100个css优化技巧

1. **启动平滑滚动**

添加scroll-behavior：smooth到元素中<html>，使整个页面能够平滑滚动。

```
html{   scroll-behavior: smooth;  }
```

2. **链接的属性选择器**

此选择器以href属性以“https”开头的链接为目标。

```
a[href^="https"] {
   color: blue;
}
```

3. **~用于合并节点**

选择作为<h2>同级元素的所有<p>元素。

```
h2 ~ p {
   color: blue;
}
```

4. **:not()伪类**

此选择器将样式应用于不具有类“Special”的 li。

```
li:not(.special) {
   font-style: italic;
}
```

5. **响应式排版的视窗单位 vw**

使用视区单位(vw, vh, vmin, vmax)可以使字体大小与视区大小相对应。

```
h1 {
   font-size: 5vw;
}
```

6. **:empty 对于空元素**

此选择器以空的<p>元素为目标并隐藏它们。

```
p:empty {
  display: none;
}
```

7. 自定义特性(变量)

可以定义和使用自定义特性，以便更轻松地创建主题和进行维护。

```
:root {
  --main-color: #3498db;
}

h1 {
  color: var(--main-color);
}
```

8. Object-fit 图像控件的适配性

object-fit 控制替换元素(如<img>)的内容应该如何调整大小。

```
img {
  width: 100px;
  height: 100px;
  object-fit: cover;
}
```

9. 简化布局的网格

Css网格提供了一种功能强大的方式来以更直接的方式创建布局。

```
.container {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr;
}
```

10. :focus-within 伪类

如果一个元素包含任何带有:focus的子元素，则:focus-Win会选择该元素。

```
form:focus-within {
  box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
}
```

11. 使用Flexbox垂直居中

使用FlexBox可轻松地将内容在容器中水平和垂直居中。

```
.container {
  display: flex;
  align-items: center;
  justify-content: center;
}
```

12. 自定义选定内容的突出显示颜色

自定义在网页上选择文本时的突出显示颜色。

```
::selection {
  background-color: #ffcc00;
  color: #333;
}
```

13. 设置占位符文本的样式

设置输入字段内占位符文本的样式。

```
::placeholder {
  color: #999;
  font-style: italic;
}
```

14. 渐变边界

使用Backback-Clip属性创建渐变边框。

```
.element {
  border: 2px solid transparent;
  background-clip: padding-box;
  background-image: linear-gradient(to right, red, blue);
}
```

15. 使用vw实现可变字体大小

根据视口宽度调整字体大小，实现更加响应式的排版。

```
body {
  font-size: calc(16px + 1vw);
}
```

16. 使用锥形渐变创建多彩元素

利用锥形渐变创建丰富多彩且动态的背景。

```
.element {
  background: conic-gradient(#ff5733, #33ff57, #5733ff);
}
```

17. 使用clamp()函数实现响应式文本

使用clamp()函数设置字体大小的范围，确保在不同屏幕尺寸下的可读性。

```
.text {
  font-size: clamp(16px, 4vw, 24px);
}
```

18. 使用font-display: swap;优化字体加载

使用font-display: swap;属性提高网页字体性能，让自定义字体加载时显示备用字体。

```
@font-face {
  font-family: 'YourFont';
  src: url('your-font.woff2') format('woff2');
```

```
  font-display: swap;
}
```

19. **自定义滚动吸附点**

为了实现更顺畅的滚动体验，特别是在图库或滑块中，实现自定义滚动吸附点。

```
.scroll-container {
  scroll-snap-type: y mandatory;
}

.scroll-item {
  scroll-snap-align: start;
}
```

20. **使用字体变体设置进行可变字体样式**

利用可变字体和font-variation-settings属性对字体的粗细、样式等进行精细调节。

```
.text {
  font-family: 'YourVariableFont', sans-serif;
  font-variation-settings: 'wght' 500, 'ital' 1;
}
```

21. **自定义下划线样式**

使用border-bottom和text-decoration的组合来自定义链接的下划线样式。

```
a {
  text-decoration: none;
  border-bottom: 1px solid #3498db;
}
```

22. **隐藏无障碍文本**

使用class sr-only来在视觉上隐藏元素，但保持其对屏幕阅读器的可访问性。

```
.sr-only {
  position: absolute;
```

```
  width: 1px;
  height: 1px;
  margin: -1px;
  padding: 0;
  overflow: hidden;
  clip: rect(0, 0, 0, 0);
  border: 0;
}
```

23. 保持元素纵横比

通过使用padding来保持元素（如图片或视频）的纵横比。

```
.aspect-ratio-box {
  position: relative;
  width: 100%;
  padding-bottom: 75%; /* 根据需要调整 */
}

.aspect-ratio-box > iframe {
  position: absolute;
  width: 100%;
  height: 100%;
}
```

24. 选择偶数和奇数元素

使用:nth-child伪类来为交替元素设置样式。

```
li:nth-child(even) {
  background-color: #f2f2f2;
}

li:nth-child(odd) {
  background-color: #e6e6e6;
```

```
}
```

25. CSS计数器

使用counter-reset和counter-increment属性在列表中创建自动编号。

```
ol {
  counter-reset: item;
}

li {
  counter-increment: item;
}

li::before {
  content: counter(item) ". ";
}
```

26. 多重背景图片

使用不同属性为元素应用多个背景图片。

```
.bg {
  background-image: url('image1.jpg'), url('image2.jpg');
  background-position: top left, bottom right;
  background-repeat: no-repeat, repeat-x;
}
```

27. 优化文本流畅性的连字符

通过使用hyphens属性允许自动连字符以提高文本的可读性。

```
p {
  hyphens: auto;
}
```

28. 使用CSS变量进行动态样式

利用CSS变量创建动态且可重用的样式。

```
:root {
  --main-color: #3498db;
}

.element {
  color: var(--main-color);
}
```

29. 键盘导航的焦点样式

改善焦点样式以提高键盘导航和可访问性。

```
:focus {
  outline: 2px solid #27ae60;
}
```

30. 平滑渐变过渡

为渐变背景应用平滑过渡效果，增强视觉效果。

```
.gradient-box {
  background: linear-gradient(45deg, #3498db, #2ecc71);
  transition: background 0.5s ease;
}

.gradient-box:hover {
  background: linear-gradient(45deg, #e74c3c, #f39c12);
}
```

31. 文本描边效果

为文本添加描边，产生独特的视觉效果。

```
h1 {
  color: #3498db;
  -webkit-text-stroke: 2px #2c3e50;
}
```

32. 纯CSS汉堡菜单

创建一个简单的汉堡菜单，无需使用JavaScript。

```
.menu-toggle {
  display: none;
}

.menu-toggle:checked + nav {
  display: block;
}
/* 在这里添加汉堡菜单图标和菜单样式 */
```

33. CSS :is()选择器

使用:is()伪类简化复杂的选择器。

```
:is(h1, h2, h3) {
  color: blue;
}
```

34. CSS变量中的计算

在CSS变量中进行计算，实现动态样式。

```
:root {
  --base-size: 16px;
  --header-size: calc(var(--base-size) * 2);
}
```

```
h1 {
  font-size: var(--header-size);
}
```

35. attr()函数用于内容

使用attr()函数检索和显示属性值。

```
div::before {
  content: attr(data-custom-content);
}
```

36. CSS遮罩效果

为图像应用遮罩，创造出独特的效果。

```
.masked-image {
  mask: url(mask.svg);
  mask-size: cover;
}
```

37. 混合模式

尝试使用混合模式创建有趣的色彩效果。

```
.blend-mode {
  background: url(image.jpg);
  mix-blend-mode: screen;
}
```

38. 纵横比属性

使用纵横比属性简化纵横比盒子的创建。

```
.aspect-ratio-box {
```

```
  aspect-ratio: 16/9;
}
```

39. **shape-outside实现文本环绕**

使用shape-outside属性使文本围绕指定形状，实现更动态的布局。

```
.shape-wrap {
  float: left;
  width: 150px;
  height: 150px;
  shape-outside: circle(50%);
}
```

