
ADVENTURES
IN WEB ANIMATIONS

Guiding Users

We guide users online with the design tools we have available: color, font, and **motion**.

Users, like pieces on a game board, must be guided.

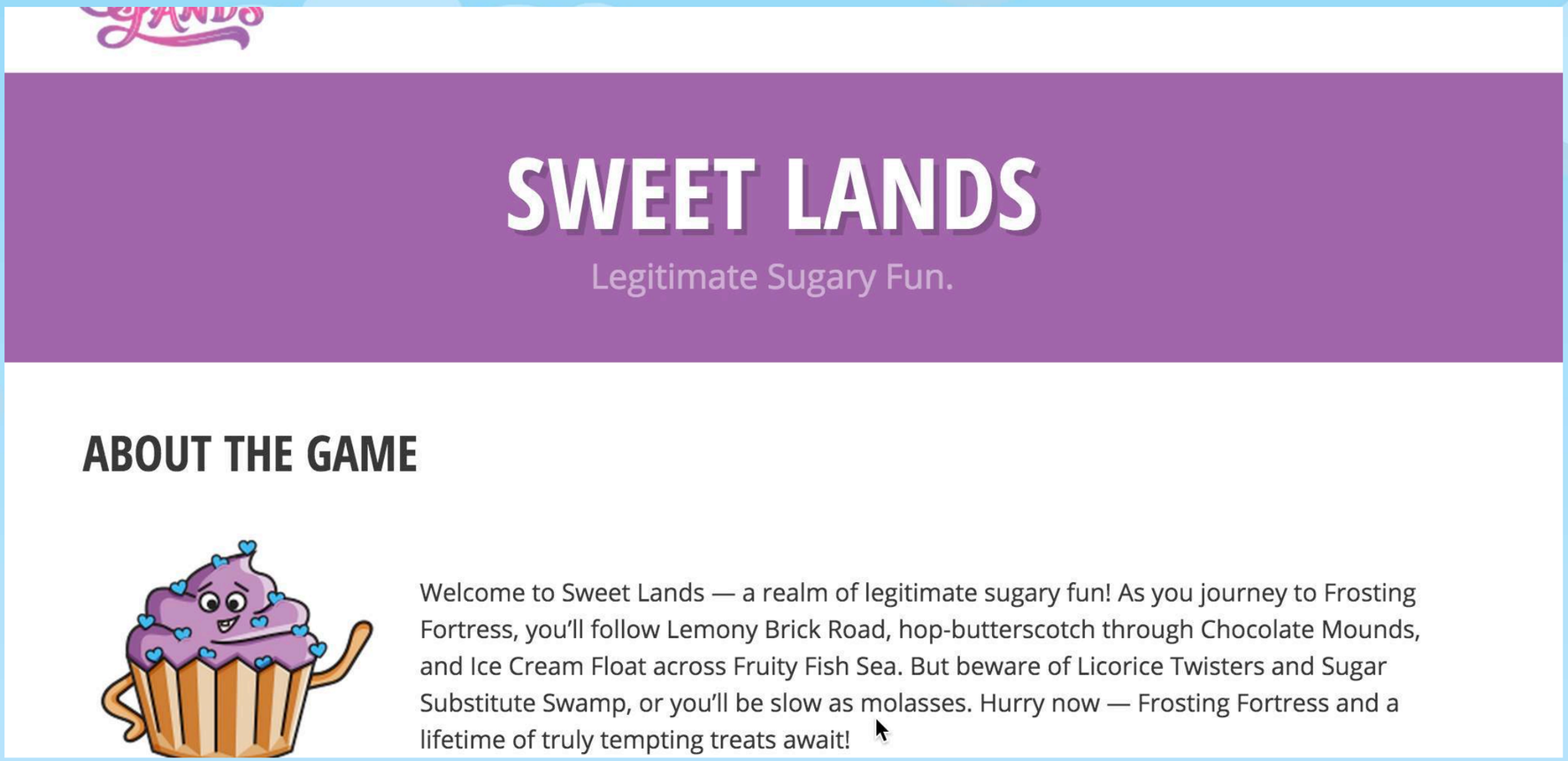
Course Outline

- Transitions
- Transforms
- Keyframes Animations
- SVG Animations

Adventures in Web Animations

Sweet Lands App

Learning to animate this site, which sells the game “Sweet Lands.”



Working knowledge of HTML and CSS are a pre-req for this course.
A bit of JavaScript and jQuery are helpful, but not needed.

Level 1 – Transitions

SECTION 1

Transitioning Color

What Is a Transition?

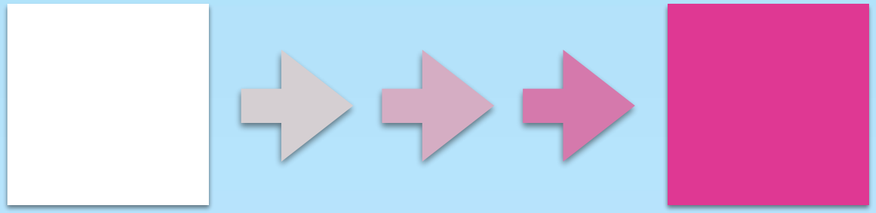
Transitions cause changes to a property and take place over a period of time.

Instant Color Change

Transition Color Change

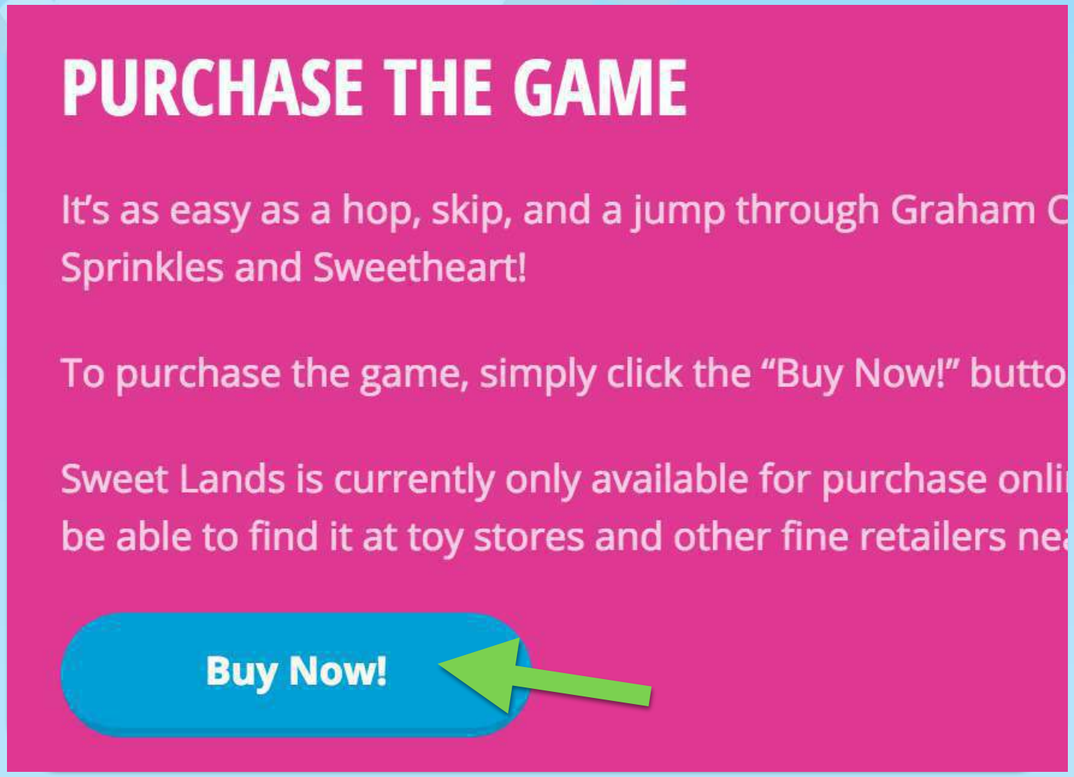
instant

over time



Prominent Call to Action

A call to action (CTA) is generally the primary goal for a user on a site. Here, purchasing the game is top priority.



We can use CSS transitions to make the "Buy Now!" button a prominent call to action on the site.

Styling the Button's Initial and Hover States

HTML

```
<a href='#' class='btn'>
  Buy Now!
</a>
```

CSS

```
.btn {
  background-color: #00A0D6;
}

.btn:hover {
  background-color: #007DA7;
}
```

You can transition color in many forms:
white, #FFF, rgb(255, 255, 255).

Transitioning Background Color

Stay *subtle* with transitions and only dramatize when you really need to call attention to an element.

The fastest transition easily seen by the human eye is .256s.

Transition recipe:

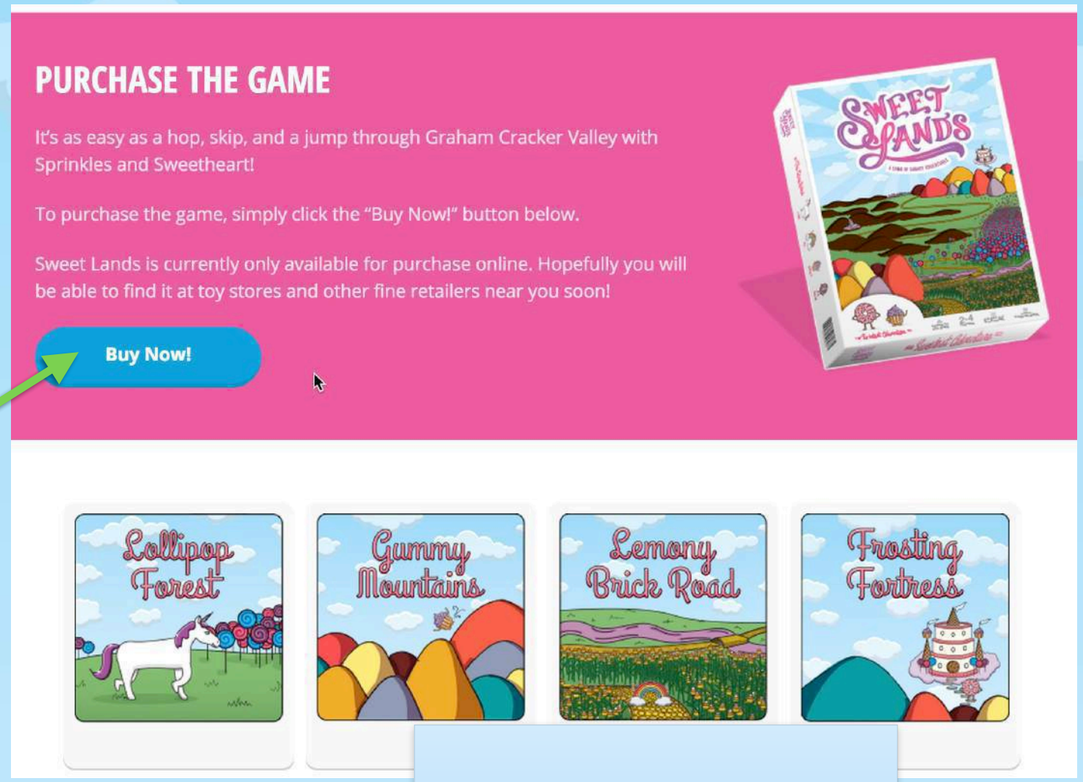
```
transition: <property> <duration>;
```

The transition is added to the starting state.

CSS

```
.btn {
  background-color: #00A0D6;
  transition: background-color 0.4s ;
}

.btn:hover {
  background-color: #007DA7;
}
```



Transitioning Background Color

The transition is added to the starting state.

CSS

```css
.btn {
  background-color: #00A0D6;
  transition: background-color 0.4s ;
}

.btn:hover {
  background-color: #007DA7;
}
```

Transitioning Multiple Properties

You can transition multiple comma-separated properties.

CSS

```css
.btn {
  background-color: #00A0D6;
  color: #FFFFFF;
  transition: background-color 0.4s, color 0.4s;
}

.btn:hover {
  background-color: #007DA7;
  color: #E3E3E3;
}
```

initial state

Buy Now!

hover state

Buy Now!

A Keyword to Transition All Properties

Use the `all` keyword to transition every changing property.

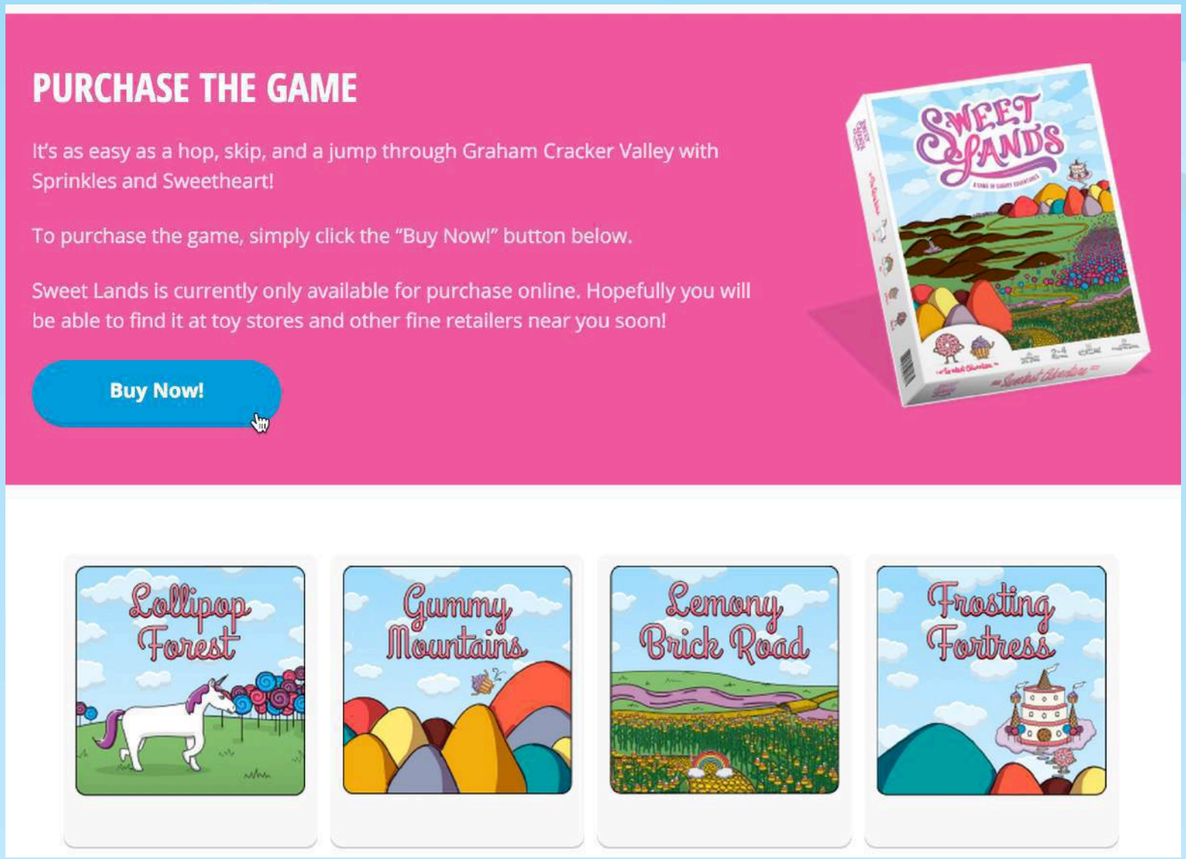
```
transition: background-color 0.4s, color 0.4s;
```

CSS

```
.btn {
  background-color: #00A0D6;
  color: #FFFFFF;
  transition: all 0.4s;
}

.btn:hover {
  background-color: #007DA7;
  color: #AD7EB6;
}
```

Be careful, though, because any property that can animate will animate.



Transition Recipe

Order is irrelevant as long as you have your duration number specified before the delay number.

```
.btn {
  transition: <property> <duration> <timing-function> <delay>;
}
```

Defaults to all

Defaults to ease

Defaults to 0

CSS

```
.btn {
  background-color: #00A0D6;
  color: #92539E;
  transition: all 0.4s ease 0;
}
```

=

CSS

```
.btn {
  background-color: #00A0D6;
  color: #92539E;
  transition: 0.4s;
}
```

Leave defaults out unless you need to change them

When to Use Vendor Prefixes

All of our examples are without vendor prefixes, but you might need to include them.



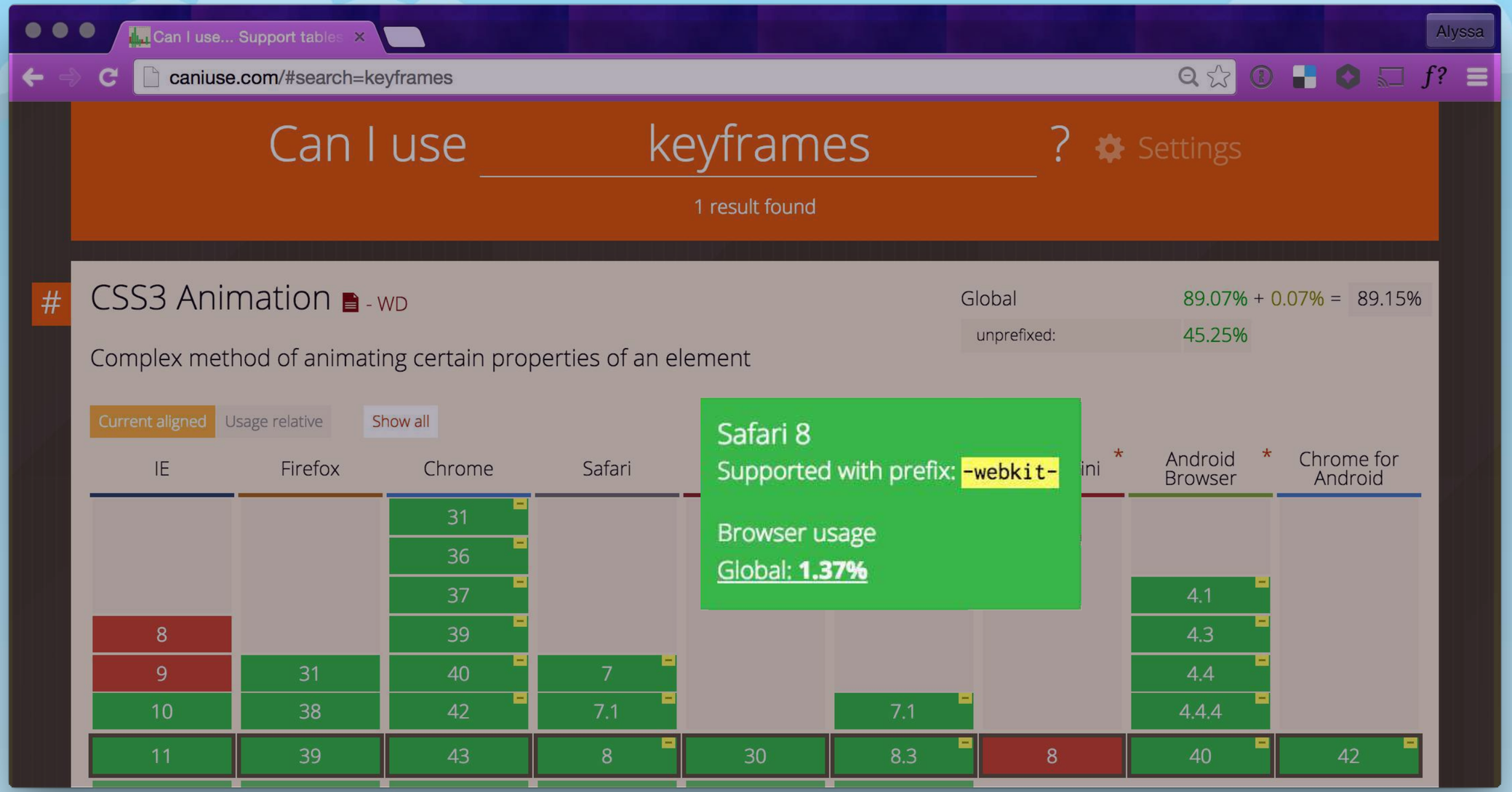


CSS

```css
.btn {
  -webkit-transition: background-color .4s;
  -moz-transition: background-color .4s;
  -ms-transition: background-color .4s;
  -o-transition: background-color .4s;
  transition: background-color .4s;
}
```

Which Properties Need Prefixes and When?

