Ajax

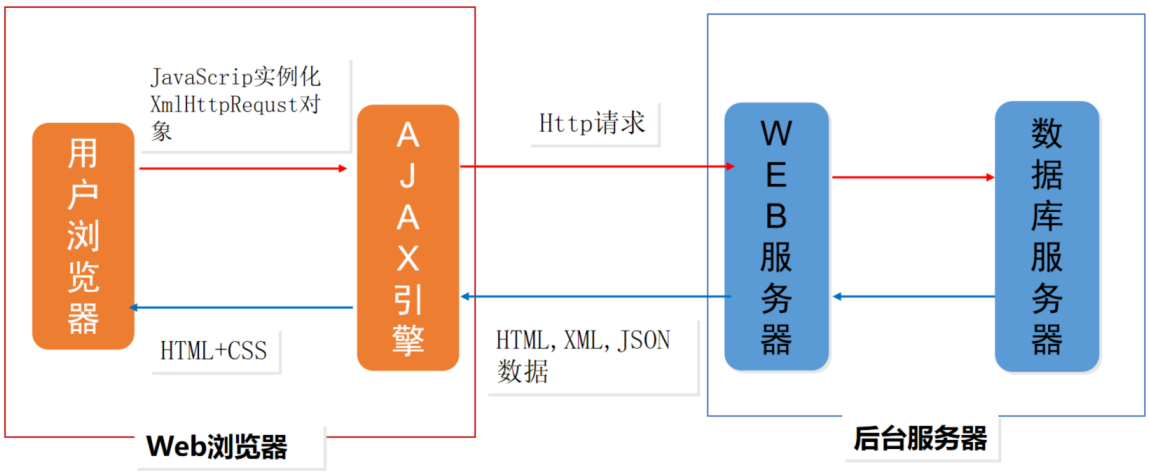
1、Ajax 概述

- Asynchronous Javascript And XML（异步JavaScript和XML），标准读音 [ˈeɪˌdʒæks]，中文音译：阿贾克斯
- AJAX是浏览器提供的一套方法，可以实现页面无刷新更新数据，提高用户浏览网站应用的体验
- AJAX并不是一种语言，而是一种创建交互式网页应用的网页开发技术
- AJAX是Javascript、XHTML和CSS、DOM、XML和XSTL、XMLHttpRequest 等技术的组合
- 使用XHTML+CSS来标准化呈现；
- 使用XML和XSLT进行数据交换及相关操作；
- 使用XMLHttpRequest对象与Web服务器进行异步数据通信；
- 使用Javascript操作DOM进行动态显示及交互；
- AJAX核心对象是**XMLHttpRequest**

2、Ajax 的应用场景

- 页面下拉加载更多数据
- 列表数据无刷新分页
- 表单项离开焦点数据验证

3、AJAX工作原理



4、AJAX优缺点

4.1 AJAX优点

- 减轻服务器的负担，AJAX一般只从服务器获取只需要的数据。
- 无刷新页面更新，减少用户等待时间。
- 更好的客户体验，可以将一些服务器的工作转移到客户端完成，节约网络资源，提高用户体验
- 无平台限制
- 促进显示与数据相分离

4.2 AJAX缺点

- 页面中存在大量js，给搜索引擎带来困难
- AJAX干掉了Back和History功能，即对浏览器机制的破坏
- 存在跨域问题
- 只能传输及接收utf-8编码数据

5、Ajax 的实现步骤

5.1 初始化XMLHttpRequest对象

```
if (window.XMLHttpRequest) {// Mozilla, Safari, ...
    var xhr = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE 5 ,6
    var xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
```

5.2 声明Ajax 请求地址以及请求方式

- http请求中get方式和post方式的最大区别是：
    - get请求的数据会放在url地址中发送，速度快，不安全，受url长度限制
    - post请求的数据会放在请求的数据包中发送，速度稍慢，安全，不受长度限制

```
// GET请求方式
xhr.open("GET","请求地址",true);

// POST请求方式
xhr.open("POST","请求地址",true);
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
```

