JS常用函数（一）

一、JS生成随机字符串

> 在创建订单时，我们希望订单号是唯一的，那么，则可能需要用到：随机字符串+时间格式

代码如下：

```
/**
 * randomWord 产生任意长度随机字母数字组合
 * @param randomFlag 是否任意长度 min-任意长度最小位[固定位数] max-任意长度最大位
 * @param min
 * @param max
 * @returns {string}
 */
function randomWord(randomFlag, min, max){
    let str = "",
        range = min,
        arr = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'];

    // 随机产生
    if(randomFlag){
        range = Math.round(Math.random() * (max-min)) + min;
    }
    for(let i=0; i<range; i++){
        pos = Math.round(Math.random() * (arr.length-1));
        str += arr[pos];
    }
    return str;
```

```
}
```

调用方法

```
// 生成 3 - 32 位随即字符串
console.log(randomWord(true, 3, 32))
//  生成 32 位随机字符串
console.log(randomWord(false, 32))
```

二、JS生成UUID

> **很多时候，我们需要保证标识符唯一，那么，使用UUID是不错的选择。**

代码如下：

```
/**
 * generateUUID 生成UUID
 * @returns {string} 返回字符串
 */
function generateUUID(){
    let d = new Date().getTime();
    let uuid = 'xxxxxxxx-xxxx-4xxx-yxxx-
xxxxxxxxxxxx'.replace(/[xy]/g, function(c) {
        let r = (d + Math.random()*16)%16 | 0;
        d = Math.floor(d/16);
        return (c=='x' ? r : (r&0x7|0x8)).toString(16);
    });
    return uuid;
}
```

调用方法：

```
console.log(generateUUID()) // 例如：7ceb31a7-41b9-45ed-
915b-14c7ad0fddf6
```

三、JS验证手机格式

> **很多时候，我们需要在JS当中去验证手机格式是否正确**

代码如下：

```
/**
 * @param str 对应手机号码
 * @returns {boolean} 结果返回 true 和 false。
 * true 为正确手机号码
 * false 为错误手机号码
 */
function verifyPhoneNumber(str){
    let phoneReg = /^(((13[0-9]{1})|(15[0-9]{1})|(17[0-9]{1})|(18[0-9]{1}))+\d{8})$/;
    return phoneReg.test(str);
}
```

调用方法：

```
console.log(verifyPhoneNumber("18818592555"))
```

四、验证身份证格式

> **很多时候，我们需要在JS当中去验证用户输入身份证格式是否正确**

代码如下：

```
/**
 * 根据〖中华人民共和国国家标准 GB 11643-1999〗中有关公民身份号码的规定，公民身份号码是特征组合码，由十七位数字本体码和一位数字校验码组成。排列顺序从左至右依次为：六位数字地址码，八位数字出生日期码，三位数字顺序码和一位数字校验码。
 * 地址码表示编码对象常住户口所在县(市、旗、区)的行政区划代码。
 * 出生日期码表示编码对象出生的年、月、日，其中年份用四位数字表示，年、月、日之间不用分隔符。
```

```
 * 顺序码表示同一地址码所标识的区域范围内，对同年、月、日出生的人员编定的顺序号。顺序码的奇数分给男性，偶数分给女性。
 * 校验码是根据前面十七位数字码，按照ISO 7064:1983.MOD 11-2校验码计算出来的检验码。
 * 出生日期计算方法。
 * 15位的身份证编码首先把出生年扩展为4位，简单的就是增加一个19或18,这样就包含了所有1800-1999年出生的人;
 * 2000年后出生的肯定都是18位的了没有这个烦恼，至于1800年前出生的,那啥那时应该还没身份证号这个东东，⊙﹏⊙b汗...
 * 下面是正则表达式:
 * 出生日期1800-2099  (18|19|20)?\d{2}(0[1-9]|1[12])(0[1-9]|[12]\d|3[01])
 * 身份证正则表达式 /^\d{6}(18|19|20)?\d{2}(0[1-9]|1[12])(0[1-9]|[12]\d|3[01])\d{3}(\d|X)$/i
 * 15位校验规则 6位地址编码+6位出生日期+3位顺序号
 * 18位校验规则 6位地址编码+8位出生日期+3位顺序号+1位校验位
 * 校验位规则     公式:∑(ai×Wi)(mod 11)……(1)
 * 公式(1)中:
 * i----表示号码字符从由至左包括校验码在内的位置序号;
 * ai----表示第i位置上的号码字符值;
 * Wi----示第i位置上的加权因子，其数值依据公式Wi=2^(n-1）(mod 11)计算得出。
 * i 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
 * Wi 7 9 10 5 8 4 2 1 6 3 7 9 10 5 8 4 2 1
 *
==========================================================================
 * 身份证号合法性验证
 * 支持15位和18位身份证号
 * 支持地址编码、出生日期、校验位验证
 * @param code
 * @returns {*[]} 该函数返回一个数组 [true,''] 或 [false,"身份证号格式错误"]
 * @constructor
 */
function identityCodeValid (code) {
```

```
  let city = { 11: "北京", 12: "天津", 13: "河北", 14: "山
西", 15: "内蒙古", 21: "辽宁", 22: "吉林", 23: "黑龙江 ", 31:
"上海", 32: "江苏", 33: "浙江", 34: "安徽", 35: "福建", 36:
"江西", 37: "山东", 41: "河南", 42: "湖北 ", 43: "湖南", 44:
"广东", 45: "广西", 46: "海南", 50: "重庆", 51: "四川", 52:
"贵州", 53: "云南", 54: "西藏 ", 61: "陕西", 62: "甘肃", 63:
"青海", 64: "宁夏", 65: "新疆", 71: "台湾", 81: "香港", 82:
"澳门", 91: "国外 " }
  let tip = ""
  let pass = true

  if (!code || !/^\d{6}(18|19|20)?\d{2}(0[1-9]|1[012])
(0[1-9]|[12]\d|3[01])\d{3}(\d|[xX])$/i.test(code)) {
    tip = "身份证号格式错误"
    pass = false
  }

  else if (!city[code.substr(0, 2)]) {
    tip = "地址编码错误"
    pass = false
  }
  else {
    //18位身份证需要验证最后一位校验位
    if (code.length == 18) {
      code = code.split('')
      //∑(ai×Wi)(mod 11)
      //加权因子
      let factor = [7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9,
10, 5, 8, 4, 2]
      //校验位
      let parity = [1, 0, 'X', 9, 8, 7, 6, 5, 4, 3, 2]
      let sum = 0
      let ai = 0
      let wi = 0
      for (let i = 0; i < 17; i++) {
        ai = code[i]
        wi = factor[i]
        sum += ai * wi
      }
      let last = parity[sum % 11]
      if (parity[sum % 11] != code[17]) {
        tip = "校验位错误,结尾是字母请注意大小写"
```

```
        pass = false
      }
    }
  }
  return [pass, tip]
}
```

