

2.5.4 Add Shadows and More

Our first code edit will be to add shadows to some of our HTML elements. This is something the design team requested to give the elements more depth.

In CSS, shadows come in two flavors: text shadows and box shadows. They're represented by the following CSS declarations:

```
.some-class {  
  text-shadow: 5px 10px 15px black;  
  box-shadow: 5px 10px 15px black;  
}
```

PAUSE

The `text-shadow` property is obviously used for text, so when and where would you use `box-shadow`?

Adding a shadow to containers like `<div>` elements.

[Hide Answer](#)

There are quite a few value possibilities for shadows, but the most common usage is to define the horizontal offset (`5px` in our example), vertical offset (`10px`), radius of the shadow blur (`15px`), and finally the color. The offsets start in the top-left corner of the shadowed element, and these numbers can be negative.

Here's an example of positive and negative box shadow declarations:



DEEP DIVE ▲

DEEP DIVE

For more information, see the [MDN web docs on box shadows](https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow) (<https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow>).

The black shadow in the example is a little too bold. We can lighten it by applying transparency to it, though this does require using RGB values instead of color keywords like `black` or hexadecimal values like `#024e76`. **RGB** stands for red/green/blue and allows us to define how much of these three colors to blend together on a scale of 0 to 255. The value `rgb(0, 0, 0)` represents black and `rgb(255, 255, 255)` is white.

DEEP DIVE ▲

DEEP DIVE

Another color model you might hear about is **CMYK**, which stands for cyan/magenta/yellow/black. CMYK is used for physical printing, whereas RGB is used for displaying images electronically. When red, green, and blue lights overlap, the result is white. If only red and green overlap, the result is yellow. Codewise, we could write that as `rgb(255, 255, 0)`. To make a softer yellow, we can add a little blue to bring it closer to white: `rgb(255, 255, 200)`.

You can experiment with RGB and CMYK values using an online color picker. These are so common that Google provides a color picker directly in its search results!

When playing with RGB values, we still end up with a solid color that can't be looked through. We'll actually need to use `rgba()` instead of `rgb()`, where the "a" stands for "alpha." In other words, alpha is the transparency, defined by a decimal number between 0.0 and 1.0. An alpha value of 1 would be fully opaque while 0 would be fully transparent.

Let's revisit our previous example:

```
.some-class {  
  text-shadow: 5px 10px 15px black;  
  box-shadow: 5px 10px 15px rgba(0, 0, 0, 0.25);  
}
```

Both of these declarations are still black, but the `box-shadow` declaration turns the opacity down to 25% (0.25).

Here's another example, which we'll use in Run Buddy:

```
text-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
```

The first `0` means there is no horizontal offset; the shadow will start at the left side of the text. The second `0` means no vertical offset. The value `10px` defines the radius (i.e., size) of the blur. Finally, `rgba(0, 0, 0, 0.5)` sets the color to black at half transparency.

Per the mock-up, add the following `text-shadow` property to the following three CSS rules in `style.css`:

```
header h1 {  
  /* add alongside your existing declarations */  
  text-shadow: 0 0 10px rgba(0, 0, 0, 0.5);  
}  
  
header nav ul li a {  
  /* and here */  
}  
  
.hero-cta h2 {  
  /* and here */  
}
```

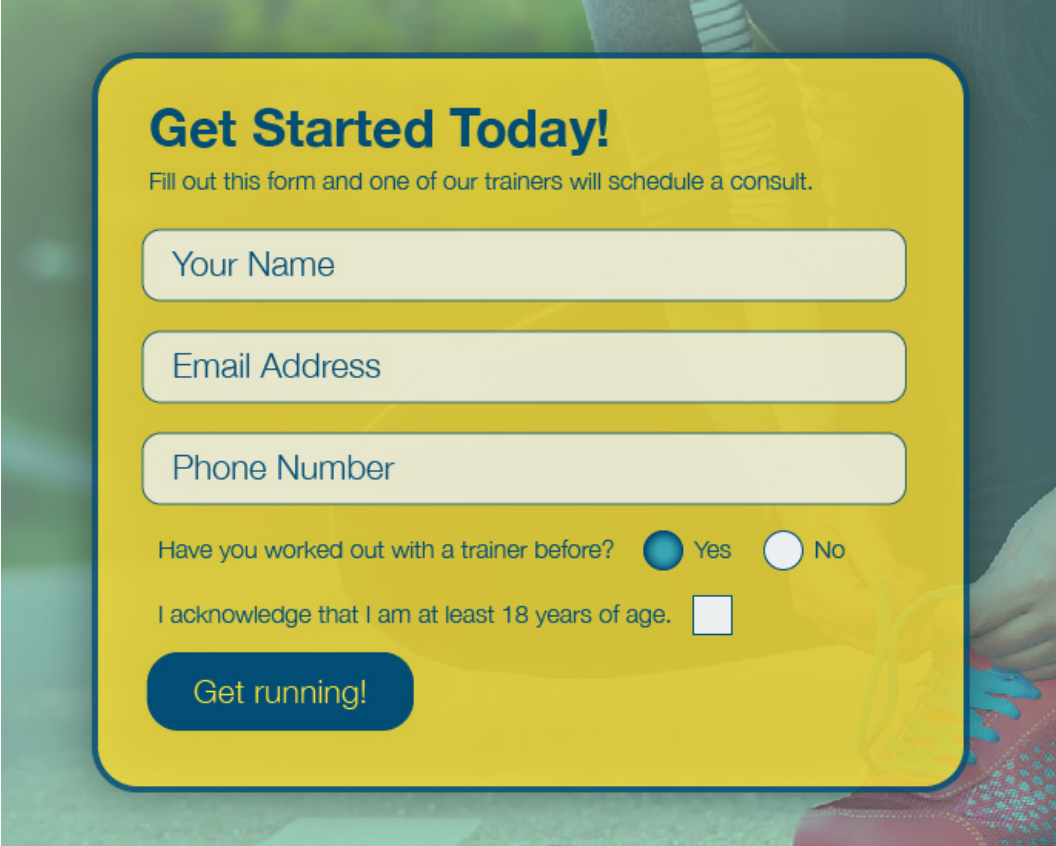
Save the file, refresh the browser, and enjoy your new shadows!

