JavaScript基础

JavaScript的作用

为什么要使用JavaScript

HTML网页运行在浏览器端，与用户没有交互功能。用户访问网页的时候只能看，如果网页没有程序员去更新，永远是一成不变的。JavaScript就是可以让程序运行在网页上，提高客户访问网页时体验。可以让网页有交互的功能。

网页中各技术的作用

技术	作用
HTML	制作网页的结构
CSS	对网页进行美化操作
JavaScript	提高用户体验，让网页具体交互的功能。运行在浏览器端，由浏览器去解析脚本的运行。

JavaScript的基本概述

在1995年时，由Netscape公司在网景导航者浏览器上首次设计实现而成。因为Netscape与Sun合作，Netscape管理层希望它外观看起来像Java，因此取名为JavaScript。

布兰登.艾奇 10天开发js 原名LiveScript

JavaScript与Java的区别

特点	Java	JavaScript
面向对象	完全面向对象语言	基于对象，不完全面向对象
运行方式	编译型的语言，生成中间的文件class	解释型语言，js引擎，直接在浏览器中执行
跨平台	Java可以运行在不同系统上，安装虚拟机	运行浏览器上，只要操作系统有浏览器就可以执行
数据类型	强类型的语言 String str = 35; //错误	弱类型语言，同一个变量可以赋值为不同的数据类型 var str = "abc"; str = 35; //对

JavaScript的基础语法

JavaScript语言的组成

组成部分	作用
ECMA Script	这是一种脚本语言规范，构成了JavaScript的核心语法基础。 如：定义变量，控制结构
BOM	Browser Object Model 浏览器对象模型，操作浏览器中的各种对象。如：window
DOM	Document Object Model 文档对象模型，操作网页中标签元素。如：input, span

JavaScript的存放位置

1. 放在HTML文件中的任意位置，建议放在body或head中
2. 单独的js文件存在，使用的时候需要导入

script标签的说明：

1. <script>中的src属性和type属性：

<script type="text/javascript" src="js/out.js"></script>
type：可选，指定脚本的类型
src：导入外部文件地址

2. <script>标签个数：在一个网页可以同时出现多个标签，每个标签会依次执行。
3. 每行JS代码以分号结束，如果一条语句一行代码，分号可以省略。

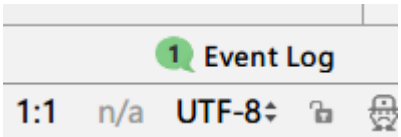
- HTML代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

<script type="text/javascript">
    alter("我是js");
</script>

<!--导入外部js文件-->
<script type="text/javascript" src="js/out.js"></script>

</body>
</html>
```

- 外部的JS代码：

```
document.write("我是外部js文件"+"<br/>")
```

- 注：保存JS指定为UTF-8的编码，与网页的编码一致。方法：点右下角的位置：



JavaScript的注释

语言	注释语法
HTML	<!-- 注释 -->
CSS	/* 注释 */
JavaScript	// 单行注释 :可以嵌套　　/* 多行注释 */：不可以嵌套

变量

什么是变量

变量是内存中一块小空间，我们把数据存放在内存中，他就会有一个地址值，地址值是16进制的，非常难记住，所以我们就给这个小空间起了一个名字，这个名字就叫做变量。

怎么定义一个变量

声明一个变量 var 变量名；

var name;

初始化一个变量 var 变量名=值；

var name="小东"

var a="小东"

var !num=2;

var firstName;

变量定义的特点

1. var关键字可以省略，建议加上
2. 变量名要有意义
3. 以字母、$、_开头
4. 变量名一般小写，多个单词使用小驼峰式
5. 不能使用js中的关键字做变量名
6. 变量可以重复定义：var a = 5; var a="abc"，后面的会覆盖前面的

7. 在Java大括号变量的作用范围，在JS中不受这个限制。

案列：交换两个变量的值

方式一：借助第三方变量

```
<script type="text/javascript">
    var num1=10;
    var num2=20;

    var temp=num1;
    num1=num2;
    num2=temp;
    console.log(num1);
    console.log(num2);
</script>
```

方式二：总和交换，一般适用于数字的交换

```
<script type="text/javascript">
    var num1=10;
    var num2=20;

    num1=num1+num2;//30
    num2=num1-num2;//10
    num1=num1-num2;//20
    console.log(num1);
    console.log(num2);
</script>
```

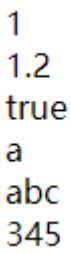
数据类型

六种数据类型：

关键字	说明
number	数值型：包括整数和浮点数/NaN not a number
boolean	布尔类型：true / false
string	字符串：包含字符和字符串。可以使用双引号或单引号 "字符串" '字符串'
null	是object对象类型，指这个对象没有值 var a=null;
undefined	变量未赋值或者函数没有明确的返回值
object	对象类型：JS内置对象或自定义对象

案例：输出不同类型的数据

- 案例需求：分别输出数值、浮点、布尔、字符串类型JS的变量
- 案例效果：



- 案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

<script type="text/javascript">
    {//不能限制变量的作用范围
        //var可以省略不写，推荐写上
        i=1;//数值
        var a=1.2;//数值 浮点
        var b=true;//布尔
        var c='a';//字符串
        var d="abc";//字符串
    }
    //将变量输出
    document.write(i+"<br/>");
    document.write(a+"<br/>");
    document.write(b+"<br/>");
    document.write(c+"<br/>");
    document.write(d+"<br/>");

    //变量可以重复定义
    var d=345;
    document.write(d+"<br/>");

</script>

</body>
</html>
```

