# **Advanced topics**

The difference between a site that looks good and one that looks great is attention to detail. After you lay out and style a component of your page, train yourself to slow down and look at it with a critical eye. Does it look better if you increase, or decrease, paddings? Adjust the colors a bit—do they look better a little

darker or little lighter, or a little less vivid? If you’re working from a detailed mockup by the designer, does your implementation match everything as closely as possible? Your designer spent a lot of time on those details. Make sure you’re doing the design justice.

## **Gradients**

[https://developer.mozilla.org/en-US/docs/ Web/CSS/radial-gradient.]

there’s a lot more to explore when it comes to the background property; it’s, in fact, a shorthand for eight properties:

* **background-image**—Specifies an image from a file or a generated color gradient.
* **background-position**—Sets the initial position of the background image.
* **background-size**—Specifies how large to render the background image within the element.
* background-repeat—Determines whether to tile the image if necessary to fill the entire element.
* **background-origin**—Determines whether background positioning is relative to the element’s border-box, padding-box (initial value), or content-box.
* **background-clip**—Specifies whether the background should fill the element’s border-box (initial value), padding-box, or content-box.
* **background-attachment—**Specifies whether the background image will scroll up and down along with the element (the initial value), or if the image will be fixed in place in the viewport. Note that using the value fixed can have negative performance implications on the page.
* **background-color**—Specifies a solid background color. This will render behind any background image.

I tend to prefer using individual properties when I’m doing anything that requires more than a few

of them.

The linear-gradient function has three parameters defining its behavior: angle, starting color, and ending color. The angle here is to right, meaning the gradient starts on the left edge of the element (where it’s white) and blends evenly to the right edge (where it’s blue). You can also use other color syntaxes, such as hex (#0000ff), RGB (rgb(0, 0, 255)), or the **transparent** keyword. You can specify the angle of the gradient in several ways. In this example, you used ***to right***, but you can also use **to top or to bottom.** You can even specify a corner such as **to bottom right**;

For more precise control of the angle, you can use more specific units, such as degrees. The value 0deg ,90deg;

### **Using multiple color stops**

you can define a gradient with multiple colors, which are each called a *color stop;*

|  |
| --- |
| background-image: linear-gradient(90deg, red, white, blue); // **Specifies multiple color stops** |

You can also explicitly set the position of these color stops in the gradient function.

|  |
| --- |
| linear-gradient(90deg, red 0%, white 50%, blue 100%) |

### **STRIPES**

If you place two color stops at the same position, the gradient’s color will instantly switch from one to the

other, rather than a smooth transition;

|  |
| --- |
| **background-image: linear-gradient(90deg, red 40%, white 40%, white 60%, blue 60%);** |

The first color stop is red at 40%, so the gradient is solid red from the left edge all theway to 40%. The next color stop is white, also at 40%, so the gradient makes a hard switch to white. This is followed by another white color stop at 60%, so the gradient is pure white from 40% to 60%. Then the final color stop, also at 60%, is blue. This makes a hard switch to blue, then remains blue all the way to the right edge.

### **REPEATING GRADIENTS**

the technique can be used for some interesting effects. In particular, it can be used with a slightly different gradient function, repeating-linear-gradient(). This works like the regular linear-gradient

function, except the pattern repeats; With repeating gradients, it’s better to use a specific length rather than a percentage,because specified values determine the size of the pattern to repeat.

|  |
| --- |
| **background-image: repeating-linear-gradient(-45deg,#57b, #57b 10px, #148 10px, #148 20px);** |

### **Using radial gradients**

radial gradients start at a single point and proceed outward in all directions; By default, the gradient is centered in the element, transitioning evenly to its corners. It’s elliptical in shape, matching the proportions of the element; It supports all the properties of the linear gradient;

|  |
| --- |
| radial-gradient(circle, white, midnightblue)  …  radial-gradient(circle, midnightblue 0%, white 75%, red 100%) |

## **Shadows**

Two properties that create shadows are box-shadow, which creates a shadow of an element’s box

shape, and text-shadow, which creates a shadow of rendered text.

The declaration ***box-shadow: 1em 1em black*** produces a shadow; The 1em values are the offsets: how far the shadow will be shifted from the element’s position (horizontal, then vertical). If these have a value of 0, then the shadow will be directly behind the element. The value black specifies the color of

the shadow;

box-shadow: horizontal offset (*x*), vertical offset (*y*), and color should always be specified for the shadow. Two other values can optionally be added: a blur radius and a spread radius.

|  |
| --- |
| box-shadow: 2px 2px 2px 1px black; |

The blur radius controls how much the edges of the shadow are to be blurred. This will give it a softer, slightly transparent edge. The spread radius controls the size of the shadow. A positive spread radius makes the shadow larger in all directions; a negative value makes the shadow smaller.

### **Defining depth with gradients and shadows**

Let’s use gradients and shadows to build a button as shown in figure 11.10. A top-to bottom gradient gives it a curved, 3-D appearance. The shadow enhances this effect. In this example, you’ll also use the :**active** pseudo-class to create an alternate type of shading when the button is depressed.

|  |
| --- |
| .button {  padding: 1em;  border: 0;  font-size: 0.8rem;  color: white;  border-radius: 0.5em;  background-image: linear-gradient(to bottom, #57b, #148);  box-shadow: 0.1em 0.1em 0.5em #124;  }  .button:active {  box-shadow: inset 0 0 0.5em #124,  inset 0 0.5em 1em rgba(0,0,0,0.4);  } |

I’ve done two new things here. Instead of a normal box shadow, I’ve added the inset keyword. This makes the shadow appear inside the border of the element, rather than outside. I’ve also added more than one shadow definition, separating them with a comma. Multiple shadows can be added in this way.

The first inset shadow (inset 0 0 0.5em #124) has offsets of zero and a slight blur. This adds a ring of shade inside the edges of the element. The second (inset 0 0.5em 1em rgba(0,0,0,0.4)) has a bit of vertical offset, making the shadow more prevalent along the top of the button. The RGBA color definition defines a semi transparent black.

## **Creating elements with a flat design**

Flat design embraces the modern world’s digital nature. It emphasizes vivid, uniform colors and a simpler

appearance. This means fewer gradients, shadows, and rounded corners. Ironically, this trend emerged only after these long-awaited effects arrived in CSS.

|  |
| --- |
| .button {  padding: 1em;  border: 0;  color: white;  background-color: #57b;  font-size: 1rem;  padding: 0.8em;  box-shadow: 0 0.2em 0.2em rgba(0, 0, 0, 0.15);  }  .button:hover {  background-color: #456ab6;  }  .button:active {  background-color: #148;  } |

### **Creating buttons with a more modern look**