HOWCOLORS.WORK

A CSS color notation guide.

Table of Contents

Intro

This booklet helps you understand the current and future color notations *(Color notations are different ways to write down colors in your CSS code.)* in CSS.

The aim is to teach you all different color notations available in CSS. Tell you whether the browsers support these notations already and show you some code examples.

The color spaces *(Color spaces are organizations of colors. A color space may define color swatches, like Pantone. Or have a more mathematical notation like sRGB or CMYK.)*, which CSS uses in these different ways to express colors are not part of this booklet. In some places I mention them but do not explain any color space in detail.

To make full use of this booklet, all you need to know is how to create an HTML tag and apply CSS to it. All the different color options you will learn here!

Most of the CSS colors are in the sRGB *(sRGB (standard RED GREEN BLUE) is an RGB color space created to for the use on monitors and the web.)* color space, except Lab and LCH, where the computed value is a CIELab *(The CIELAB color space defined by the International Commission on Illumination (CIE).)* color. Not to forget the new `color()` function. This makes it possible to pass your own or predefined color profiles!

All these color notations offer an optional parameter for the alpha value. The alpha value indicates how transparent the color is. In the booklet, there is an extra section in each chapter that shows how to use the alpha values.

Before you use colors in your HTML documents, please consider the W3C Web Content Accessibility Guidelines *(https://www.w3.org/TR/WCAG20/)*. Most important is the contrast between your foreground and background color. For the current color notations, you can check the contrast with my Color Contrast Checker *(https://convertingcolors.com/color-contrast-checker.html)*.

Current Ways to Express Colors!

Current Ways to Express Colors!

Code Samples

All the examples in this booklet use the same HTML structure. To make the examples look nice, I added a simple CSS grid and some basic formatting. The colors I define as CSS variables on the top.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
        :root{
            --color:#128458;
            --color-alpha:#12845877;
        }

        body{
            font-size:20pt;
            font-family:monospace;
            margin:1rem;
        }

        h2{
            margin:0;
        }

        h3{
            margin:0;
            margin-top:2rem;
```

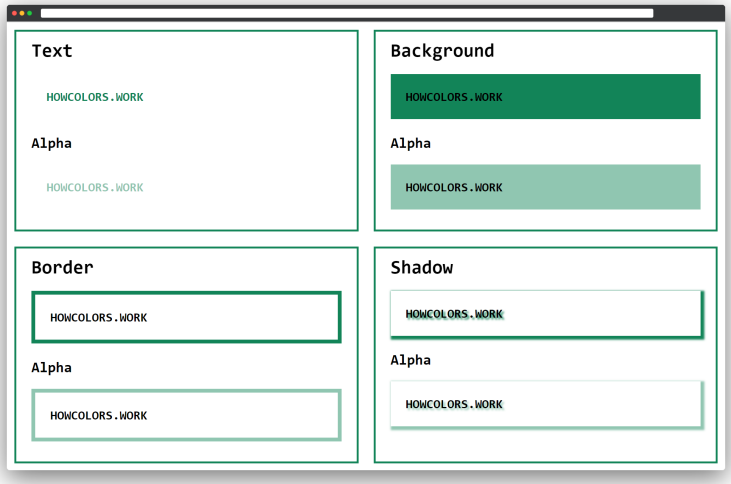
```
        }

        p{
            padding:2rem;
            font-weight:bold;
        }

        .grid{
            display: grid;
            grid-template-columns: repeat(auto-fill,minmax(640px,1fr));
            grid-gap:2rem;
        }

        .grid div{
            border:4px solid #128458;
            padding:2rem;
            padding-top:1rem;
            padding-bottom:1rem;
        }

        .textsample{
            color:var(--color);
        }

        .textsample_alpha{
            color:var(--color-alpha);
        }

        .bgsample{
            background-color:var(--color);
        }

```

```
55          .bgsample_alpha{
56              background-color:var(--
color-alpha);
57          }
58
59          .bordersample{
60              border:8px solid var(--
color);
61          }
62
63          .bordersample_alpha{
64              border:8px solid var(--
color-alpha);
65          }
66
67          .shadowsample{
68              text-shadow:4px 4px 4px
var(--color);
69              box-shadow:4px 4px 4px 4px
var(--color);
70          }
71
72          .shadowsample_alpha{
73              text-shadow:4px 4px 4px
var(--color-alpha);
74              box-shadow:4px 4px 4px 4px
var(--color-alpha);
75          }
76      </style>
77    </head>
78    <body>
79      <div class="grid">
80        <div>
81          <h2>Text</h2>
```

```
          <p class="textsample">HOWCOLORS.WORK</p>
          <h3>Alpha</h3>
          <p class="textsample_alpha">HOWCOLORS.WORK</p>
        </div>
        <div>
          <h2>Background</h2>
          <p class="bgsample">HOWCOLORS.WORK</p>
          <h3>Alpha</h3>
          <p class="bgsample_alpha">HOWCOLORS.WORK</p>
        </div>
        <div>
          <h2>Border</h2>
          <p class="bordersample">HOWCOLORS.WORK</p>
          <h3>Alpha</h3>
          <p class="bordersample_alpha">HOWCOLORS.WORK</p>
        </div>
        <div>
          <h2>Shadow</h2>
          <p class="shadowsample">HOWCOLORS.WORK</p>
          <h3>Alpha</h3>
          <p class="shadowsample_alpha">HOWCOLORS.WORK</p>
        </div>
      </div>
```

```
  </body>
</html>
```

The result of this code will give you four boxes for the text, background, border, and shadow CSS examples.

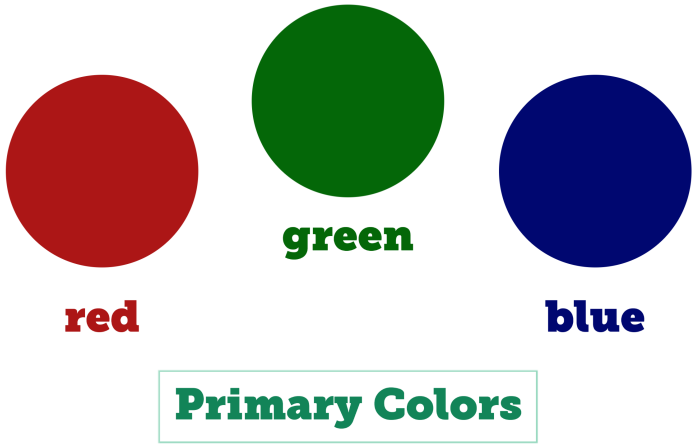


The browser output of the above code

In the following CSS examples, we will change the `--color` and `--color-alpha` variables. Instead of a variable as color `color:var(--color);` you could of course also pass the color value `color:#FF0000;`.

Hex

The hex color notation allows you to define sRGB colors as hexadecimal numbers. It starts with a hash #, and then you have the three primary colors, red, green and last the blue color.



The primary colors, red, green, and blue.

The hex notation only allows ten numerals (0-9) and six letters (a-f). Where `00` would be `0` in the RGB functional notation, and `FF` would be `255`.
The higher the value of a primary color is, the brighter that color is. This means that the hex color `#000000` has zero brightness in all three primary colors and is black. The color `#FFFFFF` is the opposite as all colors have the full light, which leads to white.

A example of brighter and darker versions of red `#FF0000`

If you use any RGB color format to express colors, you should consider writing them in this notation. This is true in case the file size is the most important to you and readability comes second. The hex notation gives you the shortest color strings. You can define some colors in the three-digit notation. That three-digit short-form reduces your color string (including #) to four or five characters.

Syntax

#442B82

Breakdown

1	**2**	**3**	**4**
#	44	2B	82

1

The # indicates that the following is a hex color notation.

2

The red part of the color, two hexadecimal characters (0–9, A–F).

3

The green part of the color, two hexadecimal characters.

4

The blue part of the color, two hexadecimal characters.

Alpha

#442B8277

Breakdown

1	**2**	**3**	**4**	**5**
#	44	2B	82	77

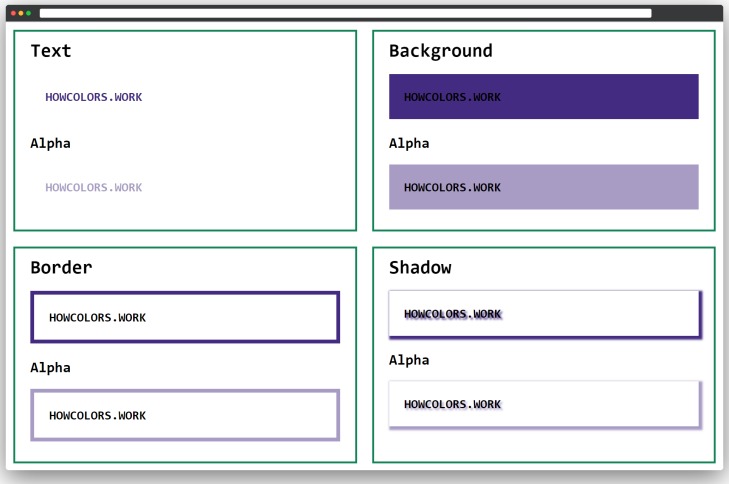
1 till 4

These parts are the same as for the color notation without an alpha value. To get details have a look at the section above.

5

The opacity part of the color, two hexadecimal characters. Where `FF` = 100%, `CC` = 80%, `99` = 60%, `66` = 40%, `33` = 20%, and `00` = 0%.

Code Samples



Result with the color values #442B82 and #442B8277

