# Array.join();

将数组中的所有元素都转化为字符串并连接在一起，返回最后生成的字符串；可以指定一个可选的字符串在生成的字符串中来分隔数组的各个元素；

Array.join()是String.split()方法的逆向操作，后者是将字符串分割成若干块来创建一个数组；

# Array.reverse()

将数组中的元素颠倒顺序，返回逆向的数组。不通过重新排列的方式来创建新数组，而是直接替换原有数组。

# Array.sort()

将数组中的元素排序并返回排序后的数组。不通过重新排列的方式来创建新数组，而是直接替换原有数组。

当不带参数调用sort()方法时，数组元素以字母表顺序排列

# Array.concat()

将创建并返回一个新数组，它的元素包括调用concat()方法的原始数组和concat()的每个参数。如果这些参数中的任何一个自身是数组，则连接是数组元素而非数组本身。但是concat()不会递归扁平化数组，也不会修改调用concat()的数组

# Array.slice()

将返回指定数组的一个片段或子数组。它的两个参数分别指定了片段开始和结束的位置。返回数组包含第一个参数指定的位置和所有但不包含第二个参数指定的位置之间的所有数组元素。如果参数中出现负数，它表示相对于数组中最后一个的位置。-1表示数组中最后一个元素。

# Array.splice()

在数组中插入或删除元素的通用方法。

不同于slice(),concat().splice()会修改调用的数组。

Splice()能够从数组中删除元素、插入元素到数组中或者同时完成这两种操作。在插入或删除点之后的数组元素会根据需要增加或减少它们的索引值，因此数组的其他部分任然保持连续。Splice()第一个参数指定了插入或删除的起始位置。第二个参数指定了删除的元素个数。如果省略第二个参数那么将删除从起始点到元素结尾的所有元素。Splice()返回一个由删除元素组成的数组，如果没有删除元素那么返回一个空数组。Splice()前两个参数指定了需要删除的数组元素，后面的任意个参数指定了插入到数组中参数。

# Array.push(),Array.pop()

Push()方法在数组尾部添加一个或多个元素并返回数组的新长度。Pop()方法相反删除数组的最后一个元素并返回它删除的值。这两个方法都修改并替换原始数组而非生成一个修改后的新数组。

# unshift(),shift()

unshift()在数组的头部添加一个或个元素，并将已存在的元素移动到更高的索引位置来获取足够的空间，最后返回数组的新长度shift()方法删除数组的第一个元素并将其返回，然后把随后的所有元素下移一个位置来填补数组头部的空缺。

# Array.toString()

将数组的每个元素转化为字符串并且输出用逗号分隔的字符串列表。

# ECMAScript5新方法

ECMAScript5数组方法。大多数的方法接收一个函数，并且对数组的每一个元素（或一些元素）调用一次改函数。如果是稀疏数组，对不存在的元素不调用传递的函数。在大多数情况下调用提供的函数使用三个参数：数组元素、元素的索引、数组本身。通常只需要第一个参数。大多数ECMAScript5数组方法的第一个参数是一个函数，第二个参数是可选的。如果有第二个参数，则调用的函数被看做是第二个参数的方法。也就是说，在调用函数时传递进去第二个参数作为它的this关键字的值来使用。被调用函数的返回值是非常重要，但不同方法处理返回值的方式也不同。ECMAScript5数组方法都不会修改它们调用的原始数组

# Array.forEach()

从头到尾遍历数组，为每个元素调用指定的函数。

forEach()无法在所有元素都传递给调用的函数之前终止遍历。也就是说没有像for循环中用break来提前终止必须把forEach()方法放在一个try块中，并能抛出一个异常。如果forEach()调用的函数抛出foreach.break异常，循环会提前终止。

# Array.map()

将调用的数组的每个元素传递给指定的函数，并返回一个数组

# Array.filter()

返回的数组元素是调用数组的一个子集。传递的函数是用来逻辑判断的；如果返回值为true，那么传递给判定函数的元素就是这个子集的成员，它将被添加到一个作为返回值的数组中

Array.filter()会跳过稀疏数组中缺少的元素，它返回的总是稠密数组

# Array.every()和Array.some()

它们对数组元素应用指定的函数进行判断，返回false或true；

Array.every()仅当数组中所有的元素调用判定函数都返回true，它才返回true；

Array.some()当数组中至少有一个元素调用判定函数返回true，它就返回true；当数组中所有元素调用判定函数都返回false时，它才返回false。

一旦every(),some()确定该返回什么值时它们就会停止遍历数组

空数组调用every()返回true, some()返回false。

# Array.reduce()和Array.reduceRight()

使用指定的函数将元素进行组合，生成单个值。

reduce()需要2个参数，第一个是执行化简操作的函数（用某种方法把两个值化简为一个值，并返回简化后的值）。第二个是传递给函数的初始值

在空数组上不带初始值参数调用reduce()方法将导致类型错误异常。如果调用它的时候只有一个值-数组只有一个元素并且没有指定初始值，或者有一个空数组并且指定一个初始值-reduce()只是简单的返回那个值并不会调用化简函数。

reduceRight()工作原理和reduce()一样，不同的是他是按照数组索引从高到低（从右到左）处理数组。

数学计算并不是reduce()和reduceRight()的唯一意图。它可以用来计算两个对象的并集并返回另一个新对象，新对象具有二者的属性。详情参考page159

# Array.indexOf(),Array.lastIndexOf()

Array.indexOf(),Array.lastIndexOf()搜索整个数组中具有给定值的元素，返回找到的第一个元素的索引或者没找到就返回-1.indexOf()从头至尾搜索lastIndexOf()则相反。

indexOf()第一个参数是需要索引的值，第二个参数是可选的；它用来指定数组中的一个索引，从那里开始搜索。如果省略则从头开始搜索lastIndexOf()则从尾开始搜索，也可以是负数代表数组末尾的偏移量。同splice()方法

# 数组对象的特性

* 当有新的元素添加到列表中时，自动更新length属性
* 设置length为一个较小值时截断数组
* 从Array.prototype中继承一些有用的方法
* 其类属性为“Array”

Javascript数组方法特意定义为通用的，因此他们不仅应用在真正的数组而且在类数组对象上都能正确的工作。在ECMAScript5中，所有的数组方法都是通用的，在ECMAScript3中除了toString()和toLocalString()以外其他的方法也是通用的(concat()方法是一个特例，虽然可以用在类数组对象上，但是它没有将那个对象扩充进返回的数组中)既然数组对象没有继承自Array.prototype那就不能在他们上面直接调用数组方法，可以间接的使用Function.call方法例如Array.prototype.join.call()；