Depois que um site é exibido, as aparências visuais de vários elementos podem mudar por vários motivos.

Por exemplo:

- Mover o mouse sobre um link pode alterar a cor ou a aparência desse link.
- Alterar o tamanho da janela pode alterar o layout.
- A rolagem faz com que alguns elementos desapareçam e outros apareçam.

Com *as transições CSS*, podemos controlar como essas mudanças acontecem.

Essas mudanças são um tipo de *mudança de estado*. As transições CSS nos permitem controlar o tempo das mudanças de estado visual.

Podemos controlar os seguintes aspectos de transição de um elemento:

- Quais propriedades do CSS fazem a transição;
- Quanto tempo dura uma transição;
- Quanto tempo há antes de uma transição começar;
- Como uma transição acelera;

Duration

Para criar uma transição simples em CSS, devemos especificar dois dos quatro aspectos:

1. A propriedade que queremos fazer a transição.
2. A duração da transição.

```
a {
  transition-property: color;
  transition-duration: 1s;
}
```

No exemplo acima, transition-property declara em qual propriedade CSS vamos fazer a transição, no caso a cor do texto.

A segunda propriedade, transition-duration, declara quanto tempo a transição levará — no exemplo um segundo.

As mudanças de estado de muitas propriedades podem ser transicionadas. O tipo de transição depende da propriedade que você escolher.

Propriedades diferentes transitam de maneiras diferentes, por exemplo:

- Os valores de cor, como color e background-color, serão combinados com uma nova cor.
- Valores de comprimento como font-size, width e height aumentarão ou diminuirão.

A duração é especificada em segundos ou milissegundos, como 3s, 0.75s, 500ms. O valor padrão é 0s, ou instantâneo, como se não houvesse transição.

Ao escolher uma duração, pense em quanto tempo as ações levam na vida real. Por exemplo, um piscar de olhos humano leva cerca de 400ms. As pessoas podem esperar que a animação de clicar em um botão seja tão repentina quanto um piscar de olhos.

Função de temporização

A próxima propriedade de transição é transition-timing-function. A função de temporização descreve o ritmo da transição.

O valor padrão é ease, que inicia a transição lentamente, acelera no meio e desacelera novamente no final.

Outros valores válidos incluem:

- ease-in - começa devagar, acelera rapidamente, pára abruptamente
- ease-out - começa abruptamente, diminui e termina lentamente
- ease-in-out - começa devagar, fica rápido no meio e termina devagar
- linear- velocidade constante em todo

```
transition-property: color;
transition-duration: 1s;
transition-timing-function: ease-out;
```

No exemplo acima, a cor do texto será animada por um segundo. A função de temporização é ease-out o que significa que começará abruptamente e diminuirá quando terminar.

Atraso

A próxima propriedade de transição é transition-delay. A função de temporização descreve o ritmo da transição, transition-delay é 0s, o que significa sem atraso.

```
transition-property: width;
transition-duration: 750ms;
transition-delay: 250ms;
```

No exemplo acima, uma alteração na largura do elemento começará após um quarto de segundo e será animada em três quartos de segundo.

Shorthand

Agora que exploramos cada propriedade de transição, você pode se deparar com muitos conjuntos de regras CSS semelhantes ao código abaixo.

```
transition-property: color;
transition-duration: 1.5s;
transition-timing-function: linear;
transition-delay: 0.5s;
```

Como essas quatro propriedades são frequentemente declaradas juntas, o CSS fornece uma propriedade que pode ser usada para declará-las todas em uma linha: `transition`. Essa propriedade abreviada descreve cada aspecto do quebra-cabeça de transição em uma única declaração.

As propriedades devem ser especificadas nesta ordem: `transition-property`, `transition-duration`, `transition-timing-function`, `transition-delay`.

```
transition: color 1.5s linear 0.5s;
```

No exemplo acima, refatoramos as quatro linhas de código do exemplo anterior em uma linha concisa. Este exemplo fará com que qualquer alteração na cor do texto transicione em velocidade constante por mais de 1,5 segundos, após um atraso de 0,5 segundos.

