

属性的操作

HTML 元素包括标签名和若干个键值对，这个键值对就称为“属性”（attribute）。

```
```html

 链接

```
```

上面代码中，`a`元素包括两个属性：`id`属性和`href`属性。

属性本身是一个对象（`Attr`对象），但是实际上，这个对象极少使用。一般都是通过元素节点对象（`HTMLElement`对象）来操作属性。本章介绍如何操作这些属性。

Element.attributes 属性

元素对象有一个`attributes`属性，返回一个类似数组的动态对象，成员是该元素标签的所有属性节点对象，属性的实时变化都会反映在这个节点对象上。其他类型的节点对象，虽然也有`attributes`属性，但返回的都是`null`，因此可以把这个属性视为元素对象独有的。

单个属性可以通过序号引用，也可以通过属性名引用。

```
```javascript
// HTML 代码如下
// <body bgcolor="yellow" onload="">
document.body.attributes[0]
document.body.attributes.bgcolor
document.body.attributes['ONLOAD']
```
```

注意，上面代码的三种方法，返回的都是属性节点对象，而不是属性值。

属性节点对象有`name`和`value`属性，对应该属性的属性名和属性值，等同于`nodeName`属性和`nodeValue`属性。

```
```javascript
// HTML代码为
// <div id="mydiv">
var n = document.getElementById('mydiv');

n.attributes[0].name // "id"
n.attributes[0].nodeName // "id"

n.attributes[0].value // "mydiv"
n.attributes[0].nodeValue // "mydiv"
```
```

下面代码可以遍历一个元素节点的所有属性。

```
```javascript
var para = document.getElementsByTagName('p')[0];
var result = document.getElementById('result');

if (para.hasAttributes()) {
 var attrs = para.attributes;
 var output = '';
 for(var i = attrs.length - 1; i >= 0; i--) {
 output += attrs[i].name + '->' + attrs[i].value;
 }
 result.textContent = output;
} else {
 result.textContent = 'No attributes to show';
}
```
```

元素的标准属性

HTML 元素的标准属性（即在标准中定义的属性），会自动成为元素节点对象的属性。

```
```javascript
var a = document.getElementById('test');
a.id // "test"
a.href // "http://www.example.com/"
```
```

上面代码中，`a` 元素标签的属性 `id` 和 `href`，自动成为节点对象的属性。

这些属性都是可写的。

```
```javascript
var img = document.getElementById('myImage');
img.src = 'http://www.example.com/image.jpg';
```
```

上面的写法，会立刻替换掉 `img` 对象的 `src` 属性，即会显示另外一张图片。

这种修改属性的方法，常常用于添加表单的属性。

```
```javascript
var f = document.forms[0];
f.action = 'submit.php';
f.method = 'POST';
```
```

上面代码为表单添加提交网址和提交方法。

注意，这种用法虽然可以读写属性，但是无法删除属性，`delete` 运算符在这里不会生效。

HTML 元素的属性名是大小写不敏感的，但是 JavaScript 对象的属性名是大小写敏感的。转换规则是，转为 JavaScript 属性名时，一律采用小写。如果属性名包括多个单词，则采用骆驼拼写法，即从第二个单词开始，每个单词的首字母采用大写，比如`onClick`。

有些 HTML 属性名是 JavaScript 的保留字，转为 JavaScript 属性时，必须改名。主要是以下两个。

- `for`属性改为`htmlFor`
- `class`属性改为`className`

另外，HTML 属性值一般都是字符串，但是 JavaScript 属性会自动转换类型。比如，将字符串`true`转为布尔值，将`onClick`的值转为一个函数，将`style`属性的值转为一个`CSSStyleDeclaration`对象。因此，可以对这些属性赋予各种类型的值。

属性操作的标准方法

概述

元素节点提供六个方法，用来操作属性。

- `getAttribute()`
- `getAttributeNames()`
- `setAttribute()`
- `hasAttribute()`
- `hasAttributes()`
- `removeAttribute()`

这有几点注意。

(1) 适用性

这六个方法对所有属性（包括用户自定义的属性）都适用。

(2) 返回值

`getAttribute()`只返回字符串，不会返回其他类型的值。

(3) 属性名

这些方法只接受属性的标准名称，不用改写保留字，比如`for`和`class`都可以直接使用。另外，这些方法对于属性名是大小写不敏感的。

```
```javascript
var image = document.images[0];
image.setAttribute('class', 'myImage');
```

...

上面代码中，`setAttribute`方法直接使用`class`作为属性名，不用写成`className`。

### Element.getAttribute()

`Element.getAttribute`方法返回当前元素节点的指定属性。如果指定属性不存在，则返回`null`。

