# JAVASCRIPT

## js

### 变量定义

a = 1.3；

var i=10;

var j="abc";

var string = String(100);//类型转换

数字与字符串相加为字符串

### 基本数据类型

#### Number

-5, -5.5 , 0xd都是数字类型

测试变量是非数字：

isNaN("4")会得到返回值false，因为4是数字

isNaN("four")返回ture，因为four不是数字

数学函数：

Math.PI 得到圆周率

Math.pow(x,y); 返回x的y次幂

常量：

Infinity

Number.MAX\_VALUE

Number.MIN\_VALUE

Number.NEGATIVE\_INFINITY 负无穷大

Number.POSITIVE\_INFINITY 正无穷大

#### 字符和字符串

‘a’ ‘abc’ “abc”

#### Boolean

Boolean b=true;

#### null

代表没有值，不等价于空

#### undefined

是一种状态

#### 数据转换

parseInt("123", 10) 字符串转整数，第二个参数默认10进制

parseFloat() 字符串转浮点数

String(obj) 转换为字符串

### 函数

function fun(\*，\*){........}

参数无类型，只需要标识

返回值如果有需要直接写

一个函数返回值时，如果函数没有返回，则接收为undefined

function fun(a,b)

{

alert(a);

}

fun(12); // 调用时可以直接缺省参数

#### 带参函数

function ab(txt){........}

调用时：onclick=”ab(‘hello’)”

外部有双引号时，内部用单引号

#### 默认参数

定义函数时可以不定义参数，函数中使用arguments数组直接拿参数

a = arguments[0];

b = arguments[1];

#### 返回值

function ab(){ return “hello”;}

调用时：document.write(ab());

#### 匿名函数

也就是函数直接量

var i = function (x, y) { return x + y }

alert(i(1, 1));

#### 闭包

嵌套函数可以访问外部函数的变量

function f() {

var num = 10;

function show() {

alert(num);

}

return show();

}

f();//调用

### 事件

===============标签中调用

<script language = "javascript" type ="text/jscript">

function check()

{

if (f1.user.value == "")

{

alert("请输入");

f1.user.focus();

}

else

{

f1.submit();

}

}

</script>

Name或id属性都可以用.语法访问到

<form name="f1" action="b.html" method="post" target="\_blank">

用户名:<input type="text" name="user" />

<input type="button" onclick = "check()" value="按钮名"/>

</form>

=========放在元素下

Js代码加载必须在相应标签加载完后执行

document.getElementById('bt').onclick = function() { alert(1); };

或者

document.getElementById('bt').addEventListener('click', function() { alert(1); }, false);

### 对象

#### 使用函数构造

**function** *People*(name)  
{  
 **this**.**name** = name;  
 **this**.show = **function**(age){alert(**this**.**name** + age);};  
}

**var** people = **new** *People*(**"nn"**);  
people.show(10);

#### 声明后添加

**var** people = **new** *Object*();  
people.**name** = **"xx"**;  
people.show = **function**(age){alert(**this**.**name** + age);};  
  
people.show(12);

#### json构造

**var** people = {**name**:**"aa"**,show: **function**(age){alert(**this**.**name** + age)}};  
people.show(10);

并列数据之间使用逗号（,）

映射用冒号（:）

并列数据集合（数组）用方括号（[]）

映射的集合（对象）用大括号({})

### 操作符

#### in

用来鉴定一个给定的属性是否包含在一个对象内，in搜索的是属性是否存在。

var obj =

{

a: "aaa",

b: "bbb"

};

if ("a" in obj) {

alert("xx");

}

#### instanceof

用来测试一个给定的表达式（通常是一个变量）是否是类的一个对象。

var date = new Date();

if (date instanceof Date) {

alert("xx");

}

#### delete

将对象的一个值删除或者变为未定义

delete N[0]; //删除数组中第一个元素

#### typeof

返回操作数的变量类型

var i = 10;

if (typeof (i) == "number") {

alert("xx");

}

#### void

常用来提交一个表单或打开一个新的窗口

void(window.open());

### 错误处理

try{...执行代码...

throw(“ab”);

}

catch(err){..处理错误.

if(err==”ab”){......}.

