# 下载jQuery

1. 官网：<https://jquery.com/>
2. 版本介绍

jQuery如今有三个大版本，分别是1.x、2.x、3.x

其中只有1.x支持IE6、7、8，其他版本均不支持

每个大版本还分为compressed版和uncompressed版，其中compressed版是压缩版，

在发布项目时一般选择它，因为它体积小，加载速度快；而uncompressed版是未压缩版，体积大，但有注释以及缩进，适合开发时使用。

# 原生js入口函数和jQuery入口函数

1.入口函数的加载时机

    <!--

        原生js会等到DOM元素加载完毕，包括图片加载完毕才会执行

        而jQuery会等到DOM元素加载完，但不会等图片加载完毕就会执行

    -->

    <img src="../images/baby甜甜呀.png">

    <script>

        window.onload = function(ev){

            //通过js入口函数拿到DOM元素

            var img = document.getElementsByTagName("img")[0];

            console.log(img);

            //通过原生js函数拿到DOM元素的宽高

            var width = window.getComputedStyle(img).width;

            console.log('onload' , width);

        }

    </script>

    <script>

        $(document).ready(function(){

            //通过jQuery入口函数拿到DOM元素

            var img = $("img")[0];

            console.log(img);

            //jQuery1.12.4的函数拿不到DOM元素的宽高

            var width = img.width;

            console.log('ready' , width);

        });

    </script>

2.原生的和jQuery的入口函数是否会覆盖前面的入口函数

<script>

    /\*

         一个页面中，如果原生的js编写了多个入口函数，

         后面的会覆盖前面的，导致前面的入口函数不会执行

    \*/

    window.onload = function(ev){

       alert('onload 1');

    }

    window.onload = function(ev){

       alert('onload 2');

    }

    //编写多个jQuery的入口函数，后面的不会覆盖前面的，都会执行

    $(document).ready(function(){

        alert('ready 1');

    });

    $(document).ready(function(){

        alert('ready 2');

    });

</script>

3.jQuery入口函数的四种书写方式(其实是两种)，jQuery = $，二者都是jQuery的对象名

<script>

    //第一种写法

    $(document).ready(function(){

        alert('第一种写法');

    });

    //第二种写法

    jQuery(document).ready(function(){

        alert('第二种写法');

    });

    //第三种写法(推荐)

    $(function(){

        alert('第三种写法');

    });

    //第四种写法

    jQuery(function(){

        alert('第四种写法');

    });

</script>

# jQuery变量名的冲突问题

<script src="../js/jquery-1.12.4.min.js"></script>

<script src="../js/test.js"></script>

<script>

    /\*

        已知当js变量名重复时，后面的会覆盖前面的值。

        test.js里有变量名$，且它在jQuery.js后面引入，

        所以jQuery的$会被test的$所覆盖，无法正常使用jQuery

    \*/

    //方式一(不再使用$，使用jQuery作为变量名)

    jQuery.noConflict();//禁用jQuery的$

    //方式二(重新给jQuery的变量赋名字)

    var j = jQuery.noConflict();

    j(function(){

        alert('j');

    });

    //jQuery仍然可以用

    jQuery(function(){

        alert('jQuery');

    });

</script>

# jQuery核心函数

<script>

    $();//这就是jQuery的核心函数，用处如下。

    //1.接收一个函数

    $(function(){

        alert('接收一个函数');

        //2.1接收一个字符串形式的选择器

        var box1 = $('.box1');

        var box2 = $('#box2');

        console.log(box1);

        console.log(box2);

        //2.2接收一个字符串代码片段

        var p = $('<p>我是段落</p>');

        console.log(p);

        box1.append(p);//box1添加子节点p

        //3.接收一个DOM元素

        var span = document.getElementsByTagName('span')[0];

        console.log(span);

        span = $(span);

        console.log(span);

    });

</script>

# jQuery对象

<script>

    $(function(){

        /\*

            jQuery对象是一个伪数组。它有属性length

            为其长度；有下标，即0~length-1

        \*/

        var div = $('div');

        console.log(div);

    });

</script>

<div>div1</div>

<div>div2</div>

<div>div3</div>

# 静态方法和实例方法

<script>

   //定义一个类(函数模拟的类)

   function AClass(){}

   //给这个类添加静态方法

   AClass.staticMethod = function(){

       alert('staticMethod');