typeof操作符

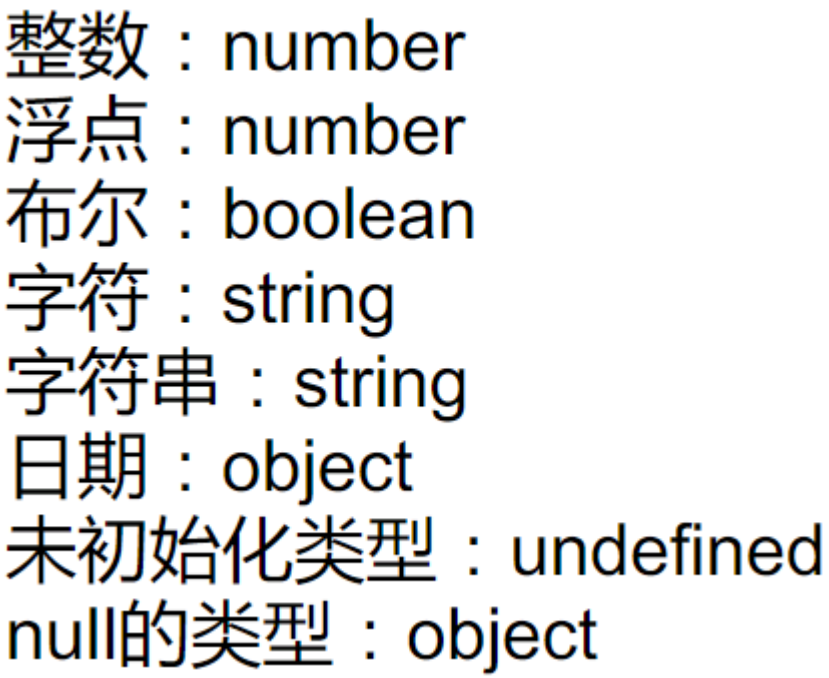
1. 作用： 用于判断某个变量的数据类型，返回这种数据类型的名字
2. 写法：typeof 变量名 或 typeof(变量名)

案例：数据类型的演示

- 案例需求：

分别输出整数、浮点数、字符串(单引号和双引号)、布尔、未定义、对象、null的数据类型

- 案例效果：



- 案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

<script type="text/javascript">
    var i = 5;
    var a = 3.14;
    var b = true;
    var c = 'a';
    var d = "abc";
    //输出每种数据类型
    document.write("整数: " + typeof(i) + "<br/>");
    document.write("浮点: " + typeof(a) + "<br/>");
    document.write("布尔: " + typeof(b) + "<br/>");
    document.write("字符: " + typeof(c) + "<br/>");

    document.write("字符串: " + typeof(d) + "<br/>");
```

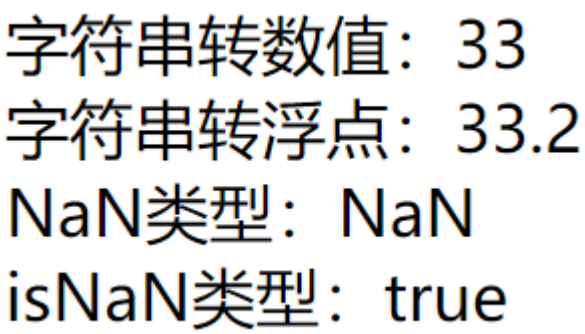
```
    // JS内置对象
    var date = new Date();
    document.write("日期: " + typeof (date) + "<br/>");
    //定义变量，但没有赋值
    var u;
    document.write("未初始化类型: " + typeof (u) + "<br/>");
    //创建变量
    var n = null;
    document.write("null的类型: " + typeof (n) + "<br/>");
</script>

</body>
</html>
```

字符串转换成数字类型

转换函数	作用
parseInt(变量)	将一个字符串类型的数字转成整数类型，如果转换失败，返回NaN = Not a Number 不是一个数字，将数字开头可以转一部分。var a="1a";
parseFloat(变量)	将一个字符串类型的数字转成浮点数，如果转换失败，返回NaN
isNaN(变量)	非数字返回true，数字返回false。isNaN = is not a number 如："abc"返回true，"123abc"返回true, "123"返回false

代码演示：字符串转数字

- 案例效果：



- 案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">

    <title>Title</title>
```