Use a site like caniuse.com to check browser support for prefixes.



Level 1 – Transitions

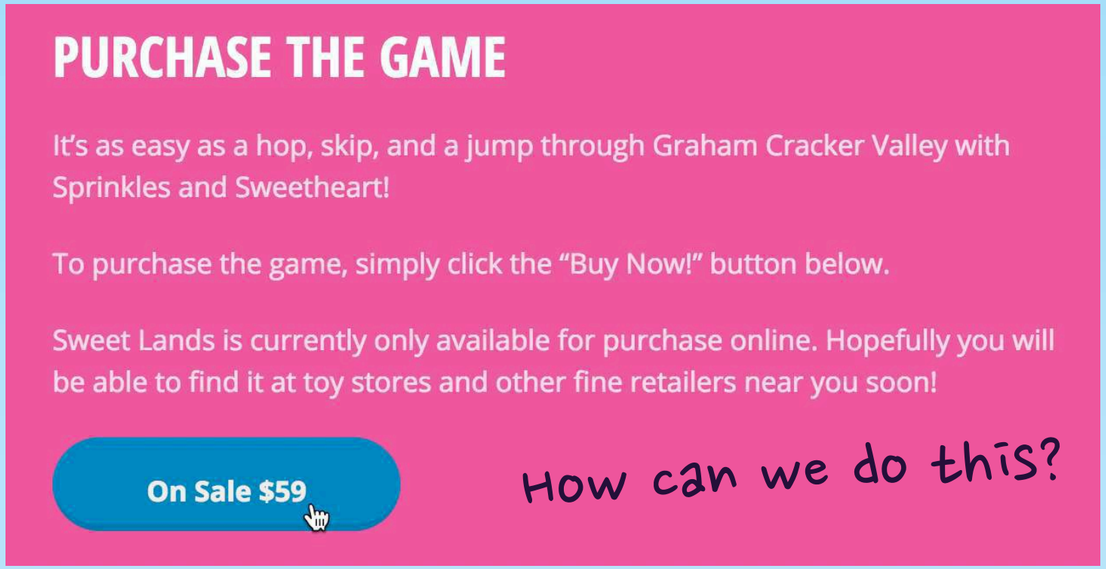
SECTION 2

Transitioning Position

Transition In Hidden Content

Moving hidden content onto the screen is another common use for transitions. This also adds personality and provides more info to the user on hover.

Button with extra hidden content



Steps to Transition In Hidden Button Content

Current button

HTML

```
<section>
  <a href='#' class='btn buy-button'>
    Buy Now!
  </a>
</section>
```

1. Create 2 inner spans to hold the current and additional information to be shown on button hover

2. Style the initial and hover states of the button

3. Create a transition between initial and hover states

Creating Spans Inside the Button

HTML

```
<section>
  <a href='#' class='btn buy-button'>
    <span class="top content">Buy Now!</span>
    <span class="bottom content">On Sale $59</span>
  </a>
</section>
```

Adventures in Web Animations

Setting Initial Position of Each Span

HTML

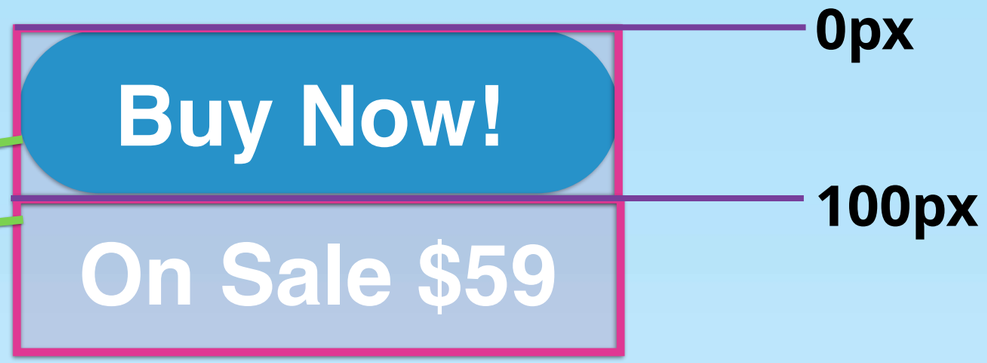
```
<section>
  <a href='#' class='btn buy-button'>
    <span class="top content">Buy Now!</span>
    <span class="bottom content">On Sale $59</span>
  </a>
</section>
```

“Absolute” means “position me relative to my nearest positioned parent or document.”

CSS

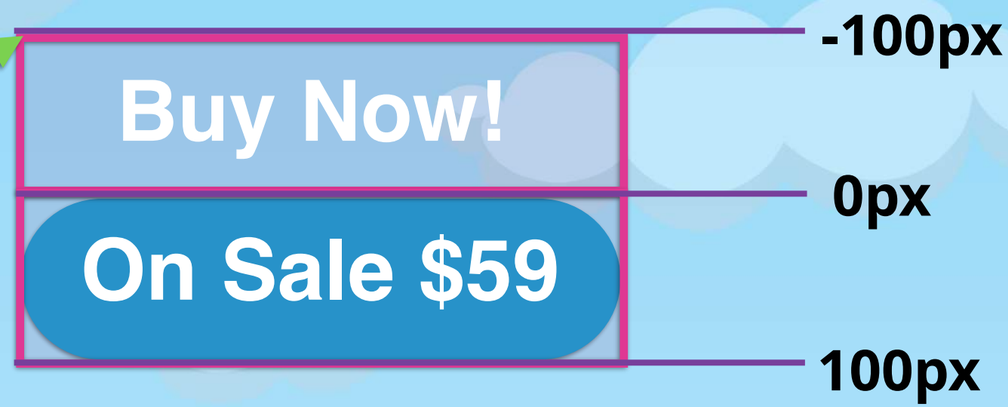
```
.btn { position: relative; }
.content { position: absolute; }

.top { top: 0;}
.bottom { top: 100px;}
```



Setting a New Position on Hover

```css
.top {top: 0px;}
.btn:hover .top {top: -100px;}
```



Move each span 100 pixels up when button is hovered.

```css
.bottom {top: 100px;}
.btn:hover .bottom {top: 0px;}
```

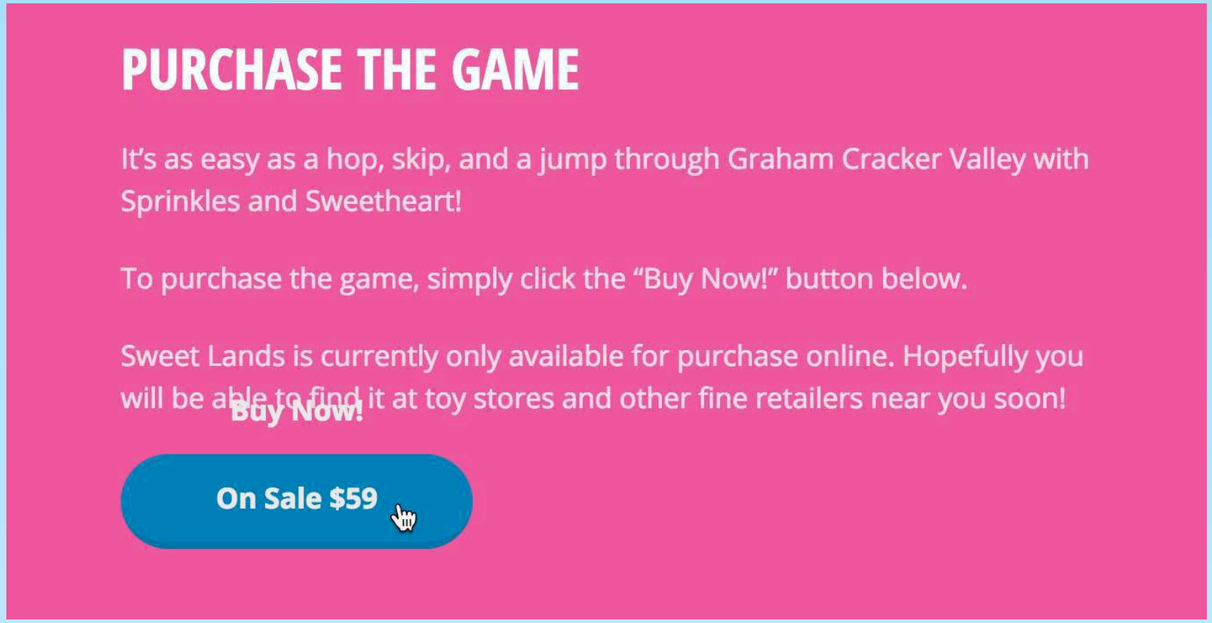
Transitioning Position

Now we will transition both top and bottom position properties to slide both content divs up.

Position is not on the list of animatable properties, so we transition top, bottom, right, left, or all.

CSS

```css
.top {top: 0px;}
.btn:hover .top {top: -100px;}
.bottom {top: 100px;}
.btn:hover .bottom {top: 0px;}
.content {
  position: relative;
  transition: top 0.3s;
}
```



Need to hide the overflowing text

Hiding Content Overflowing the Button

CSS

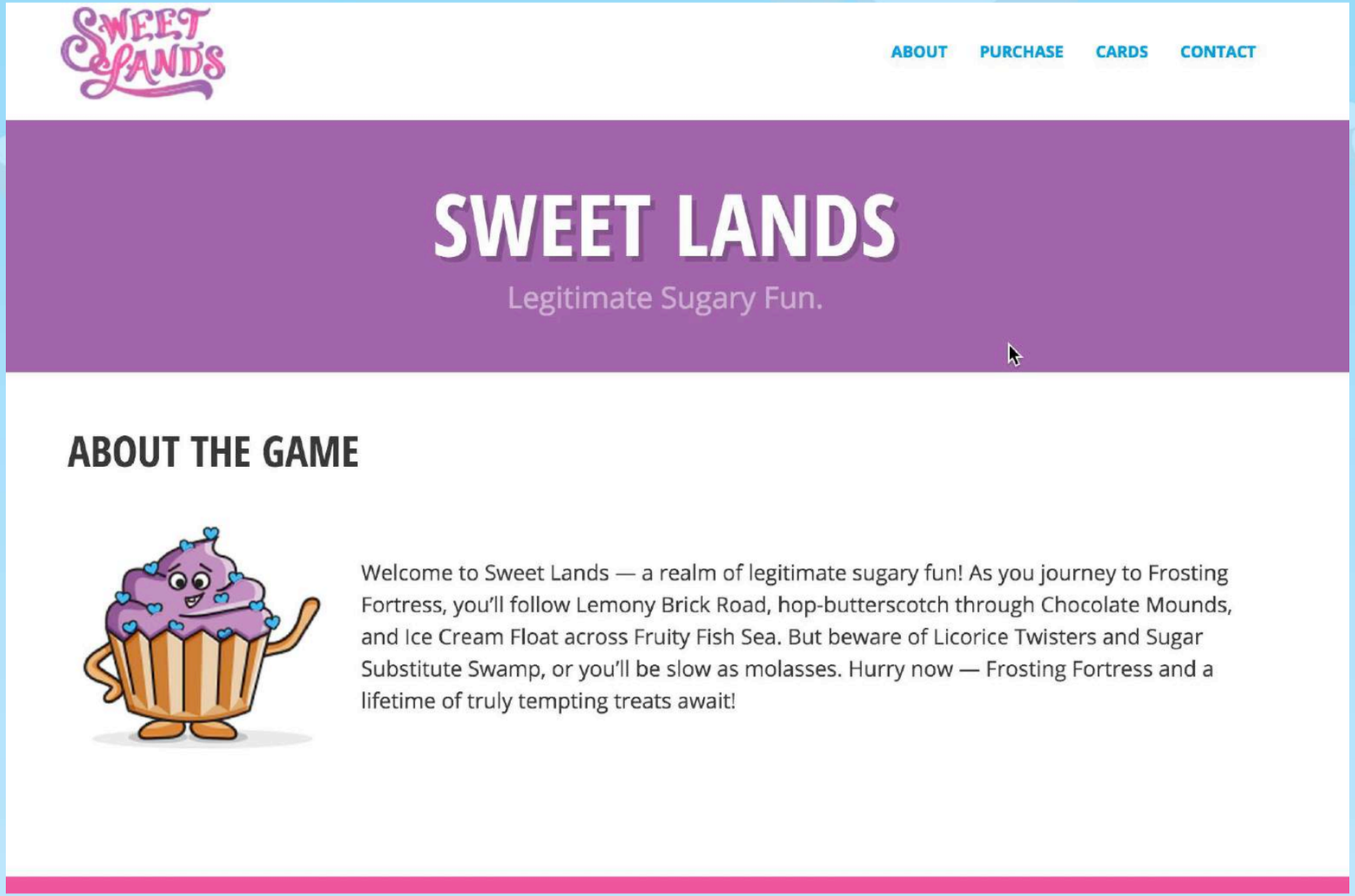
```css
.content {
  position: relative;
  transition: top 0.3s;
  overflow: hidden;
}
```

Level 1 – Transitions

SECTION 3

Transitioning Visibility

Current Application



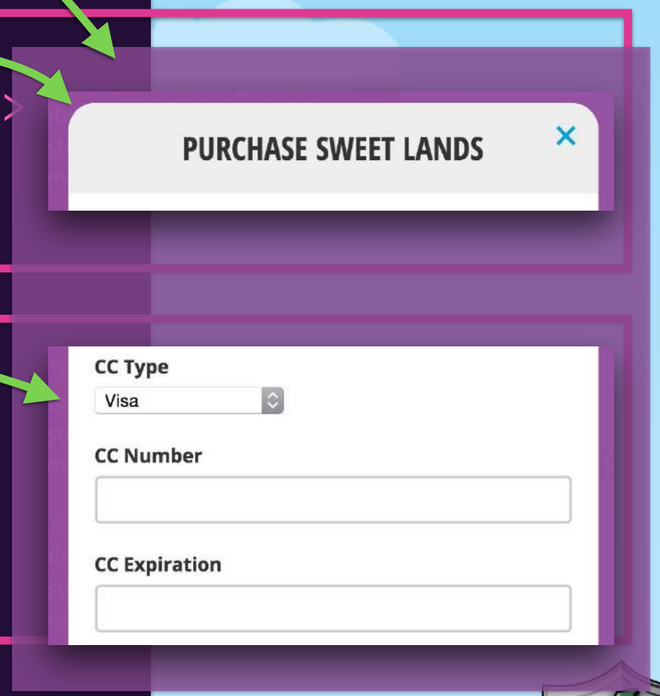
Creating the Form and Overlay

Here is the form we are going to display on button click.

```
<div class='modal-overlay'></div>

<div class='modal'>
  <div class='modal-header'>
    <a class='modal-close'>&times;</a>
    <h3>Purchase Sweet Lands</h3>
  </div>

  <div class='modal-content'>
    <form class='form' action=''>
      ...
    </form>
  </div>
</div>
```

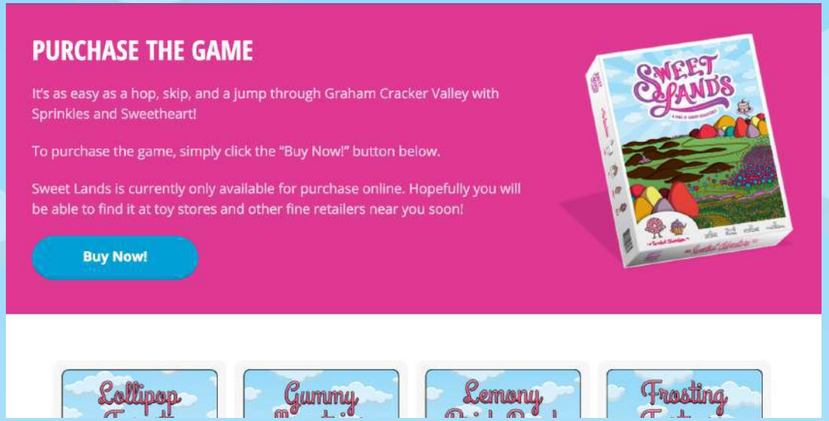


Setting the Initial and Active Modal Styles

Initial Modal State: Hidden

CSS

```css
.modal,
.modal-overlay {
  visibility: hidden;
  opacity: 0;
}
```

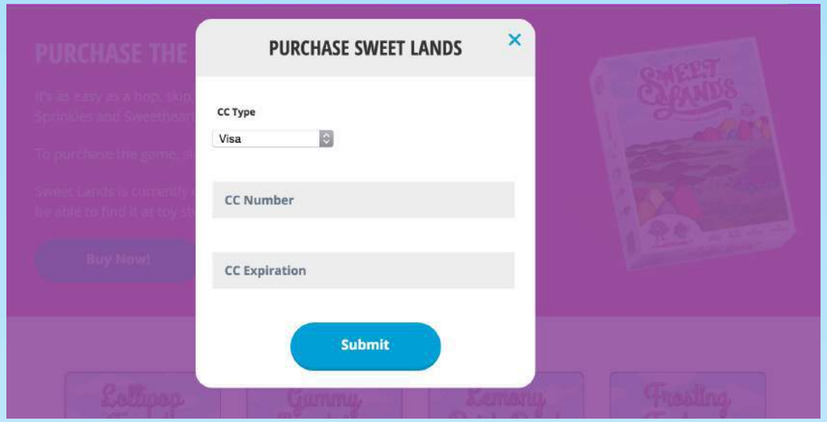


Active Modal State: Visible

CSS

```css
.modal.active,
.modal-overlay.active {
  opacity: 1;
  visibility: visible;
}
```

.active class is added when the button is clicked



Not All Properties Can Be Transitioned

`opacity: 0;` + `visibility: hidden;` =

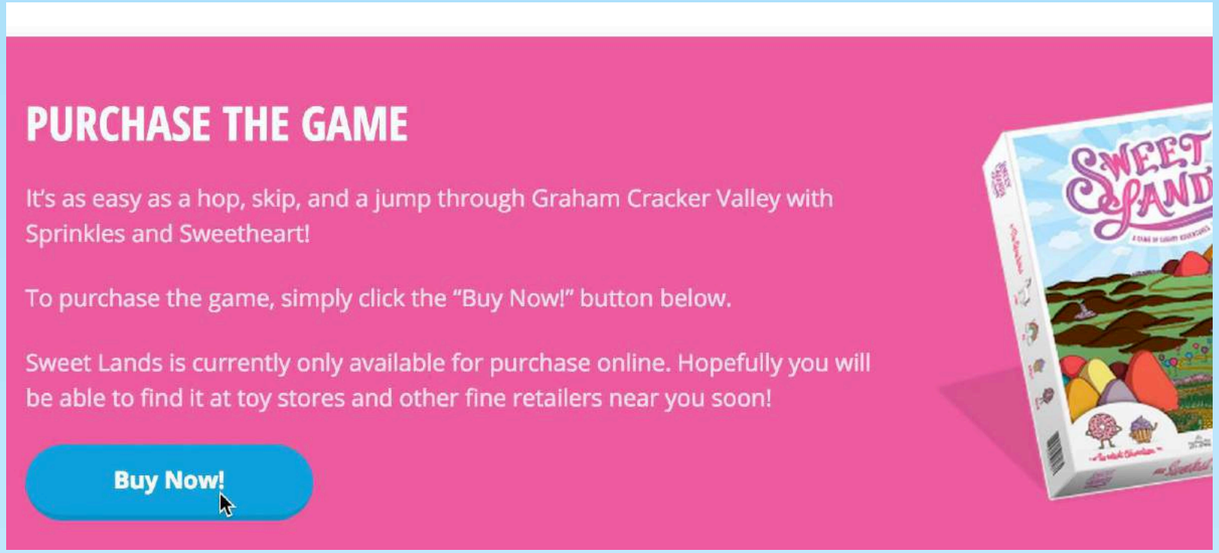
Hides element — element still takes up same width/height.