   }

   //静态方法通过类名调用

   AClass.staticMethod();

   //给这个类添加实例方法

   AClass.prototype.instanceMethod = function(){

       alert('instanceMethod');

   }

   //通过对象名调用实例方法

   var a = new AClass();

   a.instanceMethod();

</script>

# 静态方法each

<script>

   var arr = [1,3,5,7,9];

   //原生forEach方法只能遍历数组。index是下标，value是对应的值。

   arr.forEach(function(value,index){

        console.log(index,value);

   });

    var obj = {0:1,1:3,2:5,3:7,4:9,length:5};

   //jQuery的each方法即可以遍历数组，也可以遍历伪数组

   $.each(arr,function(index,value){

        console.log(index,value);

   });

   $.each(obj,function(index,value){

        console.log(index,value);

   });

</script>

# 静态方法map

<script>

   var arr = [1,3,5,7,9];

   var obj = {0:1,1:3,2:5,3:7,4:9,length:5};

   /\*

      原生map方法，index是下标，value是对应的值，哪个数组调map，

      参数array就是哪个数组，这里array就是arr。map方法同样只能遍历数组。

   \*/

   arr.map(function(value,index,array){

      console.log(index,value,array);

   });

   //map方法和each的作用一样，遍历数组和伪数组

   $.map(arr,function(value,index){

      console.log(index,value);

   });

   $.map(obj,function(value,index){

      console.log(index,value);

   });

   /\*

      map和each方法的区别:

      (1)map方法默认返回一个空数组

      (2)each方法固定返回自己的形参数组(伪数组)

      (3)map可以在函数体内使用return语句来修改返回的空数组

   \*/

   var res1 = $.map(obj,function(value,index){

      console.log(index,value);

      //使用return语句修改返回默认的空数组

      return value + index;

   });

   var res2 = $.each(obj,function(index,value){

      console.log(index,value);

   });

   console.log(res1);

   console.log(res2);

</script>

# 其他静态方法

<script>

   //$.trim()：去除字符串两端空格，返回去除空格之后的字符串

   var str = '   lnj   ';

   var res = $.trim(str);

   console.log('---'+str+'---');//---   lnj   ---

   console.log('---'+res+'---');//---lnj---

   //$.isWindow()：判断一个对象是否是window对象，返回true/false

   var a = '123';

   var w = window;

   console.log($.isWindow(a));//false

   console.log($.isWindow(w));//true

   //$.isArray()：判断一个对象是否是数组，返回true/false

   var arr = [1,3,5,7,9];

   console.log($.isArray(arr));//true

   console.log($.isArray(a));//false

   //$.isFunction()：判断一个对象是否是函数，返回true/false

   var fun = function(){}

   console.log($.isFunction(fun));//true

   console.log($.isFunction(jQuery));//true

   console.log($.isFunction(a));//false

</script>

# 静态方法holdReady

<script>

    //$.holdReady(true);可以停止ready函数的执行

    $.holdReady(true);

    //当页面所有DOM元素加载完后才会执行ready函数

    $(document).ready(function(){

        alert('ready');

    });

</script>

<button>恢复ready事件</button>

<script>

    $('button').click(function(){

        //恢复ready函数执行

        $.holdReady(false);

    });

</script>

# 内容选择器

<script>

    $(function(){

        //查找div中没有内容且没有子元素的

        var div1 = $('div:empty');

        //查找div中有内容或有子元素的

        var div2 = $('div:parent');

        //查找div中包含(包含此字符串，不用一模一样)内容是 我是div 的

        var div3 = $('div:contains("我是div")');

        //查找div中包含子元素span的

        var div4 = $('div:has("span")');

    });

</script>

# 属性和属性节点

<script>

    $(function(){

        //1.对象身上保存的变量就是属性

        function Person(){}

        var p = new Person();

        p.name = null;//这里name就是对象p的属性

        //2.给属性name赋值

        p.name = 'lnj';//方式一

        p['name'] = 'ha';//方式二

        console.log(p.name);//ha

        console.log(p['name']);//ha

        /\*

            3.在HTML标签(DOM对象)身上添加的属性即为属性节点

                如：<span name="it"></span>

                name就是属性节点

        \*/

        /\*

            4.js原生操作属性节点

                (1)设置属性节点的值

                    var span = $('span');

                    span.setAttribute("name","lnj");

                (2)得到属性节点的值

                    var val = span.getAttribute("name");

