# class操作函数

function addClass(node,className){

var reg=new RegExp("\\b"+className+"\\b");

if(!reg.test(node.className)){

node.className +=(" "+className);

}

}

function removeClass(node,className){

if(node.className){

var reg=new RegExp("\\b"+className+"\\b");

var classes = node.className;

node.className=classes.replace(reg,"");

if(/^\s\*$/g.test(node.className)){

node.removeAttribute("class");

}

}else{

node.removeAttribute("class");

}

}

# 2D变换函数

(function(w){

w.transformCss = function(node,name,value){

if(!node.obj){

node.obj = {}

}

if(arguments.length > 2){

node.obj[name] = value;

var result = '';

for(var k in node.obj){

switch(k){

case 'translateX':

case 'translateY':

case 'translateZ':

case 'translate':

result += k+'('+ node.obj[k] +'px )';

break;

case 'scale':

case 'scaleX':

case 'scaleY':

result += k+'('+ node.obj[k] +' )';

break;

case 'rotateX':

case 'rotateY':

case 'rotateZ':

case 'rotate':

case 'skewX':

case 'skewY':

case 'skew':

result += k+'('+ node.obj[k] +'deg )';

break;

}

}

node.style.transform = result;

}else{

var result = ''

if(node.obj[name] == undefined){

if(name == 'scale' || name == 'scaleX' || name == 'scaleY'){

result = 1;

}else{

result = 0;

}

}else{

result = node.obj[name];

}

return result

}

};

})(window);

# 移动端滑屏模板

(function(w) {

/\*\*

\*

\* @param {外部包裹器} wrap

\* @param {回调函数} callback

\*/

w.contentDrag = function(wrap, callback) {

//获取内部移动的区域

var list = wrap.firstElementChild;

transformCss(list, 'translateZ', 0.1);

var startY = 0;

var startX = 0;

var eleY = 0;

var isFirst = true;

var s1 = 0;

var s2 = 0;

var t1 = 0;

var t2 = 0;

var tween = {

Linear: function(t, b, c, d) {

return c \* t / d + b;

},

easeOut: function(t, b, c, d, s) {

if(s == undefined) s = 3;

return c \* ((t = t / d - 1) \* t \* ((s + 1) \* t + s) + 1) + b;

}

};

var timer = null;

wrap.addEventListener('touchstart', function(event) {

var touch = event.changedTouches[0];

list.style.transition = 'none';

eleY = transformCss(list, 'translateY');

startY = touch.clientY;

startX = touch.clientX;

s1 = eleY;

t1 = new Date().getTime(); //毫秒

clearInterval(timer);

isFirst = true;

if(callback && typeof callback['start'] == 'function') {

callback['start']();

}

});

wrap.addEventListener('touchmove', function(event) {

var touch = event.changedTouches[0];

if(!isFirst) {

return;

}

var endY = touch.clientY;

var endX = touch.clientX;

var disY = endY - startY;

var disX = endX - startX;

if(Math.abs(disX) > Math.abs(disY)) {

isFirst = false;

return;

}

var lastY = eleY + disY;

//当滑到边界的时候，会出现橡皮筋效果

if(lastY > 0) {

//每一次的移动 lasY不断增大，但是系数再不断减小。也就是每一次拉的时候，走的实际距离会越小

var scale = 0.6 - lastY / (3 \* wrap.clientHeight);

lastY = lastY \* scale;

} else if(lastY < wrap.clientHeight - list.offsetHeight) {

var temp = Math.abs(lastY) - Math.abs(wrap.clientHeight - list.offsetHeight);

var scale = 0.6 - temp / (3 \* wrap.clientHeight);

temp = temp \* scale;

lastY = wrap.clientHeight - list.offsetHeight - temp;

}

transformCss(list, 'translateY', lastY);

if(callback && typeof callback['move'] == 'function') {

callback['move']();

}

});

wrap.addEventListener('touchend', function() {

//手指离开后会有一个加速的效果/求一个速度

s2 = transformCss(list, 'translateY');

t2 = new Date().getTime();

var speed = (s2 - s1) / (t2 - t1);

var lastY = transformCss(list, 'translateY');

lastY = lastY + speed \* 100;

var timeAll = 1;

//我们得把加速的距离进行分段移动，使用tween算法进行实现

var type = 'Linear';

if(lastY > 0) {

lastY = 0;

type = 'easeOut';

} else if(lastY < wrap.clientHeight - list.offsetHeight) {

lastY = wrap.clientHeight - list.offsetHeight;

type = 'easeOut';

