# jquery

## 什么jquery

jQuery(javaScript Query)是一个优秀的JavaScript库，是一个凭借简洁的语法和跨平台的兼容性，极大地简化了JavaScript开发人员遍历HTML文档，操作DOM，执行动画和开发Ajax的操作。jQuery封装了很多预定义的对象和函数。其理念：write less,do more。

功能：选择器、过滤器、时间、效果、ajax。

1、jQuery(javaScript Query)是一个优秀的JavaScript库。

2、是一个快速的简洁的javascript框架，可以简化查询DOM对象、处理事件、制作动画、处理Ajax交互过程。

## js的功能

jQuery使用户能更方便地处理HTML、events、实现动画效果，并且方便地为网站提供AJAX交互。

jQuery库包含以下功能：HTML元素选取、HTML元素操作、CSS操作、HTML事件函数、JavaScript特效和动画、HTML|DOM遍历和修改、AJAX、Utilities。

jQuery使用户方便快捷获取DOM元素、动态修改页面样式、动态改变DOM内容、响应用户的交互操作、为页面添加动态效果、统一Ajax操作、简化常见的JavaScript任务。

## Jquery的版本

1.x：兼容ie678，但相对其它版本文件较大，官方只做BUG维护，功能不再新增，最终版本：1.12.4 (2016年5月20日).

2.x：不兼容ie678，相对1.x文件较小，官方只做BUG维护，功能不再新增，最终版本：2.2.4 (2016年5月20日)

3.x：不兼容ie678，只支持最新的浏览器，很多老的jQuery插件不支持这个版本，相对1.x文件较小，提供不包含Ajax/动画API版本。

开发板: 所有代码没有经过压缩,体积更大(200-300KB)，利于阅读和学习。

生产版:所有代码经过压缩,提及更小(30-40KB)，开发使用，利于提高响应速度。

初学者为了更好的理解jQuery编码时使用开发板,项目上线时为了提升访问速度使用生产版

## 常见的javascript库

Prototype:是最早成型的JS库之一，对于JS的内置对象做了大量的扩展。

Do jo：提供了很多奇特JS库没有提供的功能。例如：离线存储的API，生成图标的组件等等。

YUI：是由Yahoo公司开发的一套完备的，扩展性良好的富交互网页程序工作集。

Ext JS：原本是对YUI的一个扩展，主要用于创建前端用户界面。

Moo Tools：是一套轻量、简洁、模拟化和面向对象的JS框架。

jQuery：同样是一个轻量级的库，拥有强大的选择器等更多优点，吸引了更多开发者去学习使用它。

## jQuery函数

jQuery库只提供了一个叫jQuery的函数，该函数中以及该元素的原型中定义了大量的方法。jQuery函数具有四种参数：

1）选择器（字符串）

jQuery函数通过该选择器获取对应的DOM，然后将这些DOM封装到一个jQuery对象中并返回。

2）DOM对象（即Node实例）

jQuery函数将该DOM封装成jQuery对象并返回。

3）HTML文本字符串

jQuery函数根据传入的文本创建好HTML元素并封装成jQuery对象并返回。

$("<div class="one">one</div>");

4）一个匿名函数

$(function(){});

当文当加载完毕之后jQuery函数调用匿名函数。

## jQuery对象

jQuery对象是jQuery函数的一个实例，是一个类数组对象，数组中存放的是DOM对象，而DOM对象是Node的实例。

对jQuery对象的操作实际上是对jQuery数组中的DOM对象的批量操作。jQuery对象和DOM对象可以相互转化。

jQuery对象的获取通常是使用选择器来获取的。比如，获取所有clss为one元素:$(".one");

## Jquery的入口函数

### 引入Jquery的入口函数

|  |
| --- |
| <head>  <meta charset="UTF-8">  <title>01-初识jQuery</title>  <script src="code/js/jquery-1.12.4.js"></script>  </head> |

编写代码

|  |
| --- |
| <script>  $(document).ready(function () {  // 所有jQuery代码写在这里面  alert("hello LNJ");  });  </script> |

### jQuery与JavaScript加载模式对比

1、原生JS和jQuery入口函数的加载模式不同。

原生JS会等到DOM元素加载完毕，并且图片也加载完毕才会执行。

jQuery会等到DOM元素加载完毕，但不会等到图片也加载完毕就会执行。

2、多个window.onload只会执行一次, 后面的会覆盖前面的。多个$(document).ready()会执行多次,后面的不会覆盖前面的。

|  |
| --- |
| <script>  window.onload = function () {  alert("hello lnj1"); // 不会显示  }  window.onload = function () {  alert("hello lnj2"); // 会显示  }  </script>  <script>  $(document).ready(function () {  alert("hello lnj1"); //会显示  });  $(document).ready(function () {  alert("hello lnj2"); // 会显示  });  </script> |

