jQuery:http://www.runoob.com/manual/jquery/

- jQuery是一个JavaScript函数库。
- jQuery是一个轻量级的"写的少，做的多"的JavaScript库。
- jQuery库包含以下功能：

```
- HTML 元素选取
- HTML 元素操作
- CSS 操作
- HTML 事件函数
- JavaScript 特效和动画
- HTML DOM 遍历和修改
- AJAX 异步加载
```

jQuery介绍

> **jQuery是目前使用最广泛的javascript函数库。据统计，全世界排名前100万的网站，有46%使用jQuery，远远超过其他库。**
>
> **微软公司甚至把jQuery作为他们的官方库。**
>
> **jQuery的版本分为1.x系列和2.x、3.x系列，1.x系列兼容低版本的浏览器，2.x、3.x系列放弃支持低版本浏览器，目前使用最多的是1.x系列的**

- **jquery是一个函数库，一个js文件，页面用script标签引入这个js文件就可以使用**

```
<script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
```

下载jQuery

- 官网：http://jquery.com/
- 下载：https://code.jquery.com/

基础语法

-
```
<script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
<script type="text/javascript">
    $(document).ready( function(){
        $("p").click( function(){
            console.log($(this).css("color"))
            $(this).css("color","red");
        });
    })
</script>
<body>
    <p style="color: blue;">我会变红</p >
</body>
```

- Jquery基础语法：Query 语法是通过选取 HTML 元素，并对选取的元素执行某些操作
    - ```
      $(selector).action()
      ```
    - 美元符号定义 jQuery
    - 选择符（selector）"查询"和"查找" HTML 元素
    - jQuery 的 action() 执行对元素的操作

文档就绪事件

- 这是为了防止文档在完全加载（就绪）之前运行 jQuery 代码，jQuery的代码经常会位于一个 document ready 函数中：
    - ```
      $(document).ready(function(){

         // 开始写 jQuery 代码...

      });
      ```

- **注意**

> jQuery 入口函数与 JavaScript 入口函数的区别：
>
> - jQuery 的入口函数是在 html 所有标签(DOM)都加载之后，就会去执行
> - JavaScript 的 window.onload 事件是等到所有内容，包括外部图片之类的文件加载完后，才会执行

jQuery选择器

元素选择器：$("h1")

- jQuery 选择器允许您对 HTML 元素组或单个元素进行操作
    - ```
      <head>
          <title></title>
          <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>

          <script type="text/javascript">
              $(document).ready(function(){
                  $("p").click(function(){
                      $(this).css("color","red");
                  });
              });
          </script>
          <!-- 当某个p被点击时，都会触发该js函数 -->
      </head>
      <body>
          我是第一个p
          我是第二个p
          我是第三个p
      </body>
      ```

ID选择器：$("#id")
      ```

- ID选择器通过HTML元素的ID属性选取指定的元素
- 页面中的元素的id应该是唯一的，在页面中选取唯一的元素需要#id选择器
    - ```
      <head>
          <title></title>
          <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
          <script type="text/javascript">
              $(document).ready(function(){
                  $("#change").click(function(){
                      $(this).css("color","red"); // css属性函数
                  });
              });
          </script>
          <!-- 当Id值为change的p标签被点击时，触发该js函数 -->
      </head>
      <body>
          我是第一个p
          我是第二个p
          我是第三个p
      </body>
      ```

类选择器：$(".class")

- 通过元素的class属性进行查找
    - ```
      <head>
          <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
          <script>
              $(document).ready(function(){
                  $("button").click(function(){
                      $(".pClass").hide();
                  });
              });
          </script>
      </head>
      <body>
          <p class="pClass">
              这是测试内容，点击按钮后，这里的东西会消失
          </p>
          <button>
              点击
          </button>
      </body>
      ```

