CSS

Funções
Animações

Funções

Permitem-nos agregar funcionalidade ao nosso css.

Funções

Existem funções com argumentos específicos e outras com alguma flexibilidade.


```
.selector {
  background-image: url(”/assets/bg.jpeg”);
}
```

Funções

Existem várias funções que cobrem os mais variados propósitos.

Custom Properties	Calculations	Color	Pseudo Class Selector	Animation	Transform
Filter	Comparison	Logical Combinators	Gradient	Grid	Shape

Custom Properties

Podemos declarar propriedades no início do documento.

Podemos depois re-utilizar estas propriedades ao longo do nosso css.

```
:root {
  —size: 1200px;
}

html {
  width: var(—size);
}
```

```
:root {
  —size: 1200px;
}

html {
  width: 1200px
}
```

Cálculo

A função calc() permite utilizar aritmética simples para retornar valores calculados.

```
:root {
  —size: 1200px;
}

html {
  width: calc(var(—size) / 2);
}
```

```
:root {
  —size: 1200px;
}

html {
  width: 600px;
}
```

Cálculo

A função calc() tem algumas particularidades que devem ser cumpridas.

```
calc( n + y )
calc( n - y )
calc( n / y )
calc( n * y )
```

```
calc( 1200px + 2 )
calc( 1200px  - 2 )
calc( 1200px  / 2 )
calc( 1200px  * 2 )
```

Cálculo

A função calc() tem algumas particularidades que devem ser cumpridas.

```
calc( n + y )
calc( n - y )
calc( n / y )
calc( n * y )
```

```
calc ( 10px + 2 )
calc ( 10px - 2 )
calc ( 10px / 5px )
calc ( 10px * 5px )
```

Nenhuma destas funções tem um retorno que demonstre um valor existente.

Cálculo

Podemos através do nesting tornar os nossos cálculos mais complexos.

```
:root {
  --width: 1200px;
  --cols: 12;
  --col-width: calc(var(--width) / var(--cols))
  --gap: 12px;
}

.nav {
  width: calc(calc(120px + 12px) / (calc(12 / 4))
}
```

Cores

As funções que trabalham com cores são úteis para a utilização e transformação de cores.

rgb() ou rgba()
hsl() ou hsla()

rgba
red | green | blue | alpha

hsl
hue saturation lightness

```
rgba(0,0,0,.5)
// preto com 50% de transparência

rgb(0,0,0) // preto

hsla(0,0,0,.5)
// preto com 50% de transparência

hsl(0,0,0) // preto
```

Pseudo Class Selectors

Permite-nos interagir com elementos específicos em que, destes elementos existam *n* instâncias no mesmo nível.

```
<ul>
  <li>a</li>
  <li>b</li>
  <li>c</li>
</ul>
```

Pseudo Class Selectors

Permite-nos interagir com elementos específicos em que, destes elementos existam *n* instâncias no mesmo nível.

```
<ul>
  <li>a</li>      n = 1
  <li>b</li>      n = 2
  <li>c</li>      n = 3
  <li>d</li>      n = 4
  <li>e</li>      n = 5
  <li>f</li>      n = 6
</ul>
```

- a
- b
- c
- d
- e
- f

```
li:nth-child(1) {
  color: red;
}
```

Pseudo Class Selectors

Permite-nos interagir com elementos específicos em que, destes elementos existam *n* instâncias no mesmo nível.

```
<ul>
  <li>a</li>      n = 1
  <li>b</li>      n = 2
  <li>c</li>      n = 3
  <li>d</li>      n = 4
  <li>e</li>      n = 5
  <li>f</li>      n = 6
</ul>
```

- a
- b
- c
- d
- e
- f

```
li:nth-child(n+1) {
  color: red;
}
```

Pseudo Class Selectors

Permite-nos interagir com elementos específicos em que, destes elementos existam *n* instâncias no mesmo nível.

```
<ul>
  <li>a</li>      n = 1
  <li>b</li>      n = 2
  <li>c</li>      n = 3
  <li>d</li>      n = 4
  <li>e</li>      n = 5
  <li>f</li>      n = 6
</ul>
```

- a
- b
- c
- d
- e
- f

```
li:nth-child-of-type(odd) {
  color: red;
}
```

Pseudo Class Selectors

Permite-nos interagir com elementos específicos em que, destes elementos existam *n* instâncias no mesmo nível.

```
<ul>
  <li>a</li>      n = 1
  <li>b</li>      n = 2
  <li>c</li>      n = 3
  <li>d</li>      n = 4
  <li>e</li>      n = 5
  <li>f</li>      n = 6
</ul>
```

- a
- b
- c
- d
- e
- f

```
li:nth-last-child() {
  color: red;
}

li:nth-first-child() {
  color: yellow;
}
```


