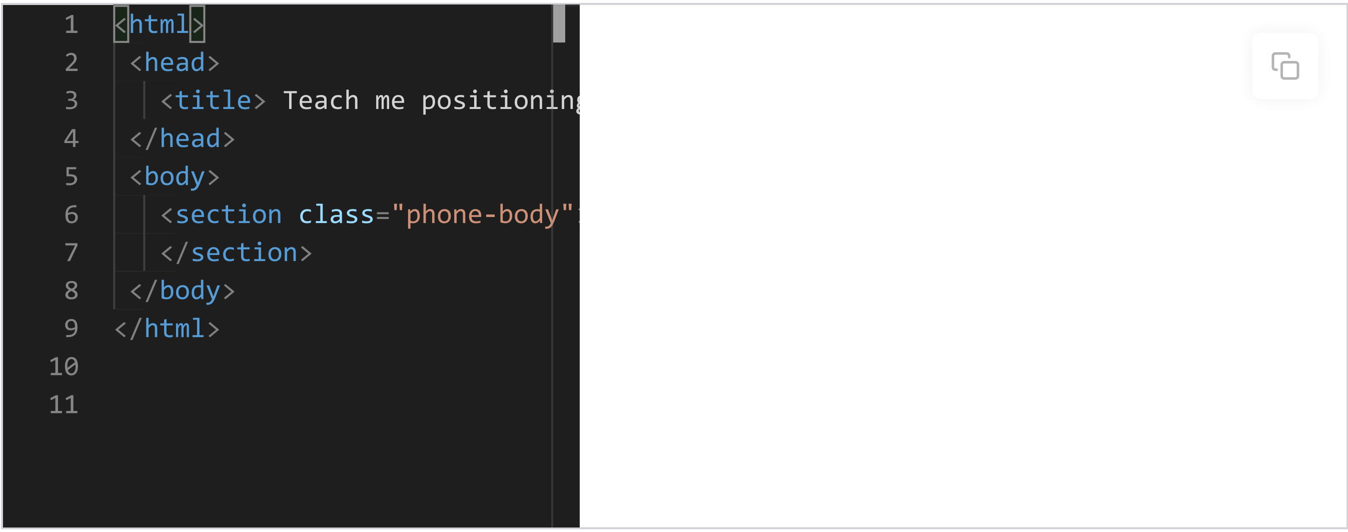
Beginning the Project

Before exploring positioning in CSS I want to get you started on the project we will build in this section.

Getting Started

The project begins with the modest markup below:



```
<html>
  <head>
    <title> Teach me positionin
  </head>
  <body>
    <section class="phone-body"
    </section>
  </body>
</html>


```

As you may have guessed, `.phone-body` refers to the outer phone container.

I'll go ahead and give it a `width` and `height`:

```
  width: 350px;
height: 700px;
```

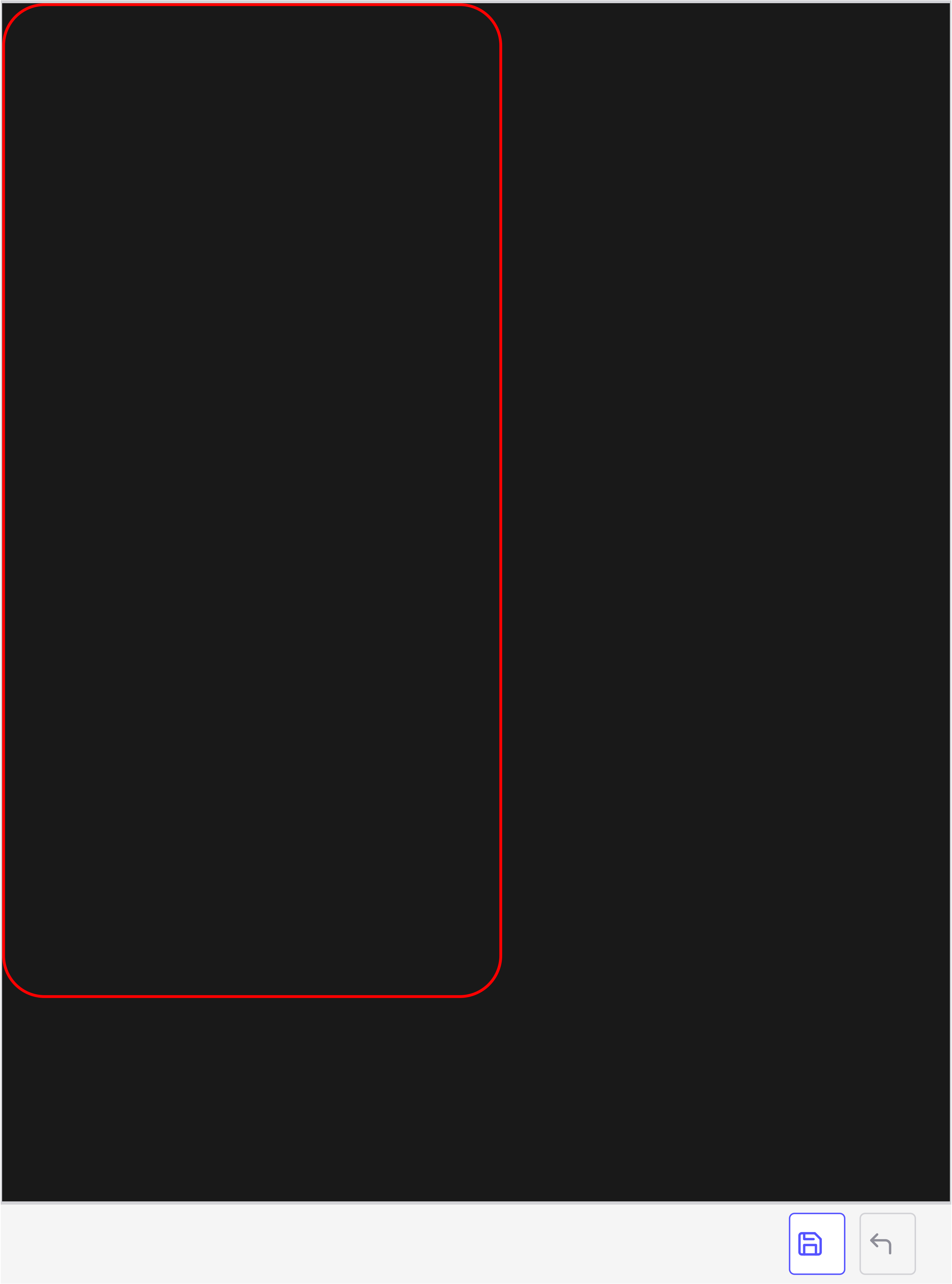
Well, that's good but not good enough to get anyone excited.

