

Funções



Translated?
Transformed?



O que são?

Funções são blocos de código presentes em qualquer linguagem de programação.

No caso do css também temos funções, a diferença é que não podemos criar as nossas próprias funções (para já).

As funções são blocos que nos permitem realizar operações com os argumentos fornecidos.

```
.example {  
  width: calc(100% / 3);  
}
```

Estrutura

```
.example {  
  width: calc(100% / 2);  
}
```

↑
propriedade

↑
função

↑
Argumento

Estrutura

```
.example {  
  width: calc(100% / 2);  
}
```

↑
propriedade

↑
função

↑
Argumento

```
.example {  
  width: 50%; // retorno  
}
```

Tipos de funções

transform

math

filter

color

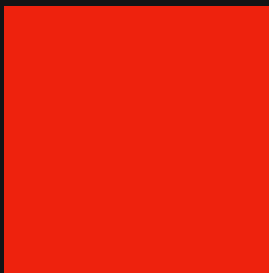
images

counter

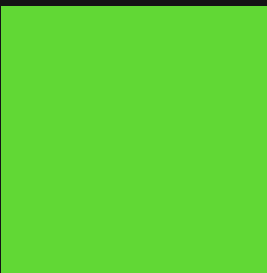
reference

easing

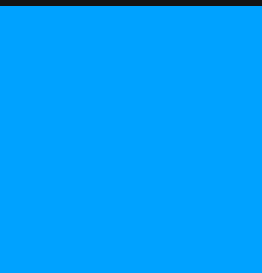
Color



R



G



B

Color



Color

R

255

G

0

B


0


=

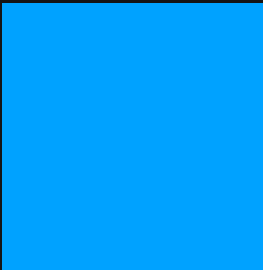
Vermelho

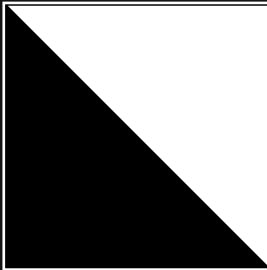
```
.example {  
  background: rgb(255,0,0);  
}
```


Color

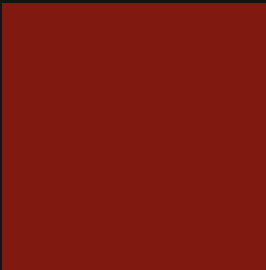








=



R

G

B

A

Vermelho
com
50% de alpha
(opacidade)

255

0

0

0.5

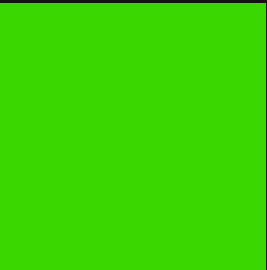
```
.example {  
  background: rgba(255,0,0,0.5);  
}
```

Color



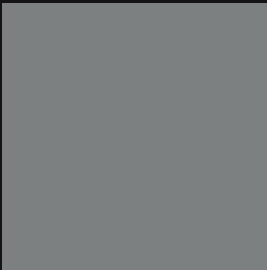
H

0deg



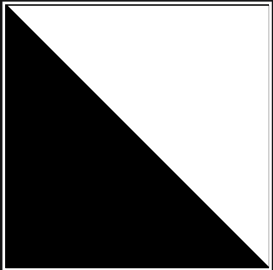
S

100%



L

50%



A

1


=




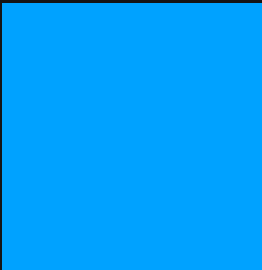
Vermelho

```
.example {  
  background: hsla(0,100%,50%,1);  
}
```