40. **ch单位用于一致的尺寸**

ch单位表示所选字体中字符“0”的宽度，可用于创建一致且响应式的布局。

```
h1 {
  font-size: 2ch;
}
```

41. **::marker伪元素**

使用::marker伪元素为列表项标记设置样式。

```
li::marker {
  color: blue;
}
```

42. **element()函数用于背景**

使用element()函数动态引用元素作为背景。

```
.background {
  background: element(#targetElement);
}
```

43. **Flexbox实现粘性底部**

使用Flexbox创建粘性底部布局。

```
body {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

main {
  flex: 1;
}
```

44. scroll-padding实现平滑滚动

通过调整scroll padding来改善滚动行为。

```
html {
  scroll-padding: 20px;
}
```

45. 交互式高亮效果

使用CSS变量创建交互式高亮效果。

```
.highlight {
  --highlight-color: #e74c3c;
  background-image: linear-gradient(transparent 0%, var(--highlight-color) 0%);
  background-size: 100% 200%;
  transition: background-position 0.3s;
}

.highlight:hover {
  background-position: 0 100%;
}
```

46. 自定义单选框和复选框样式

无需图像，样式化单选框和复选框。

```
input[type="radio"] {
  appearance: none;
  -webkit-appearance: none;
  border-radius: 50%;
  width: 16px;
  height: 16px;
  border: 2px solid #3498db;
}

input[type="checkbox"] {
  appearance: none;
  -webkit-appearance: none;
  width: 16px;
  height: 16px;
  border: 2px solid #e74c3c;
}
```

47. textarea的resize属性

使用resize属性控制textarea的调整大小行为。

```
textarea {
  resize: vertical;
}
```

48. 文本渐变效果

使用background-clip和text-fill-color属性为文本创建渐变效果。

```
.gradient-text {
  background-image: linear-gradient(45deg, #3498db, #2ecc71);
  background-clip: text;
  color: transparent;
}
```

49. 对长单词使用word-break属性

使用word-break属性控制长单词或没有空格的字符串的断行和换行。

```
.long-words {
  word-break: break-all;
}
```

50. font-variation-settings用于可变字体

使用font-variation-settings属性微调可变字体样式。

```
.custom-font {
  font-family: 'MyVariableFont';
  font-variation-settings: 'wght' 600, 'ital' 1;
}
```

51. 混合模式用于创意叠加效果

使用混合模式为元素应用叠加效果，创造出有趣的视觉效果。

```
.overlay {
  mix-blend-mode: overlay;
}
```

52. 为损坏的图像应用样式

使用:broken伪类为损坏的图像应用样式。

```
img:broken {
  filter: grayscale(100%);
}
```

53. CSS形状

使用CSS形状为设计创建有趣的效果。

```
.shape {
  shape-outside: circle(50%);
}
```

54. 属性选择器用于子字符串匹配

使用*=操作符的属性选择器进行子字符串匹配。

```
[data-attribute*="value"] {
  /* 样式 */
}
```

55. backdrop-filter用于模糊背景

使用backdrop-filter为背景应用模糊效果，实现毛玻璃效果。

```
.element {
  backdrop-filter: blur(10px);
}
```

56. CSS环境变量

使用env()函数在CSS中访问环境变量。

```
.element {
  margin-top: env(safe-area-inset-top);
}
```

57. CSS属性计数器

使用:nth-child选择器计算特定属性值的出现次数。

```
[data-category="example"]:nth-child(3) {
  /* 第三次出现的样式 */
}
```

58. CSS形状实现文本环绕

使用shape-outside结合polygon()函数精确地控制文本围绕不规则形状的布局。

```
.text-wrap {
  shape-outside: polygon(0 0, 100% 0, 100% 100%);
}
```

