

网易微专业之《前端开发工程师》

学习笔记

开始时间：2015.12.17

《页面制作》

CSS（一）

CSS 简介

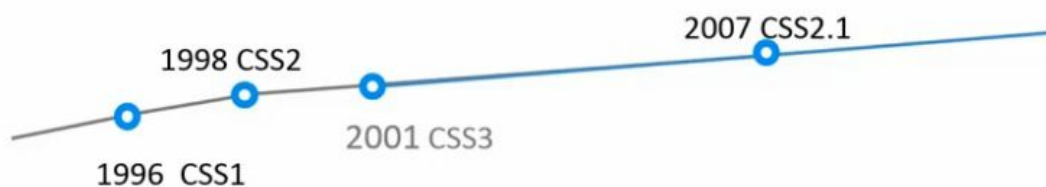
概念：

CSS 全称为“层叠样式表 (Cascading Style Sheets)”，它主要是用于定义 HTML 内容在浏览器内的显示样式，如文字大小、颜色、字体加粗等。

- CSS 指层叠样式表 (**C**ascading **S**tyle **S**heets)
- 样式定义**如何显示** HTML 元素
- 样式通常存储在**样式表**中
- 把样式添加到 HTML 4.0 中，是为了解决**内容与表现分离的问题**
- **外部样式表**可以极大提高工作效率
- 外部样式表通常存储在 **CSS 文件**中

多个样式定义可**层叠**为一

历史



CSS 引用：

在 HTML 中引入 CSS 共有 3 种方式：

- **外部样式表；**
- **内部样式表；**
- **内联样式表；**

1. 外部样式表

当样式需要应用于很多页面时，外部样式表将是理想的选择。在使用外部样式表的情况下，你可以通过改变一个文件来改变整个站点的外观。每个页面使用 `<link>` 标签链接到样式表。`<link>` 标签在（文档的）头部：

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

浏览器会从文件 `mystyle.css` 中读到样式声明，并根据它来格式文档。

外部样式表可以在任何文本编辑器中进行编辑。文件不能包含任何的 `html` 标签。样式表应该以 `.css` 扩展名进行保存。下面是一个样式表文件的例子：

```
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("/images/back40.gif");}
```

2. 内部样式表：

当单个文档需要特殊的样式时，就应该使用内部样式表。你可以使用 `<style>` 标签在文档头部定义内部样式表，就像这样：

```
<head>
<style>
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("/images/back40.gif");}
</style>
</head>
```

3. 内联样式表：

要使用内联样式，你需要在相关的标签内使用样式（`style`）属性。`Style` 属性可以包含任何 `CSS` 属性。本例展示如何改变段落的颜色和左外边距：

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

由于要将表现和内容混杂在一起，内联样式会损失掉样式表的许多优势。请慎用这种方法，例如当样式仅需要在一个元素上应用一次时。

层叠次序

当同一个 `HTML` 元素被不止一个样式定义时，会使用哪个样式呢？

一般而言，所有的样式会根据下面的规则层叠于一个新的虚拟样式表中，其中数字 4 拥有最高的优先权。

1. 浏览器缺省设置

2. 外部样式表
3. 内部样式表（位于 <head> 标签内部）
4. 内联样式（在 HTML 元素内部）

因此，**内联样式（在 HTML 元素内部）拥有最高的优先权**，这意味着它将优先于以下的样式声明：

内联样式 > 标签中的样式声明 > 外部样式表中的样式声明（或者浏览器中的样式声明（缺省值））

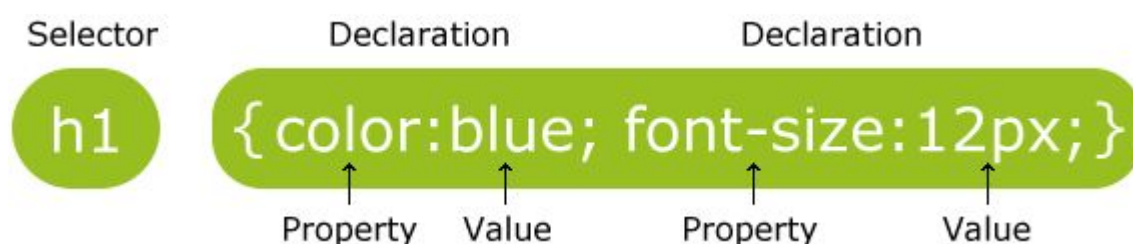
提示：如果你使用了外部文件的样式在 <head>中也定义了该样式，则内部样式表会取代外部文件的样式。

其实总结来说，就是--**就近原则（离被设置元素越近优先级别越高）**。

但注意上面所总结的优先级是有一个前提：内联式、嵌入式、外部式样式表中 css 样式是在的**相同权值**的情况下。

CSS 语法

CSS 规则由两个主要的部分构成：选择器，以及一条或多条声明：



选择器 (selector) 通常是您需要改变样式的 HTML 元素。

每条声明 (Declaration) 由一个属性 (property) 和一个值 (value) 组成。在英文大括号 “{ }” 中的的就是声明。

属性 (property) 是您希望设置的样式属性 (style attribute)。每个属性有一个值。属性和值被冒号分开。

浏览器私有属性

由于 CSS3 很多属性尚未成为 W3C 标准的一部分，因此每种内核的浏览器都只能识别带有自身私有前缀的 CSS3 属性。我们在书写 CSS3 代码时，需要在属性前加上浏览器的私有前缀，然后该种内核的浏览器才能识别相应的 CSS3 属性。

CSS3 浏览器私有前缀

私有前缀	对应的浏览器
<code>-webkit-</code>	chrome 和 safari
<code>-moz-</code>	Firefox
<code>-ms-</code>	IE
<code>-o-</code>	opera

举个例子，假如我们想要使用 CSS3 实现半径为 10px 的圆角效果的话，可能会这样写：

```
1border-radius:10px;
```

但是并非所有浏览器都能识别 `border-radius` 这个属性，例如 chrome 浏览器只能识别 `-webkit-border-radius`（前缀为 `-webkit-`），而 firefox 浏览器只能识别 `-moz-border-radius`（前缀为 `-moz-`）。因为为了让主流浏览器都能实现圆角效果，

我们需要这样写：

```
1border-radius:10px;
2-webkit-border-radius:10px; /*兼容 chrome 和 Safari*/
3-moz-border-radius:10px; /*兼容 Firefox*/
4-ms-border-radius:10px; /*兼容 IE*/
5-o-border-radius:10px; /*兼容 opera*/
```

