The CSS Positioning Lesson You Never Had

After this lesson, you'll never have problems with positioning in CSS again. Period!

Introduction

Meet Janice and Roughnut.

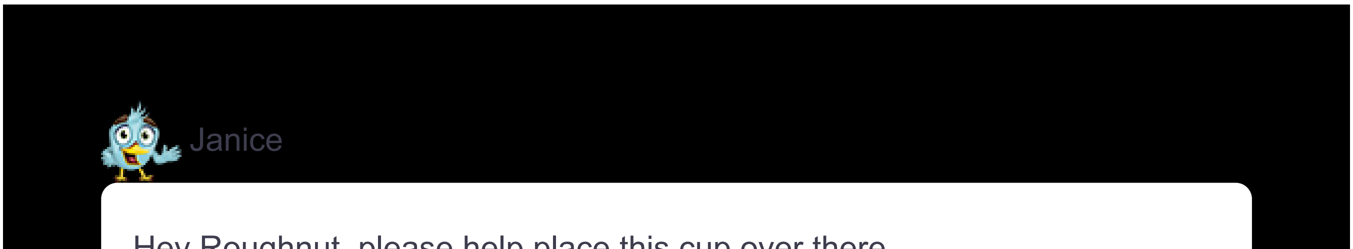
Janice

Hey there, pleased to meet you!

Roughnut

Yo! It's Roughnut here :)

Both are great buddies, and after grabbing coffee at a nearby coffee shop, the following conversation ensued.



Janice

Hey Roughnut, please help place this cup over there

Hey Roughnut, please help place this cup over there

Roughnut

Alright, done.

Janice

Jeez! You didn't bother asking for the precise location to place the cup.

Roughnut

Don't be so nerdy. Alright, where do I place it?

Janice

About a quarter meter from the table over there

Roughnut

Seriously? Go do that yourself

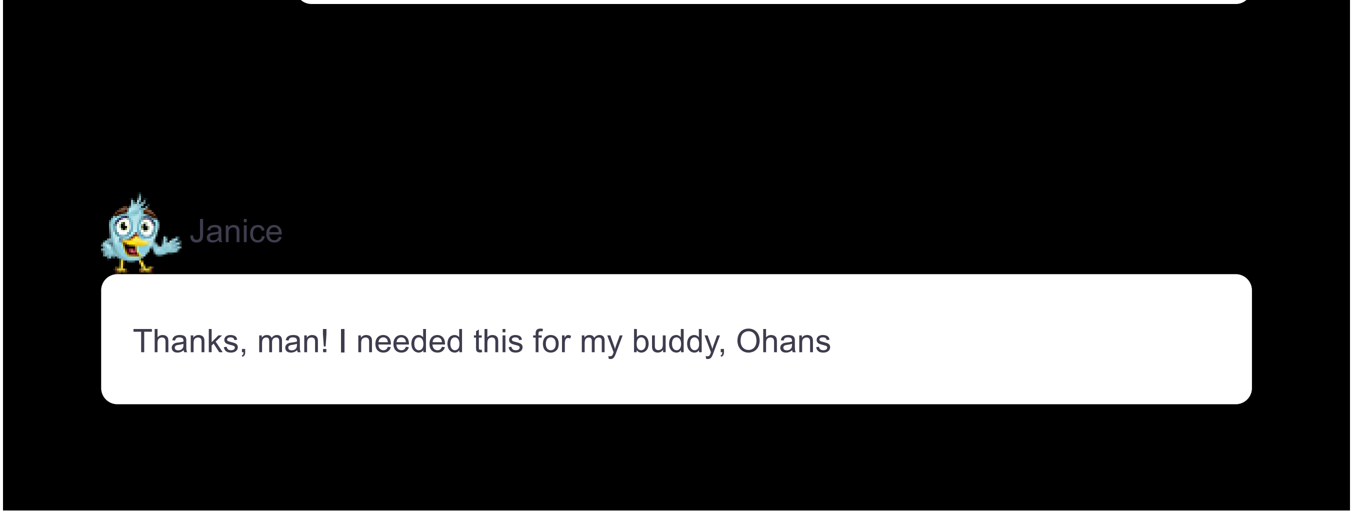
Janice

Please Roughnut.

Roughnut

Alright, I'll do that

(Grudgingly places the cup about a quater meter from the table. Thanks to the Math classes)



Janice

Thanks, man! I needed this for my buddy, Ohans

Making Sense of the Story

I wasn’t intending to waste your time by reading a pointless story. Now, let me show you where this all comes to play.

In the real world, objects are typically placed with respect to some other object in a scene.

Take for example the cup in the story above.

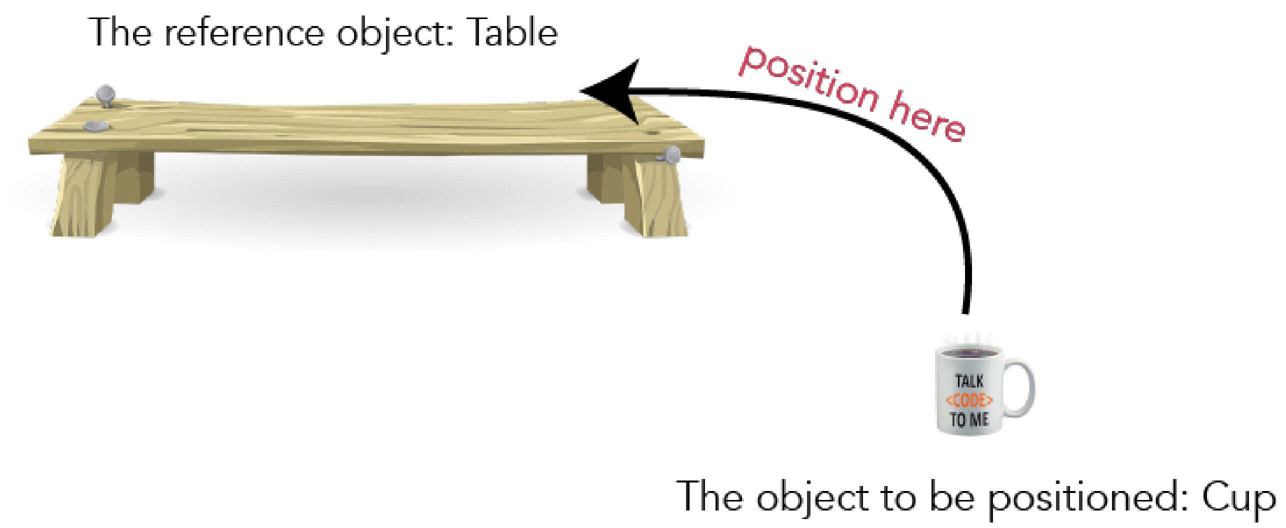
Roughnut, like everyone else, went ahead and just placed the cup somewhere close to the table.

