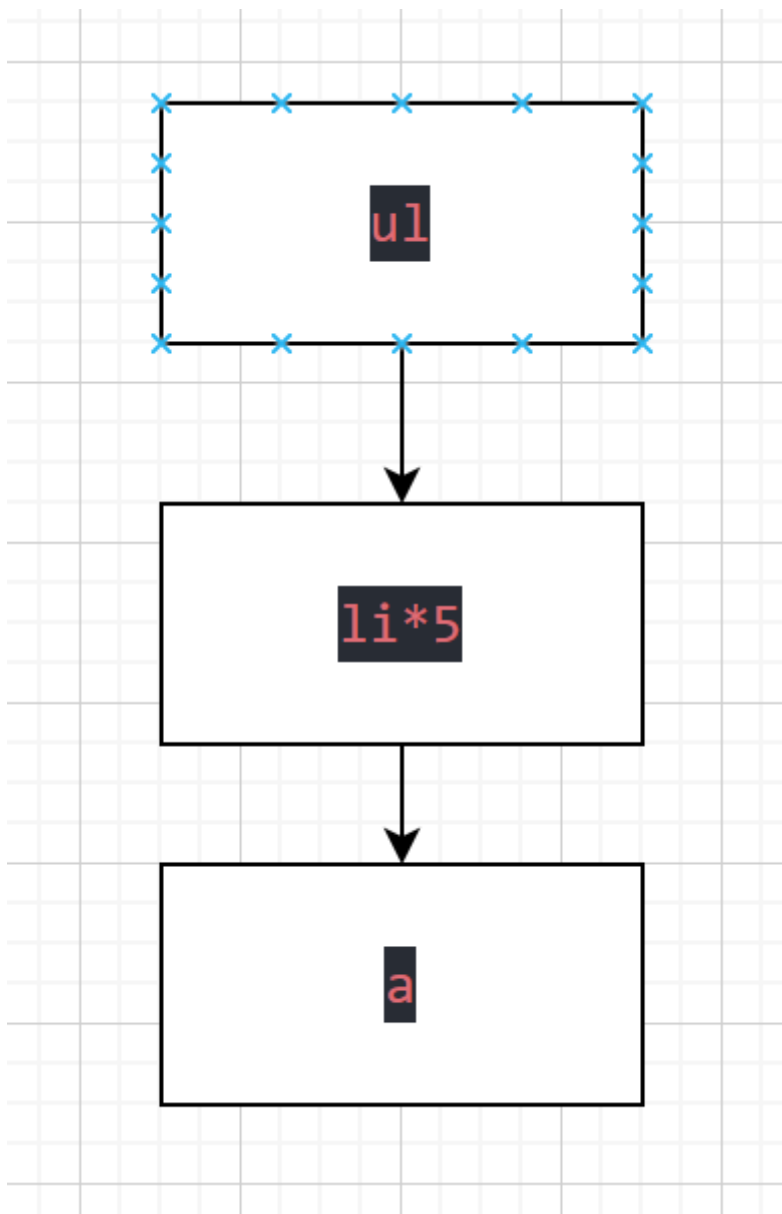



菜单悬停

一.html结构



```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@700&family=Sacramento&display=swap" rel="stylesheet">
```

- 我们注意到link rel = 'preconnect'

preconnect 	向浏览器提供提示，建议浏览器提前打开与链接网站的连接，而不会泄露任何私人信息或下载任何内容，以便在跟随链接时可以更快地获取链接内容。	<code><link></code>	<code><a></code> , <code><area></code>
--	--	---------------------------	---

- 而crossorigin

crossorigin

此枚举属性指定在加载相关资源时是否必须使用 CORS. [启用 CORS 的图片](#) 可以在 [<canvas>](#) 元素中重复使用, 并避免其被污染. 可取的值如下:

"anonymous"

会发起一个跨域请求 (即包含 `Origin:` HTTP 头). 但不会发送任何认证信息 (即不发送 cookie, X.509 证书和 HTTP 基本认证信息). 如果服务器没有给出源站凭证 (不设置 `Access-Control-Allow-Origin:` HTTP 头), 资源就会被污染并限制使用.

"use-credentials"

会发起一个带有认证信息 (发送 cookie, X.509 证书和 HTTP 基本认证信息) 的跨域请求 (即包含 `Origin:` HTTP 头). 如果服务器没有给出源站凭证 (不设置 `Access-Control-Allow-Origin:` HTTP 头), 资源就会被污染并限制使用. 当不设置此属性时, 资源将会不使用 CORS 加载 (即不发送 `Origin:` HTTP 头), 这将阻止其在 [<canvas>](#) 元素中进行使用. 若设置了非法的值, 则视为使用 **anonymous**. 前往 [CORS settings attributes](#) 获取更多信息.

```
<li style="--clr: #81ced;"><a href="#" data-text="Home">Home</a></li>
```

- 我们注意到在html中就可以设置格式, 这里相当于给此处格式设置了一个表示颜色的变量, 其作用域仅在这个li中
- a标签中的data-text属性是何意思呢?

