原生态纯 JavaScript 100 大技巧大收集---你值得拥有

1、原生 JavaScript 实现字符串长度截取

```
function cutstr(str, len) {
    var temp;
    var icount = 0;
    var patrn = /[^\x00-\xff]/;
    var strre = "";
    for (var i = 0; i < str.length; i++) {
        if (icount < len - 1) {
            temp = str.substr(i, 1);
            if (patrn.exec(temp) == null) {
                icount = icount + 1
            } else {
                icount = icount + 2
            }
            strre += temp
        } else {
            break
        }
    }
    return strre + "..."
}
```

2、原生 JavaScript 获取域名主机

```
function getHost(url) {
    var host = "null";
    if(typeof url == "undefined"|| null == url) {
        url = window.location.href;
    }
    var regex = /^\w+\:\/\/([^\/]*).*/;
    var match = url.match(regex);
    if(typeof match != "undefined" && null != match) {
        host = match[1];
    }
    return host;
}
```

3、原生 JavaScript 清除空格

```
String.prototype.trim = function() {
    var reExtraSpace = /^\s*(.*?)\s+$/;
    return this.replace(reExtraSpace, "$1")
```

```
}
```
4、原生 JavaScript 替换全部

```
String.prototype.replaceAll = function(s1, s2) {
    return this.replace(new RegExp(s1, "gm"), s2)
}
```
5、原生 JavaScript 转义 html 标签

```
function HtmlEncode(text) {
    return text.replace(/&/g, '&').replace(/\"/g, '"').replace(/</g, '<').replace(/>/g, '>')
}
```
6、原生 JavaScript 还原 html 标签

```
function HtmlDecode(text) {
    return text.replace(/&/g, '&').replace(/"/g, '\"').replace(/</g, '<').replace(/>/g, '>')
}
```
7、原生 JavaScript 时间日期格式转换

```
Date.prototype.Format = function(formatStr) {
    var str = formatStr;
    var Week = ['日', '一', '二', '三', '四', '五', '六'];
    str = str.replace(/yyyy|YYYY/, this.getFullYear());
    str = str.replace(/yy|YY/, (this.getYear() % 100) > 9 ? (this.getYear() % 100).toString() : '0' +
(this.getYear() % 100));
    str = str.replace(/MM/, (this.getMonth() + 1) > 9 ? (this.getMonth() + 1).toString() : '0' +
(this.getMonth() + 1));
    str = str.replace(/M/g, (this.getMonth() + 1));
    str = str.replace(/w|W/g, Week[this.getDay()]);
    str = str.replace(/dd|DD/, this.getDate() > 9 ? this.getDate().toString() : '0' + this.getDate());
    str = str.replace(/d|D/g, this.getDate());
    str = str.replace(/hh|HH/, this.getHours() > 9 ? this.getHours().toString() : '0' + this.getHours());
    str = str.replace(/h|H/g, this.getHours());
    str = str.replace(/mm/, this.getMinutes() > 9 ? this.getMinutes().toString() : '0' +
this.getMinutes());
    str = str.replace(/m/g, this.getMinutes());
    str = str.replace(/ss|SS/, this.getSeconds() > 9 ? this.getSeconds().toString() : '0' +
this.getSeconds());
    str = str.replace(/s|S/g, this.getSeconds());
    return str
}
```
8、原生 JavaScript 判断是否为数字类型

```
function isDigit(value) {
    var patrn = /^[0-9]*$/;
```

```
    if (patrn.exec(value) == null || value == "") {
        return false
    } else {
        return true
    }
}
```

9、原生 JavaScript 设置 cookie 值

```
function setCookie(name, value, Hours) {
    var d = new Date();
    var offset = 8;
    var utc = d.getTime() + (d.getTimezoneOffset() * 60000);
    var nd = utc + (3600000 * offset);
    var exp = new Date(nd);
    exp.setTime(exp.getTime() + Hours * 60 * 60 * 1000);
    document.cookie = name + "=" + escape(value) + ";path=/;expires=" + exp.toGMTString() +
";domain=360doc.com;"
}
```

10、原生 JavaScript 获取 cookie 值

```
function getCookie(name) {
    var arr = document.cookie.match(new RegExp("(^| )" + name + "=([^;]*)(;|$)"));
    if (arr != null) return unescape(arr[2]);
    return null
}
```

11、原生 JavaScript 加入收藏夹

```
function AddFavorite(sURL, sTitle) {
    try {
        window.external.addFavorite(sURL, sTitle)
    } catch(e) {
        try {
            window.sidebar.addPanel(sTitle, sURL, "")
        } catch(e) {
            alert("加入收藏失败，请使用 Ctrl+D 进行添加")
        }
    }
}
```

12、原生 JavaScript 设为首页

```
function setHomepage() {
    if (document.all) {
        document.body.style.behavior = 'url(#default#homepage)';
        document.body.setHomePage('http://www.jq-school.com')
```

```
    } else if (window.sidebar) {
        if (window.netscape) {
            try {
                netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect")
            } catch(e) {
                alert("该操作被浏览器拒绝，如果想启用该功能，请在地址栏内输入 about:config,然后将项 signed.applets.codebase_principal_support 值该为 true")
            }
        }
        var prefs = Components.classes['@mozilla.org/preferences-service;1'].getService(Components.interfaces.nsIPrefBranch);
        prefs.setCharPref('browser.startup.homepage', 'http://www.jq-school.com')
    }
}
```

13、原生 JavaScript 判断 IE6

```
var ua = navigator.userAgent.toLowerCase();
var isIE6 = ua.indexOf("msie 6") > -1;
if (isIE6) {
    try {
        document.execCommand("BackgroundImageCache", false, true)
    } catch(e) {}
}
```

14、原生 JavaScript 加载样式文件

```
function LoadStyle(url) {
    try {
        document.createStyleSheet(url)
    } catch(e) {
        var cssLink = document.createElement('link');
        cssLink.rel = 'stylesheet';
        cssLink.type = 'text/css';
        cssLink.href = url;
        var head = document.getElementsByTagName('head')[0];
        head.appendChild(cssLink)
    }
}
```

15、原生 JavaScript 返回脚本内容

```
function evalscript(s) {
    if(s.indexOf('<script') == -1) return s;
    var p = /<script[^\>]*?>([^\x00]*?)<\/script>/ig;
    var arr = [];
    while(arr = p.exec(s)) {
```

```
            var p1 = /<script[^\>]*?src=\"([^\>]*?)\"[^\>]*?(reload=\"1\")?(?:charset=\"([\w\-]+?)\")?><\/script>/i;
            var arr1 = [];
            arr1 = p1.exec(arr[0]);
            if(arr1) {
                appendscript(arr1[1], '', arr1[2], arr1[3]);
            } else {
                p1 = /<script(.*?)>([^\x00]+?)<\/script>/i;
                arr1 = p1.exec(arr[0]);
                appendscript('', arr1[2], arr1[1].indexOf('reload=') != -1);
            }
        }
        return s;
}
```

