JavaScript 学习笔记

核心内容概述

1.JavaScript 加强，涉及到 ECMAScript 语法、BOM 对象、DOM 对象以及事件。
2.Ajax 传统编程。
3.jQuery 框架，九种选择器为核心学习内容
4.JQuery UI 插件
5.jQuery Ajax 编程
6.jQuery 第三方插件
7.反向 Ajax 编程(彗星)

一、JavaScript 基础加强

JavaScript 是在浏览器内容运行，无需编译、解释执行动态脚本语言，是一种弱类型语言，所有变量使用 var 定义。

JavaScript 的 3 个组成部分分别为：核心(ECMAScript)、文档对象模型(DOM)、浏览器对象模型(BOM)

1.ECMAScript 核心语法

①：代码编写位置

分为内部JS和外部JS【使用src进行引入】

```
<meta http-equiv="Content-Type" content="text/html; charset=gbk">
<title>JavaScript程序编写</title>
<!-- 内部JS -->
<script type="text/javascript">
    // 编写JavaScript代码
    alert(1);
</script>
<!-- 外部JS-->
<script type="text/javascript" src="1.js"></script>
```

②：学习顺序

JavaScript 依次从**变量**(标示符、关键字)，运算符，程序结构(if while for)，以及**函数**来进行学习。

(1)所有的变量使用 var 来定义，是弱类型变量，不代表没有类型，变量本身还是有类型的。【var a=10，var b=1.5；他们分别为整数以及浮点数类型】

(2)每行结尾分号可有可无，建议编写。

(3)注释和 Java 类似，支持单行注释(//)和多行注释(/* */)

③：数据类型

JavaScript 分为原始数据类型和引用数据类型，分别存储于栈和堆中。

带格式的：字体：(默认) 微软雅黑，(中文) 微软雅黑，10 磅，加粗，字体颜色：蓝色

带格式的：字体颜色：蓝色

批注 [ThinkPad1]：注意编码的设置
1.如果是内部 JS，html 编码要与浏览器编码一致；
2.如果是外部 JS，确保 JS 编码集与 html 编码集一致。

批注 [ThinkPad2]：这里不建议使用 language 属性！

带格式的：无，行距：单倍行距

批注 [ThinkPad3]：这里重点讲解**变量以及函数**，运算符和程序结构与 Java 类似。

原始数据类型：number、string、boolean、null 和 undefined
引用数据类型：存在很多种，每种都是 object 对象
可以使用 typeof 查看数据类型，使用 instanceof 判断变量数据类型

```
<script type="text/javascript">
    // 定义所有变量都用var，但是变量本身具有类型
    var a = 10;    // 整数
    var b = 1.5;   // 浮点数
    var c = true;  // 布尔
    var d = "abc"; // 字符串 基本数据类型
    var e = 'abc'; // 字符串

    // 通过typeof查看数据类型
    alert(typeof d);
    // 通过instanceof判断变量数据类型
    alert(d instanceof Object);//falae
    alert(a instanceof Object);//falae

    var s = new String("abc"); // 对象类型
    alert(s instanceof Object);
</script>
```

批注 [ThinkPad4]：显示数据类型为 string

批注 [ThinkPad5]：结果为 false，变量 d 为 string 类型，属于原始数据类型，不是引用数据类型(对象)

批注 [ThinkPad6]：这里的 s 是对象类型 object，前面的 d 是基本数据类型

带格式的：字体：（中文）+中文正文（宋体）

带格式的：缩进：首行缩进： 0 厘米

④：**null 和 undefined 的区分**
null：对象不存在；
undefined：对象存在，访问属性或者方法不存在(对象未初始化)

2.ECMAScript 对象

批注 [ThinkPad7]：核心对象的常见方法仅作为学习的思路，不会没啥关系，查手册即可。

ECMAScript 常用的有 7 个对象，依次为 String、Number、Boolean、Math、Date、Array 以及 Regxp。

批注 [ThinkPad8]：这里提到的为常用的，还有很多其他对象，可查看手册。

①：**String 类型常用属性方法**
建议查看手册，这里需要注意的为length属性以及match方法
charAt()、concat()、indexOf()、lastIndexOf()、match()、replace()、split()、substr()、substring()、toLowerCase()、toUpperCase()
Java中提供matches方法 例如："1234".matches("\\d+") ---- 返回true
JavaScript 与 matches方法等价的那个方法，是 RegExp 对象提供test方法
例如：/^\d+$/.test("1234") --- 返回true
/^\d+$/ 等价于 new RegExp("^\\d+$")

批注 [ThinkPad9]：^和&分别为开始和结束符号

"1234".match("^\\d+$") 返回是匹配正则表达式内容，而不是布尔值，等价于/^\d+$/.exec("1234")

②：**Math 常用属性和方法**
PI 属性
round(x) 把数四舍五入为最接近的整数

random() 返回 0 ~ 1 之间的随机数

pow(x,y) 次幂

sqrt(x) 平方根

③：Date 常用属性和方法

toLocaleString() 返回当地本地化日期格式 2012年12月12日 11:07:52

getTime() 返回从1970年1月1日到目前为止 毫秒值

```
<script type="text/javascript">
    var s1 = "abc";                 // s1是基本数据类型
    var s2 = new String("abc") ; // s2是对象类型
//  alert(s1 == s2); //

//  alert("98"==98);// true
//  alert("true"==true); // false
//  alert(1==true); // true

    var d = 010;// 八进制
    var d2 = 0x10; // 十六进制

    // match方法 类似 Java中 matches，有区别
//  alert(/^\d+$/.test("1234abc")); // 等价于 java中matches
//  alert("1234".match("^\\d+$")); // math方法返回的是匹配正则表达式内容，而不是布尔值
//  alert(/^\d+$/.exec("1234abc1234"));// 返回匹配的内容

    // Date使用
    var date = new Date(); //当前日期
    alert(date.toLocaleString());// 返回当地国际化日期格式
    var dateStr = date.getFullYear()+"-"+date.getMonth()
+"-"+date.getDate()+" "+date.getHours()+":"+date.getMinutes()
+":"+date.getSeconds();
    alert(dateStr);
</script>
```

批注 [ThinkPad10]: ==比较，自动进行类型转换，如果要使它不进行自动转型，我们可以使用===

批注 [ThinkPad11]: 自定义日期格式！

带格式的：字体：（中文）+中文正文（宋体）

带格式的：缩进：首行缩进： 0 厘米

④：Array 常用属性方法

push() 加入元素到数组

pop() 从数组移除最后一个元素

reverse()反转

join() 连接数组元素 通过特定内容 返回字符串

sort() 排序

slice() 截取数组中指定元素 从start到end

```
<script type="text/javascript">
    // 定义数组 使用Array对象
    // 方式一
    var arr1 = [1,2,3];
    // 数组的遍历 通过长度和下标
    for(var i=0;i< arr1.length ; i++){
//      alert(arr1[i]);
    }
    // 方式二
    var arr2 = new Array(3);// 定义长度为3的数组
    arr2[0] = "aa";
    arr2[1] = "bb";
    arr2[2] = "cc"
    arr2["100"] = "dd";
//  alert(arr2.length);
//  alert(arr2[4]);
    // 方式三
    var arr3 = new Array(1,2,3);// 数组三个元素 1, 2 ,3
//  alert(arr3.join("-")); // 1-2-3
    alert(arr3.slice(1,3)); // 从1下标，截取到3下标，1下标包含，3下标不包含
</script>
```

批注 [ThinkPad12]: JS 中，{}代表对象，[]代表数组

批注 [ThinkPad13]: JS 中不存在越界问题

批注 [ThinkPad14]: JS 中，length 为最大角标+1，此结果为 101.

批注 [ThinkPad15]: 结果为 undefined，对应的引用存在，只是值不存在，所以是 undefined，而不是 null!

带格式的: 行距：最小值 1.1 磅，调整中文与西文文字的间距，调整中文与数字的间距

带格式的: 字体：（中文）+中文正文（宋体）

带格式的: 缩进：首行缩进： 0 厘米

3.ECMAScript 核心语法——函数

①：函数定义的三种方式

带格式的: 字体：（默认）微软雅黑，（中文）微软雅黑，10 磅，加粗，字体颜色：蓝色

注意：第二种方式使用越来越多，第三种不常用，第一种常用

带格式的: 字体：（中文）+中文正文（宋体）

```
<script type="text/javascript">
    // 方式一
    function add(a,b){ // 没有返回值，形参不需要声明类型
        return a+b; // 可以返回
    }
//  alert(add(1,2));

    // 方式二  function 匿名函数， sub成为函数名称
    var sub = function(a,b){
        return a-b;
    }
//  alert(sub(10,8));

    // 方式三 使用Function对象 定义函数
    // 语法 new Funtion(arg1,arg2 ... , body)
```

```
    var mul = new Function("a","b","return a*b;"); // 不常用
//  alert(mul(10,20));

    // 所有函数 都是Function实例
    alert(mul instanceof Function);// true
</script>
```

②：JavaScript 全局函数(内置函数)

一组与编码解码相关的函数

encodeURI()&decodeURI()

encodeURIComponent()&decodeURIComponent()

escape()&unescape()

此块具体内容请参照 W3C 文档查看。

4.ECMAScript 核心——JavaScript 面向对象编程

Java 是面向对象，写 Java 程序，写类和对象。JavaScript 是基于对象，写 Js，不用创建类，使用 Js 内部已经定义好的对象。

①：定义 JavaScript 对象的两种方式

方式一：使用已经存在的对象，通过关键字进行创建

```
var s = new String("aaaa");
var o = new Object();
var date = new Date();
//  alert(date instanceof Object);// true

    // JS对象 类似一个map结构
    var arr = new Array(3);
    arr[0] = 100;// 使用数组下标 为数组元素赋值
    arr['aaa'] = 1000; // 定义对象属性
//  alert(arr['aaa']);
    arr.showInfo = function(){// 定义对象方法
        alert(arr.join(","));
    };
//  arr.showInfo();  //100, ,
```

Js 其实就是一个类似 map 结构，key 为属性名和方法名，value 为属性值和方法定义

方式二：通过{}创建

```
var obj = {
    name : '张三',
    age : 20,
    getName : function(){
```

带格式的：字体：（中文）+中文正文（宋体）

带格式的：字体：（中文）+中文正文（宋体）

带格式的：字体：（默认）微软雅黑，（中文）微软雅黑，10 磅，加粗，字体颜色：蓝色

带格式的：字体：（中文）+中文正文（宋体）

批注 [ThinkPad16]：之后类似此种格式均为运行结果。

```
        // 访问对象属性 通过关键字 this
            return this.name;
        }
    };
    // 访问对象 属性 [] 和 .
// alert(obj.name);
// alert(obj["age"]);
    alert(obj.getName());

    // 添加一个方法到 obj对象
    obj.getAge = function(){
        return this.age;
    }
    alert(obj.getAge());
```

JavaScript 中的对象是通过 new function 创建的，在 Js 中 function 等同于一个类结构

```
    // 定义类 结构
    var Product = function(name,price){
        this.name = name; // 保存name的值 到对象属性中
        this.price = price;
    }
    // 基于类结构创建对象，使用new 关键字
    var p1 = new Product("冰箱",1000);
    var p2 = new Product("洗衣机",1500);

// alert(p1.name);
// alert(p1.price);
```

function 本身代表一个函数，JavaScript 对象通过 new function 来获得的，理解 function 就是对象构造函数

②：Object 和 function 的关系

JavaScript 中所有引用类型都是对象 Object 实例 ------- Function instanceOf Object //true

JavaScript 中所有对象都是通过 new Function 实例(function) 获得 ------ Object instance Function //true

JavaScript 所有对象构造函数都是 function 实例；JavaScript 所有对象都是 object 实例，function 也是 object 实例。

