ANIMATIONS

TBC

📖 ANIMATIONS

Angular's animation system is built on CSS functionality, which means you can animate any property that the browser considers animatable. This includes positions, sizes, transforms, colors, borders, and more.

📖 IMPORT

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

@NgModule({
  imports: [
    BrowserModule,
    BrowserAnimationsModule
  ],
  declarations: [ ],
  bootstrap: [ ]
})
export class AppModule { }
```

TBC

📖 IMPORT

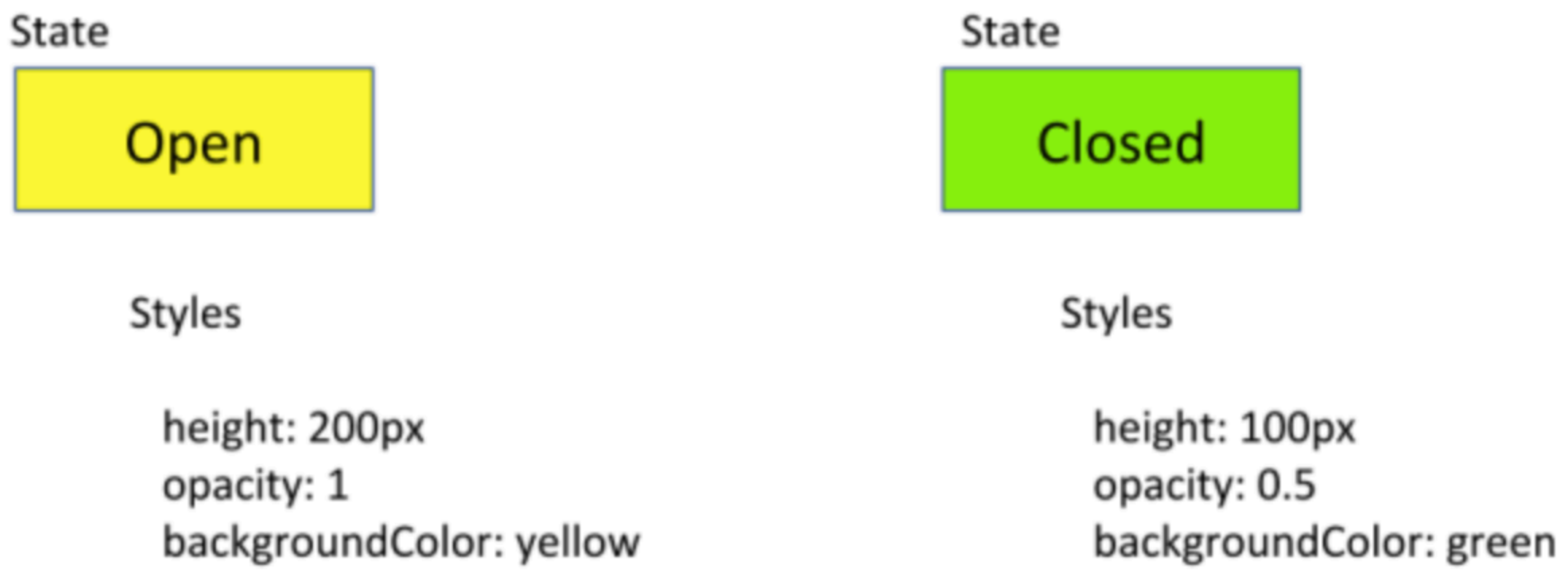
```
import { Component, HostBinding } from '@angular/core';
import {
  trigger,
  state,
  style,
  animate,
  transition,
  // ...
} from '@angular/animations';
```

TBC

📖 IMPORT