If only we were all a little more concious of this, we’d all notice that nothing is placed in empty space.

You’re always placing objects with respect to some other object in space.

Are you putting the cup on the table or the children to bed?

Did you see that?



Reference object + object to be positioned

While positioning anything, there are always at least two objects involved.

1. The object to be placed.
2. The reference object.

Please refer to the graphic above.

Having established that, there's one more thing to note.

After placing an object with respect to a reference object, you can accurately measure where they were placed.

For instance, when you place a cup on a table, you can measure exactly how many inches from the edges of the table the cup sits.

TALK
<CODE>
TO ME
y
x

Don't have painful math memories. This is just an example

While nobody goes around monitoring the geometry of objects around them, this is important to the concept of positioning.

Let's see how this applies to CSS.

Relative Positioning

```
.rel {
   position: relative;
}
```

What does the code above mean?

Let's get back to the story.

When I say, 'place the cup on the table', I have made mention of two different objects.

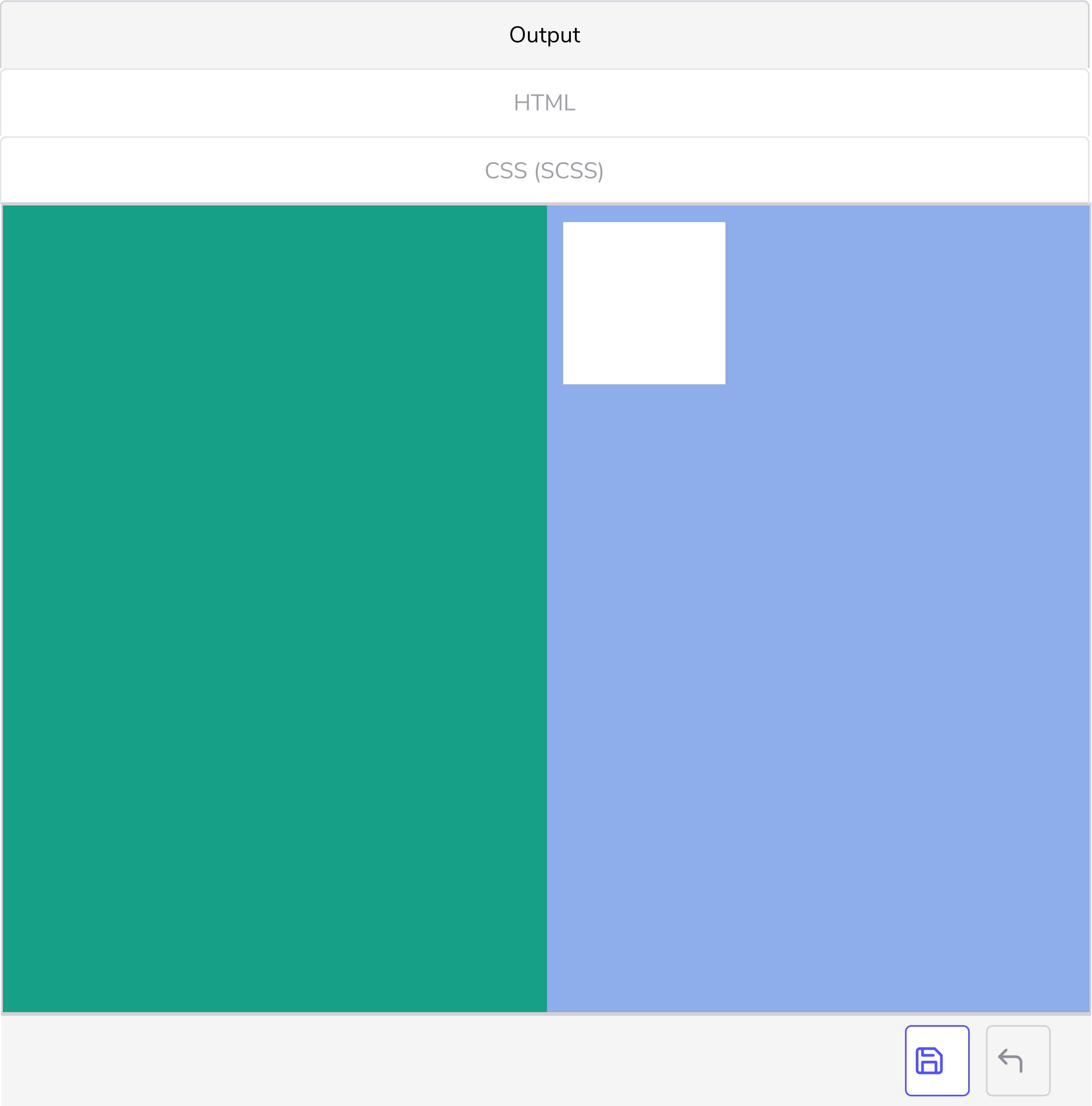
1. The object to be placed, the cup.
2. The reference object, the table

2. The reference object, the table.

When you set the position property of an element to a value of `relative`, you’re setting it up as a reference object to position some other element.

Let’s get practical.

Consider the code output below:



The code above represents two `divs` placed side by side.

1. The green div.
2. The blue div.

In the blue `div`, there’s a child element represented by the small white box.

Consider yourself faced with the challenge of positioning the white box within the blue box.

Here’s my solution.

First off, note the two objects at play here:

1. The object to be positioned, the white box.
2. The reference object, the blue container.

In CSS, you specify the reference object by setting the `position` property value to `relative`

I’ll go ahead and do that.

Output

HTML

CSS (SCSS)

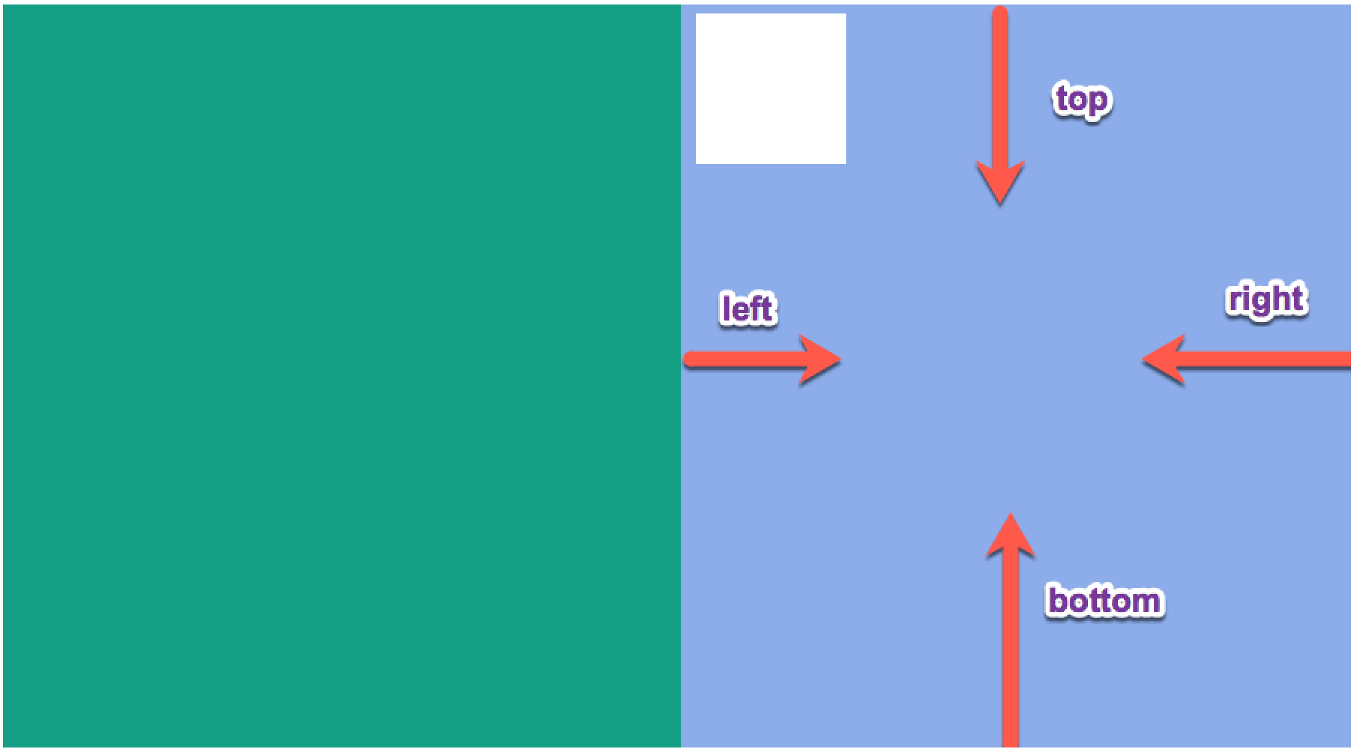
```
body {
  min-height: 100vh;
  display: flex;
  background: red;
  margin: 0;
}

.green-box,
.blue-box {
  height: 100vh;
  flex: 1 1 0;
}

.green-box {
  background-color: #16a085;
}
.blue-box {
  background-color: #8cacea;
  position: relative;
}

.child {
  background-color: #fff;
  width: 100px;
  height: 100px;
  margin: 10px;
}
```

Click the output tab above. Nothing’s changed 🤪

Well, that shouldn’t be surprising. All we’ve done is make the blue `div` the **reference** object for positioning the white box. Nothing magical yet.

Here comes the magical bit 😂

Setting up a reference object in CSS comes with some goody. Firstly, a positioning context is established for all child elements.



positioning context set around the blue box

The Positioning Context

When you style an element with `position: relative`, it creates a positioning context for every child element within the element.

What this means is, you can go ahead and accurately position the child element with respect to any sides of the `reference` object.

Don’t forget the table and cup analogy.

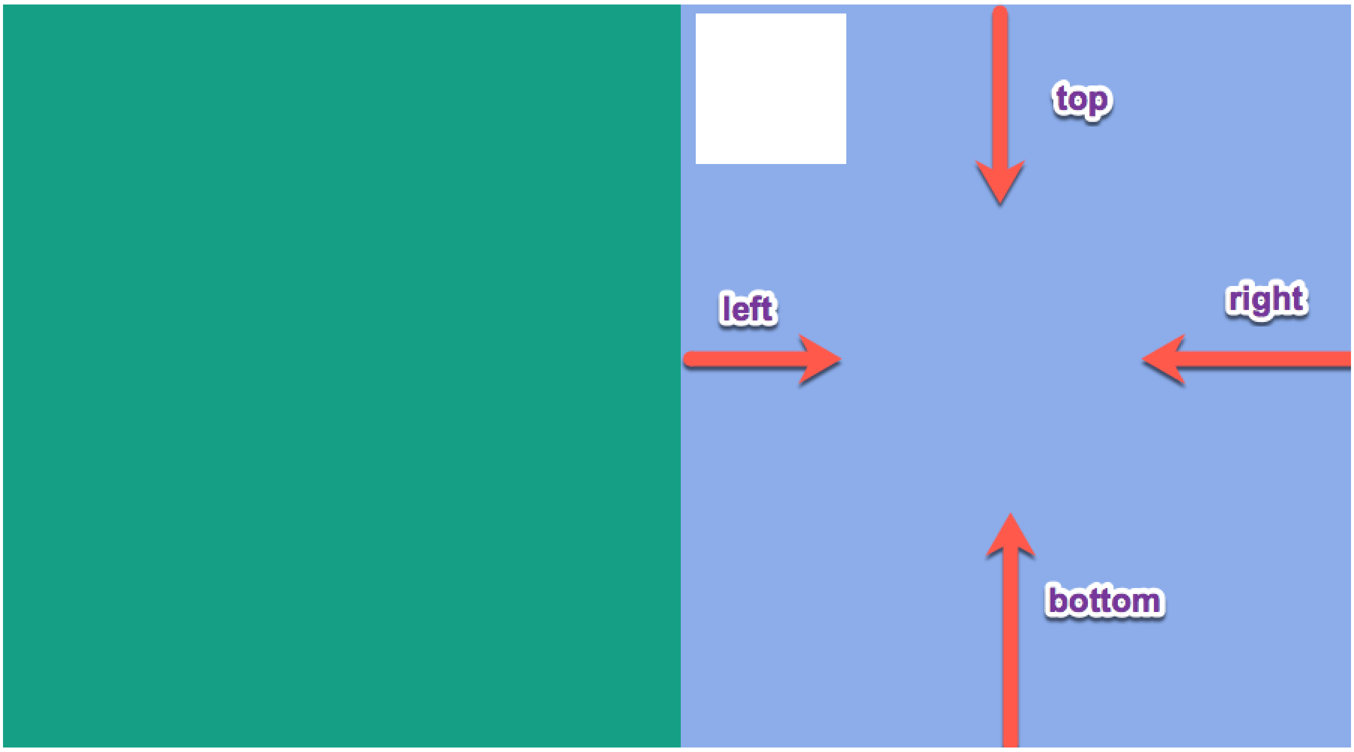
If the table is made the reference object, you can place the cup `1cm` from the top of the table, or `3cm` from the `bottom` of the table.

top of the table, or `5cm` from the `bottom` of the table.

