传智播客.黑马程序员

第3章 jQuery操作DOM

- jQuery操作元素样式的方法
- jQuery操作元素属性的方法
- jQuery操作元素内容的方法
- jQuery操作DOM节点的方法

学习目标

1. 掌握jQuery操作元素样式和属性的方法
2. 掌握jQuery操作元素内容和DOM节点的方法
3. 了解jQuery链式编程的使用

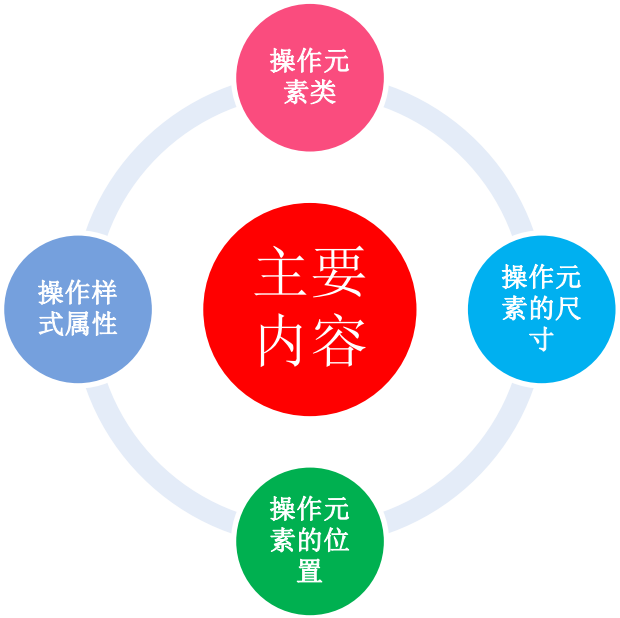
目录

返回目录

3.4 操作DOM节点

3.1 操作元素样式

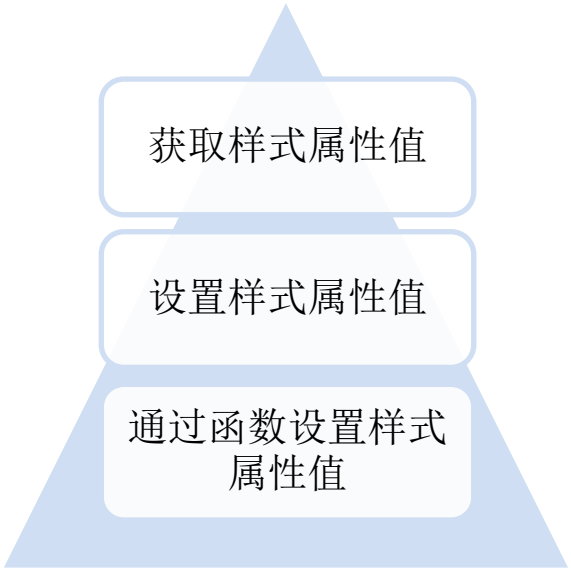
虽然传统的CSS样式表可以实现对元素的修饰，但如果可以任意操作元素现有的样式，或者为元素添加新的样式，Web页面就会变得灵活多变。



1. 操作样式属性

提供了专门操作元素样式属性的css()方法。

用途：利用该方法可以很容易的修改style样式里的属性。



1. 操作样式属性——获取样式属性值

获取样式属性：把需要获取的样式作为属性名传递到css()方法中，即样式值。

获取多个属性：$(selector).css(['property1', 'property2'...])。

获取单个属性：$(selector).css('property')。

原理：

- 以数组的形式传入css()方法中；
- 返回的结果是对象的属性名和属性值；

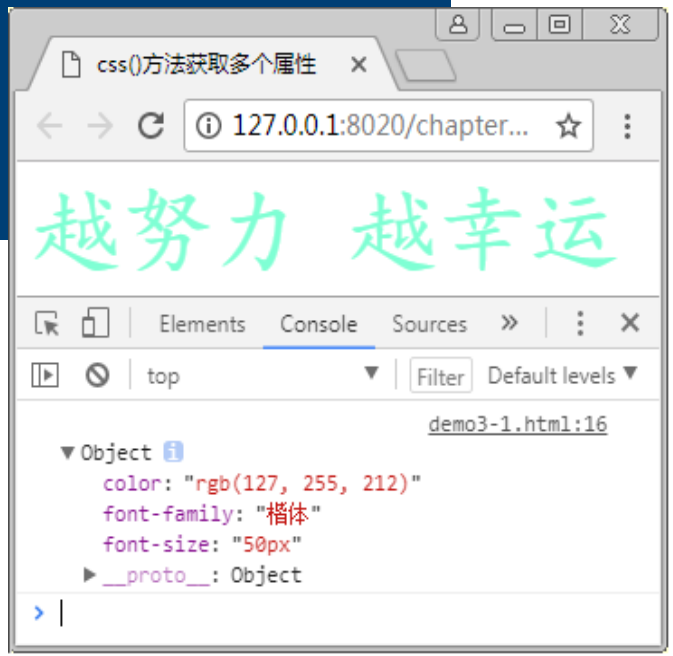
3.1 操作元素样式

1. 操作样式属性——获取样式属性值

案例演示：使用css()方法获取多个样式属性的具体返回结果。

```
// 获取多个属性的值
var obj = $('div').css(['color', 'font-size', 'font-family']);
// 在控制台查看obj
console.log(obj);
```

获取多个属性值



3.1 操作元素样式

1. 操作样式属性——设置样式属性值

设置样式属性：把需要获取的样式作为属性名传递到css()方法中，即样式值。

设置多个样式属性：$(selector).css({'property1': 'value', 'property2': 'value'...})；

设置单个样式属性：$(selector).css('property', 'value')；

原理：

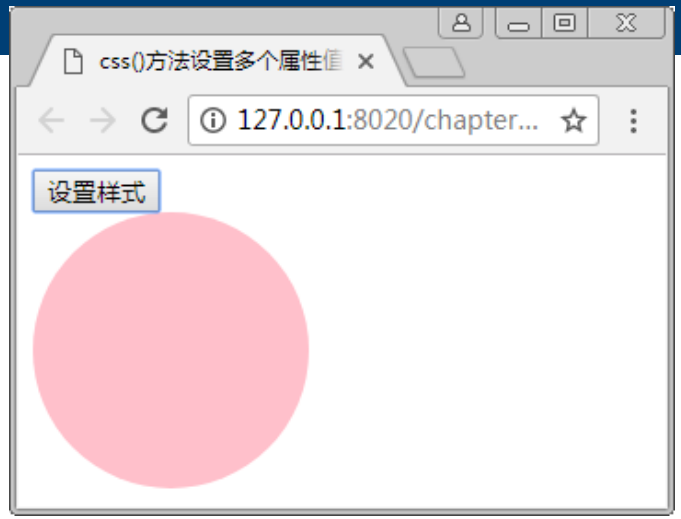
- css()方法中传入一个对象，该对象中包含多个键值对；
- 每个键值对都是元素样式属性名以及对应的属性值；

1. 操作样式属性——设置样式属性值

案例演示：如何设置多个样式属性值。

设置多个属性值

```
$('div').css({backgroundColor: 'pink', width: '150px',
   height: '150px', 'border-radius': '50%'});
});
```



1. 操作样式属性——通过函数设置样式属性值

函数设置样式属性：每个样式属性对应的value值还可以替换为函数形式。

元素的索引值从0开始

元素样式属性的当前值

```
$(selector).css('property', function(index, value) {
    return newValue;
}) ;
```

函数的返回值

1. 操作样式属性——通过函数设置样式属性值

函数设置多个样式属性：

```
$(selector).css({
    'property1': function(index, value1) {
        return newValue;
    },
    'property2': function(index, value2) {
        return newValue;
    }
});
```

1. 操作样式属性——通过函数设置样式属性值

通过获取元素的value属性，可以在原有的样式基础上改变样式。

举例：将div的宽度修改为原div宽度的2倍。

示例

```
$('div').css({width,function(index,value){
   return parseFloat(value)*2+'px';
});
```

1. 操作样式属性——通过函数设置样式属性值

通过获取元素的index属性，根据index的值来判断改变哪一个元素的样式。

举例：将第4个div的宽度修改为原div宽度的2倍。

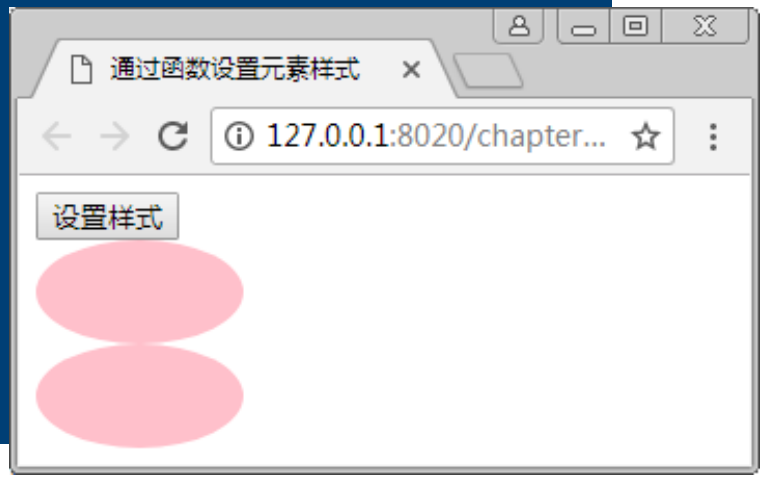
示例

```
$('div').css({width,function(index,value){
   if(index===3){
      return parseFloat(value)*2;
   }
});
```

1. 操作样式属性——通过函数设置样式属性值

默认样式

```
<input id="btn" type="button" value="设置样式">
<div></div>
<div></div>
$('div').css({backgroundColor: 'pink',
   width: '100px',
   height: '50px',
   'border-radius': '50% ‘
});
```



1. 操作样式属性——通过函数设置样式属性值

通过函数设置样式的属性值

```
width: function(index, value) {
        return parseFloat(value) * 2;
},
height: function(index, value) {
        if (index === 1) {
                return parseFloat(value) * 2;
        }
 }
```

2. 操作元素类

jQuery提供了专门的操作类的方法，包括添加类、移除类、切换类以及判断某个类是否存在等常用的方法。

方法	描述
addClass()	将指定的类添加到匹配元素中
removeClass()	从所有匹配的元素中删除全部或者指定的类
toggleClass()	对设置或移除被选元素的一个或多个类进行切换
hasClass()	确定是否有匹配的元素被分配了给定类

2. 操作元素类——addClass()方法

addClass()方法：在保留已有的样式上，再去添加新的一些样式。

添加多个类：使用空格隔开，调用addClass()方法后，元素将具备这三个类定义的样式。

```
$(selector).addClass('c');
```

添加单个类“c”

```
$(selector).addClass('c1 c2 c3');
```

添加多个类c1，c2，c3

2. 操作元素类——removeClass()方法

在网页开发中，有时需要移除一些元素已经存在的类，需要用到removeClass()方法。

removeClass()的参数：

- 是可选的，不传递参数时，会移除当前元素的所有类名。
- 一次移除指定的多个类时，需要使用空格隔开。

2. 操作元素类——removeClass()方法

语法展示：

```
$(selector).removeClass();
```
移除所有类

```
$(selector).removeClass('c1');
```
移除单个类

```
$(selector).removeClass('c1 c2  c3');
```
移除多个类

