JS常用方法封装

使用css选择器选择单个元素

```
$:function(elem){
•    return document.querySelector(elem);
}
```

使用css选择器选择一组元素

```
$$:function(elem){
•    return document.querySelectorAll(elem);
}
```

判断浏览器类型

```
browserType:function(){
•    var browser = window.navigator.userAgent; *//获取浏览器*
•    if (userAgent.indexOf("Opera") > -1) {
•    return "Opera"
•    }; *//判断是否Opera浏览器*
•    if (userAgent.indexOf("Firefox") > -1) {
•      return "FireFox";
•    } *//判断是否Firefox浏览器*
•    if (userAgent.indexOf("Chrome") > -1){
•      return "Chrome";
•    }
•    if (userAgent.indexOf("Safari") > -1) {
•      return "Safari";
•    } *//判断是否Safari浏览器*
•    if (userAgent.indexOf("compatible") > -1 && userAgent.indexOf("MSIE") > -1
&& !isOpera) {
•      return "IE";
•    }; *//判断是否IE浏览器*
}
```

判断终端类型（手机端、PC端）

```
terminalType:function(){
•    var browser = window.navigator.userAgent; *//获取浏览器*
•    if (userAgent.indexOf("Android") > -1) {
•    return "Android"
•    }; *//判断是否安卓手机*
•    if (userAgent.indexOf("iPhone") > -1) {
•      return "iPhone";
•    } *//判断是是否苹果手机*
}
```

移除所有节点

```
removeNode:function (elem){
•    for(var i = 0,len = elem.length;i < len;i++){
•      elem[i].parentNode.removeChild(elem[i]);
•    }
}
```

清空所有子元素

```
emptyChildElem:function(elem){
•    elem.innerHTML = "";
}
```

获取子元素

```
getChildNodes:function(elem){
•    var childNodes = [];
•    var node = elem.childNodes;
•    for(var i = 0;i < node.length;i++){
•      if(node[i].nodeType == 1){
•        childNodes[childNodes.length] = node[i];
•      }
•    }
•    return childNodes;
}
```

获取所有兄弟元素

```
getSiblingElems:function (elem){
•    var result = [], parentNode ,childNodes;
•    parentNode = elem.parentNode;
•    childNodes = parentNode.childNodes;
•    for(var i = 0,len = childNodes.length;i < len;i++){
•      if(childNodes[i].nodeType == 1 && childNodes[i] != elem){
•        result[result.length] = childNodes[i];
•      }
•    }
•    return result;
}
```

获取下一个兄弟元素

```
nextSiblingElem:function(elem){
•    var siblings = zycTools.getChildNodes(elem.parentNode);
•    var index = zycTools.getIndex(siblings,elem);
•    var result = siblings[parseInt(index)+1];
•    return result;
},
```

获取指定元素所有后面的同辈元素

```
nextSiblingsAll:function(elem){
    var index,result = [];
    var childNodes = zycTools.getChildNodes(elem.parentNode);
    for(var i = 0,len = childNodes.length;i < len;i++){
      if(elem == childNodes[i]){
        index = i+1;
      }
    }
    for(var j = index;j < len;j++){
      result[result.length] = childNodes[j];
    }
    return result;
}
```

获取前一个兄弟元素

```
prevSiblingElem:function(elem){
    var siblings = zycTools.getChildNodes(elem.parentNode);
    var index = zycTools.getIndex(siblings,elem);
    var result = siblings[parseInt(index)-1];
    return result;
}
```

获取指定元素所有前面的同辈元素

```
prevSiblingsAll:function(elem){
    var index,result = [];
    var childNodes = zycTools.getChildNodes(elem.parentNode);
    for(var i = 0,len = childNodes.length;i < len;i++){
      if(elem == childNodes[i]){
        index = i-1;
      }
    }
    for(var j = 0;j <= index;j++){
      result[result.length] = childNodes[j];
    }
    return result;
}
```

指定位置之后插入元素

```
insertAfter:function(newEl, targetEl){
    var parentEl = targetEl.parentNode;
    if(parentEl.lastChild == targetEl){
      parentEl.appendChild(newEl);
    }else{
      parentEl.insertBefore(newEl,targetEl.nextSibling);
    }
}
```

往数组里面添加Dom元素

```
addDom:function(elem, addelem){
•    elem[elem.length] = addelem;
•    return elem;
}
```

查找后代元素

```
findElem:function (elemSelector, hasElemSelector){
•    var result =
document.querySelector(elemSelector).querySelectorAll(hasElemSelector);
•    return result;
}
```

获取元素对应的元素数组的索引

```
getIndex:function(elem, targetE){
•    var index;
•    for(var i = 0;i < elem.length;i++){
•      if(elem[i] == targetE){
•        index = i;
•      }
•    }
•    return index;
}
```

获取元素的文本,获取的元素必须使$$()方法获取或者必须是一组Dom数组

```
getText:function(elem){
•    var text = "";
•    for(var j = 0, lenj = elem.length;j < lenj;j++){
•      for(var i = 0, leni = elem[j].childNodes.length;i < leni;i++){
•        if(elem[j].childNodes[i].nodeType == 3){
•          var empText = elem[j].childNodes[i].nodeValue.replace(/[
]/g,"").replace(/[\r\n]/g,"");
•          text += empText + " ";
•        }
•      }
•    }
•    return text;
}
```

添加class

```
addClass:function(elem,classname){
•    var oldClass = elem.className;
•    if(elem.className){
•      if(elem.className.indexOf(classname) == -1){
•        elem.className = oldClass + " " + classname;
•      }
•    }else{
•      elem.className = classname;
•    }
}
```

移除class

```
removeClassName:function(elem,classname){
•    var oldClass = elem.className;
•    if(elem.className && oldClass.indexOf(classname) != -1){
•      elem.className = elem.className.replace(classname, "");
•    }else{
•      return false;
•    }
}
```

获取元素样式

```
getStyle:function(elem,prop){
•    if(window.getComputedStyle){
•      return window.getComputedStyle(elem)[prop];
•    }else{
•      return elem.currentStyle[prop];
•    }
}
```

设置元素样式

```
setCss:function(elem, cssObj){
•    for(var i in cssObj){
•      elem.style[i] = cssObj[i]
•    }
}
```

切换样式

```
toggleClass:function(elem,classname){
•    this.flag = false;
•    var oldClass = elem.className;
•    if(oldClass.indexOf(classname) != -1 && !flag){
•      flag = true;
•      elem.className = elem.className.replace(" "+classname,"");
•    }else{
•      flag = false;
•      elem.className = oldClass + " " + classname;
•    }
}
```

元素左右或者上下运动

```
elemMove:function(elem, dir, time, callback){
•    var flag = true, value;
•    dir == "top" ? value = "height" : value = "width";
•    var elemWith = parseInt(zycTools.getStyle(elem,value));  *//获取宽度值*
•    var elemPosition = zycTools.getStyle(elem,"position");  *//获取position属性*
•    if(elemPosition == "static"){
•      elem.style.position = "relative";
•      elem.style[dir] = "0px";
•    }
•    setInterval(function(){
•      var left = parseInt(zycTools.getStyle(elem,dir));
•      if(left < elemWith && flag){
•        elem.style[dir] = (left + 1) + "px";
•      }else{
•        flag = false;
•        elem.style[dir] = (left - 1) + "px";
•        if(parseInt(zycTools.getStyle(elem,dir)) < 0){
•          elem.style[dir] = "0px";
•          flag = true;
•        }
•      }
•    },time);
}
```

