Tip 51: Leverage CSS for Animations

In this tip, you'll learn how to use CSS for animations.

We'll cover the following

- Using css for animations
  - Example
  - Adding functionality to the button
  - CSS transition

The last tip isn't even a JavaScript tip. Instead, it's a tip about when to stop using JavaScript.

Using css for animations

The key to writing readable code is to use the best tool for the job. JavaScript used to be the best tool for animations. In fact, there were entire libraries dedicated to using JavaScript to create *drop-down* menus or to animate *slidein* elements.

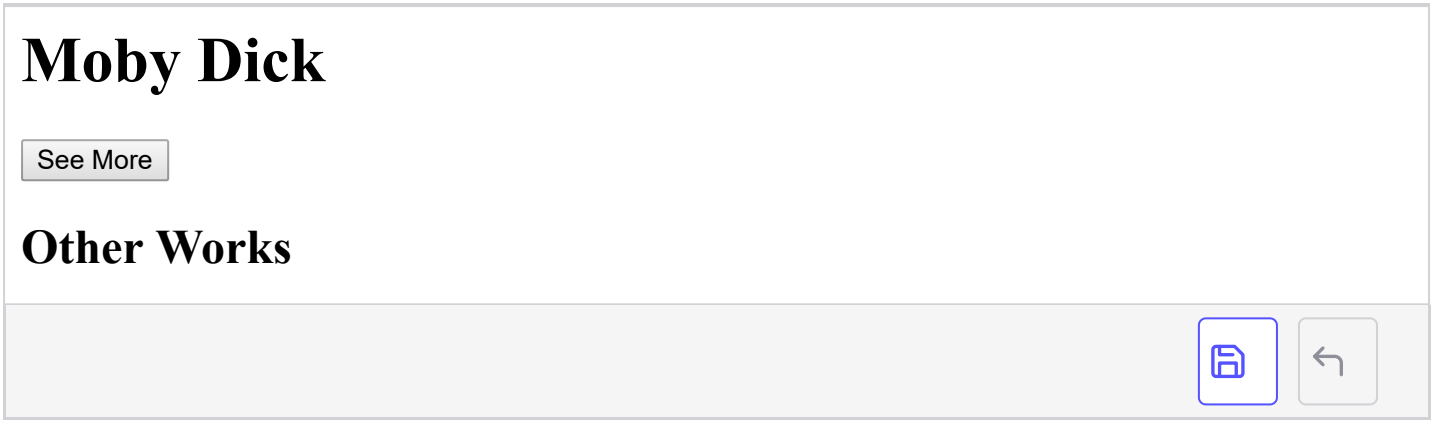
It's much easier now. CSS is replacing JavaScript for simple animations. That's great. Now you don't have to worry about using the right timeouts or calculating odd-size constraints. CSS will take care of all that for you. You'll still need JavaScript for more complicated animations, but for most common tasks, CSS works great.

Example

Start by creating a simple page that has a *menu* on the right. The markup is very simple. You need some *text*, a *button* to toggle the side menu, and the menu itself.