16、原生 JavaScript 清除脚本内容

```
function stripscript(s) {
        return s.replace(/<script.*?>.*?<\/script>/ig, '');
}
```

17、原生 JavaScript 动态加载脚本文件

```
function appendscript(src, text, reload, charset) {
        var id = hash(src + text);
        if(!reload && in_array(id, evalscripts)) return;
        if(reload && $(id)) {
            $(id).parentNode.removeChild($(id));
        }

        evalscripts.push(id);
        var scriptNode = document.createElement("script");
        scriptNode.type = "text/javascript";
        scriptNode.id = id;
        scriptNode.charset = charset ? charset : (BROWSER.firefox ? document.characterSet : document.charset);
        try {
            if(src) {
                scriptNode.src = src;
                scriptNode.onloadDone = false;
                scriptNode.onload = function () {
                    scriptNode.onloadDone = true;
                    JSLOADED[src] = 1;
                };
                scriptNode.onreadystatechange = function () {
                    if((scriptNode.readyState == 'loaded' || scriptNode.readyState == 'complete')
```

```
&& !scriptNode.onloadDone) {
                        scriptNode.onloadDone = true;
                        JSLOADED[src] = 1;
                    }
                };
            } else if(text){
                scriptNode.text = text;
            }
            document.getElementsByTagName('head')[0].appendChild(scriptNode);
        } catch(e) {}
}
```

18、原生 JavaScript 返回按 ID 检索的元素对象

```
function $(id) {
    return !id ? null : document.getElementById(id);
}
```

19、原生 JavaScript 返回浏览器版本内容

```
function browserVersion(types) {
    var other = 1;
    for(i in types) {
        var v = types[i] ? types[i] : i;
        if(USERAGENT.indexOf(v) != -1) {
            var re = new RegExp(v + '(\\/|\\s)([\\d\\.]+)', 'ig');
            var matches = re.exec(USERAGENT);
            var ver = matches != null ? matches[2] : 0;
            other = ver !== 0 && v != 'mozilla' ? 0 : other;
        }else {
            var ver = 0;
        }
        eval('BROWSER.' + i + '= ver');
    }
    BROWSER.other = other;
}
```

20、原生 JavaScript 元素显示的通用方法

```
function $(id) {
    return !id ? null : document.getElementById(id);
}
function display(id) {
    var obj = $(id);
    if(obj.style.visibility) {
        obj.style.visibility = obj.style.visibility == 'visible' ? 'hidden' : 'visible';
    } else {
```

```
        obj.style.display = obj.style.display == '' ? 'none' : '';
    }
}
```

21、原生 JavaScript 中有 insertBefore 方法,可惜却没有 insertAfter 方法?用如下函数实现

```
function insertAfter(newChild,refChild){
    var parElem=refChild.parentNode;
    if(parElem.lastChild==refChild){
        refChild.appendChild(newChild);
    }else{
        parElem.insertBefore(newChild,refChild.nextSibling);
    }
}
```

22、原生 JavaScript 中兼容浏览器绑定元素事件

```
function addEventSamp(obj,evt,fn){
    if (obj.addEventListener) {
        obj.addEventListener(evt, fn, false);
    }else if(obj.attachEvent){
        obj.attachEvent('on'+evt,fn);
    }
}
```

23、原生 JavaScript 光标停在文字的后面，文本框获得焦点时调用

```
function focusLast(){
    var e = event.srcElement;
    var r =e.createTextRange();
    r.moveStart('character',e.value.length);
    r.collapse(true);
    r.select();
}
```

24、原生 JavaScript 检验 URL 链接是否有效

```
function getUrlState(URL){
    var xmlhttp = new ActiveXObject("microsoft.xmlhttp");
    xmlhttp.Open("GET",URL, false);
    try{
        xmlhttp.Send();
    }catch(e){
    }finally{
        var result = xmlhttp.responseText;
        if(result){
            if(xmlhttp.Status==200){
                return(true);
```

```
                }else{
                    return(false);
                }
            }else{
                return(false);
            }
        }
}
```

25、原生 JavaScript 格式化 CSS 样式代码

```
function formatCss(s){//格式化代码
    s = s.replace(/\s*([\{\}\:\;\,])\s*/g, "$1");
    s = s.replace(/;\s*;/g, ";"); //清除连续分号
    s = s.replace(/\,[\s\.\#\d]*{/g, "{");
    s = s.replace(/([^\s])\{([^\s])/g, "$1 {\n\t$2");
    s = s.replace(/([^\s])\}([^\n]*)/g, "$1\n}\n$2");
    s = s.replace(/([^\s]);([^\s\}])/g, "$1;\n\t$2");
    return s;
}
```

26、原生 JavaScript 压缩 CSS 样式代码

```
function yasuoCss (s) {//压缩代码
    s = s.replace(/\/\*(.|\n)*?\*\//g, ""); //删除注释
    s = s.replace(/\s*([\{\}\:\;\,])\s*/g, "$1");
    s = s.replace(/\,[\s\.\#\d]*\{/g, "{"); //容错处理
    s = s.replace(/;\s*;/g, ";"); //清除连续分号
    s = s.match(/^\s*(\S+(\s+\S+)*)\s*$/); //去掉首尾空白
    return (s == null) ? "" : s[1];
}
```

27、原生 JavaScript 获取当前路径

```
var currentPageUrl = "";
if (typeof this.href === "undefined") {
    currentPageUrl = document.location.toString().toLowerCase();
}
else {
    currentPageUrl = this.href.toString().toLowerCase();
}
```

28、原生 JavaScriptIP 转成整型

```
function _ip2int(ip){
    var num = 0;
    ip = ip.split(".");
    num = Number(ip[0]) * 256 * 256 * 256 + Number(ip[1]) * 256 * 256 + Number(ip[2]) * 256 +
```

```
Number(ip[3]);
    num = num >>> 0;
    return num;
}
```

29、原生 JavaScript 整型解析为 IP 地址

```
function _int2iP(num){
    var str;
    var tt = new Array();
    tt[0] = (num >>> 24) >>> 0;
    tt[1] = ((num << 8) >>> 24) >>> 0;
    tt[2] = (num << 16) >>> 24;
    tt[3] = (num << 24) >>> 24;
    str = String(tt[0]) + "." + String(tt[1]) + "." + String(tt[2]) + "." + String(tt[3]);
    return str;
}
```

