

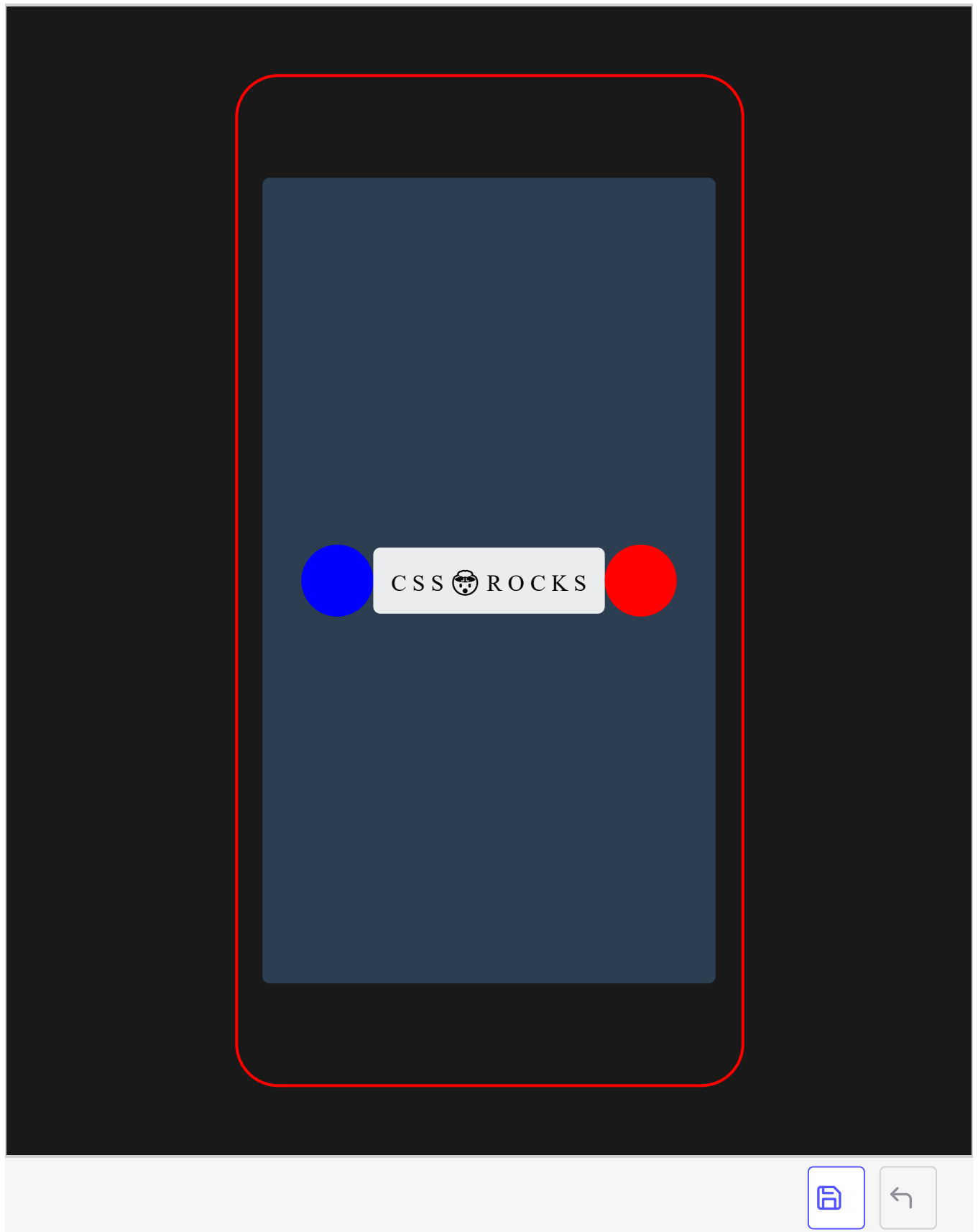
# Putting Positioning to Practice

We learned a great deal about CSS positioning in the last lesson. In this lesson, we will apply some of the new knowledge gained to finsih off the iPhone project.

The code playground below shows the current state of the iPhone project.

Here it is:

Output
HTML
CSS (SCSS)



You remember how we got here, right?