5.3 发送请求

```
// GET请求方式
xhr.send();

// POST请求方式
xhr.send('name=zhangsan&password=123');
```

5.4 获取服务器端的响应数据

```
xhr.onreadystatechange = function(){
    if(xhr.readyState == 4  && xhr.status == 200){
        console.log(xhr.responseText)
    }
}
```

```
/* readyState为Ajax 状态码
在创建ajax对象，配置ajax对象，发送请求，以及接收完服务器端响应数据，这个过程中的每一个步骤都会对应一个数值，这个数值就是ajax状态码
0：请求未初始化(还没有调用open())
1：请求已经建立，但是还没有发送(还没有调用send())
2：请求已经发送
3：请求正在处理中，通常响应中已经有部分数据可以用了
4：响应已经完成，可以获取并使用服务器的响应了
*/

/*status为http状态码，200表示请求成功，请求所希望的响应头或数据将随响应返回*/
```

5.5 响应数据的格式

- XML:笨重,解析困难,但是XML是通用的数据交换格式
- HTML:不需要解析,直接可以存放到文档中,若仅更新一部分区域,但传输的数据不是很方便,,且HTML代码需要拼装完成
- JSON: 小巧,有面向对象的特征,且很多很多前后端语言都有封装好的JSON字符串转换方法

JSON:一种与开发语言无关的、轻量级的数据存储格式，全称JavaScript Object Notation，一种数据格式的标准规范，起初来源于JavaScript这门语言，后来随着使用的广泛，几乎每门开发语言都有处理JSON的API。

优点：易于人的阅读和编写，易于程序解析与生产。

JSON样例：首先一个花括号{}，整个代表一个对象，同时里面是一种Key-Value的存储形式，它还有不同的数据类型来区分

数据类型表示

数据结构：Object、Array

基本类型：string，number，true，false，null

（1）Object

{key:value,key:value...}

key：string类型

value：任何基本类型或数据结构

（2）Array

[value,value...]

value：任何基本类型或数据结构。

比如：{"name":"李广", "values":[1,2,45,"你好"] }

JSON数据示例

示例网址：

http://note.youdao.com/noteshare?id=84831efce8bfe03ffa2a19dff5e10175&sub=17BC8D897C864FACB6E1396DCB8503DD

https://note.youdao.com/ynoteshare1/index.html?id=18f5f94a87d5871bc9368d8739c6b21c&type=note

6、Express中参数的获取

6.1 GET参数的获取

- Express框架中使用req.query即可获取GET参数，框架内部会将GET参数转换为对象并返回。

```
app.get('/路由名称', (request, response) => {
    //接收get数据
    console.log(request.query);
});
```

6.2 POST参数的获取

- Express中接收post请求参数需要借助第三方包 body-parser
  - 安装body-parser模块

```
#在命令行下执行如下命令
npm i -S body-parser
#安装完成后，重启web服务
nodemone website1.js
```

  - 引入body-parser模块

```
// 引入body-parser模块
const bodyParser = require('body-parser');
// 配置body-parser模块
app.use(bodyParser.urlencoded({ extended: false }));
// 接收请求
app.post('/路由名称', (request, response) => {
    // 接收请求参数
    console.log(request.body);
})
```

demo1: 验证用户名是否存在

1.客户端发送请求

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>表单验证</title>
</head>
<body>
    <form action="">
        <label for="">用户名: </label>
        <input type="text" name="username" placeholder="请输入用户名">
        <span></span>
    </form>
</body>
<script>
    var username = document.querySelector("input[name='username']");
    // 失去焦点进行验证
    username.onblur = function() {
        // 初始化XHR对象
        if (window.XMLHttpRequest) {
            var xhr = new XMLHttpRequest();
        } else if (window.ActiveXObject) {
            var xhr = new ActiveXObject("Microsoft.XMLHTTP");
        }

        // GET请求方式
        xhr.open("GET", "http://127.0.0.1:3000/chkUsername?username=" +
username.value, true);