使用 JavaScript 的传递性进行推论！

批注 [ThinkPad17]: 使用[]访问对象属性要使用引号

批注 [ThinkPad18]: 注意定义类结构与定义对象的区别，不要混淆！

带格式的: 段落间距段前：0.5 行，段后：0.5 行

A：function 是用来定义一个函数，所有函数实例都是 Function 对象

B：JavaScript 中，所有对象都是通过 new function 得到的

```
Var Object = function(){...}
Var String = function(){...}
Var Array = function(){...}
Var Date = function(){...}
```

结论：所有对象构造器都是 Function 实例

```
alert(String instanceOf Function)   //true
alert(Object instanceOf Function)   //true
```

C：创建一个对象，需要使用 new function

```
Var s = new String()
Var o = new Object()
Var arr = new Array()
Var date = new Date()
```

结论：JavaScript 中，一切对象都是 object 实例

```
alert(s instanceOf Object)    //true
alert(Function instanceOf Object)    //true
```

```
var f = new Function(); // 实例化Function对象
var o = new Object();   // 实例化Object对象
alert(f instanceof Function); // true
alert(f instanceof Object); // true
alert(o instanceof Function); // false
alert(o instanceof Object); // true
```

③：**function 原型属性**

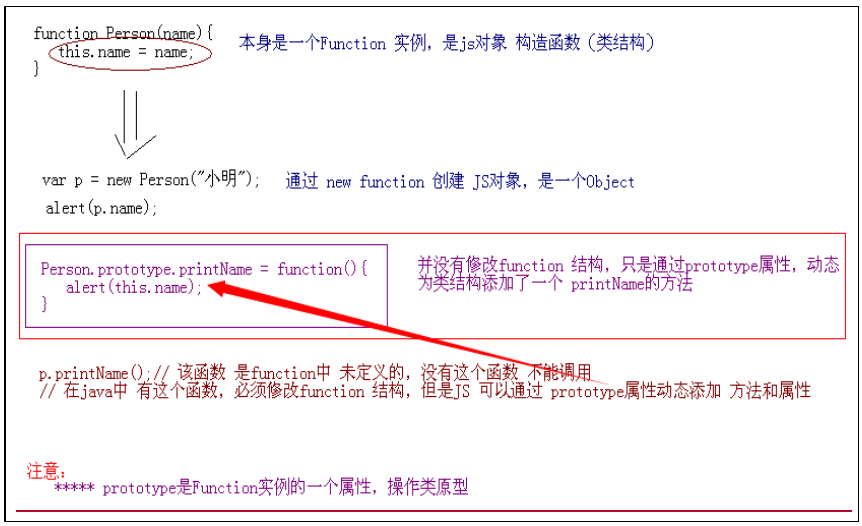
JavaScript 所有对象都由 function 构造函数得来的 ，通过修改 function 构造函数 prototype 属性，动态修改对象属性和方法。

批注 [ThinkPad19]：这的 String()就是具体的 function

批注 [ThinkPad20]：在 Js 中，小写的 function(f) 代表的是一个类的结构

批注 [ThinkPad21]：o 代表的是一个对象

批注 [ThinkPad22]：对象本身不是 Function 实例，他是 Object 实例，Object 才是 Function 的实例。



带格式的：字体：（中文）+中文正文（宋体）

④：**继承**

A：使用原型链完成 JavaScript 单继承

```
    var A = function(){
        this.name = 'xxx';
    }
    var B = function(){
        this.age = 20;
    }
    // 方式一 可以通过 prototype 原型完成单继承 B 的原型指向 A
    B.prototype = new A(); // 从A实例中，继承所有属性

    var b = new B();
    alert(b.name);

    //  练习：通过prototype为String类添加一个trim方法
    String.prototype.trim = function(){
        return this.replace(/(^\s*)(\s*$)/g, "");
    }
```

B：对象冒充完成多继承

```
    var C = function(){
        this.info = 'c';
    }
    var D = function(){
        this.msg = 'd';
```

```
    }
    var E = function(){
        // 同时继承C和D
        this.methodC = C;
        this.methodC();
        delete this.methodC;

        this.methodD = D;
        this.methodD();
        delete this.methodD;

        this.desc = 'e';
    }

    var e = new E();
//  alert(e.info);
//  alert(e.msg);
//  alert(e.desc);
```

⑤：**动态方法调用**

可以改变 this 的指向，可以完成对象多继承

```
    // 定义函数
    function printInfo(){
        alert(this.name);
    }

    // 属性name 值 张三
    var o = {name: '张三'};
//  o.printInfo();// 函数不属于对象o
// JS提供动态方法调用两个方法，允许一个对象调用不是属于它自己的方法(call apply)
//  printInfo.call(o);
//  printInfo.apply(o);

    function add(a,b){
        this.sum = a+b;
    }
    // call传 多个参数
//  add.call(o,8,10);
    // apply 传递参数数组
    add.apply(o,new Array(8,10));
//  alert(o.sum);

    // 动态方法调用 ，实现多重继承，原理就是对象冒充
```

批注 [ThinkPad23]: 在 E 中定义方法，引用 C 方法。

批注 [ThinkPad24]: 执行方法，实际是在执行 C，--->this.info = ‘c’；其中 this 是 E，info 被 E 继承了。

```
var A = function(){
    this.info = 'a';
}
var B = function(){
    // 动态方法调用继承
    A.call(this);
}
var b = new B();
alert(b.info);
```

批注 [ThinkPad25]: ①：改变了 this 指向；
②：此语句等价于====>
this.methodA = A;
this.methodA();
Delete this.methodA;

二、JavaScript 浏览器对象 BOM

DOM Window 代表窗体
DOM History 历史记录
DOM Location 浏览器导航
DOM Navigator 浏览器信息 不讲
DOM Screen 屏幕 不讲

重点：window、history、location ， 最重要的是 window 对象

1.window 对象

Window 对象表示浏览器中打开的窗口，如果文档包含框架（frame 或 iframe 标签），浏览器会为 HTML 文档创建一个 window 对象，并为每个框架创建一个额外的 window 对象

window.frames 返回窗口中所有命名的框架
parent 是父窗口（如果窗口是顶级窗口，那么 parent==self==top）
top 是最顶级父窗口（有的窗口中套了好几层 frameset 或者 iframe）
self 是当前窗口（等价 window）
opener 是用 open 方法打开当前窗口的那个窗口

批注 [ThinkPad26]: 常用必备技能，必须掌握！

①：**父子窗体之间的通讯**

在页面内嵌入一个 iframe，在 iframe 中提供一个输入项，输入后，在 iframe 外面窗口中显示内容

显示结果如上图所示，实现思路如下：

子窗体：2.html

```
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=gbk">
    <title>Untitled Document</title>
    <script type="text/javascript">
        function showOutter(){
            // 获得输入内容
            var content = document.getElementById("content").value;
            // 将输入的内容显示到主窗体info 中
    window.parent.document.getElementById("info").innerHTML = content;
        }
    </script>
</head>
<body>
    <h1>子窗体</h1>
    <input type="text" id="content" />
    <input type="button" value="显示到主窗体" onclick="showOutter();"/>
</body>
```

带格式的：缩进：首行缩进： 0 厘米

主窗体：1.html

```
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=gbk">
    <title>父子窗体通信</title>
```

```
    <script type="text/javascript">
        function showContent(){
            // 用主窗体读取子窗体内容
            var content = window.frames[0].document.getElementById("content").value;
            alert(content);
        }
    </script>
</head>
<body>
    <h1>主窗体</h1>
    <div id="info"></div>
    <!-- 在主窗体中获得子窗体内容 -->
    <input type="button" value="获取子窗体输入内容" onclick="showContent();" />
    <iframe src="2.html"></iframe>
</body>
```

批注 [ThinkPad27]: 子窗体只有 1 个，这里角标为 0。

带格式的: 字体: (中文) +中文正文 (宋体)

②：window 的 open close

带格式表格

```
<head>
    <title>打开关闭窗体</title>
    <meta http-equiv="content-type" content="text/html; charset=gbk">
    <script type="text/javascript">
        //用一个变量记录打开的网页
        var openNew;
        function openWindow(){
            openNew = window.open("http://www.itcast.cn");
        }
```

```
        //关闭的时候需要注意关闭的是打开的网页，而不是本身
        function closeWindow(){
            openNew.close();
        }
    </script>
  </head>
  <body>
    <input type="button" value="打开传智播客网页" onclick="openWindow()">
    <input type="button" value="关闭传智播客网页" onclick="closeWindow()">
  </body>
```

③：window 弹出对话框相关的 3 个方法

alert()警告框　confirm()确认框　prompt()输入框

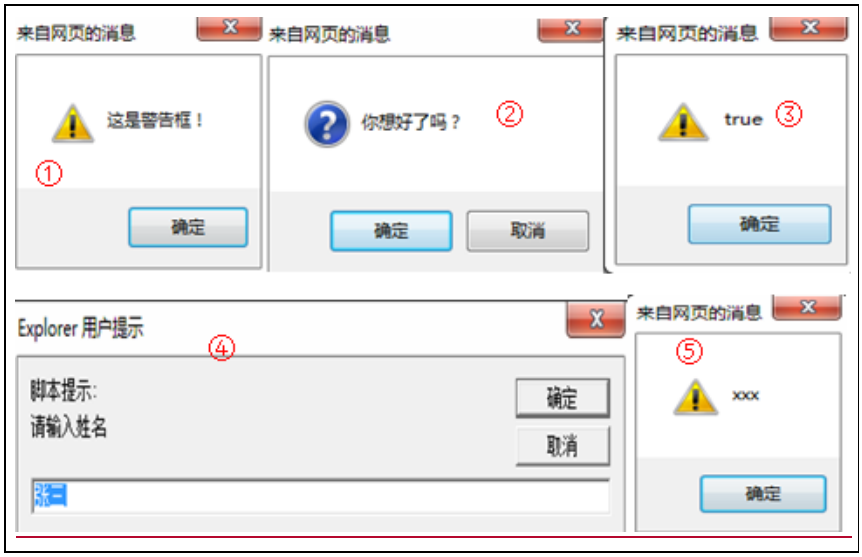
```
<script type="text/javascript">
    alert("这是警告框！")
    var con = confirm("你想好了吗？");
    alert(con);
    var msg = prompt("请输入姓名","张三");
    alert(msg);
</script>
```

批注 [ThinkPad28]：此处 2 个参数分别为**提示信息和默认信息**，均为 String 类型

显示效果截图如下：

带格式的：段落间距段前：0.5 行，段后：0.5 行

带格式的：字体：（中文）+中文正文（宋体）



④：定时操作 setInterval & setTimeout

带格式的：字体：（默认）微软雅黑，（中文）微软雅黑，10 磅，加粗，字体颜色：蓝色

带格式的：字体：10 磅，加粗，字体颜色：蓝色

setInterval：定时任务会重复执行

setTimeout：定时任务只执行一次

在页面动态显示当前时间

```
<script type="text/javascript">
    window.onload = function(){
    var date = new Date();
        document.getElementById("time1").innerHTML =date.toLocaleString();
        document.getElementById("time2").innerHTML =date.toLocaleString();
        setInterval("show1();",1000); //间隔1秒后重复执行
        setTimeout("show2();",1000); //1秒后执行，执行1次
    }
    function show1(){
        var date = new Date();
        document.getElementById("time1").innerHTML =date.toLocaleString();
    }
    function show2(){
        var date = new Date();
        document.getElementById("time2").innerHTML =date.toLocaleString();
        setTimeout("show2();",1000);//不终止
    }
</script>
<body>
   <div id="time1"></div>
   <div id="time2"></div>
</body>
```

2.history 对象

代表历史记录，常用来制作页面中返回按钮

```
<input type="button" value="返回" onclick="history.back();" />
<input type="button" value="返回" onclick="history.go(-1);" />
```