Makes element transparent to click events

when transitioning, use these instead of display

`display: none;`

Removes element from DOM — does not transition.

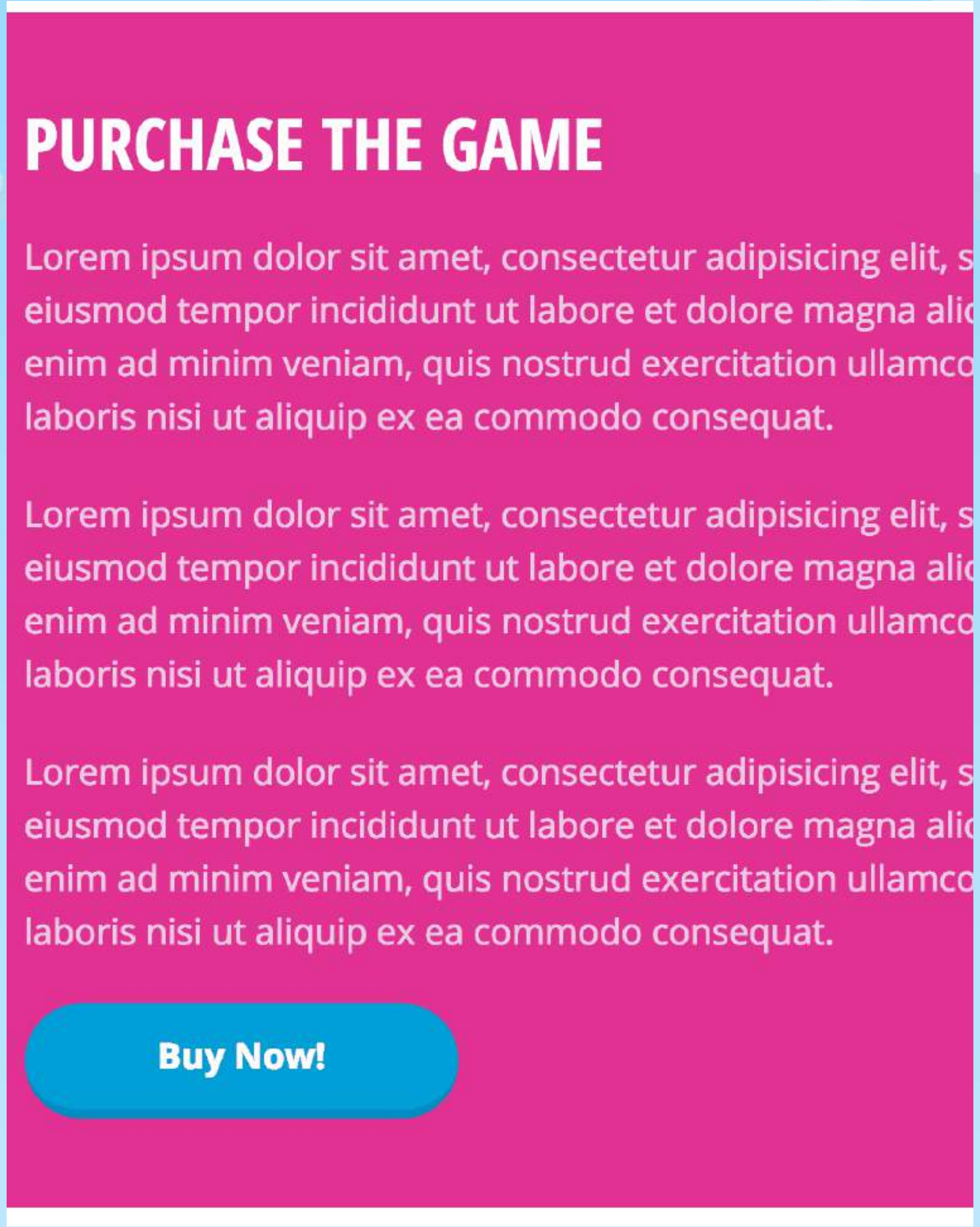


Transitioning Opacity

On button click, the form and overlay don't simply appear, but transition in nicely!

CSS

```css
.modal,
.modal-overlay {
  visibility: hidden;
  opacity: 0;
  transition: opacity .5s;
}
```



But why is it disappearing right away? What happened to the fade out?

Visibility Is Not Transitioning Out

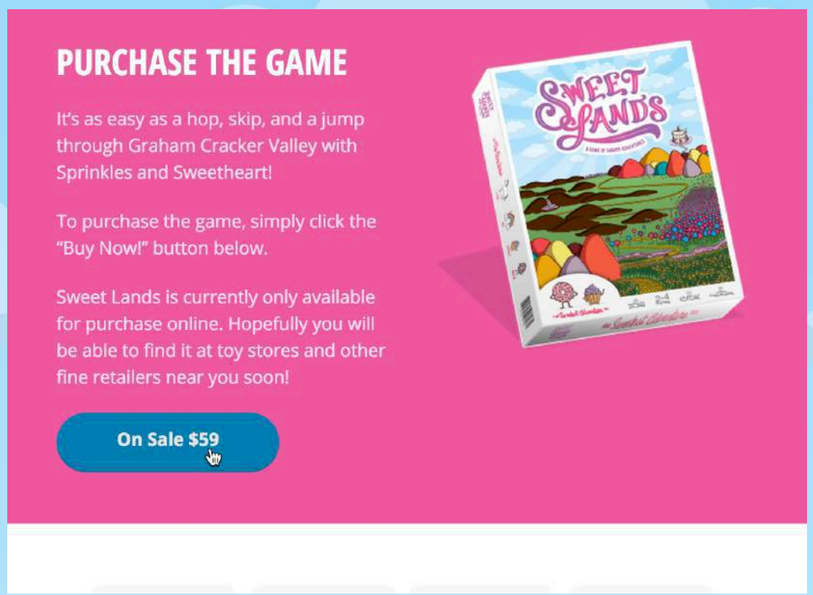
We need to also transition the visibility property.

CSS

```css
.modal,
.modal-overlay {
  visibility: hidden;
  opacity: 0;
  transition: opacity .5s, visibility .5s;
}
```

we can also just specify all

```css
transition: all .5s;
```



ADVENTURES IN WEB ANIMATIONS

How Can We Tell If a Property Can Be Transitioned?

Good rule of thumb: Ask, “Does it have a middle transitioning state?”

	start state	transitioning	end state
✓ **opacity**	0	0.5	1
✗ **display**	none	no possible middle state	block

what would that even be?
display: sort-of?

Properties That Can Be Transitioned

background-color	margin-bottom	padding-bottom
background-position	margin-left	padding-left
border-left-color	margin-right	padding-right
border-left-width	margin-top	padding-top
border-spacing	max-height	right
color	max-width	text-indent
font-size	min-height	text-shadow
font-weight	min-width	vertical-align
left	opacity	visibility
letter-spacing	outline-color	word-spacing
line-height	outline-width	z-index

They are not set in stone, so always check the full list of **properties that can be transitioned:**

http://go.codeschool.com/transitionable-properties

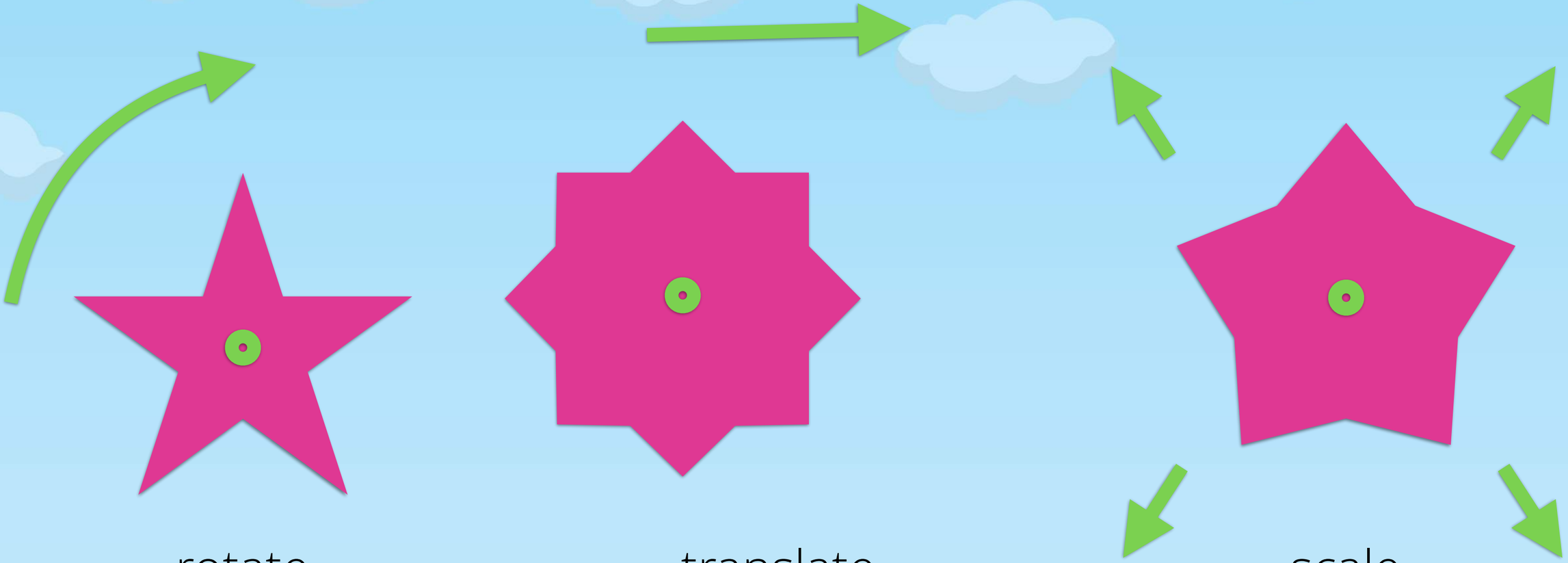
Level 2 – Transforms

SECTION 1

Transforming Rotate

Transforms

CSS transforms let you modify elements in their coordinate space. They can be rotated, translated, scaled, and skewed.

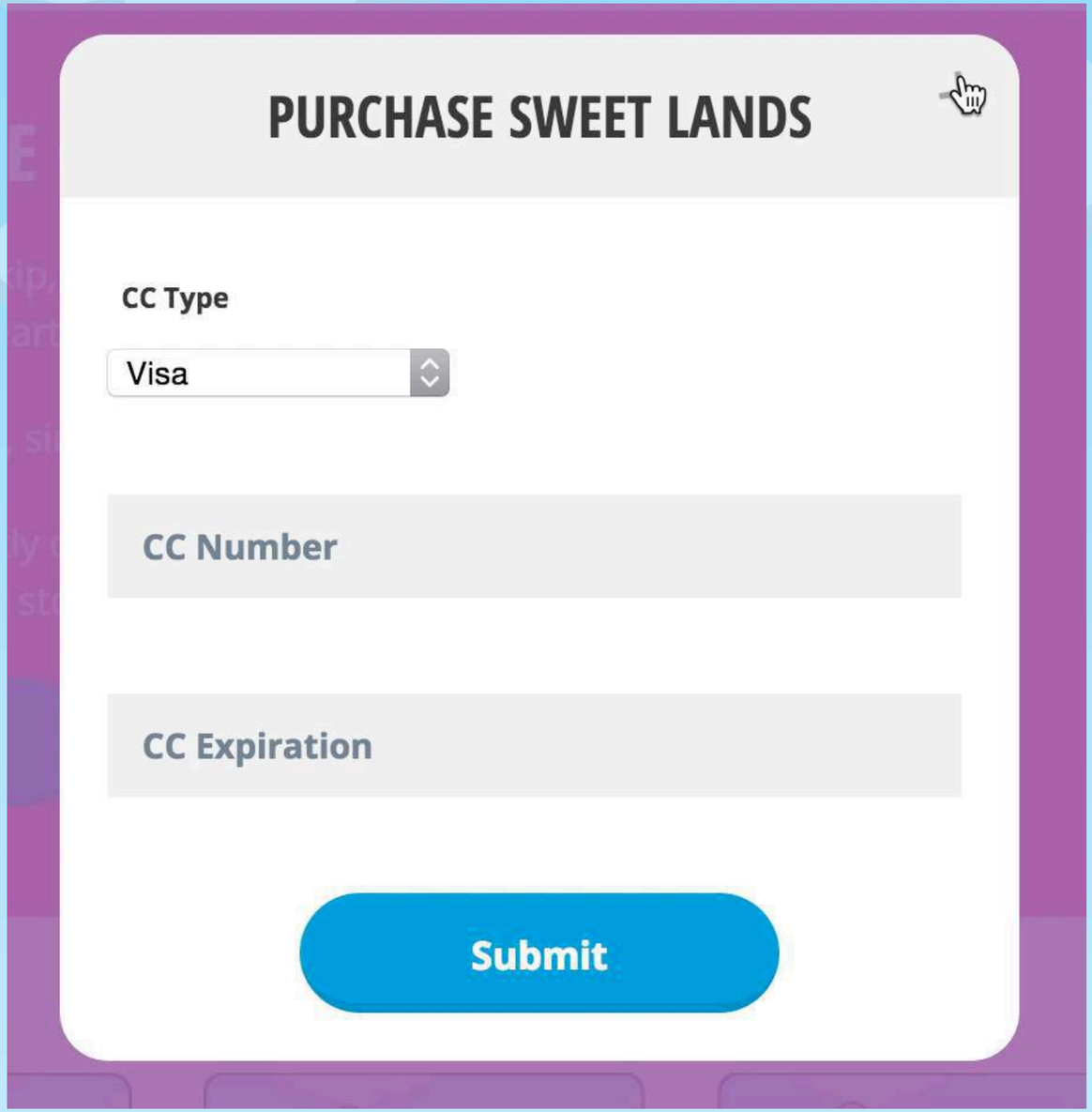


Rotating the Modal Close Icon

initial style for the X

CSS

```css
.modal-close {
  font-size: 200%;
  right: 15px;
  top: 0;
  position: absolute;
}
```

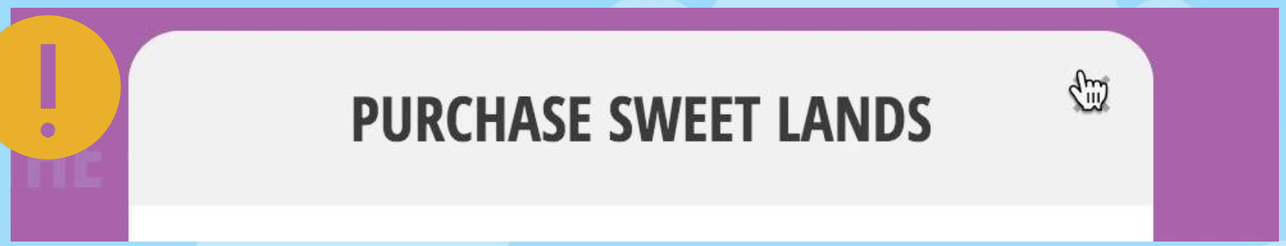


Using Transform to Rotate

CSS

```css
.modal-close {
  font-size: 200%;
  right: 15px;
  top: 0;
  position: absolute;
}

.modal-close:hover {
  transform: rotate(360deg);
}
```



The transformation is jumping from the start state immediately to the end state.

Rotate takes any number value with “deg” or “turn” unit suffix.

```css
  transform: rotate(1turn);
```

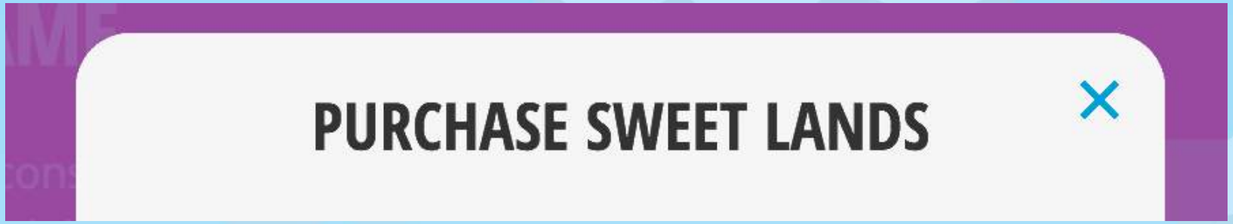
Transitioning the Transform Rotate

CSS

```css
.modal-close {
  font-size: 200%;
  right: 15px;
  top: 0;
  position: absolute;
  transition: transform 4s;
}

.modal-close:hover {
  transform: rotate(360deg);
}
```

Adding `transition: transform` will allow us to see the icon changing state over time:

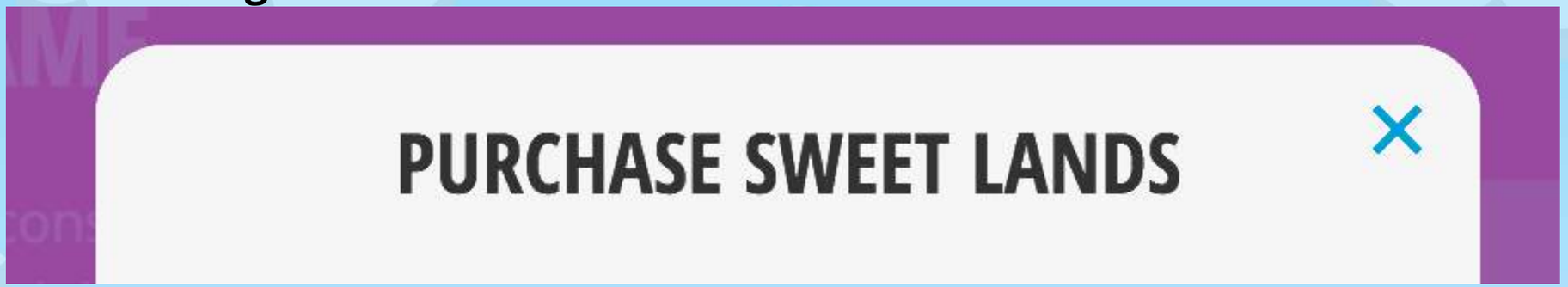


Changing the Timing Function

```
transition: transform 4s;
```

Default timing function is ease.

Ease Timing Function




```
transition: transform 4s ease-out;
```

See the subtle change a different timing function like ease-out can make?

Ease-out Timing Function

Timing Functions

```
ease
linear
ease-in
ease-out
initial
inherit
ease-in-out
cubic-bezier
```

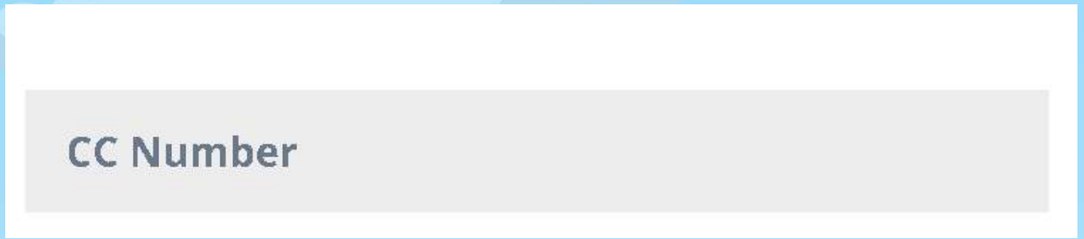
Level 2 – Transforms

SECTION 2

Transforming Scale and Translate

Creating Interactivity With Inputs

Form inputs are an excellent use of animations on the web.

