河南中医药大学信息技术学院（智能医疗行业学院）智能医学工程专业《互联网医疗服务开发》课程

第03章：CSS

冯顺磊

河南中医药大学信息技术学院（智能医疗行业学院）

河南中医药大学信息技术学院智能医疗教研室

https://aitcm.hactcm.edu.cn

2025/2/27

本章概要

- 基础
- 选择器
- 文字样式
- 背景与边框
- 动画

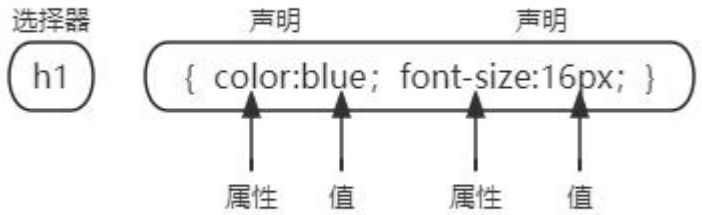
1. 基础：1.1 CSS语法

- CSS（Cascading Style Sheets，层叠样式表）是一种样式表语言，用来描述HTML或XML（包括如SVG、MathML或XHTML之类的XML分支语言）文档的呈现方式。

```css
body {
    background-color:#d0e4fe;
}
h1 {
    color:orange;
    text-align:center;
}
p {
    font-family:"Times New Roman";
    font-size:20px;
}
```

1. 基础：1.1 CSS语法

- CSS规则由两个主要部分组成：选择器和声明（1条或多条）。选择器通常是需要改变样式的HTML元素。声明由属性和值组成，属性和值被冒号分隔开。



1. 基础：1.1 CSS语法

- CSS注释是用来解释代码，浏览器会忽略注释代码，CSS注释以“/*”开始，以“*/”结束。

```css
/* 这是个注释 */
p{
    text-align:center;
    /* 这是另一个注释 */
    color:black;
    font-family:arial;
}
```

1. 基础：1.1 CSS语法

- HTML中引入CSS有4种方式，分别是内联方式、嵌入方式、链接方式、导入方式。
  - 内联方式
    - 内联方式是在HTML元素中的“style”属性中添加CSS。
    - 内联方式添加的CSS样式只改变当前元素的样式，若想要多个元素拥有同样的样式，需要重复地为每个元素添加相同的样式。

```
<div style="background: red"></div>
```

1. 基础：1.1 CSS语法

- HTML中引入CSS有4种方式，分别是内联方式、嵌入方式、链接方式、导入方式。
  - 嵌入方式
    - 嵌入方式是在HTML头部中“<style>”元素下书写CSS代码。
    - 嵌入方式的CSS代码是在HTML文件中，CSS代码会比较集中，可以一目了然地查看HTML结构和CSS样式，嵌入方式的CSS代码只对当前网页有效。

```
<head>
    <style>
    .content {
        background:red;
    }
    </style>
</head>
```

1. 基础：1.1 CSS语法

- HTML中引入CSS有4种方式，分别是内联方式、嵌入方式、链接方式、导入方式。
  - 链接方式
    - 链接方式是在HTML头部的“<head>”标签中引入外部的CSS文件。
    - 使用链接方式，所有的CSS代码存放于单独的CSS文件中，具有良好的可维护性。CSS文件会在第一次加载时引入，后续切换页面时只需加载HTML文件即可。

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

1. 基础：1.1 CSS语法

- HTML中引入CSS有4种方式，分别是内联方式、嵌入方式、链接方式、导入方式。
  - 导入方式
    - 导入方式是使用CSS规则引入外部CSS文件。

```
<style>
    @import url(style.css);
</style>
```

2. 选择器：2.1 CSS选择器

- CSS选择器是CSS的一个核心组成部分，用于在HTML或XML文档中定位和选择元素，以便应用样式规则。
- CSS选择器指定了哪些元素应当被样式化，一个CSS选择器通常由一个或多个简单的选择器组合而成，每个选择器可以基于元素的名称、属性、伪类、伪元素或这些的任意组合来定位元素。常见的选择器有元素选择器（Element Selector）、类选择器（Class Selector）、ID 选择器（ID Selector）、属性选择器（Attribute Selector）等。
- CSS的选择器类型包括：基础选择器、层次选择器、伪类选择器、伪元素选择器、属性选择器。

2. 选择器： 2.2 基础选择器

- 通配符选择器
  - 通配符选择器使用星号 * 表示，它可以匹配 HTML 文档中的所有元素。
  - 性能影响：由于通配符选择器会匹配页面上的所有元素，当页面元素较多时，频繁使用可能会对页面性能产生一定影响。因此，在实际开发中应谨慎使用，尤其是在大型项目中。
  - 优先级问题：通配符选择器的优先级较低。如果某个元素同时被通配符选择器和其他更具体的选择器选中，其他选择器的样式会覆盖通配符选择器的样式。

示例3-2-1：通配符选择器

2. 选择器：2.2 基础选择器

- 元素选择器
  - 元素选择器是根据 HTML 元素的名称来选择页面中的元素，它会选中文档中所有指定类型的元素，并将对应的 CSS 样式应用到这些元素上。
  - 元素选择器的优先级相对较低。如果一个元素同时被元素选择器和其他优先级更高的选择器（如类选择器、ID 选择器）选中，那么高优先级选择器的样式会覆盖元素选择器的样式。
  - 当需要为页面中某一类型的所有元素设置统一的基础样式时，元素选择器非常实用。例如，为所有的 <body> 元素设置默认的字体、背景颜色等。
  - 快速样式调整：在开发过程中，如果需要快速对某种元素的样式进行整体调整，使用元素选择器可以方便快捷地实现。比如，要将所有的 <a> 元素（链接）的下划线去掉。

示例3-2-2：元素选择器

2. 选择器：2.2 基础选择器

- 类选择器
  - 类选择器允许你根据元素的 class 属性值来选择 HTML 元素。在 HTML 中，你可以为一个或多个元素添加 class 属性，并给它赋予一个特定的类名；在 CSS 里，通过在类名前加上点号（.）来创建类选择器。
  - 类选择器的优先级高于元素选择器，但低于 ID 选择器。如果一个元素同时被元素选择器、类选择器和 ID 选择器选中，ID 选择器的样式会覆盖类选择器的样式，类选择器的样式会覆盖元素选择器的样式。
  - 复用样式：当多个元素需要应用相同的样式时，使用类选择器可以避免代码重复。例如，在一个网站的导航栏中，多个导航项可能需要相同的字体、颜色和背景样式，就可以为这些导航项添加同一个类名并使用类选择器设置样式。
  - 模块化设计：在前端开发中，常常会将页面拆分成多个模块，每个模块有自己独特的样式。可以通过类选择器为每个模块的元素设置样式，使代码结构更清晰，易于维护和扩展。

示例3-2-3：类选择器

2. 选择器：2.2 基础选择器

- ID选择器
  - 在 HTML 中，每个元素的 id 属性值应当是唯一的，即一个页面里不能有两个元素拥有相同的 id。ID 选择器就是依据元素的 id 属性值来选取元素的。在 CSS 里，ID 选择器以井号（#）开头，后面紧跟 id 属性值。
  - ID 选择器的优先级高于类选择器和元素选择器。当一个元素同时被 ID 选择器、类选择器和元素选择器选中时，ID 选择器的样式会覆盖类选择器和元素选择器的样式。
  - 唯一性：HTML 元素的 id 属性值必须是唯一的。若违反此规则，可能会引发一些不可预期的问题，比如 JavaScript 操作元素时出现混乱。
  - 性能影响：尽管 ID 选择器的性能通常不错，但如果过度依赖 ID 选择器，会让 CSS 代码的可维护性变差，因为 ID 选择器的针对性过强，不利于样式的复用。
  - 页面布局：在页面布局中，可使用 ID 选择器为一些关键的结构元素设置样式，如页面头部、侧边栏、页脚等。
  - JavaScript 交互：ID 选择器常与 JavaScript 配合使用，因为通过 id 能方便地在 JavaScript 中获取特定的元素。例如，在 JavaScript 里可以使用 document.getElementById('elementId') 来获取指定 id 的元素。

示例3-2-4：ID选择器

2. 选择器： 2.3 层次选择器

- 后代选择器与子选择器
  - 后代选择器用于选择某个元素的所有后代元素（包括子元素、孙元素等）。它由两个或多个选择器组成，选择器之间用空格分隔。
  - 子选择器用于选择某个元素的直接子元素，即只选择该元素下一级的特定子元素。

示例3-2-5：后代选择器与子选择器

2. 选择器： 2.3 层次选择器

- 兄弟选择器
  - 相邻兄弟选择器用于选择紧接在另一个元素之后的同级元素。两个元素必须有相同的父元素，且第二个元素必须紧跟在第一个元素之后。
  - 通用兄弟选择器用于选择某个元素之后的所有同级兄弟元素。两个元素必须有相同的父元素，但第二个元素不需要紧跟在第一个元素之后。

示例3-2-6：兄弟选择器

2. 选择器： 2.4 伪类选择器

- 链接与行为伪类选择器
  - 用于处理链接的不同状态

伪类	说明
:link	用于选取未被访问过的链接
:hover	当鼠标指针悬浮在元素上时，该元素会被选取
:active	在元素被激活（通常是鼠标按下但未释放）的瞬间选取该元素
:visited	选取已经被访问过的链接。不过出于隐私考量，浏览器对该伪类可修改的样式有所限制
:focus	当元素获得焦点（如输入框被点击）时选取该元素