3.Location 对象

代表浏览器导航 在js函数中发起href链接效果

location.href='跳转后url' ；等价于

三、JavaScript 文档对象模型 DOM

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=gbk">
        <title>History和Location使用</title>
    </head>
    <body>
        <input type="button" value="返回" onclick="history.back();" />
```

带格式的：字体：五号

带格式的：不调整西文与中文之间的空格，不调整中文和数字之间的空格

带格式的：字体颜色：青色

带格式的：字体：（中文）+中文正文（宋体）

批注 [ThinkPad29]：空格即为 1 个文本节点。

</body>
</html>

DOM 解析模型，将文档加载到 内存，形成一个树形结构 <html> 就是根节点，每个标签会成为一个元素节点、标签的属性成为属性节点，标签内部的文本内容成为文本节点

注意：属性节点，它不属于 DOM 树形结构，不属于任何节点父节点，也不属于任何节点的子节点 ，属性节点依附于元素节点上 一种附加节点

【上面代码 产生 6 个元素节点，5 个属性节点，9 个文本节点】

HTML 本身也是 XML，所有可以使用 XML DOM API 规范

DOM Element

DOM Attr

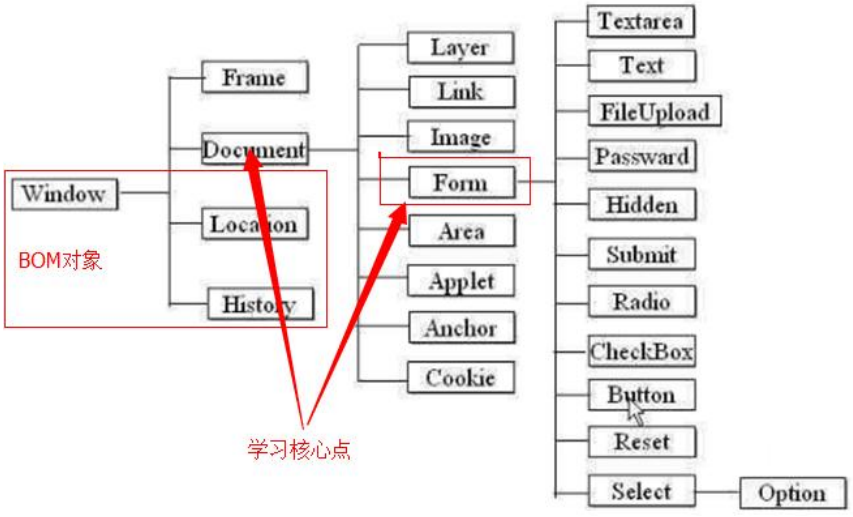
DOM Text

DOM Document

HTML DOM 是对 XML DOM 的扩展， HTML DOM 比 XML DOM 开发 JS 来说更加简单方便！

HTML DOM 最优秀的地方是，操作 form 对象和 table 数据

1.BOM 和 HTML DOM 关系图



学习 DOM 编程，从 Document 对象开始，document 代表当前 HTML 网页文档对象，是 window 对象一个属性，可以直接使用 ，所有 HTML DOM 对象都是 Document 子对象

2.DOM 编程开发

window.document 代表整个 HTML 文档

①：**通过 document 获得 Node 节点对象**

document.forms 获得页面中所有 form 元素集合

document.body 访问页面中<body> 元素

document.cookie 用 JS 操作网页 cookie 信息

批注 [ThinkPad30]: Node 是整个开发的核心。**这里简单的总结一下：**

①：通过 BOM 对象 window 的 document 属性进行全文检索。

②：检索结果不是 Node 就是 NodeList

③：如果是 NodeList 就进行遍历，如果是 Node 操作文本或者属性了。

全局检索提供了三个重要的方法：

document.getElementById():通过 id 属性检索，获得 Node 节点（Element 元素）

document.getElementsByName 通过 name 属性检索 ，获得 NodeList

document.getElementsByTagName 通过标签元素名称 获得 NodeList

其中 NodeList 可以作为数组进行操作

Demo:在每一个 h1 标签后追加 itcast

```
<script type="text/javascript">
        //在每一个h1标签内追加一个itcast
    window.onload = function(){
        var nodeList = document.getElementsByTagName("h1");
        for(var i=0; i<nodeList.length;i++){
 //         var h1 = nodeList[i];
            var h1 = nodeList.item(i);
            alert(h1.innerHTML);
            h1.innerHTML += "itcast";
        }
    }
</script>
<body>
   <h1>AAA</h1>
   <h1>BBB</h1>
   <h1>CCC</h1>
   <h1>DDD</h1>
 </body>
```

批注 [ThinkPad31]: NodeList 有一个 length 属性和一个 item 方法。

批注 [ThinkPad32]: alert(h1.firstChild.nodeValue);

带格式的: 字体: (中文) +中文正文 (宋体), 字体颜色: 青色

②：**获得 node 后**

如果 node 是元素，去操作里面的文本内容 innerHTML (HTML 页面内所有元素，一定是 HTML 元素，innerHTML 是所有 HTML 元素通用属性)

批注 [ThinkPad33]: 它不是所有 xml 元素通用属性

XML 取得一个元素内部文本内容 element.firstChild.nodeValue(看批注 32)

③：**通过节点 Node 相对位置关系访问元素**

childNodes

firstChild

lastChild

nextSibling

parentNode

previousSibling

用 2 种方式打印——明天休息

```
<h1 id="h1">明天休息</h1> <br>
    var h1 = document.getElementById("h1");
    alert(h1.innerHTML);//方式一
```

alert(h1.firstChild.nodeValue);//方式二

3.DOM 元素常见操作

DOM 获取节点：节点查询 参上

DOM 改变节点： 元素属性修改 setAttribute(name,value)

内部文本元素的修改 innerHTML

DOM 删除节点：removeChild 删除子元素 、removeAttribute(name)删除节点指定属性

* 要删除节点 o o.parentNode.removeChild(o)

DOM 替换节点：replaceChild：父节点.replaceChild(新节点,被替换节点) ；

如果对于一个已经存在节点，执行 appendChild、 replaceChild 都会造成该节点一个移动效果，可以采取先克隆再复制来消除此效果。

DOM 创建节点:document 对象提供 createElement() 创建元素、createAttribute(name)创建属性、createTextNode() 创建文本节点

DOM 添加节点 appendChild 父元素.appendChild(新的子节点) ；

insertBefore 父节点.insertBefore(新子节点，已经存在子节点)

DOM 克隆节点 源节点.cloneNode(true); 该方法可以返回一个节点的克隆节点， 克隆隆节点包含原节点的属性和子元素

此节内容有大量的练习，建议大家做写，增强代码的熟练度。

四、JavaScript 事件

事件通常与函数配合使用，这样就可以通过发生的事件来驱动函数执行。事件是基于对象存在，事件通常可以修饰多种对象。

1.为对象添加事件的 2 种方式

①：在 HTML 元素中添加对象的事件

```
<head>
    <title>事件</title>
    <meta http-equiv="content-type" content="text/html; charset=gbk">
    <script type="text/javascript">
        function overtest(){
            alert("移动到图片上方");
        }
    </script>
</head>
<body>
    <img src="1.jpg" width="200" height="300" onmouseover = "overtest()";/>
</body>
```

带格式表格

批注 [ThinkPad34]: 方式一：在 HTML 元素中添加对象事件

带格式的: 字体：（中文）+中文正文（宋体），字体颜色：自动设置

带格式的: 行距：多倍行距 1.12 字行，调整中文与西文文字的间距，调整中文与数字的间距

②：在 JS 代码中为元素添加事件

```
<head>
    <title>事件</title>
```

```
    <meta http-equiv="content-type" content="text/html; charset=gbk">
    <script type="text/javascript">
        function overtest(){
            alert("移动到图片上方");
        }
        window.onload = function(){
            document.getElementById("myimg").onmouseover = overtest;
        }
    </script>
  </head>
  <body>
    <img src="1.jpg" width="200" height="300" id="myimg";/>
  </body>
```

批注 [ThinkPad35]: 捆绑函数不能加括号！

总结：优先使用第二种，将 js 代码与 HTML 元素代码分离开，更加方便统一管理维护。

问题：HTML 元素添加事件， 与 JS 添加事件是否可以完全等价？

在实际开发中，如果传参数，使用 HTML 元素绑定事件，如果不传参数，使用 JS 绑定事件。传参数也可以使用与 JS 绑定事件**【使用匿名函数】**。示例代码如下：

```
<head>
    <title>HTML事件绑定与JS绑定</title>
    <meta http-equiv="content-type" content="text/html; charset=gbk">
    <script type="text/javascript">
        function clicktest(o){
            alert(o);
        }
        window.onload = function(){
            document.getElementById("mybutton").onclick = function(){
                clicktest('次奥');
            }
        }
    </script>
</head>
<body>
    <input type="button" id="mybutton" value="点击我！">
    <input type="button" value="别碰我！" onclick = "clicktest('次奥')"/>
</body>
```

批注 [ThinkPad36]: 使用匿名函数可以解决此类问题

带格式的: 字体: (中文) +中文正文 (宋体), 字体颜色: 自动设置

带格式的: 行距: 多倍行距 1.12 字行, 调整中文与西文文字的间距, 调整中文与数字的间距

2.鼠标移动事件

Mousemove： 鼠标移动时触发事件 鼠标跟随效果

Mouseover： 鼠标从元素外，移动元素之上，信息提示、字体变色

Mouseout： 鼠标从元素上，移出元素范围，和 mouseover 一起使用

3.鼠标点击事件（左键相关事件）

click 鼠标单击事件
dbclick 鼠标双击事件
mousedown/mouseup 鼠标按下、按键弹起 click = mousedown + mouseup;
oncontextmenu 鼠标右键菜单事件 （不是浏览器兼容事件）

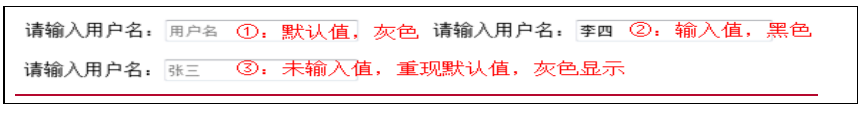
4.聚焦离焦事件

focus 聚焦 页面焦点定位到目标元素
blur 离焦 页面焦点由目标元素移开

Demo:

```
<script type="text/javascript">
    window.onload= function(){
        //页面加载后，在输入框加入默认值，并以灰色显示
        document.getElementById("username").value= "用户名";
        document.getElementById("username").style.color="gray";
        //聚焦事件
        document.getElementById("username").onfocus= function(){
            document.getElementById("username").value="";
            document.getElementById("username").style.color="black";
        }
        //离焦事件
        document.getElementById("username").onblur=function(){
            var name = document.getElementById("username").value;
            if(name==""){
                document.getElementById("username").value="张三";
                document.getElementById("username").style.color="gray";
            }
        }
    }
</script>
</head>
<body>
    请输入用户名：<input type="text" id="username"><br/>
</body>
```

显示效果截图如下：



5.键盘事件

使用场景：没有提交按钮，我们一般采用回车进行提交

带格式的：两端对齐
带格式的：字体：（中文）+中文正文（宋体），非加粗，字体颜色：自动设置
带格式的：字体：非加粗，字体颜色：文字 1
带格式的：段落间距段前：0.5 行，段后：0.5 行

Demo:

```
<script type="text/javascript">
    window.onload = function(){
        document.getElementById("message").onkeypress = function(e){
            // 判断用户 按键是否为 回车键
            if(e && e.which){
                // 火狐浏览器
                if(e.which == 13){
                    alert("提交")
                }
            }else{
                // IE浏览器
                if(window.event.keyCode ==13 ){
                    alert("提交")
                }
            }
        }
    }
</script>
<body>
    发送消息 <input type="text" name="message" id="message"/>
</body>
```

批注 [ThinkPad37]: 这里需要考虑浏览器兼容问题

带格式的: 字体: (中文) +中文正文 (宋体), 非加粗, 字体颜色: 自动设置

IE 中 window 对象，提供 event 属性，所以在 IE 中可以直接使用 event 对象

火狐没有全局 event 对象，必须在发生事件时，产生一个 event 对象 ，传递默认方法

6.form 的提交、重置事件

submit/reset

onsubmit = "return validateForm" 对表单进行校验

7.改变事件

onchange 制作 select 联动效果 ---- 省市联动

重点 ： onclick 、onchange 、onblur、 onsubmit

8.默认事件的阻止和传播阻止

使用场景极为常见，超链接用户点击后，取消了不发生跳转。

批注 [ThinkPad38]: 如果以后开发涉及到写原始 JavaScript，那么兼容性需考虑！

