布局

盒尺寸重置

重置盒子模型，以便`width`s和`height`s并没有受到`border`还是`padding`他们的影响；

```
html {
  box-sizing: border-box;
}
*,
*::before,
*::after {
  box-sizing: inherit;
}
```

说明

1. `box-sizing: border-box`添加`padding`或者`border`不影响元素的`width`或者`height`。
2. `box-sizing: inherit`使元素尊重其父元素`box-sizing`规则。

清除浮动

确保元素自清除其子元素。

注意:只有当您仍然使用浮动来构建布局时，这才是有用的。请考虑使用一种现代的方法与柔性盒布局或网格布局。

```
<div class="clearfix">
  <div class="floated">float a</div>
  <div class="floated">float b</div>
  <div class="floated">float c</div>
</div>
```

```
.clearfix::after {
  content: '';
  display: block;
  clear: both;
}
.floated {
  float: left;
}
```

说明

1. `.clearfix::after`定义伪元素。
2. `content: ''`允许伪元素影响布局。
3. `clear: both`指示元素的左侧、右侧或两侧不能与同一块格式上下文中较早浮动的元素相邻。

为了使此代码段正常工作，您需要确保容器中没有非浮动子级，并且clearfixed容器之前没有高浮动，但格式上下文相同(例如浮动列)。

等宽高比

给定可变宽度的元素，它将确保其高度以响应的方式保持成比例(即，其宽度与高度的比率保持恒定)。

```
<div class="constant-width-to-height-ratio"></div>
```

```
.constant-width-to-height-ratio {
  background: #333;
  width: 50%;
}
.constant-width-to-height-ratio::before {
  content: '';
  padding-top: 100%;
  float: left;
}
.constant-width-to-height-ratio::after {
  content: '';
  display: block;
  clear: both;
}
```

说明

`padding-top` 在...上 `::before` 伪元素使元素的高度等于其宽度的百分比。`100%` 因此表示元素的高度始终为 `100%` 的宽度，创建一个响应正方形。

此方法还允许内容正常放置在元素内部。

均匀分布的子元素和Flexbox垂直居中

在父元素中均匀分布子元素。

使用 `flexbox` 在父元素内水平和垂直居中放置子元素。

```
<div class="evenly-distributed-children">
  <p>Item1</p>
  <p>Item2</p>
  <p>Item3</p>
</div>
```

```
.evenly-distributed-children {
  display: flex;
  justify-content: space-between;
}

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

说明（需要前缀才能获得完全支持）

1. `display: flex` 启用弹性箱。
2. `justify-content: space-between` 水平均匀分布子元素。第一个项目位于左边缘，而最后一个项目位于右边缘。或者，使用 `justify-content: space-around` 给子元素们分配空间，而不是在他们之间。

3. justify-content: center ` 将子元素水平居中。
4. align-items: center` 将子元素垂直居中。

网格居中、网格布局

```
<!--使用`grid`在父元素内水平和垂直居中放置子元素-->
<div class="grid-centering">
  <div class="child">Centered content.</div>
</div>
<!--基本网站布局使用`grid-->
<div class="grid-layout">
  <div class="header">Header</div>
  <div class="sidebar">Sidebar</div>
  <div class="content">
    Content
    <br>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit.
  </div>
  <div class="footer">Footer</div>
</div>
```

```
.grid-centering {
  display: grid;
  justify-content: center;
  align-items: center;
}

.grid-layout {
  display: grid;
  grid-gap: 10px;
  grid-template-columns: repeat(3, 1fr);
  grid-template-areas:
    'sidebar header header'
    'sidebar content content'
    'sidebar footer  footer';
  color: white;
}
.grid-layout > div {
  background: #333;
  padding: 10px;
}
.sidebar {
  grid-area: sidebar;
}
.content {
  grid-area: content;
}
.header {
  grid-area: header;
}
.footer {
  grid-area: footer;
}
```

说明

1. `display: grid` 启用网格。
2. `justify-content: center` 将子元素水平居中。
3. `align-items: center` 将子元素垂直居中。
4. `grid-gap: 10px` 定义元素之间的间距。
5. `grid-template-columns: repeat(3, 1fr)` 定义3列大小相同。
6. `grid-template-areas` 定义网格区域的名称。
7. `grid-area: sidebar` 使元素使用名称为的区域`sidebar` 。

文本操作

- 如果文本长度超过一行，它将被截断并以省略号结束。

