CSS 指层叠样式表 (Cascading Style Sheets)，定义了如何显示 HTML 元素

教程

# 1 类别/位置

## 1.1 内联样式

- 在HTML元素内使用 **style属性**定义适用的样式表属性
- 语法格式如下：

```
<div style="内容"><div>
<!-- 例如 -->
<div style="background-color:red"><div>
```

## 1.2 内部样式表

- 由 `<style></style>` 标记对放在 `<head></head>` 中
- 在`<style>`中有一个类型属性 `type`，后面接 `test/css`，表示CSS文本
- 语法格式如下：

```
<style type="text/css">
    /* 这里写CSS内容 */
    #id { background-color:red; }
</style>
```

## 1.3 外部样式表 (推荐)

- **把CSS文件和HTML文件分割开**，通过链接 `<link>` 使CSS文件对本网页的样式有效
- 语法格式如下：

```
<link type="text/css" rel="stylesheet" href="css文件的存放地址">
```

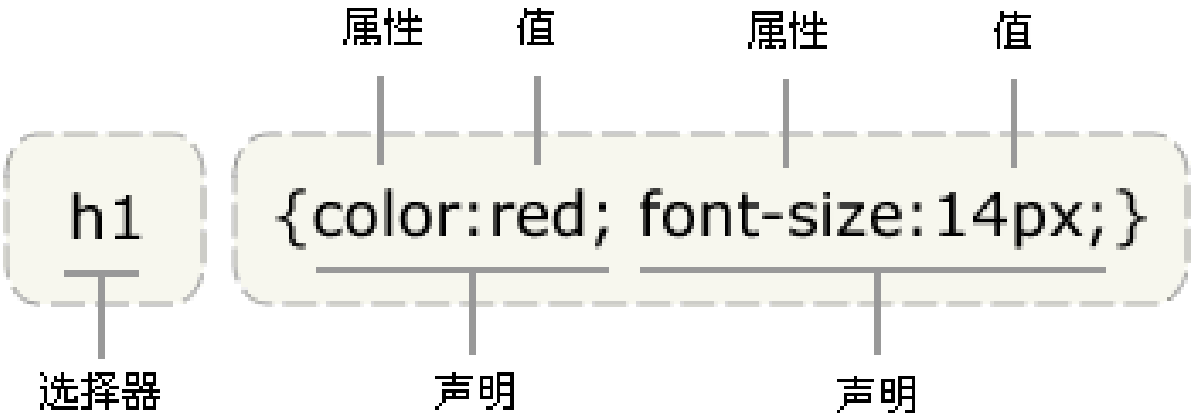
**优先级：内联样式 > 内部样式 > 外部样式**

# 2 语法

基本写法：

- 选择器
- 声明
    - 属性
    - 值

下面这行代码的作用是将 `h1` 元素内的文字颜色定义为红色，同时将字体大小设置为 14 像素




**每行只描述一个属性可以增强样式定义的可读性**，例如：

```
p {
    text-align: center;
    color: black;
    font-family: arial;
}
```

常用CSS样式属性（较为完整的CSS样式说明）：

| 类型 | 属性 | 作用 | 例子 |
| --- | --- | --- | --- |
| 布局相关 | position | 设置布局定位方法 \| position: absolute;` | |
| | top, bottom, left, right | 在**绝对布局**时，设置相对边距 | `top: 10%;` |
| | width, height | 设置宽、高 \| width: 100px;` | |
| | margin, border, padding | 设置外边距、边框、内边距 | `border: 1px solid black;` |
| | transform | 设置形变（不影响布局占位） | `transform:** **translateX**** (1px);` |
| | z-index | 设置层高度（影响遮挡） | `z-index: 999;` |
| | overflow | 内容超出边界时处理方法 | `overflow: auto hidden;` |
| 显示效果 | background, color | 背景、前景颜色 | `color: white;` |
| | box-shadow | 元素边界阴影 | `box-shadow: 0 0 1px 1px black;` |
| | animation, transition | 动画 | `transition: height 2s ease-in-out;` |
| 文本相关 | line-height,font-size, font-family, font-style | 行高、字体大小、字体、加粗斜体等样式 | `font-size: 1.3rem;` |