}

TweenMove(lastY, type, timeAll);

if(callback && typeof callback['endTrue'] == 'function') {

callback['endTrue'](timer);

}

function TweenMove(lastY, type, timeAll) {

/\*  
 t:(从第几次开始)；  
 b:(初始值)；  
 c:(变化量)；  
 d:(变化的次数)。

\*/

var t = 0;

var b = transformCss(list, 'translateY');

var c = lastY - b;

var d = timeAll / 0.02;

timer = setInterval(function() {

t++;

if(t > d) {

clearInterval(timer);

if(callback && typeof callback['end'] == 'function') {

callback['end']();

}

} else {

lastY = tween[type](t, b, c, d);

transformCss(list, 'translateY', lastY);

if(callback && typeof callback['move'] == 'function') {

callback['move']();

}

}

}, 20)

}

})

}

})(window);

# 鼠标滚轮事件

*Scroll*();  
**function** *Scroll*() {  
 *//ie/chrome*

*//系统调用的函数会传入event参数* **document**.onmousewheel = **function** (event) {

*//添加定时器优化滚轮事件,否则会滚动太快*

*//每一次滚动之前先将上次滚动产生的定时器清除掉*  
 clearTimeout(timer);  
 timer = setTimeout(**function** () {  
 *//这里的event不能直接使用,必须从上面传进来,因为这里不是系统调 //用的函数*

scrollMove(event); }, 200)  
 };  
 *//firefox* **if** (**document**.addEventListener) {  
 **document**.addEventListener(**'DOMMouseScroll'**, **function** (event) {  
 clearTimeout(timer);  
 timer = setTimeout(**function** () {  
 *scrollMove*(event);  
 }, 200)  
 })  
 }  
  
 **function** *scrollMove*(event) {  
 *//兼容性处理*

event = event || **window**.**event**;  
  
 **var** flag = **''**;  
 **if** (event.**wheelDelta**) {  
 *//ie/chrome* **if** (event.**wheelDelta** > 0) {  
 *//上* flag = **'up'**;  
 } **else** {  
 *//下* flag = **'down'** }  
 } **else if** (event.**detail**) {  
 *//firefox* **if** (event.**detail** < 0) {  
 *//上* flag = **'up'**;  
 } **else** {  
 *//下* flag = **'down'** }  
 }  
  
 **switch** (flag) {  
 **case 'up'**:  
 *//鼠标向上滚动* **if** (index > 0) {  
 index--;  
 }  
 *contentChange*(index);  
 **break**;  
 **case 'down'**:  
 *//鼠标向下滚动* **if** (index < contentItem.**length** - 1) {  
 index++;  
 }  
 *contentChange*(index);  
 **break**;  
 }  
 *//取消默认行为* event.preventDefault && event.preventDefault();  
 **return false**;  
 }  
}

