



SEMINAR

CALC() CSS FUNCTION

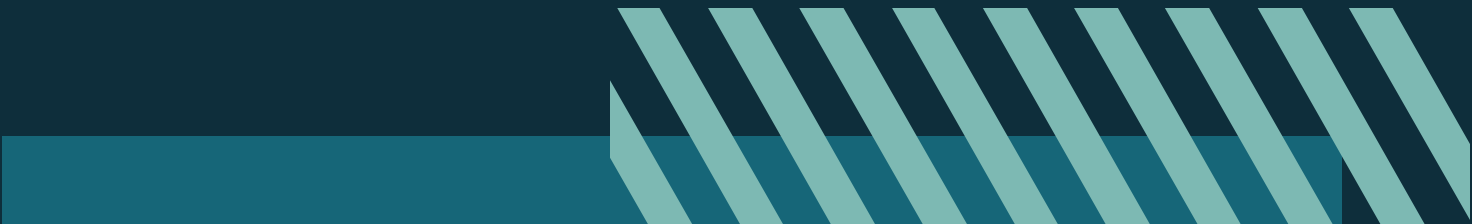
Presented by: Fasika Kelile

Date: Feb 19 2025



What is the `calc()` CSS Function?

`calc()` is a CSS function that lets you perform basic arithmetic operations (addition (+), subtraction(-), multiplication(*), division(/) to determine CSS property values in real-time.





Its syntax

```
element {  
  property: calc(expression);  
}  
width: calc(100vw - 10%);
```

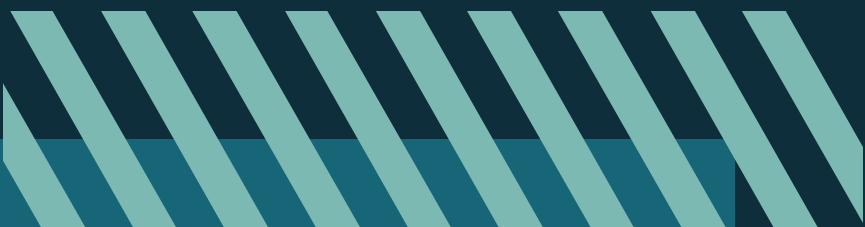
Here -> expression parameter represents a **calculation**. Giving you a single value for CSS property.





How do I use it?

Can use calc() when using css properties that accept length within the values

- Few properties in css that take length
 - (width, height, padding, margin, border, box sizing and fonts).
 - Length Units of CSS
 - (px, %, em, rem, vh, vw, vmin, vmax, in, mm, cm, pt, pc, ex, ch).
- 

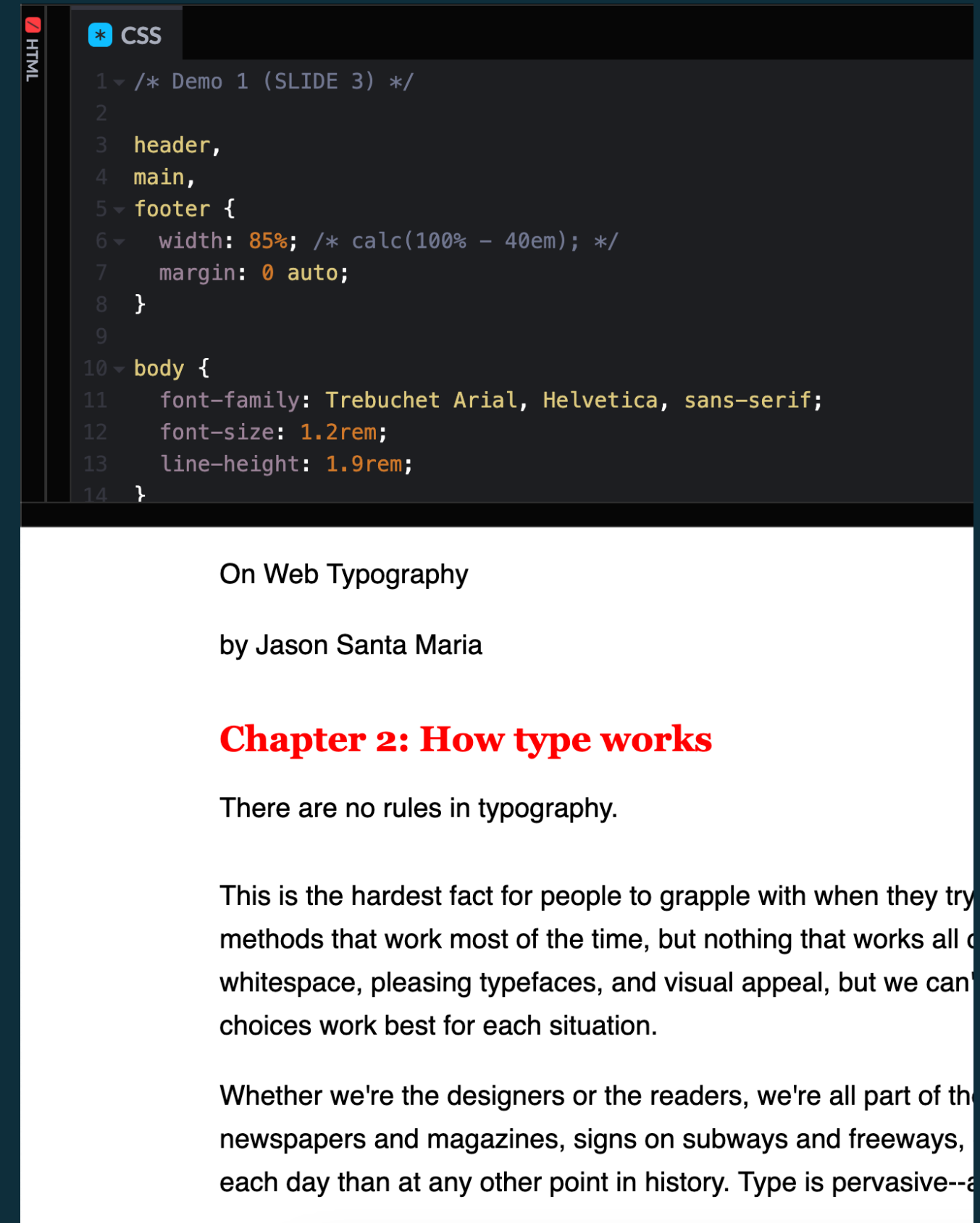
Why Should I use it?

