语法由一个 [选择器(selector)](https://developer.mozilla.org/zh-CN/docs/Glossary/CSS_Selector)起头。 它 选择(selects) 了我们将要用来添加样式的 HTML 元素。 在这个例子中我们为一级标题（主标题[<h1> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements)）添加样式

接着输入一对大括号{ }。 在大括号内部定义一个或多个形式为 **属性(property):值(value);** 的 **声明(declarations)**。每个声明都指定了我们所选择元素的一个属性，之后跟一个我们想赋给这个属性的值。

冒号之前是属性，冒号之后是值。不同的 CSS [属性(properties) (en-US)](https://developer.mozilla.org/en-US/docs/Glossary/property/CSS) 对应不同的合法值。

**浏览器渲染dom生命周期，CSS是如何渲染的。**

浏览器载入HTML文件（比如从网络上获取）。

将HTML文件转化成一个DOM（Document Object Model），DOM是文件在计算机内存中的表现形式，下一节将更加详细的解释DOM。

html 加载时发生了什么

在页面加载时，浏览器把获取到的HTML代码解析成1个DOM树，DOM树里包含了所有HTML标签，包括display:none隐藏，还有用JS动态添加的元素等。

浏览器把所有样式(用户定义的CSS和用户代理)解析成样式结构体

DOM Tree 和样式结构体组合后构建render tree, render tree类似于DOM tree，但区别很大，因为render tree能识别样式，render tree中每个NODE都有自己的style，而且render tree不包含隐藏的节点(比如display:none的节点，还有head节点)，因为这些节点不会用于呈现，而且不会影响呈现的，所以就不会包含到 render tree中。我自己简单的理解就是DOM Tree和我们写的CSS结合在一起之后，渲染出了render tree。

接下来，浏览器会拉取该HTML相关的大部分资源，比如嵌入到页面的图片、视频和CSS样式。JavaScript则会稍后进行处理，简单起见，同时此节主讲CSS，所以这里对如何加载JavaScript不会展开叙述。

**浏览器拉取到CSS之后会进行解析，根据选择器的不同类型（比如element、class、id等等）把他们分到不同的“桶”中。浏览器基于它找到的不同的选择器，将不同的规则（基于选择器的规则，如元素选择器、类选择器、id选择器等）应用在对应的DOM的节点中，并添加节点依赖的样式（这个中间步骤称为渲染树）。**

上述的规则应用于渲染树之后，渲染树会依照出现的结构进行布局。

网页展示在屏幕上（这一步被称为着色）。

回流必将引起重绘，而重绘不一定会引起回流

**重绘:当render tree中的一些元素需要更新属性，而这些属性只是影响元素的外观，风格，而不会影响布局的，比如background-color。则就叫称为重绘。**

**回流：添加或者删除可见的DOM元素，元素位置改变，元素尺寸改变——边距、填充、边框、宽度和高度，内容改变**

**浏览器的帮忙**

所以我们能得知回流比重绘的代价要更高，回流的花销跟render tree有多少节点需要重新构建有关系
因为这些机制的存在，所以浏览器会帮助我们优化这些操作，浏览器会维护1个队列，把所有会引起回流、重绘的操作放入这个队列，等队列中的操作到了一定的数量或者到了一定的时间间隔，浏览器就会flush队列，进行一个批处理。这样就会让多次的回流、重绘变成一次回流重绘。

<https://www.jianshu.com/p/e081f9aa03fb>

[当浏览器遇到无法解析的CSS代码会发生什么](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/First_steps/How_CSS_works#%E5%BD%93%E6%B5%8F%E8%A7%88%E5%99%A8%E9%81%87%E5%88%B0%E6%97%A0%E6%B3%95%E8%A7%A3%E6%9E%90%E7%9A%84css%E4%BB%A3%E7%A0%81%E4%BC%9A%E5%8F%91%E7%94%9F%E4%BB%80%E4%B9%88)

答案就是浏览器什么也不会做，继续解析下一个CSS样式！

如果一个浏览器在解析你所书写的CSS规则的过程中遇到了无法理解的属性或者值，它会忽略这些并继续解析下面的CSS声明。在你书写了错误的CSS代码（或者误拼写），又或者当浏览器遇到对于它来说很新的还没有支持的CSS代码的时候上述的情况同样会发生（直接忽略）。