Cristian Oliveira | 2022

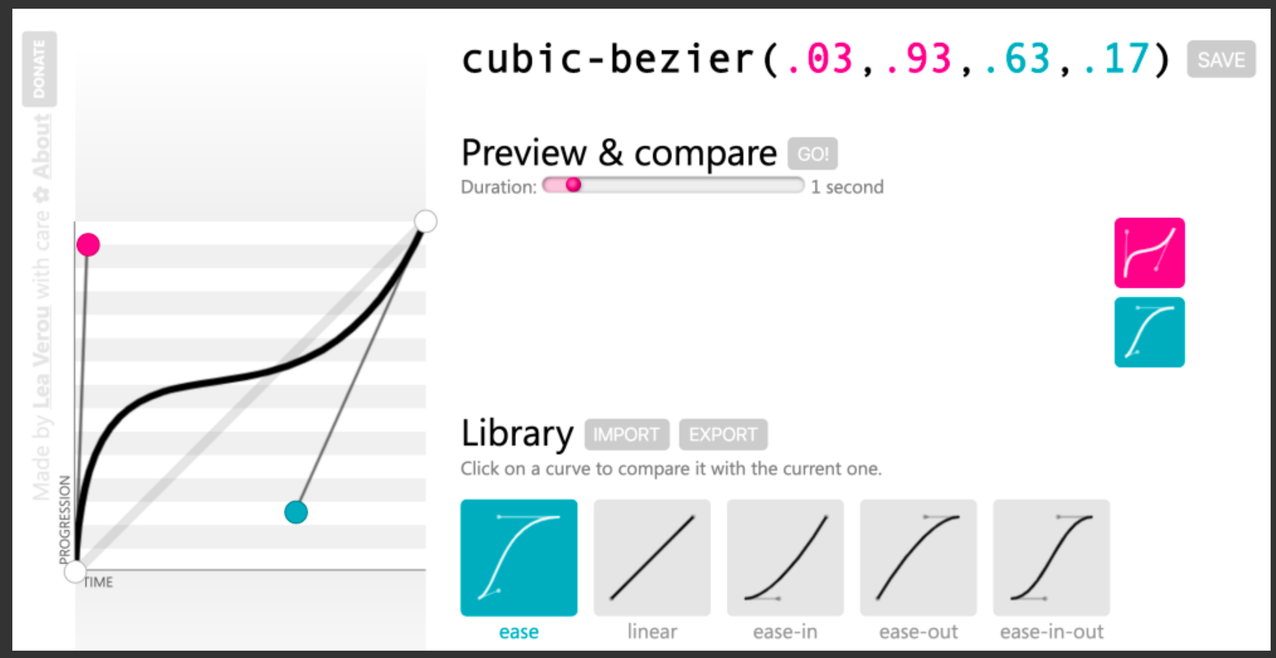
Animation

Animações são importantes para gerir o fluxo na história do utilizador.

Permitem-nos realizar animações e transições mais “sofisticadas” ao nosso css.

cubic-bezier()

- Permite realizar transições controlando o tempo da transição



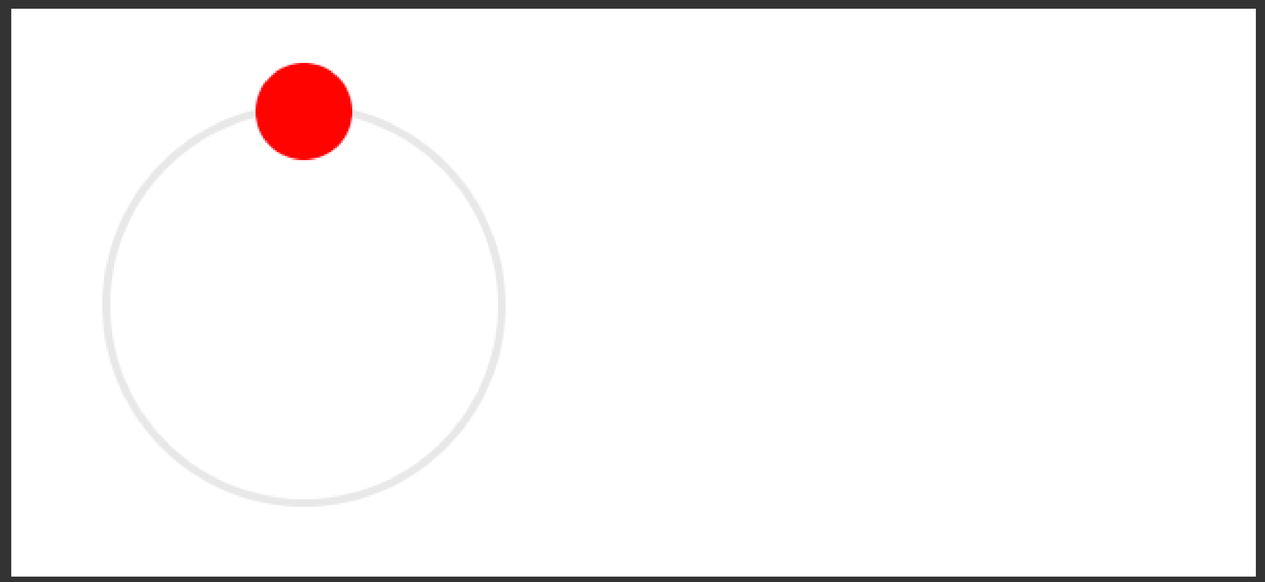
Animation

Animações são importantes para gerir o fluxo na história do utilizador.

