
</html>
```

刷新浏览器，效果如图所示：



hello js

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：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JS-基本语法</title>
</head>
<body>

</body>
<script>
    /* alert("JS"); */

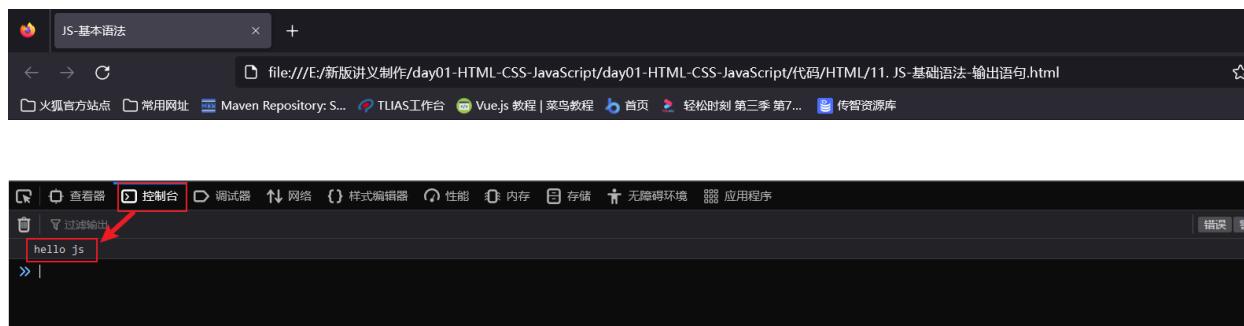
    // 方式一：弹出警告框
    // window.alert("hello js");

    // // 方式二：写入html页面中
    // document.write("hello js");

    // 方式三：控制台输出
    console.log("hello js");
</script>
</html>

```

浏览器f12抓包，去控制台页面，如图所示：



## 3.2 变量

变量是一门编程语言必不可少的，在JS中变量的声明和Java中不同。

JS主要通过如下3个关键字来声明变量：

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

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 **03-JS变量.html** 的文件：

第二步：编写代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JS-基础语法</title>
</head>
<body>

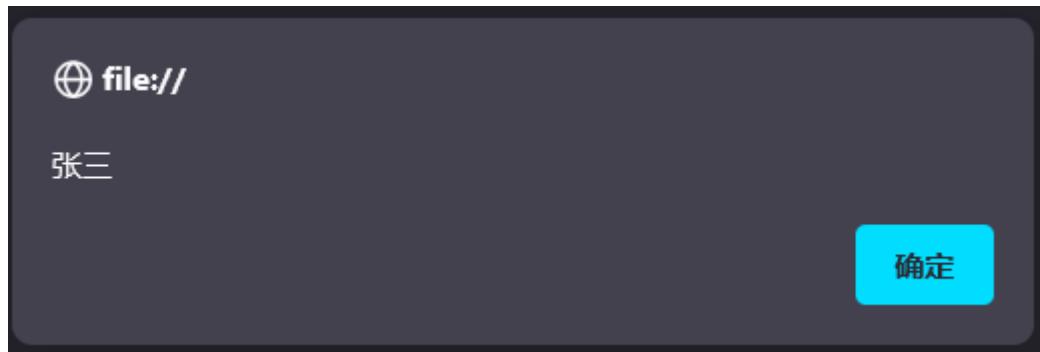
</body>
```

```
<script>

    //var定义变量
    var a = 10;
    a = "张三";
    alert(a);

</script>
</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声明的变量的作用域是全局的，注释掉之前的代码，添加如下代码：

```
<script>
    //var定义变量
    // var a = 10;
    // a = "张三";
    // alert(a);

    //特点1：作用域为全局
    {
        var x = 1;
    }
    alert(x);
</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：

```
{  
    var x = 1;  
    var x = "A";  
}  
alert(x);
```

浏览器弹出内容是A



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：

```
<script>  
  
//var 定义变量
```

```

// var a = 10;
// a = "张三";
// alert(a);

//特点1：作用域比较大，全局变量
//特点2：可以重复定义
// {
//     var x = 1;
//     var x = "A";
// }
// alert(x);

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

</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VS Code直接帮我们报错了，说明let声明的变量不能重复定义。

```

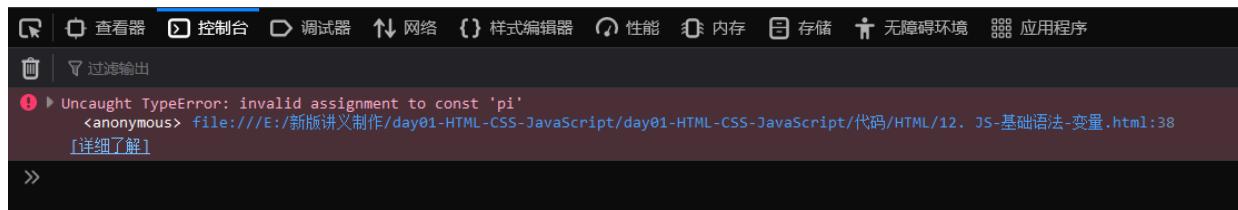
//let：局部变量；不能重复定义
{
    let x = 1;
    let x = 2;
    alert(x);
}

```

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：

```
const pi = 3.14;  
pi = 3.15;  
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：

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-  
    scale=1.0">  
    <title>JS-基础语法</title>  
</head>  
<body>  
  
</body>  
<script>  
  
    //var 定义变量  
    // var a = 10;  
    // a = "张三";  
    // alert(a);  
  
    //特点1：作用域比较大，全局变量  
    //特点2：可以重复定义的
```

```
// {
//     var x = 1;
//     var x = "A";
// }
// alert(x);

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

//const: 常量,其值不可以修改
const pi = 3.14;
pi = 3.15;
alert(pi);

</script>
</html>
```

### 3.3 类型

虽然JS是弱数据类型的语言，但是JS中也存在数据类型，可分为：**原始类型**和**引用类型**。

具体如下：

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

注意：使用 **typeof** 运算符 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 **04-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

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JS-数据类型</title>
</head>
<body>

</body>
<script>

    //原始数据类型
    alert(typeof 3); //number
    alert(typeof 3.14); //number

    alert(typeof "A"); //string

```

```

    alert(typeof 'Hello');//string

    alert(typeof true); //boolean
    alert(typeof false); //boolean

    alert(typeof null); //object ?

var a;
alert(typeof a); //undefined

</script>
</html>

```

**注释：**您也许会问，为什么 `typeof` 运算符对于 `null` 值会返回 "Object"。这实际上是 JavaScript 最初实现中的一个错误，然后被 ECMAScript 沿用了。现在，`null` 被认为是对象的占位符，从而解释了这一矛盾，但从技术上来说，它仍然是原始值。

## 3.4 运算符

熟悉了JS的数据类型了，那么我们需要学习JS中的运算符，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	<code>+ , - , * , / , % , ++ , --</code>
赋值运算符	<code>= , += , -= , *= , /= , %=</code>
比较运算符	<code>&gt; , &lt; , &gt;= , &lt;= , != , == , ===</code>
逻辑运算符	<code>&amp;&amp; ,    , !</code>
三元运算符	<code>条件表达式 ? true_value: false_value</code>

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

- ===：不光比较值，还要比较类型，如果类型不一致，直接返回false

接下来我们通过代码来演示JS中的运算符，主要区分JS中和Java中不一致的地方

- 第一步：在vs Code中创建名为 **05-JS运算符.html** 的文件
- 第二步：编写代码

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JS-运算符</title>
</head>
<body>

</body>
<script>
    var age = 20;
    var _age = "20";
    var $age = 20;

    alert(age == _age);      //true 只比较值
    alert(age === _age);    //false 类型不一样
    alert(age === $age);    //true 类型一样，值一样

</script>
</html>
```

