

JQuery

1. JQuery概述

1.1 JavaScript库

即library，是一个封装好的特定的集合（方法和函数），这些经过原生JS封装好的就叫做JS库，这是为了更加高效的使用这些封装好的功能

常见的JS库: JQuery, Prototype, YUI, Dojo, Ext JS等等

1.2 JQuery

J就是JavaScript，Query就是查询，把JS中的DOM操作做了封装，我们可以快速查询里面的功能

优化了DOM操作、事件处理、动画设计、Ajax交互

学习JQuery的本质就是学习调用这些函数（方法）

1.3 JQuery下载

下载网址: <https://jquery.com/download/>

production是工程的Jquery，这个是经过压缩的文件

development是开发的jQuery，里面还保留了相关的注释

jQuery

For help when upgrading jQuery, please see the [upgrade guide](#) most relevant to your version. We also recommend using the [jQuery Migrate plugin](#).

[Download the compressed, production jQuery 3.6.0](#)

[Download the uncompressed, development jQuery 3.6.0](#)

Download the map file for jQuery 3.6.0

You can also use the slim build, which excludes the [ajax](#) and [effects](#) modules:

[Download the compressed, production jQuery 3.6.0 slim build](#)

[Download the uncompressed, development jQuery 3.6.0 slim build](#)

[Download the map file for the jQuery 3.6.0 slim build](#)

[jQuery 3.6.0 blog post with release notes](#)

2. JQuery基本语法

2.1 入口函数

也就是原生JS的load的加载事件

下面是第一种写法

`$('#div').hide()`是隐藏元素

```
<!DOCTYPE html>
<html lang="zh">
  <head>
```

```

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title></title>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<style type="text/css">
    .main{
        width: 200px;
        height: 200px;
        background-color: pink;
    }
</style>
<script type="text/javascript">
    $(document).ready(function(){
        $('div').hide();
    })
</script>
</head>
<body>
    <div class="main"></div>
</body>
</html>

```

下面是第二种写法

```

<!DOCTYPE html>
<html lang="zh">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title></title>
        <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
        <style type="text/css">
            .main{
                width: 200px;
                height: 200px;
                background-color: pink;
            }
        </style>
        <script type="text/javascript">
            $(function(){
                $('div').hide();
            })
        </script>
    </head>
    <body>
        <div class="main"></div>

    </body>
</html>

```

1. 等着DOM结构渲染完毕既可执行内部代码，不必等到所有外部资源加载完成，jQuery帮我们完成了封装
2. 相当于原生JS里面的DOMContentLoaded
3. 不同原生JS中的load事件是等页面结构、外部JS、css、图片加载完毕才执行

2.2 顶级对象\$

其实\$符号和jQuery是一样的，但是为了方便，通常直接使用\$

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <script src="js/Jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <style type="text/css">
      .main{
        width: 200px;
        height: 200px;
        background-color: pink;
      }
    </style>
    <script type="text/javascript">
      jQuery(function(){
        jQuery('div').hide();

        // $(function){
        //   $('div').hide();
        // })
      </script>
    </head>
    <body>
      <div class="main"></div>

    </body>
  </html>
```

\$相当于原生JS中的window

2.3 JQuery对象和DOM对象

Jquery对象和DOM对象获取的方式是不一样的

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```

<title></title>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<style type="text/css">
    .main {
        width: 200px;
        height: 200px;
        background-color: pink;
    }
</style>
<script type="text/javascript">
    $(function() {
        //原生JS获取的方式
        var divs = document.querySelector('div');
        //jQuery获取的方式
        $('div');
    })
</script>
</head>
<body>
    <div class="main"></div>
</body>
</html>

```

并且获取来的信息也是不一样的，很明显原生JS获取来的信息更加多

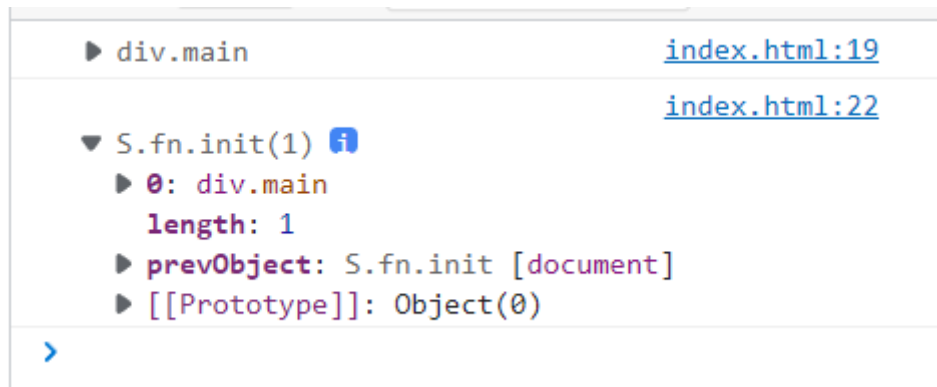
```

<!DOCTYPE html>
<html lang="zh">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title></title>
        <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
        <style type="text/css">
            .main {
                width: 200px;
                height: 200px;
                background-color: pink;
            }
        </style>
        <script type="text/javascript">
            $(function() {
                var divs = document.querySelector('div');
                console.dir(divs);

                $('div');
                console.dir($('div'))
            })
        </script>
    </head>
    <body>
        <div class="main"></div>
    </body>
</html>

```

```
</body>
</html>
```



其实jQuery对象的本质是利用\$对DOM对象包装后产生的对象，只不过是采用伪数组来存储的

要记住的是原生JS获取的DOM对象，只能使用原生JS的语法，不能使用jQuery的语法，jQuery对象只能使用jQuery的方法

DOM对象和jQuery对象之间是可以相互转换的，因为原生JS比jQuery更加大，原生的一些属性和方法jQuery没给我们封装进去，要想使用原生的JS，就需要将jQuery对象转换为DOM对象才能使用

DOM对象转换为jQuery对象

注意转换的格式问题，将DOM对象转换为jQuery对象是不用加引号的

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <style type="text/css">
      .main {
        width: 200px;
        height: 200px;
        background-color: pink;
      }
    </style>
    <script type="text/javascript">
      $(function() {
        //直接获取jQuery对象
        $('.main');
        console.dir($('.main'));

        //将DOM对象转换为jQuery对象
        var JQuery_main = document.querySelector('.main');
        $(JQuery_main);
        console.dir$(JQuery_main));
      })
    </script>
  </head>
```

```
<body>
  <div class="main"></div>
</body>
</html>
```

jQuery对象转换为DOM对象

`$('#获取的值')[index]`

`$('#获取的值').get(index)`

这两种方式都是可以转换的

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <script src="js/Jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <style type="text/css">
      .main {
        width: 200px;
        height: 200px;
        background-color: pink;
      }
    </style>
    <script type="text/javascript">
      $(function() {
        $('#.main')[0].style.backgroundColor = 'red';
        // $('#.main').get(0).style.backgroundColor = 'red';
      })
    </script>
  </head>
  <body>
    <div class="main"></div>
  </body>
</html>
```

3. jQuery的常见API

3.1 jQuery选择器

3.1.1 jQuery各种选择器

原生JS获取元素的方式很多很杂，而且兼容性不同，因此jQuery给我们做了封装，使得获取元素统一标准

```
$('#选择器')
```

名称	用法	描述
ID选择器	<code>\$("#id")</code>	获取指定ID的元素
全选选择器	<code>\$("*")</code>	匹配所有元素
类选择器	<code>\$(".class")</code>	获取同一类class的元素
标签选择器	<code>\$("div")</code>	获取同一类标签的所有元素
并集选择器	<code>\$("div,p,li")</code>	选取多个元素
交集选择器	<code>\$("li.current")</code>	交集元素

子代选择器	<code>\$("ul>li");</code>	使用>号，获取亲儿子层级的元素；注意，并不会获取孙子层级的元素
后代选择器	<code>\$("ul li");</code>	使用空格，代表后代选择器，获取ul下的所有li元素，包括孙子等