### Jquert入口函数的其他写法

|  |
| --- |
| <script type="text/javascript">  //第一种方式  $(document).ready(function(){  alert("sx");    });  //第二种方式  jQuery(document).ready(function(){  alert("sx");    });  //第三种写法(推荐)  $(function(){  alert("sx");  });  //第四种写法  jQuery(function(){  alert("sx");  });  </script> |

### Jquery的冲突问题

|  |
| --- |
| <script src="js/jquery.min.js"></script>\  <script type="text/javascript">  //引入的文件可能有$符号，会覆盖Jquery中的$  /\*<script type="引入的文件">\*/  /\*解决方法:  1、释放Juery的使用权  注意点：释放操作必须在编写其它jQuery代码之前编写  释放之后就不能再使用3，改为使用jQuery  jQuery.noConflict();  2、如果任然想用符号代表jQuery  自定义一个符号，使用sx可以作为$符号  var sx = jQuery.noConflict();  \*  \* \*/  var sx = jQuery.noConflict();  sx(function(){  alert("哈哈");  })  </script> |

### 讲解$(function(){});

1、$是jQuery别名。如$()也可jQuery()这样写,相当于页面初始化函数，当页面加载完毕，会执行jQuery()。

2、希望在做所有事情之前，JQuery操作DOM文档。必须确保在DOM载入完毕后开始执行，应该用ready事件做处理HTML文档的开始。

3、$(document).ready(function(){});

类似于js的window.onload事件函数，但是ready事件要先于onload事件执行。

window.onload = function(){};。

4、为方便开发，jQuery简化这样的方法，直接用$()表示。

5、JQuery的ready事件不等于Js的load：

执行时机不同：load需要等外部图片和视频等全部加载才执行。ready是DOM绘制完毕后执行，先与外部文件。

用法不同：load只可写一次，ready可以多次。

# Jquery

## jQuery函数

jQuery库只提供了一个叫jQuery的函数，该函数中以及该元素的原型中定义了大量的方法。jQuery函数具有四种参数：

1)选择器（字符串）

jQuery函数通过该选择器获取对应的DOM，然后将这些DOM封装到一个人jQuery对象中并返回。

2)DOM对象（即Node实例）

jQuery函数将该DOM封装成jQuery对象并返回。

3)HTML文本字符串

jQuery函数根据传入的文本创建好HTML元素并封装成jQuery对象并返回。

$("<div class="one">one</div>");

4)一个匿名函数

$(function(){});

当文当加载完毕之后jQuery函数调用匿名函数。

注：

1、$在JQuery库中，$是JQuery的别名，$()等效于就jQuery()。

2、$()是JQuery方法,赞可看作是JQuery的选择器，与css选择器相似（可做对比）。

3、var jQuery==$ =function(){} $()本质就是一个函数也就是 jQuery的核心函数。

4、只要是jQuery的对象都这样变量加上一个符号$ 方便识别：var $div = $("#")。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <title></title>  <script src="js/jquery.min.js"></script>  <script type="text/javascript">  /\*1、接收一个函数\*/  $(function(){  alert("sx");  //2、接收一个字符串你选择器，包装成Jquery对象并返回  //该对象保存了dom元素。  var $box1 = $(".box1");  console.log($box1);  //3接收一个字符代码片段，创建一个p标签，会执行代码，  //创建一个jquery对象，并返回，该对象保存了dom元素。  var $p = $("<p>字符串代码片段</p>");  console.log($p);  $box1.append($p);    //4、接收一个dom元素，会将dom元素包装成一个Jquery对象返回，保存了dom元素。  var span = document.getElementsByTagName("span");  var $span = $(span);  console.log($span);  var span1 = document.getElementsByTagName("span");  })    </script>  </head>  <body>  <div id="box1">  </div>  <span >  span元素  </span>          </body>  </html> |

## jQuery对象

jQuery对象是jQuery函数的一个实例，是一个类数组对象，数组中存放的是DOM对象，而DOM对象是Node的实例。对jQuery对象的操作实际上是对jQuery数组中的DOM对象的批量操作。jQuery对象和DOM对象可以相互转化。

jQuery对象的获取通常是使用选择器来获取的。比如，获取所有clss为one元素:$(".one");

并且拥有length属性，范围为0-length-1。

## 原生对象与jquery代理对象转换

|  |
| --- |
| $(传入的原生对象);  //原生对象转化成jQuery对象  var nav = document.getElementById("nav");  var $nav = $(nav);  alert($nav.get(0) == nav);//true  <div class="se-preview-section-delimiter"></div> |

