# mdn网址

https://developer.mozilla.org/zh-CN/

# html网页在线编辑器

https://www.w3schools.cn/tryit/tryit.asp?filename=tryhtml\_default

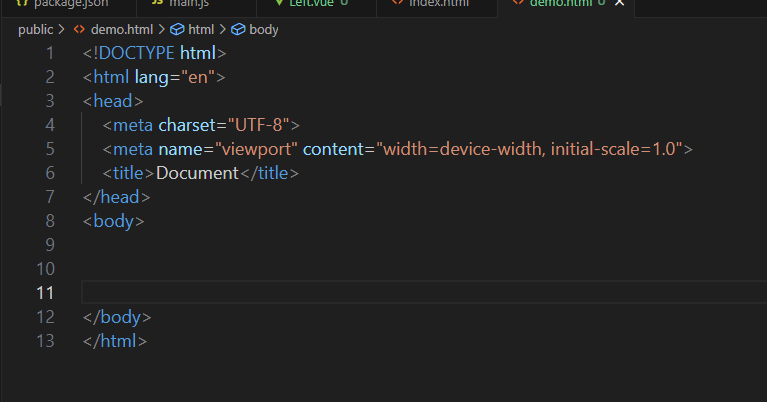
# 快捷键

## !!!

输出<!DOCTYPE html>

## !

输出下面的内容



# html注释

<!-- 这里是注释内容 -->

注释以 <!-- 开始，以 --> 结束，中间是注释的具体内容。注释可以跨越多行，并且可以包含任何文本、标签或其他内容。

# js注释

## 单行注释

使用双斜线 //开始，直到行末的所有内容都将被视为注释，不会被执行。

## 多行注释

使用斜线和星号 /\* 开始，使用星号和斜线 \*/ 结束，之间的所有内容都将被视为注释，不会被执行。

# <meta>

## 基本概念

<meta>标签是HTML中的元数据（metadata）标签，它用于提供关于HTML文档的信息，而不会直接显示在页面上。它位于<head>标签内部，用于告诉浏览器和搜索引擎一些关于网页的额外信息。

## 属性

### charset

指定HTML文档的字符编码。

例如：<meta charset="UTF-8">

表示文档使用UTF-8编码。

### name和content

用于定义各种元数据。例如：

<meta name="description" content="这是一个网页描述">

用于指定网页的描述。

### http-equiv和content

用于模拟HTTP头部字段。例如：

<meta http-equiv="refresh" content="5;url=[http://example.com](http://example.com/)">

用于在5秒后跳转到example.com网站。

### viewport

用于控制移动设备上的视口（viewport）。

例如：

<meta name="viewport" content="width=device-width, initial-scale=1.0">

用于适配移动设备的屏幕宽度。

# <script>

## 基本概念

用于在 HTML 文档中嵌入 JavaScript 代码或引入外部 JavaScript 文件。

## 用法

### 1、内联脚本

可以通过在 <script> 标签内直接编写 JavaScript 代码来创建内联脚本。例如：



在这种情况下，浏览器会解析并执行 <script> 标签内的 JavaScript 代码。

### 2、外部脚本

# <link>

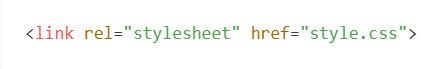
## 基本概念

用于在 HTML 文档中引入外部资源，主要包括 CSS 样式表和图标。

## 用法

### 1、引入外部 CSS 样式表

可以使用 <link> 标签来引入外部的 CSS 样式表文件，例如：



在上述示例中，rel 属性指定关系类型为 stylesheet，href 属性指定外部 CSS 文件的路径，这样就可以将样式表应用到 HTML 文档中。

### 2、指定样式表类型

可以使用 <link> 标签的 type 属性来指定外部资源的 MIME 类型。对于 CSS 样式表，可以使用 type="text/css"。例如：



### 3、引入网站图标

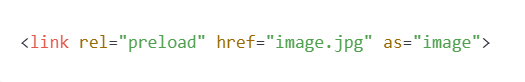
可以使用 <link> 标签来引入网站的图标，通常是一个 .ico 或 .png 文件，例如：