下面就是关于jQuery选择器的使用，下面的代码证明后代选择器也是可以的，相应的伪类选择器也是可以生效的

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <style type="text/css">
      .main {
        width: 200px;
        height: 200px;
        background-color: pink;
      }
      .small{
        width: 100px;
        height: 100px;
        background-color: blueviolet;
      }
    </style>
    <script type="text/javascript">
      $(function() {
        $('.main .small')[0].style.backgroundColor = 'red';
        console.dir($('.main .small'));
      })
    </script>
  </head>
  <body>
    <div class="main">
      <span>123</span>
      <div class="small">123</div>
    </div>
  </body>
</html>
```

123

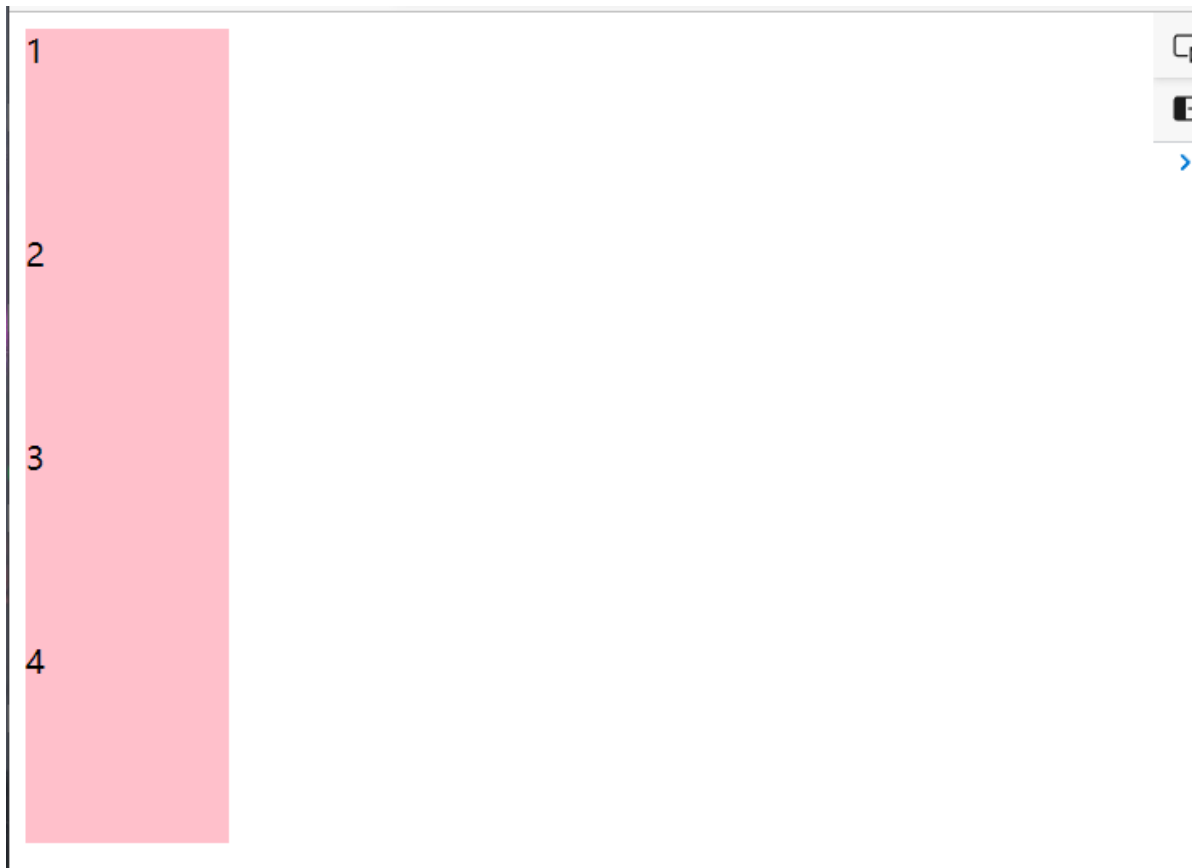
123

3.1.2 隐式迭代

遍历内部DOM元素（伪数组形式存储）的过程就叫做隐式迭代

是将匹配到的所有元素内部进行遍历循环，而不是自己写

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <style type="text/css">
      div{
        width: 100px;
        height: 100px;
      }
    </style>
    <script type="text/javascript">
      $(function() {
        $('div').css('background-color', 'pink');
      })
    </script>
  </head>
  <body>
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
  </body>
</html>
```

我们这边可以查看一下jQuery获取来的元素，其实就是伪数组里面4个div

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <style type="text/css">
      div{
        width: 100px;
        height: 100px;
      }
    </style>
    <script type="text/javascript">
      $(function() {
        $('div').css('background-color', 'pink');
        console.dir($('div'));
      })
    </script>
  </head>
  <body>
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
  </body>
</html>
```

[index.html:18](#)

```
▼ S.fn.init(4) ⓘ  
  ▶ 0: div  
  ▶ 1: div  
  ▶ 2: div  
  ▶ 3: div  
    length: 4  
  ▶ prevObject: S.fn.init [document]  
  ▶ [[Prototype]]: Object(0)
```

>

3.1.3 JQuery筛选选择器

语法	用法	描述
:first	\$('#li:first')	获取第一个li元素
:last	\$('#li:last')	获取最后一个li元素
:eq(index)	\$('#li:eq(2)')	获取到的li元素中，选择索引号为2的元素，索引号index从0开始。
:odd	\$('#li:odd')	获取到的li元素中，选择索引号为奇数的元素
:even	\$('#li:even')	获取到的li元素中，选择索引号为偶数的元素

3.1.4 JQuery筛选方法

下面图有一个错误，prevAll，没有t

语法	用法	说明
parent()	\$('#li').parent();	查找父级
children(selector)	\$('#ul').children('li')	相当于 \$('#ul>li')，最近一级（亲儿子）
find(selector)	\$('#ul').find('li');	相当于 \$('#ul li')，后代选择器
siblings(selector)	\$('#.first').siblings('li');	查找兄弟节点，不包括自己本身
nextAll([expr])	\$('#.first').nextAll()	查找当前元素之后所有的同辈元素
prevAll([expr])	\$('#.last').prevAll()	查找当前元素之前所有的同辈元素
hasClass(class)	\$('#div').hasClass("protected")	检查当前的元素是否含有某个特定的类，如果有，则返回true
eq(index)	\$('#li').eq(2);	相当于 \$('#li:eq(2)'), index 从0开始

其本质基本和DOM操作，CSS选择是一样的

```
<!DOCTYPE html>  
<html lang="zh">  
  <head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```

</title>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<style type="text/css">
    div{
        width: 100px;
        height: 100px;
    }
</style>
<script type="text/javascript">
    $(function() {
        console.dir($('.son').parent());
        console.dir($('.father').children('li'));
        console.dir($('.father').find('li'));
        console.dir($('.son').siblings('li'));
        console.dir($('.son').nextAll('li'));
        console.dir($('.son').prevAll('li'));
        console.dir($('.father li').hasClass('bro1'));
        console.dir($('.father li').eq(2));
    })
</script>
</head>
<body>
    <ul class="father">
        <li class="bro1">1</li>
        <li class="bro2">2</li>
        <li class="son">3</li>
        <li class="bro3">4</li>
        <li class="bro4">
            5
            <ul>
                <li class="bro5"></li>
                <li class="bro6"></li>
                <li class="bro7"></li>
            </ul>
        </li>
    </ul>
</body>
</html>

```

▶ S.fn.init(1)	index.html:17
▶ S.fn.init(5)	index.html:18
▶ S.fn.init(8)	index.html:19
▶ S.fn.init(4)	index.html:20
▶ S.fn.init(2)	index.html:21
▶ S.fn.init(2)	index.html:22
true	index.html:23
▶ S.fn.init(1)	index.html:24

>

其中siblings可以作为排他思想来使用

3.2 链式编程

下面这段代码是案例里面排他思想的一段代码

我们可以发现在写jQuery排他思想的时候，核心代码有两段是一样的，代码是不允许一样的代码出现多次，这里就出现了我们的jQuery的链式编程

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <style type="text/css">
      .nav{
        width: 500px;
        list-style: none;
        display: flex;
        justify-content: space-around;
      }
      .nav li{
        color: pink;
        font-size: 25px;
        cursor: pointer;
      }
    </style>
    <script type="text/javascript">
      $(function(){
        $('.nav').children('li').mouseover(function(){
          $(this).css('color','red');
          $(this).siblings('li').css('color','');
        })
      })
    </script>
  </head>
  <body>
    <ul class="nav">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
    </ul>
  </body>
</html>
```

下面就是链式编程的写法，代替上面的两段代码

```
$(this).css('color','red').siblings('li').css('color','');
```

我们还可以换一个案例来实现链式编程，在案例3.8.3淘宝服饰就可以改为链式编程

```
$('.main_img img').eq(index).show().siblings('img').hide();
```

这样可以节省很多代码量，让编写更加优雅

3.3 JQuery样式操作

3.3.1 CSS操作

查看CSS样式的属性值的话，你只在css()方法里面写属性名就可以了，返回值就是属性值

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">
      .main{
        width: 200px;
        height: 200px;
        background-color: pink;
      }
    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        console.log($('.main').css('width'));
      })
    </script>
  </head>
  <body>
    <div class="main"></div>
  </body>
</html>
```



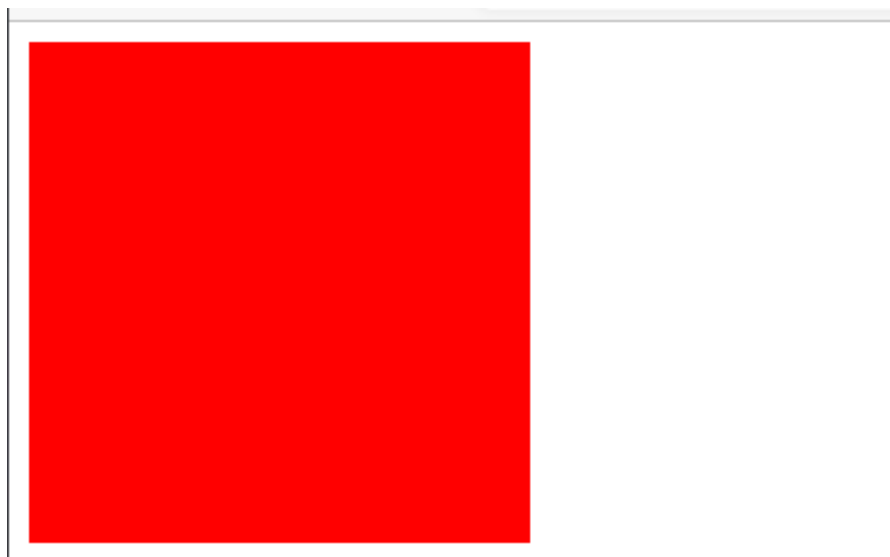
当然我们也可以去修改css的值

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```

<title></title>
<style type="text/css">
    .main{
        width: 200px;
        height: 200px;
        background-color: pink;
    }
</style>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">
    $(function(){
        $('.main').css('background-color','red');
    })
</script>
</head>
<body>
    <div class="main"></div>
</body>
</html>

```



参数也可以是对象的形式，设置多个css样式值，记得里面还有一个括号

```

<!DOCTYPE html>
<html lang="zh">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title></title>
        <style type="text/css">
            .main{
                width: 200px;
                height: 200px;
                background-color: pink;
            }
        </style>
        <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>

```

```
<script type="text/javascript">
    $(function(){
        $('.main').css({
            'background-color':'red',
            'width':'400px'
        });
    })
</script>
</head>
<body>
    <div class="main"></div>
</body>
</html>
```