```
<p class="truncate-text">If I exceed one line's width, I will be truncated.</p>
```

```
.truncate-text {
  overflow: hidden;
  white-space: nowrap;
  text-overflow: ellipsis;
  width: 200px;
}
```

说明-仅适用于单个行元素

1. overflow: hidden ` 防止文本溢出其尺寸(对于块，100 %宽度和自动高度)。
2. `white-space: nowrap` 防止文本高度超过一行。
3. `text-overflow: ellipsis` 使其在文本超出其维度时以省略号结尾。
4. `width: 200px;` 确保元素具有维度，以知道何时获取省略号

- 更改文本选择的样式。

```
<p class="custom-text-selection">Select some of this text.</p>
```

```
::selection {
  background: aquamarine;
  color: black;
}
.custom-text-selection::selection {
  background: deeppink;
  color: white;
}
```

说明-需要前缀来表示完全支持，实际上不在任何规范中。

`::selection` 定义元素上的伪选择器，以便在选定元素时设置其中文本的样式。请注意，如果不组合任何其他选择器，则样式将在文档根级别应用于任何可选元素。

- 创建文本显示为“蚀刻”或刻在背景中的效果。

```
<p class="etched-text">I appear etched into the background.</p>
```

```
.etched-text {
  text-shadow: 0 2px white;
  font-size: 1.5rem;
  font-weight: bold;
  color: #b8bec5;
}
```

说明

`text-shadow: 0 2px white` 创建白色阴影偏移 `0px` 水平和 `2px` 垂直于原点位置。

背景必须比阴影暗，效果才能发挥作用。

文字颜色应该稍微褪色，使其看起来像是刻在背景上的。

- 为文本提供渐变颜色。

```
<p class="gradient-text">Gradient text</p>
```

```
.gradient-text {
  background: -webkit-linear-gradient(pink, red);
  -webkit-text-fill-color: transparent;
  -webkit-background-clip: text;
}
```

说明-使用非标准属性

1. `background: -webkit-linear-gradient(...)` 为文本元素提供渐变背景。
2. `webkit-text-fill-color: transparent` 使用透明颜色填充文本。
3. `webkit-background-clip: text` 用文本剪辑背景，用渐变背景作为颜色填充文本。

- 比...更好的选择 `text-decoration: underline` 其中后代不裁剪下划线。本机实现为 `text-decoration-skip-ink: auto` 但它对下划线的控制较少。

```
<p class="pretty-text-underline">Pretty text underline without clipping
descending letters.</p>
```

```
.pretty-text-underline {
  font-family: Arial, sans-serif;
  display: inline;
  font-size: 18px;
  text-shadow: 1px 1px 0 #f5f6f9, -1px 1px 0 #f5f6f9, -1px -1px 0 #f5f6f9, 1px
-1px 0 #f5f6f9;
  background-image: linear-gradient(90deg, currentColor 100%, transparent 100%);
  background-position: 0 0.98em;
  background-repeat: repeat-x;
  background-size: 1px 1px;
}
.pretty-text-underline::-moz-selection {
  background-color: rgba(0, 150, 255, 0.3);
  text-shadow: none;
}
.pretty-text-underline::selection {
  background-color: rgba(0, 150, 255, 0.3);
  text-shadow: none;
}
```

说明-下划线与文本的距离取决于字体的内部度量，因此必须确保每个人看到相同的字体(即，不会出现基于操作系统而更改的系统字体)。

1. `text-shadow: ...` 具有4个偏移值，覆盖4x 4 px区域，以确保下划线具有“厚”阴影，该阴影覆盖后代裁剪下划线的线。使用与背景匹配的颜色。对于较大字体，请使用较大字体 `px` 大小。
2. `background-image: linear-gradient(...)` 使用当前文本颜色(`currentColor`)创建90度渐变.
3. `background-*` 属性将渐变的大小设置为1x1px，并沿x轴重复。
4. `::selection` 伪选择器确保文本阴影不会干扰文本选择。

- 使用操作系统的本机字体来接近本机应用程序感觉。