CSS 属性值定义语法：

- margin : [<length> | <percentage> | auto] {1,4}

基本元素

组合符号

数量符号

CSS 属性值定义语法（CSS value definition syntax）是用来限定 CSS 属性合法取值的专门语法。在此基础之上，一个 CSS 属性的合法取值也由语义所限制，比如一些数字必须是正数。

CSS 属性值定义语法描述了哪些值是可取的 CSS 属性，基本组成元素包括关键字、符号与带类型的参数。

基本组成元素

1. 关键字

- **一般关键字：**一般关键字都有预先的定义，不需要引号，如 `auto`, `smaller` 或 `ease-in`。
- **特殊关键字：inherit 与 initial：**所有 CSS 属性值都可以使用 `inherit` 或者 `initial`。这两个关键字不会出现在 CSS 合法值定义中，但都是隐含可用的。

2. 符号

CSS 中，有一些符号是可以出现的，比如斜杠（`/`）或者逗号（`,`）等。它们用来分隔属性值：逗号用来分隔数个并列值，或者分隔函数的参数；斜杠用来分隔一个值的多个部分，通常用在 CSS 缩写中。这两种符号会以其字面意义出现在 CSS 属性值定义中。

3. 带类型的参数

- **基本类型：**一些类型在 CSS 中经常出现，CSS 规范中将其专门定义，称为基本类型，用一对尖括号表示：'`<`'与'`>`'，例如：`<length>`, `<percentage>`, `<color>` ...
- **其他类型：**其他类型同样也用一对尖括号表示：'`<`'与'`>`'。

其他类型分为两种：

1. 取值与其属性取值范围一致的类型，使用包含在尖括号中的带有引号的属性名表示（例如 `<'border-top-color'>` 等）
2. 取值不同于其属性的类型，使用包含在尖括号中的属性名表示（例如 `<margin-width>`、`<border-width>` 等）

组合符号：

值可以按照下面的方法组织，类似于正则表达式

1. 几个并列的词(词中间用空格隔开)：**并列的词必须全部出现**，次序和列出时一样
2. 两个与符号 `&&` 分割两个或多个选择：**必须全部出现的**，次序任意
3. 双竖线 `||` 分割两个或多个选择：**其中一个或多个出现**，次序任意
4. 竖线 `|` 分割两个或多个选择：**其中只有一个出现**

5. 方框[]用来分组

方框优先于并列，并列优先于两个与符号&&，两个与符号&&优先于双竖线||，双竖线||优先于竖线|。因此，下面两行的含义是一样的：

```
a b | c || d && e f  
[ a b ] | [ c || [ d && [ e f ] ] ]
```

1. 并置（也就是上面的组合符号 1）是指将数个关键字、符号或类型，用空格分开写在一起。并置中所有的元素都必须出现并且按所规定的顺序出现。

例如：<'font-size'> <'font-family'>

合法值：12px Arial

不合法值：

- 2em --两个元素必须出现，此处少一个值；
- Arial 14px --顺序错误。

2. && ---与组合符

“与”组合符连接的各个部分都必须出现，但是顺序任意。例如：<length>&&<color>

合法值：

- Green 2px
- 1em blue

不合法值：

- Blue --两个元素必须都出现，此处少一个。
- 20px

3. || ---或组合符

“或”组合符表示其连接的所有组成元素是可选的，次序任意，但是至少其中一个要出现。

“或”组合符通常用来描述属性缩写中的各部分。

如：<'border-width'> || <'border-style'> || <'border-color'>

以下均为该例的合法取值：

- 1em solid blue
- blue 1em
- solid 1px yellow

但以下不是合法取值：

- blue yellow 因为一个组成部分最多出现一次。
- bold 因为它不允许出现。

4. | ---互斥组合符

“互斥”组合符表示各组成部分中只能恰好出现一个，通常用来分隔一个属性的所有可选值。

例如：<percentage> | <length> | left | center | right | top | bottom

以下均为该例的合法取值：

- 3%
- 0

- 3.5em
- left
- center
- right
- top
- bottom

但以下不是合法取值：

- center 3% ---只能出现一个，这里出现两个。
- 3em 4.5em

5. [] ---方框

方框将数个基本元素组成一个整体，用来强调组合的优先级。例如：

`bold [thin && <length>]`

以下均为该例的合法取值：

- bold thin 2vh
- bold 0 thin
- bold thin 3.5em

但以下不是合法取值：

`thin bold 3em` 因为 `bold` 被放置在方括号定义的整体之中。根据定义，`bold` 必须出现在方括号定义的整体之前。

数量符号：

1. 无：

数量符号用来描述一个元素可以出现多少次。若不标注，则这个元素比如恰好出现一次。注意数量描述符不能叠加出现，并且优先级最高。

例如：`<length>`

以下为合法值：

- 1px
- 10em

单以下不是合法取值：

`1px 2px` ---此处只能出现一次

2. 加号 (+)

加号表示可以出现一次或多次。例如：

`<color-stop>[,<color-stop>]+`

以下均为该例的合法取值：

- #fff, red
- Blue, green 50%, gray

但以下不是合法取值：

`red`

3. 问号 (?)

问号表示可选，即出现零次或一次。

例如：inset?&&<color>

以下均为该例的合法取值：

- Inset blue
- Red

但以下是不合法值：

- Inset blue red
- Inset

4. 大括号 ({ })

大括号包含两个以逗号分隔的整数 A 与 B，表示最少出现 A 次，且最多出现 B 次。例如：

bold smaller{1,3}

以下均为该例的合法取值：

- bold smaller
- bold smaller smaller
- bold smaller smaller smaller

但以下不是合法取值：

- bold 因为 smaller 至少要出现一次。
- bold smaller smaller smaller smaller 因为 smaller 最多出现三次。
- smaller 因为 bold 是并置关系，必须出现在 smaller 之前。

5. 星号 (*)

星号表示可以出现零次（即不出现），一次，或任意多次。例如：

bold smaller*

以下均为该例的合法取值：

- bold
- bold smaller
- bold smaller smaller
- bold smaller smaller smaller and so on.