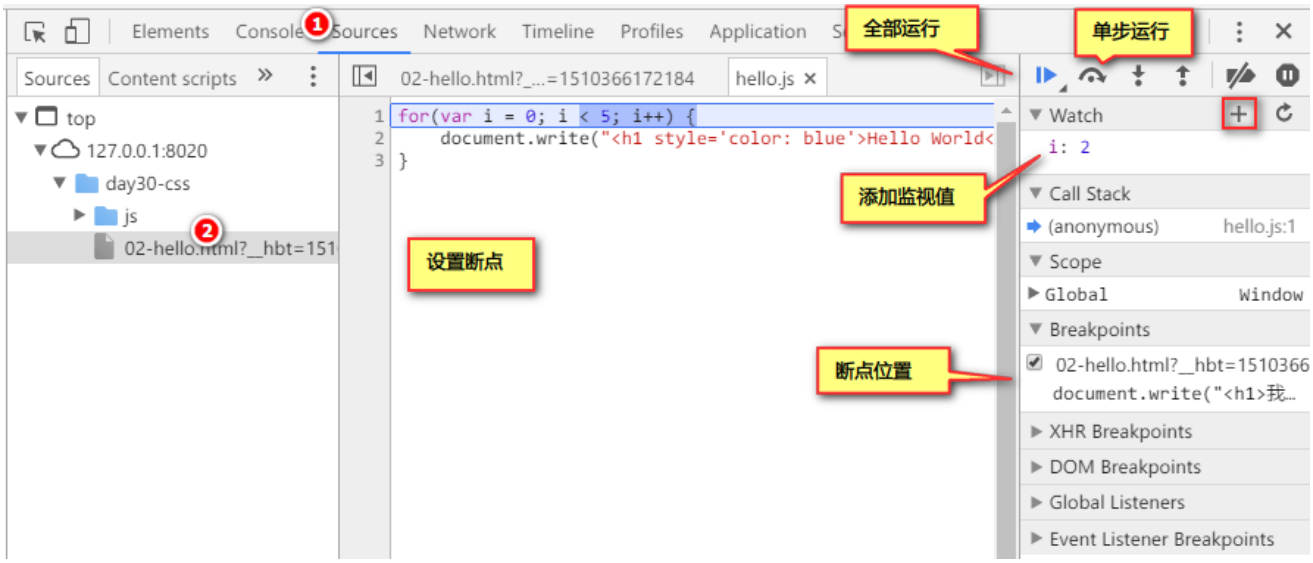
```
</head>
<body>
<script type="text/javascript">
    var x = '11';
    var y = "22.2";
    //将字符串转换为数值类型
    var result=parseInt(x)+parseInt(y);
    //将结果输出
    document.write("字符串转数值：“+result+"<br/>");
    //将字符串转换为浮点类型
    var result2=parseFloat(x)+ parseFloat(y);
    //将结果输出
    document.write("字符串转浮点：“+result2+"<br/>");

    var a="a";
    var number = parseInt(a);
    //NaN：不是一个数字
    document.write("NaN类型：“+number+"<br/>");

    //isNaN：非数字返回true 数字返回false
    var b = isNaN(a);
    document.write("isNaN类型：“+b+"<br/>");

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

在浏览器中的调试

IE、Chrome、FireFox中调试的快捷键：F12

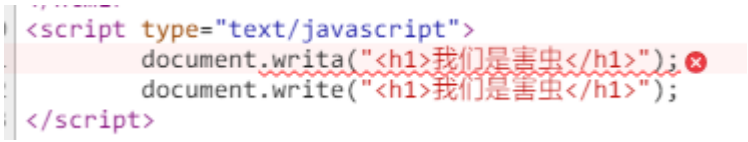
设置断点

- 注：设置断点以后要重新刷新页面才会在断点停下来



语法错误

- 注：如果有语法错误，有些浏览器会出现提示



流程控制：代码的执行过程

三种方式：

顺序结构、分支结构、循环结构

if判断

if 语句：

先判断条件表达式是否为true，如果为true则执行代码块，如果为false则不执行代码块

```
if(条件表达式) {
    //代码块;
}
```

案列：

1、如果8大于6，请输出8

2、问：小东的年龄是否大于18岁，如果大于18岁，则提示：可以交女朋友了！

if...else 语句

只执行一个分支，先判断条件表达式是否为true,如果为true则执行代码块1，如果为false则执行代码块2

```
if(条件表达式) {
    //代码块1;
}
else {
    //代码块2;
}
```

案列：

1、问：小东的年龄是否大于18岁，如果大于，则提示：可以谈恋爱，否则提示：好好学习，多敲代码

2、问：找的两个数字中的最大值

分析：

1、定义两个数值类型的变量并给它赋值

2、假设第一个数字大于第二个数字，输出第一个数字

3、否则输出第二个数字

3、问：判断这个数字是基数还是偶数

分析：

1、定义一个数字的变量num并赋值

2、判断num%2==0,返回 这是一个偶数

3、 返回 这是一个基数

预习：

弹框：prompt（“msg”） 返回的是字符串类型

if...else if....else 语句

多个分支，最终只执行一个分支。那个条件成立则执行那个条件下的代码块

```
if（条件表达式）{
    //代码块;
}
else if(条件表达式）{
    //代码块;
}
......
else {
    //代码块;
}
```

- 条件判断可以使用非逻辑运算符

数据类型	**为真**	**为假**
number	非0	0
string	**非空串**	空串：""
undefined		为假
NaN(Not a Number)		为假
object	非空 date	null

案例：求指定星期

案列需求：

有星期一至星期日，用户输入一个数，判断用户输入的数字对应的是星期几?

- 案列效果：

今天是星期五！

- 案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script type="text/javascript">
    var week=5;
    if(week<0 || week>7){
        document.write("没有对应的星期！")
    }else if(week==1){
        document.write("今天是星期一！");
    }else if(week==2){
        document.write("今天是星期二！");
    }else if(week==3){
        document.write("今天是星期三！");
    }else if(week==4){
        document.write("今天是星期四！");
    }else if(week==5){
        document.write("今天是星期五！");
    }else if(week==6){
        document.write("今天是星期六！");
    }else if(week==7){
        document.write("今天是星期日！");
    }
</script>

</body>
</html>
```

脑筋急转弯：

我的年龄是6岁，我妹妹的年龄是我的年龄的一半，当我50岁时，请问我妹妹多少岁？

swtich多分支

语法一：case后使用变量

```
switch(变量名) {
  case 值1:
     break;
  case 值2:
     break;
    .....
  default:
      break;
}
```