Permitem-nos realizar animações e transições mais “sofisticadas” ao nosso css.

path()

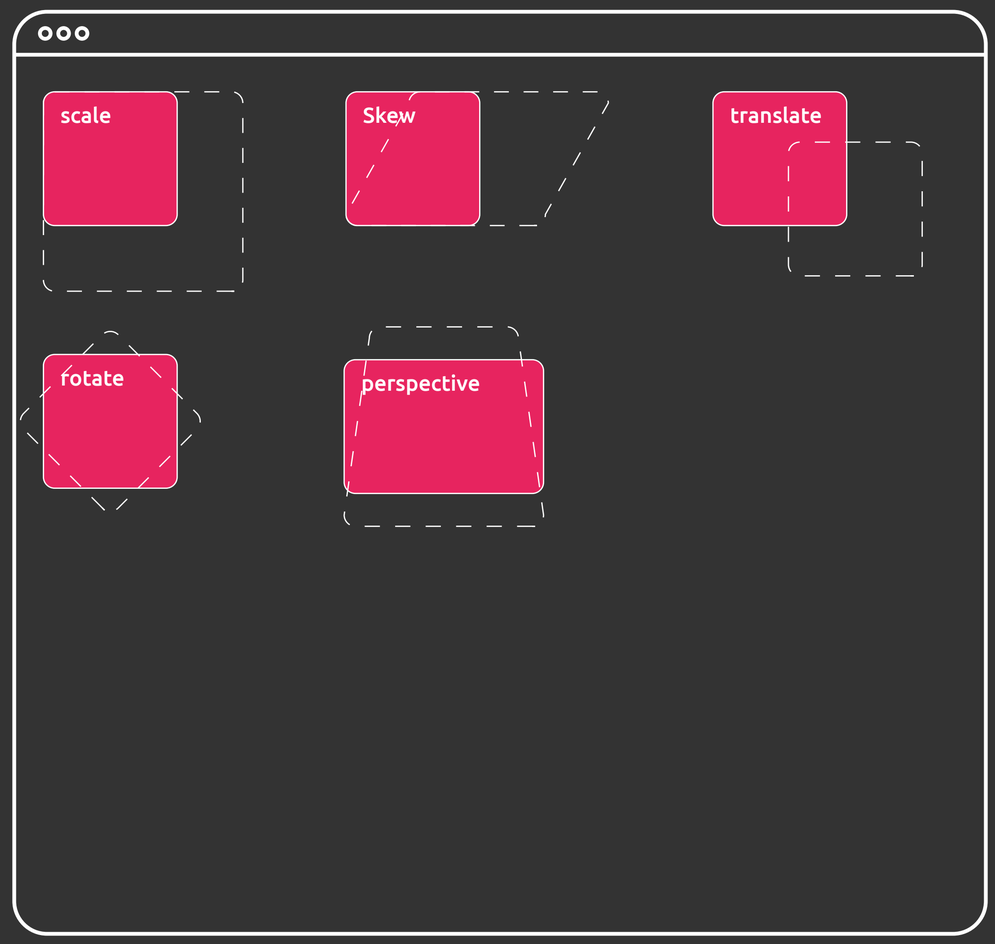
- Permite-nos realizar percursos, entre outras coisas, aquando a transição do nosso elemento.



Transform

Transform ajuda-nos a alterar o elemento e a sua natural forma.

scale( ) || z( ) || y( ) || 3d( )
translate( ) || z( ) || y( ) || 3d( )
perspective()
rotate( ) || z( ) || y( ) || 3d( )



Filter

Tal como em algumas ferramentas de edição de imagem, também no css podemos adicionar alguns tipos de filtros nos nossos elementos.

brightness()
blur()
contrast()
grayscale()
invert()
opacity()
saturate()

Filter

Tal como em algumas ferramentas de edição de imagem, também no css podemos adicionar alguns tipos de filtros nos nossos elementos.

