

DOMAssistant

2.7.1 版

DOMAssistant 代码库由六个独立模块组成。只有核心模块为必要，其他模块都可以无依赖性地随意组合使用。此文档集合了所有可用的方法和使用范例。

目录

<u>DOMAssistant 核心模块</u>	3
<u>\$(cssSelector / elementReference)</u>	3
<u>\$(elementId)</u>	3
<u>cssSelect(cssSelector)</u>	4
<u>elmsByClass(className, tag)</u>	4
<u>elmsByAttribute(attr, attrVal, tag)</u>	5
<u>elmsByTag(tag)</u>	6
<u>each(functionRef)</u>	6
<u>first()</u>	7
<u>end()</u>	7
<u>DOMAssistantAJAX 模块</u>	7
<u>get(url, callBack)</u>	8
<u>post(url, callBack)</u>	8
<u>load(url, add)</u>	9
<u>ajax(ajaxObject)</u>	10
<u>getReadyState()</u>	10
<u>getStatus()</u>	11
<u>getStatusText()</u>	11

<u>DOMAssistantContent 模块</u>	12
<u>prev()</u>	12
<u>next()</u>	12
<u>create(name, attr, append, content)</u>	13
<u>setAttributes(attr)</u>	14
<u>addContent(content)</u>	14
<u>replaceContent(newContent)</u>	15
<u>remove()</u>	15
<u>DOMAssistantCSS 模块</u>	16
<u>addClass(className)</u>	16
<u>removeClass(className)</u>	16
<u>hasClass(className)</u>	17
<u>setStyle(style, value)</u>	17
<u>getStyle(cssRule)</u>	18
<u>DOMAssistantEvents 模块</u>	18
<u>addEvent(evt, func)</u>	18
<u>removeEvent(evt, func)</u>	19
<u>preventDefault(evt)</u>	20
<u>cancelBubble(evt)</u>	20
<u>DOMAssistantLoad 模块</u>	21
<u>DOMReady()</u>	21
<u>setErrorHandler()</u>	21
<u>支援的 CSS 选择符 (CSS1、CSS2、CSS3)</u>	22

DOMAssistant 核心模块

DOMAssistant 核心模块为所有 DOMAssistant 功能奠基，是必要的模块。它集合了核心功能和一些重要的方法。

\$(cssSelector / elementReference)

\$ 方法用以取得一个或多个元素引用。它支援输入 CSS 选择符 字串，或一个已知的元素引用。它将匹配元素添加了所有 DOMAssistant 方法后返回。如果 \$ 返回空数组，接下来的方法呼叫将默默失灵。

参数

输入 CSS 选择符或元素引用。

返回值

返回 CSS 选择符的匹配引用数组。如果输入的是单一对象，返回的就是那对象引用。输入的是多个对象，则返回对象引用数组。

呼叫范例

```
$("#container input[type=text]);  
  
$("#navigation a");  
  
$("#news-list");  
  
$("#container", "#navigation .important-item", "#content");  
  
$(document.body);
```

\$(elementId)

\$\$ 方法用以快速取得单一元素引用，功能就像 document.getElementById。它返回一个 DOM 元素的直接引用，并添加所有 DOMAssistant 的方法。

和 \$ 方法不同的是，如果 \$\$ 没有返回任何匹配元素，接下来的方法呼叫将抛出异常。

参数

输入元素的 id。

返回值

返回一个 DOM 引用。

呼叫范例

```
$$("container");
```

```
$$("navigation");
```

cssSelect(cssSelector)

适用于现有的对象引用，使用CSS选择符寻找该对象的子元素。

参数

cssSelector

用以选择元素。必要。

返回值

返回元素引用数组。

呼叫范例

```
$(document).cssSelect(".mandatory");
```

```
$(DOM元素引用).cssSelect(".important[type=test]");
```

elmsByClass(className, tag)