```
<p class="system-font-stack">This text uses the system font.</p>
```

```
.system-font-stack {
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen-
Sans, Ubuntu,
    Cantarell, 'Helvetica Neue', Helvetica, Arial, sans-serif;
}
```

说明

浏览器查找每个连续的字体，如果可能，则首选第一种字体，如果找不到该字体(在系统上或在CSS中定义)，则返回到下一种字体。

1. `-apple-system` 是旧金山，用于iOS和macOS（不是Chrome）
2. `BlinkMacSystemFont` 是旧金山，在macOS Chrome上使用
3. `Segoe UI` 在Windows 10上使用
4. `Roboto` 在安卓系统上使用
5. `Oxygen-Sans` 是在GNU + Linux上使用的
6. `Ubuntu` 是在Linux上使用的
7. `"Helvetica Neue"` 和 `Helvetica` 在macOS 10.10及以下版本中使用(因为有空格，所以用引号括起来)
8. `Arial` 是所有操作系统都广泛支持的字体
9. `sans-serif` 如果不支持任何其他字体，则为备用无衬线字体

- 内容不可选择

```
<p>你可以选择我。</p>
<p class="unselectable">你不能选择我！</p>
```

```
.unselectable {
  user-select: none;
}
```

说明-需要前缀才能获得完全支持。这不是防止用户复制内容的安全方法

`user-select: none` 指定无法选取文字。

视觉的

计数器

计数器本质上是由CSS维护的变量，其值可以通过CSS规则递增以跟踪它们的使用次数。

```
<ul>
  <li>List item</li>
  <li>List item</li>
  <li>List item</li>
</ul>
```

```
ul {
  counter-reset: counter;
}
li::before {
  counter-increment: counter;
  content: counters(counter, '.') ' ';
}
```

说明

您可以使用任何类型的HTML创建有序列表。

1. `counter-reset` 初始化计数器，值为计数器的名称。默认情况下，计数器从0开始。此属性还可用于将其值更改为任何特定数字。
2. `counter-increment` 用于可计数的元素中。一次 `counter-reset` 初始化后，计数器的值可以增加或减少。
3. `counter(name, style)` 显示区段计数器的值。一般用于 `content` 财产。此函数可以接收两个参数，第一个参数作为计数器的名称，第二个参数可以是 `decimal` 或者 `upper-roman` （ `decimal` 默认情况下)。
4. `counters(counter, string, style)` 显示区段计数器的值。一般用于 `content` 财产。此函数可以接收三个参数，第一个参数作为计数器的名称，第二个参数可以包括在计数器后面的字符串，第三个参数可以是 `decimal` 或者 `upper-roman` （ `decimal` 默认情况下)。
5. CSS计数器对于生成大纲列表特别有用，因为计数器的新实例是在子元素中自动创建的。使用 `counters()` 函数，可以在不同级别的嵌套计数器之间插入分隔文本。

自定义滚动条

在WebKit平台上自定义具有可滚动溢出的文档和元素的滚动条样式。

```
<div class="custom-scrollbar">
  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Iure id
exercitationem nulla qui repellat laborum vitae, molestias tempora velit natus.
Quas, assumenda nisi. Quisquam enim qui iure, consequatur velit sit?</p>
</div>
```

```
/* Document scrollbar */
::-webkit-scrollbar {
  width: 8px;
}
::-webkit-scrollbar-track {
  box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);
  border-radius: 10px;
}
::-webkit-scrollbar-thumb {
  border-radius: 10px;
  box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.5);
```

```
}
/* Scrollable element */
.some-element::webkit-scrollbar {}
```

说明-滚动条样式不在任何标准轨道上

1. `::-webkit-scrollbar` 定位整个滚动条元素。
2. `::-webkit-scrollbar-track` 仅针对滚动条轨道。
3. `::-webkit-scrollbar-thumb` 瞄准滚动条拇指。

动态阴影

创建与类似的阴影 `box-shadow` 而是基于元素本身的颜色。

```
<div class="dynamic-shadow-parent">
  <div class="dynamic-shadow"></div>
</div>
```