- brightness()
- blur()
- contrast()
- grayscale()
- invert()
- opacity()
- saturate()

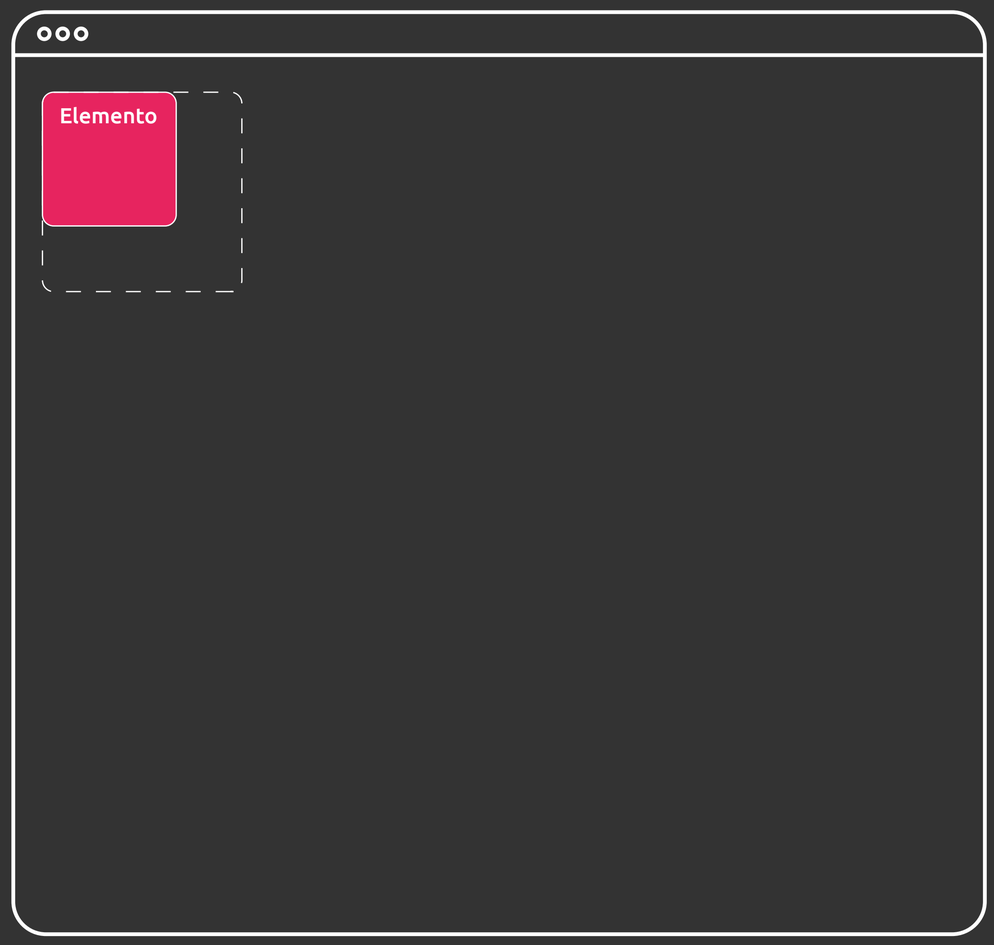
Comparison

As funções de comparação ajudam-nos a flebilizar os nossos elementos para melhor corresponder as exigências do responsive.

clamp()

min()

max()

