

Building Realistic CSS Effects with Box-shadow

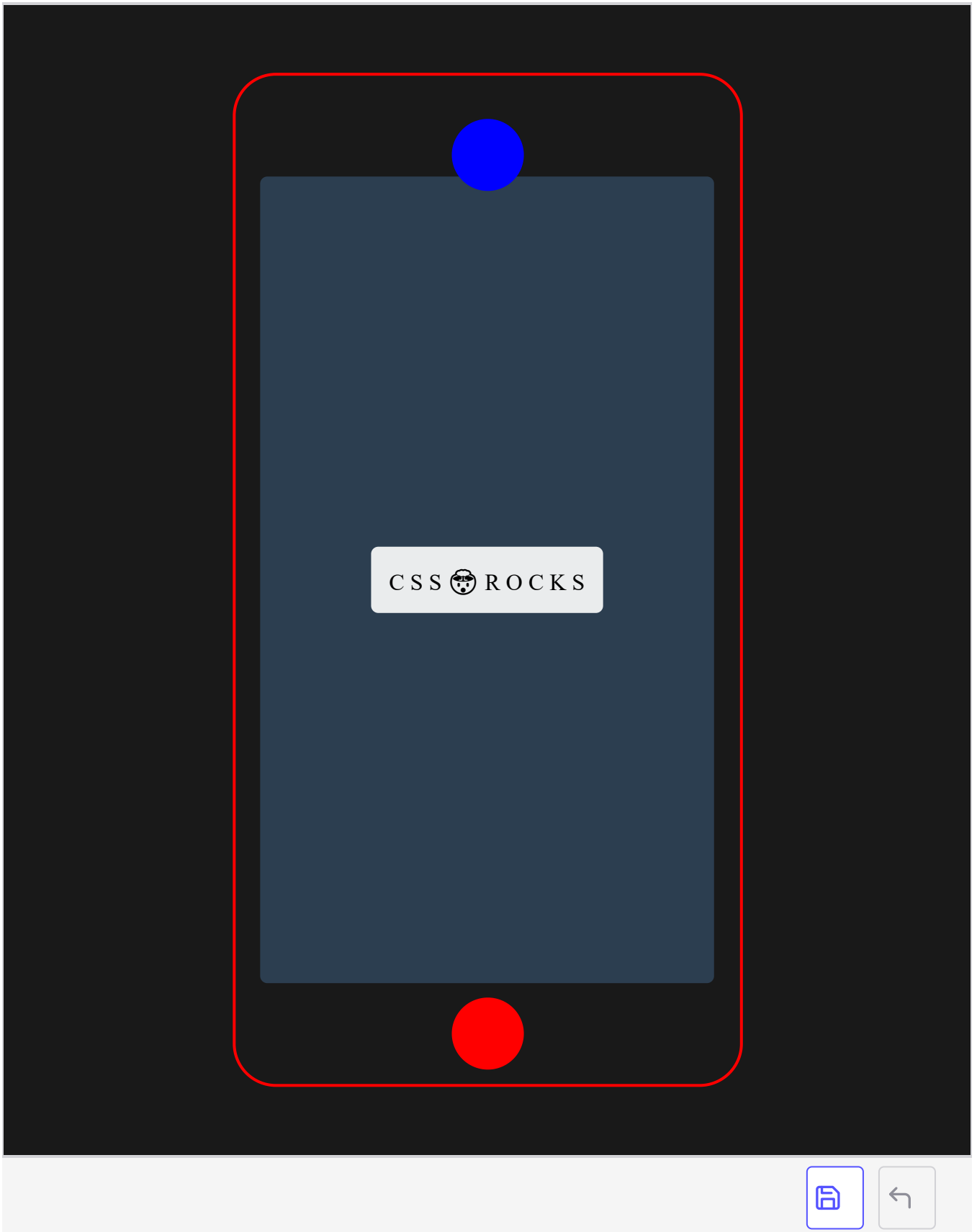
At the end of the last lesson, we were able to position some of the elements of the phone.

While that's good stuff, the phone is far from realistic. You can't show that off to friends yet!

