# [better-scroll](https://www.npmjs.com/package/better-scroll) NPM

窗体顶端

窗体底端

**[IMG_256](https://www.npmjs.com/package/better-scroll)**

inspired by iscroll, and it has a better scroll perfermance **<https://ustbhuangyi.github.io/better-scroll/>**

## **立即使用**

<body>

  <div id="wrapper">

    <ul>

       <li>...</li>

       <li>...</li>

       ...

    </ul>

  </div>

<script type="text/javascript" src="better-scroll.js"></script>

<script type="text/javascript">

  new BScroll(document.getElementById('wrapper'));

</script>

</body>

搞定 !

## **通过npm引入**

安装better-scroll

npm install better-scroll --save-dev

引入better-scroll

**import** BScroll **from** 'better-scroll'

如果不支持import, 请使用

var BScroll **=** require('better-scroll')

## **DEMO**

better-scroll 的源码是基于 Webpack 构建的

首先，clone项目源码

git clone https://github.com/ustbhuangyi/better-scroll.git

安装依赖

cd better-scroll

npm install

测试demo页

npm run dev

打开浏览器访问如下地址, 查看效果

localhost:9090

## **Options 参数**

Example:

let scroll **=** **new** BScroll(document.getElementById('wrapper'), {

  startX**:** 0,

  startY**:** 0

})

Options List:

* startX: 0 开始的X轴位置
* startY: 0 开始的Y轴位置
* scrollY: true 滚动方向
* click: true 是否启用click事件
* directionLockThreshold: 5
* momentum: true 是否开启拖动惯性
* bounce: true 是否启用弹力动画效果，关掉可以加速
* selectedIndex: 0
* rotate: 25
* wheel: false 该属性是给 picker 组件使用的，普通的列表滚动不需要配置
* snap: false 是否开启捕捉元素，当为 true 时，捕捉的元素会根据可滚动的位置和滚动区域计算得到可滑动几页。
* snapLoop: false 是否创建当前滚动元素子集的拷贝
* snapThreshold: 0.1 滑动的长度限制，只有大于这个距离才会触发事件
* swipeTime: 2500 swipe 持续时间
* bounceTime: 700 弹力动画持续的毫秒数
* adjustTime: 400
* swipeBounceTime: 1200
* deceleration: 0.001 滚动动量减速越大越快，建议不大于0.01
* momentumLimitTime: 300 惯性拖动的回弹时间
* momentumLimitDistance: 15 惯性拖动的回弹距离
* resizePolling: 60 重新调整窗口大小时，重新计算better-scroll的时间间隔
* probeType: 1 监听事件的触发时间，1为即时触发，3为延迟到事件完毕后触发
* preventDefault: true 是否阻止默认事件
* preventDefaultException: { tagName: /^(INPUT|TEXTAREA|BUTTON|SELECT)$/ } 阻止默认事件
* HWCompositing: true 是否启用硬件加速
* useTransition: true 是否使用CSS3的Transition属性，否则使用-requestAnimationFram代替
* useTransform: true 是否使用CSS3的Transform属性
* probeType: 1 滚动的时候会派发scroll事件，会截流。2滚动的时候实时派发scroll事件，不会截流。 3除了实时派发scroll事件，在swipe的情况下仍然能实时派发scroll事件

## **Events 事件**

Example:

let scroll **=** **new** BScroll(document.getElementById('wrapper'),{

   probeType**:** 3

})

scroll.on('scroll', (pos) => {

  console.log(pos.x **+** '~' **+** posx.y)

**...**

})

Events 列表

* beforeScrollStart - 滚动开始之前触发
* scrollStart - 滚动开始时触发
* scroll - 滚动时触发
* scrollCancel - 取消滚动时触发
* scrollEnd - 滚动结束时触发
* flick - 触发了 fastclick 时的回调函数
* refresh - 当 better-scroll 刷新时触发
* destroy - 销毁 better-scroll 实例时触发

## **派发滚动**

* scrollTo(x, y, time, easing) 滚动到某个位置，x,y 代表坐标，time 表示动画时间，easing 表示缓动函数

Example:

let scroll **=** **new** BScroll(document.getElementById('wrapper'))

scroll.scrollTo(0, 500)

**...**

* scrollToElement(el, time, offsetX, offsetY, easing) 滚动到  某个元素，el（必填）表示 dom 元素，time 表示动画，offsetX 和 offsetY 表示坐标偏移量，easing 表示缓动函数

# **[better-scroll](https://github.com/ustbhuangyi/better-scroll) GitHub**

[IMG_256](https://www.npmjs.com/package/better-scroll)

inspired by iscroll, and it has a better scroll perfermance <https://ustbhuangyi.github.io/better-scroll/>

## **立即使用**

<body>

<div id="wrapper">

<ul>

<li>...</li>

<li>...</li>

...