2. 操作元素类——removeClass()方法

案例演示：获取当前节点的html代码，可以看到div元素class属性的改变。

```
<script>
  var div = $('div');
  div.addClass('c1');
  div.removeClass('c2 c3');
  div.removeClass('c1');
  div.removeClass();
</script>
```

添加单个类

移除多个类

移除单个类

移除全部类

2. 操作元素类——removeClass()方法

注意

css()方法与addClass()、removeClass()方法虽然都可以改变元素的样式，但适用的场景不同。建议使用addClass()、removeClass()对类进行操作，而css()方法适合CSS样式值不固定的情况。例如，为元素随机生成背景色。

2. 操作元素类——toggleClass()方法

toggleClass()方法：在网页开发中，当用户在多个标签中，选择一个标签时，会增加特定的样式，否则，取消该样式。

语法：

- $(selector).toggleClass('c')，参数c表示一个自定义的类。
- 指定元素中若没有c，则添加，否则执行移出操作。
- addClass()和removeClass()方法虽然可以实现，但不够简便。
- 直接对元素的类进行切换。

2. 操作元素类——toggleClass()方法

案例演示：匹配到的元素对指定的类名进行切换无则加，否则去掉。

```
<script>
  var div = $('div');
  div.toggleClass('c');
  console.log(div[0].outerHTML);
  div.toggleClass('c');
  console.log(div[0].outerHTML);
  div.toggleClass('c');
  console.log(div[0].outerHTML
</script>
```

2. 操作元素类——toggleClass()方法

toggleClass()方法：还支持使用第2个参数手动控制类的添加或移除。

参数：是一个布尔值，若值为true表示添加类，值为false表示移除类。

```
$(selector).toggleClass('c', true);
```
添加类

```
$(selector).toggleClass('c', false);
```
移除类

```
$(selector).toggleClass('c', 条件判断);
```
条件返回值

2. 操作元素类——toggleClass()方法

案例演示：还可以通过条件判断的返回值情况设置。

示例

```
var count = 0;
$('div').click(function() {
    $(this).toggleClass('c', ++count % 3 === 0);
});
```

多 学 一 招

在操作元素类的方法中使用函数

在上面讲解的jQuery提供的用于操作元素类的addClass()方法、removeClass()方法和toggleClass()方法，都支持使用函数作为参数，通过函数的返回值来操作元素的类。

示例

```
$('div').addClass(function(index, value) {
    console.log('元素的索引：' + index);
    console.log('元素原来的class值：' + value);
    return 'item-' + index; // 将返回值作为要添加的class
});
```

2. 操作元素类——hasClass()方法

在网页开发过程中，有时需要判断元素的某个类是否存在，然后执行某些操作。

举例：结合其他方法，实现切换效果等。

语法：$(selector).hasClass('c')；

分析：必选参数c用于检测该类名是否存在，存在时返回true，否则返回false。

2. 操作元素类——hasClass()方法

案例演示：首先为div元素注册单击事件，当用户单击div元素时，判断div元素是否有类名“c”，如果有则移除，没有则添加。

示例

```
$('div').click(function() {
    if ($(this).hasClass('c')) {
        $(this).removeClass('c');
    } else {
        $(this).addClass('c');
    }
});
```

3. 操作元素的尺寸

元素尺寸的操作是Web开发中常用的功能之一，例如登录框的拖拽特效、图片的放大等功能。

方法	描述
width()	获取或设置元素的宽度
height()	获取或设置元素的宽度
innerWidth()	获取元素的宽度(包括内边距)
innerHeight()	获取元素的高度(包括内边距)
outerWidth()	获取元素的宽度(包括内边距和边框)

3. 操作元素的尺寸

方法	描述
outerHeight()	获取元素的高度(包括内边距和边框)
outerWidth(true)	获取元素的宽度(包括内边距、边框和外边距)
outerHeight	获取元素的高度(包括内边距、边框和外边距)

3. 操作元素的尺寸——操作元素的宽度

width()方法：包括设置元素的宽度和获取元素的宽度。

示例

```
$(selector).width();          // 获取宽度
$(selector).width('30px');    // 设置宽度
```

3. 操作元素的尺寸——操作元素的宽度

注意

在设置宽度时，参数的引号可以省略。但是如果不添加引号，里面传递的只能是数字，不能包含单位，如果包含单位就会出现语法错误。

3. 操作元素的尺寸——操作元素的高度

height()方法：包括设置元素的高度和获取元素的高度。

示例

```
$(selector).height();                    // 获取高度
$(selector).height('30px');              // 设置高度
```

3. 操作元素的尺寸——操作元素的高度

案例演示：

```
div {
    width: 100px; height: 100px;
    background: darkorange;
    border-radius:25%;color: aliceblue;
    text-align: center;padding: 10px;
    line-height: 25px;
}
<input type="button" value="按钮">
<div>只要努力，人生的长度和宽度都可以改变</div>
```

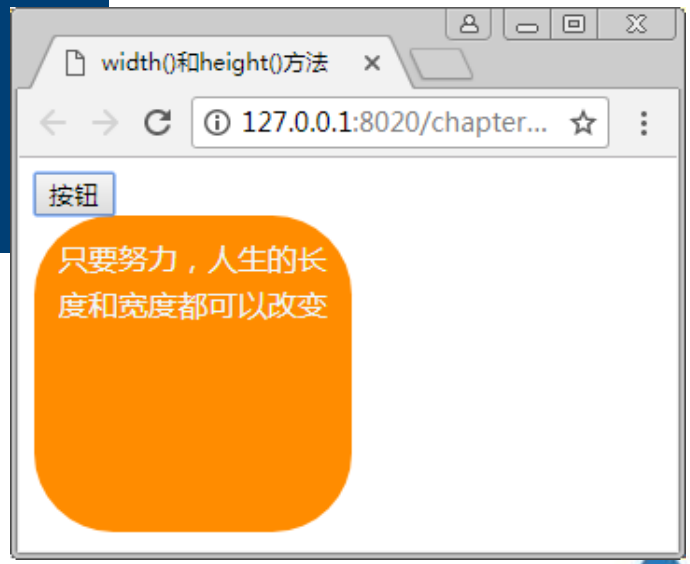
默认页面

3. 操作元素的尺寸——操作元素的高度

使用width()方法和height()方法在原div元素宽高的基础上增加了50像素。

第一次点击

```
$('input').click(function() {
    $('div').width($('div').width() + 50);
    $('div').height($('div').height() + 50);
});
```

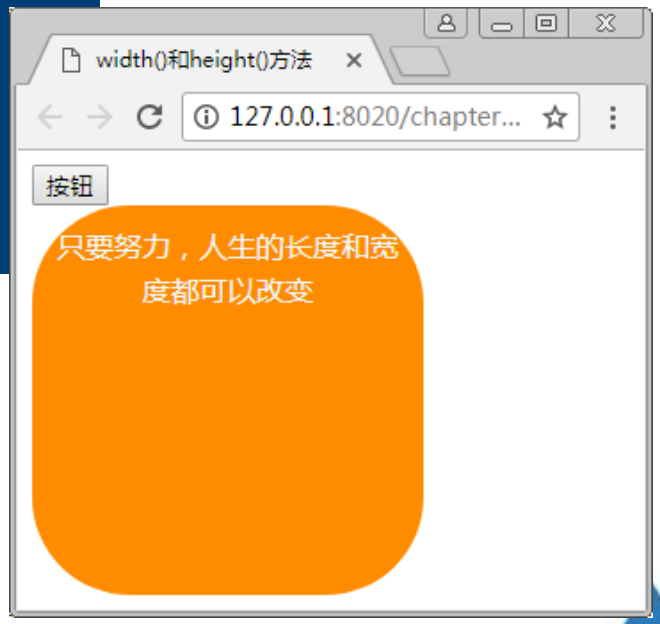


3. 操作元素的尺寸——操作元素的高度

width()和height()方法：每次获取的div宽高，都是当前div显示时的宽高。

第二次点击

```
$('input').click(function() {
    $('div').width($('div').width() + 50);
    $('div').height($('div').height() + 50);
});
```



3. 操作元素的尺寸——获取元素的内部宽度和高度

jQuery中innerWidth()和innerHeight()方法也是用于获取元素的宽度和高度。

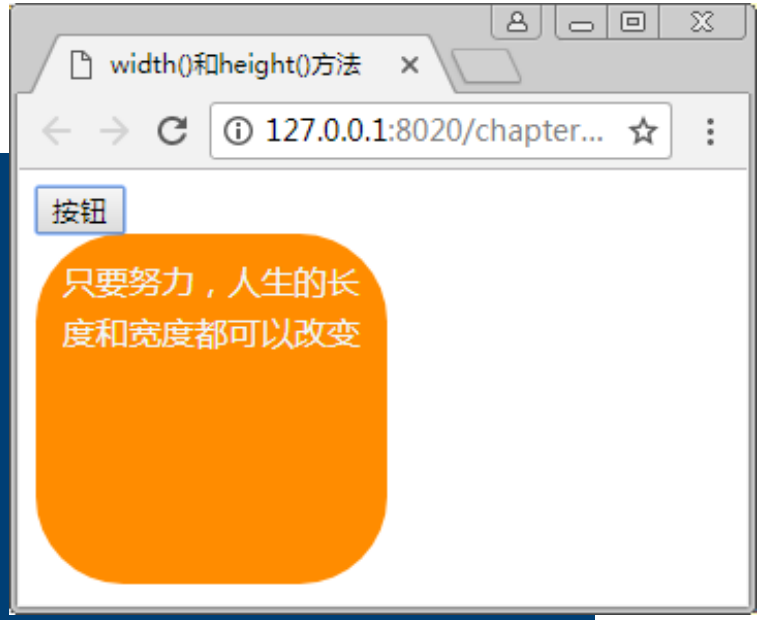
与width()方法和height()方法不同的是：这两个方法包括元素的内边距。

3. 操作元素的尺寸——获取元素内部的宽度和高度

案例演示：

默认页面

```
div {
    width: 100px;
    height: 100px;
    background: pink;
    padding: 10px;
}
<input type="button" value="获取元素的宽高">
<div>你是自己人生的导演</div>
```