```
<script type="text/javascript">
    // 阻止默认事件发生
    function confirmDel(e){
        var isConfirm = window.confirm("确认删除吗？");
        if(!isConfirm){// 用户选择了取消
            // 阻止默认事件
            if(e && e.preventDefault){
```

```
                // 火狐
                e.preventDefault();
            }else{
                // IE
                window.event.returnValue = false;
            }
        }
    }
    // 阻止事件冒泡
    function aclick(e){
        alert("a");
        if(e && e.stopPropagation){
            // 火狐浏览器
            e.stopPropagation();
        }else{
            // IE 浏览器
            window.event.cancelBubble = true;
        }
    }

    function divclick(){
        alert("div");
    }
</script>
<body>
    <h1>默认事件</h1>
    <!-- 删除时，询问用户是否删除，然后再跳转-->
    <a href="del?id=1" onclick="confirmDel(event);">这是一个链接</a>
    <h1>事件传播</h1>
    <!-- 事件冒泡传播 -->
    <div onclick="divclick();"><a href="#" onclick="aclick(event);">这个链接 会触发两个事件执行</a></div>
</body>
```

HTML DOM Event 对象

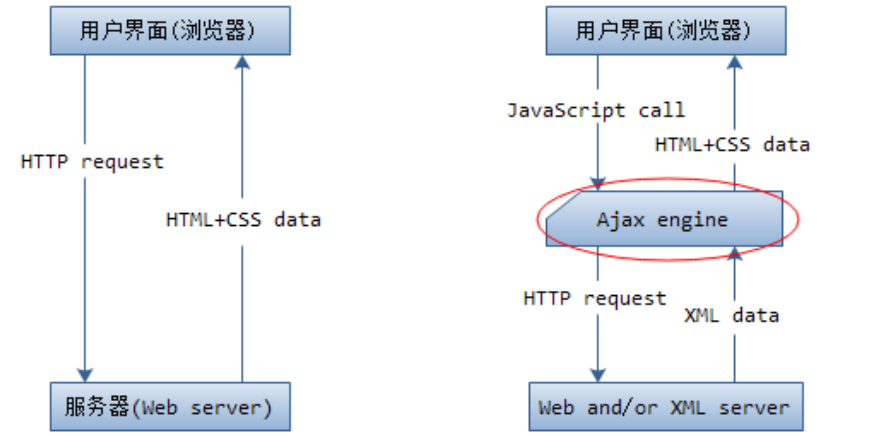
提供 preventDefault()用于阻止默认事件的发生， 该方法 IE 不支持 ，在 IE 中使用 returnValue

提供 stopPropagation()用与阻止事件传播，该方法 IE 不支持，在 IE 中 cancelBubble

五、Ajax 编程入门

1.web 交互的 2 种模式对比

①：2 种交互模式的流程



文字描述
①传统web交互模式：浏览器直接将请求发送给服务器，服务器回送响应，直接发给浏览器
②Ajax交互模式：浏览器首先将请求发送Ajax引擎（以XMLHttpRequest为核心），AJax引擎再将请求发送给服务器，服务器回送响应先发给Ajax引擎，再由引擎传给浏览器显示

②：**2 种交互模式用户体验**

同步交互模式：客户端提交请求，等待，在响应回到客户端前，客户端无法进行其他操作

异步交互模型：客户端将请求提交给 Ajax 引擎，客户端可以继续操作，由 Ajax 引擎来完成与服务武器端通信，当响应回来后，Ajax 引擎会更新客户页面，在客户端提交请求后，用户可以继续操作，而无需等待 。

2.Ajax 快速入门

①：**开发步骤**

1).创建 XMLHttpRequest 对象

2).将状态触发器绑定到一个函数

3).使用 open 方法建立与服务器的连接

4).向服务器端发送数据

5).在回调函数中对返回数据进行处理

Demo：

```
<script type="text/javascript">
    // 第一步 创建XMLHttpRequest
    function createXMLHttpRequest(){
        try {
            xmlHttp = new XMLHttpRequest();
        }catch (tryMS) {
            try {
                xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
            }catch (otherMS) {
```

```
            try {
                xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
            }catch (failed) {
                xmlHttp = null;
                // 这里可以报一个错误，无法获得 XMLHttpRequest对象
            }
        }
    }
    return xmlHttp;
}
var xmlHttp = createXMLHttpRequest();

// 第二步  响应从服务器返回后，Ajax引擎需要更新页面，绑定一个回调函数
xmlHttp.onreadystatechange = function(){
    // 第五步，响应返回后执行
        // 状态依次 是 0 - 4
        // 0 未初始化   1 正在加载   2 已经加载   3 交互中   4 响应完成
    if(xmlHttp.readyState == 4){
        // 判断数据是否正确
        if(xmlHttp.status == 200){
            // 响应有效
            alert("响应返回了..." + xmlHttp.responseText);
        }
    }
};

// 第三步 建立与服务器连接
//xmlHttp.open("GET","helloworld?name=张三");//helloworld代表  Servlet URL

xmlHttp.open("POST","helloworld");

// 第四步 发送数据
// xmlHttp.send(null);
xmlHttp.setRequestHeader("CONTENT-TYPE","application/x-www-form-urlencoded");
xmlHttp.send("name=李四");
</script>
```

批注 [ThinkPad39]: XMLHttpRequest 对象的方法：
open send
属性：
①onreadystatechange ：状态回调函数
②readyState ：对象状态
③status：服务器返回的 HTTP 状态码
④responseText/responseXML：服务器的响应字符串
⑤statusText： 服务器返回的 HTTP 状态信息

批注 [ThinkPad40]: 回调函数编写模式固定，主要负责处理数据

②：**XMLHttpRequest 对象的属性及方法**

属性：

1)onreadystateonchange：状态回调函数

2)readystate：对象状态

3)status：服务器返回的 http 状态码

4)reqsponseText/responseXML：服务器响应的字符串

5)statusText：服务器返回的 http 状态信息

方法：

1)open:

2)send:

③：**客户端向服务器提交数据**

1)get 方式发送数据

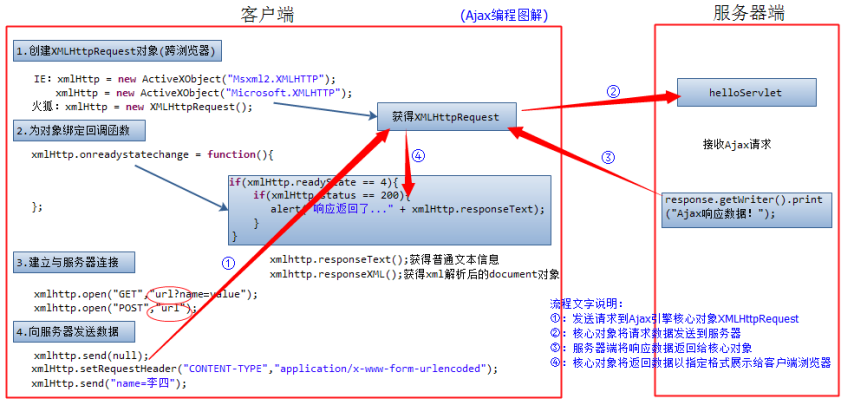
```
xmlHttp.open("GET","url?key=value"); // 参数已经在 url 上
xmlHttp.send(null);
```

2)post 方式发送数据

```
xmlHttp.open("POST","url"); // 不需要写参数
xmlHttp.setRequestHeader("CONTENT-TYPE","application/x-www-form-urlencoded"); // post 发送参数前，需要设置编码格式
xmlHttp.send("name=xxx&pwd=xxx"); // 发送 post 数据
```

④：**Ajax 编程图解**

说明：查看时，请将文档放到为 180%较为合适！



结合编程图解，我们将第一个案例的代码进行解剖：

用 demo 中的请求进行访问服务器，得到结果：

3.三种不同服务器响应数据类型编程

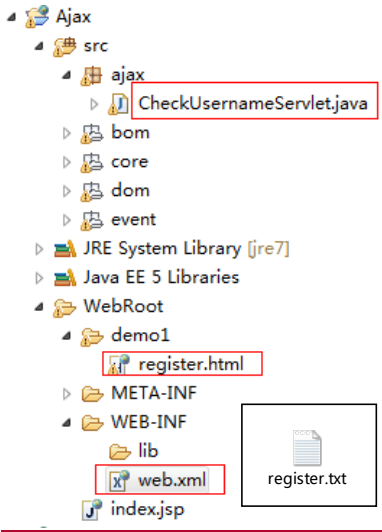
常见的服务器响应数据类型：html 片段、JSON 格式数据、xml 格式数据

①：HTML 片段的数据处理

练习1：验证用户名是否有效

- 通过 xmlhttp.responseText 获得返回数据
- 通过 Dom 查找获得元素
- 调用元素的 innerHTML 进行操作

批注 [ThinkPad41]： 重点练习！接下来的 3 个练习希望能独立完成！



从 Ajax 编程图解我们可以知道，我们只需要分别编写客户端和服务器端就好。客户端我们以 register.html 文件为主，服务器端程度为 CheckUsernameServlet 以及配置文件 web.xml。

下面分别对三个文件内容进行展示：

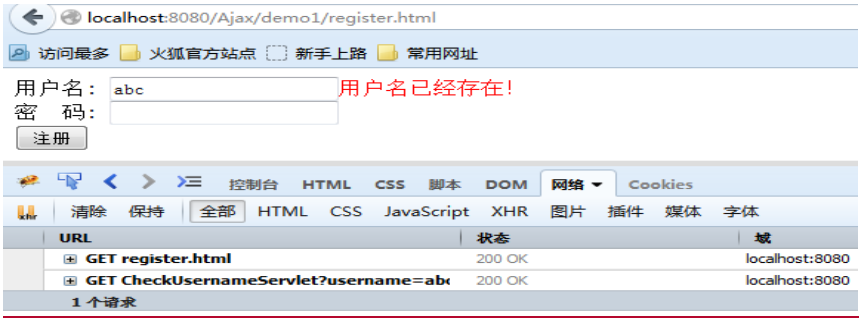
①：register.html

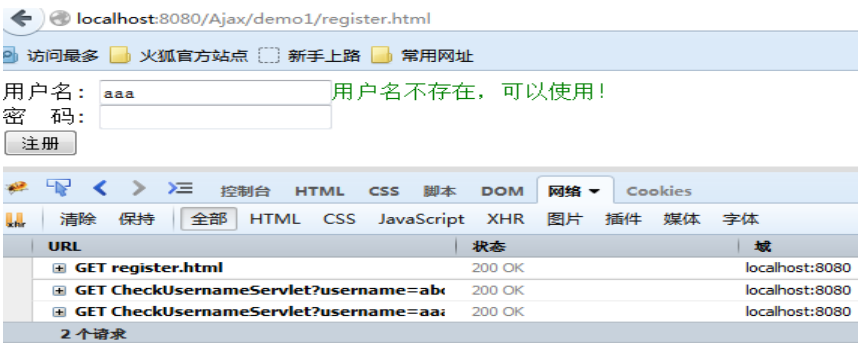
②：CheckUsernameServlet.java

③：web.xml

说明：三个文件的具体内容请双击方框内的图标即可查看，这里推荐使用 Notepad++ 进行关联

效果图展示：(数据使用 list 存储，abc 和 def 存在)





注意：在 register.html 文件中，与服务器建立连接是，url 一定要写对，否则返回数据时出现 404 错误!【"/Ajax/CheckUsernameServlet?username="】

②：JSON 格式数据处理

练习 2：通过链接获得 table 数据，显示 --- 返回 HTML 片段

通过 product.jsp 生成 HTML 片段，返回客户端，客户端 Ajax 引擎接收，通过 innerHTML 将 table 元素嵌入到页面内部，其思路与案例一类似，这里不做详细介绍。这里需要提出的是第二种解决方案 JSON(HTML 片段容易造成返回数据带回过多冗余的 HTML 标签元素)

JSON 是一种 JavaScript 轻量级数据交互格式，主要应用于 Ajax 编程。易于人阅读和编写，同时也易于机器解析和生成。

格式一： {key:value,key:value,key:value } 键值对直接用 ， 分开，键值之间用 ： **键**本身必须是字符串常量

{name : '张三'}

{'name':'张三'}

是等价的。

值加不加引号，是有区别的，不加引号是变量，加引号是常量字符串

格式二： [值 1，值 2 ,值 3] 数组结构

组合后复杂格式

[{name:'aaa'}，{name:'bbb'}，{name:ccc}] 表示三个对象数组

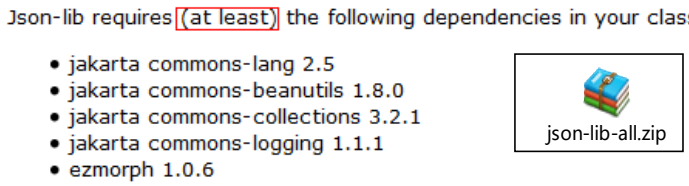
JSON 应用场景： AJAX 请求参数和响应数据

问题： 服务器端如何生成 json 格式数据------->>依赖第三方开源类库

③：JSON-lib 的使用

是 java 类库 ，支持 javabean map list array 转换 json 格式字符串， 支持将 json 字符串转换 javabean 对象(**反过来只支持这一种，使用很少**)

在使用 JSON-lib 时必须导入至少 5 个 jar 包



开发时我们使用 6 个 jar 包，双击 json-lib-all.zip 即可获取所需 jar 包。

1）将数组/list 集合解析成 JSON 串

使用 JSONArray 可以解析 Array 类型

JSONArray jsonArray = JSONArray.fromObject(list 变量);

批注 [ThinkPad42]: 截图内容注意看转后后 json 串的格式