类型转换：

在JS中虽然不区分数据类型，但是涉及到数值计算时，需要进行**类型转换**。

JS中可通过 **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：

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

此外，在JS中**其他类型转换为boolean类型**时：

- **0、null、undefined、""、NaN** 自动转换为false
- 其他情况转换为true

注释掉之前的代码，添加如下代码：

```
if(0){ //false  
    alert("0 转换为 false");  
}
```

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：

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

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；

⊕ file://

除0和NaN其他数字都转为 true

确定

其他情况请自行演示，完整代码如下：

```
// if(0) { //false
//     alert("0 转换为false");
// }

// if(NaN) { //false
//     alert("NaN 转换为false");
// }

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

// if("") { //false
//     alert("空字符串为false, 其他都是true");
// }

// if(null) { //false
//     alert("null转化为false");
// }

// if(undefined) { //false
//     alert("undefined转化为false");
// }
```

## 3.5 语句

流程控制语句if、switch、for等和Java保持一致。

可参考官方文档：[https://www.w3school.com.cn/jsref/jsref\\_statements.asp](https://www.w3school.com.cn/jsref/jsref_statements.asp)

相关案例 [06-JS语句.html](#)：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>06-JS语句</title>
</head>
<body>
    <div id="did" style="color: blue;font-size: 20px;">
        Hello World
    </div>

    <div>
        您喜欢的水果: <input id="myInput" type="text"
value="Banana">
    </div>
</body>
<script>
    //if语句
    var time = new Date().getHours();
    if (time < 9) {
        //修改标签值
        document.getElementById("did").innerHTML = "Good
Morning";
    }

    // 添加分割线
    document.write("<hr>");

```

```
// switch语句
var text;
var fruits = document.getElementById("myInput").value;

switch(fruits) {
    case "Banana":
        text = "Banana is good!";
        break;
    case "Orange":
        text = "I am not a fan of orange.";
        break;
    case "Apple":
        text = "How you like them apples?";
        break;
    default:
        text = "I have never heard of that fruit...";
}
alert(text);

// for循环
var text = "";
var i;
for (i = 0; i < 5; i++) {
    text += "The number is " + i + "<br>";
}
document.write(text);

// 添加分割线
document.write("<hr>");

// while循环
var text = "";
var i = 10;
while (i > 5) {
    text += "<br>The number is " + i;
    i--;
}
```

```
document.write(text);

</script>
</html>
```

## 4 函数

---

在Java中我们为了提高代码的复用性，可以定义方法。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。

### 4.1 格式1

第一种定义格式如下：

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

#### 注意：

- 形式参数不需要声明类型  
原因：JavaScript是弱数据类型的语言，不管什么类型都是let或者var去声明，加上没有意义
- 返回值也不需要声明类型，直接return即可

#### 示例：

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

接下来我们需要在VS Code中编写代码来演示

- 第一步：创建名为 **07-JS函数.html** 文件，然后在<script>中定义函数：

```
<script>  
    function add(a,b){  
        return a + b;  
    }  
</script>
```

上述只是定义函数，**函数需要被调用才能执行！**

- 第二步：调用add函数，并接受返回值输出到浏览器上

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

完整代码：

```
11  </body>  
12  <script>  
13  
14      //定义函数  
15      function add(a,b) {  
16          return a + b;  
17      }  
18  
19      //函数调用  
20      var result = add(10,20);  
21      alert(result);  
22  
23  
24  </script>  
25  </html>
```

查看浏览器运行结果：



## 4.2 格式2

第二种可以通过var去定义函数的名字，具体格式如下：

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

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

```
<script>  
    //定义函数-1  
    // function add(a,b){  
    //     return a + b;  
    // }  
  
    //定义函数-2  
    var add = function(a,b){  
        return a + b;  
    }  
  
    //函数调用  
    var result = add(10,20);  
    alert(result);  
  
</script>
```

浏览器弹框效果和上述一致



**注意：JS函数调用可以传递任意个数的参数！**

我们在调用add函数时，再添加2个参数，修改代码如下：

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

浏览器打开，发现没有错误，依然弹出30！

原因：在JavaScript中，函数的调用只需要名称正确即可，参数列表不管的。

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。

## 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：

The screenshot shows the W3School homepage. On the left, there is a sidebar with a tree view of HTML/CSS topics. A red arrow points to the '浏览器脚本' (JavaScript) section, which is highlighted with a red box. Below it, the text '点击这里' (Click here) is written. The main content area features a banner with the text '领先的 Web 技术教程 - 全部免费' (Advanced Web Technology Tutorials - All Free) and '从基础的 HTML 到 CSS, 乃至进阶的 XML、SQL、JS、PHP 和 ASP.NET.' (From basic HTML to CSS, and even advanced XML, SQL, JS, PHP, and ASP.NET). To the right, there is a sidebar titled '参考手册' (Reference Manual) listing various web technologies.

然后进入到如下界面，点击右侧的参考书

The screenshot shows a JavaScript tutorial page from W3School. The left sidebar has a tree view under 'JS 教程'. A red arrow points to the '参考书' (Reference) button in the sidebar, which is highlighted with a green box. The text '点击这里，进入参考书界面' (Click here, enter reference book interface) is written next to it. The main content area displays the 'JavaScript 教程' (JavaScript Tutorial) with sections like 'JavaScript 是属于 HTML 和 Web 的编程语言.' (JavaScript is a programming language for HTML and Web), '编程令计算机完成您需要它们做的工作.' (Programming makes computers do what you need them to), and 'JavaScript 很容易学习.' (JavaScript is easy to learn).

然后进入到如下页面，此页面列举出了JavaScript中的所有对象

The screenshot shows a browser window with the URL <https://www.w3school.com.cn/jsref/index.asp>. The page title is "JavaScript 和 HTML DOM 参考手册". The left sidebar has sections for "HTML/CSS", "Browser Side", "Server Side", "Programming", "XML", "Web Building", and "Reference". Under "Reference", there is a "JS 参考手册" section with a sub-section "JS 参考手册 (类别排序)". This sub-section lists various JavaScript objects: JS Array, JS Boolean, JS Class, JS Date, JS Error, JS Global, JS JSON, JS Math, JS Number, JS RegExp, JS String, JS 运算符, and JS 语句. To the right of the main content area, there is an advertisement for "ZEGO 视频通话SDK" with a "打开" (Open) button. On the far right, there is a sidebar titled "工具箱" (Toolbox) containing links to "参考书" (Reference Books), "实验室" (Laboratory), "小测验" (Quiz), and "必修课" (Mandatory Courses). There is also a "赞助商链接" (Sponsorship Link) section with a logo for "永中".

可以大体分页3大类：

## 第一类：基本对象，我们主要学习Array、JSON和String

### JavaScript 参考手册

这些参考手册描述了所有 JavaScript 对象的属性和方法以及示例。

- [Array](#)
- [Boolean](#)
- [Classe](#)
- [Date](#)
- [Error](#)
- [Global](#)
- [JSON](#)
- [Math](#)
- [Number](#)
- [RegExp](#)
- [String](#)
- [运算符](#)
- [语句](#)

## 第二类：BOM对象，主要是和浏览器相关的几个对象

- Location
- Navigator
- Screen
- Style
- Window

**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中的Array对象

## 5.1 Array对象

### 1) 语法

Array对象用来定义数组。常用语法格式有2种：

方式1：

```
var 变量名 = new Array(元素列表);
```

例如：

```
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：

```
var 变量名 = [元素列表];
```

例如：

```
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：

```
arr[索引] = 值；
```

接下来，我们在VS Code中创建名为 [08-Array对象.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。

```
<script>
    // 第一种数组定义格式
    var arr = new Array(1,2,3,4,5);
    console.log("arr[1]: " + arr[1]);

    // 第二种数组定义格式
    var arr2 = [10,9,8,7];
    console.log("arr2[1]: " + arr2[1]);
</script>
```

浏览器控制台观察的效果如下：输出1和2



## 2) 特点

与Java中不一样的是，JavaScript中数组相当于Java中的集合：

- 数组长度是可以变化的
- 数组中可以存储任意数据类型值（JavaScript是弱类型语言）

接下来我们通过代码来演示上述特点。

```
// 注意事项1：JS数组长度可变  
arr[9] = 10;  
console.log("arr[9]: " + arr[9]);  
  
// 注意事项2：JS数组可以存放任意类型  
arr[5] = "hello";  
arr[6] = 3.14;  
arr[7] = null;  
  
console.log("-----");  
  
//遍历数组  
for(let i = 0; i < arr.length; i++) {  
    console.log("arr[" + i + "]: " + arr[i]);  
}
```

浏览器控制台输出结果如下：