但以下不是合法取值：

- smaller 因为 bold 并置于 smaller，必须出现在任何 smaller 之前。

6. 井号 (#)

井号表示可以出现一次或多次，与加号相似。但是其多次出现必须以逗号分隔。例如：

bold smaller#

以下均为该例的合法取值：

- bold smaller
- bold smaller, smaller
- bold smaller, smaller, smaller and so on.

但以下不是合法取值：

- bold 因为 smaller 必须至少出现一次。
- bold smaller smaller smaller 因为多个 smaller 必须以逗号分隔。
- smaller 因为 bold 是并置关系，必须出现在 smaller 之前。

属性值列子



- padding-top: <length>|<percentage>
`padding-top:1px;`
`padding-top:1em 5%;`
- border-width:[<length> | thick | medium | thin]{1,4}
`border-width:2px;`
`border-width:2px small;`
- box-shadow: [inset? && [<length>{2,4} && <color>?]]#|none
`box-shadow:3px 3px rgb(50%, 50%, 50%), red 0 0 4px inset;`
`box-shadow:inset 2px 4px, 2px blue;`

(绿色的为合法值，红色的为不合法值。)

总结：

符号	名称	描述	示例
组合符号			
	并置	各部分必须出现且按顺序出现	<code>solid <length></code>
<code>&&</code>	“与”组合符	各部分必须出现，但可以不按顺序	<code><length> && <string></code>
<code> </code>	“或”组合符	各部分至少出现一个，可以不按顺序	<code><'border-image-outset'> <'border-image-slice'></code>
<code> </code>	“互斥”组合符	各部分恰好出现一个	<code>smaller small normal big bigger</code>
<code>[]</code>	方括号	强调优先级	<code>bold [thin && <length>]</code>
数量符号			
	无数量符号	恰好一次	<code>solid</code>
<code>*</code>	星号	零次、一次或多次	<code>bold smaller*</code>
<code>+</code>	加号	一次或多次	<code>bold smaller+</code>
<code>?</code>	问号	零次或一次（即可选）	<code>bold smaller?</code>
<code>{A,B}</code>	大括号	至少 A 次，至多 B 次	<code>bold smaller{1,3}</code>
<code>#</code>	井号	一次或多次，但多次出现必须以逗号分隔	<code>bold smaller#</code>

CSS 属性值定义参考资源：

https://developer.mozilla.org/zh-CN/docs/Web/CSS/Value_definition_syntax

http://www.dreamdu.com/css/property_value/

@规则及语法

1. @media :

指定样式表规则用于指定的媒体类型和查询条件。

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

IE8 及以下只能实现 CSS2 中的部分，即只可以设置媒体类型。

媒体查询可以被用在 CSS 中的 `@media` 和 `@import` 规则上，也可以被用在 HTML 和 XML 中。

示例代码：

```
@media screen and (width:800px) { ... }

@import url(example.css) screen and (width:800px);
```

2. @keyframes

指定动画名称和动画效果。

定义动画时，简单的动画可以直接使用关键字 **from** 和 **to**，即从一种状态过渡到另一种状态：

示例代码：

```
@keyframes testanimations {

    from { opacity: 1; }

    to { opacity: 0; }

}
```

其中 **testanimations** 是该动画的名字，该动画表示某个东西将逐渐消失。

如果复杂的动画，可以混合 `<percentage>` 去设置某个时间段内的任意时间点的样式：

示例代码：

```
@keyframes testanimations {

    from { transform: translate(0, 0); }

    20% { transform: translate(20px, 20px); }

    40% { transform: translate(40px, 0); }

    60% { transform: translate(60px, 20); }

    80% { transform: translate(80px, 0); }

    to { transform: translate(100px, 20px); }

}
```

当然，也可以不使用关键字 **from** 和 **to**，而都使用 `<percentage>`，如将上面的示例代码中的 **“from”**、**“to”** 分别换成 **“0%”**、**“100%”**。

3. @font-face

设置嵌入 HTML 文档的字体。

需要兼容当前的主流浏览器，需同时使用 TureTpe(.ttf)、Web Open Font Format(.woff)、Embedded Open Type(.eot)、SVG(.svg)四种字体格式。

嵌入 HTML 文档的字体是指将 OpenType 字体（压缩的 TrueType 字体）文件映射到客户端系统，用来提供 HTML 文档使用该字体，或取代客户端系统已有的同名字体。即让客户端显示客户端所没有安装的字体。

.eot(Embedded Open Type)为 IE 的私有字体格式。Safari3.1 开始支持.ttf(TrueType)和.otf(OpenType)。

未来.woff(Web Open Font Format)将会取代.ttf(TrueType)和.otf(OpenType)两种字体格式。

示例：使用一个全浏览器兼容的自定义字体

代码如下：

```
@font-face {
    font-family: 'diyfont';

    src: url('diyfont.eot'); /* IE9+ */

    src: url('diyfont.eot?#iefix')
format('embedded-opentype'), /* IE6-IE8 */

        url('diyfont.woff') format('woff'), /*
chrome、firefox */

        url('diyfont.ttf') format('truetype'), /*
chrome、firefox、opera、Safari, Android, iOS 4.2+*/

        url('diyfont.svg#fontname') format('svg');
/* iOS 4.1- */
}
```

你需要同时提供 4 种格式的字体

另外，还有其他几种 @ 规则，如 @import/@charset/@namespace/@page/

@supports/@document 等，这几类规则用的比较少，具体可以查看 CSS 参考手册：

<http://www.css88.com/book/css/>

CSS（二）

CSS 选择器

选择器，说白了就是**用一种方式把你想要的那一个标签选中**！把它选中了，你才能操作这个标签的 CSS 样式。CSS 有很多把你所需要的标签选中的方式，这些不同的方式就是不同的选择器。

选择器的不同，在于它选择方式不同，但是他们的目的都是相同的，那就是**把你需要的标签选中**，然后让你定义该标签的 CSS 样式。当然，你也有可能用某一种选择器代替另一种选择器，这仅仅是由于选择方式不一样罢了，目的还是一样的。

