NodeList 接口，HTMLCollection 接口

节点都是单个对象，有时需要一种数据结构，能够容纳多个节点。DOM 提供两种节点集合，用于容纳多个节点：`NodeList`和`HTMLCollection`。

这两种集合都属于接口规范。许多 DOM 属性和方法，返回的结果是`NodeList`实例或`HTMLCollection`实例。主要区别是，`NodeList`可以包含各种类型的节点，`HTMLCollection`只能包含 HTML 元素节点。

NodeList 接口

概述

`NodeList`实例是一个类似数组的对象，它的成员是节点对象。通过以下方法可以得到`NodeList`实例。

- `Node.childNodes`
- `document.querySelectorAll()`等节点搜索方法

```javascript
document.body.childNodes instanceof NodeList // true
```

`NodeList`实例很像数组，可以使用`length`属性和`forEach`方法。但是，它不是数组，不能使用`pop`或`push`之类数组特有的方法。

```javascript
var children = document.body.childNodes;

Array.isArray(children) // false

children.length // 34
children.forEach(console.log)
```

上面代码中，NodeList 实例`children`不是数组，但是具有`length`属性和`forEach`方法。

如果`NodeList`实例要使用数组方法，可以将其转为真正的数组。

```javascript
var children = document.body.childNodes;
var nodeArr = Array.prototype.slice.call(children);
```

除了使用`forEach`方法遍历 NodeList 实例，还可以使用`for`循环。

```javascript
var children = document.body.childNodes;
```

```
for (var i = 0; i < children.length; i++) {
  var item = children[i];
}
```

注意，NodeList 实例可能是动态集合，也可能是静态集合。所谓动态集合就是一个活的集合，DOM 删除或新增一个相关节点，都会立刻反映在 NodeList 实例。目前，只有`Node.childNodes`返回的是一个动态集合，其他的 NodeList 都是静态集合。

```javascript
var children = document.body.childNodes;
children.length // 18
document.body.appendChild(document.createElement('p'));
children.length // 19
```

上面代码中，文档增加一个子节点，NodeList 实例`children`的`length`属性就增加了1。

NodeList.prototype.length

`length`属性返回 NodeList 实例包含的节点数量。

```javascript
document.querySelectorAll('xxx').length
// 0
```

上面代码中，`document.querySelectorAll`返回一个 NodeList 集合。对于那些不存在的 HTML 标签，`length`属性返回`0`。

NodeList.prototype.forEach()

`forEach`方法用于遍历 NodeList 的所有成员。它接受一个回调函数作为参数，每一轮遍历就执行一次这个回调函数，用法与数组实例的`forEach`方法完全一致。

```javascript
var children = document.body.childNodes;
children.forEach(function f(item, i, list) {
  // ...
}, this);
```

上面代码中，回调函数`f`的三个参数依次是当前成员、位置和当前 NodeList 实例。`forEach`方法的第二个参数，用于绑定回调函数内部的`this`，该参数可省略。

NodeList.prototype.item()

`item`方法接受一个整数值作为参数，表示成员的位置，返回该位置上的成员。

```javascript
document.body.childNodes.item(0)
```

上面代码中，`item(0)`返回第一个成员。

如果参数值大于实际长度，或者索引不合法（比如负数），`item`方法返回`null`。如果省略参数，`item`方法会报错。

所有类似数组的对象，都可以使用方括号运算符取出成员。一般情况下，都是使用方括号运算符，而不使用`item`方法。

```javascript
document.body.childNodes[0]
```

NodeList.prototype.keys()，NodeList.prototype.values()，NodeList.prototype.entries()

这三个方法都返回一个 ES6 的遍历器对象，可以通过`for...of`循环遍历获取每一个成员的信息。区别在于，`keys()`返回键名的遍历器，`values()`返回键值的遍历器，`entries()`返回的遍历器同时包含键名和键值的信息。

```javascript
var children = document.body.childNodes;

for (var key of children.keys()) {
  console.log(key);
}
// 0
// 1
// 2
// ...

for (var value of children.values()) {
  console.log(value);
}
// #text
// <script>
// ...

for (var entry of children.entries()) {
  console.log(entry);
}
// Array [ 0, #text ]
// Array [ 1, <script> ]
// ...
```

HTMLCollection 接口

概述

`HTMLCollection`是一个节点对象的集合，只能包含元素节点（element），不能包含其他类型的节点。它的返回值是一个类似数组的对象，但是与`NodeList`接口不同，`HTMLCollection`没有`forEach`方法，只能使用`for`循环遍历。

返回`HTMLCollection`实例的，主要是一些`Document`对象的集合属性，比如`document.links`、`document.forms`、`document.images`等。

```javascript
document.links instanceof HTMLCollection // true
```

`HTMLCollection`实例都是动态集合，节点的变化会实时反映在集合中。

如果元素节点有`id`或`name`属性，那么`HTMLCollection`实例上面，可以使用`id`属性或`name`属性引用该节点元素。如果没有对应的节点，则返回`null`。

```javascript
// HTML 代码如下
// <img id="pic" src="http://example.com/foo.jpg">

var pic = document.getElementById('pic');
document.images.pic === pic // true
```

上面代码中，`document.images`是一个`HTMLCollection`实例，可以通过`<img>`元素的`id`属性值，从`HTMLCollection`实例上取到这个元素。

HTMLCollection.prototype.length

`length`属性返回`HTMLCollection`实例包含的成员数量。

```javascript
document.links.length // 18
```

HTMLCollection.prototype.item()

`item`方法接受一个整数值作为参数，表示成员的位置，返回该位置上的成员。

```javascript
var c = document.images;
var img0 = c.item(0);
```

上面代码中，`item(0)`表示返回0号位置的成员。由于方括号运算符也具有同样作用，而且使用更方便，所以一般情况下，总是使用方括号运算符。

如果参数值超出成员数量或者不合法（比如小于0），那么`item`方法返回`null`。

HTMLCollection.prototype.namedItem()

`namedItem`方法的参数是一个字符串，表示`id`属性或`name`属性的值，返回对应的元素节点。如果没有对应的节点，则返回`null`。

```javascript
// HTML 代码如下
// <img id="pic" src="http://example.com/foo.jpg">

var pic = document.getElementById('pic');
document.images.namedItem('pic') === pic // true
```