防抖

⭐等待一段时间如何这个事件在等待的过程中再次触发，那么重新计时，最终只有当事件停止触发一段事件后，才会执行对应的处理，比如 用户在输入一段时间后再执行的操作

function debounce(func, wait) {

var timeId = null; // 存储setTimeout 的返回值

return function(...args){

if(timeId){

// 进入此if 说明目前在等待当中，但是由于又调用了，所以需要重新开始计时

clearTimeout(timeId);

}

timeId = setTimeout(() => {

func(...args)

}, wait);

}

}

节流

⭐节流就是在一定的时间内只触发一次事件梳理函数，无论事件触发的频率多高，只要在指定的时间内触发一次处理即可

// 1. 时间戳的形式

/\*\*

\* func : 要执行的函数

\* wait : 间隔的时间

\*/

function throttle(func,wait){

var pre = 0; // 上一次时间,开始为0

return function(...args){

var now = new Date(); // 获取最新的时间戳

if(now - pre > wait){

// 进入此if 说明时间已经到了，等待时间那么长，可以执行一次

func(...args);

// 更新上一次的时间

pre = now;

}

}

}

⭐使用定时器实现

function throttle(func,wait){

var timeId = null;

return function(...args){

if(!timeId){

func(...args);

timeId = setTimeout(function(){

timeId = null;

},wait)

}

}

}

移动端rem适配

// 方案二：优化方案 可以非常轻松的实现转换

(

function (doc, win, DesignWidth) {

// 获取html

var html = doc.documentElement;

// 计算rem

function refreshRem() {

var width = html.clientWidth; // 获取CSS像素(设备的宽度)

if (width >= DesignWidth) {

// 说明设备宽度都已经大于设计稿,测量出来是多少就是多少。

// 这里设置100px相当于1rm = 100px为了保持不同屏幕展示的效果都一样

html.style.fontSize = '100px';

} else {

// 计算比例

// 拿iPhone6 (375px) 375px / 750px = 0.5 => 1 / DPR

// 相当于现在每一列的宽度为0.5px 相当于375px / 0.5 = 750列

// 但是现在我们设置fontSize 为 0.5px 浏览器不给设置这么小的数值，因此 \* 100

// 每一列的宽度就变为了 50px, 375/50 = 7.5列

// 假设设计稿量出来的是375px --> 设备像素 187.5px / 50 = 3.75列(所占列数)

// 假设里量出来100px --> 1列 100px -> 50px/50px -> 1 -> 1rem

// 假设量出来是1180px --> 1180px -> 590/50px -> 11.8列 ->11.8rem

html.style.fontSize = 100 \* (width / DesignWidth) + 'px';

}

}

doc.addEventListener('DOMContentLoaded', refreshRem);

}

)(document, window, 750);

// 这里的750是设计稿的宽度