Colors

Rgb

Red , green , blue

It is use for an electronics

R =0 mean 0% 255 mean max 100 %

R =0 mean 0% 255 mean max 100 %

R =0 mean 0% 255 mean max 100 %

Primary Color

When one colors combine together

Rgb(255,0,0) for red

Secondary Colors

When two colors combine together

Rgb(255,127,0)

Tertiary color

When three colors combine together

Rgb(255,255,255) black color and (0,0,0) for white colors

Hex value

A very common way to apply color to an element with CSS is with hexadecimal or hex values. While hex values sound complicated, they're really just another form of RGB values.

Hex color values start with a # character and take six characters from 0-9 and A-F. The first pair of characters represent red, the second pair represent green, and the third pair represent blue. For example, #4B5320.

In the .green class selector, set the background-color property to a hex color code with the values 00 for red, FF for green, and 00 blue.

**Linar Gradient**

**A micture of color in box called linear gradient**

**Syntax**

**Background:linear-gradient(90deg,rgb(255,0,0),rgb(0,255,0)**

**Step 47**

You may already be familiar with decimal, or base 10 values, which go from 0 - 9. Hexadecimal, or base 16 values, go from 0 - 9, then A - F:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

With hex colors, 00 is 0% of that color, and FF is 100%. So #00FF00 translates to 0% red, 100% green, and 0% blue, and is the same as rgb(0, 255, 0).

Lower the intensity of green by setting the green value of the hex color to 7F.

# Step 48

The HSL color model, or hue, saturation, and lightness, is another way to represent colors.

The CSS hsl function accepts 3 values: a number from 0 to 360 for hue, a percentage from 0 to 100 for saturation, and a percentage from 0 to 100 for lightness.

If you imagine a color wheel, the hue red is at 0 degrees, green is at 120 degrees, and blue is at 240 degrees.

Saturation is the intensity of a color from 0%, or gray, to 100% for pure color. You must add the percent sign % to the saturation and lightness values.

Lightness is how bright a color appears, from 0%, or complete black, to 100%, complete white, with 50% being neutral.

In the .blue CSS rule, use the hsl function to change the background-color property to pure blue. Set the hue to 240, the saturation to 100%, and the lightness to 50%.

# Step 50

The linear-gradient function is very flexible -- here is the basic syntax you'll use in this tutorial:

linear-gradient(gradientDirection, color1, color2, ...);

gradientDirection is the direction of the line used for the transition. color1 and color2 are color arguments, which are the colors that will be used in the transition itself. These can be any type of color, including color keywords, hex, rgb, or hsl.

Now you'll apply a red-to-green gradient along a 90 degree line to the first marker.

First, in the .red CSS rule, set the background property to linear-gradient(), and pass it the value 90deg as the gradientDirection.

# Step 51

You'll use the rgb function for the colors of this gradient.

In the linear-gradient function, use the rgb function to set the first color argument to pure red.

# Step 52

You won't see gradient yet because the linear-gradient function needs at least two color arguments to work.

In the same linear-gradient function, use the rgb function to set the second color argument to pure green.

# Step 49

You've learned a few ways to set flat colors in CSS, but you can also use a color transition, or gradient, on an element.

A gradient is when one color transitions into another. The CSS linear-gradient function lets you control the direction of the transition along a line, and which colors are used.

One thing to remember is that the linear-gradient function actually creates an image element, and is usually paired with the background property which can accept an image as a value.

In the .red CSS rule, change the background-color property to background.

# Step 53Passed

As you can see, the linear-gradient function produced a smooth red-green gradient. While the linear-gradient function needs a minimum of two color arguments to work, it can accept many color arguments.

Use the rgb function to add pure blue as the third color argument to the linear-gradient function.

# Step 54

Color-stops allow you to fine-tune where colors are placed along the gradient line. They are a length unit like px or percentages that follow a color in the linear-gradient function.

For example, in this red-black gradient, the transition from red to black takes place at the 90% point along the gradient line, so red takes up most of the available space:

linear-gradient(90deg, red 90%, black);

In the linear-gradient function, add a 75% color stop after the first red color argument. Do not add color stops to the other colors arguments.