Let's display something on the screen.

Output

HTML

CSS (SCSS)



Not bad.

Be sure to check the code tabs above.

The `html` is the same as the former. No additions made here.

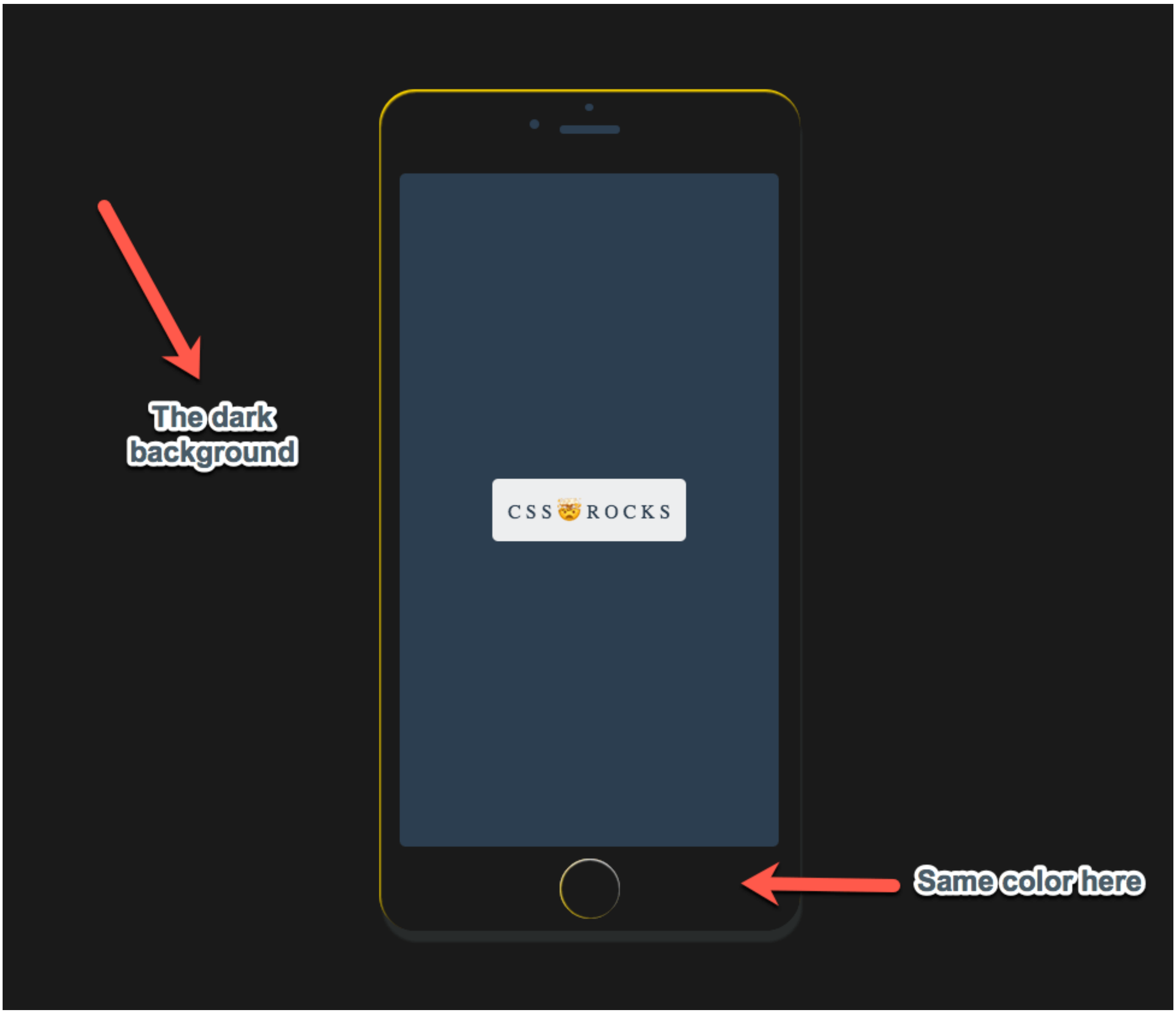
On the `css` side of things though, there's a bit in there. Look closely, and you'll realize all of that is familiar terrrain.