在上述示例中，rel 属性指定关系类型为 icon，href 属性指定图标文件的路径，这样就可以将指定的图标显示在浏览器标签页或书签栏上。

### 4、预加载资源

可以使用 <link> 标签的 rel 属性值为 preload 来指示浏览器预加载资源，以提高网页的加载速度。例如：



在上述示例中，href 属性指定要预加载的资源路径，as 属性指定资源类型，这样可以告诉浏览器在加载页面时提前下载指定的资源。

## 总结

HTML <link> 标签用于引入外部资源，包括 CSS 样式表和图标。通过 <link> 标签，可以将外部样式表应用到 HTML 文档中，指定图标文件以及进行预加载资源等操作，从而实现对网页样式和功能的扩展。

# <style>

## 基本概念

<style> 标签用于在 HTML 文档中定义内部样式表，允许您直接在 HTML 文件中编写 CSS 样式。

# <span>

## 基本概念

<span> 标签是一个行内元素，它通常用于对文本的一部分进行样式化或添加特殊处理，而不会对文本内容本身产生影响。

## 用法

<span> 标签用于在行内文本中创建一个容器，用于应用样式、脚本或其他行为。它不会对文本的布局产生影响，只是为其内容创建一个逻辑容器。

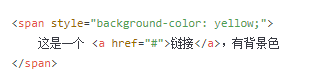
## 样式化

<span> 标签通常与CSS一起使用，用于对其中的文本进行样式化。你可以在 <span> 标签中使用内联样式或者将其与外部样式表结合使用，以实现特定的样式效果。



## 包裹其他元素

<span> 标签可以包裹其他HTML元素，比如链接、图片等。这在需要对特定部分进行样式化或操作时很有用。



## 脚本和事件

<span> 标签也可以用于包裹需要与JavaScript事件处理程序关联的文本。通过为 <span> 元素添加事件处理程序，可以在用户交互时触发相应的动作。



## 语义化

<span> 标签本身并不具有特定的语义，它主要用于提供样式化或脚本处理的钩子。因此，应该根据具体的语义需求来选择使用 <span> 标签还是其他更具语义的标签。

## 总结

总的来说，HTML <span> 标签是一个非常灵活的元素，可以用于对行内文本进行样式化、脚本处理或包裹其他元素，为开发者提供了更多的控制权和灵活性。

# <img>

## 基本概念

<img> 标签是 HTML 中用于插入图像的标签，它是一个空元素，没有闭合标签。

## 作用

<img> 标签用于在网页中插入图片，可以显示各种格式的图像文件，如 JPEG、PNG、GIF 等。

## 属性

### src

指定要显示的图像的URL。

<img src="example.jpg">

### alt

提供图像的替代文本，用于在图像无法显示时显示给用户。这对于视觉障碍用户和一些无法加载图像的环境（如网络问题）非常重要。

<img src="example.jpg" alt="这是一个示例图像">

### width

指定图像的宽度，以像素（px）为单位

<img src="example.jpg" width="200">

### height

指定图像的高度，以像素（px）为单位。

<img src="example.jpg" height="150">

### title

提供一个标题，当用户将鼠标悬停在图像上时会显示。

<img src="example.jpg" title="这是一个示例图像">

### loading(HTML5)

指定图像的加载方式，可以是 "eager"（立即加载）或 "lazy"（延迟加载，即当图像进入视口时才加载）。

<img src="example.jpg" loading="lazy">

# <ul>

## 基本概念

<ul>标签是HTML中用于创建无序列表（unordered list）的元素。无序列表是一种用于显示项目之间没有特定顺序关系的列表。<ul>标签通常与<li>标签配合使用，<li>标签用于定义列表中的每个项目。列表项目会按照它们在HTML代码中的顺序进行显示，并且每个项目前面会有一个符号（通常是圆点）作为项目的标记。

# <ol>

## 基本概念

<ol> 标签用来创建有序列表（Ordered List），即按照一定顺序排列的列表项。

## 属性

### type

指定列表项的计数类型，可以是数字（1, 2, 3...）、大写字母（A, B, C...）、小写字母（a, b, c...）、罗马数字（I, II, III...）或罗马数字小写（i, ii, iii...）。