# Step 55

Now that you know the basics of how the linear-gradient function and color-stops work, you can use them to make the markers look more realistic.

In the linear-gradient function, set gradientDirection to 180deg.

# Step 57

Now that the color-stops are set, you'll apply different shades of red to each color argument in the linear-gradient function. The shades on the top and bottom edges of the marker will be darker, while the one in the middle will be lighter, as if there's a light above it.

For the first color argument, which is currently pure red, update the rgb function so the value for red is 122, the value for green is 74, and the value for blue is 14.

# Step 58Passed

Now modify the second color argument in the linear-gradient function, which is currently pure green.

Update the rgb function so the value for red is 245, the value of green is 62, and the value of blue is 113.

# Step 60Passed

The red marker is looking much more realistic. Now you'll do the same for the green marker, using a combination of the linear-gradient function and hex colors.

In the .green CSS rule, change the background-color property to background.

# Step 61

For this marker, you'll use hex color codes for your gradient.

Use the linear-gradient function and set gradientDirection to 180deg. And for the first color argument, use a hex color code with the values 55 for red, 68 for green, and 0D for blue.

# Step 59

Finally, modify the third color argument in the linear-gradient function, which is currently pure blue.

Update the rgb function so the value for red is 162, the value of green is 27, and the value of blue is 27.

Opacity describes how opaque, or non-transparent, something is. For example, a solid wall is opaque, and no light can pass through. But a drinking glass is much more transparent, and you can see through the glass to the other side.

With the CSS opacity property, you can control how opaque or transparent an element is. With the value 0, or 0%, the element will be completely transparent, and at 1.0, or 100%, the element will be completely opaque like it is by default.

In the .sleeve CSS rule, set the opacity property to 0.5.

# Step 78

It looks like your sleeve disappeared, but don't worry -- it's still there. What happened is that your new cap div is taking up the entire width of the marker, and is pushing the sleeve down to the next line.

This is because the default display property for div elements is block. So when two block elements are next to each other, they stack like actual blocks. For example, your marker elements are all stacked on top of each other.

To position two div elements on the same line, set their display properties to inline-block.

Create a new rule to target both the cap and sleeve classes, and set display to inline-block.

# Step 79

In the last project, you learned a little bit about borders and the border-color property.

All HTML elements have borders, though they're usually set to none by default. With CSS, you can control all aspects of an element's border, and set the border on all sides, or just one side at a time. For a border to be visible, you need to set its width and style.

In the .sleeve CSS rule, add the border-left-width property with the value 10px.

# Step 81

Your border should be visible now. If no color is set, black is used by default.

But to make your code more readable, it's better to set the border color explicitly.

In the .sleeve CSS rule, add the border-left-color property with the value black.

# Step 82

The border-left shorthand property lets you to set the left border's width, style, and color at the same time.

Here is the syntax:

border-left: width style color;

In the .sleeve CSS rule, replace the border-left-width, border-left-style, and border-left-color properties with the border-left shorthand property. The values for the width, style, and color of the left border should be the same.

# Step 84

The black color of your border looks pretty harsh against the more transparent sleeve. You can use an alpha channel to lower the opacity of the black border.

For the border-left shorthand property, use the rgba function to set the color value to pure black with 75% opacity.

# Step 85

Awesome. Your red marker is looking good. Now all you need to do is add the caps and sleeves to your other markers.

Add a cap and sleeve to both the green and blue markers. You can just copy the div elements from the red marker and paste them into the other two markers.

**Step 86**

The last thing you'll do is add a slight shadow to each marker to make them look even more realistic.

The box-shadow property lets you apply one or more shadows around an element. Here is basic syntax:

box-shadow: offsetX offsetY color;

Here's how the offsetX and offsetY values work:

* both offsetX and offsetY accept number values in px and other CSS units
* a positive offsetX value moves the shadow right and a negative value moves it left
* a positive offsetY value moves the shadow down and a negative value moves it up
* if you want a value of zero (0) for any or both offsetX and offsetY, you don't need to add a unit. Every browser understands that zero means no change.