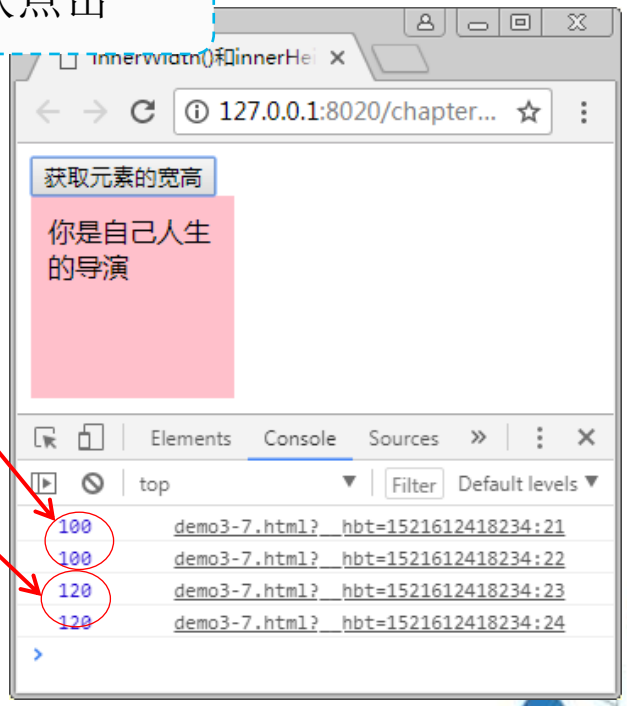


3. 操作元素的尺寸——获取元素内部的宽度和高度

前两个是div元素本身的宽高值，后两个值是div元素内部的宽高值；

```
$('input').click(function() {
    console.log($('div').width());
    console.log($('div').height());
    console.log($('div').innerWidth());
    console.log($('div').innerHeight());
});
```

第一次点击



3. 操作元素的尺寸——获取元素外部的宽度和高度

outerWidth()和outerHeight()方法也可获取元素的宽度和高度。

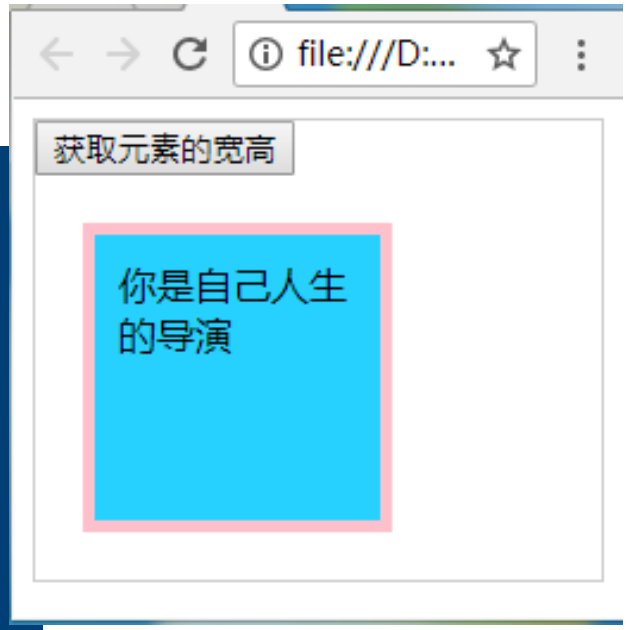
与width()方法和height()方法不同的是：

- 只是除了元素本身的宽高值，还包括内边距和边框值。
- 若在调用方法时传递参数true，获取到的结果中还会包含元素的外边距。

3. 操作元素的尺寸——获取元素外部的宽度和高度

案例演示：

默认页面



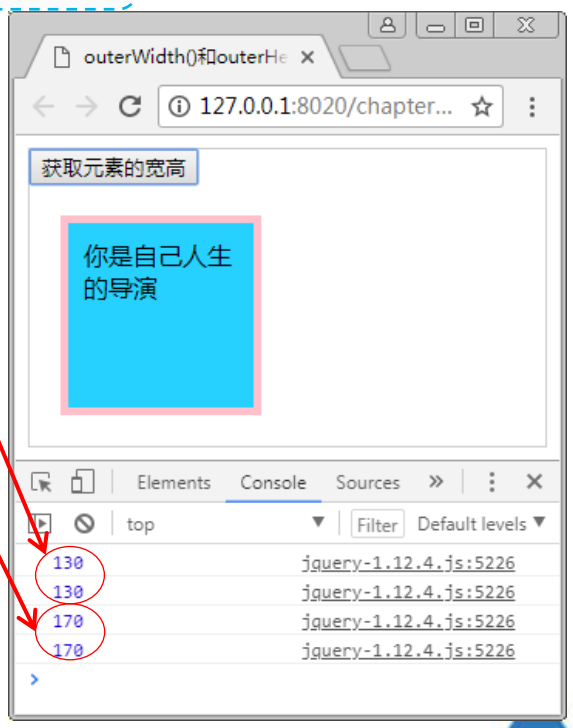

```
body {border: 1px solid #ccc;}
div {
        width: 100px;height: 100px;
        background: #27d1ff;padding: 10px;
        border: 5px solid pink;margin: 20px;
    }
  <input type="button" value="获取元素的宽高">
  <div>你是自己人生的导演</div>
```

3. 操作元素的尺寸——获取元素外部的宽度和高度

案例演示：

对比

```
$('input').click(function() {
    console.log($('div').outerWidth());
    console.log($('div').outerHeight());
    console.log($('div').outerWidth(true));
    console.log($('div').outerHeight(true));
});
```

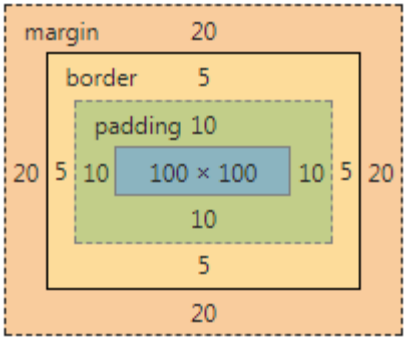


3. 操作元素的尺寸——获取元素外部的宽度和高度

div元素的外边距margin： 上下左右都是20像素。

带有参数true的outerWidth()方法和outerHeight()方法：

- 与不带有参数的这两个方法取得的外部元素宽高值，相差了40像素；
- 这40像素就是div元素的外边距。



4. 操作元素的位置

jQuery提供了专门的操作元素位置的方法，可以操作元素在页面中的位置以及相对滚动条的位置等，包括offset()、position()、scrollLeft()等方法。

方法	描述
offset()	获取匹配元素的第一个元素的坐标位置，或设置每个元素的坐标
offsetParent()	获取距离匹配元素最近的含有定位信息的元素
position()	获取匹配元素相对父元素的偏移
scrollLeft()	获取或设置匹配元素相对滚动条左侧的偏移
scrollTop()	获取或设置匹配元素相对滚动条顶部的偏移

4. 操作元素的位置——offset()方法

offset()方法：可以获取到匹配元素中的第一个元素在当前页面的坐标位置。

语法：$(selector).offset()。

调用offset()方法：

- 会返回匹配到的第一个元素在整个页面的偏移位置left和top。
- left和top分别表示元素距离浏览器的左偏移和上偏移。
- 若元素的样式属性display设置为none，则获取到的值为0。

4. 操作元素的位置——offset()方法

举例：获取和设置div元素的位置。

```
// 获取div的top和left值
console.log($('div').offset().left);
console.log($('div').offset().top);
// 设置div的left和fop值
// 向下移动100像素并向右移动200像素
$('div').offset({left: 200, top: 100});
// 向右移动200像素
$('div').offset({left: 200});
```

4. 操作元素的位置——offset()方法

注意

jQuery提供的position()方法也可以获取匹配元素中的第一个元素在当前页面的坐标位置。offset()方法与position()方法的区别在于，前者获取元素相对于当前窗口的偏移；后者获取元素相对于父元素（含有定位）的偏移，当父元素没有设置定位时，后者的左右与前者等价。

4. 操作元素的位置——offsetParent()方法

offsetParent()方法：会返回距离指定元素最近的“被定位”的祖辈元素对象。

position属性值为：

- relative、absolute或fixed；
- 不包括position属性的默认值static；

语法：$(selector).offsetParent()。

4. 操作元素的位置——offsetParent()方法

举例：为距离div元素最近的“被定位”的祖辈元素对象，设置背景色。

示例

```
$('div').offsetParent().css('background-color', 'green');
```

4. 操作元素的位置——scrollLeft()和scrollTop()方法

scrollLeft()和scrollTop()方法：可以获取或者设置指定元素相对滚动条左侧和顶部的偏移值。

语法：

语法

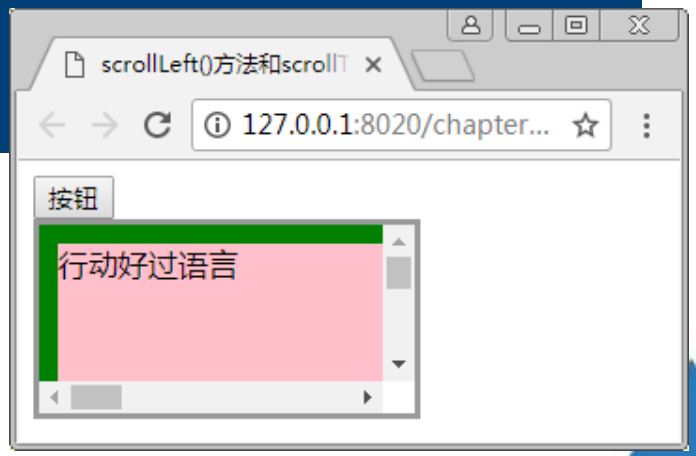
```
$(selector).scrollLeft();          // 获取元素相对左侧的偏移值
$(selector).scrollLeft(value);     // 设置元素相对左侧的偏移值
```

4. 操作元素的位置——scrollLeft()和scrollTop()方法

案例演示：

默认页面

```
div {
        background: green;border: 3px solid #999;
        width: 200px;height: 100px;overflow: auto;
}
p {background: pink;margin: 10px;width: 1000px;height:  1000px;}
<input type="button" value="按钮">
<div><p>行动好过语言</p></div>
```



4. 操作元素的位置——scrollLeft()和scrollTop()方法

案例演示：

- 设置div元素相对滚动条的左偏移和上偏移。
- 滚动条卷去的距离就是500像素和50像素。

```
$('input').click(function() {
    $('div').scrollTop(500);
    $('div').scrollLeft(50);
});
```