}

Throw抛出的异常参数，由err变量接收

### 执行代码

将字符串解释为js代码执行

eval("var i = 10; alert(i);")

### 浏览器调用

#### 写入HTML

<script>

document.write("<h1>xxxxxx</h1>");

</script>

包含函数的脚本位于文档的head部分，这样可以确保调用函数前，脚本已经载入。

包含执行内容的脚本一般放在body末尾，有利于载入，防止阻塞造成的页面载入慢。

在xhtml上，<和&被解释为xml，为了避免干扰，在xhtml中使用

<script type="text/javascript">

#### JS文件载入

将脚本保存为.js，可以被多个文件调用

JS文件中：

document.write("<h1>xxxxxx</h1>");

function fun(){alert("xx");}

Html文件中：

<script language = "javascript" type ="text/jscript" src = "a.js">

</script>

<button onclick = "fun();">btn</button>

#### 写入浏览器

在浏览器的地址栏直接写入代码

javascript:alert("hello");

#### JS文件交互

通过<script>标签引入的两个js之间不能互相调用

a.js需要调用b.js中的函数和变量时，body中载入b.js

在body节点后用scrpit标签引入a.js

newElement = document.createElement("script");

newElement.setAttribute("src","b.js");

newElement.setAttribute("type","text/javascript");

document.body.appendChild(newElement);

## js对象

### json

{"key1":obj1,"key2":obj2}

[obj1,obj2]

obj可以是{}、[]，或其他类型数据

JSON.parse() 将JSON 字符串转换为 JavaScript 对象。

JSON.stringify() 将 JavaScript 值转换为 JSON 字符串。

### console

console.log("xxx");

### 数组

#### 构造

1）var N = ["abc",2];

2）var N = new Array();

N[0] = "abc";

N[1] = 2;

1. var N = new Array("abc",2);

#### 数组操作

arr.pop(); 在尾部删除元素，返回被删除元素

arr.push(7); 在尾部添加元素，返回新数组长度

arr.shift(); 在头部删除元素

arr.unshift(7); 在头部添加元素

var str = arr.join(); 将数组内容变成字符串

var arr3 = arr.concat(arr2); 连接两个数组

#### 遍历

for in对于数组对象属性进行循环

for(index in N){

alert(N[index]);

}

var CC = {a:1,b:2};

for(var x in CC)

{

var key = x;

var val = CC[x];

}

### String

Hello \

Abc 使用\可以对字符串进行折行处理

“abc”，”A” 都是字符串类型

"\*\*\*'str'\*\*\*", '\*\*\*"str"\*\*\*' 字符串用单引号或者双引号包含，单引号内可以包含双引号，双引号内可以包含单引号。

转译字符：\”

|  |  |
| --- | --- |
| 功能 | 用法 |
| length | "\*\*\*".length可以获取字符串长度 |
| substr | substr(1,2) 返回下标1开始，长度2的字符串。第二个参数可省略 |
| indexOf | indexOf("bc") 从字符串开头查找，返回bc所在的位置下标 |
| lastIndexOf | lastIndexOf("bc") 从字符串末尾查找，返回bc所在的位置下标 |
| split | 将字符串根据-分割  var arr = str.split("-");  for(i = 0 ; i < arr.length; ++i){} |
| replace | 匹配str中xcc，全部替换成##。 /g表示全部替换，如果不加则替换匹配的第一个  var reg = /xc{2}m/g;  var ss = str.replace(reg,"##"); |
| 正则表达式 | var str = "http://www.abc.org";  var Regex = /http:\/\/w+\.(.\*)/g; //正则表达式 /g表示匹配全部  var result = Regex.exec(str); //result为一个字符串数组，第一个元素是源字符串，后面的元素为正则表达式中每个（）内的内容  String的match,search,replace可以使用正则表达式 |

### Date

date = new Date(); 获取当前时间

new Date(2010,1,1); 构造，小时，分钟，秒，毫秒缺省

date.getSeconds() 获取秒数

date.getFullYear()获取年，getYear()获取的是19\*\*年差

显示时间

<p id = "clock"></p>

function show()

{

var now = new Date();

clock.innerHTML = now.getHours() + ":" + now.getMinutes() + ":" + now.getSeconds();

}

window.setInterval("show()",1000);