示例3-2-7：超链接的伪类选择器

2. 选择器： 2.4 伪类选择器

- 结构伪类选择器
  - 依据元素在文档树中的位置来选取元素

伪类	说明	伪类	说明
:first-child	选取父元素的第一个子元素	:last-of-type	选取父元素下特定类型的最后一个元素。
:last-child	选取父元素的最后一个子元素。	:nth-of-type(n)	选取父元素下特定类型的第 n 个元素
:nth-child(n)	选取父元素的第 n 个子元素，n 可以是数字、关键字（even 表示偶数，odd 表示奇数）或公式（如 2n+1）。	:nth-last-of-type(n)	从父元素下特定类型的最后一个元素开始计数，选取第 n 个元素。
:nth-last-child(n)	从父元素的最后一个子元素开始计数，选取第 n 个子元素。	:only-child	若元素是其父元素的唯一子元素，则选取该元素。
:first-of-type	选取父元素下特定类型的第一个元素。	:only-of-type	若元素是其父元素下特定类型的唯一元素，则选取该元素。

示例3-2-8：结构伪类选择器

2. 选择器：2.4 伪类选择器

- 目标伪类选择器
  - :target：选取当前活动的目标元素（例如点击锚点链接后对应的目标元素）。

示例3-2-9：目标伪类选择器

2. 选择器： 2.4 伪类选择器

- 语言伪类选择器
  - :lang()：根据元素的语言代码来选取元素。

示例3-2-10：语言伪类选择器

2. 选择器： 2.4 伪类选择器

- 表单相关伪类选择器

伪类	说明	伪类	说明
:enabled	选取可用（未被禁用）的表单元素。	:out-of-range	当输入框的值超出规定的范围时选取该输入框。
:disabled	选取被禁用的表单元素。	:required	选取设置了 required 属性的表单元素。
:checked	选取被选中的表单元素（如单选框、复选框）。	:optional	选取未设置 required 属性的表单元素
:indeterminate	选取处于不确定状态的表单元素（如一组复选框中部分被选中）。	:read-only	选取设置了 readonly 属性的表单元素。
:default	选取表单中默认选中的元素。	:read-write	选取可读写的表单元素（未设置 readonly 属性）。
:valid	选取满足表单验证规则的表单元素。	:not()	选取不满足指定选择器的元素。
:invalid	选取不满足表单验证规则的表单元素。	:is()	选取匹配括号内任意一个选择器的元素。
:in-range	当输入框的值在规定的范围之内时选取该输入框。	:where()	与 :is() 类似，但 :where() 的优先级始终为 0。

示例3-2-11：表单相关伪类选择器

2. 选择器： 2.4 伪类选择器

- 逻辑组合伪类选择器

伪类	说明
:not()	选取不满足指定选择器的元素。
:is()	选取匹配括号内任意一个选择器的元素。
:where()	与 :is() 类似，但 :where() 的优先级始终为 0。

示例3-2-12：逻辑组合伪类选择器

2. 选择器： 2.4 伪类选择器

- 空伪类选择器
  - :empty：选取没有子元素（包括文本节点）的元素。

示例3-2-13：空伪类选择器

2. 选择器：2.4 伪类选择器

- 根伪类选择器
  - :root：选取文档的根元素，在 HTML 中即为 <html> 元素。

示例3-2-14：根伪类选择器

2. 选择器：2.5 伪元素选择器

- 伪元素选择器
  - ::before 和 ::after 伪元素用于在所选元素的内容前面或后面插入内容。通常需要配合 content 属性使用，插入的内容默认是行内元素。

示例3-2-15：::before 和 ::after

2. 选择器： 2.5 伪元素选择器

- 伪元素选择器
  - ::first-letter 伪元素用于选择元素文本的第一个字母，并为其设置样式。

示例3-2-16：::first-letter

2. 选择器： 2.5 伪元素选择器

- 伪元素选择器
  - ::first-line 伪元素用于选择元素文本的第一行，并为其设置样式。

示例3-2-17：::first-line

2. 选择器： 2.5 伪元素选择器

- 伪元素选择器
  - ::selection 伪元素用于选择用户选中的文本部分，并为其设置样式。

示例3-2-18：::selection

2. 选择器： 2.5 伪元素选择器

- 伪元素选择器
  - ::marker 伪元素用于选择列表项的标记（如有序列表的数字、无序列表的圆点），并为其设置样式。

示例3-2-19：::marker

2. 选择器： 2.6 属性选择器

- 属性选择器
  - 属性选择器允许你根据元素的属性或属性值来选择 HTML 元素。这在需要根据特定属性来应用样式时非常有用。

伪类	说明
[attribute]	选择具有特定属性的元素，无论该属性的值是什么。
[attribute="value"]	选择具有特定属性，且该属性的值精确等于指定值的元素。
[attribute~="value"]	选择具有特定属性，且该属性的值包含指定单词（以空格分隔）的元素。
[attribute^="value"]	选择具有特定属性，且该属性的值以指定字符串开头的元素。
[attribute$="value"]	选择具有特定属性，且该属性的值以指定字符串结尾的元素。
[attribute*="value"]	选择具有特定属性，且该属性的值包含指定字符串的元素。
[attribute|="value"]	选择具有特定属性，且该属性的值以指定字符串开头，后面可以跟连字符 - 再接其他字符的元素。常用于语言属性等场景。

示例3-2-20：属性选择器

3. 文字样式：3.1 文本样式

- color属性
  - color 属性主要用于设置元素内文本的颜色，其取值类型为：
    - 颜色关键字：CSS 预定义了一系列颜色关键字，如 red、blue、green、yellow、black、white 等，使用这些关键字能直接指定颜色。
    - 十六进制颜色值：十六进制颜色值是由 # 符号后跟 6 位十六进制数字组成，每两位分别表示红（R）、绿（G）、蓝（B）三种颜色的强度，取值范围是 00 到 FF。也可以使用 3 位十六进制缩写形式，如 #f00 等价于 #ff0000。
    - RGB 和 RGBA 颜色值：RGB：使用 rgb() 函数，语法为 rgb(red, green, blue)，其中 red、green、blue 的取值范围是 0 到 255。RGBA：rgba() 是在 rgb() 的基础上增加了一个透明度参数，语法为 rgba(red, green, blue, alpha)，alpha 的取值范围是 0.0（完全透明）到 1.0（完全不透明）。
    - HSL 和 HSLA 颜色值：HSL：使用 hsl() 函数，语法为 hsl(hue, saturation, lightness)。hue 表示色相，取值范围是 0 到 360 度；saturation 表示饱和度，以百分比形式表示，0% 为灰色，100% 为全彩；lightness 表示亮度，同样以百分比形式表示，0% 为黑色，100% 为白色。HSLA：hsla() 是在 hsl() 的基础上增加了透明度参数，语法为 hsla(hue, saturation, lightness, alpha)，alpha 的取值和意义与 rgba() 中的相同。

示例3-3-1：文本颜色

3. 文字样式：3.1 文本样式

- text-indent属性
  - text-indent属性用于设置元素内文本的首行缩进，也就是段落第一行相对于其他行的缩进量，其取值类型为：
    - 长度值：可以使用固定的长度单位，如 px（像素）、em（相对于元素的字体大小）、rem（相对于根元素的字体大小）等。
    - 百分比值：使用百分比值时，缩进量是相对于包含该元素的块级元素的宽度。
    - 负值：可以使用负值来实现文本的首行突出显示（即首行不缩进，反而突出）。

示例3-3-2：文本缩进

3. 文字样式：3.1 文本样式

- line-height属性
  - text-height属性主要用于设置元素内每行文本的间距，也就是行与行之间的垂直距离，其取值类型为：
    - 无单位数字：这是最常用的取值方式，无单位数字会被视为当前元素字体大小的倍数。
    - 长度值：可以使用固定的长度单位，如 px（像素）、em（相对于元素的字体大小）、rem（相对于根元素的字体大小）等。
    - 百分比值：相对于当前元素的字体大小来计算的。
    - normal： line-height 的默认值，它会根据浏览器和字体的不同而有不同的表现，通常在 1.2 到 1.3 之间。
    - 继承性：line-height 属性具有继承性，子元素会继承父元素的 line-height 值。如果使用无单位数字，子元素会根据自身的字体大小重新计算行高；如果使用长度值或百分比值，子元素会直接继承该固定值。
    - 对单行文本垂直居中的影响：当元素只有一行文本时，可以将 line-height 设置为与元素的高度相同，实现文本的垂直居中。

示例3-3-3：行高

3. 文字样式：3.1 文本样式

- letter-spacing属性
  - letter-spacing 属性用于设置元素内文本中字符之间的间距，也就是调整字符与字符之间的水平距离，其取值类型为：
    - 长度值：可以使用固定的长度单位，如 px（像素）、em（相对于元素的字体大小）、rem（相对于根元素的字体大小）等。
      - 正值：增加字符之间的间距。
      - 负值：减小字符之间的间距，使字符更加紧凑，但过度使用可能会导致字符重叠，影响可读性。
    - normal：letter-spacing 的默认值，表示使用浏览器的默认字符间距，通常字符之间的间距比较紧凑。
    - 继承性：letter-spacing 属性具有继承性，子元素会继承父元素的 letter-spacing 值。如果不想让子元素继承该属性，可以为子元素显式设置 letter-spacing: normal;。
    - 对文本布局的影响：调整 letter-spacing 可能会影响文本的整体布局和长度。增加字符间距会使文本在水平方向上占据更多空间，减小字符间距则相反。在设计响应式布局时，需要考虑这种影响，确保文本在不同屏幕尺寸下都能正常显示。