# 自定义栅格系统

**.container {  
 margin-right: auto;  
 margin-left: auto;  
 padding-left: 15px;  
 padding-right: 15px;  
}  
.container:before,  
.container:after {  
 content: " ";  
 display: table;  
}  
.container:after {  
 clear: both;  
}  
@media (min-width: 768px) {  
 .container {  
 width: 750px;  
 }  
}  
@media (min-width: 992px) {  
 .container {  
 width: 970px;  
 }  
}  
@media (min-width: 1200px) {  
 .container {  
 width: 1170px;  
 }  
}  
.container-fluid {  
 margin-right: auto;  
 margin-left: auto;  
 padding-left: 15px;  
 padding-right: 15px;  
}  
.container-fluid:before,  
.container-fluid:after {  
 content: " ";  
 display: table;  
}  
.container-fluid:after {  
 clear: both;  
}  
.row {  
 margin-left: -15px;  
 margin-right: -15px;  
}  
.row:before,  
.row:after {  
 content: " ";  
 display: table;  
}  
.row:after {  
 clear: both;  
}  
.col-xs-1,  
.col-sm-1,  
.col-md-1,  
.col-lg-1,  
.col-xs-2,  
.col-sm-2,  
.col-md-2,  
.col-lg-2,  
.col-xs-3,  
.col-sm-3,  
.col-md-3,  
.col-lg-3,  
.col-xs-4,  
.col-sm-4,  
.col-md-4,  
.col-lg-4,  
.col-xs-5,  
.col-sm-5,  
.col-md-5,  
.col-lg-5,  
.col-xs-6,  
.col-sm-6,  
.col-md-6,  
.col-lg-6,  
.col-xs-7,  
.col-sm-7,  
.col-md-7,  
.col-lg-7,  
.col-xs-8,  
.col-sm-8,  
.col-md-8,  
.col-lg-8,  
.col-xs-9,  
.col-sm-9,  
.col-md-9,  
.col-lg-9,  
.col-xs-10,  
.col-sm-10,  
.col-md-10,  
.col-lg-10,  
.col-xs-11,  
.col-sm-11,  
.col-md-11,  
.col-lg-11,  
.col-xs-12,  
.col-sm-12,  
.col-md-12,  
.col-lg-12 {  
 position: relative;  
 min-height: 1px;  
 padding-left: 15px;  
 padding-right: 15px;  
}  
.col-xs-1,  
.col-xs-2,  
.col-xs-3,  
.col-xs-4,  
.col-xs-5,  
.col-xs-6,  
.col-xs-7,  
.col-xs-8,  
.col-xs-9,  
.col-xs-10,  
.col-xs-11,  
.col-xs-12 {  
 float: left;  
}  
.col-xs-12 {  
 width: 100%;  
}  
.col-xs-11 {  
 width: 91.66666667%;  
}  
.col-xs-10 {  
 width: 83.33333333%;  
}  
.col-xs-9 {  
 width: 75%;  
}  
.col-xs-8 {  
 width: 66.66666667%;  
}  
.col-xs-7 {  
 width: 58.33333333%;  
}  
.col-xs-6 {  
 width: 50%;  
}  
.col-xs-5 {  
 width: 41.66666667%;  
}  
.col-xs-4 {  
 width: 33.33333333%;  
}  
.col-xs-3 {  
 width: 25%;  
}  
.col-xs-2 {  
 width: 16.66666667%;  
}  
.col-xs-1 {  
 width: 8.33333333%;  
}  
.col-xs-pull-12 {  
 right: 100%;  
}  
.col-xs-pull-11 {  
 right: 91.66666667%;  
}  
.col-xs-pull-10 {  
 right: 83.33333333%;  
}  
.col-xs-pull-9 {  
 right: 75%;  
}  
.col-xs-pull-8 {  
 right: 66.66666667%;  
}  
.col-xs-pull-7 {  
 right: 58.33333333%;  
}  
.col-xs-pull-6 {  
 right: 50%;  
}  
.col-xs-pull-5 {  
 right: 41.66666667%;  
}  
.col-xs-pull-4 {  
 right: 33.33333333%;  
}  
.col-xs-pull-3 {  
 right: 25%;  
}  
.col-xs-pull-2 {  
 right: 16.66666667%;  
}  
.col-xs-pull-1 {  
 right: 8.33333333%;  
}  
.col-xs-pull-0 {  
 right: auto;  
}  
.col-xs-push-12 {  
 left: 100%;  
}  
.col-xs-push-11 {  
 left: 91.66666667%;  
}  
.col-xs-push-10 {  
 left: 83.33333333%;  
}  
