# 函数

## 内容:

函数的概念

函数的使用

函数的参数

函数的返回值

arguments的使用

函数案例

函数的两种声明方式

## 目标:

能够说出为什么需要函数

能够根据语法书写函数

能够根据需求封装函数

能够说出形参和实参的传递过程

能够使用函数的返回值

能够使用arguments获取函数的参数

# 函数的概念

函数是一个完成特定任务的代码语句块

函数分类：系统函数和自定义函数

## 案例1

使用js完成计算1~100的和

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript">  var sum=0;  // 使用for循环  for (var i = 1; i <=100; i++) {  sum=sum+i;  }  //显示sum  // alert(sum)  console.log(sum);  </script>  </head>  <body>  </body>  </html> |

# 函数的使用

函数在使用时分为两步：声明函数和调用函数

### 声明函数

function 函数名() {

//函数体代码

}

说明:

function 是声明函数的关键字,必须小写

由于函数一般是为了实现某个功能才定义的， 所以通常我们将函数名命名为动词，比如 getSum

### 调用函数

函数名(); // 通过调用函数名来执行函数体代码

说明:

调用的时候千万不要忘记添加小括号

口诀：函数不调用，自己不执行。

注意：声明函数本身并不会执行代码，只有调用函数时才会执行函数体代码。

### 案例

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script>  // 1.声明函数  function sayHello(){  console.log("Hello World");  }  // 2.调用函数  sayHello();  </script>  </head>  <body>  </body>  </html> |

## 提问

1. 函数是做什么的（作用）？

封装一段被重复调用的语句块

1. 声明函数用什么关键词？

function

1. 如何调用函数？

函数名(实参1,实参2,...)

1. 封装是什么意思？

封装就是将一段代码放到一起,使用的时候调用.

## 阶段总结

### 函数的概念

完成特定任务的代码块,(封装一段被重复调用的语句块)

### 函数的使用

①声明函数:

function 函数名(形参1, 形参2,...){

//语句块...

}

②调用函数

函数名(实参1,实参2,...)

# 函数的参数

## 形参和实参

①形参:函数声明的时候在函数名后的小括号里面的参数

②实参:函数调用的时候在函数名后的小括号里面的参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| 形参 | 形式上的参数,在函数声明的时候传递的参数,不知道他是什么 |
| 实参 | 实际的参数,在函数调用的时候传递的参数,实参传递给形参 |

③参数的作用

参数的作用 : 在函数内部某些值不能固定，我们可以通过参数在调用函数时传递不同的值进去

## 带参数的函数声明

function 函数名(形参1, 形参2 , 形参3...) { // 可以定义任意多的参数，用逗号分隔

// 函数体

}

## 带参数的函数调用

函数名(实参1, 实参2, 实参3...);

### 带一个参数的函数调用

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script>  //封装一个做饭的函数--声明函数(定义函数)  function cook(food){  console.log(food);//console:代表浏览器控制台对象.log:代表函数,打印参数的日志信息 ;console.log(food)代表在浏览器控制台打印参数的日志信息  }  //调用函数(执行函数)  cook("醋溜白菜")  cook("果木烤鸭");  cook("酱排骨");  </script>  </head>  <body>  </body>  </html> |

### 案例：利用函数求任意两个数的和

思路:

1. 定义函数,
2. 2个形式参数
3. 函数体:求2个形参的和,并在控制台输出
4. 调用函数,传递2个实参

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script>  //1.定义函数  function getSum(a,b){  console.log(a+b);  }  //2.调用函数  getSum(10,20)  </script>  </head>  <body>  </body>  </html> |

## 形参个数和实参个数不匹配问题

|  |  |
| --- | --- |
| 参数个数 | 说明 |
| 形参个数和实参个数相等 | 输出正确的结果 |
| 实参个数多于形参个数 | 只取到形参的个数 |
| 实参个数少于形参个数 | 多的形参(未匹配上的形参)定义为undefined,结果是NaN(Not a Number不是一个数字) |