```
:root{
  --color:#442B82;
  --color-alpha:#442B8277;
}
```

Three-digit notation

It is possible to use a three-digit or four-digit (including the alpha value) notation. In this short-form the single digits get doubled. So, for example, `#0FC` would be `#00FFCC` in the standard form. The downside is you only have 4096 colors available in the short-form.

Three-digit Syntax

#840

Breakdown

1	**2**	**3**	**4**
#	8	4	0

1

The # indicates that the following is a hex color notation.

2

The red part of the color, one hexadecimal character (0–9, A–F).

3

The green part of the color, one hexadecimal character.

4

The blue part of the color, one hexadecimal character.

Alpha

#8407

Breakdown

1	**2**	**3**	**4**	**5**
#	8	4	0	7

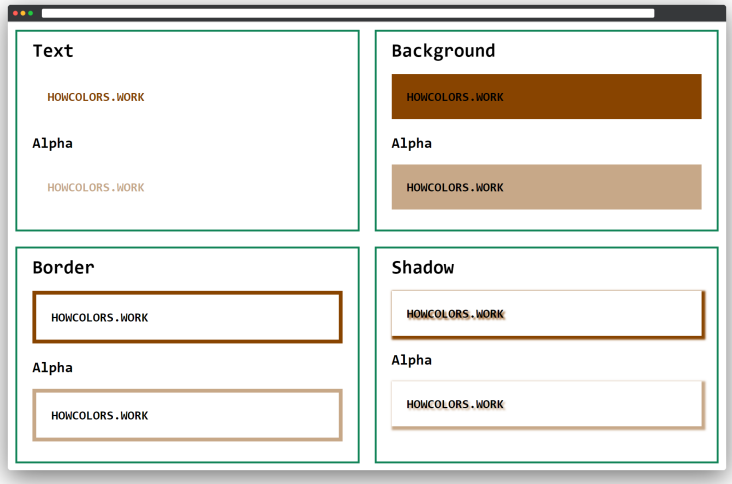
1 till 4

These parts are the same as for the color notation without an alpha value. To get details have a look at the section above.

5

The opacity part of the color, one hexadecimal character. Where `F` = 100%, `C` = 80%, `9` = 60%, `6` = 40%, `3` = 20%, and `0` = 0%.

Code Samples



Result with the color values #840 and #8407

```
:root{
  --color:#840;
  --color-alpha:#8407;
}
```

Browser Support

The hex notation, is supported in all browsers *(https://caniuse.com/#feat=mdn-css_types_color_rgb_hexadecimal_notation)*. The four and eight-digit short-form has support in all major browsers *(https://caniuse.com/#feat=css-rrggbbaa)*.

Exercise

1. How many characters can a hex color have (without #)?

    1. 4
    2. 5
    3. 2

2. The hexadecimal alpha value `66` in percent?

3. How many colors are available in the three-digit notation?

    1. Same as in the long format
    2. 16.777.216
    3. 4096

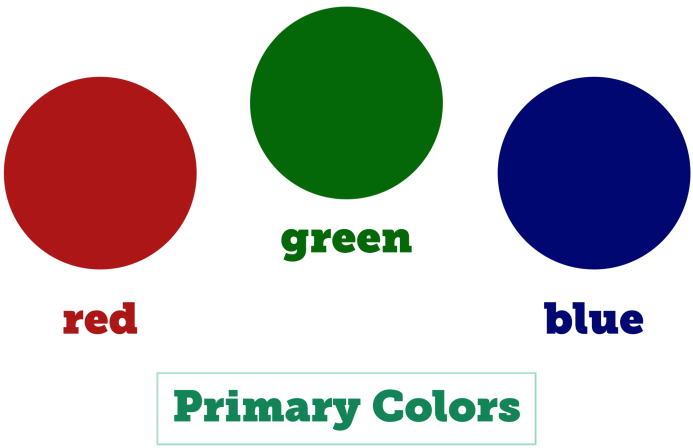
RGB

The RGB functions build the colors from a combination of red, green, and blue. Each of these color values defines the intensity of the color as an integer value. The values range from `0` to `255` or from `0%` to `100%` percent. If you pass `0` to all three values you will get black, if you pass `255` to all you get white.

The computed value of the following functions is a color in the sRGB color space.

The functional RGB notation is longer than the hex, but it is easier to read as we have integer values.

The Primary Colors



The primary colors, red, green, and blue.

Red

Red	Green	Blue
`255`	`0`	`0`

If you only set the red value to the highest possible value `255`, it gives you the brightest red, which is possible.

Green

Red	Green	Blue
`0%`	`100%`	`0%`

Here only the green value is set to `100%`, which gives you the brightest green possible.

Blue

Red	Green	Blue
`0`	`0`	`255`

The same logic applies for blue, so by setting the value to `255` and the other two values to `0`, you get the brightest blue.

Old Syntax

rgb(15, 64, 174)

Breakdown

1	2	3	4	5	6	7	8	9	10
rgb	(	15	,		64	,		174	)

1

The function starts with an `rgb`.

2

Following the function name, an opening bracket.

3

The red value of the color can be a number or a percentage. The numbers are in the range of `0` - `255`, where `255` equals `100%`.

4

A comma as the separator between the color values.

5

An optional space.

6

The green value of the color. Allowed are the same values as for red.

7

A comma as the separator between the color values.

8

An optional space.

9

The blue value of the color. Allowed are the same values as for red and green.

10

The functions closing bracket.

Alpha

rgba(15, 64, 174, 80%)

Breakdown

1	2	3	4	5	6	7	8	9	10	11	12	13
rgba	(	15	,		64	,		174	,		80%	)

1

If you want to add the colors alpha value, the old syntax's function needs to start with a `rgba`.

2 till 9

These parts are the same as for the color notation without an alpha value. To get details have a look at the section above.

10

A comma as the separator between the color values and the alpha value.

11

An optional space.

12

You can define the alpha (opacity) value as a number between `0` and `1` or as a percentage. For example, `1` or `100%` means full opacity, while `0.5` or `50%` is means half opacity.

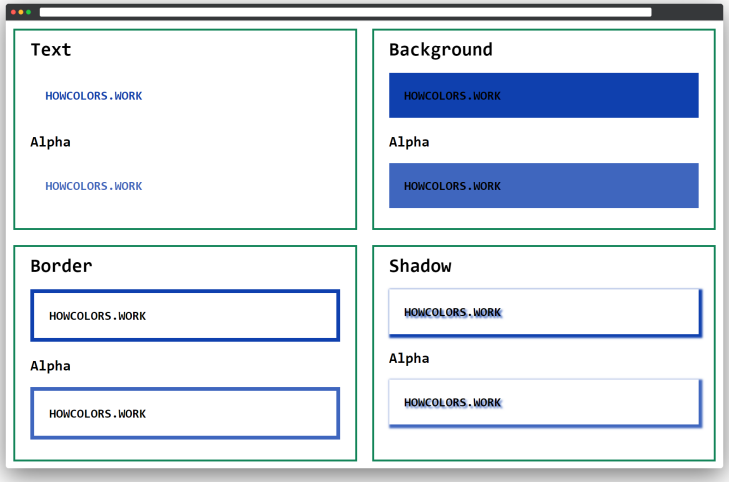
13

The functions closing bracket.

Browser Support

All major browsers support *(https://caniuse.com/#feat=mdn-css_types_color_rgb_functional_notation)* the RGB functional notation.

Code Samples



Result with the color values rgb(15, 64, 174) and rgba(15, 64, 174, 80%)

