Html

基础结构标签

标准骨架：

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

</body>

</html>

标题标签

为了使网页更具有语义化，我们经常会在页面中用到标题标签。

HTML 提供了 6 个等级的网页标题，即：<h1> 到 <h6>。

<h1>一级标题</h1>

<h2>二级标题</h2>

<h3>三级标题</h3>

<h4>四级标题</h4>

<h5>五级标题</h5>

<h6>六级标题</h6>

段落和换行标签

在网页中，要把文字有条理地显示出来，就需要将这些文字分段显示，在 HTML 标签中，<p> 标签用于定义段落，它可以将整个网页分为若干个段落。

<p>我是一个段落标签</p>

如果希望某段文本强制换行显示，就需要使用换行标签 <br>。

<br>

<br> 是个单标签

<br> 标签只是简单地开始新的一行，跟段落不一样，所以不会产生段间距

分割线标签：<hr>

生成一条横线分割线，不常用

<div>和<span>标签

<div> 和 <span> 是没有语义的，它们就是两种盒子，用来对网页进行布局和装其他内容。

<div>这是头部</div>

<span>今日价格</span>

特点：

<div> 标签用来布局，一行只能放一个 <div>，大盒子

<span> 标签用来布局，一行上可以放多个 <span>，小盒子

<a>标签

<a href="https://www.runoob.com" target="\_self或\_blank">这是一个链接</a>

Href:用于指定链接目标的 url 地址，（必须属性）当标签应用 href 属性时，它就具有了超链接的功能

Target: 用于指定链接页面的打开方式，\_self 在当前页面打开的方式（为默认值），\_blank 在新窗口中打开的方式

除了打开外部网站，<a>标签也可以实现页面内跳转：（用选择器定位）

<a href="#section2">跳转到第二部分</a>

<img>标签

<img src="/images/logo.png" alt=" " title="这是图片" width="" height="" />

Src：图片路径 必须属性

alt：文本 替换文本，图像显示失败时显示（建议都把 alt 写上）

Title：文本 提示文本，鼠标放到图片上，显示的提示文字

Width，height，border：像素 设置图像的宽度（了解，后面通过 CSS 设置）

路径

相对路径：以引用文件所在位置为参考基础，而建立出目录路径。

相对路径分类 符号 说明

同一级路径 . 如：<img src="baidu.png" />

下一级路径 / 如：<img src="image/baidu.png" />

上一级路径 ../ 如：<img src="../image/baidu.png" />

绝对路径：指目录下的绝对位置，直接到达目的位置，通常是从盘符开始的路径。

如 Windows 系统的绝对路径：D:\web\img\logo.png

网络地址

https://github.com/JERRY-Z-J-R/I-love-you-3-thousand/raw/master/mark-img/I-Love-You.gif

注意：

相对路径为 /（正斜杆），绝对路径为 \（反斜杆）

实际开发中建议使用相对路径或网络地址（都是 / 正斜杆）

注释标签

注释以：<!-- 开头，以 --> 结束

| **特殊字符** | **描述** | **字符的代码** |
| --- | --- | --- |
|  | 空格符 | &nbsp; |
| < | 小于号 | &lt; |
| > | 大于号 | &gt; |
| & | 和号 | &amp; |
| ± | 正负号 | &plusmn; |
| × | 乘号 | &times; |
| ÷ | 除号 | &divide; |
| ² | 上标 | &sup2; |

文本格式化标签

加粗 <strong> </strong> 或 <b> </b>

倾斜 <em> </em> 或 <i> </i>

删除线 <del> </del> 或 <s> </s>

下划线 <ins> </ins> 或 <u> </u>

表格标签

基本语法

<table>

<!-- 头部区域 -->

<thead>

<tr>

<th>姓名</th>（表示表头）

<th>性别</th>

...

</tr>

</thead>

<!-- 主体区域 -->

<tbody>

<tr>（表示第一行）

<td>单元格</td>（表示第一列）

...

</tr>

...

</tbody>

</table>

表头单元格标签<th> 标签表示 HTML 表格的表头部分

一般表头单元格位于表格的第一行或第一列，作用是：突出重要性，表头单元格里面的文本内容默认加粗居中显示。

表格属性

写在<table>标签内，<table align="center" border="1" cellpadding="0"。。。>

Align：left、center、right 表示对齐方式

Border：1 或 "" 规定表格单元是否拥有边框，默认为 ""，表示没有边框

Cellpadding：像素值 规定单元边沿与单元内内容之间的空白，默认 1 像素

Cellspacing：像素值 规定单元格之间的空白，默认 2 像素

Width：像素值 或 百分比 规定表格的宽度

Height：像素值 或 百分比 规定表格的高度

合并单元格

跨行合并（上下合并）：rowspan="合并单元格的个数"

跨列合并（左右合并）：colspan="合并单元格的个数"

无序列表

<ul> 标签表示 HTML 页面中项目的无序列表，一般会以项目符号呈现列表项，而列表项使用 <li> 标签定义（开发中经常使用）。

无序列表的基本语法格式如下：

<ul>

<li>列表项1</li>

<li>列表项2</li>

<li>列表项3</li>

...

</ul>

表示为：

・列表项1

・列表项2

・列表项3

无序列表会带有自己的样式属性（比如圆点），但在实际开发中，我们会使用 CSS 来设置

去除圆点：style="list-style: none;"

有序列表

有序列表即为有序排列顺序的列表，其各个列表项会按照一定的顺序排列定义（开发中不太常用）。

有序列表的基本语法格式如下：

<ol>

<li>列表项1</li>

<li>列表项2</li>

<li>列表项3</li>

...