.col-xs-push-9 {  
 left: 75%;  
}  
.col-xs-push-8 {  
 left: 66.66666667%;  
}  
.col-xs-push-7 {  
 left: 58.33333333%;  
}  
.col-xs-push-6 {  
 left: 50%;  
}  
.col-xs-push-5 {  
 left: 41.66666667%;  
}  
.col-xs-push-4 {  
 left: 33.33333333%;  
}  
.col-xs-push-3 {  
 left: 25%;  
}  
.col-xs-push-2 {  
 left: 16.66666667%;  
}  
.col-xs-push-1 {  
 left: 8.33333333%;  
}  
.col-xs-push-0 {  
 left: auto;  
}  
.col-xs-offset-12 {  
 margin-left: 100%;  
}  
.col-xs-offset-11 {  
 margin-left: 91.66666667%;  
}  
.col-xs-offset-10 {  
 margin-left: 83.33333333%;  
}  
.col-xs-offset-9 {  
 margin-left: 75%;  
}  
.col-xs-offset-8 {  
 margin-left: 66.66666667%;  
}  
.col-xs-offset-7 {  
 margin-left: 58.33333333%;  
}  
.col-xs-offset-6 {  
 margin-left: 50%;  
}  
.col-xs-offset-5 {  
 margin-left: 41.66666667%;  
}  
.col-xs-offset-4 {  
 margin-left: 33.33333333%;  
}  
.col-xs-offset-3 {  
 margin-left: 25%;  
}  
.col-xs-offset-2 {  
 margin-left: 16.66666667%;  
}  
.col-xs-offset-1 {  
 margin-left: 8.33333333%;  
}  
.col-xs-offset-0 {  
 margin-left: 0%;  
}  
@media (min-width: 768px) {  
 .col-sm-1,  
 .col-sm-2,  
 .col-sm-3,  
 .col-sm-4,  
 .col-sm-5,  
 .col-sm-6,  
 .col-sm-7,  
 .col-sm-8,  
 .col-sm-9,  
 .col-sm-10,  
 .col-sm-11,  
 .col-sm-12 {  
 float: left;  
 }  
 .col-sm-12 {  
 width: 100%;  
 }  
 .col-sm-11 {  
 width: 91.66666667%;  
 }  
 .col-sm-10 {  
 width: 83.33333333%;  
 }  
 .col-sm-9 {  
 width: 75%;  
 }  
 .col-sm-8 {  
 width: 66.66666667%;  
 }  
 .col-sm-7 {  
 width: 58.33333333%;  
 }  
 .col-sm-6 {  
 width: 50%;  
 }  
 .col-sm-5 {  
 width: 41.66666667%;  
 }  
 .col-sm-4 {  
 width: 33.33333333%;  
 }  
 .col-sm-3 {  
 width: 25%;  
 }  
 .col-sm-2 {  
 width: 16.66666667%;  
 }  
 .col-sm-1 {  
 width: 8.33333333%;  
 }  
 .col-sm-pull-12 {  
 right: 100%;  
 }  
 .col-sm-pull-11 {  
 right: 91.66666667%;  
 }  
 .col-sm-pull-10 {  
 right: 83.33333333%;  
 }  
 .col-sm-pull-9 {  
 right: 75%;  
 }  
 .col-sm-pull-8 {  
 right: 66.66666667%;  
 }  
 .col-sm-pull-7 {  
 right: 58.33333333%;  
 }  
 .col-sm-pull-6 {  
 right: 50%;  
 }  
 .col-sm-pull-5 {  
 right: 41.66666667%;  
 }  
 .col-sm-pull-4 {  
 right: 33.33333333%;  
 }  
 .col-sm-pull-3 {  
 right: 25%;  
 }  
 .col-sm-pull-2 {  
 right: 16.66666667%;  
 }  
 .col-sm-pull-1 {  
 right: 8.33333333%;  
 }  
 .col-sm-pull-0 {  
 right: auto;  
 }  
 .col-sm-push-12 {  
 left: 100%;  
 }  
 .col-sm-push-11 {  
 left: 91.66666667%;  
 }  
 .col-sm-push-10 {  
 left: 83.33333333%;  
 }  
 .col-sm-push-9 {  
 left: 75%;  
 }  
 .col-sm-push-8 {  
 left: 66.66666667%;  
 }  
 .col-sm-push-7 {  
 left: 58.33333333%;  
 }  
 .col-sm-push-6 {  
 left: 50%;  
 }  
 .col-sm-push-5 {  
 left: 41.66666667%;  
 }  
 .col-sm-push-4 {  
 left: 33.33333333%;  
 }  
 .col-sm-push-3 {  
 left: 25%;  
 }  
 .col-sm-push-2 {  
 left: 16.66666667%;  
 }  
 .col-sm-push-1 {  
 left: 8.33333333%;  
 }  
 .col-sm-push-0 {  
 left: auto;  
 }  
