

56个JavaScript 实用工具函数助你提升开发效率！

前端阳光 今天



文章来自掘金：<https://juejin.cn/post/7019090370814279693#heading-56>

作者：CUGGZ

又到周五啦，快来一起摸鱼呀！今天来看看JavaScript中的一些实用的工具函数，希望能帮助你提高开发效率！整理不易，如果觉得有用就点个赞吧！



1. 数字操作

(1) 生成指定范围随机数

```
export const randomNum = (min, max) => Math.floor(Math.random() * (max - min + 1)) + min;
```

复制代码

(2) 数字千分位分隔

```
export const format = (n) => {
  let num = n.toString();
  let len = num.length;
  if (len <= 3) {
    return num;
  } else {
    let temp = '';
    let remainder = len % 3;
    if (remainder > 0) { // 不是3的整数倍
      return num.slice(0, remainder) + ',' + num.slice(remainder, len).match(/\d{3}/g).join(',') + temp;
    } else { // 3的整数倍
      return num.slice(0, len).match(/\d{3}/g).join(',') + temp;
    }
  }
}
```

复制代码

2. 数组操作

(1) 数组乱序

```
export const arrScrambling = (arr) => {
  for (let i = 0; i < arr.length; i++) {
    const randomIndex = Math.round(Math.random() * (arr.length - 1 - i)) + i;
    [arr[i], arr[randomIndex]] = [arr[randomIndex], arr[i]];
  }
  return arr;
}
```

复制代码

(2) 数组扁平化

```
export const flatten = (arr) => {
  let result = [];

  for(let i = 0; i < arr.length; i++) {
    if(Array.isArray(arr[i])) {
      result = result.concat(flatten(arr[i]));
    } else {
      result.push(arr[i]);
    }
  }
  return result;
}
```

复制代码

(3) 数组中获取随机数

```
export const sample = arr => arr[Math.floor(Math.random() * arr.length)];
```

复制代码

3. 字符串操作

(1) 生成随机字符串

```
export const randomString = (len) => {
  let chars = 'ABCDEFGHJKMNPQRSTWXYZabcdefhijkmnprstxyz123456789';
  let strLen = chars.length;
  let randomStr = '';
  for (let i = 0; i < len; i++) {
    randomStr += chars.charAt(Math.floor(Math.random() * strLen));
  }
  return randomStr;
};
```

复制代码

(2) 字符串首字母大写

```
export const fistLetterUpper = (str) => {
  return str.charAt(0).toUpperCase() + str.slice(1);
};
```

复制代码

(3) 手机号中间四位变成*

```
export const telFormat = (tel) => {
  tel = String(tel);
  return tel.substr(0,3) + "****" + tel.substr(7);
};
```

复制代码

(4) 驼峰命名转换成短横线命名

```
export const getKebabCase = (str) => {
  return str.replace(/[A-Z]/g, (item) => '-' + item.toLowerCase())
}
```

复制代码

(5) 短横线命名转换成驼峰命名

```
export const getCamelCase = (str) => {
  return str.replace( /-([a-z])/g, (i, item) => item.toUpperCase())
}
```

复制代码

(6) 全角转换为半角

```
export const toCDB = (str) => {
  let result = '';
  for (let 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;
}
```

复制代码

(7) 半角转换为全角

```
export const toDBC = (str) => {
  let result = '';
  for (let 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;
}
```

复制代码

4. 格式转化

(1) 数字转化为大写金额

```
export const digitUppercase = (n) => {
  const fraction = ['角', '分'];
  const digit = [
    '零', '壹', '贰', '叁', '肆',
    '伍', '陆', '柒', '捌', '玖'
  ];
  const unit = [
    ['元', '万', '亿'],
    ['', '拾', '佰', '仟']
  ];
  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 s.replace(/(零.)*零元/, '元')
    .replace(/(零.)/g, '零')
    .replace(/^整$/, '零元整');
};
```

复制代码

(2) 数字转化为中文数字

```
export const intToChinese = (value) => {
  const str = String(value);
  const len = str.length-1;
  const idxs = ['', '+', '百', '千', '万', '+', '百', '千', '亿', '+', '百', '千', '万', '+', '百', '千', '亿'];
  const num = ['零', '一', '二', '三', '四', '五', '六', '七', '八', '九'];
  return str.replace(/([1-9]|0+)/g, ( $, $1, idx, full) => {
    let pos = 0;
    if($1[0] !== '0'){
      pos = len-idx;
      if(idx == 0 && $1[0] == 1 && idxs[len-idx] == '+'){
        return idxs[len-idx];
      }
      return num[$1[0]] + idxs[len-idx];
    }
    return num[$1[0]] + idxs[len-idx];
  });
}
```

```
    } else {
      let left = len - idx;
      let right = len - idx + $1.length;
      if(Math.floor(right / 4) - Math.floor(left / 4) > 0){
        pos = left - left % 4;
      }
      if( pos ){
        return idxs[pos] + num[$1[0]];
      } else if( idx + $1.length >= len ){
        return '';
      }else {
        return num[$1[0]]
      }
    }
  });
}
```