Color







=



R

G

B

Vermelho

FF

00

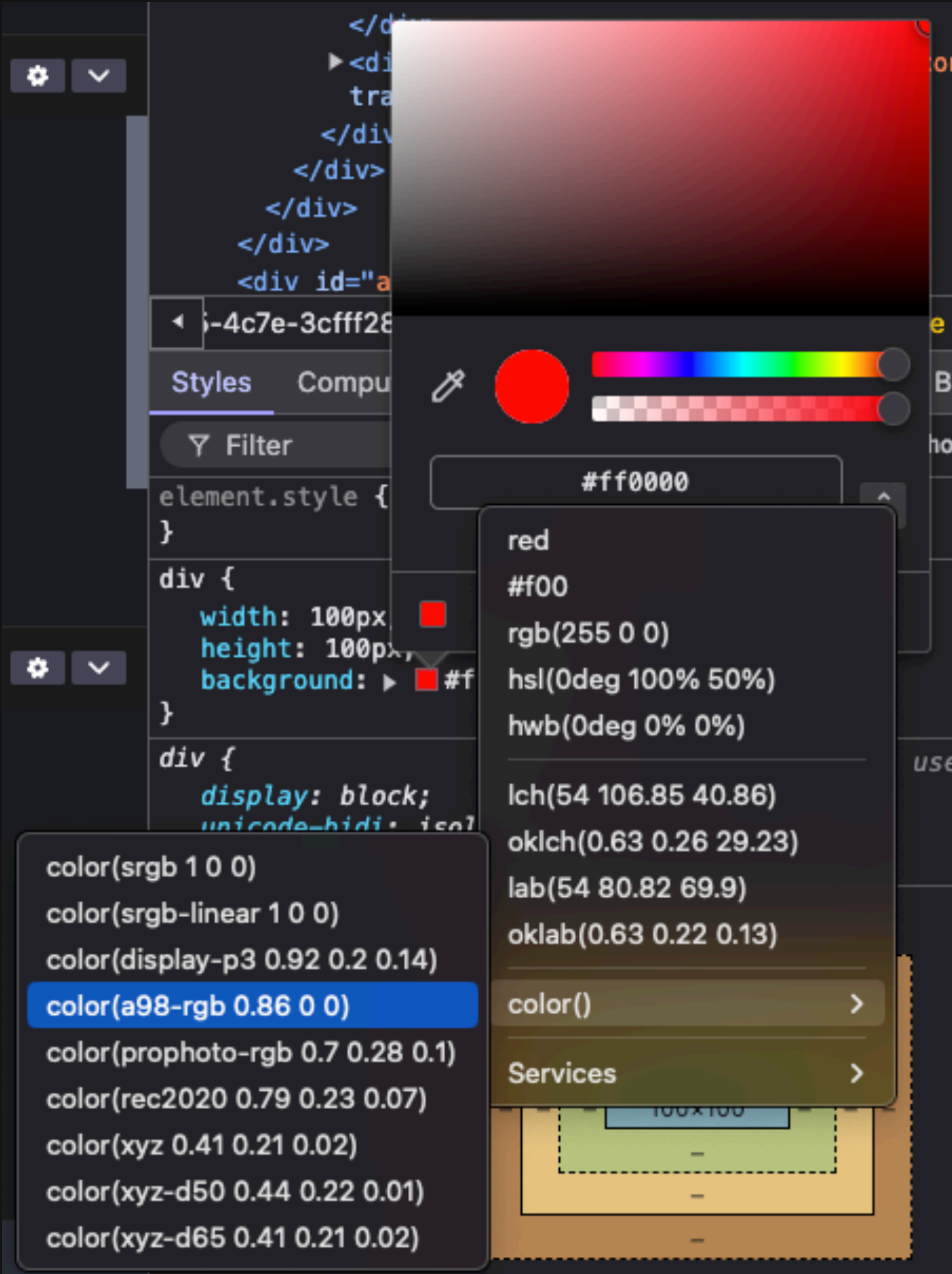
00

```
.example {  
  background: #FF0000;  
}
```

Color

Há várias maneiras de definir o mesmo resultado.

A escolha da ferramenta correcta depende da flexibilidade, a leitura e de caso para caso.



Filter

Para os familiarizados com a edição de imagem, não é muito diferente.

A utilização de filters são formas de manipulação de conteúdo visual.

