

# jQuery

最流行的JavaScript程序库

## jQuery的用途

- 访问和操作DOM元素
- 控制页面样式
- 对页面样式进行处理
- 扩展新的jQuery插件
- 与Ajax技术完美结合

\* **DOM** : 文件对象模型 ( Document Object Model ) 是一种理念 , 一种思想 , 一种使 Web 开发人员可以访问 HTML 元素的功能 , 不是具体的方法。

\* **Ajax** : 异步 JavaScript 和 XML ( Asynchronous Javascript And XML ) , 是指一种创建交互式网页应用的网页开发技术 ; 是一种用于创建快速动态网页的技术 ; 是一种在无需重新加载整个网页的情况下 , 能够更新部分网页的技术。

## jQuery的优势

- 体积小 , 压缩后只有 100KB 左右
- 强度的选择器
- 出色的 DOM 封装
- 可靠的事件处理机制
- 出色的浏览器兼容性
- 使用隐式迭代简化编程
- 丰富的插件支持

## jQuery使用

### 引入jQuery库文件

直接在 HTML 文件 body 标签内容的地步调用即可

```
<script type="text/javascript" src="plug/jquery-1.11.0.js"></script>
```

### 使用jQuery

#### 1. 使用jQuery弹出提示框

```
<script>
$(document).ready(function(){
```

```
    alert("Hello jQuery");
});
</script>
```

## 2. 语法

`$(选择器).方法名();`

`$( )` :工厂函数，将DOM对象转化为jQuery对象

选择器：需要操作的DOM元素

方法名( )：jQuery中提供的方法，包括操作元素和事件。

## 3.DOM元素的操作

`var $input = $('#userName');` 表示获取id为userName的jQuery对象

## 4.将DOM对象转换为jQuery对象

`var userName = document.getElementById('userName');` //获得DOM对象

`var $userName = $(userName);` //获得jQuery对象

## 5.通过选择器直接获得jQuery对象

`var $userName = $('#userName');` //直接获得id为userName的jQuery对象

## 6.通过jQuery对象获得DOM对象

`var $userName = $('#userName');` //直接获得id为userName的jQuery对

象

`var userName = $userName[0];` //通过数组方式获得

`var userName = $userName.get(0);` //通过get方式获得

# jQuery选择器

## 1. 基本选择器

名称	语法构成	描述	示例
标签选择器	element	根据给定的标签名匹配元素	<code>\$("h2")</code> 选取所有h2元素
类选择器	.class	根据给定的class匹配元素	<code>\$(".title")</code> 选取所有class为title的元素
ID选择器	#id	根据给定的id匹配元素	<code>\$("#title")</code> 选取id为title的元素
并集选择器	selector1,selector2,...,selectorN	将每一个选择器匹配的元素合并后一起返回	<code>\$(".div,p,.title")</code> 选取所有div、p和拥有class为title的元素
交集选择器	element.class或element#id	匹配指定class或id的某元素或元素集合	<code>\$(".h2.title")</code> 选取所有拥有class为title的h2元素
全局选择器	*	匹配所有元素	<code>\$("*")</code> 选取所有元素

## 2. 层次选择器

名称	语法构成	描述	示例
后代选择器	ancestor descendant	选取ancestor元素里的所有descendant（后代）元素	<code>\$("#menu span")</code> 选取#menu下的 <span>元素</span>
子选择器	parent>child	选取parent元素下的child（子）元素	<code>\$("#menu&gt;span")</code> 选取#menu的子元素 <span></span>
相邻元素选择器	prev+next	选取紧邻prev元素之后的next元素	<code>\$(".h2+dl")</code> 选取紧邻 <h2>元素之后的同辈元素<dl></dl></h2>
同辈元素选择器	prev~siblings	选取prev元素之后的所有siblings元素	<code>\$(".h2~dl")</code> 选取 <h2>元素之后所有的同辈元素<dl></dl></h2>

## 3. 属性过滤选择器

名称	语法构成	描述	示例
属性过滤选择器	[attribute]	选取包含给定属性的元素	<code>\$("[href]")</code> 选取含有href属性的元素
	[attribute=value]	选取等于给定属性是某个特定值的元素	<code>\$("[href ='#']")</code> 选取href属性值为“#”的元素
	[attribute!=value]	选取不等于给定属性是某个特定值的元素	<code>\$("[href != '#']")</code> 选取href属性值不为“#”的元素
	[attribute^=value]	选取给定属性是以某些特定值开始的元素	<code>\$("[href^='en']")</code> 选取href属性值以en开头的元素
	[attribute\$=value]	选取给定属性是以某些特定值结尾的元素	<code>\$("[href\$='jpg']")</code> 选取href属性值以.jpg结尾的元素
	[attribute*=value]	选取给定属性是以包含某些值的元素	<code>\$("[href*='txt']")</code> 选取href属性值中含有txt的元素
	[selector] [selector2] [selectorN]	选取满足多个条件的复合属性的元素	<code>\$("li[id][title=新闻要点]")</code> 选取含有id属性和title属性为新闻要点的 <li>元素</li>