### 浏览器对象

#### window

window对象是一个全局对象，表示浏览器目前正打开的窗口，是其他对象的顶层对象，所以可以省略名称，直接调用

|  |  |
| --- | --- |
| alert | 弹出提示框  alert(....); |
| location | location.href = "page" |
| location.host | 当前页面的地址，如：127.0.0.1:8080 |
| confirm | 弹出确认框,确认返回true  confirm(...); |
| prompt | 弹出提示对话框，并要求输入一个字符串  defster是要求输入位置的默认值  prompt("xx","defstr"); |
| open | var windowVar = open("b.html","target","width=500,height=200");  执行成功返回新窗口对象。第一个参数为url地址，空字符串则打开空窗口。第二个参数制定新窗口名称，该名称可作为<a>和<form>的target属性。第三个参数可选。 |
| close | 关闭浏览器窗口  close(); |
| setInterval  　　　clearInterval | 定时器，1s调用1次fun  setInterval("fun",1000);  清除定时器，id从1开始，是每个定时器设置的先后次序  clearInterval(1); |
| setTimeout  　　　clearTimeout | 1s后执行fun函数  setTimeout("fun()",1000);  取消代码延迟  clearTimeout(1); |
| print | 调用浏览器打印机  print(); |
| moveTo  　　　moveBy | 移动浏览器到指定位置  moveTo(1,1);  移动偏移量  moveBy(-10,100); |
| scrollTo | 移动右侧和下方滚动条 |
| resizeTo | 设置窗口宽高  resizeTo(500, 500); |

#### DOM

文档对象模型（DOM）document对象提供了一种访问和修改HTML文档内容的方法。

DOM是树形，对于html，树根是<html>标记，<head>和<body>是枝干

##### document方法

|  |  |
| --- | --- |
| write | document.write("<h1>xxxxxx</h1>");  直接写html文件，如果是在开始载入，则会写在文件头。如果是中间调用则会清空文件 |
| getElementById  直接用id也可以代表一个标记对 | 通过id获取一个标记对  innerHTML 表示标记对在页面显示的内容  通过标记对中id属性获取，如<p id="id1"></p>  var x = document.getElementById("id1");  x.innerHTML = "xfasdfsdf"; //修改标签内的内容//也可以使用id1.innerHTML  alert(x.tagName); |
| getElementsByTagName | 通过标记对名获取一组标记对  获取所有p标签，遍历获得每个的内容  var content = document.getElementsByTagName("p");  for(i = 0 ; i < content.length; ++i)  {  alert(content[i].innerHTML);  } |
| createElement | 创建标记对  根据标签名td创建td标记对元素  td\_con = document.createElement("td"); |

##### 节点方法

|  |  |
| --- | --- |
| setAttribute | 设置标记对属性值  \*\*.setAttribute("bgcolor","red"); |
| removeAttribute | 删除属性  \*\*.removeAttribute("bgcolor"); |
| appendChild | 将con加入到\*\*节点下  \*\*.appendChild(con); |
| insertBefore | 在子节点refChil前面插入新节点newElement  \*\*.insertBefore(newElement,refChil); |
| replaceChild | 将子节点odl替换成newElement  \*\*.replaceChild(newElement,old) |
| removeChild | 删除子节点ch  \*\*.removeChild(ch); |
| cloneNode | 复制\*\*节点，包括属性。参数为true会通过递归方法克隆子节点  newNode= \*\*.cloneNode(true); |
| hasChildNodes | 判断一个元素是否有子节点（标记对中没有任何内容，包括空白）  文本节点和属性节点不可能再包含子节点，返回false  nodeType  1 元素节点类型  2 属性节点类型  3 文本节点类型 |

##### 遍历所有节点

var elemList = "";

function getElement(node)

{

var total = 0;

if (node.nodeType == 1) //检查是否是elment对象

{

total++;

elemList = elemList + node.nodeName + " ";

}

var chil = node.childNodes;

for(var m = node.firstChild; m != null; m = m.nextSibling)

{

total += getElement(m);

}

return total;

}

var num = getElement(document);

alert("包含" + num + "个标记" + " " + elemList);

##### HTML集合

document.anchors 包含所有a元素的组(具有name属性的参数)

document.forms 包含所有form元素的组

document.images 包含所有图像元素的组

document.links 包含所有具有href属性的a元素的值

## function

### 获取尺寸

BODY对象宽度

document.body.clientWidth

可见区域宽度

document.documentElement.clientWidth

### canvas绘图

<canvas id="canv"></canvas>

var canvas = document.getElementById('canv');

var ctx = canvas.getContext("2d");