```
```javascript
// HTML 代码为
// <div id="div1" align="left">
var div = document.getElementById('div1');
div.getAttribute('align') // "left"
```
```

### Element.getAttributeNames()

`Element.getAttributeNames`返回一个数组，成员是当前元素的所有属性的名字。如果当前元素没有任何属性，则返回一个空数组。使用`Element.attributes`属性，也可以拿到同样的结果，唯一的区别是它返回的是类似数组的对象。

```
```javascript
var mydiv = document.getElementById('mydiv');

mydiv.getAttributeNames().forEach(function (key) {
  var value = mydiv.getAttribute(key);
  console.log(key, value);
})
```
```

上面代码用于遍历某个节点的所有属性。

### Element.setAttribute()

`Element.setAttribute`方法用于为当前元素节点新增属性。如果同名属性已存在，则相当于编辑已存在的属性。该方法没有返回值。

```
```javascript
// HTML 代码为
// <button>Hello World</button>
var b = document.querySelector('button');
b.setAttribute('name', 'myButton');
b.setAttribute('disabled', true);
```
```

上面代码中，`button`元素的`name`属性被设为`myButton`，`disabled`属性被设为`true`。

这里有两个地方需要注意，首先，属性值总是字符串，其他类型的值会自动转成字符串，比如布尔值`true`就会变成字符串`true`；其次，上例的`disable`属性是一个布尔属性，对于`<button>`元

素来说，这个属性不需要属性值，只要设置了就总是会生效，因此`setAttribute`方法里面可以将`disabled`属性设成任意值。

### ### Element.hasAttribute()

`Element.hasAttribute`方法返回一个布尔值，表示当前元素节点是否包含指定属性。

```
```javascript
var d = document.getElementById('div1');

if (d.hasAttribute('align')) {
  d.setAttribute('align', 'center');
}
```
```

上面代码检查`div`节点是否含有`align`属性。如果有，则设置为居中对齐。

### ### Element.hasAttributes()

`Element.hasAttributes`方法返回一个布尔值，表示当前元素是否有属性，如果没有任何属性，就返回`false`，否则返回`true`。

```
```javascript
var foo = document.getElementById('foo');
foo.hasAttributes() // true
```
```

### ### Element.removeAttribute()

`Element.removeAttribute`方法移除指定属性。该方法没有返回值。

```
```javascript
// HTML 代码为
// <div id="div1" align="left" width="200px">
document.getElementById('div1').removeAttribute('align');
// 现在的HTML代码为
// <div id="div1" width="200px">
```
```

## ## dataset 属性

有时，需要在HTML元素上附加数据，供 JavaScript 脚本使用。一种解决方法是自定义属性。

```
```html
<div id="mydiv" foo="bar">
```
```

上面代码为`div`元素自定义了`foo`属性，然后可以用`getAttribute()`和`setAttribute()`读写这个属性。

```
```javascript
var n = document.getElementById('mydiv');
n.getAttribute('foo') // bar
n.setAttribute('foo', 'baz')
```
```

这种方法虽然可以达到目的，但是会使得 HTML 元素的属性不符合标准，导致网页代码通不过校验。

更好的解决方法是，使用标准提供的`data-\*`属性。

```
```html
<div id="mydiv" data-foo="bar">
```
```

然后，使用元素节点对象的`dataset`属性，它指向一个对象，可以用来操作 HTML 元素标签的`data-\*`属性。

```
```javascript
var n = document.getElementById('mydiv');
n.dataset.foo // bar
n.dataset.foo = 'baz'
```
```

上面代码中，通过`dataset.foo`读写`data-foo`属性。

删除一个`data-\*`属性，可以直接使用`delete`命令。

```
```javascript
delete document.getElementById('myDiv').dataset.foo;
```
```

除了`dataset`属性，也可以用`getAttribute('data-foo')`、`removeAttribute('data-foo')`、`setAttribute('data-foo')`、`hasAttribute('data-foo')`等方法操作`data-\*`属性。

注意，`data-`后面的属性名有限制，只能包含字母、数字、连词线（`-`）、点（`.`）、冒号（`:`）和下划线（`\_`）。而且，属性名不应该使用`A`到`Z`的大写字母，比如不能有`data-helloWorld`这样的属性名，而要写成`data-hello-world`。

转成`dataset`的键名时，连词线后面如果跟着一个小写字母，那么连词线会被移除，该小写字母转为大写字母，其他字符不变。反过来，`dataset`的键名转成属性名时，所有大写字母都会被转成连词线+该字母的小写形式，其他字符不变。比如，`dataset.helloWorld`会转成`data-hello-world`。