# 3 选择器

## 3.1 基本选择器

| 选择器 | 类型 | 功能 |
| --- | --- | --- |
| * | 通配选择器 | 选择文档中所有HTML元素 |
| E | 元素选择器 | 选择指定类型的HTML元素 |
| #id | ID选择器 | 选择指定ID属性值为“id”的任意类型元素，最好唯一 |
| .class | 类选择器 | 选择指定class属性值为“class”的任意类型的**任意多个元素** |
| selector1, selectorN | 群组选择器 | 将每一个选择器匹配的元素集合并 |

```
/* 选中某个 HTML 元素，比如 div、p、h1、a */
html {color: black;}
h1 {color: blue;}
h2 {color: silver;}
/* id 选择器以 "#" 来定义 */
#red {color: red;}
#green {color: green;}
/* class 选择器以 "." 来定义 */
.important {color: red;}
/* h1, h2 元素颜色都设置为red */
h1, h2 {color: red;}
```

## 3.2 层次选择器

| 选择器 | 类型 | 功能 |
| --- | --- | --- |
| `E F` | 后代选择器（包含选择器） | 选择匹配的F元素，且匹配的F元素被包含在匹配的E元素内 |
| `E > F` | 子选择器 | 选择匹配的F元素，且匹配的F元素是所匹配的E元素的子元素 |
| `E + F` | 相邻兄弟选择器 | 选择匹配的F元素，且匹配的F元素紧位于匹配的E元素的后面 |
| `E ~ F` | 通用选择器 | 选择匹配的F元素，且位于匹配的E元素后的所有匹配的F元素 |

```
<html>
    <head>
    <style type="text/css">
        /* 后代选择器 */
        p em {color:red;}
        /* 子选择器 */
        h1 > strong {color:red;}
        /* 相邻兄弟选择器 */
        h2 + p {color: blue;}
        /* 通用选择器 */
        h3 ~ p {color: green;}
    </style>
    </head>

    <body>
        <h1>This is a <em>important</em> heading</h1>
        <p>This is a <em>important</em> paragraph.</p>
        <!-- 只有h2紧跟着的p元素会变成蓝色 -->
        <h2>This is a heading.</h2>
        <p>This is paragraph.</p>
        <p>This is paragraph.</p>
        <!-- h3后面的所有p元素会变成绿色 -->
        <h3>This is a heading.</h3>
        <p>This is paragraph.</p>
        <p>This is paragraph.</p>
        <!-- 两个very都会变成红色，因为strong是h1的子元素 -->
        <h1>This is <strong>very</strong> <strong>very</strong> important.</h1>
```

```
        <!-- very不会变成红色，因为strong是em的子元素 -->
        <h1>This is <em>really <strong>very</strong></em> important.</h1>
        <p>This is paragraph.</p>
    </body>
</html>
```

# This is a *important* heading

This is a *important* paragraph.

## This is a heading.

This is paragraph.

This is paragraph.

### This is a heading.

This is paragraph.

This is paragraph.

# This is very very important.

# This is *really very* important.

This is paragraph.

## 3.3 属性选择器

| 选择器 | 功能 |
| --- | --- |
| [*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value开头的所有元素 |
| [*attribute$=value*] | 匹配属性值以value结尾的所有元素 |
| [*attribute**=value*] | 匹配属性值中包含子串value的所有元素 |

```
<!DOCTYPE HTML>
    <html>
        <head>
        <style type="text/css">
            /* 把包含标题（title）的所有元素变为红色 */
            [title] {
                color:red;
            }
            /* 只对有 href 属性的锚（a 元素）应用样式 */
            a[href] {
```

```
                color:red;
            }
            /* 同时有 href 和 title 属性的 HTML 超链接的文本设置为红色 */
            a[href][title] {
                color:red;
            }
            /* 属性值完全匹配 */
            p[class="important warning"] {
                color: red;
            }
            /* 根据部分属性值选择 */
            p[class~="hello"] {
                color: blue;
            }
            /* 会选择 lang 属性等于 en 或以 en- 开头的所有元素 */
            *[lang|="en"] {
                color: green;
            }
        </style>
    </head>

    <body>
        <h2 title="Hello world">Hello world</h2>

        <a href="http://w3school.com.cn">W3School</a><br>
        <a name="w3school">W3School</a><br>

        <a title="W3School Home" href="http://w3school.com.cn">W3School</a><br>

        <p class="important warning">This is a paragraph.</a>

        <p class="hello world">This is a paragraph.</a>
        <p class="hello">This is a paragraph.</a>

        <p lang="en">Hello!</p>
        <p lang="en-us">Greetings!</p>
        <p lang="en-au">G'day!</p>
        <p lang="fr">Bonjour!</p>
    </body>
</html>
```

# Hello world

W3School
W3School
W3School

This is a paragraph.

This is a paragraph.

This is a paragraph.

Hello!

Greetings!

G'day!

Bonjour!

## 3.4 伪类与伪元素

### 3.4.1 伪类

伪类是选择器的一种，它用于**选择处于特定状态的元素**，例如：

- 当它们是这一类型的第一个元素时
- 当鼠标指针悬浮在元素上面的时候

它们表现得会像是你向你的文档的某个部分应用了一个类一样，帮你在你的标记文本中减少多余的类，让你的代码更灵活、更易于维护

- 伪类就是开头为冒号的关键字：`:pseudo-class-name`

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            p {
              display: none;
              background-color: yellow;
              padding: 20px;
            }
            <!-- 把鼠标悬停到 <div> 元素以显示 <p> 元素 -->
            div:hover p {
              display: block;
            }
        </style>
    </head>
    <body>
        <div>鼠标移到我上面来显示 p 元素
            <p>哈哈！我在这里！</p>
        </div>
    </body>
</html>
```

- 所有 CSS 伪类：

| 选择器 | 例子 | 例子描述 |
| --- | --- | --- |
| :active | a:active | 选择活动的链接 |
| :checked | input:checked | 选择每个被选中的 <input> 元素 |
| :disabled | input:disabled | 选择每个被禁用的 <input> 元素 |
| :empty | p:empty | 选择没有子元素的每个 <p> 元素 |
| :enabled | input:enabled | 选择每个已启用的 <input> 元素 |
| :first-child | p:first-child | 选择作为其父的首个子元素的每个 <p> 元素 |
| :first-of-type | p:first-of-type | 选择作为其父的首个 <p> 元素的每个 <p> 元素 |
| :focus | input:focus | 选择获得焦点的 <input> 元素 |
| :hover | a:hover | 选择鼠标悬停其上的链接 |
| :in-range | input:in-range | 选择具有指定范围内的值的 <input> 元素 |
| :invalid | input:invalid | 选择所有具有无效值的 <input> 元素 |
| :lang(language) | p:lang(it) | 选择每个 lang 属性值以 "it" 开头的 <p> 元素 |
| :last-child | p:last-child | 选择作为其父的最后一个子元素的每个 <p> 元素 |
| :last-of-type | p:last-of-type | 选择作为其父的最后一个 元素的每个 <p> 元素 |
| :link | a:link | 选择所有未被访问的链接 |
| :not(selector) | :not(p) | 选择每个非 <p> 元素的元素 |
| :nth-child(n) | p:nth-child(2) | 选择作为其父的第二个子元素的每个 <p> 元素 |
| :nth-last-child(n) | p:nth-last-child(2) | 选择作为父的第二个子元素的每个 <p> 元素，从最后一个子元素计数 |
| :nth-last-of-type(n) | p:nth-last-of-type(2) | 选择作为父的第二个 <p> 元素的每个 <p> 元素，从最后一个子元素计数 |
| :nth-of-type(n) | p:nth-of-type(2) | 选择作为其父的第二个 <p> 元素的每个 <p> 元素 |
| :only-of-type | p:only-of-type | 选择作为其父的唯一 <p> 元素的每个 <p> 元素 |
| :only-child | p:only-child | 选择作为其父的唯一子元素的 <p> 元素 |

| 选择器 | 例子 | 例子描述 |
| --- | --- | --- |
| `:optional` | `input:optional` | 选择不带 "required" 属性的 `<input>` 元素 |
| `:out-of-range` | `input:out-of-range` | 选择值在指定范围之外的 [ ] 元素 |
| `:read-only` | `input:read-only` | 选择指定了 "readonly" 属性的 `<input>` 元素 |
| `:read-write` | `input:read-write` | 选择不带 "readonly" 属性的 `<input>` 元素 |
| `:required` | `input:required` | 选择指定了 "required" 属性的 `<input>` 元素 |
| `:root` | `root` | 选择元素的根元素 |
| `:target` | `#news:target` | 选择当前活动的 #news 元素（单击包含该锚名称的 URL） |
| `:valid` | `input:valid` | 选择所有具有有效值的 `<input>` 元素 |
| `:visited` | `a:visited` | 选择所有已访问的链接 |

### 3.4.2 伪元素

伪元素以类似方式表现，不过表现得是像你往标记文本中加入全新的HTML元素一样，而不是向现有的元素上应用类。伪元素开头为双冒号`::`： `::pseudo-element-name`

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            <!-- 为所有 <p> 元素中的首行添加样式 -->
            p::first-line {
                color: #ff0000;
                font-variant: small-caps;
            }
        </style>
    </head>
    <body>
        <p>您可以使用 ::first-line 伪元素将特殊效果添加到文本的第一行。一些更多的文字。越来越多，越来越多，越来越多，越来越多，越来越多，越来越多，越来越多，越来越多，越来越多，越来越多。
</p>
    </body>
</html>
```