```
@Component({
  selector: 'app-root',
  templateUrl: 'app.component.html',
  styleUrls: ['app.component.css'],
  animations: [
    // animation triggers go here
  ]
})
```

📖 **ANIMATION**



State

Open

Styles

height: 200px
opacity: 1
backgroundColor: yellow

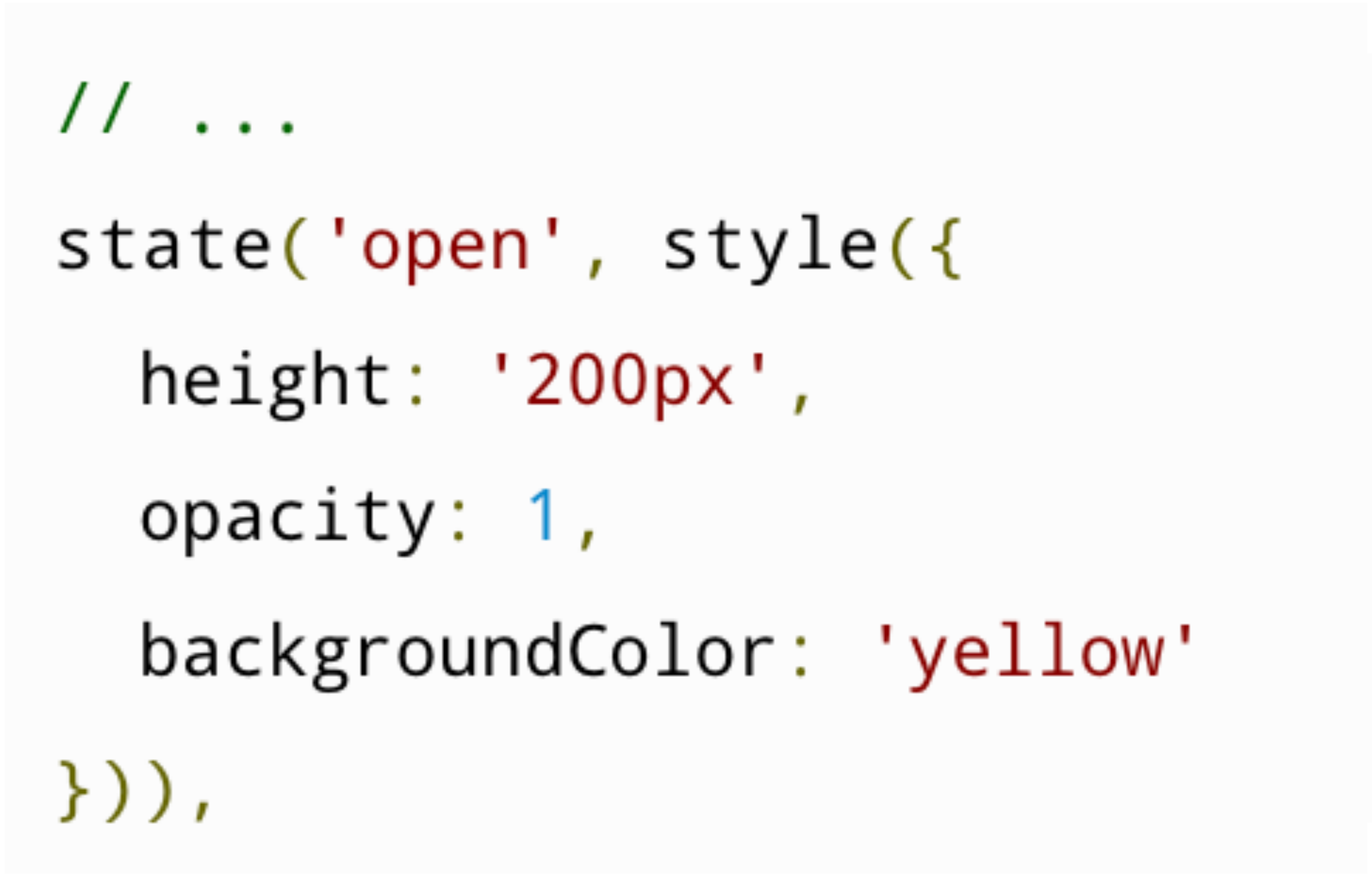
State

Closed

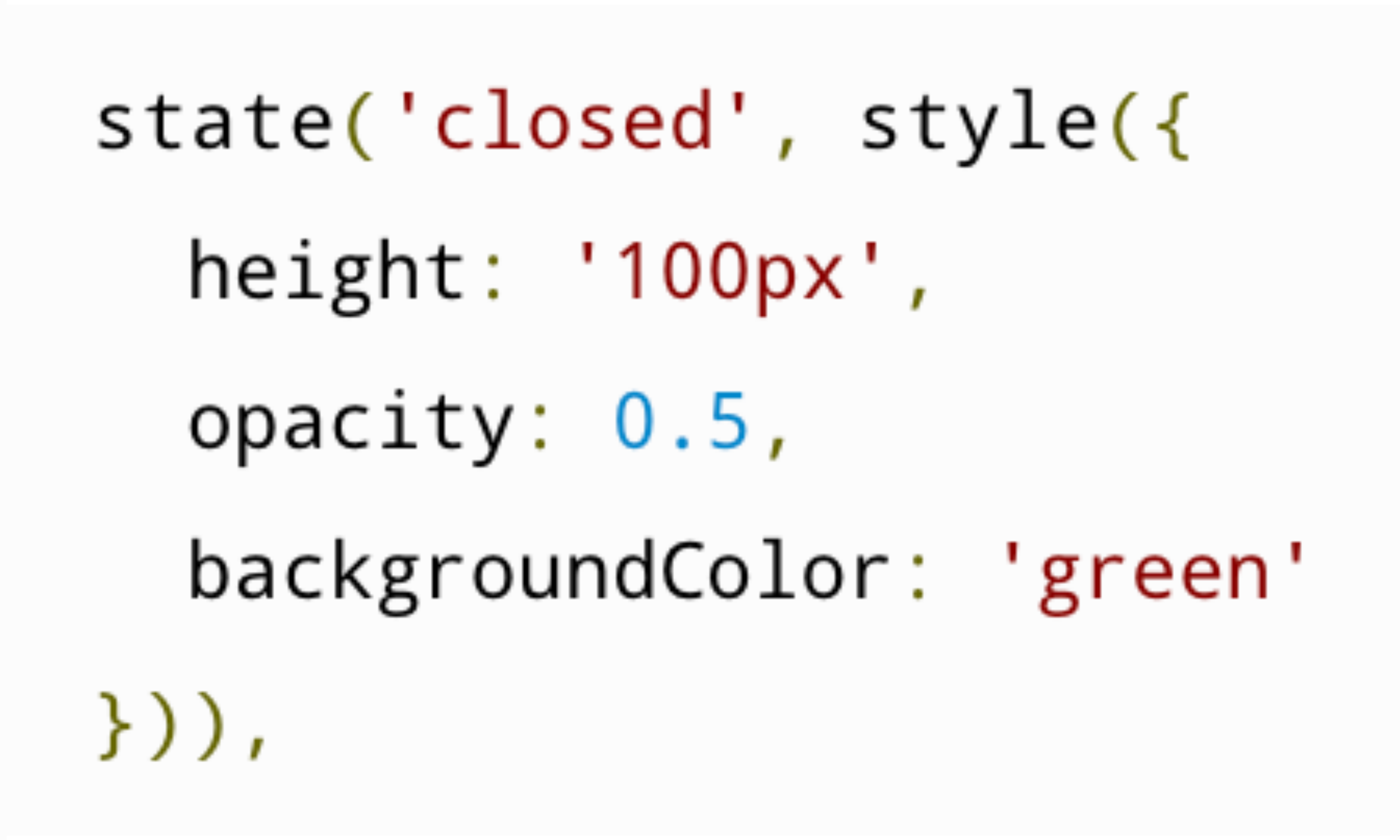
Styles

height: 100px
opacity: 0.5
backgroundColor: green

📖 STATE

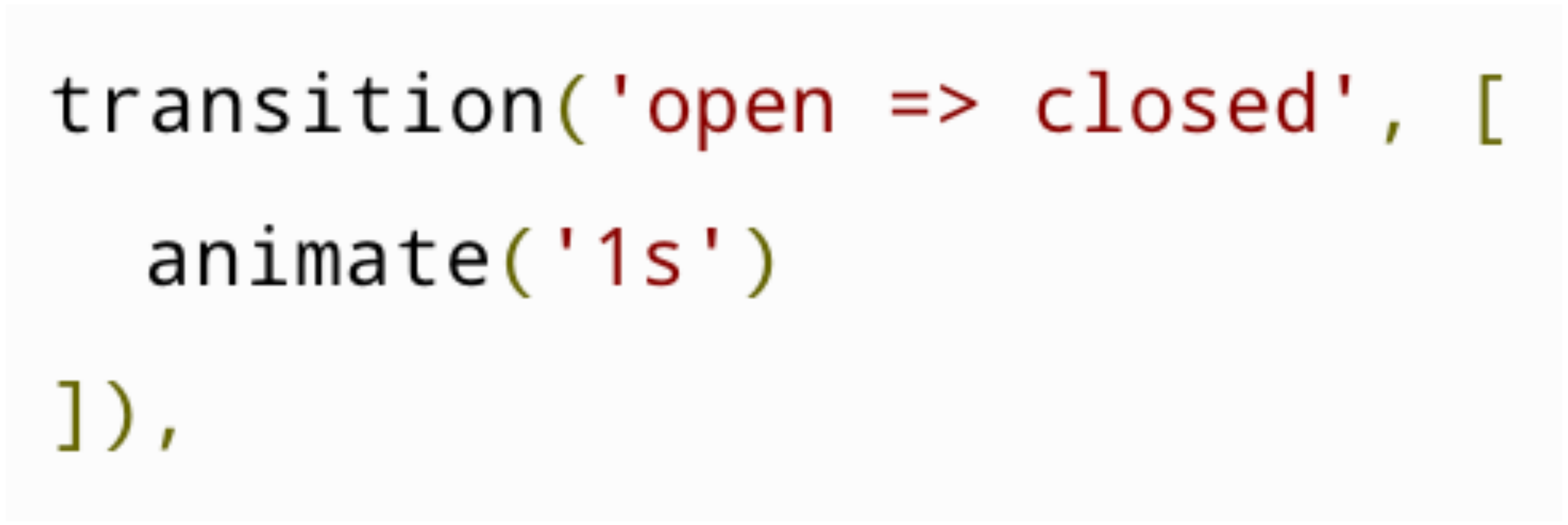


