## Overview

In this activity, you’ll build some common animations that you’ve probably seen across the internet.

**Note:** Keyframe animations are frequently used to build an engaging interface that delights users. CSS animations consist of two components: a style describing the CSS animation and a set of keyframes that indicate the start, intermediate, and end states of the animation’s style.

## Instructions

1. Open the index.html and index.css files.

**Animation 1: Bouncing Arrow**

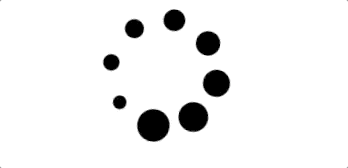
****

1. Add the property animation: bounceArrow .7s infinite to the .bounceMe selector.
   * **Note:** Refer to documentation and other resources as needed.
   * Also, note the name of the animation must be the same as the keyframe specified later in the file.
   * There are three things we just specified. A name of bounceArrow, a duration of .7s and infinite repetition.
2. Now it’s time to modify the keyframes. Start with the start state:
   * Add the property margin-top: 0px; to the start state for bounceArrow.
   * The bounceArrow animation will now start with a 0px top margin.
3. Next, add margin-top: 7px; to the intermediary keyframe.
   * This indicates our bounceArrow animation will have a top margin of 7px at its halfway point.
4. Lastly, add a property to the end state so our animation returns to its original starting position.

**Resources**

* CSS animation documentation: <https://www.w3schools.com/css/css3_animations.asp>
* CSS @keyframe documentation: <https://www.w3schools.com/cssref/css3_pr_animation-keyframes.asp>

**Animation 2: Loading Circle**

****

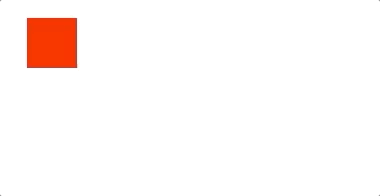
1. Next, you will animate a loading circle. Add the property animation: spin 2s infinite; to the .spinMe selector.
2. Modify the start state by adding the property transform: rotate(180deg);.
3. Next, bring the animation full circle:
   * Add the property transform: rotate(-180deg); to the end state.
4. Take a minute to inspect these two states.
   * **Note:** Try to notice why we use only two states to make an animation.

**Animation 3: Loading Bar**

****

1. Next, you will animate a loading bar:
   * Add the properties animation: loadBar 4s; and animation-fill-mode: forwards; to the #progressBar > .orangeBar selector.
   * The first property names and adds a duration to the animation.
   * The second states the direction of the animation.
2. Modify the start state by adding the property width: 40%;.
3. Finish the loading bar by adding the property width: 100%; to the end state.

**Animation 4: Moving Box**

****

1. Animate the red box by adding the values to the .movingBox selector that declares the name of the keyframes (movingBox) and duration (5s).
2. Modify the start state by adding the property transform: translate3d(0px, 0px, 0px);.
   * There’s a lot happening here, so take a moment to review the syntax:
   * The first value is the x-axis, the second value is the y-axis, and the third value is the z-axis. We’re not transforming along the z-axis in this case, but we still have to declare it.
   * **Note:** Read more about translate3d [here](https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function/translate3d).
3. Add the following properties to the remaining keyframes in the same order:
   * transform: translate3d(50px, 25px, 0px);
   * translate3d(100px, 50px, 0px);
   * translate3d(200px, 100px, 0px);
   * translate3d(0px, 0px, 0px);

**Animation 5: Bonus**

1. It's time for you to get creative. If you're feeling adventurous and think you have a solid understanding of keyframe animations, try to create your own:
   * Loading bar
   * Animated UI element
   * Icon that animates in some way.