**实现元素水平居中**

**分页容器**

**html**

<div class="pagination">

<ul>

<li><a href="#">Prev</a></li>

<li><a href="#">1</a></li>

<li><a href="#">2</a></li>

<li><a href="#">3</a></li>

<li><a href="#">4</a></li>

<li><a href="#">5</a></li>

<li><a href="#">Next</a></li>

</ul>

</div>

**css**

.pagination li{

line-height: 25px;

list-style: none;

display: inline;

float: left;

}

.pagination a{

text-decoration: none;

display: block;

color: #000;

padding: 0 10px;

border-radius: 2px;

border: 1px solid #F2F2F2;

background: #333;

}

.pagination a:hover{

background: orange;

}

**1、方法一：margin和width实现**

这个是最常见的方案：在容器上定义一个固定的宽度，然后配合margin左右的值为auto。

.pagination{

width**:** 340px;

margin: 0 auto;

}

效果实现了，扩展性不强，因为宽度无法确定，也就无法确定容器宽度。

**优点**：实现简单，浏览器兼容性强

**缺点**：扩展性差，无法自适应未知情况。

**2、方法二：inline-block和父元素text-align**

仅inline-block属性是无法让元素水平居中，他的关键之处要在元素的父容器中设置text-align的属性为“center”，这样才能达到效果：

.pagination {

**text-align: center;**

font-size: 0;

letter-spacing: -4px;

word-spacing: -4px;

}

.pagination li {

line-height: 25px;

margin: 0 5px;

**display: inline-block;**

\*display: inline;

zoom:1;

letter-spacing: normal;

word-spacing: normal;

font-size: 12px;

}

inline-block解决了水平居中的问题，但是又有一个新的问题，就是分页项与分页项由回车符带来的空白间距，所以需要解决inline-block带来的间距。详细的解决方法可以阅读《[如何解决inline-block元素的空白间距](http://www.w3cplus.com/css/fighting-the-space-between-inline-block-elements)》一文。

**优点**：简单易懂，扩展性强。

**缺点**：需要额外处理inline-block的浏览器兼容性问题。

**3、方法三：浮动实现水平居中**

感到很意外，元素都浮动了，他还能水平居中？大家都知道，浮动要么靠左、要么靠右，还真少见有居中的。其实略加处理就有了。

.pagination{

margin-top: 50px;

**float: left;**

**width:** 100%;

overflow: hidden;

position: relative;

}

.pagination ul{

clear: left;

float: left;

position: relative;

left: 50%;/\*整个分页向右边移动宽度的50%\*/

text-align: center;

}

.pagination li{

line-height: 25px;

list-style: none;

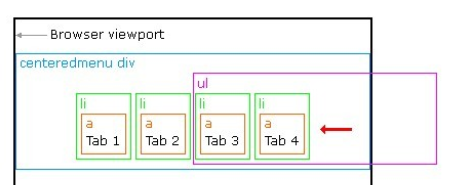
display: inline;

float: left;

**position: relative;**

**right:** 50%;

}



在这里是通过“position:relative”属性实现，首先在列表项“ul”上向右移动50%（left:50%;），

整个分页向右移动了50%的距离，紧接着我们在“li”上也定义“position:relative”属性，但其移动的方向和列表“ul”移动的方向刚好是反方向，而其移动的值保持一致。这样一来就实现了float浮动居中的效果。

**优点**：兼容性强，扩展性强。

**4、方法四：绝对定位实现**

绝对定位实现水平居中，我想大家也非常的熟悉了，并且用得一定不少。

.ele {

position: absolute;

width: 宽度值;

left: 50%;

margin-left: -(宽度值/2);

}

但这种实现我们有一个难题，我并不知道元素的宽度是多少，这样也就存在如方法一所说的难题，但我们可以借助方法三做一点变通：

.pagination {

position: relative;

}

.pagination ul {

position: absolute;

left: 50%;

}

.pagination li {

line-height: 25px;

margin: 0 5px;

float: left;

position: relative;/\*注意，这里不能是absolute，大家懂的\*/

right: 50%;

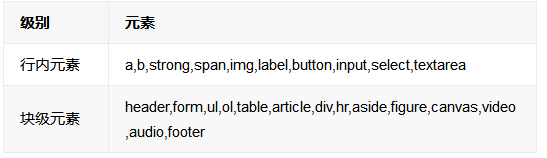
**优点**：扩展性强，兼容性强;

行内元素一般是内容的容器，而块级元素一般是其他容器的容器，行内元素适合显示具体内容，而块级元素适合做布局。

**块级元素(block)：**独占一行，对宽高的属性值生效；如果不给宽度，块级元素就默认为浏览器的宽度，即就是100%宽。

**行内元素(inline)：**可以多个标签存在一行，对宽高属性值不生效，完全靠内容撑开宽高。

**行内块元素(inline-block)：**结合的行内和块级的优点，既可以设置长宽，可以让padding和margin生效，又可以和其他行内元素并排。



其中****img和input为行内块元素****。

**行内元素与块状元素之间的转换:**

1.float: 当把行内元素设置完float:left/right后，该行内元素的display属性会被赋予block值，且拥有浮动特性。行内元素去除了之间的莫名空白。

2. position: 当为行内元素进行定位时，position:absolute与position:fixed.都会使得原先的行内元素变为块级元素。

3 .display:

1、块级标签转换为行内标签：display:inline;

2、行内标签转换为块级标签：display:block;

3、转换为行内块标签：display：inline-block;