        //发送请求
        xhr.send();
        //获取服务器端的响应数据
        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4 && xhr.status == 200) {
                //console.log(xhr.responseText)
                // 将返回的JSON字符串转换为JSON对象
                data = JSON.parse(xhr.responseText);
                // 根据返回结果，显示验证信息
                var color = "red"
                if (data.status == "ok") {
                    color = "green";
                }
                username.nextElementSibling.innerHTML = data.msg;
```

```
                username.nextElementSibling.style.color = color;
            }
        }
    }
</script>
</html>
```

2、服务器端接收请求并响应

```
// 引入Express框架
const express = require('express');
// 使用框架创建web服务器
const app = express();
// 声明静态资源目录
app.use("/", express.static("views"));

// 假设用户数据信息为(实际项目中应该从数据库查询得到):
var users = [
    { username: '张三', password: "123" },
    { username: '李四', password: "123" },
    { username: '王五', password: "123" },
];
// 创建路由接收用户请求
app.get('/chkUsername', (request, response) => {
    //接收get数据
    var data = request.query;
    //判断是否存在该用户
    var result = { status: 'ok', msg: '用户名可以使用' };
    for (var i = 0; i < users.length; i++) {
        if (users[i].username == data.username) {
            result = { status: 'error', msg: '用户名已存在' };
            break;
        }
    }

    // 对客户端做出响应,send方法会根据内容的类型自动设置请求头
    response.send(result);
});
// 程序监听3000端口
app.listen(3000);
```

demo2:登录验证（用户名密码是否正确）

```
//要求采用POST请求方式实现
```

四、封装ajax函数

```
/*
 * AJAX请求
 * @param  {string}  url              请求的URL地址
 * @param  {function}  callback       回调函数
 * @param  {string}  type             请求方式（GET或POST）
 * @param  {string}  data             POST方式发送的数据
*/
```

```
function ajax(url, callback, type = "GET", data = "") {
        if (window.XMLHttpRequest) {
            var xhr = new XMLHttpRequest();
        } else if (window.ActiveXObject) {
            var xhr = new ActiveXObject("Microsoft.XMLHTTP");
        }

        xhr.open(type, url, true);
        // GET请求方式
        if (type == "POST") {
            xhr.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded");
            xhr.send(data);
        } else {
            xhr.send();
        }

        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4) {
                if (xhr.status == 200) {
                    callback(JSON.parse(xhr.responseText));
                } else {
                    console.log('error:' + xhr.status);
                }
            }
        }

    }
```

项目地址：http://47.105.96.139:8082/

作业

一、什么是AJAX

二、AJAX的工作原理是什么

三、AJAX优缺点

四、get和post的区别

五、AJAX实现的基本步骤

六、安装NodeJS，使用express框架搭建web站点

七、异步验证用户名是否存在

八、异步登录

九、封装AJAX函数

jQuery之AJAX、常用插件

一、AJAX

jQuery中针对不同的http异步请求方式对ajax请求进行了封装，主要有三种方式：

1.1 ajax()

语法:

```
$.ajax({
    url:'服务器url地址',
    //data:'名=值&名=值&...',
    data:{名:值,名:值,...},
    type:'post|get',
    dataType:'xml|html|text|json|script',
    success:function(data){
        console.log(data);
    },
    error:function(xhr){
        console.log(xhr);
    },
    timeout:2000,
    async:true  ,
    cache:false
})
```

1.2 get()

语法:

```
$.get('服务器url地址', "json格式或字符串格式", function(data){
    console.log(data);
}), "xml|html|json|text|script")
```

1.3 post()

语法:

