# CSS

## \*\*\*盒子模型

* margin 外边距
* border 边框
* padding 内边距
* content（内容区域）

width属性等于是content的宽度

总宽度 = 左右外边距 + 左右边框 + 左右内边距 + 内容宽度

总高度 = 上线外边距 + 上下边框 + 上下内边距 + 内容高度

IE5.X和6的width 属性等于 内边距+边框+content的宽度

## flex弹性布局

布局的传统解决方案，基于盒装模型，依赖display属性 + position属性 + float属性。它对于那些特殊布局非常不方便，比如，垂直居中就不容易实现。

使用flex布局以后，子元素的float、clear和vertical-align属性将失效。

主轴：X轴

交叉轴：Y轴

**Flex容器的属性**

**1、flex-directio**n属性决定X周的方向，row | row-reverse | column | column-reverse

**2、flex-wrap** 默认情况下，项目都排在一条线上，如果一条轴线排不下，如何换行。

nowrap（默认），不换行

wrap 换行，第一行再上方

wrap-reverse 换行，第一行在下方

**3、flex-flow** 是flex-direction和flex-wrap的简写形式，默认值为row || nowrap

**4、justify-content** 定义项目再主轴上的对齐方式

flex-start（默认值）左对齐

flex-end 右对齐

center 居中

space-between 两端对齐，项目之间的间隔都相等

space-around 每个项目两侧的间隔相等。

所以，项目之间的间隔比项目与边框的间隔大一倍。

**5、align-items** 定义项目在y轴上如何对齐

flex-start 起点对齐

flex-end 终点对齐

center 终点对齐

baseline 项目的第一行文字的基线对齐

stretch（默认值）如果项目未设置高度或者设置为auto，将占满整个容器的高度

**6、align-content** 定义了很多轴线的对齐方式。

如果项目只有一根轴线，该属性不起作用。

**Flex项目的属性**

1. **order** 定义项目的排列顺序，数值越小，排列越靠前，默认为0
2. **flex-grow** 定义项目的放大比例，默认为0，即如果存在剩余空间，也不放大

如果所有项目的flex-grow属性都为1，则它们将等分剩余空间（如果有的话）；

如果一个项目的flex-grow属性为2，其他都为1，则前者占据的剩余空间比其他项多一倍。

举例：

* flex容器总宽度为1000px，项目1、项目2、项目3三个项目宽度为100px，

当项目1的flex-grow为1，项目2、项目3不设置，则项目1的宽度为800px；

当项目1的flex-grow为1，项目2的flex-grow为1，项目3的flex-grow为0，则项目1和项目2平分剩余空间，则项目1和项目2为450px；

1. **flex-shrink** 定义项目的缩小比例，默认为1，即如果空间不足，将项目缩小

flex-shrink为0则项目不缩小

1. **flex-basis** 定义了在分配多余空间之前，项目占据的主轴空间。浏览器根据这个属性，计算主轴是否有多余空间。它的默认值是auto，即项目的未来大小。

它可以设置为跟width或height属性一样的值（比如350px），则项目将占据固定空间。

flex-basis设置的宽度会覆盖width；

1. **flex** 是flex-grow、flex-shrink和flex-basis的缩写，默认值为0 1 auto。后两个属性可选
2. **flex-self** 允许单个项目有与其他项目不一样的水平对齐方式，可以覆盖align-items属性。

默认为auto，表示继承父元素的align-items属性。

如果没有父元素，则等同于stretch

## fixed什么时候不相对浏览器定位？

当元素祖先的transform属性非none时，定位容器有视口改为该祖先。

并不是所有能够生成层叠上下文的元素都会是的position:fixed失效，但也不止transform

## 如何触发一个元素形成层叠上下文

1. 根元素<html>
2. z-index值不为auto的绝对和相对定位
3. 一个z-index值不为auto的flex项目（flex item），即父元素display：flex|inline-flex
4. opacity [əʊˈpæsəti] 属性值小于1的元素
5. transform属性值不为none的元素
6. mix-blend-mode [mɪks] [məʊd] 属性值不为normal的元素

mix-blend-mode属性设置图片元素与父容器背景（黄色）进行混合，图片会变暗

1. filter值不为none的元素

filter属性定义了元素（通常是img的）可视效果（如：模糊与饱和度）

1. position：fixed
2. ……..

## px、em、rem，VW，VH

em相对长度单位，相对于当前对象内文字的字体尺寸，如当前对行内文字的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。

rem是css3新增的一个相对单位（root em，根em），rem为元素设定字体大小时，依然是相对大小，但相对的只是HTML根元素。

VM：视口的宽度为100VM，相对于把视口分为100份；

VH：视口的高度为100VH，同理。

## 交互过程中布局layout [ˈleɪaʊt]的优化

## Js执行的优化

## line-height：2 /2em/2%

浏览器有一个默认文字大小是16px，

 1、line-height:2em、line-height:200%

　　　  根据父元素的字体大小（font-size）计算行高

         子元素继承父元素计算后的行高，如果子元素的字体很大，而父元素字体小，这样计算出来的行高有可能无法容量子元素的显示，会被截掉(或是重叠)。

     2、line-height:2（推荐）

　　    根据子元素的font-size计算行高（取最大字体值\*2）

如何父元素和子元素都没有设置font-size，就按照浏览器默认的16px\*2

## \*\*\*响应式布局（自适应网页设计）

让一张网页自动适应不同大小的屏幕，根据屏幕宽度，自动调整布局。

自动识别屏幕宽度，并做出响应调整的网页设计。

首先，在网页代码的头部，加入一行viewport元标签。

<meta name=”viewport” content=”width=device-width, initial-scale=1” />

viewport是网页默认的宽度和高度，

width=device-width网页宽度默认等于屏幕宽度

initial-scale=1 原始缩放比例为1.0，即网页初始大小占屏幕面积的100%。

* **媒体查询**

自动探测屏幕宽度，然后加载相应的css。

css3媒体查询可以让我们针对不同的媒体类型定义不同的样式。

当你重置浏览器大小的过程中，页面也会根据浏览器的宽度和高度重新渲染页面。

* **百分比布局**
* **rem布局**

rem是css3新增的单位

rem单位是相对于根元素html的font-size来决定大小的。

根据视图容器的大小，动态的改变font-size即可。

rem布局是目前多屏幕适配的最佳方式。

**动态改变font-size的方式：**

1. 媒体查询方式
2. Js动态计算
3. Scss计算，指定rem基准值为37.5px，也有设置基准值为75px的

|  |
| --- |
| @function px2rem($px) {     $rem: 37.5px;      @return ($px / $rem) + rem;  } |

* **视口单位**

css中引入vm/vh，与视图窗口有关

vm表示相对于视图的宽度；vh表示相对于视图窗口的高度。

Vmin vw和vh中的较小值；Vmax vw和vh中的较大值

vw和vh很类似百分比单位，区别是%大部分相对于祖先元素，也有相对于自身的情况，比如border-radius、translate等；vm、vh是相对于视口的尺寸

* **图片响应式**

## 移动端适配

<https://www.runoob.com/w3cnote/moible-web-front-source.html>

meta篇

1、视图宽度

<meta name="viewport" content="width=device-width,initial-scale=1.0,minimum-scale=1.0,maximum-scale=1.0,user-scalable=no"/>

width=device-width 设置视图宽度为设备视图宽度，还可以固定宽度，例如：width=640则是640px的宽度（常见于微信）；

initial-scale=1.0 设置缩放比例为1.0

minimum-scale=1.0 和maximum-scale=1.0 最小缩放比例和最大缩放比例

user-scalable=no 禁止用户自由缩放，默认值为yes

提示：刚刚那个是带全部参数的，一般常用的，有user-scalable=no就不用使用minimum-scale=1.0和maximum-scale=1.0来强制禁止缩放了