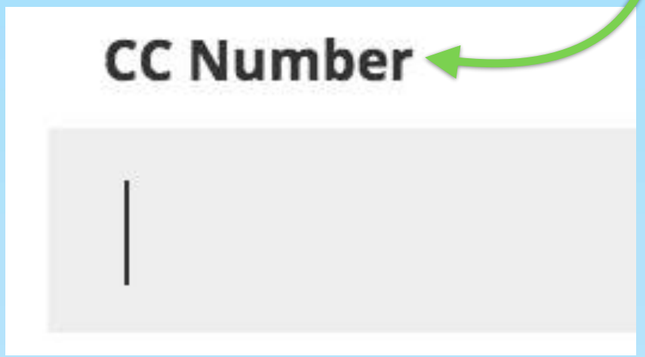


Analyzing the States of the Input Animation

If we break it down, we want our input label to have 2 different states.

We want the initial state of our label to provide information as a text placeholder.

On input:focus, we want the label to slide up and scale down, becoming your average label for an input.

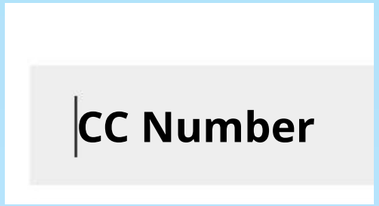


Step 1: Transition the Text Color

1. The input label text is **Transitioning Color**.

initial state

focused state



CSS

```css
.form-input + .form-label {
  position: relative;
  padding: 0 1em;
  color: #6A7989;
  transition: color 0.3s;
}

.form-input:focus + .form-label {
  color: #333333;
}
```

Only select the **first label after each input**

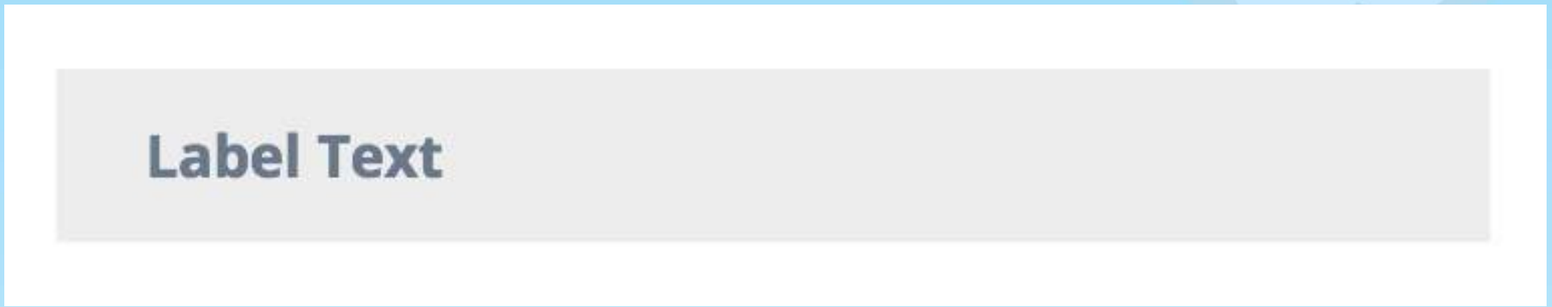
darker gray

ADVENTURES IN WEB ANIMATIONS

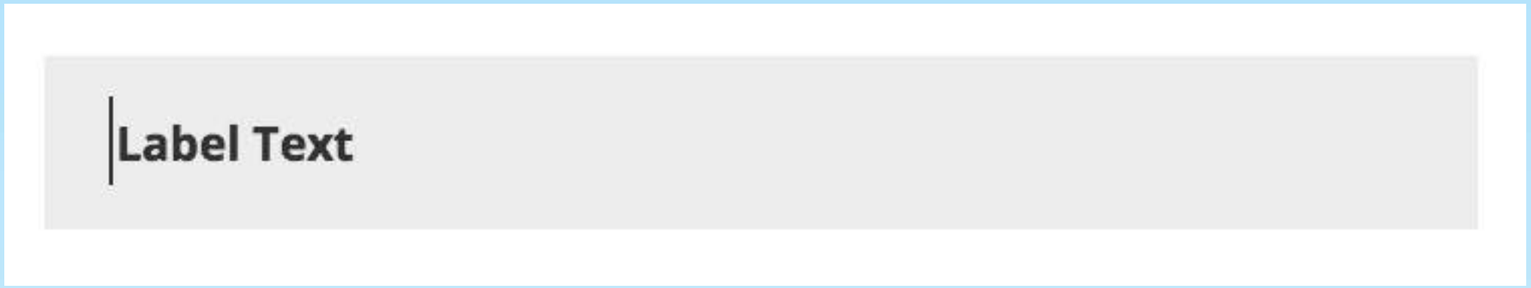
Step 2: Scaling the Label Size

1. The input label is **Transitioning Color**.
2. The input label is **Scaling Down** 80% of its original state.

initial state



focused state

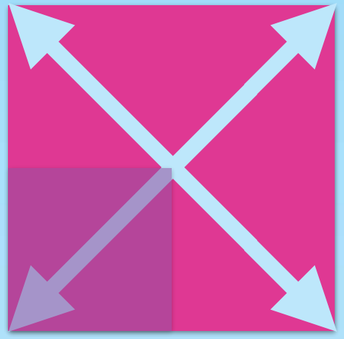


Transform Scale

Scale: to stretch an element based on the value multiplier

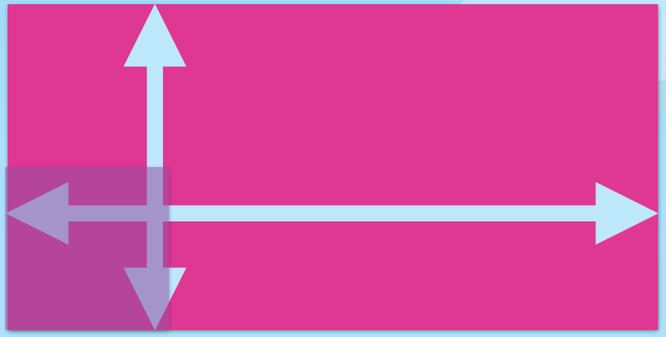
If only 1 value is provided, it will scale the element in both directions by that value.

```
transform: scale(2);
```



make it twice the size

```
transform: scale(4, 2);
```



stretch horizontally by 4, stretch vertically by 2

You can also specify the X and the Y separately:

```
transform: scaleX(value);
```

```
transform: scaleY(value);
```

Scaling Down the Label as It Moves Up

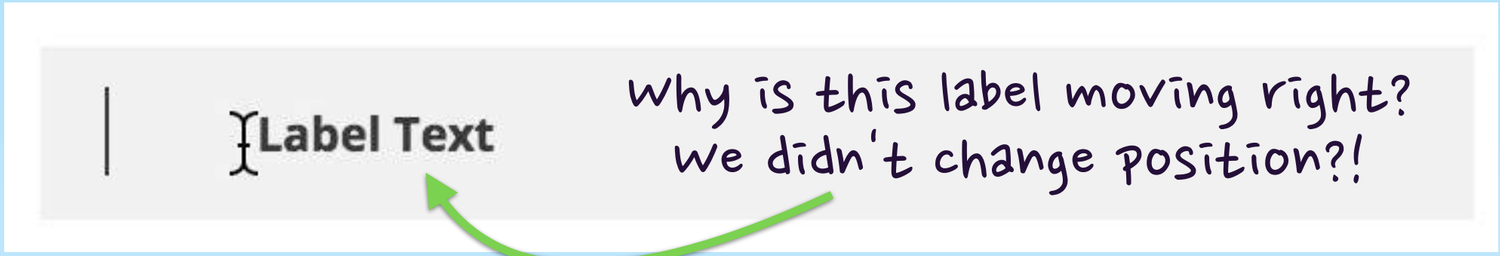
Scaling down to 80% of its original size and transitioning scale:

CSS

```css
.form-input + .form-label {
  position: relative;
  transition: color 0.3s, transform 0.3s;
}

.form-input:focus + .form-label {
  color: #333333;
  transform: scale(0.8);
}
```

transition both color and scale

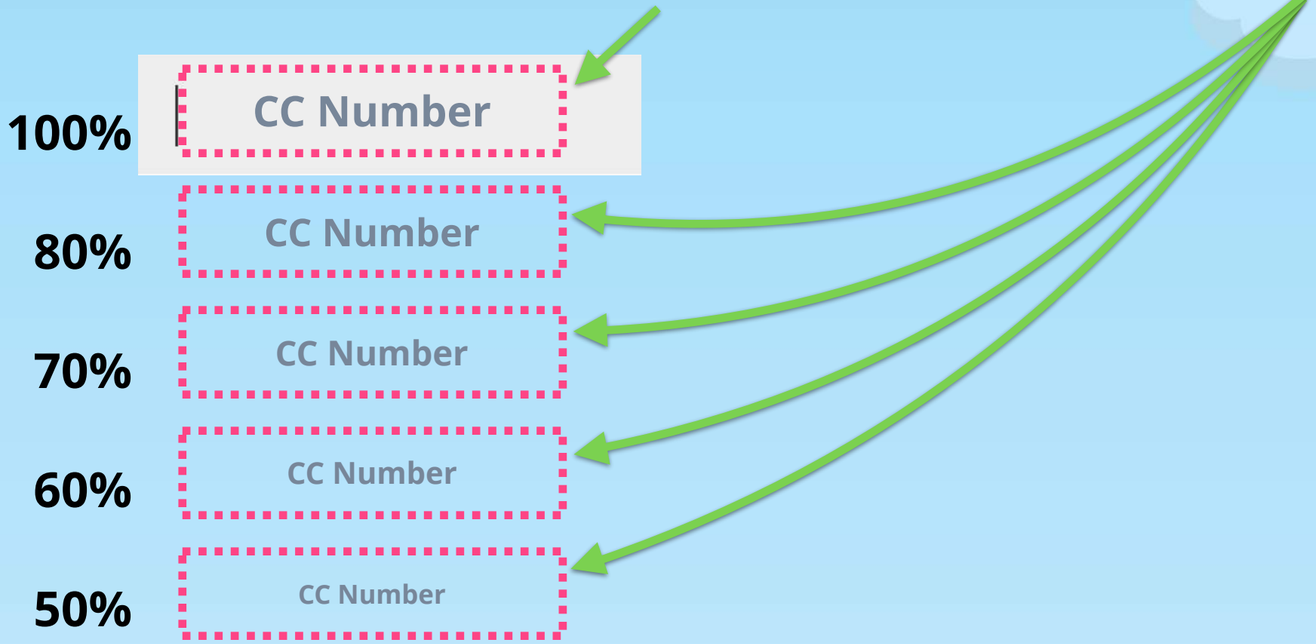


The Box Model and Scaling Gotcha

When you scale something down, it still maintains its original box model size.

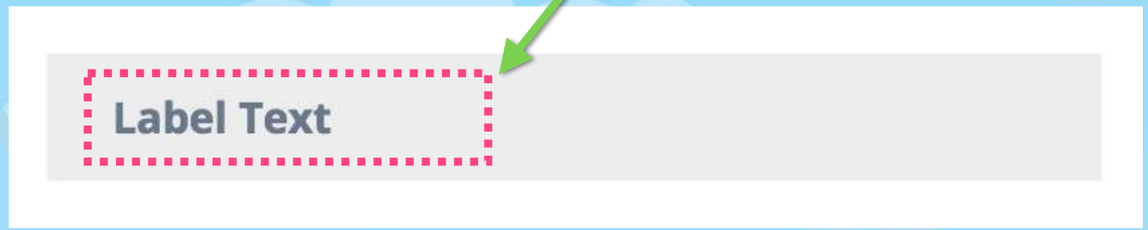
Let’s say the “CC Number” label takes up 100px height and 400px width.

No matter how small you scale, the label will still take up 400px x 100px.

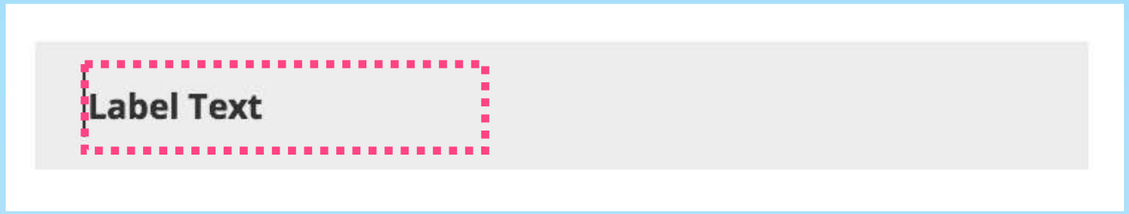


The Desired Label vs. the Current Label

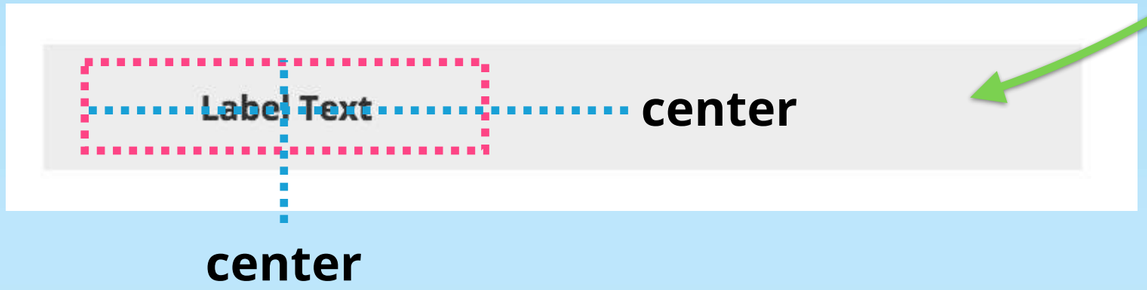
initial state



desired focused state



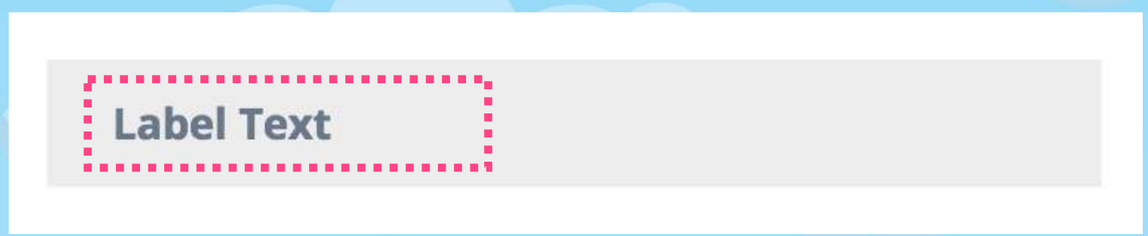
current focused state



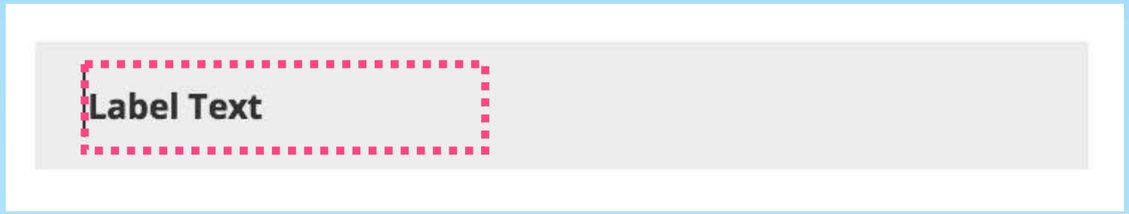
As we scale the text down inside this permanent 400x100 label, it scales around an origin of center.

The Label's Origin

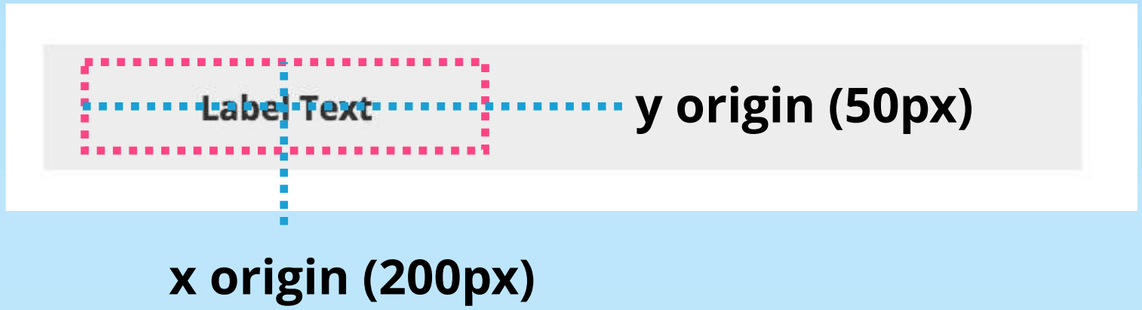
initial state



desired focused state



current focused state



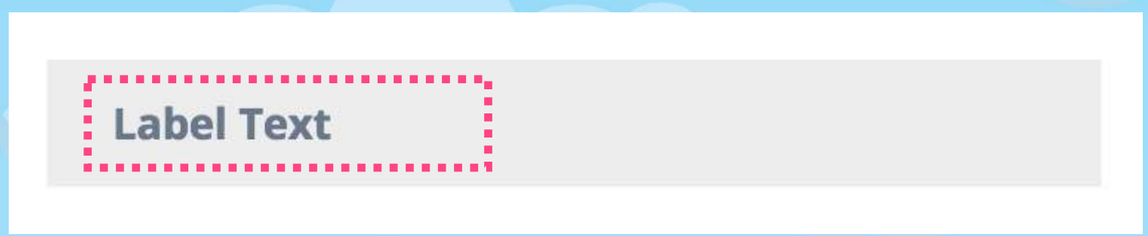
As we scale the text down, inside this permanent 400x100 label, it scales around the center (200x50).

```
transform-origin: center center;
```

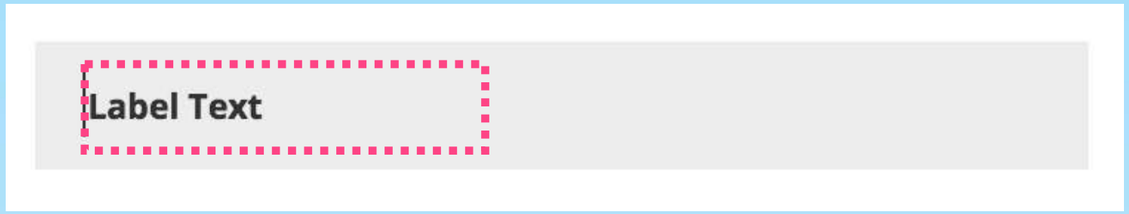
y origin **x origin**

Changing the Label's Origin

initial state

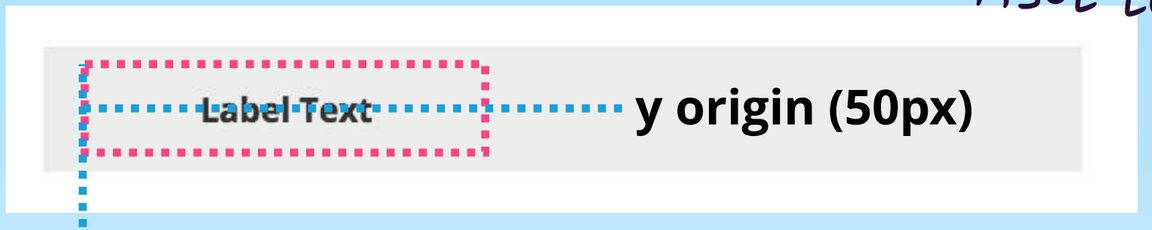


desired focused state



current focused state

changing our x offset to left



We can set this transform-origin in either keywords (center, right, top...) or in pixels.

```
transform-origin: center left;
```

