uniapp js实用工具包

目　录

介绍

插件为uniapp前端js工具包，工具包包含了众多模块，数组、颜色、日期、加解密、函数、数字、对象、请求、字符串、验证等模块，可以让您的开发得心应手，不需要为了一个功能需要花费大量的时间从网上再去搜索，本插件已集成了大部分常用的功能，**本插件会持续更新。**

联系作者

关注微信公众号可联系作者

如何使用

下载插件

本插件支持uni_modules，可通过插件市场uni_modules直接导入插件即可
插件地址：https://ext.dcloud.net.cn/plugin?id=12012

联系作者

QQ：334106817
微信：lw0911913

全局引入

在main.js文件中添加以下代码

```
// 加载js工具插件
import jsTools from "uni_modules/leven-js-tools/index.js"
Vue.use(jsTools);
```

然后就可以直接使用了，不需要在页面中单独引入插件

自定义插件引用

如果您觉得uni.$lv不是您想用的，你也可以自定义插件的引用，只需要在main.js里添加以下代码即可

```
// 自定义插件引用
uni.$v = uni.$lv;
```

main.js示例文件如下：

```
import App from './App'

// #ifndef VUE3
import Vue from 'vue'
Vue.config.productionTip = false
App.mpType = 'app'

// 加载js工具插件
```

如何使用

```
import jsTools from "uni_modules/leven-js-tools/index.js"
Vue.use(jsTools);

// 自定义插件引用
uni.$v = uni.$lv;

const app = new Vue({
  ...App
})
app.$mount()
// #endif

// #ifdef VUE3
import {
  createSSRApp
} from 'vue'
export function createApp() {
  const app = createSSRApp(App)
  return {
    app
  }
}
// #endif
```

使用方式

每一种插件都是以uni.$lv开头，具体的功能会在单独的章节中讲解
1.数组

```
uni.$lv.array
```

2.颜色

```
uni.$lv.color
```

3.日期

```
uni.$lv.date
```

4.加解密

如何使用

```
uni.$lv.encrypt
```

5.函数

```
uni.$lv.func
```

6.数字

```
uni.$lv.number
```

7.对象

```
uni.$lv.object
```

8.请求

```
uni.$lv.request
```

9.字符串

```
uni.$lv.string
```

10.定时器

```
uni.$lv.timer
```

11.验证器

```
uni.$lv.validate
```

更新日志

2023-09-12 V1.1.6

日期

- 【新增】加多少天后的日期【转到】
- 【新增】减多少天后的日期【转到】

2023-08-17 v1.1.5

字符串

- 字符串转十六进制，新增参数：是否UrlEncode
- 十六进制转字符串，新增参数：是否UrlDecode

数组

- 求和，新增参数hex（进制）目前只支持10进制或16进制

2023-06-06 v1.1.4

数字

1.新增高精度加法【转到】
2.新增高精度减法【转到】
3.新增高精度乘法【转到】
4.新增高精度除法【转到】

2023-05-30 v1.1.3

更新luch-request版本至3.1.0

2023-05-11 v1.1.2

字符串

1.转大写【转到】
2.转小写【转到】
3.下划线转驼峰【转到】
4.驼峰转下划线【转到】
5.url解析【转到】
6.将字符串转换为16进制【转到】
7.十六进制转换为字符串【转到】

数组

1.求和【转到】

2023-05-09 v1.1.1

定时器

1.新增设置一个定时器【转到】

验证

1.新增验证是否是微信浏览器【转到】
2.新增验证是否是支付宝浏览器【转到】

2023-05-04 v1.1.0

日期

1.两个日期之间的天数【转到】
2.某个月有多少天【转到】
3.两个日期之间的所有日期【转到】

数字

1.金额转大写【转到】

颜色

1.生成随机颜色【转到】

字符串

1.随机字符串【转到】

2023-04-28 v1.0.1

对象新增按键名排序【转到】

2023-04-26 v1.0.0

首次发布

赞助作者

您的赞助是作者持续更新的动力，如果您觉得插件还可以的话您可以赞助作者，赞助时请备注“uniapp插件”

支付宝/微信 赞助名单

赞助作者

查看

插件赞赏名单

2024-10-22 **臧*宏** 0.30元
2023-10-09 **程*** 0.30元
2023-09-21 **胡*华** 0.30元
2023-08-17 **潘*平** 5元

排序

方法名

`sort`

参数说明

参数名	参数类型	是否必填	默认值	描述
arr	Array	是	无	要排序的数组
type	String	否	asc	排序方式，asc:升序，desc：倒序

示例

```
let newArr = uni.$lv.array.sort(this.numberArr, this.sortNumberType);
this.sortNumberType = this.sortNumberType == "asc" ? "desc" : "asc";
console.log(newArr);
```

返回值

返回排序后的数组

数组分割

方法名

`split`

参数说明

参数名	参数类型	是否必填	默认值	描述
arr	Array	是	无	要分割的数组
len	Integer	是	无	分割长度

示例

```
let newArr = uni.$lv.array.split(this.numberArr, 4);
console.log(newArr);
```

返回值

返回分割后的数组

```
//分割前
[3, 12, 7, 9, 36, 24, 45, 11, 22, 43, 64, 53, 8, 19, 36, 12, 23]
//分割后
[[3,12,7,9],[36,24,45,11],[22,43,64,53],[8,19,36,12],[23]]
```

数组是否相同

方法名

> 该方法会先排序再比较

参数说明

参数名	参数类型	是否必填	默认值	描述
arr1	Array	是	无	要比较的数组
arr2	Array	是	无	要比较的数组

示例

```
```

返回值

true/false
true:相同，false：不相同

移除元素

方法名

`remove`

> 如果有多个相同的值只会移除第一个，建议您可以先去重再移除

参数说明

参数名	参数类型	是否必填	默认值	描述
arr1	Array	是	无	原始数组
value	Integer	是	无	要移除的值

示例

```
let newArr = uni.$lv.array.remove(this.numberArr, 11);
console.log(newArr);
```

返回值

移除值后的新数组

添加元素

参数名	参数类型	是否必填	默认值	描述
arr	Array	是	无	原始数组
index	Integer	是	无	添加的位置，从0开始
value	Integer	是	无	要添加的值

添加值后的新数组

最大值

方法名

`max`

参数说明

参数名	参数类型	是否必填	默认值	描述
arr	Array	是	无	原始数组

示例

```
let value = uni.$lv.array.max(this.numberArr);
console.log(value);
```

返回值

最大的值

最小值

方法名

`min`

参数说明

参数名	参数类型	是否必填	默认值	描述
arr	Array	是	无	原始数组

示例

```
let value = uni.$lv.array.min(this.numberArr);
console.log(value);
```

返回值

最小的值

数组打乱

方法名

`disruption`

参数说明

参数名	参数类型	是否必填	默认值	描述
arr	Array	是	无	原始数组

示例

```
let value = uni.$lv.array.disruption(this.numberArr);
console.log(value);
```