其他选择器

语法	描述
$(this)	选取当前 HTML 元素
$("*")	选取所有元素
$("p.intro")	选取 class 为 intro 的 <p> 元素
$("p:first")	选取第一个 <p> 元素
$("ul li:first")	选取第一个 元素的第一个 元素
$("ul li:first-child")	选取每个 元素的第一个 元素
$("[href]")	选取带有 href 属性的元素
$("a[target='_blank']")	选取所有 target 属性值等于 "_blank" 的 <a> 元素
$(":button")	选取所有 type="button" 的 <input> 元素 和 <button> 元素
$("tr:odd")	选取奇数位置的 <tr> 元素

jQuery遍历

- 在查找时,可以通过遍历,相对于某些元素进行位置查找,从而获取到想要的元素位置
- 先要分清楚在HTML中元素的级别关系
- ```

      <ul>
          <li> a </li>
          <li> <a> b </a> </li>
      </ul>

  ```
- ```
  - <div> 元素是 <ul> 的父元素，同时是其中所有内容的祖先。
  - <ul> 元素是 <li> 元素的父元素，同时是 <div> 的子元素
  - 左边的 <li> 元素是 <span> 的父元素，<ul> 的子元素，同时是 <div> 的后代。
  - <span> 元素是 <li> 的子元素，同时是 <ul> 和 <div> 的后代。
  - 两个 <li> 元素是同胞（拥有相同的父元素）。
  - 右边的 <li> 元素是 <a> 的父元素，<ul> 的子元素，同时是 <div> 的后代。
  - <a> 元素是右边的 <li> 的子元素，同时是 <ul> 和 <div> 的后代。
  ```

祖先遍历

- **parent**()：返回被选择元素的直接父元素
  - ```
    <head>
        <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>

        <script>
            $(document).ready(function(){
                $("span").parent().css("border","1px solid red");
    ```
    ```

```
        });
    </script>
    // 这里选择到了全部span标签的父标签,li标签,并且设置红色2像素边框
</head>

<body>
    <div>
        <ul>
            <li>
                <span>普通内容1</span>
            </li>
            <br>
            <li>
                <span>普通内容2</span>
                <br>
                <a href="https://www.baidu.com">百度</a>
            </li>
        </ul>
    </div>
</body>
```

- **parents**()：返回被选择元素的全部父元素

```
<head>
    <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
    <script>
        $(document).ready(function(){
            $("span").parents().css("border","1px solid red");
        });
        // 选中了li标签 ul标签 div标签
    </script>
</head>
<body>
    <div>
        <ul>
            <li>
                <span>普通内容1</span>
            </li>
            <br>
            <li>
                <a href="https://www.baidu.com">百度</a>
            </li>
        </ul>
    </div>
</body>
```

  - 该方法也可以指定父元素中的某些元素进行二次过滤

- 比如选择所有父元素中类为**father**的元素

```
<!DOCTYPE html>
<html>
```

```
<head>
    <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
    <script>
        $(document).ready(function(){
            $("span").parents(".father").css("border","1px solid red");
        });
        // 找到了 div标签和ul标签
    </script>
</head>
<body>
    <div class="father">
        <ul class="father">
            <li>
                <span>普通内容1</span>
            </li>
            <br>
            <li>
                <a href="https://www.baidu.com">百度</a>
            </li>
        </ul>
    </div>
</body>
</html>
```

- **parentsUntil**("limit"): 向上查找,直到找到元素界限为止之前的所有父元素
  - ```
    <!DOCTYPE html>
    <html>
    <head>
        <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
        <script>
            $(document).ready(function(){
                $("span").parentsUntil("div").css("border","1px solid red");
            });
            // 找到了 ul标签 第一个li标签
        </script>
    </head>
    <body>

            <ul class="father">
                <li>
                    普通内容1
                </li>
                <br>
                <li>
                    <a href="https://www.baidu.com">百度</a>
                </li>
            </ul>

    </body>
    </html>
    ```

后代遍历
    ```

- **children**()：返回被选元素的直接子元素，不会继续向深层次遍历
    - ```
      <!DOCTYPE html>
      <html>
      <head>
          <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
          <script>
              $(document).ready(function(){
                  $("li").children().css("border","1px solid red");
              });
              // 找到了 span标签 a标签
          </script>
      </head>
      <body>

              <ul class="father">
                  <li>
                      普通内容1
                  </li>
                  <br>
                  <li>
                      <a href="https://www.baidu.com">百度</a>
                  </li>
              </ul>

