12 个 JavaScript 技巧

作者：virtualpier
原文链接：https://ld246.com/article/1463055776333
来源网站：链滴

在这篇文章中将给大家分享12个有关于JavaScript的小技巧。这些小技巧可能在你的实际工作中许能帮助你解决一些问题。
<h2>使用<code>!!</code>操作符转换布尔值</h2>
有时候我们需要对一个变量查检其是否存在或者检查值是否有一个有效值，如果存在就返回<cod>true</code>值。为了做这样的验证，我们可以使用<code>!!</code>操作符来实现是非常的方与简单。对于变量可以使用<code>!!variable</code>做检测，只要变量的值为:<code>0</code><code>null</code>、<code>" "</code>、<code>undefined</code>或者<code>NaN</co>e>都将返回的是<code>false</code>，反之返回的是<code>true</code>。比如下面的示例：
<pre class="brush: js">function Account(cash) {
  this.cash = cash;
  this.hasMoney = !!cash;
}
var account = new Account(100.50);
console.log(account.cash); // 100.50
console.log(account.hasMoney); // true

var emptyAccount = new Account(0);
console.log(emptyAccount.cash); // 0
console.log(emptyAccount.hasMoney); // false</pre>

在这个示例中，只要<code>account.cash</code>的值大于<code>0</code>，那么<code account.hasMoney</code>返回的值就是<code>true</code>。
<h2>使用<code>+</code>将字符串转换成数字</h2>
这个技巧非常有用，其非常简单，可以交字符串数据转换成数字，不过其只适合用于字符串数据否则将返回<code>NaN</code>，比如下面的示例：
<pre class="brush: js">function toNumber(strNumber) {
  return +strNumber;
}
console.log(toNumber("1234")); // 1234
console.log(toNumber("ACB")); // NaN</pre>
这个也适用于<code>Date</code>，在本例中，它将返回的是时间戳数字：
<pre class="brush: js">console.log(+new Date()) // 1461288164385</pre>
<h2>并条件符</h2>
如果你有一段这样的代码：
<pre class="brush: js">if (conected) {
  login();
}</pre>
你也可以将变量简写，并且使用<code>&&</code>和函数连接在一起，比如上面示例，可以简写成这样：
<pre class="brush: js">conected && login();</pre>
如果一些属性或函数存在于一个对象中，你也可以这样做检测，如下面的代码所示：
<pre class="brush: js">user && user.login();</pre>
<h2>使用<code>||</code>运算符</h2>
在ES6中有默认参数这一特性。为了在老版本的浏览器中模拟这一特性，可以使用<code>||</co>e>操作符，并且将将默认值当做第二个参数传入。如果第一个参数返回的值为<code>false</code>，那么第二个值将会认为是一个默认值。如下面这个示例：
<pre class="brush: js">function User(name, age) {
  this.name = name || "Oliver Queen";
  this.age = age || 27;
}
var user1 = new User();