返回值

打乱后的数组

随机值

方法名

`random`

参数说明

参数名	参数类型	是否必填	默认值	描述
arr	Array	是	无	原始数组
length	Integer	否	1	取出的长度
isSame	Boolean	否	false	是否可以相同

示例

```
let value = uni.$lv.array.random(this.numberArr, 5);
console.log(value);
```

返回值

取出的值（数组）

去重

方法名

`unique`

参数说明

参数名	参数类型	是否必填	默认值	描述
arr	Array	是	无	原始数组

示例

```
let value = uni.$lv.array.unique(this.numberArr);
console.log(value);
```

返回值

去重后的数组

合并

方法名

`union`

参数说明

参数名	参数类型	是否必填	默认值	描述
arr1	Array	是	无	原始数组
arr2	Array	是	无	要合并的数组

示例

```
let value = uni.$lv.array.union(this.numberArr, [1, 5, 7, 3, 6, 9]);
console.log(value);
```

返回值

合并后的数组，结果会返回去重后的值

交集

参数名	参数类型	是否必填	默认值	描述
arr1	Array	是	无	原始数组
arr2	Array	是	无	取交集的数组

取交集后的数组

是否是数组

方法名

`isArray`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	any	是	无	原始值

示例

```
let value = uni.$lv.array.isArray(this.numberArr);
console.log(value);
let value1 = uni.$lv.array.isArray(1);
console.log(value1);
let value2 = uni.$lv.array.isArray("123");
console.log(value2);
```

返回值

true/false
true:是数组
false:不是数组

平均值

参数名	参数类型	是否必填	默认值	描述
arr	Array	是	无	原始数组

```
```

平均值(包含小数点)

求和

方法名

`sum`

参数说明

参数名	参数类型	是否必填	默认值	描述
arr	Array	是	无	原始数组
hex `v1.1.5`	Integer	否	10	进制，10或16

示例

```
let value = uni.$lv.array.sum([3, 12, 7, 9, 36, 24, 45, 11, 22, 43, 64, 53, 8, 19
console.log(value);
```

返回值

Integer 计算结果

返回示例

```
427
```

十六进制转换为rgb

方法名

`hexToRgb`

参数说明

参数名	参数类型	是否必填	默认值	描述
sColor	String	是	无	十六进制颜色值
str	Boolean	否	true	是否以字符串返回,true.返回格式rgb(0,0,0), false.返回格式：[0,0,0]

示例

```
let result = uni.$lv.color.hexToRgb("#234d72");
console.log(result);
let result1 = uni.$lv.color.hexToRgb("#199076", false);
console.log(result1);
```

返回值

String/Array
参数中str为true时返回String，为false时返回Array，详见参数描述

渐变值

参数名	参数类型	是否必填	默认值	描述
startColor	String	否	rgb(0, 0, 0)	开始的颜色
endColor	String	否	rgb(255, 255, 255)	结束的颜色
step	Integer	否	10	颜色等分的份额

十六进制颜色值数组

rgb转十六进制

方法名

`rgbToHex`

参数说明

参数名	参数类型	是否必填	默认值	描述
rgb	String	是	无	rgb颜色值

示例

```
let result = uni.$lv.color.rgbToHex('rgb(12, 233, 124)');
console.log(result);
let result1 = uni.$lv.color.rgbToHex('rgb(45, 10, 110)');
console.log(result1);
```

返回值

十六进制颜色值

随机颜色

方法名

`random`

参数说明

参数名	参数类型	是否必填	默认值	描述
returnType	Integer	否	1	返回格式，1.十六机制，2.rgb格式，3.数组(rgb的三个值)

示例

```
let result = uni.$lv.color.random();
console.log(result);
let result1 = uni.$lv.color.random(2);
console.log(result1);
let result2 = uni.$lv.color.random(3);
console.log(result2);
```

返回值

十六进制/rgb字符串/rgb数组

返回示例

```
#22D830
rgb(63, 93, 78)
[245,231,42]
```

说明

本插件使用Date函数处理，只提供了一些常用的功能。如果您需要更高级的功能可以使用dayjs或moment

dayjs

Day.js是一个极简的JavaScript库，可以为现代浏览器解析、验证、操作和显示日期和时间。
官网：https://dayjs.fenxianglu.cn/
使用文档：https://dayjs.fenxianglu.cn/category/

moment js

JavaScript 日期处理类库
官网：http://momentjs.cn/
使用文档：http://momentjs.cn/docs/

获取时间戳

方法名

`getTimestamp`

参数说明

参数名	参数类型	是否必填	默认值	描述
time	String	否	无	日期字符串，如：2023-04-26 09:22:56
type	String	否	s	返回格式：s:返回秒，ms:返回毫秒

示例

```
// 获取秒
let timestamp = uni.$lv.date.getTimestamp();
console.log(timestamp);
// 获取毫秒
let timestamp1 = uni.$lv.date.getTimestamp(null, "ms");
console.log(timestamp1);
```

返回值

Integer 秒/毫秒

格式化时间

方法名

`format`

参数说明

参数名	参数类型	是否必填	默认值	描述
dateTime	Integer	是	无	日期时间戳，支持秒和毫秒
formatStr	String	否	yyyy-mm-dd	返回格式：支持自定义

自定义格式说明

yyyy 年
mm 月
dd 日
hh 时
MM 分
ss 秒

示例

```
let today = uni.$lv.date.format();
console.log(today);
let date = uni.$lv.date.format(1681033676);
console.log(date);
let date1 = uni.$lv.date.format(1681033676, "yyyy-mm-dd hh:MM:ss");
console.log(date1);
let date2 = uni.$lv.date.format(1681033676, "yyyy年mm月dd日 hh:MM:ss");
console.log(date2);
```

返回值

格式化后的时间

多久之前

方法名

`from`

参数说明

参数名	参数类型	是否必填	默认值	描述
timestamp	Integer	是	无	时间戳
format	String/Boolean	否	yyyy-mm-dd	格式化规则

格式化规则说明

格式化规则如果为时间格式字符串，超出一定时间范围，返回固定的时间格式
如果为布尔值false，无论什么时间，都返回多久以前的格式

示例

```
let date = uni.$lv.date.from(1682043676);
console.log(date);
let date1 = uni.$lv.date.from(1681043676);
console.log(date1);
let date2 = uni.$lv.date.from(1581043676);
console.log(date2);
let date3 = uni.$lv.date.from(1581043676, "yyyy-mm-dd hh:MM:ss");
console.log(date3);
```

返回值

格式化后的时间

日期补零

方法名

`padZero`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	Integer	是	无	需要补零的值