Styling the Home Button and Speaker

Below is the current state of the project:

Output
HTML
CSS (SCSS)



Let's kick off with styling the home button represented by `.phone-inner:after`

The overall style for the home button is this:

```
/* Home button */  
.phone-inner:after{  
  content: '';
```



```
position: absolute;  
width: 50px;  
height: 50px;  
  
border-radius: 50%;  
bottom: -60px;  
box-shadow: 0px 1px 1px #ecf0f,  
            1px 0px 1px gold;  
}
```

I have removed the red background color. Now the background defaults to being fully transparent.

The added lines of code are highlighted above.

Can you make sense of it?

How the CSS box-shadow Property Works

The property values look weird at first, but on close look, it's really easier than it seems.

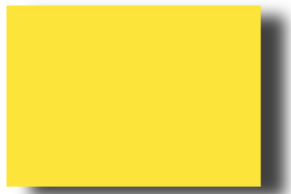
This is what a simple `box-shadow` declaration looks like:

```
box-shadow: 2px 4px 1px 2px red;
```

Daunting, huh?

Let's break it down.

Consider two elements represented by yellow boxes and styled with 2 different box-shadow values. The result may be seen below.



(a)



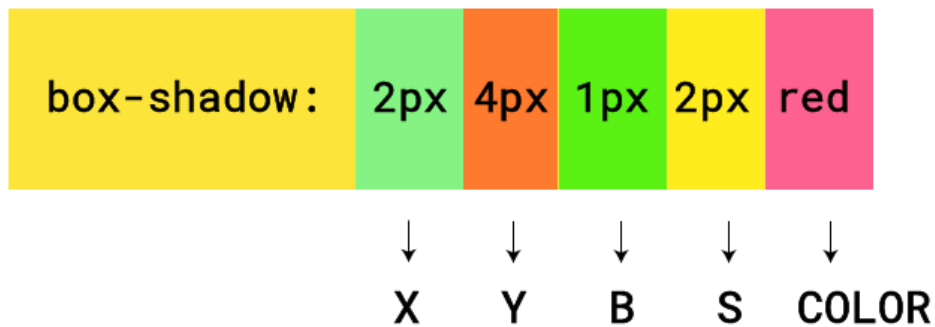
(b)

Without caring much about what code values yielded this, have a look at the elements above.

Do you see what I see?

1. The shadow acts like a clone of the element slightly nudged from the position of the element.
2. The shadow can have different colors.
3. The edges of the shadow look blurred.
4. The shadow in (b) seems to be nudged down a little more than (a)

This leads me to discussing the components that make up a `box-shadow` value.



The box shadow is largely defined by 5 properties.

1. X and Y refer to **horizontal** and **vertical** length values used to offset the shadow from the position of the element.
2. B refers to the **blur** length value of the shadow. This controls how much the edges of the shadow is blurred.
3. S refers to the **spread** value of the shadow. The length value specifies how much the shadow spreads on all sides.
4. **color** refers to the color of the shadow.

Pretty simple?

The way I remember this is, **X Y BullShit**. I never forget the order of the values with that!

Now, back to where we started.

Can you figure out what `box-shadow: 2px 4px 1px 2px red` does?

```
box-shadow: 2px 4px 1px 2px red;
```

X (horizontal offset): Position the shadow 2px from the element's position

Y (vertical offset): Position the shadow 4px from the element's position

B (blur length): Blur 1px of the shadow's edge

S (Spread length): Increase the shadow 2px on all sides!

C (color): Make the shadow red

Note that the direction of positive X and Y offsets is from left to right and top to bottom

I hope now you understand the code below:

```
/* Home button */
.phone-inner:after{
  content: '';
  position: absolute;
  width: 50px;
  height: 50px;
  border-radius: 50%;
  bottom: -60px;
  box-shadow: 0px 1px 1px #ecf0f,
              1px 0px 1px gold;
}
```



Oh, I missed something else.

Unlike the example before this, the `box-shadow` above has a comma followed by other values. This is a multiple declaration i.e setting more than one shadow on an element - yes that's possible!

If you still don't get how the `box-shadow` works, don't worry, toward the end of the lesson, I've added a playground where you can toy with some box-shadow values.

Styling the Speaker

This one is simple, and here's all the code:

```
/* speaker-rect */
.phone-inner:before{
  content: '';
  position: absolute;
  width: 50px;
  height: 7px;
  top: -40px;
  border-radius: 5px;
  background: var(--primary-bg);
}
```



I've made the element rectangular and given it a background color.

Put these together and we have the result below:

Output
HTML
CSS (SCSS)



Not looking bad. Yeah!

Styling the front camera

Like we did with the home button and speaker, we'll also make use of pseudo-

elements here. This time, the psedo-elements of the `.phone-body` element.

Since `.phone-body` will act as the reference element, it must be relatively positioned:

```
.phone-body {  
  position: relative;  
  /** other styles we've written **/  
}
```

It's just about the same process. Style the pseudo-elements and position them.

Here's code for that. I'm sure you can figure it out. It's just like we discussed earlier.

```
.phone-body:before,  
.phone-body:after {  
  content: '';  
  position: absolute;  
  width: 6px;  
  height: 6px;  
  border-radius: 50%;  
  background: var(--primary-bg);  
  top: 12px;  
}  
.phone-body:after {  
  top: 25px;  
  left: 125px;  
  width: 8px;  
  height: 8px;  
}
```



You'll notice that I grouped the selectors like so:

```
.phone-body:before,  
.phone-body:after {  
  /** write same styles here */  
}
```

I then secified the unique styles on `.phone-body:after`:

```
.phone-body:after {  
  /** unique styles **/  
}
```

This will override whatever styles were the same in the grouped selection.

The results

Add these to to the project and you have the result below:

Output
HTML
CSS (SCSS)



Finishing up the Project

This time, all that is left is removing the ugly red border and making it fancy.

Here's the secret sauce:

```
.phone-body {  
  
  box-shadow: 1px 2px 1px gold inset,  
             2px 10px 3px rgba(149,165,166 ,0.1);  
  /** other styles we've written **/  
}
```

Hey, `box-shadow` again!

By default, shadows are created outside the element.
However, the optional `inset` keyword will create the shadow
inside the element.