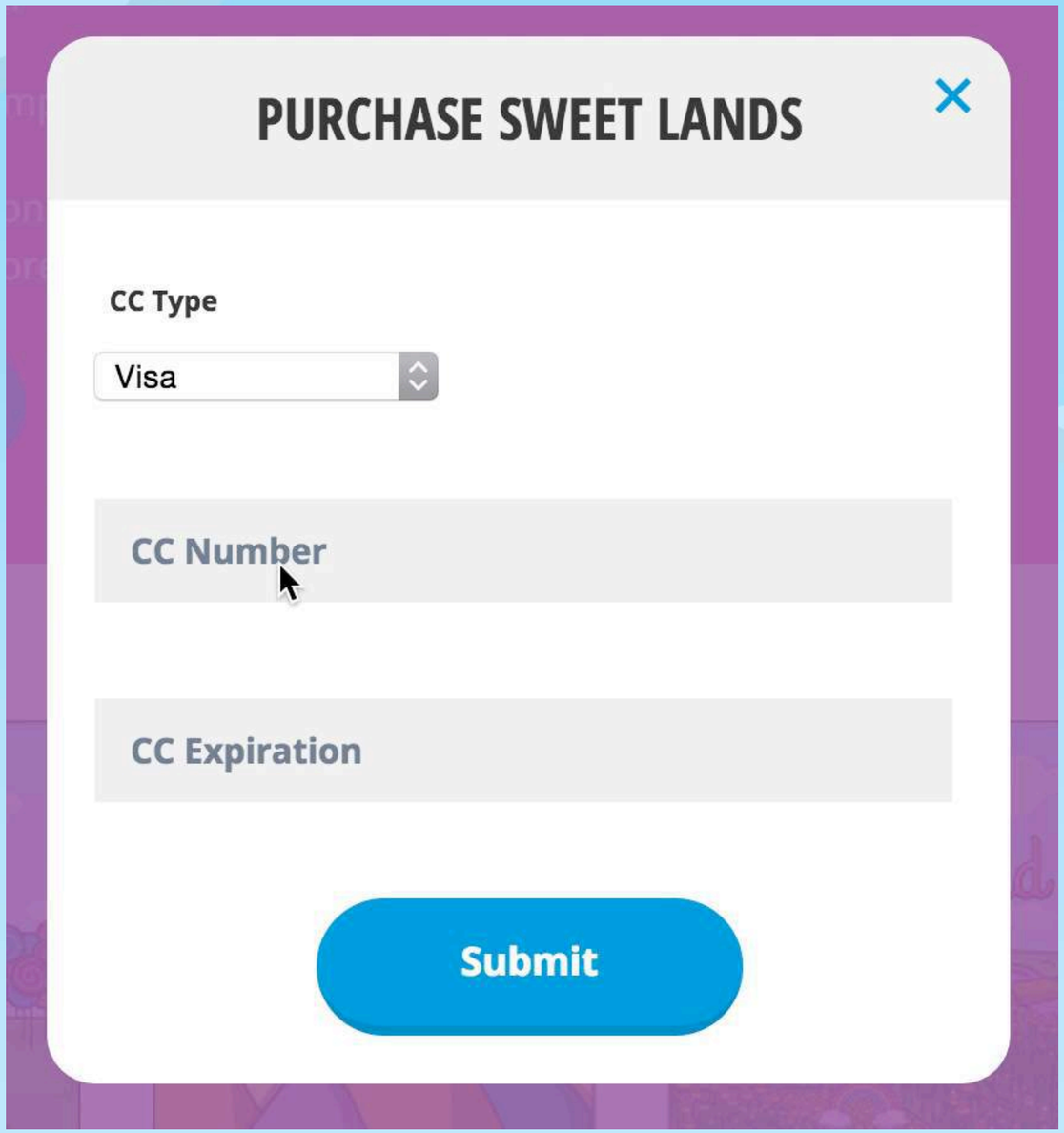
y origin **x origin**

Solution: Set Transform Origin to Center, Left

CSS

```css
.form-input + .form-label {
  position: relative;
  transform-origin: center left;
  transition: all 0.3s;
}

.form-input:focus + .form-label {
  color: #333333;
  transform: scale(0.8);
}
```



Step 3: Translate the Text Position

1. The input label is **Transitioning Color**.
2. The input label is **Scaling Down** to 80%.
3. The input label is **Translating Up** above the input.

initial state

focused state

What Is Translation?

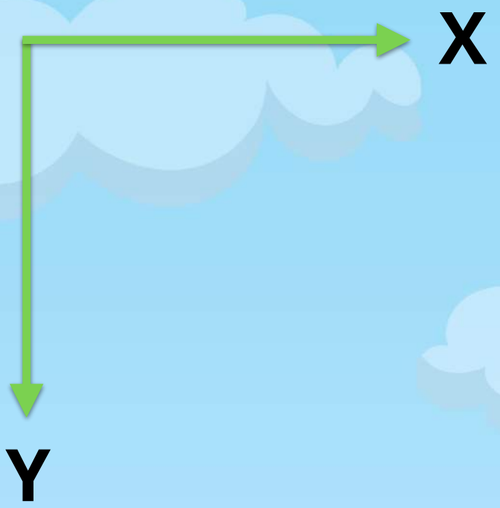
Translate simply means **to move** something.

move right 3

```
transform:translateX(3px);
```

move up 3

```
transform:translateY(-3px);
```



move right 3

move down 3

```
transform:translate(3px);
```

Moving the Label Up With TranslateY

On input:focus, the label of the input should move up out of the way.

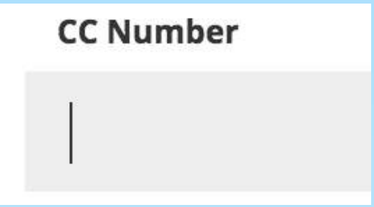
CSS

```css
.form-input + .form-label {
  position: relative;
  transition: color 0.3s, transform 0.3s;
}

.form-input:focus + .form-label {
  color: #333333;
  transform: scale(0.8) translateY(-40px);
}
```

initial state

focused state



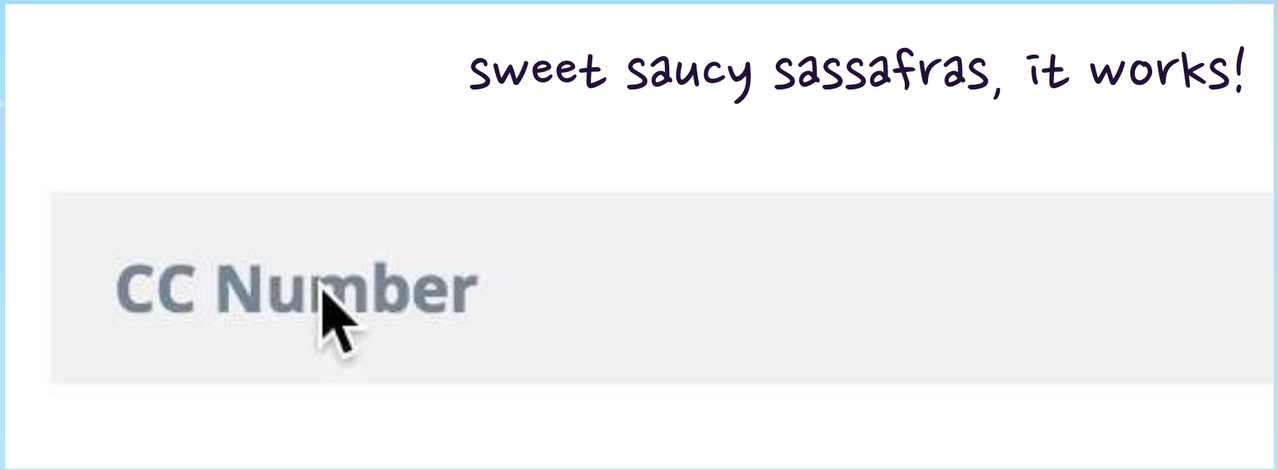
Can shorten this transition's line by using the `all` keyword.

```css
transition: all 0.3s;
```

move up 40px

Our Input Animates Properly!

1. The input label is **Transitioning Color**.
2. The input label is **Scaling Down** to 80%.
3. The input label is **Translating Up** above the input.

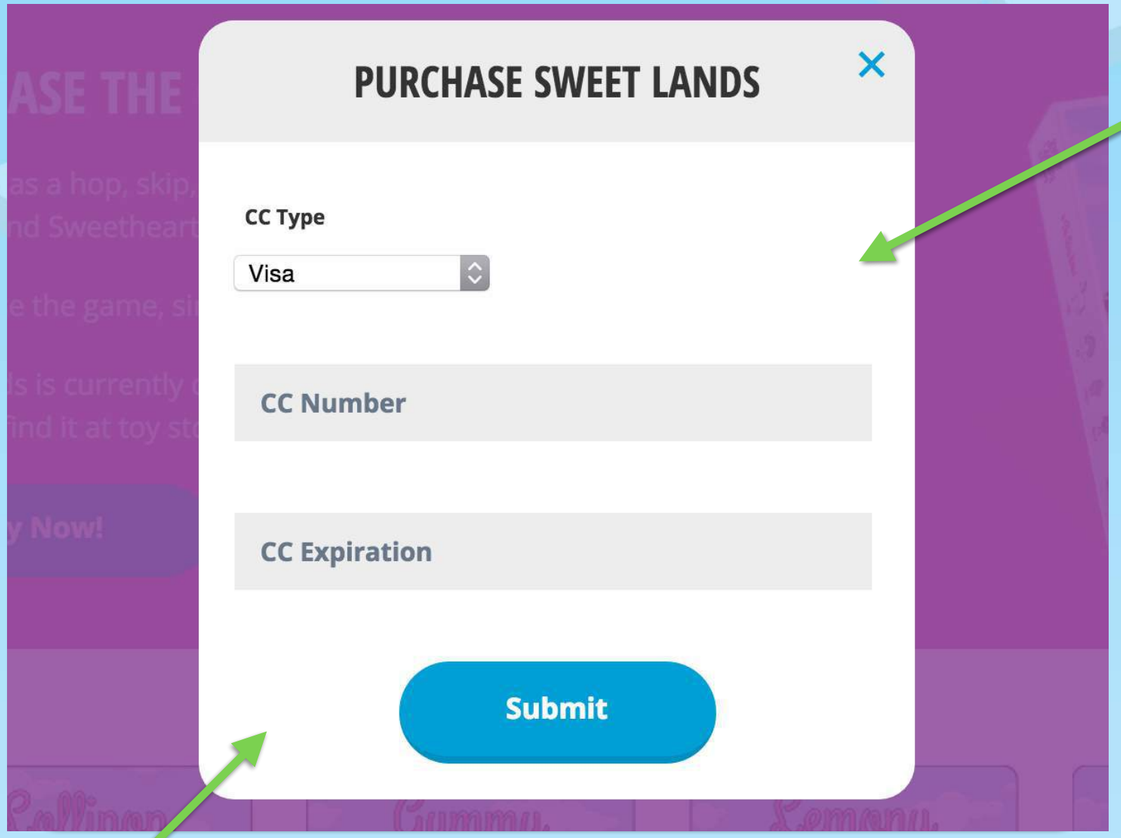


Level 3 – Keyframes

SECTION 1

Creating and Reusing Keyframes

Our Site So Far

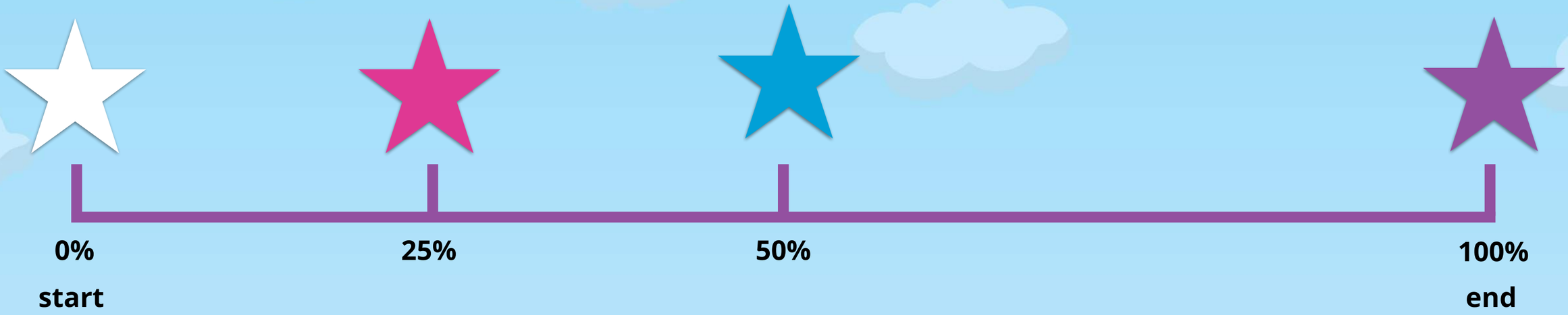


Transitions!

Transforms!

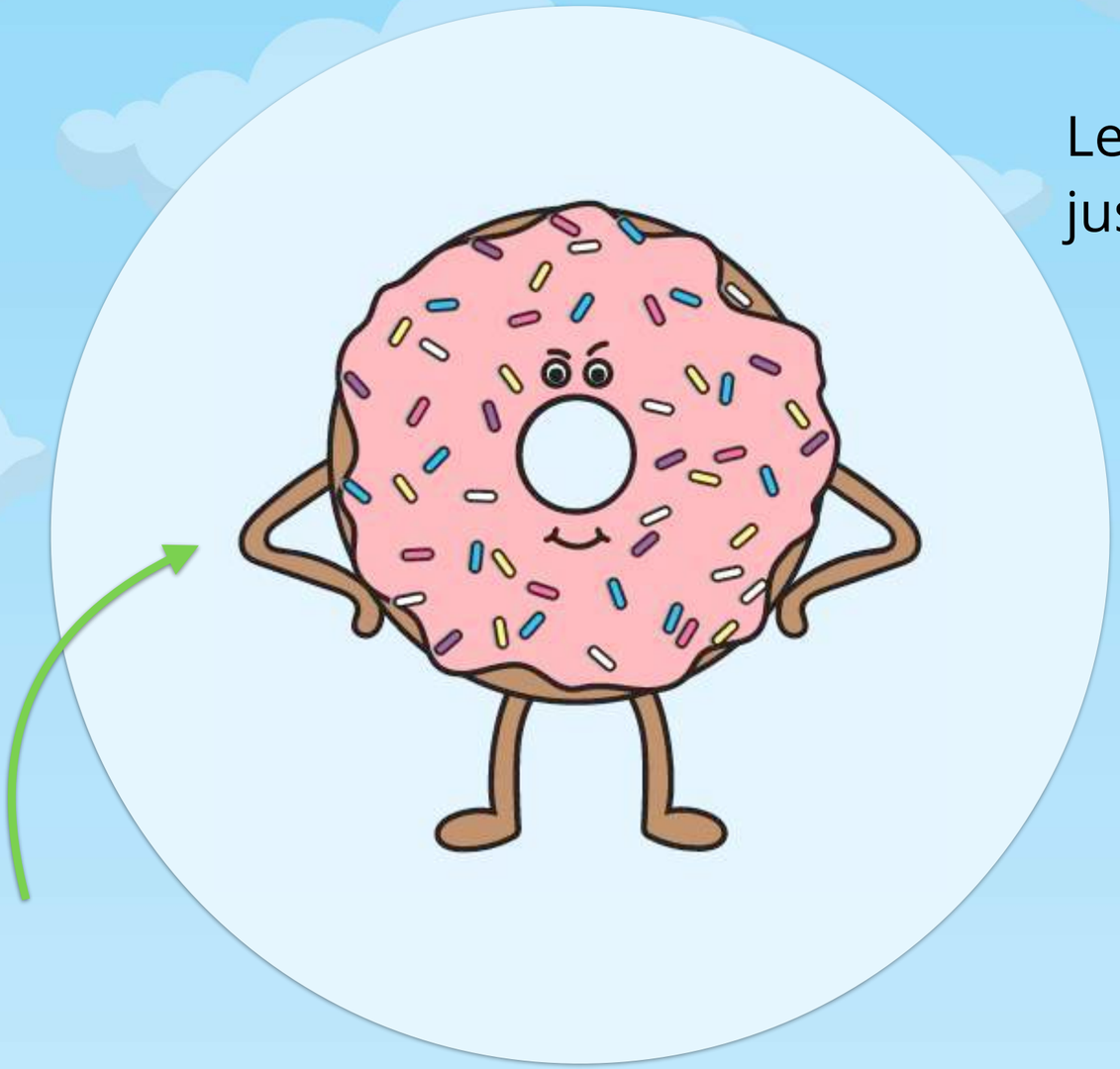
Keyframe Animations

A list of what should happen over the course of the animation — which properties should change, how, and when.



Using @keyframes to Animate Characters

Animating PNGs on your site can add a lot of personality and fun!



Let’s start animating Beau Knut by moving just his arm inside a reusable keyframe.

Creating the Swing Keyframe Animation

There are 2 parts to keyframe animations:

1. **Create the Animation**
2. **Assign the Animation**

This custom name could be 'beau-knut-arm-swing'

CSS

```
@keyframes swing {
  0% {transform: rotate(0deg);}
  100% {transform: rotate(-10deg);}
}
```

Declare the animation

keyframe recipe

```
@keyframes <name-animation> {
  <step 1> {<property>: <value>;}
  <step 2> {<property>: <value>;}
}
```

Define steps of the animation

from is a shortcut for writing **0%**

to is a shortcut for writing **100%**

Assign the Animation to an Element

There are 2 parts to keyframe animations:

1. **Create the Animation** ✅
2. **Assign the Animation**

CSS

```
@keyframes swing {
  0% {transform: rotate(0deg);}
  100% {transform: rotate(-10deg);}
}
```

Duration must come before delay!

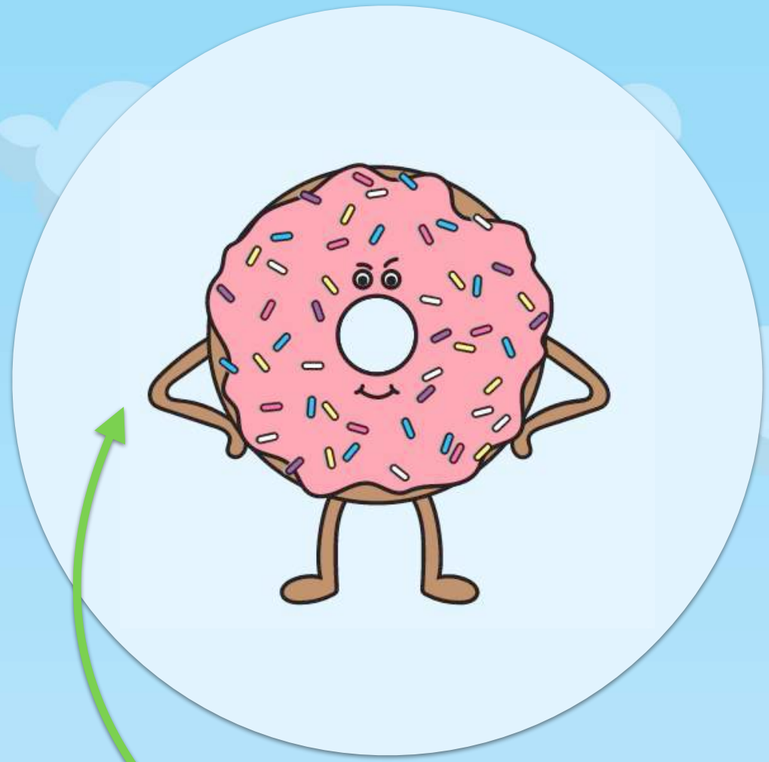
duration **delay**

CSS

```
#left-arm {
  animation: swing 2s 0s infinite ease;
}
```

name **iteration** **timing function**

how many times to run



our donut's arm looks weird!