复制代码

5. 操作存储

(1) 存储loalStorage

```
export const loalStorageSet = (key, value) => {
  if (!key) return;
  if (typeof value !== 'string') {
    value = JSON.stringify(value);
  }
  window.localStorage.setItem(key, value);
};
```

复制代码

(2) 获取localStorage

```
export const loalStorageGet = (key) => {
  if (!key) return;
  return window.localStorage.getItem(key);
};
```

复制代码

(3) 删除localStorage

```
export const loalStorageRemove = (key) => {
  if (!key) return;
  window.localStorage.removeItem(key);
};
```

复制代码

(4) 存储sessionStorage

```
export const sessionStorageSet = (key, value) => {
  if (!key) return;
  if (typeof value !== 'string') {
    value = JSON.stringify(value);
  }
  window.sessionStorage.setItem(key, value);
};
```

复制代码

(5) 获取sessionStorage

```
export const sessionStorageGet = (key) => {
  if (!key) return;
  return window.sessionStorage.getItem(key);
};
```

```
};
```

复制代码

(6) 删除sessionStorage

```
export const sessionStorageRemove = (key) => {
  if (!key) return;
  window.sessionStorage.removeItem(key)
};
```

复制代码

6. 操作cookie

(1) 设置cookie

```
export const setCookie = (key, value, expire) => {
  const d = new Date();
  d.setDate(d.getDate() + expire);
  document.cookie = `${key}=${value};expires=${d.toUTCString()}`
};
```

复制代码

(2) 读取cookie

```
export const getCookie = (key) => {
  const cookieStr = unescape(document.cookie);
  const arr = cookieStr.split('; ');
  let cookieValue = '';
  for (let i = 0; i < arr.length; i++) {
    const temp = arr[i].split('=');
    if (temp[0] === key) {
      cookieValue = temp[1];
      break
    }
  }
  return cookieValue
};
```

复制代码

(3) 删除cookie

```
export const delCookie = (key) => {
  document.cookie = `${encodeURIComponent(key)}=;expires=${new Date()}`
};
```

复制代码

7. 格式校验

(1) 校验身份证号码

```
export const checkCardNo = (value) => {
  let reg = /^(^\\d{15}$)|(\\d{18}$)|(\\d{17}(\\d|X|x)$)/;
  return reg.test(value);
};
```

复制代码

(2) 校验是否包含中文

```
export const haveCNChars => (value) => {
  return /[\\u4e00-\\u9fa5]/.test(value);
};
```

```
}
复制代码
```

(3) 校验是否为中国大陆的邮政编码

```
export const isPostCode = (value) => {
  return /^[1-9][0-9]{5}$/.test(value.toString());
}
复制代码
```

(4) 校验是否为IPv6地址

```
export const isIPv6 = (str) => {
  return Boolean(str.match(/:/g)?str.match(/:/g).length<=7:false && /::/.test(str)?/^([\da-f]{1,4}(:|::)){1,6}[\da-f]{1,4}$/i.test(str):false);
}
复制代码
```

(5) 校验是否为邮箱地址

```
export const isEmail = (value) {
  return /^[a-zA-Z0-9_-]+@[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+)+$/.test(value);
}
复制代码
```

(6) 校验是否为中国大陆手机号

```
export const isTel = (value) => {
  return /^[1,3,4,5,6,7,8,9][0-9]{9}$/.test(value.toString());
}
复制代码
```

