LEARN THE CSS BOX MODEL BY BUILDING A ROTHKO PAINTING

Introduction

Every HTML element is its own box – with its own spacing and a border. This is called the Box Model.

In this course, you'll use CSS and the Box Model to create your own Rothko-style rectangular art pieces.

Step 1:

By now, you should be familiar with the basic elements an HTML page should have. Set up your code with a `DOCTYPE` declaration, an `html` element with the language set to English, a `head` element, and a `body` element.

Step 2:

Within the `head` element, add a `meta` tag which sets the `charset` to `UTF-8`, and a `title` element with the value `Rothko Painting`.

Within the `body` element, add an `img` element with a `src` of https://cdn.freecodecamp.org/curriculum/css-box-model/diagram-1.png.

Step 3:

In the CSS box model, every HTML element is treated as a box with four areas.

Imagine you receive a box from your favorite online retailer -- the content is the item in the box, or in our case, a header, paragraph, or image element.

Change the `src` attribute in the `<img>` from `https://cdn.freecodecamp.org/curriculum/css-box-model/diagram-1.png` to https://cdn.freecodecamp.org/curriculum/css-box-model/diagram-2.png.

Step 4:

The content is surrounded by a space called padding, similar to how bubble wrap separates an item from the box around it.

Think of the border like the cardboard box your item was shipped in.

Change the `src` attribute to https://cdn.freecodecamp.org/curriculum/css-box-model/diagram-3.png

Step 5:

Margin is the area outside of the box, and can be used to control the space between other boxes or elements.

Here the bottom element has a larger top margin, pushing it further down the page.

Now that you understand the CSS box model, let's get started on the Rothko painting. Remove the `<img>` element.

Step 6:

Add a `div` element in the `body`. Set the `class` attribute equal to `canvas`. This will act as the canvas for your painting.

Step 7:

Before you can start styling the `div` you added, you need to link your CSS to your HTML.

Add a `link` element to link your `styles.css` file. Set the `href` to `styles.css`, and remember to set the `rel` attribute to `stylesheet`.

Step 8:

Time for CSS.

Even though your `<div>` has no text, it's still treated as a box with content. Write a CSS rule that uses the `.canvas` class selector and set its `width` to 500 pixels. Here's a CSS rule that sets the width of the class `card` to 300 pixels:

Example Code:

```
.card {
  width: 300px;
}
```

Step 9:

Add the `height` property with the value `600px` to your `.canvas` rule.

Step 10:

Change the `background-color` of the canvas to `#4d0f00`.

Step 11:

Every painting needs a frame. Wrap the .canvas element in another div. Give that div the frame class.

Step 12:

Write a new rule using the .frame class selector. Use the border shorthand declaration to give the .frame element a solid, black border with a width of 50px.

Step 13:

The frame is much too wide. In .frame, set its width to 500 pixels.

Step 14:

Use padding to adjust the spacing within an element.

In .frame, use the padding shorthand property to increase the space between the .frame and .canvas elements by 50px. The shorthand will increase space in the top, bottom, left, and right of the element's border and canvas within.

Step 15:

Use margins to adjust the spacing outside of an element.

Using the margin property, give the .frame element vertical margin of 20px, and horizontal margin of auto. This will move the frame down 20 pixels and horizontally center it on the page.

Step 16:

Add a new div element inside of your .canvas element. Give the new div the class attribute with a value of one. This will be your first rectangle.

Step 17:

Write a new rule that targets .one and set its width to 425 pixels.

Step 18:

Now set the height for .one to 150 pixels.

Step 19:

Set the background-color of .one to #efb762.

Step 20:

Use margins to position the .one element on the canvas.

Add the shorthand margin property with a vertical margin of 20px and a horizontal margin of auto.

Step 21:

Now .one is centered horizontally, but its top margin is pushing past the canvas and onto the frame's border, shifting the entire canvas down 20 pixels.

Add padding of 1px to the .canvas element to give the .one element something solid to push off of.

Step 22:

Adding 1 pixel of padding to the top, bottom, left, and right of the canvas changed its dimensions to 502 pixels x 602 pixels.

Replace the `padding` property with `overflow` set to `hidden` - changing the canvas back to its original dimensions.

Step 23:

Add another `div` with a `class` value of `two` just below your `one` element. This will be your second rectangle.

Step 24:

Create a new CSS rule using the `.two` selector and set its `width` to 475 pixels.

Step 25:

Set the `height` of the `.two` element to 200 pixels.

Step 26:

Set the `background-color` of the `.two` element to `#8f0401`.

Step 27:

Center the `.two` element by setting its `margin` to `auto`.

Step 28:

Create a new div with a class value of three right under the .two element. This will be your third rectangle.

Step 29:

You don't always have to use pixels when sizing an element. Create a new rule, .three, and set its width to 91%.

Step 30:

Set the height of .three to 28%.

Step 31:

Change the background-color of .three to #b20403.

Step 32:

Center the .three element on the canvas by setting its margin to auto.

Step 33:

It's helpful to have your margins push in one direction.

In this case, the bottom margin of the .one element pushes .two down 20 pixels.

In the .two selector, use margin shorthand property to set top margin to 0, horizontal margin to auto, and bottom margin to 20px. This will remove its top margin, horizontally center it, and set its bottom margin to 20 pixels.

Step 34:

The colors and shapes of your painting are too sharp to pass as a Rothko.

Use the `filter` property to `blur` the painting by `2px` in the `.canvas` element.

Here's an example of a rule that add a 3px `blur`:

Example Code:

```
p {

  filter: blur(3px);

}
```

Step 35:

Create a rule that targets both `.one` and `.two` and increase their `blur` effect by 1 pixel.

Step 36:

Increase the `blur` of `.three` by 2 pixels.

Step 37:

The rectangles are too small and their edges don't have the soft quality of a painting.

Increase the area and soften the edges of `.one` by setting its `box-shadow` to `0 0 3px 3px #efb762`.

Step 38:

Use the same box-shadow declaration for .two, but change the color from #efb762 to #8f0401.

Step 39:

Add a box-shadow to .three with the values 0 0 5px 5px #b20403.

Step 40:

The corners of each rectangle are still too sharp.

Round each corner of the .one element by 9 pixels, using the border-radius property.

Step 41:

Use the border-radius property on the .two selector, to set its top-left radius and bottom-right radius to 8px, and top-right radius and bottom-left radius to 10px.

Step 42:

The border-radius property accepts up to four values to round the top-left, top-right, bottom-right, and bottom-left corners.

Round the top-left corner of .three by 30 pixels, the top-right by 25 pixels, the bottom-right by 60 pixels, and bottom-left by 12 pixels.

Step 43:

Rotate each rectangle to give them more of an imperfect, hand-painted look.

Use the `transform` property on the `.one` selector to `rotate` it counter clockwise by `0.6` degrees.

Step 44:

Rotate the `.two` element clockwise by `0.4` degrees.

Step 45:

Rotate `.three` counter clockwise by `0.2` degrees.

With this final step, your Rothko painting is now complete.