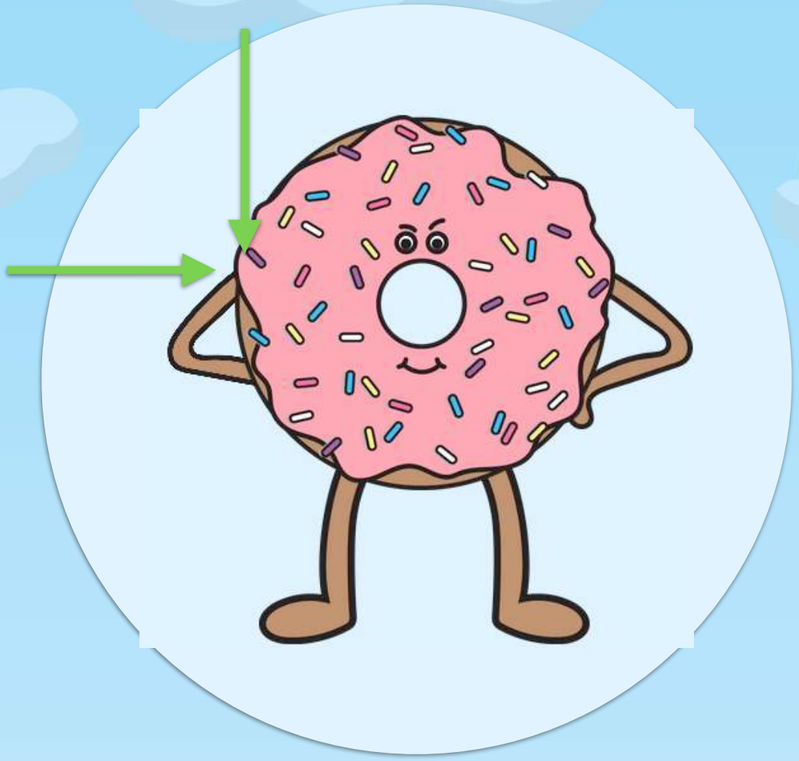
Fixing the Arm By Adjusting the Origin

Specify a different transform origin so our arm rotates properly from the top and not around the elbow.

CSS

```css
@keyframes swing {
  0% {transform: rotate(0deg);}
  100% {transform: rotate(-10deg);}
}

#left-arm {
  transform-origin: top center;
  animation: swing 2s infinite;
}
```



Reuse the Swing Animation Keyframe

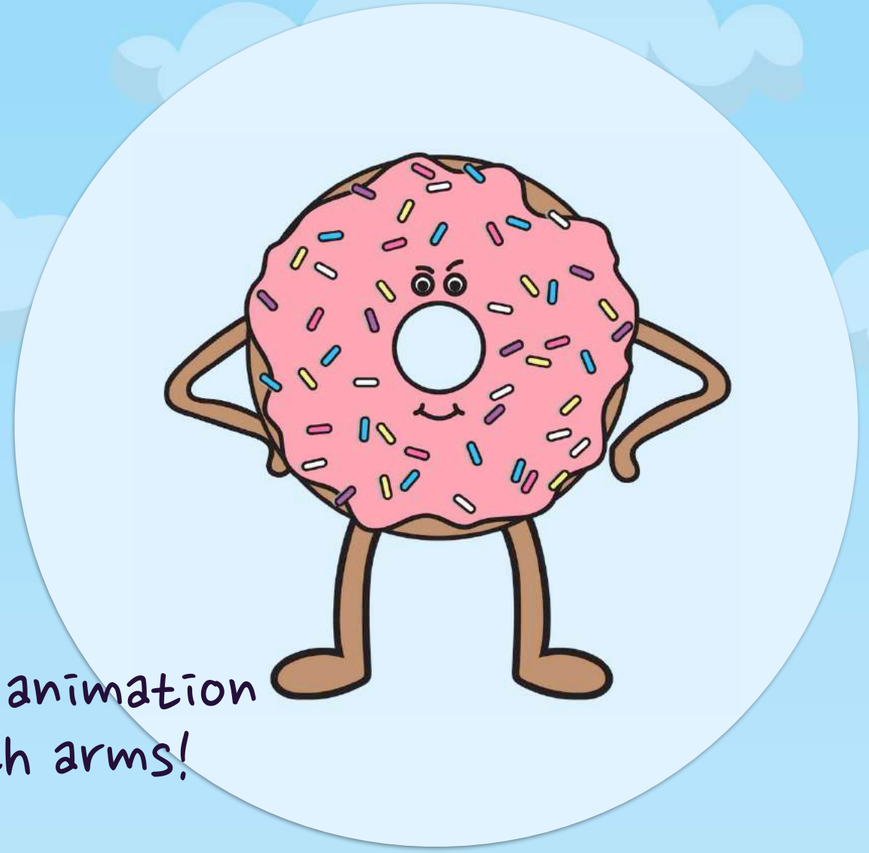
Now let's animate the right arm by reusing our swing animation.

CSS

```css
@keyframes swing {
  0% {transform: rotate(0deg);}
  100% {transform: rotate(-10deg);}
}

#left-arm {
  transform-origin: top center;
  animation: swing 2s infinite;
}

#right-arm {
  transform-origin: top center;
  animation: swing 2s infinite;
}
```



Level 3 – Keyframes

SECTION 2

Multi-step Keyframes

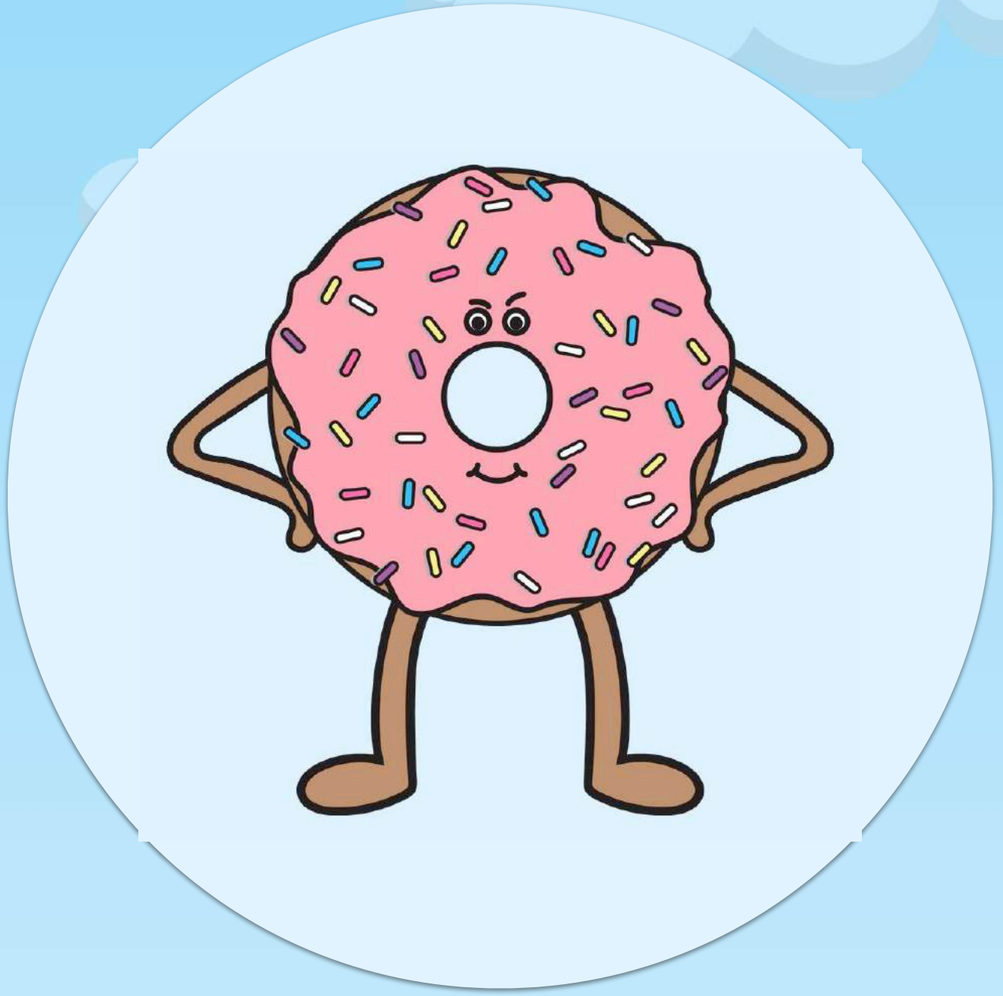
Adding More Steps to the Animation

Our donut's arm swing doesn't feel right with only 2 steps, though.

Let's add more than 2 steps to make the arm swing look more natural.

CSS

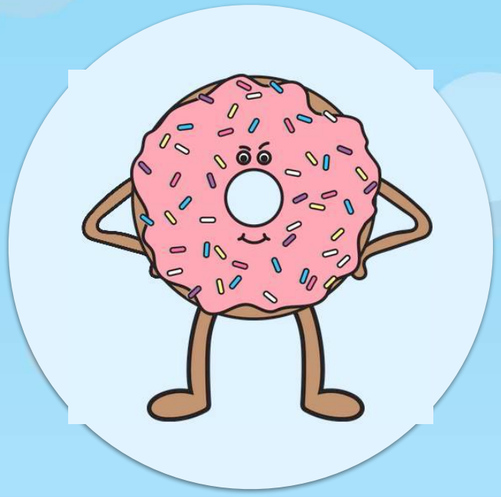
```css
@keyframes swing {
  0%   {transform: rotate(0deg);}
  25%  {transform: rotate(-10deg);}
  50%  {transform: rotate(0deg);}
  75%  {transform: rotate(10deg);}
  100% {transform: rotate(0deg);}
}
```

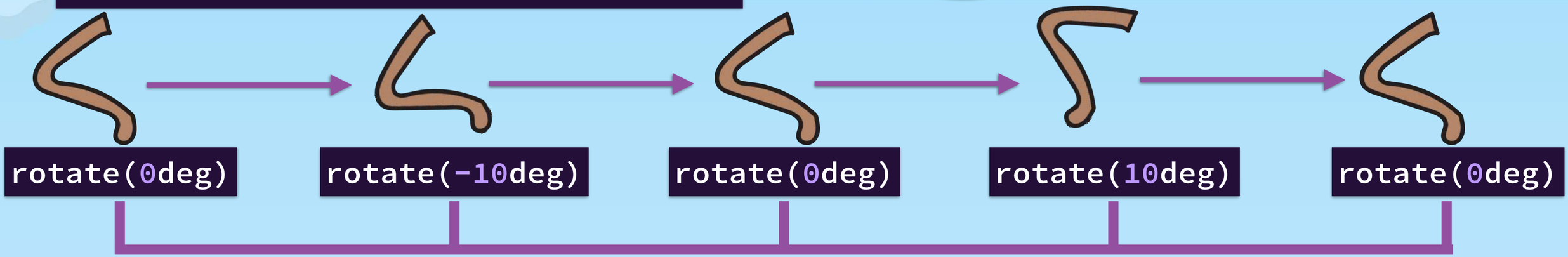


An Even, Linear Spacing of Steps

CSS

```
@keyframes swing {
  0%   {transform: rotate(0deg);}
  25%  {transform: rotate(-10deg);}
  50%  {transform: rotate(0deg);}
  75%  {transform: rotate(10deg);}
  100% {transform: rotate(0deg);}
}
```

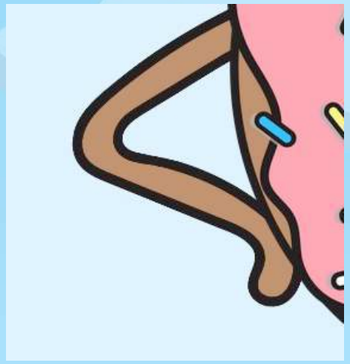




full duration is 2 seconds

An Ease-out Spacing of Steps

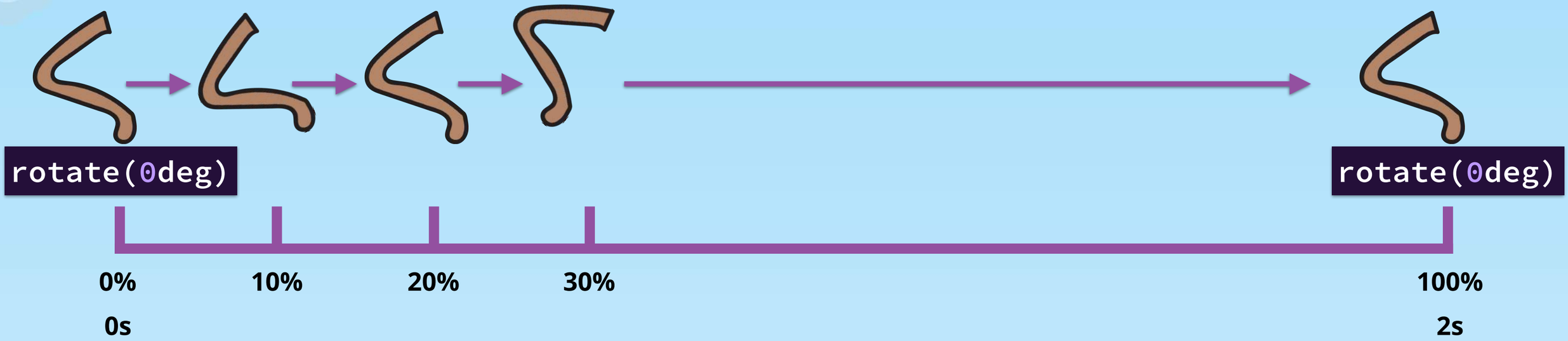
You can space out your keyframes manually, or use a timing function to do the work.



CSS

```
@keyframes swing {
  0%   {transform: rotate(0deg);}
  10%  {transform: rotate(-10deg);}
  20%  {transform: rotate(0deg);}
  30%  {transform: rotate(10deg);}
  100% {transform: rotate(0deg);}
}
```

similar to ease-out timing function



full duration is 2 seconds

Condensing Similar Steps

CSS

```
@keyframes swing {
  0%   {transform: rotate(0deg);}
  25%  {transform: rotate(-10deg);}
  50%  {transform: rotate(0deg);}
  75%  {transform: rotate(10deg);}
  100% {transform: rotate(0deg);}
}
```

If you have any duplicate animation code...

...you can condense the duplicates to one comma-separated line.

CSS

```
@keyframes swing {
  0%, 50%, 100% {transform: rotate(0deg);}
  25% {transform: rotate(-10deg);}
  75% {transform: rotate(10deg);}
}
```

Arms Aren’t Moving in Sync

Both arms are rotating to the right and left at the same time, but we want 1 to move to the left when the other is moving to the right.

Opposites!

Two Options

1. Write 2 different swings for each arm
2. Use a delay on 1 arm to start halfway through

Adding a Delay to the Right Arm

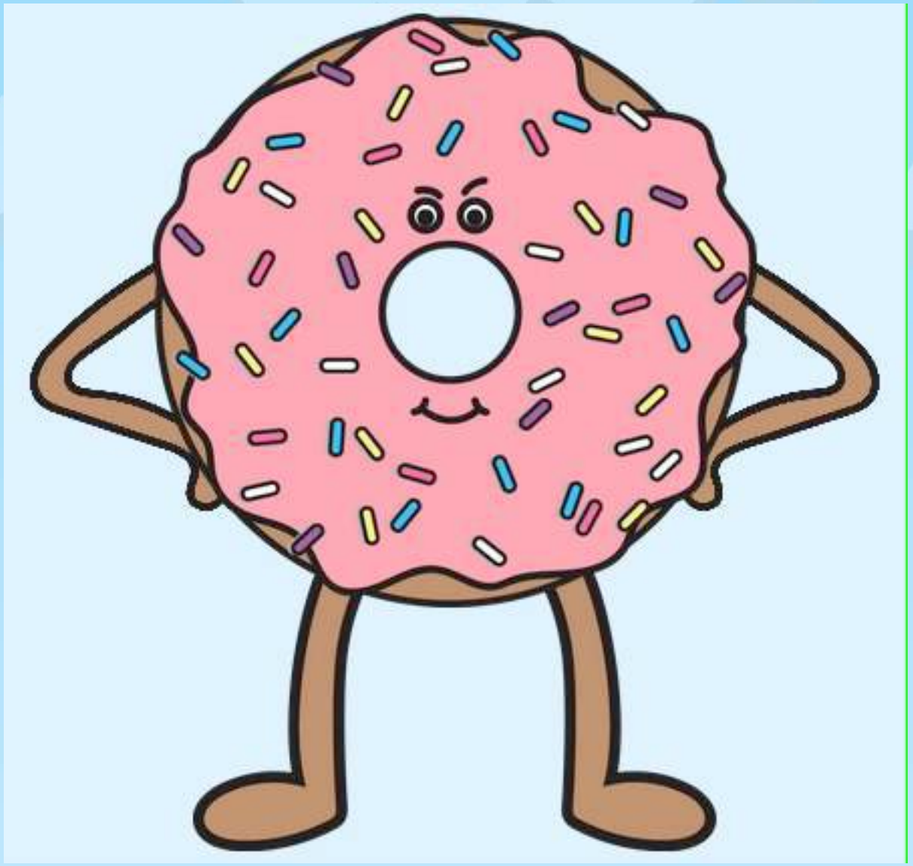
Delaying the right arm by a second will cause the swinging arms to be in sync.

CSS

```css
#left-arm{
  transform-origin: top;
  animation: swing 2s infinite linear;
}

#right-arm {
  transform-origin: top;
  animation: swing 2s infinite 1s linear;
}
```

Giving the right arm a delay will cause the arms to go in and out together.



Another Keyframe Animation

Now, let's animate the left foot.

CSS

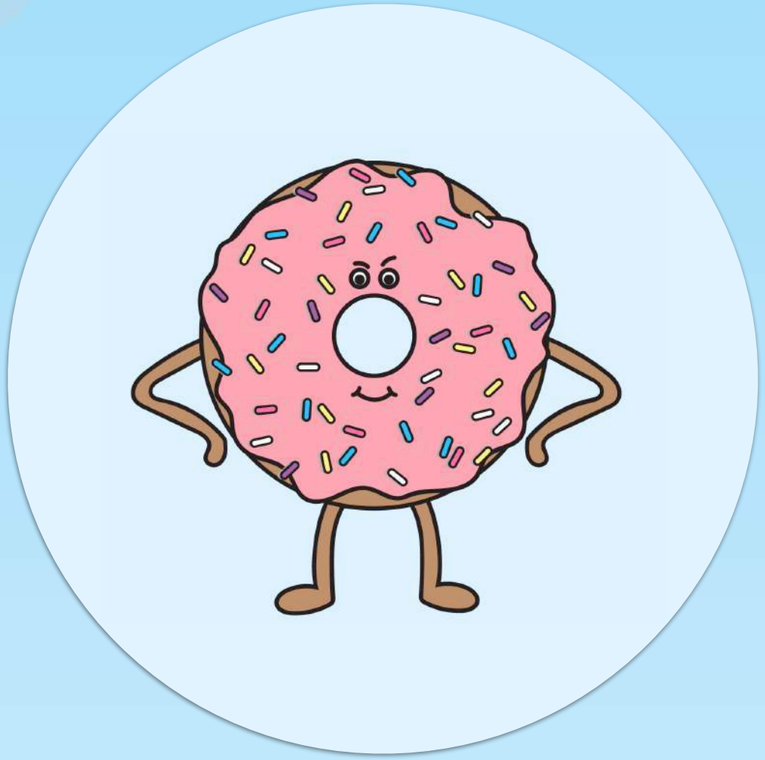
```
@keyframes swing {
  0%, 50%, 100% {transform: rotate(0deg);}
  25% {transform: rotate(-10deg);}
  75% {transform: rotate(10deg);}
}

@keyframes tap {
  0%, 100% {transform: translateY(0px);}
  50% {transform: translateY(-5px);}
}
```

We will have it loop infinite every second.