示例

```
let date = uni.$lv.date.padZero(5);
console.log(date);
let date1 = uni.$lv.date.padZero(10);
console.log(date1);
```

返回值

补0后的值

昨天

方法名

`yesterday`

参数说明

无参数

示例

```
let date = uni.$lv.date.yesterday();
console.log(date);
```

返回值

昨天的日期，返回格式：yyyy-mm-dd

周开始和结束日期

方法名

`week`

参数说明

参数名	参数类型	是否必填	默认值	描述
operator	Integer	否	1	前后周

示例

```
// 本周
let date = uni.$lv.date.week();
console.log(date);
// 上周
let date1 = uni.$lv.date.week(-1);
console.log(date1);
// 上上周
let date2 = uni.$lv.date.week(-2);
console.log(date2);
// 下周
let date3 = uni.$lv.date.week(2);
console.log(date3);
// 下下周
let date4 = uni.$lv.date.week(3);
console.log(date4);
```

返回值

Array
周的开始日期和结束日期，[周一,周日]
如下返回示例

```
["2023-04-24","2023-04-30"]
```

月开始和结束日期

方法名

`month`

参数说明

参数名	参数类型	是否必填	默认值	描述
operator	Integer	否	1	前后月

示例

```
// 本月
let date = uni.$lv.date.month();
console.log(date);
// 上月
let date1 = uni.$lv.date.month(-1);
console.log(date1);
// 上上月
let date2 = uni.$lv.date.month(-2);
console.log(date2);
// 前6个月前的日期
let date3 = uni.$lv.date.month(-6);
console.log(date3);
// 下个月
let date4 = uni.$lv.date.month(2);
console.log(date4);
// 下下个月
let date5 = uni.$lv.date.month(3);
console.log(date5);
// 10个月后的日期
let date6 = uni.$lv.date.month(11);
console.log(date6);
```

返回值

Array
月的开始日期和结束日期

月开始和结束日期

如下返回示例

```
["2023-04-01","2023-04-30"]
["2023-03-01","2023-03-31"]
["2023-02-01","2023-02-28"]
["2022-10-01","2022-10-31"]
["2023-05-01","2023-05-31"]
["2023-06-01","2023-06-30"]
["2024-02-01","2024-02-29"]
```

是否是有效日期

方法名

`isDate`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	日期的值
separator	Sting	否	-	年月日之间的分隔符

示例

```
let result = uni.$lv.date.isDate("2023-04-21");
console.log(result);
let result1 = uni.$lv.date.isDate("2023-04-31");
console.log(result1);
let result2 = uni.$lv.date.isDate("2023-02-28");
console.log(result2);
```

返回值

true/false
true:有效日期
false:无效日期

日期之间的天数

方法名

`betweenDays`

参数说明

参数名	参数类型	是否必填	默认值	描述
startDate	String	是	无	开始日期，格式：YYYY-mm-dd
endDate	String	是	无	结束日期，格式：YYYY-mm-dd

示例

```
let result = uni.$lv.date.betweenDays("2023-04-21", "2023-05-15");
console.log(result);
```

返回值

Integer

返回示例

```
24
```

某个月的天数

方法名

`monthDays`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	月份，格式：YYYY-mm

示例

```
let result = uni.$lv.date.monthDays("2023-04");
console.log(result);
let result1 = uni.$lv.date.monthDays("2023-02");
console.log(result1);
```

返回值

Integer

返回示例

```
30
28
```

日期之间的所有日期

方法名

`betweenDates`

参数说明

参数名	参数类型	是否必填	默认值	描述
startDate	String	是	无	开始日期，格式：YYYY-mm-dd
endDate	String	是	无	结束日期，格式：YYYY-mm-dd

示例

```
let result = uni.$lv.date.betweenDates("2023-04-21", "2023-05-15");
console.log(result);
```

返回值

Array

返回示例

```
["2023-04-21","2023-04-22","2023-04-23","2023-04-24","2023-04-25","2023-04-26","2(
```

加多少天后的日期

方法名

`addDay`

参数说明

参数名	参数类型	是否必填	默认值	描述
day	Integer	是	无	需要加的天数

示例

```
let result = uni.$lv.date.addDay(1);
console.log(result);
```

返回值

Integer

返回示例

```
2023-09-12
```

减多少天后的日期

方法名

`reduceDay`

参数说明

参数名	参数类型	是否必填	默认值	描述
day	Integer	是	无	需要减的天数

示例

```
let result = uni.$lv.date.reduceDay(1);
console.log(result);
```

返回值

Integer

返回示例

```
2023-09-10
```

md5

方法名

`md5`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串

示例

```
let result = uni.$lv.encrypt.md5("123456");
console.log(result);
let result2 = uni.$lv.encrypt.md5("Yx9RUbhjxDLwzPaNXV%EvZQp&kQvmkPp");
console.log(result2);
```

返回值

加密后的值

AES加密

方法名

`aesEncode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串
key	String	是	无	加密秘钥

示例

```
let result = uni.$lv.encrypt.aesEncode("123456abcdefg这是一条加密字符串", "M@FWNJVtCs
console.log(result);
```

返回值

加密后的值

AES解密

方法名

`aesDecode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	解密字符串
key	String	是	无	解密秘钥

示例

```
let result = uni.$lv.encrypt.aesDecode("LLgx3btXTpnb8ThYwc/DFK44kjTT80PhJCQ2ESrKo
console.log(result);
```

返回值

解密后的值

base64加密

方法名

`base64Encode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串

示例

```
let result = uni.$lv.encrypt.base64Encode("KNGT^$Sat#je5VX9Hdp3Y@kLUTegwg82这是一条
console.log(result);
```

返回值

加密后的值

base64解密

方法名

`base64Decode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	解密字符串

示例

```
let result = uni.$lv.encrypt.base64Decode("S05HVF4kU2F0I2plNVZYOUhkcDNZQGtMVVRlZ3
console.log(result);
```

返回值

解密后的值

DES加密

方法名

`desEncode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串
key	String	是	无	加密秘钥

示例

```
let result = uni.$lv.encrypt.desEncode("abcdefg123456这是一条加密字符串7892345", "#p
console.log(result);
```

返回值

加密后的值

DES解密

方法名

`desDecode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	解密字符串
key	String	是	无	解密秘钥

示例

```
let result = uni.$lv.encrypt.desDecode("PtIXJtlLuhqspkRVag5fCOllHUMWHdCAKM/+1B0Y74
console.log(result);
```

返回值

解密后的值

RC4加密

方法名

`rc4Encode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串
key	String	是	无	加密秘钥

示例

```
let result = uni.$lv.encrypt.rc4Encode("abcdefg123456这是一条加密字符串7892345", "4A
console.log(result);
```