语法二：case后使用表达式

```
switch(true) {  //这里的变量名写成true
  case 表达式:     //如: n > 5
    break;
  case 表达式:
    break;
  default:
    break;
}
```

案例：判断一个学生的等级

案例需求：

通过prompt输入的分数，如果90~100之间，输出优秀。80~90之间输出良好。60~80输出及格。60以下输出不及格。其它分数输出:分数有误。

window对象的方法名	作用
string　prompt("提示信息","默认值”)	在浏览器上出现一个输入框 参数1：提示信息 参数2：默认出现在输入框中的值 返回：字符串
alert("提示信息")	在浏览器上弹出一个信息框，只有一个确定按钮
confirm("提示信息");	弹出提示框，显示确认、取消按钮，点击确认返回true，点击取消返回false

- 注：只要是window对象的方法，都可以省略window对象

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script type="text/javascript">
    //在浏览器打开时弹出一个输入框
    var score = window.prompt("请输入您的分数","60");
    //得到返回值
    document.write("你的分数为：“+score+"<br/>");
    //判断返回值类型
    var type=typeof (score);
    document.write("返回值类型为：“+type+"<br/>");

    //在浏览器中弹出一个提示框
    window.alert("您的分数为：“+score);

</script>

</body>
</html>
```

案例效果：

良好

案例分析：

1. 使用prompt得到输入的分数
2. 使用switch对分数进行判断
3. 如果在90到100之间，则输出优秀，其它依次类推。

案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script type="text/javascript">
    //用户输入分数并接收
    var score = window.prompt("请输入你的分数: ","60");
    /*
        使用switch判断分数等级
        100-90  优秀
        90-80   良好
        80-60   及格
        60以下    不及格
     */
    switch (true) {
        //两种不同的类型会自动进行转换，转成整数类型比较
        case score<=100 && score>90:
            document.write("优秀");
            break;
        case score>=80:
            document.write("良好");
            break;
        case score>=60:
            document.write("及格");
            break;
        case score>=0:
            document.write("不及格");
```

```
            break;
        default:
            document.write("成绩输入有误！");
            break;
    }
</script>
</body>
</html>
```

循环

while语句：

```
while（条件表达式）{
    需要执行的代码;
}
```

练习：

1、输出十次 我想玩游戏了

2、计算1-100之间所有数字的和

3、求6的阶乘

4、求1-100之间的偶数和

5、求1-100之间的基数和

6、求1-100之间能被7整除的数字

do-while语句：

最少执行1次循环

```
do {
    需要执行的代码;
}
while（条件表达式）
```

练习：

1、输出十次 我要暴富了

2、求1-100之间能被3整除的数字

for 语句

循环指定次数

```
for (var i=0; i<10; i++) {
    需要执行的代码;
}
```

练习：

1、画★，五排、每排五个

2、画三角形的★

break和continue

- break: 结束整个循环

  只能在switch和循环中使用
- continue：跳过本次循环，执行下一次循环

案列：

计算1-100之间总和，除了3的倍数

1、for 循环一百次

2、定义一个总和变量存储总和

3、使用if将3的倍数使用continue跳过，直接执行下一次循环

定义一个字符串为用户名，如果用户名没有输入正确就一直循环提示请输入用户名，如果输入正确停止循环

案例：乘法表

案例需求：

以表格的方式输出乘法表，其中行数通过用户输入

案例效果：

9x9乘法表

1x1=1								
1x2=2	2x2=4							
1x3=3	2x3=6	3x3=9						
1x4=4	2x4=8	3x4=12	4x4=16					
1x5=5	2x5=10	3x5=15	4x5=20	5x5=25				
1x6=6	2x6=12	3x6=18	4x6=24	5x6=30	6x6=36			
1x7=7	2x7=14	3x7=21	4x7=28	5x7=35	6x7=42	7x7=49		
1x8=8	2x8=16	3x8=24	4x8=32	5x8=40	6x8=48	7x8=56	8x8=64	
1x9=9	2x9=18	3x9=27	4x9=36	5x9=45	6x9=54	7x9=63	8x9=72	9x9=81

案例分析：

1. 由用户输入乘法表的行数
2. 使用循环嵌套的方式，每个外循环是一行tr，每个内循环是一个td
3. 输出每个单元格中的计算公式
4. 给表格添加样式，设置内间距

案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>99乘法表</title>
    <!--添加表格样式-->
    <style type="text/css">
        table{
            /*表格居中*/
            margin: auto;
            /*边框设置为细线模式*/
            border-collapse: collapse;
        }
        td {
```

```
            /*设置内边距*/
            padding: 5px;
        }
        /*鼠标悬浮时*/
        td:hover {
            background-color: yellow;
        }
    </style>
</head>
<body>
<script type="text/javascript">
    //用户输入行数
    var num = window.prompt("请输入乘法表的行数：","9");
    document.write("<h3 style='text-align: center'>" + num + "&times;" + num +"乘法表</h3>");
    //创建一个表格
    document.write("<table border='1'>");
    //使用for循环嵌套  外层循环表示几行
    for (var i = 1; i <= num; i++) {
        document.write("<tr>");
        for (var j = 1; j <=i ; j++) {//内层循环表示一行有几个单元格
            document.write("<td>");
            document.write(j+"&times;"+i+"="+(i*j));
            document.write("</td>");
        }
        document.write("</tr>");
    }
    document.write("</table>")
</script>
</body>
</html>
```

思考题：

求三个数的最大值?

函数的使用

函数的基本概述

什么是函数：与Java中的方法类似。函数的关键字：function

两种定义方式：

1. 命名函数
2. 匿名函数

命名函数的使用

函数的格式