```java
public class JsonlibTest {
    @Test
    public void test(){
        List<Product> products = new ArrayList<Product>();
        Product p1 = new Product();
        p1.setName("苹果手机");
        p1.setPrice(4999);

        Product p2 = new Product();
        p2.setName("thinkpad笔记本");
        p2.setPrice(8888);

        products.add(p1);
        products.add(p2);

        JSONArray jsonArray= JSONArray.fromObject(products);
        System.err.println(jsonArray);
    }
}
```

```
Problems Tasks Web Browser Console Servers JUnit
<terminated> JsonlibTest [JUnit] C:\Program Files\Java\jre7\bin\javaw.exe (2014-7-21 下午3:36:17)
[{"name":"苹果手机","price":4999},{"name":"thinkpad笔记本","price":8888}]
```

2）将 Javabean/Map 解析成 JSON 串

使用 JSONObject 可以解析 javabean 类型

JSONObject jsonObject = JSONObject.*fromObject*(javabean);

```java
    //将Javabean转化成JSON串
    @Test
    public void test2(){
        Product p1 = new Product();
        p1.setName("苹果手机");
        p1.setPrice(4999);

        JSONObject jsonObject = JSONObject.fromObject(p1);
        System.err.println(jsonObject);
    }
}
```

```
Problems Tasks Web Browser Console Servers JUnit
<terminated> JsonlibTest.test2 [JUnit] C:\Program Files\Java\jre7\bin\javaw.exe (2014-7-21 下午3:
{"name":"苹果手机","price":4999}
```

3）对象属性过滤转换 JSON 串

通过 JsonConfig 对象配置对象哪些属性不参与转换。

JsonConfig jsonConfig = new JsonConfig();

jsonConfig.setExcludes(new String[]{"price"});

批注 [ThinkPad43]: 创建 1 个 jsonConfig 对象

批注 [ThinkPad44]: 配置 product 对象的 price 属性不参与转换，这里需要的是一个 String 类型的数组

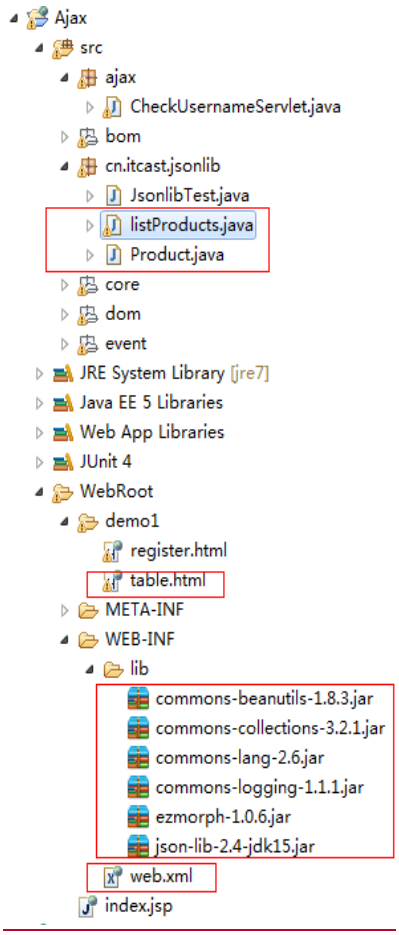
```java
    //对象属性过滤转换
    @Test
    public void test3(){
        Product p1 = new Product();
        p1.setName("苹果手机");
        p1.setPrice(4999);

        JsonConfig jsonConfig = new JsonConfig();
        jsonConfig.setExcludes(new String[]{"price"});

        JSONObject jsonObject = JSONObject.fromObject(p1, jsonConfig);
        System.err.println(jsonObject);
    }
}
```

过滤price属性

过滤后仅剩下name属性

```
Problems Tasks Web Browser Console Servers JUnit
<terminated> JsonlibTest.test3 [JUnit] C:\Program Files\Java\jre7\bin\javaw.exe (2014-7-21 下午4:01:43)
{"name":"苹果手机"}
```

重构练习 2



运行效果：

导入 jsonlib 需要的 jar 包

①：table.html(双击图标即可查看代码)

②：listProducts.java

③：web.xml(需要注意访问路径)

使用 json 数据格式，优化了性能，但是添加了程序员的代码量。

需要注意：

1)在 table.html 中点击超链接不发生跳转可以使用如下方法控制：

```
href="javascript:void(0);"
```

此外表格的拼接也需要注意

2)在 listProducts.java 中，注意使用 JSONArray 将 list 集合转化成 json 串：JSONArray.fromObject(products)

另外，发送数据的时候要设置编码格式：response.setContentType ("text/json;charset=utf-8");

3)使用 eval 将返回的 json 转换成 js 对象：

```
var text = xmlHttp.responseText;
var obj = eval("("+text+")");
```

④：**XML 格式数据处理**

练习 3：select 完成省级联动

1）XStream 的使用

问题：服务器端如何将 java 对象，生成 XML 格式数据？需要第三方类库支持 XStream

XStream is a simple library to serialize objects to XML and back again.

XStream 主要完成 Java 对象的序列化(xstream-1.3.1.jar)以及解析(xpp3_min-1.1.4c.jar)

2）XStream 的核心方法

- xStream.toXML(obj)：将对象序列化 XML
- xStream.fromXML(inputStream/xml 片段)：将 xml 信息解析成对象
- xStream.alias(String name,Class)：将类型解析或者序列化时，取一个别名

代码案例：**(序列化)**

```
public class XstreamTest {
    @Test
    public void test1(){
        List<City> cities = new ArrayList<City>();
        City c1 = new City();
        c1.setName("北京");
        c1.setDescription("中国的心脏");

        City c2 = new City();
        c2.setName("上海");
        c2.setDescription("东方明珠");

        cities.add(c1);
        cities.add(c2);

        XStream xStream = new XStream();
        xStream.alias("cities", List.class);
        xStream.alias("city", City.class);
        String xml = xStream.toXML(cities);
        System.err.println(xml);
    }
}
```

运行结果

```
<cities>
  <city>
    <name>北京</name>
    <description>中国的心脏</description>
  </city>
  <city>
    <name>上海</name>
    <description>东方明珠</description>
  </city>
</cities>
```

解析 xml 时，要注意别名的命名规则要与序列化时保持一致！

3）XStream 注解

在 Javabean 中进行注解

把某属性的名称取别名为 city：@XStreamAlias(value="city")

注解生效：xStream.autodetectAnnotations(true);

@XStreamAsAttribute 设置变量生成属性

@XStreamOmitField 设置变量不生成到 XML

@XStreamImplicit(itemFieldName = “hobbies”) 设置集合类型变量别名

六、jQuery 框架

jQuery 1.4 是企业主流版本，从 jQuery1.6 开始引入大量新特性。最新版本 2.1.1，这里讲解以 1.8.3 为主(新版本主要是浏览器兼容问题以及新特性)

jQuery 提供 jquery-1.8.3.js 和 jquery-1.8.3.min.js

jquery-1.8.3.js jQuery 框架源码，没有被精简，体积较大（主要用来研究 jQuery 源码），企业开发时，需要导入 jquery-1.8.3.min.js （精简过）

1.jQuery 程序快速入门

window.onload = function() {...}

等价于$(document).ready(function(){...});

①：jQuery 基本使用

批注 [ThinkPad45]: jQuery 可以绑定多个 onload 函数，而传统模式只能绑定一个函数，后面的会将前面的覆盖掉。

传统Js写法：

```
<script type="text/javascript">
    window.onload = function(){
        alert("传统JS，Ok");
    }
</script>
```

jQuery写法：

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        alert("ok");
    });
    $(document).ready(function(){
        alert("OK");
    });
</script>
```

②：**jQuery 核心函数**

```
1)jQuery(callback)  // 页面 onload 函数
2)jQuery(expression, [context])  // 查找指定对象 ------ 九种选择器
3)jQuery(elements) // 将 dom 对象转换为 jQuery 对象
    * document 是 DOM 对象 jQuery(document) 成为了 jQuery 对象
4)jQuery(html, [ownerDocument])  // 将 html 转换 jQuery 对象
    * jQuery("<p>hello</p>") ----- 得到 jQuery 对象
```

Demo:

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 获得div对象
        var domObject = document.getElementById("mydiv"); // 获得DOM对象
        domObject.innerHTML = "ITCAST";

        // 通过包装DOM对象，成为jQuery对象
        var $jQueryObject = $(domObject); // DOM对象成为 jQuery对象
        $jQueryObject.html("传智播客"); // html()是jQuery对象的方法

        // 通过访问jQuery对象下标0 元素，获得DOM对象
        var dom1 = $jQueryObject[0]; // 转换jQuery对象为DOM对象
        var dom2 = $jQueryObject.get(0);
        dom2.innerHTML = "传智播客ITCAST";
    });
```

```
</script>
<body>
    <div id="mydiv">hello</div>
</body>
```

jQuery 对象无法使用 DOM 对象属性方法,DOM 对象也无法使用 jQuery 对象属性方法。但是我们可以使用 jQuery 提供方法，将 DOM 对象通过 jQuery()函数包装成为 jQuery 对象，同样我们可以把 jQuery 对象转化成 DOM 对象。

jQuery--->DOM 对象：$jQuery 对象[0]或者$jQuery 对象.get(0);

DOM 对象--->jQuery：$(DOM 对象)

2.jQuery 九种选择器

批注 [ThinkPad46]: 选择器以练习为主，练习里面结合了 jQuery 手册里面涵盖的所有方法，请大家多写练习代码

选择器是 jQuery 的根基，在 jQuery 中，对事件处理，遍历 DOM 和 Ajax 操作都依赖于选择器

jQuery(expression, [context]) 在核心函数 jQuery 中传入表达式，对页面中元素进行选择==========jQuery 核心函数第二种！

①：**基本选择器**

根据元素 id 属性、class 属性、元素名称 对元素进行选择

id 选择器： $("#元素 id 属性")

class 选择器： $(".元素 class 属性")

元素名称选择器：$("元素名称")

多个选择器同时使用 selector1,select2 例如 ： $("#xxid ,.xxxclass") 同时选择 id 和 class 匹配两类元素

练习1：

✧ 通过each() 在每个div元素内容前 加入 “传智播客”

✧ 通过size() / length 打印页面中 class属性为 itcast 的元素数量

✧ 通过 index() 打印 id 属性为 foo 的 div 标签 是页面内的第几个 div 标签

带格式表格

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 选中所有div 得到集合
        $("div").each(function(){
            // 在每个div内容前加入“传智播客”
//          this.innerHTML = "传智播客" + this.innerHTML ;
            $(this).html("传智播客" + $(this).html());
        });

        // 通过size() / length 打印页面中 class属性为 itcast 的元素数量
//      alert($(".itcast").size());
        alert($(".itcast").length);
```

```
        // 通过index()  打印 id属性为foo 的div标签 是页面内的第几个div标签
        alert($("div").index($("#foo")));
    });
</script>
<body>
    <div>DIVAAAA</div>
    <div class="itcast">DIVBBBB</div>
    <div>DIVCCCC</div>
    <div>DIVDDDD</div>
    <div class="itcast">DIVEEEE</div>
    <div id="foo">DIVFFFF</div>
    <p>PAAAA</p>
    <p class="itcast">PBBBB</p>
    <p>PCCCC</p>
</body>
```