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script>  //1.定义函数  function getSum(a,b){  console.log(a);  console.log(b);  console.log(a+b);  }  //2.调用函数  // getSum(10,20) //形参和实参个数相等，输出正确结果  // getSum(10,20,30,40)//实参个数多于形参个数 只取到形参的个数  getSum(10)//实参个数少于形参个数 多的形参(未匹配上的形参)定义为undefined,结果是NaN(Not a Number不是一个数字)  </script>  </head>  <body>  </body>  </html> |

建议:形参和实参个数相等

## 总结

1.函数可以带参数也可以不带参数

2.声明函数的时候，函数名括号里面的是形参，形参的默认值为 undefined

3.调用函数的时候，函数名括号里面的是实参

4.多个参数中间用逗号分隔

5.形参的个数可以和实参个数不匹配，但是结果不可预计，我们尽量要匹配

# 函数的返回值

## return语句

在函数体使用return,返回一个值,

### return语句的格式:

// 声明函数

function 函数名（）{

...

return 需要返回的值；

}

// 调用函数

var result=函数名(); // 此时调用函数就可以得到函数体内return 后面的值

### 案例

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script>  //声明函数,有返回值  function getSum(){  return 6;  }  //调用函数,得到返回值  var result=getSum();  //输出返回值  console.log(result);  </script>  </head>  <body>  </body>  </html> |

### 案例1

定义函数:2个参数,求2个参数的最大值返回

步骤:

1. 定义函数getMax(a,b)
2. 函数体:求2个参数的最大值返回
3. 调用函数 得到返回值就是最大值getMax(10,20)
4. 打印最大值

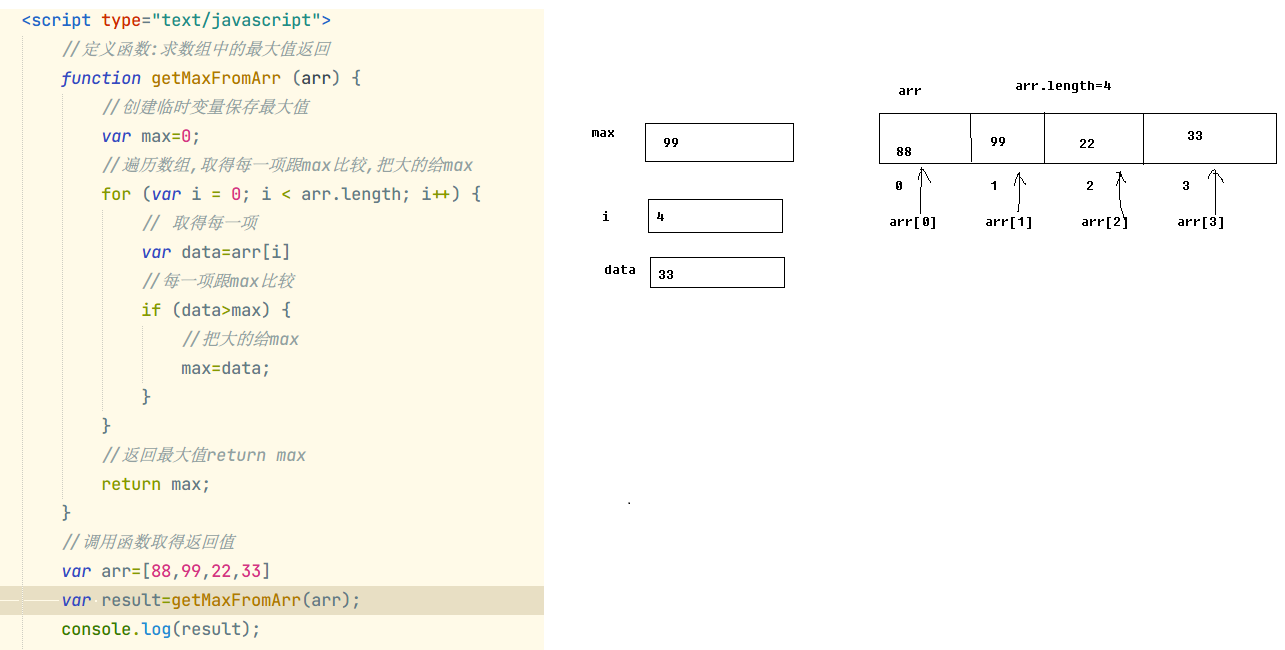
|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script>  //定义函数  /\* function getMax(a,b){  var max;  if(a>b){  max=a;  }else{  max=b;  }  return max;  } \*/  /\* function getMax(a,b){  if(a>b){  return a;  }else{  return b;  }  return max;  } \*/  function getMax(a,b){  return a>b?a:b;  }  //调用函数,得到返回值  var result=getMax(10,60);  //显示  console.log(result);  </script>  </head>  <body>  </body>  </html> |

