学习方法

1. 建立逻辑思维
   1. 多分析
   2. 理解逻辑算法
   3. 多练、多敲、多读
2. 编程习惯
3. 大胆动手
4. 保持好奇好
5. 乐观面对错误

就业能力

1. 分析能力
2. 独立思考能力
3. 程序阅读能力
4. 程序排错能力
5. 独立编写程序能力
6. 团队协作能力
7. 学习能力
8. 善于总结的能力

专业技能

正则及表单验证

1. 正则的概念
   1. 正则表达式是一种特殊的字符串模式，用于匹配一组字符串，就好比用模具做产品，而正则就是这个模具，定义一种规则去匹配符合规则的字符。
2. 创建方式
   1. 使用RegExp这个对象，第一个参数就是我们的模式字符串,第二个参数可选，模式修饰符
      1. Var 变量名 = new RegExp(“正则表达式”,”标志位”);  
         var reg = new RegExp(“study”,”ig”); //i:表示忽略大小写。G:表示全局匹配
      2. 用常量方式直接声明  
         var 变量名 = /正则表达式/标志位  
         var reg = /study/ig;
3. 正则表达式
   1. 扩展：测试正则表达式test方法
      1. 正则表达式对象名.test(“字符串”):用于检测提供的目标字符串是否包含正则表达式的匹配内容，如果包含返回true,否则返回false
         1. Var reg = /study/;  
            reg.test(“Study”) //false  
            reg.test(“study”) //true  
            除了匹配相等以外，还可以匹配更复杂的规则  
            “good good study,day day up”判断是否包含符合该正则的内容
   2. 扩展：测试正则表达式exec方法
      1. 正则表达式对象名.exec(“字符串”):用于在字符串中查找指定正则表达式，如果exec()方法执行成功，则返回包含该查找字符串的相关信息数组。如果执行失败，则返回null。
         1. Var pattern = /box/i;  
            var str = “This is a Box!”;  
            alert(pattern.exec(str));
   3. 元字符（三、三、二、三个一）
      1. 三：{}[]()
         1. {}:表示括号前面的一个字符连续出现的次数  
            {m}：表示括号前面的一个字符只能连续出现m次  
            {m,}:表示至少出现m次，即m~无限次  
            {m,n}:表示至少出现m次，最多出现n次，即m~n次
         2. []:表示范围
            1. [\u4e00-\u9fa5]:表示中文的范围
            2. [^\u4e00-\u9fa5]:表示非中文
            3. [^a-zA-Z]:表示非字母
            4. [^0-9]:表示非数字
            5. [^ ]:表示非空格
         3. （）：表示组
      2. 三：\* + ？ ：表示前面字符连续出现的次数
         1. \* ：等同于{0,}表示0~无限次
         2. + ：等同于{1,} 表示1~无限次，至少出现1次
         3. ？ ：等同于{0，1}表示0~1次，最多可出现1次，可以不出现
      3. 二：^ $
         1. ^ :定（断）头 ： 表示必须以^后面的一个字符开头
         2. $ :定（断）尾 ：表示必须以$前面的一个字符结尾
   4. 三个一： . | \
      1. . : 表示模糊匹配，一个点可以匹配一个任意字符
      2. | :表示或者
      3. \ :表示转义字符
         1. \d: 表示数字 [0-9]
         2. \D: 表示非数字[^0-9]
         3. \w: 表示字母、数字、下划线 [a-zA-Z0-9\_]
         4. \W: 表示非字母、数字、下划线 [^a-zA-Z0-9]
         5. \s: 表示空白符
         6. \S: 表示非空白符
   5. 扩展：案例分析
      1. var pattern = /(google){4,8}/; //匹配分组里的字符串 4-8 次  
         var str = 'googlegoogle';  
         alert(pattern.test(str));
      2. var pattern = /google|baidu|bing/; //匹配三种其中一种字符串  
         var str = 'google';  
         alert(pattern.test(str));
      3. var pattern = /goo\sgle/; //\s 可以匹配到空格  
         var str = 'goo gle';  
         alert(pattern.test(str));
      4. var pattern = /google\b/; //\b 可以匹配是否到了边界  
         var str = 'google';  
         var str2= 'googleaa googlexx google dsdddd';  
         alert(pattern.test(str));
      5. var pattern = /g\w\*gle/; //\w\*匹配任意多个所有字母数字\_  
         var str = 'google';  
         alert(pattern.test(str));
      6. var pattern = /google\d\*/; //\d\*匹配任意多个数字  
         var str = 'google444';  
         alert(pattern.test(str));
      7. var pattern = /\D{7,}/; //\D{7,}匹配至少 7 个非数字  
         var str = 'google8';  
         alert(pattern.test(str));
      8. var pattern = /g[a-zA-Z\_]\*gle/; //[a-z]\*表示任意个 a-z 中的字符  
         var str = 'google';  
         alert(pattern.test(str));
      9. var pattern = /g[^0-9]\*gle/; //[^0-9]\*表示任意个非 0-9 的字符  
         var str = 'google';  
         alert(pattern.test(str));
      10. var pattern = /[a-z][A-Z]+/; //[A-Z]+表示 A-Z 一次或多次  
          var str = 'gOOGLE';  
          alert(pattern.test(str));
      11. var pattern = /g.\*gle/;   
          var str = 'google';   
          alert(pattern.test(str));
      12. var pattern = /^[a-z]+\s[0-9]{4}$/i;  
          var str = 'google 2012';  
          alert(pattern.exec(str));
      13. var pattern = /^[a-z]+/i;   
          var str = 'google 2012';  
          alert(pattern.exec(str));
      14. var pattern = /^([a-z]+)\s([0-9]{4})$/i;   
          var str = 'google 2012';  
          alert(pattern.exec(str)[0]);   
          alert(pattern.exec(str)[1]);   
          alert(pattern.exec(str)[2]);
4. 字符串replace函数
   1. 使用replace替换匹配到的数据
      1. Var pattern = /Good/ig;  
         var str = “good good study!,day day up!”;  
         alert(str.replace(pattern,’hard’));//将Good替换成了hard
      2. var pattern = /(.\*)\s(.\*)/;  
         var str = 'google baidu';  
         var result = str.replace(pattern, '$2 $1'); //将两个分组的值替换输出  
         document.write(result);
      3. var pattern = /8(.\*)8/;  
         var str = 'This is 8google8';  
         var result = str.replace(pattern,'<strong>$1</strong>'); //得到替换的字符串输出  
         document.write(result);
5. 字符串match函数
   1. 使用match方法获取匹配数组
      1. Var pattern = /good/ig; //全局搜索  
         var str = ‘good good study!,day day up!’;  
         alert(str.match(pattern));   
         alert(str.match(pattern).length);
      2. var pattern = /8(.\*)8/; //获取 8..8 之间的任意字符  
         var str = 'This is 8google8, dd 8ggg8';  
         alert(str.match(pattern));  
         alert(RegExp.$1); //得到第一个分组里的字符串内容
6. 字符串search函数
   1. 使用search来查找匹配数据的下标位置
      1. var pattern = /good/ig;  
         var str = ‘good good study!,day day up!’;  
         alert(str.search(pattern)); //查找到返回位置，否则返回-1