标签选择器

标签选择器其实就是 html 代码中的标签。例如下面代码：

```
p{font-size:12px;line-height:1.6em;}
```

上面的 css 样式代码的作用：为 p 标签设置 12px 字号，行间距设置 1.6em 的样式。

类选择器

类选择器在 css 样式编码中是最常用到的。

语法：

. 类选器名称 {css 样式代码;}

注意：

- 1、英文圆点开头
- 2、其中类选器名称可以任意起名（但不要起中文噢）

使用方法：

第一步：使用合适的标签把要修饰的内容标记起来，如下：

```
<span>胆小如鼠</span>
```

第二步：使用 class="类选择器名称" 为标签设置一个类，如下：

```
<span class="stress">胆小如鼠</span>
```

第三步：设置类选器 css 样式，如下：

```
.stress{color:red;} /*类前面要加入一个英文圆点*/
```

ID 选择器

在很多方面，ID 选择器都类似于类选择符，但也有一些重要的区别：

- 1、为标签设置 id="ID 名称"，而不是 class="类名称"。
- 2、ID 选择符的前面是井号（#）号，而不是英文圆点（.）。

类和 ID 选择器的区别

相同点：可以应用于任何元素

不同点：

- 1、ID 选择器只能在文档中使用一次。与类选择器不同，在一个 HTML 文档中，ID 选择器只能使用一次，而且仅一次。而类选择器可以使用多次。
- 2、可以使用类选择器词列表方法为一个元素同时设置多个样式。我们可以为一个元素同时设多个样式，但只可以用类选择器的方法实现，ID 选择器是不可以的（不能使用 ID 词列表）。如以下代码：

```
<p>到了<span class="stress bigsize">三年级</span>下学期时，我们班上了一节公开课...</p>
```

通用选择器（通配符选择器）

通用选择器是功能最强大的选择器，它使用一个（*）号指定，它的作用是匹配 html 中所有标签元素，

如下使用下面代码使用 html 中任意标签元素字体颜色全部设置为红色：

```
* {color:red;}
```

属性选择器

选择器	描述
[attribute]	用于选取带有指定属性的元素。
[attribute=value]	用于选取带有指定属性和值的元素。
[attribute~value]	用于选取属性值中包含指定词汇的元素。
[attribute =value]	用于选取带有以指定值开头的属性值的元素，该值必须是整个单词。
[attribute^=value]	匹配属性值以指定值开头的每个元素。
[attribute\$=value]	匹配属性值以指定值结尾的每个元素。

<code>[attribute*=value]</code>	匹配属性值中包含指定值的每个元素。
---------------------------------	-------------------

实例展示：

html 代码：

```
<a href="xxx.pdf">我链接的是 PDF 文件</a>

<a href="#" class="icon">我类名是 icon</a>

<a href="#" title="我的 title 是 more">我的 title 是 more</a>
```

css 代码

```
a[class^=icon]{
    background: green;
    color:#fff;
}

a[href$=pdf]{
    background: orange;
    color: #fff;
}

a[title*=more]{
    background: blue;
    color: #fff;
}
```

结果显示：

我链接的是PDF文件 我类名是icon 我的title是more

伪类选择器

CSS 伪类用于向某些选择器添加特殊的效果。又叫伪类选择符，它允许给 html 不存在的标签（标签的某种状态）设置样式，比如说我们给 html 中一个标签元素的鼠标滑过的状态来设置字体颜色：

```
a:hover{color:red;}
```

上面一行代码就是为 a 标签鼠标滑过的状态设置字体颜色变红。

关于伪选择器：

关于伪类选择器，到目前为止，可以兼容所有浏览器的“伪类选择器”就是 a 标签上使用 :hover 了（其实伪类选择符还有很多，尤其是 css3 中，但是因为不能兼容所有浏览器，本教程只是讲了这一种最常用的）。其实 :hover 可以放在任意的标签上，比如说 p:hover，但是它们的兼容性也是很不好的，所以现在比较常用的还是 a:hover 的组合。

属性	描述	CSS
:active	向被激活的元素添加样式。	1
:focus	向拥有键盘输入焦点的元素添加样式。	2
:hover	当鼠标悬浮在元素上方时，向元素添加样式。	1
:link	向未被访问的链接添加样式。	1
:visited	向已被访问的链接添加样式。	1
:first-child	向元素的第一个子元素添加样式。	2
:lang	向带有指定 lang 属性的元素添加样式。	2

结构伪类选择器

是针对 HTML 层次“结构”的伪类选择器。例如我们想要某一个父元素下面的第 n 个子元素。

结构伪类选择器		
	选择器	说明
第一类	E:first-child	选择父元素的第一个子元素
	E:last-child	选择父元素的最后一个子元素
	E:nth-child (n)	选择父元素下的第 n 个元素或奇偶元素，n 的值为“数字 odd even”，
	E:only-child	选择父元素中唯一的子元素（该父元素只有一个子元素）
第二类	E:first-of-type	选择同类型的第 1 个同级兄弟元素
	E:last-of-type	选择同类型的最后 1 个同级兄弟元素
	E:nth-of-type (n)	选择同类型的第 n 个同级兄弟元素，n 的值为“数字 odd even”，
	E:only-of-type	选择父元素中特定类型的唯一子元素（该父元素有多个子元素）
第三类	:root	选择文档的根元素。在 HTML 中，根元素永远是 HTML
	E:not(selector)	选择某个元素之外的所有元素
	E:empty	选择一个不包含任何子元素的元素，注意文本节点也可以被看成子元素
	E:target	选取页面中的某个 target 元素

（odd:奇数； even:偶数）

总结：“:first-child” 是选择父元素下的第 1 个子元素（不区分元素类型），而

“:first-of-type” 是选择父元素下某个元素类型的第 1 个子元素（区分元素类型）。

例子：

```
1<div>
2    <h1></h1>
3    <p></p>
4    <span></span>
5    <span></span>
6</div>
```

(1) :first-child

h1:first-child: 选择的是 h1 元素，因为 h1 元素是 div 的第 1 个子元素。

p:first-child: 选择不到任何元素，因为 p 并不是 div 的第 1 个子元素，而是 div 的第 2 个子元素。

`span:first-child`:选择不到任何元素，因为 `span` 并不是 `div` 的第 1 个子元素，而是 `div` 的第 3 个子元素；

(2):first-of-type

`h1: first-of-type`:选择的是 `h1` 元素，因为 `h1` 元素是 `div` 中所有 `h1` 元素中的第 1 个 `h1` 元素，事实上也只有一个为 `h1` 的子元素；

`p: first-of-type`:选择的是 `p` 元素，因为 `p` 元素是 `div` 中所有 `p` 元素中的第 1 个 `p` 元素，事实上也只有一个为 `p` 的子元素；