CSS

```css
:root {
  --secondary-bg: rgba(0,0,0,0.9);
}

body {
  min-height: 100vh;
  margin: 0;
  font-family: serif;
  background: var(--secondary-bg);
}

.phone-body {
  width: 350px;
  height: 700px;

  border-radius: 30px;
  border: 2px solid red;
}
```

First off there's a color variable on line 2. Gosh, I love variables.

The variable is used to give the `body` a dark background, as seen on line 9. From previous lessons, you should know that the color yields an almost black color.



The CSS goodie doesn't end there.

I have made the `body` element take the entire height of the user's viewport with `100vh`

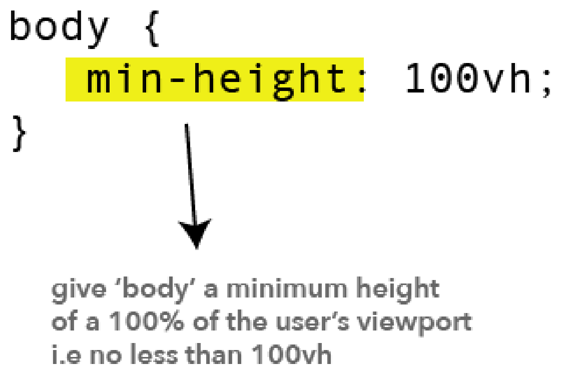
You see how all the past lessons come in handy?

CSS

```css
:root {
  --secondary-bg: rgba(0,0,0,0.9);
}

body {
  min-height: 100vh;
  margin: 0;
  font-family: serif;
  background: var(--secondary-bg);
}

.phone-body {
```

```
.phone-body {
  width: 350px;
  height: 700px;

  border-radius: 30px;
  border: 2px solid red;
}
```

Note that I used the CSS property, `min-height`. This stands for *minimum height*



```
body {
   min-height: 100vh;
}
```

give 'body' a minimum height
of a 100% of the user's viewport
i.e no less than 100vh

Be honest, you understand every other bit of CSS in there. Right? The phone is given a `border-radius` of `30px`. That gives the phone rounded edges. The `margin` on `body` is also removed because by default browsers allow some margin space. We don't need that.

Good ol' stuff.

Let's keep going.

Centering the Phone

Take a quick glance at the complete result, and you'll notice that the `phone` is perfectly centered in the center.

What, you say Flexbox?

Hell, yeah!

```
body {

  display: flex;
  justify-content: center;
  align-items: center;
}
```

This will have the phone centered within the page as it is the direct child of `body`

Output

HTML

CSS (SCSS)

```
:root {
  --secondary-bg: rgba(0,0,0,0.9);
}

body {
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  margin: 0;
  font-family: serif;
  background: var(--secondary-bg);


}

.phone-body {
  width: 350px;
  height: 700px;

  border-radius: 30px;
  border: 2px solid red;
}
```

Click the output tab above to see the perfectly centered phone.

Oh my! This is coming to live now.

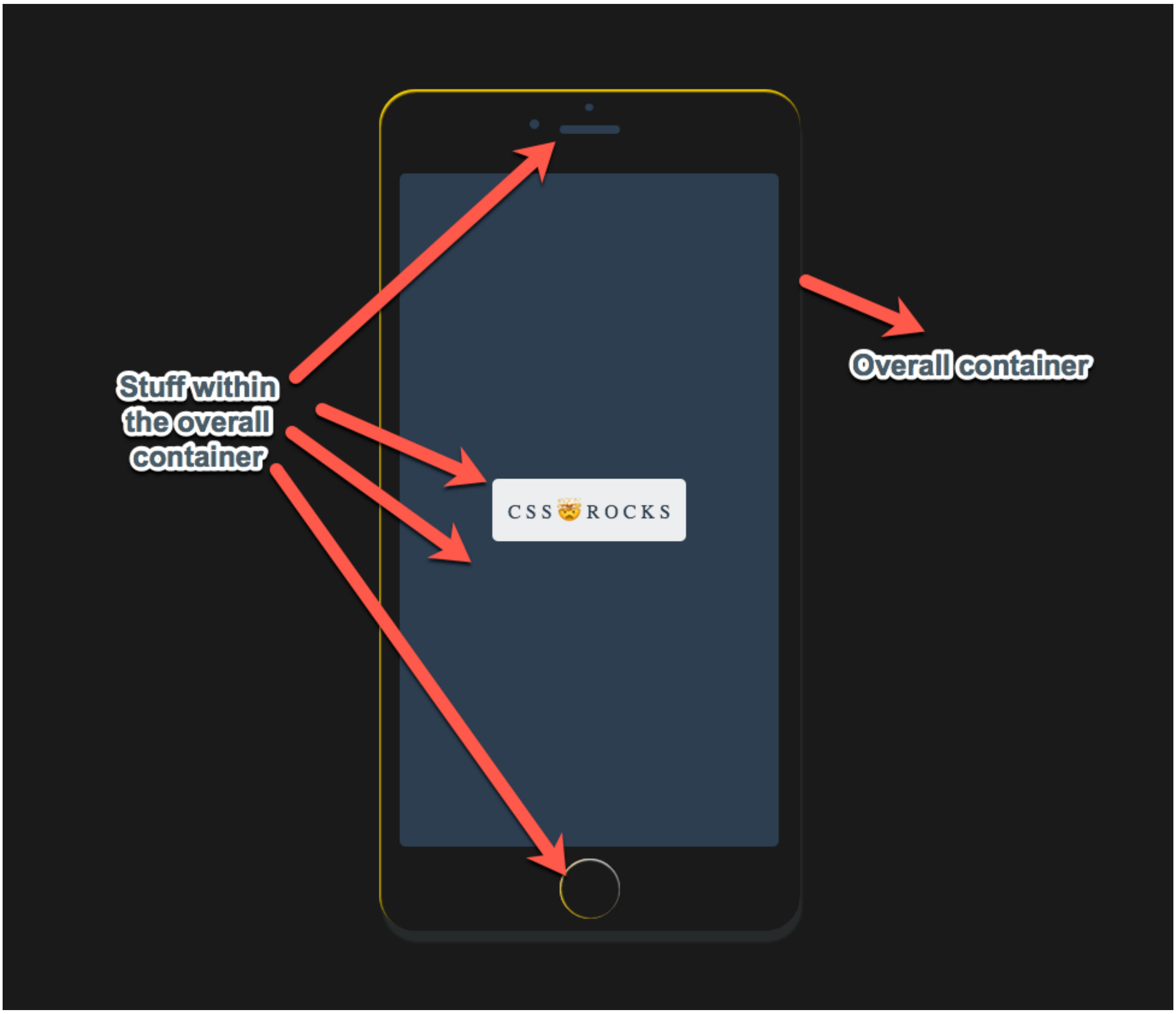
Content is King?