</ul>

</div>

<script type="text/javascript" src="better-scroll.js"></script><script type="text/javascript"> new BScroll(document.getElementById('wrapper'));</script>

</body>

搞定 !

## **通过npm引入**

安装better-scroll

npm install better-scroll --save-dev

引入better-scroll

import BScroll from 'better-scroll'

如果不支持import, 请使用

var BScroll = require('better-scroll')

## **DEMO**

better-scroll 的源码是基于 Webpack 构建的

首先，clone项目源码

git clone https://github.com/ustbhuangyi/better-scroll.git

安装依赖

cd better-scroll

npm install

测试demo页

npm run dev

打开浏览器访问如下地址, 查看效果

localhost:9090

## **Options 参数**

Example:

let scroll = new BScroll(document.getElementById('wrapper'), {

startX: 0,

startY: 0

})

Options List:

* startX: 0 开始的X轴位置
* startY: 0 开始的Y轴位置
* scrollY: true 滚动方向
* click: true 是否启用click事件
* directionLockThreshold: 5
* momentum: true 是否开启拖动惯性
* bounce: true 是否启用弹力动画效果，关掉可以加速
* selectedIndex: 0
* rotate: 25
* wheel: false 该属性是给 picker 组件使用的，普通的列表滚动不需要配置
* snap: false 是否开启捕捉元素，当为 true 时，捕捉的元素会根据可滚动的位置和滚动区域计算得到可滑动几页。
* snapLoop: false 是否创建当前滚动元素子集的拷贝
* snapThreshold: 0.1 滑动的长度限制，只有大于这个距离才会触发事件
* swipeTime: 2500 swipe 持续时间
* bounceTime: 700 弹力动画持续的毫秒数
* adjustTime: 400
* swipeBounceTime: 1200
* deceleration: 0.001 滚动动量减速越大越快，建议不大于0.01
* momentumLimitTime: 300 惯性拖动的回弹时间
* momentumLimitDistance: 15 惯性拖动的回弹距离
* resizePolling: 60 重新调整窗口大小时，重新计算better-scroll的时间间隔
* probeType: 1 监听事件的触发时间，1为即时触发，3为延迟到事件完毕后触发
* preventDefault: true 是否阻止默认事件
* preventDefaultException: { tagName: /^(INPUT|TEXTAREA|BUTTON|SELECT)$/ } 阻止默认事件
* HWCompositing: true 是否启用硬件加速
* useTransition: true 是否使用CSS3的Transition属性，否则使用-requestAnimationFram代替
* useTransform: true 是否使用CSS3的Transform属性
* probeType: 1 滚动的时候会派发scroll事件，会截流。2滚动的时候实时派发scroll事件，不会截流。 3除了实时派发scroll事件，在swipe的情况下仍然能实时派发scroll事件

## **Events 事件**

Example:

let scroll = new BScroll(document.getElementById('wrapper'),{

probeType: 3

})

scroll.on('scroll', (pos) => {

console.log(pos.x + '~' + posx.y)

...

})

Events 列表

* beforeScrollStart - 滚动开始之前触发
* scrollStart - 滚动开始时触发
* scroll - 滚动时触发
* scrollCancel - 取消滚动时触发
* scrollEnd - 滚动结束时触发
* flick - 触发了 fastclick 时的回调函数
* refresh - 当 better-scroll 刷新时触发
* destroy - 销毁 better-scroll 实例时触发

## **派发滚动**

* scrollTo(x, y, time, easing) 滚动到某个位置，x,y 代表坐标，time 表示动画时间，easing 表示缓动函数

Example:

let scroll = new BScroll(document.getElementById('wrapper'))scroll.scrollTo(0, 500)...

* scrollToElement(el, time, offsetX, offsetY, easing) 滚动到  某个元素，el（必填）表示 dom 元素，time 表示动画，offsetX 和 offsetY 表示坐标偏移量，easing 表示缓动函数