`span: first-of-type`:选择的是 `id="first"` 的 `span` 元素，因为在 `div` 元素中有 2 个 `span` 元素，我们选择的是两个之中的第 1 个。

“:nth-child(n)” 选择器

用来定位某个父元素的一个或多个特定的子元素。其中“`n`”是其参数，而且可以是整数值(1, 2, 3, 4)，也可以是表达式($2n+1$ 、 $-n+5$)和关键词(`odd`、`even`)，但参数 `n` 的起始值始终是 1，而不是 0。也就是说，参数 `n` 的值为 0 时，选择器将选择不到任何匹配的元素。

经验与技巧:当“`:nth-child(n)`”选择器中的 `n` 为一个表达式时，其中 `n` 是从 0 开始计算，当表达式的值为 0 或小于 0 的时候，不选择任何匹配的元素。如下表所示：

n	$2n+1$	$4n+1$	$4n+4$	$4n$	$5n-2$	$-n+3$
0	1	1	4	-	-	3
1	3	5	8	4	3	2
2	5	9	12	8	8	1
3	7	13	16	12	13	0
4	9	17	20	16	18	-1
5	11	21	24	20	23	-2

`:nth-last-child(n)`

“`:nth-last-child(n)`”选择器和前面的“`:nth-child(n)`”选择器非常的相似，只是这里多了一个“`last`”，所起的作用和“`:nth-child(n)`”选择器有所区别，从某父元素的最后一个子元素开始计算，来选择特定的元素。

:target 选择器

用于选取页面中的某个 target 元素。

:target 选择器称为目标选择器，用来匹配文档(页面)的 url 的某个标志符的目标元素。常用于[锚链接](#)，类似于[书签定位](#)。

```
<style type="text/css">
    :target h3
    { color:red;}
</style>
</head>
<body>
    <div>
        <a href="#music">推荐音乐</a><br />
    </div>
    .....
    <div id="music">
        <h3>推荐音乐</h3>
        <ul>
            <li>林俊杰-被风吹过的下图</li>
            <li>曲婉婷-在我的歌声里</li>
            <li>许嵩-灰色头像</li>
        </ul>
    </div>
```

1. 当设为 :target h3 时,会定位到 链接 #music 所指向的目标元素 id=music 这个 div 下的 h3 标签,使 h3 变为红色
2. 当设为 :target 时,会定位到链接#music 所指向的 id=music 这个 div 下,使 h3/ul/li 标签内容全变为红色。
3. 当设为 :target li 时,会定位到链接#music 所指向的 id=music 这个 div 下,使 li 标签内容全变为红色。

UI 元素状态伪类选择器

UI，是用户界面（User Interface）的意思。所谓的 UI 设计是指对软件的人机交互、操作逻辑、界面美观的综合设计。

UI 元素状态包括：可用、不可用、选中、未选中、获取焦点、失去焦点等。

这些选择器的共同特征是：指定的样式只有当元素处于某种状态下时才起作用，在默认状态下不起作用。UI 元素状态伪类选择器大多数都是针对表单元素来使用的。

CSS3 UI 元素状态伪类选择器

选择器	说明
E:focus	指定元素获得光标焦点时使用的样式
E:checked	选择 E 元素中所有被选中的元素

CSS3 UI 元素状态伪类选择器

选择器	说明
E::selection	改变 E 元素中被选择的网页文本的显示效果
E:enabled	选择 E 元素中所有“可用”元素
E:disabled	选择 E 元素中所有“不可用”元素
E:read-write	选择 E 元素中所有“可读写”元素
E:read-only	选择 E 元素中所有“只读”元素
E::before	在 E 元素之前插入内容
E::after	在 E 元素之后插入内容

:focus 选择器

:focus 选择器被用来指定“表单元素”获得光标焦点时使用的样式，主要在单行文本框 text、多行文本框 textarea 等表单元素获得焦点并输入文本时使用。

:checked 选择器

在表单元素中，单选按钮 radio 和复选框 checkbox 都具有“选中”和“未选中”状态。在 CSS3 中，我们可以通过使用:checked 选择器来定义选中时的 CSS 样式。

暂时只有 Opera 浏览器支持:checked 选择器。

::selection 选择器

在 CSS3 中，我们可以使用“::selection 选择器”来改变被选择的网页文本的显示效果。

注意，“::selection 选择器”是双冒号。其实**双冒号往往都是“伪元素”，而单冒号往往都是“伪类”**。

:enabled 与:disabled 选择器

在 Web 表单中，有些表单元素（如输入框、密码框、复选框等）有“可用”和“不可用”这 2 种状态。默认情况下，这些表单元素都处在可用状态。

在 CSS3 中，我们可以使用:enabled 选择器和:disabled 选择器来分别设置表单元素的可用与不可用这两种状态的 CSS 样式。

:read-write 与:read-only 选择器

在 Web 表单中，有些表单元素（如输入框、文本域等）有“可读写”和“只读”这 2 种状态。默认情况下，这些表单元素都处在“可读写”状态。

在 CSS3 中，我们可以使用:read-write 选择器和:read-only 选择器来分别设置表单元素:read-only 选择器支持 Chrome, Safari 和 Opera。Firefox 和 Internet Explorer 不支持 :read-only 选择器。Firefox 支持 :-moz-read-only 选择器作为替代方案。的“可读写”与“只读”这两种状态的 CSS 样式。

::before 和::after 选择器

在 CSS3 中，我们可以使用::before 和::after 这两个选择器在元素前面或后面添加内容。

这两个选择器常和“content 属性”配合使用，使用的场景最多的就是清除浮动和创建小图

标。

CSS 伪元素选择器

用于向某些选择器设置特殊效果。

属性	描述	CSS
<code>::first-letter</code>	向文本的第一个字母添加特殊样式。	1
<code>::first-line</code>	向文本的首行添加特殊样式。	1
<code>::before</code>	在元素之前添加内容。	2
<code>::after</code>	在元素之后添加内容。	
<code>::selection</code>	改变元素中被选择的网页文本的显示效果	

CSS 组合选择器

子选择器

即大于符号(>),用于选择指定标签元素的**第一代子元素**。如右侧代码编辑器中的代码：

```
.food > li{border:1px solid red;}
```

这行代码会使 class 名为 food 下的子元素 li 加入红色实线边框。

包含(后代)选择器

包含选择器，即加入**空格**,用于选择指定标签元素下的**后辈元素**。如右侧代码编辑器中的代码：

```
.first span{color:red;}
```