调用方法：

```
console.log(identityCodeValid("110000198304102033"))
```

五、JS判断两个数组是否相等

> 可以去判断两个数组是否相等

代码如下：

```
/**
 * JS判断两个数组是否相等
 * @param {Array} arr1
 * @param {Array} arr2
 * @returns {boolean} 返回true 或 false
 */
function arrayEqual(arr1, arr2) {
    if (arr1 === arr2) return true;
    if (arr1.length != arr2.length) return false;
    for (let i = 0; i < arr1.length; ++i) {
        if (arr1[i] !== arr2[i]) return false;
    }
    return true;
}
```

调用方法：

```
const arr1 = ['a', 'b']
const arr2 = ['a', 'b']
console.log(arrayEqual(arr1, arr2))
```

六、JS格式化金额

> 可以将金额进行格式化

代码如下：

```
/**
 * JS格式化金额
 * @param money
 * @param type
 * @returns {*}
 */
function convertMoney(money , type) {
    if (/[^0-9\.]/.test(money))
        return "0";
    if (money == null || money == "")
        return "0";
    money = money.toString().replace(/^(\d*)$/, "$1.");
    money = (money + "00").replace(/(\d*\.\d\d)\d*/,
"$1");
    money = money.replace(".", ",");
    let re = /(\d)(\d{3},)/;
    while (re.test(money))
        money = money.replace(re, "$1,$2");
    money = money.replace(/,(\d\d)$/, ".$1");
    if (type == 0) {// 不带小数位(默认是有小数位)
        let a = money.split(".");
        if (a[1] == "00") {
            money = a[0];
        }
    }
    return money;
}
```

调用方法：

```
console.log(convertMoney(311546161685)) //
311,546,161,685.00
console.log(convertMoney(311546161685,0)) //
311,546,161,685
```

七、JS数组去重

> 可以将数组重复的部分去除掉

代码如下：

```
/**
 * JS数组去重
 * @param arr 数组
 * @returns {Array} 结果返回数组
 */
function removeRepeat (array) {
  let arr = []
  let json = {}
  for (let i = 0, len = array.length; i < len; i++) {
    if (!json[array[i]]) {
      arr.push(array[i])
      json[array[i]] = 1
    }
  }
  return arr
}
```

调用方法：

```
const arr = [11, 22, 33, 46, 79, 11, 46, 97, 79, 46]
console.log(removeRepeat(arr))
```

八、JS打乱数组

> 可以将数组中重复的部分去除掉

代码如下：

```
/**
 * JS打乱数组
```

```
 * @param {array} arrOld 数组
 * @param num
 * @returns {Array} 返回数组
 */
function upsetOrder(arrOld,num){
    let result=[],_length=num||arrOld.length,arr;
    arr=Object.assign([],arrOld);
    for(let i=0,len=arr.length;i<len;i++){

 result.push(arr.splice(Math.floor(Math.random()*arr.lengt
h),1)[0]);
    }
    return result;
}
```

调用方法：

```
const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
console.log(upsetOrder(arr))
```

九、JS数组冒泡排序

> 可以将数组进行排序

代码如下：

```
/**
 * 冒泡排序
 * @param array 数组
 * @returns {*}
 */
function bubbleSort(array){
    let temp;
    for(let i=0;i<array.length;i++){ //比较的趟数，从第一趟开
始
        for(let j=0;j<array.length-i-1;j++){ //每一趟比较多少
次数
            if(array[j]>array[j+1]){
                temp=array[j];
                array[j]=array[j+1];
```

```
                array[j+1]=temp;
            }
        }
    }
    return array;
}
```

调用方法：

```
const arr = [85, 24, 63, 17, 31, 17, 86, 50]
console.log(bubbleSort(arr))
```

十、JS字符串长度截取

> 可以将字符串的长度进行截取

代码如下：

```
/**
 * 字符串长度截取
 * @param str 给定的字符串
 * @param len 给定的长度
 * @returns {string} 返回截取之后的字符串
 */
function cutStr (str, len) {
  let temp,
    count = 0,
    pattern = /[^\x00-\xff]/,
    result = ''
  for (let i = 0; i < str.length; i++) {
    if (count < len - 1) {
      temp = str.substr(i, 1)
      if (pattern.exec(temp) == null) {
        count = count + 1
      } else {
        count = count + 2
      }
      result += temp
    } else {
      break
```

```
    }
  }
  return result + '...'
}
```

调用方法：

```
let str = 'fhasjdhklsajhfnolsai'
console.log(cutStr(str,8)); //fhasjdh...
```