javascript检查属性是否存在(hasOwnProperty和propertyIsEnumerable的区别)

作者：crick77
原文链接：https://ld246.com/article/1450921621161
来源网站：链滴

<h3 id="起因">起因</h3>
JavaScript 对象就是各种属性的集合，业务场景为接收用户传来的 options 参数，如果默认 defaultOptions 存在传入的参数属性，则进行替换。所以存在一个检查对象是否包含某个属性的操作。
在原型属性中找到了 hasOwnProperty 和 propertyIsEnumerable 两个方法。那么这两个方法有什区别呢？各自的使用场景又是什么呢？
<h3 id="hasOwnProperty">hasOwnProperty</h3>
hasOwnProperty()方法可以接收一个字符参数，用来判断对象中是否存在以字符参数命名的属。它只会查找自身属性，不会根据原型链进行查找。
注：原型链会再另一篇文章中做详细的介绍！
<pre><code class="highlight-chroma">var obj = {x:"1"};
obj.y = function(){}

console.log(obj.haOwnProperty("x")); //true
console.log(obj.haOwnProperty("y")); //true 方法也是属性
console.log(obj.haOwnProperty("z")); //false 属性不存在
console.log(obj.haOwnProperty("toString")); //false hasOwnProperty是继承Object的属性，自身属性中不存在
</code></pre>
<h3 id="propertyIsEnumerable">propertyIsEnumerable</h3>
propertyIsEnumerable()方法在 hasOwnProperty()的基础上，校验属性是否为枚举属性。也就说属性必须满足为自身属性且为枚举属性时，才会返回 true。
<h4 id="枚举属性">枚举属性</h4>
枚举属性，其实是可被枚举，表示该属性可以在对象中被遍历。JavaScript 属性默认均为枚举属。如
<pre><code class="highlight-chroma">var obj = {x:"1"};
Object.defineProprty(obj, 'y', {value : '2', enumerable : true });
Object.defineProprty(obj, 'z', {value : '3', enumerable : false });
console.log(obj);//Object {x: "1", y: "2", z: "3"}
for (var i in obj) {
    console.log(i); //x,y
}
</code></pre>
上述代码中属性 z 被手动指定为非枚举属性，因此没有被遍历。
结合上面的 2 例子，可以清晰的知道 propertyIsEnumerable()函数的作用：
<pre><code class="highlight-chroma">var obj = {x:"1"};
obj.y = function(){}

Object.defineProprty(obj, 'z', {value : '2', enumerable : true });
Object.defineProprty(obj, 'w', {value : '3', enumerable : false });
console.log(obj.propertyIsEnumerable("x")); //true 属性默认为枚举属性</code></pre>

console.log(obj.propertyIsEnumerable("y")); //true 方法也是属性
console.log(obj.propertyIsEnumerable("z")); //true
console.log(obj.propertyIsEnumerable("w")); //false 属性不是枚举属性
console.log(obj.propertyIsEnumerable("v")); //false 属性不存在
console.log(obj.propertyIsEnumerable("toString")); //false propertyIsEnumerable是继承Object的属性，自身属性中存在
</code></pre>
<h3 id="扩展一-----undefined">扩展一 !== undefined</h3>
有一种简单的方法可以判断属性是否存在，通过属性!== undefined 来判断。此时会检测自身继承来的属性。之所以使用!==而不是!=是因为!==可以区分 undefined 和 null。但是此方法有一个端，当属性存在且值为 undefined 时，无法做出准确判断。如：
<pre><code class="highlight-chroma">var obj = {x:"1", y:undefined, z:null};
console.log(obj.x == undefined); //true 属性存在
console.log(obj.y == undefined); //false 此时会出现歧义，不能准确判断属性是不存在还是属性值为undefined
console.log(obj.z == undefined); //true 属性存在
console.log(obj.z = undefined); //false != 不能区分undefined和null，将两者同等对待
console.log(obj.w == undefined); //false 属性不存在
console.log(obj.toring !== undefined); //true 存在toString函数属性。
</code></pre>
<h3 id="扩展二-in">扩展二 in</h3>
为了避免!==undefined 带来的歧义，可以使用 in 运算符进行属性存在的检测。in 是根据属性而不是通过属性值来判断是否存在。
<pre><code class="highlight-chroma">var obj = {x:"1", y:undefined, z:null};
console.log("x" in bj); //true 属性存在
console.log("y" in obj); //true 属性存在
console.log("z" in bj); //true 属性存在
console.log("w" in obj); //false 属性不存在
console.log("toStrng" in obj); //true 属性存在
</code></pre>
<h3 id="总结">总结</h3>

hasOwnProperty 自身存在的属性
propertyIsEnumerable 自身存在的属性，且为枚举属性
!== undefined 自身存在的属性，继承的属性，不能识别值为 undefined 的属性
in 自身存在的属性，继承的属性

根据具体业务场景，自由选择或组合对应的方法，可以实现对属性的检测。