设置相对滚动条偏移

5. 案例高亮显示图片——案例展示

网站中一般都会有宣传或者展示图片的模块，作用是宣传人物或者产品等信息。图片展示的效果有很多。其中，高亮展示图片是常用的一种效果。

5. 案例高亮显示图片——案例分析

案例分析思路：

① HTML结构；

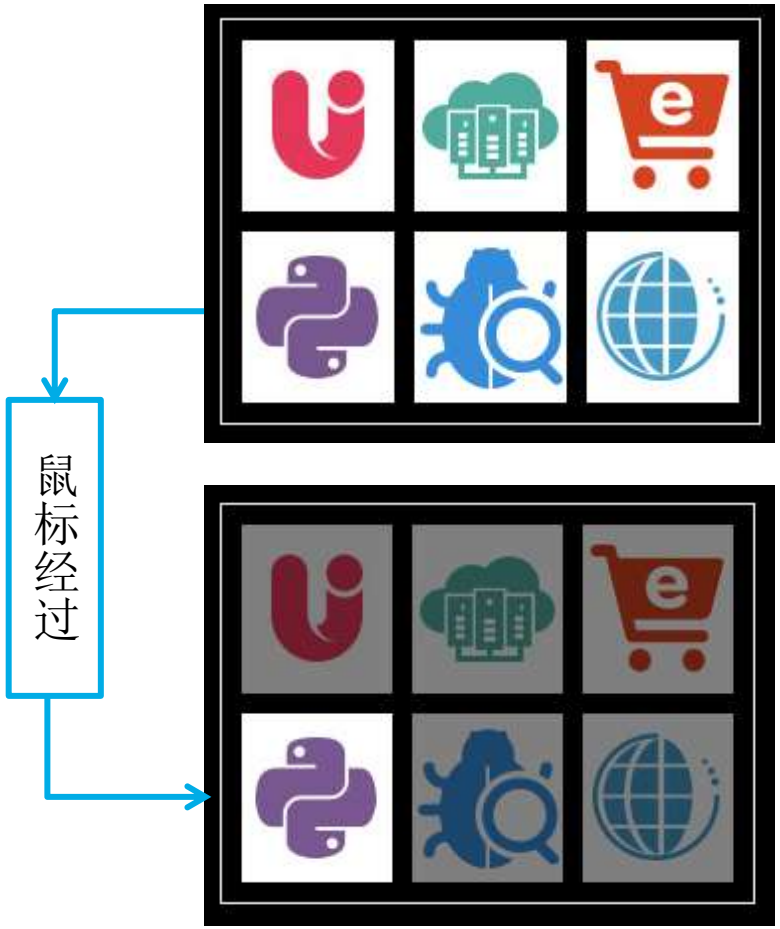
② jQuery特效；

5. 案例高亮显示图片——案例分析

案例实现思路：

① 设计HTML结构与样式；

② 添加jQuery特效；

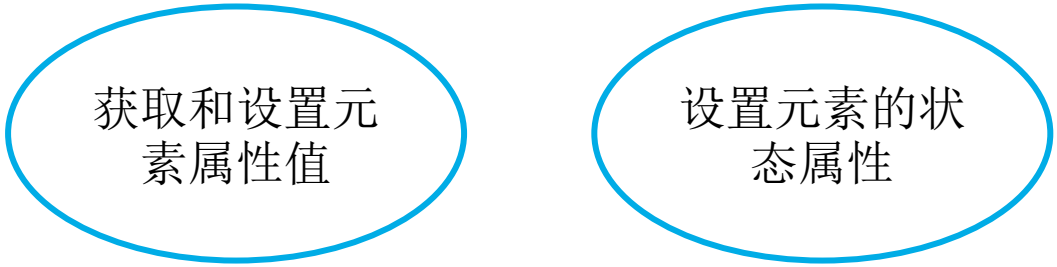


3.2 操作元素属性

元素属性是指当前元素节点的属性，

常用的元素属性：

- 有id、value、type；
- 以及用于标识元素状态的checked、disabled等；



1. 获取和设置元素的属性值——获取元素属性值

获取元素属性值：即把需要获取的属性名传递到attr()方法中。

语法：$(selector).attr('property')。

语法分析：

- property可以是元素的样式属性，如style等；
- 也可以是其他属性，如value等；
- 利用attr()方法可以操作元素的任意属性；

1. 获取和设置元素的属性值——设置元素属性值

设置元素属性即通过给元素设置属性名和属性值来改变元素。

设置多个属性：$(selector).attr({‘property1’: ‘value1’,‘property2’: ‘value2’,……});

设置单个属性：$(selector).attr('property', 'value');

语法分析：

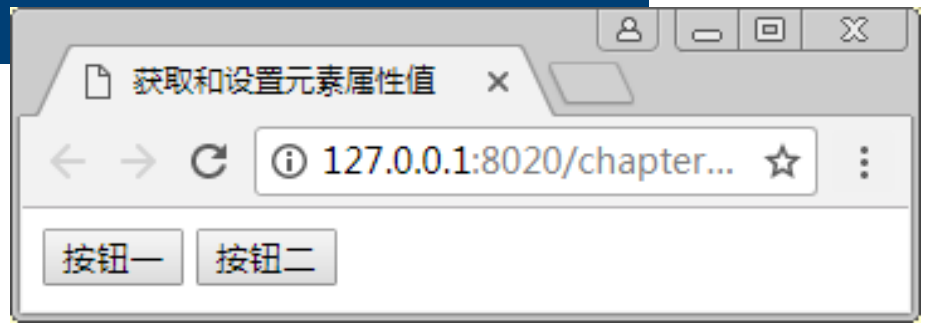
- property表示属性的名称；
- value表示属性的值；
- 具体用法和前面学习的css()方法相似；

1. 获取和设置元素的属性值——设置元素属性值

案例演示：attr()方法设置和获取元素的属性的使用。

默认页面

```
<input type="button" value="按钮一" class="btn1">
<input type="button" value="按钮二" class="btn2">
```



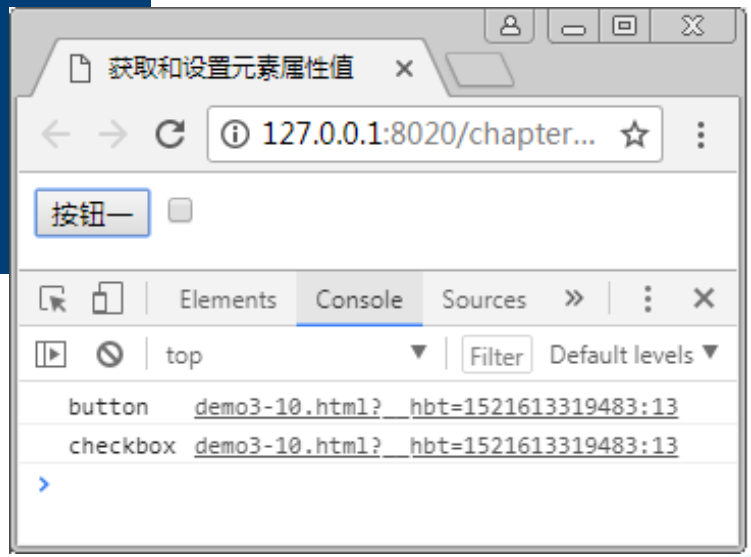
3.2 操作元素属性

1. 获取和设置元素的属性值——设置元素属性值

案例演示：将 “按钮二” type类型设置成checkbox，并输出。

attr('type', 'checkbox')

```
$('.btn1').click(function() {
    console.log($('.btn2').attr('type'));
    $('.btn2').attr('type', 'checkbox');
});
```



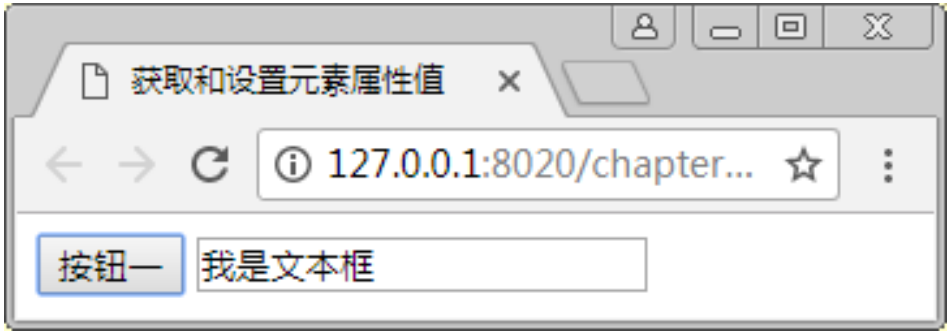
1. 获取和设置元素的属性值——设置元素属性值

案例演示：

- 将class值等于btn2的type属性设置为text；
- 将value属性的默认值设置为“我是文本框”；

attr({多个属性})

```
$('.btn2').attr({'type': 'text', 'value': '我是文本框'});
```



2. 设置元素的状态属性

jQuery中attr()方法的参数为元素状态属性时，可以用于设置元素状态；

举例：复选框是否选中的状态，文本框或者提交按钮的启用与禁用状态等。

属性	描述
checked	获取或设置表单元素的选中状态
disabled	获取胡设置表单元素的禁用状态
selected	获取或设置下拉框的选中状态

2. 设置元素的状态属性

案例演示：

- 第1个没有定义checked属性，默认未选中；
- 其余两个默认都是选中的状态；

默认状态

```
<input type="checkbox">
<input type="checkbox" checked="">
<input type="checkbox" checked="checked">
```

2. 设置元素的状态属性

案例演示：

- 第1个input控件会返回undefined，其余两个input控件则返回checked。
- 无论将checked属性设置值为checked还是true，都可以设置元素为选中状态。

获取表单元素的选中状态

```
$('input:eq(0)').attr('checked');
$('input:eq(1)').attr('checked');
$('input:eq(2)').attr('checked');
```

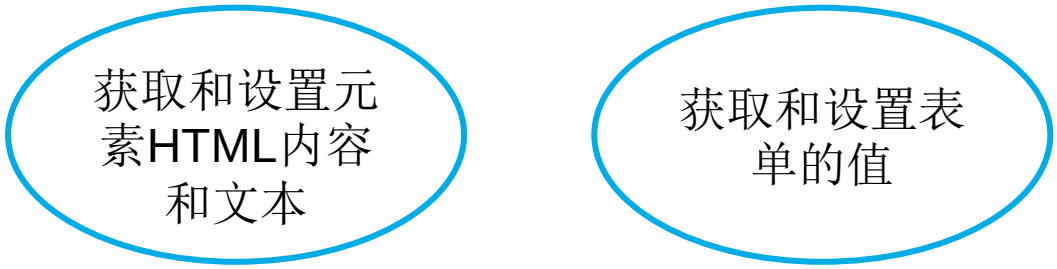
设置表单元素的选中状态

```
$('input').attr('checked',
                'checked');
$('input').attr('checked', true);
```

3.3 操作元素内容

jQuery中还提供了一些方法，可以操作元素内容。

举例：设置或者返回元素的文本值、表单的字段值等。



1. 获取和设置元素的HTML内容和文本

html()和text()方法都可以用来为元素设置文本内容。

区别：html()方法操作的元素内容包含标签，text()方法操作的内容不含标签。

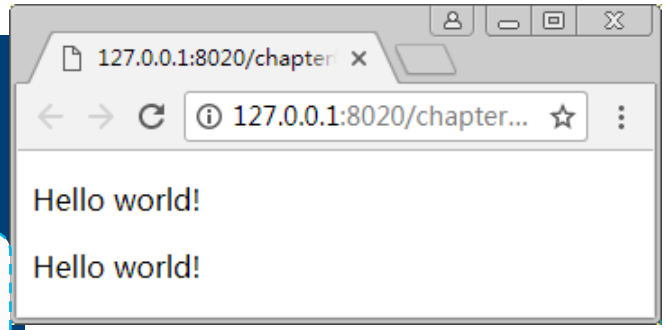
方法	参数	返回值
html()	无参数	用于获取元素的HTML内容
	字符串	用于设置元素的HTML内容
text()	无参数	用于获取元素的文本内容
	字符串	用于设置元素的文本内容

