

收藏吧 | 前端常用60余种工具方法

Vue中文社区 今天

关注 [Vue中文社区](#)，回复“加群”

加入我们一起学习，天天进步



作者: 凹凸曼-阿翔

<https://segmentfault.com/a/1190000022736837>

1.邮箱

```
export const isEmail = (s) => {  
  return /^[a-zA-Z0-9_-]+@([a-zA-Z0-9_-]+)(([a-zA-Z0-9_-]{2,3}){1,2})$/i.test(s)  
}
```

2.手机号码

```
export const isMobile = (s) => {  
  return /^1[0-9]{10}$/i.test(s)  
}
```

3.电话号码

```
export const isPhone = (s) => {  
  return /^([0-9]{3,4}-)?[0-9]{7,8}$/i.test(s)  
}
```

4.是否url地址

```
export const isURL = (s) => {  
  return /^http[s]?:\:\/\/.*$/i.test(s)  
}
```

```
}
```

5.是否字符串

```
export const isString = (o) => {  
  return Object.prototype.toString.call(o).slice(8, -1) === 'String'  
}
```

6.是否数字

```
export const isNumber = (o) => {  
  return Object.prototype.toString.call(o).slice(8, -1) === 'Number'  
}
```

7.是否boolean

```
export const isBoolean = (o) => {  
  return Object.prototype.toString.call(o).slice(8, -1) === 'Boolean'  
}
```

8.是否函数

```
export const isFunction = (o) => {  
  return Object.prototype.toString.call(o).slice(8, -1) === 'Function'  
}
```

9.是否为null

```
export const isNull = (o) => {  
  return Object.prototype.toString.call(o).slice(8, -1) === 'Null'  
}
```

10.是否undefined

```
export const isUndefined = (o) => {  
  return Object.prototype.toString.call(o).slice(8, -1) === 'Undefined'  
}
```

11.是否对象

```
export const isObj = (o) => {
```

```
    return Object.prototype.toString.call(o).slice(8, -1) === 'Object'
  }
```

12.是否数组

```
export const isArray = (o) => {
  return Object.prototype.toString.call(o).slice(8, -1) === 'Array'
}
```

13.是否时间

```
export const isDate = (o) => {
  return Object.prototype.toString.call(o).slice(8, -1) === 'Date'
}
```

14.是否正则

```
export const isRegExp = (o) => {
  return Object.prototype.toString.call(o).slice(8, -1) === 'RegExp'
}
```

15.是否错误对象

```
export const isError = (o) => {
  return Object.prototype.toString.call(o).slice(8, -1) === 'Error'
}
```

16.是否Symbol函数

```
export const isSymbol = (o) => {
  return Object.prototype.toString.call(o).slice(8, -1) === 'Symbol'
}
```

17.是否Promise对象

```
export const isPromise = (o) => {
  return Object.prototype.toString.call(o).slice(8, -1) === 'Promise'
}
```

18.是否Set对象

```
export const isSet = (o) => {  
  return Object.prototype.toString.call(o).slice(8, -1) === 'Set'  
}  
  
export const ua = navigator.userAgent.toLowerCase();
```

19.是否是微信浏览器

```
export const isWeiXin = () => {  
  return ua.match(/microMessenger/i) == 'micromessenger'  
}
```

20.是否是移动端

```
export const isDeviceMobile = () => {  
  return /android|webos|iphone|ipod|balckberry/i.test(ua)  
}
```

21.是否是QQ浏览器

```
export const isQQBrowser = () => {  
  return !!ua.match(/mqqbrowser|qzone|qqbrowser|qbwebviewtype/i)  
}
```

22.是否是爬虫

```
export const isSpider = () => {  
  return /adsbot|googlebot|bingbot|msnbot|yandexbot|baidubot|robot|careerbot|seznambot|bot|baidi  
}
```

23.是否ios

```
export const isIos = () => {  
  var u = navigator.userAgent;  
  if (u.indexOf('Android') > -1 || u.indexOf('Linux') > -1) { //安卓手机  
    return false  
  } else if (u.indexOf('iPhone') > -1) { //苹果手机  
    return true  
  } else if (u.indexOf('iPad') > -1) { //iPad  
    return false  
  } else if (u.indexOf('Windows Phone') > -1) { //winphone手机  
    return false  
  } else {
```

```

        return false
    }
}

```

24.是否为PC端

```

export const isPC = () => {
    var userAgentInfo = navigator.userAgent;
    var Agents = ["Android", "iPhone",
        "SymbianOS", "Windows Phone",
        "iPad", "iPod"];
    var flag = true;
    for (var v = 0; v < Agents.length; v++) {
        if (userAgentInfo.indexOf(Agents[v]) > 0) {
            flag = false;
            break;
        }
    }
    return flag;
}

```

25.去除html标签

```

export const removeHtmltag = (str) => {
    return str.replace(/<[^>+>/g, '')
}

```

26.获取url参数

```

export const getQueryString = (name) => {
    const reg = new RegExp('(^[&])' + name + '=(^[&]*)(&|$)', 'i');
    const search = window.location.search.split('?')[1] || '';
    const r = search.match(reg) || [];
    return r[2];
}

```

27.动态引入js

```

export const injectScript = (src) => {
    const s = document.createElement('script');
    s.type = 'text/javascript';
    s.async = true;
    s.src = src;
    const t = document.getElementsByTagName('script')[0];
    t.parentNode.insertBefore(s, t);
}

```

28.根据url地址下载

```
export const download = (url) => {
  var isChrome = navigator.userAgent.toLowerCase().indexOf('chrome') > -1;
  var isSafari = navigator.userAgent.toLowerCase().indexOf('safari') > -1;
  if (isChrome || isSafari) {
    var link = document.createElement('a');
    link.href = url;
    if (link.download !== undefined) {
      var fileName = url.substring(url.lastIndexOf('/') + 1, url.length);
      link.download = fileName;
    }
    if (document.createEvent) {
      var e = document.createEvent('MouseEvents');
      e.initEvent('click', true, true);
      link.dispatchEvent(e);
      return true;
    }
  }
  if (url.indexOf('?') === -1) {
    url += '?download';
  }
  window.open(url, '_self');
  return true;
}
```

29.el是否包含某个class

```
export const hasClass = (el, className) => {
  let reg = new RegExp('(' + className + '(\s|$)')
  return reg.test(el.className)
}
```

30.el添加某个class

```
export const addClass = (el, className) => {
  if (hasClass(el, className)) {
    return
  }
  let newClass = el.className.split(' ')
  newClass.push(className)
  el.className = newClass.join(' ')
}
```

31.el去除某个class

```
export const removeClass = (el, className) => {
```

```

    if (!hasClass(el, className)) {
      return
    }
    let reg = new RegExp('^|\\s' + className + '\\s|$)', 'g')
    el.className = el.className.replace(reg, ' ')
  }

```

32.获取滚动的坐标

```

export const getScrollPosition = (el = window) => ({
  x: el.pageXOffset !== undefined ? el.pageXOffset : el.scrollLeft,
  y: el.pageYOffset !== undefined ? el.pageYOffset : el.scrollTop
});

```

33.滚动到顶部

```

export const scrollToTop = () => {
  const c = document.documentElement.scrollTop || document.body.scrollTop;
  if (c > 0) {
    window.requestAnimationFrame(scrollToTop);
    window.scrollTo(0, c - c / 8);
  }
}

```

34.el是否在视口范围内

```

export const elementIsVisibleInViewport = (el, partiallyVisible = false) => {
  const { top, left, bottom, right } = el.getBoundingClientRect();
  const { innerHeight, innerWidth } = window;
  return partiallyVisible
    ? ((top > 0 && top < innerHeight) || (bottom > 0 && bottom < innerHeight)) &&
      ((left > 0 && left < innerWidth) || (right > 0 && right < innerWidth))
    : top >= 0 && left >= 0 && bottom <= innerHeight && right <= innerWidth;
}

```

35.洗牌算法随机

```

export const shuffle = (arr) => {
  var result = [],
      random;
  while (arr.length > 0) {
    random = Math.floor(Math.random() * arr.length);
    result.push(arr[random]);
    arr.splice(random, 1)
  }
  return result;
}

```

36.劫持粘贴板

```
export const copyTextToClipboard = (value) => {
  var textArea = document.createElement("textarea");
  textArea.style.background = 'transparent';
  textArea.value = value;
  document.body.appendChild(textArea);
  textArea.select();

  try {
    var successful = document.execCommand('copy');
  } catch (err) {
    console.log('Oops, unable to copy');
  }

  document.body.removeChild(textArea);
}
```

37.判断类型集合

```
export const checkStr = (str, type) => {
  switch (type) {
    case 'phone': //手机号码
      return /^1[3|4|5|6|7|8|9][0-9]{9}$/.test(str);
    case 'tel': //座机
      return /^(0\d{2,3}-\d{7,8})(-\d{1,4})?$/ .test(str);
    case 'card': //身份证
      return /(^\d{15}$)|(^\d{18}$)|(^\d{17}(\d|X|x)$)/.test(str);
    case 'pwd': //密码以字母开头，长度在6~18之间，只能包含字母、数字和下划线
      return /^[a-zA-Z]\w{5,17}$/.test(str)
    case 'postal': //邮政编码
      return /[1-9]\d{5}(?! \d)/.test(str);
    case 'QQ': //QQ号
      return /^[1-9][0-9]{4,9}$/.test(str);
    case 'email': //邮箱
      return /^[w-]+(\.[w-]+)*@[w-]+(\.[w-]+)+$/ .test(str);
    case 'money': //金额(小数点2位)
      return /^\d*(?:\.\d{0,2})?$/.test(str);
    case 'URL': //网址
      return /(http|ftp|https):\/\/[\w\-\_]+(\.[\w\-\_]+)+([\w\-\.,@?^=%&:/~\+]*[\w\-\_@?^=%&,
    case 'IP': //IP
      return /((?:(?:25[0-5]|2[0-4]\d|01[0-9]?\d?\d)\.){3}(?:25[0-5]|2[0-4]\d|01[0-9]?\d?\d)\.)/.test(str);
    case 'date': //日期时间
      return /^( \d{4})-( \d{2})-( \d{2}) ( \d{2}): ( \d{2}): ( \d{2}) )$/ .test(str) ||
    case 'number': //数字
      return /^[0-9]$/.test(str);
    case 'english': //英文
      return /^[a-zA-Z]+$/.test(str);
    case 'chinese': //中文
      return /^[\\u4E00-\\u9FA5]+$/.test(str);
    case 'lower': //小写
      return /^[a-z]+$/.test(str);
```



```

    case 'upper':    //大写
        return /^[A-Z]+$/.test(str);
    case 'HTML':     //HTML标记
        return /<("[^"]*"|'['']*'|[^">])*>/.test(str);
    default:
        return true;
}
}

```

38.严格的身份证校验

```

export const isCardID = (sId) => {
    if (!/^(^d{15}$)|(^d{17}(\d|X|x)$)/.test(sId)) {
        console.log('你输入的身份证长度或格式错误')
        return false
    }
    //身份证城市
    var aCity = { 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", 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: "澳门", 91: "香港"}
    if (!aCity[parseInt(sId.substr(0, 2))]) {
        console.log('你的身份证地区非法')
        return false
    }

    // 出生日期验证
    var sBirthday = (sId.substr(6, 4) + "-" + Number(sId.substr(10, 2)) + "-" + Number(sId.substr(12, 2)))
    d = new Date(sBirthday)
    if (sBirthday != (d.getFullYear() + "/" + (d.getMonth() + 1) + "/" + d.getDate())) {
        console.log('身份证上的出生日期非法')
        return false
    }

    // 身份证号码校验
    var sum = 0,
        weights = [7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2],
        codes = "10X98765432"
    for (var i = 0; i < sId.length - 1; i++) {
        sum += sId[i] * weights[i];
    }
    var last = codes[sum % 11]; //计算出来的最后一位身份证号码
    if (sId[sId.length - 1] != last) {
        console.log('你输入的身份证号非法')
        return false
    }

    return true
}

```

39.随机数范围

```

export const random = (min, max) => {
  if (arguments.length === 2) {
    return Math.floor(min + Math.random() * ((max + 1) - min))
  } else {
    return null;
  }
}

