howest
hogeschool

Web Technology: Advanced CSS

Dieter Mourisse

Topics

- Pseudo-classes
- Pseudo-elements
- CSS Variables
- CSS Nesting
- Calc
- Colors

howest
hogeschool

Pseudo-classes

Pseudoclasses

A CSS ***pseudo-class*** is a keyword added to a selector that specifies a special state of the selected element(s). For example, :hover can be used to change a button's color when the user's pointer hovers over it.

```
div {
	border: 0.2rem solid red;
}

div:hover {
	background-color: black;
	color: white;
}
```

Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur eveniet ullam ut quis nemo magni quasi omnis doloribus? Enim, quidem.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur eveniet ullam ut quis nemo magni quasi omnis doloribus? Enim, quidem.

https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

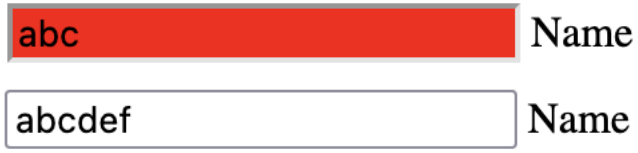
Pseudoclasses

Pseudoclass	Meaning
:hover	Matches when a user designates an item with a pointing device, for example holding the mouse pointer over it.
:focus	Matches when an element has focus.
:link	Matches links that have not yet been visited.
:visited	Matches links that have been visited.
:target	Matches the element which is the target of the document URL.
:checked	Matches when elements such as checkboxes and radiobuttons are toggled on.
:invalid	Matches an element with invalid contents. For example an input element with type 'email' with a name entered.
:heading	Matches all heading elements (h1-h6). Not fully supported yet as of 2025.

https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

Form validation

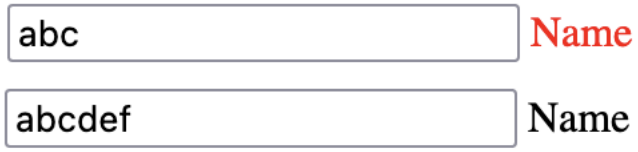
Give inputfield a red background when it’s content is invalid (empty or length is not between 5 and 25 characters).




```
<input id="name" type="text" minlength="5" maxlength="25" required>
<label for="name">Name</label>


input:invalid {
      background-color: red;
}
```

Form validation

Give label a red textcolor when the corresponding inputfield has a red background




```
<input id="name" type="text" minlength="5" maxlength="25" required>
<label for="name">Name</label>
```

```
input:invalid + label {
    color: red;
}
```

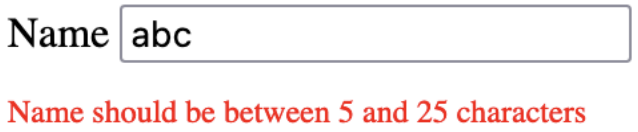
The **adjacent/next sibling combinator** (+) separates two selectors and matches the second element only if it *immediately* follows the first element, and both are children of the same parent element.

Does not work when label comes before input element

howest hogeschool

Form validation

Better to be explicit about what's wrong with the content that fails to validate.




```
<label for="name">Name</label>
<input id="name" type="text" minlength="5" maxlength="25" required>
<p class="invalid-message">Name should be between 5 and 25 characters</p>
```

```
.invalid-message {
        font-size: 0.75rem;
        display: none;
        color: red;
}

input:invalid + .invalid-message {
        display: block;
}
```


howest hogeschool

Tree-structural pseudo-classes

:root	Represents an element that is the root of the document. In HTML this is usually the <html> element.
:empty	Represents an element with no children other than white-space characters.
:nth-child	Uses A*n*+B notation to select elements from a list of sibling elements.
:nth-last-child	Uses A*n*+B notation to select elements from a list of sibling elements, counting backwards from the end of the list.
:first-child	Matches an element that is the first of its siblings.
:last-child	Matches an element that is the last of its siblings.
:only-child	Matches an element that has no siblings. For example a list item with no other list items in that list.
:nth-of-type	Uses A*n*+B notation to select elements from a list of sibling elements that match a certain type from a list of sibling elements.
:nth-last-of-type	Uses A*n*+B notation to select elements from a list of sibling elements that match a certain type from a list of sibling elements counting backwards from the end of the list.
:first-of-type	Matches an element that is the first of its siblings, and also matches a certain type selector.
:last-of-type	Matches an element that is the last of its siblings, and also matches a certain type selector.
:only-of-type	Matches an element that has no siblings of the chosen type selector.