```
// ...
state('open', style({
  height: '200px',
  opacity: 1,
  backgroundColor: 'yellow'
})),
```

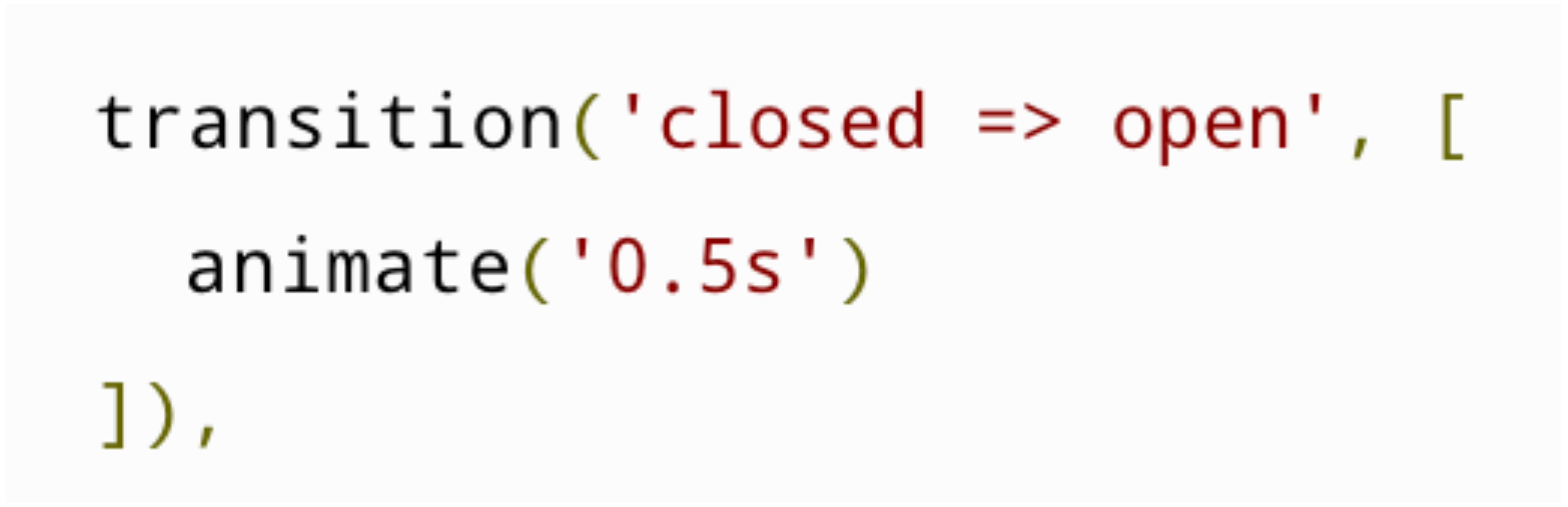