```
:root{
  --color:rgb(15, 64, 174);
  --color-alpha:rgba(15, 64, 174, 80%);
}
```

New Syntax

rgb(47 27 41)

Breakdown

1	2	3	4	5	6	7	8
rgb	(	47		27		41	)

1

Same as the old syntax, the new one starts an `rgb`.

2

An opening bracket follows the `rgb`.

3

The red value of the color can be a number or a percentage. The numbers are in the range of `0` - `255`, where `255` equals `100%`.

4

Space as the separator between the color values.

5

The green value of the color. Allowed are the same values as for the red (3) part.

6

Space as the separator between the color values.

7

The blue value of the color also can be a number or a percentage.

8

The functions closing bracket.

Alpha

rgb(47 27 41 / 0.6)

Breakdown

1	2	3	4	5	6	7	8	9	10	11	12
rgb	(	47		27		41		/		0.6	)

1

In the new syntax, the functional notation, including the alpha value, also starts with `rgb`.

2 till 7

These parts are the same as for the color notation without an alpha value. To get details have a look at the section above.

8

Space as the separator between the color values and the alpha value.

9

A `/` as the separator between the color values and the alpha value.

10

Space.

11

You can define the alpha (opacity) value as a number between `0` and `1` or as a percentage. For example, `1` or `100%` means full opacity, while `0.5` or `50%` means half opacity.

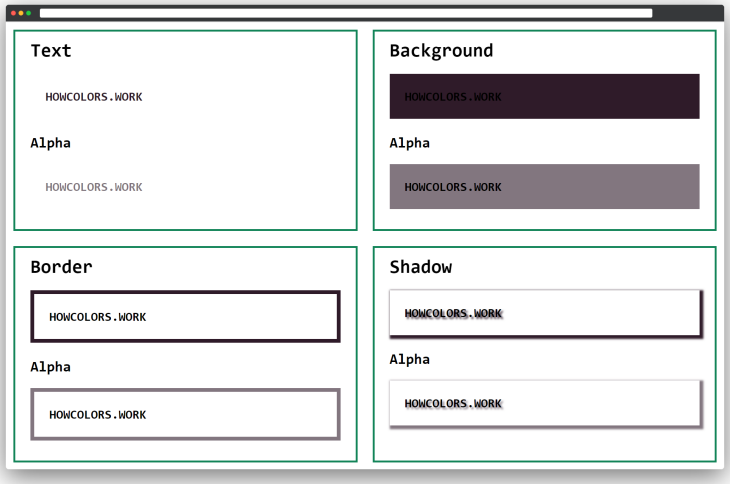
12

The functions closing bracket.

Browser Support

The browser support for the space-separated functional color notations is excellent. All major browsers support it already. For details check caniuse.com *(https://caniuse.com/#feat=mdn-css_types_color_space_separated_functional_notation)*.

Code Samples



Result with the color values rgb(47 27 41) and rgb(47 27 41 / 0.6)

```
:root{
  --color:rgb(47 27 41);
  --color-alpha:rgb(47 27 41 / 0.6);
}
```

Exercise

1. Which notation is invalid?

    1. rgb(55 10 0)
    2. rgba(200, 73, 12, .75)
    3. rgb(255 100 100, 100%)
    4. rgb(100% 10% 0%)

2. The `G` in RGB stands for?

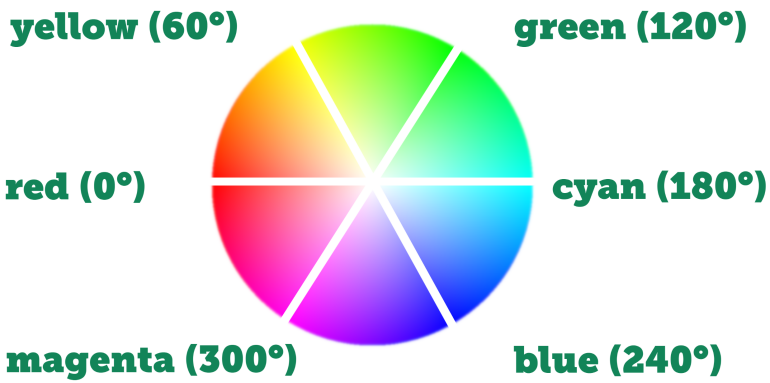
3. You can pass a percentage from 0 - 100%, but what is the range for integer values?

    1. 0 - 255
    2. 0 - 100
    3. 0 - 360

HSL

The hexadecimal and functional RGB notations are quite tricky to read and alter. The HSL functional notation is easier to read and work with. It makes it easy to, for example, get a darker version of the same hue.

The hue is an angle on the color wheel, where red equals `0` and `360` degree. Green equals `120` degree and blue `240` degree.



Color Wheel

Changing the hue makes it easy to get complementing colors. All you need to do is add 180 degrees to get the complementary color.

Old Syntax

hsl(3, 76%, 27%)

Breakdown

1	2	3	4	5	6	7	8	9	10
hsl	(	3	,		76%	,		27%	)

1

The function starts with an `hsl`.

2

An opening bracket follows the function name.

3

This part defines the hue, which is an angle on the color circle. You define it as a unitless number, which CSS takes as degrees.

4

A comma as the separator between the color values.

5

An optional space.

6

The saturation of your color as a percentage. `100%` means completely saturated, and `0%` is completely unsaturated so gray.

7

A comma as the separator between the color values.

8

An optional space.

9

The lightness of your HSL color, where `100%` is white, and `0%` lightness is black.

10

The functions closing bracket.

Alpha

hsla(3, 76%, 27%, 50%)

Breakdown

1	2	3	4	5	6	7	8	9	10	11	12	13
hsla	(	3	,		76%	,		27%	,		50%	)

1
: The function starts with an `hsla`.

2 till 9
: These parts are the same as for the color notation without an alpha value. To get details have a look at the section above.

10
: A comma as the separator between the color values and the alpha value.

11
: An optional space.

12
: You can define the alpha (opacity) value as a number between `0` and `1` or as a percentage. For example, `1` or `100%` means full opacity, while `0.5` or `50%` means half opacity.

13
: The functions closing bracket.

Browser Support

The HSL notation is supported in all browsers *(https://caniuse.com/#feat=mdn-css_types_color_hsl)*. The

alpha color values in the old HSLA syntax are also supported. Details you can find on caniuse.com *(https://caniuse.com/#feat=mdn-css_types_color_alpha)*.

Code Samples

Result with the color values hsl(3, 76%, 27%) and hsla(3, 76%, 27%, 50%)