#### 铅笔绘图

ctx.strokeStyle = "#f00";

ctx.lineWidth = 3;

function draw(sx, sy, ex, ey) {

//ctx.clearRect(0, 0, 600, 600);

//ctx.beginPath();//执行clearRect和beginPath 则重绘

ctx.moveTo(sx, sy);

ctx.lineTo(ex, ey);

ctx.stroke();

}

var oX, oY;

var flag = false;

canvas.onmousemove = function(e) {

if(!flag) return;

draw(oX, oY, e.pageX, e.pageY);

oX = e.pageX;

oY = e.pageY;

}

canvas.onmousedown = function(e) {

flag = true;

oX = e.pageX;

oY = e.pageY;

}

canvas.onmouseup = function(e) {

flag = false;

}

onmouseup = function(e)

{

flag = false;

}

#### 绘制矩形

ctx.fillStyle = "red";

ctx.strokeStyle = "red";

ctx.lineWidth = 5;

ctx.fillRect(0,0, 300, 200);

ctx.strokeRect(0,0,300,200);

#### 绘制图片

var img = new Image();

img.src = "1.jpeg";

img.onload = function()

{

ctx.drawImage(img,0,0); //从左上角开始绘制图像

ctx.drawImage(img,10,20,150,200); //从指定坐标（10,20）开始绘制图像，并设置图像宽和高 指定这些参数可以使得图像可以缩放

//裁剪一部分图像放在左上角，并调整尺寸

//ctx.drawImage(img,90,80,100,100,0,0,120,120);

}

## nodejs

### 配置

安装包下载：<http://nodejs.cn/download/>

安装后检查

node --version

npm -v

### npm

当不在nodejs的目录（node\_modules同级目录）运行npm

就会安装在C:\Users\用户名\node\_modules 目录中

安装npm镜像，使用cnpm代替npm

npm install -g cnpm --registry=https://registry.npm.taobao.org

安装mysql模块：

cnpm install mysql

### 服务搭建

var http = require('http');

var url = require("url");

http.createServer(function (request, response) {

var pathname = url.parse(request.url).pathname;

response.writeHead(200, {'Content-Type': 'text/plain;charset=utf-8'});

response.end('\*\*\*\*\*\*\*\*\*');

}).listen(8080);

console.log('Server running at <http://127.0.0.1:8080/');>

将内容写入ser.js文件

命令行启动服务： node ser.js

### 参数获取

var parseObj = url.parse(request.url, true);

request.method //请求方式 GET/POST

parseObj.pathname //无参数的url路径

request.headers["content-type"] //消息报头中的key-val

parseObj.query['x'] //参数x的值

var qs = require('querystring');

//获取post参数

if("POST" == request.method.**toUpperCase**())

{

var postData = "";

request.**addListener**("data", function (data) {

postData += data;

});

request.**addListener**("end", function () {

var query = qs.**parse**(postData);

**console**.**log**(query['passWord']);

});

}

### 读文件

var fs = require("fs");

fs.readFile(pathname, function (err, data) {

if (err) {

console.log(err);

response.writeHead(404, { 'Content-Type': 'text/html' });

} else {

response.writeHead(200, { 'Content-Type': 'image/jpeg' });

response.write(data);

}

response.end();

});

### 模块调用

fileA.js内容：

function run(param)

{

console.log(param);

}

exports.run = run;

调用时：

var modeA = require("./fileA");

modeA.run("xxx");

### mysql

cnpm install mysql

var mysql = **require**('mysql');

var connection = mysql.**createConnection**({

host : 'localhost',

user : 'root',

password : 'root',

database : 'testdb'

});

connection.**connect**(function (error)

{

if(null != error) **console**.**log**(error);

});

connection.**query**('SELECT \* FROM user', function (error, rows) {

if(error){

**console**.**log**('[SELECT ERROR] - ',error.message);

return;

}

**console**.**log**(rows);

for(i in rows)

{

**console**.**log**(rows[i].age + " " + rows[i].name);

}

});

### 事件

var events = **require**('events');

var eventEmitter = new **events**.**EventEmitter**();

var **connectHandlerA** = function **connected**() {

**console**.**log**('事件A');

}

var **connectHandlerB** = function **connected**() {

**console**.**log**('事件B');

}

*// 绑定 事件处理程序*

eventEmitter.**on**('eventA', connectHandlerA);

eventEmitter.**on**('eventB', connectHandlerB);