### 案例2

定义一个获取数组中最大数的函数

思路:

1. 定义函数getMaxFromArr(arr)
2. 函数体:求数组的最大值并返回
   1. var max=0;
   2. 遍历数组,取得每一项跟max比较,把大的给max
   3. return max
3. 调用函数传递实参数组,取得返回值打印



|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript">  //定义函数:求数组中的最大值返回  function getMaxFromArr (arr) {  //创建临时变量保存最大值  var max=arr[0];  //遍历数组,取得每一项跟max比较,把大的给max  for (var i = 1; i < arr.length; i++) {  // 取得每一项  var data=arr[i]  //每一项跟max比较  if (data>max) {  //把大的给max  max=data;  }  }  //返回最大值return max  return max;  }  //调用函数取得返回值  var arr=[88,99,22,33]  var result=getMaxFromArr(arr);  console.log(result);    </script>  </head>  <body>  </body>  </html> |

## return终止语句

return语句返回值并跳出函数体,return 语句之后的代码不被执行

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript">  //定于一个求和函数  function getSum (a,b) {  return a+b;//返回值并跳出函数体  alert("我是return后面的语句");//这个语句不被执行  }  //调用函数  var result=getSum(10,20);  //输出返回值  console.log(result);  </script>  </head>  <body>  </body>  </html> |

## return只能返回一个值

return 只能返回一个值。如果用逗号隔开多个值，以最后一个为准。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript">  //定义函数,返回a,b,以最后一个为准  function add (a,b) {  return a,b;  }  //调用  var result=add(10,20);  //输出  console.log(result);//20  </script>  </head>  <body>  </body>  </html> |

## 函数没有 return 返回 undefined

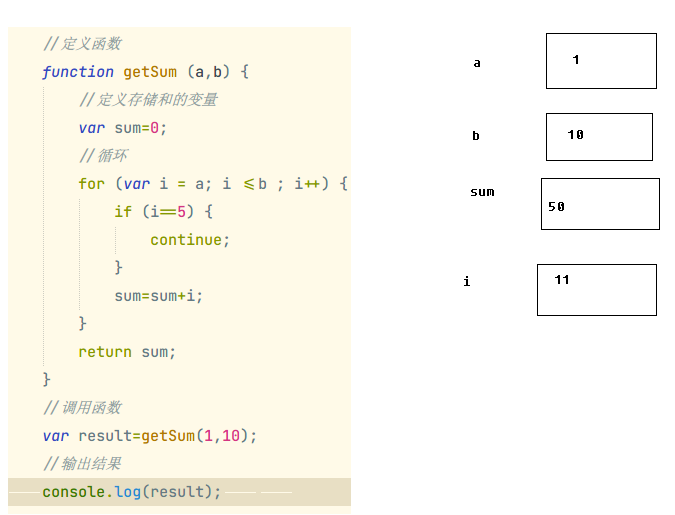
|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript">  //定义一个函数,没有return语句  function add (a,b) {  console.log("add函数");  }  //调用  var result=add(10,20)  //输出  console.log(result);  </script>  </head>  <body>  </body>  </html> |

## break,continue,return的区别

1. break ：结束当前的循环体（如 for、while）

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript">  //定义函数  function getSum (a,b) {  //定义存储和的变量  var sum=0;  //循环  for (var i = a; i <=b ; i++) {  if (i==5) {  break;  }  sum=sum+i;  }  return sum;  }  //调用函数  var result=getSum(1,10);  //输出结果  console.log(result); //10    </script>  </head>  <body>  </body>  </html> |