```
state('closed', style({
  height: '100px',
  opacity: 0.5,
  backgroundColor: 'green'
})),
```

📖 TRANSITION

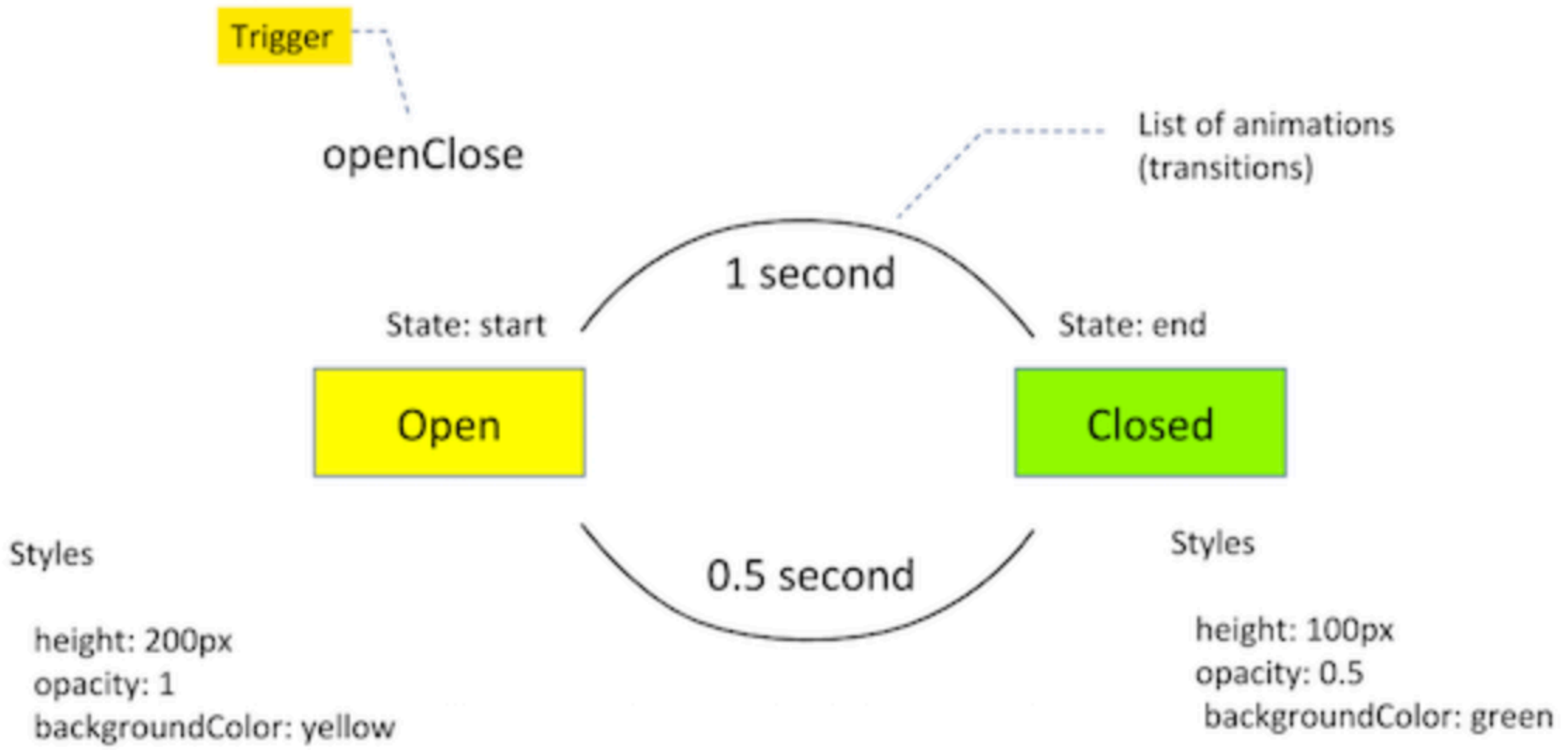


```
transition('open => closed', [
  animate('1s')
]),
```

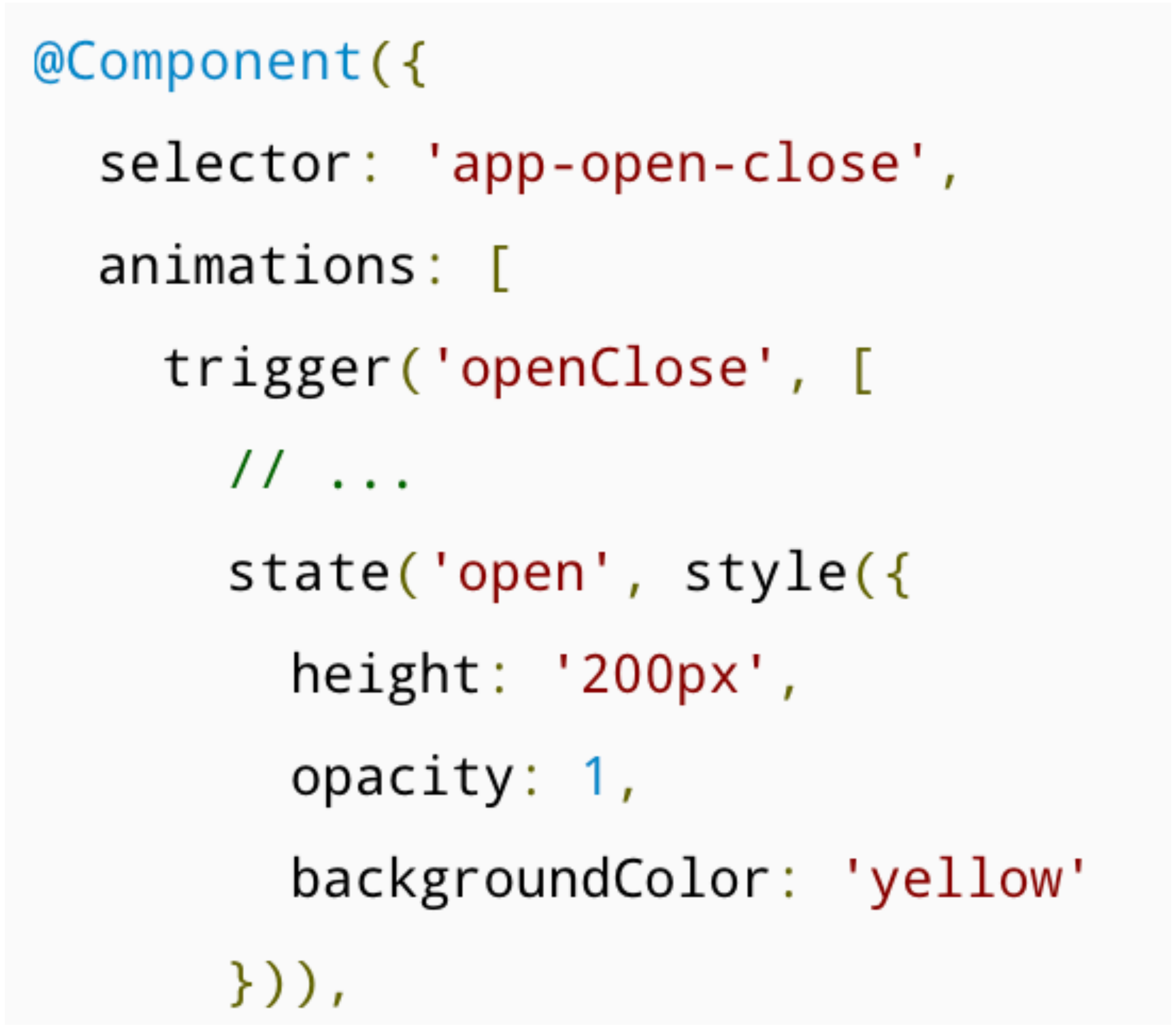