## 4. 基本过滤选择器

名称	语法构成	描述	示例
基本过滤选择器	:first	选取第一个元素	\$( " li:first" )选取所有<li>元素中的第一个<li>元素
	:last	选取最后一个元素	\$( " li:last" )选取所有<li>元素中的最后一个<li>元素
	:even	选取索引是偶数的所有元素 (index从0开始)	\$( " li:even" )选取索引是偶数的所有<li>元素
	:odd	选取索引是奇数的所有元素 (index从0开始)	\$( " li:odd" )选取索引是奇数的所有<li>元素
	:eq(index)	选取索引等于index的元素 (index从0开始)	\$( "li:eq(1)" )选取索引等于1的<li>元素
	:gt(index)	选取索引大于index的元素 (index从0开始)	\$( " li:gt(1)" )选取索引大于1的<li>元素 (注：大于1，不包括1)
	:lt(index)	选取索引小于index的元素 (index从0开始)	\$( "li:lt(1)" )选取索引小于1的<li>元素 (注：小于1，不包括1)

## 5. 可见性过滤选择器

名称	语法构成	描述	示例
可见性过滤选择器	:visible	选取所有可见的元素	\$( ":visible" )选取所有可见的元素
	:hidden	选取所有隐藏的元素	\$( ":hidden" ) 选取所有隐藏的元素

## 6. 表单过滤选择器

名称	语法构成	描述	示例
表单过滤选择器	:input	匹配所有 input, textarea, select 和 button 元素	\$( ":input" )选取所有的表单元素
	:checked	匹配所有选中的被选中元素(复选框、单选框等，不包括select中的option)	\$( "[type=checkbox]:checked" ) 选取所有选中的复选框元素
	:selected	匹配所有选中的option元素	\$( "select:selected" )匹配所有选中的下拉框
	:enabled	匹配所有可用元素	\$( ":enabled" )
	:disabled	匹配所有不可用元素	\$( "button:disabled" )

# DOM操作

## 1.DOM Core

可以用来处理任何一种使用标记语言编写出来的文档，可以用来获取表单元素、用来获取元素的属性。

例：使用DOM Core获取表单对象：

```
document.getElementsByName("form");
```

使用DOM Core获取元素的src属性：element.getAttribute("src");

## 2.HTML-DOM

提供了一些简明的记号来描述各种HTML元素属性、只能用来处理Web文档。

例：使用HTML-DOM来获取表单对象的方法：

document.forms; //HTML-DOM提供了一个forms对象

使用HTML-DOM来获取元素的src属性方法：element.src;

### 3.CSS-DOM

针对css操作、主要作用是获取和设置style对象的各种属性

例：设置某元素style对象字体颜色的方法：element.style.color = "red";

## 节点操作

### 1. 元素外部添加

语法	功能
after(content)	\$(A).after (B)表示将B插入到A之后 如：\$("ul").after(\$newNode1);
insertAfter(content)	\$(A).insertAfter (B)表示将A插入到B之后 如：\$newNode1.insertAfter("ul");
before(content)	\$(A).before (B)表示将B插入至A之前 如：\$("ul").before(\$newNode1);
insertBefore(content)	\$(A).insertBefore (B)表示将A插入到B之前 如：\$newNode1.insertBefore("ul");

### 2. 创建元素节点

创建第一个<li>元素：var \$li\_1 = \$("<li></li>");

创建第二个包含文本的<li>元素：var \$li\_1 = \$("<li>文本</li>");

创建第三个包含元素节点、文本和属性的节点：var \$li\_2 = \$(<li title='属性'>文本</li>);

在<ul>的子节点的最后一位添加第一个<li>节点：\$("ul").append(\$li\_1);

将第二个<li>节点添加到<ul>子节点的最后一位：\$li\_2.appendTo\$("ul"));