30、原生 JavaScript 实现 checkbox 全选与全不选

```
function checkAll() {
    var selectall = document.getElementById("selectall");
    var allbox = document.getElementsByName("allbox");
    if (selectall.checked) {
        for (var i = 0; i < allbox.length; i++) {
            allbox[i].checked = true;
        }
    } else {
        for (var i = 0; i < allbox.length; i++) {
            allbox[i].checked = false;
        }
    }
}
```

(31~40)移动篇

31、原生 JavaScript 判断是否移动设备

```
function isMobile(){
    if (typeof this._isMobile === 'boolean'){
        return this._isMobile;
    }
    var screenWidth = this.getScreenWidth();
    var fixViewPortsExperiment = rendererModel.runningExperiments.FixViewport ||
rendererModel.runningExperiments.fixviewport;
    var fixViewPortsExperimentRunning = fixViewPortsExperiment &&
(fixViewPortsExperiment.toLowerCase() === "new");
```

```
        if(!fixViewPortsExperiment){
            if(!this.isAppleMobileDevice()){
                screenWidth = screenWidth/window.devicePixelRatio;
            }
        }
        var isMobileScreenSize = screenWidth < 600;
        var isMobileUserAgent = false;
        this._isMobile = isMobileScreenSize && this.isTouchScreen();
        return this._isMobile;
}
```

32、原生 JavaScript 判断是否移动设备访问

```
function isMobileUserAgent(){
        return
(/iphone|ipod|android.*mobile|windows.*phone|blackberry.*mobile/i.test(window.navigator.u
serAgent.toLowerCase()));
}
```

33、原生 JavaScript 判断是否苹果移动设备访问

```
function isAppleMobileDevice(){
        return (/iphone|ipod|ipad|Macintosh/i.test(navigator.userAgent.toLowerCase()));
}
```

34、原生 JavaScript 判断是否安卓移动设备访问

```
function isAndroidMobileDevice(){
        return (/android/i.test(navigator.userAgent.toLowerCase()));
}
```

35、原生 JavaScript 判断是否 Touch 屏幕

```
function isTouchScreen(){
        return (('ontouchstart' in window) || window.DocumentTouch && document instanceof
DocumentTouch);
}
```

36、原生 JavaScript 判断是否在安卓上的谷歌浏览器

```
function isNewChromeOnAndroid(){
        if(this.isAndroidMobileDevice()){
            var userAgent = navigator.userAgent.toLowerCase();
            if((/chrome/i.test(userAgent))){
                var parts = userAgent.split('chrome/');

                var fullVersionString = parts[1].split(" ")[0];
                var versionString = fullVersionString.split('.')[0];
                var version = parseInt(versionString);
```

```
                if(version >= 27){
                    return true;
                }
            }
        }
        return false;
}
```

37、原生 JavaScript 判断是否打开视窗

```
function isViewportOpen() {
        return !!document.getElementById('wixMobileViewport');
}
```

38、原生 JavaScript 获取移动设备初始化大小

```
function getInitZoom(){
        if(!this._initZoom){
            var screenWidth = Math.min(screen.height, screen.width);
            if(this.isAndroidMobileDevice() && !this.isNewChromeOnAndroid()){
                screenWidth = screenWidth/window.devicePixelRatio;
            }
            this._initZoom = screenWidth /document.body.offsetWidth;
        }
        return this._initZoom;
}
```

39、原生 JavaScript 获取移动设备最大化大小

```
function getZoom(){
        var screenWidth = (Math.abs(window.orientation) === 90) ? Math.max(screen.height,
screen.width) : Math.min(screen.height, screen.width);
        if(this.isAndroidMobileDevice() && !this.isNewChromeOnAndroid()){
            screenWidth = screenWidth/window.devicePixelRatio;
        }
        var FixViewPortsExperiment = rendererModel.runningExperiments.FixViewport ||
rendererModel.runningExperiments.fixviewport;
        var FixViewPortsExperimentRunning = FixViewPortsExperiment && (FixViewPortsExperiment
=== "New" || FixViewPortsExperiment === "new");
        if(FixViewPortsExperimentRunning){
            return screenWidth / window.innerWidth;
        }else{
            return screenWidth / document.body.offsetWidth;
        }
}
```

40、原生 JavaScript 获取移动设备屏幕宽度

```
function getScreenWidth(){
    var smallerSide = Math.min(screen.width, screen.height);
    var fixViewPortsExperiment = rendererModel.runningExperiments.FixViewport ||
rendererModel.runningExperiments.fixviewport;
    var fixViewPortsExperimentRunning = fixViewPortsExperiment &&
(fixViewPortsExperiment.toLowerCase() === "new");
    if(fixViewPortsExperiment){
        if(this.isAndroidMobileDevice() && !this.isNewChromeOnAndroid()){
            smallerSide = smallerSide/window.devicePixelRatio;
        }
    }
    return smallerSide;
}
```

41、原生 JavaScript 完美判断是否为网址

```
function IsURL(strUrl) {
    var regular = /^\b(((https?|ftp):\/\/)?[-a-z0-9]+(\.[-a-z0-
9]+)*\.(?:com|edu|gov|int|mil|net|org|biz|info|name|museum|asia|coop|aero|[a-z][a-
z]|((25[0-5])|(2[0-4]\d)|(1\d\d)|([1-9]\d)|\d))\b(\/[-a-z0-9_:\@&?=+,.!\/~%\$]*)?)$/i
    if (regular.test(strUrl)) {
        return true;
    }
    else {
        return false;
    }
}
```

42、原生 JavaScript 根据样式名称检索元素对象

```
function getElementsByClassName(name) {
    var tags = document.getElementsByTagName('*') || document.all;
    var els = [];
    for (var i = 0; i < tags.length; i++) {
        if (tags[i].className) {
            var cs = tags[i].className.split(' ');
            for (var j = 0; j < cs.length; j++) {
                if (name == cs[j]) {
                    els.push(tags[i]);
                    break
                }
            }
        }
    }
    return els
```

```
}
```

43、原生 JavaScript 判断是否以某个字符串开头

```
String.prototype.startWith = function (s) {
    return this.indexOf(s) == 0
}
```

44、原生 JavaScript 判断是否以某个字符串结束

```
String.prototype.endWith = function (s) {
    var d = this.length - s.length;
    return (d >= 0 && this.lastIndexOf(s) == d)
}
```