## Positioning the Buttons

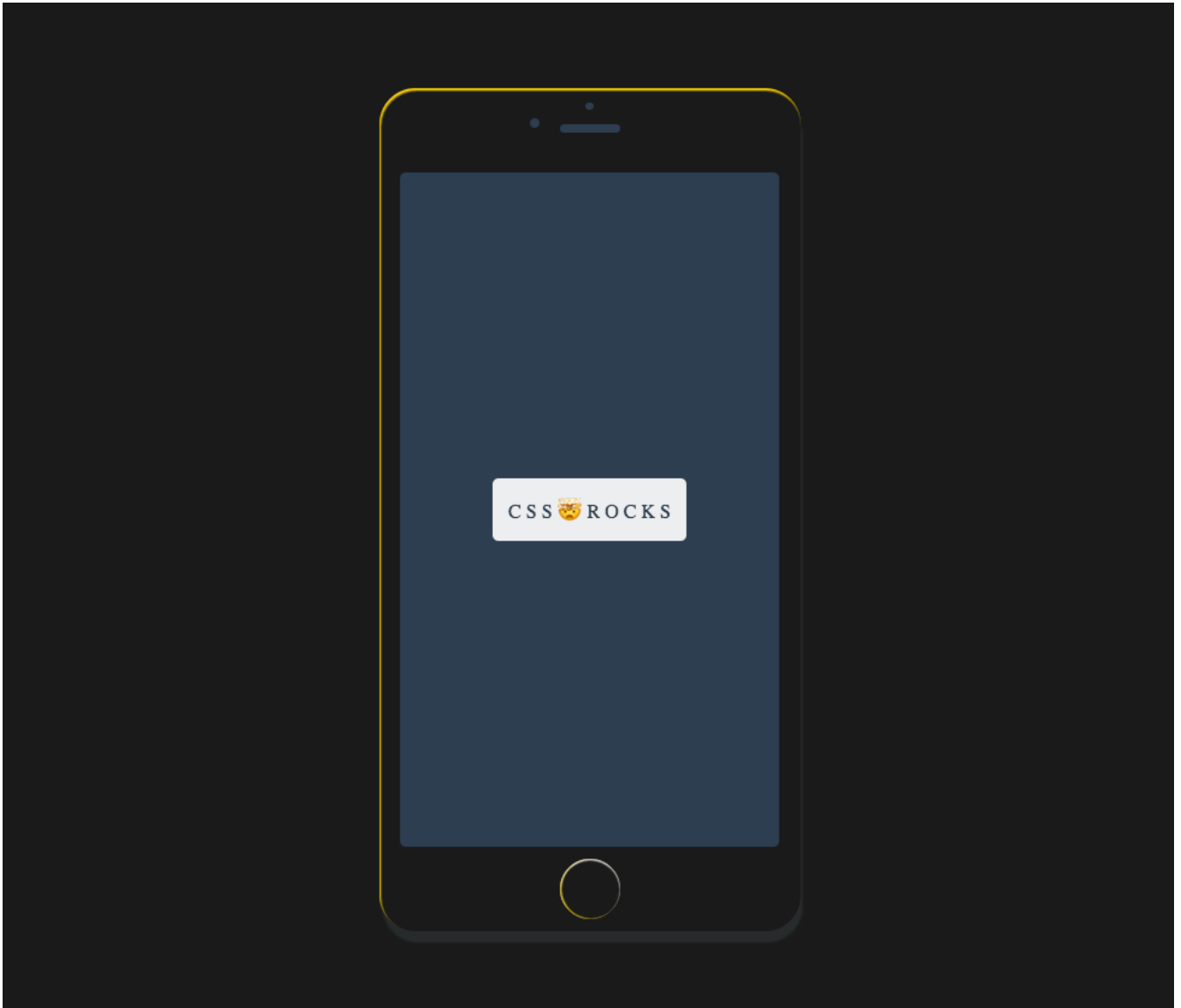
No phone has its buttons sitting in the center of the screen. That's just crazy.