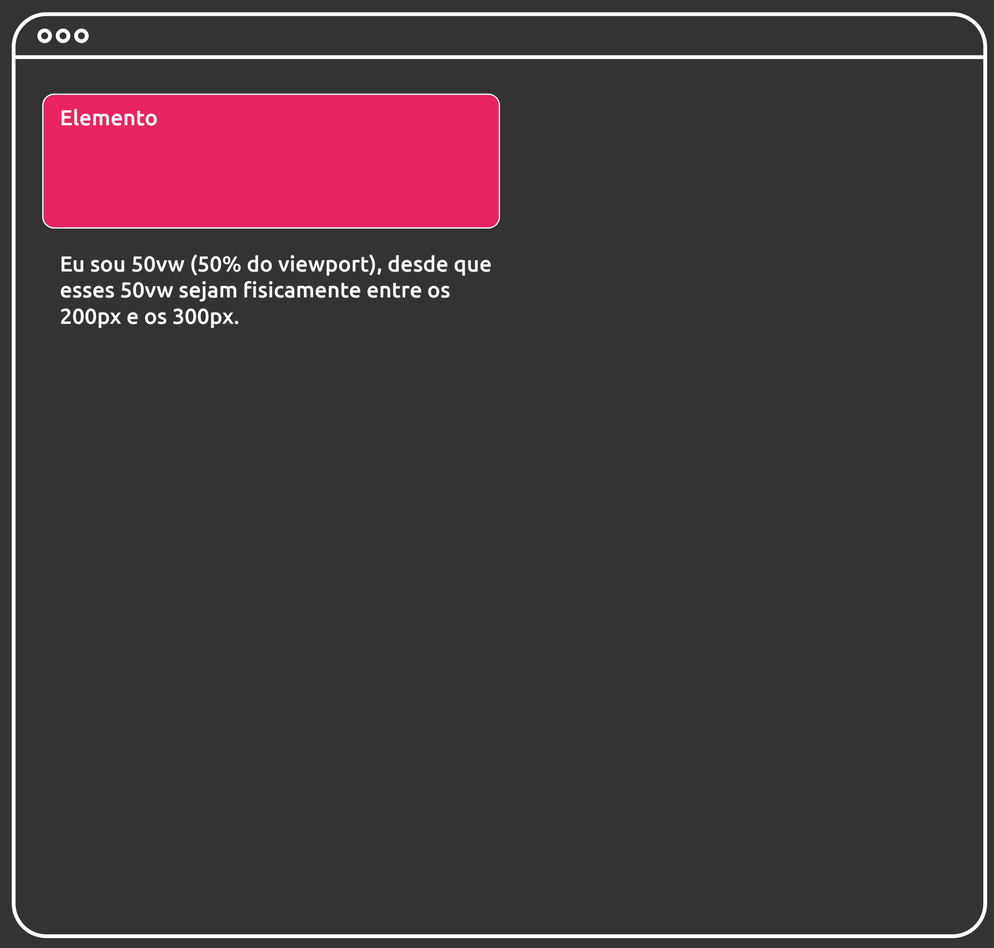


Comparison

As funções de comparação ajudam-nos a flebilizar os nossos elementos para melhor corresponder as exigências do responsive.

clamp(min, desejado, max)

clamp(200px, 50vw, 300px)



Comparison

As funções de comparação ajudam-nos a flebilizar os nossos elementos para melhor corresponder as exigências do responsive.

max(a, b)

max(100px, 50%)

Logical Operators

Com os operadores lógicos podemos criar agrupamentos em selectores de forma a interagir com vários elementos.

:is()

:where()

```
section h1, section h2, section h3, section h4, section h5,
section h6, article h1, article h2, article h3, article h4,
