JavaScript

千锋教育
QIAN FENG EDUCATION

做真实的自己，用良心做教育

张骏
H5教学部

ES5数组api
String

课程概要：

ES5新增数组api
String对象、属性
String常用api

Array.prototype.indexOf()

- **语法**
- arr.indexOf(searchElement)
- arr.indexOf(searchElement[, fromIndex = 0])
- **参数**
- searchElement
- 要查找的元素
- fromIndex
- 开始查找的位置。如果该索引值大于或等于数组长度，意味着不会在数组里查找，返回-1。如果参数中提供的索引值是一个负值，则将其作为数组末尾的一个抵消，即-1表示从最后一个元素开始查找，-2表示从倒数第二个元素开始查找，以此类推。 注意：如果参数中提供的索引值是一个负值，并不改变其查找顺序，查找顺序仍然是从前向后查询数组。如果抵消后的索引值仍小于0，则整个数组都将会被查询。其默认值为0.
- **返回值**
- 首个被找到的元素在数组中的索引位置; 若没有找到则返回 -1

Array.prototype.forEach()

- **语法**
- array.forEach(callback[, thisArg])
- **参数**
- callback为数组中每个元素执行的函数，该函数接收三个参数：
  - currentValue数组中正在处理的当前元素。
  - index可选数组中正在处理的当前元素的索引。
  - array可选forEach()方法正在操作的数组。
- thisArg可选可选参数。当执行回调 函数时用作this的值(参考对象)。
- **返回值**
- undefined

Array.prototype.filter()

- **语法**
- var new_array = arr.filter(callback(element[, index[, array]])[, thisArg])
- **参数**
- callback用来测试数组的每个元素的函数。调用时使用参数 (element, index, array)。返回true表示保留该元素（通过测试），false则不保留。它接受三个参数：
  - **element**当前在数组中处理的元素。
  - **index**可选正在处理元素在数组中的索引。
  - **array**可选调用了filter的数组。
- thisArg可选可选。执行 callback 时的用于 this 的值。
- **返回值**
- 一个新的通过测试的元素的集合的数组，如果没有通过测试则返回空数组

Array.prototype.map()

- **语法**
- var new_array = arr.map(callback(currentValue[, index[, array]]) {

```
        // Return element for new_array
}[, thisArg])
```

- **参数**
- callback生成新数组元素的函数，使用三个参数：
  - currentValuecallback 数组中正在处理的当前元素。
  - index可选callback 数组中正在处理的当前元素的索引。
  - array可选callback map 方法被调用的数组。
- thisArg可选执行 callback 函数时使用的this 值。
- **返回值**
- 一个新数组，每个元素都是回调函数的结果。
- **描述**
- 遍历数组 --- 修改(操作)数组 --- 返回数组

Array.prototype.reduce()

- **语法**
- var result = arr.reduce(callback(pre,next){

  return val;
- });
- **参数**
- callback函数遍历数组的函数有两个主要参数pre和next：
  - pre 默认情况是数组的第一个元素，遍历开始后为上一次callback的执行结果
  - next 数组的下一个值
- **返回值**
- 返回最后一次callback的执行结果
- **描述**
- 归并数组

String

- String类型是字符串的对象包装类型，可以使用构造函数来创建。
- String对象的方法也可以在所有基本的字符串值中访问到。其中，继承的valueOf() toLocale-String() 和toString()方法，都返回对象所表示的基本字符串值。

String.length

- **语法**
- *string*.length
- **描述**
- 属性String.length是一个只读整数，它声明了指定字符串string中的字符数。对于任何一个字符串s，它最后一个字符的下标都是s.length-1。用for/in循环不能枚举出字符串的length属性，用delete运算符也不能删除它。

String.charAt()

- **语法**
- *string*.charAt(*n*)
- **参数**
- *n* 应该返回的字符在string中的下标。
- **返回值**
- 字符串string的第n个字符。
- **描述**
- 方法String.charAt()返回字符串string中的第n个字符。字符串中第一个字符的下标值是0。如果参数n不在0和string.length-1之间，该方法将返回一个空字符串。

String.charCodeAt()

