CSS布局总结

选择器

css由选择器和声明组成

//.cssClass --- 选择器,一次定义多个选择器时可以使用','分割

.cssClass[,.css2Class]{

color:red; //声明

.....

}

元素选择器(类型选择器)

根据html中的标签名来设置样式声明。如：

html{} p{} h1{} ....

类选择器

该选择器可以设置html标签中所有class设置为该选择器的标签,给标签指定类的时候，可以指定多个类，如：

.import{} .class1{} ...

<div class='import class1'></div>

ID选择器

可以指定某一ID的标签的样式声明。如：

#id1{} #id2{} ....

<div id="id1"></div>

<div id="id2"></div>

属性选择器

可以给某个设有指定属性的标签设置样式。如：

[prop1]{} [prop2]{}

<div prop1="value1"></div>

同样可以给这个属性设置一个限定例如，

[prop1="123"]{}

<div prop1="123"></div>

<div prop1="3445"></div>

这样就只有prop1="123"的会有这些声明，其中的'='也可以换成'|='、'^='....

后代选择器

这个选择器是在其他选择器的基础上使用的，两个选择器之间使用空格分割，表示后者必须是前者的子标签的时候，才可以生效,例如：

h1 [ccc="123"] {color:red;}

h1 p {color:red;}

<h1>This is a <p ccc="123">important</p> heading</h1>

子元素选择器

这个选择器是在后代选择器的基础上，限制只能是子元素，而不能是子元素的子元素，例如：

h1 > p {color:red;}

<h1>This is a <p>important</p> heading</h1>

<h1>This is a <h2> <p>adsfasfsdf </p></h2> heading</h1>

相邻兄弟选择器

这个选择器可以选择紧接在前者后的后者元素，如：

h1 + p {}

<p id="id1"></p>

<h1></h1>

<p id="id2"></p>

<p id="id3"></p>

这里只有id2的p标签有效

元素定位与布局

元素的定位与布局的方式有，普通流定位、浮动定位、相对定位、绝对定位，在2009年后提出了一种新的布局方式Flax布局（伸缩盒子布局）。

普通流定位

普通的流定位中，标签分为两种类型：行内元素和块级元素。

# 块级元素包含：div、h系列、li、dt、dd等容器元素，此外还包括p这个文本级标签。块级元素特性如下：

1.这个元素会占用一整行，同一行内一般不会有其他元素

2.这个元素可以设置宽高属性

3.如果不设置宽度就是将父元素的宽度填满

# 行内元素包含：span,a,b,i,u,em等文本级标签，行内元素的特性如下：

1.这个元素不会占用一整行，会根据内容的多少来决定占用多少

2.一行内可以排列多个行内元素，除非放不下了，否则不会换行

3.该元素设置宽高属性无效

元素的类型可以通过display属性进行控制，所以可以通过这个属性修改元素的类型，如果想将行内元素修改为块级元素，需要设置为block，反之则需要修改为inline

当然普通的流定位方式局限性还是比较大，因此很多效果难以实现，我们有三种方式可以使元素跳出普通流定位方式：浮动、相对布局和绝对布局。

浮动定位

浮动定位是通过float属性来控制的，默认为none，其中还可以选取两个值left和right。

设置为浮动的元素可以再上层元素的下方跳出普通流定位，靠左或者靠右渲染，例如：

.div1{

width:100px;

height:50px;

background:blue;

}

.div2{

width:100px;

height:50px;

float:left;

background:green;

}

.div3{

width:100px;

height:100px;

background:red;

}

<div class="div1">DIV1</div>

<div class="div2">DIV2</div>

<div class="div3">DIV3</div>

此时我们可以看到的效果是：

DIV1

DIV2

DIV3

这里如果我们把DIV2的float改为right，看到的效果是这样的：

DIV1

DIV2

DIV3

这里我们需要注意两个问题：

1.如果元素设置为了浮动，那么他已经跳出了普通的流定位，因此他已经不算是第二个元素了因此，DIV1下面则是DIV3.