```
transition('closed => open', [
  animate('0.5s')
]),
```

📖 TRIGGER


Trigger
openClose
List of animations (transitions)
1 second
State: start
State: end
Open
Closed
0.5 second
Styles
height: 200px
opacity: 1
backgroundColor: yellow
Styles
height: 100px
opacity: 0.5
backgroundColor: green

TRIGGER



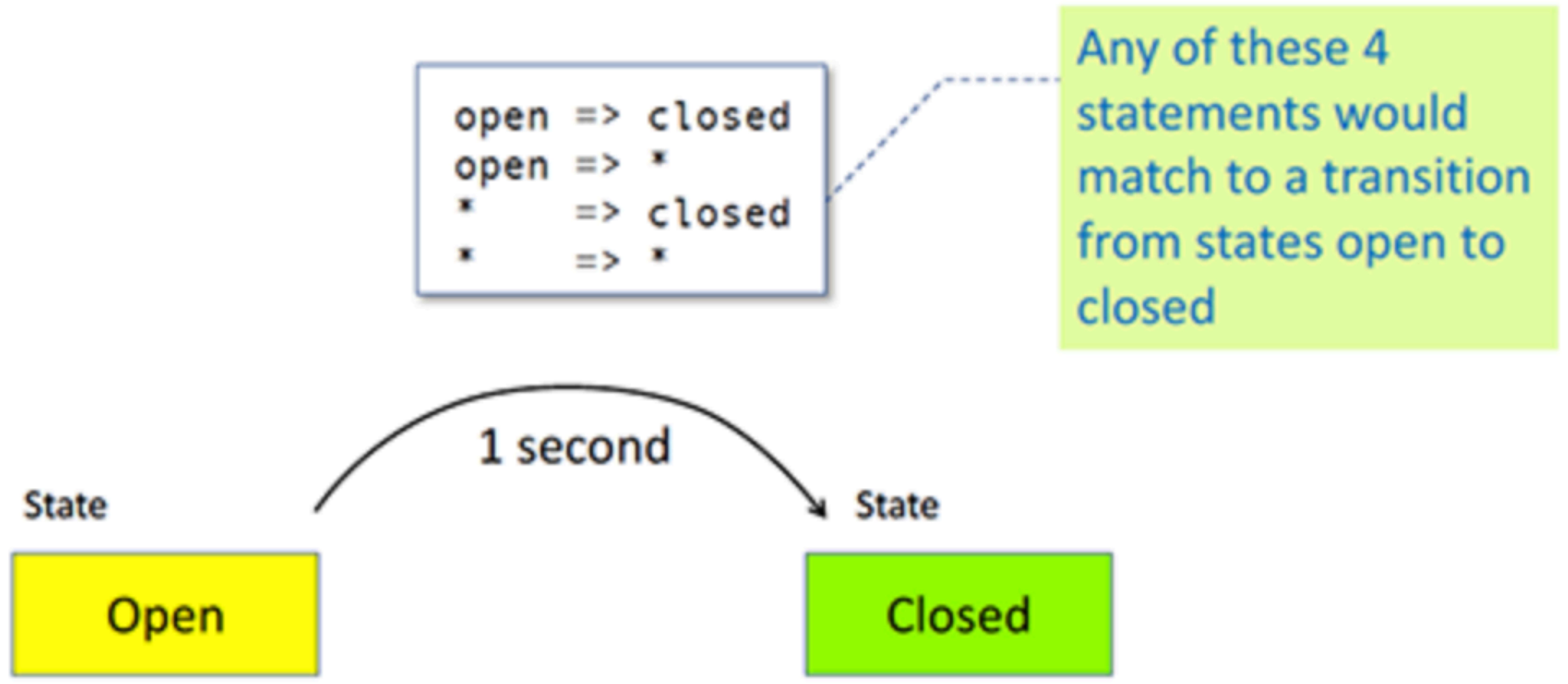
```
@Component({
  selector: 'app-open-close',
  animations: [
    trigger('openClose', [
      // ...
      state('open', style({
        height: '200px',
        opacity: 1,
        backgroundColor: 'yellow'
      })),
```

📖 TRIGGER

```
<div [@openClose]="isOpen ? 'open' : 'closed'" class="open-close-container">
  <p>The box is now {{ isOpen ? 'Open' : 'Closed' }}!</p>
</div>
```

TBC

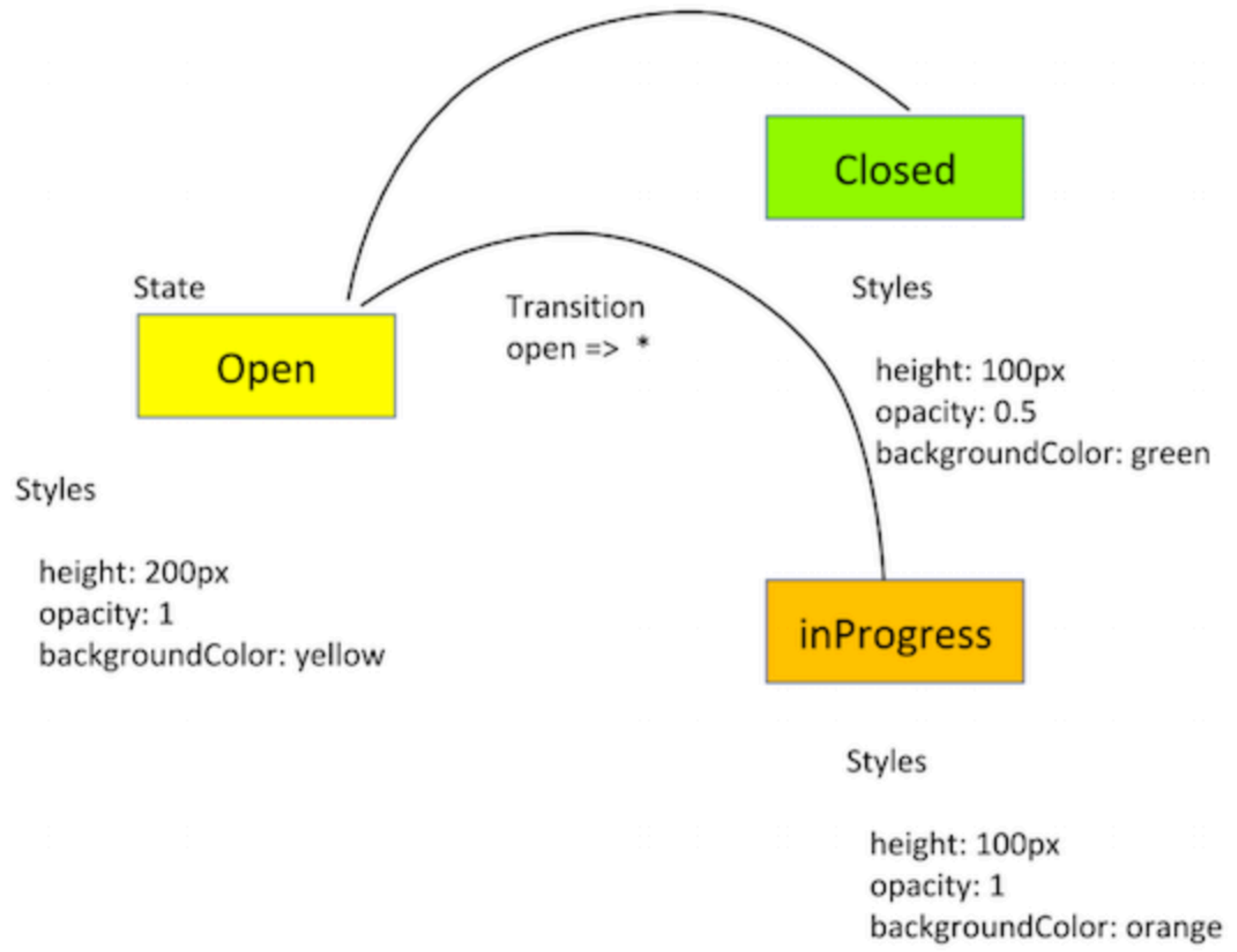
📖 STATE



