JavaScript+jQuery前端开发基础教程

微课版

第3章　数组和函数

本章主要内容：

- 数组
- 函数
- 内置函数

3.1 数组

- 数组是一组相关数据的有序集合，其中的数据项被称为数组元素。数组元素在数组中的位置称为索引或者下标，索引最小值为0。
- 数组元素用数组名和下标来表示，例如，假设a数组中有3个数组元素，这3个元素可表示为a[0]、a[1]和a[2]。
- JavaScript是弱类型的，所以数组中的各个数组元素可存放不同类型的数据，甚至可以是对象或数组。JavaScript不支持多维数组，但可通过在数组元素中保存数组来模拟多维数组。
- JavaScript的数组本质上也是一种对象，数组的类型为object。

3.1.1　创建数组

- 可用下面的几种方式创建数组。
  - 使用数组常量。
  - 使用...扩展操作符。
  - 使用Array()函数。
  - 使用Array.of()方法。
  - 使用Array.from()方法。

1．使用数组常量创建数组

- 数组常量是用“[”和“]”符号括起来的一组数据，用逗号分隔。可将数组常量赋给变量。
  - `var a = []`
    //创建一个空数组
  - `var b = [ 1 , 2 , 3 ]`
    //b[0]=1、b[1]=2、b[2]=3
  - `var c = [ ″abc″ , true , 100 ]` //c[0]=″abc″、c[1]=true、c[2]=100
- 数组元素也可以是数组。

2. 使用...扩展操作符创建数组

- ...扩展操作符将可迭代对象解析为数组元素。
  - `var a = [ 1 , 2 , 3 ]`
  - `var b = ["a" , ...a , "b"]`
    `//b=["a" , 1 , 2 , 3 , "b"]`
  - `var b = [..."abc"]`
    `//b=["a" , "b" , "c"]`

3．使用Array()函数创建数组

- Array()函数是数组对象的构造函数，可用它来创建数组。不提供参数时，Array()函数创建一个空数组（空数组长度为0）。
  - `var a=new Array()` //创建一个空数组
- 参数为一个数值时，Array()函数将其作为数组长度来创建指定长度的数组。
  - `var a=new Array(5)` //创建有5个元素的数组，元素初始值为undefined
- 参数为多个值时，Array()函数将这些值作为数组元素来创建数组。
  - `var b=new Array(1,true,"abc")` //b[0]=1、b[1]=true、b[2]="abc"

4. 使用Array.of()方法创建数组

- Array.of()方法将参数作为数组元素来创建数组。
  - `var a = Array.of( )`

    `//创建一个空数组`
  - `var b = Array.of( 5 )`

    `//创建数组为[5]`
  - `var c = Array.of( 1 , 2 , 3 )`

    `//c = [ 1 , 2 , 3 ]`

5. 使用Array.from()方法创建数组

- Array.from()方法使用可迭代对象或者类数组对象来创建数组。
  - `var a = Array.from( [ 1 , 2 , 3 ] )`
    `//复制数组，a = [ 1 , 2 , 3 ]`
  - `var b = Array.from("abc")`
    `//b=["a" , "b" , "c"]`

3.1.2　使用数组

- 1．使用数组元素
- 数组元素通过数组名和下标进行引用。
- 一个数组元素等同于一个变量，可以为数组元素赋值，或将其用于各种运算。
  - `var a = new Array(3)　　//创建数组`
  - `a[0] = 1　　//为数组元素赋值`
  - `a[1] = 2`
  - `a[2] = a[1] + a[0]　　//将数组元素用于计算`

- 在将数组用于字符串操作时，JavaScript会调用数组对象的toString()方法将其转为字符串。
- JavaScript的大多数内置对象均有toString()方法，用于将对象转换为字符串。

2. 使用多维数组

- JavaScript没有多维数组的概念，但可在数组元素中保存数组，从而实现多维数组。
  - `var a = new Array(3)`
  - `a[0] = 1`
  - `a[1] = new Array(1,2)` //将数组存入数组元素
  - `a[2] = new Array('ab', 'cd', 'ef')`

3．数组下标范围

- 在JavaScript中，数组下标最小值为0，最大值为数组长度减1。
- JavaScript没有数组下标超出范围的概念。
- 当使用了超出范围的下标时，JavaScript不会报错，引用的数组元素相当于未声明的变量，其值为undefined。
- 对超出范围的下标引用的数组元素赋值时，会为数组添加数组元素。

4. 使用数组赋值

- JavaScript允许将数组赋给变量。将引用数组的变量赋给另一个变量时，使得两个变量引用同一个数组。
- 将另一个数组赋给变量意味着改变了变量的引用，使其引用新的数组。

