01_Jquery入门

- Jquery是什么?
  JQuery是一个JS的类库文件 什么是JS类库文件?别人写好的JS代码,我们拿过来调用的这些JS代码被称作JS库文件.
- Jquery的作用
  在JS的基础部分以及JS操作DOM和JS实现Ajax等过程中.暴漏了不少问题,例如 复杂的DOM操作和烦冗的ajax操作等.为了简化JS的开发,各种JS库诞生了.各种JS库都封装了很多预定义的对象和实用函数,能够帮助使用者建立非常漂亮的浏览器中的页面,并且兼容各大浏览器.
- 自定义js库
  - 需求:自定义js库,修改div的标签内容

```
//自定义js脚本
function $(id){
    var dm = document.getElementById(id);
    dm.html = function(arg){
        if(null == arg){
            return dm.innerHTML;
        }else {

            dm.innerHTML = arg;
        }

    }
    return dm;
}

<script type="text/javascript" src="myjquery.js"></script>
<script type="text/javascript">
    window.onload = function() {
        //使用自定义js库
        $("div1").html("hello");
    }
</script>
</head>
<body>
    <div id="div1"></div>
</body>
```

- 了解Jquery
  - JQuery是2006年8月诞生的一个开源项目,现在Jquery团队主要包括核心库,UI和插件开发等.开发人员,凭借简洁的语法和跨平台的兼容性,极大地简化了Javascript开发人员遍历HTML文档,操作DOM,处理事件,执行动画和开发Ajax的操作.其独特而又优雅的代码风格改变了javascript程序员的设计思路和编写程序的方式.
- 入门案例
  - 需求:使用Jquery修改div标签内容

- 步骤
  - 1,在页面引入jquery脚本文件
  - 2,使用Jquery操作元素
- 代码实现

```
//1,在页面引入jquery脚本文件
<script type="text/javascript" src="jquery-1.4.2.js"></script>
<script type="text/javascript">
    window.onload = function(){
        //2,使用Jquery操作元素
        $("#div1").html("fuck");
    }
</script>
</head>
<body>
<div id="div1"></div>
</body>
```

02_Jquery对象和dom对象

- 什么是dom对象
  文档对象模型,每一份DOM都可以表示成一颗树.
- 什么是Jquery对象
  Jquery对象就是通过Jquery包装DOM对象后产生的对象,可以使用Jquey方法
- Jquery对象和dom对象的相互转换
  Jquery对象不能使用dom对象的方法,dom对象不能使用Jquery对象的方法.但是它们可以相互转换.
  - Jquery对象转dom对象

```
$("div").get(0);
$("div")[0];
```

  - dom对象转Jquery对象

```
 $(dm);
```

03_Jqeury选择器

- 概念
  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.如果能熟练地使用选择器,不仅能简化代码,而且可以达到事半功倍的效果.

- 基本选择器
  - 格式:

```
ID选择器   $("#id");
类选择器       $(".className");
标签选择器     $("div");
通配符选择器   $("*");
```

- 层次选择器
  - 格式

```
层次选择器:从父子关系和兄弟关系来进行选择页面节点
$("a b")   :a节点的所有后代节点b都被选中
$("a > b") :a节点的所有子节点b都被选中
$("a + b") :a节点之后的第一个兄弟节点b
$("a ~ b") :a节点之后的所有兄弟节点b
```

- 过滤选择器
  - 基本过滤选择器
    从位置的角度来对页面的标签进行过滤选择

```
 $("tagName:first"):选择第一个tagName标签
 $("tagName:last") :选择最后一个tagName标签
 $("tagName:eq(2)"):选择脚标为2的tagName标签
 $("tagName:gt(2)"):选择脚标大于2的tagName标签
 $("tagName:lt(2)");选择脚标小于2的tagName标签
```

  - 内容过滤选择器:
    节点值是否为空
    节点值是否包含指定的字符串

```
$("tagName:empty")          :tagName节点中没有子元素(文本元素,其他子元素)
$("tagName:contains('aaa')"):tagName节点是否包含指定字符串
```