```
arr[9]: 10  
-----  
arr[0]: 1  
arr[1]: 2  
arr[2]: 3  
arr[3]: 4  
arr[4]: 5  
arr[5]: hello  
arr[6]: 3.14  
arr[7]: null  
arr[8]: undefined  
arr[9]: 10
```

### 3) 属性方法

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	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，上述案例已经使用过！

- push()函数

用于向数组的末尾添加元素，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

```
//push: 添加元素到数组末尾  
arr.push('a', 'b');
```

- splice()函数

用来删除数组中的元素，需要2个参数：

- 参数1：表示从哪个索引位置开始删除
- 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

```
//删除元素 从下标2开始删，删除3个元素
arr.splice(2, 3);

console.log("-----");
for(let i = 0; i < arr.length; i++) {
    console.log("arr[" + i + "]: " + arr[i]);
}
```

浏览器执行效果如下：

```
arr[0]: 1
arr[1]: 2
arr[2]: hello ← 3 4 5 已经被删除
arr[3]: 3.14
arr[4]: null
arr[5]: undefined
arr[6]: 10
arr[7]: a
arr[8]: b
```

- **forEach()函数**

顾名思义，`forEach()`方法是用来遍历的，但是遍历时干什么呢？这个函数的参数，需要传递一个函数，该函数接受一个参数，即遍历数组的元素值。修改之前的遍历代码如下：

```
// 数组forEach方法案例
// 1. 常规形式
// function会执行arry.length次
// 每次执行arr[i]传递给参数a
arr.forEach(function(a) {
    console.log(a);
});

//也可以写成下面形式
// 2. 形式2 参数传递函数名
// var outArrayElement = function(a) {
//     console.log(a);
// }
```

```
// arr.forEach(outArrayElement);
```

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：

```
//3.形式3
arr.forEach((a) => {
    console.log(a);
})
```

浏览器控制台输出结果如下：

```
arr[1]: 2
arr2[1]: 9
arr[9]: 10
1
2
hello
3.14
null
10
a
b
»
```

没有赋值的元素不会输出

**注意：forEach只会遍历有值的元素！**

Array数组的完整代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>08-Array对象</title>
</head>
<body>

</body>
```

```
<script>

    // 第一种数组定义格式
    var arr = new Array(1,2,3,4,5);
    console.log("arr[1]: " + arr[1]);


    // 第二种数组定义格式
    var arr2 = [10,9,8,7];
    console.log("arr2[1]: " + arr2[1]);


    // 注意事项1：JS数组长度可变
    arr[9] = 10;
    console.log("arr[9]: " + arr[9]);


    // 注意事项2：JS数组可以存放任意类型
    arr[5] = "hello";
    arr[6] = 3.14;
    arr[7] = null;

    // 遍历数组
    // console.log("-----");
    // for(let i = 0; i < arr.length; i++) {
    //     console.log("arr[" + i + "]: " + arr[i]);
    // }

    // 往数组尾部添加元素
    arr.push('a','b');

    // 删除元素 从下标2开始删，删除3个元素
    arr.splice(2,3);

    // console.log("-----");
    // for(let i = 0; i < arr.length; i++) {
    //     console.log("arr[" + i + "]: " + arr[i]);
    // }

    // 数组forEach方法案例
    // 1. 常规形式
```

```
// arr.forEach(function(a) {  
//     console.log(a);  
// });  
  
// 2.形式2 参数传递函数名  
// var outArrayElement = function(a) {  
//     console.log(a);  
// }  
// arr.forEach(outArrayElement);  
  
//3.简化形式  
arr.forEach(a => console.log(a));  
  
</script>  
</html>
```

## 5.2 String对象

### 1) 语法

String对象创建方式有2种：

方式1：

```
var 变量名 = new String("...");
```

例如：

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

方式2：

```
var 变量名 = "...";
```

例如：

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

按照上述的格式，在VS Code中创建为名 `09-String对象.html` 的文件，编写代码如下：

```
<script>
    //创建字符串对象
    var s1 = new String("hello");
    var s2 = "world";

    console.log(s1);
    console.log(s2);
</script>
```

浏览器控制台输出结果如下：



## 2) 属性方法

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

- length属性

用于返回字符串的长度

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

- charAt()函数

用于返回在指定索引位置的字符，函数参数为索引（从0开始）

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

- 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

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

- trim()函数

用于去除字符串两边的空格

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

- substring()函数

用于截取字符串，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

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

完整代码如下：

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-  
scale=1.0">  
    <title>09-String对象</title>  
</head>  
<body>  
  
</body>  
<script>  
    var s1 = new String("hello");  
    var s2 = "world";  
  
    console.log(s1);  
    console.log(s2);  
  
    //对象.属性  
    console.log("s2.length: " + s2.length);  
  
    console.log("charAt(4): " + s2.charAt(4));  
  
    console.log("r1位置: " + s2.indexOf("r1"));  
  
    var s = " hello world ";  
    console.log("最初: " + s.length);  
    s = s.trim();
```

```
console.log("after trim: " + s.length);

//hello world [0,5)
console.log(s.substring(0,5)); //hello

</script>
</html>
```

浏览器执行效果如图所示：

The screenshot shows the browser's developer tools with the '控制台' (Console) tab selected. The output window displays the following log entries:

```
▶ String { "hello" }
world
s2.length: 5
charAt(4): d
r1位置: 2
最初: 16
after trim: 11
hello
```

## 5.3 自定义对象

在 JavaScript 中自定义对象特别简单，其语法格式如下：

```
var 对象名 = {
    属性名1: 属性值1,
    属性名2: 属性值2,
    ...
    函数名称: function(形参列表){
        函数体实现
    }
    ...
};
```

我们可以通过如下语法调用属性：

```
对象名.属性名
```

通过如下语法调用函数：

```
对象名.函数名()
```

接下来，我们再VS Code中创建名为 [10-自定义对象.html](#) 的文件演示自定义对象

```
<script>
    //自定义对象
    var user = {
        name: "Tom",
        age: 10,
        gender: "male",
        eat: function(){
            console.log("用膳~");
        }
    }

    console.log(user.name);
    user.eat();
<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：

```
var user = {
    name: "Tom",
    age: 10,
    gender: "male",
    // eat: function(){
    //     console.log("用膳~");
    // }
    eat(){
        console.log("用膳~");
    }
}
```

## 5.4 JSON对象

JSON：JavaScript Object Notation，JavaScript对象标记法。

JSON (JavaScript Object Notation) 是一种轻量级的数据交换格式。它以简洁和易于阅读的方式表示结构化数据，并且易于解析和生成。JSON最初是为JavaScript语言而设计的，但现在已经成为一种通用的数据格式，被广泛应用于各种编程语言和平台。

### Json在线学习资料

#### 1) JSON特点

1. 简洁性：语法简洁明了，易于理解和编写
2. 可读性：结构清晰，易于阅读和调试
3. 可扩展性：支持嵌套和复杂的数据结构，可以表示各种类型的数据
4. 平台无关性：JSON是一种通用的数据格式，可以在不同的编程语言和平台之间进行数据交换

#### 2) JSON基本语法

JSON是通过JavaScript标记法书写的文本。其格式如下：

```
{  
    "key": value,  
    "key": value,  
    "key": value  
}
```

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

**数据类型：**'string', 'number', null, true, false, '{}', '[]'。

注意事项：

- 数据在**名称/值**对中
- 数据由逗号，分隔
- 使用斜杆 \ 来转义字符
- 大括号 {} 保存对象
- 中括号 [] 保存数组，数组可以包含多个对象

### 3) JSON 取值

- 数字（整数或浮点数）

```
{ "age":30 }
```

- 字符串（在双引号中）

```
{ "name":"Bill" }
```

- 逻辑值（true 或 false）

```
{ "isVip":true }
```

- 数组（在中括号中）

注意：数据由逗号分隔，多个键值对由逗号分隔

```
{  
    "employees": [ "Bill", "Steve", "David" ]  
}
```

- 对象（在大括号中）

```
{  
    "employee": {  
        "name": "Bill Gates",  
        "age": 62,  
        "city": "Seattle"  
    }  
}
```

- null

```
{ "middlename": null }
```

## 4) JSON解析

我们可以直接百度搜索"JSON在线解析"，比如<https://www.sojson.com/>，然后编写内容如下：

```
{  
    "name": "李传播"  
}
```

The screenshot shows the SoJson JSON online parser interface. At the top, there is a navigation bar with tabs: 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, and JSON压缩成一个。 Below the tabs, there is a code editor area with three lines of JSON:

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

A red box highlights the second line of code, "name": "李传播". To the right of the code editor, there is a note: "1. 编写内容, key需要加双引号". Below the code editor, there is a note: "2.点击校验格式". In the bottom right corner of the code editor area, there is a note: "② 这里可以收缩编辑框" with a collapse icon.