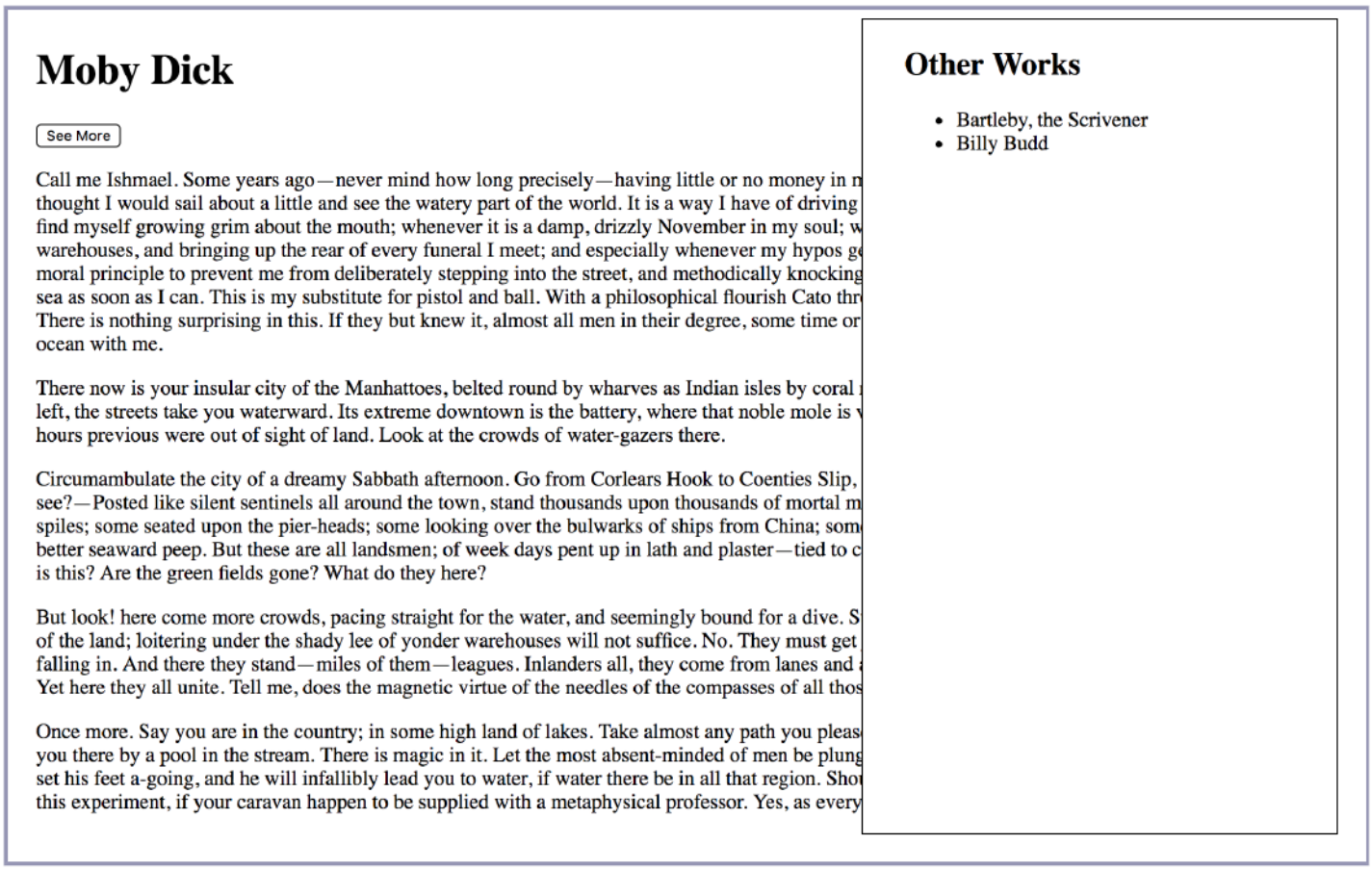
Output

HTML



Start with simple CSS to place the menu on *top* of the text.

As you can see in the figure below, when the menu is fully opened, it will cover a portion of the words. You'll change that in a moment.



Now that you have the page set up, you're going to add some CSS animations to slide the menu on and off the page.

The first step is hiding the side menu. Add the following property to your `.menu` class.

```
transform: translateX(calc(300px + 4em + 2px));
```

This property and value—`transform: translateX`—will move the page outside the container *div*, making it appear invisible. The calculation is the *width* of the menu, plus the padding, plus the border.

Output

HTML

CSS (SCSS)

Moby Dick

See More

Call me Ishmael.

With the menu hidden, it's time to add a transition. A CSS transition is an animation of a changing property. In other words, an animation is just a visual transition between two properties of the same name.

Adding functionality to the button

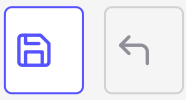
That's fine, but how do you change a property? Turns out, you'll need a little JavaScript after all. Add a `click` event listener to the button. The *callback* function for the button will toggle the `.display` class on the menu. The first time you click the button, it will add the class. The second time you click the button, it will remove the class.