  - 属性过滤选择器
    从节点的属性来过滤筛选节点:有无属性,属性值等于,不等于,包含,多重过滤

```
$("tagName[id]")           :tagName节点是否包含id属性
$("tagName[id='cc']")      :tagName节点属性值是否为cc
$("tagName[id!='cc']")     :tagName节点属性值是否不为cc
$("tagName[title*='cc']")  :tagName节点属性值是否包含cc
$("tagName[title*='cc'][name='ee'][id!='ff']"):tagName节点属性值title是否包含cc,属性值
name是否为ee,id属性是否不是ff
```

  - 子元素过滤选择器
    选择父元素下的子元素(第1个,最后1个,第几个子元素)

```
$("tagName :first-child")   :tagName节点下的第一个子节点
$("tagName :last-child")    :tagName节点下的最后一个子节点
$("tagName :nth-child(2)")  :tagName节点下的第二个子节点
```

04_Jquery的dom操作

- 内容操作
  - html()
  - text()
  - val():设置/获取输入框中的内容
- 属性操作
  - attr(): 获取/设置元素的属性
  - removeAttr():删除属性

05_Jquery范例一

- 需求:
  - 获取第2个li节点的title属性
  - 获取第2个li节点的文本内容

```
<ul>
  <li title='苹果11'>苹果</li>
  <li title='橘子22'>橘子</li>
  <li title='菠萝33'>菠萝</li>
</ul>

window.onload = function(){
    //脚标为1的li,也是第二个li
    alert($("li:eq(1)").attr("title"));
    //ul标签下的第二个li
    alert($("ul :nth-child(2)").attr("title"));

    alert($("li:eq(1)").html());
    alert($("ul :nth-child(2)").html());
}
```

06_Jquery事件

- 需求1:监听页面加载

```
//dom方式
window.onload=function(){
    alert("页面加载完成");
}
//Jquery方式
$(document).ready(function(){
    alert("页面加载完成");
});
```

```
//Jquery简写方式
$(function(){
    alert("页面加载完成");
});
```

- 需求2:输入框获取焦点时,输入框内容置为空;失去焦点时,输入内容设置为"请输入用户名"

```
<script type="text/javascript">
    $(function () {

        $("#username").focus(function () {

            var content = $("#username").val();
            if (content == "请输入用户名") {
                $("#username").val("");
            }
        });
        $("#username").blur(function () {
            var content = $("#username").val();
            if (content == "") {
                $("#username").val("请输入用户名");
            }
        });
    })
</script>
</head>
<body>
    <input type="text" id="username" value="请输入用户名">
</body>
```

07_Jquery的遍历

- js的遍历方式

```
var persons = $("ul li")
for (var i = 0; i < persons.length; i++) {
    console.log(persons[i].innerHTML);
}
```

- jquery的遍历方式
  - jq对象.each(callback)

```
jquery对象.each(function(index,element){
});
```

    - index:就是元素在集合中的索引
    - element：就是集合中的每一个元素的js对象
    - this：集合中的每一个元素对象(js对象)
    - 回调函数返回值：
      - true:如果当前function返回为false，则结束循环(break)。

        - false:如果当前function返回为true，则结束本次循环，继续下次循环(continue)

```
persons.each(function (index, element) {
    console.log($(element).html());
})
```

    - $.each(object, [callback])

```
$.each(persons,function (index, element) {
    console.log($(this).html())
})
```

08_Jquery的异步请求

- get方式

```
$.get(url,data,callback,type):使用GET 来加载远程数据
url:请求的路径
data:请求的参数
callback: 当数据被加载时，所执行的函数
type: 被返回的数据的类型 (html,xml,json,jasonp,script,text)
```

- post方式

```
$.post(url,data,callback,type):使用POST 来加载远程数据
url:请求的路径
data:请求的参数
callback: 当数据被加载时，所执行的函数
type: 被返回的数据的类型 (html,xml,json,jasonp,script,text)
```

- ajax方式

```
$.ajax({
        url:url,
        data:data,
        success:success,
        dataType:dataType,
        type:type
});
url:请求的路径
data:请求的参数
success:服务器响应成功
dataType:响应的数据的类型
type:请求方式
```