第二节：应用

1. 掌握正则的创建方式
2. 掌握正则表达式语言
3. 邮政编码检测
4. 文件格式检测
5. 字符串首尾去空格
6. 邮件格式检测

第三节

1. 手机号检测
   1. / 1(3|5|7|8|4)[\d]{9}/
   2. /1\d{10}/
2. 身份证检测

/^[1-9]\d{5}[1-9]\d{3}(0[1-9] |1[0-2])(0[1-9]|(1|2)\d|3[01])\d{3}(\d|x|X)$/

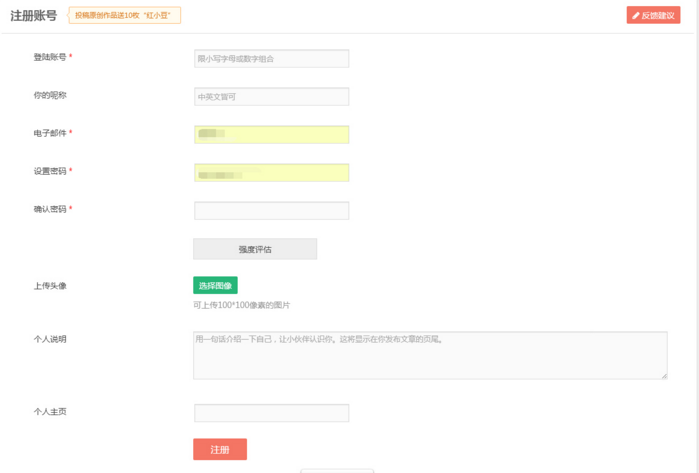
1. 日期检测
   1. Xxxx年xx月xx日 ：/(\d{4}|\d{2})年(\d{1,2})月(\d{1,2})日/

2017年3月29日  
17年3月29日

* 1. 12/25/2016 : (\d{1,2})/(\d{1,2})/(\d{4}|\d{2})
  2. 类似"12:59:59"的标准时间: /[\d]{1,2}:[\d]{1,2}:[\d]{1,2}/
  3. 类似"June 6,2014"的日期 : /([a-z]+) ([\d]{1,2}),([\d]{2,4})/

1. 中文检测
   1. /[\u4e00-\u9fa5]/
2. 用户名检测
   1. 用户名由 3-10位的字母下划线和数字组成。不能以数字或下划线开头。只能已字母开头。允许全部是字母。
      1. ^[a-zA-z][a-zA-Z0-9\_]{2,9}$
3. 字符串过滤(首尾空格)

第四节：应用

1. 表单及AJAX的综合应用

第五节：综合应用

1. 掌握正则的封装
2. 掌握综合表单验证