17.9.9 使用 Arguments 对象

Arguments 对象表示参数集合，它是一个伪类数组，拥有与数组相似的结构，可以通过数组下标的形式访问函数实参值。

【示例 1】在下面示例中，函数没有定义形参，但是在函数体内通过 Arguments 对象可以获取传递给该函数的每个实参值。

```
function f(){                          // 定义没有形参的函数
    for(var i = 0; i < arguments.length; i ++ ){
   // 循环读取函数的 arguments 对象
        alert(arguments[i]);           // 显示指定下标的实参的值
    }
}
f(3, 3, 6);                            // 逐个显示每个传递的实参
```

【示例 2】Arguments 对象仅能够在函数体内使用，作为函数的属性而存在。用户可以通过点运算符访问 Arguments 对象。由于 Arguments 对象在函数体内是可见的，也直接引用 Arguments 对象。

```
function f(){                          // 定义没有形参的函数
    for(var i = 0; i < f.arguments.length; i ++ ){
   // 循环读取函数的 arguments 对象
        alert(arguments[i]);           // 显示指定下标的实参的值
    }
}
f(3, 3, 6);                            // 逐个显示每个传递的实参
```

Arguments 对象是一个伪类数组，可以使用数组下标的形式访问每个实参值，如 arguments[i]，其中 arguments 表示对 Arguments 对象的引用，变量 i 是 arguments 集合的下标值，从 0 开始，直到 arguments.length。其中 length 是 arguments 对象的一个属性，表示 arguments 对象包含的实参个数。

【示例 3】使用 Arguments 对象可以随时编辑实参值。在下面示例中使用 for 循环遍历 arguments 对象，然后把循环变量的值传递给实参，以便动态改变实参值。

```
function f(){
    for(var i = 0; i < arguments.length; i ++ ){
   // 遍历 arguments 对象元素
        arguments[i] =i;               // 修改每个实参的值
        alert(arguments[i]);           // 提示修改的实参值
    }
}
f(3, 3, 6);                            // 返回提示 1、2、3，而不是 3、3、6
```

【示例 4】通过修改 Arguments 对象的 length 属性值，也可以达到改变函数实参个数的目的。当 length 属性值增大时，则增加的实参值为 undefined，如果 length 属性值减小，则会丢弃 arguments 数据集合后面对应个数的元素。

```
function f(){
    arguments.length = 2    ;      // 修改 arguments 对象的 length 属性值
    for(var i = 0; i < arguments.length; i ++ ){
        alert(arguments[i]);
    }
}
f(3, 3, 6);                            // 返回提示 3、3
```

Arguments 对象包含一个 callee 属性，它引用当前 Arguments 对象所属的函数，使用该属性可以在函数体内调用函数自身。在匿名函数中，callee 属性比较有用，利用它可以设计函数迭代操作。

【示例 5】在下面示例中，使用 arguments.callee 获取匿名函数，然后通过函数的 length 属性获取函数形参个数，最后比较实参与形参个数以检测用户传递的参数是否符合要求。

```
function f(x, y, z){
    var a = arguments.length;      // 获取函数实参的个数
```

```
    var b = arguments.callee.length;      // 获取函数形参的个数
    if (a != b){                          // 如果形参和实参个数不相等，则提示错误信息
      throw new Error("传递的参数不匹配");
    }
    else{                                 // 如果形参和实参数目相同，则返回它们的和
      return x + y + z;
    }
}
alert(f(3, 4, 5));                        // 返回值为 12
```

Function 对象的 length 属性返回的是函数形参个数，而 Arguments 对象的 length 属性返回的是函数实参个数。

【示例 6】如果不是匿名函数，则 arguments.callee 等价于函数名，对于上面示例可以改为如下形式。

```
function f(x, y, z){
    var a = arguments.length;         // 获取函数实参的个数
    var b = f.length;                 // 在函数体内通过函数名获取函数形参的个数
    if (a != b){                      // 如果形参和实参个数不相等，则提示错误信息
      throw new Error("传递的参数不匹配");
    }
    else{                             // 如果形参和实参数目相同，则返回它们的和
      return x + y + z;
    }
}
alert(f(3, 4, 5));                    // 返回值为 12
```

灵活使用 Arguments 对象，可以提升使用函数的灵活性，增强函数在抽象编程中的适应能力和纠错功能。下面结合两个个典型示例展示 Arguments 对象在实践中的应用。

【示例 7】使用 Arguments 对象能够增强函数应用的灵活性。例如，如果函数的参数个数不确定，或者函数的参数个数很多，而又不想为每个参数都定义一个形参变量，此时可以省略参数，直接在函数体内使用 Arguments 对象来访问调用函数的实参值。

下面示例定义一个求平均值的函数，它借助 Arguments 对象来计算函数接收参数的平均值。

```
function avg(){                       // 求平均数
    var num = 0, l = 0;               // 声明并初始化临时变量
    for(var i = 0; i < arguments.length; i ++ ){    // 遍历所有实参
      if(typeof arguments[i] != "number")         // 如果参数不是数值
        continue;                     // 则忽略该参数值
      num += arguments[i];            // 计算参数的数值之和
      l ++ ;                          // 计算参与和运算的参数个数
    }
    num /= l;                         // 求平均值
    return num;                       // 返平均值
}
alert(avg(1, 2, 3, 4));               // 返回 2.5
alert(avg(1, 2, "3", 4));             // 返回 2.3333333333333335
```

【示例 8】验证函数参数的合法性。在页面设计中经常需要验证表单输入值，下面示例检测文本框中输入的值是否为合法的邮箱地址。

```
function isEmail(){
    if(arguments.length>1) throw new Error("只能够传递一个参数");
    // 检测参数个数
    // 定义正则表达式
    var regexp = /^\w+((-\w+)|(\.\w+))*\@[A-Za-z0-9]+
            ((\.|-)[A-Za-z0-9]+)*\.[A-Za-z0-9]+$/;
    if (arguments[0].search(regexp)!= -1)       // 匹配实参的值
      return true;                    // 如果匹配则返回 true
```

```
    else
        return false;                    // 如果不匹配则返回 false
}
var email = "zhuyinhong@css8.cn";        // 声明并初始化邮箱地址字符串
alert(isEmail(email));                   // 返回 true
```