根据类别 (class) 获取元素数组。className 为必要参数。如果想要将搜索局限于单一标签，您也可以传入选择性参数 tag。

参数

className

搜索的类别名称。必要。

tag

只搜索此标签。选择性。

返回值

返回元素引用数组。

呼叫范例

```
$("#container").elmsByClass("mandatory");
```

```
$$("container").elmsByClass("external-link", "a");
```

elmsByAttribute(attr, attrVal, tag)

根据属性 (attribute) 获取元素数组。您也可以选择性指定属性值和标签。

参数

attr

属性名称。必要。

attrVal

元素属性值须匹配。选择性。如果想要指定标签但不想局限属性值，可使用通配符 ("*")。

tag

只搜索此标签。选择性。

返回值

返回元素引用数组。

呼叫范例

```
$("#container").elmsByAttribute("href");
```

```
$$("container").elmsByAttribute("name", "subscription");
```

```
$$("container").elmsByAttribute("type", "text", "input");
```

elmsByTag(tag)

根据标签 (tag) 获取元素引用。此方法有一个必要参数，即是标签名称。

参数

tag

搜索的标签名称。必要。

返回值

返回元素引用数组。

呼叫范例

```
$$("container").elmsByTag("input");
```

each(functionRef)

用以在返回的元素引用数组上逐项运行函数。

返回值

返回单一元素引用或元素引用数组。

参数

functionRef

在元素数组里逐项元素运行的函数。可以是匿名函数或函数引用。

呼叫范例

```
$("#navigation a").each(function () {  
    // 执行 JavaScript 戏法  
});
```

first()

获取第一个匹配的元素引用。

返回值

返回数组里的第一个匹配元素 (已添加全部 DOMAssistant 方法)。

参数

无。

呼叫范例

```
$("#navigation a").first();
```

end()

用以返回元素链里当前位置的上一个元素组。

返回值

返回元素链里的上一个元素组。

参数

无。

呼叫范例

```
// 返回 a 元素组  
$("#navigation a").create("span", null, true).end();
```

DOMAssistantAJAX 模块

DOMAssistantAJAX 模块提供基本的 AJAX 交互框架，用以从 URL 提取数据并传递到回调

函数，或将内容载入元素中。

get(url, callBack)

执行 GET 命令请求 URL 并设定回调函数。传入到 callBack 里的第一个参数既是此 AJAX 请求的 `responseText` 值。如果此方法运用于元素上，那么回调环境中 `this` 关键字即代表呼叫此 `get` 方法的元素。

参数

url

AJAX 请求的 URL。必要。

callBack

AJAX 请求完毕后回调的函数名称或匿名函数。选择性。

返回值

无。

呼叫范例

```
$("#news").get("news.php", insertNews);
```

```
DOMAssistant.AJAX.get("my-url.aspx", callbackFunctionName);
```

post(url, callBack)

执行 POST 命令请求 URL 并设定回调函数。传入到 callBack 里的第一个参数既是此 AJAX 请求的 `responseText` 值。如果此方法运用于元素上，那么回调环境中 `this` 关键字即代表呼叫此 `post` 方法的元素。

参数

url

AJAX 请求的 URL。必要。

callBack

AJAX 请求完毕后回调的函数名称或匿名函数。选择性。

返回值

无。

呼叫范例

```
$("#news").post("news.php?value=true", insertNews);
```

```
DOMAssistant.AJAX.post("my-url.aspx?number=10",  
callbackFunctionName);
```

load(url, add)

请求 URL 并将返回内容载入呼叫此方法的元素。

参数

url

AJAX 请求的 URL。必要。

add

布尔值，设定是否将返回内容附加于现有内容后。true 为附加，false 为取代现有内容。选择性。

返回值

无。

呼叫范例

```
$("#directions").load("maps.php");
```

```
$("#contacts").load("staff.aspx", true);
```

ajax(ajaxObject)

一个通用的 AJAX 方法。适用于处理更细腻的 AJAX 请求。

参数

ajaxObject

一个拥有各种参数的 Javascript 对象。可设置的参数有

- url
- method ("GET" 或 "POST")
- params
- callback
- headers
- responseType ("text" 或 "xml")