</ol>

自定义列表

使用场景：自定义列表常用于对术语或名词进行解释、描述和展开，定义列表的列表项前没有任何项目符号（开发中常用）。

在 HTML 标签中，<dl> 标签用于定义描述列表（或定义列表），该标签会与 <dt>（定义项目/名字）和 <dd>（描述每一个项目/名字）一起使用。

其基本语法如下：

<dl>

<dt>名词1</dt>

<dd>名词1解释1</dd>

<dd>名词1解释2</dd>

</dl>

表单域

<form> 标签用于定义表单域，以实现用户信息的收集和传递。

<form> 会把它范围内的表单元素信息提交给服务器。

<form action="url地址" method="提交方式" name="表单域名称">

<!-- 各种表单元素控件 -->

</form>

action url 地址 用于指定接收并处理表单数据的服务器程序的 url 地址

method get / post 用于设置表单数据的提交方式，其取值为 get 或 post

name 名称 用于指定表单的名称，以区分同一个页面中的多个表单域

get 和 post 区别：

GET：

将表单数据以名/值对的形式附加到 URL 中

URL 的长度是有限的（大约 3000 字符）

绝不要使用 GET 来发送敏感数据！（在 URL 中是可见的，且浏览器会缓存 URL）

对于用户希望加入浏览器书签的表单很有用（因为信息记录在 URL 中，直接保存 URL 即可）

GET 更适用于非安全数据，比如在 Google 中查询字符串

POST：

将表单数据附加到 HTTP 请求的 body 内（数据不显示在 URL 中）

没有长度限制

通过 POST 提交的表单不能加入书签

POST 数据也是不安全的，但起码不会明目张胆的直接把数据显示在地址栏 URL 上，且不会缓存数据

表单元素

1. <input> 表单元素

语法如下：

<iuput type="text" name="用户名" value="请输入用户名">

<input> 标签属性

type属性：

button 定义可点击按钮（多数情况下，用于通过 JavaScript 启动脚本）

checkbox 定义复选框，即：多选框，在一组多选中，要求它们必须拥有相同的 name

file 定义输入字段和 “浏览” 按钮，供文件上传

hidden 定义隐藏的输入字段

image 定义图像形式的提交按钮

password 定义密码字段，该字段中的字符被掩码

radio 定义单选按钮，在一组单选按钮中，要求它们必须拥有相同的 name

reset 定义重置按钮，重置按钮会清除表单中的所有数据

submit 定义提交按钮，提交按钮会把表单数据发送到服务器

text 定义单行的输入字段，用户可在其中输入文本，默认宽度为 20 个字符

其他标签：

name 由用户自定义 定义 input 元素的名称

value 由用户自定义 规定 input 元素的值，也就是提交到服务器的值

Checked checked 规定此 input 元素默认被选中

maxlength 正整数 规定输入字段中的字符的最大长度

注意：name 和 value 是每个表单元素都必须有的属性值，主要给后台人员使用，name 表单元素的名字，要求：同一组单选按钮和复选框要有相同的 name 值

placeholder（用于 <input> 和 <textarea> 元素）：在输入框中显示提示文本。

例如：<input type="text" placeholder="Enter your email">

<label> 标签为 input 元素定义标注。

<label> 标签用于绑定一个表单元素，当点击 <label> 标签内的任何东西时，浏览器就会自动将焦点（光标）转到或者选择对应的表单元素上，用来增加用户体验。

语法：

<label for="sex">

男<input id="sex" type="radio" name="sex">

</label>

核心： <label> 标签的 for 属性应当与相关元素的 id 属性相同

1. <select> 表单元素

<select> 标签控件定义下拉列表。

语法：

<select>

<option>选项1</option>

<option>选项2</option>

<option>选项3</option>

...

</select>

注意：<select> 中至少包含一对 <option>，在 <option> 中定义 selected="selected" 时，当前项即为默认选中项

1. <textarea> 表单元素

当用户输入内容较多的情况下，我们就不能使用文本框表单了，此时我们可以使用 <textarea> 标签

在表单元素中，<textarea> 标签是用于定义多行文本输入的控件。

语法：

<textarea rows="3" cols="20">

文本内容

</textarea>

注意：cols="每行中的字符数"，rows="显示的行数"，我们在实际开发中不会使用，都是用 CSS 来改变大小

表单元素总结

input 输入表单元素、select 下拉表单元素、textarea 文本域表单元素

这三组表单元素都应该包含在 form 表单域 里面，并且都应该有 name 属性

**内联框架**

<iframe> 标签规定一个内联框架。

一个内联框架被用来在当前 HTML 文档中嵌入另一个文档。

通过使用框架，你可以在同一个浏览器窗口中显示不止一个页面。

语法：

<iframe src="URL"></iframe>

CSS

CSS (Cascading Style Sheets，层叠样式表）

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

h3 {

color: deeppink;

font-size: 20px;

}

CSS选择器的作用

选择器就是根据不同的需求把不同的标签选出来，这就是选择器的作用，简单来说，就是：选择标签用的。

找到所有的 h1 标签。（选对人）

设置这些标签的样式：颜色为红色、字体大小为 25 像素。（做对事）

选择器的分类

选择器分为 基础选择器 和 复合选择器 两个大类

CSS基础选择器：

标签选择器、类选择器、id 选择器、通配符选择器

标签选择器

为页面中某一类标签指定统一的 CSS 样式

语法：

标签名 {

属性1: 属性值1;

属性2: 属性值2;

...

}

作用：标签选择器可以把某一类标签全部选择出来设置样式，比如所有的 <div> 标签和所有的 <span> 标签。

类选择器

如果想要差异化选择不同的标签，单独选一个或者某几个标签，可以使用 类选择器。

语法：

.类名 {

属性1: 属性值1;

...

}

注意：class可以理解为给这个标签起了一个别名来表示，长名称或词组可以使用中横线 - 来为类命名，一个标签可以有多个类名，同时多个标签也可以共用同一个类名

id选择器

id 选择器可以为标有特定 id 的 HTML 元素指定特定的样式。

HTML 元素以 id 属性来设置 id 选择器，CSS 中 id 选择器以 # 来定义。

语法：

#id名 {

属性1: 属性值1;

...

}

注意：相同的id 属性只能在每个 HTML 文档中出现一次，不可重复，相当于标签的身份证号，类名可以重复id不能重复，id只能一对一

通配符选择器

在 CSS 中，通配符选择器使用 \* 定义，它表示选取页面中所有元素（标签）。

语法：

\* {

属性1: 属性值1;

...

}

**CSS字体属性**

CSS Fonts（字体）属性用于定义：字体系列、大小、粗细、和 文字样式（如：斜体）。

**字体系列**： font-family 属性定义文本的字体系列。

p {

font-family: "Microsoft YaHei";

}

div {

font-family: Arial, "Microsoft YaHei";

}

**字体大小**： font-size 属性定义字体大小。

p {

font-size: 20px;

}

**字体粗细**：font-weight 属性设置文本字体的粗细。

p {

font-weight: bold（加粗）/normal（不加粗，默认）;

}

**文本倾斜**： font-style 属性设置文本的风格。

p {

font-style: normal/\* 不倾斜 \*//italic/\* 倾斜 \*/;

}

字体复合属性

字体属性可以把以上文字样式综合来写，这样可以更节约代码。

body {

font: font-style font-weight font-size/line-height font-family;

}

body {

font: normal 400 font-size/line-height "Microsoft YaHei", Arial, sans-serif;

}

**CSS文本属性**

CSS Text（文本）属性可定义文本的 外观，比如：文本颜色、文本对齐、文本装饰、文本缩进、行间距 等。

**颜色**：color 属性用于定义文本的颜色。

div {

color: red;

}

**文本对齐**：text-align 属性用于设置元素内文本内容的水平对齐方式。

div {

text-align: center;

}

属性值 解释

left 左对齐（默认值）

rigth 右对齐

center 居中对齐

**文本装饰**：text-decoration 属性规定添加到文本的修饰，可以给文本添加 下划线、删除线、上划线 等。

div {

text-decoration: underline;

}

属性值

none 默认，没有装饰线（最常用）

underline 下划线，链接 a 自带下划线（常用）

overline 上划线（几乎不用）

line-through 删除线（不常用）

**文本缩进**：text-indent 属性用来指定文本的第一行的缩进，通常是将段落的首行缩进。

div {

text-indent: 10px;