所有 CSS 伪元素：

| 选择器 | 例子 | 例子描述 |
| --- | --- | --- |
| `::after` | `p::after` | 在每个 `<p>` 元素之后插入内容 |
| `::before` | `p::before` | 在每个 `<p>` 元素之前插入内容 |
| `::first-letter` | `p::first-letter` | 选择每个 `<p>` 元素的首字母。 |
| `::first-line` | `p::first-line` | 选择每个 `<p>` 元素的首行 |
| `::selection` | `p::selection` | 选择用户选择的元素部分 |

## 优先级与继承

- **id选择器样式 > 类选择器样式 > 元素选择器样式**
- 子元素默认从父元素继承属性
- 创建一个针对子元素的特殊规则，可以摆脱父元素的规则

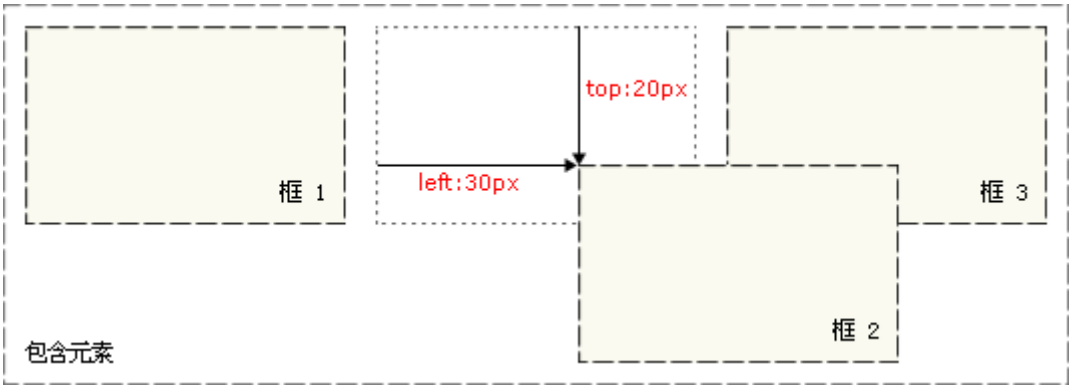
# 4 定位

## 4.1 相对定位

**设置为相对定位的元素框会偏移某个距离。元素仍然保持其未定位前的形状，它原本所占的空间仍保留**

- 如果对一个元素进行相对定位，它将出现在它所在的位置上
- 然后，可以通过设置垂直或水平位置，让这个元素**“相对于”它的起点进行移动**
- 注意，在使用相对定位时，无论是否进行移动，**元素仍然占据原来的空间。**因此，移动元素会导致它覆盖其它框

```
#box_relative {
    position: relative;
    left: 30px;
    top: 20px;
}
```