        \*/

    });

</script>

# attr方法和removeAttr方法

<script>

    $(function(){

        /\*

            attr(name|properties|key,value|fn):获取或设置属性节点的值

                (1)若只给一个参数，代表获取属性节点的值

                (2)若给两个参数，代表设置属性节点的值

        \*/

        //注意：该方法只会获取第一个元素的属性节点的值,如：

        console.log($('span').attr('class'));//s1

        //注意：但如果设置属性节点的值时，会都设置

        $('span').attr('class','box');//两个span的class值都成了box

        //如果设置的属性节点不存在，系统会自动新增

        $('span').attr('abc','123');

        /\*

            removeAttr(name):删除所有找到元素指定的属性节点

        \*/

        $("span").removeAttr("class");//删除所有span的class

        $("span").removeAttr("class name");//删除所有span的class和name

    });

</script>

<span class="s1" name="dd"></span>

<span class="s2"></span>

# prop方法和removeProp方法

<script>

    //prop()和removeProp()不仅能操作属性，也能操作属性节点

    $(function(){

        /\*

            prop():获取或设置属性的值

                (1)若只给一个参数，代表获取属性的值

                (2)若给两个参数，代表设置属性的值

        \*/

        //注意：该方法只会获取第一个元素的属性的值,如：

        console.log($('span').prop('class'));//s1

        //也可以使用eq()方法选择获取第几个span

        console.log($('span').eq(1).prop('class'));//s2

        //注意：但如果设置属性的值时，会都设置

        $('span').prop('class','box');//两个span的class值都成了box

        //如果设置的属性不存在，系统会自动新增

        $('span').eq(0).prop('abc','123');

        console.log($('span').prop('abc'));

        /\*

            removeProp(name):删除所有找到元素指定的属性的值

        \*/

        $("span").removeProp("class");//删除所有span的class的值

        /\*

            什么时候用attr或者prop来获取值呢？

            官方推荐，具有true和false两个属性的属性节点，如checked、selected

            或者disabled使用prop(),其他使用attr()

        \*/

        console.log($('input').eq(0).prop('checked'));//true

        console.log($('input').eq(0).attr('checked'));//checked

        console.log($('input').eq(1).prop('checked'));//false

        console.log($('input').eq(1).attr('checked'));//undefined

    });

</script>

<span class="s1" name="dd"></span>

<span class="s2"></span>

<input type="checkbox" checked/>

<input type="checkbox"/>

# 操作class的方法

<style>

    .class1{

        width: 100px;

        height: 100px;

        background-color: red;

    }

    .class2{

        width: 100px;

        height: 100px;

        border: 10px solid black;

    }

</style>

<script>

    $(function(){

        //addClass():给对象添加class的值

        $("button").eq(0).click(function(){

            // $("div").addClass("class1");

            $("div").addClass("class1 class2");

        });

        //removeClass():给对象删除class的值

        $("button").eq(1).click(function(){

            // $("div").removeClass("class1");

            $("div").removeClass("class1 class2");

        });

        //toggleClass():对象class有值就删除class，没有就添加

        $("button").eq(2).click(function(){

            // $("div").toggleClass("class1");

            $("div").toggleClass("class1 class2");

        });

//补充，这3个方法返回对象本身，因此可以链式调用。

//如：$("div").addClass("class1").removeClass("class2");

    });

</script>

<button>添加class</button>

<button>删除class</button>

<button>添加/删除class</button>

<div></div>

# 操作文本值的方法

<script>

    $(function(){

        //html():和原生js的innerHTML一模一样

        $("button").eq(0).click(function(){

            $("div").html("<p>段落<span>span</span></p>");

        });

        $("button").eq(1).click(function(){

            console.log($("div").html());

        });

        //text():和原生js的innerText一模一样

        $("button").eq(2).click(function(){

            $("div").text("<p>段落<span>span</span></p>");

        });

        $("button").eq(3).click(function(){

            console.log($("div").text());

        });

        //val():设置或获取属性value的值

        $("button").eq(4).click(function(){

            $("input").val("请输入内容");

        });

        $("button").eq(5).click(function(){

            console.log($("input").val());

        });

    });

</script>

<button>设置html</button>

<button>获取html</button>

<button>设置text</button>

<button>获取text</button>

<button>设置value</button>

<button>获取value</button>

<div style="width:150px;height:150px;border: 5px solid black;"></div>

<input type="text"/>

# 操作css的方法

<script>

    $(function(){

        //1.逐个设置css

        $('div').css("width","100px");