```
:root{
  --color:hsl(3, 76%, 27%);
  --color-alpha:hsla(3, 76%, 27%, 50%);
}
```

New Syntax

hsl(295 85% 28%)

Breakdown

1	2	3	4	5	6	7	8
hsl	(	295		85%		28%	)

1

The function starts with an `hsl`.

2

An opening bracket follows the function name.

3

This part defines the hue, which is an angle on the color circle. You define it as a unitless number, which CSS takes as degrees.

Red = `0` and `360` degree.
Green = `120` degree.
Blue = `240` degree.

In the new CSS Color Module Level 4, you could define the angle as degrees, rads, grads, or turns. For example, `1turn`.

4

Space as the separator between the color values.

5

The saturation of your color as a percentage. `100%` is completely saturated, and `0%` is completely unsaturated so gray.

6
: Space as the separator between the color values.

7
: The lightness of your HSL color, where `100%` is white, and `0%` lightness is black.

8
: The functions closing bracket.

Alpha

hsl(295 85% 28% / .75)

Breakdown

1	2	3	4	5	6	7	8	9	10	11	12
hsl	(	295		85%		28%		/		.75	)

1
: The function starts with an `hsl`.

2 till 7
: These parts are the same as for the color notation without an alpha value. To get details have a look at the section above.

8
: Space as the separator between the color values and the alpha value.

9
: A `/` as the separator between the color values and the alpha value.

10
: Space.

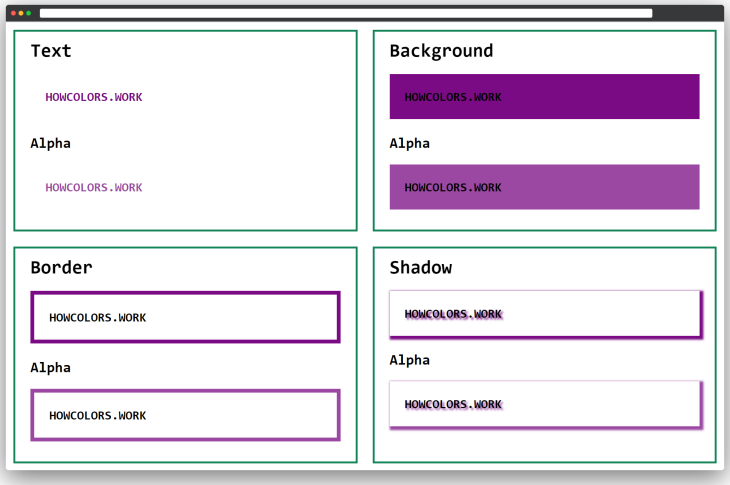
11
: You can define the alpha (opacity) value as a number between `0` and `1` or as a percentage. For example, `1` or `100%` means full opacity, while `0.5` or `50%` means half opacity.

12
: The functions closing bracket.

Browser Support

The browser support for the space-separated functional color notations is excellent. All major browsers support it already. For details check caniuse.com *(https://caniuse.com/#feat=mdn-css_types_color_space_separated_functional_notation)*.

Code Samples



Result with the color values hsl(295 85% 28%) and hsl(295 85% 28% / .75)

```
:root{
  --color:hsl(295 85% 28%);
  --color-alpha:hsl(295 85% 28% / .75);
}
```

Exercise

1. Which notation is invalid?

    1. hsl(75deg 10% 70%)
    2. hsl(300 100 50)
    3. hsl(300 100% 50%)

2. The `L` in HSL stands for?

3. At which degree on the color wheel do you find blue?

    1. 20 degree
    2. 120 degree
    3. 240 degree

Keywords

The current chapter would not be complete without a mention of the color keywords. You can use these keywords besides the already covered notations.

Transparent

The `transparent` keyword gets computed to a transparent black, like `#000000` or `rgb(0, 0, 0, 0)`.

Browser Support

All browsers support the `transparent` keyword. For details have a look at caniuse.com *(https://caniuse.com/#feat=mdn-css_types_color_transparent)*.

Code Sample

```
div {
    background-color:transparent;
    color:transparent;
}
```

Current Color

If you use the `currentcolor` keyword you get the value of the CSS `color:` property on the current element. For example, set the color of your element to yellow. Now use the `currentcolor` keyword as the background color. The browser output for this background color would also be yellow.

Browser Support

This keyword is also supported in all browsers *(https://caniuse.com/#feat=currentcolor)*.

Code Sample

```
div {
    background-color:currentcolor;
    color:yellow;
}
```

Named Colors

CSS has taken over some named colors from HTML and extended the list of named colors a lot. You can find a list of all named colors in the CSS Color Module *(https://www.w3.org/TR/css-color-4/#named-colors)*. All these color names resolve to a color in the sRGB color space.

If you look for the smallest CSS file possible. It is worth having a look at the named colors. Some names are even shorter than their hexadecimal notation.

Browser Support

Color names are supported in all browsers *(https://caniuse.com/#feat=mdn-css_types_color_keyword_color_values)*.

Code Sample

```
1 div {
2     background-color:gold;
3     color:gold;
4 }
```

New Ways to Express Colors!

New Ways to Express Colors!

Have a look at the CSS Color Module Level 4 working draft *(https://www.w3.org/TR/css-color-4/)*. You can see a lot of new ways to express colors in CSS.

All these new notations only support the space-separated functional notation. So commas as separators are not allowed.

Unfortunately, no browser supports these notations yet. But you should prepare yourself for the future and learn them now!

Code Samples Update

As these new formats are not supported yet, you should make a small change to the CSS. You can use the color variables as fallback colors and define the color in the new syntax in the selector.

Unfortunately, you can not use a second CSS variable here. I tested this, and the invalid color notation is not detected when it comes from a variable.

Still, in this way, you can already start defining colors with the new notations!

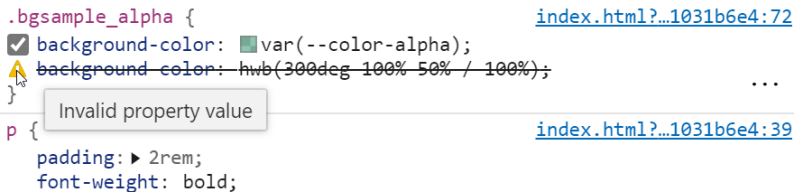
```
<style>
    :root{
        /* Use these as fallbacks now */
        --color:#808080;
        --color-alpha:#808080FF;
    }

    body{
        font-size:20pt;
        font-family:monospace;
        margin:1rem;
    }

    h2{
        margin:0;
    }

    h3{
        margin:0;
```

```
        margin-top:2rem;
    }

    p{
        padding:2rem;
        font-weight:bold;
    }

    .grid{
        display: grid;
        grid-template-columns:
repeat(auto-fill,minmax(640px,1fr));
        grid-gap:2rem;
    }

    .grid div{
        border:4px solid #128458;
        padding:2rem;
        padding-top:1rem;
        padding-bottom:1rem;
    }

    .textsample{
        color:var(--color);
        color:hwb(300deg 50% 50%);
    }

    .textsample_alpha{
        color:var(--color-alpha);
        color:hwb(300deg 50% 50% /
100%);
    }

    .bgsample{
```

```
        background-color:var(--color);
        background-color:hwb(300deg 50% 50%);
    }

    .bgsample_alpha{
        background-color:var(--color-alpha);
        background-color:hwb(300deg 50% 50% / 100%);
    }

    .bordersample{
        border:8px solid var(--color);
        border:8px solid hwb(300deg 50% 50%);
    }

    .bordersample_alpha{
        border:8px solid var(--color-alpha);
        border:8px solid hwb(300deg 50% 50% / 100%);
    }