As a refresher, the blue and red dots above are as a result of styling the `:before` and `:after` pseudo-element of `.phone-inner`

```
1 <html>
2   <head>
3     <title> Teach me positioning
4   </head>
5   <body>
6     <section class="phone-body">
7       <div class="phone-inner">
8         <article>
9           C S S 🍌 R O C K S
10        </article>
11      </div>
12    </section>
13  </body>
14 </html>
```

Now we need to position the pseudo-elements, `:before` and `:after` to sit in the right places.

So we don't lose sight of where we are headed, here is the final result again.



In most cases, positioning begins with 2 objects, right?

1. The object to be positioned
2. The reference object i.e the object with the positioning context.

Let's see if you can state what objects are present in this case.



I had said earlier that elements with `position: relative` set up a positioning context for every corresponding child element.

I must state that they also set up a positioning context for their pseudo-elements, `:before` and `:after`. Since they kind of act as child elements, this is expected.

## Enough Talking, Let's position this thing!

Select the reference object and position it relatively.

```
.phone-inner {  
  position: relative;  
}
```

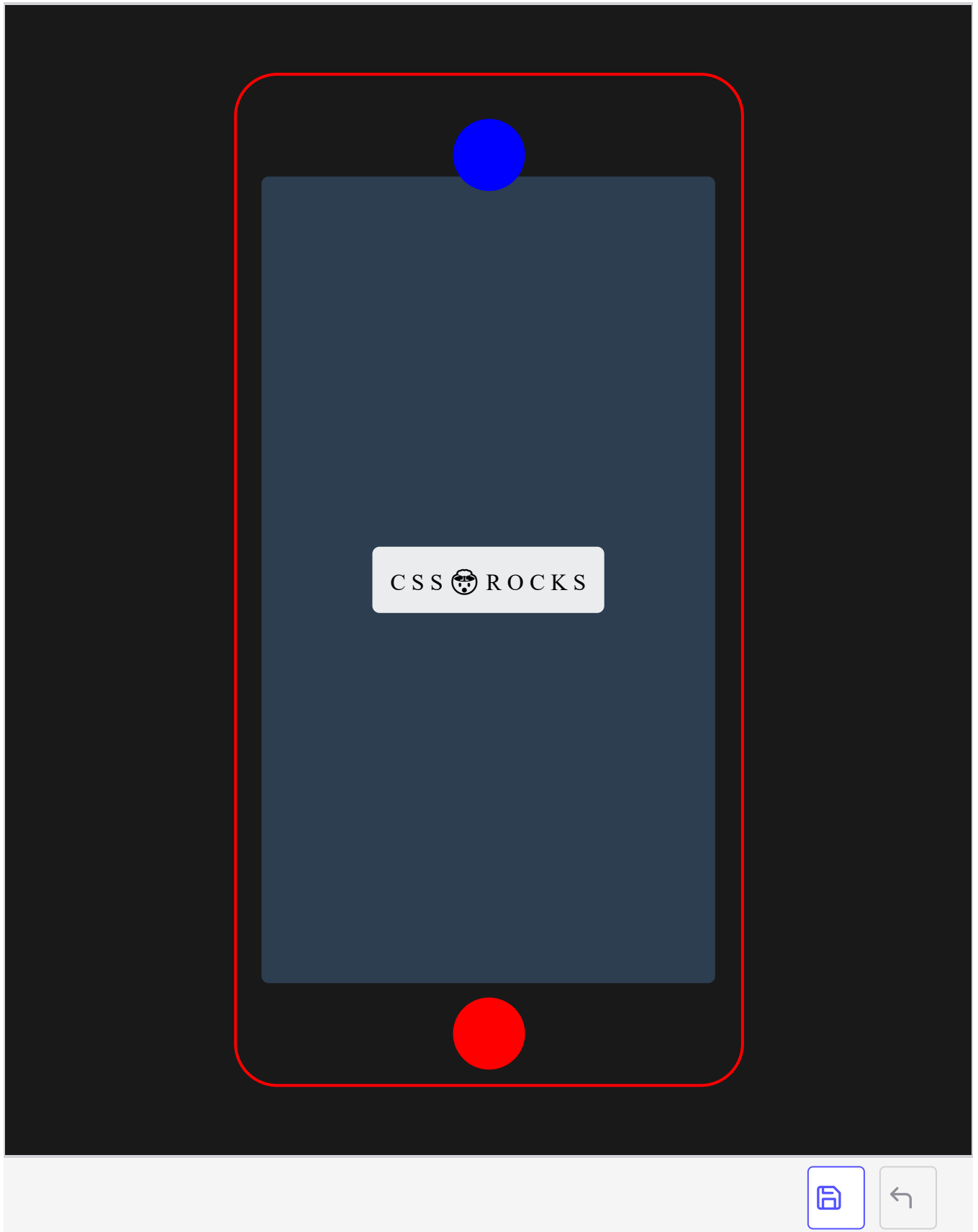
Now position the pseudo-element within the created positioning context.

```
.phone-inner:before {  
  position: absolute;  
  top: -40px;  
}  
  
.phone-inner:after {  
  position: absolute;  
  bottom: -60px;  
}
```

