8. Variables, Pseudo-Selectors, Transform and Transition Animations

Table of Content

1. Math Functions in CSS

We have already seen CSS functions like `rgb()` and `hsl()` . These functions can be used as the values of various CSS properties.

Furthermore, We have the following important functions:

- **calc()**

  The **calc(expression)** function performs a calculation to be used as the property value.

  The expression is a required mathematical expression. The result will be used as the value.

  The following operators can be used: **+ - * /**

```
#div1 {
  width: calc(100% - 100px);
}
```

▼ **max()**

The max() function uses the largest value, from a comma-separated list of values, as the property value.

```
#div1 {
  width: max(50%, 300px);
}
```

▼ **min()**

The max() function uses the smallest value, from a comma-separated list of values, as the property value.

```
#div1 {
  width: min(50%, 300px);
}
```

2. Variables in CSS

CSS variables can be used to define variables. CSS variables have access to the DOM, which means that you can create variables with local or global scope, change the variables with JavaScript, and change the variables based on media queries.

The `var()` function is used to insert the value of a CSS variable.

> **Note:** The variable name must begin with two dashes (--) and it is case sensitive!

CSS variables can have a global or local scope.

▼ **Global variables**

Global variables can be accessed/used through the entire document, while local variables can be used only inside the selector where it is declared.

To create a variable with global scope, declare it inside the `:root` selector. The `:root` selector matches the document's root element.

```
:root {
  --blue: #1e90ff;
  --white: #ffffff;
}

button {
  background-color: var(--white);
}
```

▼ Local variables

To create a variable with local scope, declare it inside the selector that is going to use it. You can even override the value of a global variable by defining it again inside a local scope.

```
button {
  --blue: #0000ff; /* local variable will override global */
  background-color: var(--white);
}
```

3. Pseudo-Classes and pseudo-elements

▼ Pseudo-Classes

A pseudo-class is used to define a special state of an element.

```
/* visited link */
a:visited {
  color: #00FF00;
}

/* mouse over link */
a:hover {
  color: #FF00FF;
}
```

Also, pseudo-classes are always preceded by a colon (`:`). Then comes the name of the pseudo-class

A few common pseudo-classes are:

- **:visited**
- **:hover**
- :**first-child**
- **:last-child**
- **:nth-child**

▼ **Pseudo-Elements**

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

Also, pseudo-elements are preceded by a colon (`::`). Then comes the name of the pseudo-class

A few common pseudo-elements are:

- ::**first-letter**
- **::first-line**
- **::before**
- **::after**

4. Transform in CSS

▼ The `transform` property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements

- `scale()`**:** Affects the size of the element. This also applies to the `font-size`, `padding`, `height`, and `width` of an element, too. It's also a a shorthand function for the `scaleX` and `scaleY` functions.

- `skewX()` **and** `skewY()`**:** Tilts an element to the left or right, like turning a rectangle into a parallelogram. `skew()` is a shorthand that combines `skewX()` and `skewY` by accepting both values.
- `translate()`**:** Moves an element sideways or up and down.
- `rotate()`: Rotates the element clockwise from its current position.

5. Transition in CSS

CSS transitions allows you to change property values smoothly, over a given duration. It handles the following properties:

- **transition :** shorthand property for the below 4 properties.

  To create a transition effect, you must specify two things:

  - the CSS property you want to add an effect to
  - the duration of the effect

  Note**:** If the duration part is not specified, the transition will have no effect, because the default value is 0.
- **transition-delay :** adds a delay after which the transition starts.
- **transition-duration :** is the duration of the transition
- **transition-property :** tells which property should this transition be applied.
- **transition-timing-function :** specifies the speed curve of the transition effect.

  The transition-timing-function property can have the following values:

  - `ease` - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
  - `linear` - specifies a transition effect with the same speed from start to end
  - `ease-in` - specifies a transition effect with a slow start
  - `ease-out` - specifies a transition effect with a slow end
  - `ease-in-out` - specifies a transition effect with a slow start and end

- `cubic-bezier(n,n,n,n)` - lets you define your own values in a cubic-bezier function

Further Readings:

https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/

Assignments

1. Build the following animations:

> https://s3-us-west-2.amazonaws.com/secure.notion-static.com/8c1e3bc6-0053-4e66-9d72-5a6a034a4a14/Screen_Recording_2022-11-16_at_8.27.34_PM.mov

2. Build this toggle button

> https://s3-us-west-2.amazonaws.com/secure.notion-static.com/37f8a296-f2c6-4a06-b080-2986238b76ac/Screen_Recording_2022-11-16_at_8.27.08_PM.mov

3. Build the button hover animations mentioned in this website:

> https://freefrontend.com/css-button-hover-effects/