3.3.2 设置类样式方法

addClass('类名'), 添加类

removeClass('类名'), 删除类

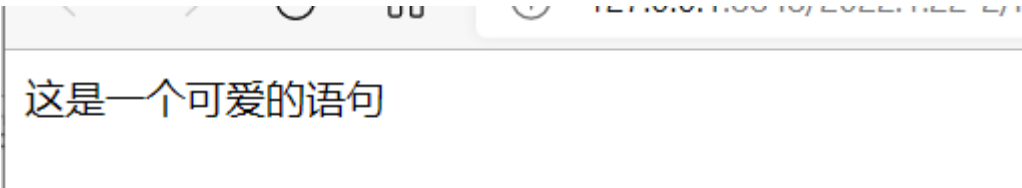
toggleClass('类名'), 切换类, 假如你有的话就删除, 假如你没有的话就添加

```
<!DOCTYPE html>
<html lang="zh">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title></title>
        <style type="text/css">
            .main{
                width: 200px;
                height: 200px;
            }
            .main_col{
                background-color: pink;
            }
            .main_text_col{
                color: cadetblue;
                font-size: 30px;
            }
        </style>
    </head>
    <body>
        <div class="main">
            <div class="main_col">
                <div class="main_text_col">
                    <h1>
                        1
                    </h1>
                </div>
            </div>
        </div>
    </body>
</html>
```

```

</style>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">
    $(function(){
        $('.main').addClass('main_col');
        $('.main').removeClass('main_text_col');
        $('.main').toggleClass('main_col');
    })
</script>
</head>
<body>
    <div class="main main_text_col">这是一个可爱的语句</div>
</body>
</html>

```



这是一个可爱的语句

在原生的JS和jQuery的类名操作有一些区别，原生JS的className会覆盖掉原先里面的类名，但是jQuery里面类的操作只是对指定类进行操作，不会影响原先的类名，和原生JS的classList是基本一样的

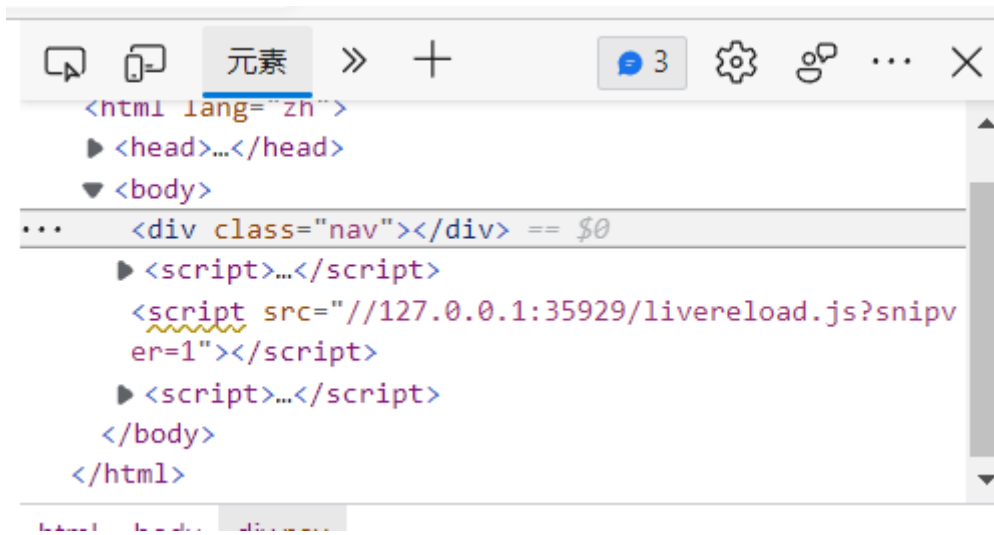
我们先来看原生JS的操作

```

<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">

    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
        window.addEventListener('load',function(){
            var main = document.querySelector('.main');
            main.className = 'nav';
        })
    </script>
  </head>
  <body>
    <div class="main"></div>
  </body>
</html>

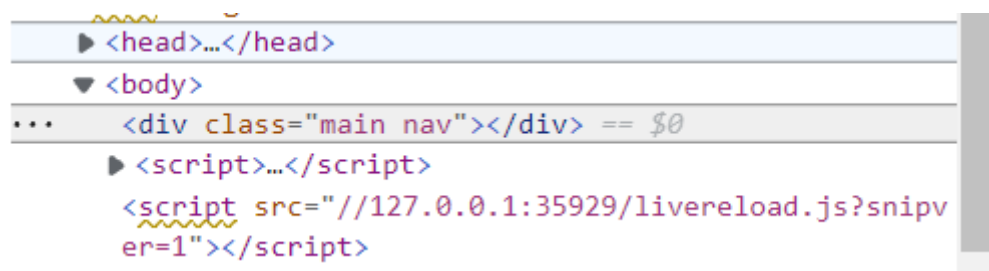
```

但是在我们的jQuery里面是不影响的

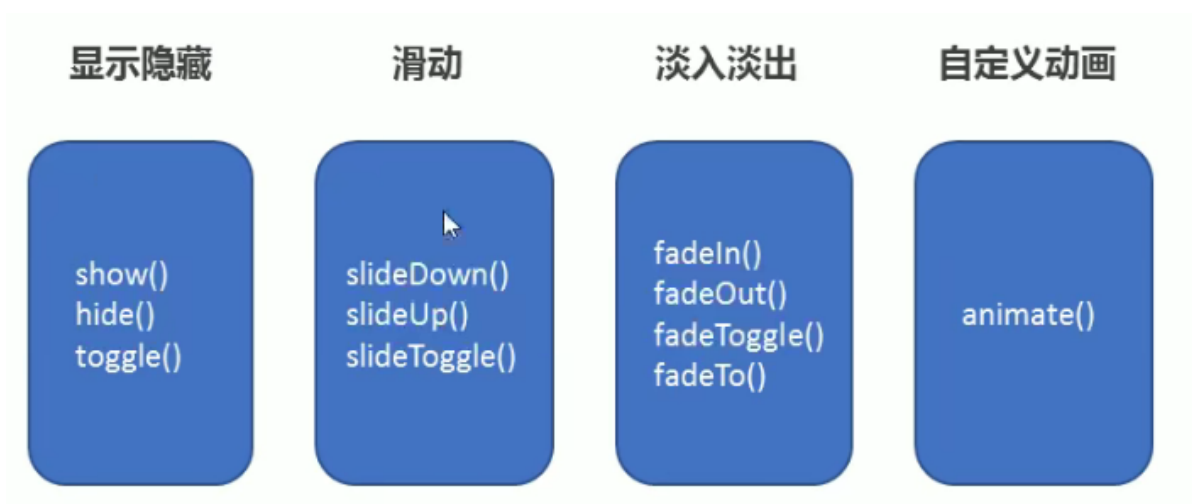
```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">

    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        $('.main').addClass('nav');
      })
    </script>
  </head>
  <body>
    <div class="main"></div>
  </body>
</html>
```



3.4 JQuery效果

JQuery封装了很多的动画效果



3.4.1 显示和隐藏

1. 显示语法规范

```
show([speed, [easing], [fn]])
```

2. 显示参数

- (1) 参数都可以省略，无动画直接显示。
- (2) speed：三种预定速度之一的字符串（“slow”，“normal”，or “fast”）或表示动画时长的毫秒数值(如：1000)。
- (3) easing：(Optional) 用来指定切换效果，默认是 “swing”，可用参数 “linear”。
- (4) fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

当然我们还有隐藏的方法

1. 隐藏语法规范

```
hide([speed, [easing], [fn]])
```

2. 隐藏参数

- (1) 参数都可以省略，无动画直接显示。
- (2) speed：三种预定速度之一的字符串（“slow”，“normal”，or “fast”）或表示动画时长的毫秒数值(如：1000)。
- (3) easing：(Optional) 用来指定切换效果，默认是 “swing”，可用参数 “linear”。
- (4) fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

toggle的语法格式是和上面一样的，但是toggle是只要隐藏了就显示，显示了就隐藏

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">
```

```
        .main {
            width: 200px;
            height: 200px;
            background-color: pink;
        }
    </style>
    <script src="js/Jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
        $(function() {
            $('.xian').click(function() {
                $('.main').show(1000, 'linear', function() {
                    alert('你显示了哦!');
                })
            })

            $('.ying').click(function() {
                $('.main').hide(1000, 'linear', function() {
                    alert('你隐藏了哦!');
                })
            })