二.style格式

1.全局设置

```
@import url('https://fonts.googleapis.com/css2?
family=Poppins:wght@200;300;400;500;600;700;800;900&display=swap');
@import url('https://fonts.googleapis.com/css2?
family=Poppins:wght@700&family=Sacramento&display=swap');
*
{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}
body
{
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  overflow: hidden;
  background: #fff;
}
```

- 导入需要用到的网址, 实际上要用的字体是显示不出来的
- font-family

CSS 属性 `font-family` 允许您通过给定一个有先后顺序的，由字体名或者字体族名组成的列表来为选定的元素设置字体。

`<family-name>`

一个字体族的名字。例如"Times" 和 "Helvetica" 都是字体族名。字体族名可以包含空格，但包含空格时应该用引号。

`<generic-name>`

通用字体族名是一种备选机制，用于在指定的字体不可用时给出较好的字体。通用字体族名都是关键字，所以不可以加引号。在列表的末尾应该至少有一个通用字体族名。以下是该属性可能的取值以及他们的定义。

`sans-serif`

无衬线字体，即笔画结尾是平滑的字体。例如， "Open Sans", "Fira Sans", "Lucida Sans", "Lucida Sans Unicode", "Trebuchet MS", "Liberation Sans", "Nimbus Sans L", sans-serif.

通用字体族即备选字体

2.一般情况下格式

```
ul
{
    position: relative;
}
ul li
{
    list-style: none;
    text-align: center;
}
ul li a
{
    color: #333;
    text-decoration: none;
    font-size: 2em;
    padding: 5px 20px;
    display: inline-flex;
    font-weight: 300;
    letter-spacing: 0.1em;
    text-transform: uppercase;
}
```

- text-decoration可以通过具体的值给字体设置属性（比如下划线等），若值为none那么意思就是没有什么设置
- display: inline-flex
外部表现: inline
内部表现:flex

inline

该元素生成一个或多个内联元素盒，它们之前或者之后并不会产生换行。在正常的流中，如果有空间，下一个元素将会在同一行上。

备注： 浏览器支持双值语法，当仅发现外部值时，例如当指定 `display: block` 或 `display: inline`，将其内部值设置为 `flow`。这种行为是预期的；例如，如果你指定一个元素是块元素，你将期望该元素的子元素将同块和内联元素一样参与正常的流布局。

flex

该元素的行为类似块级元素并且根据[弹性盒模型](#)布局它的内容。

- display形式语法

```
display =  
  [ <display-outside> || <display-inside> ]  
  <display-listitem>  
  <display-internal>  
  <display-box>  
  <display-legacy>  
  <display-outside> || [ <display-inside> | math ]  
  
<display-outside> =  
  block |  
  inline |  
  run-in  
  
<display-inside> =  
  flow |  
  flow-root |  
  table |  
  flex |  
  grid |  
  ruby  
  
<display-listitem> =  
  <display-outside>? &&  
  [ flow | flow-root ]? &&  
  list-item  
  
<display-internal> =  
  table-row-group |  
  table-header-group |  
  table-footer-group |  
  table-row |  
  table-cell |  
  table-column-group |  
  table-column |  
  table-caption |  
  ruby-base |
```

```

ruby-text      |
ruby-base-container |
ruby-text-container

<display-box> =
  contents |
  none

<display-legacy> =
  inline-block |
  inline-table |
  inline-flex   |
  inline-grid

```

- inline-flex

inline-flex

元素的行为类似于内联元素并且它的内容根据弹性盒模型布局。

它等同于 inline flex。

3. 悬浮格式

```

ul li:hover a
{
  background: #333;
  color: var(--clr);
  font-weight: 500;
}
ul:hover > li:not(:hover){
  opacity: 0;
}

```

- 当鼠标在某个具体的li上，背景颜色改变，字体的颜色改变
- 当鼠标悬浮在ul上，没有被悬浮的li就要隐去

4. 伪元素

```

ul li a::before
{
  opacity: 0;
  content: '';
  font-size: 5em;
  color: #222;
  font-weight: 400;
  text-transform: initial;
  letter-spacing: 500px;
  font-family: 'Poppins', sans-serif;
  font-family: 'Sacramento', cursive;

  position: absolute;
  top: 50%;
}

```

```

    left: 50%;
    transform: translate(-50%,-50%);
    z-index: -1;
    display: flex;
    justify-content: center;
    align-items: center;

    transition: letter-spacing 0.5s;
}
ul li a:hover::before
{
    content: attr(data-text);
    opacity: 1;
    background: var(--clr);
    /* left: 50%; */
    width: 250vh;
    height: 250vh;
    letter-spacing: 0;
}

```

- 通过观察我们发现伪元素在字上花了很多功夫，比如字体大小，颜色，粗细，间隔，字形，但我们还知道在未悬浮状态下是没有内容的，但是却在这里进行字体的设置。
- 关于位置的设置，首先让其左上角位于a的中间偏左位置，然后再在此基础上向左向上移动，使得伪元素中心位置和a相同
- 最后渐变的设置，可以看到，渐变不仅可以设置动画，还可以设置属性
- 悬浮状态下改变了内容，字体间隔，元素大小，以及透明度，背景颜色