- **语法**
- *string*.charCodeAt(*n*)
- **参数**
- *n* 返回编码的字符的下标。
- **返回值**
- string中的第n个字符的Unicode编码。这个返回值是0 ~ 65535之间的16位整数。
- **描述**
- 方法charCodeAt()与charAt()执行的操作相似，只不过前者返回的是位于指定位置的字符的编码，而后者返回的则是含有字符本身的子串。如果n是负数，或者大于等于字符串的长度，则charCodeAt()返回NaN。

String.concat()

- **语法**
- *string*.concat(*value*, ...)
- **参数**
- *value, ...* 要连接到string上的一个或多个值。
- **返回值**
- 把每个参数都连接到字符串string上得到的新字符串。
- **描述**
- 方法concat()将把它的所有参数都转换成字符串(如果必要)，然后按顺序连接到字符串string的尾部，返回连接后的字符串。注意，string自身并没有被修改。
- String.concat()与Array.concat()很相似。注意，使用 “+” 运算符来进行字符串的连接运算通常更简便一些。

String.indexOf()

- **语法**
- *string*.indexOf(*substring*)
- *string*.indexOf(*substring*,*start*)
- **参数**
- *substring* 要在字符串string中检索的子串。
- *start* 一个可选的整数参数，声明了在字符串String中开始检索的位置。它的合法取值是0(字符串中的第一个字符的位置)到string.length-1(字符串中的最后一个字符的位置)。如果省略了这个参数，将从字符串的第一个字符开始检索。
- **返回值**
- 如果在string中的start位置之后存在substring返回出现的第一个substring 的位置。如果没有找到子串substring返回-1。
- **描述**
- 方法string.indexOf()将从头到尾的检索字符串string，看它是否含有子串 substring。开始检索的位置在字符串string的start处或string的开头(没有 指定start参数时)。如果找到了一个substring那么String.indexOf()将返回 substring的第一个字符在string中的位置。string中的字符位置是从0开始的。

String.lastIndexOf()

- **语法**
- *string*.lastIndexOf(*substring*)
- *string*.lastIndexOf(*substring*, *start*)
- **参数**
- *substring* 要在字符串string中检索的子串。
- *start* 一个可选的整数参数，声明了在字符串string中开始检索的位置。它的合法取值是0(字符串中的第一个字符的位置)到string.1ength-1(字符串中的最后一个字符的位置)。如果省略了这个参数，将从字符串的最后一个字符处开始检索。
- **返回值**
- 如果在string中的start位置之前存在substring那么返回的就是出现的最后一个substring的位置。如果没有找到子串substring那么返回的是-1。
- **描述**
- 方法String.1astIndexOf()将从尾到头的检索字符串string看它是否含有子串 substring。开始检索的位置在字符串string的start处或string的结尾(没有 指定start参数时)。如果找到了一个substring，那么String.lastIndexOf()将返回substring的第一个字符在string中的位置。由于是从尾到头的检索一个字符串，所以找到的第一个substrlng其实是strlng中出现在位置start之前的最后一个substring。
- 如果在string中没有找到substring，那么该方法将返回-1。

String.match()

- **语法**
- *string*.match(*regexp*)
- **参数**
- *regexp* 声明了要匹配的模式的RegExp对象。如果该参数不是RegExp对象，则首先将把它传递给RegExp()构造函数，把它转换成RegExp对象。
- **返回值**
- 存放匹配结果的数组。该数组的内容依赖于regexp是否具有全局性质g。下面详细说明了这个返回值。
- **描述**
- 方法match()将检索字符串string，以找到一个或多个与regexp匹配的文本。这个方法的行为很大程度上依赖于regexp是否具有性质g。
- **示例**
- var *a*= "1 plus 2 equals 3";
  a.match(/\d+/g);

String.search()

- **语法**
- *string*.search(*regexp*)
- **参数**
- *regexp* 要在字符串string中检索的RegExp对象，该对象具有指定的模式。如果该参数不是RegExp对象，则首先将它传递给RegExp()构造函数，把它转换成 RegExp对象。
- **返回值**
- string中第一个与regexp相匹配的子串的起始位置。如果没有找到任何匹配的子 串，则返回-1。
- **描述**
- 方法search()将在字符串string中检索与regexp相匹配的子串，并且返回第一个匹配子串的第一个字符的位置。如果没有找到任何匹配的子串，则返回-1。
- **示例**
- var s = "JavaScript is fun"; s.search(/script/i) // 返回 4