article h5, article h6, aside h1, aside h2, aside h3, aside
h4, aside h5, aside h6, nav h1, nav h2, nav h3, nav h4, nav
h5, nav h6 {
  color: #BADA55;
}
```

Logical Operators

Com os operadores lógicos podemos criar agrupamentos em selectores de forma a interagir com vários elementos.

:is()

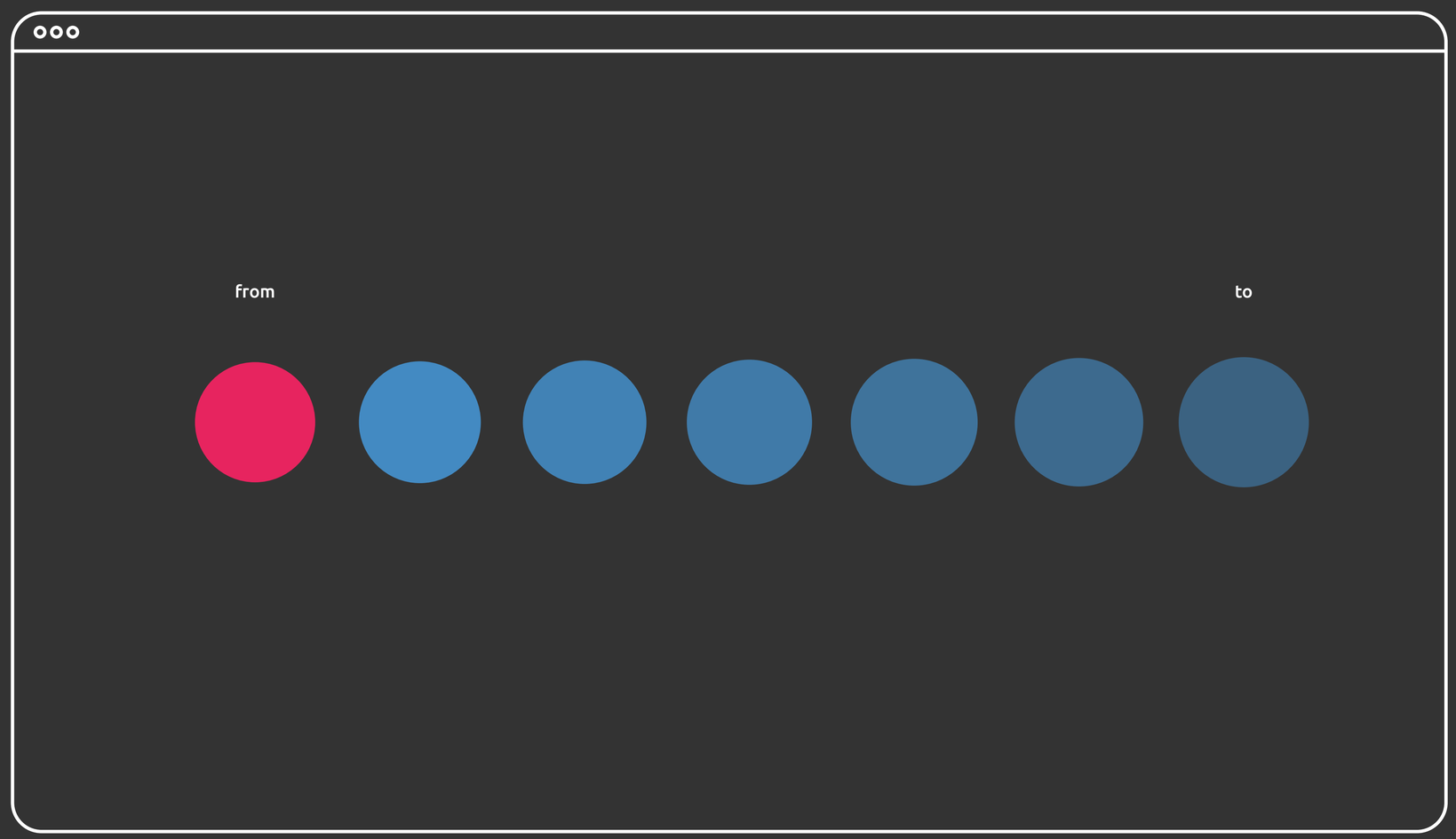
:where()

```
section h1, section h2, section h3, section h4, section h5,
section h6, article h1, article h2, article h3, article h4,
article h5, article h6, aside h1, aside h2, aside h3, aside
h4, aside h5, aside h6, nav h1, nav h2, nav h3, nav h4, nav
h5, nav h6 {
  color: #BADA55;
}

:is(section, article, aside, nav) :is(h1, h2, h3, h4, h5,
h6) {
  color: #BADA55;
}
```

