Chapter 37: Custom Properties (Variables)

CSS Variables allow authors to create reusable values which can be used throughout a CSS document.

For example, it's common in CSS to reuse a single color throughout a document. Prior to CSS Variables this would mean reusing the same color value many times throughout a document. With CSS Variables the color value can be assigned to a variable and referenced in multiple places. This makes changing values easier and is more semantic than using traditional CSS values.

Section 37.1: Variable Color

```
:root {
  --red: #b00;
  --blue: #4679bd;
  --grey: #ddd;
}
.Bx1 {
  color: var(--red); background:
  var(--grey); border: 1px solid
  var(--red);
}
```

Section 37.2: Variable Dimensions

```
:root {
  --W200: 200px;
  --W10: 10px;
}
.Bx2 {
  width: var(--W200);
  height: var(--W200);
  margin: var(--W10);
}
```

Section 37.3: Variable Cascading

CSS variables cascade in much the same way as other properties, and can be restated safely.

You can define variables multiple times and only the definition with the highest specificity will apply to the element selected.

Assuming this HTML:

```
<a class="button">Button Green</a>
<a class="button button_red">Button Red</a>
<a class="button">Button Hovered On</a>
```

We can write this CSS:

```
.button {
  --color: green;
  padding: .5rem;
  border: 1px solid var(--color);
  color: var(--color);
}
```

```
.button:hover {
  --color: blue;
}

.button_red {
  --color: red;
}
```

And get this result:

Section 37.4: Valid/Invalids

Naming When naming CSS variables, it contains only letters and dashes just like other CSS properties (eg: line-height, -moz-box-sizing) but it should start with double dashes (--)

```
//These are Invalids variable names
--123color: blue;
--#color: red;
--bg_color: yellow
--$width: 100px;

//Valid variable names
--color: red;
--bg-color: yellow
--width: 100px;
```

CSS Variables are case sensitive.

```
/* The variable names below are all different variables */
--pcolor: ;
--Pcolor: ;
--pColor: ;
```

Empty Vs Space

```
/* Invalid */
    --color:;

/* Valid */
  --color: ; /* space is assigned */
```

Concatenations

```
    /* Invalid - CSS doesn't support concatenation*/
    .logo{
        --logo-url: 'logo';
        background: url('assets/img/' var(--logo-url) '.png');
    }

    /* Invalid - CSS bug */
    .logo{
        --logo-url: 'assets/img/logo.png';
        background: url(var(--logo-url));
    }
```

```
/* Valid */
.logo{
    --logo-url: url('assets/img/logo.png');
    background: var(--logo-url);
}
```

Careful when using Units

```
/* Invalid */
--width: 10;
width: var(--width)px;

/* Valid */
--width: 10px;
width: var(--width);

/* Valid */
--width: 10;
width: calc(1px * var(--width)); /* multiply by 1 unit to convert */
width: calc(1em * var(--width));
```

Section 37.5: With media queries

You can re-set variables within media queries and have those new values cascade wherever they are used, something that isn't possible with pre-processor variables.

Here, a media query changes the variables used to set up a very simple grid:

HTML

```
<div></div>
<div></div>
<div></div>
<div></div>
```

CSS

```
:root{
    --width: 25%;
    --content: 'This is desktop';
}
@media only screen and (max-width: 767px){
    :root{
        --width:50%;
        --content: 'This is mobile';
    }
}
@media only screen and (max-width: 480px){
    :root{
        --width:100%;
    }
}

div{
    width: calc(var(--width) - 20px);
    height: 100px;
}
div:before{
    content: var(--content);
```

```
}

/* Other Styles */
body {
    padding: 10px;
}

div{
    display: flex;
    align-items: center;
    justify-content: center;
    font-weight:bold;
    float:left;
    margin: 10px;
    border: 4px solid black;
    background: red;
}
```

You can try resizing the window in this CodePen Demo

Here's an animated screenshot of the resizing in action: