一、使用jQuery

1. 使用jQuery三个步骤：
    1. 引入jQuery文件
    2. 入口函数
    3. 功能实现
2. 关于jQuery的入口函数：

```
//第一种写法
$(document).ready(function() {

});
//第二种写法
$(function() {

});
```

jQuery入口函数与js入口函数的对比：

1. JavaScript的入口函数要等到页面中所有资源（包括图片、文件）加载完成才开始执行。
2. jQuery的入口函数只会等待文档树加载完成就开始执行，并不会等待图片、文件的加载。

3. $代表什么
    1. 如果在使用jQuery时，报了这个错误:$ is not defined,就说明没有引入jQuery文件.
    2. jQuery文件结构，其实是一个自执行函数，$其实和jQuery是等价的,是一个函数
    3. $是一个函数，参数传递不同,效果也不一样
        1. 如果参数传递的是一个匿名函数--入口函数
        2. 如果参数传递的是一个字符串--选择器
        3. 如果参数是一个dom对象,那他就会把dom对象转换成jQuery对象
4. jquery对象和dom对象
    1. DOM对象：使用JavaScript中的方法获取页面中的元素返回的对象就是dom对象。
    2. jQuery对象：jquery对象就是使用jquery的方法获取页面中的元素返回的对象就是jQuery对象。
    3. jQuery对象其实就是DOM对象的包装集，包装了DOM对象的集合
    4. DOM对象与jQuery对象的方法不能混用。
    5. DOM对象转换成jQuery对象：

```
var $obj = (domObj);
```

    6. jQuery对象转换成DOM对象

```
var $li = $("li");
//第一种方法（推荐使用）
$li[0]
```

二、jQuery的常用选择器

- JavaScript获取dom元素时代码太繁琐，所以出现了jQuery选择器
- jQuery的选择器可以帮我们快速的定位到一个或多个dom元素
- 注意：jQuery选择器返回的是jQuery对象。

1. 基本选择器

名称	用法	描述
ID选择器	$(“#id”);	获取指定ID的元素
类选择器	$(“.class”);	获取同一类class的元素
标签选择器	$(“div”);	获取同一类标签的所有元素
并集选择器	$(“div,p,li”);	使用逗号分隔，只要符合条件之一就可。
交集选择器	$(“div.redClass”);	获取class为redClass的div元素

2. 层次选择器
3. 过滤选择器

名称	用法	描述
:eq（index）	$(“li:eq(2)”).css(“color”, ”red”);	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”).css(“color”, ”red”);	获取到的li元素中，选择索引号为奇数的元素
:even	$(“li:even”).css(“color”, ”red”);	获取到的li元素中，选择索引号为偶数的元素

4. 筛选选择器(方法)

名称	用法	描述
children(selector)	$(“ul”).children(“li”)	相当于$(“ul>li”)，子类选择器
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。
parent()	$(“#first”).parent();	查找父亲
eq(index)	$(“li”).eq(2);	相当于$(“li:eq(2)”),index从0开始
next()	$(“li”).next()	找下一个兄弟
prev()	$(“li”).prev()	找上一次兄弟

5. 属性选择器

- 选择type为text的dom元素

```
$(":text").css("border-color","red");
```

- 选择name为pwd的dom元素

```
$("[name=pwd]").css("border-color","blue");
```

- ：更侧重于 标签的“种类”，[] 更侧重于标签的“属性”

三、内容操作

1.内容操作

1. html()与text()：html()相当于innerHTML text()相当于innerText

   区别：html方法会识别html标签，text方法会那内容直接当成字符串，并不会识别html标签

```
//设置内容
$('div').html('<span>这是一段内容</span>');
//获取内容
$('div').html()

//设置内容
$('div').text('<span>这是一段内容</span>');
//获取内容
$('div').text()
```

2.属性操作

1. val()：用于设置和获取表单元素的value属性值，例如input、textarea的值

```
//设置值
$("#name").val('张三');
//获取值
$("#name").val();
```

2. attr()：其他属性设置

```
/*1.获取属性*/
$('li').attr('name');
/*2.设置属性*/
$('li').attr('name','tom');
/*3.设置多个属性*/
$('li').attr({
'name':'tom',
'age':'18'
});
/*4.删除属性*/
$('li').removeAttr('name');
```

3.样式操作

3.1 样式设置

```
/*1.设置一个样式*/
//两个参数  设置的样式属性,具体样式
$('li').css('color','red');
//传入对象（设置的样式属性:具体样式）
$('li').css({'color':'red'});
/*2.设置多个样式*/
$('li').css({
    'color':'green',
    'font-size':'20px'
});
```