3.3 操作元素内容

1. 获取和设置元素的HTML内容和文本

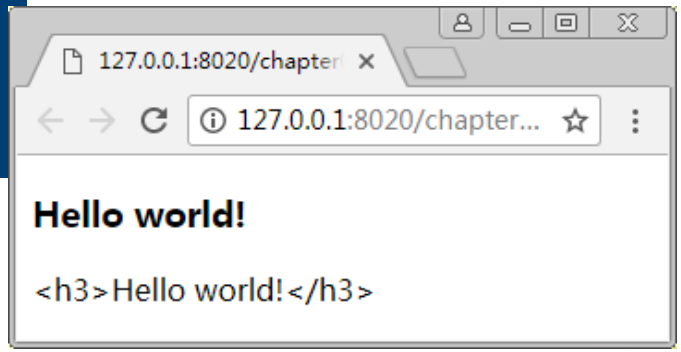
例如：分别使用html()方法和text()方法为p元素设置文本内容。

```
$('#p1').html('Hello world!');
$('#p2').text('Hello world!');
```



对比

```
$('#p1').html('<h3>Hello world!</h3>');
$('#p2').text('<h3>Hello world!</h3>');
```



2. 获取和设置表单的值

val()方法：

- 可以获取和设置表单元素的值；
- 相当于JavaScript中input控件对象value属性的作用；
- value表示表单元素的value属性的值，selector一般是指表单元素。

```
$(selector).val();
```
→ 获取表单元素的值

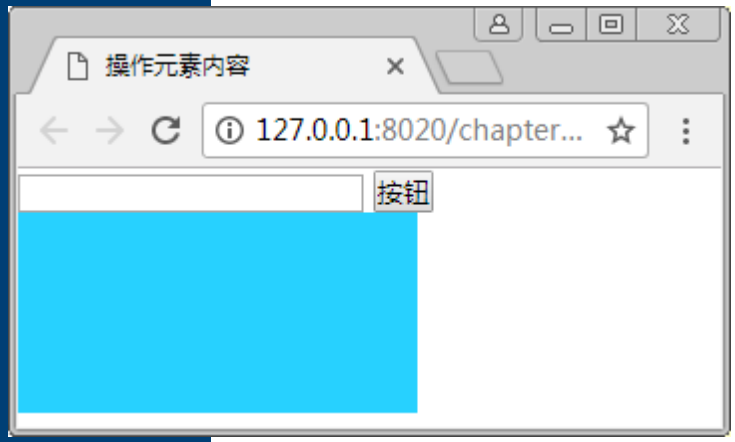
```
$(selector).val(value)
```
→ 设置表单元素的值

2. 获取和设置表单的值

案例演示：

```
* {margin: 0;padding: 0;}
    div {
        width: 200px;height: 100px;
        background: #27d1ff;
    }
<input type="text" class="txt">
<input type="button" value="按钮" class="btn">
<div></div>
```

默认页面



2. 获取和设置表单的值

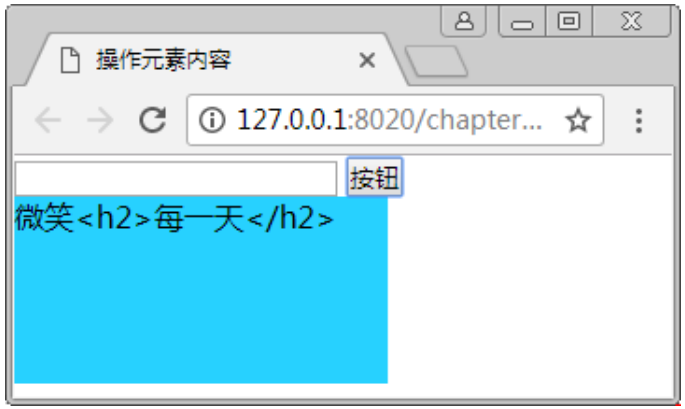
案例演示：

重置div元素内容：$('div').text($('.txt').val())。

输入框清空：$('.txt').val('')。

text()

```
$('.btn').click(function() {
    $('div').text($('.txt').val());
    $('.txt').val('');
});
```



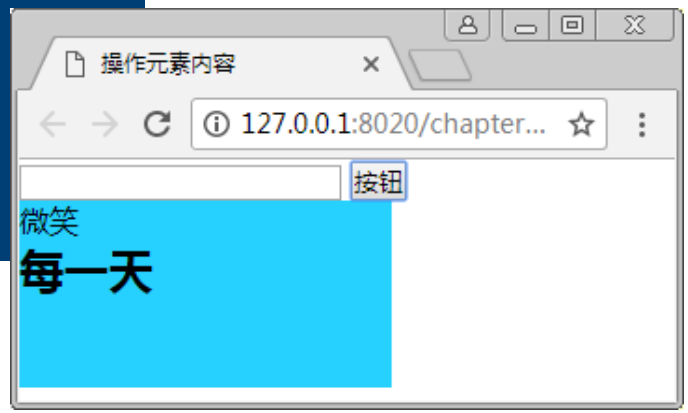
2. 获取和设置表单的值

案例演示：

- 重置div元素中的内容：$('div').html($('.txt').val())。

html()

```
$('.btn').click(function() {
    $('div').html($('.txt').val());
    $('.txt').val('');
});
```