59. 自定义鼠标样式

使用cursor属性更改鼠标样式。

```
.custom-cursor {
  cursor: pointer;
}
```

60. HSLA用于透明颜色

使用HSLA值表示透明颜色，对alpha通道进行更精细的控制。

```
.transparent-bg {
  background-color: hsla(120, 100%, 50%, 0.5);
}
```

61. text-orientation实现纵向文本

使用text-orientation属性将文本垂直旋转。

```
.vertical-text {
  text-orientation: upright;
}
```

62. font-variant用于小型大写字母

使用font-variant属性应用小型大写字母样式。

```
.small-caps {
  font-variant: small-caps;
}
```

63. box-decoration-break用于背景分割

使用box-decoration-break属性控制跨多行断开的元素的背景，实现更灵活的文本环绕。

```
.split-background {
  box-decoration-break: clone;
}
```

64. :focus-visible用于特定焦点样式

仅在元素处于焦点且焦点不是由鼠标单击提供时应用样式。

```
input:focus-visible {
  outline: 2px solid blue;
}
```

65. text-rendering实现最佳字体呈现

通过text-rendering属性改善文本呈现效果。

```
.optimized-text {
  text-rendering: optimizeLegibility;
}
```

66. initial-letter用于大写字母

使用initial-letter为块级元素的第一个字母应用样式。

```
p::first-letter {
  font-size: 2em;
}
```

67. overscroll-behavior用于滚动过度

控制用户滚动超出滚动容器边界时的行为。

```
.scroll-container {
  overscroll-behavior: contain;
}
```

68. writing-mode实现纵向布局

使用writing-mode属性创建纵向布局。

```
.vertical-layout {
  writing-mode: vertical-rl;
}
```

69. ::cue用于HTML5标题样式

使用::cue伪元素为HTML5字幕文本应用样式。

```
::cue {
  color: blue;
}
```

70. line-clamp截断多行文本

使用line-clamp属性限制元素中显示的行数。

```
.truncated-text {
  display: -webkit-box;
  -webkit-line-clamp: 3;
  -webkit-box-orient: vertical;
  overflow: hidden;
}
```

71. scroll-snap-align

scroll-snap-align属性控制滚动容器内滚动捕捉点的对齐方式，确保对滚动行为进行精确控制，提升用户体验。

```
.container {
  scroll-snap-type: x mandatory;
}

.item {
  scroll-snap-align: center;
}
```

72. overscroll-behavior

overscroll-behavior使您可以定义浏览器在滚动超出范围时的处理方式，从而避免不必要的滚动效果，提升整体滚动体验。

```
.scrollable {
  overscroll-behavior: contain;
}
```

73. font-kerning

font-kerning允许对字符间距进行微调，通过调整文本元素中字符之间的间距，确保最佳的可读性。

```
p {
  font-kerning: auto;
}
```

74. shape-margin

与CSS形状一起使用时，shape-margin指定浮动元素形状周围的边距，可以更精确地控制文本环绕和布局。

```
.shape {
  shape-margin: 20px;
}
```

75. scroll-margin

scroll-margin设置滚动容器边缘与滚动内容开始之间的边距，提升用户体验，为滚动提供缓冲空间。

```
.container {
  scroll-margin-top: 100px;
}
```

76. tab-size

tab-size属性控制文本区域中制表符的宽度，确保在不同用户代理中一致的显示。

```
pre {
  tab-size: 4;
```

```
}
```

77. **text-align-last**

text-align-last决定块级元素中最后一行文本的对齐方式，为多行块文本提供对齐控制。

```
p {
  text-align-last: justify;
}
```

78. **text-justify**

此属性控制文本两端对齐的行为，指定是使用单词间还是字符间距进行文本对齐。

```
p {
  text-align: justify;
  text-justify: inter-word;
}
```

79. **column-fill**

column-fill决定如何在多列布局中分配内容，允许内容依次或平衡分布在列中。

```
.container {
  column-count: 3;
  column-fill: auto;
}
```