Javascript 对象固然为必要，但参数中只有 url 是必要的。如果只提供 url，那 method 和 responseType 的默认值分别是 GET 和 text。

返回值

呼叫该方法的元素。

呼叫范例

```
$("#container").ajax( {  
    url: "ajax.php",  
    method: "POST",  
    params: "name=DOMAssistant",  
    callback: functionReference,  
    headers: {  
        "Content-type": "application/x-www-form-urlencoded"  
    },  
    responseType : "text"  
});
```

getReadyState()

帮助方法，用以查询 XMLHttpRequest 的 readyState 状态。

参数

无。

返回值

整数。

0 = 未初始化 (uninitialized)

1 = 载入中 (loading)

2 = 载入完成 (loaded)

3 = 交互 (interactive)

4 = 完成 (complete)

呼叫范例

```
DOMAssistant.AJAX.getReadyState();
```

getStatus()

帮助方法，用以查询 XMLHttpRequest 的 status 状态码。

参数

无。

返回值

整数。例如 200 对应 OK，404 对应 Not Found。

呼叫范例

```
DOMAssistant.AJAX.getStatus();
```

getStatusText()

帮助方法，用以查询 XMLHttpRequest 的 statusText 状态。

参数

无。

返回值

字符串。status 状态码的字释。

呼叫范例

```
DOMAssistant.AJAX.getStatusText();
```

DOMAssistantContent 模块

DOMAssistantContent 模块提供用以添加与删除内容和元素的方法。

prev()

取得上一个 HTML 元素引用，文字节点自动跳过。

参数

无。

返回值

元素的前驱兄弟。

呼叫范例

```
$("#container").prev();
```

next()

取得下一个 HTML 元素引用，文字节点自动跳过。

参数

无。

返回值

元素的后继兄弟。

呼叫范例

```
$("#container").next();
```

create(name, attr, append, content)

建立新元素，并选择性地设定属性、附加于当前元素、添加内容。

参数

name

新元素的标签名称。必要。

attr

一个装含属性与相应值的对象。选择性。

append

布尔值，定义新元素是否直接附加于当前元素中。选择性。

content

新元素的内容，可以是字串或 HTML 对象。选择性。

返回值

新建立的元素。

呼叫范例

```
$("#container").create("div");
```

```
$("#container").create("div", {id: "my-div", className: "my-class"});
```

```
$("#container").create("div", null, false, "Hello!");
```

```
$("#container").create("div", {id: "my-div", className: "my-class"
```

```
}, true, "Hi there!");
```

setAttributes(attr)

设定当前元素的属性。

参数

attr

一个装含属性与相应值的对象。必要。

返回值

呼叫此方法的元素。

呼叫范例

```
$("#container").setAttributes({id: "my-div", className: "my-class"});
```

addContent(content)

为当前元素添加内容。

参数

content

可以是字串或 HTML 对象。如果是字串，则用 `innerHTML` 添加。反之 HTML 对象会以 `appendChild` 方法附加。

返回值

呼叫此方法的元素。

呼叫范例

```
$("#container").addContent("<p>A new paragraph</p>");
```

```
$("#container").addContent(document.createElement("p"));
```

replaceContent(newContent)

取代当前元素的内容。

参数

newContent

可以是字串或 HTML 对象。如果是字串，则篡改 innerHTML 值。而 HTML 对象则会用以 appendChild 方法取代内容。

返回值

呼叫此方法的元素。

呼叫范例

```
$("#container").replaceContent("<p>A new paragraph</p>");
```

```
$("#container").replaceContent(document.createElement("p"));
```

remove()

删除当前元素。

参数

无。

返回值

Null。

呼叫范例

```
$("#container").remove();
```

DOMAssistantCSS 模块

DOMAssistantCSS 模块提供添加与删除 CSS 样式类的方法。您也可查询元素是否拥有指定的样式类，以及取得元素特定属性的显示样式。

addClass(className)

如果未有，则添加样式类于当前元素。

参数

className

样式类名称。必要。

返回值

呼叫此方法的元素。

呼叫范例

```
$("#container").addClass("selected");
```

removeClass(className)