The height and width of the shadow is determined by the height and width of the element it's applied to. You can also use an optional spreadRadius value to spread out the reach of the shadow. More on that later.

Start by adding a simple shadow to the red marker.

In the .red CSS rule, add the box-shadow property with the values 5px for offsetX, 5px for offsetY, and red for color.

.red {

  background: linear-gradient(rgb(122, 74, 14), rgb(245, 62, 113), rgb(162, 27, 27));

  box-shadow:5px 5px red;

}

# Step 87

As you can see, you added a simple red shadow around your marker that's 5 pixels to the right, and 5 pixels down.

But what if you wanted to position your shadow on the opposite side? You can do that by using negative values for offsetX and offsetY.

Update the values for the box-shadow property, and set offsetX to -5px, and offsetY to -5px.

Example :

.red {

  background: linear-gradient(rgb(122, 74, 14), rgb(245, 62, 113), rgb(162, 27, 27));

  box-shadow: -5px -5px red;

}

# Step 88

Notice that the edges of the shadow are sharp. This is because there is an optional blurRadius value for the box-shadow property:

box-shadow: offsetX offsetY blurRadius color;

If a blurRadius value isn't included, it defaults to 0 and produces sharp edges. The higher the value of blurRadius, the greater the blurring effect is.

In the .green CSS rule, add the box-shadow property with the values 5px for offsetX, 5px for offsetY, 5px for blurRadius, and green for color.

Example

.green {

  background: linear-gradient(#55680D, #71F53E, #116C31);

  box-shadow:5px 5px 5px green;

}

# Step 89

But what if you wanted to expand the shadow out further? You can do that with the optional spreadRadius value:

box-shadow: offsetX offsetY blurRadius spreadRadius color;

Like blurRadius, spreadRadius defaults to 0 if it isn't included.

Practice by adding a 5 pixel shadow directly around the blue marker.

In the .blue CSS rule, add the box-shadow property with the values 0 for offsetX, 0 for offsetY, 0 for blurRadius, 5px for spreadRadius, and blue for color.

Example

.blue {

  background: linear-gradient(hsl(186, 76%, 16%), hsl(223, 90%, 60%), hsl(240, 56%, 42%));

  box-shadow: 0 0 0 5px blue;

}

# Step 90

Now that you're familiar with the box-shadow property you can finalize the shadows, starting with the one for the red marker.

In the .red CSS rule, update the values for the box-shadow property so offsetX is 0,offsetY is 0, blurRadius is 20px, spreadRadius is 0, and color is red. Remember that you don't need to add units to a zero value

# Step 91

Next, update the color value of the red marker's box-shadow property.

Replace the named color with the rgba function. Use the values 83 for red, 14 for green, 14 for blue and 0.8 for the alpha channel.

Example

.red {

  background: linear-gradient(rgb(122, 74, 14), rgb(245, 62, 113), rgb(162, 27, 27));

  box-shadow: 0 0 20px 0 red;

}

# Step 92

The shadows for your green and blue markers will have the same position, blur, and spread. The only difference will be the colors.

In the .green and .blue CSS rules, update the values for the box-shadow properties so offsetX is 0,offsetY is 0, blurRadius is 20px, and spreadRadius is 0. Leave the colors as green and blue for now.

# Step 93Passed

For the green marker's box-shadow property, replace the named color with a hex color code. Use the values 3B for red, 7E for green, 20 for blue, and CC for the alpha channel.

Example

.green {

  background: linear-gradient(#55680D, #71F53E, #116C31);

  box-shadow: 0 0 20px 0 #3b7e20cc;

}

# Step 94

Finally, for the blue marker's box-shadow property, replace the named color with the hsla function. Use the values 223 for hue, 59% for saturation, 31% for lightness, and 0.8 for the alpha channel.

And with that, your set of colored markers is complete! Well done.

blue {

  background: linear-gradient(hsl(186, 76%, 16%), hsl(223, 90%, 60%), hsl(240, 56%, 42%));

  box-shadow: 0 0 20px 0 hsla(223,59%,31%,0.8);

}