```
open => closed
open => *
*    => closed
*    => *
```

Any of these 4 statements would match to a transition from states open to closed

📖 STATE



Closed
State
Open
Transition
open => *
Styles
height: 100px
opacity: 0.5
backgroundColor: green
Styles
height: 200px
opacity: 1
backgroundColor: yellow
inProgress
Styles
height: 100px
opacity: 1
backgroundColor: orange

TBC

📖 STATE

You can combine wildcard and void states in a transition to trigger animations that enter and leave the page:

- A transition of * => void applies when the element leaves a view, regardless of what state it was in before it left.
- A transition of void => * applies when the element enters a view, regardless of what state it assumes when entering.
- The wildcard state * matches to any state, including void.

TBC

📖 STATE

```
transition ( ':enter', [ ... ] );  // alias for void => *
transition ( ':leave', [ ... ] );  // alias for * => void
```

TBC

📖 DISABLE

```
<div [@.disabled]="isDisabled">
  <div [@childAnimation]="isOpen ? 'open' : 'closed'"
    class="open-close-container">
    <p>The box is now {{ isOpen ? 'Open' : 'Closed' }}!</p>
  </div>
</div>
```

📖 DISABLE

```
@Component({
  selector: 'app-root',
  templateUrl: 'app.component.html',
  styleUrls: ['app.component.css'],
