What is the calc() function, and how does it work?

With the calc() function, you can perform calculations directly within your stylesheets to determine property values dynamically. This means that you can create flexible and responsive user interfaces by calculating dimensions based on the viewport size or other elements.

calc() is a CSS function. You'll learn more about functions when you start learning about JavaScript, but in this lecture, you're going to learn the basics that you need to know to understand how calc() works.

A function is a block of code that performs a specific task. Some functions are already defined in CSS, so you can use them directly and pass any necessary values to them to customize how their tasks will be performed.

In the world of programming, when we run the task performed by a function, we say that we "call" the function. The values that we pass into the function are known as arguments.

Like you can see in the code below, to call a function, you write its name followed by the arguments within parentheses, separated by commas. There shouldn't be a space between the name of the function and the opening parenthesis:

function(argument1, argument2, argument3)

A function may only need one value to know what to do. In that case, it will only take one argument. That's what happens with the calc() function. It takes one argument because it needs to know what to calculate.

For this, you pass something called an expression as an argument. An expression is a combination of values and operators that produces a result.

This is how you can call the calc() function. You write the name calc, followed by parentheses, and within the parentheses, you write the expression:

calc(expression)

The expression is evaluated to calculate the final result. "Evaluated" just means that the values and operators are converted into a single value behind the scenes. The result is assigned to the CSS property where the calculation is being made.

You can perform calculations on values that represent length, angle, time, percentages, numbers, and colors. You can also combine different units like pixels, percentages, and ems.

With numbers, all the values in the expression, also called the operands, must have their corresponding units, like px, em, and percentage (%). Depending on the operator, different operands may have different units.

You can use the addition (+), subtraction (-), multiplication (\*), and division (/) operators in the expression.

If there are multiple operands and operators, calc() will follow the standard operator precedence rule. You can also add parentheses to establish the order of the operations if needed.

In the example below, you can see a div with the text Hello, World!:

<div>Hello, World!</div>

Using the CSS type selector for selecting the div, you can style it with white text and a dark blue background:

div {

color: white;

background-color: #1b1b32;

width: calc(50% - 20px);

}

What's new here is that the width is calculated dynamically. Notice how we are calling the calc() function and passing an expression as an argument. The expression has two operands with different units and one operator, the subtraction operator.

Percentage is a relative unit. The value (50%) will be determined by the width of the parent container. Then, 20px is subtracted from the value. The result of this expression will determine the width of the div.

The width of the div is approximately half the total width of its container, and if you resize the parent container, the width will be recalculated automatically.

That's the key advantage of using calc(). You can determine the value of a CSS property dynamically based on different aspects of the application or viewport.

The expression can also contain CSS functions and variables if you need to use them in your calculations. You'll learn more about CSS variables in the next lectures.

Great. Now that you know about the basics of the calc() function, let's cover some of its best practices.

First, you must surround the addition (+) and subtraction (-) operators with whitespace.

For example, the expression below would not be a valid expression because the subtraction operator is immediately before the second operand.

calc(100% -30px)

The subtraction (-) operator must be surrounded by whitespace, like this. Adding the whitespace will create a valid expression.

calc(100% - 30px)

This is not necessary for the multiplication and division operators but it's highly recommended.

You can also nest calc() function calls if you need to perform calculations and use those results in other calculations.

Also, if you use the value zero to represent length in the expression that you pass into the calc() function, you must include the units. For example, this expression would not be valid:

calc(100% - 0)

You would need to add the units, like px.

calc(100% - 0px)

You should also know that currently, if you use the multiplication or division operators, one of the operands has to be unitless. For the division operator, specifically the right operand has to be unitless. This would not be a valid expression because both operands have units (pixels). One of the operands, either 5 or 50, must be unitless:

calc(5px \* 50px)

You would need to omit the units in one of them. Both of these alternatives would be valid:

calc(5 \* 50px)

calc(5px \* 50)

And this is an example with the division operator. This would not be a valid expression since they both have units:

calc(50% / 5%)

You should remove the unit from the right operand when you have the division operator:

calc(50% / 5)

The calc() function can be very helpful for you as a web developer. With this function, you can set property values dynamically to create flexible and responsive designs.