示例3-3-4：字母间隔

3. 文字样式：3.1 文本样式

- text-align属性
  - text-align属性用于设置元素内文本的水平对齐方式，其取值类型为：
    - left：将文本左对齐，这是大多数浏览器中块级元素的默认对齐方式。
    - right：将文本右对齐。
    - center：将文本居中对齐。
    - justify：将文本两端对齐，即调整单词之间的间距，使每行文本的左右两端都与元素的边界对齐，最后一行文本通常左对齐。
    - start：文本对齐到容器的起始边缘。在从左到右的语言环境中，它等同于 left；在从右到左的语言环境中，它等同于 right。
    - end：文本对齐到容器的结束边缘。在从左到右的语言环境中，它等同于 right；在从右到左的语言环境中，它等同于 left。

    - 仅作用于块级元素和具有块级行为的元素：text-align 属性主要对块级元素（如 <p>、 、<h1> - <h6> 等）或具有 display 属性值为 block、inline-block、table-cell 等类似块级行为的元素有效。对于行内元素（如 、<a> 等），默认情况下该属性不起作用，除非将其 display 属性设置为块级相关的值。
    - 不同语言环境的影响：start 和 end 值会根据文档的语言方向（通过 direction 属性设置）而改变对齐方式。在使用时要考虑文档的语言环境，确保文本的对齐方式符合预期。

示例3-3-5：水平对齐

3. 文字样式：3.1 文本样式

- text-decoration属性
  - text-decoration 是一个用于为文本添加修饰效果的 CSS 属性，它可以控制文本的下划线、上划线、删除线等样式，还能设置修饰线的颜色、样式和厚度，其取值类型与子属性为：
    - text-decoration-line：用于指定文本修饰线的位置和类型。
      - none：不添加任何修饰线，这是默认值。
      - underline：在文本下方添加下划线。
      - overline：在文本上方添加上划线。
      - line-through：在文本中间添加删除线。
      - 可以同时使用多个值，用空格分隔，例如同时添加下划线和上划线。
    - text-decoration-color：用于设置修饰线的颜色，可以使用颜色关键字（如 red、blue）、十六进制颜色值（如 #ff0000）、RGB 或 RGBA 值等。
    - text-decoration-style：用于设置修饰线的样式。
      - solid：实线，这是默认样式。
      - dotted：点状线。
      - wavy：波浪线。
    - text-decoration-thickness：用于设置修饰线的厚度，可以使用长度值（如 px、em）或关键字 auto（浏览器自动选择合适的厚度）。
    - 继承性：text-decoration 属性具有部分继承性。如果父元素设置了 text-decoration，子元素通常会继承该样式，但 <a> 元素比较特殊，它有自己默认的下划线样式，不会继承父元素的 text-decoration 样式，除非显式指定。
    - 性能影响：频繁使用复杂的 text-decoration 样式（如波浪线）可能会对页面性能产生一定影响，尤其是在处理大量文本时。因此，在实际应用中应根据需要合理使用。

示例3-3-6：文本装饰

3. 文字样式：3.1 文本样式

- text-transform属性
  - text-transform 属性用于控制元素内文本的大小写转换，通过该属性可以轻松地将文本转换为大写、小写或每个单词的首字母大写等形式，其基本语法如下。
    - none：不进行任何大小写转换，文本保持原始输入的大小写形式，这是 text-transform 属性的默认值。
    - uppercase：将文本中的所有字母转换为大写形式。
    - lowercase：将文本中的所有字母转换为小写形式。
    - capitalize：将文本中每个单词的首字母转换为大写形式，其余字母保持不变。
    - full-width：将所有字符转换为全角形式，主要用于处理拉丁字母、数字和标点符号等，使其在显示时占据与亚洲文字相同的宽度。该取值通常在处理多语言文本排版时使用。

    - 仅改变显示效果：text-transform 属性只是改变文本的显示方式，并不会改变 HTML 元素内实际的文本内容。例如，使用 uppercase 将文本显示为大写，但在 JavaScript 获取元素的文本内容时，得到的仍然是原始输入的文本。
    - 对非字母字符的影响：text-transform 属性主要影响字母字符，对于数字、标点符号等非字母字符，除了 full-width 取值外，其他取值通常不会对它们产生影响。
    - 语言特定规则：不同语言可能有不同的大小写转换规则，浏览器会根据文档的语言设置（通过 lang 属性）来应用相应的规则。例如，某些语言中的字母大小写转换可能有特殊情况，使用时需要注意。

示例3-3-7：字符转换

3. 文字样式：3.1 文本样式

- white-space 属性
  - white-space 属性用于控制元素内空白字符（如空格、制表符、换行符等）的处理方式，以及文本是否换行，其取值类型为：
    - normal：这是 white-space 属性的默认值。它会将多个连续的空白字符合并为一个空格，并且在必要时自动换行以适应元素的宽度。
    - nowrap：该值会将多个连续的空白字符合并为一个空格，但禁止文本自动换行，文本会在一行内显示，直到遇到
 标签才会换行。
    - pre：保留文本中的所有空白字符，包括连续的空格、制表符和换行符，并且不会自动换行，文本会按照原始的格式显示，就像使用了 HTML 中的 <pre> 标签一样。
    - pre-wrap：保留文本中的所有空白字符，但会在必要时自动换行以适应元素的宽度。
    - pre-line：将多个连续的空白字符合并为一个空格，但会保留文本中的换行符，并且在必要时自动换行以适应元素的宽度。
    - break-spaces：与 pre-wrap 类似，保留所有空白字符，在必要时自动换行。不同之处在于，break-spaces 会将连续的空格视为换行点，即使空格在一行的开头或结尾，也会占用空间并可能导致换行。

    - 对布局的影响：不同的 white-space 值会影响文本的布局和显示效果。例如，nowrap 可能会导致文本溢出元素边界，需要结合 overflow 属性来处理溢出情况。
    - 与其他属性的交互：white-space 属性会与 text-align、word-wrap（或 overflow-wrap）等属性相互影响。例如，text-align 属性用于控制文本的水平对齐方式，而 white-space 决定了文本的换行和空白处理，它们共同作用于文本的显示效果。
    - 兼容性：大多数现代浏览器都支持 white-space 属性的所有取值，但在一些旧版本的浏览器中可能存在兼容性问题。在使用时，建议进行充分的测试。

示例3-3-8：元素空白

3. 文字样式：3.1 文本样式

- word-spacing 属性
  - word-spacing 属性用于设置元素内单词之间的间距，也就是调整单词与单词之间的水平距离，其取值类型为：
    - 长度值：可以使用固定的长度单位，如 px（像素）、em（相对于元素的字体大小）、rem（相对于根元素的字体大小）等。。
      - 正值：增加单词之间的间距。
      - 负值：减小单词之间的间距，使单词更加紧凑，但过度使用可能会导致单词重叠，影响可读性。
    - normal： word-spacing 的默认值，表示使用浏览器的默认单词间距。通常情况下，浏览器会根据字体和排版规则自动设置合适的单词间距。
    - 继承性：word-spacing 属性具有继承性，子元素会继承父元素的 word-spacing 值。如果不想让子元素继承该属性，可以为子元素显式设置 word-spacing: normal;。
    - 对文本布局的影响：调整 word-spacing 可能会影响文本的整体布局和长度。增加单词间距会使文本在水平方向上占据更多空间，减小单词间距则相反。在设计响应式布局时，需要考虑这种影响，确保文本在不同屏幕尺寸下都能正常显示。
    - 可读性：虽然可以通过设置负值来减小单词间距，但要注意不要过度压缩，以免影响文本的可读性。在实际应用中，应根据具体的设计需求和字体特点合理调整单词间距。

示例3-3-9：文字间隔

3. 文字样式：3.2 字体样式

- font-family 属性
  - font-family属性用于指定元素文本所使用字体的属性。通过该属性，可以为网页上的文字设置不同的字体风格，以满足设计需求，其取值类型为：
    - 具体字体名称：可以直接指定具体的字体名称，如 Arial、Times New Roman、SimSun（宋体）等。
    - 通用字体族：通用字体族是一组具有相似外观特征的字体集合，当具体字体无法使用时，可以作为后备选项。常见的通用字体族有：
      - serif：有衬线字体，字体的笔画末端有额外的装饰线条，如 Times New Roman。
      - sans-serif：无衬线字体，字体的笔画末端没有额外的装饰线条，如 Arial。
      - monospace：等宽字体，每个字符的宽度相同，常用于代码显示，如 Courier New。
      - cursive：手写风格字体，模仿手写的样式。
      - fantasy：奇幻字体，通常具有独特、夸张的外观。
    - 为了确保在不同的操作系统和设备上都能有合适的字体显示，通常会使用字体栈，即提供多个字体选项。

    - 字体可用性：不同的操作系统和设备预装的字体不同，某些字体可能只在特定的系统中存在。因此，使用字体栈可以提高字体显示的兼容性。
    - 引号的使用：如果字体名称包含空格、特殊字符或非 ASCII 字符，需要用引号括起来，如 "Microsoft YaHei"、"Times New Roman"。单引号或双引号均可，但要保持一致。
    - 自定义字体：如果需要使用系统中没有预装的字体，可以通过 @font-face 规则引入自定义字体，或者使用网络字体服务（如 Google Fonts）。