从元素中去除样式类。

参数

className

样式类名称。必要。

返回值

呼叫此方法的元素。

呼叫范例

```
$("#container").removeClass("selected");
```

hasClass(className)

查询当前元素是否拥有指定样式类。

参数

className

样式类名称。必要。

返回值

布尔值。说明元素是否拥有指定样式类。

呼叫范例

```
$("#container").hasClass("selected");
```

setStyle(style, value)

定义一个或多个内联样式。注：请使用 CSS 语法 (如 background-color) 而非 Javascript 属性语法 (如 backgroundColor)。

参数

style

样式属性。可以是字串，与属性值 value 共同使用。也可以是拥有多个样式值的对象。必要。

value

如果 style 是字串，这就是该样式的值。选择性。

返回值

呼叫该方法的元素。

呼叫范例

```
$("#container").setStyle("border", "10px solid red");

$("#container").setStyle( {
    background: "#ffffa2",
    color: "#f00"
});
```

getStyle(cssRule)

获取当前元素一个特定属性的显示样式，无论属性样式是以代码或外联方式施用。注：指定的属性必须是特定的，如 `background-color`，而不是泛式 `background` 等。

参数

cssRule

查询中的 CSS 属性。用 CSS 原语法，如 `background-color`，而不是其骆驼写法 `backgroundColor`。必要。

返回值

字符串。查询到的样式值。

呼叫范例

```
$("#container").getStyle("background-color");
```

DOMAssistantEvents 模块

DOMAssistantEvents 模块提供各种事件操作的方法。它也有取消默认动作和防止事件冒泡的功能。

addEvent(evt, func)

为当前元素添置一个事件处理器。可设置多个事件处理器。处理器函数将获取一个事件引用，

而 `this` 关键字代表引发事件的元素。如果想要最大的可及性，请只在没有 Javascript 也可处理 `click` 事件的元素上添加该事件处理器。

参数

evt

被处理的事件。用字符串定义，并省却 "on" 前缀。

func

处理事件的函数。可为函数名称或匿名函数。

返回值

呼叫此方法的元素。

呼叫范例

```
$("#container").addEvent("click", getListing);

$("#container").addEvent("click", function () {
    alert("Hello darling!");
});
```

removeEvent(evt, func)

为当前元素撤除一个事件处理器。只适用于函数引用而非匿名函数。

参数

evt

被撤除的事件。用字符串定义，并省却 "on" 前缀。

func

撤除的事件处理器函数名称。

返回值

呼叫此方法的元素。

呼叫范例

```
$("#container").removeEvent("click", getListing);
```

preventDefault(evt)

防止事件的默认动作。可从任何函数呼叫，但它并不是元素的方法。

参数

evt

被防止默认动作的事件。

返回值

无。

呼叫范例

```
DOMAssistant.preventDefault(eventReference);
```

cancelBubble(evt)

取消事件冒泡。可从任何函数呼叫，但它并不是元素的方法。

参数

evt

被取消冒泡的事件。

返回值

无。

呼叫范例

```
DOMAssistant.cancelBubble(eventReference);
```

DOMAssistantLoad 模块

DOMAssistantLoad 模块让您可以在页面 DOM 加载完成后回调函数，而不需要等待图像及其他外联内容加载。灵感取自 Dean Edwards、Matthias Miller 与 John Resig 的 [window.onload \(again\)](#)。

DOMReady()

在页面设定 DOMReady 方法并传入函数名称，让页面 DOM 加载完成后开始执行。

参数

传入无限量函数引用、匿名函数、或函数和参数形成的字串。

返回值

无。

呼叫范例

```
DOMAssistant.DOMReady(myFunc);  
  
DOMAssistant.DOMReady("myFunc('Some text')");  
  
DOMAssistant.DOMReady(myFunc, "anotherFunction()");  
  
DOMAssistant.DOMReady(myFunc, function() {  
    // 执行戏法  
});
```

setErrorHandling()