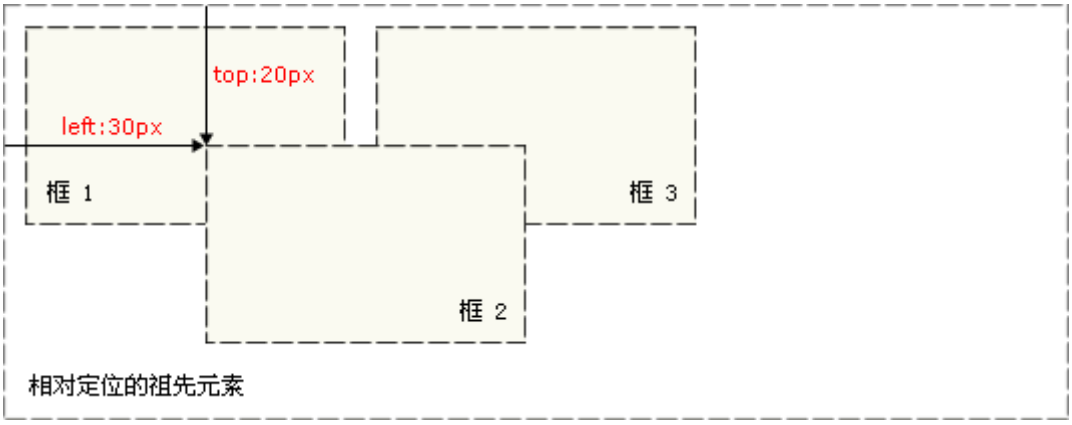

## 4.3 绝对定位

**设置为绝对定位的元素框从文档流完全删除，并相对于其包含块定位**

- 包含块可能是文档中的另一个元素或者是初始包含块
- 元素**原先在正常文档流中所占的空间会关闭**，就好像该元素原来不存在一样
- 元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框
- **普通流中其它元素的布局就像绝对定位的元素不存在一样**

- 绝对定位的元素的位置**相对于最近的已定位祖先元素**，如果元素没有已定位的祖先元素，那么它的位置相对于*最初的包含块*

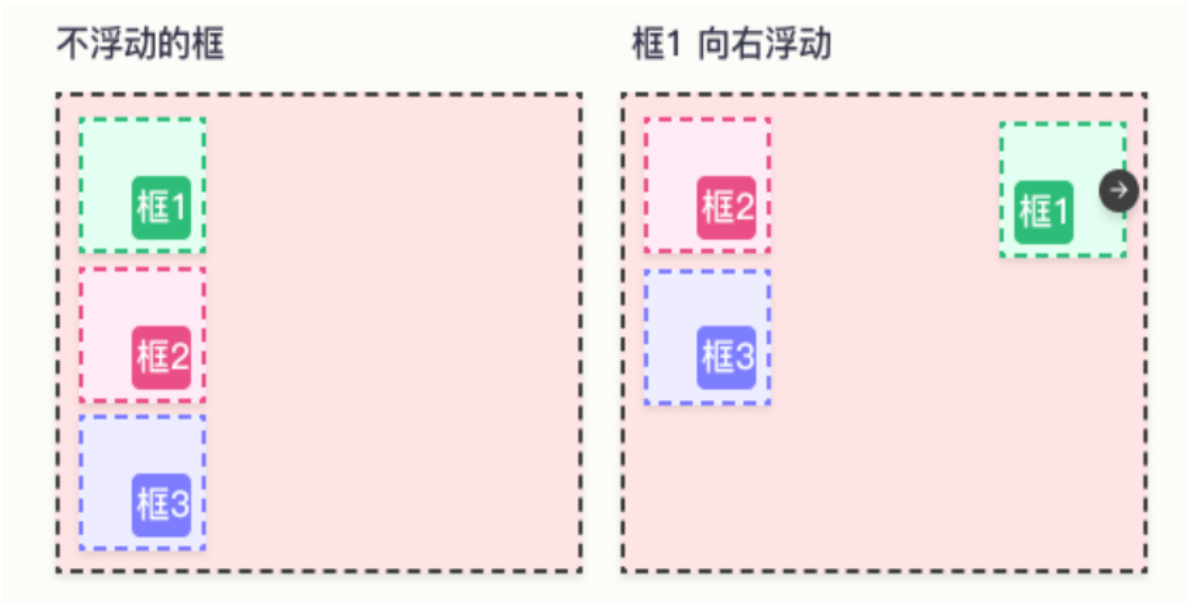
```
#box_relative {
    position: absolute;
    left: 30px;
    top: 20px;
}
```