The same is applicable here.
The object to be positioned may be placed within this positioning context.

In CSS a valid example would be to place the child element `20px` from the `top` of the `reference` element, or `30px` from the `bottom` of the reference object.

Positive values will place the child element in the directions shown below:



Note the direction in which positive length values will place the child element.

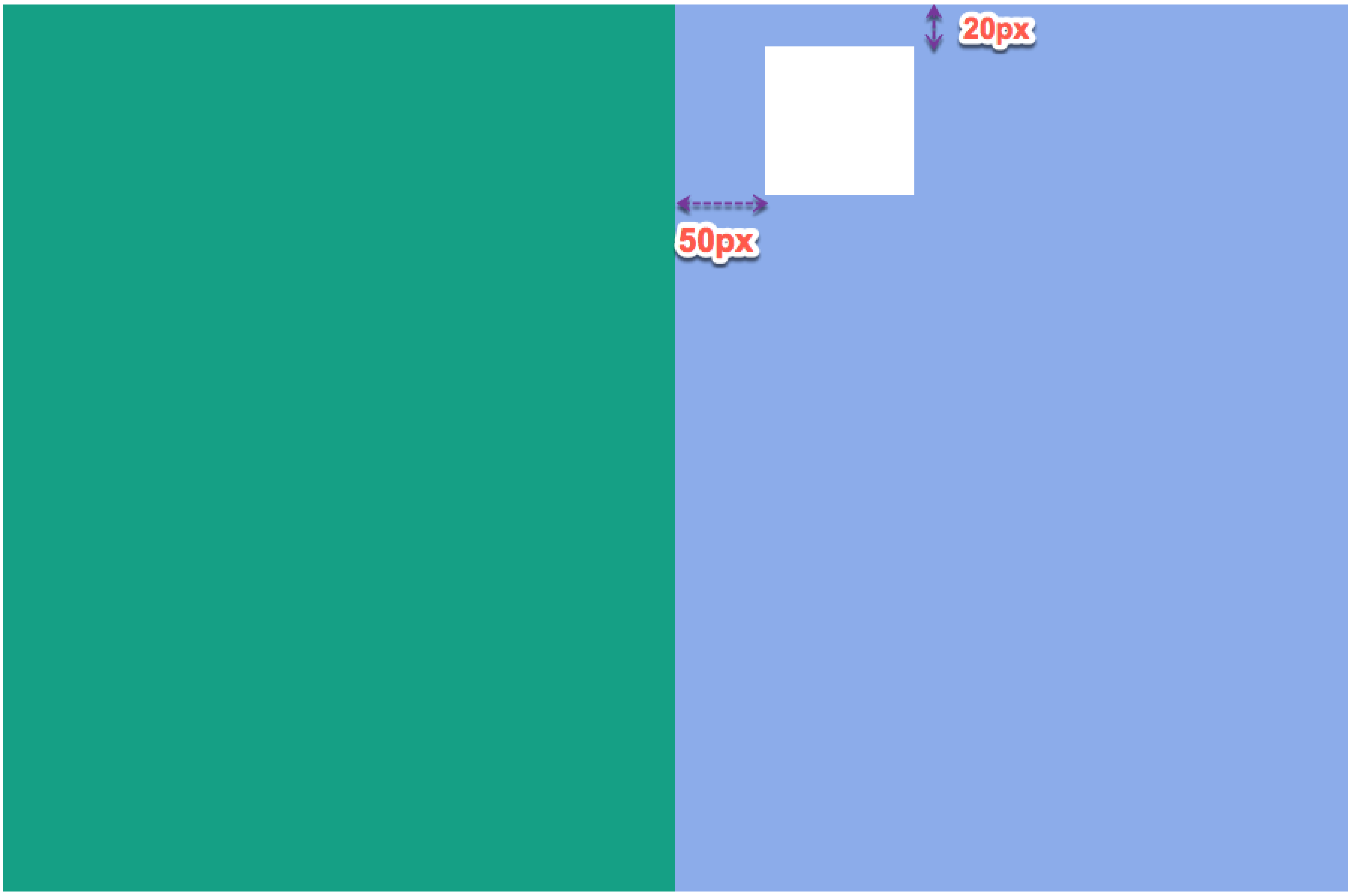
In a nutshell, you can place the child element at any distance from the `top`, `bottom`, `left` or `right` of the reference element.

Let's apply this knowledge to place the white box `20px` from the top of the blue box, and `50px` from the left of the blue box.

You do so by adding the style below:

```
top: 20px;
left: 50px
```

Using the directions set up in the positioning context (see graphic above,) this should move the white box as shown below:



The expected result

Let’s go ahead and add that to the code playground.

Output

HTML

CSS (SCSS)

```
body {
  min-height: 100vh;
  display: flex;
  background: red;
  margin: 0;
}

.green-box,
.blue-box {
  height: 100vh;
  flex: 1 1 0;
}

.green-box {
  background-color: #16a085;
}
.blue-box {
  background-color: #8cacea;
  position: relative;
}

.child {

  background-color: #fff;
  top: 20px;
  left: 50px;
  width: 100px;
  height: 100px;
  margin: 10px;
}
```

Go ahead and click the outout tab. Do we have the expected result?

I just did. No! Nothing’s changed 😫

Why?

That got me sad, but I have a fix for that.

Absolute Positioning

At this point, we have the reference object correctly set.

We have the positioning context in place and we have specified the exact `top` and `left` values we want. But nothing seems to work.

There’s one more bit of setup to do.

In order for the white box to take part in this positioning context, we need to specifcally ask it to partake in the positioning context.

