# Js

**注：基于****<script></script>标签下的开发! 书写范围可以在内部，或者外部写入后导入！**

1. 数组的定义 var i =[];
2. 函数的定义：fanction f() {};
3. 在按钮上直接书写事件：onclik=””;
4. 伪连接< href =’javascript:执行的事件’>;
5. 在js中书写html :document.write(“内容可以是html, 标签或者字符”);
6. 获取页面的元素及加入点击事件：

例:var btn = document.getElementById(“元素id”);

btn.onclick = function(){

btn.innerHtml=”显示字符”；

}

1. 弹窗：
2. alertly(“显示的字符！”)；提示类型；
3. confirm(“显示字符！”)；选择类型（有返回值；true/false）;
4. 数据类型的转换：
5. 装化为数字：parselnt（）;
6. 转化为实数：parsFloat（）；
7. 输入框弹窗：prompt();
8. 判断是不是数字：isnan（）；返回值（true/false）；
9. 对象的使用：**var *ab***={**a**:**"1"**,**arr**:[1,2,3,3]};  
    **for**(**var *abc*** in ***ab***) {  
    **document**.write(***abc***+**":"**,***ab***[***abc***]);  
    }
10. 在js中向html中添加元素：

建立组件：var nb=document.createEliment(组件);建立组件

Nb.innerHTML(“设置显示文本”)

**document**.**body**.appendChild(nb);添加件；

12.数组的操作：

Var arr=【】；

Arr.push();向数组中添加一个值；

Arr.splicel(什么地方开始，删除一个值，添加的值)；向数组中添加一个值；

数组的排序在使用之前写：arr.sort(function(n1,n2){return n1-n2/n2-n1});

13.字符串的操作：

a.截取字符串：substr(从多少开始，截取多少个)；

substring(从多少开始，到多少结束);

charAt(截取出该参数下标的1个字符)；

b.查找字符：indexof(“查找的字符”，从哪里开始查)

c.转化为整数parseInt();

14.时间的操作：var date=new Date();

console.log(date)中国标准时间；

console.log(***date***.toLocaleString())本地时间：例：2017/1/9 下午8:49:59

var time =date.getTime+1000\*60\*60\*24;

date.setTime(time);

console.log(***date***.toLocaleString()) 得到明日现在的时间；

15.随机数console.log(Math.random)取得一个0到1间的随机数；

16.窗体加载完毕函数： window.onload=function{

函数内的代码会在最后执行；

}

17.连接：loction ”地址”；

18.下拉列表：

<select id=”sel”></select>

Var sel=docmentdocument.getElementById(“sel”);

Sel.options.length=0; 列表值得清空；

Sel.options.add(new option(值，id))在js中添加一个option；】

下拉列表的点击事件：.onchange;

19.表格table的使用：

设置边框：table.border=1;

**var** table=***$***(**"#table"**).get(0); 先获取table对象

添加行：var row=table.insertRow();

添加列：row.insertcell.innerHTML=”<button></button>”

删除行：table.deleterow(行数)；

删除单元格：table.rows[行数].deletecell(列数)；

修改单元格：table.rows[行数].Cell[列数].innerHTML=””;

20.Eval(使该字符被当成代码来执行！)；

21. var a=setInterval(function(){命令},time)每time执行一次代码！

clearInterval(a);停止setInterval；

22.常用函数：

isEmptyObject() 判断对象是否为空

$(‘form’).submit() 执行表单提交

***Jquery***

1. 选择器：a.$(“可以是id,标签，样式名称”)；

例：$(“button”).html(“你被修改了”),css({color=”red”});

b.分组选择$(“a1,a2”);选择了a1和a2;

c.包含选择：(“a1 a2 ”);a1里面的a2;

d.子元素选择：(“a1>标签 ”)；a1下的一个元素；

e.属性选择器： $(“button[id]”)拥有id的按钮；

$(“button[id=xx]”)id等于xx的按钮；

$(“button[id^=x]”)id从X开始的按钮；

$(“button[id$=x]”)id从X结束的按钮；

$(“button[id\*=x]”)id包含X的按钮；

1. 筛选：

a.$(“button:eq(0)”) / $(“button”).eq(0);索引为1的按钮；

b. $(“button”).get(0); 获取索引为1的按钮；

c.$(“h1”).children(选择器该处是一个，可以为空);h1下的子元素；

d.$(“h1”).find(“button”);h1下的所有按钮包括孙级；

e. .next();获取同级的下一个元素标签；

f.parent();获取父级标签；

g. parents(null/标签);获取全部父级或某一个父级；

h. $(div).find(‘div’) 在该元素下的子孙级查找该元素

3.class样式的操作：addclass(“添加class”),removeclass(“删除class”)，toggleclass(“有就删除，没有就添加”)；