      </body>
      </html>
      ```
- **find**("*")：返回被选元素的所有符合条件的直接子元素，会继续向深层次遍历
    - ```
      <!DOCTYPE html>
      <html>
      <head>
          <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
          <script>
              $(document).ready(function(){
                  $("ul").find("*").css("border","1px solid red");
              });
              // 找到了 两个li标签 span标签 a标签
          </script>
      </head>
      <body>
          <div class="father">
              <ul class="father">
                  <li>
                      <span>普通内容1</span>
                  </li>
                  <br>
                  <li>
                      <a href="https://www.baidu.com">百度</a>
                  </li>
              </ul>
          </div>
      ```

```
</body>
</html>
```

同胞遍历

- **siblings**()：返回被选元素的所有同胞元素
  - ```
    <!DOCTYPE html>
    <html>
    <head>
        <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
        <script>
            $(document).ready(function(){
                $("span").siblings().css("border","1px solid red");
            });
            // 找到了 p标签 h标签 strong标签
        </script>
    </head>
    <body>

            一个p标签
            一个span标签
            <h3>一个h标签</h3>
            一个strong标签

    </body>
    </html>
    ```
- **next**()：返回被选元素的下一个同胞元素，只返回一个元素
- **nextAll**()：返回被选元素的所有下面的同胞元素，返回所有跟随同胞
- **nextUntil**("limit")：返回直到limit界限的所有跟随同胞，不包含limit
  - ```
    <!DOCTYPE html>
    <html>
    <head>
        <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
        <script>
            $(document).ready(function(){
                $("span").next().css("border","1px solid red");
            });
            // 找到了 h标签
        </script>
    </head>
    <body>
        <div class="father">
            <p>一个p标签</p>
            <span>一个span标签</span>
            <h3>一个h标签</h3>
            <strong>一个strong标签</strong>
        </div>
    </body>
    </html>
    ```

过滤方法

语法	描述
first()	返回被选元素的首个元素
last()	返回被选元素的最后个元素
eq()	返回被选元素中带有指定索引号的元素，索引从 0 开始
filter()	规定一个标准。不匹配这个标准的元素会被从集合中删除，匹配的元素会被返回
not()	返回不匹配标准的所有元素，与filter相反

- ```
  $("h3").filter(".suit");
  // 选择所有类名为suit的h3标签
  ```

- ```
  $("p").not(".green");
  // 过滤所有类名为green的p标签
  ```

判断是否选择到了元素

> **jquery有容错机制，即使没有找到元素，也不会出错，可以用length属性来判断是否找到了元素,length等于0，就是没选择到元素，length大于0，就是选择到了元素**

- ```
  var oh = $("h1");
  alert(oh.length) // > 1 | 0
  ```

jQuery样式操作

获取样式

> **$("selector").css("attr")**

- ```
  $("div").css("color"); // rgb(255, 0, 0)
  $("div").css("border"); // 1px solid rgb(128, 128, 128)
  ```

设置样式

> **$("selector").css("attr","xxx")**
>
> **$("selector").css({"attr1": "xxx", "attr2": "yyy"})**

```
$("div").css("color","blue");
$("div").css({"color":"blue", "border":"1px dashed yellow"});
```

- > **注意**:如果选择器div选择到了多个,在获取信息时,只取第一个

其他操作样式的方式

- **addClass**()：向被选元素添加一个或多个属性
  - ```
    <style type="text/css">
    .redFont{
        color: red;
    }
    .blueBoder{
        border: 1px solid blue;
    }
    </style>
    ```
  - ```
    $(document).ready(function(){
        $("button").click(function(){
            $(".father").addClass("redFont blueBoder");
        });
    });
    ```
  - ```

        这是个div

    <button>按钮</button>
    ```
- **removeClass**()：删除指定的class属性
  - ```
    <style type="text/css">
        .redFont{
            color: red;
        }
        .blueBoder{
            border: 1px solid blue;
        }
    </style>
    ```
  - ```
    $(document).ready(function(){
        $("button").click(function(){
            $(".redFont").removeClass("blueBoder");
        });
    });
    ```
  - ```
    <div class="blueBoder redFont">
        这是个div
    </div>
    <button>按钮</button>
    ```
- toggleClass()：设置或移除被选元素的一个或多个类进行切换
- > 该方法检查每个元素中指定的类;
  >
  > 如果不存在则添加类，如果已设置则删除之。这就是所谓的切换效果

```
.redFont{
color: red;
}
```