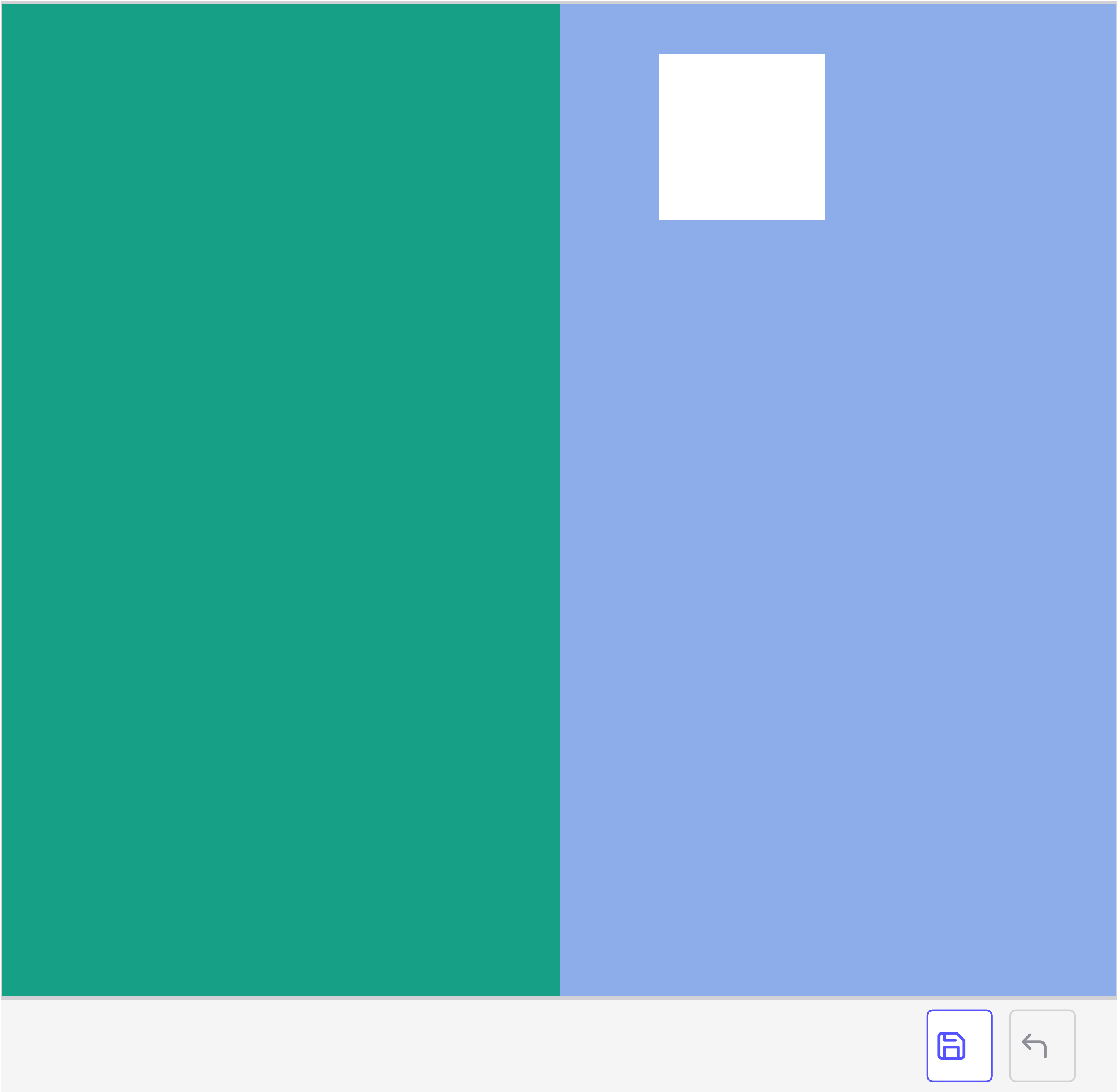
The way to do that is by setting `position: absolute` on the white box.

Have a look at the output below:

Output

HTML

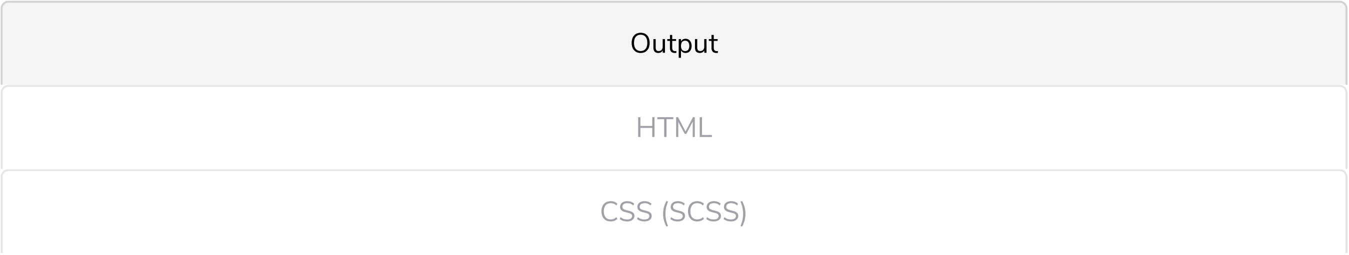
CSS (SCSS)