```
function 函数名([参数列表]) {
   //代码块;
    [return 返回值]
}
```

a(x,y);

需求：

1、求两个数的和

注意事项：

参数列表不用些var ;js弱类型语言

1、把99乘法表修改成一个函数，当用户输入几行，就打印到几?

2、求n-m之间的总和?

3、求圆的面积

4、求长方形面积

5、求正方形面积

实现自定义函数

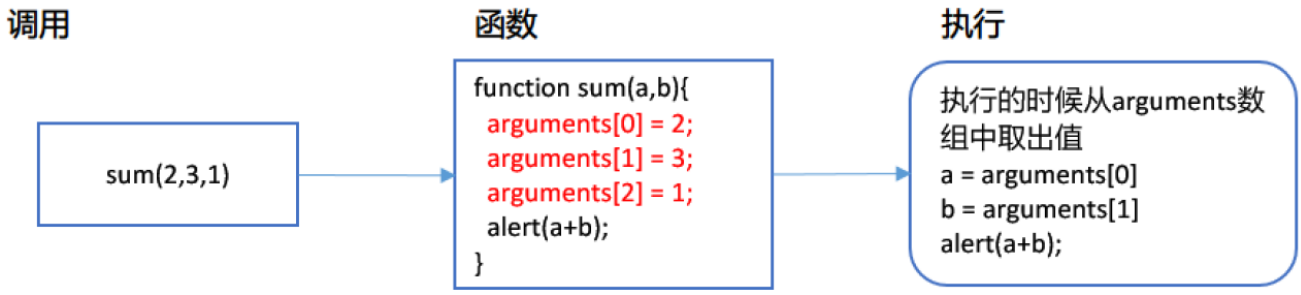
- 需求：定义一个函数实现加法功能

```
<script type="text/javascript">
    //定义一个函数实现加法功能
    function sum(a, b) {
        return a + b;
    }

    //不调用就不执行
    document.write(sum(3,5) + "<br/>");
</script>
```

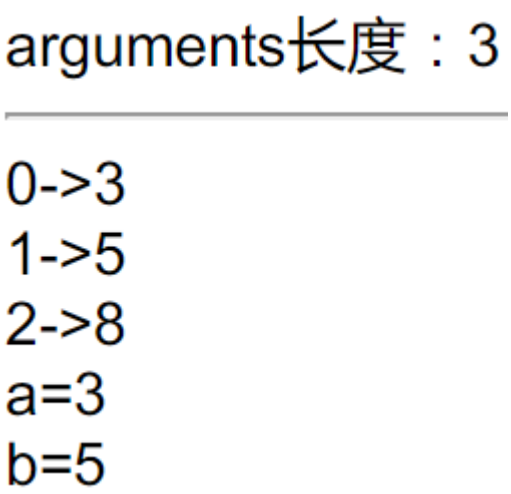
- 注意的事项

1. 形参的类型不能写，因为是弱类型，不用指定它的类型
2. 函数的返回值： 如果函数有返回值，加上return，如果没有返回值，不写return。
3. 关于函数的重载：在JS中没有函数的重载，后面定义的同名函数会覆盖前面的函数，与参数的个数无关。
4. 隐藏数组：在每个函数的内部都有一个隐藏的数组，名字叫：arguments

隐藏数组的执行过程



- 演示：定义一个函数，在函数的内部输出arguments的长度和数组中的每个元素。
- 案例效果：



- 案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>函数</title>
</head>
<body>
<script type="text/javascript">
    //定义函数
    function sum(a,b) {
        //输出隐藏数组的长度
        document.write("arguments长度: " + arguments.length +"<hr/>");
        //输出数组中的每个元素
        for (var i = 0; i < arguments.length; i++) {
            document.write(i+"---"+arguments[i]+"<br/>");
        }
        //输出形参对应的元素
        document.write("a=" + a +  "<br/>");
        document.write("b=" + b +  "<br/>");
    }
    /*//定义同名函数  在JS中没有函数的重载，后面定义的同名函数会覆盖前面的函数，与参数列数的个数无关

    function sum(){
```

```
        alert(2);
    }*/
    //调用函数
    sum(3,2,1);
</script>
</body>
</html>
```

练习：定义一个函数，传入7个参数，计算7个参数的总和

练习：定义一个函数，传入7个参数，计算7个数中的最大值

练习：定义一个函数，传入7个参数，判断7个数中是否有3的倍数

匿名函数

- 语法：没有名字的函数，如果这个函数要重用。要将函数赋值给一个变量，通过变量名去引用。

```
var 变量名 = function(参数列表) {
   //代码块
   [return 返回值];
}
```

- 函数调用：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>匿名函数</title>
</head>
<body>
<script type="text/javascript">
    //匿名函数定义
    var sum = function (a, b) {
        return a+b;
    }

