Front-End Web Development

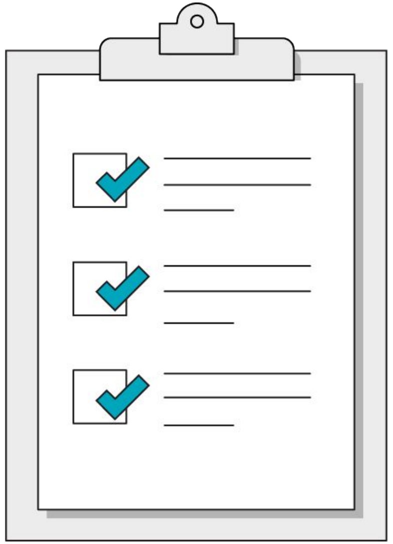
Interactive UI Components

GENERAL ASSEMBLY

Today's Learning Objectives

In this lesson, you will:

- Use JavaScript to trigger CSS animations.
- Design interactive user interfaces using CSS properties.
- Plan application states to reflect user actions.



Previously, on Front-End Web Development...

HTML	CSS	JS
● Content	● Style rules	● Programs
● Tags	● Selectors	● Variables
● Attributes	● Media queries	● Functions
● Structure	● Responsive units	● Events
● Semantics	● Flex/grid layouts	● DOM elements
● Relationships		● Listeners

Front-End Web Development

Applications and Animations

Three Types of Animation

We're going to cover three animation techniques in this lesson:

? **Mild**: `transition` property with the `:hover` pseudo-class.
  - Changes the background color of a button when the user hovers over it.

? **Medium**: `animation` property with the `@keyframes` rule.
  - Moves an element on the page with animation.

? **Spicy**: Switch application state with `.classList.toggle()`.
  - Triggers an off-canvas menu to appear or disappear.

Mild Animation

The easiest way to animate something in CSS is by using the **`transition`** property in conjunction with the **`:hover`** pseudo-class. This method includes:

- The CSS property to be animated.
- The time duration of the animation.
- The type of animation to be used.

Transition Tips

The **`transition`** property is assigned to the *off* (initial) state of your component.

Typically, you can leave out the specific thing changing and just use something like **`transition: 0.4s ease-in-out;`**. If you're sure you only want one specific property to transition, however, it's safest to be specific and leave it in.

Animating multiple properties at once can be a bit much for you and for users, so go easy with this stuff!

Transition Example

Start > End

```
.button {
  background-color: red;
  transition: background-color 0.4s ease-in-out;
}

.button:hover {
  background-color: orange;
}
```

Hiding Content

Keep your HTML block turned off with CSS. There are four methods with which to hide elements in CSS, each with slightly different results.

Property	Takes Up Space	Can Be Clicked
`display: none;`	No	No
`visibility: hidden;`	Yes	No
`opacity: 0;`	Yes	Yes
`background: rgba(0,0,0,0)`	Yes	Yes

Guided Walk-Through:

CSS Hiding Methods

Let’s explore these properties through examples in a CodePen!

Reference code:

https://codepen.io/GAmarketing/pen/LYExBWQ

Front-End Web Development

Keyframes and Animation

Guided Walk-Through:

Animations References

CSS **`transition`**, **`transform`**, **`animation`**, and **`@keyframes`** examples galore!

Reference code:

https://codepen.io/GAmarketing/pen/abzpjMG

Medium Animation

Complex animations require multiple steps. For these, you'll need to use CSS **`animation`** and **`@keyframes`** together. This will allow components to move in more interesting ways than simply appearing and disappearing!

Here's what it takes:

1. Outline animation steps and timing.
2. Set each movement instruction of a **`@keyframes`** step (usually a **`transform`**).
3. Use CSS **`animation`** properties and apply everything to the component's initial state.

CSS @keyframes

`@keyframes` work like media queries — conditional-based instructions that happen at particular times. Notice that we're giving this a custom name, `mover`.

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

CSS Animation

The property contains a bunch of settings, similar to how **background** works.

```
.box {
  animation-name: mover;
  animation-duration: 2s;
  animation-timing-function: ease-in-out;
  animation-direction: alternate;
  animation-iteration-count: infinite;
}
```

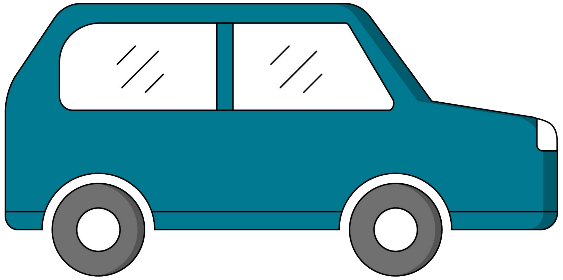
OR

```
.box {
  animation: mover 2s ease-in-out alternate infinite;
}
```