返回值

加密后的值

RC4解密

方法名

`rc4Decode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	解密字符串
key	String	是	无	解密秘钥

示例

```
let result = uni.$lv.encrypt.rc4Decode("U2FsdGVkX19FTosfD3o0xO6vxPGhcPcIdobGZ1iUO
console.log(result);
```

返回值

解密后的值

rabbit加密

方法名

`rabbitEncode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串
key	String	是	无	加密秘钥

示例

```
let result = uni.$lv.encrypt.rabbitEncode("abcdefg123456这是一条加密字符串7892345", '
console.log(result);
```

返回值

加密后的值

rabbit解密

方法名

`rabbitDecode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	解密字符串
key	String	是	无	解密秘钥

示例

```
let result = uni.$lv.encrypt.rabbitDecode("U2FsdGVkX1/m0oYx3iPHKo/xsiqfuxE1r5hDOY
console.log(result);
```

返回值

解密后的值

tripleDES加密

方法名

`tripleDESEncode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串
key	String	是	无	加密秘钥

示例

```
let result = uni.$lv.encrypt.tripleDESEncode("abcdefg123456这是一条加密字符串7892345'
console.log(result);
```

返回值

加密后的值

tripleDES解密

方法名

`tripleDESDecode`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	解密字符串
key	String	是	无	解密秘钥

示例

```
let result = uni.$lv.encrypt.tripleDESDecode(
          "U2FsdGVkX1//9GQy3LlaU+FYmNYVf4//U+yim+Rj7KJTXIHMC5o3sv1+uY8fqJo+/l9E9K(
console.log(result);
```

返回值

解密后的值

sha1

方法名

`sha1`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串

示例

```
let result = uni.$lv.encrypt.sha1("abcdefg123456这是一条加密字符串7892345");
console.log(result);
```

返回值

加密后的值

sha224

方法名

`sha224`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串

示例

```
let result = uni.$lv.encrypt.sha224("abcdefg123456这是一条加密字符串7892345");
console.log(result);
```

返回值

加密后的值

sha256

方法名

`sha256`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串

示例

```
let result = uni.$lv.encrypt.sha256("abcdefg123456这是一条加密字符串7892345");
console.log(result);
```

返回值

加密后的值

sha384

方法名

`sha384`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串

示例

```
let result = uni.$lv.encrypt.sha384("abcdefg123456这是一条加密字符串7892345");
console.log(result);
```

返回值

加密后的值

sha512

方法名

`sha512`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串

示例

```
let result = uni.$lv.encrypt.sha512("abcdefg123456这是一条加密字符串7892345");
console.log(result);
```

返回值

加密后的值

hmacSha1

方法名

`hmacSha1`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串
key	String	是	无	加密秘钥

示例

```
let result = uni.$lv.encrypt.hmacSha1("abcdefg123456这是一条加密字符串7892345", "u$DC
console.log(result);
```

返回值

加密后的值

hmacSha256

方法名

`hmacSha256`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串
key	String	是	无	加密秘钥

示例

```
let result = uni.$lv.encrypt.hmacSha256("abcdefg123456这是一条加密字符串7892345", "us
console.log(result);
```

返回值

加密后的值

hmacSha512

方法名

`hmacSha512`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串
key	String	是	无	加密秘钥

示例

```
let result = uni.$lv.encrypt.hmacSha512("abcdefg123456这是一条加密字符串7892345", "us
console.log(result);
```

返回值

加密后的值

hmacMd5

方法名

`hmacMd5`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	加密字符串
key	String	是	无	加密秘钥

示例

```
let result = uni.$lv.encrypt.hmacMd5("abcdefg123456这是一条加密字符串7892345", "u$DC
console.log(result);
```

返回值

加密后的值

延时

方法名

`sleep`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	Integer	否	30	延时时间，单位毫秒

示例

```
console.log("开始延时")
await uni.$lv.func.sleep(1000);
console.log("1秒后执行该内容")
```

是否是函数

方法名

`isFunc`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	any	是	无	判断参数

示例

```
let fun = function func() {};
let result = uni.$lv.func.isFunc(fun);
console.log(result);
let result1 = uni.$lv.func.isFunc("123");
console.log(result1);
```

返回值

true/false
true:是函数
false:不是函数

防抖

方法名

`debounce`

> 防抖原理：一定时间内，只有最后一次操作，再过wait毫秒后才执行函数

参数说明

参数名	参数类型	是否必填	默认值	描述
func	Function	是	无	要执行的回调函数
wait	Integer	否	500	延时的时间，单位毫秒
immediate	Boolean	否	false	是否立即执行，此类情况一般用不到

示例

```
let result = uni.$lv.func.debounce(() => {
  console.log(123)
});
```

返回值

无

节流

> 节流原理：在一定时间内，只能触发一次

参数名	参数类型	是否必填	默认值	描述
func	Function	是	无	要执行的回调函数
wait	Integer	否	500	延时的时间，单位毫秒
immediate	Boolean	否	true	是否立即执行

```
```

无

消息提示

方法名

`toast`

参数说明

参数名	参数类型	是否必填	默认值	描述
title	String	是	无	提示内容
func	Function	否	null	提示完成后执行的函数
duration	Integer	否	2000	执行时间，单位毫秒

示例

```
uni.$lv.func.toast("123");
uni.$lv.func.toast("123", () => {
  console.log("消息提示完成后执行的操作");
});
```

返回值

无

排列组合

方法名

`combine`

参数说明

参数名	参数类型	是否必填	默认值	描述
m	Integer	是	无	元素总数
n	Integer	是	无	选择的个数

示例

```
let result = uni.$lv.number.combine(5, 3);
console.log(result);
let result2 = uni.$lv.number.combine(7, 4);
console.log(result2);
```

返回值

组合总数

阶乘

方法名

`factorial`

参数说明

参数名	参数类型	是否必填	默认值	描述
max	Integer	是	无	最大值
min	Integer	是	无	最小值

示例

```
let result = uni.$lv.number.factorial(5, 3);
console.log(result);
let result2 = uni.$lv.number.factorial(7, 4);
console.log(result2);
```

返回值

阶乘的值

随机值

方法名

`random`

参数说明

参数名	参数类型	是否必填	默认值	描述
min	Integer	是	无	最小值
max	Integer	是	无	最大值

示例

```
let result = uni.$lv.number.random(1, 10);
console.log(result);
let result2 = uni.$lv.number.random(1, 100);
console.log(result2);
```

返回值

随机的一个值（包含最小值和最大值）

数字格式化

方法名

`format`

> 一般用于价格展示

参数说明

参数名	参数类型	是否必填	默认值	描述
number	Integer	是	无	要格式化的数字
decimals	Integer	否	0	保留几位小数
decimalPoint	String	否	.	小数点符号
thousandsSeparator	String	否	,	千分位符号