(7) 校验是否包含emoji表情

```
export const isEmojiCharacter = (value) => {
  value = String(value);
  for (let i = 0; i < value.length; i++) {
    const hs = value.charCodeAt(i);
    if (0xd800 <= hs && hs <= 0xdbff) {
      if (value.length > 1) {
        const ls = value.charCodeAt(i + 1);
        const uc = ((hs - 0xd800) * 0x400) + (ls - 0xdc00) + 0x10000;
        if (0x1d000 <= uc && uc <= 0x1f77f) {
          return true;
        }
      }
    }
    else if (value.length > 1) {
      const ls = value.charCodeAt(i + 1);
      if (ls == 0x20e3) {
        return true;
      }
    }
    else {
      if (0x2100 <= hs && hs <= 0x27ff) {
        return true;
      }
      else if (0x2B05 <= hs && hs <= 0x2b07) {
        return true;
      }
      else if (0x2934 <= hs && hs <= 0x2935) {
        return true;
      }
      else if (0x3297 <= hs && hs <= 0x3299) {
        return true;
      }
      else if (hs == 0xa9 || hs == 0xae || hs == 0x303d || hs == 0x3030
        || hs == 0x2b55 || hs == 0x2b1c || hs == 0x2b1b
        || hs == 0x2b50) {
        return true;
      }
    }
  }
}
```

```
    }  
  }  
  
  return false;  
}  
  
复制代码
```

8. 操作URL

(1) 获取URL参数列表

```
export const GetRequest = () => {  
  let url = location.search;  
  const paramsStr = /.+\?(.+)$/ .exec(url)[1]; // 将 ? 后面的字符串取出来  
  const paramsArr = paramsStr.split('&'); // 将字符串以 & 分割后存到数组中  
  let paramsObj = {};  
  // 将 params 存到对象中  
  paramsArr.forEach(param => {  
    if (/=/ .test(param)) { // 处理有 value 的参数  
      let [key, val] = param.split('='); // 分割 key 和 value  
      val = decodeURIComponent(val); // 解码  
      val = /^[^d+$/ .test(val) ? parseFloat(val) : val; // 判断是否转为数字  
      if (paramsObj.hasOwnProperty(key)) { // 如果对象有 key, 则添加一个值  
        paramsObj[key] = [].concat(paramsObj[key], val);  
      } else { // 如果对象没有这个 key, 创建 key 并设置值  
        paramsObj[key] = val;  
      }  
    } else { // 处理没有 value 的参数  
      paramsObj[param] = true;  
    }  
  })  
  return paramsObj;  
};  
  
复制代码
```

(2) 检测URL是否有效

```
export const getUrlState = (URL) => {  
  let xmlhttp = new XMLHttpRequest("microsoft.xmlhttp");  
  xmlhttp.Open("GET", URL, false);  
  try {  
    xmlhttp.Send();  
  } catch (e) {  
  } finally {  
    let result = xmlhttp.responseText;  
    if (result) {  
      if (xmlhttp.Status == 200) {  
        return true;  
      } else {  
        return false;  
      }  
    } else {  
      return false;  
    }  
  }  
}  
  
复制代码
```

(3) 键值对拼接成URL参数

```
export const params2Url = (obj) => {  
  let params = []  
  for (let key in obj) {  
    params.push(`${key}=${obj[key]}`);  
  }  
  return encodeURIComponent(params.join('&'))  
}
```



```
}  
复制代码
```

(4) 修改URL中的参数

```
export const replaceParamVal => (paramName, replaceWith) {  
  const oUrl = location.href.toString();  
  const re = eval(`/(' + paramName+'=)([^&]*)/gi`);  
  location.href = oUrl.replace(re,paramName+'='+replaceWith);  
  return location.href;  
}  
复制代码
```

(5) 删除URL中指定参数

```
export const funcUrlDel = (name) => {  
  const baseUrl = location.origin + location.pathname + "?";  
  const query = location.search.substr(1);  
  if (query.indexOf(name) > -1) {  
    const obj = {};  
    const arr = query.split("&");  
    for (let i = 0; i < arr.length; i++) {  
      arr[i] = arr[i].split("=");  
      obj[arr[i][0]] = arr[i][1];  
    }  
    delete obj[name];  
    return baseUrl + JSON.stringify(obj).replace(/[\\"\{\}\]/g,"").replace(/\\:/g,"=").replace(/\\,/g,"&");  
  }  
}  
复制代码
```

9. 设备判断

(1) 判断是移动还是PC设备

```
export const isMobile = () => {  
  if ((navigator.userAgent.match(/(iPhone|iPod|Android|ios|iOS|iPad|Backerry|WebOS|Symbian|Windows Phone|Phone)/i))) {  
    return 'mobile';  
  }  
  return 'desktop';  
}  
复制代码
```

(2) 判断是否是苹果还是安卓移动设备

```
export const isAppleMobileDevice = () => {  
  let reg = /iphone|ipod|ipad|Macintosh/i;  
  return reg.test(navigator.userAgent.toLowerCase());  
}  
复制代码
```