}

**行间距（行高）**：line-height 属性用于设置行间的距离（行高），可以控制文字行与行之间的距离。

p {

line-height: 26px;

}

行间距 = 上间距 + 文本高度 + 下间距

上下间距 = （行间距 - 文本高度）/ 2

文本高度 = font-size

**CSS的三种引入方式**

按照 CSS 样式书写的位置（或者引入的方式），CSS 样式表可以分为三大类：

行内样式表（行内式）

内部样式表（嵌入式）

外部样式表（外链式）

**行内样式表**

行内样式表（内联样式表）是在元素标签内部的 style 属性中设定 CSS 样式，适合于修改简单样式。

<div style="color: red; font-size: 12px;">

青春不常在，抓紧谈恋爱

</div>

**内部样式表**

内部样式表（嵌入样式表）时写到 HTML 页面内部，是将所有的 CSS 代码抽取出来，单独放到一个 <style> 标签中。

<style type="text/css">

div {

color: red;

font-size: 12px;

}

</style>

注意：<style> 标签理论上可以放在 HTML 文档的任何地方，但一般会放到文档的 <head> 标签中

**外部样式表**

实际开发都是外部样式表，适合于样式比较多的情况，核心是：样式单独写到 CSS 文件中，之后把 CSS 文件引入到 HTML 页面中使用。

引入外部样式表分为两步：

1，新建一个后缀名为：.css 的样式文件，把所有的 CSS 代码都放入此文件中

2，在 HTML 页面中，使用 <link> 标签引入这个文件，一般都放在文档的 <head> 标签中

<link rel="stylesheet" type="text/css" href="css文件路径">

属性 作用

rel 定义当前文档与被链接文档之间的关系，在这里需要指定为 "stylesheet"，表示被链接的文档是一个样式表文件

type 定义被链接文档的 MIME 类型，该属性最常见的 MIME 类型是 "text/css"，该类型描述样式表，目前的浏览器已经支持省略 "type" 属性

href 定义所链接外部样式表文件的 URL，可以是相对路径，也可以是绝对路径

**复合选择器**

在 CSS 中，可以根据选择器的类型把选择器分为：基础选择器 和 复合选择器，复合选择器是建立在基础选择器之上，对基础选择器进行组合形成的。

复合选择器可以更准确、更高效的选择目标元素（标签）

复合选择器是由两个或多个基础选择器，通过不同的方式组合而成的

常用的复合选择器包括：后代选择器、子选择器、并集选择器、伪类选择器等

**后代选择器**

后代选择器 又称为 包含选择器，可以选择父元素里面子元素。其写法就是把外层标签写在前面，内层标签写在后面，中间用空格分隔。当标签发生嵌套时，内层标签就成为外层标签的后代。

语法：

元素1 元素2 { 样式声明 }

上述语法表示选择 元素 1 里面的所有元素 2 （后代元素）。

例如：

ul li { 样式声明 } /\* 选择 ul 里面所有的 li 标签元素 \*/

・元素1 和 元素2 中间用 空格 隔开

・元素1 是父级，元素2 是子级，最终选择的是 元素2，即：元素1 是不会生效样式的

・元素2 可以是儿子，也可以是孙子等，只要是 元素1 的后代即可

・元素1 和 元素2 可以是任意基础选择器

**子选择器**

子元素选择器（子选择器）只能选择作为某元素的最近一级子元素，简单理解就是选亲儿子元素。

注意：是最近一级而并非最近一个！用>连接

语法：

元素1>元素2 { 样式声明 }

上述语法表示选择元素1 里面的所有直接后代（子元素）元素2。

**并集选择器**

并集选择器 可以选择多组标签，同时为他们定义相同的样式，通常用于集体声明。 并集选择器 是各选择器通过英文逗号 , 连接而成，任何形式的选择器都可以作为并集选择器的一部分。

语法：

元素1,

元素2,

元素3 {

样式声明

}

/\* 推荐写法，编码风格 \*/

上述语法表示选择元素1、元素2 和 元素3。

**伪类选择器**

伪类选择器 用于向某些选择器添加特殊的效果，比如：给链接添加特殊效果（链接伪类），或选择第 n 个元素（结构伪类）。 伪类选择器 书写最大的特点是用冒号 : 表示，比如：:hover、:first-child。 因为伪类选择器很多，比如：链接伪类、结构伪类 等，所以这里先讲解常用的链接伪类选择器。

伪类的由来：因为在页面中并没有这个真实存在的类，所以称为 “伪类”。

链接伪类选择器

链接伪类选择器注意事项：

为了确保生效且不冲突，请按照 LVHA 的顺序声明 :link，:visited，:hover，:active

记忆法：love hate 或者 lv 包包 hao

因为 a 链接在浏览器中具有默认样式，所以我们实际工作中都需要给链接单独指定样式

/\* a 是标签选择器（所有的链接）\*/

a {

color: gray;

}

/\* :hover 是链接伪类选择器（鼠标经过）\*/

a:hover {

color: red; /\* 鼠标经过的时候，由原来的灰色变成了红色 \*/

}

**:focus伪类选择器**

:focus 伪类选择器用于选取获得焦点的表单元素。

焦点就是光标，一般情况 <input> 类表单元素才能获取，因此这个选择器也主要针对于表单元素来说。

input:focus {

background-color: yellow;

}

元素显示模式

HTML 元素一般分为 块元素 和 行内元素 两种类型。

**块元素**

常见的块元素有 <h1> ~ <h6>、<p>、<div>、<ul>、<ol>、<li>、<dl>、<dt>、<dd>、<table>、<thead>、<tbody>、<tr>、<form> 等，其中 <div> 标签是最典型的块元素。

块级元素的特点：

比较霸道，自己独占一行

高度，宽度、外边距以及内边距都可以控制

宽度默认是容器（父级宽度）的 100%

是一个容器及盒子，里面可以放行内或者块级元素

注意：

文字类的块级元素内不能放置块级元素，会发生语法错误

<p> 标签主要用于存放文字，因此 <p> 里面不能放块级元素，特别是不能放 <div>

同理， <h1> ~ <h6> 等都是文字类块级标签，里面也不能放其他块级元素

**行内元素**

常见的行内元素有 <a>、<span>、<em>、<strong> 等，其中 <span> 标签是最典型的行内元素，有的地方也将行内元素称为内联元素。

行内元素的特点：

相邻行内元素在一行上，一行可以显示多个

高、宽直接设置是无效的

默认宽度就是它本身内容的宽度

行内元素只能容纳文本或其他行内元素（a 除外），链接里面不能再放链接

**行内块元素**

在行内元素中有几个特殊的标签：<img>、<input>、<th>、<td>，它们同时具有 块元素 和 行内元素 的特点，有些资料称它们为 行内块元素。

行内块元素的特点：

・和相邻行内元素（行内块）在一行上，一行可以显示多个（行内元素特点），但是他们之间会有空白缝隙。

・默认宽度就是它本身内容的宽度（行内元素特点）

・高度，行高、外边距以及内边距都可以控制（块级元素特点）

## 元素显示模式总结

| **元素模式** | **元素排列** | **设置样式** | **默认宽度** | **包含** |
| --- | --- | --- | --- | --- |
| 块级元素 | 一行只能放一个块级元素 | 可以设置宽度和高度 | 容器的 100% | 容量级可以包含任何标签 |
| 行内元素 | 一行可以放多个行内元素 | 不可以直接设置宽度和高度 | 它本身内容的宽度 | 容纳文本或其他行内元素 |
| 行内块元素 | 一行放多个行内块元素 | 可以设置宽度和高度 | 它本身内容的宽度 | 容纳文本或其他行内元素 |