At the bottom of the interface, there is a toolbar with buttons: 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, and 保存本地. The "校验 / 格式化" button is highlighted with a red border. To the right of the toolbar, there is a green bar with the text: "老铁, 这个JSON格式, 没毛病。" and "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：

The screenshot shows a JSON validation interface. At the top, there are tabs: 'JSON解析' (selected), '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', and 'JSON压缩成一行'. Below the tabs is a code editor area containing the following JSON:

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

A red box highlights the double quotes around 'name'. Below the code, the text '1.换成单引号' (Change to single quotes) is displayed. To the right, there's a note '(2) 这里可以收缩编辑框' (You can collapse the editor here) with a collapse icon. At the bottom of the editor area, there's a message: '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' (① Press and hold the left mouse button to drag and resize >> << Next time you open this tool, the size will be remembered). Below the editor is a toolbar with buttons: '校验 / 格式化' (selected), '压缩', '转义', '去转义', 'Unicode转中文', '中文转Unicode', '复制结果' (highlighted in blue), '清空', and '保存本地'.

**2.校验格式失败**

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 '')

**注意：JSON字符串中key必须使用双引号修饰，value如果是字符串类似，也必须使用双引号修饰！**

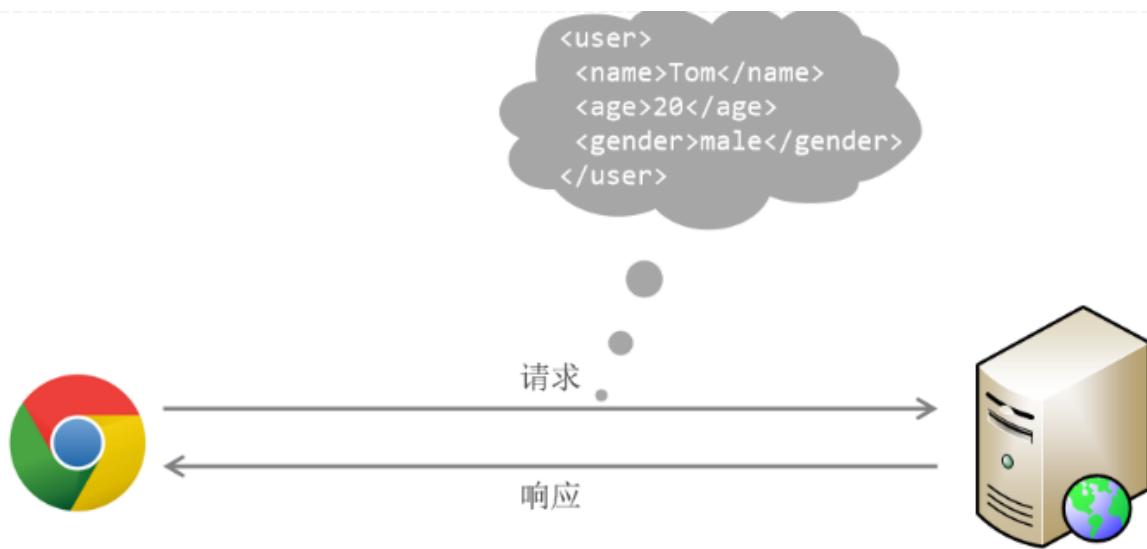
## 5) JSON应用场景

思考：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？

答案：经常用来作为前后台交互的数据载体！

思考：前后台交互时，在学习JS之前如果我们需要传输数据（Java对象），我们该怎么去实现呢？

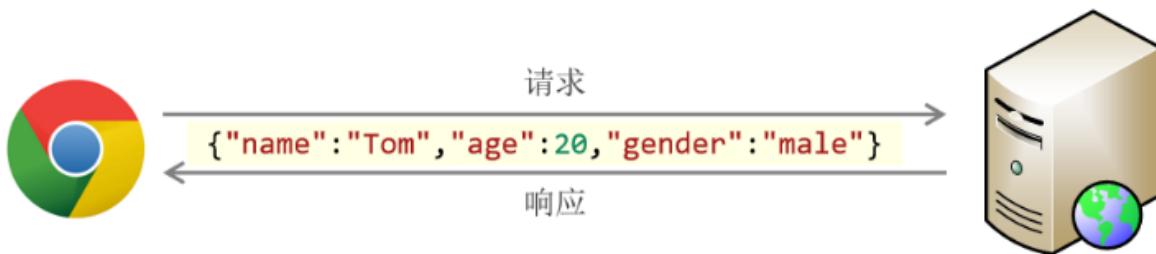
答案：可以使用xml格式传输，其能清晰的描述Java中需要传递给前端的Java对象。



但是xml数据在如下问题：

- 标签需要编写双份，占用带宽，浪费资源
- 解析繁琐

最终方案：使用JSON来传输数据，如下图所示：



接下来我们通过代码来演示JSON对象：

```

// 定义json对象
var jsonObj = {
  "name": "tom",
  "age": 21,
  "address": {
    "province": "shanxi",
    "city": "taiyuan",
    "street": "坞城中路"
  }
};

console.log(jsonObj.name);
console.log(jsonObj.age);

```

```
console.log(jsonObj.address);

// 定义json字符串
var jsonstr = '{"name":"Tom", "age":18, "addr":["北京", "上海", "西安"]}';
alert(jsonstr.name);
```

浏览器输出结果如下：



## 6) JSON转换

先思考上述案例为什么输出"undefined"？

**因为上述案例中 `jsonstr` 是一个JSON字符串，不是JSON对象。**

如果想得到期望的结果，我们需要借助转换函数将JSON字符串转换成JSON对象。

代码如下：

```
//json字符串 --> json对象
var obj = JSON.parse(jsonstr);
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对象转换成JSON字符串。

代码如下：

```
//json对象 --> json字符串
var jsonStr2 = JSON.stringify(obj);
alert(jsonStr2);
```

浏览器输出结果如图所示：



完整代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>10-自定义对象</title>
</head>
<body>
```

```
</body>
<script>
    //自定义对象
    var user = {
        name: "Tom",
        age: 10,
        gender: "male",
        eat: function(){
            console.log("用膳~");
        }
    }

    console.log(user.name);
    user.eat();

    console.log("-----");

    // 定义json 对象
    var jsonObj = {
        "name": "tom",
        "age": 21,
        "address": {
            "province": "shanxi",
            "city": "taiyuan",
            "street": "坞城中路"
        }
    };
    console.log(jsonObj.name);
    console.log(jsonObj.age);
    console.log(jsonObj.address);

    //定义JSON字符串
    var jsonstr = '{"name":"Tom", "age":18, "addr":["北京", "上海", "西安"]}';
    alert(jsonstr.name);

    //json字符串 --> json对象
```

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

//json对象 --> json字符串
var jsonStr2 = JSON.stringify(obj);
alert(jsonStr2);

var jsonStr3 = JSON.stringify(jsonObj);
console.log("jsonObj 字符串形式：" + jsonStr3)

</script>

</html>
```

## 补充内容：JSON表示对象常见用法

### Java普通对象：

```
public class Student {
    private String name;
    private int age;

    //...其他省略
}

Student stu = new Student("zs", 20);
```

### Json格式：

```
{
    "name": "zs",
    "age": 20
}
```

### Java复杂对象：

```
public class Person {  
    private String name;  
    private int age;  
    private List<String> addr;  
  
    // 省略...  
}
```

Json格式:

```
{  
    "name": "Tom",  
    "age": 18,  
    "addr": ["北京", "上海", "西安"]  
}
```

Java复合对象:

```
public class Address {  
    private String province;  
    private String city;  
}  
  
public class StudentWithAddress {  
    private String name;  
    private int age;  
    private Address address;  
  
    // 省略...  
}  
  
Address address = new Address("jiangsu", "suzhou");  
StudentWithAddress stu = new StudentWithAddress("tom", 20,  
address);
```

Json格式:

```
{  
    "name": "tom",  
    "age": 20,  
    "address": {  
        "province": "jiangsu",  
        "city": "suzhou"  
    }  
}
```

后续的web课程中我们还会进一步学习JSON。

## 5.5 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**。

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某一部分，就可以通过指定的BOM对象去操作，借助其相关属性或者函数来完成。

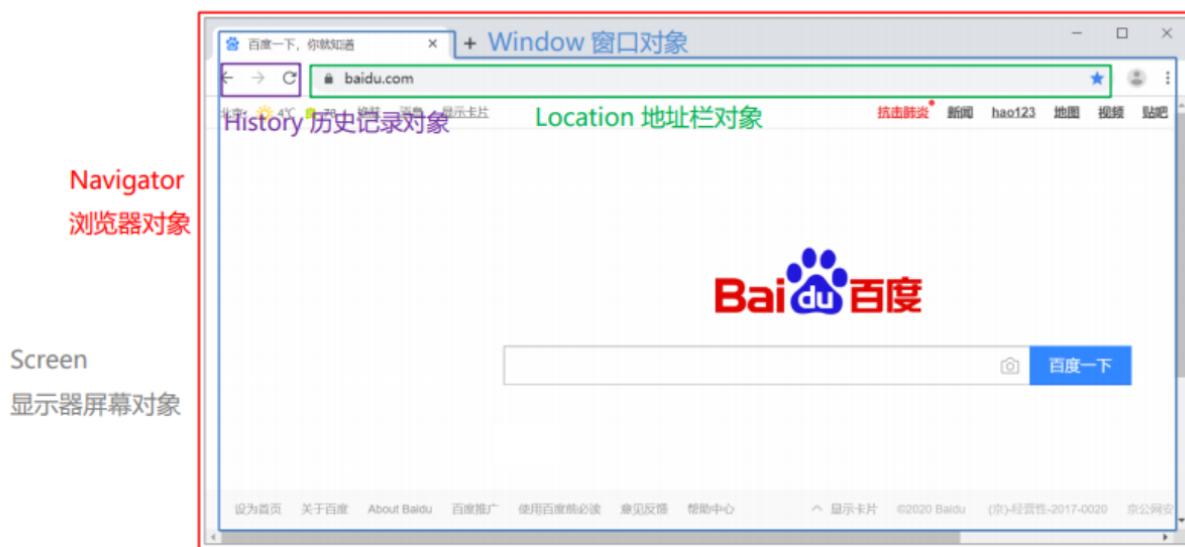
例如：我们想要将浏览器的url地址改为 `http://www.baidu.com`，就可以通过BOM中提供的 `location` 对象 的 `href` 属性 来完成。

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

BOM中提供了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	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**Window对象**、**Location对象**这2个。

## Window对象

### 1) window概述

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：

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

`window.` 可以省略，简写为：

```
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。

## 2) `window`属性和方法

`window`常用属性：

属性	描述
<code>history</code>	用于获取 <code>history</code> 对象
<code>location</code>	用于获取 <code>location</code> 对象
<code>Navigator</code>	用于获取 <code>Navigator</code> 对象
<code>Screen</code>	用于获取 <code>Screen</code> 对象

比如我们要使用`location`对象，可通过代码 `window.location` 或简写 `location` 即可使用！

`window`常用函数：

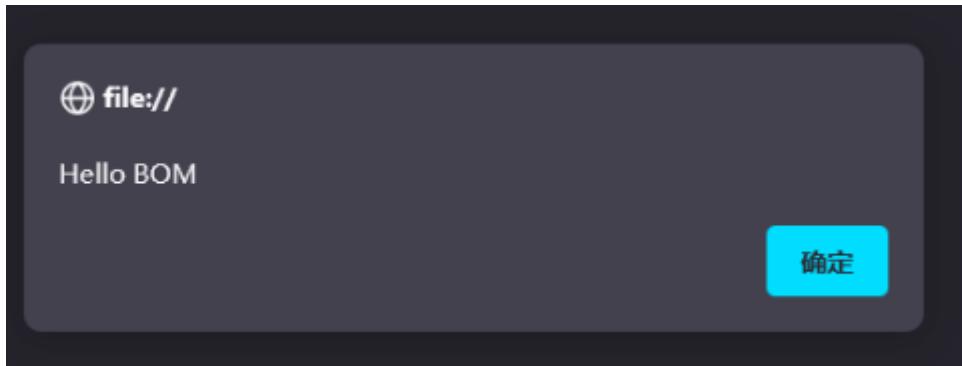
函数	描述
<code>alert()</code>	显示带有一段消息和一个确认按钮的警告框。
<code>comfirm()</code>	显示带有一段消息以及确认按钮和取消按钮的对话框。
<code>setInterval()</code>	按照指定的周期（以毫秒计）来调用函数或计算表达式。
<code>setTimeout()</code>	在指定的毫秒数后调用函数或计算表达式。

接下来，我们通过VS Code中创建名为 `11-window属性函数.html` 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

```
<script>
    //window对象是全局对象，调用window对象属性和方法时可省略window.
    window.alert("Hello BOM");
    alert("Hello BOM Window");
</script>
```

浏览器打开，依次弹框，此处只截图一张

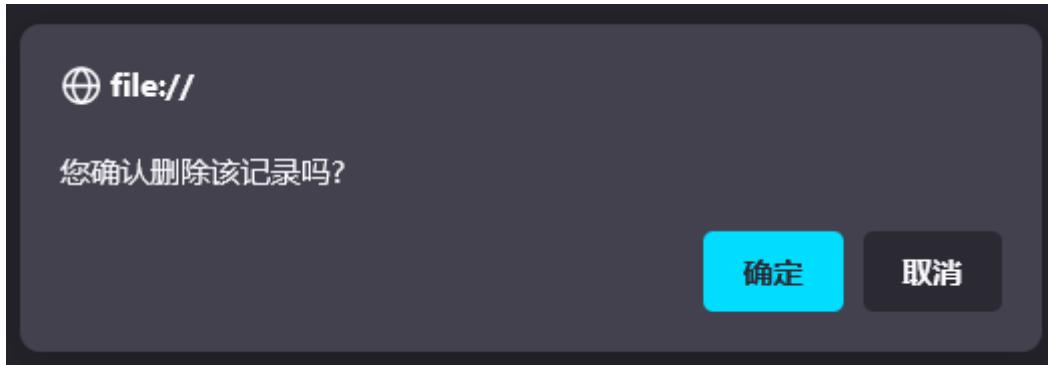


- **confirm()函数：弹出对话框**，其中包含消息以及确定和取消按钮

添加如下代码：

```
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？

该函数返回值为true时，表示用户点击确认按钮；返回false表示用户点击取消按钮。测试代码如下：

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

- `setInterval(function, milliseconds)`: 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：

`function`函数，需要周期性执行的功能代码

`milliseconds`毫秒值：间隔时间

注释掉之前的代码，添加代码如下：

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

The screenshot shows the browser's developer tools with the 'Console' tab selected. The output pane displays the following log entries:

```
定时器执行了1次  
定时器执行了2次  
定时器执行了3次  
定时器执行了4次  
定时器执行了5次
```

- `setTimeout(function, milliseconds)`：定时器，只会在一段时间后执行一次。

参数和上述`setInterval`一致

测试代码如下：

