林风工具使用指南

工具简介

该工具为林风整合的工具集，本工具内并非全部内容为林风制作，而是整合了些常用的工具，有些不常用的也被整合到了一起，下面就开始介绍该工具集的内容吧。

> 注意：工具以mtlf(是myToolLinFeng的缩写)开头的为小编写的工具，其它的全不是小编写的，只是稍作修改而已。小编可是一个励志成为一名伟大的copy工程师！！！

下载地址

未压缩林风工具js：http://linfeng.vip/assets/tools/linfengTools.js

压缩版林风工具js：http://linfeng.vip/assets/tools/linfengTools-min.js

林风工具css：http://linfeng.vip/assets/tools/linfengTools.css

全部下载：http://linfeng.vip/assets/tools/lft.zip

获取参数

获取导航传递的参数

```
/**
 * 获取导航栏的参数
 * @param {string} variable
 * @returns parm
 * 使用方式：id = myToolGetQueryVariable('id')
 */
myToolGetQueryVariable(variable)
```

传递参数

- variable：String -> 需要截取的类型

返回结果

- String or boolean -> 截取成功的字符串或false

简易写法：

```
    const {getData} = linfengTools
    console.log(getData('id'))
```

日期工具

转换日期为通用格式

```
/**
 * 日期默认转换为YY-MM-DD hh mm ss格式
 * @param {date} date 需要转换的日期 可不写
 * @returns 转换后的日期
 */
myToolFormatDateDefault(date)
```

传递参数:

- date：date -> 需要转换的日期 可不写 默认为当前时间

返回结果:

- string -> 转换后的日期

简易写法:

```
    const {date} = linfengTools
    console.log(date.fd());
```

转换日期为指定格式

```
/**
 * 根据指定格式进行格式化日期
 * @param {string} fmt 格式 例如："YYYY-MM-DD hh:mm"
 * @param {date} date 日期对象
 * @returns 返回格式化完成的字符串
 * Y年 M月 D日 h时 m分 s秒
 */
 myToolFormatDate(date, fmt)
```

传递参数:

- date：日期对象，可不写
- fmt：格式 例如："YYYY-MM-DD hh:mm:ss"

简易写法:

```
    const {date} = linfengTools
    console.log(date.format('YY:MM-DD'))
```

正则工具

正则工具小编正在快马加鞭的打造中。。

oldTools

简易写法暂不支持oldTools

由于正则工具较多，所以不再过多解释

验证特殊字符

```
/**
 * @param {string} str 需要验证的字符串
 * @desc: 特殊字符校验 除了下划线
 */
 myToolRegexContainSpecial(str)
```

```
/**
 * @param {string} str 需要验证的字符串
 * @desc: 特殊字符校验 除了下划线 斜杠 反斜杠 冒号
 */
myToolRegexContainSpecialMore(str)
```

中文校验

```
/**
 * @param {string} str 需要验证的字符串
 * @desc: 中文校验
 */
 myToolRegexCheckChinese(val)
```

用户名验证

```
/**
 * @param {string} str 需要验证的字符串
 * @desc: 特殊字符 除了下划线
 */
 myToolRegexUser(str)
```

```
/**
 * 用户名验证，只允许输入英文大小写和下划线
 * @author linfeng 林风
 * @param {String} str 需要判断的字符串
 * @param {int} star 长度最小值，可不写，默认6
 * @param {int} end 长度最大值，可不写，默认16
 * @returns 返回布尔值，符合条件为true，否则相反
 */
mtlfRegexUser(str, star, end)
// 默认是[6,16]
```

```
/**
 * 用户名验证，只允许输入英文大小写和下划线
 * @param {String} str 需要判断的字符串
 * @param {JSON} json 两个参数star和end可不写，可漏写
 * @returns
 */
mtlfRegexUserJson(str, json)
// 默认是[6,16]
```

色值转换

这是个工具JSON，也是copy的别人的代码，具体用法如下：

RGB(A)颜色转换为HEX十六进制的颜色值：

```
    /**
     * RGB(A)颜色转换为HEX十六进制的颜色值
     * @param {string} val rgb(r,g,b)例如：rgb(155,10,20)
     * 当然，也可以是rgba，rgb(155,10,20,0.5)
     * @returns 以对象形式返回alpha、r、g、b和hex十六进制色值
     */
     myToolColorChange.rgbToHex(value)
```

传递参数

- value：

  rgb(r,g,b)例如：rgb(155,10,20)

  当然，也可以是rgba，例如：rgb(155,10,20,0.5)

返回参数

- JSON：

  以对象形式返回alpha、r、g、b和hex十六进制色值

简易写法：

```
    const {color} = linfengTools
    console.log(color.toHex('rgb(255,255,255)'));
```