Don't forget to include `position: absolute` as done above or the pseudo-elements will not take part in the positioning context.

See the results below:

Output
HTML
CSS (SCSS)

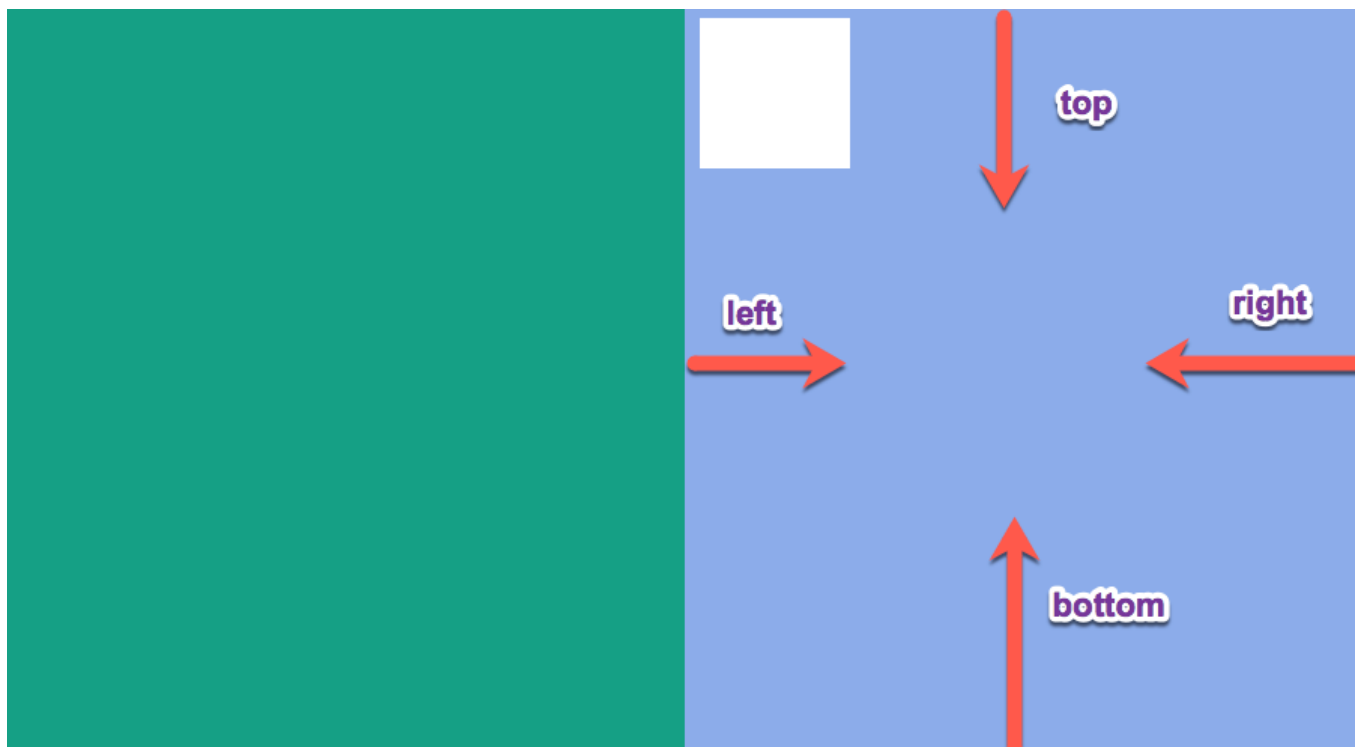


That worked. Awesome!