        $('div').css("height","100px");

        $('div').css("background","red");

        $('div').css("border","10px solid red");

        //2.链式设置css,超过三个，建议分开

        $('div').css("width","100px").css("height","100px")

        .css("background","blue");

        //3.批量设置

        $('div').css({

            width:"100px",

            height:"100px",

            background:"black"

        });

        //4.获取css样式值

        console.log($("div").css("width"));

    });

</script>

<div></div>

# 尺寸和位置操作

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    .father{

        width: 200px;

        height: 200px;

        background: red;

        border: 50px solid #000;

        margin-left: 50px;

        position: relative;

    }

    .son{

        width: 100px;

        height: 100px;

        background: blue;

        position: absolute;

        left: 50px;

        top:50px;

    }

</style>

<script>

    $(function(){

       $('button').eq(0).click(function(){

           //获取元素宽度

           console.log($(".father").width());//200

           //获取元素高度

           console.log($(".father").height());//200

           //offset():获取或设置元素距离窗口的偏移量

           //获取左偏移量

           console.log($(".son").offset().left);//150

           //position():获取元素距离定位元素的偏移量

           //获取左偏移量

           console.log($(".son").position().left);

       });

       $('button').eq(1).click(function(){

           //设置元素宽度

           $(".father").width("300px");

           //设置元素高度

           $(".father").height("300px");

           //设置左偏移量

           $(".son").offset({

               left:10

           });

       });

    });

</script>

<div class="father">

    <div class="son"></div>

</div>

<button>获取</button>

<button>设置</button>

# scrollTop()方法

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    .scroll{

        width: 100px;

        height: 100px;

        border: 1px solid red;

        overflow: auto;

    }

</style>

<script>

    $(function(){

       $("button").eq(0).click(function(){

           //获取滚动的的偏移量

           console.log($(".scroll").scrollTop());

           //获取网页滚动的偏移量

           console.log($("html").scrollTop());

           //获取网页滚动的偏移量，为了兼容IE，还得加上body

           console.log($("html").scrollTop()+$("body").scrollTop());

       });

       $("button").eq(1).click(function(){

           //设置滚动的的偏移量

           $(".scroll").scrollTop(100);

           //设置网页滚动的偏移量(兼容IE)

           $("html,body").scrollTop(200);

       });

    });

</script>

<div class="scroll">

    我是div我是div我是div我是div我是div我是div我是div

    我是div我是div我是div我是div我是div我是div我是div

    我是div我是div我是div我是div我是div我是div我是div

</div>

<button>获取</button>

<button>设置</button>

<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>

<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>

<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>

<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>

# 事件绑定

<script>

    $(function(){

       /\*

            jQuery中有两种绑定事件方式

            1.eventName(fn);

                效率高，但部分事件jQuery没有实现，不能添加。

                多个相同或者不同的事件不会覆盖。

            2.on(eventName,fn);

                效率低，但所有js事件都可以添加。

                多个相同或者不同的事件也不会覆盖。

       \*/

       //以点击事件绑定为例

       $("button").click(function(){

           alert('click');

       });

       $("button").on("click",function(){

           alert('on');

       });

    });

</script>

<button>点击</button>

# 事件绑定移除

<script>

    $(function(){

       var test1 = function(){alert('click11111');}

       var test2 = function(){alert('click22222');}

       $("button").click(test1);

       $("button").click(test2);

       $("button").mouseenter(function(){

           alert('mouseenter');

       });

       //off():移除事件绑定，分三种情况。

       //1.不传参数：移除所有绑定事件

       $("button").off();

       //2.传一个参数，移除所有指定类型的绑定事件

       $("button").off("click");

       //3.传两个参数，移除所有指定类型的指定事件

       $("button").off("click",test1);

    });

</script>

<button>点击</button>

# 事件冒泡和默认行为

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    .father{

        width: 200px;

        height: 200px;

        background: red;

    }

    .son{

        width: 100px;

        height: 100px;

        background: blue;

    }

</style>

<script>

    $(function(){

        /\*

            事件冒泡：指在调用子级触发事件时，父级相同事件随后也会触发。

                本例展示为：父级div的范围包括了子级div，实现子级点击

                    事件后，而父级的点击事件随之也被触发了。有两种方式

                    阻止事件冒泡的发生。

        \*/

        $(".son").click(function(event){

            alert("son");