CSS Transforms

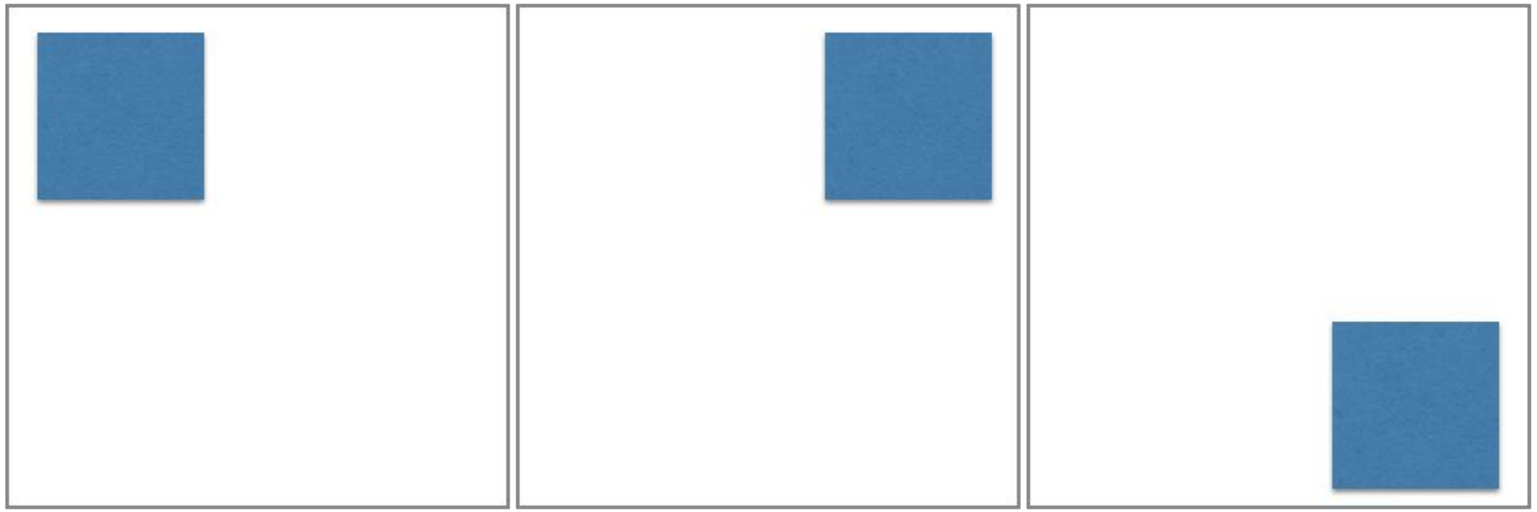
You can also adjust the shape and position of elements with **`transform`**. It's not required to do animation, but it is the easiest way to move things fluidly.

You can do a lot with **`transform`** — [CSS Tricks] has great documentation, as usual!



Storyboard Keyframes

Complex animations require something akin to pseudocode before digging in.



Step 1 (0%):
Box top, left

Step 2 (50%):
Box top, right

Step 3 (100%):
Box bottom, right

Partner Exercise:

Animations Practice

30 minutes

With a partner, practice applying these CSS properties in this CodePen.

Starter code:

https://codepen.io/GAmarketing/pen/QWwdVbd

Solution code:

https://codepen.io/GAmarketing/pen/bGNgxVe

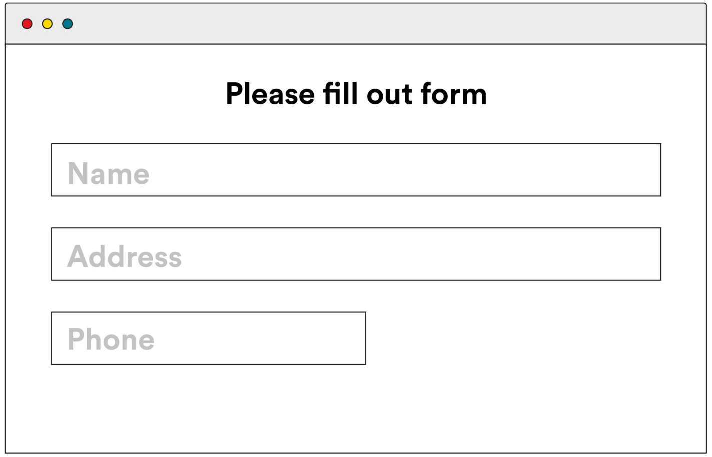
Front-End Web Development

Stateful Applications

Stateful Apps

Interactive components subtly introduce a huge idea: **state**. State can include anything the user might toggle or adjust in your site, such as:

- Showing or hiding menus, forms, or modal windows.
- Filtering, adding, or removing the data displayed on a page.
- Filling out form inputs.



How to Build Components

Building actual components means taking everything you know and putting it together. It may seem like a lot at first, but start slow and you'll get there. Here's how the steps break down:

1. Sketch out the two states of your component.
2. Build the "on" state first without turning it "off." It will likely sit on top of your other work for now, but that's OK!
3. Finally, add in the special CSS/JS you need to make the magic happen. Do this LAST!

Solo Exercise:

Navigation Menu Toggle

Add the necessary JavaScript to this CodePen so that the hamburger icon opens a navigation menu and the “Close This” menu item closes it again.

Explore the CSS as another reference for animations!

Starter code:

https://codepen.io/GAmarketing/pen/YzPNOZK

Solution code:

https://codepen.io/GAmarketing/pen/mdyRGmJ

Key Takeaways

CSS Animations Liven Up UX

- Animating HTML elements can guide users through your applications.
- The **state** of an application can be reflected with CSS.
- Don't overdo it! Animations should be subtle and unobtrusive.

For Next Time

JavaScript Conditionals

- Conditionals allow your applications to evaluate conditions and make decisions based on logic.
- Complete Homework 6.

Advanced Animation Resources

- Charts.js: good enough charts/graphs
- D3: a popular data visualization tool
- GSAP: a keyframe animation toolkit