CSS

```
#left-leg {
  animation: tap 1s infinite;
}
```



We want a tapping motion that starts and ends in the same spot and slightly moves down halfway through.

Level 3 – Keyframes

SECTION 3

More Advanced Keyframe Animations

Animating Our Form With Keyframes

ABOUT THE GAME

Welcome to Sweet Lands — a realm of legitimate sugary fun! As you journey to Frosting Fortress, you'll follow Lemony Brick Road, hop-butterscotch through Chocolate Mounds, and Ice Cream Float across Fruity Fish Sea. But beware of Licorice Twisters and Sugar Substitute Swamp, or you'll be slow as molasses. Hurry now — Frosting Fortress and a lifetime of truly tempting treats await!

PURCHASE THE GAME

It's as easy as a hop, skip, and a jump through Graham Cracker Valley with Sprinkles and Sweetheart!

To purchase the game, simply click the "Buy Now!" button below.

Sweet Lands is currently only available for purchase online. Hopefully you will be able to find it at toy stores and other fine retailers near you soon!

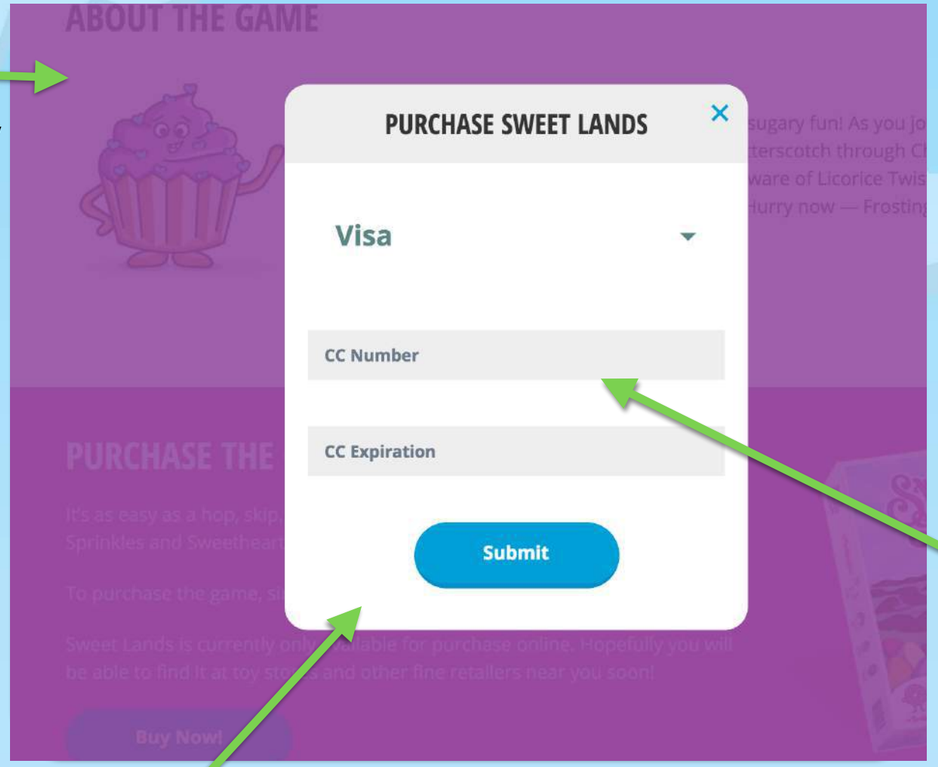
Buy Now!

What Animations Do We Need to Create?

Multiple elements need to be animated to achieve this effect.

#1

fade in the overlay



#3

slide small and fade in the elements in the modal

#2 **slide and fade in the modal**

Fading in the modal-overlay

#1

fade in the overlay

CSS

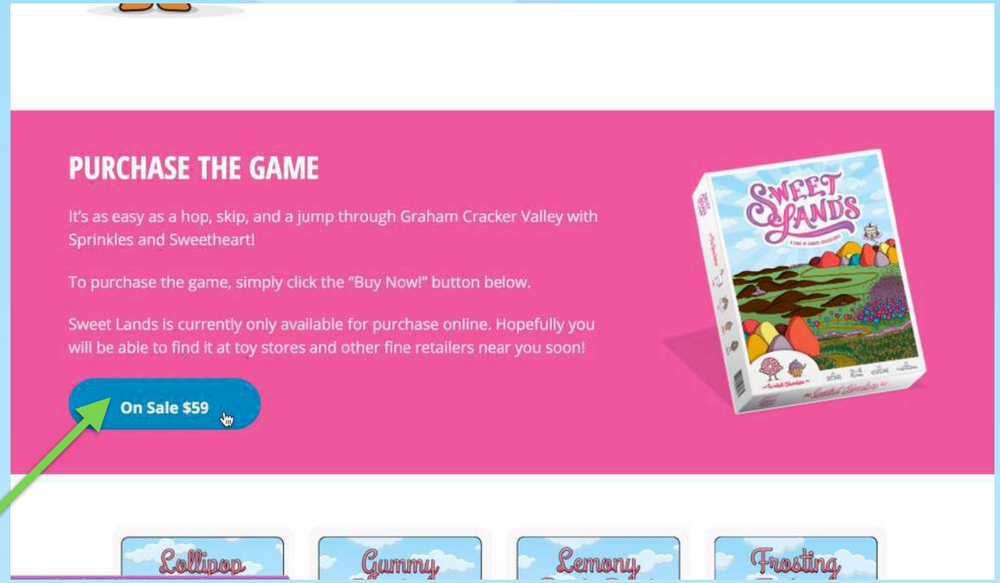
```css
@keyframes fadeIn {
  from {
    opacity: 0;
    visibility: hidden;
  }
  to {
    opacity: 1;
    visibility: visible;
  }
}
```

start hidden

end visible

CSS

```css
.modal-overlay.active {
  animation: fadeIn .25s;
}
```



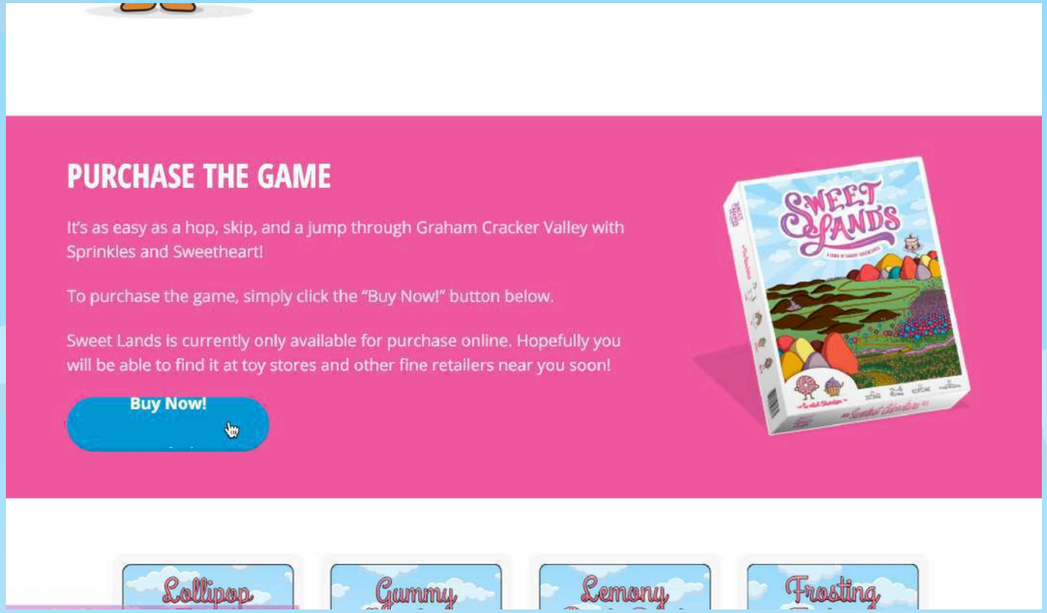
The modal is snapping back to its original state as soon as it is done.

Fill-mode Forwards

CSS

```
.modal-overlay.active {
  animation: fadeIn .25s none;
}
```

The default value for fill-mode is none.



CSS

```
.modal-overlay.active {
  animation: fadeIn .25s forwards;
}
```

Fill-mode forwards is used to set the animation's final state to the last specified step.

slideUp Keyframe for Entire Modal

CSS

```
@keyframes slideUp {
  from {
    transform: translateY(400px);
  }
  to {
    transform: translateY(-300px);
  }
}
```

start low on the page

end higher on the page

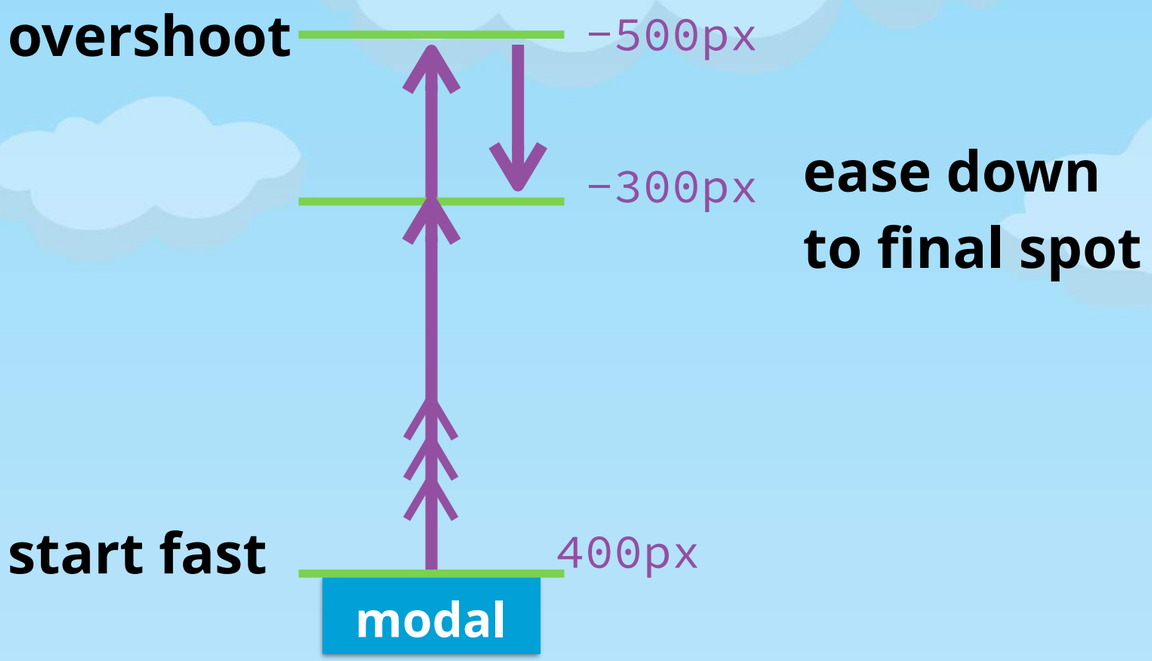
CSS

```
.modal.active {
  animation: slideUp .65s .5s forwards;
}
```

Timing Functions for the Win

When animating a single property, simply specify start/end state and use a timing function to create the desired timing.

```
@keyframes slideUp {
  from {
    transform: translateY(400px);
  }
  to {
    transform: translateY(-300px);
  }
}
```



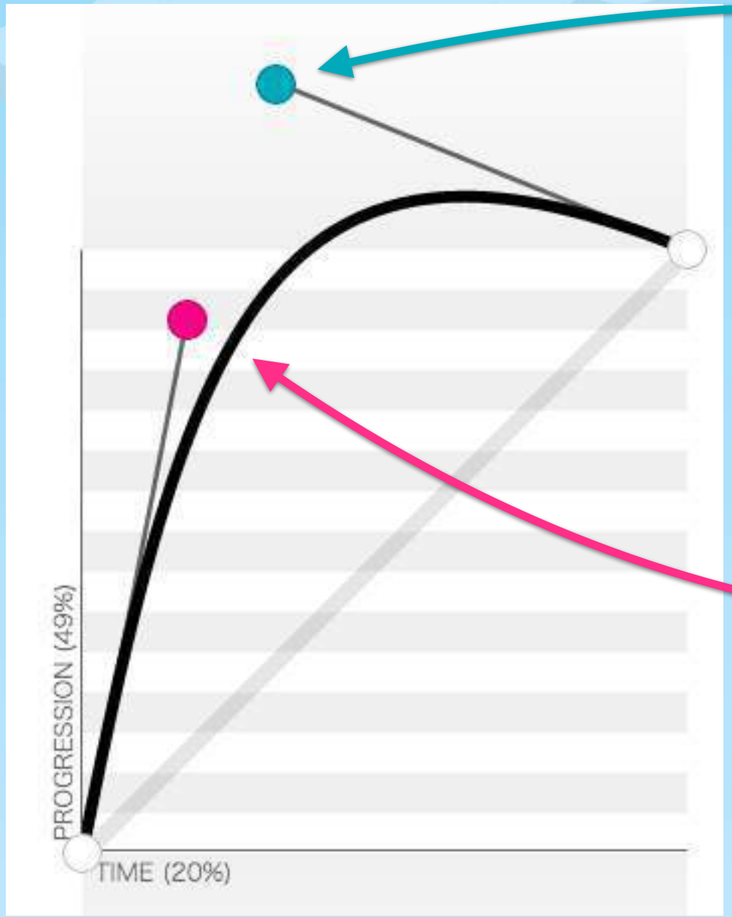
custom cubic-bezier timing function

```
.modal.active {
  animation: slideUp .65s .5s cubic-bezier(0.17, 0.89, 0.32, 1.28) forwards;
}
```

causes overshoot effect

What Is a Cubic Bezier?

WHAT ON EARTH ARE THESE NUMBERS?!



x, y

```
cubic-bezier(.17,.89,.32,1.28)
```

x, y

cubicbezier.com

All Timing Functions Are Bezier at Heart

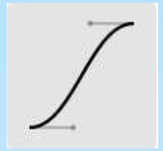
cubic Bezier curve

linear

ease

ease-in

ease-out



ease-in-out

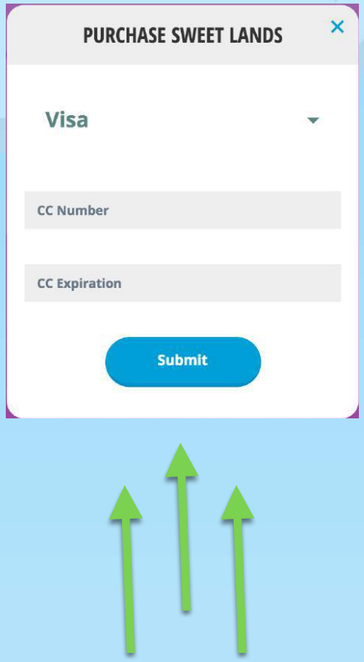
linear	cubic-bezier(0, 0, 1, 1)	Linear
ease	cubic-bezier(.25, 1, .25, 1)	Ease
ease-in	cubic-bezier(.42, 0, 1, 1)	Ease-In
ease-out	cubic-bezier(0, 0, .58, 1)	Ease-Out
ease-in-out	cubic-bezier(.42, 0, .58, 1)	Ease-In-Out

Sliding and Fading in the Modal

Assign the fadeIn animation to our .active modal as well.

CSS

```
.modal.active {
  animation: slideUp .65s .5s cubic-bezier(…) forwards,
             fadeIn .65s .5s forwards;
}
```



Finishing Form By Animating the Stuff Inside

Creating the `slideUpSmall` animation

CSS

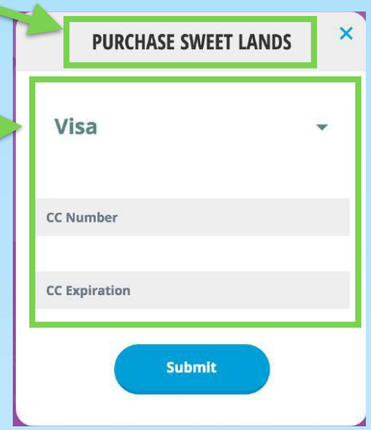
```css
@keyframes slideUpSmall {
  from {transform: translateY(80px);}
  to {transform: translateY(0);}
}
```

Assigning `slideUpSmall` and `fadeIn` to modal innards when modal is active

CSS

```css
.modal-header h3 {
  animation: slideUpSmall 0.25s 0.75s forwards,
             fadeIn 0.25s 0.75s forwards;
}

.modal.active .form-field {
  animation: slideUpSmall 0.25s 0.8s forwards,
             fadeIn 0.25s 0.8s forwards;
}
```



Our Form Looks Hot!

Welcome to Sweet Lands — a realm of legitimate sugary fun! As you journey to Frosting Fortress, you'll follow Lemony Brick Road, hop-butterscotch through Chocolate Mounds, and Ice Cream Float across Fruity Fish Sea. But beware of Licorice Twisters and Sugar Substitute Swamp, or you'll be slow as molasses. Hurry now — Frosting Fortress and a lifetime of truly tempting treats await!

PURCHASE THE GAME

It's as easy as a hop, skip, and a jump through Graham Cracker Valley with Sprinkles and Sweetheart!

To purchase the game, simply click the "Buy Now!" button below.

Sweet Lands is currently only available for purchase online. Hopefully you will be able to find it at toy stores and other fine retailers near you soon!

Buy Now!

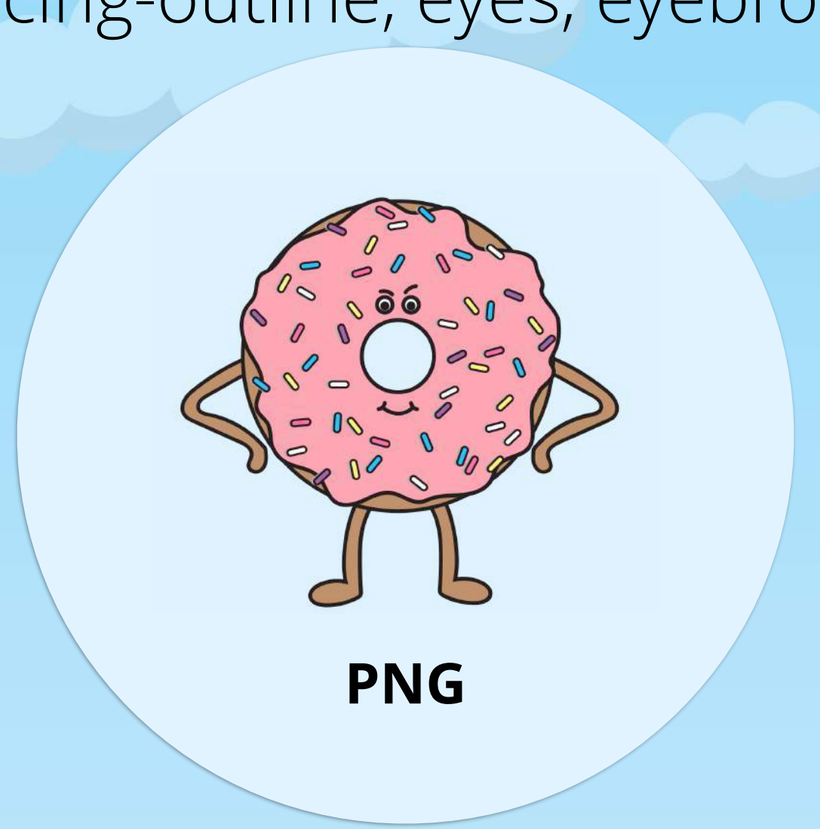
Level 4 – SVG

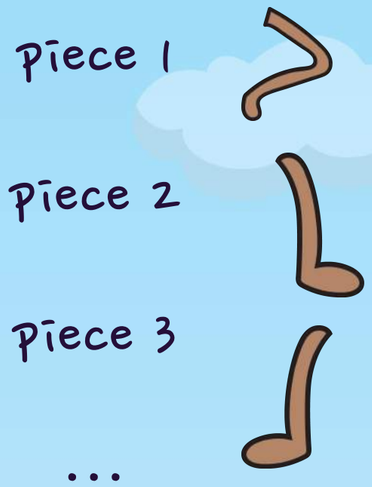
SECTION 1

Animating SVGs With CSS

Animating Lots of PNGs Together Is Tedious

The donut is comprised of a **ton** of pieces: arms, legs, body, icing-fill, icing-outline, eyes, eyebrows — the list goes on.