处理呼叫 DOMReady 时可能出现的错误。

参数

传入一个函数引用。

返回值

无。

呼叫范例

```
DOMAssistant.DOMLoad.setErrorHandling( function(e) {  
    // e 为错误对象  
});
```

支援的 CSS 选择符 (CSS1、CSS2、CSS3)

DOMAssistant 核心模块的 \$ 方法支援使用 CSS 选择符来取得一个或多个元素的引用。

CSS 1、CSS 2 和 CSS 3 规范里主要的选择符都可使用，并拥有与 CSS 文件同样的语法。

想深入了解 CSS 选择符和它们的使用法，可参考 [CSS 2.1 选择符](#) 和 [CSS 3 选择符阐述](#)。

可用的 CSS 选择符

#container

获取 id 为 "container" 的元素。

.item

获取所有类为 "item" 的元素。

#container.item

获取 id 为 "container"，与此同时类为 "item" 的元素。

p.info.error

获取所有具备 "info" 类和 "item" 类的 P 元素。

div p

获取所有是 DIV 元素后代的 P 元素。

`div > p`

获取所有是 DIV 元素直属后代的 P 元素。

`div + p`

获取所有直属前驱兄弟是 DIV 元素的 P 元素。

`div ~ p`

获取所有是 DIV 元素后继兄弟的 P 元素。（不一定直属）

`div[id]`

获取所有拥有 id 属性的 DIV 元素。

`div[id=container]`

获取所有 id 属性值为 "container" 的 DIV 元素。

`input[type=text][value=Yes]`

获取所有 type 属性值为 "text" 和 value 属性值为 "Yes" 的 INPUT 元素。

`div[id^=empt]`

获取所有 id 属性值开头为 "empt" 的 DIV 元素。

`div[id$=parent]`

获取所有 id 属性值结尾为 "parent" 的 DIV 元素。

`div[id*=mpt]`

获取所有 id 属性值包含 "mpt" 字串的 DIV 元素。

`div[foo~=bar]`

获取所有 foo 属性值是一个以空格分割的列表，且其中之一为 "bar" 的 DIV 元素。

`div[lang|=zh]`

获取所有 lang 属性值是一个以连字号分割的列表，并且以 "zh" 开头的 DIV 元素。

`div:first-child`

获取所有身为第一个子元素的 DIV。

`div:last-child`

获取所有身为最后一个子元素的 DIV。

`div:only-child`

获取所有身为唯一子元素的 DIV。

`div#container p:first-of-type`

获取 id 值为 "container" 的 DIV 元素的后代，而其必须是第一个 P 类型子元素。

`p:last-of-type`

获取所有身为最后一个 P 类型的子元素。

`p:only-of-type`

获取所有身为唯一 P 类型子元素的元素。

`p:nth-of-type(7)`

获取为第七个 P 类型子元素的元素。

`div:empty`

获取完全空的 DIV 元素（包括文字节点）。

`div:not([id=container])`

获取所有 id 属性值不是 "container" 的 DIV 元素。

`div:nth-child(3)`

获取为第三个子元素的 DIV 元素。

`div:nth-child(odd)`

获取所有 DIV 元素集中序数为奇数的元素。

`div:nth-child(even)`

获取所有 DIV 元素集中序数为偶数的元素。

`div:nth-child(5n+3)`

从第三个 DIV 子元素开始，获取所有 DIV 元素集中序数为 $(5n+3)$ 的元素 (3, 8, 13, 18 等等) 。

`tr:nth-last-of-type(-n+3)`

获取任何表格的最后三行。

`td:nth-last-child(n+1)`

除了最后一列之外，获取任何表格的所有单元格。

`input:enabled`

获取所有可用的 INPUT 元素。

`input:disabled`

获取所有禁用的 INPUT 元素。

`input:checked`

获取所有选中的 INPUT 元素。

`div:lang(zh)`

获取所有中文内容的 DIV 元素。

`p:target`

获取为文件 URL 目标的 P 元素。

p, a

获取所有 P 元素与 A 元素。