Filter

filter: blur(5px)



filter: brightness(200%);



filter: contrast(200%);



filter: grayscale(100%);



filter: hue-rotate(90deg);



filter: invert(100%);



filter: opacity(30%);



filter: saturate(8);



filter: sepia(100%);



filter: contrast(200%) brightness(150%);



Expressões matemáticas

Aritmética básica - soma, subtração, divisão, multiplicação

```
calc(100% / 2) // 50%
```

```
calc(100px / 2) // 50px
```

```
calc(100% * 2) // 200%
```

```
calc(100px + 20px) // 120px
```

```
calc(100px - 20px) // 80px
```


Expressões matemáticas

Aritmética básica - soma, subtração, divisão, multiplicação

```
calc(100% / 2) // 50%
```

```
calc(100px / 2) // 50px
```

```
calc(100% * 2) // 200%
```

```
calc(100px + 20px) // 120px
```

```
calc(100px - 20px) // 80px
```

Operações precisam de contexto

```
calc(100 / 2) // ???
```

```
calc(100 * 2) // ???
```

```
calc(100 - 2) // ???
```

```
calc(100 + 2) // ???
```

```
calc(100<% | px | pt> / 2)
```

```
// 0 resultado deve ser quantificável
```


Expressões matemáticas

Comparação

`clamp(50px, 100vw, 150px)` // 100% da width viewport, desde que entra 50px e 150px

`min(50vw, 300px)` // 50% da width do viewport, desde que até ao máximo de 300px

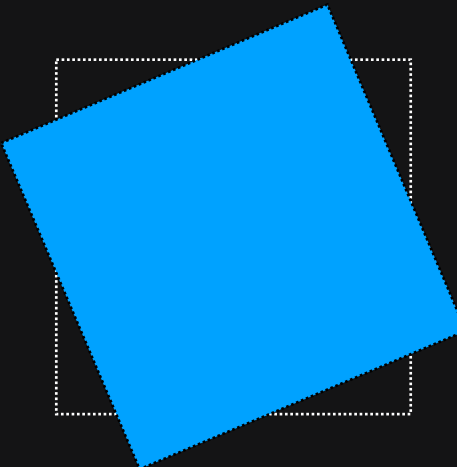
`max(50vw, 300px)` // 50% da width do viewport, desde que até ao mínimo de 300px

Transform

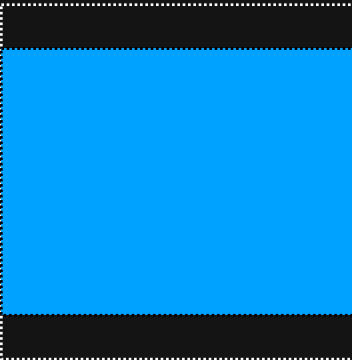
A propriedade transform da-nos acesso a funções de manipulação gráfico do elemento.

Transform

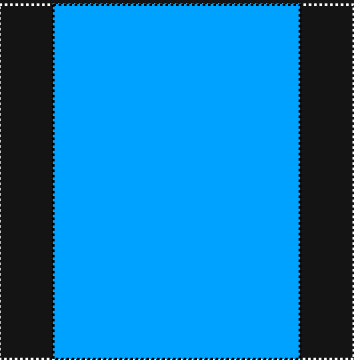
rotate<3d | X | Y | Z>



transform: rotateX(45deg)



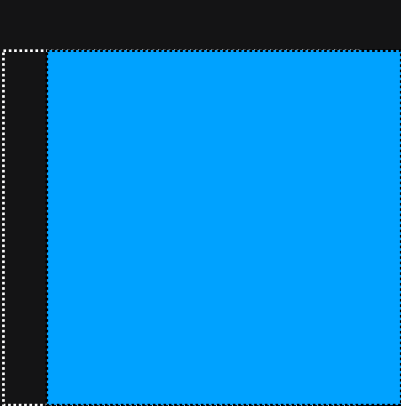
transform: rotateZ(45deg)



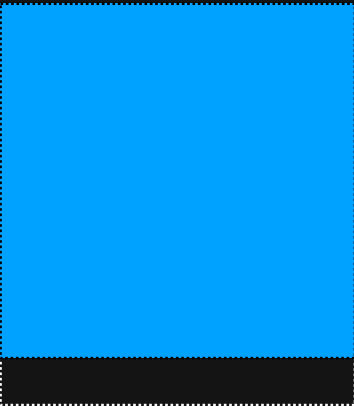
transform: rotateY(45deg)