 .col-sm-offset-12 {  
 margin-left: 100%;  
 }  
 .col-sm-offset-11 {  
 margin-left: 91.66666667%;  
 }  
 .col-sm-offset-10 {  
 margin-left: 83.33333333%;  
 }  
 .col-sm-offset-9 {  
 margin-left: 75%;  
 }  
 .col-sm-offset-8 {  
 margin-left: 66.66666667%;  
 }  
 .col-sm-offset-7 {  
 margin-left: 58.33333333%;  
 }  
 .col-sm-offset-6 {  
 margin-left: 50%;  
 }  
 .col-sm-offset-5 {  
 margin-left: 41.66666667%;  
 }  
 .col-sm-offset-4 {  
 margin-left: 33.33333333%;  
 }  
 .col-sm-offset-3 {  
 margin-left: 25%;  
 }  
 .col-sm-offset-2 {  
 margin-left: 16.66666667%;  
 }  
 .col-sm-offset-1 {  
 margin-left: 8.33333333%;  
 }  
 .col-sm-offset-0 {  
 margin-left: 0%;  
 }  
}  
@media (min-width: 992px) {  
 .col-md-1,  
 .col-md-2,  
 .col-md-3,  
 .col-md-4,  
 .col-md-5,  
 .col-md-6,  
 .col-md-7,  
 .col-md-8,  
 .col-md-9,  
 .col-md-10,  
 .col-md-11,  
 .col-md-12 {  
 float: left;  
 }  
 .col-md-12 {  
 width: 100%;  
 }  
 .col-md-11 {  
 width: 91.66666667%;  
 }  
 .col-md-10 {  
 width: 83.33333333%;  
 }  
 .col-md-9 {  
 width: 75%;  
 }  
 .col-md-8 {  
 width: 66.66666667%;  
 }  
 .col-md-7 {  
 width: 58.33333333%;  
 }  
 .col-md-6 {  
 width: 50%;  
 }  
 .col-md-5 {  
 width: 41.66666667%;  
 }  
 .col-md-4 {  
 width: 33.33333333%;  
 }  
 .col-md-3 {  
 width: 25%;  
 }  
 .col-md-2 {  
 width: 16.66666667%;  
 }  
 .col-md-1 {  
 width: 8.33333333%;  
 }  
 .col-md-pull-12 {  
 right: 100%;  
 }  
 .col-md-pull-11 {  
 right: 91.66666667%;  
 }  
 .col-md-pull-10 {  
 right: 83.33333333%;  
 }  
 .col-md-pull-9 {  
 right: 75%;  
 }  
 .col-md-pull-8 {  
 right: 66.66666667%;  
 }  
 .col-md-pull-7 {  
 right: 58.33333333%;  
 }  
 .col-md-pull-6 {  
 right: 50%;  
 }  
 .col-md-pull-5 {  
 right: 41.66666667%;  
 }  
 .col-md-pull-4 {  
 right: 33.33333333%;  
 }  
 .col-md-pull-3 {  
 right: 25%;  
 }  
 .col-md-pull-2 {  
 right: 16.66666667%;  
 }  
 .col-md-pull-1 {  
 right: 8.33333333%;  
 }  
 .col-md-pull-0 {  
 right: auto;  
 }  
 .col-md-push-12 {  
 left: 100%;  
 }  
 .col-md-push-11 {  
 left: 91.66666667%;  
 }  
 .col-md-push-10 {  
 left: 83.33333333%;  
 }  
 .col-md-push-9 {  
 left: 75%;  
 }  
 .col-md-push-8 {  
 left: 66.66666667%;  
 }  
 .col-md-push-7 {  
 left: 58.33333333%;  
 }  
 .col-md-push-6 {  
 left: 50%;  
 }  
 .col-md-push-5 {  
 left: 41.66666667%;  
 }  
 .col-md-push-4 {  
 left: 33.33333333%;  
 }  
 .col-md-push-3 {  
 left: 25%;  
 }  
 .col-md-push-2 {  
 left: 16.66666667%;  
 }  
 .col-md-push-1 {  
 left: 8.33333333%;  
 }  
 .col-md-push-0 {  
 left: auto;  
 }  
 .col-md-offset-12 {  
 margin-left: 100%;  
 }  
 .col-md-offset-11 {  
 margin-left: 91.66666667%;  
 }  
 .col-md-offset-10 {  
 margin-left: 83.33333333%;  
 }  
 .col-md-offset-9 {  
 margin-left: 75%;  
 }  
 .col-md-offset-8 {  
 margin-left: 66.66666667%;  
 }  
 .col-md-offset-7 {  
 margin-left: 58.33333333%;  
 }  
 .col-md-offset-6 {  
 margin-left: 50%;  
 }  