    .shadowsample{
        text-shadow:4px 4px 4px var(--color);
        text-shadow:4px 4px 4px hwb(300deg 50% 50%);
        box-shadow:4px 4px 4px 4px var(--color);
        box-shadow:4px 4px 4px 4px hwb(300deg 50% 50%);
```

```
76      }
77
78      .shadowsample_alpha{
79          text-shadow:4px 4px 4px var(--
color-alpha);
80          text-shadow:4px 4px 4px
hwb(300deg 50% 50% / 100%);
81          box-shadow:4px 4px 4px 4px
var(--color-alpha);
82          box-shadow:4px 4px 4px 4px
hwb(300deg 50% 50% / 100%);
83      }
84 </style>
```

This will give you the same result as the initial code sample. But in your developer tools, you can see a difference. The browser first tries to understand the new color syntax and after that goes to the fallback.



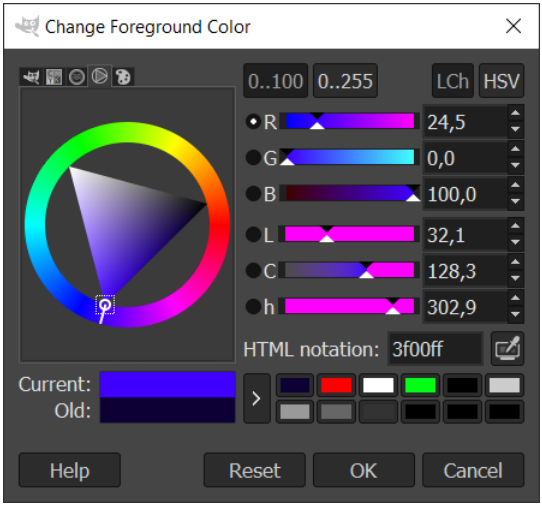
Developer Tools – Invalid property value

In the CSS examples below, you will only see the text color example. You can check other cases, like the background, here as the color notation works the same.

HWB

HWB is another functional notation to define colors, similar to HSL. The hue is an angle of `0` - `360` degrees, and values of `0%` - `100%` for the whiteness and blackness.

This new notation is even easier to understand then the HSL notation. That's the reason why many color pickers, offer the possibility to select colors based on HWB.



GIMP Color Picker

Syntax

hwb(193deg 4% 54%)

Breakdown

1	2	3	4	5	6	7	8
hwb	(	193deg		4%		54%	)

1

The function starts with an `hwb`.

2

An opening bracket follows the function name.

3

This part defines the hue, which is an angle on the color circle.

Red = `0` and `360` degree.
Green = `120` degree.
Blue = `240` degree.

4

Space as the separator between the color values.

5

The whiteness of your color as a percentage. `100%` is full whiteness, and `0%` is no whiteness at all.

6

Space as the separator between the color values.

7

The blackness of your HWB color, where `100%` is full blackness, and `0%` no blackness at all.

8
: The functions closing bracket.

Alpha

hwb(193deg 4% 54% / 90%)

Breakdown

1	2	3	4	5	6	7	8	9	10	11	12
hwb	(	193deg		4%		54%		/		90%	)

1 till 7
: These parts are the same as for the color notation without an alpha value. To get details have a look at the section above.

8
: Space as the separator between the color values and the alpha value.

9
: A `/` as the separator between the color values and the alpha value.

10
: Space.

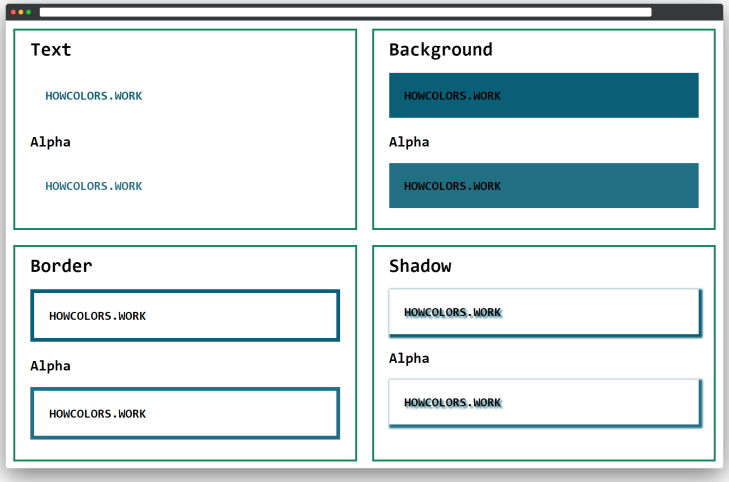
11
: You can define the alpha (opacity) value as a number between `0` and `1` or as a percentage. For example, `1` or `100%` means full opacity, while `0.5` or `50%` means half opacity.

12
: The functions closing bracket.

Browser Support

No browser supports this way of expressing colors yet.

Code Samples



Result using the fallback color values #0A5E75 and #0A5E75E6

```
:root{
    /* Use these as fallbacks now */
    --color:#0A5E75;
    --color-alpha:#0A5E75E6;
}

.textsample{
    color:var(--color);
    color:hwb(193deg 4% 54%);
}

.textsample_alpha{
    color:var(--color-alpha);
    color:hwb(193deg 4% 54% / 90%);
}
```

Exercise

1. Which notation is invalid?

    1. hwb(75deg 30% 70%)
    2. hwb(300 25% 75% / 0.5)
    3. hwb(300deg, 50, 50)

2. The `H` in HWB stands for?

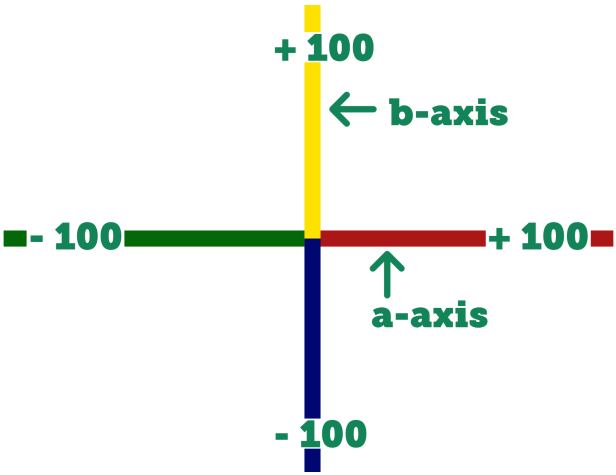
3. At which degree on the color wheel do you find green?

    1. 20 degree
    2. 120 degree
    3. 240 degree

Lab

The CIELab color space represents the complete range of colors a human can see. Once major browsers support this notation it means way more colors for you!

For the second and third function parameter we move along the a and b axes of the color space. There are positive and negative values allowed. Positive values along the a-axis give you a red while negative values are green. On the b-axis, positive values are yellow and negative values are blue.



A visual of the CIELab a and b axes.

Syntax

lab(36% 54 -6)

Breakdown

1	2	3	4	5	6	7	8
lab	(	36%		54		-6	)

1

The function starts with a `lab`.

2

An opening bracket follows the function name.

3

The first value is the CIE lightness; usually, it is between `0%` (black) and `100%` (white). Still, it can exceed that range for extra bright whites up to `400%`.

4

Space as the separator between the color values.

5

The second parameter is the distance along the a-axises in the Lab colorspace.

6

Space as the separator between the color values.

7

The third parameter is the distance along the b-axises in the Lab colorspace.

8

The functions closing bracket.

Alpha

lab(36% 54 -6 / 0.4)

Breakdown

1	2	3	4	5	6	7	8	9	10	11	12
lab	(	36%		54		-6		/		0.4	)

1 till 7
: These parts are the same as for the color notation without an alpha value. To get details have a look at the section above.

8
: Space as the separator between the color values and the alpha value.

9
: A `/` as the separator between the color values and the alpha value.

10
: Space.

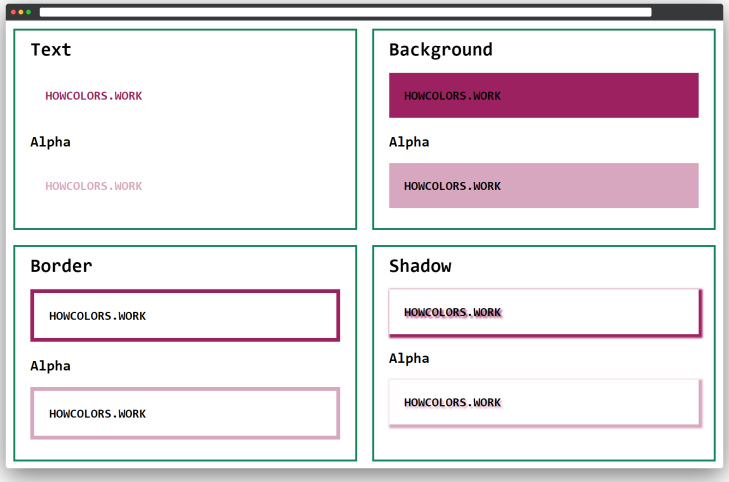
11
: You can define the alpha (opacity) value as a number between `0` and `1` or as a percentage. For example, `1` or `100%` means full opacity, while `0.5` or `50%` means half opacity.

12
: The functions closing bracket.

Browser Support

No browser supports this way of expressing colors yet.

Code Samples



Result using the fallback color values #9C2160 and #9C216066