```
$.post('服务器url地址', "json格式或字符串格式", function(data){
    console.log(data);
}), "xml|html|json|text|script")
```

> get和post请求方式区别:

	GET	POST
缓存	能被缓存	不能缓存
编码类型	application/x-www-form-urlencoded	application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码
历史	参数保留在浏览器历史中	参数不会保存在浏览器历史中
对数据长度的限制	当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）	无限制
对数据类型的限制	只允许 ASCII 字符	没有限制，也允许二进制数据
安全性	与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分	POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中
可见性	数据在 URL 中对所有人都是可见的	数据不会显示在 URL 中

二、常用插件

jQuery 功能比较有限，想要更复杂的特效效果，可以借助于 jQuery 插件完成。 这些插件也是依赖于jQuery来完成的，所以必须要先引入jQuery文件，因此也称为 jQuery 插件。

jQuery 插件常用的网站：

1. jQuery 插件库 http://www.jq22.com/
2. jQuery 之家 http://www.htmleaf.com/
3. github官网 https://github.com/

jQuery 插件使用步骤：

1. 引入jQuery 文件 和 插件文件。
2. 复制相关html、css、js (调用插件)。

这里主要给大家介绍5款开发中最常用的插件：

2.1 layer

layer是一款近年来备受青睐的web弹层组件，*layer* 采用 MIT 开源许可证，*将会永久性提供无偿服务*。

layer 已经成为国内乃至全世界百万人使用的 Web 弹层解决方案

使用步骤：

1、下载layer，并将layer目录放到项目目录下

官网地址：https://layer.layui.com/

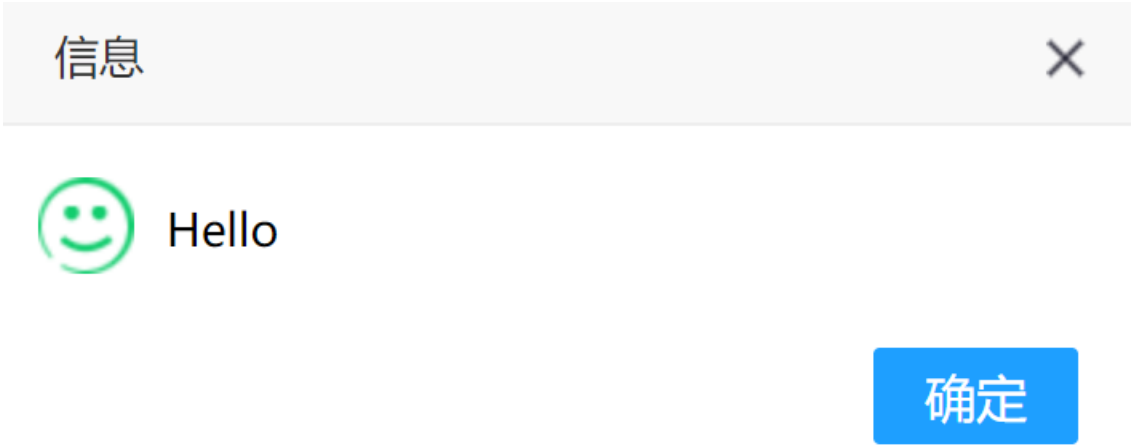
2、引入jQuery以及layer

```
<script src="jquery-3.5.1.min.js"></script>
<script src="layer/layer.js"></script>
```

3、使用layer实现弹层效果

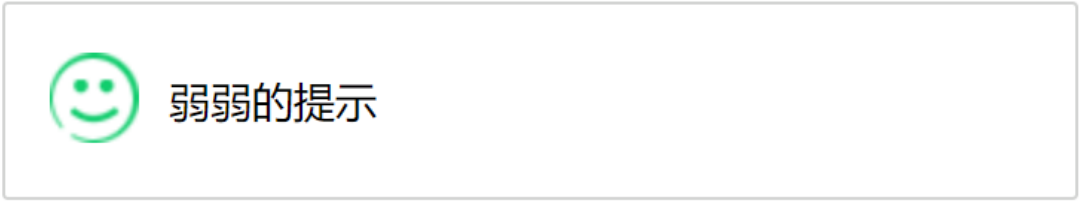
- 警告层