1. continue ：跳出本次循环，继续执行下次循环（如 for、while）



|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript">  //定义函数  function getSum (a,b) {  //定义存储和的变量  var sum=0;  //循环  for (var i = a; i <=b ; i++) {  if (i==5) {  continue;  }  sum=sum+i;  }  return sum;  }  //调用函数  var result=getSum(1,10);  //输出结果  console.log(result);    </script>  </head>  <body>  </body>  </html> |

3.return ：不仅可以退出循环，还能够返回 return 语句中的值，同时还可以结束当前的函数体内的代码

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript">  //定义函数  function getSum (a,b) {  //定义存储和的变量  var sum=0;  //循环  for (var i = a; i <=b ; i++) {  if (i==5) {  return sum;  }  sum=sum+i;  }  return sum;  }  //调用函数  var result=getSum(1,10);  //输出结果  console.log(result); //10    </script>  </head>  <body>  </body>  </html> |



# 案例

案例：创建一个函数，实现两个数之间的加减乘除运算，并将结果返回

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript">  //定义函数  function calculate (a,b) {  return [a+b,a-b,a\*b,a/b];  }  //调用函数,取得返回值  var a=parseInt(prompt("请输入第一个数:"))//parseInt将字符串转成整数  var b=parseInt(prompt("请输入第二个数:"))  var result=calculate(a,b);  console.log(result);  </script>  </head>  <body>  </body>  </html> |

# 案例

1写一个函数，用户输入任意两个数字的任意算术运算（简单的计算器小功能），并能弹出运算后的结果。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript">  //定义函数  function calculate (a,b,op) {  switch (op){  case "+":  return a+b;  break;  case "-":  return a-b;  break;  case "\*":  return a\*b;  break;  case "/":  return a/b;  break;  default:  break;  }  }  //用户输入信息  var a=parseFloat(prompt("请输入第一个数:"))  var b=parseFloat(prompt("请输入第二个数:"))  var op=prompt("请输入运算符:")  //调用函数  var result=calculate(a,b,op)  //alert弹出  alert(result)  </script>  </head>  <body>  </body>  </html> |

# 案例

2写一个函数，求出用户输入任意两个数字的最大值，并能出弹运算后的结果

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript">  //定义函数  function getMax (a,b) {  return a>b?a:b  }  //提示输入  var a=parseFloat(prompt("请输入第一个数:"))  var b=parseFloat(prompt("请输入第二个数:"))  //调用函数  var result=getMax(10,5)  //显示  alert(result);  </script>  </head>  <body>  </body>  </html> |

# 总结

## 函数的概念

执行特定功能的代码块

## 函数的使用

1. 函数声明

function 函数名(形参1,形参2,...){

//代码块

}

1. 函数调用

函数名(实参1,实参2,...)

## 函数的参数

1. 形参和实参
   1. 形参:形式上的参数.函数声明时的参数,没有确定的值undefined
   2. 实参:实际的参数,函数调用的时候给的参数,有确定的值,实参传递给形参
2. 形参个数和实参个数不匹配问题

|  |  |
| --- | --- |
| 参数个数 | 说明 |
| 形参个数和实参个数相等 | 输出正确的结果 |
| 实参个数多于形参个数 | 只取到形参的个数 |
| 实参个数少于形参个数 | 多的形参(未匹配上的形参)定义为undefined,结果是NaN(Not a Number不是一个数字) |

## 函数的返回值

1.return语句的格式:

// 声明函数

function 函数名（）{

...

return 需要返回的值；

}

// 调用函数

var result=函数名(); // 此时调用函数就可以得到函数体内return 后面的值

1. return终止语句

return语句返回值并跳出函数体,return 语句之后的代码不被执行

1. return只能返回一个值

return 10,6 返回的是最后一个值

1. 如果函数体不写return,返回的是undefined
2. break,continue,return的区别
   1. break:跳出循环
   2. continue:跳出本次循环,并继续下一次循环
   3. return:跳出循环和函数体