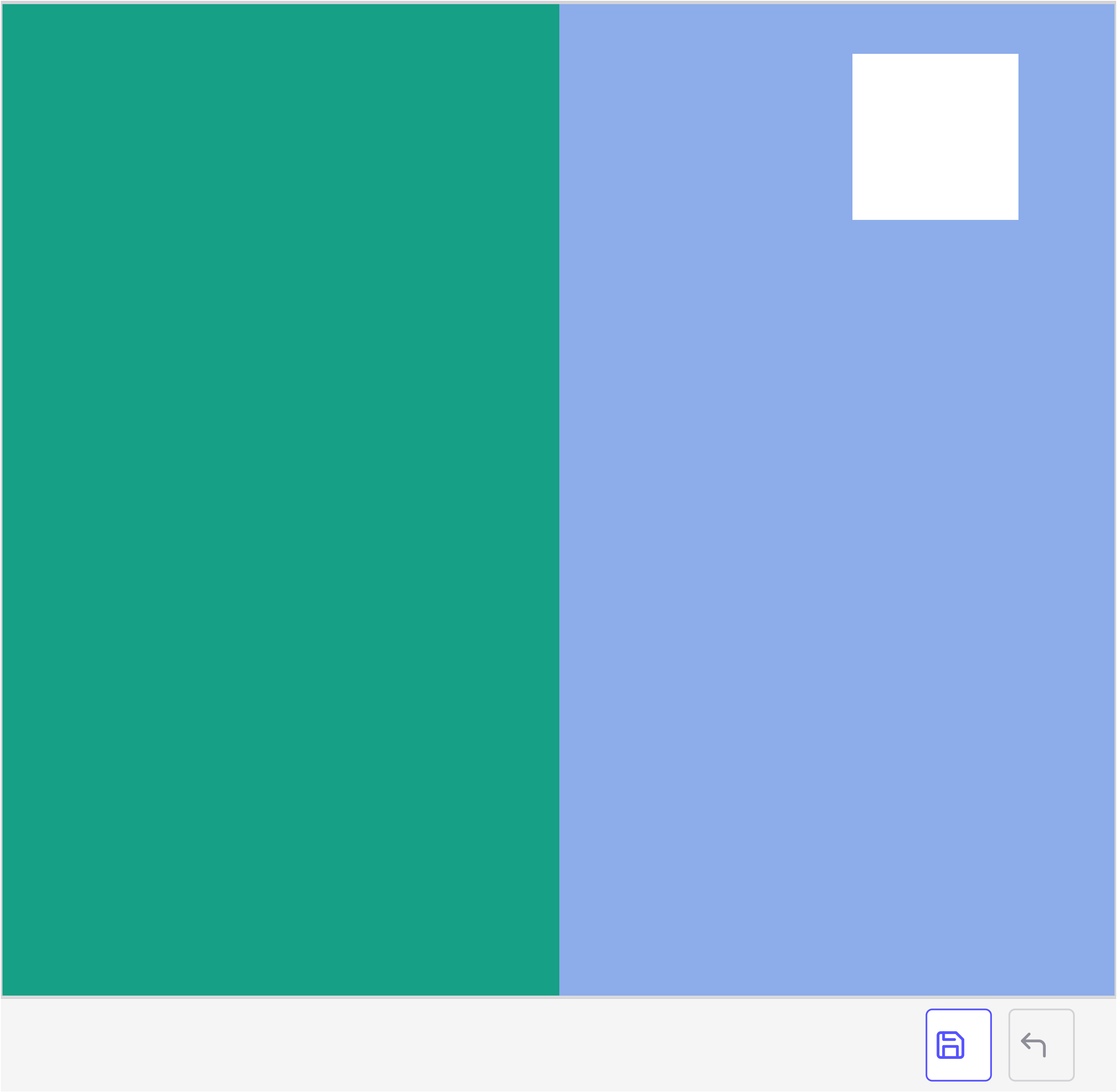


The white box moved!

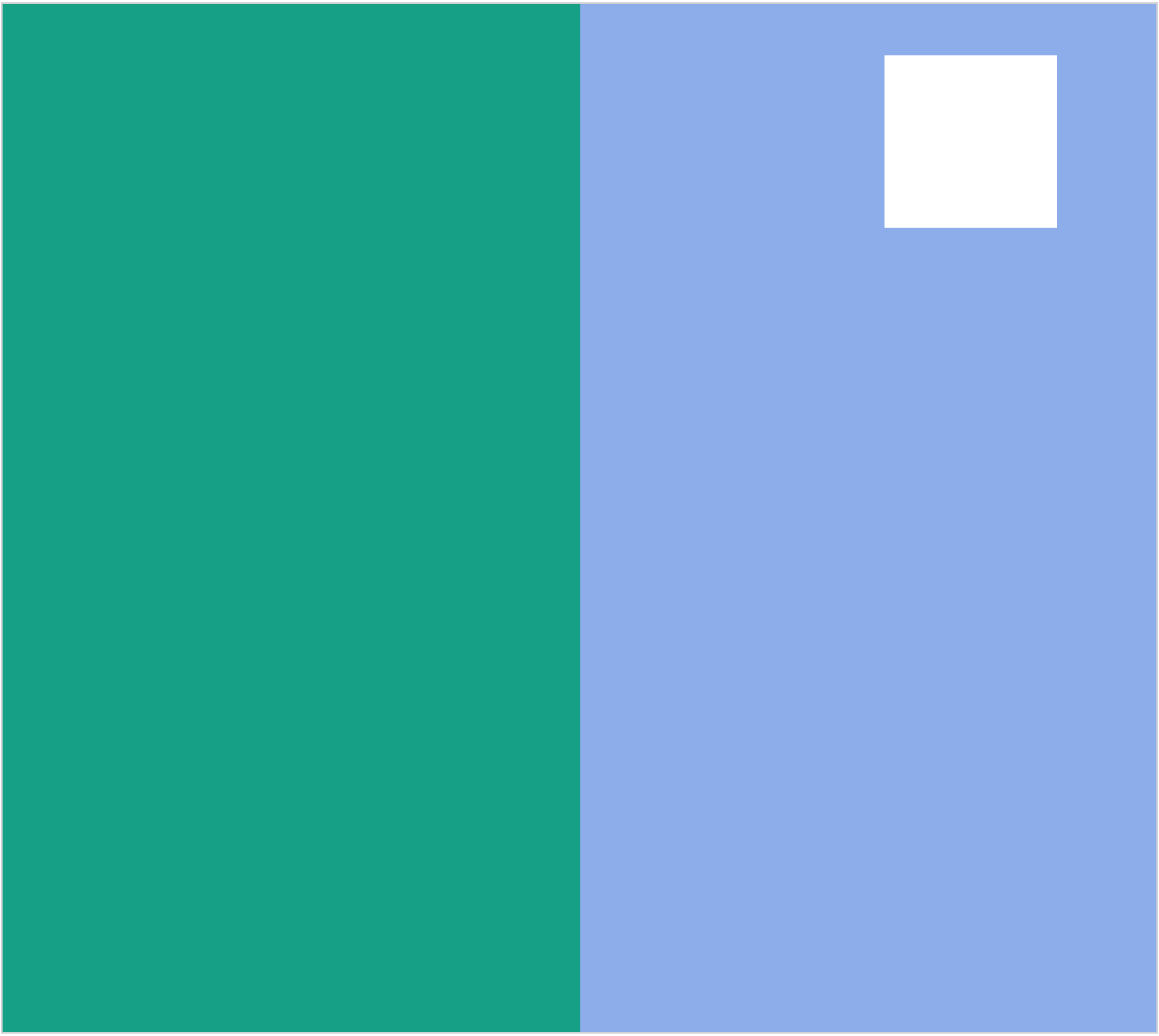
Click the `CSS` tab to see what’s changed.