Let’s put in some content.

HTML

```
<html>
 <head>
   <title> Teach me positioning </title>
 </head>
 <body>
   <section class="phone-body">
     <div class="phone-inner">
      <article>
         C S S 🤯 R O C K S
      </article>
    </div>
   </section>
 </body>
</html>
```

You know we need some stuff in the phone

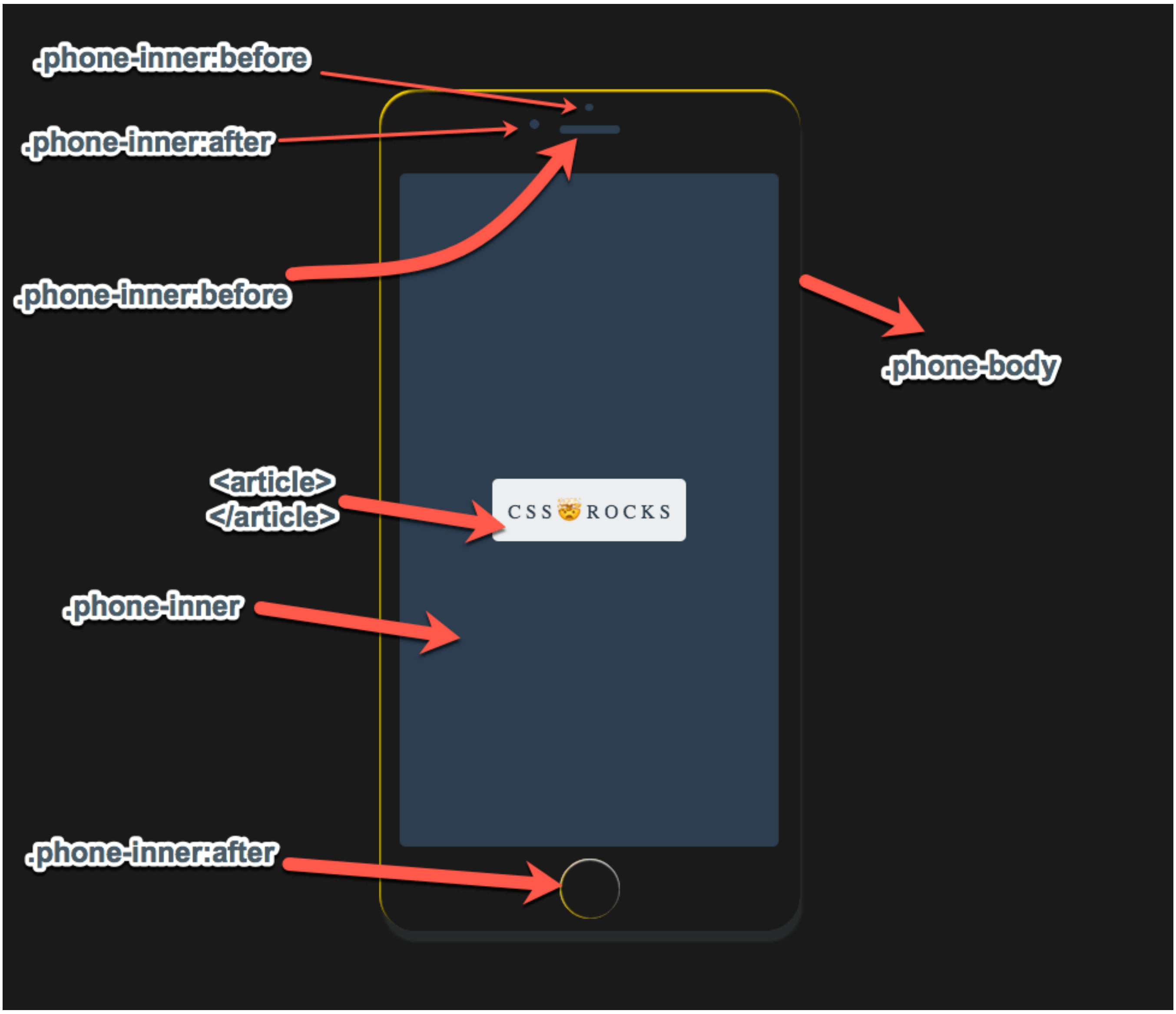


The cool thing is that the markup I shared above is all we need to bring the iPhone to life!