80. **outline-offset**

outline-offset调整轮廓与元素边缘之间的空间，不影响布局，为轮廓的外观提供更细致的控制。

```
button {
  outline: 2px solid blue;
  outline-offset: 4px;
}
```

81. **font-variant-numeric**

此属性允许精细控制数字字体渲染，启用特性如数字大小写和分数、序数指示器等。

```
p {
```

```
  font-variant-numeric: lining-nums;
}
```

82. font-optical-sizing

启用或禁用字体光学大小调整，以调整字符的间距和比例，实现在不同字体大小下更好的视觉和谐。

```
p {
  font-optical-sizing: auto;
}
```

83. text-decoration-thickness

精确控制文本装饰（如下划线、上划线和删除线）的粗细，实现精细化定制。

```
p {
  text-decoration-thickness: 2px;
}
```

84. text-decoration-skip-ink

text-decoration-skip-ink属性控制文本装饰（如下划线）在被墨水遮挡的情况下是否继续绘制，提升可读性。

```
a {
  text-decoration-skip-ink: auto;
}
```

85. word-spacing

word-spacing属性控制字词间距，调整文本的紧凑度和可读性。

```
p {
  word-spacing: 2px;
}
```

86. hyphenation

通过调整断字点和断字行为，提高文本在窄列中的美观度和可读性。

```
p {
  hyphens: auto;
}
```

87. background-blend-mode

background-blend-mode属性允许背景图像与其下方的背景颜色进行混合，产生出独特的视觉效果。

```
.element {
  background-image: url('image.jpg');
  background-color: #3498db;
  background-blend-mode: multiply;
}
```

88. text-decoration-color

text-decoration-color属性控制文本装饰的颜色，为链接和装饰文本提供更灵活的样式。

```
a {
  text-decoration: underline;
  text-decoration-color: red;
}
```

89. overflow-anchor

overflow-anchor属性指定了在容器的滚动范围内滚动时哪些内容应保持可见，提升用户体验。

```
.container {
  overflow-anchor: none;
}
```

90. contain-intrinsic-size

contain-intrinsic-size属性定义了内联大小计算函数的大小，实现更精确的布局控制。

```
img {
  contain-intrinsic-size: 200px 300px;
}
```

91. line-gap

line-gap属性定义了元素的行间距，可以为文字和其他行内元素提供更大的空间。

```
p {
  line-gap: 1.5;
}
```

92. text-underline-offset

text-underline-offset属性控制文本下划线相对于基线的垂直偏移量，实现更加精确的下划线定位。

```
a {
  text-decoration: underline;
  text-underline-offset: 3px;
}
```

93. text-decoration-line

text-decoration-line属性指定要绘制的装饰线的类型，可用于单独控制上划线、下划线、删除线等。

```
a {
  text-decoration-line: underline overline;
}
```

94. text-emphasis

text-emphasis属性为文本设置强调标志，用于提高关键字的可读性。

```
em {
  text-emphasis: filled circle;
}
```

95. text-orientation

text-orientation属性控制文本的方向，适用于纵向文本和日文排版等情况。

```
.vertical-text {
  text-orientation: sideways-right;
```

```
}
```

96. background-origin

background-origin属性确定背景图片的起始位置，影响背景图片与边框的相对位置。

```
.element {
  background-image: url('image.jpg');
  background-origin: content-box;
}
```

97. counter-set

counter-set属性在使用counter-reset创建的计数器值基础上，显式设置一个新的值。

```
ol {
  counter-set: section 1;
}
```

98. hanging-punctuation

hanging-punctuation属性控制标点符号是否在行框之外悬挂，以提高排版的美观度和可读性。

```
p {
  hanging-punctuation: last allow-end;
}
```

99. line-break

line-break属性控制文本换行行为，确保在指定的断点处进行断行。

```
p {
  line-break: anywhere;
}
```

100. text-justify

text-justify属性指定如何分配额外的空间，以便充分利用容器的宽度。

```
p {
   text-align: justify;
   text-justify: inter-word;
}
```