相似的，当浏览器遇到无法解析的选择器的时候，他会直接忽略整个选择器规则，然后解析下一个CSS选择器。

你使用最新的CSS优化的过程中浏览器遇到无法解析的规则也不会报错

**1.层叠、优先级和继承**

[层叠](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance#%E5%B1%82%E5%8F%A0)

Stylesheets **cascade（样式表层叠）** — 简单的说，css规则的顺序很重要；

[优先级](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance#%E4%BC%98%E5%85%88%E7%BA%A7)

浏览器是根据优先级来决定当多个规则有不同选择器对应相同的元素的时候需要使用哪个规则。它基本上是一个衡量选择器具体选择哪些区域的尺度：

一个元素选择器不是很具体 — 会选择页面上该类型的所有元素 — 所以它的优先级就会低一些。

一个类选择器稍微具体点 — 它会选择该页面中有特定

class

 属性值的元素 — 所以它的优先级就要高一点。

[继承](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance#%E7%BB%A7%E6%89%BF)

继承也需要在上下文中去理解 —— 一些设置在父元素上的css属性是**可以被子元素继承的**，有些则不能，**一些属性是不能继承的**。

https://www.cnblogs.com/thislbq/p/5882105.html

**可继承属性**

字体属性，font

文字属性，text

3、元素可见性：visibility

4、表格布局属性：caption-side、border-collapse、border-spacing、empty-cells、table-layout

5、列表布局属性：list-style-type、list-style-image、list-style-position、list-style

6、生成内容属性：quotes

7、鼠标光标属性：cursor

8、页面样式属性：page、page-break-inside、windows、orphans

9、声音样式属性：speak、speak-punctuation、speak-numeral、speak-header、speech-rate、volume、voice-family、pitch、pitch-range、stress、richness、、azimuth、elevation

**不可继承属性**

[控制继承](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance#%E6%8E%A7%E5%88%B6%E7%BB%A7%E6%89%BF)

CSS 为控制继承提供了四个特殊的通用属性值。每个css属性都接收这些值。

[inherit](https://developer.mozilla.org/zh-CN/docs/Web/CSS/inherit)

设置该属性会使子元素属性和父元素相同。实际上，就是 "开启继承".

[initial](https://developer.mozilla.org/zh-CN/docs/Web/CSS/initial)

设置属性值和浏览器默认样式相同。如果浏览器默认样式中未设置且该属性是自然继承的，那么会设置为 inherit 。

[unset](https://developer.mozilla.org/zh-CN/docs/Web/CSS/unset)

将属性重置为自然值，也就是如果属性是自然继承那么就是 inherit，否则和 initial一样

**注**: 还有一个新的属性, [revert (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/revert)， 只有很少的浏览器支持。

[重设所有属性值](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance#%E9%87%8D%E8%AE%BE%E6%89%80%E6%9C%89%E5%B1%9E%E6%80%A7%E5%80%BC)

CSS  属性 all 可以用于同时将这些继承值中的一个应用于（几乎）所有属性

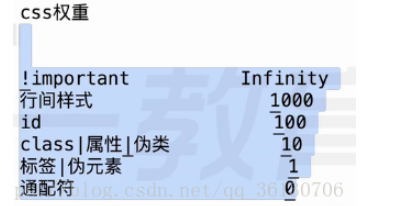
[理解层叠](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance#%E7%90%86%E8%A7%A3%E5%B1%82%E5%8F%A0)

重要程度

优先级

会发现在一些情况下，有些规则在最后出现，但是却应用了前面的规则。这是因为前面的有更高的**优先级**

**权重**

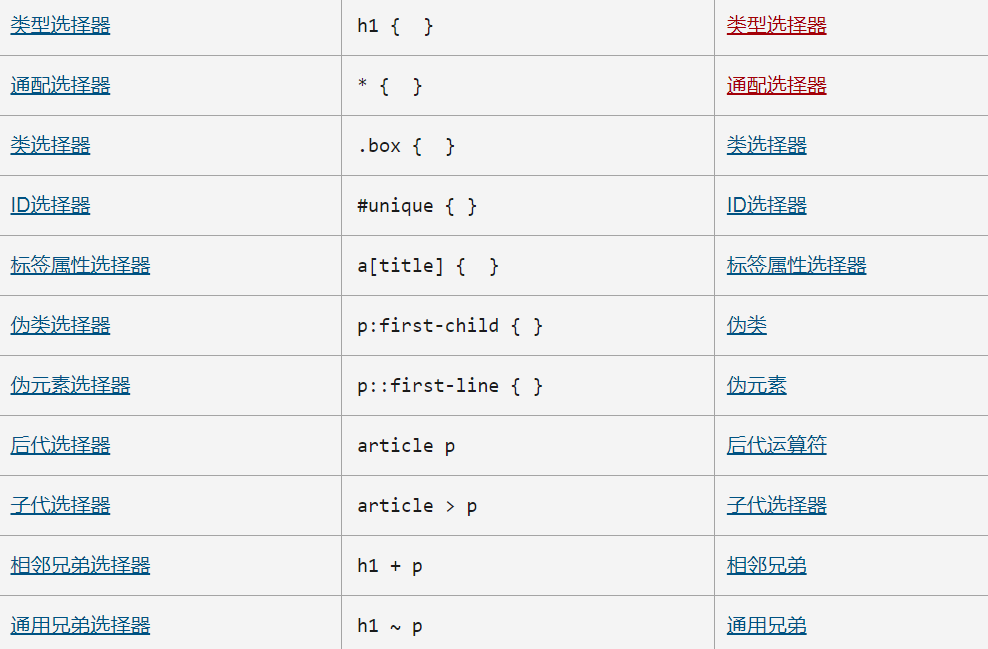


 覆盖 !important 唯一的办法就是另一个 !important 具有 相同**优先级** 而且顺序靠后，或者更高优先级。 **但是，强烈建议除了非常情况不要使用它**!important

资源顺序

如果你有超过一条规则，而且都是相同的权重，那么最后面的规则会应用。可以理解为后面的规则覆盖前面的规则，直到最后一个开始设置样式

**2.**[选择器列表](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Selectors#%E9%80%89%E6%8B%A9%E5%99%A8%E5%88%97%E8%A1%A8)



**3.盒模型**

**块级盒子** (**block box**) 和 **内联盒子** (**inline box**)**。**这两种盒子会在**页面流**（page flow）和**元素之间的关系**方面表现出不同的行为。

**正常文档流**

块级盒子：

盒子会在内联的方向上扩展并占据父容器在该方向上的所有可用空间，在绝大数情况下意味着盒子会和父容器一样宽

每个盒子都会换行

width和height 属性可以发挥作用

内边距（padding）, 外边距（margin） 和 边框（border） 会将其他元素从当前盒子周围“推开

内联盒子

盒子不会产生换行。

width 和 height属性将不起作用。

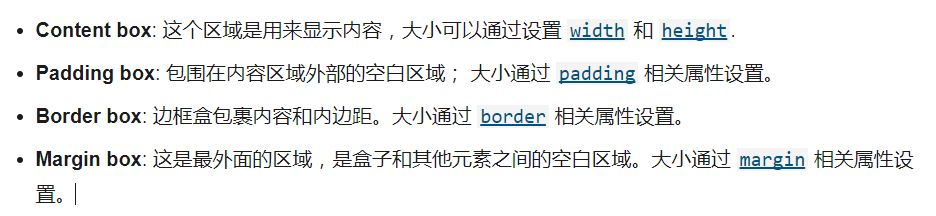
垂直方向的内边距、外边距以及边框会被应用但是不会把其他处于 inline状态的盒子推开。

水平方向的内边距、外边距以及边框会被应用且会把其他处于 inline 状态的盒子推开

[什么是CSS 盒模型?](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/The_box_model#%E4%BB%80%E4%B9%88%E6%98%AFcss_%E7%9B%92%E6%A8%A1%E5%9E%8B)

完整的 CSS 盒模型应用于块级盒子，内联盒子只使用盒模型中定义的部分内容。模型定义了盒的每个部分 —— margin, border, padding, and content —— 合在一起就可以创建我们在页面上看到的内容。

**1.标准盒模型：**在标准模型中，如果你给盒设置 width 和 height，实际设置的是 **content box**。 padding 和 border 再加上设置的宽高一起决定整个盒子的大小。



2.[替代（IE）盒模型](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/The_box_model#%E6%9B%BF%E4%BB%A3%EF%BC%88ie%EF%BC%89%E7%9B%92%E6%A8%A1%E5%9E%8B)

所有宽度都是可见宽度，所以内容宽度是该宽度减去边框和填充部分。

默认浏览器会使用标准模型。如果需要使用替代模型，您可以通过为其设置 box-sizing: border-box 来实现。

[使用调试工具来查看盒模型](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/The_box_model#%E4%BD%BF%E7%94%A8%E8%B0%83%E8%AF%95%E5%B7%A5%E5%85%B7%E6%9D%A5%E6%9F%A5%E7%9C%8B%E7%9B%92%E6%A8%A1%E5%9E%8B)

[外边距](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/The_box_model#%E5%A4%96%E8%BE%B9%E8%B7%9D)

外边距是盒子周围一圈看不到的空间。它会把其他元素从盒子旁边推开。 外边距属性值可以为正也可以为负。**设置负值会导致和其他内容重叠。**无论使用标准模型还是替代模型，**外边距总是在计算可见部分后额外添加。**

**尝试更改外边距的值，来查看当前元素和其包含元素，在外边距设置为正时是如何推开周边元素，以及设置为负时，是如何收缩空间的。**

**外边距折叠:理解外边距的一个关键是外边距折叠的概念。如果你有两个外边距相接的元素，这些外边距将合并为一个外边距，即最大的单个外边距的大小。**

[边框](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/The_box_model#%E8%BE%B9%E6%A1%86)

边框是在边距和填充框之间绘制的。如果您正在使用标准的盒模型，边框的大小将添加到框的宽度和高度。

宽度，边框状态，颜色

[内边距](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/The_box_model#%E5%86%85%E8%BE%B9%E8%B7%9D)

内边距位于边框和内容区域之间。与外边距不同，不能有负数量的内边距，值必须是0或正的值。应用于元素的任何背景都将显示在内边距后面，内边距通常用于将内容推离边框。

[CSS的背景样式](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Backgrounds_and_borders#css%E7%9A%84%E8%83%8C%E6%99%AF%E6%A0%B7%E5%BC%8F)

.box {

background: linear-gradient(105deg, rgba(255,255,255,.2) 39%, rgba(51,56,57,1) 96%) center center / 400px 200px no-repeat,

url(big-star.png) center no-repeat, rebeccapurple;

}

[background-color](https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-color) **背景颜色**

[background-image](https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-image) [背景图片](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Backgrounds_and_borders#%E8%83%8C%E6%99%AF%E5%9B%BE%E7%89%87)

也可以有多个背景图像——在单个属性值中指定多个background-image值，用逗号分隔每个值。

background-image: url(image1.png), url(image2.png), url(image3.png), url(image1.png);

background-repeat: no-repeat, repeat-x, repeat;

background-position: 10px 20px, top right;

[background-repeat](https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-repeat) 控制背景平铺

[background-size](https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-size) **调整背景图像的大小**

cover —浏览器将使图像足够大，使它完全覆盖了盒子区，同时仍然保持其高宽比。

contain — 浏览器将使图像的大小适合盒子内。

[background-position](https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-position) 背景图像定位

[background-image](https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-image) [渐变背景](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Backgrounds_and_borders#%E6%B8%90%E5%8F%98%E8%83%8C%E6%99%AF)

[background-attachment](https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-attachment)属性只有在有内容要滚动时才会有效果

scroll：使元素的背景在页面滚动时滚动。如果滚动了元素内容，则背景不会移动。实际上，背景被固定在页面的相同位置，所以它会随着页面的滚动而滚动。

fixed:：使元素的背景固定在视图端口上，这样当页面或元素内容滚动时，它就不会滚动。它将始终保持在屏幕上相同的位置。

local：这个值是后来添加的，因为滚动值相当混乱，在很多情况下并不能真正实现您想要的功能。局部值将背景固定在设置的元素上，因此当您滚动元素时，背景也随之滚动。

**border:边框**

通过使用[border-radius](https://developer.mozilla.org/zh-CN/docs/Web/CSS/border-radius)属性和与方框的每个角相关的长边来实现方框的圆角。

[处理不同方向的文本](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Handling_different_text_directions)（详细查阅MDN文档）

writing-mode的三个值分别是：

horizontal-tb: 块流向从上至下。对应的文本方向是横向的。

vertical-rl: 块流向从右向左。对应的文本方向是纵向的。

vertical-lr: 块流向从左向右。对应的文本方向是纵向的。

**溢出的内容**

可以通过给[width](https://developer.mozilla.org/zh-CN/docs/Web/CSS/width)和[height](https://developer.mozilla.org/zh-CN/docs/Web/CSS/height)（或者 [inline-size](https://developer.mozilla.org/zh-CN/docs/Web/CSS/inline-size) 和 [block-size](https://developer.mozilla.org/zh-CN/docs/Web/CSS/block-size)）赋值的方式来约束盒子的尺寸。溢出是在你往盒子里面塞太多东西的时候发生的，所以盒子里面的东西也不会老老实实待着