请注意这个选择器与子选择器的区别，**子选择器 (child selector)** 仅是指它的直接后代，或者你可以理解为作用于子元素的第一代后代。而后代选择器是作用于所有子后代元素。后代选择器通过空格来进行选择，而子选择器是通过“>”进行选择。

总结：> 作用于元素的**第一代**后代，**空格**作用于元素的**所有**后代。

相邻选择器

相邻选择器只会命中符合条件的**相邻**的兄弟元素，使用“+”号。

如：p+p{color:#f00;}

即设置与 p 元素相邻的 p 元素的颜色。

兄弟选择器

兄弟选择器会命中所有符合条件的兄弟元素，而不强制是紧邻的元素，使用“~”符号

如：p~p{color:#f00;}

群组选择器

群组选择器，就是同时对几个选择器进行相同的操作。

语法：h3,div,p,span{color:red;}

CSS 继承

CSS 继承

CSS 的某些样式是具有继承性的，那么什么是继承呢？继承是一种规则，它允许样式不仅应用于某个特定 html 标签元素，而且应用于其后代。比如下面代码：如某种颜色应用于 p 标签，这个颜色设置不仅应用 p 标签，还应用于 p 标签中的所有子元素文本，这里子元素为 span 标签。

```
p{color:red;}
```

```
<p>三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>
```

注意: 有一些 css 样式是不具有继承性的。如 border:1px solid red;

```
p{border:1px solid red;}
```

```
<p>三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>
```

在上面例子中它代码的作用只是给 p 标签设置了边框为 1 像素、红色、实心边框线，而对于子元素 span 是没用起到作用的。

CSS 优先级（特殊性）

CSS 的优先级也可以称为 CSS 的特殊性 (specificity) 或权值：**对于每个样式表规则, 浏览器**

都会计算选择器的特殊性,从而使元素属性声明在有冲突的情况下能够正确显示.

有的时候我们为同一个元素设置了不同的CSS样式代码,那么元素会启用哪一个CSS样式呢?我们来看一下下面的代码:

```
p{color:red;}
```

```
.first{color:green;}
```

```
<p class="first">三年级时,我还是一个<span>胆小如鼠</span>的小女孩。</p>
```

p 和 .first 都匹配到了 p 这个标签上,那么会显示哪种颜色呢? green 是正确的颜色,那么为什么呢? 是因为浏览器是根据权值来判断使用哪种 css 样式的,权值高的就使用哪种 css 样式。

下面是权值的规则:

标签的权值为 1, 类选择符的权值为 10, ID 选择符的权值最高为 100。例如下面的代码:

```
p{color:red;} /*权值为 1*/
```

```
p span{color:green;} /*权值为 1+1=2*/
```

```
.warning{color:white;} /*权值为 10*/
```

```
p span.warning{color:purple;} /*权值为 1+1+10=12*/
```

```
#footer .note p{color:yellow;} /*权值为 100+10+1=111*/
```

注意: 还有一个权值比较特殊—继承也有权值但很低,有的文献提出它只有 0.1, 所以可以理解为继承的权值最低。

如何改变 CSS 的优先级?

1. 改变 CSS 的先后顺序;
2. 提升权重;
3. 使用 !important.

CSS 层叠

层叠就是在 html 文件中对于同一个元素可以有多个 css 样式存在, 当有相同权重的样式存在时, 会根据这些 css 样式的前后顺序来决定, 处于最后面的 css 样式会被应用。

如下面代码:

```
p{color:red;}
```

```
p{color:green;}
```

```
<p class="first">三年级时,我还是一个<span>胆小如鼠</span>的小女孩。</p>
```

最后 p 中的文本会设置为 green, 这个层叠很好理解, 理解为后面的样式会覆盖前面的样式。

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

所以前面的 css 样式优先级就不难理解了：

内联样式表（标签内部） > 嵌入样式表（当前文件中） > 外部样式表（外部文件中）。

重要性(!important)

我们在做网页代码的时，有些特殊的情况需要为某些样式设置具有最高权值，怎么办？这时候我们可以使用 **!important** 来解决。

如下代码：

```
p{color:red !important;}
```

```
p{color:green;}
```

```
<p class="first">三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>
```

这时 p 段落中的文本会显示的 red 红色。

注意：**!important 要写在分号的前面**

这里注意当网页制作者不设置 css 样式时，浏览器会按照自己的一套样式来显示网页。并且用户也可以在浏览器中设置自己习惯的样式，比如有的用户习惯把字号设置为大一些，使其查看网页的文本更加清楚。这时注意样式优先级为：

浏览器默认的样式 < 网页制作者样式 < 用户自己设置的样式

但记住!important 优先级样式是个例外，权值高于用户自己设置的样式。

参考资料：

<http://www.quirksmode.org/css/selectors/>

CSS（三）

文本、颜色

CSS 字体属性

属性	描述
font	简写属性。作用是把所有针对字体的属性设置在一个声明中。
font-family	设置字体系列。
font-size	设置字体的尺寸。
font-style	设置字体风格。
font-variant	以小型大写字体或者正常字体显示文本。
font-weight	设置字体的粗细。

Font-family:

CSS 定义了 5 种通用字体系列：

- Serif 字体（衬线字体，如 Times New Rome）
- Sans-serif 字体（无衬线字体，如 Arial）
- Monospace 字体（等宽字体）
- Cursive 字体（草书字体, 手写字体）
- Fantasy 字体（相当于印刷学中的装饰体）

默认情况下，浏览器的字体是宋体。中文字体常用的有宋体、微软雅黑，英文字体比较常用的是 Times New Roman、Arial。

font-size 语法：

font-size: <absolute-size> | <relative-size> | <length> | <percentage>

<absolute-size> = xx-small | x-small | small | medium | large | x-large | xx-large