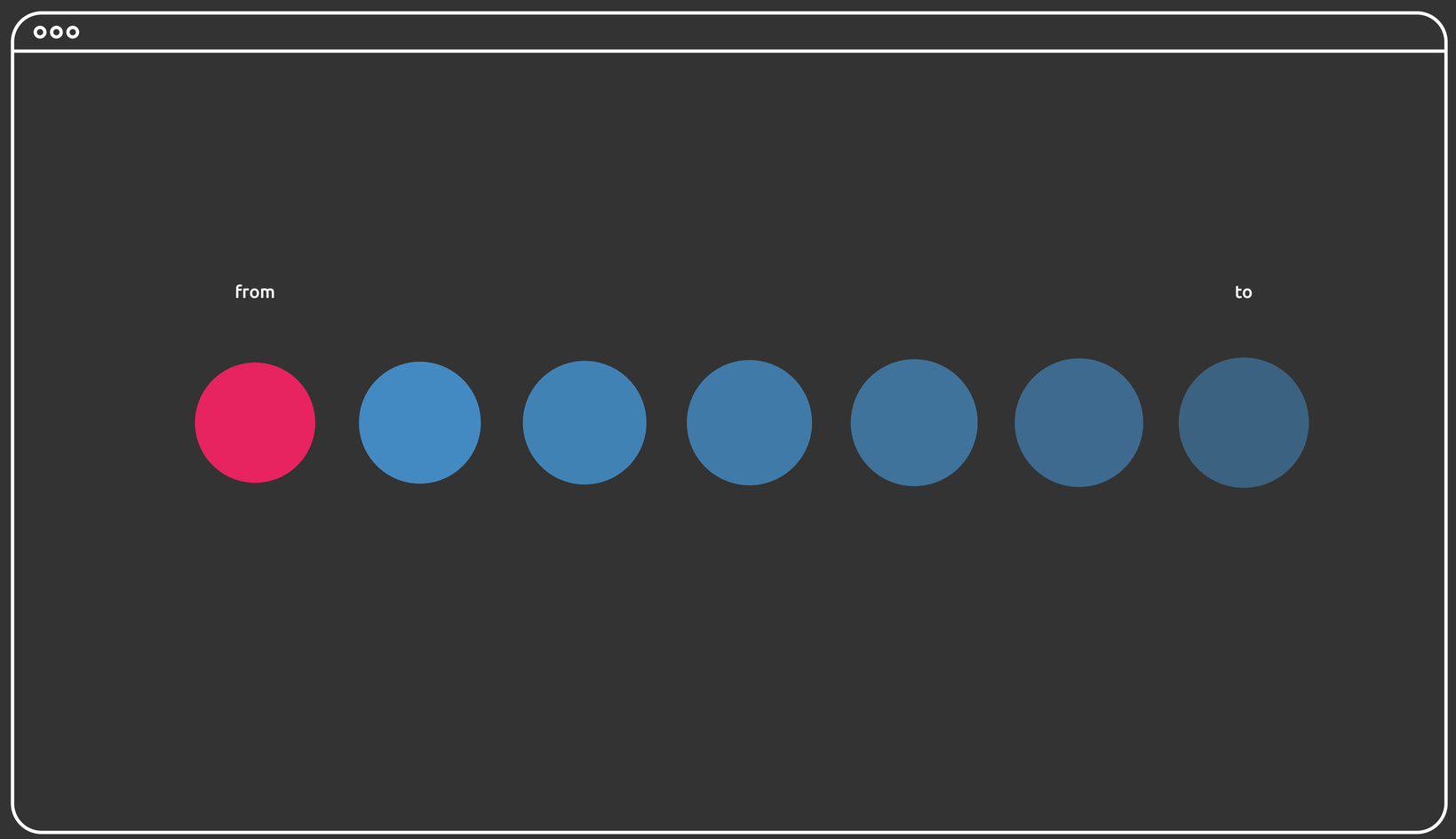
Animações

Podemos animar transições entre propriedades com css.



Animações

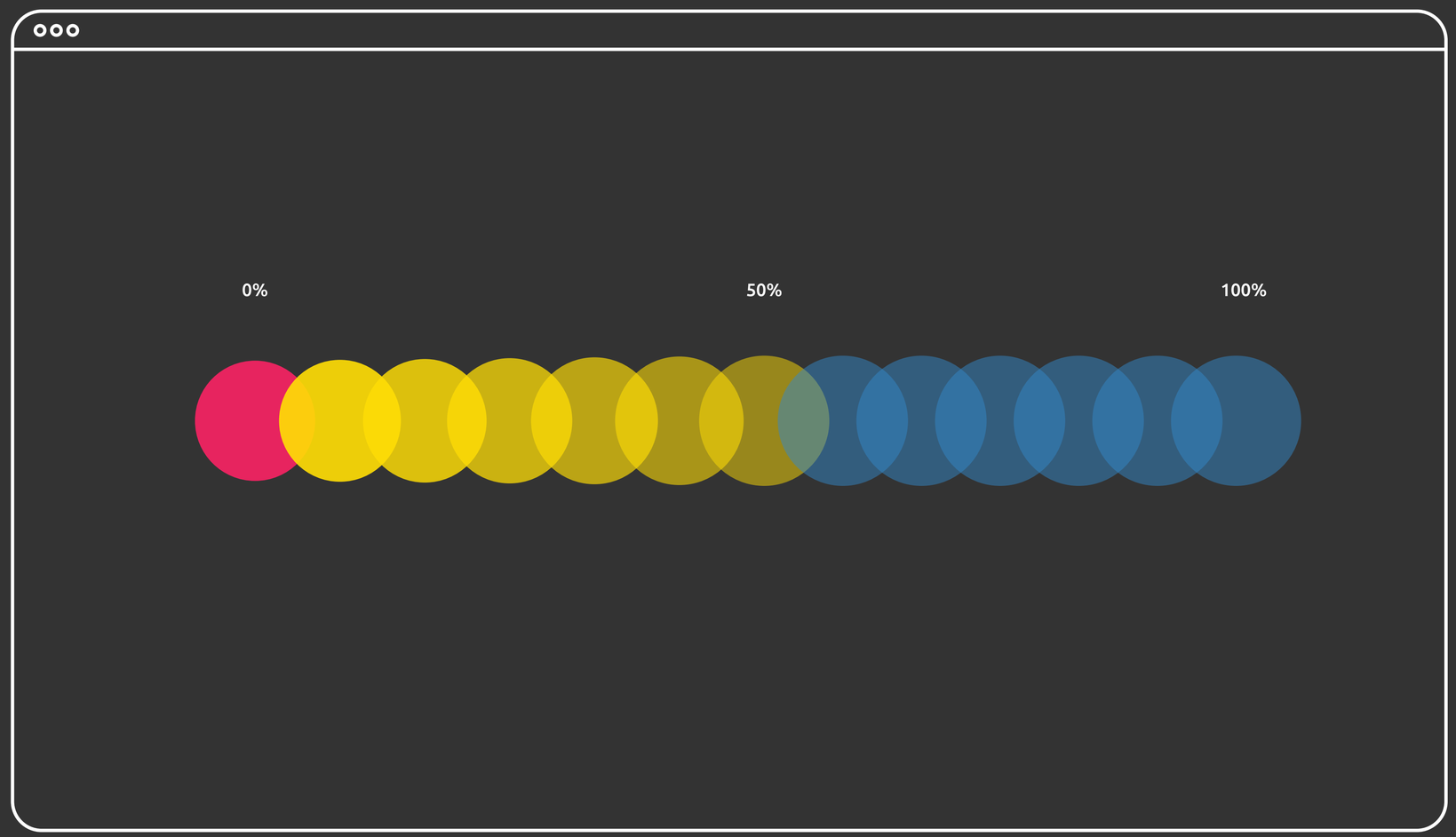
Podemos animar transições entre propriedades com css.




```
@keyframes pulse {
  from {
    background-color: #001F3F;
  }
  to {
    background-color: #FF4136;
  }
}
```