【案例】留言板

在实际的网站中，经常会提供用户留言功能，例如论坛或微博等。

功能：

① 用户在留言版上输入内容；

② 提交后会在页面中展示，并且显示发表留言者的用户名。

③ 下一次发表新留言的时候，或者其他用户发表了留言，都会在最上面显示。

1. 案例留言板——案例展示

留言板组成部分：

- 留言列表：用于存放所有用户发布的留言；
- 编辑区域：是一个为用户输入留言提供的；
- “发表留言”按钮：用于发布用户输入的留言内容；

1. 案例留言板——案例展示

单击发布留言：就会在留言列表的最上方显示用户名以及用户输入的内容。

需要注意的是：最新的留言总是显示留言列表的最上面。

2. 案例留言板——案例分析

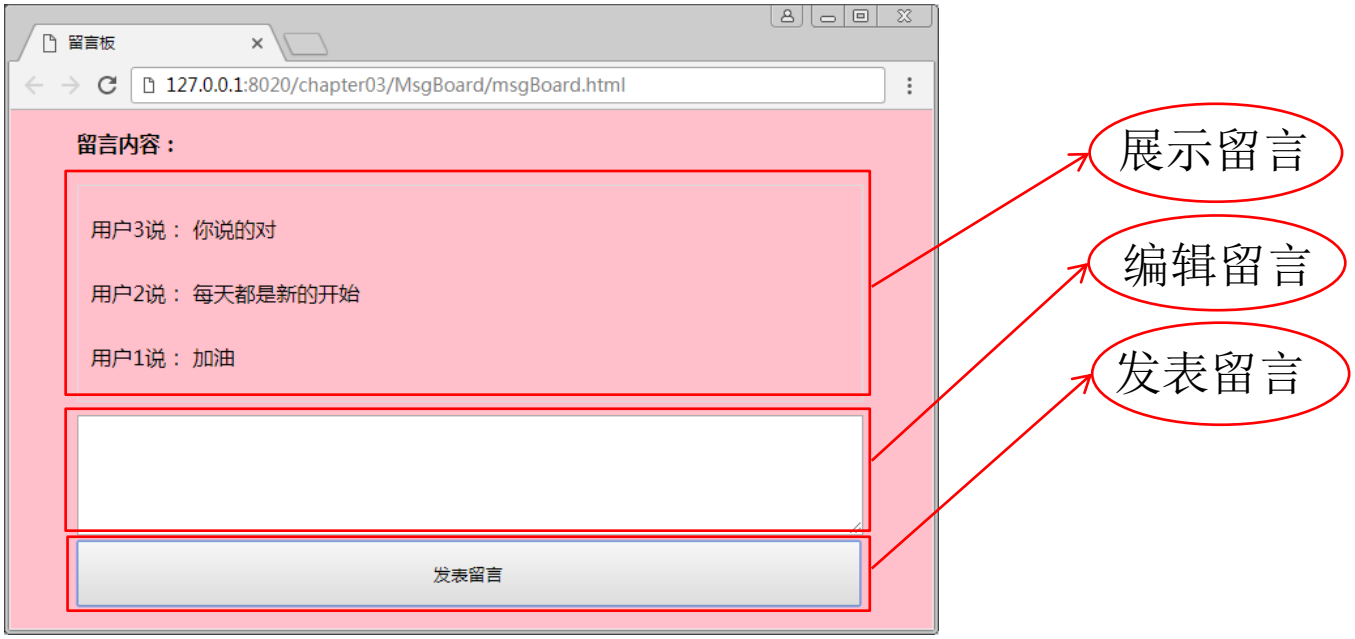
代码实现思路：

① HTML结构；

2. 案例留言板——案例分析

代码实现思路：

② jQuery特效。

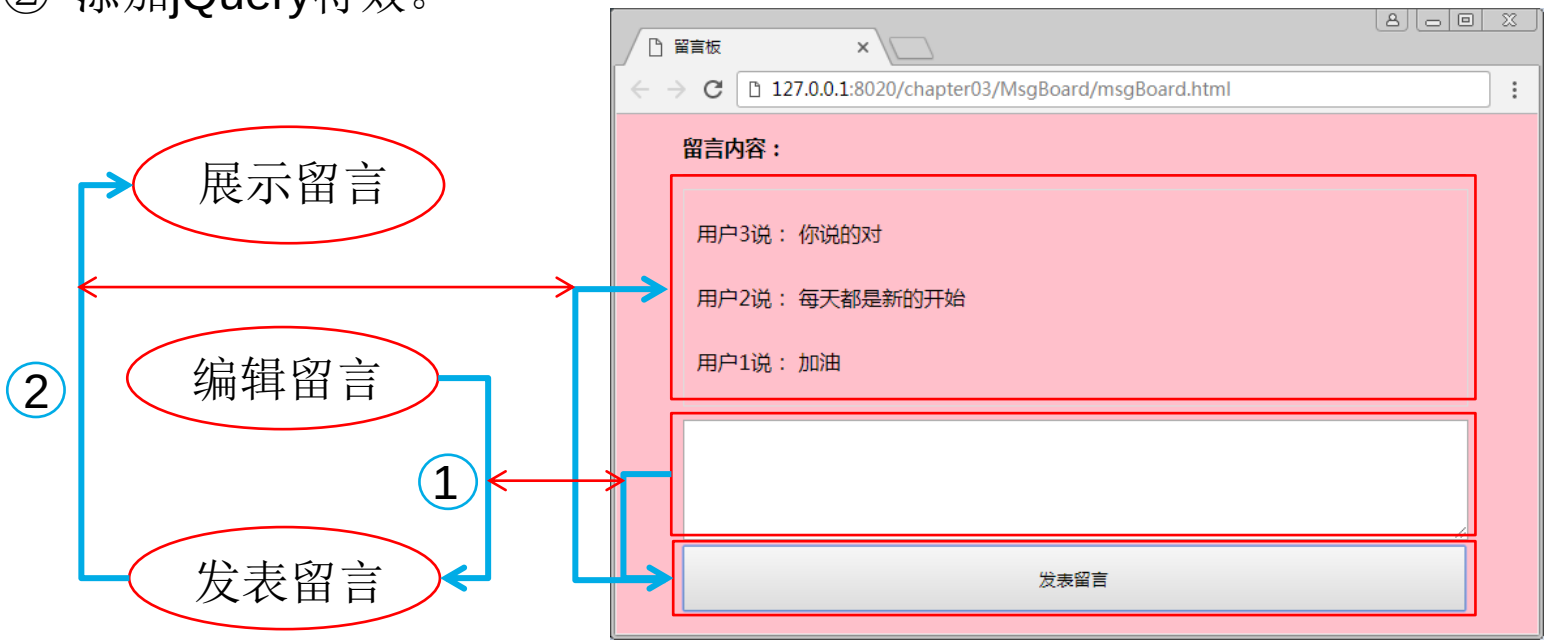


3. 案例留言板——案例实现

代码实现思路：

① 设计HTML结构与样式；

② 添加jQuery特效。



3.4 操作DOM节点

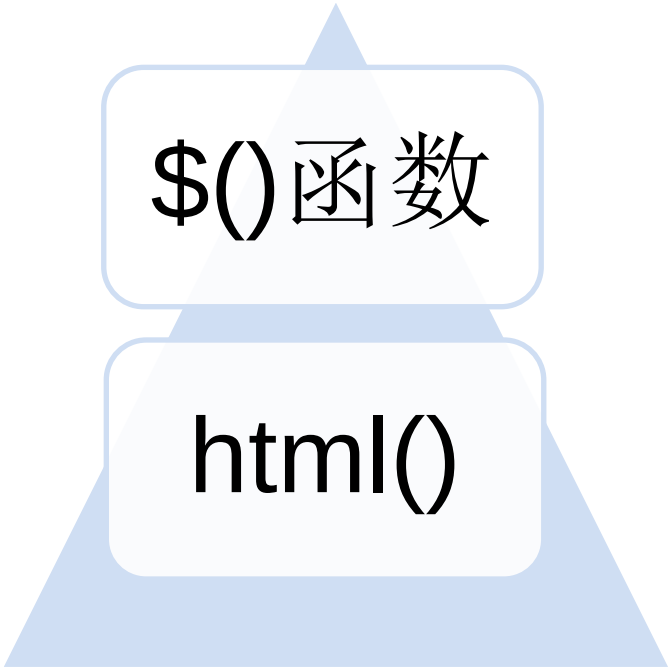
如果需要更灵活的操作网页上的动态效果很多时候需要直接操作DOM节点。

比如：_如在网页中的某个位置动态添加a链接。

1. 创建节点

jQuery提供了动态创建节点的方法，创建节点后返回jQuery对象。

创建节点的方式：



1. 创建节点——$()函数

$()函数在jQuery中有很多作用。

语法：$('HTML代码')；

举例：

- 将DOM对象转换成为jQuery对象；
- 将HTML代码转换成DOM对象，并将其包装成jQuery对象；

1. 创建节点——$()函数

例如：将obj对象传入append()方法中，便可以在DOM中插入节点。

示例

```
var p = $('<p>这是一个段落</p>');
$('body').append(p);
```

1. 创建节点——html()方法

html()方法可用来设置或返回所选元素的HTML内容。

html('HTML代码')：可以在DOM中动态创建节点。

其作用：与JavaScript中的 innerHTML属性类似。

语法：$(selector).html('HTML代码')。

示例

```
$('body').html('<p>这是一个段落</p>')
```

1. 创建节点——html()方法

案例演示：jQuery中创建节点的两种方式$()函数和html()方法。

默认页面

```
<input id="btn" type="button" value="创建节点">
<div id="dv"></div>
 div {
    width: 200px;height: 100px;
    background-color: yellow;
 }
```

1. 创建节点——html()方法

案例演示：当单击按钮时，会在div中创建两个超链接。

创建节点的两种方式

```
$('#btn').click(function() {
    $('#dv').html('<a href="#">腾讯</a><br>');
    var aObj = $('<a href="#">百度</a>');
    $('#dv').append(aObj);
});
```

2. 插入节点

jQuery提供了一些方法用于将创建好的节点插入到DOM的不同位置。

例如：前面用过的append()方法。

方法	描述
append(ele)	用于在匹配元素的最后一个子元素后插入ele，插入ele作为匹配元素的最后一个子元素
prepend(ele)	用于在匹配元素的第一个子元素前面插入ele，插入的ele作为匹配元素的第一个子元素
appendTo(ele)	将匹配元素插入到ele中，该匹配元素作为ele的最后一个子元素

2. 插入节点

方法	描述
prependTo(ele)	将匹配元素插入到ele中，该匹配元素作为ele的第一个子元素
before(ele)	在匹配元素前面插入ele元素
insertBefore(ele)	将匹配元素插入到ele之前
after(ele)	在匹配元素后面插入ele元素
insertAfter(ele)	将匹配元素插入到ele之后

2. 插入节点

案例演示：利用jQuery提供的不同插入节点的方法在ul元素后面插入一个p元素。

HTML

```
<nav>
    <ul>
        <li>序列号1</li>
        <li>序列号2</li>
        <li>序列号3</li>
    </ul>
</nav>
```

2. 插入节点——append()和appendTo()方法

案例演示：append()和appendTo()方法区别这两个方法的调用对象不同。

```
var $p = '<p>插入的节点</p>';
// 将新创建的p节点插入到nav容器的内容底部
$('nav').append($p);
```

VS

```
var $p = '<p>插入的节点</p>';
// 将新创建的p节点插入到nav容器的内容底部
$($p).appendTo('nav');
```

HTML

```
<nav>
  <ul>
      <li>序列号1</li>
      <li>序列号2</li>
      <li>序列号3</li>
  </ul>
  <p>插入的节点</p>
</nav>
```

2. 插入节点——after()和insertAfter()方法

案例演示：after()和insertAfter()方法区别在于这两个方法的调用对象不同。

```
var $p = '<p>插入的节点</p>';
// 将新创建的p节点插入到ul元素之后
$('ul').after($p);
```

VS

```
var $p = '<p>插入的节点</p>';
// 将新创建的p节点插入到ul元素之后
$($p).insertAfter('ul');
```

2. 插入节点——after()和insertAfter()方法

类似：

- prepend()和prependTo()：在某元素中插入第一个子元素。
- before()和insertBefore()：在某元素前面插入元素。

2. 插入节点——after()和insertAfter()方法

案例演示：使用before()方法ul元素的前面插入p元素。

HTML

```
<nav>
    <p>插入的节点</p>
    <ul>
        <li>序列号1</li>
        <li>序列号2</li>
        <li>序列号3</li>
    </ul>
</nav>
```

before($p)

```
var $p = '<p>插入的节点</p>';
// 将新创建的p节点插入到ul元素之前
$('ul').before($p);
```

2. 插入节点——after()和insertAfter()方法

案例演示：在ul元素中的前面插入多个节点时，使用逗号“,”分隔。

HTML

```
<nav>
    <p>插入的p元素</p>
    <a>插入的a元素</a>
    <ul>
        <li>序列号1</li>
        <li>序列号2</li>
        <li>序列号3</li>
    </ul>
</nav>
```

before($p,$a)

```
var $p = '<p>插入的p元素</p>';
var $a = '<a>插入的a元素</a>';
// 将新创建的两个节点插入到ul元素之后
$('ul').before($p, $a);
```

3. 删除节点

在网页开发中，有时需要动态的删除某个节点。

方法	描述
remove()	从DOM中删除所有匹配的元素
detach()	从DOM中删除所有匹配的元素
empty()	删除匹配的元素集合中所有的子节点

3. 删除节点——remove()方法

remove()方法:待删除元素对象调用remove()方法即可完成删除操作。

语法：$(‘p’).remove()；

语法分析：

- $('p')用于获取待删除的元素对象。
- 只会从DOM中移除匹配到的元素，但该元素还存在于jQuery对象中。

3. 删除节点——remove()方法

注意

jQuery对象中不会保留元素的jQuery 数据。例如，被删除的p元素如果绑定了事件或有附加的数据等都会被移除。

3. 删除节点——detach()方法

detach()方法：detach()方法的使用方式与remove()基本相同。

区别：

- detach()方法不仅会保留jQuery对象中的匹配元素；
- 还会保留该元素所有绑定的事件以及附加的数据。

恢复删除节点

```
var obj = $(selector).detach();          // 删除元素节点
$(obj).appendTo(selector);               // 恢复元素节点
```

3. 删除节点——detach()方法

案例演示：detach()方法删除和恢复元素的效果。

默认页面

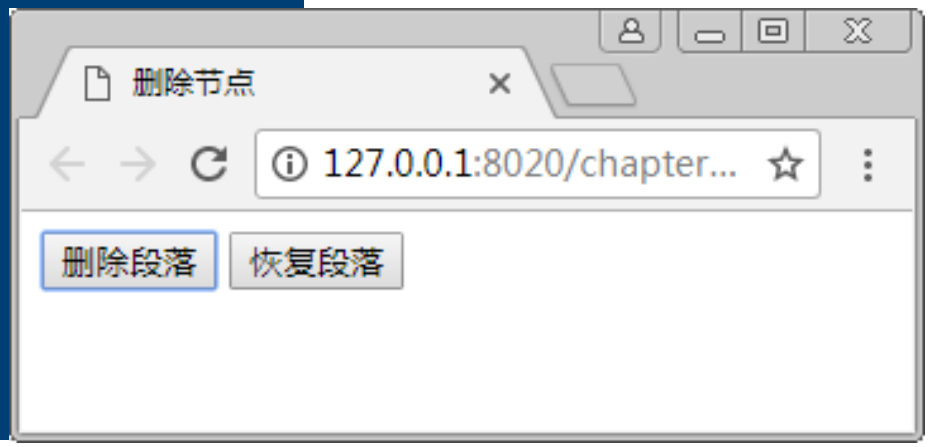
```
<button id="btn1">删除段落</button>
<button id="btn2">恢复段落</button>
<p>今天天气不错!</p>
```

3. 删除节点——detach()方法

案例演示：当单击“删除段落”按钮时，删除p元素，单击“恢复段落”按钮，插入p元素。

删除(恢复)

```
var $p;
$('#btn1').click(function() {
    $p = $('p').detach();
});
$('#btn2').click(function() {
    $($p).appendTo('body');
});
```



3. 删除节点——empty()方法

empty()方法：清空元素中的所有后代节点。

与remove()和detach()方法不同：empty()方法并不是删除节点。

清空div元素中的所有内容

```
$('div').empty();
```

3. 删除节点——empty()方法

案例演示：

HTML默认结构

```
<ul id="ul1">
    <li><span>序列号1</span></li>
    <li><span>序列号2</span></li>
    <li><span>序列号3</span></li>
</ul>
<ul id="ul2">
    <li><span>序列号1</span></li>
    <li><span>序列号2</span></li>
    <li><span>序列号3</span></li>
 </ul>
```