JavaScript

HTML

CSS (SCSS)

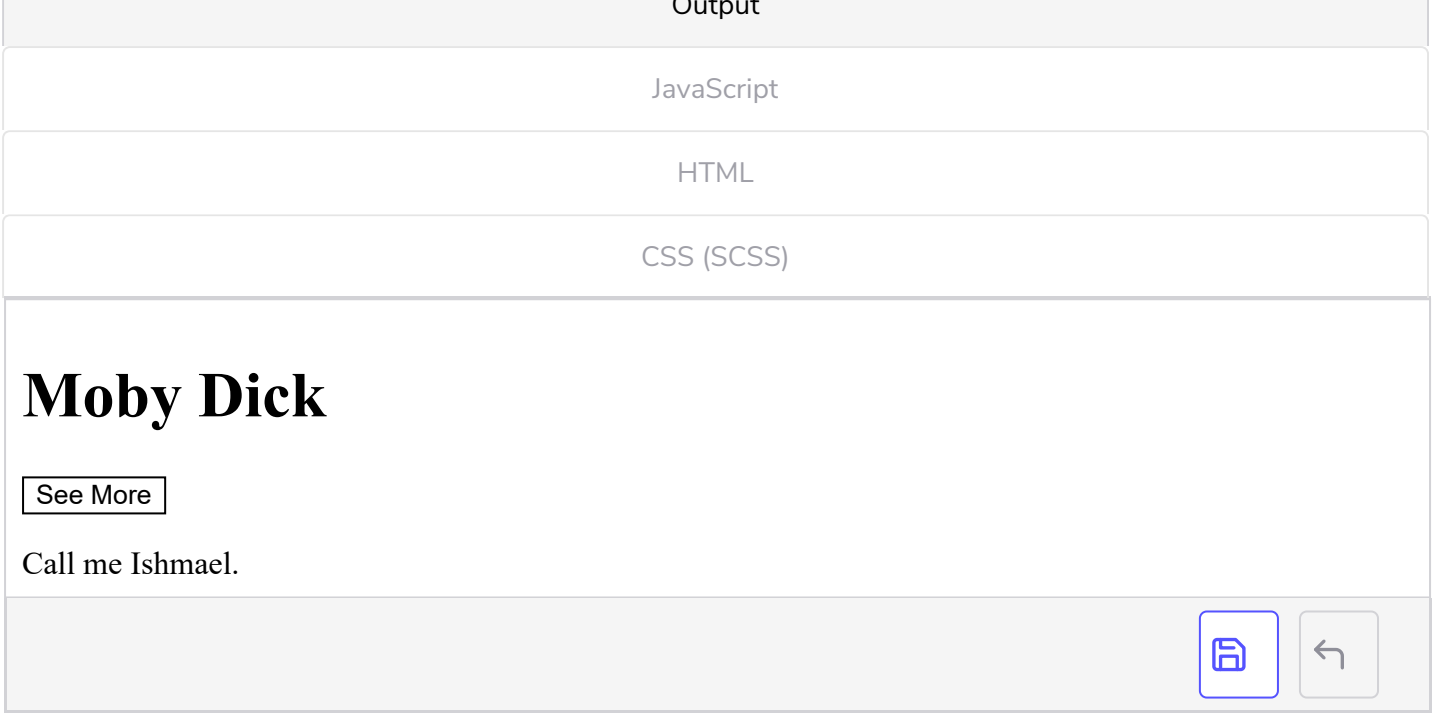
```
const sidebar = document.getElementById('sidebar');
document.getElementById('show')
  .addEventListener('click', () => {
    sidebar.classList.toggle('display');
  });
```



CSS transition

Next, you need to update your stylesheet to include styles for a `.menu.display`

```
.menu {
    /* Other styles from before */
    transform: translateX(calc(300px + 4em + 2px));
}
.menu.display {
    transform: translateX(0);
}
```



When you add the class, you move from a transform of `calc(300px + 4em + 2px)` to a transform of `0`. Because the browser knows that property is changing, it can trigger an animation. All that's left is to tell the element how to respond to the changing property.

A **CSS transition** is a set of instructions telling the page what to do when moving from an initial property value to the final property value. Check out the Mozilla Developer Network documentation for different options.

First, declare the property that will need a transition with `transition-property`. In this case, you only want to animate the transform, so set the value to `transform`. Next, set the length of the durations with `transition-duration`.