示例3-3-10：字体系列

3. 文字样式：3.2 字体样式

- font-size 属性
  - font-size 属性用于设置元素中文本的字体大小，它在网页设计中是一个非常基础且关键的属性，能直接影响页面的可读性和整体视觉效果，其取值类型为：
    - 绝对大小
      - 关键字：CSS 提供了一些关键字来设置字体大小，如 xx-small、x-small、small、medium、large、x-large、xx-large。这些关键字是相对于浏览器的默认字体大小而言的，medium 通常对应浏览器的默认字体大小，一般为 16px。
      - 固定长度单位：可以使用固定的长度单位，如 px（像素）。px 是最常用的单位，它能精确控制字体的大小，不受浏览器或操作系统的影响。
    - 相对大小
      - 百分比：以父元素的字体大小为基准，按照百分比来设置字体大小。
      - em：同样以父元素的字体大小为基准，1em 等于父元素的字体大小。
      - rem：以根元素（<html>）的字体大小为基准，不受父元素字体大小的影响。
      - vw 和 vh：vw 表示视口宽度的百分比，vh 表示视口高度的百分比。例如，1vw 等于视口宽度的 1%。
    - 继承性：font-size 属性具有继承性，子元素会继承父元素的字体大小。如果需要修改子元素的字体大小，可以通过相对单位或直接指定新的字体大小来覆盖继承的值。
    - 响应式设计：在进行响应式网页设计时，建议使用相对单位（如 em、rem、vw、vh 等），这样可以使字体大小根据页面布局或视口大小自动调整，提高页面在不同设备上的显示效果。
    - 浏览器兼容性：大多数现代浏览器都支持 font-size 属性的各种取值方式，但在一些旧版本的浏览器中可能存在兼容性问题。在使用新的单位或特性时，建议进行充分的测试。

示例3-3-11：字体大小

3. 文字样式：3.2 字体样式

- font-weight 属性
  - font-weight 属性用于设置元素中文本的字体粗细程度，它可以让文本在页面上呈现出不同的视觉强调效果，其取值类型为：
    - 关键字
      - normal：这是默认值，表示正常的字体粗细，等同于数值 400。
      - bold：表示粗体字体，等同于数值 700。
      - lighter：表示比父元素更细的字体。同样，具体粗细程度取决于浏览器和字体。
    - 数值
      - 可以使用 100 到 900 之间以 100 为间隔的数值来精确指定字体的粗细，数值越大字体越粗。
        - 100：最细的字体。
        - 200：比 100 稍粗。
        - 300：细体。
        - 400：等同于 normal。
        - 500：中等粗细。
        - 600：比 500 稍粗。
        - 700：等同于 bold。
        - 800：粗体程度比 700 更高。
        - 900：最粗的字体。
    - 字体支持：并非所有字体都支持 font-weight 属性的所有取值。有些字体可能只提供了正常和粗体两种粗细样式，对于其他取值，浏览器可能会使用最接近的可用样式来显示。
    - 继承性：font-weight 属性具有继承性，子元素会继承父元素的字体粗细。如果需要修改子元素的字体粗细，可以通过重新设置 font-weight 属性来覆盖继承的值。
    - 视觉效果：不同的字体对于相同的 font-weight 取值可能会有不同的视觉表现。在设计网页时，需要根据具体的字体来选择合适的字体粗细，以达到理想的视觉效果。

示例3-3-12：字体加粗

3. 文字样式：3.2 字体样式

- font-style 属性
  - font-style 属性用于设置元素内文本的字体样式，主要用于控制文本是正常显示、倾斜显示还是以斜体显示，其取值类型为：
    - normal：这是 font-style 属性的默认值，表示文本以正常的字体样式显示，没有倾斜或斜体效果。
    - italic：将文本设置为斜体样式。斜体通常是字体专门设计的一种风格，具有独特的倾斜形态，与正常字体相比，它的笔画可能会有一定的倾斜角度和形状变化。
    - oblique：同样是将文本设置为倾斜样式，但与 italic 不同的是，oblique 是通过将正常字体倾斜一定角度来实现倾斜效果，而不是使用字体专门设计的斜体样式。不过，在实际应用中，很多浏览器会将 oblique 处理成与 italic 相同的效果。
    - 字体支持：并非所有字体都有专门设计的斜体样式。当使用 italic 时，如果字体本身没有斜体版本，浏览器可能会尝试用 oblique 的方式来模拟斜体效果。
    - 继承性：font-style 属性具有继承性，子元素会继承父元素的字体样式。如果需要修改子元素的字体样式，可以通过重新设置 font-style 属性来覆盖继承的值。
    - 语义和样式：在 HTML 中， 和 <i> 元素通常用于表示斜体文本，但它们也有一定的语义含义。 表示强调，而 <i> 通常用于表示一些不同的语气或文本的不同风格。在使用 font-style 属性时，要结合 HTML 元素的语义和实际的设计需求来选择合适的方式。

示例3-3-13：字体风格

3. 文字样式：3.3 文本效果

- text-overflow 属性
  - text-overflow 属性用于规定当文本溢出包含它的元素时，应该如何显示溢出的部分。它通常与 white-space 和 overflow 属性一起使用，以实现特定的文本溢出效果，其取值类型为：
    - clip：这是 text-overflow 属性的默认值。当文本溢出元素的边界时，直接将溢出的部分裁剪掉，不会显示任何额外的提示，文本会被截断在元素的边界处。
    - ellipsis：当文本溢出元素的边界时，在溢出的位置显示省略号（...），以此来提示用户这里有被隐藏的文本内容。
    - 自定义字符串（实验性）：在某些浏览器中，还支持使用自定义字符串来替代省略号。不过这是一个实验性的特性，兼容性可能不太好。
    - 与其他属性配合使用：text-overflow 属性本身不会产生溢出效果，它需要与 white-space: nowrap 和 overflow: hidden 一起使用。white-space: nowrap 用于防止文本换行，overflow: hidden 用于隐藏溢出的内容，这样 text-overflow 才能正常发挥作用。
    - 兼容性问题：自定义字符串作为 text-overflow 的取值是一个实验性特性，不同浏览器对其支持程度不同。在实际开发中，建议优先使用 clip 和 ellipsis 这两个较为稳定的取值。
    - 多行文本溢出：text-overflow 通常只对单行文本溢出有效。对于多行文本溢出，实现起来相对复杂，可能需要借助 JavaScript 或者一些特定的 CSS 技巧（如 -webkit-line-clamp，但它是 WebKit 内核浏览器的私有属性）。

示例3-3-13：字体风格

3. 文字样式： 3.3 文本效果

- text-shadow属性
  - text-shadow 属性用于为元素内的文本添加阴影效果，能增强文本的立体感和视觉吸引力，让页面更加生动美观。

```
selector {
    text-shadow: h-shadow v-shadow blur-radius color;
}
```

  - 其中，selector 是要应用文本阴影样式的元素选择器，后面的参数含义如下：
    - h-shadow：必需，指定水平阴影的偏移量。正值表示阴影在文本的右侧，负值表示在左侧。
    - v-shadow：必需，指定垂直阴影的偏移量。正值表示阴影在文本的下方，负值表示在上方。
    - blur-radius：可选，指定阴影的模糊半径。值越大，阴影越模糊；默认值为 0，表示没有模糊效果。
    - color：可选，指定阴影的颜色。可以使用颜色关键字、十六进制颜色值、RGB 或 RGBA 值等；默认值由浏览器决定，通常为黑色。
  - 性能影响：过多或复杂的文本阴影效果可能会对页面性能产生一定影响，尤其是在处理大量文本或在性能较低的设备上。因此，在实际应用中应根据需要合理使用，避免过度使用复杂的阴影效果。
  - 兼容性：大多数现代浏览器都支持 text-shadow 属性，但在一些旧版本的浏览器中可能存在兼容性问题。在使用时，建议进行充分的测试。
  - 颜色透明度：如果需要设置阴影的透明度，可以使用 RGBA 颜色值。

示例3-3-13：字体风格

3. 文字样式： 3.3 文本效果

- overflow-warp属性
  - overflow-wrap 是一个 CSS 属性，用于指定当文本溢出其容器时，是否允许在单词内部进行换行，以避免文本溢出容器。
    - normal：这是 overflow-wrap 属性的默认值。当设置为 normal 时，文本只会在正常的单词边界处进行换行。也就是说，如果一个单词太长，超出了容器的宽度，它将不会被拆分，而是会溢出容器。
    - break-word：当取值为 break-word 时，允许浏览器在必要时将一个不能在单词边界处换行的长单词进行拆分，并在单词内部换行，以防止文本溢出容器。
  - 与 word-wrap 的关：overflow-wrap 是 word-wrap 的标准化名称，word-wrap 是早期的非标准属性，现在已经被弃用。虽然大多数浏览器仍然支持 word-wrap，但为了代码的兼容性和遵循标准，建议使用 overflow-wrap。
  - 兼容性：overflow-wrap 在大多数现代浏览器中都有很好的支持，但在一些旧版本的浏览器中可能需要添加浏览器前缀来确保兼容性。例如，在 WebKit 内核的浏览器中，可能需要使用 -webkit-overflow-wrap。
  - 与其他属性的配合：overflow-wrap 通常与 width、max-width 等属性一起使用，以定义容器的宽度，从而更好地控制文本的换行效果。同时，它也可能会受到 white-space 属性的影响，white-space 属性用于控制空白字符的处理方式，可能会影响文本是否换行以及如何换行。例如，当 white-space 设置为 nowrap 时，文本不会换行，即使 overflow-wrap 设置为 break-word 也可能无效。