```
:root{
    /* Use these as fallbacks now */
    --color:#9C2160;
    --color-alpha:#9C216066;
}

.textsample{
    color:var(--color);
    color:lab(36% 54 -6);
}

.textsample_alpha{
    color:var(--color-alpha);
    color:lab(36% 54 -6 / 0.4);
}
```

Exercise

1. Which notation is invalid?

    1. lab(100%, 100, 160)
    2. lab(50% 50 -160)
    3. lab(350% 50 160)

2. The `a` in Lab stands for?

3. What range does the first parameter allow?

    1. 0 - 400%
    2. 0 - 100%
    3. 0 - 360 degree

LCH

Another new way to express colors in the CSS 4 working draft is the `lch()` notation.

Syntax

lch(12% 8 211)

Breakdown

1	2	3	4	5	6	7	8
lch	(	12%		8		211	)

1

The function starts with an `lch`.

2

An opening bracket follows the function name.

3

The first value is the lightness; usually, it is between `0%` (black) and `100%` (white). Still, it can exceed that range for extra bright whites up to `400%`.

4

Space as the separator between the color values.

5

The second value is the amount of color or chroma, usually between `0` and `230`. A value above `77` exceeds the boundaries of the sRGB color space which most of the CSS color notations use.

6

Space as the separator between the color values.

7

This part defines the hue, which is an angle on the color circle (or wheel). See the HSL section for a visual of the color wheel.

8

The functions closing bracket.

Alpha

lch(12% 8 211 / 60%)

Breakdown

1	2	3	4	5	6	7	8	9	10	11	12
lch	(	12%		8		211		/		60%	)

1 till 7
: These parts are the same as for the color notation without an alpha value. To get details have a look at the section above.

8
: Space as the separator between the color values and the alpha value.

9
: A `/` as the separator between the color values and the alpha value.

10
: Space.

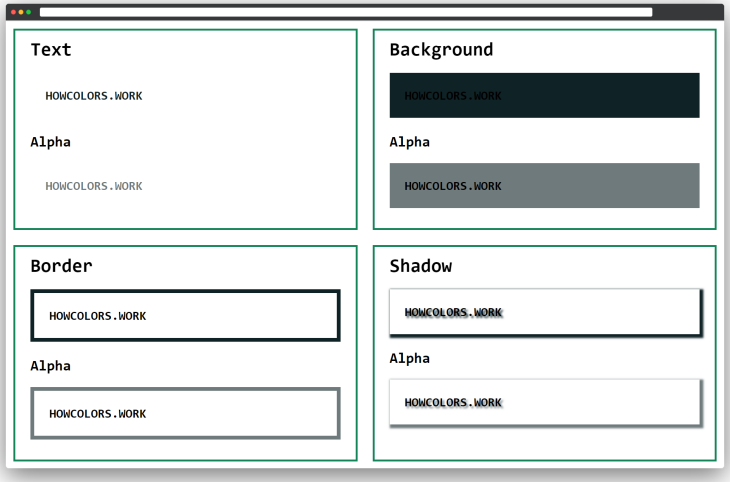
11
: You can define the alpha (opacity) value as a number between `0` and `1` or as a percentage. For example, `1` or `100%` means full opacity, while `0.5` or `50%` means half opacity.

12
: The functions closing bracket.

Browser Support

No browser supports this way of expressing colors yet.

Code Samples



Result using the fallback color values #0F2225 and #0F222599

```
:root{
    /* Use these as fallbacks now */
    --color:#0F2225;
    --color-alpha:#0F222599;
}

.textsample{
    color:var(--color);
    color:lch(12% 8 211);
}

.textsample_alpha{
    color:var(--color-alpha);
    color:lch(12% 8 211 / 60%);
}
```

Exercise

1. Which notation is invalid?

    1. lch(100% 100 160deg)
    2. lch(50% 50 60deg)
    3. lch(50% 50 380deg)

2. The h in Lab stands for?

3. What range of values can you use for the second parameter?

    1. 0 - 100
    2. 0 - 230
    3. 0 - 360

Color

With the notation `color()`, CSS will allow you to specify colors in reference to a color profile. You can pass the color profile as the first parameter. You can pass one of the predefined color profiles (`srgb`, `prophoto-rgb`, `display-p3`, `rec-2020`, `a98-rgb`) or you define an own profile.

If you want to add your own profile, you need to specify it first with a `@color-profile` rule. Then you can use it within the `color()` function, this works almost like defining new fonts via `@font-face`.

Syntax

color(display-p3 0.4087 0.1048 0)

Breakdown

1	2	3	4	5	6	7	8	9	10
color	(	display-p3		0.4087		0.1048		0	)

1
: The function starts with a `color`.

2
: An opening bracket follows the function name.

3
: This is an optional parameter defining the color space. Either you choose one of the predefined ones or one you created yourself.

4
: Space as the separator between the color values.

5 till 9
: Depending on the colorspace, you either pass a string or a set of numeric parameters here. If you provide fewer color values than the colorspace requires, CSS will take `0` for the remaining. This repeats with space as separator till the last value for the color space.

  In our example, we pass 3 values between `0` and `1`, all separated by a space.

10
: The functions closing bracket.

Alpha

color(display-p3 0.4087 0.1048 0 / .8)

Breakdown

1	2	3	4	5	6	7	8	9	10	11	12	13	14
color	(	display-p3		0.4087		0.1048		0		/		.8	)

1 till 9
: These parts are the same as for the color notation without an alpha value. To get details have a look at the section above.

10
: Space as the separator between the color values and the alpha value.

11
: A `/` as the separator between the color values and the alpha value.

12
: Space.

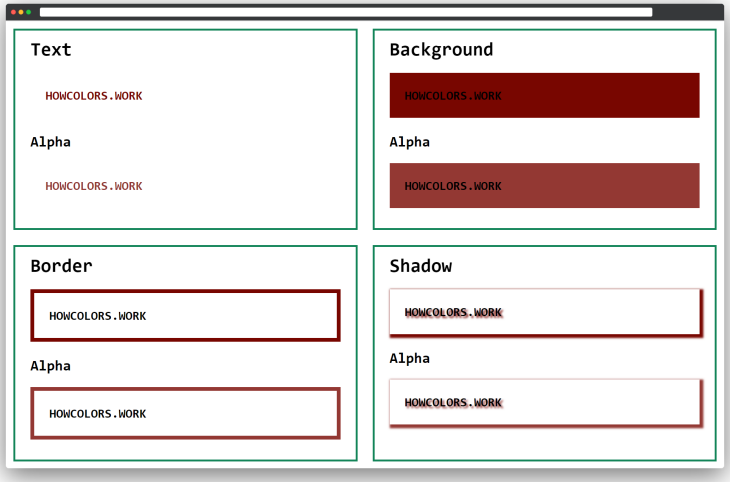
13
: You can define the alpha (opacity) value as a number between `0` and `1` or as a percentage. For example, `1` or `100%` means full opacity, while `0.5` or `50%` means half opacity.

14
: The functions closing bracket.

Browser Support

No browser supports this way of expressing colors yet.

Code Samples



Result using the fallback color #780600