②：**层级选择器**

根据祖先、后代、父子关系、兄弟关系 进行选择

ancestor descendant 获取 ancestor 元素下边的所有元素 $("form input")

parent > child 获取 parent 元素下边的所有直接 child 子元素 $("form > input")

prev + next 获取紧随 pre 元素的后一个兄弟元素 $("label + input")

prev ~ siblings 获取 pre 元素后边的所有兄弟元素 $("form ~ input")

练习2：

✧ 将class属性值为itcast的元素下所有a元素字体变为红色

✧ 将class属性值为itcast的元素下直接a元素字体变为蓝色

✧ 将div元素后所有兄弟a元素，字体变为黄色，大小变为30px

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 将class属性值为itcast的元素下所有a元素字体变为红色
        $(".itcast a").css("color","red");

        // 将class属性值为itcast的元素下直接a元素字体变为蓝色
        $(".itcast>a").css("color","blue");

        // 将div元素后所有兄弟a元素，字体变为黄色，大小变为30px
        $("div~a").css({color:'yellow','font-size':'30px'});
```

```
    });
</script>
<body>
    <div class="itcast">
        <a>div link</a>
        <p>info
            <a>p link</a>
        </p>
    </div>
    <a>link</a>
    <p class="itcast">
        <a>p link</a>
    </p>
    <a>link</a>
</body>
```

③：**基本过滤选择器**

:first 选取第一个元素 $("tr:first")

:last 选取最后一个元素 $("tr:last")

:not(selector) 去除所有与给定选择器匹配的元素 $("input:not(:checked)")

:even 选取所有元素中偶数索引的元素，从 0 开始计数 $("tr:even") ----- 选取奇数元素

:odd 选取所有元素中奇数索引的元素 ，从 0 开始计数 $("tr:odd") ------ 选取偶数元素

:eq(index) 选取指定索引的元素 $("tr:eq(1)")

:gt(index) 选取索引大于指定 index 的元素 $("tr:gt(0)")

:lt(index) 选取索引小于指定 index 的元素 $("tr:lt(2)")

:header 选取所有的标题元素 如：h1, h2, h3 $(":header")

:animated 匹配所有正在执行动画效果的元素

练习3：

✧ 设置表格第一行，显示为红色

✧ 设置表格除第一行以外 显示为蓝色

✧ 设置表格奇数行背景色 黄色

✧ 设置表格偶数行背景色 绿色

✧ 设置页面中所有标题 显示为灰色

✧ 设置页面中正在执行动画效果 div 背景黄色，不执行动画 div 背景绿色

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 设置表格第一行，显示为红色
```

```
        $("tr:first").css("color","red");

        // 设置表格除第一行以外 显示为蓝色
//      $("tr:not(:first)").css("color","blue");
        $("tr:gt(0)").css("color","blue");

        // 设置表格奇数行背景色 黄色 /设置表格偶数行背景色 绿色
        $("tr:even").css("background-color","yellow");
        $("tr:odd").css("background-color","green");

        // 设置页面中所有标题 显示为灰色
        $(":header").css("color","gray");

        // 设置页面中正在执行动画效果div背景黄色，不执行动画div背景绿色
        // 无法选中正在执行动画的元素
        $("div:not(:animated)").css("background-color","green");
        $("div").click(function(){
            $(this).css("background-color","yellow");  //设置执行动画元素
            $(this).slideUp("slow");
        });
    });
</script>
<body>
    <h1>表格信息</h1>
    <h2>这是一张商品表</h2>
    <table border="1" width="600">
        <tr>
            <th>商品编号</th>
            <th>商品名称</th>
            <th>售价</th>
            <th>数量</th>
        </tr>
        <tr>
            <td>001</td>
            <td>冰箱</td>
            <td>3000</td>
            <td>100</td>
        </tr>
        <tr>
```

```
            <td>002</td>
            <td>洗衣机</td>
            <td>2000</td>
            <td>50</td>
        </tr>
        <tr>
            <td>003</td>
            <td>热水器</td>
            <td>1500</td>
            <td>20</td>
        </tr>
        <tr>
            <td>004</td>
            <td>手机</td>
            <td>2188</td>
            <td>200</td>
        </tr>
    </table>

    <div>
        slideDown(speed, [callback])
        概述
        通过高度变化（向下增大）来动态地显示所有匹配的元素，在显示完成后可选地触发一个回调函数。
        这个动画效果只调整元素的高度，可以使匹配的元素以“滑动”的方式显示出来。在jQuery 1.3中，上下的padding和margin也会有动画，效果更流畅。
        参数
        speedString,Number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)
        callback (可选)FunctionFunction在动画完成时执行的函数
    </div>
    <div>
        fadeOut(speed, [callback])
        概述
        通过不透明度的变化来实现所有匹配元素的淡出效果，并在动画完成后可选地触发一个回调函数。
        这个动画只调整元素的不透明度，也就是说所有匹配的元素的高度和宽度不会发生变化。
        参数
        speedString,Number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)
        callback (可选)Function在动画完成时执行的函数
    </div>
</body>
```

④：**内容过滤选择器**

内容选择器是对子元素和文本内容的操作

:contains(text) 选取包含 text 文本内容的元素 $("div:contains('John')") 文本内容含有 john 的所有 div

:empty 选取不包含子元素或者文本节点的空元素 $("td:empty") td 元素必须为空

:has(selector) 选取含有选择器所匹配的元素的元素 $("div:has(p)").addClass("test"); 含有 p 子元素的 div

:parent 选取含有子元素或文本节点的元素 $("td:parent") 所有不为空 td 元素选中

练习4：

✧ 设置含有文本内容 ”传智播客” 的 div 的字体颜色为红色

✧ 设置没有子元素的div元素 文本内容 ”这是一个空DIV“

✧ 设置包含p元素 的 div 背景色为黄色

✧ 设置所有含有子元素的span字体为蓝色

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 设置含有文本内容 ”传智播客” 的 div 的字体颜色为红色
        $("div:contains('传智播客')").css("color","red");

        // 设置没有子元素的div元素 文本内容 ”这是一个空DIV“
        $("div:empty").html('这是一个空DIV');

        // 设置包含p元素 的 div 背景色为黄色
        $("div:has(p)").css("background-color","yellow");

        // 设置所有含有子元素的span字体为蓝色
        $("span:parent").css("color","blue");
    });
</script>
<body>
    <div>今天是个晴天</div>
    <div>明天要去传智播客学 java</div>
    <div><span>JavaScript</span> 是网页开发中脚本技术</div>
    <div>Ajax 是异步的 JavaScript和 XML</div>
```

```
    <div> <p>jQuery</p> 是 JavaScript一个 轻量级框架</div>
    <div></div>
</body>
```

⑤：**可见性过滤选择器**

根据元素的可见与不可见状态来选取元素

:hidden 选取所有不可见元素 $("tr:hidden")

:visible 选取所有可见的元素 $("tr:visible")

练习5：

✧ 为表单中所有隐藏域 添加 class属性，值为itcast

✧ 设置table所有 可见 tr 背景色 黄色

✧ 设置table所有 隐藏tr 字体颜色为红色，显示出来 ，并输出tr中文本值

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        //  为表单中所有隐藏域  添加 class属性，值为itcast
        $("input:hidden").addClass("itcast");

        // 设置table所有 可见 tr 背景色 黄色
        $("tr:visible").css("background-color","yellow");

        // 设置table所有 隐藏tr 字体颜色为红色，显示出来 ，并输出tr中文本值
        $("tr:hidden").each(function(){
            alert($(this).text());
        });

        $("tr:hidden").css("color","red");
        $("tr:hidden").css("display","block");
    });
</script>
<body>
    <form>
        订单号 itcast-xxxx 金额 100元
        <!-- 隐藏令牌号 -->
        <input type="hidden" name="token"
                        value="12312-0xccx-zx-asd-21-asd-gdfgd" />
        <input type="submit" value="确认支付" />
    </form>
    <table>
        <tr style="display:none;">
```

```
            <td>冰箱</td>
        </tr>
        <tr style="visibility:hidden;">
            <td>洗衣机</td>
        </tr>
        <tr>
            <td>热水器</td>
        </tr>
    </table>
</body>
```

⑥：**属性过滤选择器**

通过元素的属性来选取相应的元素

[attribute] 选取拥有此属性的元素 $("div[id]")

[attribute=value] 选取指定属性值为 value 的所有元素

[attribute !=value] 选取属性值不为 value 的所有元素

[attribute ^= value] 选取属性值以 value 开始的所有元素

[attribute $= value] 选取属性值以 value 结束的所有元素

[attribute *= value] 选取属性值包含 value 的所有元素

练习6：

✧ 设置所有含有id属性的div，字体颜色红色

✧ 设置所有class属性值 含有itcast元素背景色为黄色

✧ 对所有既有id又有class属性div元素，添加一个点击事件，打印div标签中内容

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 设置所有含有id属性的div，字体颜色红色
        $("div[id]").css("color","red");

        // 设置所有class属性值 含有itcast元素背景色为黄色
        $("[class *= 'itcast']").css("background-color","yellow");

    // 对所有既有id又有class属性div元素，添加一个点击事件，打印div标签中内容
        $("div[id][class]").click(function(){
            alert($(this).html());
        });
    });
</script>
<body>
    <div>AAAA</div>
```

```
    <div id="mydiv" class="itcast1">BBBB</div>
    <div class="itcast2">CCCC</div>
    <div id="mydiv2">DDDD</div>
    <div class="divclass">EEEE</div>
    <div id="xxx" class="itcast3">FFFF</div>
    <p class="p-itcast">PPPPPP</p>
</body>
```

⑦：**子元素过滤选择器**

对某元素中的子元素进行选取

:nth-child(index/even/odd) 选取索引为 index 的元素、索引为偶数的元素、索引为奇数的元素 ----- index 从 1 开始 区别 eq

:first-child 选取第一个子元素

:last-child 选取最后一个子元素

:only-child 选取唯一子元素，它的父元素只有它这一个子元素

练习7：

✧ 选择id属性mytable 下3的倍数行，字体颜色为红色

✧ 表格 奇数行 背景色 黄色

✧ 表格 偶数行 背景色 灰色

✧ 只有一个td的 tr元素 字体为 蓝色

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 选择id属性mytable 下3的倍数行，字体颜色为红色
        $("#mytable tr:nth-child(3n)").css("color","red");

        // 表格 奇数行 背景色 黄色 / 表格 偶数行 背景色 灰色
        $("table tr:nth-child(even)").css("background-color","gray");
//    $("table  tr:nth-child(odd)").css("background-color","yellow");
// 从1计数
        $("tr:even").css("background-color","yellow");// 从0计数

        // 只有一个td的 tr元素 字体为 蓝色
        $("tr td:only-child").css("color","blue");
    });
</script>
</head>
<body>
    <table border="1" width="400" id="mytable">
        <tr><td>1</td><td>冰箱</td></tr>
```

```
        <tr><td>2</td><td>洗衣机</td></tr>
        <tr><td>3</td><td>热水器</td></tr>
        <tr><td>4</td><td>电饭锅</td></tr>
        <tr><td>5</td><td>电磁炉</td></tr>
        <tr><td>6</td><td>豆浆机</td></tr>
        <tr><td>7</td><td>微波炉</td></tr>
        <tr><td>8</td><td>电视</td></tr>
        <tr><td>9</td><td>空调</td></tr>
        <tr><td>10</td><td>收音机</td></tr>
        <tr><td>11</td><td>排油烟机</td></tr>
        <tr><td>12</td><td>加湿器</td></tr>
        <tr><td>13 电暖气</td>加湿器</tr>
    </table>
</body>
```