    //调用函数
    document.write(sum(2,5) + "<br/>");
</script>
</body>
</html>
```

案例：实现轮播图

案例中相关的方法说明

使用到的方法	描述
document.getElementById("id")	通过标签的id得到网页上唯一的元素
window.setInterval("函数名()",时间) window.setInterval(函数名,时间)	每过指定的时间间隔之后调用指定的函数，循环不断的调用。 参数1：被调用的函数 参数2：每隔多久以后调用，单位是：毫秒

- 注：HTML中的元素应该出现在JS中得到元素的代码之前，因为如果先执行JS代码，HTML的元素还没有被浏览器加载到内存中，会出现得到元素为null的情况。

案例需求：

实现每过3秒中切换一张图片的效果，一共5张图片，当显示到最后1张的时候，再次显示第1张。

案例效果：

案例分析：

1. 创建HTML页面，页面中有一个div标签，div标签内包含一个img标签。
2. body的背景色为黑色；div的样式为：设置为居中，加边框，宽度为500px；img的id为pic，宽度500px;
3. 五张图片的名字依次是1~5.jpg，放在项目的img文件夹下，图片一开始的src为第1张图片。
4. 编写函数：changePicture()，使用setInterval()函数，每过3秒调用一次。

setInterval(方法,3000);

5. 定义全局变量：num=1。
6. 在changePicture()方法中，设置图片的src属性为img/num.jpg。

   获取img的id得到图片元素对象

   图片对象.src="/img/num.jpg";
7. 判断num是否等于6，如果等于6，则num=1；否则num++。

   if判断num=？最后一张图片，num=1;num++

案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>轮播图</title>
    <style type="text/css">
        div{
            margin: auto;
            border: black solid 1px;
            width: 500px;
        }
    </style>
</head>
<body>
<div>
    <img src="img/1.jpg" id="pic" width="500" height="300">
</div>
<script type="text/javascript">
    //定义一个全局变量
    var num=1;
    //定义改变图片的函数
    function changePic() {
        //得到图片的元素
        var pic = document.getElementById("pic");
        //更换图片的地址
        pic.src="img/"+num+".jpg";
        //图片名字+1
        num++;
        //判断是否是最后一张图片
        if(num==6){
            num=1;//回到第一张图片
        }
    }

    //设置定时器 动态更换图片 每过3秒，替换1次图片的地址src
    /*
     * 参数1：被调用的函数
     * 参数2：  过多久调用一次函数
     */

    window.setInterval(changePic,1000);
```

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

练习：

1、进入页面显示轮播图！十秒过后提示 想要继续观看，请支付！

事件的处理

事件的概述

什么是事件：网页上存在各种表单元素，每个元素会进行：点击，双击，失去焦点等操作，我们可以针对这些操作进行编程。在不同的操作中产生不同的效果。

设置事件的两种方式：

1. 方式一：命名函数

<input type="button" onclick="被调用的函数()" >
当点击button的时候，调用指定的函数

2. 方式二：匿名函数

得到元素对象.onclick = function() { //代码 }
点击指定的对象，运行function中的代码

设置事件的示例

有两个按钮，点第1个弹出一个信息框，点第2个弹出另一个信息框。分别使用两种不同的方式激活事件

- 代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<input type="button" value="按钮1" id="b1" onclick="clickMe()">
<input type="button" value="按钮2" id="b2">


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

```
    //命名函数的调用方式
    function clickMe() {
        alert("我是按钮1");
    }

    //使用匿名函数
    //得到b2这个对象
    document.getElementById("b2").onclick = function () {
        alert("我是按钮2");
    }
</script>
</body>
</html>
```

常用的事件：

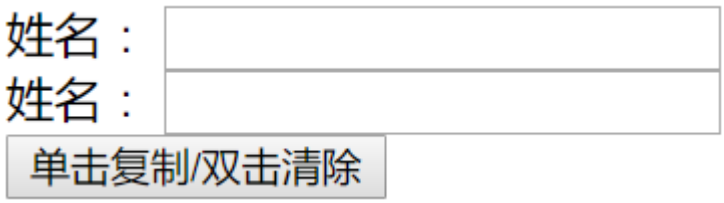
加载完成事件

- onload 元素是在网页加载完毕以后运行
- 示例：页面加载完毕以后，才执行相应的JS代码

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script type="text/javascript">
    //命名函数的调用方式
    function clickMe() {
        alert("我是按钮1");
    }

    //在页面加载完毕以后再执行这个代码
    window.onload = function () {
        //使用匿名函数
        //得到b2这个对象
        document.getElementById("b2").onclick = function () {
            alert("我是按钮2");
        }
    };
</script>

<input type="button" value="按钮1" id="b1" onclick="clickMe()">
<input type="button" value="按钮2" id="b2">

</body>
</html>
```

鼠标点击

- onclick 单击事件
- ondblclick 双击事件



- 示例：单击复制文本内容，双击清除文本内容

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
用户名：<input type="text" id="t1"><br>
用户名：<input type="text" id="t2"><br>
<hr>
<input type="button" value="单击复制/双击清除" id="b1">

<script type="text/javascript">
    //得到b1这个对象，设置单击事件
    document.getElementById("b1").onclick = function () {
        //得到t1的值
        document.getElementById("t2").value = document.getElementById("t1").value;
    }

    //双击事件
    document.getElementById("b1").ondblclick = function () {
        document.getElementById("t1").value = "";
        document.getElementById("t2").value = "";
    }
</script>
</body>
</html>
```

鼠标移动：

- onmouseover 鼠标移到某个元素的上面
- onmouseout 鼠标移出某个元素

- 示例：将鼠标移动到img上显示图片，移出则显示另一张图片。图片设置边框，宽500px

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<img src="img/1.jpg" width="500" id="pic">