2、自动识别格式

<meta name="format-detection" content="telephone=no"/>

Content里面的参数：telephone=no是禁止浏览器自动识别手机号码；email=no是禁止浏览器自动识别email。

Onorientationchange 手机屏幕旋转

Rezie 浏览器窗口缩放

document.documentElement 返回的是<html>……..</html>

document.documentElement.getBoundingClientRect().width （html的宽）

window.screen.width（设备的宽）

window.getComputedStyle(document.documentElement, null).getPropertyValue('font-size')

获取html标签设置的font-size

当浏览器没有最大化的时候，html的宽度小于设备的宽度

大多数浏览器的默认字号是16px，所以1rem=16px

设计稿750

Flexible将设计稿分为10份，可以理解为页面的width=10rem，即1rem=75px，所以根font-size（基准值）=75px

返利根节点data-screenwidth和font-size的设置

data-screenwidth（html的宽度）

data-screenwidth > 450时，font-size=60

data-screenwidth < 450时，font-size= data-screenwidth /750 \* 100

定时器实时监听屏幕大小，然后重新设置屏幕宽度和font-size

返利默认设置body的font-size=14px

当根html的font-size = 60px

则相当于1rem=60px

当设置某一个div的font-size=0.25rem时，font-size换算后为60\*0.25=15px

## Css样式权重

 !important > 行内 > 头部 > 引入

## line-height/font-size/height

## css伪类

input:checked 所有选中的表单元素

input:disabled 所有禁用的表单元素

input:enabled 所有启用的表单元素

input:in-range 元素指定范围内的值

input-invalid 无效的元素

p:empty 没有子元素的p元素

p:first-of-type 第一个p元素

p:last-child p元素下的最后一个子元素

## css三大预处理器（Sass、Less、stylus）

预处理器是一种用来为CSS增加一些编程特性的语言，无需考虑浏览器的兼容问题。

## BFC

## \*\*清除浮动的方法

[**https://www.cnblogs.com/plsmile/p/11125641.html**](https://www.cnblogs.com/plsmile/p/11125641.html)

标准流：盒子会各占整行位置。子盒子若是标准流，父盒子虽然没有高度，但是会撑开父盒子高度。

浮动：盒子浮起来，不会占据原来的位置，若父盒子没有定义高度，则不会撑开父盒子，父盒子高度为0。

浮动可以让多个块级元素在一行显示，且块与块之间没有间隙，但要注意给父盒子清除浮动，否则父盒子不会被撑开。

**为什么要清楚浮动呢？清除浮动的本质是什么?**

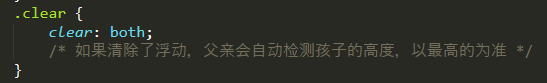
清除浮动主要是为了解决父级元素因为子级浮动引起的内部高度为0的问题。

**清除浮动的方法：**

* 额外标签法：给谁清除浮动，就在其后额外添加一个空白标签，空白标签添加样式clear:both

优点：通俗易懂，书写方便（不推荐使用）

缺点：添加需要无意义的标签，结构化比较差



* 父级添加overflow方法：可以通过触发BFC的方式，实现清除浮动效果。

优点：代码简洁（慎重使用，若该父盒子里还有position定位会引起麻烦）

缺点：内容增多的时候容易造成不会自动换行导致内容被隐藏掉，无法显示需要溢出的元素

Overflow:hidden;

注意：别加错位置，是给父亲加（并不是所有的浮动都需要清除，谁影响布局，才清除谁。）

* 使用after伪元素清除浮动

After方式是空元素的升级版，好处是不用单独加标签了。（较常用）

优点：符合闭合浮动思想，结构语义化正确

缺点：由于IE6-7不支持after，使用zoom：1触发hasLayout



* 使用before和after双伪元素清除浮动（较常用）



## \*\*\*垂直居中的实现

* **对单行元素进行垂直居中**

1、设置line-height等于父元素的height

* **对文本进行垂直居中**

设置display：flex和align-items：center实现文本居中

* **对已知高度块级元素进行垂直居中**

1. 绝对定位，配合top:50%和margin-top（元素高度一半）进行垂直居中
2. 绝对定位，配合top:0; bottom:0; 和margin:auto 进行垂直居中 margin [ˈmɑːdʒɪn]
3. 绝对定位，配合top:calc(50% - 元素本身高度一半) 函数进行垂直居中
4. ~~使用浮动float实现元素垂直居中~~