[CSS尽力减少“数据损失”](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Overflowing_content#css%E5%B0%BD%E5%8A%9B%E5%87%8F%E5%B0%91%E2%80%9C%E6%95%B0%E6%8D%AE%E6%8D%9F%E5%A4%B1%E2%80%9D)

CSS就不会隐藏你的内容，隐藏引起的数据损失通常会造成困扰。

[overflow属性](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Overflowing_content#overflow%E5%B1%9E%E6%80%A7)**:控制一个元素溢出的方式，浏览器想怎样处理溢出。**

overflow:visible 默认属性看的见的

overflow: hidden 隐藏掉溢出

overflow: scroll 滚动

overflow-y: scroll y轴方向滚动。

overflow-x: scroll x轴方向滚动。

overflow: auto 显示滚动

word-break

normal:使用默认的断行规则。

break-all:对于non-CJK (CJK 指中文/日文/韩文) 文本，可在任意字符间断行。

keep-all:CJK 文本不断行。 Non-CJK 文本表现同 normal。

break-word :他的效果是word-break: normal 和 overflow-wrap: anywhere  的合，不论 [overflow-wrap](https://developer.mozilla.org/zh-CN/docs/Web/CSS/overflow-wrap)的值是多少。

overflow-wrap: normal;

overflow-wrap: break-word;

normal:行只能在正常的单词断点处中断。（例如两个单词之间的空格）。

break-word:表示如果行内没有多余的地方容纳该单词到结尾，则那些正常的不能被分割的单词会被强制分割换行。

CSS中有所谓**块级排版上下文（Block Formatting Context，BFC）**的概念

[什么是CSS的值?](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Values_and_units#%E4%BB%80%E4%B9%88%E6%98%AFcss%E7%9A%84%E5%80%BC)

颜色:

关键字、十六进制值、rgb()函数,rgba()函数

[数字，长度和百分比](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Values_and_units#%E6%95%B0%E5%AD%97%EF%BC%8C%E9%95%BF%E5%BA%A6%E5%92%8C%E7%99%BE%E5%88%86%E6%AF%94)

**绝对长度单位:px**

**相对长度单位:em,rem,vw,vh**

在排版属性中 em 单位的意思是“父元素的字体大小”。带有ems类的[<ul>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/ul)内的[<li>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/li)元素从它们的父元素中获取大小。因此，**每一个连续的嵌套级别都会逐渐变大，因为每个嵌套的字体大小都被设置为1.3em—是其父嵌套字体大小的1.3倍。**

**rem单位的意思是“根元素的字体大小”**。(“根em”的rem标准。)[<ul>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/ul)内的[<li>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/li)元素和一个rems类从根元素(<html>)中获取它们的大小。这意味着每一个连续的嵌套层都不会不断变大。

[百分比](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Values_and_units#%E7%99%BE%E5%88%86%E6%AF%94)

百分比的问题在于，它们总是相对于其他值设置的。例如，如果将元素的字体大小设置为百分比，那么它将是元素父元素字体大小的百分比。如果使用百分比作为宽度值，那么它将是父值宽度的百分比。

[数字](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Values_and_units#%E6%95%B0%E5%AD%97)

有些值接受数字，不添加任何单位。接受无单位数字的属性的一个例子是不透明度属性（opacity ），它控制元素的不透明度(它的透明程度)。此属性接受0(完全透明)和1(完全不透明)之间的数字。

[图片](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Values_and_units#%E5%9B%BE%E7%89%87)

[<image>](https://developer.mozilla.org/en-US/docs/Web/CSS/image) 数据类型用于图像为有效值的任何地方。它可以是一个通过

url()函数指向的实际图像文件，也可以是一个渐变。

background-image

[<position>](https://developer.mozilla.org/en-US/docs/Web/CSS/position_value) 数据类型表示一组2D坐标，用于定位一个元素，如背景图像(通过

[background-position](https://developer.mozilla.org/en-US/docs/Web/CSS/background-position))。它可以使用关键字(如top, left, bottom, right, 以及center )将元素与2D框的特定边界对齐，以及表示框的顶部和左侧边缘偏移量的长度。