4.滚动条的操作：$(“body”). scrollTop()/scrollLeft()

5.删除子级元素empty();

6.组件位置的偏移：

var x= $(‘a’).offset() //offset下有方法top和left;

x.left //获取位置

$("p").offset({top:100,left:0});设置组件的偏移位置，但是原来的位置任然占位，

7．组件属性的添加 :

例：$(a).aattr(“属性名”,值)；

删除属性：removeAttr();

8. appendTo（）方法在被选元素的结尾（仍然在内部）插入指定内容：

例1：创建1个元素放入：$(‘<button>,{html(‘text’)}’). appendTo (指定元素)

例2：克隆一个自己放入：$(this).clone. appendTo (指xzas定元素)

例3：克隆事件自己放入：$(this).clone(true). appendTo (指定元素)

相同功能的几个方法：

Append/prepand()/prepandTo()/insetAfter()/insetBefore()/remove()

9.id的标准写法：data-id=””获取：$(this).data(‘id’)；

***事件***

1.希望所有按钮都有事件，包括js生成的按钮：

列：$(‘body’).on(‘click’,’button’,function{

事件内容！！！！！！

})

2.希望按钮只执行1次事件：

$(‘button’).one(‘click’,function{

事件内容！！！！！！

})

3.移入移出事件：

$(a).hovar.(function{

移入事件!!

},function{

移出事件！

})

1. input 失去焦点监听blur()

input 选取内容时的监听select()

1. 其他事件：

移出事件：mouseout()

移入事件：mouseover()

移动事件：mousemove()

右键鼠标事件：contextmenu()

$(body).contextmenu(function(e){

e.pageX ,e.pageY //(右键坐标的获取)// })

1. 默认事件：阻止事件的发生/阻止事件的扩散：return fales;

例：$(‘a’).click(function(){

**Return** confirm(**"text"**)sh

})

1. 键盘事件：

1、keydown()   
keydown事件会在键盘按下时触发.   
2、keyup()   
keyup事件会在按键释放时触发,也就是你按下键盘起来后的事件   
3、keypress()   
keypress事件会在敲击按键时触发,我们可以理解为按下并抬起同一个按键

Anima(动画)

1.基础动画的写法:

$(**'div'**).**animate**({**width**:$(**'div'u**).**width**()>100?80:100},100)

1. script内置动画:
2. show 打开
3. hide 关闭
4. toggle 切换
5. slidToggle 卷动切换
6. fadeIn 淡入
7. fadeout 淡出
8. dadeTo 改变透明度

2.动画模拟器（驱动jquery.easing-1-3.js）模拟事物的运动轨迹

例：.animate({width:btn.width()>100?80:1000},1000,”方法名”)

也可设置所有动画为该方法：

例：jQuery.easing.def=”方法名”

4.基于class(导入jquery-ui.min.js)的动画：

例1：$(‘.a’).toggleclass(‘class’,1000)

例2：$(‘.a’).toggle(‘动画样式’,1000)

Ui中的动画有：

Bling/clip/drop/explode/fold/paff/slide/size/pulsate

1. 块级元素的拖动：draggable()

Hide(0,())

jsp

Servlet的操作：

request.setCharacterEncoding("btf-8");

过滤器（Filter接口）

重写三个方法

在方法doFilter（可以新建Filter文件系统会在web.xhl里面自动生成配置文件）里面实现过滤器

命令如下：chain.doFilter(request, response);

注修改web.xhl下<url-pattern>标签内容为：

<url-pattern>/\*</url-pattern>// 该路径的所有文件都会经过顾虑

Jsp(java server pages/服务器端运行的网页)的应用：

Jsp类编码的修改：preferences>web>jsp Files【encoding:utf-8】

* + - 1. 在webcontent建立jsp文件(test.jsp)
      2. Liset<string>liset=new ArraLiset<>();
      3. List.add(rs.getstring(‘name’));
      4. req.setAttribute(“data”,liset);
      5. req.getRequestDispatcher(‘/test.jsp’).forward(req,resp) //请求转发！！

///以上是servlet里的操作

Jsp内的操作：

<%=request.getAttribute%>

Jsp中java的嵌套：

例： <%**for** (**int** a=0;a<10;a++){ %>

<h1> 测试文字 <%=a %></h1>

<%} %>

1.Jsp中函数的写法：

<%=request.getAttribute("data") %>jsp源代码的地址查看命令//》》》 work\Catalina\localhost\txt\org\apache\jsp路径！！！