```
.dynamic-shadow-parent {
  position: relative;
  z-index: 1;
}
.dynamic-shadow {
  position: relative;
  width: 10rem;
  height: 10rem;
  background: linear-gradient(75deg, #6d78ff, #00ffb8);
}
.dynamic-shadow::after {
  content: '';
  width: 100%;
  height: 100%;
  position: absolute;
  background: inherit;
  top: 0.5rem;
  filter: blur(0.4rem);
  opacity: 0.7;
  z-index: -1;
}
```

说明-需要前缀才能获得完全支持

代码片段需要一些复杂的情况来正确堆叠上下文，这样伪元素将定位在元素本身的下面，同时仍然可见。

1. `position: relative` 在父元素上为子元素建立笛卡尔定位上下文。
2. `z-index: 1` 建立新的堆叠内容。
3. `position: relative` 在子级上建立伪元素的定位上下文。
4. `::after` 定义伪元素。
5. `position: absolute` 从文档流中取出伪元素，并将其相对于父元素定位。
6. `width: 100%` 和 `height: 100%` 调整伪元素的大小以填充其父元素的尺寸，使其大小相等。
7. `background: inherit` 使伪元素继承在元素上指定的线性渐变。
8. `top: 0.5rem` 将伪元素从其父元素稍微向下偏移。
9. `filter: blur(0.4rem)` 将模糊伪元素以在下面创建阴影的外观。
10. `opacity: 0.7` 使伪元素部分透明。

11. `z-index: -1` 将伪元素定位在父元素后面。

细线边框

为元素提供一个边框，宽度等于1个本地设备像素，可以显得非常清晰和清晰。

```
<div class="hairline-border">text</div>
```

```
.hairline-border {
  box-shadow: 0 0 0 1px;
}
@media (min-resolution: 2dppx) {
  .hairline-border {
    box-shadow: 0 0 0 0.5px;
  }
}
@media (min-resolution: 3dppx) {
  .hairline-border {
    box-shadow: 0 0 0 0.33333333px;
  }
}
@media (min-resolution: 4dppx) {
  .hairline-border {
    box-shadow: 0 0 0 0.25px;
  }
}
```

说明-需要备用语法和JavaScript用户代理检查以获得完全支持

1. `box-shadow` ，当仅使用扩展时，添加可以使用子像素*的伪边框。
2. 使用 `@media (min-resolution: ...)` 为了检查器件像素比(`1dppx` 等于96 DPI)，将 `box-shadow` 的分布设置为 `1 / dppx` 。

Chrome不支持上的子像素值 `border` 。safari不支持上的子像素值 `box-shadow` 。Firefox支持两者的子像素值。

鼠标光标渐变跟踪

渐变跟随鼠标光标的悬停效果。

```
<button class="mouse-cursor-gradient-tracking">
  <span>Hover me</span>
</button>
```

```
.mouse-cursor-gradient-tracking {
  position: relative;
  background: #7983ff;
  padding: 0.5rem 1rem;
  font-size: 1.2rem;
  border: none;
  color: white;
  cursor: pointer;
  outline: none;
  overflow: hidden;
}
.mouse-cursor-gradient-tracking span {
```

```
  position: relative;
}
.mouse-cursor-gradient-tracking::before {
  --size: 0;
  content: '';
  position: absolute;
  left: var(--x);
  top: var(--y);
  width: var(--size);
  height: var(--size);
  background: radial-gradient(circle closest-side, pink, transparent);
  transform: translate(-50%, -50%);
  transition: width 0.2s ease, height 0.2s ease;
}
.mouse-cursor-gradient-tracking:hover::before {
  --size: 200px;
}
```

```
var btn = document.querySelector('.mouse-cursor-gradient-tracking')
btn.onmousemove = function(e) {
  var x = e.pageX - btn.offsetLeft
  var y = e.pageY - btn.offsetTop
  btn.style.setProperty('--x', x + 'px')
  btn.style.setProperty('--y', y + 'px')
}
```

注意

如果元素的父级具有定位上下文(`position: relative`)，您还需要减去它的偏移量。

```
var x = e.pageX - btn.offsetLeft - btn.offsetParent.offsetLeft
var y = e.pageY - btn.offsetTop - btn.offsetParent.offsetTop
```

:非选择器

这`:not` psuedo选择器用于设置一组元素的样式，而保留最后一个(或指定的)元素未设置样式。

```
<ul class="css-not-selector-shortcut">
  <li>One</li>
  <li>Two</li>
  <li>Three</li>
  <li>Four</li>
  <li>Five</li>
</ul>
```