Transform

translate<3d | X | Y | Z>



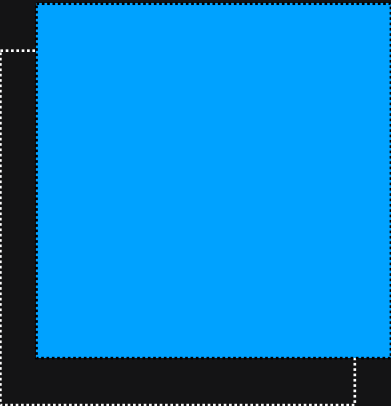
transform: translateX(10px)



transform: translateY(10px)



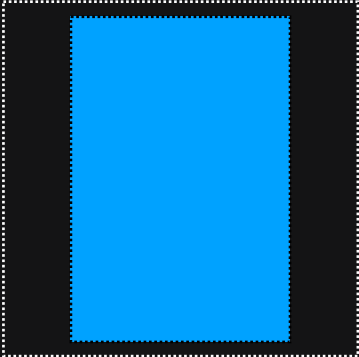
transform: translateZ(10px)



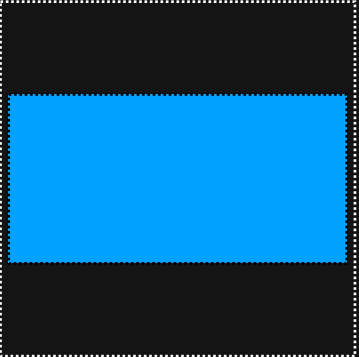
transform: translate(10px, 10px, 10px)

Transform

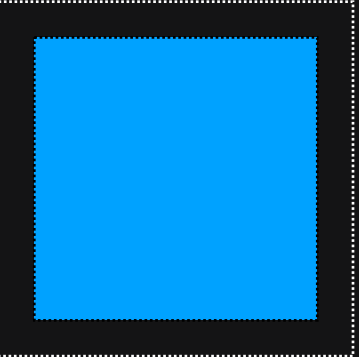
scale<3d | X | Y | Z>



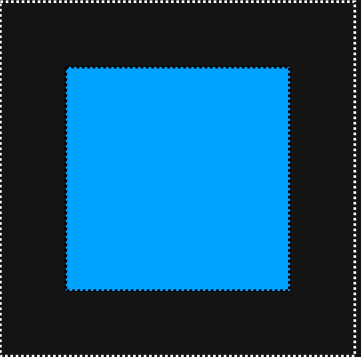
transform: scaleX(0.5)



transform: scaleY(0.5)



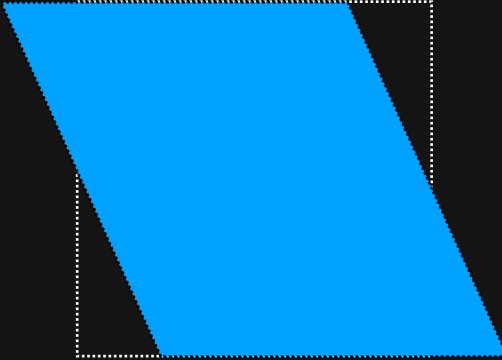
transform: scaleZ(0.5)



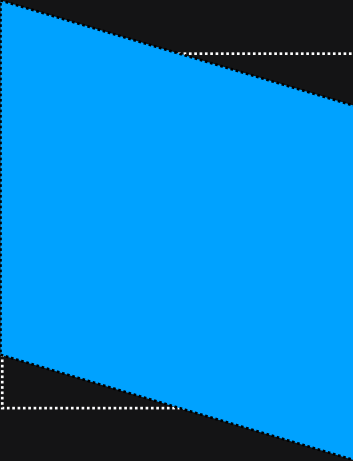
transform: scale(0.5)