<script type="text/javascript">
    //得到图片对象
    document.getElementById("pic").onmouseover = function () {
        //修改它自己的src
        //document.getElementById("pic").src = "img/2.jpg";
        this.src = "img/2.jpg";
    }

    //鼠标移出事件
    document.getElementById("pic").onmouseout = function () {
        //修改它自己的src
        this.src = "img/1.jpg";
    }
</script>
</body>
</html>
```

this关键字的作用：

1. 出现在控件的事件方法中：

```
<!--this相当于button对象-->
<input type="button" value="按钮1" id="b1" onclick="clickMe(this)">
<input type="button" value="按钮23" id="b1" onclick="clickMe(this)">
<input type="button" value="按钮1" id="b1" onclick="clickMe(this)">
//命名函数的调用方式
function clickMe(obj) {
    alert(obj);  //按钮对象
}
```

1. 出现在匿名函数的代码中：

```
//鼠标移出事件
document.getElementById("pic").onmouseout = function () {
    //修改它自己的src
    this.src = "img/1.jpg";
}
```

焦点相关的：

- 什么是焦点： 如果某个网页元素得到了光标，处于一种可以操作状态，得到焦点。
- onblur：失去焦点
- onfocus: 得到焦点
- 示例：当文本框获取到焦点的同时,文本框后面显示红色文字。失去焦点，文本框后面的文字消失。


```
<input type="text" id="user"><span id="info" style="color: red"></span>
<script type="text/javascript">
    //得到文本框，设置得到焦点的事件
    document.getElementById("user").onfocus = function () {
        //得到span对象，修改它内部的html的内容
        document.getElementById("info").innerHTML = "用户名不能为空";
    }

    //失去焦点
    document.getElementById("user").onblur = function () {
        //得到span对象，修改它内部的html的内容
        document.getElementById("info").innerHTML = "";
    }
</script>
```

改变事件：

- onchange：

  某个元素的内容或选项发生变化的事件，文本框必须在失去焦点以后才会激活这个事件

- 示例：

1. 选中不同的城市出现一个信息框，显示你选中的城市的名字
2. 用户输入英文字母以后，文本框的字母全部变成大写

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
城市:
<select id="city">
    <option value="广州">广州</option>
    <option value="深圳">深圳</option>
    <option value="珠海">珠海</option>
</select>
<hr>
英文: <input type="text" id="user">

<script type="text/javascript">
    //改变事件
    document.getElementById("city").onchange = function () {
        alert(this.value);   //this是select对象
    }

    //文本框改变事件
    document.getElementById("user").onchange = function () {
        //变成大写
        this.value = this.value.toUpperCase();
    }
</script>
</body>
</html>
```

键盘相关

- onkeyup: 松开按键
- 示例：用户输入英文以后马上变成大写

- onkeydown： 按下按键
- 示例：在文本框中按下任意键，在h4中显示这个按键的键盘码

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

```
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
城市:
<select id="city">
    <option value="广州">广州</option>
    <option value="深圳">深圳</option>
    <option value="珠海">珠海</option>
</select>
<hr>
英文: <input type="text" id="user">
<hr>
<h2 id="key"></h2>

<script type="text/javascript">
    //改变事件
    document.getElementById("city").onchange = function () {
        alert(this.value);   //this是select对象
    }

    //文本框改变事件
   /* document.getElementById("user").onchange = function () {
        //变成大写
        this.value = this.value.toUpperCase();
    }*/

   //使用松开按键事件
    document.getElementById("user").onkeyup = function () {
        //变成大写
        this.value = this.value.toUpperCase();
    }

    //按下键，在h2中显示按下键键盘码，传入一个对象，通过这个对象keyCode属性得到按键编码
    document.getElementById("user").onkeydown = function (event) {
        document.getElementById("key").innerHTML = event.keyCode;
    }
</script>
</body>
</html>
```

键盘码

irfirefox2.0中不支持 window.event.keyCode,

但是我们可以用event.which代替。但是为了使其能更具有普遍的兼容性，最好用event.keyCode|| event.which.

Keycode对照表

字母和数字键的键码值(keyCode)							
按键	键码	按键	键码	按键	键码	按键	键码
A	65	J	74	S	83	1	49
B	66	K	75	T	84	2	50
C	67	L	76	U	85	3	51
D	68	M	77	V	86	4	52
E	69	N	78	W	87	5	53
F	70	O	79	X	88	6	54
G	71	P	80	Y	89	7	55
H	72	Q	81	Z	90	8	56
I	73	R	82	0	48	9	57

数字键盘上的键的键码值(keyCode)				功能键键码值(keyCode)			
按键	键码	按键	键码	按键	键码	按键	键码
0	96	8	104	F1	112	F7	118
1	97	9	105	F2	113	F8	119
2	98	*	106	F3	114	F9	120
3	99	+	107	F4	115	F10	121
4	100	Enter	108	F5	116	F11	122
5	101	-	109	F6	117	F12	123
6	102	.	110				
7	103	/	111				

控制键键码值(keyCode)							
按键	键码	按键	键码	按键	键码	按键	键码
BackSpace	8	Esc	27	Right Arrow	39	-_	189
Tab	9	Spacebar	32	Dw Arrow	40	.>	190
Clear	12	Page Up	33	Insert	45	/?	191
Enter	13	Page Down	34	Delete	46	`~	192
Shift	16	End	35	Num Lock	144	[{	219
Control	17	Home	36	;:	186	|	220
Alt	18	Left Arrow	37	=+	187	]}	221
Cape Lock	20	Up Arrow	38	,<	188	'"	222