```
// 普通版
layer.alert('Hello');
// 表情版
layer.alert('Hello', {icon: 6}); //0感叹号 1对勾 2 叉  3问号 4 锁  5哭  6笑
// 回调版
layer.alert('Hello', {icon: 6}, function() {
    // 函数内容
 });
```



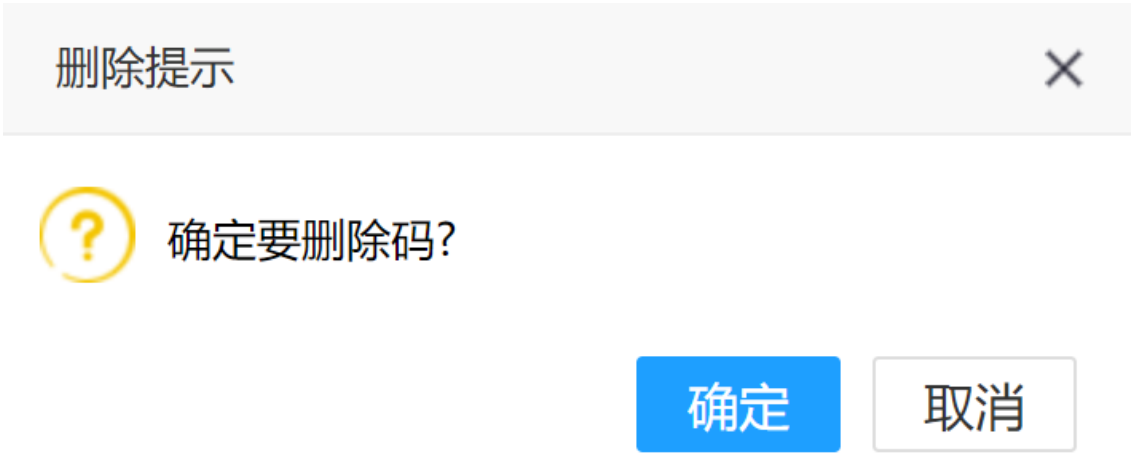
- 提示层

```
// 普通版
layer.msg('弱弱的提示');
// 表情版
layer.msg('弱弱的提示',{icon:6});
// 回调版
layer.msg('弱弱的提示',{icon:6},function(){
    layer.msg('提示完成')
});
// 停留时间版
layer.msg('弱弱的提示', {
    icon: 6,
    time: 2000
}, function() {
    layer.msg('提示完成')
})
```



- 询问层

```
layer.confirm("确定要删除码?", {
    btn: ['确定', '取消'],
    icon: 3,
    title: '删除提示'
    },
    function() {
        layer.msg('是的,我要删除')
    },
    function() {
        layer.msg('你走吧,我不想杀你')
    }
)
```



- tips层

```
layer.tips('我会在指定元素的右边出现','#div1');          //默认右面
layer.tips('我会在指定元素的右边出现','#div1',{tips:1}); //上面
layer.tips('我会在指定元素的右边出现','#div1',{tips:2}); //右面
layer.tips('我会在指定元素的左边出现','#div1',{tips:3}); //下面
layer.tips('我会在指定元素的左边出现','#div1',{tips:4}); //左面
layer.tips('我会在指定元素的左边出现', '#div1', {
    tips: [4, '#f00']
});
```

- 弹出层

```
// 弹出content内容:
layer.open({
    type: 1,
    title: '用户登录',
    area: ['600px', '400px'],
    content: "<h1>登录表单</h1>"
})

// 弹出页面内指定内容
layer.open({
```

```
    type:1,
    title:'用户登录',
    area:['600px','400px'],
    content:$('#form1')
})

// 弹出其它页面内容
layer.open({
    type: 2,
    title: '用户登录',
    area: ['1000px', '500px'],
    content:['http://www.baidu.com','no']
})
```

2.2 layDate

layDate 是一款很棒的时间插件

官网地址：https://www.layui.com/laydate/

1、引入laydate.js

```
<script src="laydate/laydate.js"></script>
```

2、HTML设置

```
<input type="text" name="start_time" id="start_time"/>
<input type="text" name="end_time" id="end_time"/>
```

3、调用ladate