## 静态方法

|  |
| --- |
| <script type="text/javascript">  window.onload = function(){  /\*静态方法\*/  //1、定义一个类  function AClass(){    }  //2、给这个类添加一个静态方法  //直接添加给类的就是静态方法  AClass().staticMethod = function(){  alert("staticMethod");  }  //3、调用静态方法  //通过类名调用  AClass().staticMethod();    /\*添加一个实例方法\*/  AClass().prototype.instanceMethod = function(){  alert("instanceMethod");  }  //创建实例方法  var a = new AClass();  //通过实例调用实例方法  a.instanceMethod();  }  </script> |

### each

|  |
| --- |
| <script type="text/javascript">  /\*  \* 参数1：遍历到的元素。  \* 参数2：当前遍历的元素。  \* 注：原生的foreach只能遍历数组，不能遍历伪数组  \*/  /\* var arr = [1,3,5,7,9];  var arr1 = {0:1,1:3,2:5,3:7,4:9,length:5};  arr.forEach(function(value,index){  console.log(index,value);  })\*/    $().ready(function(){  /\*1、利用Jquery的each静态方法遍历数组\*/  /\*  参数1：当前遍历到的索引  参数2：遍历到的元素  遍历为真数组或者伪数组  \* \*/  var arr1 = {0:1,1:3,2:5,3:7,4:9,length:5};  $.each(arr1, function(index,value) {  console.log(index);  console.log(value);  });    });  </script> |

### map

|  |
| --- |
| <script type="text/javascript">  var arr = [1,3,5,7,9];  var arr1 = {0:1,1:3,2:5,3:7,4:9,length:5};  //1、原生map方法遍历  /\*  参数1：当前遍历到的元素  参数2：当前遍历到的索引  参数3：遍历的数组  注意点：不能遍历weu数组  \* \*/  arr.map(function(value,index,array){  console.log(index,value,array);  });  //2、Jquery中的map  /\*  第一个参数：要遍历的数组  第二个参数：每遍历一个元素之后执行的回调函数  回调函数的参数：  第一个参数：遍历到的元素  第二个参数：遍历到的索引  和jQuery中的each静态方法一样，map静态方法也可以遍历伪数组  \* \*/  $.map(arr1,function(value,index){  console.log(index,value);  });  </script> |

### each和map的

|  |
| --- |
| //map和each的区别  /\*jQuery中的each静态方法和map静态方法的区别：  each静态方法默认的返回值就是，遍历谁就返回谁  each静态方法不支持在回调函数中对遍历的数组进行处理  map静态方法可以在回调函数中通过return对遍历的数组进行处理，  然后生成一个新的数组返回。  \*/  var arr1 = {0:1,1:3,2:5,3:7,4:9,length:5};  var res = $.map(arr1,function(value,index){  console.log(index,value);  return index+value;//这里可以对返回的值进行操作  });  var res1 = $.each(arr1, function(index,value) {  console.log(index);  console.log(value);  });  </script> |

### 其他方法

|  |
| --- |
| /\*$.trim()；  作用：去除字符串两端的空格  参数：需要去除空格的字符串  返回值：去除空格之后的字符串\*/  var str = " str ";  var str1 = $.trim(str);  alert(str1);  //isWindow()判断传入的对象是否为window  var res = $.isWindow(window);  alert(res);//true  //isArray：  //作用：判断传入的对象是否是真数组  //返回值：true/false  /\*$.isArray()；  作用：判断传入的对象是否是一个函数  返回值：true/false  注意点：  jQuery框架本质上是一个函数  \*/  //jquery实质为一个匿名函数  (function(window,undefined){    })(window);  test(); |

### holdReady()

$.holdReady() 函数用于暂停或恢复.ready() 事件的执行。

$.holdReady(true);表示暂停ready

$.holdReady(flase);表示开启ready；

## Jquery效果方法

### 隐藏和显示

|  |
| --- |
| $("#hide").click(function(){  $("p").hide();//选择器选中元素，然后隐藏。  });  $("#show").click(function(){  $("p").show();//选择器选中元素，然后显现。  }); |

# 选择器

## 常用的选择器

|  |  |  |
| --- | --- | --- |
| 选择器 | 语法 | 描述 |
| 标签选择器 | E{css规则} | 以文档元素作为选择符 |
| ID选择器 | #ID{css规则} | ID作为选择符 |
| 类选择器 | E.className{css规则} | class作为选择符 |
| 群组选择器 | E1,E2,E3{css规则} | 多个选择符应用同样的样式 |
| 后代选择器 | E F{css规则} | 元素E的任意后代元素F |