3.2 类名设置

```
/*1.添加一个类*/
$('li').addClass('now');
/*2.删除一个类*/
$('li').removeClass('now');
/*3.切换一个类  有就删除没有就添加*/
$('li').toggleClass('now');
/*4.匹配一个类  判断是否包含某个类  如果包含返回true否知返回false*/
$('li').hasClass('now');
```

4.节点操作

4.1 创建节点

```
/*创建节点*/
var $a = $('<a href="http://www.baidu.com" target="_blank">百度1</a>');
```

4.2 克隆节点

```
/*如果想克隆事件  false  true克隆事件*/
var $cloneP = $('p').clone(true);
```

4.3 添加节点&移动节点

```
/*追加自身的最后面  传对象和html格式代码*/
$('#box').append('<a href="http://www.baidu.com" target="_blank"><b>百度3</b></a>');
$('#box').append($('a'));
/*追加到目标元素最后面  传目标元素的选择器或者对象*/
$('<a href="http://www.baidu.com" target="_blank"><b>百度3</b>
</a>').appendTo($('#box'));
$('a').appendTo('#box');

prepend();
prependTo();
after();
before();
```

4.4 删除节点&清空节点

```
/*1.清空box里面的元素*/
/* 清理门户 */
$('#box').empty();
/*2.删除某个元素*/
/* 自杀 */
$('#box').remove();
```

四、jQuery事件机制

- JavaScript中已经学习过了事件，但是jQuery对JavaScript事件进行了封装，增加并扩展了事件处理机制 jQuery不仅提供了更加优雅的事件处理语法，而且极大的增强了事件的处理能力。

• jQuery事件发展历程(了解)

简单事件绑定>bind事件绑定>delegate事件绑定>on事件绑定(推荐)

1. 简单事件注册

```
click(handler)          //单击事件
mouseenter(handler)         //鼠标进入事件
mouseleave(handler)         //鼠标离开事件
```

缺点：不能同时注册多个事件

2. bind方式注册事件

```
//第一个参数：事件类型
//第二个参数：事件处理程序
$("p").bind("mouseover mouseout", function(){
    //事件响应方法
});
//可以这样
$("li").bind({
    mouseover() {
        $(this).css("background-color","red")
    },
```

```
    mouseout() {
        $(this).css("background-color","blue")
    }
})
```

缺点：

1. 这里用了隐式迭代的方法，如果匹配到的元素特别多的时候，比如如果我在url里放了50个li元素，就得执行绑定50次。对于大量元素来说，影响到了性能。如果是id选择器，因为id唯一，用bind()方法就很快捷了
2. 对于尚未存在的元素，无法绑定

3. delegate注册委托事件

```
// 第一个参数：selector，要绑定事件的元素
// 第二个参数：事件类型
// 第三个参数：事件处理函数
$(document).ready(function () {
    //事件委托delegate写法事件并没有加到li上面，而是加到了ul的上面
    // 是$(this)触发的时候指向了li；利用了冒泡原理
    /*$("ul").delegate("li","click", function () {
        $(this).css("background-color", "red");
    })*/
    $("ul").delegate("li",{
        mouseover() {
            $(this).css("background-color","red")
        },
        mouseout() {
            $(this).css("background-color","blue")
        }
    })
    //ul里面新创建的li标签也可以使用li事件
    $("#btn").click(function () {
        $li = $("<li>新添加的li</li>")
        $("ul").append($li)
    })
});
```

缺点：这种方式采用了事件委托的概念。不是直接为li元素绑定事件，而是为其父元素（或祖先元素也可）绑定事件，当在ul内任意元素上点击时，事件会一层层从event target向上冒泡，直至到达你为其绑定事件的元素，如此例中的ul元素。冒泡的过程中，如果事件的currentTarget与选择器匹配时，就会执行代码。

这样就解决了用bind()方法的上面两个问题，不用再一个个地去为li元素绑定事件，也可以为动态添加进来的li元素绑定。甚至，如果你将事件绑定到document上，都不用等document准备好就可执行绑定。