String.replace()

- **语法**
- *string*.replace(*regexp*, *replacement*)
- **参数**
- *regexp* 声明了要替换的模式的RegExp对象。如果该参数是一个字符串，则将它作为要检索的直接量文本模式，而不是首先被转换成RegExp对象。
- *replacement*
- 一个字符串，声明的是替换文本或生成替换文本的函数。详见描述部分。
- **返回值**
- 一个新字符串，是用replacemenc替换了与regexp的第一次匹配或所有匹配之后得到的。
- **描述**
- 字符串string的方法replace()执行的是查找并替换的操作。它将在string中查找与regexp相匹配的子串，然后用replacement替换这些子串。如果regexp具有全局性质g，那么replace()将替换所有的匹配子串。否则，它只替换第一个匹配子串。

String.slice()

- **语法**
- *string*.slice(*start*, *end*)
- **参数**
- *start*
- 要抽取的片段的起始下标。如果是负数，那么该参数声明了从字符串的尾部开始算起的位置。也就是说，-1指字符串中的最后一个字符，-2指倒数第二个字符，以此类推。
- *end*
- 紧接着要抽取的片段的结尾的下标。如果没有指定这一参数，那么要抽取的子串包括start到原字符串结尾的字符串。如果该参数是负数，那么它声明了从字符串的尾部开始算起的位置。
- **返回值**
- 一个新字符串，包括字符串string从start开始(包括start)到end为止(不包括end)的所有字符。
- **描述**
- 方法slice()将返回一个含有字符串string的片段的字符串或返回它的一个子串。 但是该方法不修改string。

String.split()

- **语法**
- *string*.split(*delimiter*, *limit*)
- **参数**
- *delimiter*
- 字符串或正则表达式，从该参数指定的地方分割string。
- *limit*
- 这个可选的整数指定了返回的数组的最大长度。
- **返回值**
- 一个字符串数组，是通过在delimiter指定的边界处将字符串string分割成子串创建的。返回的数组中的子串不包括delimiter自身。
- **描述**
- 方法split()将创建并返回一个字符串数组，该数组中的元素是指定的字符串string 的子串，最多具有limit个。
- **示例**
- "1:2:3:4:5".split(":"); // 返回 ["1","2","3","4","5"]
- "|a|b|c|".split("|"); // 返回 ["", "a", "b", "c", ""]

String.substr()

- **语法**
- *string*.substr(*start*, *length*)
- **参数**
- *start*
- 要抽取的子串的起始下标。如果是一个负数，那么该参数声明从字符串的尾部开始算起的位置。也就是说，-1指字符串中的最后一个字符，-2指倒数第二个字符，以此类推。
- *length*
- 子串中的字符数。如果省略了这个参数，那么返回从string的开始位置到结尾的子串。
- **返回值**
- 一个字符串的副本，包括从string的start处(包括start所指的字符)开始到length个字符。如果没有指定length，返回的字符串包含从start到string结尾的字符。
- **描述**
- substr()将在string中抽取并返回一个子串。但是它并不修改string。

String.substring()

- **语法**
- *string*.substring(*from*, *to*)
- **参数**
- *from*
- 一个整数，声明了要抽取的子串的第一个字符在string中的位置。
- *to*
- 一个可选的整数，比要抽取的子串的最后一个字符在string中的位置多1。如果省略了该参数，返回的子串直到字符串的结尾。
- **返回值**
- 一个新字符串，其长度为to-from，存放的是字符串string的一个子串。这个新字符串含有的字符是从string中的from处到to-1处复制的。
- **描述**
- String.substring()将返回字符串string的子串，由from到to之间的字符构成， 包括位于from的字符，不包括位于to的字符。

String.toLowerCase()

- **语法**
- *string*.toLowerCase( )
- **返回值**
- string的一个副本，其中所有大写字符都被转换成了小写字符。

String.toUpperCase()

- **语法**
- *string*.toUpperCase( )
- **返回值**
- string的一个副本，其中所有小写字符都被转换成了大写的。

THANK YOU

千锋教育
QIANFENG EDUCATION

做真实的自己，用良心做教育