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").toggleClass("redFont");
    });
});
```

```
<p class="redFont">第一段文字</p>
<p class="redFont">第二段文字</p>
<p>第三段文字</p>
<p>第四段文字</p>
<h3>这是h3标题</h3>
<button>按钮</button>
```

jQuery事件

鼠标事件

- **click**：点击

```
$(selector).click(function(){
    ...
});
```

- **dbclick**：双击

```
$(selector).dbclick(function(){
    ...
});
```

- **mouseenter**：穿过某元素

```
$(selector).mouseenter(function(){
    ...
});
```

- **moseleave**：鼠标离开

```
$(selector).mouseleave(function(){
   ...
});
```

- **hover**：鼠标悬停

```
$(selector).hover(function(){
    ...
});
```

键盘事件

- **keydown**：键按下的过程
  - ```
    $(selector).keydown(function(){
        //
    });
    ```
- **keypress**：键被按下
  - ```
    i = 0
    $(document).ready(function(){
        $("input").keypress(function(){
            $("span").text(i+=1);
        });
    });
    // 在input表单中按了多少次
    ```
  - ```
    <input type="text">
    按键的次数: 0
    ```
- **keyup**：键被松开
  - ```
    ￥(selector).keyup(function(){
       //
    });
    ```

表单事件

- **submit**：表单提交
  - ```
    $("form").submit(function(){
       alert("表单被提交");
    });
    ```
- **change**：表单修改
  - ```
    $("input").change(function(){
        alert("文本已被修改");
    });
    ```
  - > **注意：**当用于 **select** 元素时，**change** 事件会在选择某个选项时发生。当用于 **text field** 或 **text area** 时，**change** 事件会在元素失去焦点时发生
- **focus**：光标选中
  - ```
    $("input").focus(function(){
        $("label").fadeOut(2000);
    });
    // 当输入框被选中时，label标签淡出 fadeOut
    ```
    ```

```
<label>看看这个文字</label>
<input type="text">
```

- **blur**：光标离开
  - ```
    $("input").blur(function(){
        alert("输入框失去了焦点");
    });
    ```

文档/窗口事件

- **load**：指定元素已加载，
  - > load() 方法在 jQuery 版本 1.8 中已废弃
  - ```
    $("img").load(function(){
        alert("图片已载入");
    });
    ```
- **resize**：当调整浏览器窗口大小时，发生 resize 事件
  - ```
    $(window).resize(function(){
       $("span").text(i+=1);
    });
    ```
  - ```
    <span>0</span>次
    ```
- **scroll**：当用户滚动指定的元素时，会发生 scroll 事件
  - > scroll 事件适用于所有可滚动的元素和 window 对象（浏览器窗口）
  - ```
    $("div").scroll(function(){
        $("span").text(x+=1);
    });
    ```
- **unload**：
  - > unload() 方法在 jQuery 版本 1.8 中已废弃，在 3.0 版本被移除
  - ```
    $(window).unload(function(){
        alert("Goodbye!");
    });
    // unload() 方法只应用于 window 对象
    ```

获取内容和属性

- **text()**：设置或返回所选元素的文本内容
- **html()**：设置或返回所选元素的内容（包括 HTML 标记）

```
$("button").click(function(){
    console.log($("p").text()); // 这是个p标签
    console.log($("p").html()); // 这是个<b>p</b>标签
});
```

```
<p>这是个<b>p</b>标签</p>
 <button>按钮</button>
```

- **val()**：设置或返回表单字段的值

```
$("button").click(function(){
    console.log($("input").val());
});
```

```
<input type="text" value="123">
<button>按钮</button>
```

- **attr("src")**：获取属性

```
$("button").click(function(){
     console.log($("a").attr("href")); // https://www.baidu.com
});
// 获取a标签的href属性
```

```
<a href="https://www.baidu.com">百度</a>
<button>按钮</button>
```