### start

指定起始值，即列表中第一个项的值。

### reversed

如果设置了该属性，则列表将以递减顺序计数。

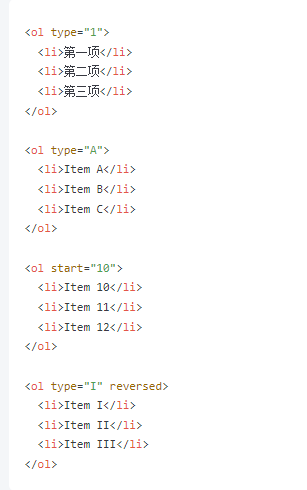
## 注意事项

1、默认情况下，列表项将按照数字顺序进行编号。

2、可以嵌套其他列表，如嵌套 <ul> 或 <ol> 来创建多层次的列表结构。

3、通常与 <li> 标签一起使用，<li> 标签用来定义列表中的每个项目。

## 示例代码



# <li>

## 基本概念

<li> 标签用于在无序列表（<ul>）或有序列表（<ol>）中定义每个列表项。

## 属性

<li> 标签通常没有特定的属性，其作用主要是定义列表中的每个项目。

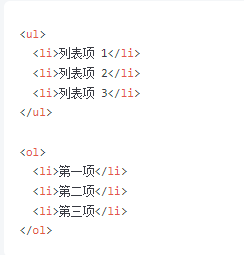
## 注意事项

1、在使用无序列表时（<ul>），每个 <li> 标签会显示为一个带有符号（如圆点、方块等）的项目。

2、在使用有序列表时（<ol>），每个 <li> 标签会按照指定的顺序编号显示。

3、可以在 <li> 标签内部嵌套其他 HTML 元素，如文本、图片、链接等，来丰富列表项的内容。

## 示例代码



# <i>

## 基本概念

<i> 标签用于表示文本中的斜体（italic）样式。它是一个内联元素，常用于对单词、短语或句子的某部分进行强调、突出显示或标记。

需要注意的是，<i> 标签仅仅表示文本的视觉样式，而不代表特定的语义含义。在 HTML5 中，推荐使用 <em> 标签来表示强调语气或重要性，而不是仅仅使用 <i> 标签来表示斜体。<em> 标签除了应用斜体样式外，还提供了语义上的强调。

总结起来，<i> 标签用于在 HTML 文档中将文本显示为斜体。它是一个内联元素，用于强调、突出显示或标记文本的一部分。在 HTML5 中，更推荐使用 <em> 标签来表示强调语气或重要性。

# <label>

## 基本概念

在 HTML 中，<label> 元素用于为表单控件（如输入框、复选框、单选框等）定义标签。它的作用是提供对用户更好的可访问性和用户体验，让用户知道每个表单控件代表什么内容。

## 功能

### 关联表单控件

<label> 元素通过与表单控件的 id 属性进行关联，可以使用户点击标签时自动聚焦到相关的表单控件，提高表单的易用性。

### 表单标签

1、<label> 元素通常用于包裹文本内容，作为表单控件的标签，帮助用户理解每个表单控件的用途。

2、用户点击 <label> 元素时，关联的表单控件会获得焦点或选中，增强了表单的交互性。

## 属性

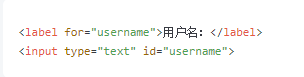
### for

指定与 <label> 元素关联的表单控件的 id 属性值。

<label for="username">用户名：</label>

## 使用方法

1、使用 for 属性关联：



2、使用嵌套关系关联



## 注意事项

1、使用 <label> 标签能够提高表单的可访问性，特别是对于视力受损用户或使用辅助技术的用户。

2、建议使用 <label> 标签来描述表单控件，而不是仅依靠控件本身的标题或提示信息。

3、在使用 <label> 标签时，要确保其文本内容清晰明了，能够准确描述相应的表单控件。

## 总结

通过合理使用 <label> 标签，可以提高网页表单的易用性和可访问性，使用户更容易理解和操作表单控件。

# <select>

## 基本概念

<select>标签是用于创建下拉列表框的元素，它允许用户从一组选项中选择一个或多个选项。<select>标签必须与<option>标签一起使用，<option>标签定义下拉列表中的每个选项。