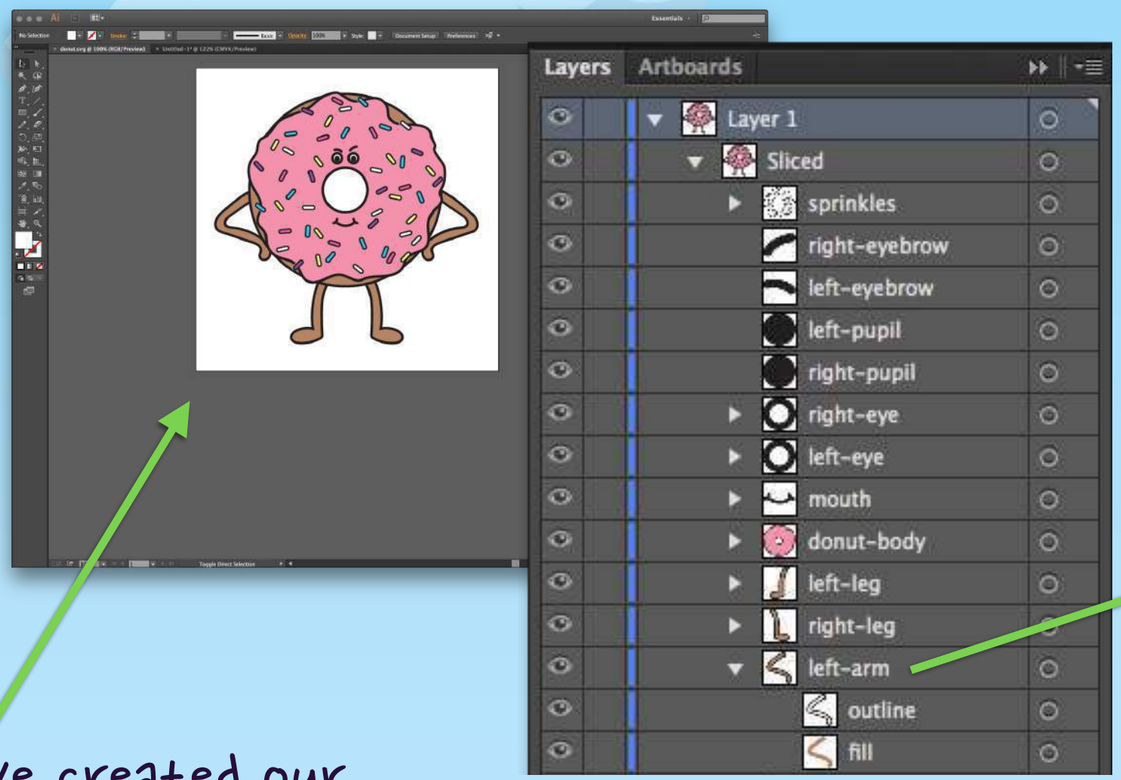




If we use an SVG donut image, we'll be able to easily animate the icing, sprinkles, and any other part of the donut!

Getting SVG Assets

You can create your own SVG asset or find a free/paid asset online.



we created our SVG donut in Adobe Illustrator

Labeling layers in Illustrator will assign ids in the SVG file when saving it out.

SVG

```
<g id="left-arm">
  <path id="fill" d="M123...."/>
  <path id="outline" d="M123..."/>
</g>
```

What Is SVG?

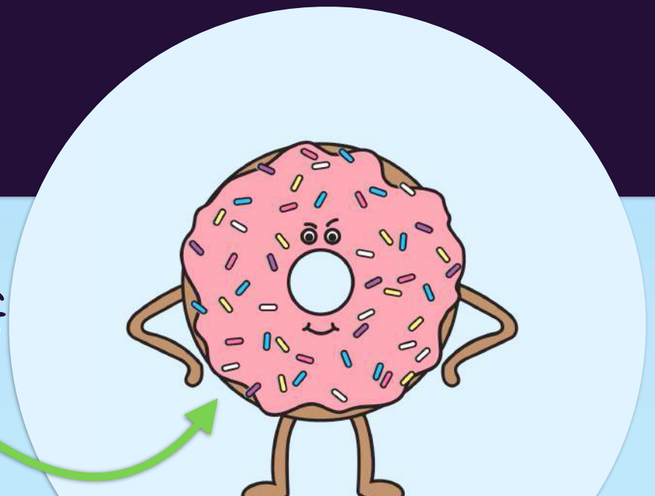
SVG is a file format that contains vector-based images.

SVG

```
<?xml version="1.0" encoding="utf-8"?>
<svg x="0px" y="0px" viewBox="0 91 612 612">
   <style type="text/css">...</style>
   <g id="donut">
      <g id="right-arm">...</g>
      <g id="left-arm">...</g>
      <g id="right-leg">...</g>
      <g id="left-leg">...</g>
      <g id="donut-body">...</g>
      ...
   </g>
</svg>
```

different animatable parts are in different elements

SVG version of the donut



SVG is written in XML, another tag-based language!

Replacing PNGs With SVGs

SVG can be dropped in your HTML file wherever you're normally loading PNG (or other) images.

HTML

```
<section class='contact'  id='contact'>
  <div class='cell well'>
    <h2>Contact Us</h2>
    <div id="donut">
      <image id="left-arm" src="https://s3.../left-arm.png"></image>
       ...
    </div>
    <p>...</p>
  </div>
</section>
```

Let's delete these PNG images...

...and replace them with one SVG image.

SVG

```
<?xml version="1.0" encoding="utf-8"?>
<svg x="0px" y="0px" viewBox="0 91 612 612">
  <style type="text/css">...</style>
  <g id="donut">
    <g id="right-arm">...</g>
    <g id="left-arm">...</g>
    <g id="right-leg">...</g>
```

Bonus: SVG Images Are Always Crisp

Without any additional work, SVG donut is as crisp as the day he was freshly fried!

No matter how much you zoom, SVG donut is still lookin' good!

SVG zoomed in

PNG zoomed in

Accessing Elements in SVG

HTML

```
<section class='contact'  id='contact'>
   <div class='cell well'>
     <h2>Contact Us</h2>
     <?xml version="1.0" encoding="utf-8"?>
     <svg x="0px" y="0px" viewBox="0 91 612 612">
       <style type="text/css">...</style>
       <g id="donut">
         <g id="right-arm" class="st0" d="M302...">...</g>
         <g id="left-arm" class="st1" d="M302...">...</g>
         <g id="right-leg" class="st2" d="M302...">...</g>
         <g id="left-leg" class="st3" d="M302...">...</g>
         <g id="donut-body" class="st4" d="M302...">...</g>
         <g id="sprinkles" class="st5" d="M302…">…</g>
         ...
       </g>
     </svg>
     ...
 </section>
```

We can access specific SVG tags with CSS selectors

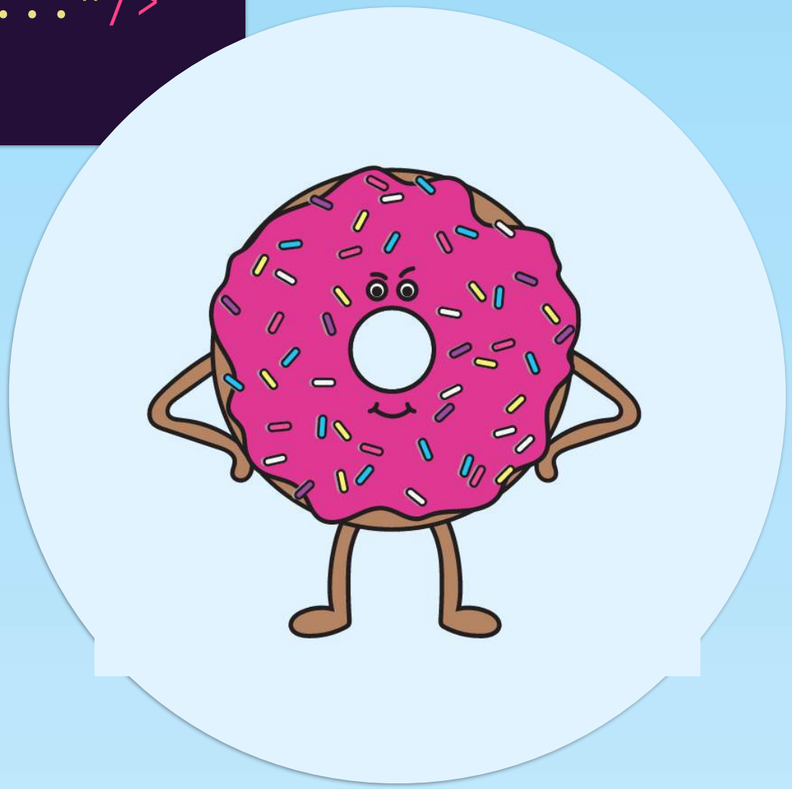
Elements Accessed in SVG

HTML

```
<g id="donut-body">
  <path id="donut-fill" class="st0" d="M302..."/>
  <path id="donut-outline" class="st1" d="M302..."/>
  <path id="icing-fill" class="st2" d="M503..."/>
  <path id="icing-outline" class="st1" d="M237..."/>
</g>
```

CSS

```
#icing-fill {
  fill: #DD3D93;
}
```



we couldn't do this with PNG!

Animating the Icing-fill Color

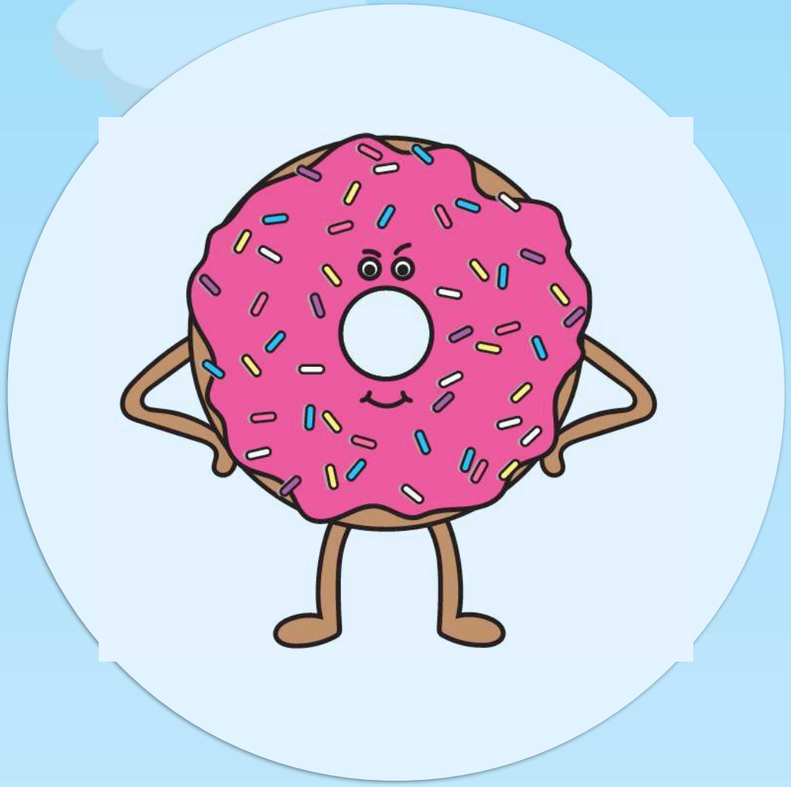
We can now animate our icing darker to show our donut's frustration with those pesky sprinkles!

CSS

```
@keyframes darken {
  0% {fill: #FCA9B7;}
  100% {fill: #DD3D93;}
}


#icing-fill {
  animation: darken 3s infinite;
}
```

SVG uses fill instead of background-color



Unique Properties for Styling SVGs

SVG has some unique CSS properties that can be animated.

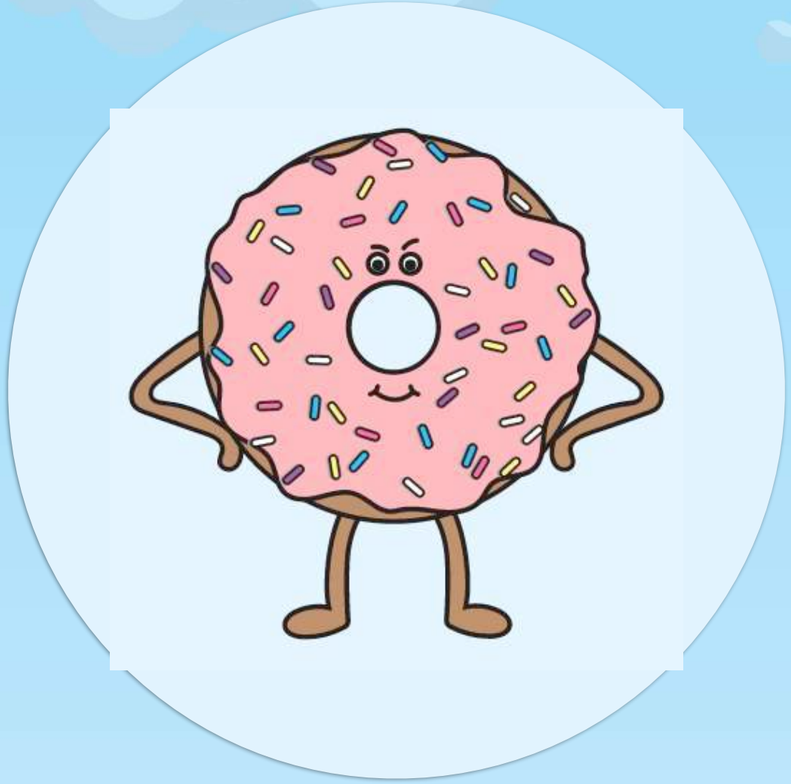
```
enable-background
fill
fill-opacity
filter
mask
stroke
stroke-dasharray
stroke-dashoffset
viewport-fill
viewport-fill-opacity
```

Check out MDN for the full list of **SVG properties**.

http://go.codeschool.com/svg-css-properties

Shake It Off!

We are getting close! The last animations that need to happen are the sprinkles and icing shaking and the sprinkles flying off.



Animating Multiple Properties in a Keyframe

This keyframe animation will be used to rotate, scale, and fade the sprinkles, effectively imitating what sprinkles flying off would look like.

CSS

```css
@keyframes flyoff {
  0% {transform: rotate(0deg);}
  10% {transform: rotate(30deg);}
  20% {transform: rotate(0deg);}
  30% {transform: rotate(-30deg);}
  40% {transform: rotate(0deg);}
  45% {opacity: 1;}
  50% {
    transform: rotate(100deg) scale(3);
    opacity: 0;
  }
  100% {
    opacity: 0;
    transform: scale(1);
  }
}
```

rotating sprinkles back and forth

rotating further and scaling really large for "flyoff effect"

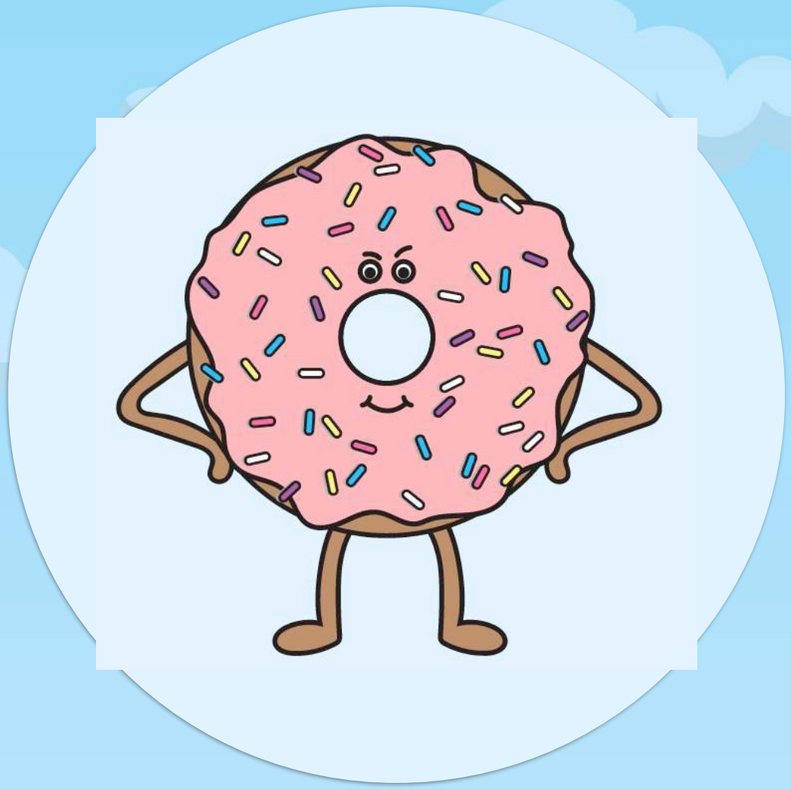
reset the scale at the end of the animation so the loop looks more natural

The Sprinkle Animation in Action

CSS

```
@keyframes flyoff {
    ...
}


#sprinkles {
  transition: transform 2s;
  transform-origin: 302px 337px;
  animation: flyoff 3s infinite ease-in;
}
```



Animating the Icing

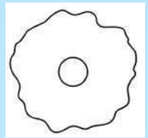
The icing is made up of 2 parts: the outline and the filled color.

we've added this same animation to the icing-fill...

CSS

```
#icing-fill {
  animation: darken 3s infinite,
             shake 3s infinite ease-out;
  transform-origin: 302px 337px;
}
#icing-outline {
  animation: shake 3s infinite ease-out;
  transform-origin: 302px 337px;
}
```

...and to the icing-outline.

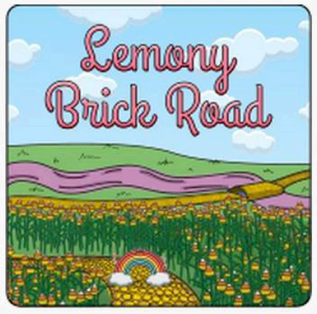


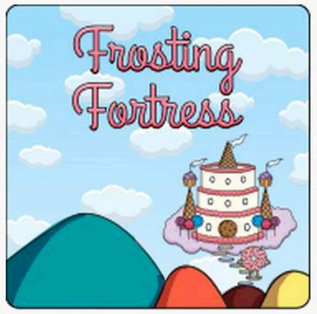
Shake Animation for Icing

CSS

```
@keyframes shake {
  0% {transform: rotate(0deg);}
  10% {transform: rotate(30deg);}
  20% {transform: rotate(0deg);}
  30% {transform: rotate(-30deg);}
  40% {transform: rotate(0deg);}
  50% {transform: rotate(100deg);}
}
```

Our Donut SVG Is Now Animating Wonderfully





CONTACT US

You don’t need to tangle with the Tangy Tart Trickster to get a hold of us! Simply send a message through our site or reach us on our social media pages. We strive to make sure every customer is a happy one, so feel free to let us know if you have any comments or questions!