            $('.qie').click(function(){
                $('.main').toggle(1000, 'linear', function(){
                    alert('你切换了哦');
                })
            })
        })
    </script>
</head>
<body>
    <button type="button" class="xian">显示</button>
    <button type="button" class="ying">隐藏</button>
    <button type="button" class="qie">切换</button>
    <div class="main"></div>
</body>
</html>
```



3.4.2 滑动

其实和上面的语法格式是一样的，所以这里就不再赘述了

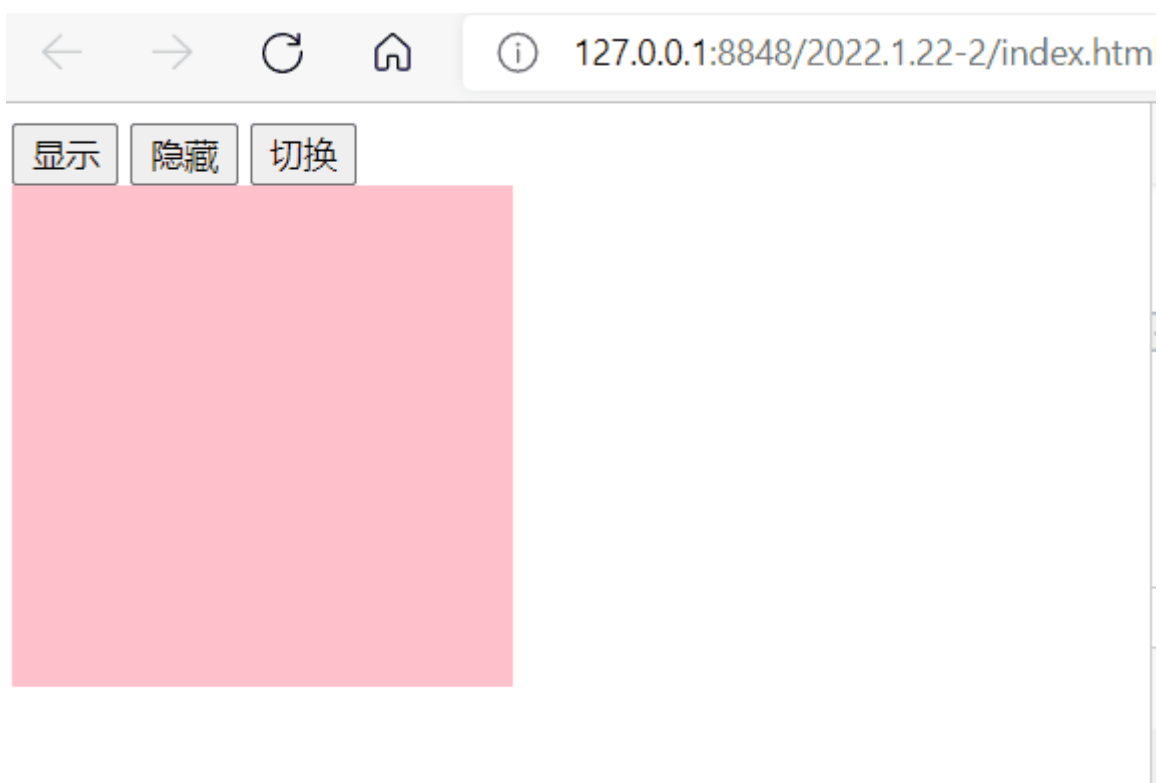
```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">
      .main {
        width: 200px;
        height: 200px;
        background-color: pink;
      }
    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function() {
        $('.xian').click(function() {
          $('.main').slideDown(1000, 'linear', function() {
            alert('你显示了哦!');
          });
        });

        $('.ying').click(function() {
          $('.main').slideUp(1000, 'linear', function() {
            alert('你隐藏了哦!');
          });
        });
      });
    </script>
  </head>
  <body>
    <div class="main">
    </div>
  </body>
</html>
```

```

        $('.qie').click(function(){
            $('.main').slideToggle(1000,'linear',function(){
                alert('你切换了哦');
            })
        })
    })
</script>
</head>
<body>
    <button type="button" class="xian">显示</button>
    <button type="button" class="ying">隐藏</button>
    <button type="button" class="qie">切换</button>
    <div class="main"></div>
</body>
</html>

```



3.4.3 事件切换

我们在案例3.8.1的下拉菜单中就使用了鼠标经过和移开，这里我们介绍另一个，这个也可以进行鼠标的经过和移开，这个更加简单，更加高效

使用hover方法就可以了，houver(1,2)，其中1表示mouseenter，也就是鼠标移入的意思，其中2就是mouseleave，也就是鼠标移出的意思

```
$(function() {
    $(".nav>li").hover(
        function(){
            $(this).children('ul').slideDown(200);
        },function() {
            $(this).children('ul').slideUp(200);
        }
    );
})
```

当然我们也可以更加简单，假如你在hover里面只写一个的话，就是你鼠标移入执行这个函数，鼠标移开也移动这个函数

我们只要将上面的写法改成这样下面的就可以了

```
$(function() {
    $('.nav>li').hover(function(){
        $(this).children('ul').slideToggle(200);
    });
})
```

3.4.5 停止动画排队

假如我们按照上面的写法来写的话，就会有一些小问题，我们将鼠标快速滑动的时候就会有这样的问题

下拉

下拉

下拉

下拉

这个时候我们就需要动画停止函数，也就是stop()，这个使用有一个注意的点，就是要写在动画的前面，不然不能阻止上面的问题，当然我们也可以写在其他的地方

假如我们要修正的话，就要写成下面这个样子

```
$(function() {
    $('.nav>li').hover(function(){
        $(this).children('ul').stop().slideToggle(200);
    });
})
```

3.4.6 淡入淡出

淡入fadeTo，淡出fadeOut，切换fadeToggle，还有透明度切换fadeTo，其中语法格式基本和上面都是一样的，除了fadeTo(speed,opacity,easing,fn)是这样的语法格式

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title></title>
<style type="text/css">
    .main {
        width: 200px;
        height: 200px;
        background-color: pink;
    }
</style>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">
    $(function() {
        $('.ru').click(function() {
            $('.main').fadeIn(1000, 'linear', function() {
                alert('这是淡入哦');
            })
        })

        $('.chu').click(function() {
            $('.main').fadeOut(1000, 'linear', function() {
                alert('这是淡出哦');
            })
        })

        $('.qie').click(function(){
            $('.main').fadeToggle(1000, 'linear', function(){
                alert('这是淡入淡出切换哦');
            })
        })

        $('.tou').click(function(){
            $('.main').fadeTo(1000, 0.5, function(){
                alert('渐进方式调整');
            })
        })
    })
</script>
</head>
<body>
    <button type="button" class="ru">淡入</button>
    <button type="button" class="chu">淡出</button>
    <button type="button" class="qie">淡入淡出切换</button>
    <button type="button" class="tou">修改透明度</button>
    <div class="main"></div>
</body>
</html>
```

淡入 淡出 淡入淡出切换 修改透明度



3.4.7 自定义动画

1. 语法

```
animate(params,[speed],[easing],[fn])
```

2. 参数

- (1) **params**: 想要更改的样式属性, 以对象形式传递, 必须写。属性名可以不用带引号, 如果是复合属性则需要采取驼峰命名法 **borderLeft**。其余参数都可以省略。
- (2) **speed**: 三种预定速度之一的字符串(“slow”, “normal”, or “fast”)或表示动画时长的毫秒数值(如: 1000)。
- (3) **easing**: (Optional) 用来指定切换效果, 默认是 “swing”, 可用参数 “linear”。
- (4) **fn**: 回调函数, 在动画完成时执行的函数, 每个元素执行一次。

其实这个本质和CSS3的动画格式是差不多的, 其中params是按照对象的格式来写的

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">
      .ani{
        width: 200px;
        height: 200px;
        background-color: pink;
        position: absolute;
      }
    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        $('button').click(function(){
          $('.ani').animate({
            left:100,
```



```

        top:100,
        opacity:.5,
        height:300,
        width:300
    },500)
});
    })
</script>
</head>
<body>
    <button type="button">动画效果</button>
    <div class="ani"></div>
</body>
</html>

```



3.5 JQuery属性操作

3.5.1 固有属性值prop()

固有属性值就是元素本身自带的属性，比如说a元素里面href，input里面的type

获取的语法格式是prop('属性名')

```

<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title></title>
<style type="text/css">

</style>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">
    $(function(){
        console.log($('a').prop("href"));
    })
</script>
</head>
<body>
    <a href="www.baidu.com">点击链接</a>
</body>
</html>

```



当然我们还可以修改属性值，其语法格式是prop('属性名','属性值')

```

<!DOCTYPE html>
<html lang="zh">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title></title>
        <style type="text/css">

        </style>
        <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
        <script type="text/javascript">
            $(function(){
                console.log($('a').prop("href", 'www.bilibili.com'));
            })
        </script>
    </head>
    <body>
        <a href="www.baidu.com">点击链接</a>
    </body>
</html>

```

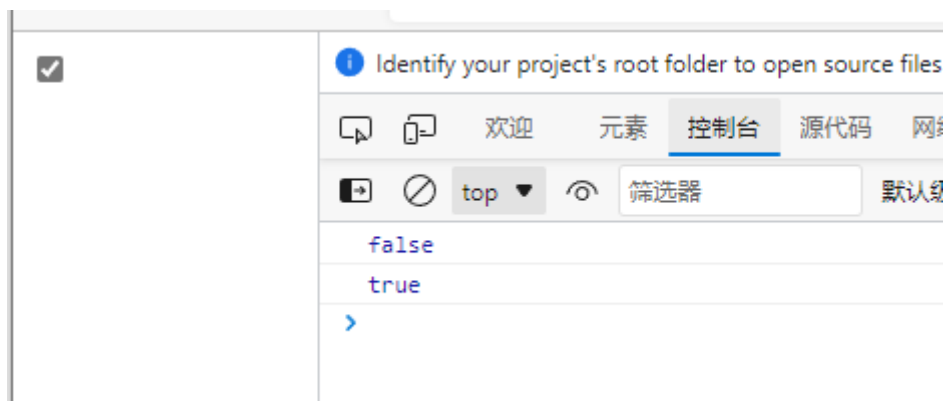
当然我们还可以看到input的checked的属性

change()方法是，只要你改变就会触发里面的函数

当你点击了复选框的话，就会检查checked

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">

    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        $('input').change(function(){
          console.log($(this).prop('checked'));
        })
      })
    </script>
  </head>
  <body>
    <input type="checkbox" name="" id="" value="" checked="checked"/>
  </body>
</html>
```



3.5.2 自定义属性attr()