原文链接：12 个 JavaScript 技巧

```
console.log(user1.name); // Oliver Queen
console.log(user1.age); // 27

var user2 = new User("Barry Allen", 25);
console.log(user2.name); // Barry Allen
console.log(user2.age); // 25</pre>

<h2>在循环中缓存<code>array.length</code></h2>
<p>这个技巧很简单，这个在处理一个很大的数组循环时，对性能影响将是非常大的。基本上，大家会写一个这样的同步迭代的数组：</p>
<pre class="brush: js">for(var i = 0; i &lt; array.length; i++) {
    console.log(array[i]);
}</pre>
<p>如果是一个小型数组，这样做很好，如果你要处理的是一个大的数组，这段代码在每次迭代都将重新计算数组的大小，这将会导致一些延误。为了避免这种现象出现，可以将<code>array.length</ode>做一个缓存：</p>
<pre class="brush: js">var length = array.length;
for(var i = 0; i &lt; length; i++) {
    console.log(array[i]);
}</pre>
<p>你也可以写在这样：</p>
<pre class="brush: js">for(var i = 0, length = array.length; i &lt; length; i++) {
    console.log(array[i]);
}</pre>
<h2>检测对象中属性</h2>
<p>当你需要检测一些属性是否存在，避免运行未定义的函数或属性时，这个小技巧就显得很有用。果你打算定些一些跨兼容的浏览器代码，你也可能会用到这个小技巧。例如，你想使用<code>docu ent.querySelector()</code>来选择一个<code>id</code>，并且让它能兼容IE6浏览器，但是在IE浏览器中这个函数是不存在的，那么使用这个操作符来检测这个函数是否存在就显得非常的有用，如面的示例：</p>
<pre class="brush: js">if ('querySelector' in document) {
    document.querySelector("#id");
} else {
    document.getElementById("id");
}</pre>
<p>在这个示例中，如果<code>document</code>不存在<code>querySelector</code>函数那么就会调用<code>docuemnt.getElementById("id")</code>。</p>
<h2>获取数组中最后一个元素</h2>
<p><code>Array.prototype.slice(begin,end)</code>用来获取<code>begin</code>和<code end</code>之间的数组元素。如果你不设置<code>end</code>参数，将会将数组的默认长度值作<code>end</code>值。但有些同学可能不知道这个函数还可以接受负值作为参数。如果你设置个负值作为<code>begin</code>的值，那么你可以获取数组的最后一个元素。如：</p>
<pre class="brush: js">var array = [1,2,3,4,5,6];
console.log(array.slice(-1)); // [6]
console.log(array.slice(-2)); // [5,6]
console.log(array.slice(-3)); // [4,5,6]</pre>
<h2>数组截断</h2>
<p>这个小技巧主要用来锁定数组的大小，如果用于删除数组中的一些元素来说，是非常有用的。例，你的数组有<code>10</code>个元素，但你只想只要前五个元素，那么你可以通过<code>array. ength=5</code>来截断数组。如下面这个示例：</p>
<pre class="brush: js">var array = [1,2,3,4,5,6];
console.log(array.length); // 6
array.length = 3;
```

console.log(array.length); // 3
console.log(array); // [1,2,3]</pre>
<h2>替换所有</h2>
<code>String.replace()</code>函数允许你使用字符串或正则表达式来替换字符串，本身这函数只替换第一次出现的字符串，不过你可以使用正则表达多中的<code>/g</code>来模拟<code>eplaceAll()</code>函数功能：
<pre class="brush: js">var string = "john john";
console.log(string.replace(/hn/, "ana")); // "joana john"
console.log(string.replace(/hn/g, "ana")); // "joana joana"</pre>
<h2>合并数组</h2>
如果你要合并两个数组，一般情况之下你都会使用<code>Array.concat()</code>函数：
<pre class="brush: js">var array1 = [1,2,3];
var array2 = [4,5,6];
console.log(array1.concat(array2)); // [1,2,3,4,5,6];</pre>
然后这个函数并不适合用来合并两个大型的数组，因为其将消耗大量的内存来存储新创建的数组在这种情况之个，可以使用<code>Array.pus().apply(arr1,arr2)</code>来替代创建一个新数组。种方法不是用来创建一个新的数组，其只是将第一个第二个数组合并在一起，同时减少内存的使用：
<pre class="brush: js">var array1 = [1,2,3];
var array2 = [4,5,6];
console.log(array1.push.apply(array1, array2)); // [1,2,3,4,5,6];</pre>
<h2>将<code>NodeList</code>转换成数组</h2>
如果你运行<code>document.querySelectorAll(“p”)</code>函数时，它可返回DOM元素的数组，也就是<code>NodeList</code>对象。但这个对象不具有数组的函数功能比如<code>sort()</code>、<code>reduce()</code>、<code>map()</code>、<code>filter(</code>等。为了让这些原生的数组函数功能也能用于其上面，需要将节点列表转换成数组。可以使<code>[].slice.call(elements)</code>来实现：
<pre class="brush: js">var elements = document.querySelectorAll("p"); // NodeList
var arrayElements = [].slice.call(elements); // Now the NodeList is an array
var arrayElements = Array.from(elements); // This is another way of converting NodeList to Array</pre>
<h2>数组元素的洗牌</h2>
对于数组元素的洗牌，不需要使用任何外部的库，比如Lodash，只要这样做：
<pre class="brush: js">var list = [1,2,3];
console.log(list.sort(function() { Math.random() - 0.5 })); // [2,1,3]</pre>
<h2>总结</h2>
现在你学会了些有用的JavaScript小技巧。希望这些小技巧能在工作中帮助你解决一些麻烦，或说这篇文章对你有所帮助。如果你有一些优秀的JavaScript小技巧，欢迎在评论中与我们一起分享。

转载自：http://www.w3cplus.com/javascript/12-extremely-useful-hacks-fo-javascript.html