 .col-md-offset-5 {  
 margin-left: 41.66666667%;  
 }  
 .col-md-offset-4 {  
 margin-left: 33.33333333%;  
 }  
 .col-md-offset-3 {  
 margin-left: 25%;  
 }  
 .col-md-offset-2 {  
 margin-left: 16.66666667%;  
 }  
 .col-md-offset-1 {  
 margin-left: 8.33333333%;  
 }  
 .col-md-offset-0 {  
 margin-left: 0%;  
 }  
}  
@media (min-width: 1200px) {  
 .col-lg-1,  
 .col-lg-2,  
 .col-lg-3,  
 .col-lg-4,  
 .col-lg-5,  
 .col-lg-6,  
 .col-lg-7,  
 .col-lg-8,  
 .col-lg-9,  
 .col-lg-10,  
 .col-lg-11,  
 .col-lg-12 {  
 float: left;  
 }  
 .col-lg-12 {  
 width: 100%;  
 }  
 .col-lg-11 {  
 width: 91.66666667%;  
 }  
 .col-lg-10 {  
 width: 83.33333333%;  
 }  
 .col-lg-9 {  
 width: 75%;  
 }  
 .col-lg-8 {  
 width: 66.66666667%;  
 }  
 .col-lg-7 {  
 width: 58.33333333%;  
 }  
 .col-lg-6 {  
 width: 50%;  
 }  
 .col-lg-5 {  
 width: 41.66666667%;  
 }  
 .col-lg-4 {  
 width: 33.33333333%;  
 }  
 .col-lg-3 {  
 width: 25%;  
 }  
 .col-lg-2 {  
 width: 16.66666667%;  
 }  
 .col-lg-1 {  
 width: 8.33333333%;  
 }  
 .col-lg-pull-12 {  
 right: 100%;  
 }  
 .col-lg-pull-11 {  
 right: 91.66666667%;  
 }  
 .col-lg-pull-10 {  
 right: 83.33333333%;  
 }  
 .col-lg-pull-9 {  
 right: 75%;  
 }  
 .col-lg-pull-8 {  
 right: 66.66666667%;  
 }  
 .col-lg-pull-7 {  
 right: 58.33333333%;  
 }  
 .col-lg-pull-6 {  
 right: 50%;  
 }  
 .col-lg-pull-5 {  
 right: 41.66666667%;  
 }  
 .col-lg-pull-4 {  
 right: 33.33333333%;  
 }  
 .col-lg-pull-3 {  
 right: 25%;  
 }  
 .col-lg-pull-2 {  
 right: 16.66666667%;  
 }  
 .col-lg-pull-1 {  
 right: 8.33333333%;  
 }  
 .col-lg-pull-0 {  
 right: auto;  
 }  
 .col-lg-push-12 {  
 left: 100%;  
 }  
 .col-lg-push-11 {  
 left: 91.66666667%;  
 }  
 .col-lg-push-10 {  
 left: 83.33333333%;  
 }  
 .col-lg-push-9 {  
 left: 75%;  
 }  
 .col-lg-push-8 {  
 left: 66.66666667%;  
 }  
 .col-lg-push-7 {  
 left: 58.33333333%;  
 }  
 .col-lg-push-6 {  
 left: 50%;  
 }  
 .col-lg-push-5 {  
 left: 41.66666667%;  
 }  
 .col-lg-push-4 {  
 left: 33.33333333%;  
 }  
 .col-lg-push-3 {  
 left: 25%;  
 }  
 .col-lg-push-2 {  
 left: 16.66666667%;  
 }  
 .col-lg-push-1 {  
 left: 8.33333333%;  
 }  
 .col-lg-push-0 {  
 left: auto;  
 }  
 .col-lg-offset-12 {  
 margin-left: 100%;  
 }  
 .col-lg-offset-11 {  
 margin-left: 91.66666667%;  
 }  
 .col-lg-offset-10 {  
 margin-left: 83.33333333%;  
 }  
 .col-lg-offset-9 {  
 margin-left: 75%;  
 }  
 .col-lg-offset-8 {  
 margin-left: 66.66666667%;  
 }  
 .col-lg-offset-7 {  
 margin-left: 58.33333333%;  
 }  
 .col-lg-offset-6 {  
 margin-left: 50%;  
 }  
 .col-lg-offset-5 {  
 margin-left: 41.66666667%;  
 }  
 .col-lg-offset-4 {  
 margin-left: 33.33333333%;  
 }  
 .col-lg-offset-3 {  
 margin-left: 25%;  
 }  
 .col-lg-offset-2 {  
 margin-left: 16.66666667%;  
 }  
 .col-lg-offset-1 {  
 margin-left: 8.33333333%;  
 }  
 .col-lg-offset-0 {  
 margin-left: 0%;  
 }  
}  
\* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
}**