## 属性

### name

指定下拉列表的名称

### size

可选属性，规定下拉列表中可见的行数

### multiple

可选属性，如果存在则允许用户选择多个选项。

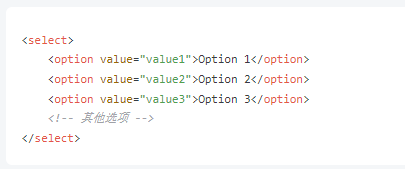
### disabled

可选属性，禁用下拉列表，使其不可选。

### form

规定 <select> 元素所属的表单，允许将 <select> 元素与表单外的表单控件关联。

## 示例



# <option>

## 基本概念

用于定义下拉列表中的选项。<option> 标签之间的文本内容是选项的显示文本。

## 属性

### value

规定选项的值，在提交表单时会发送给服务器。

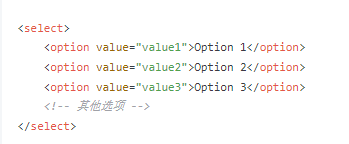
### selected

可选属性，规定默认选中的选项。

### disabled

可选属性，禁用选项，使其不可选。

## 示例



# <template>

## 基本概念

<template> 标签是 HTML5 中的一个新标签，用于定义文档片段或模板。它通常用于存放 HTML 页面中的可复用部分，比如组件、模块或页面的结构。

## 作用

### 1、定义模板

<template> 标签可以用来定义一个模板或文档片段，并将其中的内容作为一整块进行处理。在这个标签中可以包含任何合法的 HTML 内容，包括标签、文本、属性等。

### 2、隐藏内容

<template> 标签中的内容在页面加载时不会显示出来，也不会被浏览器渲染。这意味着你可以在模板中编写和组织 HTML 结构，而无需担心它会在页面中显示出来。

### 3、可复用性

<template> 标签中的内容可以被多次使用，通过 JavaScript 或者其他方式将其插入到页面中。这使得开发者可以将一些常见的 HTML 结构或代码片段封装为模板，以便在不同的地方进行重复使用。

### 4、样式隔离

<template> 标签中的内容具有作用域隔离的特性。这意味着在模板内定义的 CSS 样式仅适用于模板内部，不会影响到外部的样式。这有助于确保模板内的样式不会与页面其他部分产生冲突。

### 5、JavaScript 访问

通过 JavaScript可以选择性地获取 <template> 标签中的内容，并对其进行处理或操作。可以使用 content 属性或 querySelector 方法来获取模板中的内容，并根据需要进行动态修改。

### 6、浏览器支持

<template> 标签在现代浏览器中得到了广泛的支持，包括 Chrome、Firefox、Safari 等主要浏览器。对于不支持的旧版浏览器，它们会将 <template> 标签中的内容作为普通文本处理，因此仍然可以保持基本的兼容性。

## 属性

### id

给模板一个唯一标识，可通过 JS 获取

### class

添加样式类（一般无实际作用）

### type

可指定 MIME 类型（如 text/x-template，常见于 Vue 模板）

## 其他说明

vue项目中的.vue文件中都是使用了<template>标签去编写模版内容，那为什么这些内容会在页面中显示出来。

Vue 中的 <template> 标签和原生 HTML 的 <template> 标签虽然名字相同，但**完全是两个不同的概念**。

在 .vue 文件中，<template> 是 Vue 的**模板语法标识符**，不是原生 HTML 元素。



关键区别：

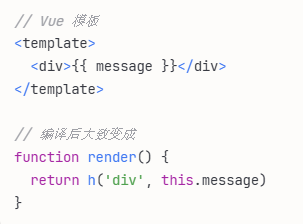
1、处理时机不同

（1）原生 <template>：在浏览器中直接解析，默认隐藏

（2）Vue <template>：在构建阶段被 Vue 编译器处理，转换为 JavaScript 代码。

2、编译过程

Vue 的构建工具（如 Vite、Webpack）会



3、最终输出

Vue 组件最终会生成真实的 DOM 元素



原生 <template>：浏览器特性，内容不显示，需要 JS 手动克隆使用

Vue <template>：Vue 框架的语法糖，用于定义组件模板，会被编译成渲染函数。

Vue 选择用 <template> 这个名字是因为它确实在定义"模板"，但实现机制完全不同于原生 HTML 的 <template> 标签。

## 总结

<template> 标签是 HTML5 中用于定义模板或文档片段的标签。它提供了一种组织和复用 HTML 结构的机制，并具有隐藏内容、样式隔离和 JavaScript 访问等特性。通过使用 <template> 标签，可以更好地管理和组织 HTML 代码，提高代码的可读性和可维护性。

# <div>

## 语法

<div>内容</div>

## 作用

作为网页中的通用容器，用于布局和分组内容。

## 特点

块级元素，占满一整行，可嵌套其他标签。

## 属性

### id

唯一标识

### class

添加样式类

### style

内联样式

### title

鼠标悬停提示

## 示例代码



# DOM对象

## 基本概念

HTML DOM（Document Object Model）指的是 HTML 文档对象模型，它是浏览器将 HTML 文档解析成树状结构的方式。HTML DOM 将 HTML 文档中的每个元素（如标签、属性、文本内容等）都看作是一个对象，并且这些对象之间存在父子关系，形成了一个层级结构。HTML DOM 提供了一种通过 JavaScript 或其他编程语言来访问和操作 HTML 元素的标准接口。通过 HTML DOM，开发者可以轻松地使用脚本语言来改变页面的内容、结构和样式，实现动态交互效果。