## 选择器引擎规则($('字符串'))

## 基本选择器

基本选择器是jQuery中最常用的选择器，也是最简单的选择器，它通过元素、id、class和标签名来查找DOM元素，在网页中id名称只能用一次，class允许重复使用。

|  |  |  |  |
| --- | --- | --- | --- |
| 选择器 | 描述 | 返回值 | 示例 |
| $("element") | 根据给定的元素名匹配元素 | 多个元素 | $("p")选取文档中所有的<p>元素 |
| $("#id") | 根据给定的id匹配一个元素 | 单个元素 | $("#test")选取文档中id为test的元素 |
| $(".class") | 根据给定的类名匹配元素 | 多个元素 | $(".test")选取文档中class为test的所有元素 |
| $("\*") | 匹配所有元素 | 多个元素 | #("\*")选取文档中的所有元素 |

## 关系选择器

|  |  |  |  |
| --- | --- | --- | --- |
| 关系选择器 | 格式 | 描述 | 示例 |
| 交集选择器 | $("element.className")或$("element#idName") | 将匹配到class为className的element元素合并后一起返回 | $("div.test")选取文档中<div>标签中class为test的所有元素 |
| 并集选择器 | $("element.className")或$("element#idName") | 将每一个选择器匹配到的元素合并后一起返回 | $("div, span")选取文档中所有<div>和<span>标签的元素 |
| 后代选择器 | $("ancestor descendant") | 选取ancestor元素里的所有descendant(后代)元素 | $("div span")选取<div>元素中的所有<span>后代子元素 |
| 子选择器 | $("parent > child") | 选取parent元素下的child（亲儿子）元素 | $("div span")选取<div>元素中的所有<span>后代子元素 |
| 相邻选择器 | $("prev + next") | 配所有紧接在prev元素后的next元素。 |  |
| 兄弟选择器 | $("prev ~ siblings") | 匹配prev元素之后的所有sibling元素 |  |

## 过滤选择器

过滤选择器主要是通过特定的过滤规则来筛选出所需的DOM元素，过滤规则与CSS中的伪类选择器语法相同，即选择器都以一个冒号(:)开头。

### 基本过滤选择器

|  |  |  |  |
| --- | --- | --- | --- |
| :first | 选取第一个元素 | 单个元素 | $("div:first")选取所有<div>元素中的第一个<div>元素 |
| :last | 选取最后一个元素 | 单个元素 | $("div:last")选取所有<div>元素中的最后一个<div>元素 |
| :not(selector) | 去除所有与给定选择器匹配的元素 | 多个元素 | $("input:not([type='file'])")选取type值不是“file”的所有<input>元素 |
| :even | 选取索引是偶数的所有元素，索引从0开始 | 多个元素 | $("li:even")选取索引是偶数的<li>元素 |
| :odd | 选取索引是奇数的所有元素，索引从0开始 | 多个元素 | $("li:odd")选取索引是奇数的<li>元素 |
| :eq(index) | 选取索引等于index的元素，索引从0开始 | 多个元素 | $("li:eq(1)")选取索引等于1的<li>元素 |
| :gt(index) | 匹配所有大于给定索引值的元素 | 多个元素 |  |
| :lt(index) | 匹配所有小于给定索引值的元素 | 多个元素 |  |
| :animated | 选取当前正在执行动画的所有元素 |  |  |
| focus | 选取当前正在获取焦点的元素 |  |  |

### 内容过滤选择器

|  |  |  |
| --- | --- | --- |
| :contains(text) | 选取含有文本内容为text的元素 |  |
| :empty | 选取不包含子元素获取文本的空元素 |  |
| :has(selector) | 选择含有选择器所匹配的元素的元素 |  |
| :parent | 选取含有子元素或者文本的元素 |  |

### 可见过滤选择器

|  |  |  |
| --- | --- | --- |
| :hidden | 选择所有不可见的元素 |  |
| :visible | 选取所有可见的元素 |  |

### 属性过滤选择器

|  |  |  |
| --- | --- | --- |
| [attribute] | 选取拥有此属性的元素 |  |
| [attribute=value] | 选取属性值为value值的元素 |  |
| [attribue^=value] | 选取属性的值以value开始的元素 |  |
| [attribue$=value] | 选取属性的值以value结束的元素 |  |

### 子元素过滤选择器

|  |  |  |
| --- | --- | --- |
| :nth-child(index/even/odd) | 选取每个父元素下的第index个子元素或者奇偶元素（index从1算起） |  |
| :first-child | 选取每个元素的第一个子元素 |  |
| :last-child | 选取每个元素的最后一个子元素 |  |

:nth-child()选择器是很常用：

 :nth-child(even)选择每个父元素下的索引值是偶数的元素 。

- :nth-child(odd)选择每个父元素下的索引值是奇数的元素 。

- :nth-child(2)选择每个父元素下的索引值是2的元素 。

- :nth-child(3n)选择每个父元素下的索引值是3的倍数的元素 (n从1开始)。

### 表单对象属性过滤选择器

|  |  |  |
| --- | --- | --- |
| :enabled | 选取所有可用元素 |  |
| :disabled | 选取所有不可用元素 |  |
| :checked | 选取所有被选中的元素（单选框、复选框） |  |
| :selected | 选取所有被选中的元素（下拉列表） |  |