            //1.返回false，便能阻止事件冒泡

            // return false;

            //2.使用event回调参数的方法也能阻止事件冒泡

            event.stopPropagation();

        });

        $(".father").click(function(){

            alert("father");

        });

        /\*

            默认行为：点击<a>标签默认会跳转，点击提交按钮默认会提交数据，这

                便是默认行为。有时候需要阻止这种默认行为。有两种方式阻止默认

                行为的发生。

        \*/

        $("a").click(function(event){

            //1.返回false，能阻止默认行为的发生

            // return false;

            //2.使用event回调参数的方法也能阻止默认行为

            event.preventDefault();

        });

    });

</script>

<div class="father">

    <div class="son"></div>

</div>

<a href="http://www.baidu.com">我是百度</a>

# 自动触发事件

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    .father{

        width: 200px;

        height: 200px;

        background: red;

    }

    .son{

        width: 100px;

        height: 100px;

        background: blue;

    }

</style>

<script>

    $(function(){

        /\*

            自动触发事件：顾名思义，即打开页面时自动触发的事件，有两种实现方式。

                1.trigger()方法

                2.triggerHandler()方法

            二者区别：

                1.当子级没有阻止事件冒泡时，它使用trigger()会出现事件冒泡；

                    但使用triggerHandler()不会出现事件冒泡。

                2.除了<a>标签，当其他标签有默认行为而没有阻止时，它使用

                    trigger()会触发默认行为；但使用triggerHandler()不会。

        \*/

        $(".son").click(function(event){

            alert("son");

            //没有阻止事件冒泡

        });

        $(".father").click(function(){

            alert("father");

        });

        // $(".son").trigger("click");

        // $(".son").triggerHandler("click");

        /\*

            如果你仍然想用<a>的自动触发方法且能正常跳转页面，那么你可以用<a>

                包裹一个<span>标签，而点击方法和trigger()方法的对象都用该

                <span>对象就行了。

        \*/

        $("span").click(function(event){

            alert('<a>包裹的<span>');

            //没有阻止事件冒泡

        });

        $("span").trigger("click");

        // $("span").triggerHandler("click");

    });

</script>

<div class="father">

    <div class="son"></div>

</div>

<a href="http://www.baidu.com"><span>我是百度</span></a>

# 自定义事件

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    .father{

        width: 200px;

        height: 200px;

        background: red;

    }

</style>

<script>

    $(function(){

        /\*

            自定义事件：即自制的事件，需要满足两个条件。

                1.该事件必须通过on绑定。

                2.该事件必须通过trigger或triggerHandler来触发。

            如下例自定义的myClick事件：

        \*/

        //1.使用on绑定

        $(".father").on("myClick",function(){

            alert("father");

        });

        //2.通过trigger或triggerHandler来触发

        // $("father").trigger("myClick");

        $(".father").triggerHandler("myClick");

    });

</script>

<div class="father"></div>

# 事件命名空间

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    .father{

        width: 200px;

        height: 200px;

        background: red;

    }

</style>

<script>

    $(function(){

        /\*

            事件命名空间：同一个页面可能由多人开发，导致多人可能对同一个元素

                添加了多个相同类型的事件，这会造成混乱，我们可以使用事件命名

                空间避免这种事情发生。

                如何实现(如下例)：

                    1.必须通过on绑定事件，on函数的第一个参数形式为：

                        "事件类型.自定义命名"

                    2.通过trigger或triggerHandler来触发

        \*/

        $(".father").on("click.zs",function(){

            alert("click1");

        });

        $(".father").on("click.ls",function(){

            alert("click2");

        });

        // $(".father").trigger("click.zs");

        $(".father").triggerHandler("click.zs");

        /\*

            注意：

                1.当子级不阻止事件冒泡且使用了trigger()触发带有命名空间的事

                    件时，如果它的父级使用了相同的事件和命名空间，那么该父级

                    的这个事件也会被触发，但这个父级的其他相同事件因为没有或

                    命名空间不同而不会被触发。

                2.当子级不阻止事件冒泡且使用了trigger()触发不带有命名空间的

                    事件时，该子级的所有相同类型的事件会被触发，其父级的所有

                    相同类型的事件也会被触发，命名空间此时不会影响事件的触发。

        \*/

    });