```
laydate.render({
    elem:'#start_time',          // 绑定input标签
})

laydate.render({
    elem:'#end_time',
    type:'datetime',             // 插件类型：date、time、datatime、year、month
    theme:'#000',                // 主题颜色
    min:'2015-01-01 00:00:00',   // 最小日期
```

```
    max:'2029-01-01 00:00:00',  // 最大日期
    value:new Date(),           // 默认日期
    //value:new Date('2020-01-01 00:00:00')
    calendar: true              //公历节日
})
```

2.3 distpicker

distipicker是一款功能强大省市县三级联动插件

1、引入jQuery和distpicker

```
<script src="jquery-3.5.1.min.js"></script>
<script src="distpicker.min.js"></script>
```

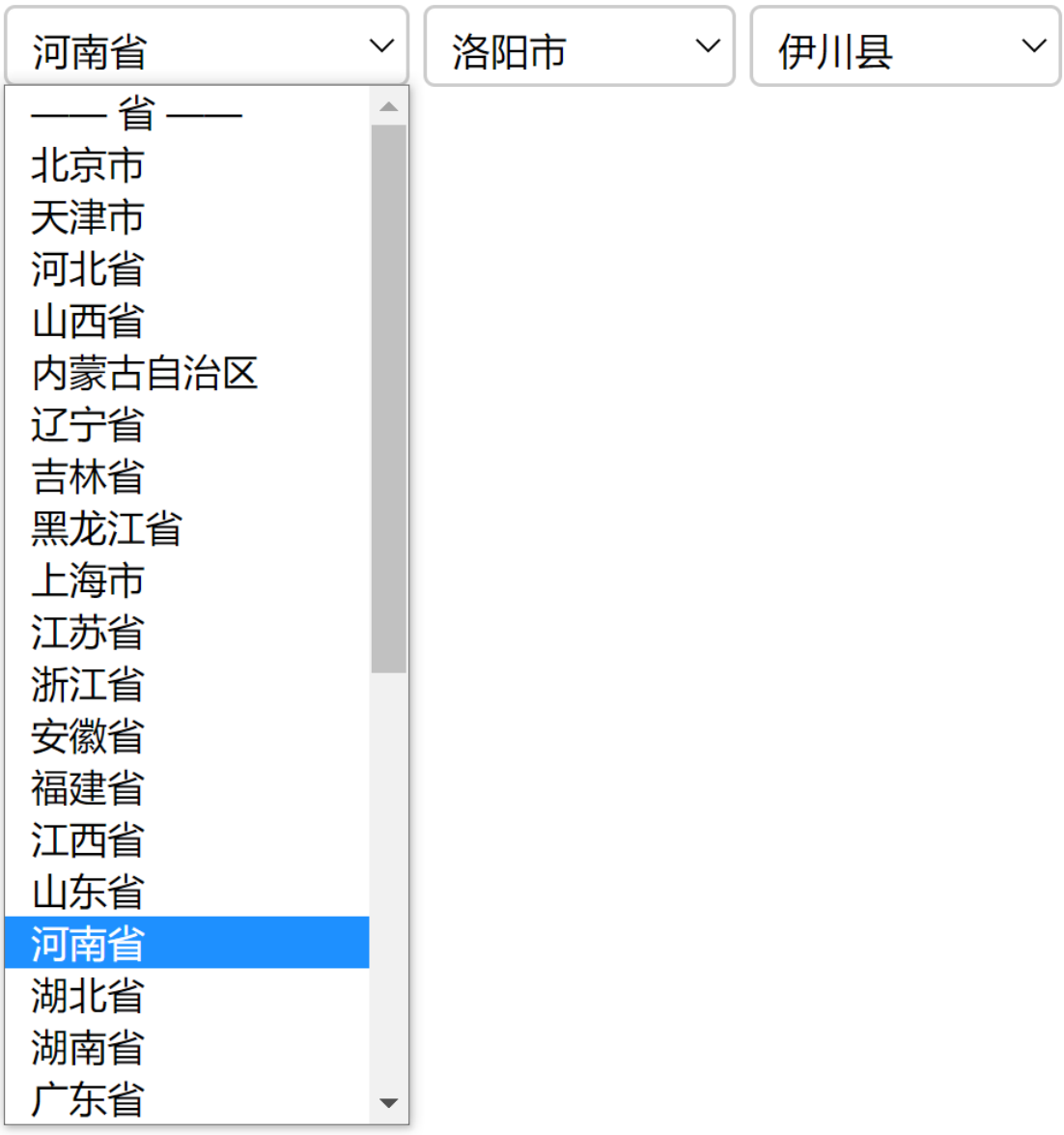
2、HTML设置