**setTimeout**(function() {

*//触发事件*

eventEmitter.**emit**('eventA');

}, 1000);

**setTimeout**(function() {

eventEmitter.**emit**('eventB');

}, 2000);

监听一次消息：

**once(event, listener)**

## Vue

cnpm install webpack-dev-server -g

cnpm install -g vue-cli

vue init webpack vuetest

进入项目目录vuetest

安装依赖：cnpm install

运行项目：npm run dev

打包：

npm run build (npm run build:prod)

放入tomcat方法：

修改config\index.js中build节点下的assetsPublicPath: '/prj',

生成文件在dist目录下

生产文件放入tomcat服务器，访问<http://127.0.0.1:8080/prj>

## jQuery

### 基础语法

$(selector).action()

$("p").hide() - 隐藏所有段落

#### 元素选择器

$("p") 选取 <p> 元素。

$("p.intro") 选取所有 class="intro" 的 <p> 元素。

$("p#demo") 选取所有 id="demo" 的 <p> 元素。

$("p[xx='#']"") 选取所有 xx="#" 的 <p> 元素。

#### 属性选择器

$("[xx]") 选取所有带有 xx属性的元素。

$("[xx='#']") 选取所有带有 xx值等于 "#" 的元素。

$("[xx!='#']") 选取所有带有 xx值不等于 "#" 的元素。

$("[xx$='.jpg']") 选取所有 xx值以 ".jpg" 结尾的元素。

#### CSS选择器

$("p").css("background-color","red");

所有 p 元素的背景颜色更改为红色

### 名称冲突

var jq=jQuery.noConflict()；

使用自己的名称（比如 jq）来代替 $ 符号。

### 初始化调用

$(document).ready(

function(){...} 或 fun()

);

文档加载完成时进行调用,该函数可以写多个

### 按钮绑定函数

$("#btn1").click(function(){alert("xx");}); -设置按钮响应函数

只能使用以下方法绑定：

$(document).ready(

function(){

$("button").click(function(){

btnFun();

return false;//拦截原有的操作

});

}

);

### 效果

Callback参数是函数执行成功后调用，执行失败不调用

#### 隐藏显示

$("p").hide(); 隐藏所有p标签

$("p").show(); 显示所有p标签

#### 淡入淡出

参数可以是"slow"、"fast" 或毫秒

$("p").fadeOut(1000); 所有p标签1s内逐渐消失

$("p").fadeIn(1000); 所有p标签1s内逐渐出现

$("p").toggle(1000); 如果处于隐藏则淡入，如果显示状态则淡出

$("p").fadeTo(1000,0.5); 变成0.5的透明度

#### 滑动

参数可以是"slow"、"fast" 或毫秒

$("p").slideUp(); 向上滑动（消失）

$("p").slideDown(); 向下滑动（出现）

$("p").slideToggle(); 自动根据状态切换

#### 动画

第一个参数是css效果，第二个参数是变化时间

var v=$("p");

v.animate({height:'30px',opacity:'0.4'},"slow");

停止动画效果

$(selector).stop(stopAll,goToEnd);

可选的 stopAll 参数规定是否应该清除动画队列。默认是 false，即仅停止活动的动画，允许任何排入队列的动画向后执行。

可选的 goToEnd 参数规定是否立即完成当前动画。默认是 false。

因此，默认地，stop() 会清除在被选元素上指定的当前动画。

### 连续操作

$("p").css("color","red").slideUp(2000).slideDown(2000);

### HTML元素操作

#### 设置获取

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）

val() - 设置或返回表单字段的值

#### 添加

$("p").append(" XXX"); 结尾加入文字

$("p").prepend(" XXX"); 前方加入文字

after() 和 before() 会加入换行

var txt1="<p>Text.</p>"; // 以 HTML 创建新元素

var txt2=$("<p></p>").text("Text."); // 以 jQuery 创建新元素

var txt3=document.createElement("p"); // 以 DOM 创建新元素

txt3.innerHTML="Text.";