## 重要概念

### 节点

在 HTML DOM 中，每个 HTML 元素、属性、文本均被视为一个节点。节点之间存在父子、兄弟等关系，形成了一个树状结构。

### 元素节点

HTML 中的标签（如 <div>、<p> 等）被表示为元素节点，其可以拥有子节点。

### 属性节点

HTML 元素的属性被表示为属性节点，开发者可以通过 DOM 操作属性，如修改元素的 class、id 等属性。

### 文本节点

HTML 中的文本内容被表示为文本节点，如 <p>这是文本内容</p> 中的“这是文本内容”就是一个文本节点。

### DOM方法和属性

HTML DOM 提供了众多方法和属性，用于访问和操作 HTML 元素，如 getElementById()、appendChild()、innerHTML 等。

## 方法

### getElementById(id)

根据ID查找元素

### querySelector(selector)

根据CSS选择器查找第一个匹配元素

### querySelectorAll(selector)

查找所有匹配的元素

### createElement(tagName)

创建新元素

### appendChild(child)

添加子元素

### removeChild(child)

删除子元素

### remove()

直接删除元素

### getAttribute(name)

获取属性值

### setAttribute(name, value)

设置属性

### hasAttribute(name)

检查是否有某属性

### innerHTML

获取/设置HTML内容

### textContent

获取/设置文本内容

### innerText

获取/设置可见文本

### style.property

直接设置样式

### classList.add()

添加CSS类

### addEventListener(type, handler)

添加事件监听

### removeEventListener(type, handler)

移出事件监听

### parentNode

父节点

### childNodes

所有子节点

### firstChild

第一个子节点

### lastChild

最后一个子节点

### nextSibling

下一个兄弟节点

### previousSibling

上一个兄弟节点

# CSS

## 选择器

### type选择器

#### 基本概念

根据标签名进行匹配（元素选择器）

#### 示例代码



### class选择器

#### 基本概念

根据元素的class属性进行匹配。class取值可以重复

#### 语法

.类名 { 样式 }

#### 示例代码



### id选择器

#### 基本概念

根据元素的id属性进行匹配，id取值是唯一的，不能重复。

#### 语法

#id名 { 样式 }

#### 示例代码



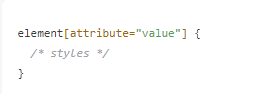
### 属性选择器

#### 基本概念

CSS 属性选择器允许您根据元素的属性及其相应的值来选择元素，并对其应用样式。

#### 语法

属性选择器使用方括号（[]）来定义，其基本语法如下：



element：表示要选择的 HTML 元素，可以是元素的标签名，也可以为空表示选择所有元素。

attribute：表示要匹配的属性名。

value：表示要匹配的属性值。

#### 示例代码

假设我们有如下 HTML 元素：



我们可以使用属性选择器来选择具有特定属性及其对应值的元素，比如：



#### 特殊匹配

属性选择器还支持一些特殊的匹配方式，如：

存在和值属性选择器：[attribute] 选择具有指定属性的元素，而不论其值如何。

部分值匹配：[attribute\*="value"] 选择属性值中包含指定值的元素。