```

40.将阿拉伯数字翻译成中文的大写数字

```

export const numberToChinese = (num) => {
  var AA = new Array("零", "一", "二", "三", "四", "五", "六", "七", "八", "九", "十");
  var BB = new Array("", "十", "百", "仟", "萬", "億", "点", "");
  var a = ("" + num).replace(/(^0*)/g, "").split("."),
    k = 0,
    re = "";
  for (var i = a[0].length - 1; i >= 0; i--) {
    switch (k) {
      case 0:
        re = BB[7] + re;
        break;
      case 4:
        if (!new RegExp("0{4}//d{" + (a[0].length - i - 1) + "}$" )
            .test(a[0]))
          re = BB[4] + re;
        break;
      case 8:
        re = BB[5] + re;
        BB[7] = BB[5];
        k = 0;
        break;
    }
    if (k % 4 == 2 && a[0].charAt(i + 2) != 0 && a[0].charAt(i + 1) == 0)
      re = AA[0] + re;
    if (a[0].charAt(i) != 0)
      re = AA[a[0].charAt(i)] + BB[k % 4] + re;
    k++;
  }

  if (a.length > 1) // 加上小数部分(如果有小数部分)
  {
    re += BB[6];
    for (var i = 0; i < a[1].length; i++)
      re += AA[a[1].charAt(i)];
  }
  if (re == '一十')
    re = "十";
  if (re.match(/^一/) && re.length == 3)
    re = re.replace("一", "");
  return re;
}

```

41.将数字转换为大写金额

```
export const changeToChinese = (Num) => {
  //判断如果传递进来的不是字符的话转换为字符
  if (typeof Num == "number") {
    Num = new String(Num);
  };
  Num = Num.replace(/,/g, "") //替换tomoney()中的“,”
  Num = Num.replace(/ /g, "") //替换tomoney()中的空格
  Num = Num.replace(/¥/g, "") //替换掉可能出现的¥字符
  if (isNaN(Num)) { //验证输入的字符是否为数字
    //alert("请检查小写金额是否正确");
    return "";
  };
  //字符处理完毕后开始转换，采用前后两部分分别转换
  var part = String(Num).split(".");
  var newchar = "";
  //小数点前进行转化
  for (var i = part[0].length - 1; i >= 0; i--) {
    if (part[0].length > 10) {
      return "";
      //若数量超过拾亿单位，提示
    }
    var tmpnewchar = ""
    var perchar = part[0].charAt(i);
    switch (perchar) {
      case "0":
        tmpnewchar = "零" + tmpnewchar;
        break;
      case "1":
        tmpnewchar = "壹" + tmpnewchar;
        break;
      case "2":
        tmpnewchar = "贰" + tmpnewchar;
        break;
      case "3":
        tmpnewchar = "叁" + tmpnewchar;
        break;
      case "4":
        tmpnewchar = "肆" + tmpnewchar;
        break;
      case "5":
        tmpnewchar = "伍" + tmpnewchar;
        break;
      case "6":
        tmpnewchar = "陆" + tmpnewchar;
        break;
      case "7":
        tmpnewchar = "柒" + tmpnewchar;
        break;
      case "8":
        tmpnewchar = "捌" + tmpnewchar;
        break;
```

```

        case "9":
            tmpnewchar = "玖" + tmpnewchar;
            break;
    }
    switch (part[0].length - i - 1) {
        case 0:
            tmpnewchar = tmpnewchar + "元";
            break;
        case 1:
            if (perchar != 0) tmpnewchar = tmpnewchar + "拾";
            break;
        case 2:
            if (perchar != 0) tmpnewchar = tmpnewchar + "佰";
            break;
        case 3:
            if (perchar != 0) tmpnewchar = tmpnewchar + "仟";
            break;
        case 4:
            tmpnewchar = tmpnewchar + "万";
            break;
        case 5:
            if (perchar != 0) tmpnewchar = tmpnewchar + "拾";
            break;
        case 6:
            if (perchar != 0) tmpnewchar = tmpnewchar + "佰";
            break;
        case 7:
            if (perchar != 0) tmpnewchar = tmpnewchar + "仟";
            break;
        case 8:
            tmpnewchar = tmpnewchar + "亿";
            break;
        case 9:
            tmpnewchar = tmpnewchar + "拾";
            break;
    }
    var newchar = tmpnewchar + newchar;
}
//小数点之后进行转化
if (Num.indexOf(".") != -1) {
    if (part[1].length > 2) {
        // alert("小数点之后只能保留两位,系统将自动截断");
        part[1] = part[1].substr(0, 2)
    }
    for (i = 0; i < part[1].length; i++) {
        tmpnewchar = ""
        perchar = part[1].charAt(i)
        switch (perchar) {
            case "0":
                tmpnewchar = "零" + tmpnewchar;
                break;
            case "1":
                tmpnewchar = "壹" + tmpnewchar;
                break;
            case "2":