示例

```
let result = uni.$lv.number.format(1234567.56);
console.log(result);
let result1 = uni.$lv.number.format(1234567.56, 2);
console.log(result1);
```

返回值

格式化后的数据

是否是数字

方法名

`isNumber`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	any	是	无	要验证的内容

示例

```
let result = uni.$lv.number.isNumber(123);
console.log(result);
let result1 = uni.$lv.number.isNumber(1234567.56);
console.log(result1);
let result2 = uni.$lv.number.isNumber("1234");
console.log(result2);
let result3 = uni.$lv.number.isNumber("1234abc");
console.log(result3);
let result4 = uni.$lv.number.isNumber("1234567.56");
console.log(result4);
```

返回值

true/false
true:是数字（包含小数点）
false:不是数字

是否是整数

方法名

`isDigits`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	any	是	无	要验证的内容

示例

```
let result = uni.$lv.number.isDigits(123);
console.log(result);
let result1 = uni.$lv.number.isDigits(1234567.56);
console.log(result1);
let result2 = uni.$lv.number.isDigits("1234");
console.log(result2);
let result3 = uni.$lv.number.isDigits("1234abc");
console.log(result3);
let result4 = uni.$lv.number.isDigits("1234567.56");
console.log(result4);
```

返回值

true/false
true:是整数
false:不是整数

金额转大写

方法名

`priceToUpper`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	Float/Integer	是	无	金额数据

示例

```
let result = uni.$lv.number.priceToUpper(123);
console.log(result);
let result1 = uni.$lv.number.priceToUpper(1234567.56);
console.log(result1);
let result2 = uni.$lv.number.priceToUpper(1234);
console.log(result2);
let result3 = uni.$lv.number.priceToUpper(12345);
console.log(result3);
let result4 = uni.$lv.number.priceToUpper(123456789.12);
console.log(result4);
```

返回值

转换后的数据

返回示例

```
壹佰贰拾叁元整
壹佰贰拾叁万肆仟伍佰陆拾柒元伍角陆分
壹仟贰佰叁拾肆元整
壹万贰仟叁佰肆拾伍元整
壹亿贰仟叁佰肆拾伍万陆仟柒佰捌拾玖元壹角贰分
```

高精度加法

方法名

`add`

参数说明

参数名	参数类型	是否必填	默认值	描述
a	Float/Integer	是	无	加数
b	Float/Integer	是	无	被加数

示例

```
let a = 0.1;
let b = 0.2;
console.log(a + b)
let result = uni.$lv.number.add(a, b);
console.log(result)
```

返回值

计算后的值

返回示例

```
0.30000000000000004
0.3
```

高精度减法

方法名

`sub`

参数说明

参数名	参数类型	是否必填	默认值	描述
a	Float/Integer	是	无	减数
b	Float/Integer	是	无	被减数

示例

```
let a = 0.3;
let b = 0.1;
console.log(a - b)
let result = uni.$lv.number.sub(a, b);
console.log(result)
```

返回值

计算后的值

返回示例

```
0.19999999999999998
0.2
```

高精度乘法

方法名

`mul`

参数说明

参数名	参数类型	是否必填	默认值	描述
a	Float/Integer	是	无	乘数
b	Float/Integer	是	无	被乘数

示例

```
let a = 0.6;
let b = 3;
console.log(a * b)
let result = uni.$lv.number.mul(a, b);
console.log(result)
```

返回值

计算后的值

返回示例

```
1.7999999999999998
1.8
```

高精度除法

方法名

`div`

参数说明

参数名	参数类型	是否必填	默认值	描述
a	Float/Integer	是	无	被除数
b	Float/Integer	是	无	除数

示例

```
let a = 1;
let b = 6;
console.log(a / b)
let result = uni.$lv.number.div(a, b);
console.log(result)
```

返回值

计算后的值

返回示例

```
0.16666666666666666
0.16666666666666666
```

对象转url参数

方法名

`queryParams`

参数说明

参数名	参数类型	是否必填	默认值	描述
data	Object	是	无	要转换的对象
isPrefix	Boolean	否	true	是否自动加上"?"
arrayFormat	String	否	brackets	对象属性中数组的转换方式，参考 arrayFormat 格式

arrayFormat 格式

1.indices
`结果: ids[0]=1&ids[1]=2&ids[2]=3`
2.brackets
`结果: ids[]=1&ids[]=2&ids[]=3`
3.repeat
`结果: ids=1&ids=2&ids=3`
4.comma
`结果: ids=1,2,3`

示例

```
let obj = {
  a: 1,
  b: 2,
  c: 3,
  d: [1, 2, 3]
};
let result = uni.$lv.object.queryParams(obj);
console.log(result);
let result2 = uni.$lv.object.queryParams(obj, true, "indices");
console.log(result2);
```

对象转url参数

```
let result3 = uni.$lv.object.queryParams(obj, true, "brackets");
console.log(result3);
let result4 = uni.$lv.object.queryParams(obj, true, "repeat");
console.log(result4);
let result5 = uni.$lv.object.queryParams(obj, true, "comma");
console.log(result5);
```

返回值

转换后的字符串

克隆

方法名

`clone`

参数说明

参数名	参数类型	是否必填	默认值	描述
obj	Object	是	无	要克隆的对象

示例

```
let obj = {
  a: 1,
  b: 2,
  c: 3,
  d: [1, 2, 3]
};
let result = uni.$lv.object.clone(obj);
console.log(result);
result.a = 123;
console.log(result);
console.log(obj);
```

返回值

克隆后的对象

合并对象

方法名

`merge`

参数说明

参数名	参数类型	是否必填	默认值	描述
target	Object	是	无	要合并的源对象
source	Object	是	无	要合并的目标对象

示例

```
let obj = {
  a: 1,
};
let obj2 = {
  b: 2,
  c: 2
};
let result = uni.$lv.object.merge(obj, obj2);
console.log(result);
console.log(obj);
console.log(obj2);
```

返回值

合并后的对象，如果参数中不为对象则返回false

获取对象的属性

方法名

`getProperty`

参数说明

参数名	参数类型	是否必填	默认值	描述
obj	Object	是	无	对象
key	String	是	无	需要获取的属性字段，用于通过类似'a.b.c'的形式去获取一个对象的的属性的形式

示例

```
let object = {
  userInfo: {
    address: {
      province: '深圳'
    }
  }
}
let result = uni.$lv.object.getProperty(object, "userInfo");
console.log(result);
let result1 = uni.$lv.object.getProperty(object, "userInfo.address");
console.log(result1);
let result2 = uni.$lv.object.getProperty(object, "userInfo.home.address");
console.log(result2);
```