[字符串和标识符](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Values_and_units#%E5%AD%97%E7%AC%A6%E4%B8%B2%E5%92%8C%E6%A0%87%E8%AF%86%E7%AC%A6)

键字被用作值的地方(例如

<color>关键字被**更准确地描述为标识符，**一个CSS可以理解的特殊值。因此它们没有使用引号括起来——它们不被当作字符串。

引号括起来就是被当做是字符串。

函数

函数是一段可重用的代码，可以多次运行，以完成重复的任务，对开发人员和计算机都是如此

**rgb()、**

**hsl()、**

**url()、**

calc()

**在CSS中调整大小**

[原始尺寸，或固有尺寸](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Sizing_items_in_CSS#%E5%8E%9F%E5%A7%8B%E5%B0%BA%E5%AF%B8%EF%BC%8C%E6%88%96%E5%9B%BA%E6%9C%89%E5%B0%BA%E5%AF%B8)

在受CSS设置影响之前，HTML元素有其原始的尺寸。一个直观的例子就是图像。一副图像的长和宽由这个图像文件自身确定。这个尺寸就是固有尺寸。

[设置具体的尺寸](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Sizing_items_in_CSS#%E8%AE%BE%E7%BD%AE%E5%85%B7%E4%BD%93%E7%9A%84%E5%B0%BA%E5%AF%B8)

 当给元素指定尺寸（然后其内容需要适合该尺寸）时，我们将其称为**外部尺寸**。

[使用百分数](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Sizing_items_in_CSS#%E4%BD%BF%E7%94%A8%E7%99%BE%E5%88%86%E6%95%B0)

百分数是以包含盒子的块为根据解析的。如果我们的<div>没有被指定百分数的值，那么它会占据100%的可用空间，因为它是块级别的元素。如果我们给了它一个百分数作为宽度，那么这就是它原来情况下可以占据空间的百分数。

[把百分数作为内外边距](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Sizing_items_in_CSS#%E6%8A%8A%E7%99%BE%E5%88%86%E6%95%B0%E4%BD%9C%E4%B8%BA%E5%86%85%E5%A4%96%E8%BE%B9%E8%B7%9D)

margins和padding设置为百分数的话，值是以**内联尺寸**进行计算的

[min-和max-尺寸](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Sizing_items_in_CSS#min-%E5%92%8Cmax-%E5%B0%BA%E5%AF%B8)

CSS给定一个元素的最大或最小尺寸。有一个包含了变化容量的内容的盒子，让它至少有个确定的高度，给它设置一个[min-height](https://developer.mozilla.org/zh-CN/docs/Web/CSS/min-height)属性。盒子一直保持大于最小高度，如果有比这个盒子在最小高度状态下所能容纳的更多内容，那么盒子就会变大。

百分比可以让图片可响应式,在更小的设备上浏览的时候，它们会合适地缩放。

[视口单位](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Sizing_items_in_CSS#%E8%A7%86%E5%8F%A3%E5%8D%95%E4%BD%8D)

视口，即你在浏览器中看到的部分页面，有尺寸的。视口尺寸相关的度量单位，即意为视口宽度的

vw单位，以及意为视口高度的vh单位。

1vh等于视口高度的1%，1vw则为视口宽度的1%.你可以用这些单位约束盒子的大小，还有文字的大小。

如果你改变了vh和vw的对应值，盒子和字体的大小也会改变；视口大小的变化也会让它们的大小变化，因为它们是依照视口来定大小的。

[替换元素](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Images_media_form_elements#%E6%9B%BF%E6%8D%A2%E5%85%83%E7%B4%A0)

图像和视频被描述为替换元素。 这意味着CSS不能影响这些元素的内部布局-仅影响它们在页面上于其他元素中的位置。

某些替换元素（例如图像和视频）也被描述为具有宽高比。 这意味着它在水平（x）和垂直（y）尺寸上均具有大小，并且默认情况下将使用文件的固有尺寸进行显示。

一个常用的技术是将一张图片的[max-width](https://developer.mozilla.org/zh-CN/docs/Web/CSS/max-width)设为100%。允许图片尺寸上小于但不大于盒子。技术也会对其他替换元素[<video>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/video)，或者[<iframe>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/iframe)）起作用