Deixar de fora uma das propriedades faz com que o valor padrão dessa propriedade seja aplicado. Há uma exceção: você deve definir a duração se quiser definir o atraso. Como ambos são valores de tempo, o navegador sempre interpretará o primeiro valor de tempo que ele vê como duração.

Combinações

A regra de transição abreviada tem uma vantagem sobre o conjunto de transition-<property>regras separadas: você pode descrever transições exclusivas para várias propriedades e combiná-las.

Para combinar transições, adicione uma vírgula (,) antes do ponto e vírgula (;) em sua regra. Após a vírgula, use a mesma sintaxe abreviada. Por exemplo:

```
transition: color 1s linear,
font-size 750ms ease-in 100ms;
```

O código acima faz a transição de duas propriedades ao mesmo tempo. A cor do texto transita em um segundo com tempo linear e sem atraso. Ao mesmo tempo, o tamanho da fonte transita mais de 750 milissegundos com um ease-in tempo e um atraso de 100 milissegundos. Este "encadeamento" é uma ferramenta poderosa para expressar animações complicadas.

All

Mesmo com a abreviação, especificar transições para muitas propriedades pode ser tedioso. É comum usar a mesma duração, função de temporização e atraso para várias propriedades. Quando este for o caso, você pode definir o transition-property valor para all. Isso aplicará os mesmos valores a todas as propriedades.

Para efetuar isso, você pode usar all como um valor para transition-property.

All significa que todos os valores alterados serão transferidos da mesma maneira. Você pode usar all com as propriedades de transição separadas ou a sintaxe abreviada. Isso permite que você descreva a transição de muitas propriedades com uma única linha:

```
transition: all 1.5s linear 0.5s;
```

Neste exemplo, qualquer alteração será animada durante um segundo e meio após um atraso de meio segundo com tempo linear.

Como tornar as transições mais interativas usando a propriedade animation e keyframes

Podemos fazer mais com as transições do CSS para tornar essa animação mais criativa e interativa.

Exemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta http-equiv="X-UA-Compatible" content="ie=edge" />
<title>Static Template</title>
</head>
<style>
        .elem {
        background: orange;
        width: 300px;
        height: 150px;
        animation: moveToRight 2s ease-in-out;
        animation-delay: 1000ms;
        }

        @keyframes moveToRight {
                0% {
                transform: translateX(0px);
                }
                100% {
                transform: translateX(300px);
                }
                }
</style>
<body>
<div class="elem"></div>
</body>
</html>
```

Neste exemplo foi usado propriedades como animation e keyframes.

Usamos a propriedade animation para definir o nome da animação e a duração, e usamos keyframes para descrever como deve ser a movimentação do elemento.

Existem outras funções de tempo que podem ser usadas, como ease-in, linear, ease-out, que, basicamente, tornam a animação mais suave.

@keyframes recebe o nome da animação.

Neste caso, o nome é moveToRight.

```
@keyframes moveToRight {
        0% {
         transform: translateX(0px);
        }
        100% {
        transform: translateX(300px);
          }
         }
```

keyframes executará a animação em diversas etapas. O exemplo acima usa a porcentagem para representar o intervalo ou a ordem das transições.

Também podemos usar os métodos from e to, como vemos abaixo:

```
 @keyframes moveToRight {
        from {
                transform: translateX(0px);
                 }
        to {
                 transform: translateX(300px);
                }
 }
```

- from representa o ponto de início ou a primeira parte da animação.
- to é o ponto final ou a última parte da animação a ser executada.

Podemos querer usar uma porcentagem em alguns casos. Por exemplo, digamos que queiramos adicionar mais do que duas transições que serão executadas em sequência, como vemos abaixo:

```
@keyframes moveToRight {
        0% {
        transform: translateX(0px);
         }
        50% {
        transform: translateX(150px);
         }
        100% {
        transform: translateX(300px);
         }
}
```