Really?

In the graphic below, I have represented the components of the phone with their corresponding CSS class names.

Have a look.



Don't concern yourself with this if you don't get it at a glance. I'll be explaining every process along the way.

Welcome to the cool world of CSS. We will be creating the iPhone with as little markup as possible.

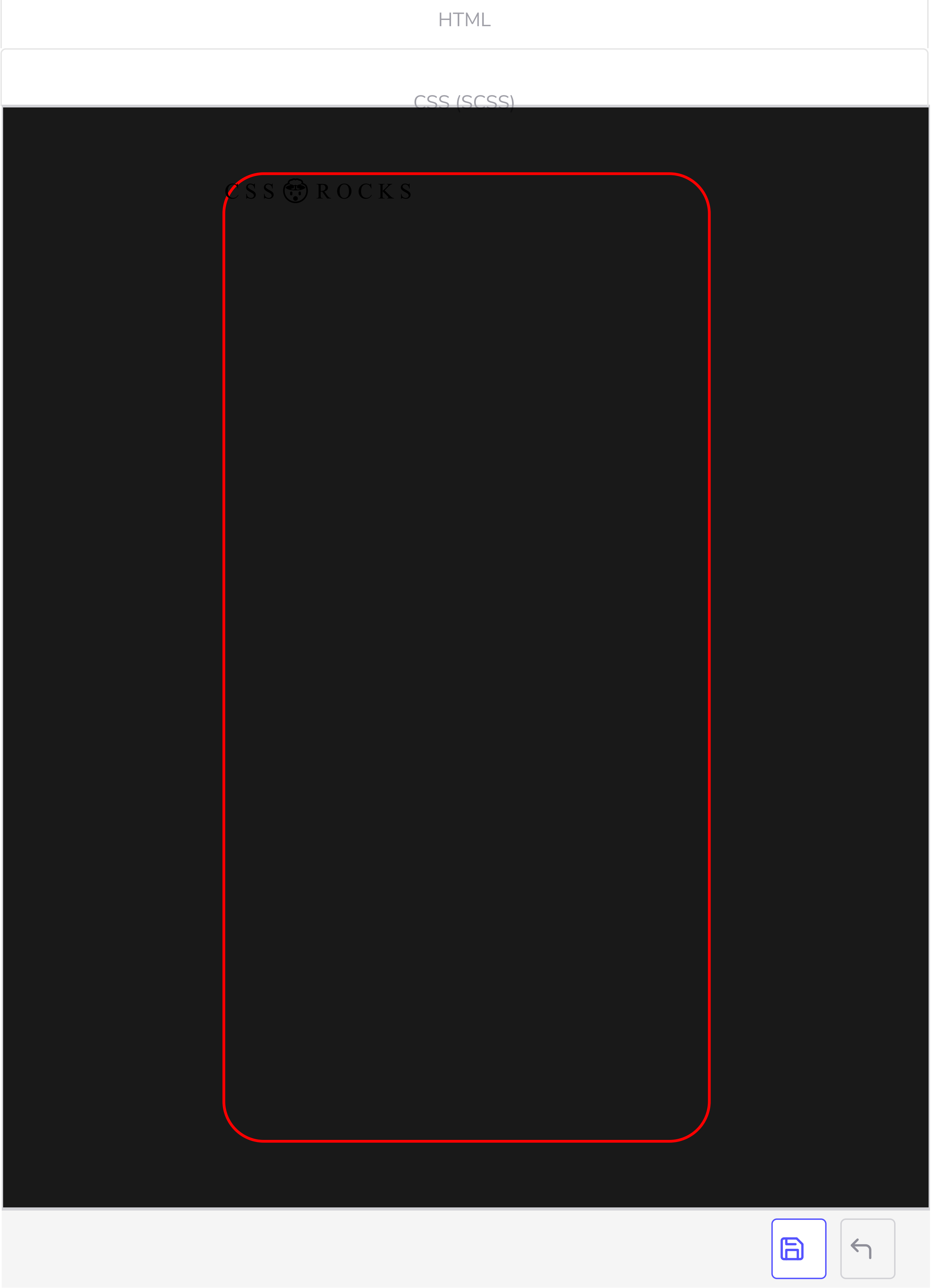
We will take advantage of the CSS pseudo-elements, `:before` and `:after` to create the iPhone’s home button and speaker.

Hang on!

Styling the Inner Content

This one is simple. Here is the current state of the project:

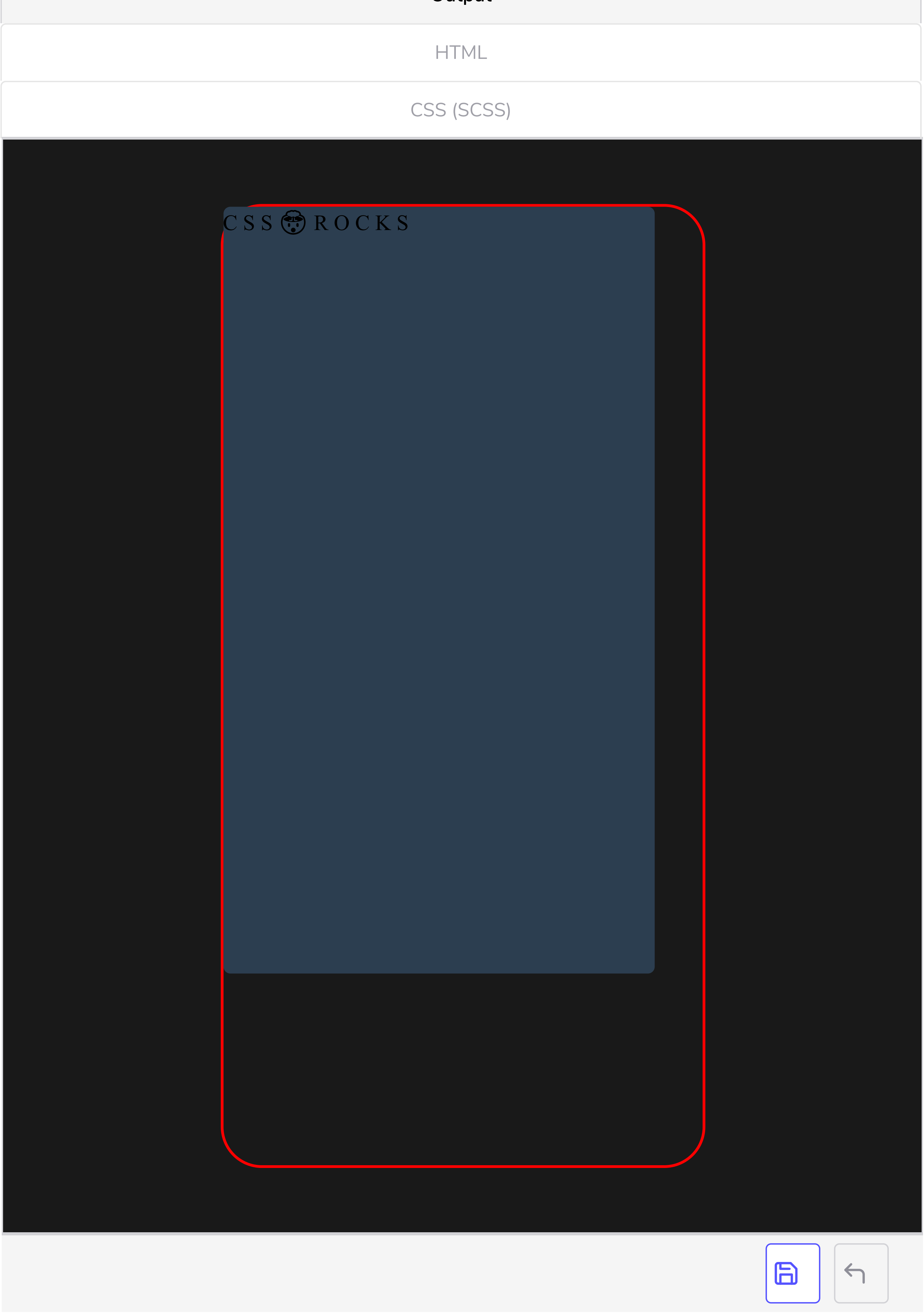
Output



I have gone ahead to give `.phone-inner` a width, height, blue background color and some `border-radius` below.

Be sure to hit the CSS pane to see the highlihted CSS code.