  animations: [
    slideInAnimation
    // animation triggers go here
  ]
})
export class AppComponent {
  @HostBinding('@.disabled')
  public animationsDisabled = false;
}
```

TBC

📖 EVENT

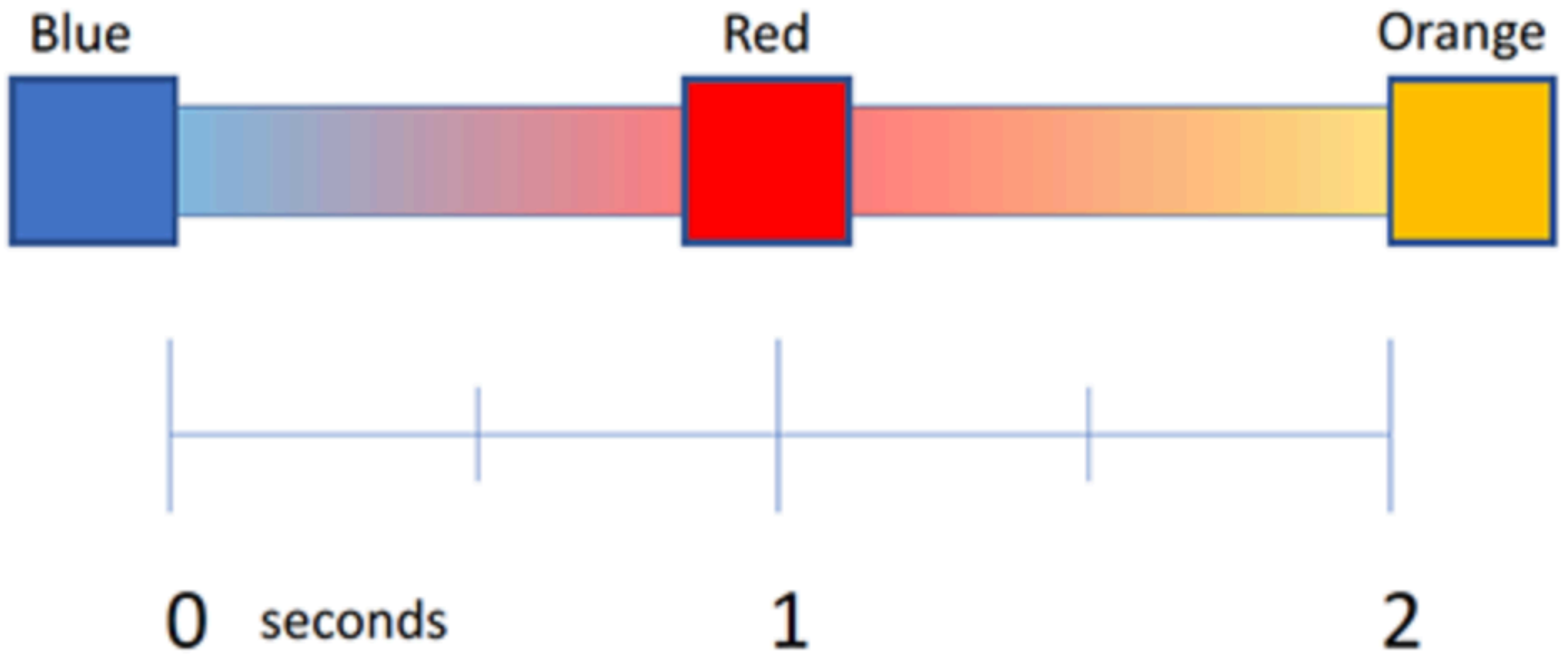
```
<div [@openClose]="isOpen ? 'open' : 'closed'"
    (@openClose.start)="onAnimationEvent($event)"
    (@openClose.done)="onAnimationEvent($event)"
    class="open-close-container">
</div>
```

TBC

📖 EVENT

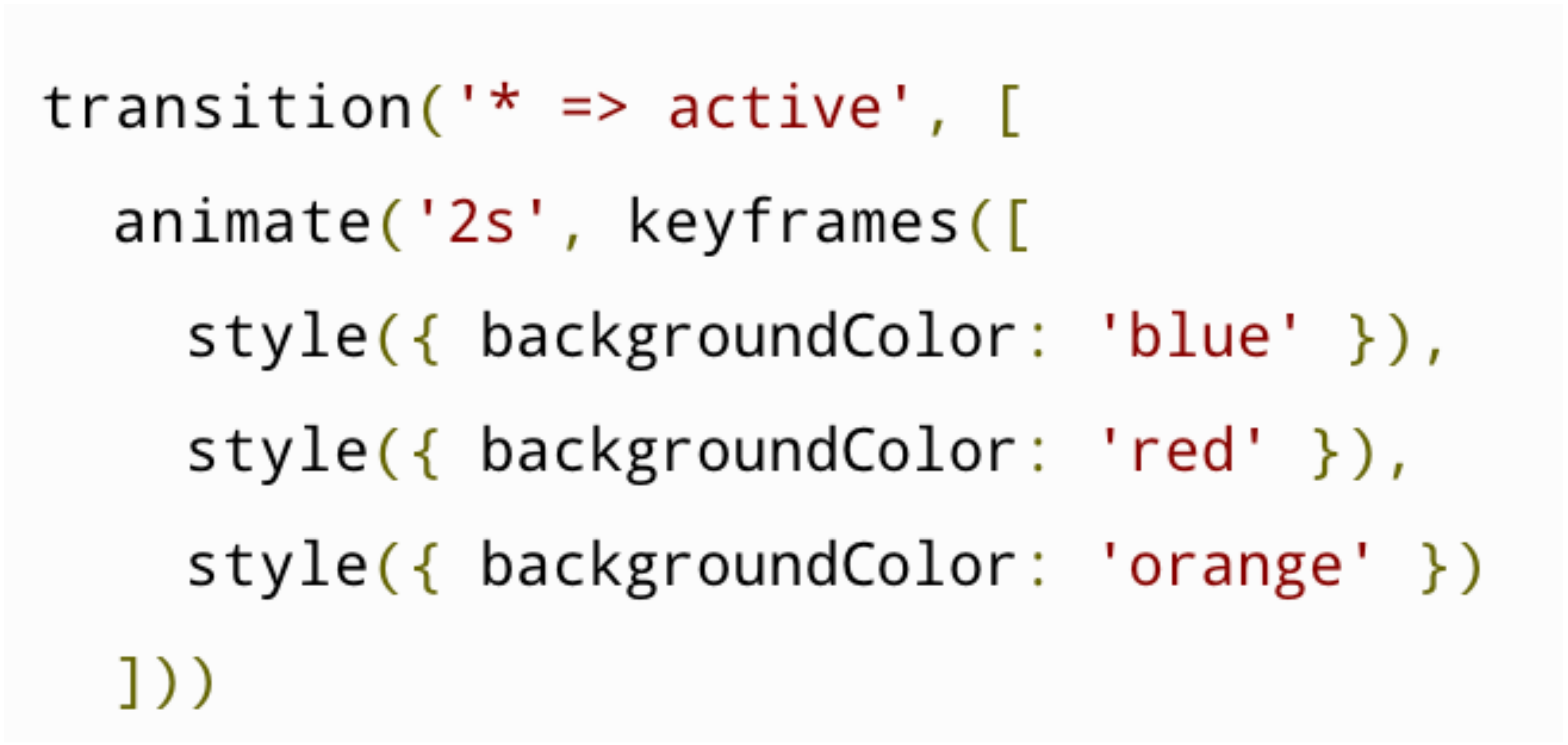
```
export class OpenCloseComponent {
  onAnimationEvent ( event: AnimationEvent ) {
    // openClose is trigger name in this example
    console.warn(`Animation Trigger: ${event.triggerName}`);

    // phaseName is start or done
    console.warn(`Phase: ${event.phaseName}`);

    // in our example, totalTime is 1000 or 1 second
    console.warn(`Total time: ${event.totalTime}`);

    // in our example, fromState is either open or closed
    console.warn(`From: ${event.fromState}`);

    // in our example, toState either open or closed
    console.warn(`To: ${event.toState}`);

    // the HTML element itself, the button in this case
    console.warn(`Element: ${event.element}`);
  }
}
```

📖 KEYFRAMES



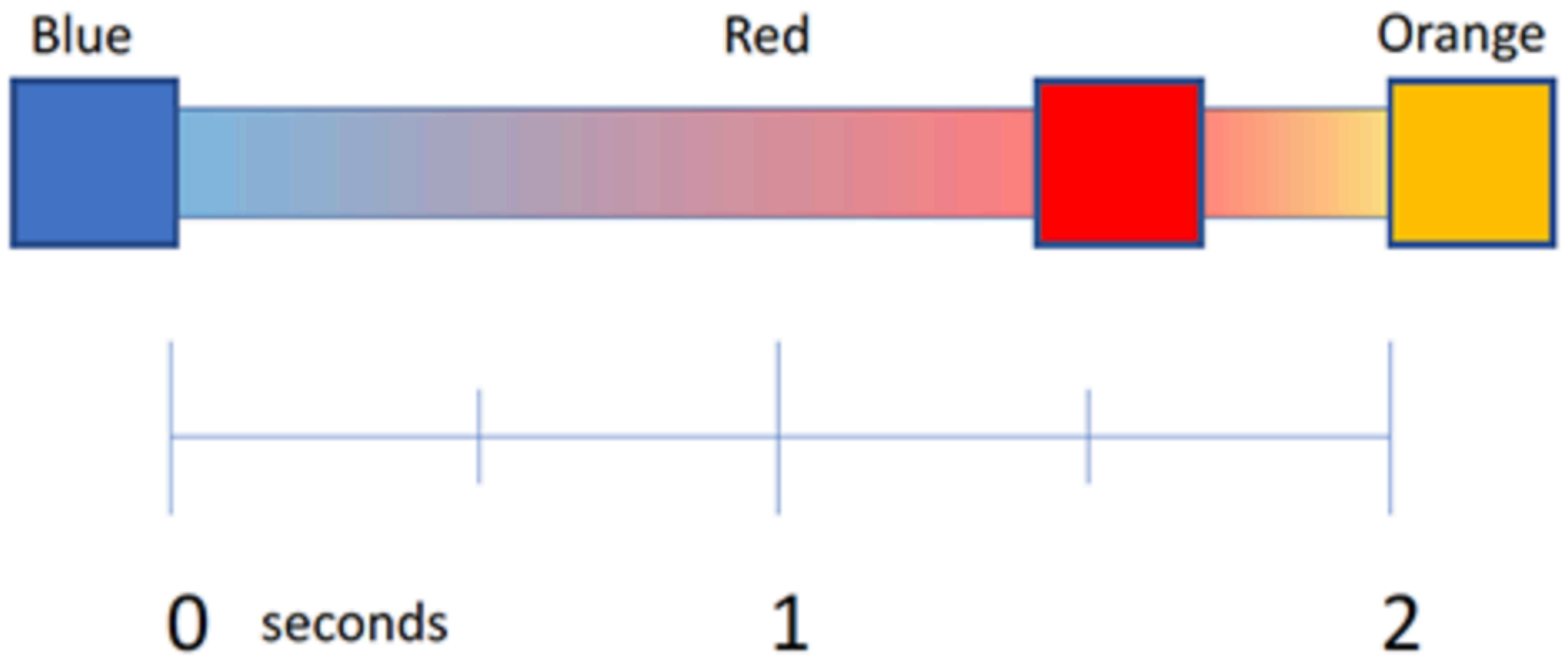
TBC

KEYFRAMES