以某值开头：[attribute^="value"] 选择属性值以指定值开头的元素。

以某值结尾：[attribute$="value"] 选择属性值以指定值结尾的元素。

##### 示例

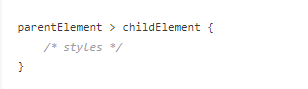


### 子元素选择器

#### 基本概念

CSS子元素选择器用于选择某个元素的直接子元素。

#### 语法



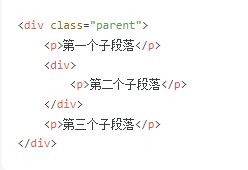
1、parentElement 是要选择的父元素的选择器。

2、childElement 是要选择的子元素的选择器。

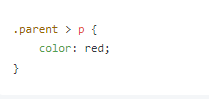
3、> 符号用于表示子元素选择器。

#### 示例

假设我们有以下 HTML 结构：



如果我们想选择 .parent 下直接的 p 元素，可以使用子元素选择器：



这样，只有 .parent 下直接的 p 元素会被选择并应用样式，第二个子段落不会被选择，因为它被一个 div 包裹着。

#### 注意事项

1、子元素选择器 > 只会选择父元素下直接的子元素，不会选择孙子元素或更深层次的后代元素。

2、子元素选择器对于网页布局和样式控制非常有用，可以精确地选择特定层次的元素应用样式。

3、在实际应用中，要确保选择器的准确性，避免选择到不必要的元素，以提高代码的可维护性和性能。

## 选择器的优先级

id选择器>class选择器>type选择器

## 属性和值

### display

### background-color

## 示例代码



## 布局

### div+css

### template

# Less

## 基本概念

Less 是一种 **CSS 预处理语言**，在 CSS 基础上增加了编程能力，如变量、嵌套、函数等，最终会被编译成标准 CSS。

## 安装

在服务器端最容易的安装方式就是通过 npm （[node.js](http://nodejs.org/) 的包管理器），方法如下：

npm install -g less

## 查看less的版本

lessc --version

## 注释



## 嵌套规则

### 基本概念

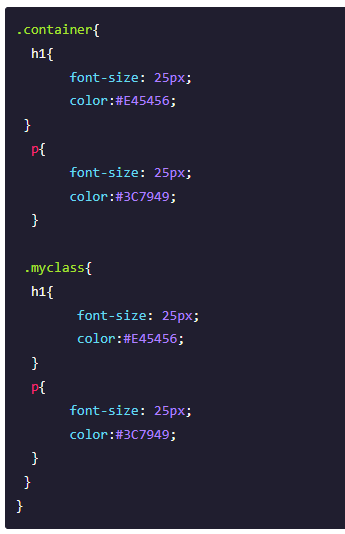
它是一组 CSS 属性，允许将一个类的属性用于另一个类，并且包含类名作为其属性。 在 LESS 中，可以使用类或 id 选择器以与 CSS 样式相同的方式声明 mixin。 它可以存储多个值，并且可以在必要时在代码中重复使用。

### 示例代码

#### 1、创建html文件



#### 2、创建style.less文件



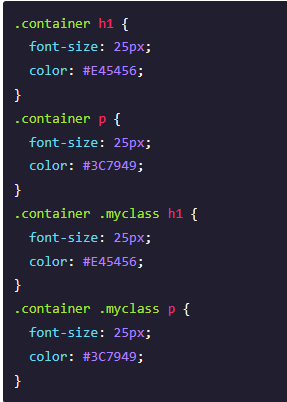
#### 3、编译less文件

使用下面的命令将style.less 文件编译为 style.css文件

lessc style.less style.css

#### 4、编译后的文件

style.css文件

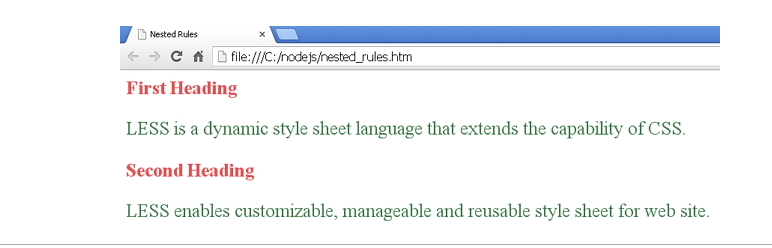


#### 5、输出

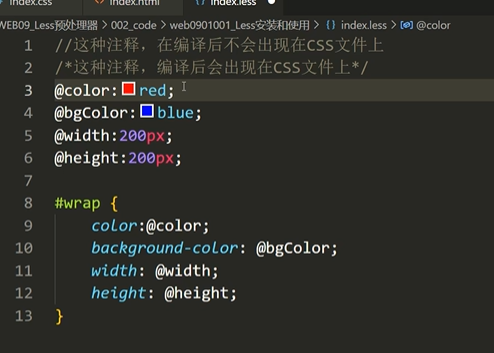
让我们执行以下步骤，看看上面的代码如何工作:

将上述 html 代码保存在 nested\_rules.html 文件中。

在浏览器中打开此 HTML 文件，将显示如下输出。



## 变量





### 选择器变量

### 语法

### 属性变量

# 怎么调试前端代码

## 快捷键

### F8

暂停脚本执行（继续执行脚本）

### F10

跳过下一个函数调用

### F11

进入下一个函数调用

### shift+F11

跳出当前函数

### F9

单步调试

### ctrl+F8

停用断点