改变内容和属性

- **text**()
- **html**()
- **val**()

```
$("button").click(function(){
    $("#test1").text("Hello world!");
});
$("button").click(function(){
    $("#test2").html("<b>Hello world!</b>");
});
$("button").click(function(){
    $("#test3").val("RUNOOB");
});
```

- **attr**()：

```
$(document).ready(function(){
    $("button").click(function(){
      $("a").attr("href","https://www.sougou.com");
    });
});
// 同时设置多个属性

$(document).ready(function(){
    $("a").attr({
        "href" : "https://www.sougou.com",
        "class" : "sougou"
    });
});
```

- ```
  <a href="https://www.baidu.com">百度</a>
  <button>按钮</button>
  ```

jQuery效果

显示|隐藏

- **hide**(speed:[slow|fast], *callback*)：隐藏元素
- **show**(speed:[slow|fast], *callback*)：显示元素
  - ```
    $(document).ready(function(){
        $("#hide").click(function(){
            $("p").hide("slow");
        });

        $("#show").click(function(){
            $("p").show();
        });
    });
    ```
  - ```
    这是一段文字
    <button id="hide" >hide</button>
    <button id="show" >show</button>
    ```
- **toggle**()：显示被隐藏的元素，并隐藏已显示的元素
  - ```
    $("button").click(function(){
      $("p").toggle();
    });
    ```
  - ```
    这是一段文字
    <button>按钮</button>
    ```

淡入|淡出
    ```

- **fadeIn**()：淡入已隐藏的元素
  - ```
    $(document).ready(function(){
        $("p").hide() // 隐藏元素
        $("button").click(function(){
            $("#p1").fadeIn();
            $("#p2").fadeIn("slow");
            $("#p3").fadeIn(3000);
        });
    });
    ```
  - ```
    <p id="p1">这是一段文字</p>
    <p id="p2">这是一段文字</p>
    <p id="p3">这是一段文字</p>
    <button>按钮</button>
    ```
- **fadeOut**()：淡出可见元素
  - ```
    $(document).ready(function(){
        $("button").click(function(){
            $("#p1").fadeOut();
            $("#p2").fadeOut("slow");
            $("#p3").fadeOut(3000);
        });
    });
    ```
  - ```
    <p id="p1">这是一段文字</p>
    <p id="p2">这是一段文字</p>
    <p id="p3">这是一段文字</p>
    <button>按钮</button>
    ```
- **fadeToggle**()：在 fadeIn() 与 fadeOut() 方法之间进行切换
  - ```
    <!DOCTYPE html>
    <html>
    <head>
        <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
        <script>
            $(document).ready(function(){
                $("button").click(function(){
                    $("#p1").fadeToggle();
                    $("#p2").fadeToggle("slow");
                    $("#p3").fadeToggle(3000);
                });
            });
        </script>
    </head>

    <body>
        这是一段文字
        这是一段文字
        这是一段文字
    ```
    ```

```
    <button>按钮</button>
</body>
</html>
```

滑动

```
- slideDown(speed,callback) // 向下滑动元素
- slideUp(speed,callback) // 向上滑动元素
- slideToggle(speed,callback) // 在 slideDown() 与 slideUp() 方法之间进行切换
// 可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。
// 可选的 callback 参数是滑动完成后所执行的函数名称
```

jQuery动画

- $(**selector**).**animate**({**params**}, **speed**, **callback**);
  - params：可选参数；动画形成的属性，要改变的样式值，写成字典
  - speed：可选参数；动画持续的时间，单位毫秒
  - callback：可选参数；动画完成后执行的函数名称
- **注意**：当使用 animate() 时，必须使用 Camel 标记法书写所有的属性名，比如，必须使用 paddingLeft 而不是 padding-left，使用 marginRight 而不是 margin-right，等等、

  同时，色彩动画并不包含在核心 jQuery 库中
- ```
  $(document).ready(function(){
      $("button").click(function(){
          $("p").animate(
              {
                  left: "+=300px",
                  fontSize: "100px",
              },
              "slow",
              function(argument) {
                  alert("动画完成")
              }
          )
      })
  });
  ```
- ```
  <p style="position: relative;">这是一段文字</p>
  <button>按钮</button>
  ```

jQuery获取表单数据

- 单选框：**radio**
  - ```
    $("input[type=radio]:checked").val()
    // 当一个页面有两个radio时
    $("input[name='gender']:checked").val();
    // 通过name分类进行过滤 checked代表选中元素
    ```
    ```