示例3-3-13：字体风格

4. 背景与边框：4.1 背景属性

- background 属性
  - background 是一个 CSS 简写属性，用于一次性设置元素的背景相关样式，包括背景颜色、背景图像、背景重复方式、背景位置、背景大小、背景附着方式等，其取值类型及子属性说明如下。
    - background-color：用于设置元素的背景颜色，可以使用颜色关键字（如 red、blue）、十六进制颜色值（如 #ff0000）、RGB 或 RGBA 值等。
    - background-image：用于设置元素的背景图像，可以使用 url() 函数指定图像的路径，也可以使用渐变来创建背景效果。
    - background-repeat：用于指定背景图像的重复方式，常见取值有：
      - repeat：默认值，背景图像在水平和垂直方向上都重复。
      - repeat-x：背景图像只在水平方向上重复。
      - repeat-y：背景图像只在垂直方向上重复。
      - no-repeat：背景图像不重复。
    - background-attachment：用于设置背景图像相对于元素的滚动方式，常见取值有：
      - scroll：默认值，背景图像会随着元素内容的滚动而滚动。
      - fixed：背景图像固定在浏览器窗口的某个位置，不会随着元素内容的滚动而滚动。
      - local：背景图像会随着元素的内容一起滚动。
    - background-position：用于指定背景图像的起始位置，可以使用关键字（如 top、bottom、left、right、center）或长度值（如 px、em）、百分比值来表示。

4. 背景与边框：4.1 背景属性

- background 属性
  - background 是一个 CSS 简写属性，用于一次性设置元素的背景相关样式，包括背景颜色、背景图像、背景重复方式、背景位置、背景大小、背景附着方式等，其取值类型及子属性说明如下。
    - background-size：用于设置背景图像的大小，可以使用长度值、百分比值，也可以使用关键字 cover 或 contain。
      - cover：缩放背景图像以完全覆盖元素，可能会导致部分图像超出元素范围。
      - contain：缩放背景图像以使其完全包含在元素内，可能会在元素边缘留下空白。
    - background-origin：用于指定背景图像的定位原点，常见取值有：
      - padding-box：默认值，背景图像相对于内边距框定位。
      - border-box：背景图像相对于边框框定位。
      - content-box：背景图像相对于内容框定位
    - background-clip：用于指定背景的绘制区域，常见取值有：
      - border-box：默认值，背景延伸到边框外边缘。
      - padding-box：背景延伸到内边距外边缘，不覆盖边框。
      - content-box：背景只绘制在内容区域。
    - 兼容性：大多数现代浏览器都支持 background 属性及其子属性，但在一些旧版本的浏览器中可能存在兼容性问题。特别是一些较新的特性，如 background-size 的 cover 和 contain 值，在旧浏览器中可能需要添加前缀（如 -webkit-、-moz- 等）来保证兼容性。
    - 性能影响：使用大尺寸的背景图像或复杂的渐变效果可能会对页面性能产生影响，尤其是在移动设备或性能较低的设备上。因此，在使用时应尽量优化图像大小，并避免过度使用复杂的背景效果。
    - 简写属性的覆盖：当使用 background 简写属性时，会覆盖之前单独设置的 background 相关子属性。如果需要单独调整某个子属性的值，建议直接使用该子属性进行设置。

案例演示

- 示例3-4-1：background-clip属性
- 示例3-4-2：background-origin属性
- 示例3-4-3：background-size属性

4. 背景与边框：4.1 背景属性

- linear-gradient() 属性
  - background: linear-gradient()是一个用于创建线性渐变背景的属性值。它可以让你在元素的背景中实现从一种颜色到另一种颜色或多种颜色的平滑过渡

```
background: linear-gradient([<angle> | to <side-or-corner>], <color-stop-list>);
```

  - <angle>：指定渐变的方向角度，以度数为单位，0deg 表示从上到下的渐变，90deg 表示从左到右的渐变，以此类推。
  - to <side-or-corner>：也用于指定渐变方向，例如to right表示从左到右的渐变，to bottom right表示从左上角到右下角的渐变。
    - <angle>：指定渐变的方向角度，以度数为单位，0deg 表示从上到下的渐变，90deg 表示从左到右的渐变，以此类推。
    - to <side-or-corner>：也用于指定渐变方向，例如to right表示从左到右的渐变，to bottom right表示从左上角到右下角的渐变。
    - <color-stop-list>：是一系列颜色值和它们的位置，用于定义渐变过程中不同位置的颜色。颜色值可以使用预定义颜色名称、十六进制值、RGB 值、RGBA 值等，位置可以用百分比、长度值或关键字start、center、end来表示。
  - <color-stop-list>：是一系列颜色值和它们的位置，用于定义渐变过程中不同位置的颜色。颜色值可以使用预定义颜色名称、十六进制值、RGB 值、RGBA 值等，位置可以用百分比、长度值或关键字start、center、end来表示。

4. 背景与边框：4.1 背景属性

- linear-gradient() 属性
  - 浏览器兼容性：虽然大多数现代浏览器都支持 linear-gradient() 函数，但在一些旧版本的浏览器中可能需要添加浏览器前缀来确保兼容性。

示例3-4-4：线性渐变属性

4. 背景与边框：4.1 背景属性

- radial-gradient() 属性
  - radial-gradient() 是 CSS 中用于创建径向渐变效果的函数。径向渐变是从一个中心点开始，颜色向四周辐射状扩散的渐变效果。

```
radial-gradient([<shape> || <size>]? [at <position>]?, <color-stop-list>)
```

  - 形状（<shape>，可选）：指定渐变的形状，有两个取值：
    - circle：表示渐变形状为圆形。
    - ellipse：表示渐变形状为椭圆形。如果不指定，默认根据元素的尺寸和 size 参数来确定是圆形还是椭圆形。
  - 大小（<size>，可选）：定义渐变的大小，有以下几种取值：
    - closest-side：渐变的边缘刚好接触到离中心点最近的元素边缘。
    - farthest-side：渐变的边缘刚好接触到离中心点最远的元素边缘。
    - closest-corner：渐变的边缘刚好接触到离中心点最近的元素角。
    - farthest-corner：渐变的边缘刚好接触到离中心点最远的元素角（默认值）。
    - 长度值或百分比：可以使用具体的长度单位（如 px、em）或百分比来指定渐变的大小。

4. 背景与边框：4.1 背景属性

- radial-gradient() 属性
  - radial-gradient() 是 CSS 中用于创建径向渐变效果的函数。径向渐变是从一个中心点开始，颜色向四周辐射状扩散的渐变效果。

```
radial-gradient([<shape> || <size>]? [at <position>]?, <color-stop-list>)
```

  - 位置（at <position>，可选）：指定渐变的中心点位置，可以使用关键字（如 top、bottom、left、right、center）或长度值、百分比来表示。例如：
    - at center：中心点位于元素中心（默认值）。
    - at top left：中心点位于元素的左上角。
    - at 20% 30%：中心点位于元素宽度的 20% 和高度的 30% 处。
  - 颜色停止点列表（<color-stop-list>）：这是一个或多个颜色停止点的组合，每个颜色停止点由颜色值和可选的位置组成。颜色值可以是任何合法的 CSS 颜色表示方式，位置可以用百分比或长度值来指定，表示该颜色在渐变中开始的位置。

示例3-4-5：放射渐变属性

4. 背景与边框：4.2 边框属性

- border属性
  - border 是一个 CSS 简写属性，用于一次性设置元素的边框样式、宽度和颜色。它可以为页面元素增添视觉上的分隔和层次感，使页面布局更加清晰，其取值类型及子属性说明如下。
    - border-width：用于设置边框的宽度，可以使用长度值（如 px、em、rem 等）或关键字（如 thin、medium、thick）
    - border-style：用于指定边框的样式，常见取值有：
      - none：默认值，不显示边框。
      - dashed：虚线边框。
      - ridge：具有 3D 凸起效果的边框。
      - solid：实线边框。
      - double：双实线边框。
      - inset：具有 3D 内陷效果的边框。
      - dotted：点状边框。
      - groove：具有 3D 凹槽效果的边框。
      - outset：具有 3D 外凸效果的边框。
    - border-color：用于设置边框的颜色，可以使用颜色关键字（如 red、blue）、十六进制颜色值（如 #ff0000）、RGB 或 RGBA 值等。
    - 单独设置各边的边框：除了使用 border 简写属性，还可以分别设置元素四条边的边框样式，可以使用 border-top、border-right、border-bottom 和 border-left 分别设置顶部、右侧、底部和左侧的边框。
    - 兼容性：大多数现代浏览器都支持 border 属性及其相关子属性，但在一些旧版本的浏览器中可能存在兼容性问题。特别是一些较新的边框样式（如 groove、ridge 等 3D 效果的边框），在不同浏览器中的显示效果可能会有细微差异。
    - 盒模型影响：边框的宽度会影响元素的实际尺寸，因为边框是盒模型的一部分。在进行页面布局时，需要考虑边框宽度对元素大小和位置的影响。例如，设置了较宽边框的元素可能会比预期占用更多的空间。
    - 样式优先级：如果同时使用了 border 简写属性和单独的边框子属性，单独的子属性会覆盖简写属性中相应的值

示例3-4-6：边框属性

4. 背景与边框：4.2 边框属性

- border-radius属性
  - border-radius 是一个 CSS 属性，用于为元素的边框添加圆角效果，它能让元素的外观更加柔和、美观，在现代网页设计中应用广泛。
  - border-radius 既可以作为简写属性一次性设置元素四个角的圆角，也可以分别设置每个角的圆角。
  - 长度值：可以使用固定的长度单位，如 px（像素）、em（相对于元素的字体大小）、rem（相对于根元素的字体大小）等。
  - 百分比值：使用百分比值时，圆角的半径是相对于元素的宽度和高度来计算的。
  - 不同数量值的含义
    - 一个值：四个角都使用相同的圆角半径。
    - 两个值：第一个值用于左上角和右下角，第二个值用于右上角和左下角。
    - 三个值：第一个值用于左上角，第二个值用于右上角和左下角，第三个值用于右下角。
    - 四个值：依次用于左上角、右上角、右下角和左下角。
  - 椭圆圆角：可以使用 / 分隔水平和垂直方向的半径，创建椭圆形状的圆角。