[object-fit](https://developer.mozilla.org/zh-CN/docs/Web/CSS/object-fit)属性:替换元素可以以多种方式被调整到合乎盒子的大小。

cover缩小了图像，维持了图像的比例

ontain作为值，图像将会缩放到足以放到盒子里面的大小。

fill值，图像充满盒子，不会维持比例。

[布局中的替换元素](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Images_media_form_elements#%E5%B8%83%E5%B1%80%E4%B8%AD%E7%9A%84%E6%9B%BF%E6%8D%A2%E5%85%83%E7%B4%A0)

在一个flex或者grid布局中，元素默认会把拉伸到充满整块区域。

图像不会拉伸，而是会被对齐到网格区域或者弹性容器的起始处。

只要记住替换元素在成为网格或者弹性布局的一部分时，有不同的默认行为，这很必要，避免了他们被布局奇怪地拉伸。

img {

width: 100%;

height: 100%;

}

[form元素](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Images_media_form_elements#form%E5%85%83%E7%B4%A0)

[继承和表单元素](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Images_media_form_elements#%E7%BB%A7%E6%89%BF%E5%92%8C%E8%A1%A8%E5%8D%95%E5%85%83%E7%B4%A0)

表单元素默认不会继承字体样式，因此如果你想要确保你的表单填入区域使用body中或者一个父元素中定义的字体，

你需要向你的CSS中加入这条规则：font-family : inherit;

button,

input,

select,

textarea {

font-family: inherit;

font-size: 100%;

box-sizing: border-box;

padding: 0; margin: 0;

}

textarea {

overflow: auto;

}

**备注**：通用样式表被许多开发者用作所有项目的一系列基础样式，典型就是那些做了和以上提到相似的事情的那些，在你开始自己的CSS作业前，它确保了跨浏览器的任何事情都被默认设定为统一样式。它们不像以往那么重要了，因为浏览器显著地要比以往更加统一。但是，如果你想要看一个例子，可以看看这个[Normalize.css](http://necolas.github.io/normalize.css/)，它被许多项目用作基础，是非常流行的样式表。

间距和布局

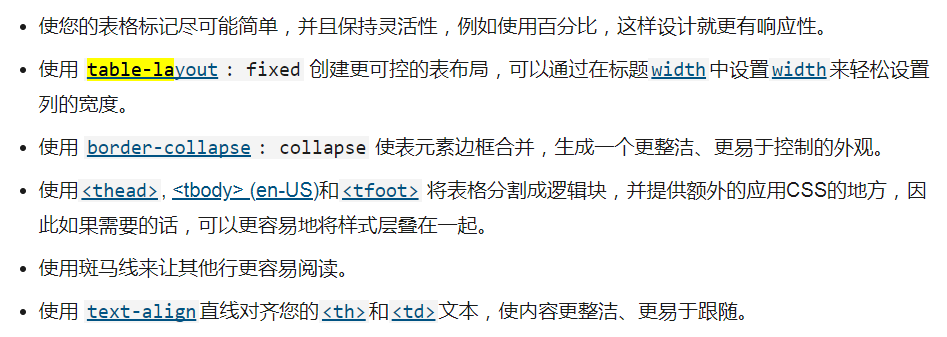
table-layout: fixed;

:nth-child

:nth-child选择器用于选择特定的子元素。它也可以用一个公式作为参数，来选择一个元素序列。公式

2n-1会选择所有奇数的子元素(1、3、5等)，而公式2n会选择所有偶数的子元素(2、4、6等等)。我们在代码中使用了

odd和even的关键字，这与前面提到的公式作用完全相同。在这里，我们给奇数行和偶数行不同的(醒目的)颜色。



**了解调试 CSS 工具**

[如何使用浏览器开发者工具](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Debugging_CSS#%E5%A6%82%E4%BD%95%E4%BD%BF%E7%94%A8%E6%B5%8F%E8%A7%88%E5%99%A8%E5%BC%80%E5%8F%91%E8%80%85%E5%B7%A5%E5%85%B7)

[比较 DOM 和 View Source](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Debugging_CSS#%E6%AF%94%E8%BE%83_dom_%E5%92%8C_view_source)

[HTML-dom调试详解1](https://developer.mozilla.org/zh-CN/docs/Tools/Page_Inspector/How_to/Examine_and_edit_HTML#html_tree)

[保持你的CSS整洁的技巧](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Organizing#%E4%BF%9D%E6%8C%81%E4%BD%A0%E7%9A%84css%E6%95%B4%E6%B4%81%E7%9A%84%E6%8A%80%E5%B7%A7)

**CSS代码风格规范**

[保持统一](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Organizing#%E4%BF%9D%E6%8C%81%E7%BB%9F%E4%B8%80)，[为你的CSS加注释](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Organizing#%E4%B8%BA%E4%BD%A0%E7%9A%84css%E5%8A%A0%E6%B3%A8%E9%87%8A) ：在你的样式表里面的逻辑段落之间，加入一块注释，是个好技巧。

[MDN的CSS代码规范的示例。 (en-US)](https://developer.mozilla.org/en-US/docs/MDN/Guidelines/Code_guidelines/CSS).