### 表单选择器

|  |  |  |
| --- | --- | --- |
| :input | 选取所有input textarea select button元素 |  |
| :text | 选取所有单行文本框 |  |
| :password | 选取所有密码框 |  |
| :radio | 选取所有单选框 |  |
| :checkbox | 选取所有多选框 |  |
| :image | 选取所有的图像按钮 |  |
| :reset | 选取所有的重置按钮 |  |
| :file | 选取所有的上传域 |  |
| :submit | 选取所有的提交按钮 |  |
| :hidden | 选取所有的不可见元素 |  |

## 优化选择器

### 使用合适的选择器表达式可以提高性能、增强语义并简化逻辑。常用的选择器中，ID选择器速度最快，其次是类型选择器。

a. 多用ID选择器

b. 少直接使用class选择器

c. 多用父子关系，少用嵌套关系

d. 缓存jQuery对象

### 使用过滤器

jQuery提供了2种选择文档元素的方式：选择器和过滤器

a. 类过虑器：根据元素的类属性来进行过滤操作。

hasClass(className)：判断当前jQuery对象中的某个元素是否包含指定类名，包含返回true，不包含返回false

b. 下标过滤器：精确选出指定下标元素

eq(index)：获取第N个元素。index是整数值，下标从0开始

c. 表达式过滤器

filter(expr)/(fn)：筛选出与指定表达式/函数匹配的元素集合。

功能最强大的表达式过滤器，可接收函数参数，也可以是简单的选择器表达式

d. 映射 map(callback)：将一组元素转换成其他数组

e. 清洗 not(expr)：删除与指定表达式匹配的元素

f. 截取 slice(start,end)：选取一个匹配的子集

### 查找

向下查找后代元素

children():取得所有元素的所有子元素集合（子元素）

find():搜索所有与指定表达式匹配的元素(所有后代元素中查找)

查找兄弟元素 siblings()查找当前元素的兄弟

# 代理对象属性和样式操作

## 代理对象属性和样式操作

### attr(attribute)

格式:attr(name pro key,val fn)

作用：获取或者设置属性节点的值，返回值为String类型。

可以传递一个参数，也可以传递两个参数

如果传递一个参数，代表获取属性节点的值

如果传递两个参数，代表设置属性节点的值

注意点：

1、无论找到多少个元素，都只会返回第1个元素指定的属性节点的值。

2、如果是设置：找到多少个元素就会设置多少个元素。

3、如果是设置：如果设置的属性节点不存在，那么系统会自动新增。

4、可以同时设置多个属性值。

|  |
| --- |
| // 设置单个属性  $("div").attr("title", "我是一个div");  // 设置多个属性  $("img").attr({  "src" : "img/11.jpg",  "alt" : "我是一张风景图片"  });  // 获取alt属性值  console.log($("img").attr("src"));  // 获取title属性值  console.log($("div").attr("title")); |

从jQuery1.6开始，使用attr()获取这些属性的返回值为String类型，如果被选中(或禁用)就返回checked、selected或disabled，否则(即元素节点没有该属性)返回undefined。

jQuery认为：attribute的checked、selected、disabled就是表示该属性初始状态的值，property的checked、selected、disabled才表示该属性实时状态的值(值为true或false)。

### removeAttr(name)

删除属性节点

注意点：

会删除所有找到元素指定的属性节点。

如果要删除多个属性节点，在同一个字符串中间可以空格隔开。

|  |
| --- |
| // 删除alt属性值  $("img").removeAttr("alt");  // 删除title属性值  $("div").removeAttr("title");  //同时删除alt、title属性值。  $("div").removeAttr("title alt"); |

### prop(property)

在jQuery中，通过prop()方法操作属性，只能操作固有属性，特别用于：checked、disabled、selected等表单属性。注意：checked、disabled、selected等表单属性只能通过prop()方法来操作。

比如访问input的disabled属性的时候，会有些问题。在有些浏览器里，只要写了disabled属性就可以，有些则要写：disabled。所以，从1.6版开始，jQuery提供新的方法prop()来获取这些属性。使用prop()的时候，返回值是标准属性：true/false，比如$（#checkbox).prop(Cdisabled)，不会返回”disabled”或者“”，只会返回true/false。当然赋值的时候也是如此。这样，便统一了所有操作，无论是从语法上还是语义上。