Transform

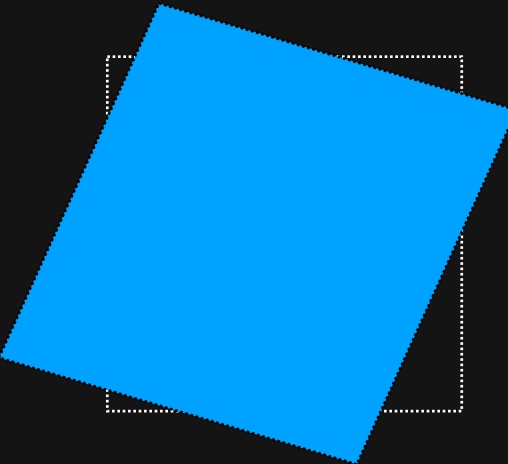
skew<3d | X | Y | Z>



transform: skewX(10deg)



transform: skewY(10deg)



transform: skew(10deg, 10deg)

Transform

Outros métodos de transformação são mais claros em elementos 3d.

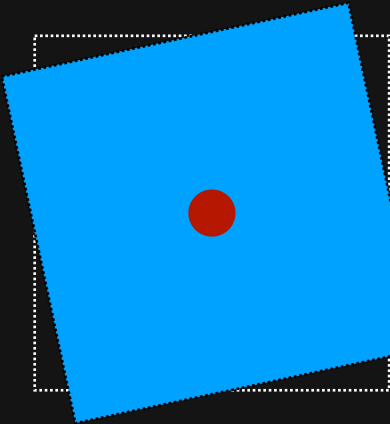
matrix

matrix3d

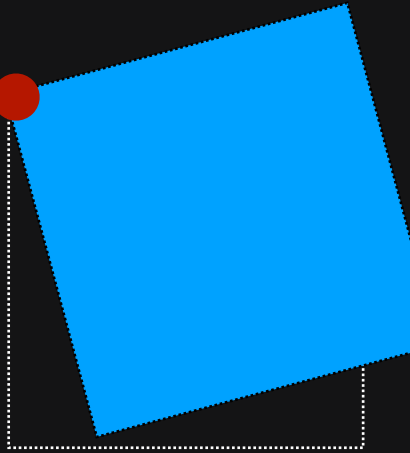
perspective

Transform

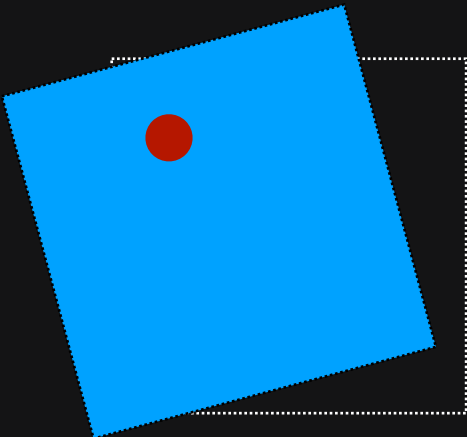
Outros atributos de transform depende da origem da transformação. Por default a ancora da transform é o meio do elemento em sí, mas isto pode ser alterado.



`transform-origin:center;`



`transform-origin:top left;`



`transform-origin:20% 20%;`

Counters

São variáveis que ficam em memória e podem ser incrementadas juntamente com os elementos e funcionar como index.

