JS常用函数（三）

二十一、JS获取地址栏参数的值

> JS获取地址栏参数的值

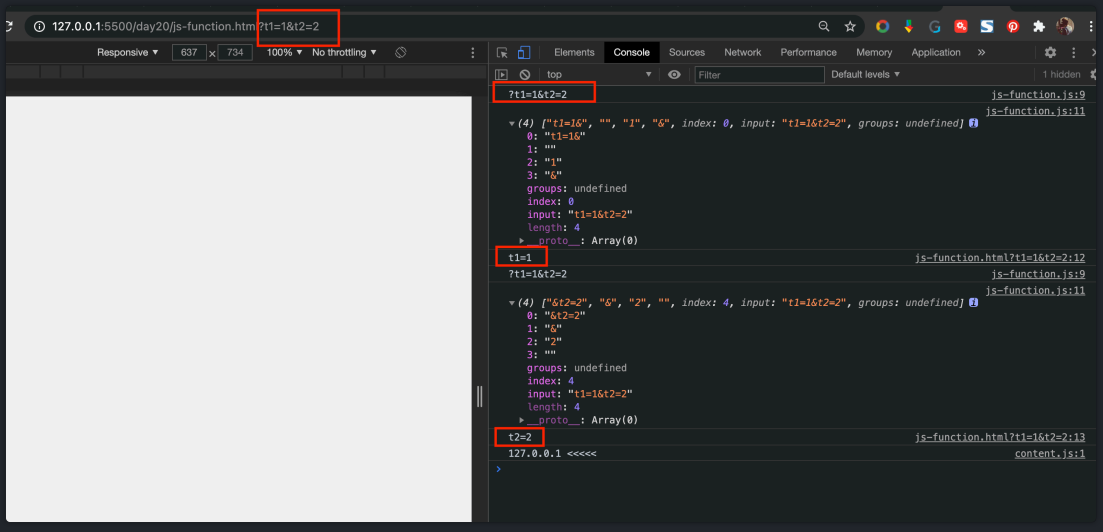
代码如下：

```
/**
 * JS获取地址栏参数的值
 * @param name 对应的参数
 * @returns {*} 如果有，则返回参数值，没有则返回null
 */
function getUrlParam(name){
    var reg = new RegExp("(^|&)" + name + "=([^&]*)
(&|$)");
    var r = window.location.search.substr(1).match(reg);
    if (r != null) {
        return unescape(r[2]);
    } else {
        return null;
    }
}
```

调用方法：

```
// 若当前的URL地址为：****.html?t1=1&t2=2&t3=3
console.log(getUrlParam("t1")); // 1
```

如下图使用：



二十二、JS字符串反序

JS字符串反序

代码如下：

```
/**
 * JS字符串反序
 * @param text 需要进行反序的字符串
 * @returns {string} 返回反序之后的字符串
 * @constructor
 */
function reverseStr(text){
    return text.split('').reverse().join('');
}
```

调用方法：

```
console.log(reverseStr("Hello!"));  //  !olleH
```

二十三、JS现金额转大写

JS现金额转大写

代码如下：

```
/**
 * @desc   JS现金额转大写
 * @param  {Number} n
 * @return {String}
 */
function digitUppercase (n) {
  let fraction = ['角', '分']
  let digit = [
    '零', '壹', '贰', '叁', '肆',
    '伍', '陆', '柒', '捌', '玖'
  ]
  let unit = [
    ['元', '万', '亿'],
    ['', '拾', '佰', '仟']
  ]
  let head = n < 0 ? '欠' : ''
  n = Math.abs(n)
  let s = ''
  for (let i = 0; i < fraction.length; i++) {
    s += (digit[Math.floor(n * 10 * Math.pow(10, i)) % 10]
+ fraction[i]).replace(/零./, '')
  }
  s = s || '整'
  n = Math.floor(n)
  for (let i = 0; i < unit[0].length && n > 0; i++) {
    let p = ''
    for (let j = 0; j < unit[1].length && n > 0; j++) {
      p = digit[n % 10] + unit[1][j] + p
      n = Math.floor(n / 10)
    }
    s = p.replace(/(零.)*零$/, '').replace(/^$/, '零') +
unit[0][i] + s
  }
  return head + s.replace(/(零.)*零元/, '元')
    .replace(/(零.)+/g, '零')
    .replace(/^整$/, '零元整')
}
```

调用方法：

```
console.log(digitUppercase(1023456789.56)); // 壹拾亿贰仟叁佰肆拾伍万陆仟柒佰捌拾玖元伍角伍分
```