~~原理：通过再要进行垂直居中的元素a前面添加一个无内容的元素，并将该无内容元素的高设置为50%，再利用clear:both清除浮动，则元素a相对于父元素来说是垂直居中。~~

* **对未知高度块级元素进行垂直居中**

1. 绝对定位，配合top:50%和transform:translate(x,y)实现水平垂直居中

transform:translate(-50%,-50%); 针对元素本身向左以及向上移动50%

1. 居于视口单位的解决办法

通过使用margin-top:50vh配合transform:translateY(-50%),实现视口居中

1. 通过display:table-cell和vertical-aligin:middle，实现垂直居中

设置父块display:table，子块display:table-cell，vertical-aligin：middle

1. 基于flex的实现方案

设置父块display：flex，子块margin:auto

## transform

css transform属性允许你旋转、缩放、倾斜或平移给定的元素。

|  |  |  |
| --- | --- | --- |
| translate | 转换、平移 | 3D，x，y，z |
| scale [skeɪl] | 缩放 | 3D，x，y，z |
| rotate [rəʊˈteɪt] | 旋转 | 3D，x，y，z |
| skew [skjuː] | 倾斜 | x，y |
| perspective [pəˈspektɪv] | 3D元素定义透视视图 |  |

应用：

1. 画出小于1px的直线

transform: scaleY(0.2);

1. 实现box居中

 #box{

            width: 300px;

            background-color: green;

            top: 50%;

            transform: translateY(-50%);

            position: fixed;

        }

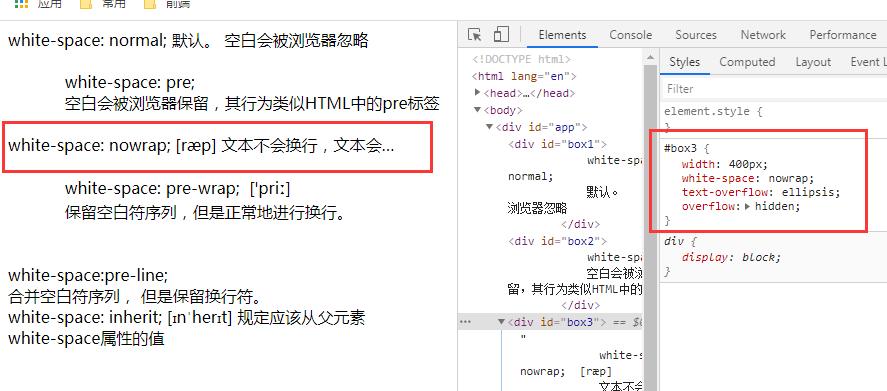
## white-space

white-space属性设置如何处理元素内的空白

## text-overflow

text-overflow属性规定当文本溢出包含元素时发生的事情。

## 一行显示多余显示省略号



## 实现table的所有列相等table-layout

实现方式如下：

注意：一定要设置width:100%，不然不生效

layout [ˈleɪaʊt] 布局

 table{

            width: 100%;

            table-layout: fixed;

            }

table-layout属性用来显示表格单元格、行、列的算法规则。

* auto：大多数浏览器采用自动表格布局算法对表格布局。表格及单元格的宽度取决于其包含的内容。
* fixed：表格和列的宽度通过表格的宽度来设置。

## Position（定位）属性

* static 默认值

元素使用正常的布局行为，即元素在文档常规流中当前的布局位置。此时top/right/bottom/left和z-index属性无效。

* relative [ˈrelətɪv]