```
<div id="address">
    <!-- 省 -->
    <select></select>
    <!-- 市 -->
    <select></select>
    <!-- 区 -->
    <select></select>
</div>
```

3、调用distpicker

```
// 普通调用
$('#address').distpicker();

// 调用时设置默认地址
$('#address').distpicker({
    province: '河南省',
    city: '洛阳市',
    district: '伊川县'
});
```



2.4 lazyload

图片的懒加载就是：当页面滑动到有图片的位置，图片才进行加载，用以提升页面打开的速度及用户体验。

1、引入jQuery和lazyload

```
<script src="../jquery-3.5.1.min.js"></script>
<script src="jquery.lazyload.js"></script>
```

2、HTML设置

```
<img src="images/mi1.jpg" alt="" />
<img src="images/mi2.jpg" alt="" />
<img class="lazy" data-original="images/mi3.jpg" alt="" />
<img class="lazy" data-original="images/mi4.jpg" alt="" />
<img class="lazy" data-original="images/mi6.jpg" alt="" />
<img class="lazy" data-original="images/mi7.jpg" alt="" />
<img class="lazy" data-original="images/mi8.jpg" alt="" />
```

3、调用lazyload

```
$('img.lazy').lazyload({
    effect: 'fadeIn' ,
    placeholder : "images/loading.gif"
    //threshold :180 //在距离元素180像素时预加载
})
```

2.5 validate

1、引入jQuery和jquery.validate.js

```
<script src="../static/jquery-3.5.1.min.js"></script>
<script src="../static/jquery.validate.js"></script>
```

2、HTML设置

```
<form action="server.jsp" method="post" id="form1">
    <table>
        <tr>
            <td>用户名</td>
            <td><input type="text" name="username"/></td>
        </tr>
        <tr>
            <td>密码</td>
            <td><input type="password" name="password" id="pwd"/></td>
        </tr>
        <tr>
            <td>确认密码</td>
            <td><input type="password" name="repwd"/></td>
        </tr>
        <tr>
            <td>电子邮箱</td>
            <td><input type="text" name="mail"/></td>
        </tr>
```

```
        <tr>
            <td>手机</td>
            <td><input type="text" name="mobile"/></td>
        </tr>
        <tr>
            <td colspan="2">
              <input type="submit" value="开始注册"/>
            </td>
        </tr>
    </table>
</form>
```

3、调用validate

```
var myValidate=$('#form1').validate({
     // 验证规则
     rules:{
         username:{
             required:true,
             minlength:2,
             maxlength:10,
             remote:{
                 url:"http://127.0.0.1:8000/checkusername",
                 type:"get"
             }
         },
         mail:{
             required:true,
             email:true
         },
         password:{
             required:true
         },
         repwd:{
             required:true,
             equalTo:"#pwd"
         },
         mobile:{
             required:true,
             mobile:true
         }
     },
     // 错误提示信息
     messages:{
         username:{
             required:'用户名不能为空',
             minlength:'用户名长度至少需要2个字符',
             maxlength:'用户名长度最多10个字符',
             remote:'用户已被占用'
         },
         mail:{
             required:'邮箱不能为空',
             email:'邮箱格式不正确'
         },
         password:{
             required:'密码不能为空'
         },
```