4. 背景与边框：4.2 边框属性

- border-radius属性
  - 兼容性：大多数现代浏览器都支持 border-radius 属性，但在一些旧版本的浏览器中可能需要添加浏览器前缀（如 -webkit-、-moz- 等）来保证兼容性。
  - 对背景和内容的影响：border-radius 不仅会影响边框的形状，还会裁剪元素的背景和内容，使其适应圆角的形状。但在某些情况下，背景图像或渐变可能需要额外的处理才能正确显示圆角效果。
  - 与其他属性的交互：border-radius 与 overflow 属性密切相关。当 overflow 属性值为 hidden 时，元素内容会被裁剪以适应圆角形状；当 overflow 属性值为 visible 时，内容可能会溢出圆角边界。

示例3-4-7：圆角边框属性

4. 背景与边框：4.2 边框属性

- border-image属性
  - border-image 是一个 CSS 简写属性，用于使用图像来替代元素的边框样式。它可以让你为元素创建独特、个性化的边框效果，使页面更加美观和吸引人。

```
border-image: source slice width outset repeat|initial|inherit;
```

  - border-image 属性可以一次性设置多个与边框图像相关的子属性，这些子属性也可以单独设置。
  - border-image-source：用于指定边框图像的来源，可以使用 url() 函数指定图像的路径，也可以取值为 none 表示不使用边框图像。
  - border-image-slice：该属性用于将边框图像分割成 9 个区域：4 个角、4 条边和 1 个中间区域。可以使用 1 - 4 个值来指定分割线的位置，取值可以是数字（表示像素或矢量坐标）或百分比。
    - 一个值：四个边的分割线位置相同。
    - 两个值：第一个值表示上下边的分割线位置，第二个值表示左右边的分割线位置。
    - 三个值：第一个值表示上边的分割线位置，第二个值表示左右边的分割线位置，第三个值表示下边的分割线位置。
    - 四个值：依次表示上、右、下、左边的分割线位置。

4. 背景与边框：4.2 边框属性

- border-image属性
  - border-image 是一个 CSS 简写属性，用于使用图像来替代元素的边框样式。它可以让你为元素创建独特、个性化的边框效果，使页面更加美观和吸引人。

```
border-image: source slice width outset repeat|initial|inherit;
```

  - border-image-width：用于指定边框图像的宽度，可以使用长度值（如 px、em 等）、百分比或数字（表示相对于 border-width 的倍数）。同样可以使用 1 - 4 个值来分别指定四个边的宽度。
  - border-image-outset：指定边框图像相对于元素边框盒的偏移量，可以使用长度值或数字。也可以使用 1 - 4 个值分别指定四个边的偏移量。
  - border-image-repeat：用于指定边框图像的四条边如何重复填充，可以取值为：
    - stretch：默认值，拉伸边框图像以填充边框区域。
    - repeat：重复边框图像以填充边框区域，可能会导致图像失真。
    - round：重复边框图像以填充边框区域，会自动调整图像的大小以避免出现裁剪。
    - space：重复边框图像以填充边框区域，会在图像之间添加间距以避免裁剪。

4. 背景与边框：4.2 边框属性

- border-image属性
  - border-image 是一个 CSS 简写属性，用于使用图像来替代元素的边框样式。它可以让你为元素创建独特、个性化的边框效果，使页面更加美观和吸引人。

```
border-image: source slice width outset repeat|initial|inherit;
```

  - 兼容性：大多数现代浏览器都支持 border-image 属性，但在一些旧版本的浏览器中可能存在兼容性问题。特别是在不同浏览器中，对于一些特殊的取值（如 round、space）可能会有不同的显示效果。在使用时，建议进行充分的测试。
  - 图像尺寸和分割：要确保边框图像的尺寸和分割方式能够满足设计需求。如果分割不合理，可能会导致边框显示异常。例如，如果分割线位置设置不当，可能会使角部图像显示不完整或重复部分出现拼接问题。
  - 与 border 属性的关系：在使用 border-image 时，需要先设置 border 属性，因为 border-image 是基于 border 的宽度和样式来应用的。如果 border-width 为 0，那么边框图像将不会显示。

示例3-4-8：图片边框

4. 背景与边框：4.2 边框属性

- linear-gradient
  - 可使用 linear-gradient() 函数结合 border-image 属性为元素创建渐变边框。linear-gradient() 能够生成线性渐变图像，将其作为边框图像应用到元素上，从而实现独特的渐变边框效果。

示例3-4-9：渐变边框

4. 背景与边框：4.2 边框属性

- box-shadow 属性
  - box-shadow 属性用于为元素添加阴影效果，使元素在页面中具有立体感和层次感，增强视觉效果。

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

  - h-shadow：必需，指定水平阴影的偏移量。正值表示阴影在元素的右侧，负值表示在左侧。
  - v-shadow：必需，指定垂直阴影的偏移量。正值表示阴影在元素的下方，负值表示在上方。
  - blur：可选，指定阴影的模糊半径。值越大，阴影越模糊；默认值为 0，表示没有模糊效果。
  - spread：可选，指定阴影的扩展半径。正值会使阴影扩大，负值会使阴影缩小；默认值为 0。
  - color：可选，指定阴影的颜色。可以使用颜色关键字、十六进制颜色值、RGB 或 RGBA 值等；默认值由浏览器决定，通常为黑色。
  - inset：可选，关键字，若指定该关键字，则阴影为内阴影；若省略，则为外阴影。

4. 背景与边框：4.2 边框属性

- box-shadow 属性
  - box-shadow 属性用于为元素添加阴影效果，使元素在页面中具有立体感和层次感，增强视觉效果。

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

  - 性能影响：过多或复杂的阴影效果可能会对页面性能产生一定影响，尤其是在处理大量元素或在性能较低的设备上。因此，在实际应用中应根据需要合理使用，避免过度使用复杂的阴影效果。
  - 兼容性：大多数现代浏览器都支持 box-shadow 属性，但在一些旧版本的浏览器中可能需要添加浏览器前缀来确保兼容性。
  - 阴影顺序：当使用多个阴影时，前面的阴影会覆盖后面的阴影。例如，在 box-shadow: 2px 2px 5px red, -2px -2px 5px blue; 中，红色阴影会显示在蓝色阴影之上。

示例3-4-10：阴影效果

5. 动画：5.1 Web动画

- 动画是由一系列静态的图像（帧）以一定的速度（如每秒24帧或每秒30帧）连续播放而形成的。当这些帧快速连续地切换时，会因为视觉残像而产生错觉，将这一系列的静态图像看作是连续运动的画面。

- 几个Web动画示例
  - https://loading.io/
  - http://www.species-in-pieces.com/#
  - https://codepen.io/marianab/full/XPOQaR/
  - https://codepen.io/matchboxhero/pen/JrLJeb?editors=1100
  - https://tympanus.net/Tutorials/OriginalHoverEffects/index.html

5. 动画：5.1 Web动画

- GIF动画
  - GIF（Graphics Interchange Format，图形互换格式）是为跨平台而开发的动画格式，Web前端开发者可以通过GIF实现简单的网页动画。GIF89a格式是将多幅GIF组合在一起，并按照一定的时间间隔顺序显示出来，从而实现动态效果。GIF支持透明背景图像，适合在不同背景的网页上展示。

示例3-5-1：在页面中使用GIF动画

5. 动画：5.1 Web动画

- JS动画
  - JavaScript动画是Web前端中另外一种动画形式，其实现主要基于JavaScript对HTML的操作，通过JavaScript与CSS样式的结合实现丰富多彩的动画效果。Web前端开发者可使用JavaScript实现各种各样的动画效果，如图片轮转、新闻头条播报、用户交互操作等。JavaScript动画的制作偏向编程，开发使用难度较高，需要开发者具有丰富的Web前端开发经验和编程经验。

示例3-5-2：Javascript动画

5. 动画：5.1 Web动画

- CSS动画
  - CSS动画就是使元素逐渐从一种样式变为另一种样式。可以随意更改任意数量的 CSS 属性以平滑、流畅的方式移动、旋转、缩放、改变透明度等。想要使用 CSS 动画，必须先为动画指定关键帧。关键帧包含元素在特定时间所拥有的样式。相比传统的脚本实现动画技术，使用 CSS 动画有三个主要优点：
    - 能够非常容易地创建简单动画，甚至不需要了解 JavaScript 就能创建动画。
    - 动画在低性能的系统上也能运行流畅。渲染引擎会使用跳帧或者其他技术以保证动画表现尽可能的流畅。而使用 JavaScript 实现的动画通常表现不佳。
    - 让浏览器控制动画序列，允许浏览器优化性能和效果，如降低位于隐藏选项卡中的动画更新频率。