Counters

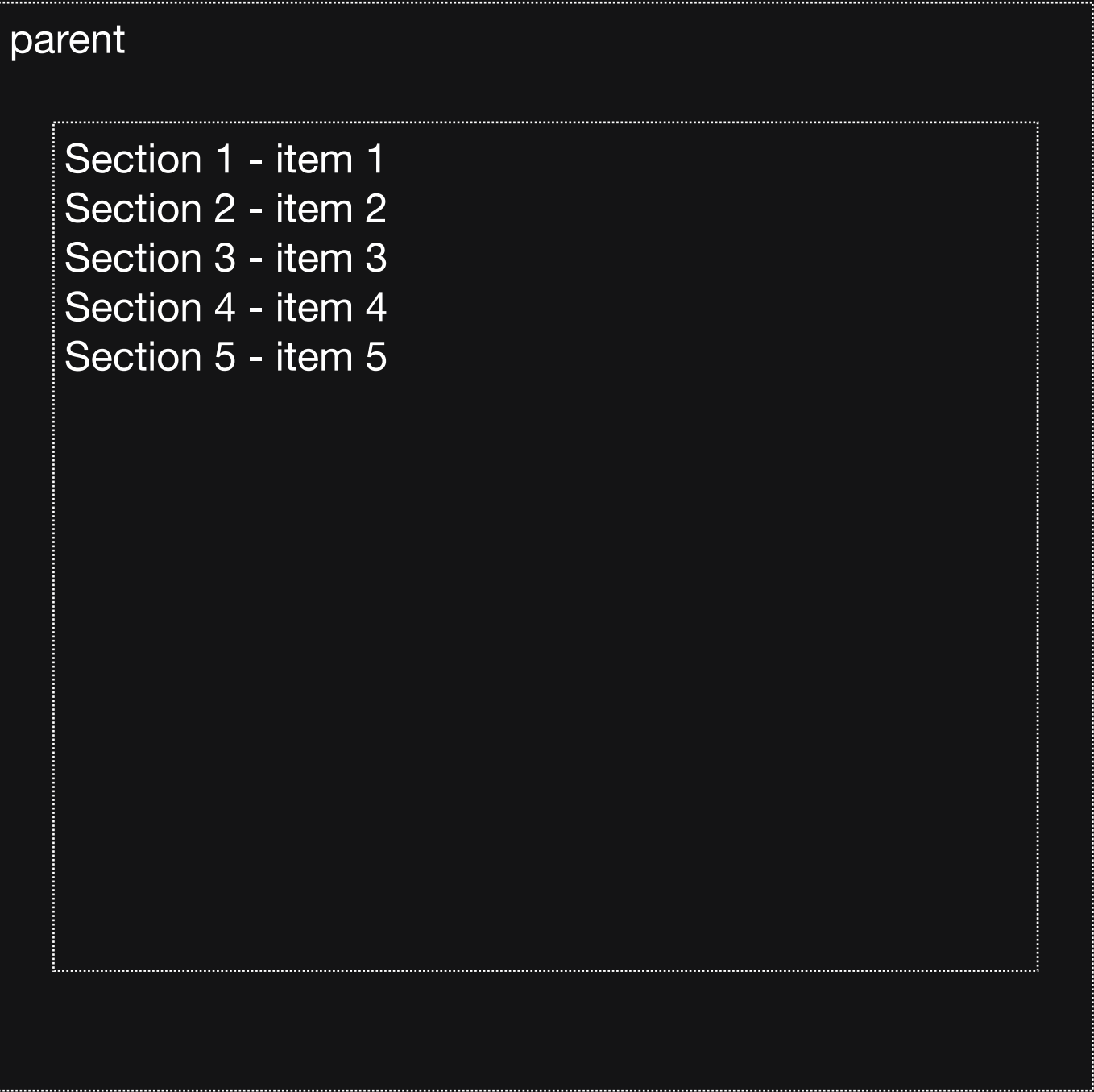
São variáveis que ficam em memória e podem ser incrementadas juntamente com os elementos e funcionar como index.

```
body {
  counter-reset: section;
}

li::before {
  counter-increment: section;
  content: "Section " counter(section) "- ";
}
```

Counters

São variáveis que ficam em memória e podem ser incrementadas juntamente com os elementos e funcionar como index.



Imagens

São métodos usados para execução de imagens e / ou atribuição de gradientes, entre outros.

Imagens



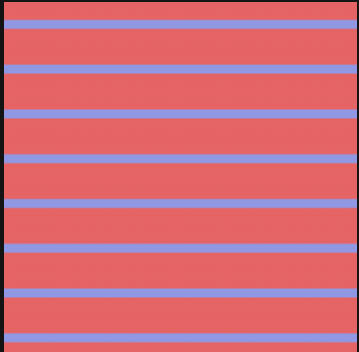
background:
linear-gradient(#e66465, #9198e5);



background:
radial-gradient(#e66465, #9198e5);



background: conic-gradient(red, orange, yellow, green,
blue);



background: repeating-radial-gradient(#e66465, #9198e5 20%);

Referência

As seguintes funções são usadas como valor de propriedades para referenciar um valor definido noutro local

```
attr() // Usa atributos definidos no html, como title, etc.  
  
var() // Acede a parâmetros guardados normalmente no selector :root
```