⑧：**表单过滤选择器**

选取表单元素的过滤选择器

:input 选取所有<input>、<textarea>、<select >和<button>元素

:text 选取所有的文本框元素

:password 选取所有的密码框元素

:radio 选取所有的单选框元素

:checkbox 选取所有的多选框元素

:submit 选取所有的提交按钮元素

:image 选取所有的图像按钮元素

:reset 选取所有重置按钮元素

:button 选取所有按钮元素

:file 选取所有文件上传域元素

:hidden 选取所有不可见元素

练习8：

✧ 对所有text框和password框，添加离焦事件，校验输入内容不能为空

✧ 对button 添加 点击事件，提交form表单

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 对所有text框和password框，添加离焦事件，校验输入内容不能为空
        $(":text,:password").blur(function(){
            // 获得表单元素内容 val()
            var value  = $(this).val(); // 获得value 属性
            // 将输入内容 trim
            if($.trim(value) == "" ){
```

```
            alert("用户名和密码不能为空");
        }
    });
    // 对button 添加 点击事件，提交form表单
    $(":button").click(function(){
        $("#myform").submit();
    });
});
</script>
<body>
    <form action="regist" method="post" id="myform">
        用户名 <input type="text" name="username" /><br/>
        密码 <input type="password" name="password" /><br/>
        性别 <input type="radio" name="gender" value="男" />男
            <input type="radio" name="gender" value="女"/><br/>
        城市 <select name="city">
            <option value="北京">北京</option>
            <option value="上海">上海</option>
        </select>
        个人简介 <textarea rows="5" cols="60" name="introduce"></textarea>
        <input type="button" value="提交"/>
    </form>
</body>
```

⑨：**表单对象属性过滤选择器**

选取表单元素属性的过滤选择器

:enabled 选取所有可用元素

:disabled 选取所有不可用元素

:checked 选取所有被选中的元素，如单选框、复选框

:selected 选取所有被选中项元素，如下拉列表框、列表框

练习9：

✧ 点击 button 打印 radio checkbox select 中选中项的值

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 点击button 打印radio checkbox select 中选中项的值
        $("#mybutton").click(function(){
            // 打印选中性别的值
            $("input[name='gender']:checked").each(function(){
                alert($(this).val());
            });
```

```
            // 打印爱好
            $("input[name='hobby']:checked").each(function(){
                alert($(this).val());
            });

            // 打印城市
            $("select[name='city'] option:selected").each(function(){
                alert($(this).val());
            });
        });
    });
</script>
</head>
<body>
    性别 : <input type="radio" name="gender" value="男" /> 男
           <input type="radio" name="gender" value="女"/> 女 <br/>
    爱好: <input type="checkbox" name="hobby" value="体育" />体育
    <input type="checkbox" name="hobby" value="读书" />读书
    <input type="checkbox" name="hobby" value="音乐" />音乐
    <input type="checkbox" name="hobby" value="旅游" />旅游  <br/>
    城市 : <select name="city">
        <option value="北京">北京</option>
        <option value="上海">上海</option>
        <option value="广州">广州</option>
    </select><br/>
    <input type="button" value="获取选中的值 " id="mybutton" />
</body>
```

3.jQuery 选择器总结

①：**对象访问核心方法**

each(function(){}) 遍历集合

size()/length 属性 返回集合长度

index() 查找目标元素是集合中第几个元素

②：**CSS 样式操作**

css(name,value) css({name:value,name:value}); 同时修改多个 CSS 样式

基本过滤选择器与 筛选过滤 API 功能是相同

$("tr:first") 等价于 $("tr").first()

③：**九种选择器重点**

- 基本选择器和层级选择器 **锁定元素**
- 使用属性过滤选择器和内容过滤选择器 **具体选中元素**
- 表单操作 :checked :selected 选中 **表单选中元素**

配合基本过滤选择器，缩小选中的范围

4.jQuery 的 DOM 操作

使用 jQuery 的九种选择器可以基本选中需要操作的对象，但是为了提高 jQuery 的查询效率，可以结合 jQuery 的内置查找函数一起使用

①：**查询**

children([expr]) 获取指定的子元素
find(expr) 获取指定的后代元素
parents([expr]) 获得祖辈元素
parent() 获取父元素
next([expr]) 获取下一个兄弟元素
prev([expr]) 获取前一个兄弟元素
siblings([expr]) 获取所有兄弟元素
在 XML 解析中 find 方法使用最多

对查找结果进行遍历操作 each(function(){… }) ，在 each 函数中可以通过 this 获得 DOM 对象，$(this) 获得 jQuery 对象

②：**属性操作**

设置属性 attr(name,value)
读取属性 attr(name)
同时设置多个属性 attr({name:value,name:value... });
attr("checked","true") 等价于 prop("checked")

练习1：

✧ 点击一个 button，动态设置 div 的属性 id name class
✧ 尝试能否设置一个不存在的属性？

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    // 点击一个button，动态设置 div的属性 id name class
    // 尝试能否设置一个不存在的属性？
    $(function(){
        $("#mybutton").click(function(){
            // 可以设置一个不存在属性
            $("#mydiv").attr({'id':'xxxdiv','name':'xxxxname','class':'xxxclass','itcastinfo':'xxxxitcast'});
```

```
        });
    });
</script>
</head>
<body>
    <div id="mydiv">itcast</div>
    <input type="button" value="设置属性" id ="mybutton" />
</body>
```

③：**CSS 操作**

通过 attr 属性设置/获取 style 属性

```
    attr('style','color:red'); // 添加 style 属性
```

设置 CSS 样式属性

```
    css(name, value)  设置一个 CSS 样式属性
    css(properties)  传递 key-value 对象，设置多个 CSS 样式属性
```

设置 class 属性

```
    addClass(class)  添加一个 class 属性
    removeClass([class])  移除一个 class 属性
    toggleClass(class)如果存在（不存在）就删除（添加）一个类
```

练习2：

✧ 点击button，使一个div的背景颜色变为 黄色

✧ 通过toggleClass(class) 实现点击 字体变为紅色，再点击样式还原

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 点击button，使一个div的背景颜色变为 黄色
        $("#button1").click(function(){
            $("#div1").css("background-color","yellow");
        });

        // 通过toggleClass(class) 实现点击 字体变为紅色，再点击样式还原
        $("#button2").click(function(){
            $("#div1").toggleClass("divclass");
        });
    });
</script>
<style type="text/css">
    .divclass {
        color:red;
    }
```

```
</style>
<body>
    <div id="div1">AAAAAA</div>
    <input type="button" value="背景颜色变为黄色" id="button1" />
    <input type="button" value="字体颜色开关" id="button2" />
</body>
```

④：HTML 代码&文本&值操作

- 读取和设置某个元素中 HTML 内容

  html() 读取 innerHTML

  html(content) 设置 innerHTML

- 读取和设置某个元素中的文本内容

  text() 读取文本内容

  text(content) 设置文本内容

- 文本框、下拉列表框、单选框 选中的元素值

  val() 读取元素 value 属性

  val(content) 设置元素 value 属性

练习3：

✧ 传智播客 获取 div 中 html 和 text 对比

✧ 使用 val() 获得文本框、下拉框、单选框选中的 value

✧ 测试能否通过 val() 设置单选框、下拉框的选中效果

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // <div><p>传智播客</p></div>  获取div中 html和text 对比
        var $obj = $("<div><p>传智播客</p></div>");
//      alert($obj.html());
//      alert($obj.text());

        //使用val() 获得文本框、下拉框、单选框选中的value
        $("#mybutton").click(function(){
            alert($("#username").val());
            alert($("input[name='gender']:checked").val());
            alert($("#city").val());
        });

        //测试能否通过 val() 设置单选框、下拉框的选中效果
        $("#city").val("广州");
        $("input[name='gender']").val(['女']);
    });
```

批注 [ThinkPad47]: 这个函数不能用于 XML 文档。但可以用于 XHTML 文档

批注 [ThinkPad48]: 该方法既可以用于 XHTML 也可以用于 XML 文档

运行结果3.png

```
</script>
</head>
<body>
    用户名 <input type="text" id="username" /> <br/>
    性别 <input type="radio" name="gender" value="男" />男
        <input type="radio" name="gender" value="女" /> 女<br/>
    城市 <select id="city">
        <option value="北京">北京</option>
        <option value="上海">上海</option>
        <option value="广州">广州</option>
    </select> <br/>
    <input type="button" value="获取val" id="mybutton"/>
</body>
```

设置 val 控制 radio select checkbox 选中

```
    $("#city").val("广州");
    $("input[name='gender']").val(['女']);
```

练习4：

✧ 输出所有 select 元素下的所有 option 元素中对应的文本内容

例如:<option value="中专">中专^^</option> 输出--->中专^^

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        $("#edu option").each(function(){
            alert($(this).text());
        });
    });
</script>
</head>
<body>
<select id="edu">
    <option>博士</option>
    <option>硕士</option>
    <option>本科</option>
    <option>大专</option>
</select>
```

⑤：jQuery 添加元素

● 创建元素

拼接好 HTML 代码片段 $(html 片段) ---- 产生 jQuery 对象

- 内部插入：

  $node.append($newNode) 内部结尾追加

  $node.prepend($newNode) 内部开始位置追加

- 外部插入：

  $node.after($newNode) 在存在元素后面追加 -- 兄弟

  $newNode.insertBefore($node) 在存在元素前面追加

练习5：

✧ 在 id=edu 下增加<option value="大专">大专</option>

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 追加 option 内容大专
        // 创建元素
        var $newNode = $("<option value='大专'>大专</option>");
        // 添加元素
//      $("#edu").append($newNode); // 内部结尾
//      $("#edu").prepend($newNode); // 内部开始

//      $("option[value='本科']").after($newNode);
        $newNode.insertBefore($("option:contains('高中')"));
    });
</script>
<body>
<select id="edu">
    <option value="博士">博士</option>
    <option value="硕士">硕士</option>
    <option value="本科">本科</option>
    <option value="高中">高中</option>
</select>
```

⑥：**jQuery 删除元素**

选中要删除元素.remove() ---- 完成元素删除

选中要删除元素.remove(expr) ----- 删除特定规则元素

remove 删除节点后，事件也会删除

detach 删除节点后，事件会保留 从 1.4 新 API

练习6：

✧ 分别使用 detach 和 remove 删除带有 click 事件的 p 标签，删除后再将 p 重新加入 body 查看事件是否存在

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
```

```
<script type="text/javascript">
    $(function(){
        $("p").click(function(){
            alert($(this).text());
        });
        // 使用remove方法删除 p元素，连同事件一起删除
//      var $p = $("p").remove();
        // 使用detach删除，事件会保留
        var $p = $("p").detach();

        $(document.body).append($p);
    });
</script>
</head>
<body>
    <p>AAA</p>
    <div>BBB</div>
</body>
```

练习7：

✧ 表格数据添加与删除练习

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        $("#mybutton").click(function(){
            var name = $("#name").val();
            var email = $("#email").val();
            var phone = $("#phone").val();
            // DOM添加
            var $tr =
 $("<tr><td>"+name+"</td><td>"+email+"</td><td>"+phone+"</td><td><a
href='javascript:void(0)' onclick='del(this)'>删除</a></td></tr>");
            $("table").append($tr);
        });
    });

    function del(o) {
        // 对象o 代表a 标签
        $(o).parents("tr").remove();
    }
```

批注 [ThinkPad49]: jQuery 不推荐这么干，在后面事件绑定章节会进行优化！【实时追加事件】