示例3-5-2：Javascript动画

5. 动画：5.2 2D转换

- CSS3 转换可对元素进行移动、缩放、转动、拉长或拉伸。

属性	描述
transform	适用于2D或3D转换的元素
transform-origin	允许您更改转化元素位置

5. 动画：5.2 2D转换

- CSS3 转换可对元素进行移动、缩放、转动、拉长或拉伸。

函数	描述
matrix(n,n,n,n,n,n)	定义 2D 转换，使用六个值的矩阵。
translate(x,y)	定义 2D 转换，沿着 X 和 Y 轴移动元素。
translateX(n)	定义 2D 转换，沿着 X 轴移动元素。
translateY(n)	定义 2D 转换，沿着 Y 轴移动元素。
scale(x,y)	定义 2D 缩放转换，改变元素的宽度和高度。
scaleX(n)	定义 2D 缩放转换，改变元素的宽度。
scaleY(n)	定义 2D 缩放转换，改变元素的高度。
rotate(angle)	定义 2D 旋转，在参数中规定角度。
skew(x-angle,y-angle)	定义 2D 倾斜转换，沿着 X 和 Y 轴。
skewX(angle)	定义 2D 倾斜转换，沿着 X 轴。
skewY(angle)	定义 2D 倾斜转换，沿着 Y 轴。

5. 动画： 5.2 2D转换

- translate方法
  - ranslate()方法，根据左(X轴)和顶部(Y轴)位置给定的参数，从当前元素位置移动。

示例3-5-3：translate方法

5. 动画： 5.2 2D转换

- rotate方法
  - rotate()方法，在一个给定度数顺时针旋转的元素。负值是允许的，这样是元素逆时针旋转。

示例3-5-4：rotate方法

5. 动画： 5.2 2D转换

- scale方法
  - scale()方法，该元素增加或减少的大小，取决于宽度（X轴）和高度（Y轴）的参数。

示例3-5-5：scale方法

5. 动画： 5.2 2D转换

- scale方法
  - scale()方法，该元素增加或减少的大小，取决于宽度（X轴）和高度（Y轴）的参数。

示例3-5-5：scale方法

5. 动画： 5.2 2D转换

- skew方法
  - skew()方法包含两个参数值，分别表示X轴和Y轴倾斜的角度，如果第二个参数为空，则默认为0，参数为负表示向相反方向倾斜。
    - skewX(<angle>);表示只在X轴(水平方向)倾斜。
    - skewY(<angle>);表示只在Y轴(垂直方向)倾斜。

示例3-5-6：skew方法

5. 动画：5.2 2D转换

- matrix方法
  - matrix()方法和2D变换方法合并成一个。
  - matrix 方法有六个参数，包含旋转，缩放，移动（平移）和倾斜功能。
  - matrix() 方法可以将多个 2D 变换（平移、旋转、缩放、倾斜）组合成一个单一的变换矩阵。
  - matrix() 函数接受六个参数 matrix(a, b, c, d, e, f)，其具体含义如下：
    - a：水平缩放。
    - b：垂直倾斜。
    - c：水平倾斜。
    - d：垂直缩放。
    - e：水平平移。
    - f：垂直平移。

示例3-5-7：matrix方法

5. 动画： 5.3 3D转换

- 转换属性

属性	描述
transform	向元素应用 2D 或 3D 转换。
transform-origin	允许你改变被转换元素的位置。
transform-style	规定被嵌套元素如何在 3D 空间中显示。
perspective	规定 3D 元素的透视效果。
perspective-origin	规定 3D 元素的底部位置。
backface-visibility	定义元素在不面对屏幕时是否可见。
transform	向元素应用 2D 或 3D 转换。

5. 动画： 5.3 3D转换

- 3D转换方法

函数	描述
matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)	定义 3D 转换，使用 16 个值的 4x4 矩阵。
translate3d(x,y,z)	定义 3D 转化。
translateX(x)	定义 3D 转化，仅使用用于 X 轴的值。
translateY(y)	定义 3D 转化，仅使用用于 Y 轴的值。
translateZ(z)	定义 3D 转化，仅使用用于 Z 轴的值。
scale3d(x,y,z)	定义 3D 缩放转换。
scaleX(x)	定义 3D 缩放转换，通过给定一个 X 轴的值。
scaleY(y)	定义 3D 缩放转换，通过给定一个 Y 轴的值。
scaleZ(z)	定义 3D 缩放转换，通过给定一个 Z 轴的值。
rotate3d(x,y,z,angle)	定义 3D 旋转。
rotateX(angle)	定义沿 X 轴的 3D 旋转。
rotateY(angle)	定义沿 Y 轴的 3D 旋转。
rotateZ(angle)	定义沿 Z 轴的 3D 旋转。
perspective(n)	定义 3D 转换元素的透视视图。

5. 动画：5.3 3D转换

- translate3d方法
  - translate3d() 用于在 3D 空间中对元素进行平移操作。它接受三个参数，分别表示元素在 X 轴、Y 轴和 Z 轴上的平移距离。

示例3-5-8：translate3d()方法

5. 动画： 5.3 3D转换

- scale3d方法
  - scale3d() 是 CSS 中用于在 3D 空间对元素进行缩放操作的函数，它接收三个参数，分别对应元素在 X 轴、Y 轴和 Z 轴方向上的缩放比例。

示例3-5-9：scale3d()方法

5. 动画： 5.3 3D转换

- rotate3d方法
  - rotate3d() 是 CSS 中用于在 3D 空间对元素进行旋转操作的函数。它接受四个参数，前三个参数分别表示旋转轴在 X、Y、Z 轴上的向量分量，最后一个参数表示旋转的角度。

示例3-5-10：rotate3d()方法

5. 动画： 5.3 3D转换

- transform-origin属性
  - transform-origin 属性用于设置元素变换的原点，默认情况下，变换原点是元素的中心点。通过改变 transform-origin 的值，可以改变元素变换的起始位置。

示例3-5-11：transform-origin

5. 动画：5.3 3D转换

- transform-style属性
  - transform-style 属性用于控制嵌套元素在 3D 空间中的呈现方式，它有两个取值：flat（默认值）和 preserve-3d。

示例3-5-12：transform-style

5. 动画： 5.3 3D转换

- perspective 与perspective-origin属性
  - perspective 属性用于为 3D 变换的元素创建透视效果，它模拟了人眼观察物体时近大远小的视觉感受。
  - perspective-origin 属性用于定义 perspective 透视效果的源点位置，也就是观察者的视角位置。它与 perspective 属性一起使用，帮助你控制 3D 变换元素在空间中的呈现效果。

```
perspective-origin: x-axis y-axis;
```

  - x-axis：指定水平方向的位置，可以使用百分比、长度值（如 px、em 等）或关键字（left、center、right）。
  - y-axis：指定垂直方向的位置，可以使用百分比、长度值或关键字（top、center、bottom）。

示例3-5-13：perspective 属性
示例3-5-14：perspective-origin属性
示例3-5-15：3D 立方体效果

5. 动画：5.3 3D转换

- backface-visibility属性
  - backface-visibility 用于控制元素在进行 3D 变换时，其背面是否可见。当一个元素在 3D 空间中旋转或翻转时，原本朝向观察者的一面可能会转到背面，这时 backface-visibility 属性就可以决定该元素的背面是否显示出来，其取值如下。
    - visible：默认值，表示元素的背面是可见的。
    - hidden：表示元素的背面是不可见的，当元素的背面朝向观察者时，元素会变得透明。

示例3-5-16：backface-visibility

5. 动画：5.4 过渡

- 过渡是元素从一种样式逐渐改变为另一种的效果，要实现这一点，必须规定两项内容：
  - 指定要添加效果的CSS属性
  - 指定效果的持续时间

属性	描述
transition	简写属性，用于在一个属性中设置四个过渡属性。
transition-property	规定应用过渡的 CSS 属性的名称。
transition-duration	定义过渡效果花费的时间。默认是 0。
transition-timing-function	规定过渡效果的时间曲线。默认是 "ease"。
transition-delay	规定过渡效果何时开始。默认是 0。

5. 动画：5.4 过渡

- transition 属性
  - transition 属性是 CSS 中用于实现元素状态过渡动画的重要属性，它可以让元素在两个状态之间的变化过程变得平滑，而不是瞬间切换。

```
transition: property duration timing-function delay;
```

  - property：指定要过渡的 CSS 属性名称，例如 width、background-color 等。也可以使用 all 表示所有可过渡的属性。
  - duration：过渡效果的持续时间，以秒（s）或毫秒（ms）为单位，例如 1s 或 500ms。
  - timing-function：定义过渡的速度曲线，常见值有 linear（匀速）、ease（默认，慢 - 快 - 慢）、ease-in（慢开始）、ease-out（慢结束）、ease-in-out（慢开始慢结束）等。还可以使用贝塞尔曲线来自定义速度曲线。
  - delay：过渡效果开始前的延迟时间，同样以秒（s）或毫秒（ms）为单位。

示例3-5-17：transition

5. 动画：5.5 动画

- CSS 动画允许元素在一段时间内平滑地从一种样式变换到另一种样式，为网页增添生动和交互性。

属性	描述
@keyframes	规定动画。
animation	所有动画属性的简写属性。
animation-name	规定 @keyframes 动画的名称。
animation-duration	规定动画完成一个周期所花费的秒或毫秒。默认是 0。
animation-timing-function	规定动画的速度曲线。默认是 "ease"。
animation-fill-mode	规定当动画不播放时（当动画完成时，或当动画有一个延迟未开始播放时），要应用到元素的样式。
animation-delay	规定动画何时开始。默认是 0。
animation-iteration-count	规定动画被播放的次数。默认是 1。
animation-direction	规定动画是否在下一周期逆向地播放。默认是 "normal"。
animation-play-state	规定动画是否正在运行或暂停。默认是 "running"。

5. 动画：5.5 动画

- animation属性
  - animation 是一个简写属性，用于将多个动画相关的子属性组合在一起。

```
animation: name duration timing-function delay iteration-count direction fill-mode play-state;
```