二十四、JS限制输入小数的时候只允许数字

JS允许输入小数位的数字

代码如下：

```
/**
 * JS允许输入小数位的数字
 * @param curObj
 */
function clearNoFloat(curObj){
    curObj.value = curObj.value.replace(/[^\d.]/g,"");  //清除“数字”和“.”以外的字符
    curObj.value = curObj.value.replace(/^\./g,"");  //验证第一个字符是数字而不是.
    curObj.value = curObj.value.replace(/\.{2,}/g,"."); //只保留第一个. 清除多余的.
    curObj.value = curObj.value.replace(".","$#$").replace(/\./g,"").replace("$#$",".");
}
```

调用方法：

```
<input type="text" onkeyup="clearNoFloat(this)"> // JS允许输入小数，只能是数字，如果是其他字符，则会替换掉
```

二十五、JS限制输入只能是整数

JS限制只能是整数

代码如下：

```
/**
 * JS限制只能是整数，不能是小数
 * @param curObj
 */
function clearNoInt(curObj){
    curObj.value = curObj.value.replace(/[^\d]/g,"");  //清除“数字”和“.”以外的字符
}
```

调用方法：

```
<input type="text" onkeyup="clearNoInt(this)">  //文本框输入非数字会被清除替换
```

二十六、JS检测字符串是否为空

JS检测字符串是否为空

代码如下：

```
/**
 * JS检测字符串是否为空
 * @param str
 * @returns {boolean}
 */
function checkIsEmpty(str) {
    if (null != str && undefined != str && "" != str) {
        return false;
    }
    return true;
}
```

调用方法：

```
console.log(checkIsEmpty(""));  //  true
```

二十七、JS四舍五入保留小数

> JS将数值四舍五入到保留的小数位数

代码如下：

```
/**
 * 将数值四舍五入到保留的小数位数
 * @param num 待四舍五入数值
 * @param len 保留小数位数
 * @returns {number}
 */
function getRound(num,len) {
    return Math.round(num * Math.pow(10, len)) /
Math.pow(10, len);
}
```

调用方法：

```
getRound(6.123456,4); // 6.1235
```

二十八、JS切割小数位数

> JS切割相应小数点后位数，并将小数点后多余的0 清空

代码如下：

```
/**
 * 切割相应小数点后位数，并将小数点后多余的0 清空
 * @param val 需要切割的数值
 * @param num 需要的小数位数
 * @returns {string}
 */
function cutRoundNum (val, num) {
```

```
  let value = val.toString()
  value = value.substr(0, value.indexOf('.') + 1) +
value.substr(value.indexOf('.') + 1, num)

  let reg = value.match(/\d+\.\d+/g)
  for (let index in reg) {
    value = value.replace(reg[index],
parseFloat(reg[index]))
  }
  return value
}
```

调用方法：

```
cutRoundNum(2.333000,4);  // 2.333
```

二十九、JS判断是否是邮箱的正确格式

> **JS判断是否是邮箱的正确格式**

代码如下：

```
/**
 * 判断是否是邮箱的正确格式
 * @param str 对应的需要验证的邮箱地址
 * @returns {boolean} 返回真或假
 */
function isEmail (str) {
  let reg = /^([a-z0-9A-Z]+[-|\.]?)+[a-z0-9A-Z]@([a-z0-9A-
Z]+(-[a-z0-9A-Z]+)?\.)+[a-zA-Z]{2,}$/
  return reg.test(str)
}
```

调用方法：

```
console.log(isEmail("ye21st@gmail.com"))  //true
console.log(isEmail("ye21st!gmail.com"))  //false
```

三十、JS将手机号格式化

> JS将手机号格式化，中间部分以 * 号代替

代码如下：

```
/**
 * JS将手机号格式化，中间部分以 * 号代替
 * @param phone
 * @returns {string | * | void}
 */
function phoneToStar( phone ) {
    return phone.replace(/(\d{3})\d{4}(\d{4})/,
"$1****$2");
}
```

调用方法：

```
console.log(phoneToStar('13951905171')) // 139****5171
```

三十一、JS检查用户名是否满足要求

> JS检查用户名是否满足要求，只能是英文或英文数字

代码如下：

```
/**
 * JS检查用户名是否满足要求，只能是英文或英文数字
 * @returns {*}
 */
function checkLoginName(loginName) {
    return /^[A-Za-z0-9]*$/.test(loginName) && !/(^\d*$)|
(^\S+\s+\S+$)/.test(loginName);
}
```