语法格式基本和上面的prop()是一样的

用户自己给元素添加的属性，我们称为自定义属性。比如给 div 添加 index = "1" 。

1. 获取属性语法

```
attr("属性") // 类似原生 getAttribute()
```

2. 设置属性语法

```
attr("属性", "属性值") // 类似原生 setAttribute()
```

3.5.3 数据缓存

data() 方法可以在指定的元素上存取数据，并不会修改 DOM 元素结构。一旦页面刷新，之前存放的数据都将被移除。

1. 附加数据语法

```
data("name","value") // 向被选元素附加数据
```

2. 获取数据语法

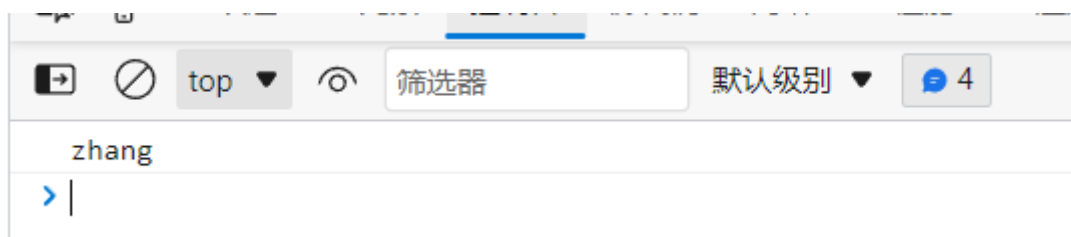
```
date("name") // 向被选元素获取数据
```

同时，还可以读取 HTML5 自定义属性 data-index，得到的是数字型

这个是存在我们的内存里面，只要你刷新页面的话，就会丢失

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">

    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        $('div').data('uname','zhang');
        console.log($('div').data('uname'));
      })
    </script>
  </head>
  <body>
    <div index="1"></div>
  </body>
</html>
```



但是在我们的H5里面还有一个data-index






我们在元素上面写上data-index的话，在jQuery里面用attr查看，和用data查看是不一样的，attr查看的是自定义属性，也就是把data-index当作为自定义属性值了，但是使用data()的话，就是直接获取的H5里面的data-index的值，再各个方法里面填入的属性值也是不一样的

```

<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">

    </style>
    <script src="js/Jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        console.log($('div').attr('data-index'));
        console.log($('div').data('index'));
      })
    </script>
  </head>
  <body>
    <div index="1" data-index="2"></div>
  </body>
</html>

```

		top ▼		筛选器	默认级别 ▼	 4	3 hidden 
2						index.html:14	
2						index.html:15	
>							

3.6 JQuery内容文本值

3.6.1 普通内容

1. 普通元素内容 html() (相当于原生innerHTML)

html() // 获取元素的内容

html("内容") // 设置元素的内容

2. 普通元素文本内容 text() (相当与原生 innerText)

text() // 获取元素的文本内容

text("文本内容") // 设置元素的文本内容

大体是一样的，其中的差别就是Innerhtml和innertext是差不多的

```

<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">

    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        console.log($('.main').html());
        $('.main').html('这是一个被html改变的语句');
        console.log($('.main').html());

        console.log('-----');

        console.log($('.no').text());
        $('.no').text('这是一个被text改变的语句');
        console.log($('.no').text());
      })
    </script>
  </head>
  <body>
    <div class="main">这是一个可爱的语句</div>
    <div class="no">这不是一个可爱的语句</div>
  </body>
</html>

```

这是一个可爱的语句	index.html:14
这是一个被html改变的语句	index.html:16
-----	index.html:18
这不是一个可爱的语句	index.html:20
这是一个被text改变的语句	index.html:22

3.6.2 表单内容

表单的话，可以使用val()方法来表示和修改

但是要注意val()里面的值，要修改的话，必须使用双引号""，不然会导致无法修改

```

<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">

```

```

</style>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">
    $(function(){
        console.log($('input').val());
        $('input').val("我被修改了");
        console.log($('input').val());
    })
</script>
</head>
<body>
    <input type="text" id="" value="这是一个寂寞的输入框" />
</body>
</html>

```



3.7 JQuery遍历

语法1：

```
$( "div" ).each( function (index, domEle) { xxx; } )
```

1. each() 方法遍历匹配的每一个元素。主要用DOM处理。each 每一个
2. 里面的回调函数有2个参数： index 是每个元素的索引号; domEle 是每个DOM元素对象，不是jquery对象
3. 所以要想使用jquery方法，需要给这个dom元素转换为jquery对象 \$(domEle)

我们来看下面的代码

each就很像前面的foreach，就是一个个循环，其中回调函数里面的index是表示的序号，从上到下依次是0,1,2，这个是自动编好的，domEle是表示的dom对象，注意这里是dom对象，不是JQuery的对象，所以在这里要将DOM对象转换为JQuery对象，这个在前面有写

随后就是执行css方法，然后转回来继续循环

```

<!DOCTYPE html>
<html lang="zh">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title></title>
        <style type="text/css">

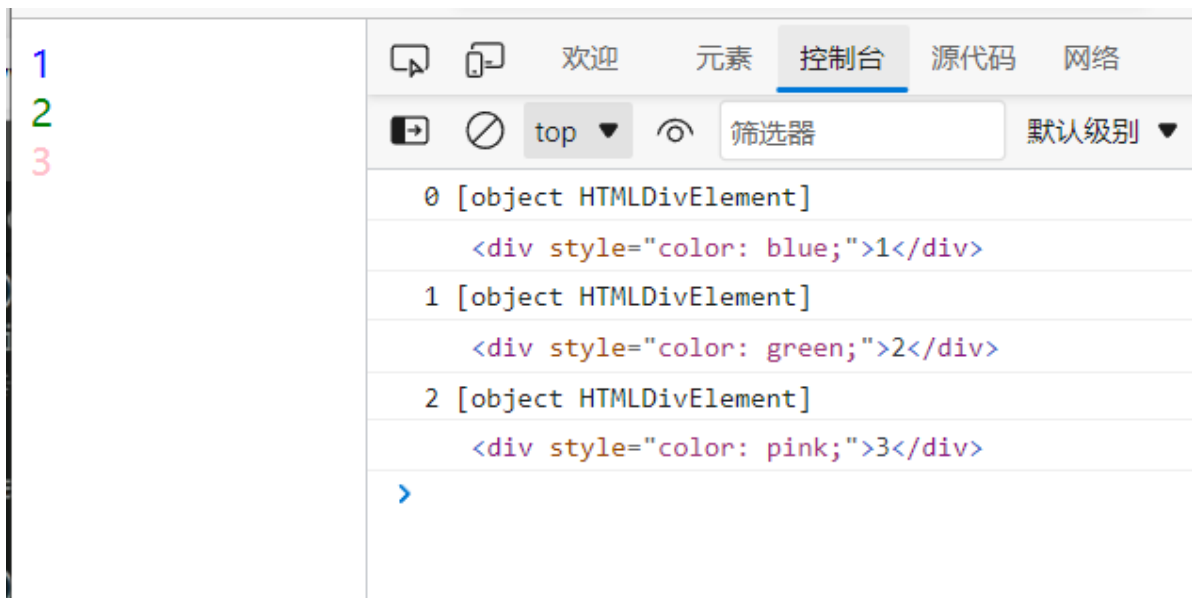
```

```

</style>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">
    $(function(){
        var color = ["blue","green","pink"];
        $('div').each(function(index,domEle){
            console.log(index + ' ' + domEle);
            console.log(domEle);

            $(domEle).css('color',color[index]);
        });
    })
</script>
</head>
<body>
    <div>1</div>
    <div>2</div>
    <div>3</div>
</body>
</html>

```



在jQuery里面还有一个遍历的方式，这个是可以遍历任何对象

语法2：

```
$.each(object, function (index, element) { xxx; })
```

1. \$.each()方法可用于遍历任何对象。主要用于数据处理，比如数组，对象
2. 里面的函数有2个参数：index 是每个元素的索引号; element 遍历内容

而且输出的东西和上面是一样的，但是我们来看下面的遍历的东西，这个是可以遍历任何对象的


```

<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">

    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        var color = ["blue","green","pink"];
        $.each(color,function(index,ele){
          console.log(index);
          console.log(ele);
        });
      })
    </script>
  </head>
  <body>
    <div>1</div>
    <div>2</div>
    <div>3</div>
  </body>
</html>

```

0
blue
1
green
2
pink
>

3.8 JQuery尺寸、位置

3.8.1 JQuery尺寸

假如你要修改的话，就在里面添加数字，就可以修改

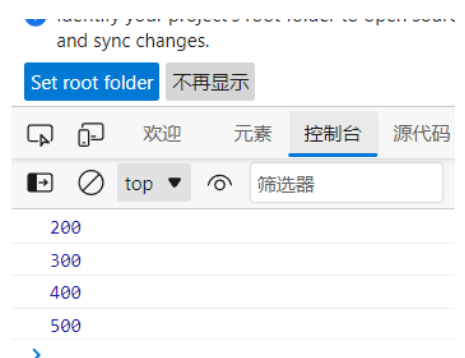
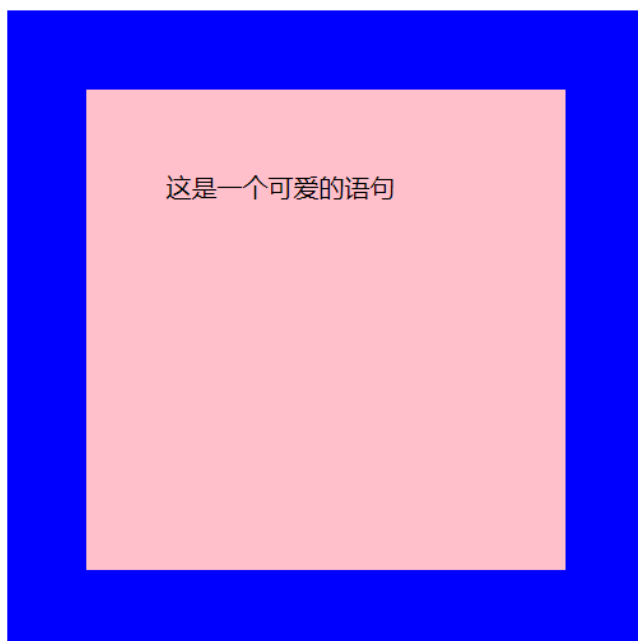
语法	用法
width() / height()	取得匹配元素宽度和高度值 只算 width / height
innerWidth() / innerHeight()	取得匹配元素宽度和高度值 包含 padding
outerWidth() / outerHeight()	取得匹配元素宽度和高度值 包含 padding、border
outerWidth(true) / outerHeight(true)	取得匹配元素宽度和高度值 包含 padding、border、margin