We've got to celebrate that 🍰 🍰 🍰 🍰

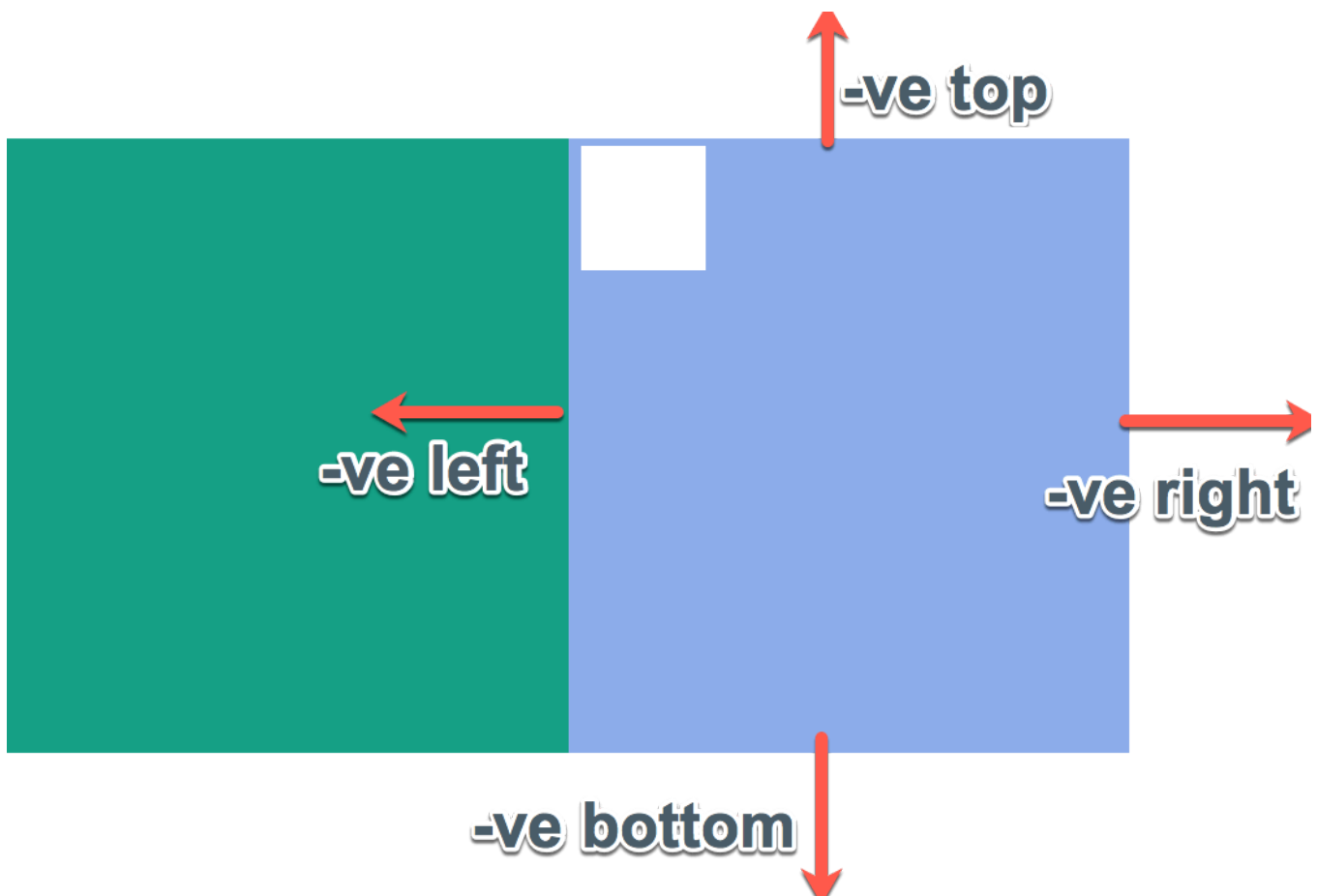
Notice that I used negative `top` and `bottom` values. This places the elements in the opposite direction.

Remember the positive directions are as seen below:



Positive directions.

Therefore, `top: -40px` will position the `.phone-inner:before` 40px from the top of `.phone-inner` in the opposite direction of what is shown above.