返回值

获取的属性值

设置对象的属性值

方法名

`setProperty`

参数说明

参数名	参数类型	是否必填	默认值	描述
obj	Object	是	无	对象
key	String	是	无	属性字段
value	Any	是	无	设置的值

示例

```
let object = {
  userInfo: {
    address: {
      province: '深圳'
    }
  }
}
uni.$lv.object.setProperty(object, "userInfo.address.province", "武汉");
console.log(object);
uni.$lv.object.setProperty(object, "userInfo.home.address", "湖北省武汉市");
console.log(object);
```

返回值

无

是否是对象

方法名

`isObject`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	Any	是	无	要验证的内容

示例

```
let object = {
  userInfo: {
    address: {
      province: '深圳'
    }
  }
}
let result = uni.$lv.object.isObject(object);
console.log(result);
let result1 = uni.$lv.object.isObject("123");
console.log(result1);
```

返回值

true/false
true:是对象
false:不是对象

移除空值

方法名

`trim`

参数说明

参数名	参数类型	是否必填	默认值	描述
obj	Object	是	无	要处理的对象

示例

```
let object = {
  a: 1,
  b: 2,
  c: "",
  d: " "
}
let result = uni.$lv.object.trim(object);
console.log(result);
```

返回值

移除空值后的对象

排序

方法名

`sort`

参数说明

参数名	参数类型	是否必填	默认值	描述
obj	Object	是	无	要处理的对象
type	String	否	asc	排序方式，asc:升序，desc:降序

示例

```
let object = {
  b: 2,
  a: 1,
  d: " ",
  c: "",
  f: "",
  e: "我是一段中文",
}
let result = uni.$lv.object.sort(object);
console.log(result);
let result2 = uni.$lv.object.sort(object, "desc");
console.log(result2);
```

返回值

排序后的对象

使用

说明

此插件集成自优秀的开源请求库：luch-request
官方网站：https://www.quanzhan.co/luch-request
官方文档：https://www.quanzhan.co/luch-request/guide/3.x/
插件市场：https://ext.dcloud.net.cn/plugin?id=392
具体使用方法可参考官方文档

插件优势：

1. 基于 Promise 对象实现更简单的 request 使用方式，支持请求和响应拦截
2. 支持全局挂载
3. 支持多个全局配置实例
4. 支持自定义验证器
5. 支持文件上传/下载
6. 支持task 操作
7. 支持自定义参数
8. 支持多拦截器
9. 对参数的处理比uni.request 更强

项目中的使用

```
```

请求示例

```
```

生成uuid

方法名

`uuid`

参数说明

参数名	参数类型	是否必填	默认值	描述
len	Integer	否	32	uuid的长度
first	String	否	空	自定义首字母
radix	Integer	否	null	生成uuid的基数(意味着返回的字符串都是这个基数),2-二进制,8-八进制,10-十进制,16-十六进制

示例

```
let result = uni.$lv.string.uuid();
console.log(result);
let result1 = uni.$lv.string.uuid(64, "", 2);
console.log(result1);
let result2 = uni.$lv.string.uuid(64, "", 8);
console.log(result2);
let result3 = uni.$lv.string.uuid(64, "", 10);
console.log(result3);
let result4 = uni.$lv.string.uuid(64, "", 16);
console.log(result4);
let result5 = uni.$lv.string.uuid(64, "l");
console.log(result5);
```

返回值

生成的uuid字符串

去除空格

方法名

`trim`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	需要去除空格的字符串
pos	String	否	both	去除空格类型，both(左右)，left，righ，all

示例

```
let str = " 123456 789 0 123 111 ";
console.log(str);
let result = uni.$lv.string.trim(str);
console.log(result);
let result1 = uni.$lv.string.trim(str, "right");
console.log(result1);
let result2 = uni.$lv.string.trim(str, "left");
console.log(result2);
let result3 = uni.$lv.string.trim(str, "all");
console.log(result3);
```

返回值

返回去除空格后的字符串

是否是字符串

方法名

`isString`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	Any	是	无	要验证的内容

示例

```
let result = uni.$lv.string.isString(123);
console.log(result);
let result1 = uni.$lv.string.isString("123");
console.log(result1);
```

返回值

true/false
true:是字符串
false:不是字符串

验证是否包含某个值

方法名

`isContains`

参数说明

参数名	参数类型	是否必填	默认值	描述
str1	String	是	无	原始字符串
str2	String	是	无	验证的字符串

示例

```
let result = uni.$lv.string.isContains("abcdefg", "aa");
console.log(result);
let result1 = uni.$lv.string.isContains("abcdefg", "ab");
console.log(result1);
```

返回值

true/false
true:包含
false:不包含

是否是json字符串

方法名

`isJsonString`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	验证的字符串

示例

```
let result = uni.$lv.string.isJsonString("abcdefg");
console.log(result);
let result1 = uni.$lv.string.isJsonString("{\"a\":1,\"b\":2}");
console.log(result1);
let result2 = uni.$lv.string.isJsonString({
  "a": 1,
  "b": 2
});
console.log(result2);
```

返回值

true/false
true:是
false:不是

字节格式化

方法名

`sizeFormat`

参数说明

参数名	参数类型	是否必填	默认值	描述
size	Integer	是	无	字节大小

示例

```
let result = uni.$lv.string.sizeFormat(1234567);
console.log(result);
let result1 = uni.$lv.string.sizeFormat(123456789);
console.log(result1);
let result2 = uni.$lv.string.sizeFormat(123456789123);
console.log(result2);

//输出结果
1.18MB
117.74MB
114.98GB
```

返回值

格式化后的字符串

字符串截取

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	要截取的字符串
len	Integer	否	10	截取长度
end	String	否	...	截取后的结束字符串

```
```

截取后的字符串

隐藏字符

方法名

`hideString`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	要处理的字符串
starCount	Integer	否	4	显示隐藏字符的个数
prevCount	Integer	否	3	开始显示的字符串个数
endCount	Integer	否	4	结束显示的字符串个数

示例

```
let result = uni.$lv.string.hideString("这是一条很长的字符串这是一条很长的字符串这是一条很
console.log(result);
let result1 = uni.$lv.string.hideString("13412345678");
console.log(result1);

//输出结果
这是一**********的字符串
134****5678
```

返回值

处理后的字符串

字符串反转

方法名

`reverse`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	要处理的字符串

示例

```
let str = "这是一条很长的字符串这是一条很长的字符串这是一条很长的字符串";
let result = uni.$lv.string.reverse(str);
console.log(result);
console.log(str);
```

返回值

处理后的字符串

随机字符串

方法名

`random`

参数说明

参数名	参数类型	是否必填	默认值	描述
length	Integer	是	无	长度
num	Boolean	否	true	是否含有数字
lowerLetters	Boolean	否	true	是否含有小写字母
upperLetters	Boolean	否	true	是否含有大写字母
specialCharacters	Boolean	否	true	是否含有特殊字符