使用原则：

第一个原则：只添加属性名称该属性就会生效应该使用pro()；

第二个原则：只存在true/false的属性应该使用pro()。

一般用于checked、disabled、selected属性的获取和设置。

其他的设置，和获取的用法与atrr()相同。

### removeProp

特点和attr相同。

### prop与attr的区别

从jQuery1.6开始，使用attr()获取这些属性的返回值为String类型，如果被选中(或禁用)就返回checked、selected或disabled，否则(即元素节点没有该属性)返回undefined。

jQuery认为：attribute的checked、selected、disabled就是表示该属性初始状态的值，property的checked、selected、disabled才表示该属性实时状态的值(值为true或false)。

1、很明显，attr和prop分别是单词attribute和property的缩写，并且它们均表示"属性"的意思。不过，在jQuery中，attribute和property却是两个不同的概念。attribute表示HTML文档节点的属性，property表示JS对象的属性。

|  |
| --- |
| <!-- 这里的id、class、data\_id均是该元素文档节点的attribute -->  <div id="message" class="test" data\_id="123"></div>    <script type="text/javascript">  // 这里的name、age、url均是obj的property  var obj = { name: "CodePlayer", age: 18, url: "http://www.365mini.com/" };  </script> |

2、在jQuery中，prop() 函数的设计目标是，用于设置或获取指定DOM元素(指的是JS对象，Element类型)上的属性(property)；attr() 函数的设计目标是，用于设置或获取指定DOM元素所对应的文档节点上（指HTML的元素，元素节点）的属性(attribute)。

<https://blog.csdn.net/chenchunlin526/article/details/77426796>

### addClass / removeClass

addClass() 为每个匹配的元素添加指定的类名，如需添加多个类，请使用空格分隔类名。 该方法不会移除已存在的 class 属性，仅仅添加一个或多个 class 属性。

|  |
| --- |
| // 添加一个class  $("div").addClass("box1");  // 添加多个class  $("div").addClass("box2 box3"); |

在项目开发中，我们可以通过jQuery的方式来给元素添加class，从而实现给元素添加CSS样式（比通过jQuery添加CSS样式效率更高）。

### removeClass()

removeClass() 方法从被选元素移除一个或多个类。如果没有规定参数，则该方法将从被选元素中删除所有类。

|  |
| --- |
| // 移除一个class  $("div").removeClass("box1");  // 移除多个class  $("div").removeClass("box1 box2");  // 移除全部class  $("div").removeClass(); |

### hasClass()

hasClass()可以用来判断元素中是否含有某个class。如果有，则返回true，否则返回false。

|  |
| --- |
| // 判断div中是否包含值为box的class  if($("div").hasClass("box")) {  // 如果存在，则移除这个class  $("div").removeClass("box");  } |

### toggleClass()

toggleClass() 对设置或移除被选元素的一个或多个类进行切换。

该方法检查每个元素中指定的类。如果不存在则添加类，如果已设置则删除之。这就是所谓的切换效果。

|  |
| --- |
| // 切换一个class  $("div").toggleClass("box");  // 切换多个class  $("div").toggleClass("box1 box2"); |

问题：添加类选择器覆盖问题。

## 操作原生DOM

### 操作属性

### 操作样式

### 操作类样式

### 获取DOM的子元素children属性

### DOM里面添加一个子元素

# 操作样式

## css()

### 逐个设置

$（"div").css("width","100px")；

$("div").css("height","100px")；

$（"div").css("background","red")；

### 链式设置

$（"div").css("width","100px").css("height","100px").css（"background"，"blue"）;

### 批量设置

$("#box").css({

"color": "red",

"font-size": "30px",

"text-align": "center"

});

### 获取样式值

// 获取背景颜色值

console.log($("#box").css("background-color"));

// 获取字体大小值

console.log($("#box").css("font-size"));

## width() 和 height() 方法

width() 方法设置或返回元素的宽度（不包括内边距、边框或外边距）。

height() 方法设置或返回元素的高度（不包括内边距、边框或外边距）。

|  |
| --- |
| // 1.设置宽高  $("div").css({"width" : "200px", "height" : "200px"});  $("div").width(300).height(300);  // 2.获取宽高  console.log($("div").css("width"));  console.log($("div").width());  console.log($("div").css("height"));  console.log($("div").height());  // 3.获取窗口的宽高  console.log("窗口宽度: " + $(window).width());  console.log("窗口高度: " + $(window).height());  // 4.获取文档的宽高  console.log("文档宽度: " + $(document).width());  console.log("文档高度: " + $(document).height()); |

## innerWidth() 和 innerHeight() 方法

innerWidth() 方法返回元素的宽度（包括内边距）。

innerHeight() 方法返回元素的高度（包括内边距）。

|  |
| --- |
| // 1.返回元素的宽度（包括内边距）  console.log($("div").innerWidth());  // 2.返回元素的高度（包括内边距）  console.log($("div").innerHeight()); |