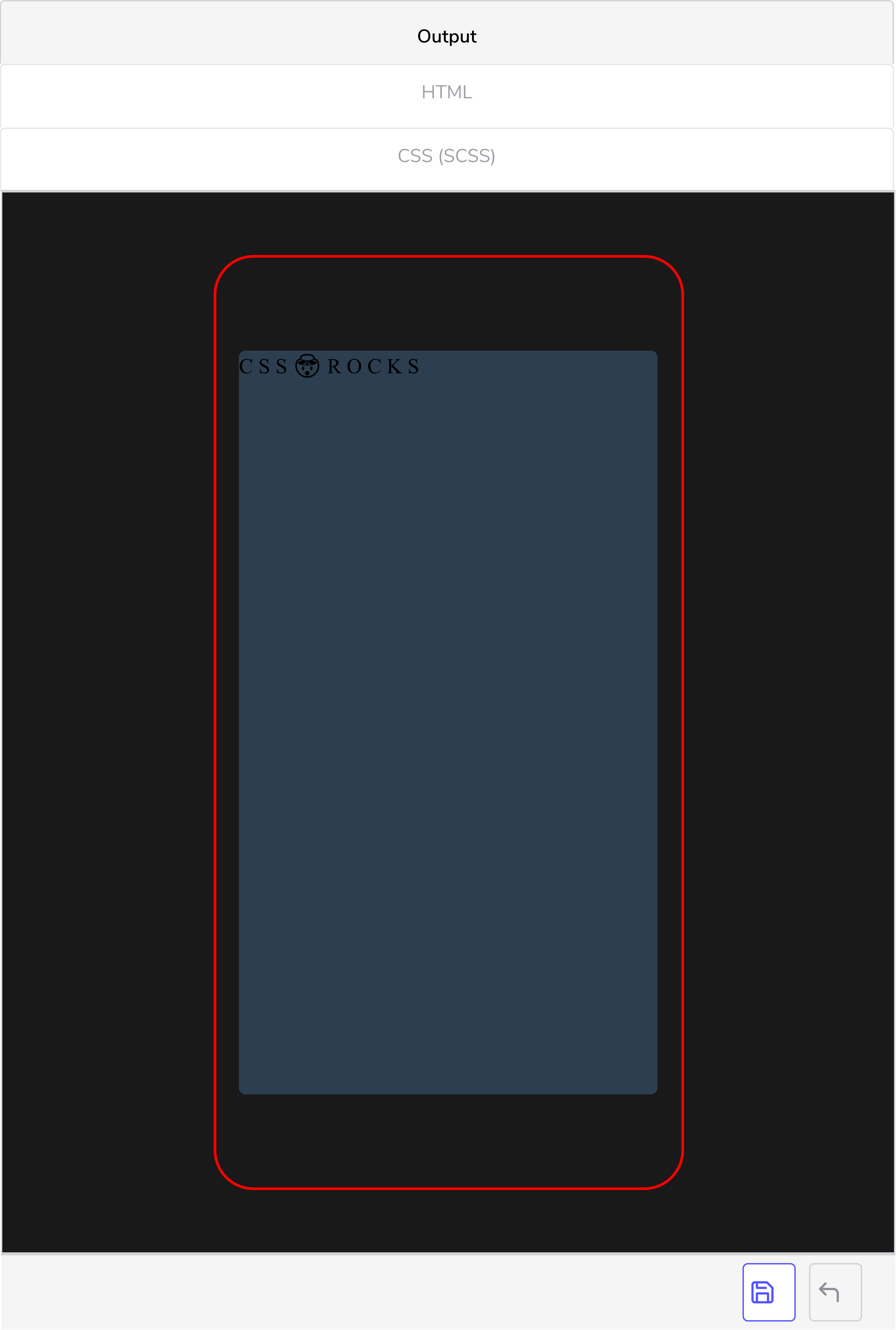
Output



If you wont say it, I will. That was super ugly. We’ll fix that.

Let’s align `.phone-inner` and have it sit perfectly centered within `.phone-inner`.

Flexbox to the rescue!



I smell progress!

It’s not magic, see the highlighted CSS by clicking the CSS tab above.

Center Center Center

Since we’re in the business of centering stuff, let’s center the “css rocks” container within `.phone-inner` too.

As always, Flexbox again to the rescue.

```
.phone-inner {
  background: var(--primary-bg);
  height: 80%;
  width: 90%;

  display: flex;
  justify-content: center;
  align-items: center;

  border-radius: 5px;
}
```

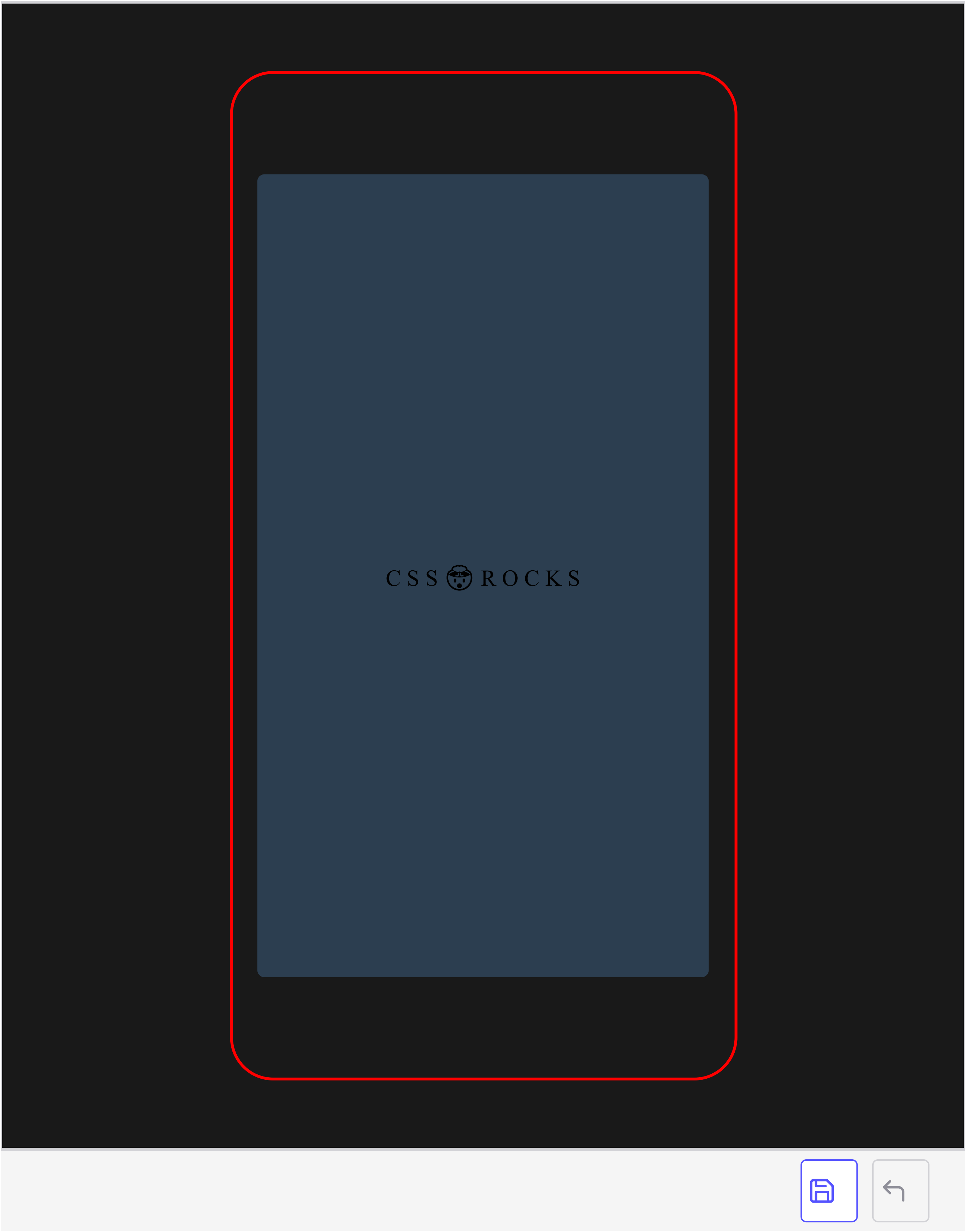
That’s it!