There's one more shadow we need to add, though, and that's to the hero form itself. The `text-shadow` property wouldn't be appropriate here because it would add shadows to all of the text inside the form. Instead, let's use `box-shadow`.

Add the following declaration to the `.hero-form` rule:

```
box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
```

While you're fiddling with the hero form, look closely at the mock-up and notice how the background color is slightly transparent:



The style sheet currently defines the background color as the following solid color:

```
.hero-form {  
  background-color: #fce138;  
}
```

How can we add that transparency? First, we need to know the RGB equivalent of `#fce138`, which we can obtain by using a converter. Perform a quick [Google search for HEX to RGB converters](#)

(<https://www.google.com/search?q=convert+hex+to+rgb>) to find one. Once you have the correct values, rewrite the `background-color` declaration to be 80% opaque.

HIDE HINT

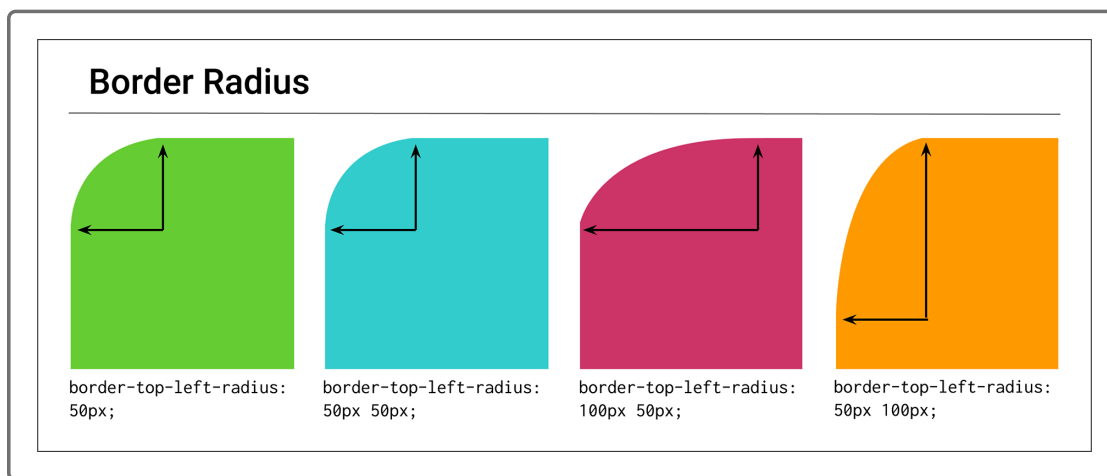
The RGB values are 252, 225, and 56. And you'll need to use `rgba()` and set the alpha to `0.8`.

The hero form also has several rounded corners, not only for the form itself but for the form elements inside. The CSS property for this is called `border-radius`.

Add the following declaration to your `.hero-form` rule:

```
border-radius: 15px;
```

This is shorthand to add a rounded corner to all four corners of the element. This essentially means move 15 pixels in on the x-axis, move 15 pixels in on the y-axis, and draw a circle with a radius of 15 pixels. Other units of measurement could be used, as well:



Next, let's add a small rounded corner to all of the form elements on the page. Update the following CSS rules to have a `border-radius` of `10px` (we'll let you fill these in all on your own!):

```
.form-input {  
  
}  
  
.hero-form button {  
  
}  
  
.contact-form input, .contact-form textarea {  
  
}  
  
.contact-form button {  
  
}
```

LEGACY LORE

Before the invention of `border-radius`, developers had to jump through some annoying hoops to simulate rounded corners. One common trick was to add four images inside the `<div>`, one for each corner, and use absolute positioning to move them into the appropriate places.

© 2020 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.