3. 删除节点——empty()方法

案例演示：id为ul1的第2个列表项被完全移除，而id为ul2的第2个列表项只有子元素span被移除。

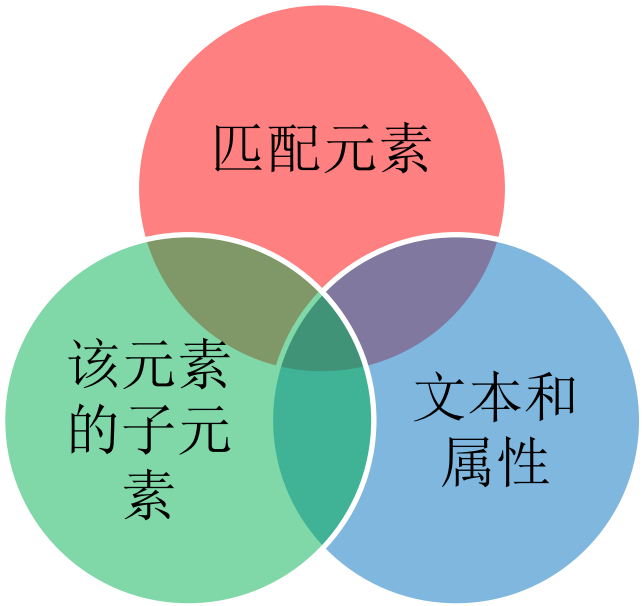
对比

```
<script>
    // 删除序列号2列表项
    $('#ul1 li:eq(1)').remove();
    // 删除序列号2列表项的内容
    $('#ul2 li:eq(1)').empty();
</script>
```

4. 复制节点

jQuery提供一个clone()方法，专门用于处理DOM节点的复制。

复制的内容包括：



4. 复制节点——clone()方法

语法：$(selector).clone()；

语法分析：

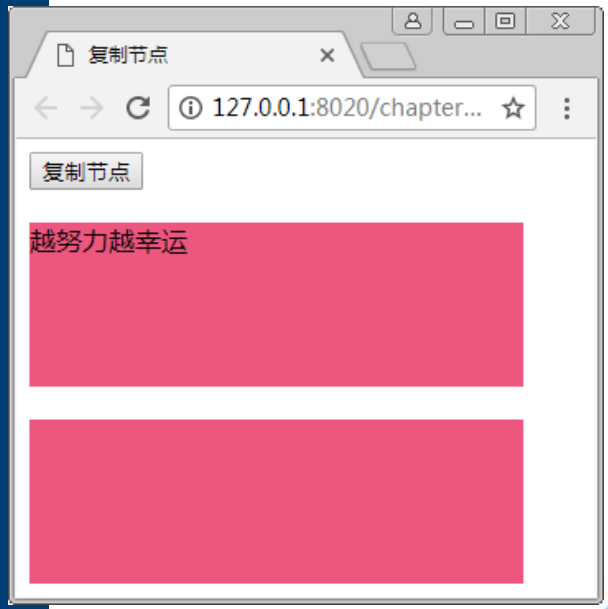
- 参数selector可以是选择器或HTML内容；
- 调用clone()方法后会生成一个被选元素的副本；
- 该副本需要利用插入节点的方法才能显示到DOM中；

4. 复制节点——clone()方法

案例演示：

HTML默认结构

```
div {
    width: 300px;height: 100px;
    background-color: #ed577f;
    margin-top: 20px;
}
<input type="button" value="复制节点" id="btn">
<div id="dv">
    <span>越努力越幸运</span>
</div>
<div id="dv1"></div>
```

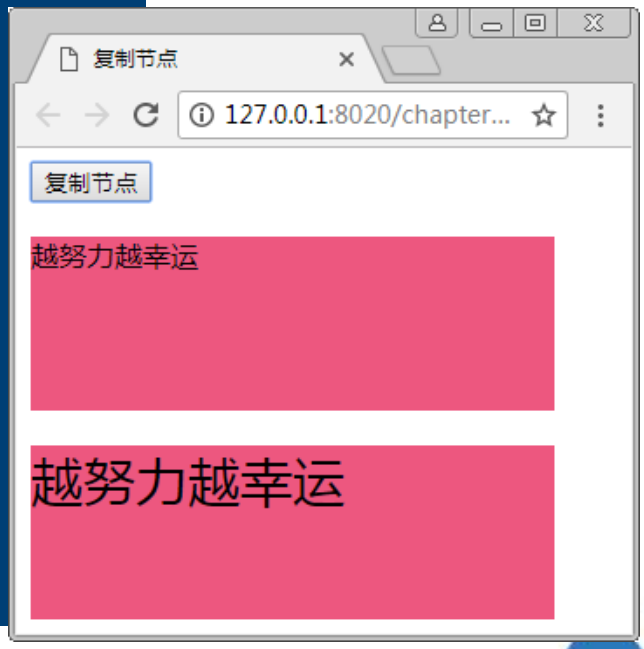


4. 复制节点——clone()方法

案例演示：调用clone()方法生成id为dv下的span元素副本及修改样式。

clone()方法

```
<script>
    $('#btn').click(function() {
        // 复制第一个div中的span元素生成副本
        var spanObj = $('#dv>span').clone();
        // 设置副本的样式
        spanObj.css('fontSize', '30px');
        $('#dv1').append(spanObj);
    });
</script>
```



5. 替换节点

replaceWith()和replaceAll()方法：替换元素或元素中的内容

其区别：在于调用方法的对象以及参数设置的不同。

语法分析：

- content可以是DOM元素对象或HTML内容。

语法

```
$(selector).replaceWith(content);
$(content).replaceAll(selector);
```

5. 替换节点

例如：将p元素替换为span元素。

分析：

- replaceWith()方法的调用对象是待替换的元素对象。
- replaceAll()方法的使用正好与之相反。

语法对比

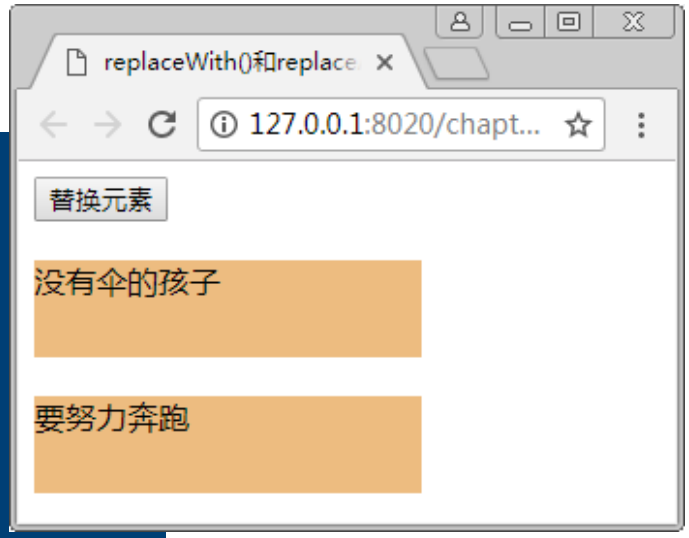
```
$('p').replaceWith('<span>替换喽</span>');      // 实现方式一
$('<span>替换喽</span>').replaceAll('p');       // 实现方式二
```

5. 替换节点

案例演示：

HTML默认结构

```
div {
        width: 200px;
        height: 50px;
        background-color: #edbc80;
        margin-top: 20px;
    }
<input type="button" id="btn" value="替换元素">
<div id="dv">没有伞的孩子</div>
<div id="dv1">要努力奔跑</div>
```

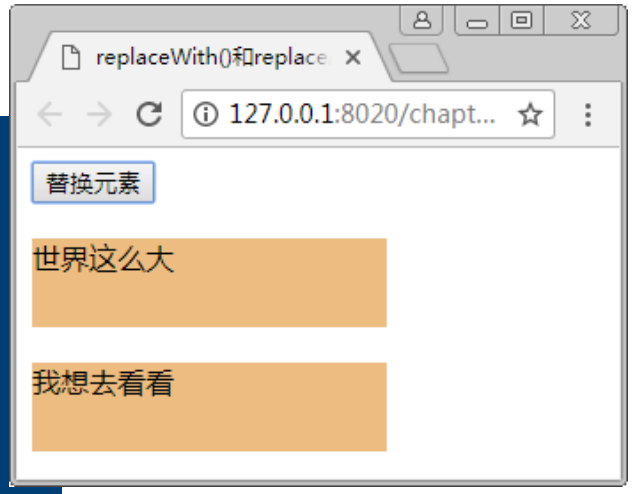


5. 替换节点

案例演示：使用replaceWith()方法替换id值为dv的div元素，使用replaceAll()方法替换id值为dv1的div元素。

替换

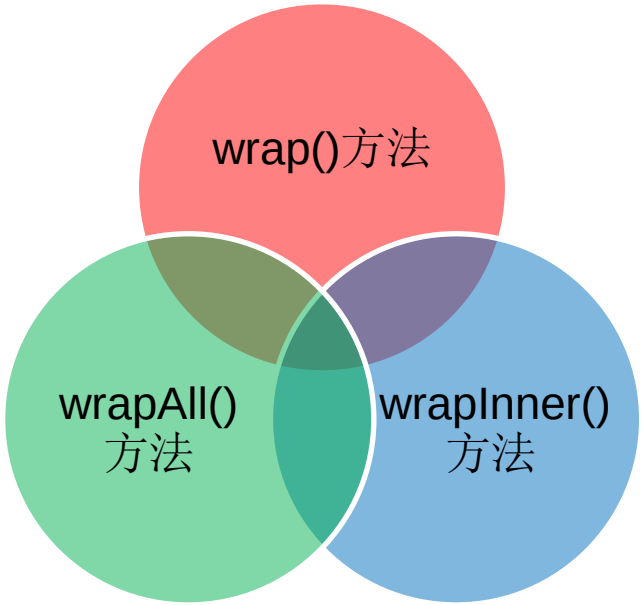
```
<script>
  $('#btn').click(function() {
    $('#dv').replaceWith('<div>世界这么大</div>');
    $('<div>我想去看看</div>').replaceAll('#dv1');
  });
</script>
```



6. 包裹节点

包裹节点是指在某个元素的外层添加父元素，将其“包裹”起来。

包裹节点方法：



6. 包裹节点——wrap()方法

wrap()方法用于为每个匹配到的元素添加父元素，将匹配元素包裹在其中。

语法：$(selector).wrap(wrapper)；

语法分析：参数wrapper表示包裹元素的结构化标记。

示例

```
$('li').wrap('<strong></strong>');
<ul>
  <strong><li>北京</li></strong>
  <strong><li>广州</li></strong>
  <strong><li>深圳</li></strong>
</ul>
```

6. 包裹节点——wrapAll()方法

wrapAll()方法： 将所有匹配元素一起包裹起来。

语法：$(selector).wrapAll(wrapper)；

语法分析：参数wrapper表示包裹元素的结构化标记。

示例

```
$('li').wrapAll('<strong></strong>');
<strong>
   <li>北京</li>
   <li>广州</li>
   <li>深圳</li>
 </strong>
```