45、原生 JavaScript 返回 IE 浏览器的版本号

```
function getIE(){
    if (window.ActiveXObject){
        var v = navigator.userAgent.match(/MSIE ([^;]+)/)[1];
        return parseFloat(v.substring(0, v.indexOf(".")))
    }
    return false
}
```

46、原生 JavaScript 获取页面高度

```
function getPageHeight(){
    var g = document, a = g.body, f = g.documentElement, d = g.compatMode == "BackCompat"
            ? a
            : g.documentElement;
    return Math.max(f.scrollHeight, a.scrollHeight, d.clientHeight);
}
```

47、原生 JavaScript 获取页面 scrollLeft

```
function getPageScrollLeft(){
    var a = document;
    return a.documentElement.scrollLeft || a.body.scrollLeft;
}
```

48、原生 JavaScript 获取页面可视宽度

```
function getPageViewWidth(){
    var d = document, a = d.compatMode == "BackCompat"
            ? d.body
            : d.documentElement;
    return a.clientWidth;
}
```

49、原生 JavaScript 获取页面宽度

```
function getPageWidth(){
    var g = document, a = g.body, f = g.documentElement, d = g.compatMode == "BackCompat"
            ? a
            : g.documentElement;
    return Math.max(f.scrollWidth, a.scrollWidth, d.clientWidth);
}
```

50、原生 JavaScript 获取页面 scrollTop

```
function getPageScrollTop(){
    var a = document;
    return a.documentElement.scrollTop || a.body.scrollTop;
}
```

51、原生 JavaScript 获取页面可视高度

```
function getPageViewHeight() {
    var d = document, a = d.compatMode == "BackCompat"
            ? d.body
            : d.documentElement;
    return a.clientHeight;
}
```

52、原生 JavaScript 跨浏览器添加事件

```
function addEvt(oTarget,sEvtType,fnHandle){
    if(!oTarget){return;}
    if(oTarget.addEventListener){
        oTarget.addEventListener(sEvtType,fnHandle,false);
    }else if(oTarget.attachEvent){
        oTarget.attachEvent("on" + sEvtType,fnHandle);
    }else{
        oTarget["on" + sEvtType] = fnHandle;
    }
}
```

53、原生 JavaScript 跨浏览器删除事件

```
function delEvt(oTarget,sEvtType,fnHandle){
    if(!oTarget){return;}
    if(oTarget.addEventListener){
        oTarget.addEventListener(sEvtType,fnHandle,false);
    }else if(oTarget.attachEvent){
        oTarget.attachEvent("on" + sEvtType,fnHandle);
    }else{
        oTarget["on" + sEvtType] = fnHandle;
    }
```

```
}
```

54、原生 JavaScript 去掉 url 前缀

```
function removeUrlPrefix(a){
   a=a.replace(/：/g,":").replace(/．/g,".").replace(/／/g,"/");
   while(trim(a).toLowerCase().indexOf("http://")==0){
      a=trim(a.replace(/http:\/\//i,""));
   }
   return a;
}
```

55、原生 JavaScript 随机数时间戳

```
function uniqueId(){
      var a=Math.random,b=parseInt;
      return Number(new Date()).toString()+b(10*a())+b(10*a())+b(10*a());
}
```

56、原生 JavaScript 全角半角转换,iCase: 0 全到半，1 半到全，其他不转化

```
function chgCase(sStr,iCase){
      if(typeof sStr != "string" || sStr.length <= 0 || !(iCase === 0 || iCase == 1)){
            return sStr;
      }
      var i,oRs=[],iCode;
      if(iCase){/*半->全*/
            for(i=0; i<sStr.length;i+=1){
                  iCode = sStr.charCodeAt(i);
                  if(iCode == 32){
                        iCode = 12288;
                  }else if(iCode < 127){
                        iCode += 65248;
                  }
                  oRs.push(String.fromCharCode(iCode));
            }
      }else{/*全->半*/
            for(i=0; i<sStr.length;i+=1){
                  iCode = sStr.charCodeAt(i);
                  if(iCode == 12288){
                        iCode = 32;
                  }else if(iCode > 65280 && iCode < 65375){
                        iCode -= 65248;
                  }
                  oRs.push(String.fromCharCode(iCode));
            }
      }
```

```
    return oRs.join("");
}
```

57、原生 JavaScript 确认是否键盘有效输入值

```
function checkKey(iKey){
    if(iKey == 32 || iKey == 229){return true;}/*空格和异常*/
    if(iKey>47 && iKey < 58){return true;}/*数字*/
    if(iKey>64 && iKey < 91){return true;}/*字母*/
    if(iKey>95 && iKey < 108){return true;}/*数字键盘 1*/
    if(iKey>108 && iKey < 112){return true;}/*数字键盘 2*/
    if(iKey>185 && iKey < 193){return true;}/*符号 1*/
    if(iKey>218 && iKey < 223){return true;}/*符号 2*/
    return false;
}
```

58、原生 JavaScript 获取网页被卷去的位置

```
function getScrollXY() {
    return document.body.scrollTop ? {
        x: document.body.scrollLeft,
        y: document.body.scrollTop
    }: {
        x: document.documentElement.scrollLeft,
        y: document.documentElement.scrollTop
    }
}
```

59、原生 JavaScript 另一种正则日期格式化函数+调用方法

```
Date.prototype.format = function(format){ //author: meizz
  var o = {
    "M+" : this.getMonth()+1, //month
    "d+" : this.getDate(),    //day
    "h+" : this.getHours(),   //hour
    "m+" : this.getMinutes(), //minute
    "s+" : this.getSeconds(), //second
    "q+" : Math.floor((this.getMonth()+3)/3),  //quarter
    "S" : this.getMilliseconds() //millisecond
  }
  if(/(y+)/.test(format)) format=format.replace(RegExp.$1,
    (this.getFullYear()+"").substr(4 - RegExp.$1.length));
  for(var k in o)if(new RegExp("("+ k +")").test(format))
    format = format.replace(RegExp.$1,
      RegExp.$1.length==1 ? o[k] :
        ("00"+ o[k]).substr((""+ o[k]).length));
  return format;
```

```
}
alert(new Date().format("yyyy-MM-dd hh:mm:ss"));
```