# 函数节流

***函数节流：让函数在单位时间内只调用一次，第一次生效，后面的不生效***

**let *num* = 0;  
*// 监听/绑定滚轮事件*document.onscroll = *throttle*(function (e) {  
 *console*.log(this);  
 *console*.log(e);  
 *console*.log('滚轮事件触发了~', ++*num*);  
}, 50); */\*\*  
 \* 节流函数  
 \* @param fn 针对哪个函数做节流  
 \* @param time 单位时间  
 \*/*function *throttle*(fn, time) {  
 let lastTime = 0;*// 上一次触发函数的时间*   
 *// 返回一个新函数：最终会作为事件的回调函数* return function (e) {const nowTime = Date.now();*// 获取当前的时间  
 // 缓存this --> 这个this是事件回调函数的this=>document* const \_self = this;  
 *// 判断时间差是否大于time，如果大于就执行，如果小与就不执行* if (nowTime - lastTime > time) {  
 fn.apply(\_self, arguments);  
 *// 更新时间：将上一次时间置为当前触发函数的时间* lastTime = nowTime;  
 }  
 }  
}**

# 函数防抖

***函数防抖：让函数在单位时间内只调用一次，最后一次生效，前面的不生效*function *debounce*(fn, time) {  
 let timeId = null;  
 return function () {  
 *// 清楚上一次调用的定时器  
 clearTimeout*(timeId);  
 const \_self = this;  
 const args = arguments;  
 timeId = *setTimeout*(function () {  
 fn.apply(\_self, args);  
 }, time);  
 }  
}  
  
const *scroll* = function (e) {  
 *console*.log('函数调用了~');  
}  
document.onscroll = *debounce*(*scroll*, 100);**

# 图片懒加载

<**img src="./imgs/loading.gif" data-src="./imgs/1.jpg" class="lazy-load" alt="img"**>  
<**img src="./imgs/loading.gif" data-src="./imgs/2.jpg" class="lazy-load" alt="img"**>  
<**img src="./imgs/loading.gif" data-src="./imgs/3.jpg" class="lazy-load" alt="img"**>  
<**img src="./imgs/loading.gif" data-src="./imgs/4.jpg" class="lazy-load" alt="img"**>  
<**img src="./imgs/loading.gif" data-src="./imgs/5.jpg" class="lazy-load" alt="img"**>  
<**img src="./imgs/loading.gif" data-src="./imgs/6.jpg" class="lazy-load" alt="img"**>

......

***// 视口的高度*const *viewHeight* = document.documentElement.clientHeight;  
*// 一上来需要手动触发lazyLoad: 目的为了第一次加载的时候出现在可视区域内的图片能够加载到***

***lazyLoad*();**

**function *lazyLoad*() {  
 const imgs = document.querySelectorAll('.lazy-load'); *// dom2* imgs.forEach((img) => {  
 *// 当前图片距离视口的top值* const { top } = img.getBoundingClientRect();  
 *// 判断图片是否出现在可视区域内* if (top < *viewHeight*) {  
 *// 图片出现在可视区域内，加载图片真实地址  
 // 通常来讲 data-xxx 使用用户自定义属性（用来保存数据，浏览器不能识别）。  
 // 读取 data-xxx 属性的值，需要通过 dom.dataset.xxx 读取对应自定义属性的值* const src = img.dataset.src;  
 *// 将图片src属性修改为图片真实地址* img.src = src;  
 *// 删除dom元素上的属性* img.removeAttribute('class');  
 img.removeAttribute('data-src');  
 }  
 })  
}**

***// 给document绑定滚轮事件*document.onscroll = *lazyLoad*;**

# webp格式图片的处理

**<img src="./img/1.jpg\_\_.webp" alt="img">**

**const *img* = document.querySelector('img');  
*// 图片加载成功触发的事件(可不写)  
img*.onload = function () {  
 console.log('图片加载成功~');  
}  
*// 图片加载失败触发的事件  
img*.onerror = function () {  
 console.log('图片加载失败，当前浏览器不支持webp格式图片');  
 *img*.src = *img*.src.slice(0, -7);  
}**

# 自定义数组flat方法

**const *arr* = [1,[2],[[3],[4]],[[[5]]]];  
function *flat*(arr) {  
 let result = [];  
 arr.forEach((item)=>{  
 if(Array.isArray(item)){  
 result = result.concat(*flat*(item));  
 } else {  
 result.push(item)  
 }  
 });  
 return result;  
}  
*console*.log(*flat*(*arr*));  
*//添加到原型对象上*Array.prototype.flat = function () {  
 let result = [];  
 this.forEach((item)=>{  
 if(Array.isArray(item)){  
 result = result.concat(item.flat());  
 } else {  
 result.push(item)  
 }  
 });  
 return result;  
};  
*console*.log(*arr*.flat());**