调用方法：

```
console.log(checkLoginName("ye21st"))  // true
console.log(checkLoginName("sam!"))    // false
```

三十二、JS验证密码，必须是字母和数字结合

> JS验证密码，必须是字母和数字结合

代码如下：

```
/**
 * JS验证密码，必须是字母和数字结合
 * @param password 密码
 * @returns {boolean} 返回true或false
 */
function checkPasswordValidate(password) {
    return /^(?!^\d+$)(?!^[a-zA-Z]+$)[0-9a-zA-Z]
{6,20}$/.test(password);
}
```

调用方法：

```
console.log(checkPasswordValidate("dasdas1132156"))
 //true
console.log(checkPasswordValidate("dsadasdas"))
 //false
```

三十三、JS检查输入的邮政编码是否正确

> JS检查输入的邮政编码是否正确

代码如下：

```
/**
 * JS检查输入的邮政编码是否正确
 * @param str
 * @returns {Boolean}
 */
function checkPostcode(str){
    if (str.match(/^[1-9][0-9]{5}$/) == null) {
        return false;
    } else {
        return true;
    }
}
```

调用方法：

```
console.log(checkPostcode("423000"))   //true
console.log(checkPostcode("029000"))   //false
```

三十四、JS验证是否为正整数

> JS验证是否为正整数

代码如下：

```
/**
 * JS验证是否为正整数
 * @param str
 * @returns
 */
function checkNumber(str){
    return /^[1-9]\d*$/.test(str);
}
```

调用方法：

```
console.log(checkNumber(100))   //true
console.log(checkNumber(-100))  //false
```

三十五、JS验证是否包含全角

JS验证是否包含全角

代码如下：

```
/**
 * JS判断是否包含全角
 * @param str
 */
function checkHasFull(str) {
    for ( var i = 0; i < str.length; i++) {
        strCode = str.charCodeAt(i);
        if ((strCode > 65248) || (strCode == 12288)) {
            return true;
            break;
        }
    }
    return false;
}
```

调用方法：

```
console.log(checkHasFull("AaBb1234@#%&; , . : "))   //true
console.log(checkHasFull("AaBb1234@#%&;,.:"))     //false
```

三十六、JS检查字符串是否全部是数字或者英文

> JS检查输入的一串字符是否全部是数字或者英文

代码如下：

```
/**
 * JS检查输入的一串字符是否全部是数字或者英文
 * @param str 字符串
 * @returns true 或 false; true表示为数字或者英文
 */
function checkEnNum(str) {
    for ( var i = 0; i < str.length; i++) {
        var strTmp = str.charAt(i);
        if (!(/[A-Za-z0-9]/.test(strTmp))) {
            return false;
        }
    }
    return true;
}
```

调用方法：

```
console.log(checkEnNum("dasdasdas"))    //true
console.log(checkEnNum("dasdasdas13615!"))  //false
```

三十七、JS将数字转换成字符串的通用方法

> JS将数字转换成字符串的通用方法

代码如下：

```
/**
 * JS将数字转换成字符串的通用方法
```

```
 * 说明:直接使用 toFixed 方法会进行四舍五入，因此写一个将数字转换为指定小数位数字符串的方法
 * @param sourceData 源数据
 * @param decimalLen 小数的位数
 */
function numberToString(sourceData, decimalLen)
{
    decimalLen = typeof(decimalLen) == undefined ? 0 : decimalLen;
    let result = sourceData + ""
    let integerStr = null // 整数部分
    let decimalStr = null // 小数部分
    if(result.indexOf(".") == -1)
    {
        result = Number(result).toFixed(decimalLen);
    }
    else
    {
        integerStr = result.substring(0, result.indexOf(".")); // 整数部分
        decimalStr = /\.\d+/.exec(result); // 小数部分
        decimalStr = Number(decimalStr);
        decimalStr = decimalStr.toPrecision(decimalLen).substr(0, decimalLen + 2);
        result = integerStr + decimalStr.substr(1);
    }
    return result;
}
```

调用方法:

```
console.log(numberToString(233, 2))  // 233.00
```

三十八、JS判断浏览器

> JS判断浏览器

代码如下:

```
/**
 * JS判断浏览器
 * @returns {string}
 */
function getOs() {
    if (navigator.userAgent.indexOf("MSIE 8.0") > 0) {
        return "MSIE8";
    } else if (navigator.userAgent.indexOf("MSIE 6.0") >
0) {
        return "MSIE6";
    } else if (navigator.userAgent.indexOf("MSIE 7.0") >
0) {
        return "MSIE7";
    } else if (isFirefox =
navigator.userAgent.indexOf("Firefox") > 0) {
        return "Firefox";
    }
    if (navigator.userAgent.indexOf("Chrome") > 0) {
        return "Chrome";
    } else {
        return "Other";
    }
}
```