```

<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">
      .main{
        width: 200px;
        height: 200px;
        background-color: pink;

        border: 50px solid blue;
        margin: 50px;
        padding: 50px;
      }
    </style>
    <script src="js/Jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function() {
        console.log($('.main').width());
        console.log($('.main').innerwidth());
        console.log($('.main').outerwidth());
        console.log($('.main').outerwidth(true));
      })
    </script>
  </head>
  <body>
    <div class="main">这是一个可爱的语句</div>
  </body>
</html>

```



3.8.2 JQuery位置

3.8.2.1 offset()

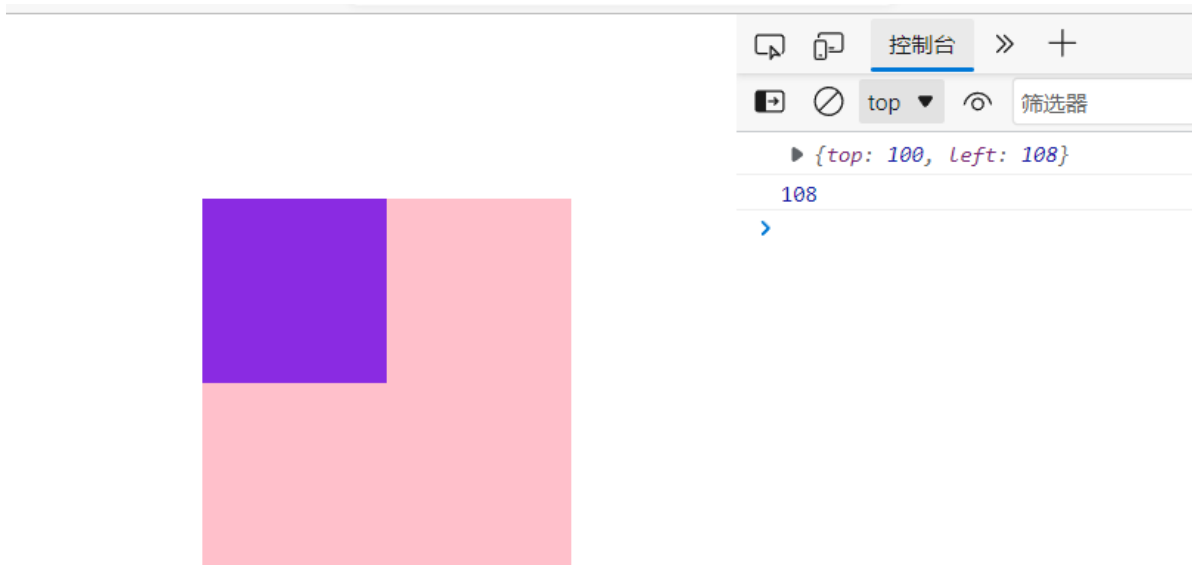
1. offset() 设置或获取元素偏移

- ① offset() 方法设置或返回被选元素相对于文档的偏移坐标，跟父级没有关系。
- ② 该方法有2个属性 left、top。offset().top 用于获取距离文档顶部的距离，offset().left 用于获取距离文档左侧的距离。
- ③ 可以设置元素的偏移：offset({ top: 10, left: 30 });

offset()是查看元素距离文档的位置，假如我们单纯查看的话，就是返回的对象，对象里面的值有left和top值，假如我们要查看的话，就在后面写上.top和.left就可以查看相应的值了

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">
      .father {
        width: 200px;
        height: 200px;
        background-color: pink;
        margin: 100px;
      }

      .son {
        width: 100px;
        height: 100px;
        background-color: blueviolet;
      }
    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function() {
        console.log($('.son').offset());
        console.log($('.son').offset().left);
      })
    </script>
  </head>
  <body>
    <div class="father">
      <div class="son"></div>
    </div>
  </body>
</html>
```



当然我们也可以去修改里面的值，并且这是关于文档的距离，和父元素没关系

```
$(function() {  
    $('.son').offset({  
        left:200,  
        top:200  
    });  
})
```

3.8.2.2 position()

2. position() 获取元素偏移

① position() 方法用于返回被选元素相对于**带有定位的父级**偏移坐标，如果父级都没有定位，则以文档为准。

```
<!DOCTYPE html>  
<html lang="zh">  
  <head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <meta http-equiv="X-UA-Compatible" content="ie=edge">  
    <title></title>  
    <style type="text/css">  
      * {  
        margin: 0px;  
        padding: 0px;  
      }  
  
      .father {  
        width: 200px;  
        height: 200px;  
        background-color: pink;  
        margin: 100px;  
        position: relative;  
      }  
  
      .son {
```

```

        width: 100px;
        height: 100px;
        background-color: blueviolet;
        position: absolute;
        left: 100px;
        top: 100px;
    }
</style>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">
    $(function() {
        console.log($('.son').position());
    })
</script>
</head>
<body>
    <div class="father">
        <div class="son"></div>
    </div>
</body>
</html>

```



3.8.2.3 被卷去的头部

下面代码写的是文档被卷曲的部分，在jQuery里面可以用scrollTop()来是实现

```

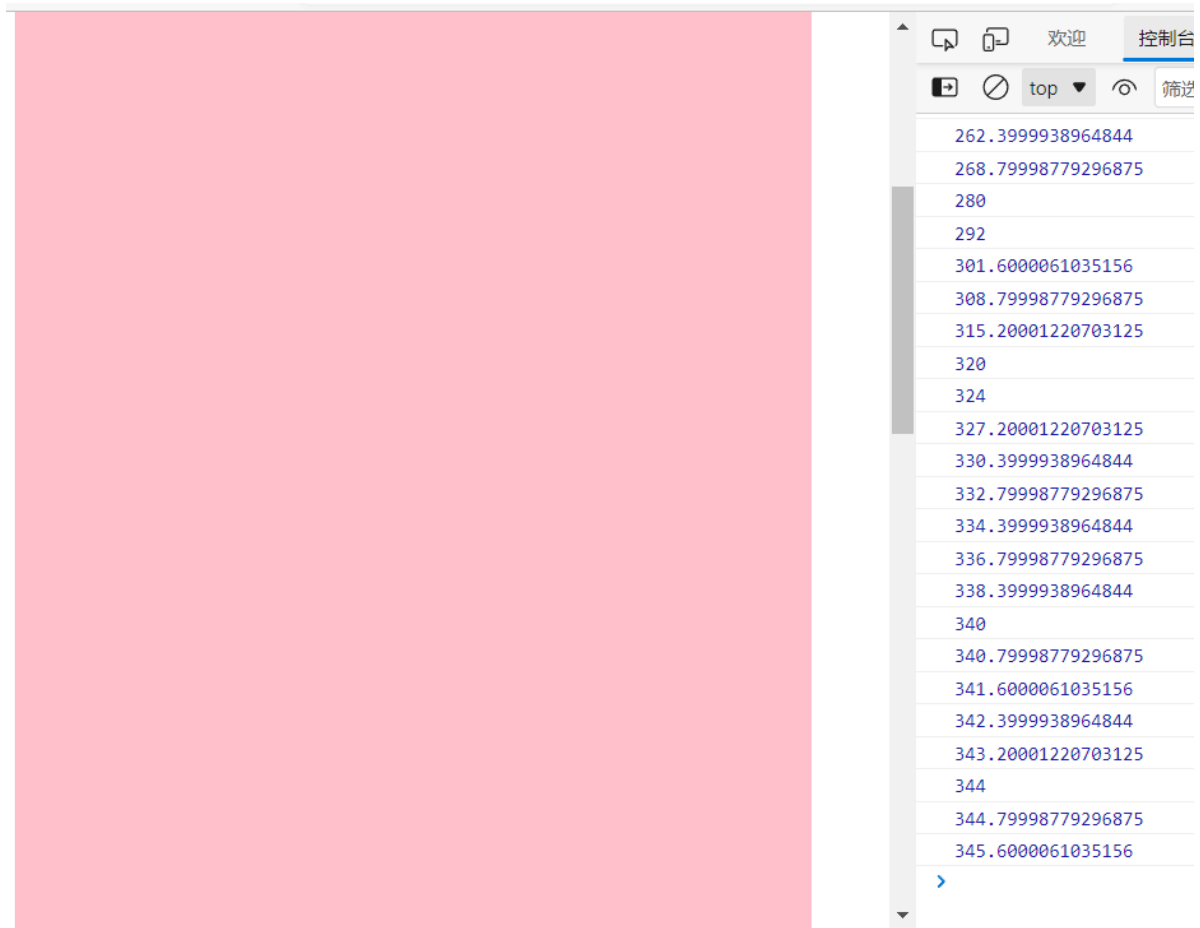
<!DOCTYPE html>
<html lang="zh">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title></title>
        <style type="text/css">
            .father{
                width: 500px;
                height: 2000px;
            }
        </style>
    </head>
    <body>
        <div class="father">
            <div class="son"></div>
        </div>
    </body>
</html>

```

```

        background-color: pink;
    }
</style>
<script src="js/Jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">
    $(function() {
        $(window).scroll(function(){
            console.log($(document).scrollTop());
        })
    })
</script>
</head>
<body>
    <div class="father"></div>
</body>
</html>

```



3.9 JQuery事件

3.9.1 事件注册

基本和上面的格式是一样的

语法：

```
element.事件(function() {})
```

```
$("#div").click(function() { 事件处理程序 })
```