6. 包裹节点——wrapInner()方法

wrapInner()方法：为匹配元素添加子元素并且包裹匹配元素中的所有内容。

语法：$(selector).wrapInner(wrapper)；

语法分析：参数wrapper表示包裹元素的结构化标记。

示例

```
$('li').wrapInner('<strong></strong>');
<ul>
  <li><strong>北京</strong></li>
  <li><strong>广州</strong></li>
  <li><strong>深圳</strong></li>
</ul>
```

7. 遍历节点

在DOM元素操作中，为ul元素下的li元素内容都为添加内容。

代码实现：$('ul li').text('测试');

代码分析：

- 我们并没有去取得所有li元素然后循环添加；
- 这种实现方式被称为“隐式迭代”。它是通过jQuery内部机制实现的；
- 一般适用于对指定的元素做相同操作的处理；

7. 遍历节点——each()方法

在实际开发中，有些需求是“隐式迭代”所不能处理的。

例如：为ul元素下的奇数行li元素添加内容。

解决方案：通过jQuery提供的each()方法遍历所有元素，并进行相关的处理。

语法：$(selector).each(function(index,element){})；

语法分析：

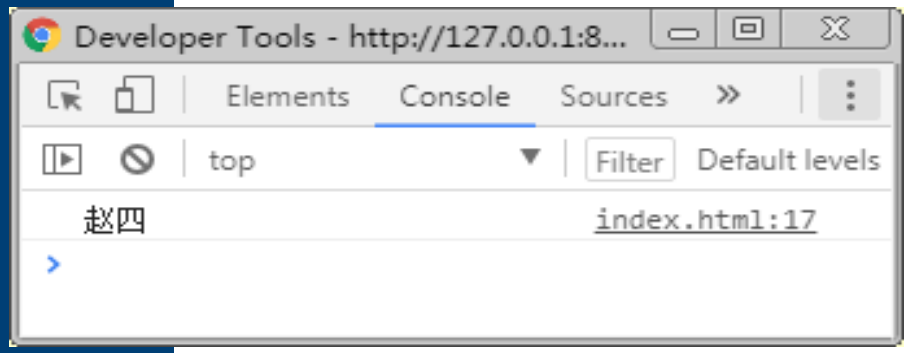
- index表示遍历索引，索引默认从0开始；
- element表示当前元素，一般使用this关键字表示当前元素；

7. 遍历节点——each()方法

例如：获取第2个li元素中的文本内容。

示例

```
<li>刘三</li>
<li>赵四</li>
<li>王小五</li>
$('li').each(function(index) {
    if (index == 1) {
        console.log($(this).text());
    }
});
```

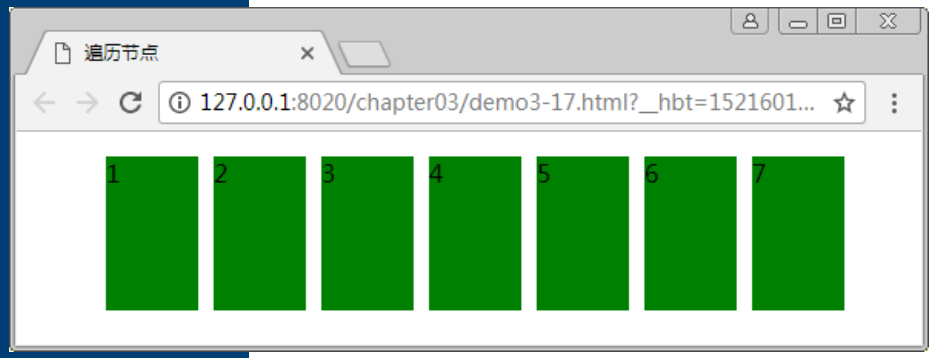


7. 遍历节点——each()方法

案例演示：

```
ul li {
    width: 60px;height: 100px;
    background-color: green;
    list-style-type: none;float: left;
    margin-left: 10px;
}
<ul id="uu">
    <li>1</li><li>2</li><li>3</li>
    <li>4</li><li>5</li><li>6</li><li>7</li>
</ul>
```

默认页面

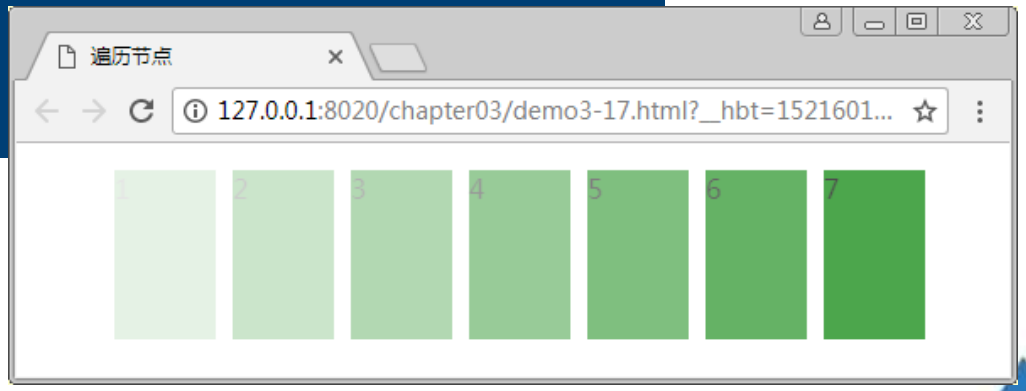


7. 遍历节点——each()方法

案例演示：让ul列表中的每个li元素的背景色由浅入深。

遍历节点

```
<script>
    $('#uu>li').each(function(index, element) {
        // 改变每个元素的透明度
        $(element).css('opacity', (index + 1) / 10);
    });
</script>
```



8. 案例权限选择

内容管理系统是：

- 用于管理内容创办和办公流程的软件系统；
- 使用内容管理系统可以提交、删除、修改、审批、发布内容等；
- 使用者可以包括管理员、创作人员、编辑人员、发布人员等；
- 管理员作为拥有最高权限的角色，需要为每个角色用户设置权限；

8. 案例权限选择——案例展示

案例演示：

- 包含2个下拉列表和4个按钮，左侧下拉列表为所有权限；
- 当某个权限被移动到右侧下拉列表，则代表当前用户添加了该权限；

8. 案例权限选择——案例分析

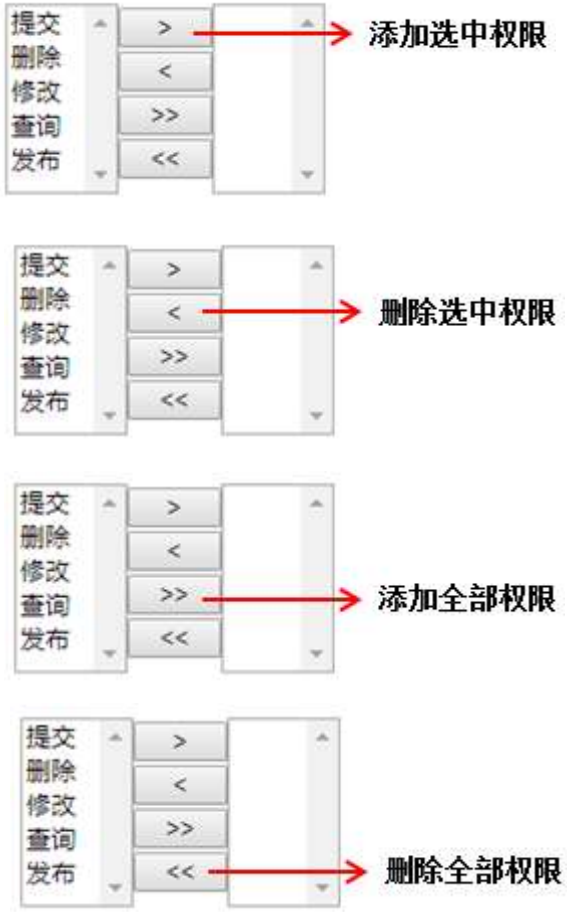
案例演示：

① 添加选中权限；

② 删除选中权限；

③ 添加全部权限；

④ 删除全部权限；



8. 案例权限选择——案例实现

案例演示：在左侧下拉列表中选中“提交”和“修改”两个权限，然后单击“＞”按钮。

多 学 一 招 jQuery链式编程

利用jQuery获取DOM元素以后，可以对该元素对象进行一系列操作，并且所有操作可以通过点号“.”的形式连接在一起形成一句代码，这种类似“链条”的调用方式称之为链式编程。

```
$('#pic').css('border', 'solid 1px #FF0000');
$('#pic').attr('alt', 'myPhoto');
```

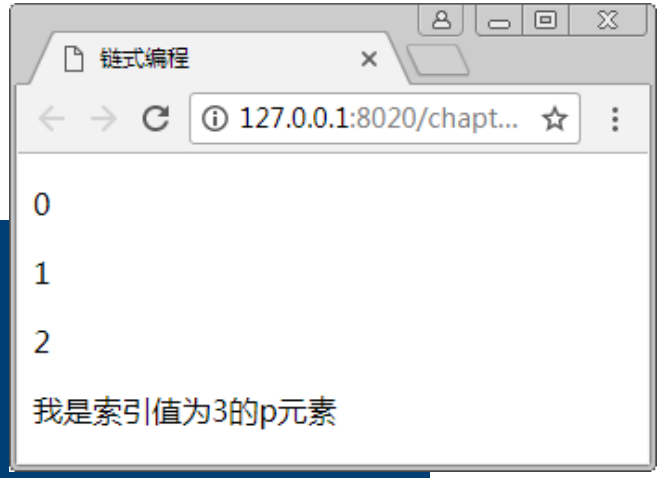
VS

```
$('#pic').css('border', 'solid 1px #FF00030').attr('alt', 'myPhoto');
```

“$('#pic').css('border', 'solid 1px#FF00030')”代码执行后，会返回当前元素的jQuery对象，这个jQuery对象与$('#pic')返回的结果相同，所以可以直接使用“.”调用attr()方法。

多 学 一 招 jQuery链式编程

案例演示：修改div元素下的第4个p元素的内容。




```
<p>0</p>
<p>1</p>
<p>2</p>
<p>3</p>
<script>
    $('div').find('p').eq(3).html('我是索引值为3的p元素');
</script>
```

链式编程

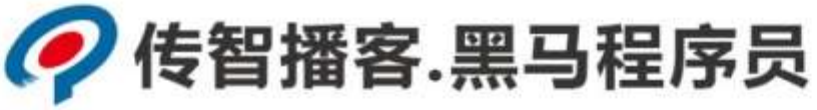
多 学 一 招 jQuery链式编程

值得一提

jQuery之所以可以实现链式编程，就是因为每个函数或方法被调用后返回的都是jQuery对象，在jQuery对象上可以继续调用jQuery方法执行其他操作。但是链式编程的语句不宜过长，否则会造成代码难以阅读的问题。

本章小结

本章首先介绍了jQuery操作DOM相关方法，包括操作元素样式、操作元素属性、操作元素内容和操作DOM节点等，然后介绍了jQuery中链式编程的应用。



Thank You!