```css
.css-not-selector-shortcut {
  display: flex;
}
li {
  list-style-type: none;
  margin: 0;
  padding: 0 0.75rem;
}
li:not(:last-child) {
  border-right: 2px solid #d2d5e4;
}
```

`li:not(:last-child)` 指定样式应应用于所有 `li` 元素除外 `:last-child` 。

溢出滚动渐变

向溢出元素添加渐变以更好地指示有更多内容需要滚动。

```html
<div class="overflow-scroll-gradient">
  <div class="overflow-scroll-gradient__scroller">
    Content to be scrolled
  </div>
</div>
```

```css
.overflow-scroll-gradient {
  position: relative;
}
.overflow-scroll-gradient::after {
  content: '';
  position: absolute;
  bottom: 0;
  width: 240px;
  height: 25px;
  background: linear-gradient(
    rgba(255, 255, 255, 0.001),
    white
  ); /* transparent keyword is broken in Safari */
  pointer-events: none;
}
.overflow-scroll-gradient__scroller {
  overflow-y: scroll;
  background: white;
  width: 240px;
  height: 200px;
  padding: 15px 0;
  line-height: 1.2;
  text-align: center;
}
```

说明

1. `position: relative` 在父级上为伪元素建立笛卡尔定位上下文。
2. `::after` 定义伪元素。
3. `background-image: linear-gradient(...)` 添加从透明渐变为白色(从上到下)的线性渐变。
4. `position: absolute` 从文档流中取出伪元素，并将其相对于父元素定位。

5. `width: 240px` 匹配滚动元素的大小(它是具有伪元素的父元素的子元素)。
6. `height: 25px` 是衰落梯度伪元素的高度，应当保持相对较小。
7. `bottom: 0` 将伪元素定位在父元素的底部。
8. `pointer-events: none` 指定伪元素不能是鼠标事件的目标，允许其后面的文本仍然是可选/交互式的。

形状分离器

使用SVG形状分隔两个不同的块，与标准水平分隔相比，可以创建更有趣的视觉外观。

```
<div class="shape-separator"></div>
```

```
.shape-separator {
  position: relative;
  height: 48px;
  background: #333;
}
.shape-separator::after {
  content: '';
  background-image:
url();
  position: absolute;
  width: 100%;
  height: 24px;
  bottom: 0;
}
```

说明

1. `position: relative` 在元素上为伪元素建立笛卡尔定位上下文。
2. `::after` 定义伪元素。
3. `background-image: url(...)` 添加SVG形状(base64格式的24 x24同级变淡形)作为伪元素的背景图像，默认情况下重复。它必须与要分隔的块颜色相同。
4. `position: absolute` 从文档流中取出伪元素，并将其相对于父元素定位。
5. `width: 100%` 确保元素拉伸其父元素的整个宽度。
6. `height: 24px` 与形状高度相同。
7. `bottom: 0` 将伪元素定位在父元素的底部。

使用纯CSS创建三角形形状

```
<div class="triangle"></div>
```

```
.triangle {
  width: 0;
  height: 0;
  border-top: 20px solid #333;
  border-left: 20px solid transparent;
  border-right: 20px solid transparent;
}
```

说明

边框的颜色是三角形的颜色。 三角形尖端点的边对应于相反的`border-*`属性。 例如， border-top的颜色表示箭头指向下方。

`px`值来改变三角形的比例。

动画

弹跳加载中

```
<div class="bouncing-loader">
  <div></div>
  <div></div>
  <div></div>
</div>
```

```
@keyframes bouncing-loader {
  from {
    opacity: 1;
    transform: translateY(0);
  }
  to {
    opacity: 0.1;
    transform: translateY(-1rem);
  }
}
.bouncing-loader {
  display: flex;
  justify-content: center;
}
.bouncing-loader > div {
  width: 1rem;
  height: 1rem;
  margin: 3rem 0.2rem;
  background: #8385aa;
  border-radius: 50%;
  animation: bouncing-loader 0.6s infinite alternate;
}
.bouncing-loader > div:nth-child(2) {
  animation-delay: 0.2s;
}
.bouncing-loader > div:nth-child(3) {
  animation-delay: 0.4s;
}
```

说明

注:`1rem`通常是`16px`。

1. `@keyframes`定义了一个具有两种状态的动画，其中元素更改`opacity`并使用`transform: translateY()`在2D平面上进行`transform: translateY()`。
2. `.bouncing-loader`是弹跳圆圈的父容器，使用`display: flex`和`justify-content: center`将它们放置在中心位置。

3. `.bouncing-loader > div` ，将父级的三个子`div`作为样式。 `div`的宽度和高度为`1rem` ，使用`border-radius: 50%`将它们从正方形变成圆形。
4. `margin: 3rem 0.2rem` 指定每个圆的上边距/下边距为`3rem` 和左/右边距`0.2rem` 以便它们不直接接触对方，给它们一些呼吸空间。
5. `animation`是各种动画属性的缩写属性：使用`animation-name` ， `animation-duration` ，`animation-iteration-count` ， `animation-direction` 。
6. `nth-child(n)` 目标是其父元素的第n个子元素。
7. `animation-delay`分别用于第二个和第三个`div` ，以便每个元素不会同时启动动画。

用于指示内容加载的圆环微调器