2.第一种效果的时候，实际上是DIV2把DIV3的上半部分给盖住了，但是DIV3上面的内容这个时候并不会被盖住，而是会挤到可显示的区域。

此外，除了浮动的方法，我们还有一个清楚浮动的方法，就是clear属性，此时如果div3设置了与div2的float相同的属性，或者both,那么DIV2所在的一行会空出来。

相对定位

相对定位的属性名称叫做position，值为relative,已上面的float:left的例子为例,如果我们给div2加上两个属性:position:relative; top:50px; 那么就会变成下面这样：

DIV1

DIV3

DIV2

\*这里的DIV3的文字实际上是会被挡住一半的

相对定位是在原来的定位的基础上，对元素进行偏移，而且不会影响到其他的元素。

绝对定位

绝对定位使用的控制属性也是position,其中分为两种，第一种是根据父元素的绝对定位，使用的是absolute,另一种是根据窗口的绝对定位，使用的是fixed。例如：

.father{

background:red;

width:200px;

height:200px;

position:absolute;

top:50%;

left:50%;

}

.father2{

background:blue;

width:100px;

height:100px;

position: absolute;

top:50%;

left:50%;

}

<div class="father">

<div class="father2"></div>

</div>

这种情况效果如下：

如果我们将father2的position改为fixed。则会变成下面的效果：

如果相同的两个绝对定位的元素，可以通过z-Index属性来绝对哪个显示在上面。

Flex布局（伸缩盒子布局）

这个布局模式也是通过display属性来控制的，但是因为比较新，所以设置的方式上存在兼容问题，如下：

display: -webkit-box; /\* Chrome 4+, Safari 3.1, iOS Safari 3.2+ \*/

display: -moz-box; /\* Firefox 17- \*/

display: -webkit-flex; /\* Chrome 21+, Safari 6.1+, iOS Safari 7+, Opera 15/16 \*/

display: -moz-flex; /\* Firefox 18+ \*/

display: -ms-flexbox; /\* IE 10 \*/

display: flex; /\* Chrome 29+, Firefox 22+, IE 11+, Opera 12.1/17/18, Android 4.4+ \*/

一.基础知识

Flex布局有主轴和副轴的概念，默认的情况主轴的方向是横向的从左往右。相对的副轴就是纵向的从上到下。

二、Flex布局常用属性

Flex布局有几个主要的属性,包括针对容器的和针对item的，如下：

针对容器的：

**flex-direction**

该属性是修改主轴属性，可以修改为row、row-reverse、column、column-reverse这四个

**flex-wrap**

设置如果容器显示的item一行不够放了，是否需要换行，可以设置的值有：nowrap、wrap、wrap-reverse这三个

**flex-flow**

这个属性可以同时设置以上两个属性

**justify-content**

设置主轴方向的排列方式，可以设置的属性有flex-start、flex-end、center、space-between、space-around五个属性。

**align-items**

设置副轴方向的排列方式，可以设置的属性有flex-start、flex-end、center、baseline、stretch(默认值)五个属性。

**align-content**

属性定义了多根轴线的对齐方式。如果项目只有一根轴线，该属性不起作用。实际上就是在item过多需要换行的时候，这个值才会有效果。可以设置的属性有flex-start、flex-end、center、space-between、space-around五个属性。

针对**item**的：

**order**

这个属性是设置item的排序位置的，数值越小越靠前，默认为0

**flex-grow**

这个属性是设置如果容器存在剩余空间时，该item所能占的空间的大小。

**flex-shrink**

这个属性是设置如果容器的空间不足时，item是否会缩小，缩小多少。默认为1，如果设置为0的话，就算空间不足，也不会缩小。

**flex-basis**

这个属性是设置，这个item占据多少主轴方向的空间，如果不设置就是auto，即item是多大就占用多少主轴方向的空间。

**flex**

这个属性可以同时设置上面的三个属性

**align-self**

这个属性是设置这个item在副轴上的对齐方式，可以用于特殊指定某一个item和容器内的其他item不一样。

小程序实践（公众号承接页面）