</script>

<div class="father"></div>

# 事件委托

<script>

    $(function(){

        $("button").click(function(){

            $("ul").append("<li>我是新增的li</li>");

        });

        /\*

            通过选择器获取了多个对象，这些对象都会被绑定点击事件

                但新增的<li>没有被绑定点击事件。

        \*/

        $("ul>li").click(function(){

            console.log($(this).html());

        });

        /\*

            事件委托：自己做不了的事，请别人(别的对象和方法)去做。

            delegate() 方法为指定的元素（属于被选元素的子元素）添

                加一个或多个事件处理程序，并规定当这些事件发生时运

                行的函数。使用 delegate() 方法的事件处理程序适用于

                当前或未来的元素（比如由脚本创建的新元素）。

            $(selector).delegate(childSelector,event,data,function)

        \*/

        //例：新增的li也能绑定点击事件。

        $("ul").delegate("li","click",function(){

            console.log($(this).html());

        });

    });

</script>

<ul>

    <li>我是第一个li</li>

    <li>我是第二个li</li>

    <li>我是第三个li</li>

</ul>

<button>新增一个li</button>

# 移入移出事件

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    .father{

        width: 200px;

        height: 200px;

        background: red;

    }

    .son{

        width: 100px;

        height: 100px;

        background: blue;

    }

</style>

<script>

    $(function(){

        //mouseover/mouseout事件，子元素被移入移出也会触发父元素的这些事件

        /\* $(".father").mouseover(function(){

            console.log("father被移入了");

        });

        $(".father").mouseout(function(){

            console.log("father被移出了");

        });\*/

        //mouseenter/mouseleave事件,子元素被移入移出不触发父元素的这些事件

        //推荐使用mouseenter/mouseleave

        /\*  $(".father").mouseenter(function(){

            console.log("father被移入了");

        });

        $(".father").mouseleave(function(){

            console.log("father被移出了");

        }); \*/

        //有一个函数能同时监听移入和移出的事件

        //hover()其实是封装了mouseenter/mouseleave事件

        $(".father").hover(function(){

            //第一个形参代表移入时触发的函数

            console.log("father被移入了");

        },function(){

            //第二个形参代表移出时触发的函数

            console.log("father被移出了");

        });

        $(".father").hover(function(){

            //若只有一个形参代表移入/移出时触发的函数

            console.log("father被移入/移出了");

        });

    });

</script>

<div class="father">

    <div class="son"></div>

</div>

# (练习)电影排行榜

代码在F:\VScode\VSCode\_workspace\jquery\28.(练习)电影排行榜.html

# (练习)Tab选项卡

代码在F:\VScode\VSCode\_workspace\jquery\29.(练习)Tab选项卡.html

# 显示/隐藏动画

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    div{

        width: 200px;

        height: 200px;

        background: red;

        display: none;

    }

</style>

<script>

    $(function(){

        //show():显示动画。

        $("button").eq(0).click(function(){

            $("div").show(1000,function(){

                //动画显示完毕要做的事

                alert("动画显示执行完毕");

            });

        });

        //hide():隐藏动画。

        $("button").eq(1).click(function(){

            $("div").hide(1000,function(){

                //动画隐藏完毕要做的事

                alert("动画隐藏执行完毕");

            });

        });

        //toggle():显示时隐藏动画，隐藏时显示动画。即来回切换状态

        $("button").eq(2).click(function(){

            $("div").toggle(1000,function(){

                //动画显示/隐藏完毕要做的事

                alert("动画显示/隐藏执行完毕");

            });

        });

    });

</script>

<button>显示</button>

<button>隐藏</button>

<button>切换</button>

<div></div>

# (练习)对联广告

代码在F:\VScode\VSCode\_workspace\jquery\31.(练习)对联广告.html

# 展开/收起动画

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    div{

        width: 100px;

        height: 300px;

        background: red;

        display: none;

    }

</style>

<script>

    $(function(){

        $("button").eq(0).click(function(){

            $("div").slideDown(1000,function(){

                alert("展开完毕");

            });

        });

        $("button").eq(1).click(function(){

            $("div").slideUp(1000,function(){

                alert("收起完毕");

            });

        });

        $("button").eq(2).click(function(){

            $("div").slideToggle(1000,function(){

                alert("展开/收起完毕");

            });

        });

    });