```
//setTimeout定时器：过3s执行1次(有且只有1次)，输出i值
var i = 1;
window.setTimeout(function(){
    console.log("setTimeout定时器执行：" + i);
    i++;
}, 3000);
```

浏览器运行效果：

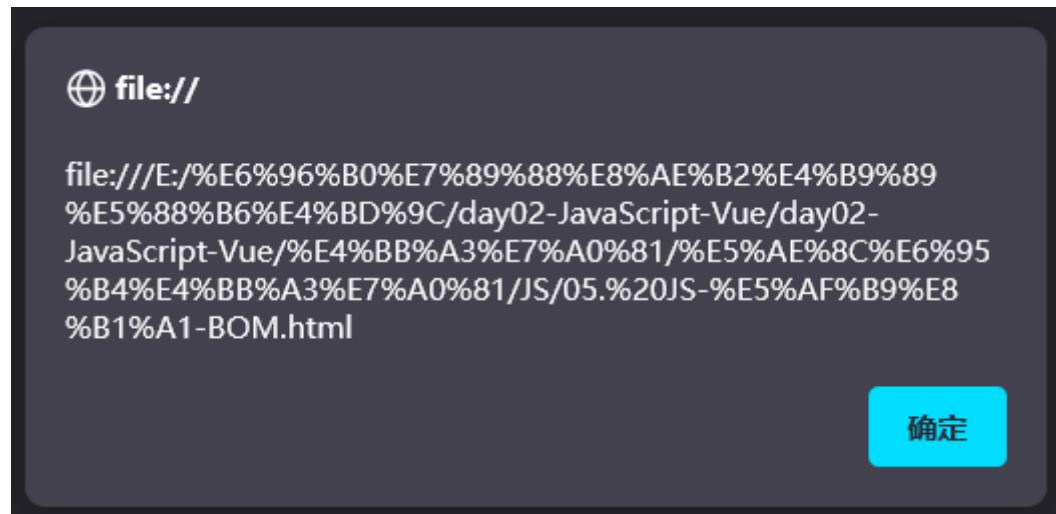


## Location对象

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：

```
//获取浏览器地址栏信息
alert(location.href);
//设置浏览器地址栏信息
location.href = "http://www.briup.com";
```

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：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>11-window属性及方法</title>
</head>
<body>

</body>
<script>
    window.alert("警告窗");

    //window.confirm("信息提示框： 提示信息 确定|取消按钮");
    var flag = window.confirm("您确认要删除吗？");
    window.console.log("flag: " + flag);

    //setInterval定时器： 每隔2s执行1次， 输出i值
    // var i = 1;
    // window.setInterval(function() {
    //     console.log("setInterval定时器执行： " + i);
    //     i++;
    // },2000);

    //setTimeout定时器： 过3s执行1次(有且只有1次)， 输出i值
    var i = 1;
    window.setTimeout(function(){
        console.log("setTimeout定时器执行： " + i);
        i++;
    },3000);

```

```
//location属性案例  
var url = window.location.href;  
console.log("地址栏url: " + url);  
  
//修改url 注意: 修改完以后浏览器会自动跳转  
location.href = "http://www.briup.com";  
  
</script>  
</html>
```

## 5.6 DOM对象

### DOM概述

DOM：Document Object Model **文档对象模型**，即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 解析时就接触过。Java语言中需要写代码解析 XML 文档，而 HTML 文档则是由浏览器解析。

#### DOM对象类别：

- 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



## DOM主要作用：

- 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

具体我们可以查看下面代码 [12-DOM演示 .html](#) 来体会DOM的效果。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>12-DOM演示</title>
</head>

<body>
    <div style="font-size: 30px; text-align: center;" id="tb1">课
程表</div>
    <table width="800px" border="1" cellspacing="0"
align="center">
        <tr>
            <th>学号</th>
            <th>姓名</th>
            <th>分数</th>
            <th>评语</th>

```

```
</tr>
<tr align="center" class="data">
    <td>001</td>
    <td>张三</td>
    <td>90</td>
    <td>很优秀</td>
</tr>
<tr align="center" class="data">
    <td>002</td>
    <td>李四</td>
    <td>92</td>
    <td>优秀</td>
</tr>
<tr align="center" class="data">
    <td>003</td>
    <td>王五</td>
    <td>83</td>
    <td>很努力, 不错</td>
</tr>
<tr align="center" class="data">
    <td>004</td>
    <td>赵六</td>
    <td>98</td>
    <td>666</td>
</tr>
</table>
<br>
<div style="text-align: center;">
    <input id="b1" type="button" value="改变标题内容"
onclick="fn1()">
    <input id="b2" type="button" value="改变标题颜色"
onclick="fn2()">
    <input id="b3" type="button" value="删除最后一个"
onclick="fn3()">
</div>
</body>

<script>
```

```
function fn1(){
    document.getElementById('tb1').innerHTML="学员成绩表";
}

function fn2(){
    document.getElementById('tb1').style="font-size: 30px;
text-align: center; color: red;";
}

function fn3(){
    var trs = document.getElementsByClassName('data');
    trs[trs.length-1].remove();
}
</script>

</html>
```

## 对象获取

DOM的作用是通过修改HTML标签的内容和样式来实现页面的各种动态效果。

学习DOM，核心有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

### 1) 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对象获取Element元素对象的API如下表：

函数	描述
document.getElementById()	根据id属性值获取，返回单个Element对象
document.getElementsByTagName()	根据标签名称获取，返回Element对象数组
document.getElementsByName()	根据name属性值获取，返回Element对象数组
document.getElementsByClassName()	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 **13-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：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>13-DOM获取元素</title>
</head>

<body>
     <br><br>

    <div class="cls">杰普软件</div> <br>
    <div class="cls">Java程序员</div> <br>

    <input type="checkbox" name="hobby"> 电影
    <input type="checkbox" name="hobby"> 旅游
    <input type="checkbox" name="hobby"> 游戏
</body>
```

```
</html>
```

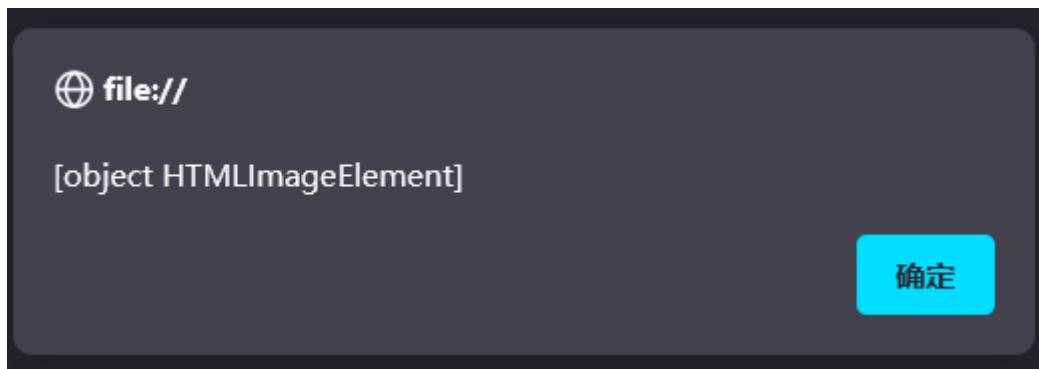
- `document.getElementById()`: 根据标签的id属性获取标签对象，id是唯一的，所以获取到是单个标签对象。

添加如下代码：

```
<script>
    //1. 获取Element元素

    //1.1 获取元素-根据ID获取
    var img = document.getElementById('h1');
    alert(img);
</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：

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

浏览器输出2次如下所示的弹框

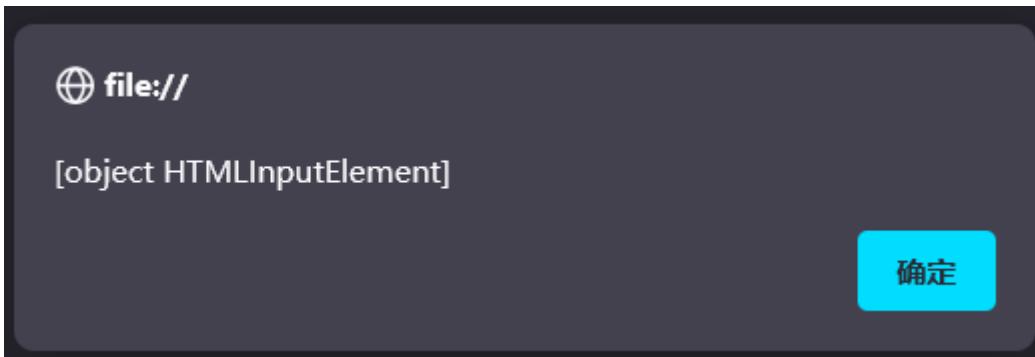


- `document.getElementsByName()` : 根据标签的name属性值获取标签对象， name属性值可以重复， 所以返回值是一个数组。

添加如下代码：

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

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：

#### //1.4 获取元素-根据class属性获取