$("p").append(txt1,txt2,txt3); // 追加新元素

#### 删除

$("div").remove(); 删除元素及子元素

$("p").remove(".aa"); 选择p的class为aa的元素进行删除

$("div").empty(); 删除子元素

#### radiobutton

##### 选择值

var val = $('#radioGroupDiv input[name="radioItemClass"]:checked').val();

选择id为radioGroupDiv 范围内的class为radioItemClass的value属性

##### 设置值

$("input[name=name属性对应的值]").val(["value属性对应值"]);

#### CSS

<style>

.ss{color:blue;}

</style>

$("p").addClass("ss"); //添加样式

$("p").css("color","red");//设置颜色

removeClass() - 从被选元素删除一个或多个类

toggleClass() - 对被选元素进行添加/删除类的切换操作

css() - 设置或返回样式属性

#### 尺寸

|  |  |
| --- | --- |
| width() | 设置或返回元素的宽度（不包括内边距、边框或外边距） |
| height() | 设置或返回元素的高度（不包括内边距、边框或外边距） |
| innerWidth() | 返回元素的宽度（包括内边距） |
| innerHeight() | 方法返回元素的高度（包括内边距） |
| outerWidth() | 方法返回元素的宽度（包括内边距和边框） |
| outerHeight() | 方法返回元素的高度（包括内边距和边框） |

#### 遍历

|  |  |  |
| --- | --- | --- |
| parent() | 返回被选元素的直接父元素 |  |
| parents() | 返回被选元素的所有祖先元素，它一路向上直到文档的根元素 (<html>) |  |
| parentsUntil() | 返回介于两个给定元素之间的所有祖先元素 | $("span").parentsUntil("div")，span处于div子节点下，选择span和div之间的关联元素。 |
| children() | 返回被选元素的所有直接子元素。该方法只会向下一级对 DOM 树进行遍历。 |  |
| find() | 返回被选元素的后代元素，一路向下直到最后一个后代。 | $("div").find("p").fadeToggle();  将div下所有p元素隐藏 |

|  |  |
| --- | --- |
| siblings() | 返回被选元素的所有同胞元素 |
| next() | 返回被选元素的下一个同胞元素  prev() 反向 |
| nextAll() | 返回被选元素的所有跟随的同胞元素  prevAll() |
| nextUntil() | 返回介于两个给定参数之间的所有跟随的同胞元素 |

|  |  |  |
| --- | --- | --- |
| first() | 选择第一个元素 | $("p").first() 选择第一个p元素  $("div p").first()选择div内第一个p |
| last() | 选择最后一个元素 |  |
| eq() | 返回被选元素中带有指定索引号的元素，索引从0开始 | $("p").eq(1) 选择第二个p元素 |
| filter() | 只选择满足筛选条件的元素 | $("p").filter(".a") 选择class为a的p元素 |
| not() | 只选择不满足筛选条件的元素 |  |

### AJAX

|  |  |
| --- | --- |
| $.ajax({ key:val,key2:val2,... }); | $.ajax({  type: "POST",  url: "......",  data:{valA: "val1",valB: "val2"},  dataType: "json",  success: function(data){}  error:function(errorData){}  //返回数据不为dataType: "json"时调用error方法  }); |
| $.get(URL,callback); | $.get("a.jsp", function(data, status) {  alert("Data: " + data + "\nStatus: " + status);  }); |
| $.post(URL,data,callback); | $.post("a.jsp", {  valA: "val1",  valB: "val2"  },  function(data, status) {  console.log("Data: " + data + "\nStatus: " + status);  }); |