元素匀速向上运动

```
silderUp:function(elem, time, callback){
•    *//var i = 1;*
•    var timmerId = setInterval(function(){
•      var height = parseInt(zycTools.getStyle(elem,"height"));  *//获取元素高度*
•      if(height > 0){
•        elem.style.height = (height - 1) + "px";
•      }else{
•        elem.style.height = "0px";
•        clearTimeout(timmerId);
•        if(typeof callback == "function"){
•          callback();
•        }
•      }
•    },time);
}
```

元素匀速向下运动

```
silderDown:function(elem, time, callback){
•    var initHeight = parseInt(zycTools.getStyle(elem,"height")); *//元素原始高度*
•    elem.style.height = "0px";
•    elem.style.display = "block";
•    var timmerId = setInterval(function(){
•      var height = parseInt(zycTools.getStyle(elem,"height"));  *//每次循环就获取
元素高度*
•      if(height < initHeight){
•        elem.style.height = (height + 1) + "px";
•       }else{
•         clearTimeout(timmerId);
•         if(typeof callback == "function"){
•           callback();
•         }
•       }
•     },time);
}
```

元素放大缩小,仅支持高版本浏览器

```
transformScale:function(elem, time, multiple, callback){
•     var flag = true;
•     elem.style.webkitTransform = "scale(1)";
•     var timeId = setInterval(function(){
•       var scaleStr = elem.style.webkitTransform.replace("scale(","");
•       var scale = parseFloat(scaleStr);
•       if(scale < multiple && flag){
•         elem.style.webkitTransform = "scale(" + (scale + 0.01) + ")";
•       }else{
•         flag = false;
•         elem.style.webkitTransform = "scale(" + (scale - 0.01) + ")";
•         if(scale < 1){
•           elem.style.webkitTransform = "scale(1)";
•           flag = true;
•           clearTimeout(timeId);
•           *//函数执行完成回调执行callback*
•           if(typeof callback == "function"){
•             callback();
•           }
•         }
•       }
•     },time);
}
```

元素旋转Z轴

```
rotate:function(elem, time, deg, callback){
•     var flag = true;
•     elem.style.webkitTransform ? {}:elem.style.webkitTransform =
"rotate(0deg)";
•     var timeId = setInterval(function(){
•       var rotateStr = elem.style.webkitTransform.replace("rotate(","");
•       *//console.log(rotateStr);*
•       var rotate = parseInt(rotateStr);
•       *//console.log(rotate);*
•       if(rotate < deg && flag){
•         elem.style.webkitTransform = "rotate(" + (rotate + 1) + "deg)";
•       }else{
•         flag = false;
•         elem.style.webkitTransform = "rotate(" + (rotate - 1) + "deg)";
•         if(rotate < 0){
•           elem.style.webkitTransform = "rotate(0deg)";
•           flag = true;
•           *//函数执行完成回调执行callback*
•           if(typeof callback == "function"){
```

```
•             callback();
•           }
•         }
•       }
•     },time);
}
```

元素淡出消失

```
fadeOut:function(elem, time, callback){
•     var elemOpcity = zycTools.getStyle(elem,"opcity");
•     if(!elemOpcity){
•       elem.style.opacity = "1";
•     }
•     setInterval(function(){
•       var intOpacityValue = parseFloat(elem.style.opacity); *//将透明度转变为整形*
•       if(intOpacityValue > 0){
•         elem.style.opacity = (intOpacityValue - 0.05) + "";
•       }else{
•         elem.style.opacity = "0";
•         elem.style.display = "none";
•         if(typeof callback == "funtion"){
•           callback();
•         }
•       }
•     },time);
}
```

元素淡入显示

```
fadeIn:function(elem, time, callback){
•     elem.style.dispaly = "block";
•     var elemOpcity = zycTools.getStyle(elem,"opcity");
•     if(!elemOpcity){
•       elem.style.opacity = "0";
•     }
•     var timerId = setInterval(function(){
•       var intOpacityValue = parseFloat(elem.style.opacity); *//将透明度转变为整形*
•       if(intOpacityValue < 1){
•         elem.style.opacity = (intOpacityValue + 0.05) + "";
•       }else{
•         elem.style.opacity = "1";
•         clearTimeout(timerId);
•         if(typeof callback == "funtion"){
•           callback();
•         }
•       }
•     },time);
}
```