I know the change is subtle, so I have gone ahead to set the `left` property to `50%`. That is to say, 'place the white box 50% from the left of its reference element.





The output below shows the change in position of the white box.



Now the change is a lot more obvious. I'm sure you didn't miss that. The white box moved!

Before I go ahead, let me reiterate some of the important ideas we've come accross so far.

1. To create a reference element for positioning, set the `position` value to `relative`.

2. Setting the `position` value to `relative` initiates a positioning context where every child element may be positioned.

3. Elements partaking in a positioning context may be positioned using any of the keywords: `top`, `bottom`, `left`, and `right`

4. To expliclty ask a child element to participate in a positioning context, use

`position: absolute`

Now, those are nuggets you don't want to forget!

Some More Positioning!

With respect to positioning, there are a few more points I haven't discussed.

Let's see an example.

In the playground above, the one before the animated output, go ahead and remove the `position: relative` on `line 19` and change the `left` value of `.child` to `30px`

Go ahead.

What do you notice?

The Default Reference Element

When you removed `position: relative`, you removed the reference element for which `.child` was placed. You left the poor `.child` lonely.

However, CSS and browsers are smart.

When you don't explicitly set a positioning context, every element with `position: absolute` set will be positioned relative to the viewport!

In this example , the white box will move `20px` and `30px` from the top and left side of the viewport.

You see that?

So, if you don't explicilty set the reference element, the entire viewport becomes your reference element.

Some More Positioning Nuggets

> 1. When you set `position: relative` on an element, it does NOT set up a positioning context for EVERY element on the page. It ONLY sets up a positioning context for its child elements.

For example, we could position the white box relative to the blue box because the white box was a direct child of the green box.

For a refresher, see the markup used below:

```
<html>
 <head>
   <title> Relative Positioning </title>
 </head>
 <body>
   <div class="green-box"></div>
   <div class="blue-box">
     <div class="child"></div>
   <div>
 </body>
</html>
```

You see where the `.child` element is?

This is why the positioning context on the parent element, `.blue-box` worked for it.

> 2. A Relatively positioned element (position: relative) also creates a positioning context for itself.

The positioning context created by a relatively positioned element doesn’t just affect its child elements. They also exist for the element itself.

Consequently, while the white box could be moved relative to the green box, the blue box itself could also be moved relative to the created positioning context.

Let me explain better.

Remember the Table and Cup story?

While the Cup could be positioned on the Table, the Table could also be moved based on its previous position.

That way, the Table exists as its own reference object.

This is the case with relatively positioned elements `position: relative`. They can be positioned based on their previous location within the document flow.

Consider the example below.

I have nudged the blue box by some values. Click to see the output below.

Output

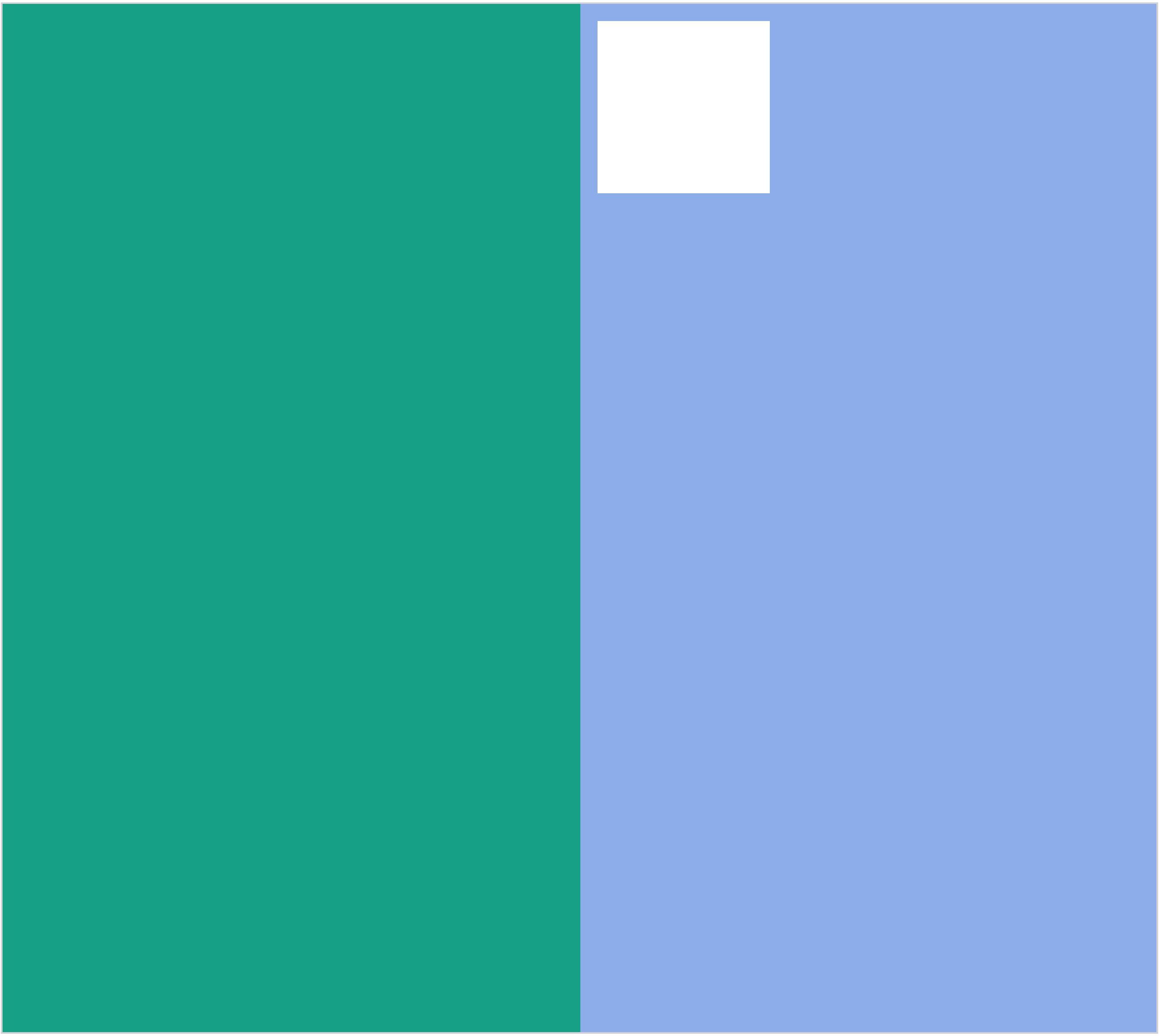
HTML

CSS (SCSS)

```
body {
  min-height: 100vh;
  display: flex;
  margin: 0;
  background-color: black;
}

.green-box,
.blue-box {
  height: 100vh;
  flex: 1 1 0;
}

.green-box {
  background-color: #16a085;
}
.blue-box {
  background-color: #8cacea;
  position: relative;
  top: 50px;
  left: 70px;
}

.child {
  background-color: #fff;
  width: 100px;
  height: 100px;
  margin: 10px;
}
```