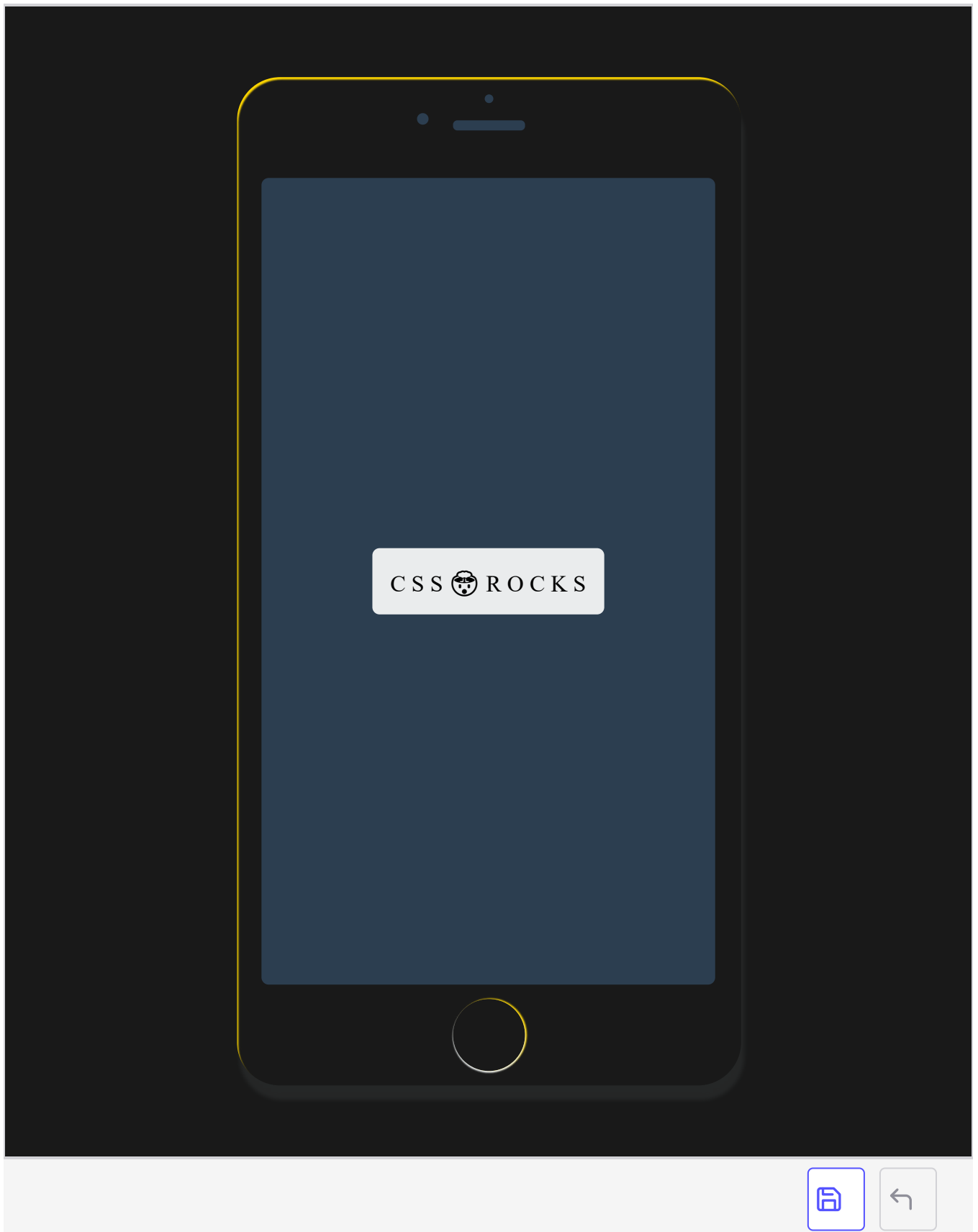
From outer shadow to inner shadow

the comma
allows the
addition
of multiple
shadows

```
box-shadow: 1px 2px 1px gold inset,  
           2px 10px 3px rgba(149,165,166 ,0.1);
```

Here's the result of that:

Output
HTML
CSS (SCSS)