```
<div class="donut"></div>
```

```
@keyframes donut-spin {
  0% {
    transform: rotate(0deg);
  }
  100% {
    transform: rotate(360deg);
  }
}
.donut {
  display: inline-block;
  border: 4px solid rgba(0, 0, 0, 0.1);
  border-left-color: #7983ff;
  border-radius: 50%;
  width: 30px;
  height: 30px;
  animation: donut-spin 1.2s linear infinite;
}
```

说明-需要前缀才能获得完全支持

使用半透明的`border` 对于整个元素，除了用作圆环加载指示器的一侧。使用`animation` 旋转元素。

从转换元素的高度 0 到`auto`当其高度未知时。

```
<div class="trigger">
  Hover me to see a height transition.
  <div class="el">content</div>
</div>
```

```
.el {
  transition: max-height 0.5s;
  overflow: hidden;
  max-height: 0;
}
.trigger:hover > .el {
  max-height: var(--max-height);
}
```

```
var el = document.querySelector('.el')
var height = el.scrollHeight
el.style.setProperty('--max-height', height + 'px')
```

说明-会在每个动画帧上引起回流，如果元素下方有大量元素在高度上转换，则回流将会滞后。

CSS

1. `transition: max-height: 0.5s cubic-bezier(...)` 指定更改为 `max-height` 应使用 `ease-out-quint` 定时函数。
2. `overflow: hidden` 防止隐藏元素的内容溢出其容器。
3. `max-height: 0` 指定元素最初没有高度。
4. `.target:hover > .el` 指定当父项悬停在上方时，以子项为目标 `.el` 并使用 `--max-height` 由JavaScript定义的变量。

JavaScript

1. `el.scrollHeight` 是包含溢出的元素的高度，它将根据元素的内容动态更改。
2. `el.style.setProperty(...)` 设置 `--max-height` 用于指定 `max-height` 目标悬停在其上的元素的，允许它从0平滑过渡到auto。

悬停下划线动画

当文本悬停在上面时创建动画下划线效果。

```
<p class="hover-underline-animation">Hover this text to see the effect!</p>
```

```
.hover-underline-animation {
  display: inline-block;
  position: relative;
  color: #0087ca;
}
.hover-underline-animation::after {
  content: '';
  position: absolute;
  width: 100%;
  transform: scaleX(0);
  height: 2px;
  bottom: 0;
  left: 0;
  background-color: #0087ca;
  transform-origin: bottom right;
  transition: transform 0.25s ease-out;
}
.hover-underline-animation:hover::after {
  transform: scaleX(1);
  transform-origin: bottom left;
}
```

说明

1. `display: inline-block` 使块p `p` 成为 一 `inline-block` 以防止下划线跨越整个父级宽度而不仅仅是内容(文本)。
2. `position: relative` 在元素上为伪元素建立笛卡尔定位上下文。
3. `::after` 定义伪元素。

4. `position: absolute` 从文档流中取出伪元素，并将其相对于父元素定位。
5. `width: 100%` 确保伪元素跨越文本块的整个宽度。
6. `transform: scaleX(0)` 最初将伪元素缩放为0，使其没有宽度且不可见。
7. `bottom: 0` 和 `left: 0` 将其放置在块的左下方。
8. `transition: transform 0.25s ease-out` 意味着 `transform` 变化将通过 `ease-out` 时间功能在0.25秒内过渡。
9. `transform-origin: bottom right` 表示变换锚点位于块的右下方。
10. `:hover::after` 然后使用 `scaleX(1)` 将宽度转换为100%，然后将 `transform-origin` 更改为 `bottom left` 以便定位点反转，从而允许其在悬停时转换到另一个方向。

鼠标光标渐变跟踪