这样，绑定是容易了，但是调用的时候也可能出现问题。如果事件目标在DOM树中很深的位置，这样一层层冒泡上来查找与选择器匹配的元素，又影响到性能了。

4. on注册事件

jQuery1.7之后，jQuery用on统一了所有事件的处理方法强烈建议使用。

on注册简单事件

```
// 表示给$(selector)绑定事件，并且由自己触发，不支持动态绑定。
$("#btn").on("click",function () {
    $li = $("<li>新添加的li</li>")
    $("ul").append($li)
})
```

on注册委托事件

```
// 表示给$(selector)绑定代理事件，当必须是它的内部元素span才能触发这个事件，支持动态绑定
$(selector).on( "click",'span', function() {});
$("ul").on({
    mouseover: function () {
        $(this).css("background-color","red")
    },
    mouseout:function() {
        $(this).css("background-color","blue")
    }
},"li")
```

5. 事件解绑

unbind方式（不用）

```
$(selector).unbind(); //解绑所有的事件
$(selector).unbind("click"); //解绑指定的事件
```

undelegate方式（不用）

```
$( selector ).undelegate(); //解绑所有的delegate事件
$( selector).undelegate( 'click' ); //解绑所有的click事件
```

off方式（推荐）

```
// 解绑匹配元素的所有事件
$(selector).off();
// 解绑匹配元素的所有click事件
$(selector).off("click");
```

6. 触发事件

```
$(selector).click(); //触发 click事件
$(selector).trigger("click");
```

五、链式编程

- 链式编程是为了节省代码量，看起来更优雅。
- 通常情况下，只有设置操作才能把链式编程延续下去。因为获取操作的时候，会返回获取到的相应的值，无法返回 jQuery对象。

```
$("#btn").on("click", function () {
    $("[name=uname]").css("background-color", "yellow").val("张
三").next().css("border-color","red").val("123456")
})
```

六、ajax

- ajax主要的功能是实现了浏览器端 异步 访问服务器：通过浏览器的XMLHttpRequest对象发出小部分数据，与服务端进行交互，服务端返回小部分数据，然后更新客户端的部分页面
- json是Ajax发送小部分数据的一种轻量级数据格式，可以简单易懂的给服务器或者浏览器交互数据，包括json对象，json数组对象
- jquery 库中已经封装了ajax请求的方法。
- 参数：
  - url：要请求的资源路径，字符串表示
  - data：请求参数
  - type："POST" 或 "GET"，请求方式
  - timeout：请求超时时间，单位为毫秒，数值表示
  - cache：是否缓存请求结果，bool表示
  - contentType：内容类型，默认为"application/x-www-form-urlencoded"
  - dataType：服务器响应的数据类型，默认是text
  - success：请求成功后，服务器回调的函数
  - error：请求失败后，服务器回调的函数
  - complete：请求完成后调用的函数，无论请求是成功还是失败，都会调用该函数；如果设置了success与error函数，则该函数在它们之后被调用
  - async：是否异步处理，bool表示，默认为true；设置该值为false后，JS不会向下执行，而是原地等待服务器返回数据，并完成相应的回调函数后，再向下执行
- 案例：判断用户名是否被占用

```
用户名：<input type="text" name="uname">
<span style="color: red" id="tips"></span><br>
密码：<input type="password" name="pwd"> <br>
<button id="btn">登录</button>
<script src="jquery-3.1.1.min.js"></script>
<script>
    //使用jquery实现
    $(document).ready(function () {
        $("#btn").click(function () {
            $.ajax({
                url: "http://localhost:8080/CheckNameServlet",
                data: {uname: $("[name=uname]").val()},
                type: "POST",
                success: function (data) {
                    $("#tips").text(data);
                },
                error: function () {
                    alert("执行失败")
                }
            })
        })
```

```
    });
</script>
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    String name = request.getParameter("uname");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter pw = response.getWriter();
    if ("张三".equals(name)) {
        pw.print("该名称已被占用");
    } else {
        pw.print("名称可用");
    }
}
```

服务端返回小部分数据，然后更新客户端的部分页面

服务端返回小部分数据，然后更新客户端的部分页面。

- json是Ajax发送小部分数据的一种轻量级数据格式，可以简单易懂的给服务器或者浏览器交互数据，包括json对象，json数组对象

服务端返回小部分数据，然后更新客户端的部分页面。

- json是Ajax发送小部分数据的一种轻量级数据格式，可以简单易懂的给服务器或者浏览器交互数据，包括json对象，json数组对象

服务端返回小部分数据，然后更新客户端的部分页面。

- json是Ajax发送小部分数据的一种轻量级数据格式，可以简单易懂的给服务器或者浏览器交互数据，包括json对象，json数组对象