表单及常用验证

百家姓验证

```
chineseFirstName: function(str){
•       var firstName='赵钱孙李周吴郑王冯陈褚卫蒋沈韩杨朱秦尤许何吕施张孔曹严华金魏陶姜戚谢邹
喻柏水窦章云苏潘葛奚范彭郎鲁韦昌马苗凤花方俞任袁柳酆鲍史唐费廉岑薛雷贺倪汤滕殷罗毕郝邬安常乐于时
傅皮卞齐康伍余元卜顾孟平黄和穆萧尹姚邵湛汪祁毛禹狄米贝明臧计伏成戴谈宋茅庞熊纪舒屈项祝董梁杜阮蓝
闵席季麻强贾路娄危江童颜郭梅盛林刁钟徐丘骆高夏蔡田樊胡凌霍虞万支柯昝管卢莫经房裘缪干解应宗丁宣贲
邓郁单杭洪包诸左石崔吉钮龚程嵇邢滑裴陆荣翁荀羊于惠甄曲家封芮羿储靳汲邴糜松井段富巫乌焦巴弓牧隗山
谷车侯宓蓬全郗班仰秋仲伊宫宁仇栾暴甘钭厉戎祖武符刘景詹束龙叶幸司韶郜黎蓟薄印宿白怀蒲邰从鄂索咸籍
赖卓蔺屠蒙池乔阴郁胥能苍双闻莘党翟谭贡劳逄姬申扶堵冉宰郦雍郤璩桑桂濮牛寿通边扈燕冀郏浦尚农柴瞿阎
充慕连茹习宦艾鱼容向古易慎戈廖庾终暨居衡步都耿满弘匡国文寇广禄阙东欧殳沃利蔚越夔隆师巩厍聂晁勾敖
融冷訾辛阚那简饶空曾毋沙乜养鞠须丰巢关蒯相查后荆红游竺权逯盖益桓公万俟司马上官欧阳夏侯诸葛闻人东
方赫连皇甫尉迟公羊澹台公冶宗政濮阳淳于单于太叔申屠公孙仲孙轩辕令狐钟离宇文长孙慕容鲜于闾丘司徒司
空丌官司寇仉督子车颛孙端木巫马公西漆雕乐正壤驷公良拓拔夹谷宰父谷梁晋楚阎法汝鄢涂钦段干百里东郭南
门呼延归海羊舌微生岳帅缑亢况后有琴梁丘左丘东门西门商牟佘佴佰赏南官墨哈谯笪年爱阳佟第五言福百家姓
终百家姓以外的:万俟司马上官欧阳夏侯诸葛闻人东方赫连皇甫尉迟公羊澹台公冶宗政濮阳淳于单于太叔申屠
公孙仲孙轩辕令狐钟离宇文长孙慕容鲜于闾丘司徒司空丌官司寇仉督子车颛孙端木巫马公西漆雕乐正壤驷公良
拓拔夹谷宰父谷梁晋楚阎法汝鄢涂钦段干百里东郭南门呼延归海羊舌微生岳帅缑亢况后有琴梁丘左丘东门西门
商牟佘佴佰赏南官墨哈谯笪年爱阳佟第五言福';
•       var nameStr1=str.substring(0,1); *//获取输入的第一个字*
•       var nameStr2=str.substring(0,2); *//获取输入的前两个字*
•       var hasName=(firstName.indexOf(nameStr1)!=-1)||
(firstName.indexOf(nameStr2)!=-1);  *//判断是否含有姓氏*
•       *//判断是否属于百家姓*
•       if(hasName){
•         return true;
•       }else{
•         return false;
•       }
}
```