```
transition('* => active', [
  animate('2s', keyframes([
    style({ backgroundColor: 'blue' }),
    style({ backgroundColor: 'red' }),
    style({ backgroundColor: 'orange' })
  ]))
```

📖 KEYFRAMES

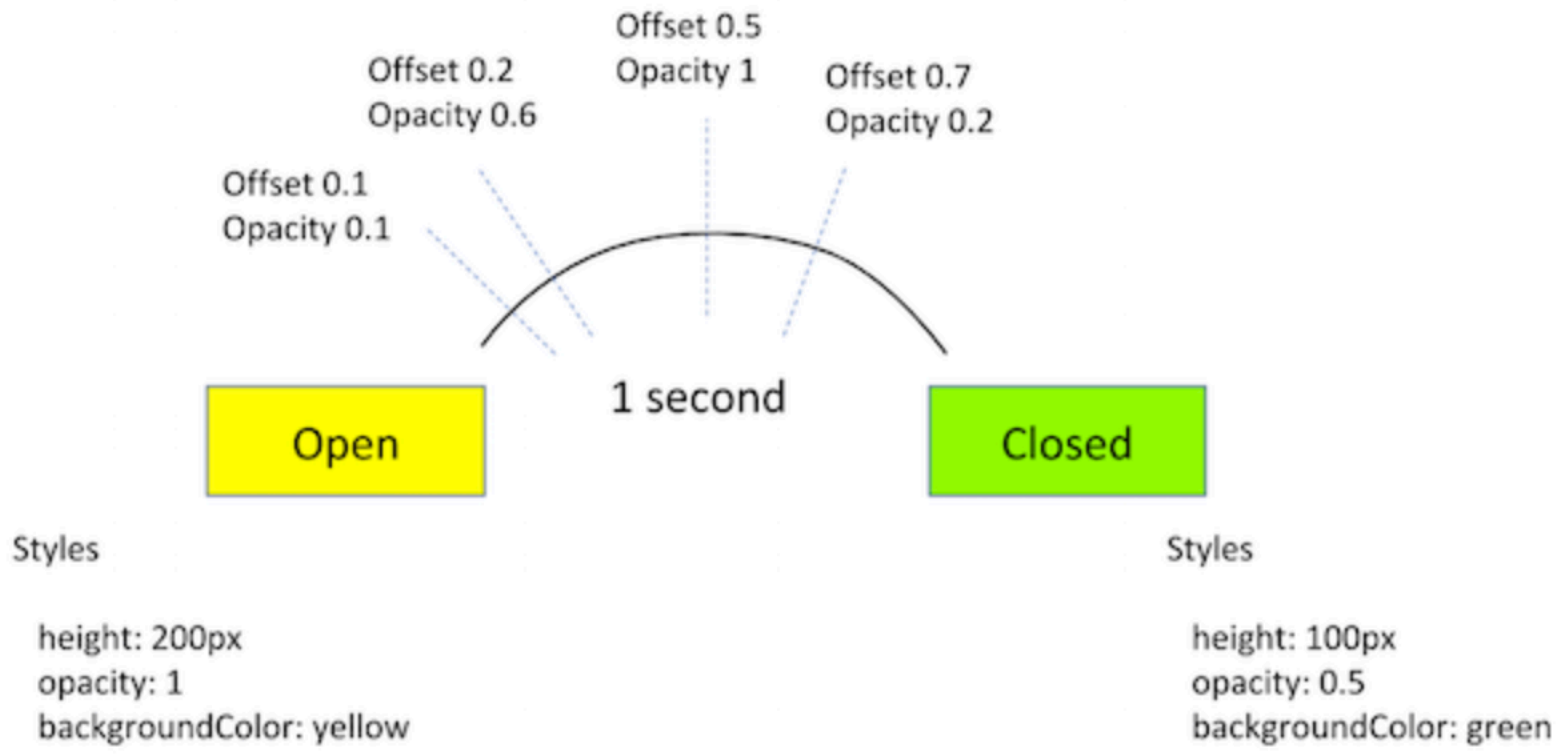


TBC

📖 KEYFRAMES

```
animate('2s', keyframes([
  style({ backgroundColor: 'blue', offset: 0}),
  style({ backgroundColor: 'red', offset: 0.8}),
  style({ backgroundColor: 'orange', offset: 1.0})
])),
```

📖 KEYFRAMES


Offset 0.5
Opacity 1
Offset 0.2
Opacity 0.6
Offset 0.7
Opacity 0.2
Offset 0.1
Opacity 0.1
1 second
Open
Closed
Styles
height: 200px
opacity: 1
backgroundColor: yellow
Styles
height: 100px
opacity: 0.5
backgroundColor: green

HERE GOES!

TBC