```
:root{
    /* Use these as fallbacks now */
    --color:#780600;
    --color-alpha:#780600CC;
}

/* Defining a new Profile */
@color-profile sample {
  src: url('https://howcolors.work/sample.icc');
}

/* Using the own Profile */
.textsample {
    color:var(--color);
    color:color(sample 0.4087 0.1048 0);
}
```

```
17
18 /* Using the predefined display-p3
Profile */
19 .textsample{
20     color:var(--color);
21     color:color(display-p3 0.4087 0.1048
0);
22 }
23
24 .textsample_alpha{
25     color:var(--color-alpha);
26     color:color(display-p3 0.4087 0.1048
0 / .8);
27 }
```

Exercise

1. Which notation is invalid?

    1. color(display-p3 0.5 0.5 0.2)
    2. color(display-p3, 0.5 0.5 0.2)
    3. color(a98-rgb 0.2 0.2 0.2)

2. What does the first parameter define?

3. What range of values you can pass for the second and following parameters?

Gray

The `gray()` functional notation gives you completely desaturated colors. You only need to pass a single parameter, or two if you want to specify an alpha value. A value of `50`, so `gray(50)` would be right between black and white.

Syntax

gray(35)

Breakdown

1	2	3	4
gray	(	35	)

1
: The function starts with a `gray`.

2
: An opening bracket follows the function name.

3
: The value for gray `50` would be right between black and white.

4
: The functions closing bracket.

Alpha

gray(35 / 50%)

Breakdown

1	2	3	4	5	6	7	8
gray	(	35		/		50%	)

1 till 3
: These parts are the same as for the color notation without an alpha value to get details have a look in the section above.

4
: Space as the separator between the color values and the alpha value.

5
: A `/` as the separator between the color values and the alpha value.

6
: Space.

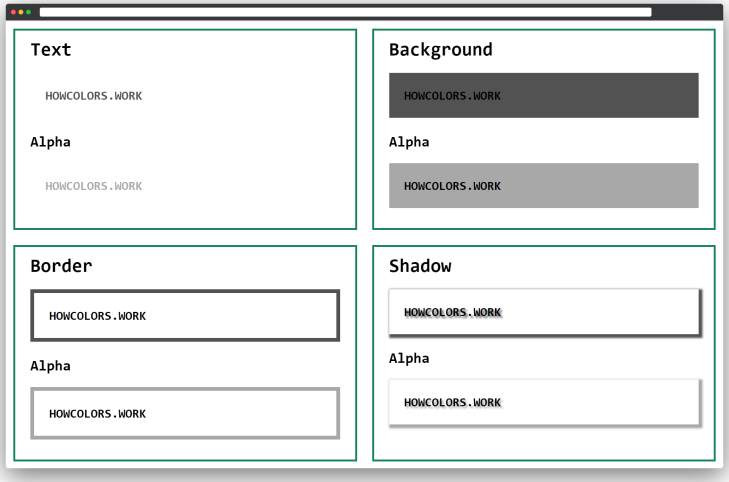
7
: You can define the alpha (opacity) value as a number between `0` and `1` or as a percentage. For example, `1` or `100%` means full opacity, while `0.5` or `50%` means half opacity.

8
: The functions closing bracket.

Browser Support

No browser supports this way of expressing colors yet.

Code Samples



Result using the fallback color for gray(35)

```
:root{
    /* Use these as fallbacks now */
    --color:#525252;
    --color-alpha:#52525280;
}

.textsample{
    color:var(--color);
    color:gray(35);
}

.textsample_alpha{
    color:var(--color-alpha);
    color:gray(35 / 50%);
}
```

Exercise

1. Which notation is invalid?

    1. gray(60 / 60%)
    2. gray(70)
    3. gray(10, 100%)

2. What does the first parameter define?

3. What range of values can you pass for the second parameter?

    1. 0 - 100
    2. 0 - 255
    3. 0 - 360 degree

Cool Ways to Use Colors!

Cool Ways to Use Colors!

In this chapter, I want to show you some cool ways to use colors in CSS. All following examples work with any of the above notations. Of course, you need to keep the notations browser support in mind, and if you use one of the new ways add a fallback color.

Text Gradient

The CSS property to change the foreground or text color is `color`. In the examples above I showed you how to set a normal color as text color with the following code.

```
.textsample{
        color:var(--color);
        /* or */
        color:#FF0000;
}
```

Now the cool part! Did you know you can use a CSS gradient as your text background? To get an idea of what is possible in modern browsers have a look at the screenshot of my personal website.

Screenshot of azettl.net

The trick here is to use the CSS properties `background-clip` and `text-fill-color`. You can

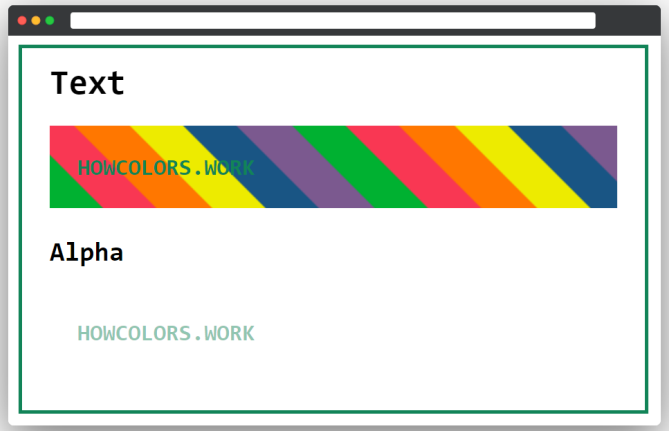
use this if you do not need to support Internet Explorer *(https://caniuse.com/#feat=mdn-css_properties_-webkit-text-fill-color)*.

First, you will have to decide on a CSS gradient you want to use. If you are not sure how to create a gradient you can use any online generator *(https://convertingcolors.com/css-gradient.html)* to create one.

Look at the example code from the previous chapters. If you want to use a gradient as text background you will have to change the `.textsample` class.

Additionally to the `color`, you have to set a `background-image`. This is where you pass your gradient, in this example a `linear-gradient`.