(3) 判断是否是安卓移动设备

```
export const isAndroidMobileDevice = () => {  
  return /android/i.test(navigator.userAgent.toLowerCase());  
}  
复制代码
```

(4) 判断是Windows还是Mac系统

```
export const osType = () => {  
  const agent = navigator.userAgent.toLowerCase();  
  if (agent.indexOf('win') > -1) {  
    return 'Windows';  
  }  
  if (agent.indexOf('mac') > -1) {  
    return 'Mac';  
  }  
  return 'Other';  
}
```

```
const isMac = /macintosh|mac os x/i.test(navigator.userAgent);
const isWindows = agent.indexOf("win64") >= 0 || agent.indexOf("wow64") >= 0 || agent.indexOf("win32") >= 0 || agent.index
if (isWindows) {
  return "windows";
}
if(isMac){
  return "mac";
}
}
```

复制代码

(5) 判断是否是微信/QQ内置浏览器

```
export const browser = () => {
  const ua = navigator.userAgent.toLowerCase();
  if (ua.match(/MicroMessenger/i) == "micromessenger") {
    return "weixin";
  } else if (ua.match(/QQ/i) == "qq") {
    return "QQ";
  }
  return false;
}
```

复制代码

(6) 浏览器型号和版本

```
export const getExplorerInfo = () => {
  let t = navigator.userAgent.toLowerCase();
  return 0 <= t.indexOf("msie") ? { //ie < 11
    type: "IE",
    version: Number(t.match(/msie ([\d]+)/)[1])
  } : !!t.match(/trident\./.+?rv:([\d.]+)/) ? { // ie 11
    type: "IE",
    version: 11
  } : 0 <= t.indexOf("edge") ? {
    type: "Edge",
    version: Number(t.match(/edge\/([\d]+)/)[1])
  } : 0 <= t.indexOf("firefox") ? {
    type: "Firefox",
    version: Number(t.match(/firefox\/([\d]+)/)[1])
  } : 0 <= t.indexOf("chrome") ? {
    type: "Chrome",
    version: Number(t.match(/chrome\/([\d]+)/)[1])
  } : 0 <= t.indexOf("opera") ? {
    type: "Opera",
    version: Number(t.match(/opera.([\d]+)/)[1])
  } : 0 <= t.indexOf("Safari") ? {
    type: "Safari",
    version: Number(t.match(/version\/([\d]+)/)[1])
  } : {
    type: t,
    version: -1
  }
}
```

复制代码

10. 浏览器操作

(1) 滚动到页面顶部

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

```
}  
复制代码
```

(2) 滚动到页面底部

```
export const scrollToBottom = () => {  
  window.scrollTo(0, document.documentElement.clientHeight);  
}  
复制代码
```

(3) 滚动到指定元素区域

```
export const smoothScroll = (element) => {  
  document.querySelector(element).scrollIntoView({  
    behavior: 'smooth'  
  });  
};  
复制代码
```

(4) 获取可视窗口高度

```
export const getClientHeight = () => {  
  let clientHeight = 0;  
  if (document.body.clientHeight && document.documentElement.clientHeight) {  
    clientHeight = (document.body.clientHeight < document.documentElement.clientHeight) ? document.body.clientHeight : document.documentElement.clientHeight;  
  }  
  else {  
    clientHeight = (document.body.clientHeight > document.documentElement.clientHeight) ? document.body.clientHeight : document.documentElement.clientHeight;  
  }  
  return clientHeight;  
}  
复制代码
```

(5) 获取可视窗口宽度

```
export const getPageViewWidth = () => {  
  return (document.compatMode == "BackCompat" ? document.body : document.documentElement).clientWidth;  
}  
复制代码
```

(6) 打开浏览器全屏

```
export const toFullScreen = () => {  
  let element = document.body;  
  if (element.requestFullscreen) {  
    element.requestFullscreen()  
  } else if (element.mozRequestFullScreen) {  
    element.mozRequestFullScreen()  
  } else if (element.msRequestFullscreen) {  
    element.msRequestFullscreen()  
  } else if (element.webkitRequestFullscreen) {  
    element.webkitRequestFullscreen()  
  }  
}  
复制代码
```

(7) 退出浏览器全屏

```
export const exitFullscreen = () => {  
  if (document.exitFullscreen) {  
    document.exitFullscreen()  
  } else if (document.msExitFullscreen) {  
    document.msExitFullscreen()  
  }  
}
```

```
document.msExitFullscreen()  
} else if (document.mozCancelFullScreen) {  
document.mozCancelFullScreen()  
} else if (document.webkitExitFullscreen) {  
document.webkitExitFullscreen()  
}  
}  
复制代码
```