调用方法（在浏览器中调用！）

```
console.log(getOs()) // Chrome
```

三十九、JS手机类型判断

> JS手机类型判断

代码如下：

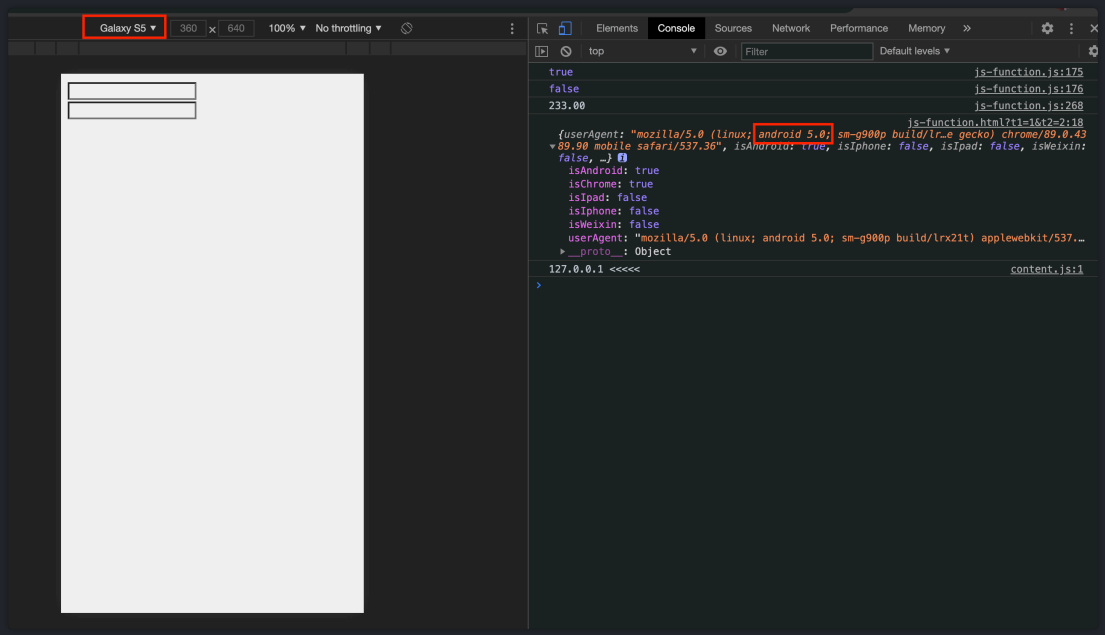
```
/**
 * JS手机类型判断
 * @type {{userAgent: string, isAndroid: boolean,
isIphone: boolean, isIpad: boolean, isWeixin: boolean,
isChrome: boolean}}
 */
var BrowserInfo = {
    userAgent: navigator.userAgent.toLowerCase(),
    isAndroid:
Boolean(navigator.userAgent.match(/android/ig)),
    isIphone:
Boolean(navigator.userAgent.match(/iphone|ipod/ig)),
    isIpad: Boolean(navigator.userAgent.match(/ipad/ig)),
    isWeixin:
Boolean(navigator.userAgent.match(/MicroMessenger/ig)),

 isChrome:Boolean(navigator.userAgent.match(/chrome/ig)),
}
```

调用方法:

```
console.log(BrowserInfo)
```

浏览器控制台切换机型，可以查看到不同结果



四十、JS将object转为formData

> JS将object转为formData，方便post提交

代码如下：

```
/**
 * JS 将object转为form data，方便post提交
 * @param {Object} obj [数据对象]
 * @return {String}
 */
function encodeFormData (obj) {
  if (!obj) return
  let pairs = []
  for (let name in obj) {
    if (!obj.hasOwnProperty(name)) continue
    if (typeof obj[name] == 'function') continue
    var value = obj[name].toString()
    name = encodeURIComponent(name.replace('%20', '+'))
    value = encodeURIComponent(value.replace('%20', '+'))
    pairs.push(name + '=' + value)
  }
  return pairs.join('&')
}
```

调用方法

```
console.log(encodeFormData({ id: 1, name: 'hello' }))   //
id=1&name=hello
```