其他事件和原生基本一致。

比如mouseover、mouseout、blur、focus、change、keydown、keyup、resize、scroll等

3.9.2 事件处理

这个就可以注册多个事件

on() 方法在匹配元素上绑定一个或多个事件的事件处理函数

语法：

```
element.on(events, [selector], fn)
```

1. events:一个或多个用空格分隔的事件类型，如"click"或"keydown"。
2. selector: 元素的子元素选择器。
3. fn:回调函数 即绑定在元素身上的侦听函数。

下面就是使用on来绑定事件

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">
      .main{
        width: 200px;
        height: 200px;
        background-color: pink;
      }
    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
    charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        $('div').on({
          mouseenter:function(){
            $(this).css('background-color','red');
          },
          click:function(){
```

```

        $(this).css('background-color','blue');
    }
    })
}
</script>
</head>
<body>
    <div class="main"></div>
</body>
</html>

```



假如你多个处理的函数是一样的话，就可以变成下面这个样子

```

$(function(){
    $('div').on("mouseenter click",function(){
        $(this).css('background-color','blue');
    });
})

```

事件的处理的优点

可以直接处理多个事件，并且可以实现事件委派的操作

下面代码就是事件委派的代码，第一个是原生事件委派的操作，父元素委派到子元素li上面

第二个是使用on的委派，是绑定在ul上，但是触发的对象是li

```

<!DOCTYPE html>
<html lang="zh">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title></title>
        <style type="text/css">

        </style>
        <script src="js/Jquery3.6-production.js" type="text/javascript"
        charset="utf-8"></script>
        <script type="text/javascript">

```



```

        $(function(){
            $('ul li').click(function(){

            }); //事件委托实现的
            $('ul').on("click","li",function(){

            });
        })
    </script>
</head>
<body>
    <ul>
        <li></li>
        <li></li>
        <li></li>
    </ul>
</body>
</html>

```

还有一个优点就是可以为动态创建的节点添加事件

这里有一个要注意的点，假如按照下面的方式来创建并且绑定是没有任何问题的

```

<!DOCTYPE html>
<html lang="zh">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title></title>
        <style type="text/css">

        </style>
        <script src="js/Jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
        <script type="text/javascript">
            $(function(){
                var li = $("<li>这是一个可爱的语句</li>");
                $('ol').append(li);
                $('ol li').click(function(){
                    alert("没有出来了哦");
                });
                // $('ul').on("click",'li',function(){
                //     alert("出来了哦");
                // });
            })
        </script>
    </head>
    <body>
        <ol>

        </ol>
    </body>
</html>

```

但是你按照下面的方式来创建的话就不行了，这是因为click不能绑定未来创建的li，也就是说在创建之前绑定，假如我们使用on的话就没有任何问题

```
$('#ol li').click(function(){
    alert("没有出来了哦");
});
// $('#ul').on("click", 'li', function(){
//     alert("出来了哦");
// });
var li = $("<li>这是一个可爱的语句</li>");
$('#ol').append(li);
```

3.9.3 事件解绑

使用off()方法来解绑事件

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">
      .main{
        width: 200px;
        height: 200px;
        background-color: pink;
      }
    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        $('#.main').on({
          click:function(){
            console.log("我点击了");
          },
          mouseover:function(){
            console.log("我经过了");
          }
        });
        // $('#.main').off(); //全部解绑
        $('#.main').off('click'); //解绑鼠标点击事件
      })
    </script>
  </head>
  <body>
    <div class="main"></div>
  </body>
</html>
```

假如说我们要接触事件委托的话

```
$('#ul1').off('click','li');
```

3.8.4 one()

用法和on是一样的，但是这个方法绑定的事件只能触发一次

3.8.5 自动触发事件

自动触发事件，就是不去执行事件的操作也可以自动执行这个事件，比如说轮播图的右按钮点击

下面的三种方法都是可以触发自动事件的，最后一个triggerHandler和上面2个区别就是不会触发元素的默认行为，比如：input的光标

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">
      .main{
        width: 200px;
        height: 200px;
        background-color: pink;
      }
    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        $('#.main').click(function(){
          console.log('我点击了');
        })
        // $('#.main').click();
        // $('#.main').trigger('click');
        // $('#.main').triggerHandler('click');
      })
    </script>
  </head>
  <body>
    <div class="main"></div>
  </body>
</html>
```

3.8.6 JQuery事件对象

语法基本和原生JS是一样的

```
$('.main').click(function(e){  
    console.log(e);  
});
```

事件被触发，就会有事件对象的产生。

```
element.on(events, [selector], function(event) {})
```

阻止默认行为：event.preventDefault() 或者 return false

阻止冒泡：event.stopPropagation()

3.10 JQuery其他方法

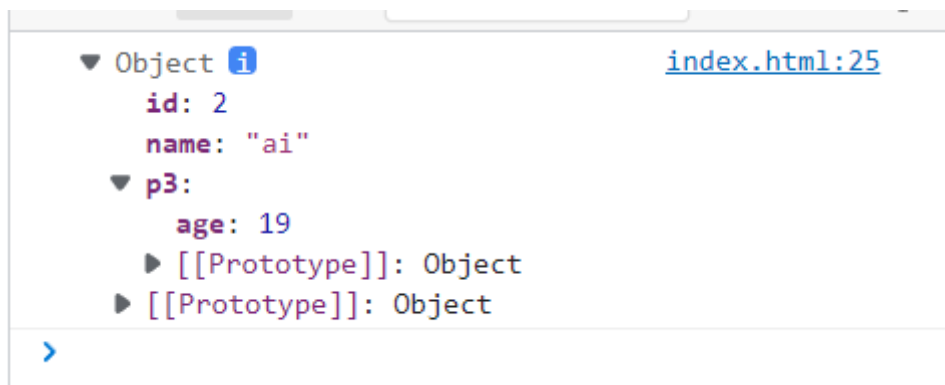
3.10.1 JQuery对象拷贝

这个东西不是很重要，记录一下留一个印象

基础的格式是\$.extend(被拷贝的对象，拷贝的对象);

首先我们来看结果，将p2所有的数据都拷给了p1，再来看，如果属性名是一样的话，就会覆盖

```
$(function(){  
    var p1 = {  
        id:1  
    };  
    var p2 = {  
        id:2,  
        name:"ai",  
        p3:{  
            age:19  
        }  
    };  
    $.extend(p1,p2);  
    console.dir(p1);  
})
```



extend默认的是浅拷贝

如果想要把某个对象拷贝（合并）给另外一个对象使用，此时可以使用 `$.extend()` 方法

语法：

```
$.extend([deep], target, object1, [objectN])
```

1. deep: 如果设为true为深拷贝，默认为false 浅拷贝
2. target: 要拷贝的目标对象
3. object1: 待拷贝到第一个对象的对象。
4. objectN: 待拷贝到第N个对象的对象。
5. 浅拷贝是把被拷贝的对象复杂数据类型中的地址拷贝给目标对象，修改目标对象会影响被拷贝对象。

如果是深拷贝的话就直接将原对象所有东西直接拷贝给被拷贝的对象，没有浅拷贝里面影响被拷贝对象

3.10.2 多库共存

下面就是将原本的\$改为sc了，这样的话，以后写jQuery的代码就是使用的sc来使用，但是原本的是可以使用的

```
var sc = $.noConflict();
```

这个就是多库共存的问题，以免你写的\$的方法和jQuery不兼容

3.8 案例

3.8.1 下拉菜单

我们使用jQuery的语法就会简单很多

这里使用jQuery的话还有一些注意的地方，首先是选择器，我们获取过来的是.nav下面的li，这有4个，按照原生JS的写法，就必须一个个绑定鼠标移动的事件，但是jQuery有隐式迭代，就不用写那么多语法格式了，另外，jQuery绑定事件也是一样的，直接在后面.事件 就可以了，里面直接写函数，\$(this)在jQuery里面代表的是this

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <style type="text/css">
      .nav{
        width: 500px;
        list-style: none;
        display: flex;
        justify-content: space-around;
      }
    </style>
  </head>
  <body>
    <div class="nav">
      <a href="#">Home</a>
      <a href="#">About</a>
      <a href="#">Contact</a>
      <a href="#">Services</a>
    </div>
  </body>
</html>
```

```

        .nav li a{
            text-decoration: none;
            color: pink;
            font-size: 20px;
        }
        .nav li ul{
            list-style: none;
            margin-left: -30px;
            margin-top: 10px;
            display: none;
        }
    </style>
    <script type="text/javascript">
        $(function(){
            $('<strong>.nav>li</strong>').mouseover(function(){
                $(this).children("ul").show();
            })

            $('<strong>.nav>li</strong>').mouseout(function(){
                $(this).children("ul").hide();
            })
        })
    </script>
</head>
<body>
    <ul class="nav">
        <li>
            <a href="#">下拉</a>
            <ul>
                <li>1</li>
                <li>2</li>
                <li>3</li>
            </ul>
        </li>
        <li>
            <a href="#">下拉</a>
            <ul>
                <li>1</li>
                <li>2</li>
                <li>3</li>
            </ul>
        </li>
        <li>
            <a href="#">下拉</a>
            <ul>
                <li>1</li>
                <li>2</li>
                <li>3</li>
            </ul>
        </li>
        <li>
            <a href="#">下拉</a>
            <ul>
                <li>1</li>
                <li>2</li>
            </ul>
        </li>
    </ul>

```

```

        <li>3</li>
      </ul>
    </li>
  </ul>
</body>
</html>

```

下拉

下拉

下拉

下拉

1
2
3

3.8.2 排他思想

jQuery的排他思想是很好做的，下面就是语法格式

```

<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
    <style type="text/css">
      .nav{
        width: 500px;
        list-style: none;
        display: flex;
        justify-content: space-around;
      }
      .nav li{
        color: pink;
        font-size: 25px;
        cursor: pointer;
      }
    </style>
    <script type="text/javascript">
      $(function(){
        $('<strong>.nav</strong>').children('<strong>li</strong>').mouseover(function(){
          $(this).css('color','red');
          $(this).siblings('<strong>li</strong>').css('color','');
        })
      })
    </script>
  </head>
  <body>
    <ul class="nav">