<%! //需在此添加感叹号！！方可写入函数

**void** f(){}

%>

2.调用外部文件：include

例：<%@ include file=*"aa.jsp"* %>

3.内置对象：

Out.printIn(a)//用于输出对象在网页中

jsp作用域：

page (pageContext)//当前页面共享数据

request //同一个请求到同一个服务器，共享数据

session //同预览器共享数据，预览器关闭后数据消失。

application //整个服务器数据共享！。服务器关闭数据消失。

4. (el表达式)数据的获取：

页面导入：

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"* %>

例1：

request.setAttribute(‘str’,’测试文字’)

直接使用：$( str )获取数据

例2：request.setAttribute(‘opt’,newlogin(1,”aa”,”bb”)) ;//login是一个封装类里面封装了get,set方法，分别是id,name,password.

使用$( opt .id)的方法获取id

例3：

当作用域属性方法名字相同时，该方法会从小到大的顺序获取值

可以使用作用域的隐藏属性获取想要的值(

requestScope/在该作用域后加Scope

)

如: request.setAttribute(‘str’,’测试文字’)

Pagecontext.setAttribute(‘str’,’测试文字’)

${requestScope.str}

5.状态的保持param.name：

例：<input name=”name” value=”${ param.name}”>请求失败时，输入框里数据不会消失！

6.容器路径获取：${ pageContext.request.contextPath }服务器根路径；

7.运算符：empty()判断字段是否不为空返回trues和false

8.el表达式 标准标签的写法

例：

<% pageContext.setAttribute("aa", **new** String[]{"asd","dsa"} ); %>

<c:forEach items=*"* ${aa }*"* var=*"v"*> //便利数组

${ v }<br/>

</c:forEach>

<c:if test=*"*${1==1 }*"*> //if

相等

</c:if>

<c:forEach begin=*"1"* end=*"10"* var=*"i"*> //for

<h4>世界你好 ${ i }</h4>

</c:forEach>

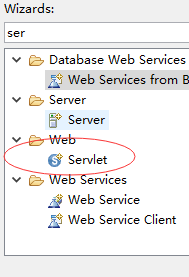
9.对象history(控制台命令)

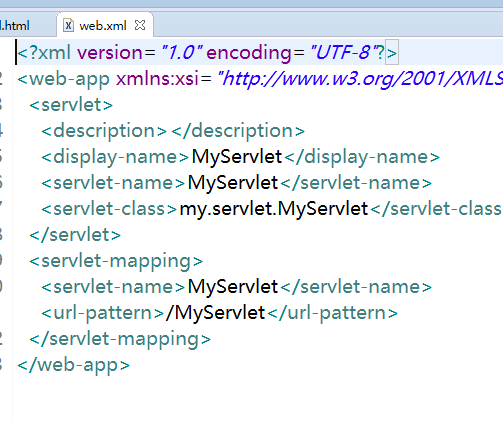
1.history.back网页后退；

2. history.forward()网页前近；

10.jsp访问的servlet服务器必须是系统建立的servlet文件

如下（应为系统会自动在web.xhl里生成访问配置）：





10.session对象在jps中的应用：

<%

String ps= (String)session.getAttribute("password");

System.out.println(ps);

**if**(ps!=**null**){ %>

<div class=*"fd f"* >账号或则密码错误!</div>

<% } %>

//注在jsp中session的调用不用创建直接调用；

并且只能String ps= (String)session.getAttribute("password");

强制装还；原因如注释：

Ajax

数据交互：

例： $（button）.click(function{

$.ajax({

url:”servlet在web.xml文件里面配置的地址，注不加斜杠”，

type(‘post’),

data:{name:$(“input[name=name]”).val()}, //数据的发送

success:function(data){

$(div).html(data)

}, //servlet响应数据的取出

error:function(){

alert(“如果访问错误的提示”)

}

})

})

对象数据的传递：导入包：







建立对象：

**private** ObjectMapper oMapper= **new** ObjectMapper();

数据的响应：response.setContentType("application/json: charset=utf-8");

response.getWriter().println(oMapper.writeValueAsString(users));

数据交互的优化：1.serialize: 例： data:$(fome).serialize(),//系统自动提交所有fome里面的表单！ //提交全部

$.post('地址',$('form').serialize(),**function**(data){

console.log(data);

},”json”)

$.postJSION('地址',$('form').serialize(),**function**(data){

console.log(data) })

ajax默认属性的设置:

$.ajaxSetup({

dataType:"json",

error:("请求数据失误！")

});

Php对ajx的支持

Ajax的验证插件：

导入以下包：//必须是该顺序！！！

<script type=*"text/javascript"* src=*"js/jquery-1.11.1.min.js"*></script>

<script type=*"text/javascript"* src=*"js/jquery.validate.min.js"*></script>

<script type=*"text/javascript"* src=*"js/additional-methods.min.js"*></script>

<script type=*"text/javascript"* src=*"js/messages\_zh.min.js"*></script>

例1：

<script>

$(**function** () {

$('form').validate({

debug:**true**, ///实验的时候用，阻止表单的发送！！！！

messages:{

a:{ ///表单的名字

pattern:"文本" //针对该表单下pattern规则，不符合时的提示

}

}

});

})

</script>

例2：equalTo校验方法 //校验2个输入框的输入内容必须相同

<form>

<input required pattern=*"123\d{3}"* name=*"a"*>

<input required equalTo=*"input[name=a]"* name=*"b"*>

<button type=*"submit"*>提交</button>

</form>

例3： number //数字校验规则

<script>

$(**function** () {

$('form').validate({

debug:**true**, ///实验的时候用，阻止表单的发送！！！！

rules:{

number:{

numble:ture

}

},

});

})

</script>

例3 : **remote =“servlet的配置路径”** //远程校验

<**form** >  
 <**input** **required name="a" remote="s"**>  
 <**input required equalTo="input[name=a]" name="b"**>  
 <**button type="submit"**>提交</**button**>  
</**form**>

例4：制作自己的校验器：

$(function () {

jQuery.validator.addMethod("my", function(value, element, params) {

return value=="888"

},"你错了");

$('form').validate({

debug:true,

rules:{

a:{

my: true

}

},

showErrors: **function** () {

**for**(**var** i=0;i<**this**.errorList.length;i++){

**var** e=**this**.errorList[i];

$(e.element).next('span').html(e.message)

}

**for** (**var** i=0;i<**this**.successList.length;i++){

**var** e=**this**.successList[i];

$(**this**.successList[i]).next('span').html("ok")

}

},

});

})

/// showErrors获取的数据



Ajax传值的校验判断！！

$(**"button"**).click(**function** () {  
 **if**($(**'form'**).valid()){  
 alert(123123131231)  
 }**else**{  
 alert(**"dsad"**)  
 };  
 })

文件的获取：

@MultipartConfig

Part part =request.getPart("file");

part.write(getServletContext().getRealPath(“文件在服务器的名称”));

form文件上传：

<form action=*"s"* method=*"post"* enctype=*"multipart/form-data"*>

<input name=*"filename"* value=*"01.jpg"*><br>

<input name=*"file"* type=*"file"*><br>

<button type=*"submit"*>上传</button>

</form>

本地文件的预览：

$(**'input[name=file]'**).**change**(**function** () {  
 **var** fr= **new FileReader**()  
 fr.**onload**=**function** () {  
 $(**'img'**).attr(**"src"**,fr.**result**)  
 }  
 fr.readAsDataURL(**this**.**files**[0]);  
 })

Ajax上传文件：

$('button').click(**function** () {

**var** formData= **new** FormData($('form').get(0))

formData.append(“名字“，”内容“)

$.ajax({

url:"s",

type:'post',

processData:**false**,

contentType:**false**,

data:formData,

success:**function**(data){

console.log(data)

}

})

})

错误调试：JUnit Text casa 类配置选项：class under text:为要测试的类；

例：

**private** User user=**new** User();

@Test

**public** **void** testQuery() **throws** Exception {

**if** (user.query(**false**)==**null**) {

**throw** **new** Exception("这不能为空");

}

}

@Test(expected=Exception.class)

**public** **void** testAdd() {

user.add();

}

@Test

**public** **void** testUpdata() {

user.updata();

}

@After

**public** **void** after(){

System.***out***.println("前置逻辑");

}

@Test

**public** **void** testQuery() **throws** Exception {

**if** (user.query(**false**)==**null**) {

**throw** **new** Exception("这不能为空");

}

}

@BeforeClass

**public** **static** **void** Before(){

System.***out***.println("前置逻辑");

}

@AfterClass

**public** **static** **void** after(){

System.***out***.println("前置逻辑");

}

Mvc(设计模式)

试图逻辑 //用于显示的模块 v

数据的模型与业务逻辑 //数据的结构 m

控制逻辑 //控制什么数据返回到什么页面 c

例：