</script>

<button>展开</button>

<button>收起</button>

<button>切换</button>

<div></div>

# (练习)折叠菜单

代码在F:\VScode\VSCode\_workspace\jquery\33.(练习)折叠菜单.html

# (练习)下拉菜单

代码在F:\VScode\VSCode\_workspace\jquery\34.(练习)下拉菜单.html

# 淡入淡出动画

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    div{

        width: 300px;

        height: 300px;

        background: red;

        display: none;

    }

</style>

<script>

    $(function(){

        $("button").eq(0).click(function(){

            $("div").fadeIn(1000,function(){

                alert("淡入完毕");

            });

        });

        $("button").eq(1).click(function(){

            $("div").fadeOut(1000,function(){

                alert("淡出完毕");

            });

        });

        $("button").eq(2).click(function(){

            $("div").fadeToggle(1000,function(){

                alert("切换完毕");

            });

        });

        $("button").eq(3).click(function(){

            //0.5表示淡入到半透明

            $("div").fadeTo(1000,0.5,function(){

                alert("淡入到一定程度完毕");

            });

        });

    });

</script>

<button>淡入</button>

<button>淡出</button>

<button>切换</button>

<button>淡入到</button>

<div></div>

# (练习)弹窗广告

代码在F:\VScode\VSCode\_workspace\jquery\36.(练习)弹框广告.html

# 自定义动画

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    div{

        width: 100px;

        height: 100px;

        margin-top: 10px;

        background: red;

    }

    .two{

        background: blue;

    }

</style>

<script>

    $(function(){

        $("button").eq(0).click(function(){

            /\*

            animate()：可以传4个参数

                第一个参数为自定义动画的生成效果

                第二个参数为完成动画的时间(毫秒)

                第三个参数为动画的节奏，默认是swing(慢->快->慢)，而linear是匀速

                第四个参数为动画完毕之后的回调函数

            \*/

            $(".one").animate({

                marginLeft:500

            },5000,function(){

                alert("自定义动画执行完毕");

            });

            $(".two").animate({

                marginLeft:500

            },5000,"linear",function(){

                alert("自定义动画执行完毕");

            });

        });

        $("button").eq(1).click(function(){

            $(".one").animate({

                //宽度会不断的累加

                width:"+=200"

            },1000,function(){

                alert("自定义动画执行完毕");

            });

        });

        $("button").eq(2).click(function(){

            $(".one").animate({

                //"hide"代表隐藏

                // width:"hide"

                //"toggle"代表切换

                width:"toggle"

            },1000,function(){

                alert("自定义动画执行完毕");

            });

        });

    });

</script>

<button>操作属性</button>

<button>累加属性</button>

<button>关键字</button>

<div class="one"></div>

<div class="two"></div>

# stop()和delay()

<style>

    \*{

        margin: 0;

        padding: 0;

    }

    .one{

        width: 100px;

        height: 100px;

        background: red;

    }

    .two{

        width: 500px;

        height: 10px;

        background: blue;

    }

</style>

<script>

    $(function(){

        /\*

        $("button").eq(0).click(function(){

            $(".one").animate({

                //动画会一起执行

                width:500,

                height:500

            },1000);

        });

        //链式编程的话执行完一个再继续执行下一个

        //delay():延迟时间,参数是毫秒值

        $("button").eq(0).click(function(){

            $(".one").animate({

                width:500,

            },1000).delay(1000).animate({

                height:500,

            },1000);

        });

        \*/

        $("button").eq(0).click(function(){

            $(".one").animate({

                width:500,

            },1000);

            $(".one").animate({

                height:500,

            },1000);

            $(".one").animate({

                width:100,

            },1000);

            $(".one").animate({

                height:100,

            },1000);

        });

        $("button").eq(1).click(function(){

            //立即停止当前动画，继续执行后续动画

            // $("div").stop();

            // $("div").stop(false);

            // $("div").stop(false,false);

            //立即停止当前和后续所有动画

            // $("div").stop(true);

            // $("div").stop(true,false);

            //立即完成当前动画，继续执行后续动画

            // $("div").stop(false,true);

            //立即完成当前动画，并停止执行后续所有动画

            $("div").stop(true,true);

        });

    });