# call、apply、bind

***相同点：三者都能改变函数调用时this指向。  
 不同点:  
 call 第一个是要改变this指向的对象，后面参数是要调用函数传入的参数（以逗号隔开）  
 apply 第一个是要改变this指向的对象，第二个参数是一个数组（数组里面有函数调用的参数）  
 bind 传参和call一样。 不会执行函数，返回一个this改变后的函数***

**const *obj1* = {  
 fn(x, y) {  
 console.log(x, y);  
 console.log(this);  
  
 return 789;  
 }  
};  
  
const *obj2* = {  
 a: 123  
};  
*/\*  
obj1.fn.call(obj2, 123, 456); // obj2  
obj1.fn(); // obj1  
obj1.fn.apply(obj2, [123, 456]);  
obj1.fn.bind(obj2, 123, 456)();  
\*/*Function.prototype.call1 = function (context, ...args) {  
 *// ...args 代表除第一个参数以外，剩下的参数  
 // context 就是要修改this指向的对象* context.\_fn = this;  
 const result = context.\_fn(...args); *// \_fn函数的this就指向context  
 // 删除对象上多余的属性* delete context.\_fn;  
 *// console.log(this); // obj1.fn --> obj1.fn.call1() 所以 call1函数的this就指向调用方法的对象  
 // this(); // window  
 // 将fn函数调用后的返回值，作为call方法的返回值* return result;  
};  
  
Function.prototype.apply1 = function (context, args) {  
 context.\_fn = this;  
 const result = context.\_fn(...args);  
 delete context.\_fn;  
 return result;  
};  
  
Function.prototype.bind1 = function (context, ...args) {  
 *// console.log(this); // 指向 obj1.fn* const \_self = this;  
 return function () {  
 return \_self.apply1(context, args);  
 };  
};  
  
const *fn* = *obj1*.fn.bind1(*obj2*, 123, 456);  
*fn*();**

# 快速排序

**function *quickSort*(arr) {  
 *// 判断数组的长度，如果小于等于1，就直接退出* if (arr.length <= 1) return arr;  
 *// 找到中间的基准值* const index = *Math*.floor(arr.length / 2);  
 *// 取出基准值对应的元素: 从数组移除元素* const result = arr.splice(index, 1)[0];  
 *// 比基准值小的放左边 比基准值大的放右边* const leftArr = [];  
 const rightArr = [];  
  
 for (let i = 0; i < arr.length; i++) {  
 let item = arr[i];  
 if (item < result) {  
 *// 小于的元素* leftArr.push(item);  
 } else {  
 *// 大于等于的元素* rightArr.push(item);  
 }  
 }  
return *quickSort*(leftArr).concat([result], *quickSort*(rightArr));  
}  
  
const *arr* = [40, 2, 5, 87, 36, 92, 5, 8 , 39, 34, 55];  
const *result* = *quickSort*(*arr*);  
console.log(*result*);**

# 数组去重

**const *arr* = [];  
for (var *i* = 0; *i* < 100000; *i*++) {  
 *arr*.push(*Math*.floor(*Math*.random() \* 10000));  
}  
  
Array.prototype.unique = function () {  
 const newArr = [];  
 for (let i = 0; i < this.length; i++) {  
 let item1 = this[i];  
 *// 定义一个重复的标识：默认是没有重复的* let isRepeat = false;  
 for (let j = 0; j < newArr.length; j++) {  
 let item2 = newArr[j];  
 if (item1 === item2) {  
 *// 说明新数组中存在item1这个元素，就不添加进行* isRepeat = true;  
 *// 新数组中已经存在原来的这个数，需要退出当前循环* break;  
 }  
 }  
 *// 只有重复的标识为false时，才添加当前元素* if (!isRepeat) {  
 newArr.push(item1);  
 }  
 }  
  
 return newArr;  
};  
  
Array.prototype.unique = function () {  
 const obj = {};  
 const newArr = [];  
 for (let i = 0; i < this.length; i++) {  
 const item = this[i];  
 *// 对象的同名属性，只会保留一个* obj[item] = 0;  
 }  
 */\*for (let key in obj) {  
 newArr.push(+key);  
 }\*/  
 // return newArr;* return Object.keys(obj);  
};  
  
Array.prototype.unique = function () {  
 return [...new Set(this)];  
};  
  
const *startTime* = Date.now();  
const *newArr* = *arr*.unique();  
const *endTime* = Date.now();  
console.log(*endTime* - *startTime*); *// 1400 ms 376ms 8ms  
// console.log(newArr);***