<relative-size> = smaller | larger

默认值：medium

Font-size 属性值可以有很多种方式：

- 使用关键字 (small/medium/large/x-small/xx-small/x-large/xx-large) ；
- 使用像素做单位的数值(如 20px),
- 使用 em、%等来做单位。

font-style 属性

font-style 属性值	说明
normal	默认值，正常体
italic	斜体，这是一个属性
oblique	将字体倾斜，将没有斜体变量(italic)的特殊字体，要应用 oblique

font-weight 属性

字体粗细值可以取关键字和 100~900 的数值。也可使用关键字。

font-weight 属性值	说明
normal	默认值，正常体
lighter	较细
bold	较粗
bolder	很粗（其实效果跟 bold 差不多）

字体粗细 font-weight 属性值可以取 100、200、...、900 这九个值。400 相当于正常字体 normal，700 相当于 bold。100~900 分别表示字体的粗细，是对字体粗细的一种量化方式，值越大就表示越粗，值越小就表示越细。

对于中文网页来说，一般仅用到 bold、normal 这两个属性值完全就可以了，粗细值不建议使用数值（100~900）。

Font-variant 属性

font-variant 属性值	说明
normal	默认值，正常效果
small-caps	小型大写字母的字体

Text-transform 属性 ---英文大小写转换

属性值	说明
none	默认值，无转换发生
uppercase	转换成大写
lowercase	转换成小写
capitalize	将每个英文单词的首字母转换成大写，其余无转换发生

Line-height--行间距（行高）

line-height 属性设置行间的距离（行高），如下例为**段落设置行间距（行高）为 2em**

```
p{line-height:2em;}
```

Color 颜色

检索或设置对象的文本颜色。无默认值。

可以使用 Color Name (颜色名称), HEX (十六进制颜色), RGB, RGBA, HSL, HSLA, transparent 来指定 color。

注意，用颜色名称指定 color 可能不被一些浏览器接受。

color 属性值被间接用来提供一个中间值 currentColor 以供其他接受颜色值的属性使用。

示例：

```
div {  
    border: 10px solid;  
    color: red;}
```

如上代码，没有定义边框的颜色，但定义了 color 的颜色为 red，那么这时 red 将会作为一个间接值 currentColor 提供给边框颜色属性，所以最终边框色也为 red。

HEX:十六进制数值表示颜色。

这个符号由红色、绿色和蓝色的值组成（RGB）。每种颜色的最小值是 0（十六进制：#00）。最大值是 255（十六进制：#FF）。如：color:#ff0000;

在 W3C 制定的 HTML 4.0 标准中，只有 16 种颜色可以用颜色名称表示：

Aqua(水色)、black、blue、fuchsia(品红)、gray、green、lime(绿黄色)、maroon(栗色)、navy、olive、purple、red、silver、teal(水鸭色)、white、yellow。

其它的颜色都要用 16 进制 RGB 颜色值表示。

```
color : red ;
```

```
color : #ff0000 ;
```

```
color : rgb(255, 0, 0) ;
```

```
color : rgba(255, 0, 0, 1) ;
```

文字颜色 : color:rgba(255,0,0,1);

```
color : rgba(255, 0, 0, 0.5) ;
```

文字颜色 : color:rgba(255,0,0,0.5);

文本对齐方式

Text-align:

text-align 属性规定元素中的文本的水平对齐方式。

该属性通过指定行框与哪个点对齐，从而设置块级元素内文本的水平对齐方式。通过允许用户代理调整行内容中字母和字之间的间隔，可以支持值 **justify**；不同用户代理可能会得到不同的结果。

默认值:	如果 direction 属性是 ltr ，则默认值是 left ；如果 direction 是 rtl ，则为 right 。
可能值	描述
left	把文本排列到左边。默认值：由浏览器决定。
right	把文本排列到右边。
center	把文本排列到中间。
justify	实现两端对齐文本效果。
inherit	规定应该从父元素继承 text-align 属性的值。

text-align 属性只能针对文本文字和 img 标签，对其他标签无效。

最后一个水平对齐属性是 **justify**。

在两端对齐文本中，文本行的左右两端都放在父元素的内边界上。然后，调整单词和字母间的间隔，使各行的长度恰好相等。您也许已经注意到了，两端对齐文本在打印领域很常见。

vertical-align

Vertical-align 属性设置元素的垂直对齐方式。

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

该属性定义行内元素的基线相对于该元素所在行的基线的垂直对齐。允许指定负长度值和百分比值。这会使元素降低而不是升高。在表单元格中，这个属性会设置单元格框中的单元格内容的对齐方式。

值	描述
baseline	默认。元素放置在父元素的基线上。
sub	垂直对齐文本的下标。
super	垂直对齐文本的上标
top	把元素的顶端与行中最高元素的顶端对齐
text-top	把元素的顶端与父元素字体的顶端对齐
middle	把此元素放置在父元素的中部。
bottom	把元素的顶端与行中最低的元素的顶端对齐。
text-bottom	把元素的底端与父元素字体的底端对齐。
length	
%	使用 "line-height" 属性的百分比值来排列此元素。允许使用负值。
inherit	规定应该从父元素继承 vertical-align 属性的值。

Text-indent

Text-indent 设置首行缩进。

中文文字中的段前习惯空两个文字的空白，这个特殊的样式可以用下面代码来实现：

```
p{text-indent:2em;}
```

注意：2em 的意思就是文字的 2 倍大小。

文本格式处理

White-space

white-space 属性设置如何处理元素内的空白（空白符）。

white-space : normal | nowrap | pre | pre-wrap | pre-line

	New Lines	Spaces and Tabs	Text Wrapping
<code>'normal'</code>	Collapse	Collapse	Wrap
<code>'nowrap'</code>	Collapse	Collapse	No wrap
<code>'pre'</code>	Preserve	Preserve	No wrap
<code>'pre-wrap'</code>	Preserve	Preserve	Wrap
<code>'pre-line'</code>	Preserve	Collapse	Wrap

可能值	描述
normal	默认。空白会被浏览器忽略。丢掉多余的空白符，换行字符（回车）会转换为空格，一行中多个空格的序列也会转换为一个空格。
pre	空白会被浏览器保留。其行为方式类似 HTML 中的 <code><pre></code> 标签。
nowrap	文本不会换行，文本会在同一行上继续，直到遇到 <code>
</code> 标签为止。
pre-wrap	保留空白符序列，但是正常地进行换行。
pre-line	合并空白符序列，但是保留换行符。
inherit	规定应该从父元素继承 white-space 属性的值。

white-space 属性的行为：

值	空白符	换行符	自动换行
pre-line	合并	保留	允许
normal	合并	忽略	允许
nowrap	合并	忽略	不允许
pre	保留	保留	不允许
pre-wrap	保留	保留	允许