示例

```
let result = uni.$lv.string.random(32);
console.log(result);
```

返回值

生成的字符串

返回示例

```
Zj9ghikkGI7s8VXWw4YqdP%@l987rD2n
89EzFILaKRWXMcPMvjFNTf$uno$JSKwO
ZL6wUwMnRHZ7TXau!%CxF0o5Bkt9AR7i
t@nD9dhQU676V^0gjbo@fvP6&ANSyJQy
```

转大写

方法名

`toUpper`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	转换字符串

示例

```
let result = uni.$lv.string.toUpper("abcdEfg");
console.log(result);
```

返回值

生成的字符串

返回示例

```
ABCDEFG
```

转小写

方法名

`toLower`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	转换字符串

示例

```
let result = uni.$lv.string.toLower("ABCDEfG");
console.log(result);
```

返回值

生成的字符串

返回示例

```
abcdefg
```

下划线转驼峰

方法名

`underlineToHump`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	转换字符串

示例

```
let result = uni.$lv.string.underlineToHump("abcd_efg_hij");
console.log(result);
let result1 = uni.$lv.string.underlineToHump("aaa_bbb");
console.log(result1);
```

返回值

生成的字符串

返回示例

```
abcdEfgHij
aaaBbb
```

驼峰转下划线

方法名

`humpToUnderline`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	转换字符串

示例

```
let result = uni.$lv.string.humpToUnderline("humpToUnderline");
console.log(result);
```

返回值

生成的字符串

返回示例

```
hump_to_underline
```

url解析

方法名

`parseUrl`

参数说明

参数名	参数类型	是否必填	默认值	描述
url	String	是	无	要解析的url

示例

```
let result = uni.$lv.string.parseUrl("http://abc.com:8080/dir/index.html?id=255&m=
console.log(result);
```

返回值

Object

返回值参数

字段名	类型	描述
file	String	文件名
hash	String	hash或锚点值
host	String	域名
params	Object	地址的参数对象格式
path	String	文件路径
port	String	端口
protocol	String	协议
query	String	地址参数字符串格式

字段名	类型	描述
relative	String	路径+参数
segments	Array	路径集合
source	String	源地址

返回示例

```
{
  "file": "index.html",
  "hash": "top",
  "host": "abc.com",
  "params": {
    "id": "255",
    "m": "hello"
  },
  "path": "/dir/index.html",
  "port": "8080",
  "protocol": "http",
  "query": "?id=255&m=hello",
  "relative": "/dir/index.html?id=255&m=hello#top",
  "segments": ["dir", "index.html"],
  "source": "http://abc.com:8080/dir/index.html?id=255&m=hello#top"
}
```

字符串转十六进制

方法名

`stringToHex`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	转换字符串
isUrlEncode `v1.1.5`	Boolean	否	true	是否将字符串先UrlEncode

示例

```
let result = uni.$lv.string.stringToHex("http://abc.com:8080/dir/index.html?id=25
console.log(result);
let result1 = uni.$lv.string.stringToHex("http://abc.com:8080这一条数据包含中文");
console.log(result1);
```

返回值

生成的字符串

返回示例

```
68747470253341253246253246616632e636f6d25334138303830253246646972253246696e646578
68747470253341253246253246616632e636f6d25334138303830254538254246253939254534254
```

十六进制转字符串

方法名

`hexToString`

参数说明

参数名	参数类型	是否必填	默认值	描述
str	String	是	无	转换字符串
isUrlDecode `v1.1.5`	Boolean	否	true	结果是否UrlDecode

示例

```
let result = uni.$lv.string.hexToString("687474703a2f2f6162632e636f6d3a383038302f6
console.log(result);
let result1 = uni.$lv.string.hexToString("68747470253341253246253246162632e636f6c
console.log(result1);
```

返回值

生成的字符串

返回示例

```
http://abc.com:8080/dir/index.html?id=255&m=hello#top
http://abc.com:8080这一条数据包含中文
```

介绍

定时器插件是为了解决频繁使用setInteval和setTimeout，如果频繁使用他们会造成内存浪费，定时器插件是通过一个setTimeout生成id分发出去，总之是在一个setTimeout处理定时任务，而不需要写更多的setInteval和setTimeout，这样也便于管理

> 建议在使用之前先clear掉所有的定时器，在页面卸载的时候您需要调用remove方法移除特定的定时器

添加定时器

方法名

`add`

参数说明

参数名	参数类型	是否必填	默认值	描述
func	Function	是	无	定时器执行的函数
name	String	是	无	定时器标识，每个定时器标识不能相同，否则会被覆盖

示例

```
// 添加定时器
add() {
    let count1 = 0;
    uni.$lv.timer.add(() => {
      count1++;
      console.log("一次性定时器：" + count1);
      if (count1 >= 20) {
        // 20秒后结束
        uni.$lv.timer.remove("timer1");
      }
    }, "timer1");

    // 创建一个轮询定时器
    this.intervalTimer();
},


// 轮询定时器
intervalTimer() {
    let count = 0;
    // 不管有没有当前的定时器，先移除
    uni.$lv.timer.remove("timer2");
    uni.$lv.timer.add(() => {
      count++;
      console.log("轮询定时器：" + count);
```

添加定时器

```
        if (count >= 30) {
          // 30秒后继续
          this.intervalTimer();
        }
      }, "timer2");
  },
```

返回值

无

移除定时器

方法名

`remove`

参数说明

参数名	参数类型	是否必填	默认值	描述
name	String	是	无	定时器标识

示例

```
uni.$lv.timer.remove("timer1");
```

返回值

无

清空定时器

方法名

`clear`

参数说明

无

示例

```
uni.$lv.timer.clear();
```

返回值

无

暂停定时器

方法名

`stop`

参数说明

无

示例

```
uni.$lv.timer.stop();
```

返回值

无

继续定时器

方法名

`start`

参数说明

无

示例

```
uni.$lv.timer.start();
```

返回值

无

设置一个定时器

方法名

`setTimer`

> 此方法您不需要考虑先移除定时器再设置，内部已做先移除后添加的功能，但是如果页面卸载您还需要移除当前定时器的

参数说明

参数名	参数类型	是否必填	默认值	描述
params	Object	是	无	参数对象，请参考下方说明

params参数说明

参数名	参数类型	是否必填	默认值	描述
name	String	是	无	定时器名称
seconds	Integer	是	无	定时器秒数
onChange	Function	否	无	定时器改变事件函数，返回结构参考下方说明
onComplate	Function	否	无	定时器结束事件，函数返回：true

onChange返回结构说明

字段名	类型	描述
outSeconds	Integer	已改变的秒数
remainSeconds	Integer	剩余的秒数

示例

设置一个定时器

```
// 添加定时器
uni.$lv.timer.setTimer({
  name: "timer1",
  seconds: 30,
  onChange: (res) => {
    console.log("定时器改变：" + JSON.stringify(res))
  },
  onComplate: (res) => {
    console.log("定时器结束：" + res)
  }
});
```