```
            repwd:{
                required:'确认密码不能为空',
                equalTo:'两次密码输入不一致'
            },
            mobile:{
                required:'手机号不能为空'
            }
        },
        //提示样式
          //设置提示标签
         errorElement:'div',
         //配置成功提示
         success:function(div){
             div.addClass('ok').html('验证通过');
         },
         validClass:'ok'
    })

    // 自定义手机号码验证
    jQuery.validator.addMethod("mobile", function(value, element) {
        var mobileReg = /^1[34578][0-9]{9}$/;
        return this.optional(element) || (mobileReg.test(value));
    }, "手机号码格式不正确");

   // 提交时验证
    $('#form1').submit(function(){
        //在提交之前判断验证是否全部通过
        if(!myValidate.form()){
           return false;
        }
    })
```

> 验证规则：

验证规则	说明
required:true	必填
remote:{url:"",type:"get"}	服务端校验
minlength:5	最小长度
maxlength:18	最大长度
rangelength:[5,18]	长度范围
min:1	最小值
max:100	最大值
range:[1,100]	取值范围
email:true	邮箱格式
url:true	url地址格式
number:true	数字
equalTo:"#pwd"	与id为pwd的元素值相等

> 自定义验证规则

```
// 邮政编码验证
jQuery.validator.addMethod("postCode", function(value, element) {
    var codeReg= /^[0-9]{6}$/;
    return this.optional(element) || (codeReg.test(value));
}, "请正确填写您的邮政编码");

// 手机号码验证
jQuery.validator.addMethod("mobile", function(value, element) {
    var mobileReg = /^1[3-9][0-9]{9}$/;
    return this.optional(element) || (mobileReg.test(value));
}, "请正确填写您的手机号码");
```

> 自定义提示效果

```
$('#form1').validate({
    //设置提示标签
    errorElement:'div',
    //配置成功提示
    success:function(div){
        div.addClass('ok').html('验证通过');
    },
    validClass:'ok'
)}
```

```
table td {
    position: relative;
    line-height: 45px;
    padding-left: 10px;
}
```

```
table td:first-child {
    width: 5em;
    text-align-last: justify;
}

table input {
    padding: 5px;
    border: 1px solid #ccc;
    border-radius: 2px;
    outline: none;
}

div.error,
div.ok {
    position: absolute;
    top: 30px;
    padding-left: 20px;
    font-size: 12px;
    font-weight: bold;
}

div.error {
    background: url(../static/images/error.png)  5px 17px no-repeat;
    color: red;
}

div.ok {
    background: url(../static/images/ok.png)  5px 17px no-repeat;
    color: green;
}
```

> 提交表单时进行验证：

```
$('#form1').submit(function() {
    //在表单提交之前判断验证是否全部通过
    // console.log(myValidate.form())
    if (!myValidate.form()) {
        return false;
    }
})
```

> 远程验证

```
// 验证规则
$('#form1').validate({
    //验证规则
    rules: {
        username: {
            required: true,
            remote: {
                url: "http://127.0.0.1:8000/checkUsername",
                type: "get"
            }
        },
    },
```

```
    //错误提示信息
        messages: {
            username: {
                required: '用户名不能为空',
                remote: '用户已被占用'
            }
        }
})
```

```
// 服务端异步响应
// 验证通过响应字符串型的"true"
// 验证不通过响应字符串型的"false"
//假设数据库中的用户信息为
var userData = [
    { username: '张三', password: '123' },
    { username: '李四', password: '123' },
    { username: '王五', password: '123' },
];
app.get("/checkUsername", function(request, response) {
    //接收get请求的数据
    var data = request.query;
    // 查询数据库，判断用户名是否存在
    var result = "true";
    for (var i = 0; i < userData.length; i++) {
        if (userData[i].username == data.username) {
            result = "false";
            break;
        }
    }
    // 响应结果给浏览器
    response.send(result);
})
```

作业

一、举例说明$.ajax的用法

二、使用$.get()实现用户名是否存在验证功能

三、使用$.post()实现异步登录功能

四、举例说明layer的用法

五、举例说明layDate的用法

六、举例说明distpicker的用法

七、举例说明lazyload的用法

八、举例说明validate的用法