Word-wrap

在 CSS3 中，我们可以使用 word-wrap 属性来设置“长单词”或“URL 地址”是否换行到下一行。

语法：word-wrap:取值；

说明：word-wrap 属性只有 2 个取值：normal 和 break-word。

word-wrap 属性取值

选择器	说明
normal	默认值，文本自动换行
break-word	“长单词”或“URL 地址”强制换行

默认情况下，文本是自动换行的，但是如果单词或者 URL 地址太长的话，就会超出区域范围，需要使用 word-wrap 属性来调节。

word-wrap 属性是针对英文来说的，中文中没有什么所谓的“长单词”之说。一般情况下，在中文网站开发中，word-wrap 属性只要采用默认值即可。当然，如果你开发的是英文网站，就很有可能用到 word-wrap 这个属性了。

Word-break

设置或检索对象内文本的字内换行行为。

对于解决防止页面中出现连续无意义的长字符打破布局，应该使用 break-all 属性值；

语法：

word-break: normal | keep-all | break-all

默认值：normal

取值：

- normal：依照亚洲语言和非亚洲语言的文本规则，允许在字内换行。
- keep-all：与所有非亚洲语言的 normal 相同。对于中文，韩文，日文，不允许字断开。适合包含少量亚洲文本的非亚洲文本。
- break-all：该行为与亚洲语言的 normal 相同。也允许非亚洲语言文本行的任意字内断开。该值适合包含一些非亚洲文本的亚洲文本，比如使连续的英文字母间断行。

文本修饰

Text-decoration -文本修饰

text-decoration 属性

属性值	说明
none	默认值，可以用这个属性值也可以去掉已经有下划线或删除线或顶划线的样式
underline	下划线

text-decoration 属性

属性值	说明
line-through	删除线
overline	顶划线

underline 会对元素加下划线，就像 HTML 中的 U 元素一样。**overline** 的作用恰好相反，会在文本的顶端画一个上划线。值 **line-through** 则在文本中间画一个贯穿线，等价于 HTML 中的 S 和 **strike** 元素。

none 值会关闭原本应用到一个元素上的所有装饰。通常，无装饰的文本是默认外观，但也不总是这样。例如，链接默认地会有下划线。如果您希望去掉超链接的下划线，可以使用以下 CSS 来做到这一点：

```
a {text-decoration: none;}
```

Text-shadow 文字阴影

语法：

```
1 text-shadow: x-offset y-offset blur color;
```

说明：

- **x-offset**：（水平阴影）表示阴影的水平偏移距离，单位可以是 px、em 或者百分比等。如果值为正，则阴影向右偏移；如果值为负，则阴影向左偏移；
- **y-offset**：（垂直阴影）表示阴影的垂直偏移距离，单位可以是 px、em 或者百分比等。如果值为正，则阴影向下偏移；如果值为负，则阴影向上偏移；
- **blur**：（模糊距离）表示阴影的模糊程度，单位可以是 px、em 或者百分比等。blur 值不能为负。如果值越大，则阴影越模糊；如果值越小，则阴影越清晰。当然，如果不需要阴影模糊效果，可以把 blur 值设置为 0；
- **color**：（阴影的颜色）表示阴影的颜色。

在 CSS3 中，可以使用 text-shadow 属性来给文字指定多个阴影，并且针对每个阴影使用不同的颜色。也就是说，text-shadow 属性可以为一个以英文逗号隔开的“值列表”，如：

```
text-shadow: 0 0 4px white, 0 -5px 4px #ff3, 2px -10px 6px #fd3;
```

当 text-shadow 属性值为“值列表”时，阴影效果会按照给定的值顺序应用到该元素的文本上，因此有可能出现互相覆盖的现象。但是 text-shadow 属性永远不会覆盖文本本身，阴影效果也不会改变边框的尺寸。

text-stroke 属性

在 CSS3 中，我们可以使用 text-stroke 属性为**文字添加描边效果**。这个描边效果，

说白了就是**给文字添加边框**。由于 CSS3 的出现，“文字”也能添加边框了呢。

语法：text-stroke:宽度值 颜色值;

text-overflow 属性

在 CSS3 中，文本溢出 text-overflow 用于设置是否使用一个省略标记 (...) 标示对象内文本的溢出。

语法：text-overflow:取值;

说明：text-overflow 属性取值只有 2 个：

属性值	说明
ellipsis	当对象内文本溢出时显示省略标记 (...)
clip	当对象内文本溢出时不显示省略标记 (...)，而是将溢出的部分裁切掉

单独使用 text-overflow 属性是无法得到上面图 1 效果的。因为 text-overflow 属性只是说明文字溢出时用什么方式显示，要实现溢出时产生省略号效果，还须定义 2 个内容：

- **(1) white-space:nowrap; (强制文本在一行内显示) ；**
- **(2) overflow:hidden; (溢出内容为隐藏) ；**

下面是实现文字溢出时产生省略号效果的完整语法：

```
1text-overflow:ellipsis;  
2overflow:hidden;  
3white-space:nowrap;
```
















这 3 个属性是必须一起使用才会有效果。

Cursor 属性

在 CSS 中，使用 cursor 属性来定义鼠标的样式。

语法：cursor:属性值;

cursor 属性取值如下，默认值为 default。在实际开发中，我们一般只用到“default”和“pointer”这两个属性值，其他的一般都很少用得上，无需记忆。

cursor 属性值	说明
default	 (默认值)
pointer	 (常用值)
text	
crosshair	
wait	
help	
move	
e-resize	
ne-resize	
nw-resize	
n-resize	
se-resize	
sw-resize	
s-resize	
w-resize	

Inherit 属性

该值使得属性能够继承祖先设置。

inherit 属于 CSS-wide 关键字，这表示**所有的属性都可以接受该值。**

如何让一个不具备继承特性的属性可以继承父元素的定义？

示例代码：

```
div {  
    position: relative;  
}  
div a {  
    position: inherit;  
}
```

上述代码，超链接 a 将会继承父元素的 **position** 定义，也会定义为 relative。

常用的 CSS 属性继承性：

font-size : inherit ;	text-align : inherit ;
font-family : inherit ;	text-indent : inherit ;
font-weight : inherit ;	white-space : inherit ;
font-style : inherit ;	word-wrap : inherit ;
line-height : inherit ;	word-break : inherit ;
color : inherit ;	text-shadow : inherit ;
text-decoration : inherit ;	