## bootstrap

### 引用

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="bootstrap.min.css">

<script src="jquery-3.3.1.min.js"></script>

<script src="popper.min.js"></script>

<script src="bootstrap.min.js"></script>

### 容器

<div class="container">

所有元素放在该容器内生效

### 灰色区域

<div class="jumbotron">

### 行列布局

<div class="row">

<div class="col-sm-4">

<h3>hhhh</h3>

<p>ppppp</p>

</div>

<div class="col-sm-4">

<h3>hhhh</h3>

<p>ppppp</p>

</div>

</div>

尺寸包括：

.col-xs- .col-sm- .col-md- .col-lg-

### 表格

<table class="table table-striped table-bordered">

<thead>

<tr class="table-active">

<th>名称</th>

<th>城市</th>

</tr>

</thead>

<tbody>

<tr class="table-warning">

<td>Tanmay</td>

<td>Bangalore</td>

</tr>

<tr class="table-success">

<td>Sachin</td>

<td>Mumbai</td>

</tr>

</tbody>

</table>

颜色包含table-active、table-danger等

### 表单

<form role="form">

<div class="form-group">

<input type="text" class="form-control" placeholder="文本输入">

</div>

</form>

### 按钮

<button type="button" class="btn btn-default">默认按钮</button>

<button type="button" class="btn btn-primary">原始按钮</button>

<button type="button" class="btn btn-success">成功按钮</button>

<button type="button" class="btn btn-info">信息按钮</button>

<button type="button" class="btn btn-warning">警告按钮</button>

<button type="button" class="btn btn-danger">危险按钮</button>

<button type="button" class="btn btn-link">链接按钮</button>

.btn-lg 比较大

.btn-sm 比较小

.btn-xs 特别小

.btn-block 横跨父元素的全部宽度

### 按钮组

<div class="btn-group btn-group-justified">

<div class="btn-group">

<button type="button" class="btn btn-primary">A</button>

</div>

<div class="btn-group">

<button type="button" class="btn btn-primary">B</button>

</div>

</div>

### 下拉按钮

<div class="btn-group">

<button type="button" class="btn btn-primary dropdown-toggle" data-toggle="dropdown">

Sony <span class="caret"></span></button>

<ul class="dropdown-menu" role="menu">

<li><a href="#">Tablet</a></li>

<li><a href="#">Smartphone</a></li>

</ul>

</div>

### 分隔按钮

<div class="btn-group">

<button type="button" class="btn btn-primary">Sony</button>

<button type="button" class="btn btn-primary dropdown-toggle" data-toggle="dropdown">

<span class="caret"></span>

</button>

<ul class="dropdown-menu" role="menu">

<li><a href="#">Tablet</a></li>

<li><a href="#">Smartphone</a></li>

</ul>

</div>

### 辅助类

文本样式：

.text-muted

.text-primary

.text-success

.text-info

.text-warning

.text-danger

背景颜色：

.bg-primary

.bg-success

.bg-info

.bg-warning

.bg-danger

其他：

.pull-left 元素浮动到左边

.center-block 设置元素为 display:block 并居中显示

.clearfix 清除浮动

.show 强制元素显示

.hidden 强制元素隐藏

.sr-only 除了屏幕阅读器外，其他设备上隐藏元素

.sr-only-focusable 与 .sr-only 类结合使用，在元素获取焦点时显示(如：键盘操作的用户)

.text-hide 将页面元素所包含的文本内容替换为背景图

.close 显示关闭按钮

.caret 显示下拉式功能

### bootstrap-table

#### 中文编码

bootstraptable刷新函数参数包含中文时，接收需要解码

String name = new String(userName.getBytes("iso-8859-1"),"utf-8");

#### 前台

<link rel="stylesheet" type="text/css" href="bootstrap.min.css" />

<link rel="stylesheet" type="text/css" href="bootstrap-table.min.css">

<script src="jquery.min.js"></script>

<script src="bootstrap.min.js"></script>

<!-- 一个页面中js引用多次会出错 -->

<script src="bootstrap-table.min.js"></script>

<script src="bootstrap-table-zh-CN.min.js"></script>

<table id="myTable" class="table table-hover"></table>

function queryParamsFun(params)

{

var id = $("#barrelDevidQuery").val();

var temp = {

limit: params.pageSize,

page: params.pageNumber,

id:id,

};

return temp;