小结

事件名	作用
onclick	单击
ondblclick	双击
onload	加载完毕
onfocus	得到焦点
onblur	失去焦点
onchange	改变事件
onmouseover	鼠标移上
onmouseout	鼠标移出
onkeyup	松开键盘按键
onkeydown	按下按键
onsubmit	表单提交事件，下次课详细讲解

JavaScript的内置对象

数组对象

数组的四种方式

创建数组的方式	说明
new Array()	创建一个长度为0的数组
new Array(5)	创建一个指定长度的数组
new Array(2,4,10,6,41)	指定数组中每个元素，创建数组

JS中数组的特点

1. 数组中每个元素的类型是可以不同的。
2. JS数组长度是可以动态变化
3. 数组中包含各种方法

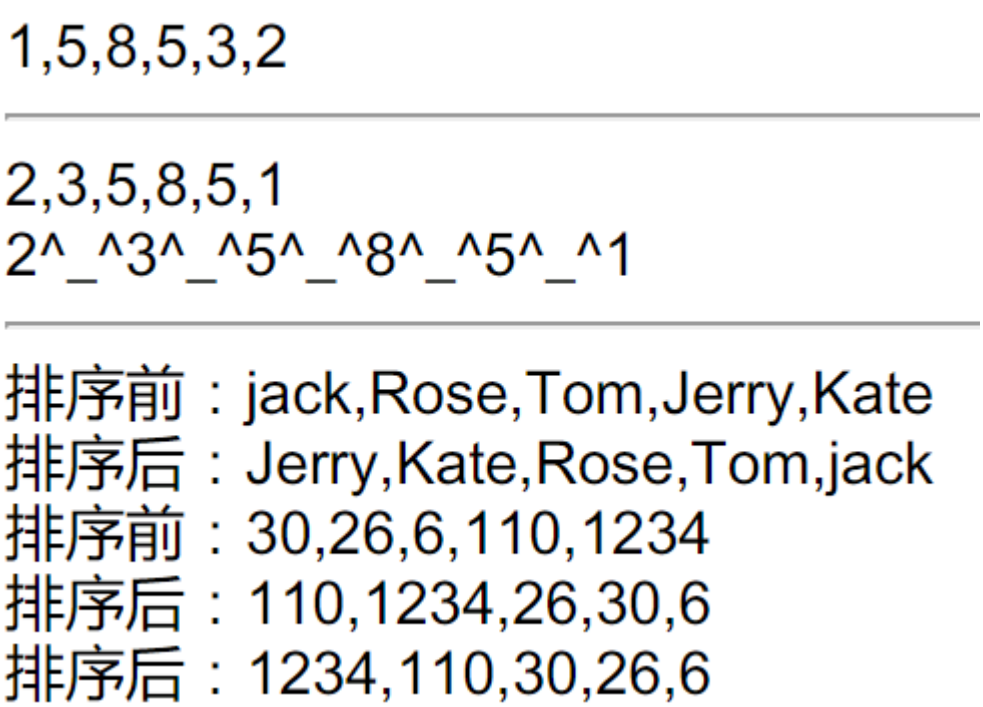
```
<!DOCTYPE html>

<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script type="text/javascript">
    var arr = [4,3,20,6];
    //数组中每个元素可以不同
    arr[2]= true;
    arr[3] = "abc";
    document.write("数组的长度: " + arr.length + "<hr/>");
    arr[5] = 100;
    document.write("数组的长度: " + arr.length + "<hr/>");
    for (var i = 0; i < arr.length; i++) {
        document.write(arr[i] + " ");
    }
    document.write("<hr>");
</script>
</body>
</html>
```

常用方法

方法名	功能
concat()	用于拼接一个或多个数组，返回拼接好的数组
reverse()	用于数组的反转
join(separator)	将一个数组通过分隔符，拼接成一个字符串。功能与java中split函数相反
sort()	对字符串数组进行排序 如果要对数字进行排序，要指定比较器函数。 sort(function(m,n)) 数字两两比较 1) 如果m大于n，则返回正整数 2) 如果m小于n，则返回负整数 3) 如果m等于n，则返回0

- 案例效果:



- 案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script type="text/javascript">
    var a1 = [1,5,8];
    var a2 = [5,3,2];
    var a3 = a1.concat(a2);   //把a1和a2拼接成一个数组
    document.write(a3 + "<hr/>");
    //把a3反转
    a3.reverse();
    document.write(a3 + "<br/>");
    //拼接成一个字符串
    var str = a3.join("^_^");
    document.write(str  + "<br/>");
    document.write("<hr/>");
    // 给字符串数组排序
    var arr = ['jack','Rose',"Tom","Jerry","Kate"];
    document.write("排序前: " + arr + "<br/>");
    arr.sort();
    document.write("排序后: " + arr + "<br/>");

    // 数字排序
    var arr = [30,26,6,110,1234];
    document.write("排序前: " + arr + "<br/>");
    //默认按字符串排序

    arr.sort();
```

```
    document.write("排序后: " + arr + "<br/>");
    //如果使用数字进行排序，必须指定比较器
    arr.sort(function (a, b) {
        return b-a;
    });
    document.write("排序后: " + arr + "<br/>");
</script>
</body>
</html>
```