60、原生 JavaScript 时间个性化输出功能

```
/*
1、< 60s, 显示为“刚刚”
2、>= 1min && < 60 min, 显示与当前时间差“XX 分钟前”
3、>= 60min && < 1day, 显示与当前时间差“今天 XX:XX”
4、>= 1day && < 1year, 显示日期“XX 月 XX 日 XX:XX”
5、>= 1year, 显示具体日期“XXXX 年 XX 月 XX 日 XX:XX”
 */
function timeFormat(time){
    var date = new Date(time)
        , curDate = new Date()
        , year = date.getFullYear()
        , month = date.getMonth() + 1
        , day = date.getDate()
        , hour = date.getHours()
        , minute = date.getMinutes()
        , curYear = curDate.getFullYear()
        , curHour = curDate.getHours()
        , timeStr;

    if(year < curYear){
        timeStr = year +'年'+ month +'月'+ day +'日 '+ hour +':'+ minute;
    }else{
        var pastTime = curDate - date
            , pastH = pastTime/3600000;

        if(pastH > curHour){
            timeStr = month +'月'+ day +'日 '+ hour +':'+ minute;
        }else if(pastH >= 1){
            timeStr = '今天 ' + hour +':'+ minute +'分';
        }else{
            var pastM = curDate.getMinutes() - minute;
            if(pastM > 1){
                timeStr = pastM +'分钟前';
            }else{
                timeStr = '刚刚';
            }
        }
    }
    return timeStr;
}
```

61、原生 JavaScript 解决 offsetX 兼容性问题

```
// 针对火狐不支持 offsetX/Y
function getOffset(e){
   var target = e.target, // 当前触发的目标对象
         eventCoord,
         pageCoord,
         offsetCoord;


   // 计算当前触发元素到文档的距离
   pageCoord = getPageCoord(target);


   // 计算光标到文档的距离
   eventCoord = {
      X : window.pageXOffset + e.clientX,
      Y : window.pageYOffset + e.clientY
   };


   // 相减获取光标到第一个定位的父元素的坐标
   offsetCoord = {
      X : eventCoord.X - pageCoord.X,
      Y : eventCoord.Y - pageCoord.Y
   };
   return offsetCoord;
}


function getPageCoord(element){
   var coord = { X : 0, Y : 0 };
   // 计算从当前触发元素到根节点为止，
   // 各级 offsetParent 元素的 offsetLeft 或 offsetTop 值之和
   while (element){
      coord.X += element.offsetLeft;
      coord.Y += element.offsetTop;
      element = element.offsetParent;
   }
   return coord;
}
```

62、原生 JavaScript 常用的正则表达式

```
//正整数
```

/^[0-9]*[1-9][0-9]*$/;
//负整数
/^-[0-9]*[1-9][0-9]*$/;
//正浮点数
/^(([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*))$/;
//负浮点数
/^(-(([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*)))$/;
//浮点数
/^(-?\d+)(\.\d+)?$/;
//email 地址
/^[\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+$/;
//url 地址
/^[a-zA-z]+://(\w+(-\w+)*)(\.(\w+(-\w+)*))*(\?\S*)?$/;
//年/月/日（年-月-日、年.月.日）
/^(19|20)\d\d[- /.](0[1-9]|1[012])[- /.](0[1-9]|[12][0-9]|3[01])$/;
//匹配中文字符
/[\u4e00-\u9fa5]/;
//匹配帐号是否合法(字母开头，允许 5-10 字节，允许字母数字下划线)
/^[a-zA-Z][a-zA-Z0-9_]{4,9}$/;
//匹配空白行的正则表达式
/\n\s*\r/;
//匹配中国邮政编码
/[1-9]\d{5}(?!\d)/;
//匹配身份证
/\d{15}|\d{18}/;
//匹配国内电话号码
/(\d{3}-|\d{4}-)?(\d{8}|\d{7})?/;
//匹配 IP 地址
/((2[0-4]\d|25[0-5]|[01]?\d\d?)\.){3}(2[0-4]\d|25[0-5]|[01]?\d\d?)/;
//匹配首尾空白字符的正则表达式
/^\s*|\s*$/;
//匹配 HTML 标记的正则表达式
< (\S*?)[^>]*>.*?|< .*? />;
63、原生 JavaScript 实现返回顶部的通用方法

```
function backTop(btnId) {
    var btn = document.getElementById(btnId);
    var d = document.documentElement;
    var b = document.body;
    window.onscroll = set;
    btn.style.display = "none";
    btn.onclick = function() {
        btn.style.display = "none";
        window.onscroll = null;
```

```
        this.timer = setInterval(function() {
            d.scrollTop -= Math.ceil((d.scrollTop + b.scrollTop) * 0.1);
            b.scrollTop -= Math.ceil((d.scrollTop + b.scrollTop) * 0.1);
            if ((d.scrollTop + b.scrollTop) == 0) clearInterval(btn.timer, window.onscroll = set);
        },
        10);
    };
    function set() {
        btn.style.display = (d.scrollTop + b.scrollTop > 100) ? 'block': "none"
    }
};
backTop('goTop');
```

64、原生 JavaScript 获得 URL 中 GET 参数值

```
// 用法：如果地址是 test.htm?t1=1&t2=2&t3=3，那么能取得：GET["t1"], GET["t2"], GET["t3"]
function get_get(){
   querystr = window.location.href.split("?")
   if(querystr[1]){
      GETs = querystr[1].split("&")
      GET =new Array()
      for(i=0;i<GETs.length;i++){
         tmp_arr = GETs[i].split("=")
         key=tmp_arr[0]
         GET[key] = tmp_arr[1]
      }
   }
   return querystr[1];
}
```

65、原生 JavaScript 实现全选通用方法

```
function checkall(form, prefix, checkall) {
    var checkall = checkall ? checkall : 'chkall';
    for(var i = 0; i < form.elements.length; i++) {
        var e = form.elements[i];
        if(e.type=="checkbox"){
            e.checked = form.elements[checkall].checked;
        }
    }
}
```

66、原生 JavaScript 实现全部取消选择通用方法

```
function uncheckAll(form) {
    for (var i=0;i<form.elements.length;i++){
        var e = form.elements[i];
```

```
            if (e.name != 'chkall')
            e.checked=!e.checked;
        }
}
```

67、原生 JavaScript 实现打开一个窗体通用方法

```
function openWindow(url,windowName,width,height){
        var x = parseInt(screen.width / 2.0) - (width / 2.0);
        var y = parseInt(screen.height / 2.0) - (height / 2.0);
        var isMSIE= (navigator.appName == "Microsoft Internet Explorer");
        if (isMSIE) {
            var p = "resizable=1,location=no,scrollbars=no,width=";
            p = p+width;
            p = p+",height=";
            p = p+height;
            p = p+",left=";
            p = p+x;
            p = p+",top=";
            p = p+y;
            retval = window.open(url, windowName, p);
        } else {
            var win = window.open(url, "ZyiisPopup", "top=" + y + ",left=" + x + ",scrollbars=" +
scrollbars + ",dialog=yes,modal=yes,width=" + width + ",height=" + height + ",resizable=no" );
            eval("try { win.resizeTo(width, height); } catch(e) { }");
            win.focus();
        }
}
```