  - name：指定要使用的 @keyframes 动画名称。
  - duration：动画的持续时间，以秒（s）或毫秒（ms）为单位。
  - timing-function：定义动画的速度曲线，如 linear、ease、ease-in 等。
  - delay：动画开始前的延迟时间。
  - iteration-count：动画的播放次数，可以是具体的数字，也可以是 infinite 表示无限循环。

5. 动画：5.5 动画

- animation属性
  - animation 是一个简写属性，用于将多个动画相关的子属性组合在一起。

```
animation: name duration timing-function delay iteration-count direction fill-mode play-state;
```

  - direction：动画的播放方向，如 normal（正常播放）、reverse（反向播放）、alternate（交替播放，先正向后反向）等。
  - fill-mode：定义动画在播放前后的状态，如 forwards（动画结束后停留在最后一帧）、backwards（动画开始前应用第一帧样式）、both（同时应用 forwards 和 backwards 的效果）。
  - play-state：控制动画的播放状态，running（播放）或 paused（暂停）。

示例3-5-17：animation动画

案例演示

- 案例1：在线课程表
- 案例2：电脑壁纸效果预览

扩展1：CSS预处理

- CSS 预处理器是一种能够通过独有的语法来生成CSS的脚本语言，它的语法类似于 CSS，但提供了更多的功能和灵活性。预处理器的代码在被浏览器执行之前，会经过一个编译过程，将预处理器代码转换为标准的CSS代码。
- CSS预处理器优点：
  - 提高开发效率：可以定义变量存储颜色、尺寸等常用值，避免重复书写；模仿HTML的嵌套结构书写CSS，使代码结构更加清晰直观，易于理解和维护，避免了手动重复书写冗长的选择器；可以将一组常用的CSS属性封装成一个混合（Mixins），可以在不同的地方重复调用，提高了代码的复用性。
  - 易于维护：因为有可以嵌套、定义变量等特性，使得CSS代码更具逻辑性和可读性，不同模块的样式可以更好地组织在一起，方便查找和修改；可以根据项目的需求轻松扩展和修改样式，新的功能可以通过定义新的变量、混合或函数来实现，而不会影响现有的代码。
  - 跨浏览器兼容性：预处理器通常会自动为一些CSS属性添加浏览器前缀，以确保样式在不同的浏览器中都能正确显示。开发人员不需要手动添加这些前缀，减少了出错的可能性。
- CSS预处理器缺点：
  - 学习曲线：对于不熟悉预处理器的开发者来说，需要花费一定的时间学习新的语法和工具。预处理器的语法虽然相对容易理解，但与普通CSS还是有一定的区别，需要一定的时间来适应。
  - 编译过程增加复杂性：需要额外的编译步骤将预处理器代码转换为普通CSS代码，增加开发流程的复杂性。在调试时如果编译过程出现错误，可能需要花费一些时间来排查问题；编译过程可能会影响开发效率，特别是在大型项目中，编译时间可能会比较长。需要合理配置编译工具，以提高编译速度。
  - 依赖工具：使用预处理器通常需要依赖特定的工具和环境，如安装预处理器的软件包、配置构建工具等。如果工具出现问题或不兼容，可能会影响开发进度。

扩展1：CSS预处理

- 常见的 CSS 预处理器
  - Sass（Syntactically Awesome Style Sheets）
  - 有两种语法形式，SCSS（Sassy CSS）和 Sass。SCSS 语法与标准 CSS 语法类似，使用大括号和分号；Sass 语法则采用缩进的方式，不使用大括号和分号。

```scss
$primary-color: #007BFF;

.button {
    background-color: $primary-color;
    color: white;
    &:hover {
        background-color: darken($primary-color, 10%);
    }
}
```

扩展1：CSS预处理

- 常见的 CSS 预处理器
  - Less（Leaner Style Sheets）
  - 受 Sass 影响，语法与 CSS 接近，易于上手，同时支持 JavaScript 运行时环境，可在浏览器端和服务器端使用。

```less
@primary-color: #007BFF;

.button {
    background-color: @primary-color;
    color: white;
    &:hover {
        background-color: darken(@primary-color, 10%);
    }
}
```

扩展1：CSS预处理

- 常见的 CSS 预处理器
  - Stylus
  - 语法简洁灵活，支持多种书写风格，既可以使用类似 CSS 的语法，也可以省略大括号、分号等。

```
primary-color = #007BFF

.button
    background-color primary-color
    color white
    &:hover
        background-color darken(primary-color, 10%)
```

扩展2：CSS字体图标

- 字体图标是一种使用字体文件（如“.ttf”、“.woff”、“.woff2”等）来展示图标的图形显示技术，这些字体文件包含了特定的字形，每个字形代表了一个图标，与传统的图形文件不同，字体图标作为文本内容来处理，因此可以使用CSS的样式规则来控制它们的大小、颜色、阴影等属性。
- 字体图标有以下几点优势。
  - 可缩放性：字体图标是矢量图形，放大缩小时不失真。
  - 灵活性：通过CSS可以轻松改变字体图标的颜色、大小、旋转角度等属性，包括实现动画效果。
  - 轻量级：相比图像文件，一个包含多个图标的字体文件通常体积更小，有助于加快网页加载效果。
  - 易于维护：所有图标都包含在同一个字体文件中，更新或修改图标时只需替换一个文件即可。
  - 跨浏览器兼容性：字体图标在大多数浏览器都能正常显示。

扩展2：CSS字体图标

- 注册并登录
  - 访问Iconfont官网（https://www.iconfont.cn/），进行注册并登录。
- 选择并添加图标到项目
  - 在Iconfont的图标库中搜索你需要的图标，点击图标下方的“添加入库”按钮，将图标添加到个人图标库中，全部选择完成之后点击“购物车”按钮点击“添加至项目”按钮将图标添加到项目（已有项目或新建项目）中。
- 下载字体图标文件
  - 进入“资源管理”导航中的“我的项目”页面，选择对应的项目，点击“下载至本地”按钮，下载包含字体文件和CSS文件的压缩包。
- 将字体图标文件添加到项目中
  - 解压下载的压缩包，将字体文件放入项目的字体文件夹中。
- 在HTML中引入CSS文件
  - 在HTML文件的“<head>”部分，使用“<link>”标签引入之前下载的CSS文件，代码示例如下。

```
<link rel="stylesheet" href="./iconfont.css">
<!-- href属性应替换为CSS文件的实际路径 -->
```

扩展2：CSS字体图标

- 使用字体图标
  - 在HTML中，使用“<i>”或“”标签，并添加两个类名来显示字体图标，第一个类名通常是“iconfont”（或CSS文件中定义的类似名称），用于指定使用哪个字体文件，第二个类名是图标对应的类名，用于指定显示哪个图标，代码示例如下。

```
<span class="iconfont icon-xxx"></span>
<!-- icon-xxx 在实际使用时需要替换为实际的图标名称 -->
```

- 调整图标样式
  - 根据项目需要调整图标的大小、颜色等样式，示例代码如下。

```
.iconfont {
    font-size: 24px; /* 调整图标大小 */
    color: #333; /*修改图标颜色*/
}
```

重点回顾

- CSS基础
  - CSS定义与文件扩展名：CSS是一种样式表语言，用于描述HTML或XML文档的呈现方式，文件扩展名为“.css”。
  - CSS规则与注释：CSS规则由选择器和声明组成，声明由属性和值组成，CSS注释以“/”开始，以“/”结束。
  - CSS引入方式：HTML中引入CSS有内联方式、嵌入方式、链接方式和导入方式四种。
  - CSS工作原理：浏览器将HTML文档内容与CSS样式结合，并在屏幕上显示出来。
- 选择器
  - 基础选择器：包括通配符选择器、元素选择器、类选择器和ID选择器。
  - 层次选择器：包括后代选择器、子选择器、兄弟选择器。
  - 伪类选择器：包括链接与行为伪类、结构伪类选择器、目标伪类选择器、语言伪类选择器、表单相关伪类选择器、逻辑组合伪类选择器、空伪类选择器和根伪类选择器。
  - 伪元素选择器：包括::before和::after、::first-letter、::first-line、::selection和::marker。。
- 文字样式
  - 文本样式：包括color、text-indent、line-height、letter-spacing、text-align、text-decoration、text-transform、white-space、word-spacing等属性。
  - 字体样式：包括font-family、font-size、font-weight、font-style等属性。
  - 文本效果：包括text-overflow、text-shadow、overflow-wrap等属性。

重点回顾

- 背景与边框
  - background属性：包括background-color、background-image、background-repeat、background-attachment、background-position、background-size、background-origin、background-clip等子属性。
  - border属性：包括border-width、border-style、border-color等子属性。
  - border-radius属性：用于为元素的边框添加圆角效果。
  - border-image属性：用于使用图像来替代元素的边框样式。
- 动画
  - Web动画：包括GIF动画、JavaScript动画和CSS动画。
  - 2D转换：包括translate、rotate、scale、skew、matrix等方法。
  - 3D转换：包括translate3d、scale3d、rotate3d、transform-origin、transform-style、perspective、perspective-origin、backface-visibility等属性。
  - 过渡：使用transition属性实现元素状态的平滑过渡。
  - 关键帧动画：使用@keyframes定义动画的关键帧，使用animation属性控制动画的播放。
- 扩展
  - CSS预处理器：包括Less和Sass（Scss）的特点、使用步骤和示例。
  - CSS字体图标：介绍了字体图标的概念、优势、使用步骤和示例。

信创智能医疗系统研发课程体系

河南中医药大学信息技术学院（智能医疗行业学院）

河南中医药大学信息技术学院（智能医疗行业学院）智能医疗教研室

河南中医药大学医疗健康信息工程技术研究所