## outerWidth() 和 outerHeight() 方法

outerWidth() 方法返回元素的宽度（包括内边距和边框）。

outerHeight() 方法返回元素的高度（包括内边距和边框）。

|  |
| --- |
| // 1.返回元素的宽度（包括内边距和边框）  console.log($("div").outerWidth());  // 2.返回元素的高度（包括内边距和边框）  console.log($("div").outerHeight()); |

outerWidth(true) 方法返回元素的宽度（包括内边距、边框和外边距）。

outerHeight(true) 方法返回元素的高度（包括内边距、边框和外边距）。

|  |
| --- |
| // 1.返回元素的宽度（包括内边距、边框和外边距）  console.log($("div").outerWidth(true));  // 2.返回元素的高度（包括内边距、边框和外边距）  console.log($("div").outerHeight(true)); |

# jQuery事件

事件三要素：事件源、事件、事件驱动程序

事件源，也就是需要操作的对象，例如：var box = $("#box");

事件，例如：onclick、onmouseenter等，例如：box.click(function () { 程序 });

事件驱动程序，就是接下来我们将要要学习的关于DOM的操作，例如：$(this).css("color", "red");

## 单击事件

|  |  |
| --- | --- |
| 事件 | 描述 |
| click |  |
| <script>  $(function () {  $("#box").click(function () {  alert("单击事件");  })  })  </script> | |

## 双击事件

|  |  |
| --- | --- |
| 事件 | 描述 |
| dblclick | 鼠标双击某个对象 |
| <script>  $(function () {  $("#box").dblclick(function () {  console.log("双击事件");  });  })  </script> | |

## 焦点事件

|  |  |
| --- | --- |
|  |  |
| focus | 元素获得焦点时触发 |
| blur | 元素失去焦点时触发 |
| <script>  $(function () {  $("input").focus(function () {  console.log("获得焦点");  });  $("input").blur(function () {  console.log("失去焦点");  });  })  </script> | |

## 改变事件

当用于 select 元素时，change 事件会在选择某个选项时发生。当用于 input 或 textarea 时，该事件会在元素失去焦点时发生。

实例：给文档中的<select>和<input>元素添加改变事件。

|  |  |
| --- | --- |
| 事件 | 描述 |
| change | 域的内容被改变触发，用于input、select和textarea标签。 |
| <body>  故乡：<select name="city" id="city">  <option value="1">北京</option>  <option value="2">上海</option>  <option value="3">广州</option>  <option value="4">武汉</option>  </select>  密码：<input type="password" placeholder="请输入密码">  <script>  $("select").change(function () {  console.log("选中的城市：" + $(this).val());  });  $("input").change(function () {  console.log("输入的内容：" + $(this).val());  });  </script>  </body> | |

## 鼠标事件

当鼠标指针穿过元素时，会发生 onmouseenter 事件。该事件大多数时候会与 onmouseleave 事件一起使用。onmouseover 事件和 onmouseout 事件一起使用。

与 onmouseover 事件不同，只有在鼠标指针穿过被选元素时，才会触发 onmouseenter 事件。如果鼠标指针穿过任何子元素，同样会触发 onmouseover 事件。

实例：给文档中的id="box"元素添加鼠标事件。

|  |  |
| --- | --- |
| 事件 | 描述 |
| onmouseover | 鼠标进入元素 |
| onmouseout | 鼠标离开元素 |
| mousedown | 鼠标被按下 |
| mouseup | 鼠标被松开 |
| mousemove | 鼠标在元素上面移动 |
| <script>  $(function () {  $("#box").mouseenter(function () {  console.log("鼠标进入");  });  $("#box").mouseleave(function () {  console.log("鼠标离开")  });  $("#box").mousedown(function () {  console.log("鼠标按下");  });  $("#box").mouseup(function () {  console.log("鼠标抬起");  });  $("#box").mousemove(function () {  console.log("鼠标移动");  });  })  </script> | |

## hover事件（jQuery新增）

语法: hover([fnOver,] fnOut)

参数1：鼠标移到元素上要触发的函数。

参数2：鼠标移出元素要触发的函数。

一个模仿悬停事件（鼠标移动到一个对象上面及移出这个对象）的方法。

当鼠标移动到一个匹配的元素上面时(mouseenter)，会触发指定的第一个函数。当鼠标移出这个元素时(mouseleave)，会触发指定的第二个函数。

|  |
| --- |
| <script>  $(function () {  $("#box").hover(function () {  console.log("鼠标进入");  }, function () {  console.log("鼠标离开");  });  })  </script> |

## 绑定/解绑事件

语法：$(selector).on(types, childSelector, data, handlerFunction)