11. 时间操作

(1) 当前时间

```
export const nowTime = () => {  
const now = new Date();  
const year = now.getFullYear();  
const month = now.getMonth();  
const date = now.getDate() >= 10 ? now.getDate() : ('0' + now.getDate());  
const hour = now.getHours() >= 10 ? now.getHours() : ('0' + now.getHours());  
const miu = now.getMinutes() >= 10 ? now.getMinutes() : ('0' + now.getMinutes());  
const sec = now.getSeconds() >= 10 ? now.getSeconds() : ('0' + now.getSeconds());  
return +year + "年" + (month + 1) + "月" + date + "日 " + hour + ":" + miu + ":" + sec;  
}  
复制代码
```

(2) 格式化时间

```
export const dateFormatter = (formater, time) => {  
let date = time ? new Date(time) : new Date(),  
Y = date.getFullYear() + '',  
M = date.getMonth() + 1,  
D = date.getDate(),  
H = date.getHours(),  
m = date.getMinutes(),  
s = date.getSeconds();  
return formater.replace(/YYYY|yyyy/g, Y)  
.replace(/YY|yy/g, Y.substr(2, 2))  
.replace(/MM/g, (M<10 ? '0' : '') + M)  
.replace(/DD/g, (D<10 ? '0' : '') + D)  
.replace(/HH|hh/g, (H<10 ? '0' : '') + H)  
.replace(/mm/g, (m<10 ? '0' : '') + m)  
.replace(/ss/g, (s<10 ? '0' : '') + s)  
}  
// dateFormatter('YYYY-MM-DD HH:mm:ss')  
// dateFormatter('YYYYMMDDHHmmss')  
复制代码
```

12. JavaScript操作

(1) 阻止冒泡事件

```
export const stopPropagation = (e) => {  
e = e || window.event;  
if(e.stopPropagation) { // W3C阻止冒泡方法  
e.stopPropagation();  
} else {  
e.cancelBubble = true; // IE阻止冒泡方法  
}  
}  
复制代码
```

(2) 防抖函数

```
export const debounce = (fn, wait) => {
  let timer = null;

  return function() {
    let context = this,
        args = arguments;

    if (timer) {
      clearTimeout(timer);
      timer = null;
    }

    timer = setTimeout(() => {
      fn.apply(context, args);
    }, wait);
  };
}
```

复制代码

(3) 节流函数

```
export const throttle = (fn, delay) => {
  let curTime = Date.now();

  return function() {
    let context = this,
        args = arguments,
        nowTime = Date.now();

    if (nowTime - curTime >= delay) {
      curTime = Date.now();
      return fn.apply(context, args);
    }
  };
}
```

复制代码

(4) 数据类型判断

```
export const getType = (value) => {
  if (value === null) {
    return value + "";
  }
  // 判断数据是引用类型的情况
  if (typeof value === "object") {
    let valueClass = Object.prototype.toString.call(value),
        type = valueClass.split(" ")[1].split("");
    type.pop();
    return type.join("").toLowerCase();
  } else {
    // 判断数据是基本数据类型的情况和函数的情况
    return typeof value;
  }
}
```

复制代码

(5) 对象深拷贝

```
export const deepClone = (obj, hash = new WeakMap()) => {
  // 日期对象直接返回一个新的日期对象
  if (obj instanceof Date){
    return new Date(obj);
  }
  // 正则对象直接返回一个新的正则对象
  if (obj instanceof RegExp){
    return new RegExp(obj);
  }
}
```

```
    }  
    // 如果循环引用, 就用 weakMap 来解决  
    if (hash.has(obj)){  
        return hash.get(obj);  
    }  
    // 获取对象所有自身属性的描述  
    let allDesc = Object.getOwnPropertyDescriptors(obj);  
    // 遍历传入参数所有键的特性  
    let cloneObj = Object.create(Object.getPrototypeOf(obj), allDesc)  
  
    hash.set(obj, cloneObj)  
    for (let key of Reflect.ownKeys(obj)) {  
        if(typeof obj[key] === 'object' && obj[key] !== null){  
            cloneObj[key] = deepClone(obj[key], hash);  
        } else {  
            cloneObj[key] = obj[key];  
        }  
    }  
    return cloneObj  
}  
}
```

复制代码



喜欢此内容的人还喜欢

如何在JavaScript中处理时区「译」

React

干掉 SQL 中的 like，我用 es 后运营小姐姐们都说好快！

写代码的渣渣鹏