68、原生 JavaScript 判断是否为客户端设备

```
function client(o){
            var b = navigator.userAgent.toLowerCase();
            var t = false;
            if (o == 'isOP'){
                    t = b.indexOf('opera') > -1;
            }
            if (o == 'isIE'){
                    t = b.indexOf('msie') > -1;
            }
            if (o == 'isFF'){
                    t = b.indexOf('firefox') > -1;
            }
            return t;
}
```

69、原生 JavaScript 获取单选按钮的值

```
function get_radio_value(field){
    if(field&&field.length){
        for(var i=0;i<field.length;i++){
            if(field[i].checked){
                return field[i].value;
            }
        }
    }else {
        return ;
    }
}
```

70、原生 JavaScript 获取复选框的值

```
function get_checkbox_value(field){
    if(field&&field.length){
        for(var i=0;i<field.length;i++){
            if(field[i].checked && !field[i].disabled){
                return field[i].value;
            }
        }
    }else {
        return;
    }
}
```

(71~80)验证篇这一篇文章主要是 10 个比较常用表单验证功能，包括了邮箱、危险字符、验证长度、验证网址、验证小数、整数、浮点数等常用的验证，有了这些代码片段，平时的表单验证也可以不需要 jquery 的验证插件了，希望可以帮到大家。。。

71、原生 JavaScript 判断是否为邮箱

```
function isEmail(str){
    var re=/^\w+((-\w+)|(\.\w+))*\@[A-Za-z0-9]+((\.|-)[A-Za-z0-9]+)*\.[A-Za-z0-9]+$/;
    if (re.test(str) != true) {
        return false;
    }else{
        return true;
    }
}
```

72、原生 JavaScript 判断是否有列表中的危险字符

```
function isValidReg(chars){
    var re=/<|>|\[|\]|\{|\}|『|』|※|○|●|◎|§|△|▲|☆|★|◇|◆|□|▼|㊣|￣|⊕|⊙|
〒|ㄅ|ㄆ|ㄇ|ㄈ|ㄉ|ㄊ|ㄋ|ㄌ|ㄍ|ㄎ|ㄏ|ㄐ|ㄑ|ㄒ|ㄓ|ㄔ|ㄕ|ㄖ|ㄗ|ㄘ|ㄙ|ㄚ|ㄛ|ㄜ|ㄝ|ㄞ|
```

```
乁|马|乜|尢|厶|儿|丨|乂|凵|■|▂|▄|\*|@|#|\^|\\/;
    if (re.test( chars) == true) {
        return false;
    }else{
        return true;
    }
}
```

73、原生 JavaScript 判断字符串是否大于规定的长度

```
function isValidLength(chars, len) {
    if (chars.length < len) {
        return false;
    }
    return true;
}
```

74、原生 JavaScript 判断字符串是为网址不区分大小写

```
function isValidURL( chars ) {
    var re=/^([hH][tT]{2}[pP]:\/\/|[hH][tT]{2}[pP][sS]:\/\/)(\S+\.\S+)$/;
    if (!isNULL(chars)) {
        chars = jsTrim(chars);
        if (chars.match(re) == null)
            return false;
        else
            return true;
    }
    return false;
}
```

75、原生 JavaScript 判断字符串是否为小数

```
function isValidDecimal( chars ) {
    var re=/^\d*\.?\d{1,2}$/;
    if (chars.match(re) == null)
        return false;
    else
        return true;
}
```

76、原生 JavaScript 判断字符串是否为整数

```
function isNumber( chars ) {
    var re=/^\d*$/;
    if (chars.match(re) == null)
        return false;
    else
```

```
        return true;
}
```

77、原生 JavaScript 判断字符串是否为浮点数

```
function isFloat( str ) {
    for(i=0;i<str.length;i++)    {
        if ((str.charAt(i)<"0" || str.charAt(i)>"9")&& str.charAt(i) != '.'){
            return false;
        }
    }
    return true;
}
```

78、原生 JavaScript 判断字符是否为 A-Za-z 英文字母

```
function isLetters( str ){
    var re=/^[A-Za-z]+$/;
    if (str.match(re) == null)
        return false;
    else
        return true;
}
```

79、原生 JavaScript 判断字符串是否邮政编码

```
function isValidPost( chars ) {
    var re=/^\d{6}$/;
    if (chars.match(re) == null)
        return false;
    else
        return true;
}
```

80、原生 JavaScript 判断字符是否空 NULL

```
function isNULL( chars ) {
    if (chars == null)
        return true;
    if (jsTrim(chars).length==0)
        return true;
    return false;
}
```

81、原生 JavaScript 用正则表达式提取页面代码中所有网址

```
var aa = document.documentElement.outerHTML.match(/(url\(|src=|href=)[\"\']*([^\"\'\(\)\<\>\[\] ]+)[\"\'\)]*|(http:\/\/[\w\-
```

```
\.]+[^\"\'\(\)\<\>\[\] ]+)/ig).join("\r\n").replace(/^(src=|href=|url\()[\"\']*|[\"\'\>\) ]*$/igm,"");
alert(aa)
```

82、原生 JavaScript 用正则表达式清除相同的数组(低效率)

```
Array.prototype.unique=function(){
    return this.reverse().join(",").match(/([^,]+)(?!.*\1)/ig).reverse();
}
```

83、原生 JavaScript 用正则表达式清除相同的数组(高效率)

```
String.prototype.unique=function(){
    var x=this.split(/[\r\n]+/);
    var y='';
    for(var i=0;i<x.length;i++){
        if(!new RegExp("^"+x[i].replace(/([^\w])/ig,"\\$1")+"$","igm").test(y)){
            y+=x[i]+"\r\n"
        }
    }
    return y
}
```

84、原生 JavaScript 用正则表达式按字母排序，对每行进行数组排序

```
function SetSort(){
    var text=K1.value.split(/[\r\n]/).sort().join("\r\n");//顺序
    var test=K1.value.split(/[\r\n]/).sort().reverse().join("\r\n");//反序
    K1.value=K1.value!=text?text:test;
}
```

85、原生 JavaScript 字符串反序

```
function IsReverse(text){
    return text.split('').reverse().join('');
}
```

86、原生 JavaScript 用正则表达式清除 html 代码中的脚本

```
function clear_script(){
   K1.value=K1.value.replace(/<script.*?>[\s\S]*?<\/script>|\s+on[a-zA-
Z]{3,16}\s?=\s?"[\s\S]*?"|\s+on[a-zA-Z]{3,16}\s?=\s?'[\s\S]*?'|\s+on[a-zA-
Z]{3,16}\s?=[^ >]+/ig,"");
}
```