```

```

        tmpnewchar = "貳" + tmpnewchar;
        break;
    case "3":
        tmpnewchar = "叁" + tmpnewchar;
        break;
    case "4":
        tmpnewchar = "肆" + tmpnewchar;
        break;
    case "5":
        tmpnewchar = "伍" + tmpnewchar;
        break;
    case "6":
        tmpnewchar = "陆" + tmpnewchar;
        break;
    case "7":
        tmpnewchar = "柒" + tmpnewchar;
        break;
    case "8":
        tmpnewchar = "捌" + tmpnewchar;
        break;
    case "9":
        tmpnewchar = "玖" + tmpnewchar;
        break;
    }
    if (i == 0) tmpnewchar = tmpnewchar + "角";
    if (i == 1) tmpnewchar = tmpnewchar + "分";
    newchar = newchar + tmpnewchar;
}
}
// 替换所有无用汉字
while (newchar.search("零零") != -1)
    newchar = newchar.replace("零零", "零");
newchar = newchar.replace("零亿", "亿");
newchar = newchar.replace("亿万", "亿");
newchar = newchar.replace("零万", "万");
newchar = newchar.replace("零元", "元");
newchar = newchar.replace("零角", "");
newchar = newchar.replace("零分", "");
if (newchar.charAt(newchar.length - 1) == "元") {
    newchar = newchar + "整"
}
return newchar;
}

```

42.判断一个元素是否在数组中

```

export const contains = (arr, val) => {
    return arr.indexOf(val) !== -1 ? true : false;
}

```

43.数组排序, {type} 1: 从小到大 2: 从大到小 3: 随机

```

export const sort = (arr, type = 1) => {
  return arr.sort((a, b) => {
    switch (type) {
      case 1:
        return a - b;
      case 2:
        return b - a;
      case 3:
        return Math.random() - 0.5;
      default:
        return arr;
    }
  })
}

```

44.去重

```

export const unique = (arr) => {
  if (Array.hasOwnProperty('from')) {
    return Array.from(new Set(arr));
  } else {
    var n = {}, r = [];
    for (var i = 0; i < arr.length; i++) {
      if (!n[arr[i]]) {
        n[arr[i]] = true;
        r.push(arr[i]);
      }
    }
    return r;
  }
}

```

45.求两个集合的并集

```

export const union = (a, b) => {
  var newArr = a.concat(b);
  return this.unique(newArr);
}

```

46.求两个集合的交集

```

export const intersect = (a, b) => {
  var _this = this;
  a = this.unique(a);
  return this.map(a, function (o) {
    return _this.contains(b, o) ? o : null;
  });
}

```

47.删除其中一个元素

```
export const remove = (arr, ele) => {  
  var index = arr.indexOf(ele);  
  if (index > -1) {  
    arr.splice(index, 1);  
  }  
  return arr;  
}
```

48.将类数组转换为数组

```
export const formArray = (ary) => {  
  var arr = [];  
  if (Array.isArray(ary)) {  
    arr = ary;  
  } else {  
    arr = Array.prototype.slice.call(ary);  
  };  
  return arr;  
}
```

49.最大值

```
export const max = (arr) => {  
  return Math.max.apply(null, arr);  
}
```

50.最小值

```
export const min = (arr) => {  
  return Math.min.apply(null, arr);  
}
```

51.求和

```
export const sum = (arr) => {  
  return arr.reduce((pre, cur) => {  
    return pre + cur  
  })  
}
```

52.平均值

```
export const average = (arr) => {
  return this.sum(arr) / arr.length
}
```

53.去除空格, **type:** 1-所有空格 2-前后空格 3-前空格 4-后空格

```
export const trim = (str, type) => {
  type = type || 1
  switch (type) {
    case 1:
      return str.replace(/\s+/g, "");
    case 2:
      return str.replace(/(^\s*)|(\s*$)/g, "");
    case 3:
      return str.replace(/(^s*)/g, "");
    case 4:
      return str.replace(/(\s*$)/g, "");
    default:
      return str;
  }
}
```