Now, I'm super excited! 😁🤖🤖

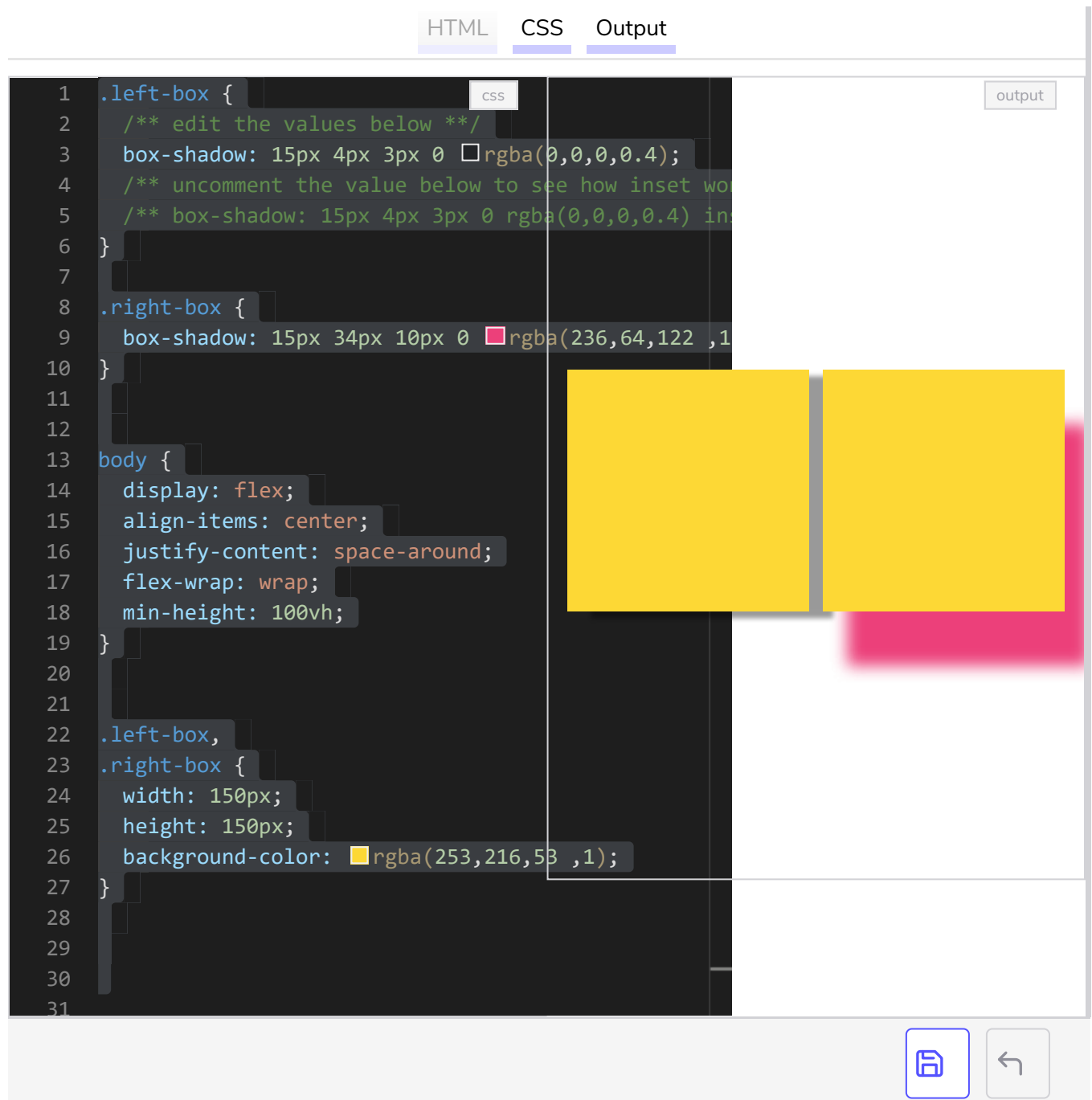
Here is an explanation of what's going on in the output above.

There are two shadows applied to `.phone-body`. The thin golden inner shadow on the left, and the darker large shadow that forms the bottom of the phone.

The fastest way to see how these work is by tweaking the values in the playground above. Don't hesitate to do so!

I still don't get this **box-shadow** thing

Here is a playground to specifically toy with how **box-shadow** works. Break the properties apart and tweak them over and over again. You just need to practice this for a bit and you'll get the hang of it.



The playground interface has three tabs: HTML, CSS, and Output. The CSS tab is selected. The code in the CSS tab is as follows:

```
1 .left-box {  
2   /** edit the values below **/  
3   box-shadow: 15px 4px 3px 0 □ rgba(0,0,0,0.4);  
4   /** uncomment the value below to see how inset works  
5   /** box-shadow: 15px 4px 3px 0 rgba(0,0,0,0.4) inset;  
6 }  
7  
8 .right-box {  
9   box-shadow: 15px 34px 10px 0 ■ rgba(236,64,122,1);  
10 }  
11  
12  
13 body {  
14   display: flex;  
15   align-items: center;  
16   justify-content: space-around;  
17   flex-wrap: wrap;  
18   min-height: 100vh;  
19 }  
20  
21  
22 .left-box,  
23 .right-box {  
24   width: 150px;  
25   height: 150px;  
26   background-color: ■ rgba(253,216,53,1);  
27 }  
28  
29  
30  
31
```

The output shows two yellow boxes. The left box has a subtle grey shadow, and the right box has a more pronounced pink shadow.

Is there a handy tool for creating box shadows?

Yes, there is. I seldom use it as I enjoy the pain of tweaking the box-shadow property values till I hit a home run.

Here's the tool. It is called [CSS MATIC](#).

You may find it helpful.

Conclusion

I hope you've had as much fun as I did writing this lesson!