```
$("a:contains('删除')")
.live("click",function(){
    $(this).parents("tr").remove();
});
```

```
</script>
</head>
<body>
<form>
    姓名 <input type="text" id="name" />
    邮箱 <input type="text" id="email" />
    手机<input type="text" id="phone" /> <br/>
    <input type="button" value="提交" id="mybutton"/>
</form>
<hr/>
<table border="1" width="400">
    <tr>
        <th>姓名</th>
        <th>邮箱</th>
        <th>手机</th>
        <th>删除</th>
    </tr>
</table>
```

姓名 aa 邮箱 bb

提交

姓名	邮箱	手机	删除
aa	bb	cc	删除
aa	bb	cc	删除
aa	bb	cc	删除
aa	bb	cc	删除

⑦：**jQuery 复制和替换**

- 复制节点

```
$(“p”).clone();  返回节点克隆后的副本，但不会克隆原节点的事件
$(“p”).clone(true);  克隆节点，保留原有事件
```

- 替换节点

```
$("p").replaceWith("<b>ITCAST</b>");  将所有 p 元素，替换为"<b>ITCAST</b>“
$(“<b>ITCAST</b>”).replaceAll(“p”);  与 replaceWith 相反
```

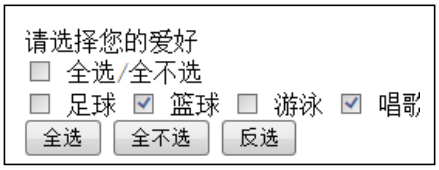
⑧：**全选以及左右移动练习**

练习8：

✧ 全选练习

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 全选/ 全不选
        $("#checkallbox").click(function(){
            var isChecked = this.checked;
            $("input[name='hobby']").each(function(){
                this.checked = isChecked;
            });
        });
```

```
        // 全选
        $("#checkAllBtn").click(function(){
            $("input[name='hobby']").attr("checked","checked");
        });
        // 全不选
        $("#checkAllNotBtn").click(function(){
            $("input[name='hobby']").removeAttr("checked");
        });

        // 反选
        $("#checkOppoBtn").click(function(){
            $("input[name='hobby']").each(function(){
                this.checked = !this.checked;
            });
        });
    });
</script>
<body>
    请选择您的爱好<br/>
    <input type="checkbox" id="checkallbox" /> 全选/全不选 <br/>
    <input type="checkbox" name="hobby" value="足球" /> 足球
    <input type="checkbox" name="hobby" value="篮球" /> 篮球
    <input type="checkbox" name="hobby" value="游泳" /> 游泳
    <input type="checkbox" name="hobby" value="唱歌" /> 唱歌 <br/>
    <input type="button" value="全选" id="checkAllBtn" />
    <input type="button" value="全不选" id="checkAllNotBtn" />
    <input type="button" value="反选" id="checkOppoBtn" />
</body>
```



练习9：

✧ 左右移动练习


```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        //选中的去右边
        $("#chooseToRight").click(function(){
            $("#right").append($("#left option:selected"));
        });

        // 全去右边
```

```
        $("#allToRight").click(function(){
            $("#right").append($("#left option"));
        });

        // 全去左边
        $("#allToLeft").click(function(){
            $("#left").append($("#right option"));
        });

        //选中的去左边
        $("#chooseToLeft").click(function(){
            $("#left").append($("#right option:selected"));
        });
    });
</script>
</head>
<body>
    <select id="left" multiple="multiple" size="15">
        <option>北京</option>
        <option>上海</option>
        <option>天津</option>
        <option>杭州</option>
        <option>武汉</option>
        <option>广州</option>
        <option>深圳</option>
        <option>南京</option>
    </select>

    <input type="button" value="-->" id="chooseToRight" />
    <input type="button" value="==>" id="allToRight" />
    <input type="button" value="<--" id="chooseToLeft" />
    <input type="button" value="<==" id="allToLeft" />

    <select id="right" multiple="multiple" size="15">
        <option>郑州</option>
    </select>
</body>
```

5.jQuery 事件

①：**事件绑定**

传统 js 一般一个对象只能绑定某种事件一个函数

jQuery 支持对同一个对象，同一个事件可以绑定多个函数

绑定事件函数到对象有两种写法

写法一

```
$("div").click(function(){
    ......
});
```

写法二

```
$("div").bind("click",function(){
    ......
});
```

取消绑定：$("div").unbind("click");

*** live 为满足条件对象，实时追加绑定 、取消 live 事件用 die 方法

Demo:

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 使用live绑定
        $("div").live("click",function(){
            alert($(this).text());
        });

//      $("div").click(function(){
//          alert($(this).text());
//      });

        // 解除绑定
//      $("div").unbind("click");

        // 新加入div元素没有之前div元素绑定事件
        $(document.body).append($("<div>CCC</div>"));
    });
</script>
</head>
<body>
    <div>AAA</div>
    <div>BBB</div>
</body>
```

批注 [ThinkPad50]: 新加元素使用 bind 绑定，无法使用绑定的事件，live 绑定可以！

②：**事件一次性绑定和自动触发**

一次性事件 one(type, [data], fn) 为对象绑定一次性事件，只有一次有效

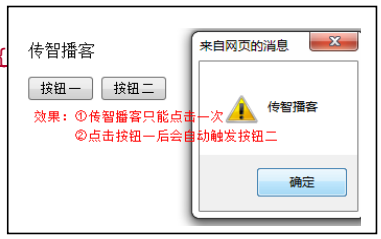
触发事件 trigger(type, [data]) 触发目标对象指定的事件执行

练习1：

✧ 为页面内所有 p 元素绑定 一次性事件，点击打印 p 元素中内容

✧ 页面内有两个按钮，点击按钮 1， 触发按钮 2 的 click 事件执行

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        //为页面内所有p 元素绑定 一次性事件，点击打印p元素中内容
        $("p").one("click",function(){
            alert($(this).text());
        });

        //页面内有两个按钮，点击按钮1， 触发按钮2的 click事件执行
        $("#mybutton1").click(function(){
            alert("点击了按钮一");
            // 触发2 click事件
            $("#mybutton2").trigger("click");
        });

        $("#mybutton2").click(function(){
            alert("点击了按钮二");
        });
    });
</script>
<body>
    <p>传智播客</p>
    <input type="button" value="按钮一" id="mybutton1" />
    <input type="button" value="按钮二" id="mybutton2" />
</body>
```



③：**事件切换**

hover(mouseover,mouseout) 模拟鼠标悬停事件

toggle(fn1,fn2,fn3...) ； 鼠标点击一次 触发一个函数

练习1：

✧ 编写一个 div 元素，光标移动上去 字体变为红色，移开后 变为蓝色

✧ 建立三张图片，页面显示第一张，点击切换到第二张，点击切换到第三张

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 编写一个div元素，光标移动上去 字体变为红色，移开后 变为蓝色
        $("div").hover(function(){
            // over
```

```
            $(this).css("color","red");
        },function(){
            // out
            $(this).css("color","blue");
        });

        // 建立三张图片，页面显示第一张，点击切换到第二张，点击切换到第三张
        $("img").toggle(function(){
            $(this).attr("src","2.jpg");
        },function(){
            $(this).attr("src","3.jpg");
        },function(){
            $(this).attr("src","4.jpg");
        });
    });
</script>
<body>
    <div>鼠标移动上 会变色的内容！ </div>
    <img src="1.jpg" width="240" height="180" />
</body>
```

④：**事件阻止默认动作和传播**

- 默认动作阻止

```
$("a").click(function(event){
        event.preventDefault();
        // do something
});
```

- 取消事件冒泡

```
$("p").click(function(event){
        event.stopPropagation();
        // do something
});
```

```
<script type="text/javascript" src="../jquery-1.8.3.min.js"></script>
<script type="text/javascript">
    $(function(){
        // 通过event阻止默认事件
        $("#dellink").click(function(event){
            var isConfirm = window.confirm("确认删除吗？");
            if(!isConfirm){
                event.preventDefault();
            }
        });

        $("div").click(function(){
            alert($(this).html());
        });

        // 阻止事件冒泡
        $("p").click(function(event){
            alert($(this).html());
            event.stopPropagation();
        });
    });
</script>
<body>
    <a href="del?id=1" id="dellink">删除资料</a>
    <div><p>信息</p></div>
</body>
```

批注 [ThinkPad51]: 之前在 IE 中是不兼容的，jQuery 中完美解决兼容性问题

七、jQuery 的 Ajax 编程

1.回顾传统 Ajax 开发步骤

①：创建 xmlHttpRequest 对象

```
var xmlHttp = creatHttpRequest();
```

②：绑定回调函数

```
xmlHttp.onreadystatechange = function(){……}
```

③：建立连接

```
xmlHttp.open(“GET”,”url”);
```

④：发送数据

```
xmlHttp.send(null) //GET 请求
```

如果是 POST 请求需要设置编码格式：

```
xmlHttp.setRequestHeader("CONTENT-TYPE","application/x-www-form-
```

urlencoded");
xmlHttp.send("key=value?key=value")
⑤：书写回调函数
if(readyState == 4){
    if(status ==200){
        ……
        //操作 xmlHttp.responseText 主要针对返回 HTML 片段和 json
        //操作 xmlHttp.responseXML 主要针对返回 XML 片段。
    }
}

2.jQuery 的 Ajax 开发

jQuery 提供了最底层的 Ajax 调用方法：$.ajax
$.ajax{
    type:"POST"
    url: "some.php"
    data: "name=John&location=Boston",
    success: function(msg){
        alert("Data Saved: " + msg);
    }
}
// 因为使用比较繁琐，所以在实际开发中，应用很少
为了简化 Ajax 开发，jQuery 提供了对$.ajax()进一步的封装方法$load、$get、$post。这三个方法不支持跨域，$getJSON、$getScript 支持跨域。
①：**load 方法**
load 方法是 jQuery 中最为简单和常用的 Ajax 方法，处理 HTML 片段此方法最为合适。
语法
    $("jquery 对象").load("url","data") ;
    url：Ajax 访问服务器地址
    data：请求参数
返回内容 HTML 片段 ，自动放入$("jquery 对象")innerHTML 中(如果返回的数据需要处理，我们可以使用 get 或者 post)

load()方法的传递参数根据参数 data 来自动自定。如过没有参数的传递，采用 GET 方式传递，否则采用 POST 方式
练习一：校验用户名是否存在
此练习在第五章的第三小节有实现代码,这里使用 jQuery 的方式进行简要的列出核心代码:
```
$(function(){
    // 为用户名添加离焦事件
    $("input[name='username']").blur(function(){
```

```
        // 获得当前输入 username
        var username = $(this).val();
        // 提交Ajax校验
        $("#info").load("/Ajax/checkUsername" , {'username': username});
    });
});
<form>
    <!-- div display:block  自动换行效果  span display:inline; 不会换行 -->
    用户名 <input type="text" name="username" /> <span id="info"></span>
id="info"></span>     <br/>
    密码 <input type="password" name="password"/><br/>
     <input type="submit" value="注册" />
</form>
```

②：**get 方法和 post 方法**

语法 ：

```
$.get/$.post("url","parameter",function(data){...});
```

url Ajax 访问服务器地址

parameter 代表请求参数

function 回调函数 data 代表从服务器返回数据内容

这里 data 代表各种数据内容 ： HTML 片段、JSON、XML

如果传递参数给服务器使用 $.post ，不需要传参数 可以使用 $.get

~~HTML~~ 知识点详解

带格式的: 字体: (中文) +中文正文 (宋体)

带格式的: 缩进: 首行缩进: 0.74 厘米

~~上面的例子包含了 3 个 HTML 元素~~

~~效果如下：~~

批注 [ThinkPad70]: 标题元素后面自动添加空行。

~~显示效果：~~

~~显示效果：~~

带格式的: 缩进: 首行缩进: 0.74 厘米