## How does that work?

In case you missed out on the party and don't understand why the code above worked, here is how.

A positioning context sets up a cartesian coordinate for which the relatively positioned element and its child element may be positioned.

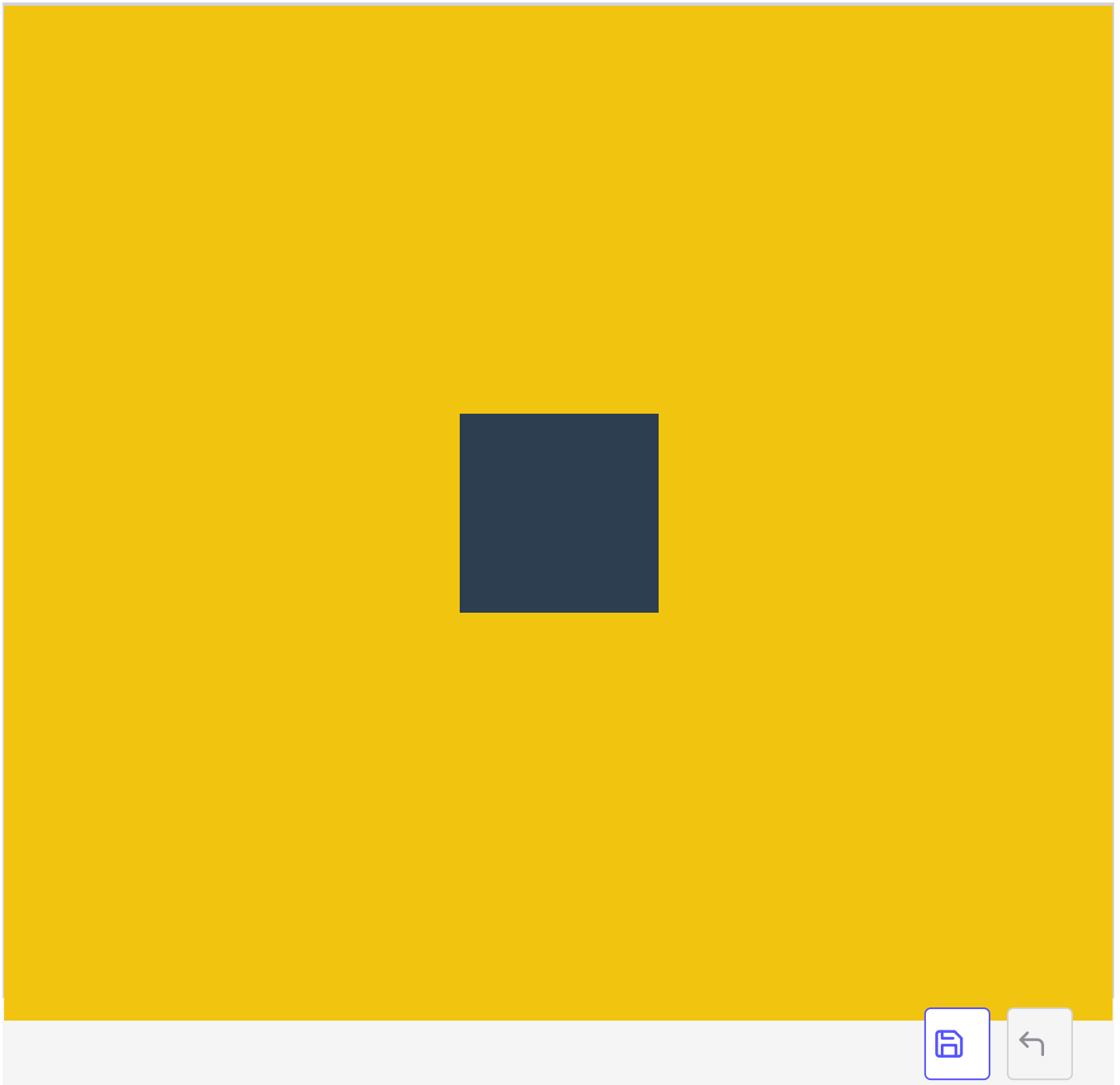
Remember that the positioning is done via the keywords, `top`, `bottom`, `left` and `right`

It is also important to note how positive and negative values influence the positioning of elements within the positioning context.