数字验证

```
number: function(number){
•       var pattern = /^[0-9]$/;
•       if(pattern.test(number)){
•         return true;
•       }else{
•         return false;
•       }
}
```

中文字符验证

```
chinese: function(str){
•       var pattern = /[\u4e00-\u9fa5]/;
•       if(pattern.test(number)){
•         return true;
•       }else{
•         return false;
•       }
}
```

字母数字中文及下划线（一般用于用户名验证）

```
wordsNumber_: function(str){
•       var pattern = /[A-Za-z0-9_\-\u4e00-\u9fa5]+/;
•       if(pattern.test(str)){
•         return true;
•       }else{
•         return false;
•       }
}
```

英文验证

```
english: function(str){
•       var pattern = /^[a-zA-Z]$/;
•       if(pattern.test(str)){
•         return true;
•       }else{
•         return false;
•       }
}
```

电子邮箱验证

```
email: function(str){
•       var pattern = /\w[-\w.+]*@([A-Za-z0-9][-A-Za-z0-9]+\.)+[A-Za-z]{2,14}/;
•       if(pattern.test(str)){
•         return true;
•       }else{
•         return false;
•       }
}
```

手机号码验证非+86

```
cellphone: function(number){
•       var pattern = /^1[3|4|5|7|8]\d{9}$/;
•       if(pattern.test(number)){
•         return true;
•       }else{
•         return false;
•       }
}
```

电话验证

```
tellphone: function(number){
•       var pattern = /[0-9-()（）]{7,18}/;
•       if(pattern.test(number)){
•         return true;
•       }else{
•         return false;
•       }
}
```

网址验证

```
website: function(str){
•       var pattern = /^((https|http|ftp|rtsp|mms)?:\/\/)[^\s]+/;
•       if(pattern.test(str)){
•         return true;
•       }else{
•         return false;
•       }
}
```

IP地址验证

```
ipAddress: function(str){
•       var pattern = /(25[0-5]|2[0-4]\d|[0-1]\d{2}|[1-9]?\d)\.(25[0-5]|2[0-4]\d|
[0-1]\d{2}|[1-9]?\d)\.(25[0-5]|2[0-4]\d|[0-1]\d{2}|[1-9]?\d)\.(25[0-5]|2[0-4]\d|
[0-1]\d{2}|[1-9]?\d)/;
•       if(pattern.test(str)){
•         return true;
•       }else{
•         return false;
•       }
}
```

身份证验证

```
idCard: function(number){
•       var pattern = /\d{17}[\d|x]|\d{15}/;
•       if(pattern.test(number)){
•         return true;
•       }else{
•         return false;
•       }
}
```

邮编验证

```
postCodes: function(number){
•       var pattern = /\d{6}/;
•       if(pattern.test(number)){
•         return true;
•       }else{
•         return false;
•       }
}
```

QQ验证

```
tencentQQ: function(number){
•       var pattern = /[1-9]([0-9]{5,11})/;
•       if(pattern.test(number)){
•         return true;
•       }else{
•         return false;
•       }
}
```

特殊字符验证

```
specialWord: function(str){
•       var pattern = /`~!@#$%^&*()_+-=[]{}\|;:'"<,>.?\//;
•       if(pattern.test(str)){
•         return true;
•       }else{
•         return false;
•       }
}
```

强密码验证(必须包含大小写字母和数字的组合，不能使用特殊字符，长度在8-16之间)

```
strongPassword: function(str){
•       var pattern = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,16}$/;
•       if(pattern.test(str)){
•         return true;
•       }else{
•         return false;
•       }
}
```

日期格式验证 2016-01-01这种格式

```
dataFormat: function(str){
•       var pattern = /^[1-9]{1}\d{3}-[0-1]{1}\d{0,1}-[0-1]{1}\d{0,1}/;
•       if(pattern.test(str)){
•         return true;
•       }else{
•         return false;
•       }
}
```