```
var divs = document.getElementsByClassName('cls');
for (let i = 0; i < divs.length; i++) {
    alert(divs[i]);
}
```

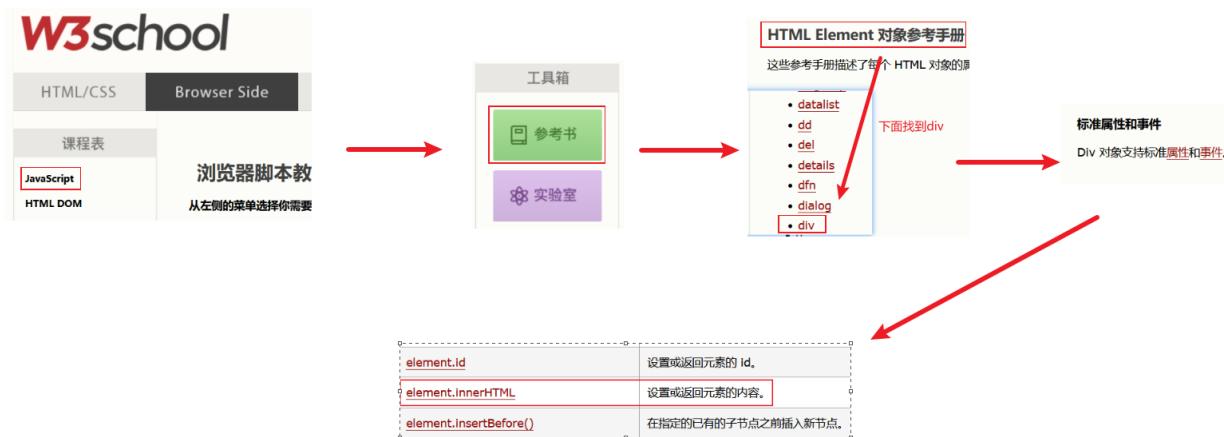
浏览器会弹框2次，都是div标签对象



## 2) 属性操作

获取到标签后，我们如何操作标签的属性呢？通过查询文档资料，我们发现通过div标签对象的innerHTML属性可以修改标签的内容。

查阅过程如下图所示：



此时我们想把页面中的Java程序员替换成九九六狂人，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：

```
var divs = document.getElementsByClassName('cls');
var div1 = divs[1];

div1.innerHTML = "九九六狂人";
```

浏览器刷新页面，展示效果如下图所示：

杰普软件

九九六狂人

完整代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>13-DOM获取元素</title>
</head>

<body>
     <br><br>

    <div class="cls">杰普软件</div> <br>
    <div class="cls">Java程序员</div> <br>

    <input type="checkbox" name="hobby"> 电影
    <input type="checkbox" name="hobby"> 旅游
    <input type="checkbox" name="hobby"> 游戏
</body>
```

```
<script>
    //1. 获取Element元素

    //1.1 获取元素-根据ID获取
    // var img = document.getElementById('h1');
    // alert(img);

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

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

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

    // 2. 修改标签内容
    var divs = document.getElementsByClassName('cls');
    var div1 = divs[1];
    div1.innerHTML = "九九六狂人";

</script>
</html>
```

## 5.7 综合案例

### 1) 需求说明

接下来通过案例来加强对DOM知识的掌握。

案例需求：

- 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：



杰普软件

Java程序员

电影  旅游  游戏



杰普软件 very good

Java程序员 very good

电影  旅游  游戏

### 2) 资料准备

在JS目录下创建img文件夹，然后拷贝2张资源图片，最终效果如下：



在VS Code中创建名为 **14-DOM课堂练习.html** 的文件，然后准备如下代码：

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>14-DOM课堂练习</title>
</head>

<body>
     <br><br>

    <div class="cls">杰普软件</div> <br>
    <div class="cls">Java程序员</div> <br>

    <input type="checkbox" name="hobby"> 电影
    <input type="checkbox" name="hobby"> 旅游
    <input type="checkbox" name="hobby"> 游戏
</body>

<script>
</script>
</html>
```

浏览器打开此时效果如图所示：



杰普软件

Java程序员

电影  旅游  游戏

### 3) 点亮灯泡

#### 功能分析：

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。

#### 操作步骤：

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

浏览器打开，效果如图所示：



杰普软件

Java程序员

电影  旅游  游戏

## 4) very good

### 需求分析：

我们需要在原有内容后面追加红色的very good。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？可以通过修改<font>标签的属性来完成。

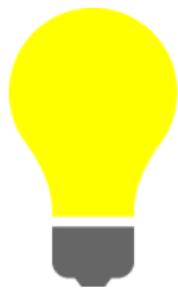
### 实现步骤：

- 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：

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

浏览器打开效果如图所示：



杰普软件 very good

Java程序员 very good

电影  旅游  游戏

## 5) 复选框选中

### 需求分析：

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

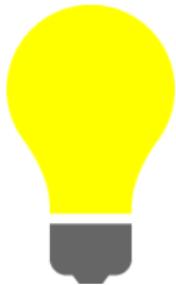
### 实现步骤：

- 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checked状态

### 代码实现：

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

浏览器刷新，效果如图所示：



杰普软件 very good

Java程序员 very good

电影  旅游  游戏

## 6) 完整代码

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>14-DOM课堂练习</title>
</head>

<body>
     <br><br>

    <div class="cls">杰普软件</div> <br>
    <div class="cls">Java程序员</div> <br>

    <input type="checkbox" name="hobby"> 电影
    <input type="checkbox" name="hobby"> 旅游
    <input type="checkbox" name="hobby"> 游戏
</body>
```

```
<script>

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

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

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

## 6 JavaScript事件

---

### 6.1 事件理解

如下图所示的百度注册页面，当我们输入完用户名，百度会自动提示该用户名是否存在。

思考：百度是怎么知道我们用户名输入完了呢？

答案：借助JavaScript事件实现。



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
- .....

事件监听：

程序员给事件绑定函数，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：上述百度注册页面中，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：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，对于JavaScript事件的学习主要围绕2点：

- 常用事件
- 事件绑定

## 6.2 事件绑定

新建 15-JS事件绑定.html，来学习测试事件绑定。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>15-事件绑定</title>
</head>

<body>
    <input type="button" id="btn1" value="事件绑定1">
    <input type="button" id="btn2" value="事件绑定2">
</body>

<script>

</script>
</html>
```

JavaScript对于事件的绑定提供了2种方式：

### 1) 绑定方式1：通过html标签中的事件属性进行绑定

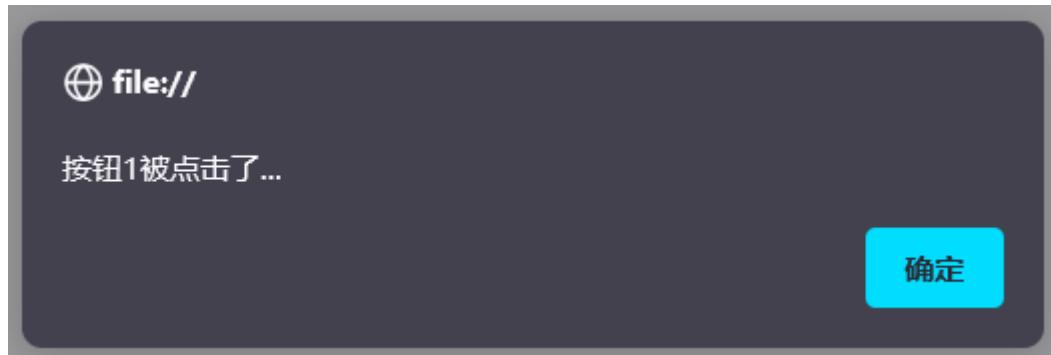
上述页面中button按钮可以借助onclick属性实现事件绑定， onclick属性值指向一个函数。

```
<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：

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

浏览器打开，然后点击按钮，弹框如下：



## 2) 绑定方式2：通过DOM中Element元素的事件属性进行绑定

根据我们学习的DOM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。

我们可以先通过id属性获取btn2按钮对象，然后操作对象的onclick属性来绑定事件，代码如下：

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

浏览器刷新页面，点击第二个按钮，弹框如下：



**注意事项：事件绑定的函数，只有在事件被触发时，函数才会被调用。**

完整代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>15-事件绑定</title>
</head>

<body>
    <input type="button" id="btn1" value="事件绑定1"
    onclick="on()>
        <input type="button" id="btn2" value="事件绑定2">
    </body>

<script>
    function on(){
        alert("按钮1被点击了...");

    }

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

</script>
</html>
```