### 3. 删除节点

- remove()方法：从DOM中删除所有匹配的元素

例：获取第一个<li>节点，并将其从网页中删除：

```
$("ul li:first").remove();
```

- detach()方法：不把匹配的元素从jQuery对象中删除，可以声明变量接收他

例：获取第二个<li>元素节点并创建jQuery对象接收该节点，然后将其从网页中删除：

```
$( "ul li:eq(1)" ).detach();
```

- **empty()方法**：清除元素中的所有后代节点（清除节点内的所有内容）

例：获取第最后一个- 节点，然后清空元素里的内容：

```
$( "ul li:last" ).empty();
```

#### 4. 复制节点 ( clone() )

复制当前点击的节点，并将其追加到

元素

```
$(this).clone().appendTo("ul"); //仅复制节点
```

```
$(this).clone(true).appendTo("ul"); //复制节点和事件
```

\* 参数**true**表示复制节点的同时将节点所设置的事件也一同复制

## 属性和样式

### 1. 属性

- 设置属性

```
$( "p" ).attr("title", "属性值") //设置

元素的title属性


```

- 获取属性

```
alert$( "p" ).attr("title")); //获取

元素的title属性


```

- 删除属性

```
$( "p" ).removeAttr("title"); //删除

元素的title属性


```

### 2. 样式 ( class )

- 获取样式

```
$( "p" ).attr("class"); //获取

元素的class样式


```

- 设置样式

```
$( "p" ).attr("class", "high"); //设置

元素的class样式


```

## \* 使用设置class样式会覆盖原有class样式

- 追加样式

```
$( "p" ).addClass( "another" ); //给<p>元素追加class样式
```

- 删除样式

```
$( "p" ).removeClass(); //删除<p>元素的所有class样式
```

```
$( "p" ).removeClass( "high" ); //删除<p>元素的指定class样式
```

- 切换样式 ( 如果样式存在就删除 , 如果样式不存在就创建样式 )

```
$( "p" ).toggleClass( "another" ); //切换<p>元素的class样式
```

- 是否包含样式

```
$( "p" ).hasClass( "another" ); //查看<p>元素是否包含指定class样式
```

```
$( "p" ).is( ".another" ); //查看<p>元素是否包含指定class样式
```

## 设置HTML文本值

### 1. 获取或设置HTML

```
$( "p" ).html(); //获取<p>元素的HTML代码
```

```
$( "p" ).html( "<span>内容</span>" ) //设置<p>元素的HTML代码
```

### 2. 获取或设置text

```
$ ( "p" ).text(); //获取<p>元素的文本内容
```

```
$ ( "p" ).text( "文本内容" ); //设置<p>元素的文本内容
```

### 3. 获取或设置值

```
$ ( "input" ).val(); //获取<input>元素的value值
```

```
$ ( "input" ).val( "默认值" ); //设置<input>元素的value值
```

\* 设置内容会将元素内原有的所有内容覆盖

\* 获取的文本内容包含<p>元素内所有子孙的文本内容 ( 不含标签 )

## 遍历节点

语法	功能
<code>children()</code>	获取元素的子元素
<code>next()</code>	获取下一个同辈元素
<code>prev()</code>	获取前一个同辈元素
<code>siblings()</code>	获取匹配的同级元素
<code>closest()</code>	获取最先匹配的祖先元素
<code>parent()</code>	获取匹配的父级元素
<code>parents()</code>	获取匹配的祖先元素

## CSS-DOM操作

### 1. 使用css()设置属性

```
css('属性', '值'); //设置单个属性  
css({'属性1':'值1','属性2':'值2',.....}); //设置多个属性
```

### 2. 获取元素的宽度和高度

```
width(); //获取元素的宽度  
height(); //获取元素的高度  
例： p.width() //获取<p>元素的宽度  
     p.height() //获取<p>元素的高度
```

### 3. 获取相对窗口的位置

```
offset();  
例： $("p").offset(); //获取<p>元素左上角相对于窗口左上角的x、y距离
```

### 4. 获取元素相对其父元素的偏移距离

```
position();  
例： $("p").position(); //获取<p>元素左上角相对于其父元素左上角的x、y距  
离
```

### 5. 获取元素的滚动条距离

```
scrollTop(); //获取元素相对于滚动条拖动到最顶部时的相对距离
```

```
scrollLeft();           //获取元素相对于滚动条拖动到最左侧时的相对距离  
scrollTop(value);    //设置元素相对于滚动条拖动到最顶部时的相对偏移量  
scrollLeft(value);   //设置元素相对于滚动条拖动到最左侧时的相对偏移量
```