元素显示模式转换

特殊情况下，我们需要元素模式的转换，简单理解: 一个模式的元素需要另外一种模式的特性

比如：想要增加链接 <a> 的触发范围。

转换为块元素：display: block;（由于经常需要设置宽高，所以通常会将行内元素转换为块元素）

转换为行内元素：display: inline;（不常用）

转换为行内块：display: inline-block;（常用）

**CSS 的背景**

通过 CSS 背景属性，可以给页面元素添加背景样式。 背景属性可以设置 背景颜色、背景图片、背景平铺、背景图片位置、背景图像固定 等。

**1，背景颜色**

background-color 属性定义了元素的背景颜色。

background-color: 颜色值;

**2，背景图片**

background-image 属性描述了元素的背景图像，实际开发常用于 logo 或者一些装饰性的小图片或者是超大的背景图片, 优点是非常便于控制位置。

background-image : none | url(路径)

参数值 作用

none 无背景图（默认的）

url 　　使用绝对或相对地址指定背景图像

**3，背景平铺**

如果需要在 HTML 页面上对背景图像进行平铺，可以使用 background-repeat 属性。

注：平铺可以简单的理解为“重复”。

background-repeat: repeat | no-repeat | repeat-x | repeat-y

参数值 　　　作用

repeat 　　背景图像在纵向和横向上平铺（默认的）

no-repeat 背景图像不平铺

repeat-x 背景图像在横向上平铺

repeat-y 背景图像在纵向上平铺

**4，背景图片位置**

利用 background-position 属性可以改变图片在背景中的位置。

background-position: x y;

参数代表的意思是：x 坐标 和 y 坐标，可以使用 方位名词 或者 精确单位。

参数值 　　　说明

length 　　　百分数 | 由浮点数字和单位标识符组成的长度值

position 　top | center | bottom | left | rigth 方位名词

参数是方位名词

・如果指定的两个值都是方位名词，则两个值前后顺序无关，比如 left top 和 top left 效果一致

・如果只指定了一个方位名词，另一个值省略，则**第二个值默认居中对齐**

参数是精确单位

・如果参数值是精确坐标，那么第一个肯定是 x 坐标，第二个一定是 y 坐标

・如果只指定一个数值，那该数值一定是 x 坐标，另一个**默认垂直居中**

参数是混合单位

・如果指定的两个值是精确单位和方位名词混合使用，则第一个值是 x 坐标，第二个值是 y 坐标

**5，背景图像固定（背景附着）**

background-attachment 属性设置背景图像是否固定或者随着页面的其余部分滚动。

background-attachment 后期可以制作 视差滚动 的效果。

background-attachment : scroll | fixed

参数 　　作用

scroll 　　背景图像是随对象内容滚动的（可见区域取决于背景图像的高度）

fixed 　　背景图像固定

背景复合写法

为了简化背景属性的代码，我们可以将这些属性合并简写在同一个属性 background 中，从而节约代码量。

当使用简写属性时，没有特定的书写顺序，一般习惯约定顺序为：

background: 背景颜色 背景图片地址 背景平铺 背景图像滚动 背景图片位置

background: transparent url(image.jpg) no-repeat fixed top;

**背景色半透明**

CSS3 为我们提供了背景颜色半透明的效果。

background: rgba(0, 0, 0, 0.3);

最后一个参数是 alpha 透明度，取值范围在 0~1 之间

习惯把 0.3 的 0 省略掉，写为 background: rgba(0, 0, 0, .3);

注意：背景半透明是指盒子背景半透明，盒子里面的内容不受影响

Emmet语法

本文档是个人对 Pink 老师课程的总结归纳及补充，转载请注明出处！

Emmet 语法的前身是 Zen coding，它使用缩写，来提高 html/css 的编写速度,，VSCode 内部已经集成该语法。

快速生成 HTML 结构语法

快速生成 CSS 样式语法

快速生成HTML结构语法

生成标签直接输入标签名按 tab 键即可，比如 div 然后 tab 键， 就可以生成 <div></div>

如果想要生成多个相同标签加上 \* 就可以了，比如 div\*3 就可以快速生成 3 个 div

如果有父子级关系的标签，可以用 > 比如 ul>li 就可以了

如果有兄弟关系的标签，用 + 就可以了，比如 div+p

如果生成带有 类名 或者 id 名字的标签， 直接写 标签.demo 或者 标签#demo 再按下 tab 键就可以了

如果生成的事物有顺序，可以用自增符号 $

如果想要在生成的标签内部写内容可以用 {} 表示

快速生成CSS样式语法

CSS 基本采取简写形式即可。

比如 w200 按 tab 可以 生成 width: 200px;

比如 lh26px 按 tab 可以生成 line-height: 26px;

快速格式化代码

VSCode 快速格式化代码：Shift + Alt + F。

WebStorm 快速格式化代码：Ctrl + Alt + L。

**CSS三大特性**

CSS 有三个非常重要的特性：层叠性、继承性、优先级。

**层叠性**

给同一个选择器设置相同的样式，此时一个样式就会覆盖（层叠）另一个冲突的样式，层叠性主要解决样式冲突的问题。

层叠性原则：

样式冲突，遵循的原则是 就近原则，哪个样式距离结构近，就执行哪个样式

样式不冲突，不会层叠

注：就近的标准是：后 > 前

**继承性**

现实中的继承：我们继承了父亲的姓。

CSS 中的继承：子标签会继承父标签的某些样式，如：文本颜色和字号，简单的理解就是：子承父业。

恰当地使用继承可以简化代码，降低 CSS 样式的复杂性

子元素可以继承父元素的样式（ text-、font-、line-、color ） 文本、字体、段落、颜色

行高的继承

body {

font: 12px/1.5 'Microsoft YaHei';