```html
<button class="mouse-cursor-gradient-tracking">
  <span>Hover me</span>
</button>
```

```css
.mouse-cursor-gradient-tracking {
  position: relative;
  background: #7983ff;
  padding: 0.5rem 1rem;
  font-size: 1.2rem;
  border: none;
  color: white;
  cursor: pointer;
  outline: none;
  overflow: hidden;
}
.mouse-cursor-gradient-tracking span {
  position: relative;
}
.mouse-cursor-gradient-tracking::before {
  --size: 0;
  content: '';
  position: absolute;
  left: var(--x);
  top: var(--y);
  width: var(--size);
  height: var(--size);
  background: radial-gradient(circle closest-side, pink, transparent);
  transform: translate(-50%, -50%);
  transition: width 0.2s ease, height 0.2s ease;
}
.mouse-cursor-gradient-tracking:hover::before {
  --size: 200px;
}
```

```js
var btn = document.querySelector('.mouse-cursor-gradient-tracking')
btn.onmousemove = function(e) {
  var x = e.pageX - btn.offsetLeft
  var y = e.pageY - btn.offsetTop
  btn.style.setProperty('--x', x + 'px')
  btn.style.setProperty('--y', y + 'px')
}
```

注意：如果元素的父级具有定位上下文(`position: relative`)，您还需要减去它的偏移量。

```
var x = e.pageX - btn.offsetLeft - btn.offsetParent.offsetLeft
var y = e.pageY - btn.offsetTop - btn.offsetParent.offsetTop
```

悬停时显示交互式弹出菜单

```
<div class="reference">
  <div class="popout-menu">
    Popout menu
  </div>
</div>
```

```
.reference {
  position: relative;
}
.popout-menu {
  position: absolute;
  visibility: hidden;
  left: 100%;
}
.reference:hover > .popout-menu {
  visibility: visible;
}
```

说明

1. `position: relative` 在参考父项上为其子项建立笛卡尔定位上下文。
2. `position: absolute` 将弹出式菜单从文档流中移出，并将其与父级相关。
3. `left: 100%` 将弹出式菜单的左侧宽度从其父宽度的`left: 100%`移开。
4. `visibility: hidden` 最初隐藏弹出菜单并允许转换(与`display: none`)。
5. `.reference:hover > .popout-menu` 意味着当`.reference` 悬停在上方时，选择具有`.popout-menu`类的直接子项并将它们的 `visibility` 更改为`visible` ，这将显示弹出窗口。

淡出悬停项目的同级

```
<div class="sibling-fade">
  <span>Item 1</span>
  <span>Item 2</span>
  <span>Item 3</span>
  <span>Item 4</span>
  <span>Item 5</span>
  <span>Item 6</span>
</div>
```

```
span {
  padding: 0 1rem;
  transition: opacity 0.2s;
}
.sibling-fade:hover span:not(:hover) {
  opacity: 0.5;
}
```

说明

1. `transition: opacity 0.2s` 指定不透明度的更改将在0.2秒内转换。
2. `.sibling-fade:hover span:not(:hover)` 指定当父项被徘徊时，选择当前没有被徘徊的任何 `span` 子项并将其不透明度更改为 `0.5` 。

其他

自定义变量

CSS变量，其中包含要在整个文档中重用的特定值。

```
<p class="custom-variables">CSS is awesome!</p>
```

```
:root {
  --some-color: #da7800;
  --some-keyword: italic;
  --some-size: 1.25em;
  --some-complex-value: 1px 1px 2px whitesmoke, 0 0 1em slategray, 0 0 0.2em
slategray;
}
.custom-variables {
  color: var(--some-color);
  font-size: var(--some-size);
  font-style: var(--some-keyword);
  text-shadow: var(--some-complex-value);
}
```

说明

变量是在 `:root` 与表示文档的树的根元素匹配的CSS伪类。如果在块中定义变量，则变量的作用域也可以限定为选择器。

使用宣告变数 `--variable-name:` 。

使用在整个文档中重用变量 `var(--variable-name)` 功能。