}

//bootstrap-table要求服务器返回的json数据包含：total和rows

$('#myTable).bootstrapTable({

url: 'GetPageData',

method: "POST",

queryParamsType: "",

queryParams: queryParamsFun,

contentType: "application/x-www-form-urlencoded",

dataType:'json',

toolbar: "#toolbar",

striped: true, // 是否显示行间隔色

uniqueId: "id", //每一行的唯一标识，一般为主键列

sortable: false, //是否启用排序

sortOrder: "asc", //排序方式

pagination: true, //是否显示分页

sidePagination: "server", //分页方式：client客户端分页，server服务端分页（\*）

pageNumber:1, //初始化加载第一页，默认第一页

pageSize: 5, //每页的记录行数

pageList: [5, 10, 25, 50, 100], //可供选择的每页的行数

strictSearch: true,

showColumns: true, //是否显示选择列按钮

showRefresh: true, //是否显示刷新按钮

minimumCountColumns: 2, //最少允许的列数

clickToSelect: true, //是否启用点击选中行

height: 500, //行高，如果没有设置height属性，表格自动根据记录条数觉得表格高度

showToggle:true, //是否显示详细视图和列表视图的切换按钮

cardView: true, //是否显示详细视图

detailView: false, //是否显示父子表

columns:[

{

field: 'none',

title: '操作',

width: 120,

align: 'center',

valign: 'middle',

formatter: actionFormatter,

},

{title: 'IP',field: 'ip',visible:true},

{title: 'port',field: 'port'},

{title: 'ID',field: 'id'},

]

});

$("#quaryBtn").click(function() {

$("#myTable").bootstrapTable('refresh',queryParamsFun);

});

function actionFormatter(value, row, index) {

//在数据刷新时调用，将数值固定写入到了result

//value:当前单元格的field

//row:当前行的数据

//index:当前行索引

var val = JSON.stringify(row).replace(/\"/g,"'");

var result = "";

result += "<a href='javascript:;' class='btn btn-xs blue' onclick=\"EditView(" + value + "," + val + ")\" title='编辑'><span class='glyphicon glyphicon-pencil'></span></a>";

return result;

}

function EditView(value,row){}

#### 后端

##### 实体类

public class Page implements Serializable {

private static final long serialVersionUID = 1L;

private Integer total; //bootstrap-table要求服务器返回的json数据包含：total和rows

private List rows = new ArrayList();

public Integer getTotal() {return total;}

public void setTotal(Integer total) {this.total = total;}

public List getRows() {return rows;}

public void setRows(List rows) {this.rows = rows;}

}

##### contoller

@RequestMapping(value = "/GetPageData", method = RequestMethod.POST)

public @ResponseBody Page getBarrelByPage(@RequestParam("page") Integer startPage, @RequestParam("limit") Integer limit, @RequestParam(id") String Id) {

return service.getByPage(startPage, limit, Id);

}

##### service

public Page getByPage(Integer startPage, Integer limit,String ID)

{

int start = (startPage - 1) \* limit;

Integer total = mapper.getcount(ID);

List<SaveData> rows = mapper.getllimit(start, limit, devID);

Page page = new Page();

page.setTotal(total);

page.setRows(rows);

return page;

}

##### mapper

<sql id="limit\_sql">

<where>

<if test="id != null and id != ''">

<![CDATA[

TempTableName.id=#{id}

]]>

</if>

</where>

</sql>

<select id="getcount" parameterType="map" resultType="int">

<![CDATA[

select count(id) from TableName TempTableName

]]>

<include refid="limit\_sql" />

</select>

<select id="getllimit" parameterType="map" resultType="map">

<![CDATA[

select \* from TableName TempTableName

]]>

<include refid="limit\_sql" />

<![CDATA[

ORDER BY id DESC LIMIT #{start}, #{limit}

]]>

</select>

### 模态对话框

#### 按钮触发

<button class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">打开对话框</button>

#### js中触发

$('#myModal').modal({key: val})

$('#myModal').modal('toggle') 手动切换

$('#myModal').modal('show') 手动打开

$('#myModal').modal('hide') 手动隐藏

#### 对话框

<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>

<h4 class="modal-title" id="myModalLabel">模态框（Modal）标题</h4>

</div>

<div class="modal-body">在这里添加一些文本</div>

<div class="modal-footer">

<button type="button" class="btn btn-default" data-dismiss="modal">关闭</button>

<button type="button" class="btn btn-primary">提交更改</button>

</div>

</div>

</div>

</div>

## vue

### 数据绑定

<script src="vue.min.js"></script>

<div id="vue\_det">

<h1>{{aaa}}</h1>

<h1>{{bbb}}</h1>

<h1>{{funA()}}</h1>

<h1>{{funB()}}</h1>

</div>

<script type="text/javascript">

var vm = new Vue({

el: '#vue\_det',

data: {

aaa: "xxxx",

bbb: "yyyy",

},

methods: {

funA: function() {

return this.aaa + "111";

},

funB: function() {

return this.bbb + "222";

}

}

})

</script>

<button onclick="vm.aaa='mmmm';vm.bbb='nnn';">modiyA</button>