```

```

        <li>1</li>
        <li>2</li>
        <li>3</li>
        <li>4</li>
    </ul>
</body>
</html>

```

1

2

3

4

3.8.3 淘宝服饰

```

<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">
      .main{
        display: flex;
        justify-content: center;
      }
      .main_ul li{
        list-style: none;
        width: 20px;
        height: 20px;
        background-color: pink;
        margin-top: 30px;
        cursor: pointer;
      }
      .main_img{
        width: 210px;
        height: 250px;
        border: 1px solid red;
      }
      .main_img img{
        width: 200px;
        height: 250px;
        position: absolute;
        left: 220px;
        top: 9px;
      }
    </style>
    <script src="js/jquery3.6-production.js" type="text/javascript"
    charset="utf-8"></script>
    <script type="text/javascript">
      $(function(){
        $('<strong>.main_ul li</strong>').mouseover(function(){
          var index = $(this).index();

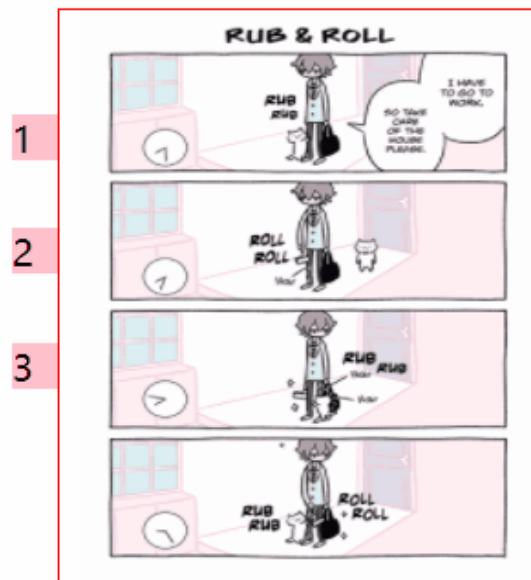
```



```

        $('.main_img img').eq(index).show();
        $('.main_img img').eq(index).siblings('img').hide();
    })
}
</script>
</head>
<body>
    <div class="main">
        <ul class="main_ul">
            <li>1</li>
            <li>2</li>
            <li>3</li>
        </ul>
        <div class="main_img">
            
            
            
        </div>
    </div>
</body>
</html>

```



3.8.4 突出显示

```

<!DOCTYPE html>
<html lang="zh">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title></title>
        <style type="text/css">
            body{
                background-color: darkcyan;
            }
            .wrap{

```

```

        display: flex;
        justify-content: center;
        align-content: center;
        flex-wrap: wrap;
    }
    .wrap li{
        list-style: none;
    }
    .wrap li img{
        width: 300px;
        height: 300px;
    }
</style>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">
    $(function(){
        $('<div>.wrap>li</div>').hover(
            function(){
                $(this).siblings().stop().fadeTo(400,0.5);
            },function(){
                $(this).siblings().stop().fadeTo(400,1);
            }
        );
    })
</script>
</head>
<body>
    <ul class="wrap">
        <li></li>
        <li></li>
        <li></li>
        <li></li>
        <li></li>
        <li></li>
    </ul>
</body>
</html>

```



3.8.5 王者手风琴

P384, P385

3.8.6 购物车

P387, P388, P390, P391-P393, P396-P399

3.8.7 带有动画的返回顶部

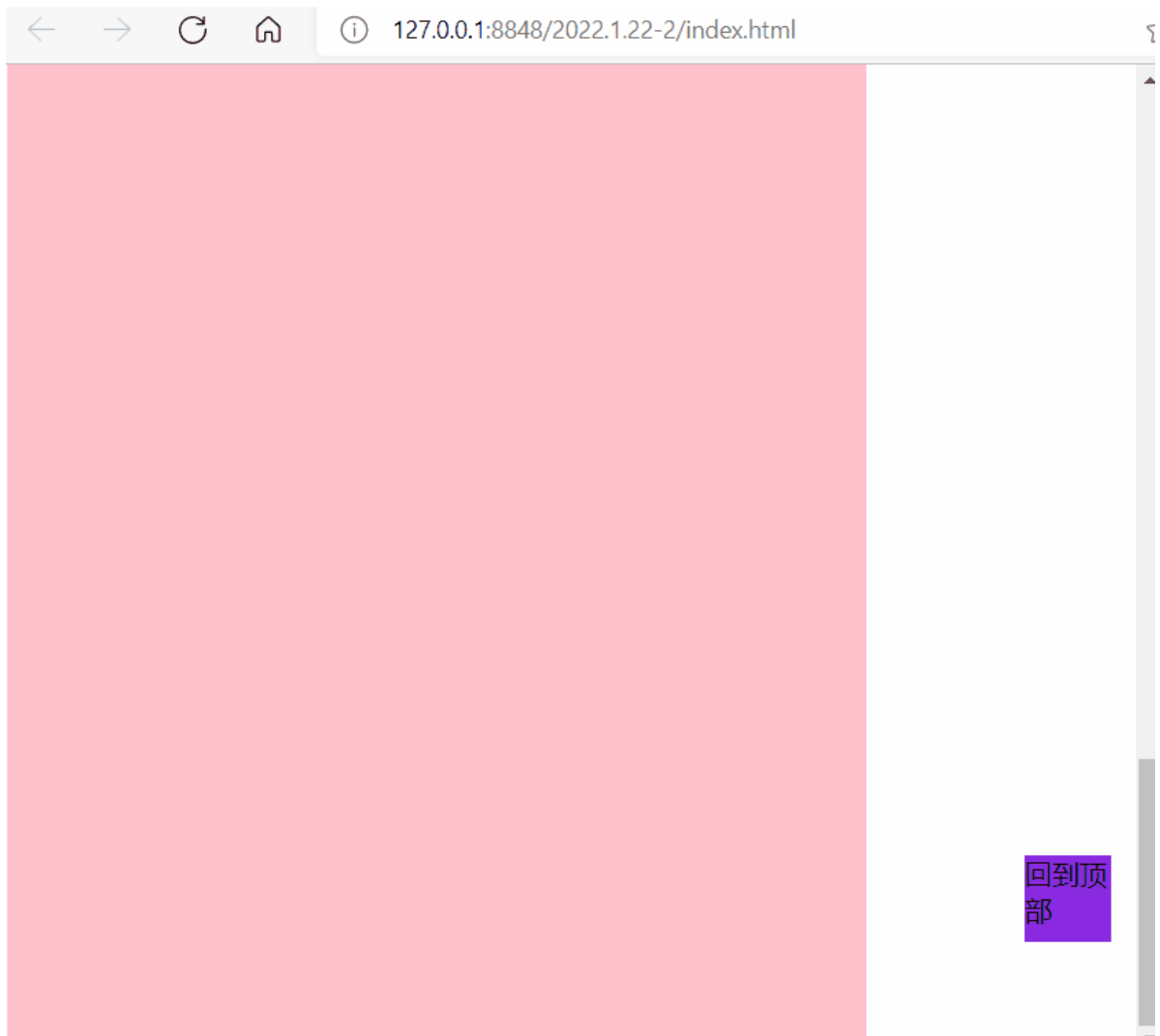
这里有一个要注意的点，就是要滚动的不是文档document，而是元素滚动，所以这里使用的是body和html

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title></title>
    <style type="text/css">
      .father{
        width: 500px;
        height: 2000px;
        background-color: pink;
      }
      .hui{
        width: 50px;
        height: 50px;
        background-color: blueviolet;
        position: absolute;
        left: 600px;
        top: 1900px;
      }
    </style>
```

```

<script src="js/Jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">
    $(function() {
        $('.hui').click(function(){
            $('body,html').stop().animate({
                scrollTop:0
            })
        });
    })
</script>
</head>
<body>
    <div class="father"></div>
    <div class="hui">回到顶部</div>
</body>
</html>

```



3.8.8 电梯导航

```

<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title></title>
<style type="text/css">
    * {
        margin: 0px;
        padding: 0px;
    }

    .one,
    .two,
    .three,
    .four {
        width: 500px;
        height: 700px;
    }

    .one {
        background-color: #8A2BE2;
    }

    .two {
        background-color: brown;
    }

    .three {
        background-color: chocolate;
    }

    .four {
        background-color: cornflowerblue;
    }

    .hui_ul {
        position: fixed;
        left: 580px;
        top: 200px;
    }

    .hui_ul li {
        width: 50px;
        height: 50px;
        list-style: none;
        margin-left: -40px;
        border: 1px solid #6495ED;
        cursor: pointer;
    }

    .current {
        background-color: pink;
    }
</style>
<script src="js/jquery3.6-production.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">

```

```

$(function() {
    var flag = true;

    var one_top = $('.one').offset().top;
    var two_top = $('.two').offset().top;
    var three_top = $('.three').offset().top;
    var four_top = $('.four').offset().top;

    //防止页面加载的时候，电梯消失
    sc_sc();
    function sc_sc() {
        if ($(document).scrollTop() >= 100) {
            $('.hui_ul').fadeIn();
        } else {
            $('.hui_ul').fadeOut();
        }
    }
    $(window).scroll(function() {
        sc_sc();
        if (flag) {
            $('.main div').each(function(index, ele) {
                if ($(document).scrollTop() >= $(ele).offset().top)
                {
                    $('.hui_ul


```

```
<li index="0" class="current">页面A</li>
<li index="1">页面B</li>
<li index="2">页面C</li>
<li index="3">页面D</li>
</ul>
</body>
</html>
```

页面A



3.8.9 微博发布案例

P412