}

行高可以跟单位也可以不跟单位

如果子元素没有设置行高，则会继承父元素的行高为 1.5

此时子元素的行高是：当前元素的文字大小 \* 1.5

body 行高 1.5 这样写法最大的优势就是里面的子元素可以根据自己文字的大小自动调整行高

**优先级**

选择器相同，则执行层叠性

选择器不同，则根据选择器权重执行

选择器 　　　　　　　　选择器权重

继承 或 \* 　　　　　　　0,0,0,0

元素选择器 　　　　　　　0,0,0,1

类选择器、伪类选择器 　0,0,1,0

ID 选择器 　　　　　　　0,1,0,0

行内样式 style="" 　　　1,0,0,0

!important 重要的 　　　∞ 无穷大

优先级注意问题：

・权重是由 4 组数字组成的，但是不会有进位！

・可以理解为：类选择器 永远大于 元素选择器，ID 选择器 永远大于 类选择器，以此类推！

・等级判断 从左到右，如果某一位数值相同，则判断下一位数值

・可以简单的记忆：通配符 和 继承 权重为 0，标签选择器 为 1，类（伪类）选择器为 10，ID 选择器为 100，行内样式表 为 1000，!important 无穷大

・继承的权重是 0，不管父元素权重多高，子元素得到的权重都是 0 ！

・a 链接浏览器默认指定了一个样式，所以它不参与继承，所以设置样式需要选中单独设置

权重的叠加：

如果是复合选择器，则会有权重叠加，需要计算权重。

注意：再次强调！权重虽然会叠加，但一定不会进位！（1万个元素选择器也比不过一个类选择器）。

从左到右逐位比较，只有左一位同样大，才比较右边一位！

**选择器优先级总结**

CSS 选择器的优先级规则决定了当多个样式应用到同一元素时，哪一个样式最终会生效。优先级越高的选择器覆盖优先级较低的选择器。

CSS 的优先级规则是基于选择器的不同类型和特定性来计算的。优先级越高的选择器，其应用的样式将优先于优先级较低的选择器。

优先级计算规则：

CSS 优先级的计算是通过为选择器分配一个数字权重来决定的。这个权重通常表示为一个四部分的数字（A, B, C, D）：

其中：

A：内联样式（style 属性）权重。值为 1（如果有内联样式），否则为 0。

B：ID 选择器的数量。例如，#header 或 #main，每个 ID 选择器计为 1。

C：类选择器、伪类选择器、属性选择器的数量。例如，.class、:hover、[type="text"] 等，计为 1。

D：元素选择器和伪元素选择器的数量。例如，div、p、::before，每个计为 1。

计算优先级的步骤：

1,内联样式（style 属性）有最高优先级，优先级为 (1, 0, 0, 0)。

2,ID 选择器：每个 ID 选择器增加 B 的分数。例如，#header 和 #main 具有相同的优先级。

3,类选择器、伪类选择器、属性选择器：每个增加 C 的分数。例如，.nav、:hover 或 [type="text"]。

4,元素选择器和伪元素选择器：每个增加 D 的分数。例如，div、p 或 ::before。

计算优先级的小技巧：

内联样式（style）总是优先级最高。

ID 选择器 比类和元素选择器的优先级高。

类选择器 和 伪类选择器 的优先级比元素选择器高。

元素选择器 和 伪元素选择器 优先级最低。

层叠性：如果多个选择器具有相同的优先级，后面的规则会覆盖前面的规则（在样式表中靠后的规则优先）。

总结：

CSS 选择器的优先级计算是通过比较选择器的类型（内联、ID、类、元素等）来决定的。通过合理组织 CSS 规则，可以有效避免样式冲突和提高代码的可维护性。

**CSS盒子模型**

页面布局要学习三大核心：盒子模型、浮动和定位。

网页布局：

A，先准备好相关的网页元素，网页元素基本都是盒子

B，利用 CSS 设置好盒子样式，然后摆放到相应位置

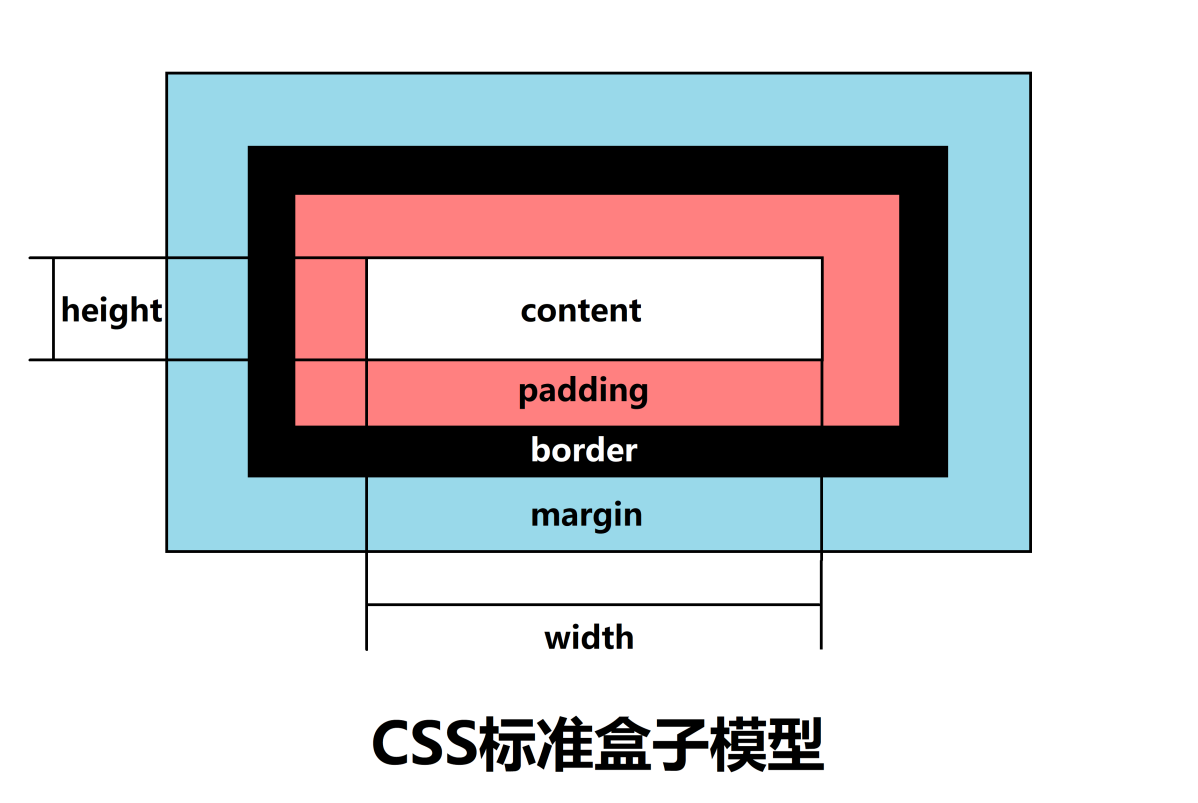
C，往盒子里面装内容

网页布局的核心本质： 就是利用 CSS 摆盒子！

盒子模型（Box Model）组成

所谓盒子模型：就是把 HTML 页面中的布局元素看作是一个矩形的盒子，也就是一个盛装内容的容器。

CSS 盒子模型本质上是一个盒子，封装周围的 HTML 元素，它包括：外边距、边框、内边距、和 内容。



**边框（border）**

border 可以设置元素的边框。

边框有三部分组成：边框宽度（粗细）、边框样式、边框颜色。

语法：

border: border-width || border-style || border-color

属性 　　　　　作用

border-width 定义边框粗细，单位是 px

border-style 边框的样式

border-color 边框颜色

边框的属性代码也可以简写

例如：border: 2px solid red;