## 6.3 常见事件

上面案例中用到了 `onclick` 事件属性，下面给大家列举一些比较常用的事件属性：

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

新增 `16-常见事件.html`，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：

源码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>16-常见事件</title>
</head>

<body onload="load()">
```

```
<form action="" style="text-align: center;"  
onsubmit="subfn()>  
    <input type="text" name="username" onblur="bfn()"  
    onfocus="ffn()" onkeydown="kfn()>  
  
    <input id="b1" type="submit" value="提交">  
  
    <input id="btn1" type="button" value="单击事件"  
    onclick="fn1()>  
    </form>  
  
<br><br><br>  
  
<table width="800px" border="1" cellspacing="0"  
align="center" onmouseover="over()" onmouseout="out()>  
    <tr>  
        <th>学号</th>  
        <th>姓名</th>  
        <th>分数</th>  
        <th>评语</th>  
    </tr>  
    <tr align="center">  
        <td>001</td>  
        <td>张三</td>  
        <td>90</td>  
        <td>很优秀</td>  
    </tr>  
    <tr align="center">  
        <td>002</td>  
        <td>李四</td>  
        <td>92</td>  
        <td>优秀</td>  
    </tr>  
    </table>  
  
</body>
```

```
<script>

    //onload : 页面/元素加载完成后触发
    function load(){
        console.log("页面加载完成...")
    }

    //onclick: 鼠标点击事件
    function fn1(){
        console.log("我被点击了...");
    }

    //onblur: 失去焦点事件
    function bfn(){
        console.log("失去焦点...");
    }

    //onfocus: 元素获得焦点
    function ffn(){
        console.log("获得焦点...");
    }

    //onkeydown: 某个键盘的键被按下
    function kfn(){
        console.log("键盘被按下了...");
    }

    //onmouseover: 鼠标移动到元素之上
    function over(){
        console.log("鼠标移入了...")
    }

    //onmouseout: 鼠标移出某元素
    function out(){
        console.log("鼠标移出了...")
    }

    //onsubmit: 提交表单事件
    function subfn(){
```

```

        alert("表单被提交了...");

    }

</script>
</html>

```

- 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，同学们通过提供好的代码去演示，自行体验。

## 6.4 综合案例

## 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：



## 2) 资料准备

在VS Code中创建名为 **17-事件案例.html** 的文件，提前准备如下代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>17-事件案例</title>
</head>
<body>
```

```
 <br>

<input type="button" value="点亮" >
<input type="button" value="熄灭" >

<br> <br>

<input type="text" id="name" value="BRIUP">
<br> <br>

<input type="checkbox" name="hobby"> 电影
<input type="checkbox" name="hobby"> 旅游
<input type="checkbox" name="hobby"> 游戏
<br>

<input type="button" value="全选" >
<input type="button" value="反选" >

</body>

</html>
```

浏览器打开如图所示：



电影  旅游  游戏

### 3) 操作灯泡

#### 功能分析：

点击按钮时触发按钮单击事件。不管是点亮还是熄灭，借助图片变化即可实现功能。故而我们需要先获取标签图片标签对象。

#### 实现步骤：

- 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()
- 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

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

##### on()和off()函数实现

```
//1. 点击“点亮”按钮，点亮灯泡；点击“熄灭”按钮，熄灭灯泡；--
onclick
function on(){
    //a. 获取img元素对象
    var img = document.getElementById("light");
    //b. 设置src属性
    img.src = "img/on.gif";
}

function off(){
```

```
//a. 获取img元素对象  
var img = document.getElementById("light");  
//b. 设置src属性  
img.src = "img/off.gif";  
}
```

## 4) 大小写切换

输入框鼠标聚焦后，展示小写；鼠标离焦后，展示大写。

### 功能分析：

先给input标签绑定onfocus和onblur事件；然后去实现事件处理函数。函数中我们要切换大小写，那么需要先获取输入框原始内容。查询资料可知，使用input标签对象的value属性，可以获取输入框内容。往输入框中重新填入内容，也是通过设置value属性实现。

### 实现步骤：

- 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：

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

lower()和upper()函数实现：

```
//2. 输入框聚焦后，展示小写；  
//    输入框离焦后，展示大写； -- onfocus, onblur  
function lower(){//小写  
    //a. 获取输入框元素对象  
    var input = document.getElementById("name");  
  
    //b. 将值转为小写  
    input.value = input.value.toLowerCase();  
}  
  
function upper(){//大写  
    //a. 获取输入框元素对象  
    var input = document.getElementById("name");  
  
    //b. 将值转为大写  
    input.value = input.value.toUpperCase();  
}
```

## 5) 全选反选

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选中状态取反。

**功能分析：**

点击按钮完成功能，我们需要给2个按钮绑定单击事件；实现"全选"功能，我们需要在事件处理函数中获取所有的复选框，然后逐个对象设置checked属性为true；实现"反选"功能，我们获得所有复选框，然后逐个元素观察其checked属性值，如果为true则设置为false，如果为false则设置为true。

## 实现步骤：

- 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，根据其checked属性值修改其值：如果为true则设置为false，如果为false则设置为true

## 代码实现：

事件绑定：

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

checkAll()和reverse()函数

```
//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选中状态取反；
function checkAll(){
    //a. 获取所有复选框元素对象
    var hobbies = document.getElementsByName("hobby");

    //b. 设置选中状态
    for (let i = 0; i < hobbies.length; i++) {
        const element = hobbies[i];
        element.checked = true;
    }
}

function reverse(){
    //a. 获取所有复选框元素对象
    var hobbies = document.getElementsByName("hobby");
```

```

//b. 设置未选中状态
for (let i = 0; i < hobbys.length; i++) {
    const element = hobbys[i];
    if(element.checked)
        element.checked = false;
    else
        element.checked = true;
}
}

```

## 6) 完整代码

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>17-事件案例</title>
</head>
<body>

     <br>

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

    <br> <br>

    <input type="text" id="name" value="BRIUP" onfocus="lower()" onblur="upper()">
    <br> <br>

    <input type="checkbox" name="hobby"> 电影

```

```

<input type="checkbox" name="hobby"> 旅游
<input type="checkbox" name="hobby"> 游戏
<br>

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

</body>

<script>
    //1. 点击 "点亮" 按钮，点亮灯泡；
    // 点击 "熄灭" 按钮，熄灭灯泡； -- onclick
    function on(){
        //a. 获取img元素对象
        var img = document.getElementById("light");
        //b. 设置src属性
        img.src = "img/on.gif";
    }

    function off(){
        //a. 获取img元素对象
        var img = document.getElementById("light");
        //b. 设置src属性
        img.src = "img/off.gif";
    }

    //2. 输入框聚焦后，展示小写；
    // 输入框离焦后，展示大写； -- onfocus, onblur
    function lower(){ //小写
        //a. 获取输入框元素对象
        var input = document.getElementById("name");

        //b. 将值转为小写
        input.value = input.value.toLowerCase();
    }

    function upper(){ //大写
        //a. 获取输入框元素对象

```

```
var input = document.getElementById("name");

//b. 将值转为大写
input.value = input.value.toUpperCase();
}

//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选中状态取反;
function checkAll(){
    //a. 获取所有复选框元素对象
    var hobbys = document.getElementsByName("hobby");

    //b. 设置选中状态
    for (let i = 0; i < hobbys.length; i++) {
        const element = hobbys[i];
        element.checked = true;
    }
}

function reverse(){
    //a. 获取所有复选框元素对象
    var hobbys = document.getElementsByName("hobby");

    //b. 反转其选中状态
    for (let i = 0; i < hobbys.length; i++) {
        const element = hobbys[i];
        if(element.checked)
            element.checked = false;
        else
            element.checked = true;
    }
}
</script>
</html>
```