HEX十六进制颜色值转换为RGB(A)颜色值

```
    /**
     * HEX十六进制颜色值转换为RGB(A)颜色值
     * @param {string} val 十六进制色值，例如：#ff0000
     * 当然也可以是三色值，例如#ff0
     * @returns 以对象形式返回rgb、r、g、b
     */
     myToolColorChange.hexToRgb(value)
```

传递参数

- value：

  十六进制色值，例如：#ff0000
  当然也可以是三色值，例如#ff0

返回参数

- JSON：

  以对象形式返回rgb、r、g、b

简易写法：

```
    const {color} = linfengTools
    console.log(color.toRgb('#f00'));
```

背景图片

该功能需要搭配linfengTools.css使用，需要将linfengTools.css引入

```
/**
 * 背景图片
 * 注意：该功能需要搭配linfengTools.css使用
 * @author linfeng 林风
 * @param {string} src 背景图片路径
 * @param {boolean} bgcBool 是否有遮罩层
 * @param {float} alpha 设置不透明度
 * @param {string} color 设置遮罩层十六进制色值
 */
mtlfBgi(src, bgcBool, alpha, color)
```

传递参数

- src： string -> 必写，背景图片路径
- bgcBool：boolean -> 选填，是否有遮罩层
- alpha：float -> 设置不透明度
- color：string -> 设置遮罩层颜色，只能是十六进制色值，可缩写

无返回值

代码示例：

```
mtlfBgi('./bgc.jpg', true, 0.3, '#000')
```

简易写法：

```
    const {bgi} = linfengTools
    bgi('./a.jpg', true, 0.3, '#000')
```

新增功能：

第五个参数，可以写宽度，支持所有width原支持的单位，但是不推荐使用

例子：

```
    const {bgi} = linfengTools
    bgi('./a.jpg', true, 0.3, '#000', '70%')
```

特殊工具

特效：浮出文字

该功能需要搭配linfengTools.css使用，需要将linfengTools.css引入

```
/**
 * 使用文字上浮消失
 * @param {Array} textArr
 * 传入的参数是你想要设定的参数
 * 若不设定会用默认值
 * 注意：该功能需要引入linfengTools.css
 */
myToolUseTheText(textArr)
```

传递参数

- textArr：Array -> 浮出的文字内容数组

无返回值

简易写法：

```
    const {animat} = linfengTools
    animat.text([0,1])
```

浏览器标题(title)文本切换

```
/**
 * 文本切换
 * @param {string} leave 离开时的文本，可为空
 * @param {string} retur 刚返回时文本，可为空
 * @param {string} selectedTitleText 返回后的文本，可为空
 * 默认返回后文本为原来title，可不写
 */
myToolTitleText(leave, retur, selectedTitleText)
```

传递参数

- leave：string -> 离开时的文本，可为空，默认为：(┬_┬)别不理我
- retur：string -> 刚返回时文本，可为空，默认为：o(￣▽￣)ブ欢迎回来
- selectedTitleText：string -> 返回后的文本，可为空，默认为自己设置的title

无返回值

简易写法：

```
    const {titleText} = linfengTools
    titleText('你干嘛', '别再有下次')
```

设备检测

该工具主要用于检测移动端与PC端之间的区别

PC检测

```
/**
 * 检测是否为PC端
 * @returns 若为PC端返回true否则返回false
 */
myToolIfMobile()
```

无参数

返回结果：

- boolean -> 若为PC端返回true否则返回false

简易写法：

```
    const {equipment} = linfengTools
    console.log(equipment.pc())
```

移动检测并跳转

```
/**
 * 检测是否为移动端并进行跳转
 * @param {string} url 需要跳转的移动端页面
 */
myToolIfGoMobile(url)
```

传递参数

- url：string -> 需要跳转的移动端页面 可不写

返回结果

- 若没写参数则返回布尔值

  移动端返回true

  pc端返回false

简易写法：

```
    const {equipment} = linfengTools
    console.log(equipment.mobile());
```

附linfengTools.css说明

清除默认样式

该css内清除了大部分标签的默认样式

启用移动端模式

如果你想做移动端样式的清楚浏览器默认样式，不妨试试在body标签中加入mobile类

```
<body class="mobile"></body>
```

这样body的默认样式就被改为移动端的常用样式了，但是还是需要自己引入normalize.css文件

清除浮动

还在为清除浮动而脑壳子疼吗？不要着急不要着急，不要998不要998，只要你引入linfengTools.css就可以把它带回家，记得在需要清除浮动的父盒子上添加clearFloat类哦

```
<div class="clearFloat"></div>
```