54.字符转换, **type:** 1:首字母大写 2: 首字母小写 3: 大小写转换 4: 全部大写 5: 全部小写

```
export const changeCase = (str, type) => {
  type = type || 4
  switch (type) {
    case 1:
      return str.replace(/\b\w+\b/g, function (word) {
        return word.substring(0, 1).toUpperCase() + word.substring(1).toLowerCase();
      });
    case 2:
      return str.replace(/\b\w+\b/g, function (word) {
        return word.substring(0, 1).toLowerCase() + word.substring(1).toUpperCase();
      });
    case 3:
      return str.split('').map(function (word) {
        if (/[a-z]/.test(word)) {
          return word.toUpperCase();
        } else {
          return word.toLowerCase()
        }
      }).join('')
    case 4:
      return str.toUpperCase();
    case 5:
      return str.toLowerCase();
    default:
  }
```



```

        return str;
    }
}

```

55.检测密码强度

```

export const checkPwd = (str) => {
    var Lv = 0;
    if (str.length < 6) {
        return Lv
    }
    if (/[\0-9]/.test(str)) {
        Lv++
    }
    if (/[a-z]/.test(str)) {
        Lv++
    }
    if (/[A-Z]/.test(str)) {
        Lv++
    }
    if (/[\.|-|_|_]/.test(str)) {
        Lv++
    }
    return Lv;
}

```

56.函数节流器

```

export const debouncer = (fn, time, interval = 200) => {
    if (time - (window.debounceTimestamp || 0) > interval) {
        fn && fn();
        window.debounceTimestamp = time;
    }
}

```

57.在字符串中插入新字符串

```

export const insertStr = (source, index, newStr) => {
    var str = source.slice(0, index) + newStr + source.slice(index);
    return str;
}

```

58.判断两个对象是否键值相同

```

export const isObjectEqual = (a, b) => {
    var aProps = Object.getOwnPropertyNames(a);
    var bProps = Object.getOwnPropertyNames(b);

```

```

    if (aProps.length !== bProps.length) {
        return false;
    }

    for (var i = 0; i < aProps.length; i++) {
        var propName = aProps[i];

        if (a[propName] !== b[propName]) {
            return false;
        }
    }

    return true;
}

```

59.16进制颜色转RGBRGBA字符串

```

export const colorToRGB = (val, opa) => {

    var pattern = /^(#?)[a-fA-F0-9]{6}$/; //16进制颜色值校验规则
    var isOpa = typeof opa == 'number'; //判断是否有设置不透明度

    if (!pattern.test(val)) { //如果值不符合规则返回空字符
        return '';
    }

    var v = val.replace(/#/g, ''); //如果有#号先去除#号
    var rgbArr = [];
    var rgbStr = '';

    for (var i = 0; i < 3; i++) {
        var item = v.substring(i * 2, i * 2 + 2);
        var num = parseInt(item, 16);
        rgbArr.push(num);
    }

    rgbStr = rgbArr.join();
    rgbStr = 'rgb' + (isOpa ? 'a' : '') + '(' + rgbStr + (isOpa ? ',' + opa : '') + ')';
    return rgbStr;
}

```

60.追加url参数

```

export const appendQuery = (url, key, value) => {
    var options = key;
    if (typeof options == 'string') {
        options = {};
        options[key] = value;
    }
    options = $.param(options);
    if (url.includes('?')) {

```

```
    url += '&' + options
  } else {
    url += '?' + options
  }
  return url;
}
```

源码地址: [github](#), 欢迎指正或star👍👍

近期

@Winter: 基于Vue3.0聊聊架构

程序员编程时戴耳机是在听什么?



在看支持一下❤️

阅读 1037

在看 17



写下你的留言

精选留言



纣王

这个手机正则请注明是中国大陆的亲

纣王

而且手机号码第二位一般不支持012, 那是国家使用的?

1

作者
看样子是踩过不少坑
纣王
给你个机会修改之下文本

作者
帅编更多的意图是想让大家借鉴，真正生产使用还是得结合自身业务测试一下

1

纣王
真羡慕你们，这么年轻就遇到有才华的我

作者
奔涌吧，后浪



久橙
我那个朋友说他可以了

作者
不如先忘记你那个朋友，咱两做个朋友先

1



Jin ChenXin
谢谢作者

作者

1



潘什么na
真的实用



Tony
不错！

作者
老师好