参数1：必需。添加事件的类型，可以添加多个事件。

参数2：可选。规定只能添加到指定的子元素上的事件处理程序。

参数3：可选。规定传递到函数的额外数据。

参数4：可选。规定当事件发生时运行的函数。

on() 方法在被选元素及子元素上添加一个或多个事件处理程序。

事件绑定原理：事件捕获和事件冒泡是刚好相反的两个过程，事件捕获是从最顶端往下开始出发。

实例：给ul上绑定一个click事件，当点击li元素的时候触发回调函数。

|  |
| --- |
| <body>  <ul>  <li>我是一个li标签1</li>  <li>我是一个li标签2</li>  <li>我是一个li标签3</li>  <li>我是一个li标签4</li>  <li>我是一个li标签5</li>  <li>我是一个li标签6</li>  </ul>  <script>  $(function () {  // 这样做效率低下，需要给每一个li标签都添加一个点击事件  /\*$("ul > li").click(function () {  alert($(this).html());  });\*/  // 我们可以通过jQuery提供的on绑定来实现  $("ul").on("click", "li", function () {  alert($(this).html()); // 点击哪一行，就显示哪一行的内容  });  })  </script>  </body> |

实例：向函数传递数据。

|  |
| --- |
| <body>  <div class="box">事件绑定</div>  <script>  $("body").on("mouseover", ".box", {msg : "这样做效率高"}, function (event) {  console.log("box被点击了, " + event.data.msg);  });  </script>  </body> |

实例：父元素自己响应事件，不把事件传递给子元素。

|  |
| --- |
| <body>  <div id="box">事件绑定</div>  <script>  $("#box").on("click", function () {  console.log("click");  });  </script>  </body> |

## submit 事件

|  |
| --- |
| 通常表单的提交有两种方式，一是直接通过html的form提交，代码如下：  <form action="" method="" id="forms">  <input type="text" name="username" value="" />  <input type="password" name="pwd" value="" />  <input type="submit" value="提交"/>  </form>  但有时候我们会出于安全角度考虑，给用户输入的密码进行加密，方法一就没办法解决这个问题，这时候我们同常会选择另一种方法，使用javaScript来进行表单提交，代码入下：  <!--HTML-->  <form action="" method="" id="test\_form">  <input type="text" name="username" value="" />  <input type="password" name="pwd" value="" />  <button type="button" onclick='doSubmitForm()'>提交<button/>  </form>  <script>  var form = document.getElementById('test\_form');  //再次修改input内容  form.submit();  </script> |

# 文本节点

### html()

html() - 设置或返回所选元素的内容（包括 HTML 标记）。此方法类似于JavaScript中的innerHTML属性，可以用来读取或设置某个元素中的HTML内容，包括标签和元素。如过没有标签，返回文本值。

|  |
| --- |
| <body>  <p id="txt"></p>  <div id="box"></div>  <button id="txt-btn">给p标签添加文本内容</button>  <button id="box-btn">给div标签添加标签</button>  <script>  $("#txt-btn").click(function () {  // 给p标签添加文本内容  $("#txt").html("我是一个p标签");  // 获取p标签内容  console.log($("#txt").html());  });  $("#box-btn").click(function () {  // 给div标签添加标签  $("#box").html("<ul>\  <li>我是第一个li标签</li> \  <li>我是第二个li标签</li> \  <li>我是第三个li标签</li> \  <li>我是第四个li标签</li> \  </ul>");  // 获取div标签内容  console.log($("#box").html());  });  </script>  </body> |

### text()方法

text() - 设置或返回所选元素的文本内容。此方法类似于JavaScript中的innerText属性，可以用来读取或设置某个元素中的文本内容（JavaScript的innerText存在兼容性问题，而jQuery的text()方法支持所有的浏览器）。

|  |
| --- |
| <body>  <p id="txt"></p>  <div id="box"></div>  <button id="txt-btn">给p标签添加文本内容</button>  <button id="box-btn">给box标签添加文本内容</button>  <script>  $("#txt-btn").click(function () {  // 给p标签添加文本内容  $("#txt").text("我是一个p标签");  // 获取p标签内容  console.log($("#txt").text());  });  $("#box-btn").click(function () {  // 给p标签添加文本内容  $("#box").text("<p>我是一个p标签</p>");  // 获取p标签内容  console.log($("#box").text());  })  </script>  </body> |

### val()方法

val() - 设置或返回表单字段的值。此方法类似于JavaScript中的value属性，可以用来设置和获取元素的值。无论元素是文本框，下拉列表还是单选框，它都可以返回元素的值。如果元素为多选，则返回一个包含所有选择的值的数组。

|  |
| --- |
| <body>  姓名：<input type="text">  <button id="text-btn">设置文本输入框内容</button>  <script>  $("#text-btn").click(function () {  // 设置文本输入框内容  $("input:text").val("我是一个p标签");  // 获取文本输入框内容  console.log($("input:text").val());  });  </script>  </body> |