Consider the simple example below where we have a dark box sitting in the center of the page.

Output
HTML
CSS (SCSS)





I have gone ahead to set `position: relative` on the dark box. A positioning context has been established.

Now you can go ahead and position the element with respect to this positioning context.

Toy with the `top` and `left` values below to see how each positioning keyword value nudges the element within the positoning context.

Go ahead.

I have placed the playground side by side with the output so it's easier to note the changes. The previous position of the dark box is also colored so you'll know the extents to which the element has moved.

For a start, try `left: 30px` and `left: -30px`

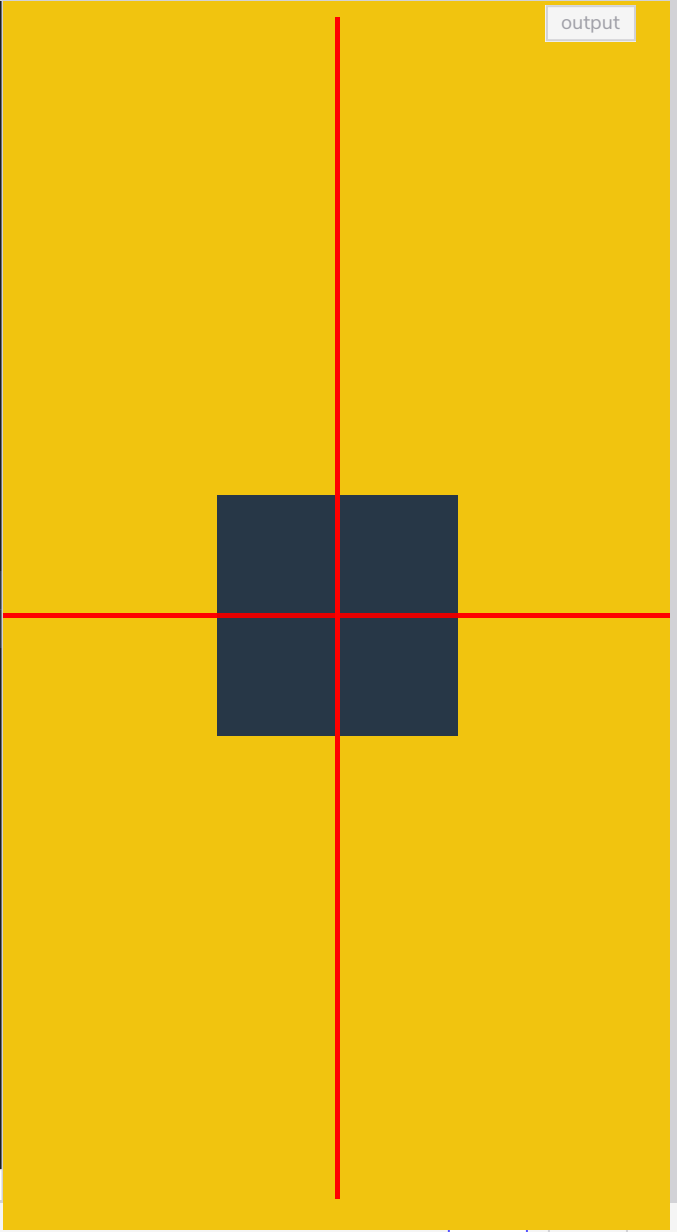
HTML

CSS

Output

```
1 body {
2   min-height: 100vh;
3   background: #f1c40f;
4   display: flex;
5   justify-content: center;
6   align-items: center;
7 }
8
9
10 .dark-box {
11   background: #2c3e50;
12   width: 120px;
13   height: 120px;
14
15   position: relative;
16   top: 0;
17   left: 0;
18 }
19
20
21 .cartesian-x {
22   position: absolute;
23   width: 100vw;
24   height: 2px;
25   background: red;
26   display: flex;
27   justify-content: center;
28   align-items: center;
29 }
30 .cartesian-x:after {
31   content: " ";
```

output



Save

↶

Again, this isn't as difficult as it seems. Use the playground above, and also make reference to the graphics that show the directions of the positioning.

You'll get the hang of it!