</script>

<button>开始动画</button>

<button>结束动画</button>

<div class="one"></div>

<div class="two"></div>

# (练习)图标特效

代码在F:\VScode\VSCode\_workspace\jquery\39.(练习)图标特效.html

# (练习)无限循环滚动

代码在F:\VScode\VSCode\_workspace\jquery\40.(练习)无限循环滚动.html

# 添加节点操作

<script>

    /\*

    内部插入

        append(content|fn)

        appendTo(content)

        将元素添加到指定元素内部的最后

        prepend(content|fn)

        prependTo(content)

        将元素添加到指定元素内部的最前面

    外部插入

        after(content|fn)

        insertAfter(content)

        将元素添加到指定元素外部的最近的后面

        before(content|fn)

        insertBefore(content)

        将元素添加到指定元素外部的最近的前面

    \*/

    $(function(){

        $("button").click(function(){

            //新建节点

            var li = $("<li>新增的li</li>");

            //内部添加节点

            // $("ul").append(li);

            // $("ul").prepend(li);

            // li.appendTo($("ul"));

            // li.prependTo($("ul"));

            //外部添加节点

            // $("ul").after(li);

            // $("ul").before(li);

            // li.insertAfter($("ul"));

            li.insertBefore($("ul"));

        });

    });

</script>

<button>添加节点</button>

<ul>

    <li>我是第1个li</li>

    <li>我是第2个li</li>

    <li>我是第3个li</li>

</ul>

# 删除节点操作

<script>

    /\*

    remove([expr])

    删除指定元素

    empty()

    只删除指定元素的内容和其子元素

    detach([expr])

    删除指定元素

    \*/

    $(function(){

        $("button").click(function(){

            // $("div").remove();

            // $("div").empty();

            $("div").detach();

        });

    });

</script>

<button>删除节点</button>

<ul>

    <li class="item">我是第1个li</li>

    <li>我是第2个li</li>

    <li class="item">我是第3个li</li>

    <li>我是第4个li</li>

    <li class="item">我是第5个li</li>

</ul>

<div>我是div

    <p>我是段落</p>

</div>

# 替换节点操作

<script>

    /\*

    replaceWith(content|fn)

    将匹配的元素替换为指定的元素

    replaceAll(selector)

    用指定的元素将匹配的元素替换

    \*/

    $(function(){

        $("button").click(function(){

            //新建元素

            var h6 = $("<h6>我是标题6</h6>");

            //替换元素

            // $("h1").replaceWith(h6);

            //替换元素

            h6.replaceAll("h1");

        });

    });

</script>

<button>替换节点</button>

<h1>我是标题1</h1>

<h1>我是标题1</h1>

<p>我是段落</p>

# 复制节点操作

<script>

    /\*

    clone([Even[,deepEven]]):

    如果形参为false为浅复制，浅复制只会复制元素

    如果形参为true为深复制，深复制不仅会复制元素，还会复制元素的事件

    \*/

    $(function(){

        $("button").eq(0).click(function(){

            //浅复制元素

            var li = $("li:first").clone(false);

            //将复制的元素添加到ul中

            $("ul").append(li);

        });

        $("button").eq(1).click(function(){

            //深复制元素

            var li = $("li:first").clone(true);

            //将复制的元素添加到ul中

            $("ul").append(li);

        });

        $("li").click(function(){

            alert($(this).html());

        });

    });

</script>

<button>浅复制节点</button>

<button>深复制节点</button>

<ul>

    <li>我是第1个li</li>

    <li>我是第2个li</li>

    <li>我是第3个li</li>

    <li>我是第4个li</li>

    <li>我是第5个li</li>

</ul>

# 实时监听textarea里的内容

<script>

    $(function(){

        //监听值的变化。第一种方式

   // $("body").delegate("#textarea","propertychange input",function(){

        //     console.log($(this).val());

        //     //判断是否输入了内容

        //     if($(this).val().length > 0){

        //         //禁用发布按钮

        //         $("button").prop("disabled",false);

        //     }else{

        //         //按钮可用

        //         $("button").prop("disabled",true);

        //     }

        // });

        //监听值的变化。第二种方式

        $(document).on("propertychange input","#textarea",function(){

            console.log($(this).val());

            //判断是否输入了内容

            if($(this).val().length > 0){

                //禁用发布按钮

                $("button").prop("disabled",false);

            }else{

                //按钮可用

                $("button").prop("disabled",true);

            }

        });

        $("button").click(function(){

            alert($("#textarea").val());

        });

    });

</script>

<textarea id="textarea"></textarea>

<button disabled>发布</button>