```
1 .textsample{
2         color:var(--color);
3         background-image:linear-
gradient(45deg,#00b131 8.33%,#f93753
8.33%,#f9\
4 3753 16.67%,#f70 16.67%,#f70 25%,#edeb00
25%,#edeb00 33.33%,#195584 33.\
5 33%,#195584 41.67%,#7b598f 41.67%,#7b598f
50%,#00b131 50%,#00b131 58.33\
6 %,#f93753 58.33%,#f93753 66.67%,#f70
66.67%,#f70 75%,#edeb00 75%,#edeb0\
7 0 83.33%,#195584 83.33%,#195584
91.67%,#7b598f 91.67%,#7b598f 100%);
8 }
```

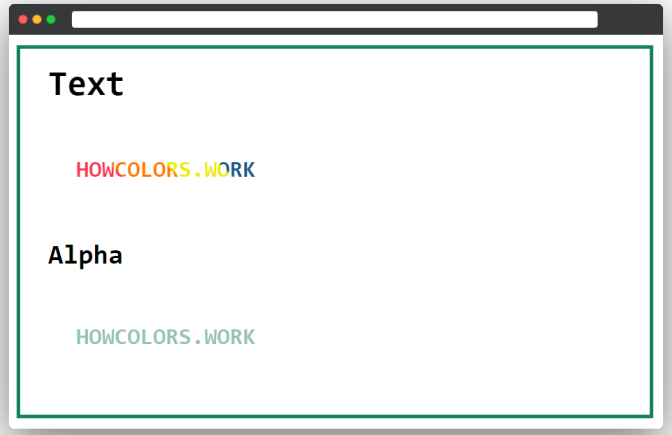


Current Result

Until now you only set the gradient as a background of your element, but the text has still the same color as before. Now you need to add the `background-clip` and `text-fill-color` properties. The `text-fill-color` you have to set to `transparent` and the `background-clip` you set to `text`. With this parameters, the background gradient gets clipped to the text and the text gets a transparent filling.

```
1 .textsample{
2         color:var(--color);
3         background-image:linear-
gradient(45deg,#00b131 8.33%,#f93753
8.33%,#f9\
4 3753 16.67%,#f70 16.67%,#f70 25%,#edeb00
25%,#edeb00 33.33%,#195584 33.\
5 33%,#195584 41.67%,#7b598f
```

```
41.67%,#7b598f 50%,#00b131 50%,#00b131
58.33\
 6 %,#f93753 58.33%,#f93753 66.67%,#f70
66.67%,#f70 75%,#edeb00 75%,#edeb0\
 7 0 83.33%,#195584 83.33%,#195584
91.67%,#7b598f 91.67%,#7b598f 100%);
 8          -webkit-background-clip:text;
 9          -webkit-text-fill-
color:transparent;
10          -moz-background-clip:text;
11          -moz-text-fill-
color:transparent;
12 }
```



The Final Result

With the CSS property `border-image` you can use your gradient as border of an element!

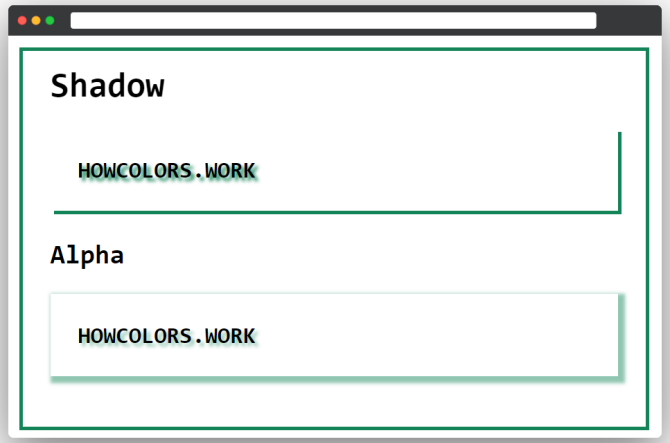
Box Shadows as Borders

Wanting more than one border on an element is one unusual but cool use case for box shadows. In this example, we will use the `.shadowsample` class of the code from the above chapters.

```
.shadowsample{
        text-shadow:4px 4px 4px var(--color);
        /* offset-x | offset-y | blur-radius | spread-radius | color */
        box-shadow:4px 4px 4px 4px var(--color);
}
```

Let's get rid of the blur and spread-radius first.

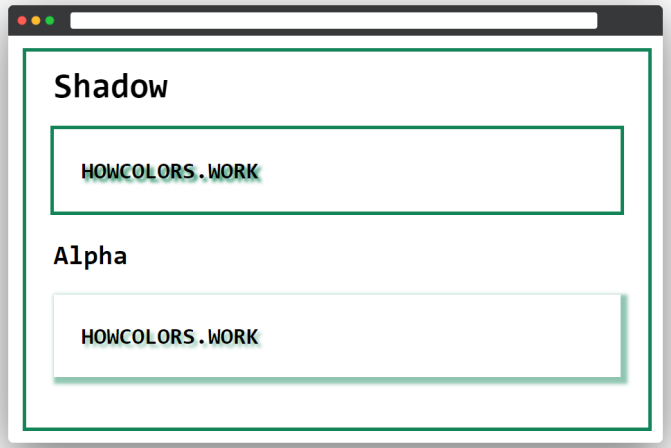
```
.shadowsample{
        text-shadow:4px 4px 4px var(--color);
        /* offset-x | offset-y | blur-radius | spread-radius | color */
        box-shadow:4px 4px 0 0 var(--color);
}
```



The result without blur and spread-radius

The `box-shadow` property allows you to pass multiple shadows by separating them with a comma. This you can use to create a basic border effect by adjusting the offset parameters.

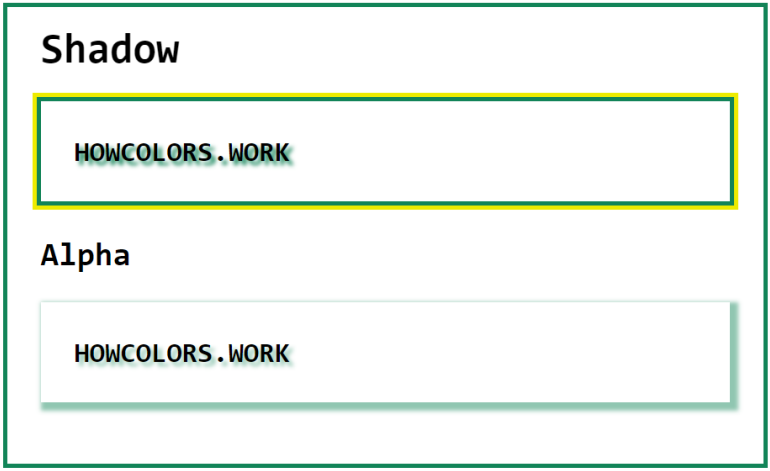
```
.shadowsample{
        text-shadow:4px 4px 4px var(--color);
        /* offset-x | offset-y | blur-radius | spread-radius | color */
        box-shadow:4px 4px 0 0 var(--color), -4px 4px 0 0 var(--color), 4px -4\
px 0 0 var(--color), -4px -4px 0 0 var(--color);
}
```



A single border via the box-shadow property

To add a second border around the first, all you need to do is add another set of box shadows. I use 8 pixels as an offset to get the same border width and the hex color `#edeb00` as border color.

```
1 .shadowsample{
2         text-shadow:4px 4px 4px var(--
color);
3         /* offset-x | offset-y | blur-
radius | spread-radius | color */
4         box-shadow:4px 4px 0 0 var(--
color), -4px 4px 0 0 var(--color), 4px -4\
5 px 0 0 var(--color), -4px -4px 0 0 var(--
color), 8px 8px 0 0 #edeb00, -\
6 8px 8px 0 0 #edeb00, 8px -8px 0 0
#edeb00, -8px -8px 0 0 #edeb00;
7 }
```



Two borders via the box-shadow property

This way you can even add a third, fourth, or any amount of borders to your element.

Thank you!

Answers to Exercises

Current Ways to Express Colors!

1. 4
2. 40%
3. 4096
4. rgb(255 100 100, 100%)
5. Green
6. 0 - 255
7. hsl(300 100 50)
8. Lightness
9. 240 degree

New Ways to Express Colors!

1. hwb(300deg, 50, 50)
2. Hue
3. 120 degree
4. lab(100%, 100, 160)
5. The distance along the a-axises in the Lab colorspace.
6. 0 - 400%
7. lch(50% 50 380deg)
8. Hue
9. 0 - 230
10. color(display-p3, 0.5 0.5 0.2)
11. The ICC color profile
12. Depends on the color space but usually 0 - 1.

13. gray(10, 100%)
14. The value for gray.
15. 0 - 100