- 多选框：**checkbox**
  - ```
    var res = new Array;
    var olike = $("input:checkbox[name='like']:checked").each(function(){
        res.push($(this).val())
    })
    // each() 方法规定为每个匹配元素规定运行的函数
    ```
- 下拉菜单：**select**
  - ```
    $("select[name='city']").val();
    ```

jQuery正则

正则规则

- **\d**：匹配一个数字
- **\D**：匹配一个非数字，即除了0-9
- **\w**：匹配字母、数字、下划线
- **\W**：匹配非单词字符，等价于[^A-Za-z0-9_]
- **\s**：匹配一个空白符
- **\S**：匹配一个非空白符
- **\b**：匹配单词边界
- **\B**：匹配非单词边界
- **.**：匹配任意字符

开头结尾

- **^**：开头匹配
- **$**：结尾匹配

正则次数

- **?**：出现零次或一次；最多出现一次
- **+**：出现一次或多次；至少出现一次
- ***:**：出现零次或多次；任意次
- **{n,m}**：出现n-m次

匹配范围

- **[a-z]**：匹配任意小写字母
- **[0-9]**：匹配任意数字

正则语法

- > var regex = /规则/参数
  - **/\d+/**：匹配所有数字

创建正则表达式

```
var regex = /[a-z]+$/ ; // 任意小写字母结尾
var regex = new RegExp()
```

- ```
  - g: 全局的匹配(匹配多次; )
  - i: 大小写不敏感匹配(忽略字符大小写)
  - m: 多行(^和$能匹配行结束符)
  ```

捕获型|非捕获型

- ```
  (?:(\d+)) // 非捕获型分组 结果并不会复制所匹配的文本
  (\d+) // 其中的\d+ 是捕获型分组，结果会放入最终的匹配结果中
  ```

正则判断方法

- **regex.exec**：将匹配到的文本保存到一个结果数组
  - 本身的表达式是一个包含分组匹配的，那么使用exec可以将每个分组保存到数组结果的依次位置中
- **regex.test**：匹配成功返回true，否则返回假

常用正则

- ```
  regAccount = /^\w{6,20}$/; // 字母数字下划线，用户名验证6-20位

  regEmail = /^[a-zA-z1-9]{8,20}@(163|126|qq)\.(com|cn)$/  // 邮箱验证

  regPass = /^[\w!@#$%^&*]{6,20}$/ // 密码验证

  regPhone = /^1[3,4,5,7,8]\d{9}$/ // 手机号验证
  ```

- ```
  sStr = "123456"
  regex = /\d+/
  alert(regex.text(sStr)); // true
  ```

AJAX

- **$.ajax([settings])**：执行ajax请求
  - ```
    $(document).ready(function(){
        $("button").click(function(){
            $.ajax({
                url: "/ajax/",// 发送请求的地址
                type: "POST", // 请求方式 默认get
                data: { // 要提交的数据
                    username:$("#username").val(),
                    // "csrfmiddlewaretoken":$("[name='csrfmiddlewaretoken']").val(),
    // django的csrf防跨站请求伪造令牌
                },

                beforeSend: function(){ // 发送请求前运行的函数
                    $("button").attr({ disabled: "disabled" })
    ```
    ```

```
            },

            success: function(result){ // 请求成功后的回调函数
                // 服务器返回根据datatype设定的类型数据
                $("#result").text(result)
                console.log(result)
            },
            complete: function(){ // 请求完成时运行的函数
                // 在请求成功或失败之后均调用，即在 success 和 error 函数之后
                $("button").removeAttr("disabled");
            },
        })
    });
});
```

- django服务端的功能：
  - ```
    # views.py
    def ajax(request):
        if request.is_ajax(): # 判断是否是ajax请求
            print(request.POST)
            print(request.body)
            username = request.POST.get("username") #获取ajax传递来的数据中的 username对应的数据
            return  HttpResponse(json.dumps("这是我获取到的:%s" % (username,)))
        response = render(request,"ajax.html")
        return response
    ```
    ```