相对定位；相对其正常位置

* absolute [ˈa b si liːt]

绝对定位；相对于最近的已定位父元素；

如果元素没有已定位的父元素，那么它的位置相对于<html>

absolute定位使元素的位置与文档流无关，因此不占据空间。

* fixed

相对于浏览器窗口的固定位置。

fixed定位使元素的位置与文档流无关，因此不占据空间。

* sticky [ˈstɪki]

<http://www.ruanyifeng.com/blog/2019/11/css-position.html>

skicky和前面四个属性值都不一样，它会产生动态效果，很像relative [ˈrelətɪv]和fixed的结合：一些时候是relative定位（定位基点是自身默认位置），另一些时候自动变成fixed定位（定位基点是视口）。

（1）因此，它能够形成“动态固定”的效果。

比如，网页的搜索工具栏，初始加载时在自己的默认位置（relative定位）；

页面向下滚动时，工具栏变成固定位置，始终停留在页面头部（fixed定位）；

等到页面重新向上滚动回到原位，工具栏也会回到默认位置。

sticky生效的前提是，必须搭配top、bottom、left、right这四个属性一起使用，不能忽略，否则等同于relative定位，不产生“动态固定”的效果。原因是这四个属性用来定义“偏移距离”，浏览器把它当做sticky的生效门槛。

（2）sticky属性值可实现堆叠效果

堆叠效果指的是页面滚动时，下方的元素覆盖上面的元素。

它的原理是页面向下滚动时，每张图片都会变成fixed定位，导致后一张图片重叠在前一张图片上面。

（3）表格的表头固定

大型表格滚动的时候，表头始终固定，也可以使用sticky实现。

需要注意的是：sticky必须设在<th>元素上面，不能设在<thead>和<tr>元素，因为这两个元素没有relative定位，也就无法产生sticky效果。

## z-index

z-index属性设置元素的堆叠顺序。拥有更高堆叠顺序的元素总是会处于堆叠顺序较低的元素前面

元素可拥有负的z-index属性值；

z-index仅能在定位元素上奏效（即position的值为relative、absolute、fixed、sticty）

## z-index失效的情况

1. 父标签position属性为relative [ˈrelətɪv]
2. 问题标签无position属性（不包含static）
3. 问题标签含有浮动（float）属性

## link和@import的区别

## CSS浏览器兼容性问题解决方案

<https://juejin.cn/post/6844903493161975822>

<https://www.jianshu.com/p/e11e2430c3f0>

1. 浏览器CSS样式初始化

Normalize.css [ˈnɔːməlaɪz]

自定义初始换样式reset.css

reset.css的作用是全部重置各浏览器的默认样式，而normalize [ˈnɔːməlaɪz] 则是可以使元素的渲染在多个浏览器下都能保持一致并且符合规范，而不是暴力的一概全部重置样式。

1. 浏览器私有属性

-moz代表firefox浏览器私有属性

-ms代表IE浏览器私有属性

-webkit代表chrome、safari私有属性

-o 代表opera私有属性

标准写法放到最后，兼容性写法放到前面

-webkit-transform:rotate(-3deg) ;

-moz-transform:rotate(-3deg);

-ms-transform:rotate(-3deg);

-o-transform:rotate(-3deg);

transform:rotate(-3deg);

每个css属性写这么一堆兼容性代码，太不方便，可以通过自动化插件来处理。

Autoprefixer是一款自动管理浏览器前缀的插件，它可以解析css文件并且添加浏览器前缀到css内容中，使用Can I Use的数据决定哪些前缀是需要的。

1. css hack [hæk]

<https://blog.csdn.net/freshlover/article/details/12132801>

针对不同浏览器写不同的css，这样的过程就叫做css hack。

## Zoom

Zoom这个属性是ie的专有属性，除了设置或者检索对象的缩放比例之外，它还可以触发ie的haslayout属性，清除浮动，清除margin重叠等作用。

## ie的haslayout属性