公共样式

有将会广泛生效的样式。典型地，你可以为以下的元素设定规则；[在你的样式表里面加入逻辑段落](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Organizing#%E5%9C%A8%E4%BD%A0%E7%9A%84%E6%A0%B7%E5%BC%8F%E8%A1%A8%E9%87%8C%E9%9D%A2%E5%8A%A0%E5%85%A5%E9%80%BB%E8%BE%91%E6%AE%B5%E8%90%BD)

/\* || General styles \*/

...

/\* || Typography \*/

...

/\* || Header and Main Navigation \*/

...

BEM命名

BEM即为块级元素修饰字符（Block Element Modifier）：[BEM命名常规](http://getbem.com/naming/)

BEM在大些的Web项目中被广泛使用，许多人用这种方式写他们的CSS。你可能会在没有提及为何CSS如此布局的时候，遇到BEM语法的例子，甚至是在教程中阅读[BEM 101](https://css-tricks.com/bem-101/) 中关于CSS Tricks的段落以了解更多和这个系统相关的信息。

[CSS的构建体系](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Organizing#css%E7%9A%84%E6%9E%84%E5%BB%BA%E4%BD%93%E7%B3%BB)

预处理工具和后处理工具

预处理工具以你的原文件为基础运行，将它们转化为样式表；后处理工具使用你已完成的样式表，然后对它做点手脚——也许是优化它以使它加载得更快。

使用这些工具的任何一部分都需要你的开发环境能运行进行处理工作的脚本。许多代码编辑器能为你做这项工作，或者你也可以安装一个命令行工具来辅助工作。

最为流行的预处理工具是[Sass](https://sass-lang.com/)，这里不是Sass的教程，所以我只会简要地解释一些Sass能做的事情，在组织的时候真的会帮到你，即使你没有用到Sass的任何其他的独特功能。

定义变量

CSS现在有原生的[自定义属性](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Using_CSS_custom_properties)。公共变量

编译组件样式表

一种组织CSS的方式是将样式表分成小的样式表。在使用Sass时，你可以在另一个层次上理解，然后得到许多小样式表——甚至到了每个组件都有一个独立样式表的地步。使用Sass中的include功能，这些都可以被编译为一个、或者少数几个真正链接到你的网站的样式表。

一个简单的尝试Sass的方式是使用[CodePen](https://codepen.io/)，你可以为一个Pen在设置中启用用于你的CSS的Sass，CodePen将会为你运行Sass语法分析功能，这样你就能看到应用了普通CSS的生成网页。有时你会发现CSS教程在它们的CodePen演示里面使用了Sass而不是普通CSS，所以了解一点点关于Sass的事情是有用的。

后处理以进行优化

如果你对加入例如许多额外的注释和空格，增大你的样式表大小有所关心的话，那么后处理会通过在生产版本中略去任何不必要的东西的方式，优化CSS。后处理解决方案是[cssnano](https://cssnano.co/)。

1.基础文字与样式化：[CSS中的文字样式涉及什么？](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Styling_text/Fundamentals#css%E4%B8%AD%E7%9A%84%E6%96%87%E5%AD%97%E6%A0%B7%E5%BC%8F%E6%B6%89%E5%8F%8A%E4%BB%80%E4%B9%88%EF%BC%9F)

2.样式列表：[一个简单的例子](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Styling_text/Styling_lists#%E4%B8%80%E4%B8%AA%E7%AE%80%E5%8D%95%E7%9A%84%E4%BE%8B%E5%AD%90)

3.样式化链接：[链接状态](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Styling_text/Styling_links#%E9%93%BE%E6%8E%A5%E7%8A%B6%E6%80%81)

[4.Web 字体](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Styling_text/Web_fonts#web_%E5%AD%97%E4%BD%93)