5. 添加、删除数组元素

- JavaScript中的数组长度是不固定的，对不存在的数组元素赋值时，会将其添加到数组中。
  - `var a = new Array()` //创建一个空数组
  - `a[0] = 1` //添加数组元素
  - `a[1] = 2`

- delete关键字可用于删除数组元素。
  - `delete  a[1]  //删除a[1]`
- 需注意的是，delete的实质是删除变量所引用的内存单元。
- 使用delete删除一个数组元素后，数组的大小不会改变。
  引用一个被删除的数组元素，得到的值为undefined。

6. 数组迭代

- 数组通常结合循环实现数组迭代（或者叫数组元素遍历）。
- 数组下标最小值为0，最大值为数组长度减1
- 对数组a用for循环“for (var i = 0; i < a.length; i++)”即可实现数组迭代。
- 如果数组元素已经使用delete删除，或者通过赋值语句给一个下标较大的不存在的数组元素赋值，就会导致数组包含一些不存在的元素。
- 使用for/in循环可忽略不存在的元素。

3.1.3　数组的属性

- 1. length属性
- 数组的length属性用于获得数组长度，例如，a.length获得数组a的长度。
- JavaScript数组的长度是可变的，通过为不存在的数组元素赋值的方式添加数组元素时，数组的长度也随之变化。
  - `var a = new Array(1, 2, 3)　　//创建数组，数组长度为3`
  - `a[5] = 10　　//添加一个数组元素，数组长度变为6`

- 数组长度为数组中元素的个数。因为数组元素下标从0开始，所以数组下标范围为0到数组长度减1。
- JavaScript允许修改length属性。
  - a.length=5
- 上面的语句将数组a的长度修改为5。
- 如果修改后的长度小于原来长度，超出新长度的数组元素将丢失。如果新长度超出原长度，增加的数组元素初始值为undefined。

2. prototype属性

- 对象的prototype属性用于为对象添加自定义的属性或方法。为数组添加自定义属性或方法的基本语法格式如下。
  - `Array.prototype.name = value`
- 其中，name为自定义的属性或方法名称，value为表达式或者函数。自定义属性和方法对当前HTML文档中的所有数组有效。

3.1.4　操作数组的方法

- 1. 连接数组
- join()方法用于将数组中的所有元素连接成一个字符串，字符串中的各个数据默认用逗号分隔。
- 也可为join()方法指定一个字符串作为分隔符。
- 基本语法格式为如下。
  - a.join()　　//将数组a中的数据连接成逗号分隔的字符串
  - a.join(x)　　//将数组a中的数据连接成变量x中的字符串分隔的字符串

2. 逆转元素顺序

- reverse()方法将数组元素以相反的顺序存放。基本语法格式如下。
  - a.reverse()

3. 数组排序

- sort()方法用于对数组排序。默认情况下，数组元素按字母顺序排列，数值会转换为字符串进行排序。
- 如果要对数字数组进行排序，可以为sort()方法提供一个排序函数作为参数，排序函数定义数组中两个相邻元素的相对顺序。
- 排序函数有两个参数，假设为x和y。若需x排在y之前，则排序函数应返回一个小于0的值。若需x排在y之后，则排序函数应返回一个大于0的值。若两个参数的位置无关紧要，排序函数返回0。
- 使用排序函数时，sort方法将两个数组元素作为排序函数的参数，根据排序函数的返回值决定数组元素的先后顺序。

4. 取子数组

- slice()方法用于从数组中取子数组，其基本语法格式如下。
  - 数组名.slice(x,y)
- 从数组中返回下标范围为x～y-1的子数组。
- 如果省略y，则返回从x开始到最后的全部数组元素。
- 如果x或y为负数，则作为最后一个元素的相对位置，如-1为倒数第1个元素位置。

5. splice()方法

- splice()方法用于添加或删除数组元素，其基本语法格式如下。
  - 数组名.splice(m,n,x1,x2,...)
- 其中，m为开始元素下标，n为从数组中删除的元素个数。x1、x2等是要添加到数组中的数据，可以省略。
- splice()方法同时会返回删除的数组元素。

6. push()和pop()方法

- push()和pop()方法用于实现数组的堆栈操作（先进后出）。
- push()方法将数据添加到数组末尾，返回数组长度。
- pop()方法返回数组中的最后一个元素，数组长度减1。

7. unshift()和shift()方法

- unshift()和shift()方法用于实现数组的队列操作（先进先出）。
- unshift()方法将数据添加到数组开头，并返回新的数组长度。
- shift()方法返回数组中的第一个元素，所有数组元素依次前移一位，数组长度减1。

8. concat()方法

- concat()方法用于将提供的数据合并成一个新的数组，其基本语法格式如下。
  - b = a.concat(x1,x2,x3,…)
- 其中，x1、x2、x3等是单个的数据或者数组变量。如果是数组变量，则将其中的数据合并到新数组中
- 。变量b保存合并后的新数组。