返回值

无

电子邮箱

方法名

`email`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.email("123456");
console.log(result);
let result1 = uni.$lv.validate.email("123456@");
console.log(result1);
let result2 = uni.$lv.validate.email("123456@qq.com");
console.log(result2);
```

返回值

true/false
true:是电子邮箱格式
false:不是电子邮箱格式

手机格式

方法名

`mobile`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.mobile("13412345678");
console.log(result);
let result1 = uni.$lv.validate.mobile("15823451234");
console.log(result1);
let result2 = uni.$lv.validate.mobile("13211111111");
console.log(result2);
```

返回值

true/false
true:是手机格式
false:不是手机格式

URL格式

方法名

`url`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.url("http://www.baidu");
console.log(result);
let result1 = uni.$lv.validate.url("http://www.baidu.com");
console.log(result1);
let result2 = uni.$lv.validate.url("rtsp://www.baidu.com");
console.log(result2);
```

返回值

true/false
true:是url格式
false:不是url格式

身份证号

方法名

`idCard`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.idCard("420113201911211003");
console.log(result);
let result1 = uni.$lv.validate.idCard("42011320191121");
console.log(result1);
```

返回值

true/false
true:是正确的身份证号
false:不是正确的身份证号

车牌号

方法名

`carNumber`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.carNumber("鄂A12349");
console.log(result);
let result1 = uni.$lv.validate.carNumber("鄂A12349A");
console.log(result1);
```

返回值

true/false
true:是车牌号格式
false:不是车牌号格式

全部中文

方法名

`chinese`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.chinese("bubaohanzhongwen");
console.log(result);
let result1 = uni.$lv.validate.chinese("不完全是中文zhongwen");
console.log(result1);
let result2 = uni.$lv.validate.chinese("全部是中文");
console.log(result2);
```

返回值

true/false
true:是
false:不是

包含中文

方法名

`containChinese`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.containChinese("bubaohanzhongwen");
console.log(result);
let result1 = uni.$lv.validate.containChinese("不完全是中文zhongwen");
console.log(result1);
let result2 = uni.$lv.validate.containChinese("全部是中文");
console.log(result2);
let result3 = uni.$lv.validate.containChinese("buwanquan不完全是中文zhongwen");
console.log(result3);
```

返回值

true/false
true:包含中文
false:不包含中文

用户名格式

方法名

`isUsername`

> 验证规则：只能包含字母数字和下划线，且不能以数字开头

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.isUsername("bubaohanzhongwen");
console.log(result);
let result1 = uni.$lv.validate.isUsername("bubaohan_zhongwen");
console.log(result1);
let result2 = uni.$lv.validate.isUsername("1bubaohan_zhongwen");
console.log(result2);
```

返回值

true/false
true:是
false:否

固定电话

方法名

`telephone`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.telephone("027-12345678");
console.log(result);
let result1 = uni.$lv.validate.telephone("0551-1234567");
console.log(result1);
```

返回值

true/false
true:是
false:不是

是否为空

方法名

`empty`

> 空对象，空数组，数字0，布尔值false(非字符串false),undefined,null都会验证为空

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.empty("0");
console.log(result);
let result1 = uni.$lv.validate.empty(0);
console.log(result1);
let result2 = uni.$lv.validate.empty(false);
console.log(result2);
let result3 = uni.$lv.validate.empty([]);
console.log(result3);
let result4 = uni.$lv.validate.empty({});
console.log(result4);
let result5 = uni.$lv.validate.empty(undefined);
console.log(result5);
let result6 = uni.$lv.validate.empty(null);
console.log(result6);
```

返回值

true/false
true:是
false:不是

图片格式

方法名

`image`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.image("123.jpg");
console.log(result);
```

返回值

true/false
true:是
false:不是

视频格式

方法名

`video`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.video("123.mp4");
console.log(result);
```

返回值

true/false
true:是
false:不是

全部字母

方法名

`letter`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.letter("123abc");
console.log(result);
let result1 = uni.$lv.validate.letter("abc");
console.log(result1);
```

返回值

true/false
true:是
false:不是

字母和空格

方法名

`letterSpace`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.letterSpace("123abc");
console.log(result);
let result1 = uni.$lv.validate.letterSpace("abc def");
console.log(result1);
let result2 = uni.$lv.validate.letterSpace("abcdef");
console.log(result2);
```

返回值

true/false
true:是
false:不是

字母和数字

方法名

`letterNumeric`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.letterNumeric("123abc");
console.log(result);
let result1 = uni.$lv.validate.letterNumeric("abc def");
console.log(result1);
let result2 = uni.$lv.validate.letterNumeric("abcdef");
console.log(result2);
```

返回值

true/false
true:是
false:不是

字母数字和空格

方法名

`letterNumericSpace`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.letterNumericSpace("123abc");
console.log(result);
let result1 = uni.$lv.validate.letterNumericSpace("abc def123");
console.log(result1);
let result2 = uni.$lv.validate.letterNumericSpace("abcdef&");
console.log(result2);
```

返回值

true/false
true:是
false:不是

数字和空格

方法名

`numericSpace`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.numericSpace("123abc");
console.log(result);
let result1 = uni.$lv.validate.numericSpace("123 456");
console.log(result1);
let result2 = uni.$lv.validate.numericSpace("abcdef123&");
console.log(result2);
```

返回值

true/false
true:是
false:不是

全部小写

方法名

`lower`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.lower("Abcdef");
console.log(result);
let result1 = uni.$lv.validate.lower("abcdef");
console.log(result1);
let result2 = uni.$lv.validate.lower("abcdef abc");
console.log(result2);
```

返回值

true/false
true:是
false:不是

全部大写

方法名

`upper`

参数说明

参数名	参数类型	是否必填	默认值	描述
value	String	是	无	要验证的内容

示例

```
let result = uni.$lv.validate.upper("Abcdef");
console.log(result);
let result1 = uni.$lv.validate.upper("ABCDEF");
console.log(result1);
let result2 = uni.$lv.validate.upper("ABCDEF ABC");
console.log(result2);
```

返回值

true/false
true:是
false:不是

是否是微信

方法名

`isWechat`

参数说明

无

示例

```
let result = uni.$lv.validate.isWechat();
uni.$lv.func.toast(result);
```

返回值

true/false
true:是
false:不是

是否是支付宝

方法名

`isAlipay`

参数说明

无

示例

```
let result = uni.$lv.validate.isAlipay();
uni.$lv.func.toast(result);
```

返回值

true/false
true:是
false:不是