Now the blue box has been positioned based on its own positioning context. Notice that the blue box moves along with its child element.

Below is an animated output to help you visualize the result better.



Let’s keep moving!

> 3. What’s the difference between relatively positioned elements (position: relative) and absolutely positioned elements (position: absolute)?

It appears that both positioned elements (relative and absolute) can be moved from their initial position with the keywords, `top`, `bottom`, `left` and `right`.

So is there any difference between them?

The major difference is that an absolutely positioned element can be positioned relative to another element on the page (position: relative), while a relatively positioned element positioned element will only be positioned relative to itself.

This has been a very lengthy lesson. There's just a little more to cover. I apologize for over-explaning some of these things. I just want to be sure you get it.

So, what is Static positioning

```
.static {
  position: static;
}
```

Besides, `position: relative` and `position: absolute`, there's also `position: static`

Back to the story.

Before Janice and Roughnut had the conversation about placing the cup on the table, both objects (Cup and Table) had no part in any positioning context. They were just objects that existed in the scene.

The same may be said for every `HTML` elements. If you don't set `position: relative` and `position: absolute` on an element, it does NOT participate in any positioning context. It is therefore said to be statically positioned.

Consequently, all elements are by default positioned "statically" on a page. They are just objects within the scene (like the Cup and Table) until you get them to be a part of a positioning context.

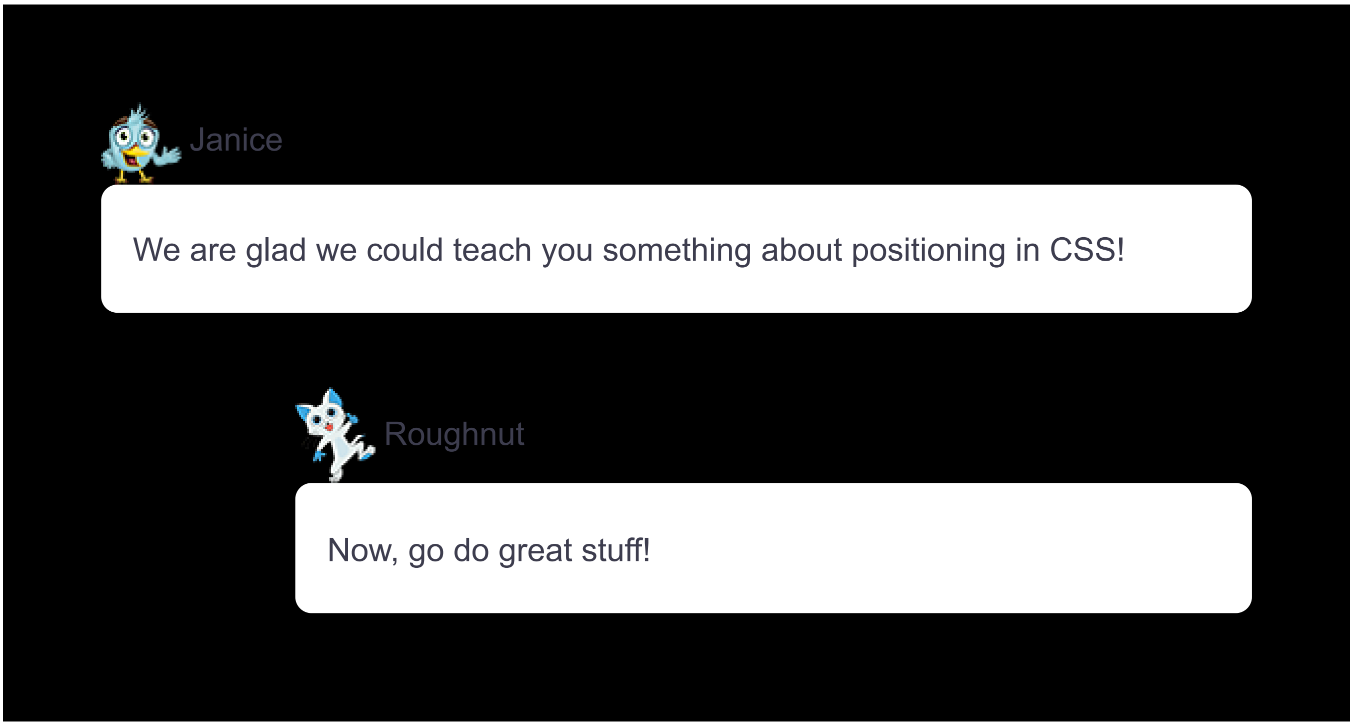
Are there Other Positioning Values?

Technically, an element is said to be positioned if its position value has been changed from being static, to one of four available values: `relative`, `absolute`, `fixed`, `sticky`.

In coming lessons, I'll discuss what `fixed` and `sticky` positioning mean. The other position values introduced in CSS3 are `center` and `page`. They aren't commonly used, so, I won't be discussing them.

Summary

1. Use `position: relative` when you need a positioning context for the child elements of a container.
2. Use `position: absolute` to position a child element with respect to its parent element.
3. A relatively positioned element can be nudged within its positioning context too.
4. If there are no parent elements with a positioning context, child elements will position themselves relative to the viewport.



We are glad we could teach you something about positioning in CSS!

Now, go do great stuff!

Don’t worry if you still don’t get some of this. We will get back to the iPhone project and apply some of the new knowledge gained here.

See you in the next lesson!

Conclusion

If you just learned CSS positioning well, thanks to Janice and Roughnut, be kind to give them a shout-out on Twitter. Huh?

Shout-out