这表示设置一个 2px 宽的 实线（solid）边框，颜色为 红色

边框样式 border-style 可以设置如下值：

none：没有边框，即忽略所有边框的宽度（默认值）

solid：边框为单实线（最为常用的）

dashed：边框为虚线

dotted：边框为点线

表格的细线边框

表格中两个单元格相邻的边框会重叠在一起，呈现出加粗的效果。

border-collapse 属性控制浏览器绘制表格边框的方式。

它控制相邻单元格的边框。

语法：

border-collapse: collapse;

collapse 单词是合并的意思

border-collapse: collapse; 表示相邻边框合并在一起

边框会影响盒子实际大小

边框会额外增加盒子的实际区域大小。因此我们有两种方案解决：

测量盒子大小的时候，不量边框

如果测量的时候包含了边框，则需要 width、height 减去边框宽度（注意减单边还是双边）

注意：盒子实际区域大小 = 内容区大小 + 内边距大小 + 边框大小 + 外边距大小

**内边距（padding）**

padding 属性用于设置内边距，即边框与内容之间的距离。

属性 作用

padding-left 左内边距

padding-rigth 右内边距

padding-top 上内边距

padding-bottom 下内边距

padding 属性（简写属性）可以有一到四个值。

值的个数 　　　　　　　　　　　　　表达意思

padding: 5px; 　　　　　　　　　1 个值，代表上下左右都有 5 像素内边距

padding: 5px 10px; 　　　　　　　2 个值，代表上下内边距是 5 像素，左右内边距是 10 像素

padding: 5px 10px 20px; 　　　3 个值，代码上内边距 5 像素，左右内边距 10 像素，下内边距 20 像素

padding: 5px 10px 20px 30px; 　4 个值，上是 5 像素，右 10 像素，下 20 像素，左是 30 像素（顺时针）

当我们给盒子指定 padding 值之后，发生了 2 件事情：

内容和边框有了距离，添加了内边距

padding 影响了盒子实际区域大小

也就是说，如果盒子已经有了宽度和高度，此时再指定内边距，会撑大盒子区域！

解决方案：

如果保证盒子跟效果图大小保持一致，则让 width、height 减去多出来的内边距大小即可

如果盒子本身没有指定 width、height 属性，则此时 padding 不会撑开盒子区域大小

**外边距（margin）**

margin 属性用于设置外边距，即控制盒子和盒子之间的距离。

属性 作用

margin-left 左外边距

margin-right 右外边距

margin-top 　　上外边距

margin-bottom 下外边距

\*margin 简写方式代表的意义跟 padding 完全一致。

外边距典型应用：

外边距可以让块级盒子水平居中，但是必须满足两个条件：

盒子必须指定了宽度 width

盒子左右的外边距都设置为 auto

.header { width: 960px; margin: 0 auto;}

常见的写法，以下三种都可以：

margin-left: auto; margin-right: auto;

margin: auto;

margin: 0 auto;

注意：以上方法是让块级元素水平居中，行内元素或者行内块元素水平居中给其父元素添加 text-align: center 即可。

**外边距合并**

使用 margin 定义块元素的垂直外边距时，可能会出现外边距的合并。

注意：内边距没有合并一说！浮动的盒子不会发生外边距合并！

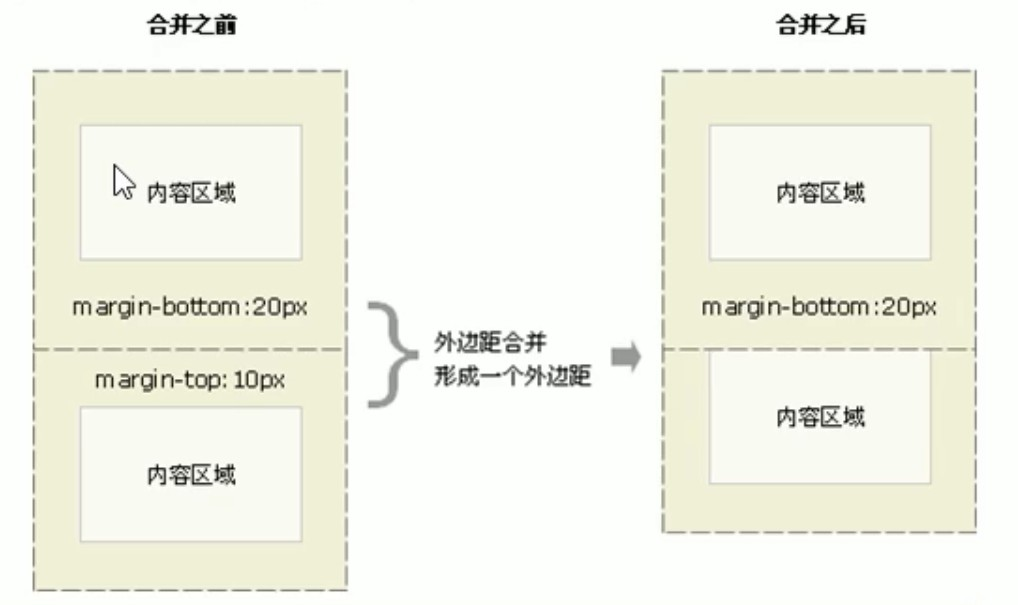
主要有两种情况:

1，相邻块元素垂直外边距的合并

2，嵌套块元素垂直外边距的塌陷

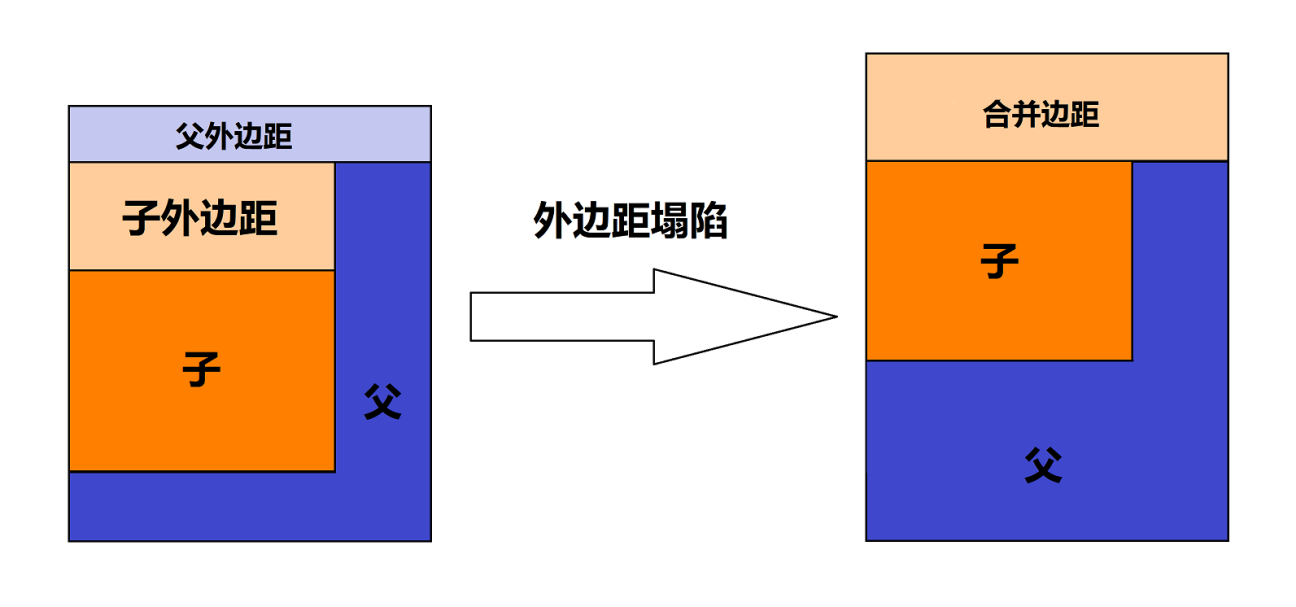
相邻块元素垂直外边距的合并

当上下相邻的两个块元素（兄弟关系）相遇时，如果上面的元素有下外边距 margin-bottom，下面的元素有上外边距 margin-top ，则他们之间的垂直间距不是 margin-bottom 与 margin-top 之和。而是取两个值中的较大者，这种现象被称为相邻块元素垂直外边距的合并（准确的描述应该是：大的外边距覆盖小的）



嵌套块元素垂直外边距的塌陷

对于两个嵌套关系（父子关系）的块元素，当子元素有上外边距，此时父元素会塌陷较大的外边距值（外边距效果显示在父元素之上）。



**清除内外边距**

网页元素很多都带有默认的内外边距，而且不同浏览器默认的也不一致。因此我们在布局前，首先要清除下网页元素的内外边距。

\* {

padding:0; /\* 清除内边距 \*/

margin:0; /\* 清除外边距 \*/

}

**圆角边框**

CSS 3 新增了圆角边框样式。

border-radius 属性用于设置元素的外边框圆角。

语法：

border-radius: length;

原理：

border-radius 顾名思义：边框半径。

注意：

参数值可以为数值或百分比的形式

如果是正方形，想要设置为圆形，那么只需要把数值修改为高度或者宽度的一半即可，或者直接写为 50%

如果是个矩形，设置为高度的一半就可以做 “胶囊” 效果了

该属性是一个简写属性，可以跟多个值

四个值：左上角、右上角、右下角、左下角（从左上开始顺时针）

三个值：左上、右上+左下、右下（对角为一组）

两个值：左上+右下、右上+左下（对角为一组）

同时可以对特定角单独设置

左上角：border-top-left-radius

右上角：border-top-right-radius

右下角：border-bottom-right-radius

左下角：border-bottom-left-radius

**盒子阴影**

CSS 3 新增了盒子阴影。

box-shadow 属性用于为盒子添加阴影。

语法：

box-shadow: h-shadow v-shadow blur spread color inset;

值 　　　　　　　　　　　　描述

h-shadow　　　必须。水平阴影的位置，允许负值。

v-shadow 　　　必须。垂直阴影的位置，允许负值。

blur 　　　　　可选。模糊距离（虚实程度）。

spread 　　　可选。阴影的尺寸（大小）。

color 　　　可选。阴影的颜色，请参阅 CSS 颜色值（阴影多为半透明颜色）。

inset 　　　　　可选。将外部阴影（outset）改为内部阴影（outset 不能指定，默认为空即可）。

**文字阴影**

CSS 3 新增了文字阴影。

text-shadow 属性用于为文本添加阴影。

语法：

text-shadow: h-shadow v-shadow blur color;

值 　　　　　　　　　描述

h-shadow 　　　　必须。水平阴影的位置。允许负值。

v-shadow 　　　　必须。垂直阴影的位置。允许负值。

blur 　　　　　　可选。模糊的距离（虚实程度）。

color 　　　　可选。阴影的颜色。参阅 CSS 颜色值。

**CSS浮动**

浮动（float）

1.1 传统网页布局的三种方式

网页布局的本质：用 CSS 来摆放盒子，把盒子摆放到相应位置。

CSS 提供了三种传统布局方式（简单说就是盒子如何进行排列）。

普通流（标准流）

浮动

定位

**标准流**（普通流/文档流）

所谓的标准流：就是标签按照规定好的默认方式排列。

块级元素会独占一行，从上向下顺序排列。

行内元素会按照顺序，从左到右顺序排列，碰到父元素边缘则自动换行。

以上都是标准流布局，我们前面学习的就是标准流，标准流是最基本的布局方式。

这三种布局方式都是用来摆放盒子的，盒子摆放到合适位置，布局自然就完成了。

\*\*注意：\*\*实际开发中，一个页面基本都包含了这三种布局方式（后面移动端学习新的布局方式） 。

**浮动**

有很多的布局效果，标准流没有办法完成，此时就可以利用浮动完成布局，因为浮动可以改变元素标签默认的排列方式。

浮动最典型的应用：可以让多个块级元素一行内排列显示。

网页布局第一准则：多个块级元素纵向排列找标准流，多个块级元素横向排列找浮动！

\*\*拓展：\*\*浮动的盒子不会发生外边距合并！

什么是浮动？

float 属性用于创建浮动框，将其移动到一边，直到左边缘或右边缘触及包含块或另一个浮动框的边缘。

语法：

选择器 { float: 属性值;}

属性 　　　　描述

none 　　　　元素不浮动（默认值）

left 　　　　元素向左浮动

right 　　　　元素向右浮动

**浮动特性（重难点）**

加了浮动之后的元素，会具有很多特性，需要我们掌握。

１　浮动元素会脱离标准流（脱标）

　　・脱离标准普通流的控制（浮） 移动到指定位置（动），（俗称脱标）

　　・浮动的盒子不再保留原先的位置

２　浮动的元素会一行内显示并且元素顶部对齐

　　・如果多个盒子都设置了浮动，则它们会按照属性值一行内显示并且顶端对齐排列。

　　・浮动的元素是互相贴靠在一起的（不会有缝隙），如果父级宽度装不下这些浮动的盒子，多出的盒子会另起一行对齐。

３　浮动的元素会具有行内块元素的特性

　　任何元素都可以浮动。不管原先是什么模式的元素，添加浮动之后具有行内块元素相似的特性。

　　块级盒子：没有设置宽度时默认宽度和父级一样宽，但是添加浮动后，它的大小根据内容来决定

　　行内盒子：宽度默认和内容一样宽，直接设置高宽无效，但是添加浮动后，它的大小可以直接设置

　　浮动的盒子中间是没有缝隙的，是紧挨着一起的

　　即：默认宽度由内容决定，同时支持指定高宽，盒子之间无空隙

**清除浮动**

语法：

选择器 { clear: 属性值; }

属性值 描述

left 不允许左侧有浮动元素（清除左侧浮动的影响）

right 不允许右侧有浮动元素（清除右侧浮动的影响）

both 同时清除左右两侧浮动的影响

我们实际工作中，几乎只用 clear: both;