Animações

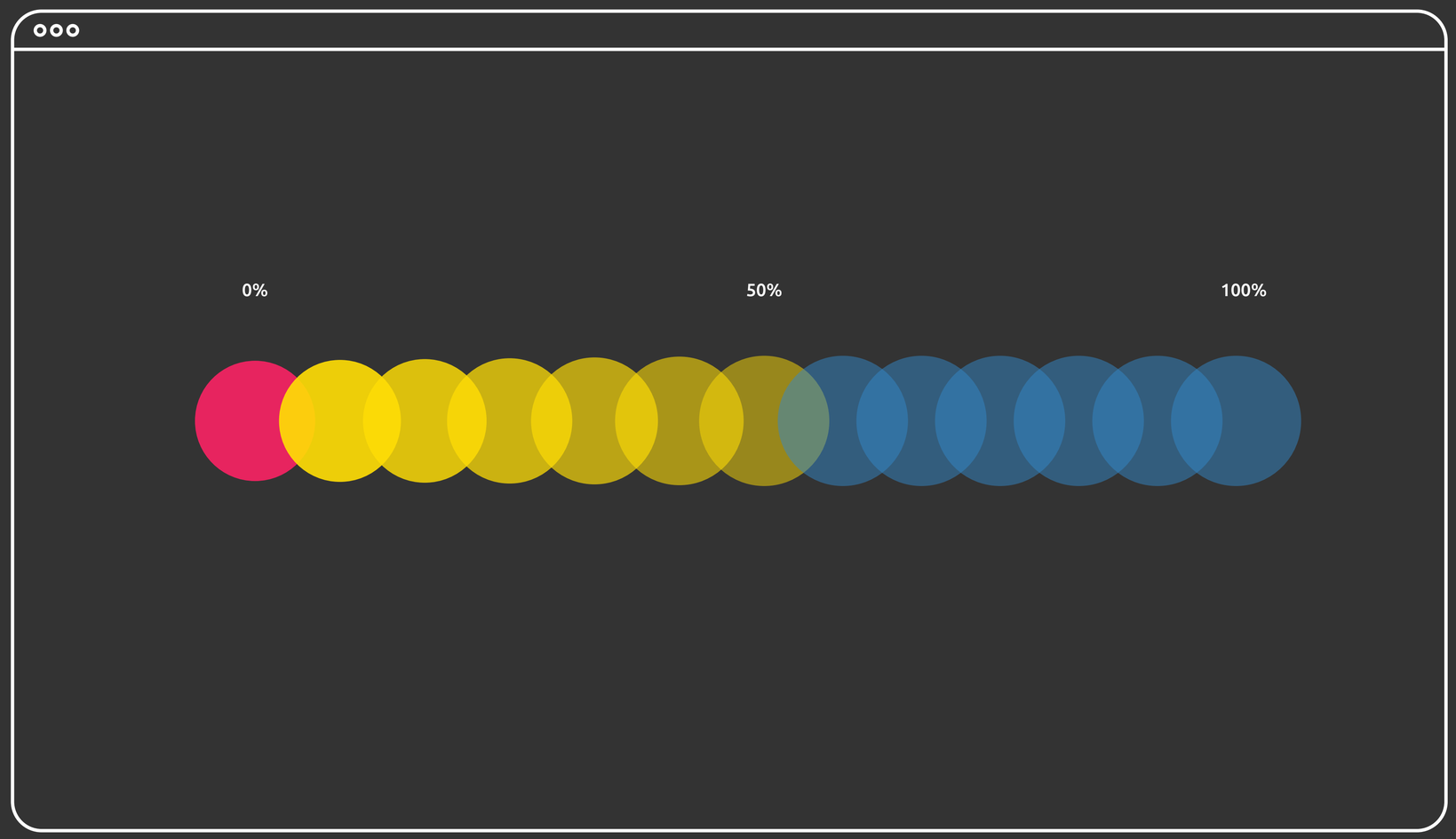
Podemos animar transições entre propriedades com css.




```
@keyframes pulse {
 0%{
    background-color: red;
 }
 50%{
    background-color: orange;
 }
 100%{
    background-color: blue;
 }
}
```

Animações

Podemos animar transições entre propriedades com css.




```
.element {
  animation-name: pulse;
  animation-duration: 1.5s;
  animation-timing-function: ease-out;
  animation-delay: 0s;
  animation-direction: alternate;
  animation-iteration-count: infinite;
  animation-fill-mode: none;
  animation-play-state: running;
}

@keyframes pulse {
  0%{
      background-color: red;
  }
  50%{
      background-color: orange;
  }
  100%{
      background-color: blue;
  }
}
```

Exercício