87、原生 JavaScript 动态执行 JavaScript 脚本

```
function javascript(){
    try{
       eval(K1.value);
    }catch(e){
```

```
            alert(e.message);
        }
}
```

88、原生 JavaScript 动态执行 VBScript 脚本

```
function vbscript(){
        try{
            var script=document.getElementById("K1").value;
            if(script.trim()=="")return;
            window.execScript('On Error Resume Next \n'+script+'\n If Err.Number<>0 Then \n MsgBox "请输入正确的 VBScript 脚本!",48,"脚本错误!" \n End If',"vbscript")
        }catch(e){
            alert(e.message);
        }
}
```

89、原生 JavaScript 实现金额大写转换函数

```
function transform(tranvalue) {
    try {
        var i = 1;
        var dw2 = new Array("", "万", "亿"); //大单位
        var dw1 = new Array("拾", "佰", "仟"); //小单位
        var dw = new Array("零", "壹", "贰", "叁", "肆", "伍", "陆", "柒", "捌", "玖"); //整数部分用
        //以下是小写转换成大写显示在合计大写的文本框中
        //分离整数与小数
        var source = splits(tranvalue);
        var num = source[0];
        var dig = source[1];
        //转换整数部分
        var k1 = 0; //计小单位
        var k2 = 0; //计大单位
        var sum = 0;
        var str = "";
        var len = source[0].length; //整数的长度
        for (i = 1; i <= len; i++) {
            var n = source[0].charAt(len - i); //取得某个位数上的数字
            var bn = 0;
            if (len - i - 1 >= 0) {
                bn = source[0].charAt(len - i - 1); //取得某个位数前一位上的数字
            }
            sum = sum + Number(n);
            if (sum != 0) {
                str = dw[Number(n)].concat(str); //取得该数字对应的大写数字，并插入到 str 字符串
的前面
```

```
            if (n == '0') sum = 0;
        }
        if (len - i - 1 >= 0) { //在数字范围内
            if (k1 != 3) { //加小单位
                if (bn != 0) {
                    str = dw1[k1].concat(str);
                }
                k1++;
            } else { //不加小单位，加大单位
                k1 = 0;
                var temp = str.charAt(0);
                if (temp == "万" || temp == "亿") //若大单位前没有数字则舍去大单位
                str = str.substr(1, str.length - 1);
                str = dw2[k2].concat(str);
                sum = 0;
            }
        }
        if (k1 == 3) //小单位到千则大单位进一
        {
            k2++;
        }
    }
    //转换小数部分
    var strdig = "";
    if (dig != "") {
        var n = dig.charAt(0);
        if (n != 0) {
            strdig += dw[Number(n)] + "角"; //加数字
        }
        var n = dig.charAt(1);
        if (n != 0) {
            strdig += dw[Number(n)] + "分"; //加数字
        }
    }
    str += "元" + strdig;
  } catch(e) {
    return "0 元";
  }
  return str;
}
//拆分整数与小数
function splits(tranvalue) {
  var value = new Array('', '');
  temp = tranvalue.split(".");
```

```
    for (var i = 0; i < temp.length; i++) {
        value[i] = temp[i];
    }
    return value;
}
```

90、原生 JavaScript 常用的正则表达式大收集

[plain] view plaincopy

匹配中文字符的正则表达式： [\u4e00-\u9fa5]

匹配双字节字符（包括汉字在内）：[^\x00-\xff]

匹配空行的正则表达式：\n[\s|]*\r

匹配 HTML 标记的正则表达式：<(.*)>.*<\/\1>|<(.*) \/>

匹配首尾空格的正则表达式：(^\s*)|(\s*$)

匹配 IP 地址的正则表达式：/(\d+)\.(\d+)\.(\d+)\.(\d+)/g

匹配 Email 地址的正则表达式：\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*

匹配网址 URL 的正则表达式：http://(/[\w-]+\.)+[\w-]+(/[\w- ./?%&=]*)?

sql 语句：^(select|drop|delete|create|update|insert).*$

非负整数：^\d+$

正整数：^[0-9]*[1-9][0-9]*$

非正整数：^((-\d+)|(0+))$

负整数：^-[0-9]*[1-9][0-9]*$

整数：^-?\d+$

非负浮点数：^\d+(\.\d+)?$

正浮点数：^((0-9)+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*))$

非正浮点数：^((-\d+\.\d+)?)|(0+(\.0+)?))$

英文字符串：^[A-Za-z]+$

英文大写串：^[A-Z]+$

英文小写串：^[a-z]+$

英文字符数字串：^[A-Za-z0-9]+$

英数字加下划线串：^\w+$

E-mail 地址：^[\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+$

URL：^[a-zA-Z]+://(\w+(-\w+)*)(\.(\w+(-\w+)*))*(\?\s*)?$ 或：^http:\/\/[A-Za-z0-9]+\.[A-Za-z0-9]+[\/=\?%\-&_~`@[\]\':+!]*([^<>\"\"])*$

邮政编码：^[1-9]\d{5}$

电话号码：^((\(\d{2,3}\))|(\d{3}\-))?(\(0\d{2,3}\)|0\d{2,3}-)?[1-9]\d{6,7}(\-\d{1,4})?$

手机号码：^((\(\d{2,3}\))|(\d{3}\-))?13\d{9}$

双字节字符（包括汉字在内）：^\x00-\xff

匹配首尾空格：(^\s*)|(\s*$)

匹配 HTML 标记：<(.*)>.*<\/\1>|<(.*) \/>

匹配空行：\n[\s|]*\r

提取信息中的网络链接：(h|H)(r|R)(e|E)(f|F) *= *('|")?(\w|\\|\/|\.)+('|"| *|>)?

提取信息中的邮件地址：\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*

提取信息中的图片链接：(s|S)(r|R)(c|C) *= *('|")?(\w|\\|\/|\.)+('|"| *|>)?