See the result below

Output

HTML

CSS (SCSS)



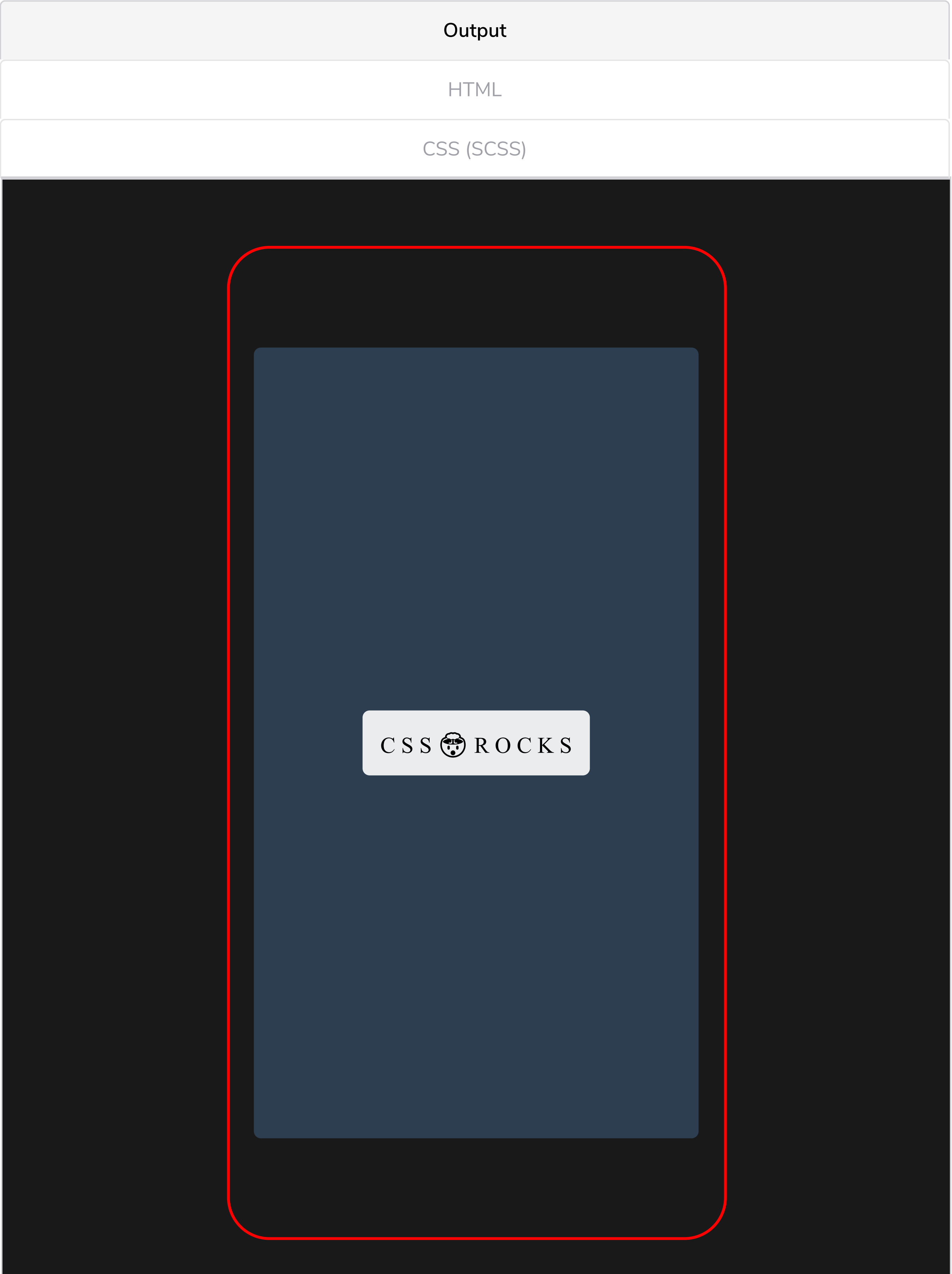
I am saying this too often, so this may be the last time I get to say this 👇

If you see a playground as in the one above, you can click any of the tabs to see the resulting code. In most cases, I will have highlighted the newer snippets of code. Okay?

Styling the Text within the Phone

The text, CSS rocks, is contained within an `article` tag.

Let’s make the text a little more obvious.



The added CSS is pretty easy to grasp. Here it is:

```
article {
  background: rgba(255,255,255,0.9);
  padding: 0.8rem;
  border-radius: 5px;
}
```

Wrapping Up

We aren’t done yet, but we’ve arrived at a reasonable place to end this lesson.

Look back at where we started. From nothing. We’ve made progress, huh?

In the next lesson, we’ll switch gears and have some more fun with positioning and generated contents.

For real, you’ll love it!