3.2 函数

- 当某一段代码需要重复使用，或者需要对批量数据执行相同操作时，就可使用函数来完成。

3.2.1　定义函数

- 1. 使用function关键字定义函数
- 用function关键字定义函数的基本语法格式如下。

```
function　函数名([参数1　，　参数2　，　...]){
                代码块
                [return　返回值]
}
```

- 在当前作用域内，函数名应该是唯一的。
- 函数参数是可选的，多个参数之间用逗号分隔。
- 花括号中的代码块称为函数体。
- 在函数体中或在函数末尾，可使用return语句指定函数返回值。
- 返回值可以是任意的常量、变量或者表达式。没有return语句时，函数没有返回值。

- 例如，下面的函数用于计算两个数的和。

```
function sum(a, b) {
    return a + b
}
```

2. 在表达式中定义函数

- JavaScript允许在表达式中定义函数。在表达式中定义求和函数如下。
  - `var sum2 = function (a, b) {`
  - `return a + b`
  - `}`
- 这里的function关键字定义了一个匿名函数（也可称未命名函数），JavaScript将其赋值给变量sum2。通过sum2可调用对应的匿名函数。

3．使用Function()构造函数

- 在JavaScript中，函数也是一种对象。函数对象的构造函数为Function()，可用它来定义函数，其基本语法格式如下。
  - var 变量 = new Function("参数1","参数2",……,"函数体")
- 例如，下面为用构造函数定义求和函数。
  - var sum3 = new Function("a","b","return a+b")

4. 箭头函数

- JavaScript允许使用箭头“=>”来定义函数表达式——箭头函数。箭头左侧为参数，右侧为函数体。
  - `var sum = (x, y) => { return x + y }  //定义函数`
  - `sum(2, 5)`
    `//函数返回值为7`
- 箭头函数的函数体通常为一个return语句。如果要返回对象常量，可将对象常量放在return语句或者一对圆括号中。

- var fruit = (x, y) => { return { type: x, price: y } }
  //返回对象常量，标准定义
- var fruit2 = (x, y) => ({ type: x, price: y })
  //返回对象常量，简略定义
- a = fruit('apple', 5)
  //a={ type: 'apple', price: 5 }
- b = fruit2('pear', 4)

3.2.2　调用函数

- 函数调用的基本语法格式如下。
  - 函数名(参数)
- 函数名是function关键字定义的函数名称，或者是引用函数对象的变量名称。
- 使用function关键字定义函数时，函数定义可以放在当前HTML文档中的任意位置，即允许函数的调用出现在函数定义之前。使用Function()构造函数定义函数时，只能在定义之后通过变量名来调用函数。
- 函数可以在脚本中调用，也可以作为HTML的事件处理程序或URL。

3.2.3　带参数的函数

- 定义函数时指定的参数称为形式参数，简称形参。
- 调用函数时指定的参数称为实际参数，简称实参。
- 在调用函数时，实参按先后顺序一一对应地传递给形参。
- JavaScript是弱类型的，形参不需要指定数据类型。
- JavaScript不会检查形参和实参的数据类型，也不会检查形参和实参的个数。

1. 函数参数的个数

- 函数的length属性返回形参的个数。在函数内部，arguments数组保存调用函数时传递的实参。

```
function getMax(a, b) {
    var max = Number.MIN_VALUE
    var len = arguments.length
                    //获得实际参数个数
```

2．使用数组作为参数

- 在使用表达式作为实参时，形参接收实参的值，所以形参值的改变不会影响到实参。
- 在使用数组作为实参时，形参接收的是数组的引用，即形参和实参引用了同一个数组。
- 这种情况下，通过形参改变数组元素的值后，在函数外通过实参获得的也是改变后的数组元素值。

3．使用对象作为参数

- 对象也可作为函数参数。与数组类似，形参和实参引用的是同一个对象。
- 如果在函数中通过形参修改了对象属性值，通过实参获得的也是修改后的对象属性值。

3.2.4　嵌套函数

- 在函数内部定义的函数称为嵌套函数，嵌套函数只能在当前函数内部使用。

```
function addArray(a, b) {
        function getMax(x, y) { return x > y ? 0 : 1 }
                                        //返回长度较大的数组的序号
        var alen = a.length
        var blen = b.length
        ……
```

3.2.5　递归函数

- 递归函数是指在函数的内部调用函数自身，形成递归调用。
- 使用递归函数必须注意递归调用的结束条件，若递归调用无法停止，则会导致运行脚本的浏览器崩溃。

```
function  fact(n)  {     //计算阶乘
                    if  (n  <=  1)
                                return  1     //递归调用结束
                    return  n  *  fact(n  -  1)  }
```

3.3 内置函数

- 1. alert()函数

- alert()函数用于显示警告对话框，对话框包括一个“确定”按钮。