提取信息中的 IP 地址：(\d+)\.(\d+)\.(\d+)\.(\d+)

提取信息中的中国手机号码：(86)*0*13\d{9}

提取信息中的中国固定电话号码：(\(\d{3,4}\)|\d{3,4}-|\s)?\d{8}
提取信息中的中国电话号码（包括移动和固定电话）：(\(\d{3,4}\)|\d{3,4}-|\s)?\d{7,14}
提取信息中的中国邮政编码：[1-9]{1}(\d+){5}
提取信息中的浮点数（即小数）：(-?\d*)\.?\d+
提取信息中的任何数字 ：(-?\d*)(\.\d+)?
IP：(\d+)\.(\d+)\.(\d+)\.(\d+)
电话区号：^0\d{2,3}$
腾讯 QQ 号：^[1-9]*[1-9][0-9]*$
帐号（字母开头，允许 5-16 字节，允许字母数字下划线）：^[a-zA-Z][a-zA-Z0-9_]{4,15}$
中文、英文、数字及下划线：^[\u4e00-\u9fa5_a-zA-Z0-9]+$
91、原生 JavaScript 实现窗体改变事件 resize 的操作（兼容所以的浏览器）

```
(function(){
    var fn = function(){
        var w = document.documentElement ? document.documentElement.clientWidth :
document.body.clientWidth
            ,r = 1255
            ,b = Element.extend(document.body)
            ,classname = b.className;
        if(w < r){
            //当窗体的宽度小于 1255 的时候执行相应的操作
        }else{
            //当窗体的宽度大于 1255 的时候执行相应的操作
        }
    }
    if(window.addEventListener){
        window.addEventListener('resize', function(){ fn(); });
    }else if(window.attachEvent){
        window.attachEvent('onresize', function(){ fn(); });
    }
    fn();
})();
```

92、原生 JavaScript 用正则清除空格分左右

```
function ltrim(s){ return s.replace( /^(\s*|    *)/, ""); }
function rtrim(s){ return s.replace( /(\s*|    *)$/, ""); }
function trim(s){ return ltrim(rtrim(s));}
```

93、原生 JavaScript 判断变量是否空值

```
/**
 * 判断变量是否空值
 * undefined, null, '', false, 0, [], {} 均返回 true，否则返回 false
 */
function empty(v){
```

```
    switch (typeof v){
        case 'undefined' : return true;
        case 'string'    : if(trim(v).length == 0) return true; break;
        case 'boolean'   : if(!v) return true; break;
        case 'number'    : if(0 === v) return true; break;
        case 'object'    :
            if(null === v) return true;
            if(undefined !== v.length && v.length==0) return true;
            for(var k in v){return false;} return true;
            break;
    }
    return false;
}
```

94、原生 JavaScript 实现 base64 解码

```
function base64_decode(data){
    var b64 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";
    var o1, o2, o3, h1, h2, h3, h4, bits, i = 0,ac = 0,dec = "",tmp_arr = [];
    if (!data) { return data; }
    data += '';
    do {
        h1 = b64.indexOf(data.charAt(i++));
        h2 = b64.indexOf(data.charAt(i++));
        h3 = b64.indexOf(data.charAt(i++));
        h4 = b64.indexOf(data.charAt(i++));
        bits = h1 << 18 | h2 << 12 | h3 << 6 | h4;
        o1 = bits >> 16 & 0xff;
        o2 = bits >> 8 & 0xff;
        o3 = bits & 0xff;
        if (h3 == 64) {
            tmp_arr[ac++] = String.fromCharCode(o1);
        } else if (h4 == 64) {
            tmp_arr[ac++] = String.fromCharCode(o1, o2);
        } else {
            tmp_arr[ac++] = String.fromCharCode(o1, o2, o3);
        }
    } while (i < data.length);
    dec = tmp_arr.join('');
    dec = utf8_decode(dec);
    return dec;
}
```

95、原生 JavaScript 实现 utf8 解码

```
function utf8_decode(str_data){
```

```
    var tmp_arr = [],i = 0,ac = 0,c1 = 0,c2 = 0,c3 = 0;str_data += '';
    while (i < str_data.length) {
        c1 = str_data.charCodeAt(i);
        if (c1 < 128) {
            tmp_arr[ac++] = String.fromCharCode(c1);
            i++;
        } else if (c1 > 191 && c1 < 224) {
            c2 = str_data.charCodeAt(i + 1);
            tmp_arr[ac++] = String.fromCharCode(((c1 & 31) << 6) | (c2 & 63));
            i += 2;
        } else {
            c2 = str_data.charCodeAt(i + 1);
            c3 = str_data.charCodeAt(i + 2);
            tmp_arr[ac++] = String.fromCharCode(((c1 & 15) << 12) | ((c2 & 63) << 6) | (c3 &
63));
            i += 3;
        }
    }
    return tmp_arr.join('');
}
```

96、原生 JavaScript 获取窗体可见范围的宽与高

```
function getViewSize(){
    var de=document.documentElement;
    var db=document.body;
    var viewW=de.clientWidth==0 ?  db.clientWidth : de.clientWidth;
    var viewH=de.clientHeight==0 ?  db.clientHeight : de.clientHeight;
    return Array(viewW ,viewH);
}
```

97、原生 JavaScript 判断 IE 版本号（既简洁、又向后兼容！）

```
var _IE = (function(){
    var v = 3, div = document.createElement('div'), all = div.getElementsByTagName('i');
    while (
        div.innerHTML = '<!--[if gt IE ' + (++v) + ']><i></i><![endif]-->',
        all[0]
    );
    return v > 4 ? v : false ;
}());
```

98、原生 JavaScript 获取浏览器版本号

```
function browserVersion(types) {
    var other = 1;
    for (i in types) {
```

```
            var v = types[i] ? types[i] : i;
            if (USERAGENT.indexOf(v) != -1) {
                var re = new RegExp(v + '(\\/|\\s|:)([\\d\\.]+)', 'ig');
                var matches = re.exec(USERAGENT);
                var ver = matches != null ? matches[2] : 0;
                other = ver !== 0 && v != 'mozilla' ? 0 : other;
            } else {
                var ver = 0;
            }
            eval('BROWSER.' + i + '= ver');
        }
        BROWSER.other = other;
}
```

99、原生 JavaScript 半角转换为全角函数

```
function ToDBC(str){
   var result = '';
   for(var i=0; i < str.length; i++){
      code = str.charCodeAt(i);
      if(code >= 33 && code <= 126){
         result += String.fromCharCode(str.charCodeAt(i) + 65248);
      }else if (code == 32){
         result += String.fromCharCode(str.charCodeAt(i) + 12288 - 32);
      }else{
         result += str.charAt(i);
      }
   }
 return result;
}
```

100、原生 JavaScript 全角转换为半角函数

```
function ToCDB(str){
   var result = '';
   for(var i=0; i < str.length; i++){
      code = str.charCodeAt(i);
      if(code >= 65281 && code <= 65374){
         result += String.fromCharCode(str.charCodeAt(i) - 65248);
      }else if (code == 12288){
         result += String.fromCharCode(str.charCodeAt(i) - 12288 + 32);
      }else{
         result += str.charAt(i);
      }
   }
 return result;
```

}