清除浮动的策略是：闭合浮动。

**CSS Display(显示) 与 Visibility（可见性）**

隐藏元素语法：

display:none

或

visibility:hidden

display:none

完全隐藏元素，并且元素不再占据空间。

使用 display: none 会从页面布局中完全移除该元素，相当于它不存在。

visibility:hidden

设置元素不可见，但元素仍然占据空间。

即使元素不可见，页面布局不会发生变化，其他元素仍然会根据该元素的位置进行布局。

**定位**

为什么需要定位？

提问： 以下情况使用标准流或者浮动能实现吗？

某个元素可以自由的在一个盒子内移动位置，并且压住其他盒子。

当我们滚动窗口的时候，盒子是固定屏幕某个位置的。

以上效果，标准流或浮动都无法快速实现，此时需要定位来实现。

所以：

浮动可以让多个块级盒子一行没有缝隙排列显示， 经常用于横向排列盒子。

定位则是可以让盒子自由的在某个盒子内移动位置或者固定屏幕中某个位置，并且可以压住其他盒子。

静态定位 static（了解）

静态定位是元素的默认定位方式，无定位的意思。

语法：

选择器 { position: static; }

静态定位按照标准流特性摆放位置，它没有边偏移

静态定位在布局时很少用到

**相对定位 relative（重要）**

相对定位是元素在移动位置的时候相对于它原来的位置来说的定位（自恋型）。

语法：

选择器 { position: relative; }

相对定位的特点：（务必记住）

它是相对于自己原来的位置来移动的（移动位置的时候参照点是自己原来的位置点）

原来在标准流的位置继续占有，后面的盒子仍然以标准流的方式对待它

因此，相对定位并没有脱标。它最典型的应用是给绝对定位当爹的。

**绝对定位 absolute（重要）**

绝对定位是元素在移动位置的时候相对于它祖先元素来说的定位（拼爹型）。

语法：

选择器 { position: absolute; }

绝对定位的特点：（务必记住）

如果没有祖先元素或者祖先元素没有定位，则以浏览器为准定位（Document 文档）

如果祖先元素有定位（相对、绝对、固定定位），则以最近一级的有定位祖先元素为参考点移动位置

绝对定位不再占有原先的位置（脱标），并且脱标的程度大于浮动（会压住浮动）

所以绝对定位是脱离标准流的

**子绝父相的由来**

弄清楚这个口诀，就明白了绝对定位和相对定位的使用场景。

这个 “子绝父相” 太重要了，是我们学习定位的口诀，是定位中最常用的一种方式这句话的意思是：子级是绝对定位的话，父级要用相对定位。

子级绝对定位，不会占有位置，可以放到父盒子里面的任何一个地方，不会影响其他的兄弟盒子

父盒子需要加定位限制子盒子在父盒子内显示

父盒子布局时，需要占有位置，因此父亲只能是相对定位

重点：竖向上布局找标准流，横向上布局找浮动，空间上布局找定位！

**固定定位 fixed （重要）**

固定定位是元素固定于浏览器可视区的位置。

主要使用场景： 可以在浏览器页面滚动时元素的位置不会改变。

语法：

选择器 { position: fixed; }

固定定位的特点（务必记住）：

以浏览器的可视窗口为参照点移动元素

跟父元素没有任何关系

不随滚动条滚动

固定定位不再占有原先的位置

固定定位也是脱标的，其实固定定位也可以看做是一种特殊的绝对定位。

固定定位小技巧：固定在版心右侧位置

小算法：

让固定定位的盒子 left: 50%，走到浏览器可视区（也可以看做版心） 一半的位置

让固定定位的盒子 margin-left: 版心宽度的一半距离，多走版心宽度的一半位置

就可以让固定定位的盒子贴着版心右侧对齐了

粘性定位 sticky（了解）

粘性定位可以被认为是相对定位和固定定位的混合。

Sticky 粘性的。

语法：

选择器 { position: sticky; top: 10px; }

粘性定位的特点：

以浏览器的可视窗口为参照点移动元素（固定定位特点）

粘性定位占有原先的位置（相对定位特点）

必须添加 top 、left、right、bottom 其中一个才有效

跟页面滚动搭配使用。 兼容性较差，IE 不支持。

未来开发的趋势，但目前并不常用（目前用 javascript 来实现粘性定位效果）。

定位叠放次序 z-index

在使用定位布局时，可能会出现盒子重叠的情况。此时，可以使用 z-index 来控制盒子的前后次序（z轴）。

语法：

选择器 { z-index: 1; }

数值可以是正整数、负整数或 0，默认是 auto，数值越大，盒子越靠上

如果属性值相同，则按照书写顺序，后来居上

数字后面不能加单位

只有定位的盒子才有 z-index 属性

**CSS3过渡（重点）**

过渡（transition）是 CSS3 中具有颠覆性的特征之一，我们可以在不使用 Flash 动画或 JavaScript 的情况下，当元素从一种样式变换为另一种样式时为元素添加效果。

过渡动画：是从一个状态渐渐的过渡到另外一个状态。

可以让我们页面更好看，更动感十足，虽然低版本浏览器不支持（IE9 以下版本） 但是不会影响页面布局。

我们现在经常和 :hover 一起搭配使用。

语法：

transition: 要过渡的属性 花费时间 运动曲线 何时开始;

属性：想要变化的 css 属性，宽度高度、背景颜色、内外边距都可以 。如果想要所有的属性都变化过渡，写一个 all 就可以

花费时间：单位是秒（必须写单位）比如 0.5s

运动曲线：默认是 ease（可以省略）

何时开始：单位是秒（必须写单位）可以设置延迟触发时间默认是 0s（可以省略）

<title>CSS3过渡效果</title>

<style>

div {

width: 200px;

height: 100px;

background-color: black;

/\* transition: 变化的属性 花费时间 运动曲线 何时开始; \*/

/\* transition: width .5s ease 0s, height .5s ease 1s; \*/

/\* 如果想要写多个属性，利用逗号进行分割 \*/

/\* transition: width .5s, height .5s; \*/

/\* 如果想要多个属性都变化，属性写 all 就可以了 \*/

/\* transition: height .5s ease 1s; \*/

/\* 谁做过渡，给谁加 \*/

transition: all 0.5s;

}

div:hover {

/\* 以下是想要过渡到的效果; \*/

width: 400px;

height: 200px;

background-color: gray;

}

</style>