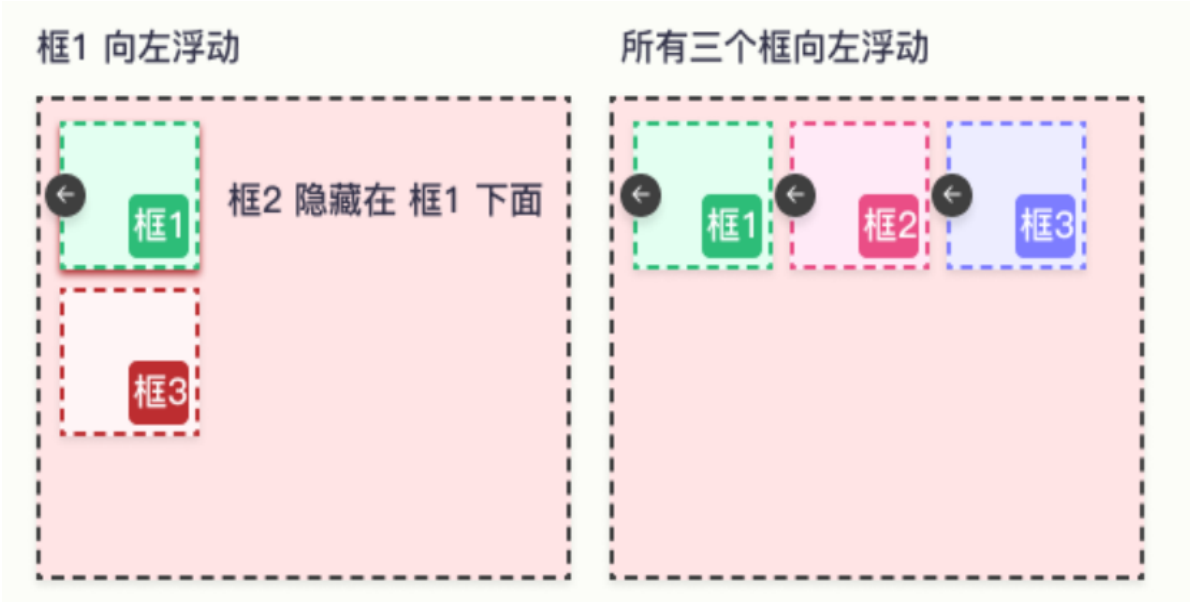

## 4.3 浮动

- 浮动的框可以向左或向右移动，**直到它的外边缘碰到包含框或另一个浮动框的边框为止**
- 由于浮动框不在文档的普通流中，所以**文档的普通流中的块框表现得就像浮动框不存在一样**

当把框 1 向右浮动时，它脱离文档流并且向右移动，直到它的右边缘碰到包含框的右边缘




当框 1 向左浮动时，它脱离文档流并且向左移动，直到它的左边缘碰到包含框的左边缘。因为它不再处于文档流中，所以它不占据空间，实际上覆盖住了框 2，使框 2 从视图中消失；如果把所有三个框都向左移动，那么框 1 向左浮动直到碰到包含框，另外两个框向左浮动直到碰到前一个浮动框




如果包含框太窄，无法容纳水平排列的三个浮动元素，那么其它浮动块向下移动，直到有足够的空间。如果浮动元素的高度不同，那么当它们向下移动时可能被其它浮动元素“卡住”