1. It's helpful to mix different CSS units (e.g., px, %, rem, vw) for a single property value.
2. Puts the user's browser to work doing math that can't be done ahead of time
3. Can simplify and create a more dynamic responsive layouts

Demo 1: Margin(spacing)

```
header, main, footer {  
  max-width: 900px;  
  width: 85%;  
  margin: 0 auto;  
}
```

```
/* width: calc(100% - 40em); , calc(100vw - 2em) */
```



[Demo Link](#)

What are the rules?

1. You must have a space within the syntax of the calc function.
 - a. `calc(100% - 10px) /* valid */`
 - b. `calc(100%-10px) /* invalid */`
2. You must not have a space between the calc and ()
 - a. `calc (100% - 10px) /* invalid */`
3. Don't over abuse it. If the math can be done then do it instead

What are the rules?

- Operators
 - + (Addition) and - (Subtraction) require units on both sides.
 - `font-size: calc(1rem + 1vw);`
 - * (Multiplication) and / (Division) require one side to be unitless.
 - Example: `width: calc(100% / 3);` (the 3 has no unit).

What can you do with it?

1. Use of CSS variables. Defining a value once, modifying later, using calc()
 - a. `:root { variables }`
 - i. `{ --width: 100px; }`
 - b. `calc(var(--width) * 2);`
2. Can nest calc() with each other
 - a. `calc(100px + calc(200px + 2%))`
3. Can mix with other css functions. or use other functions within calc()
 - a. `clamp( 100px , calc(100vw - 20em), 800px)`

Demo 2 - Sticky header

- First Set a fixed height for the header (e.g., 60px).
- Use `calc(100vh - 60px)` for the main content to leave space for the header.
- Add `overflow-y: auto;` to enable scrolling within the main section.

```
* CSS
64 background-color: #333;
65 color: white;
66 height: 60px; /* important for sticky header */
67 }
68
69
70 main {
71   height: calc(100vh - 60px);
72   overflow-y: auto;
73 }
74
75
76
77
```

[Demo Link](#)

Demo 3: clamping ()

- Clamping allows to create a more responsive Design by setting a dynamic range for values
 - min / preferred / max
 - min: the smallest value allowed.
 - preferred: the ideal or “preferred” size.
 - max : the largest value allowed.

Demo 3: clamping ()

- Ex: `font-size: clamp(16px, 3vw, 36px)`
 - This allows you to mix absolute and relative units.
- Mixing clamp & calc
 - `clamp(1.5rem, calc(1rem + 2vw), 48px)`
 - The font size can never go **below** 1.5rem.
 - The **preferred** size is `calc(1rem + 2vw)`.
 - The font size can never go **above** 48px.

Demo 3: clamping ()

Breaking Down the Preferred Value: `calc(1rem + 2vw)`

- **1rem:**
 - By default, 1rem is 16px if the user hasn't changed their browser settings.
 - So `1rem = 16px` in a typical scenario.
- **2vw:**
 - vw is **viewport width**. 1vw equals 1% of the current viewport width.
 - If the viewport width is 900px, then 2vw means **2%** of 900px, which is `900px * 2 / 100 = 18px`.
- **Adding them together:**
 - `1rem (16px) + 2vw (18px) = 34px`. <-- hence the preferred size is 34px.

[Demo Link](#)

Demo 4 : Full-Width Element with `calc()`

[Demo Link](#)

How to make images or videos span the entire viewport

- First we need a centered container
 - (e.g., `main {`
`max-width: 600px`
`margin: 0 auto;`
`})`
- We want an image or video inside this container
 - Then have it expand beyond this container to the full browser width.
- Why use `calc()` for this? It's so we can align the element's center with the container's (the viewport center)

Demo 4 : Full-Width Element with `calc()`

[Demo Link](#)

The Full-Width CSS

- `img[src*="images.unsplash"] {`
 `display: block;`
 `width: 100vw;`
 `max-width: 100vw;`
 `margin: 0 calc(-50vw + 50%);`
 `}`
- `width: 100vw` = image/video is as wide as the browser window.
- `-50vw` shifts the element left by half its own width.
- `+ 50%` nudges it back so imgs center lines up with the container's(main) center (which is already at the viewport's 50%).

Sites

1. <https://jdsteinbach.com/css/where-to-use-css-calc-real-world/>
2. <https://www.geeksforgeeks.org/css-calc-function/>
3. <https://medium.com/@anton.martyniuk/complete-guide-to-building-responsive-ui-with-css-calc-a787832dc8b3>



THANK YOU

Q & A