The transition time will seem very quick or very slow depending on how radical the transition is. Going from `0px` to `10px` in one second will seem much slower than going from `0` to `100px`. In this case, set it for `600ms`.

Finally, you need to say how the transition should act with the transition-timingfunction property. This one is a little more tricky. It can make the transition move faster at the beginning, faster at the ending, or the same speed throughout. Set the value to `linear` to keep a nice smooth slide throughout.

Here's the updated CSS for `.menu.display`:

```
.menu.display {
  /* Other styles */
  transform: translateX(0);
  transition-property: transform;
```

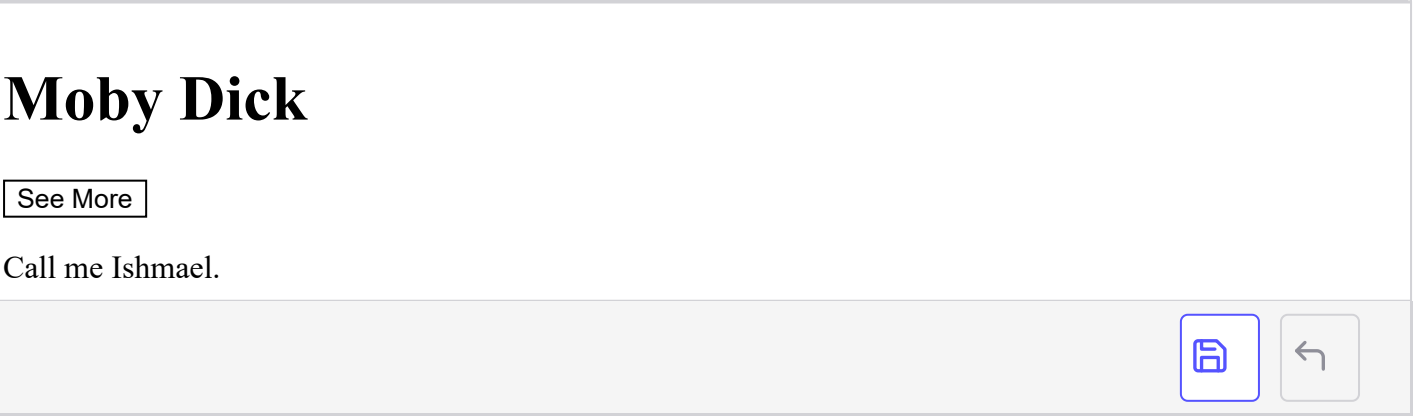
```
  transition-duration: 600ms;

  transition-timing-function: linear;
}
```

Output

JavaScript

HTML

CSS (SCSS)



When you click the button, you should see the menu slide into place. But when you click it again, the menu will just instantly disappear. The problem is that you only declared the transition for when you add the `.display` class. There's no transition for when you remove the `.display`.

No problem—all you need to do is add a transition to the base `.menu` styles. This time, though, you can shorten things up. You can add all three properties `transition-property`, `transition-duration`, and `transition-timing-function`—to a single property called `transition`. You also have the option to apply the transition to any changing property by setting the transition to all.

Once you update the stylesheet, you'll have a menu that slides in on button click and slides out on button click with very minimal JavaScript.

```
.menu {
  /* Other styles */
  transform: translateX(calc(300px + 4em + 2px));
  transition: all 600ms linear;
}
```

Output

JavaScript

HTML

CSS (SCSS)

Moby Dick

See More

Call me Ishmael.

Honestly, you only need the transition on the `.menu` class, unless you want the slide in and slide out to be different. Transitions are like any other property. They'll bubble up to all elements unless you override them with a more specific selector.

This simple slide-in used to require a lot of JavaScript code. Now it only takes a single line of CSS and a simple class toggle. Part of what makes web development such a delight is the tool set is steadily improving. HTML is more semantic. Styles are more flexible. JavaScript is simpler and easier to read.

JavaScript is a great language, and I hope you learned to love it. It's simple, expressive, and very elegant. And the best part is it's getting better all the time. You have everything you need to start writing JavaScript that you can be proud of. Now all you need to do is start building. Have fun.