An + B notation

`<An+B>`

Represents elements in a list whose indices match those found in a custom pattern of numbers, defined by `An+B`, where:

- `A` is an integer step size,
- `B` is an integer offset,
- `n` is all nonnegative integers, starting from 0.

It can be read as the *An+B*th element of a list.

An + B notation

`:nth-child(7)`

Represents the seventh element.

`:nth-child(5n)`

Represents elements **5** [=5×1], **10** [=5×2], **15** [=5×3], **etc.** The first one to be returned as a result of the formula is **0** [=5×0], resulting in a no-match, since the elements are indexed from 1, whereas `n` starts from 0. This may seem weird at first, but it makes more sense when the `B` part of the formula is `>0`, like in the next example.

`:nth-child(n+7)`

Represents the seventh and all following elements: **7** [=0+7], **8** [=1+7], **9** [=2+7], **etc.**

`:nth-child(3n+4)`

Represents elements **4** [=(3×0)+4], **7** [=(3×1)+4], **10** [=(3×2)+4], **13** [=(3×3)+4], **etc.**

`:nth-child(-n+3)`

Represents the first three elements. [=-0+3, -1+3, -2+3]

The **:not()** CSS pseudo-class represents elements that do not match a list of selectors. Since it prevents specific items from being selected, it is known as the *negation pseudo-class*.

The **:where()** CSS pseudo-class function takes a selector list as its argument, and selects any element that can be selected by one of the selectors in that list.

The **:is()** CSS pseudo-class function takes a selector list as its argument, and selects any element that can be selected by one of the selectors in that list. This is useful for writing large selectors in a more compact form.

The **:has()** CSS pseudo-class represents an element if any of the selectors passed as parameters (relative to the :scope of the given element) match at least one element.

:not

```
body > :not(nav) {
  margin: 0.5rem;
}
```

All direct children of *body* that are not *nav*

```
p:not(.fancy) {
  color: green;
}
```

All *p* elements that do not have the class *fancy*

```
h2 :not(span.foo) {
  color: red;
}
```

Elements inside a *h2* element that are not *span* elements with the class *foo*

https://developer.mozilla.org/en-US/docs/Web/CSS/:not

:where / :is

```
:is(header, main, footer) p:hover {
        color: red;
        cursor: pointer;
}
```

```
header p:hover,
main p:hover,
footer p:hover {
        color: red;
        cursor: pointer;
}
```

The difference between :where() and :is() is that :where() always has 0 specificity, whereas :is() takes on the specificity of the most specific selector in its arguments.

Prefer :is over :where

https://developer.mozilla.org/en-US/docs/Web/CSS/:where

https://developer.mozilla.org/en-US/docs/Web/CSS/:is

:has

The relational pseudo-class, *:has()*, is a functional pseudo-class taking a <forgiving-relative-selector-list> as an argument. It represents an element if any of the relative selectors would match at least one element when anchored against this element.

https://developer.mozilla.org/en-US/docs/Web/CSS/:has

:has

The css :has selector can be used to select elements that contain elements matching a selector. That's why it is sometimes called the parent selector, but you can do a lot more with it.

Try to minimize the use of *:has*, there are some performance concerns

```
div:has(h2) {
    background-color: blue;
}
```

All *div* elements that contain an *h2* element

```
a:has(> img) {
    background-color: blue;
}
```

All *a* elements that directly contain an *img* element

```
h1:has(+ p) {
    background-color: blue;
}
```

All *h1* elements directly followed by a *p* element

howest hogeschool

howest
hogeschool

Pseudo elements

Pseudo element

A CSS **pseudo-element** is a keyword added to a selector that lets you style a specific part of the selected element(s). For example, ::first-line can be used to change the font of the first line of a paragraph.

Denoted by two colons (*::*).

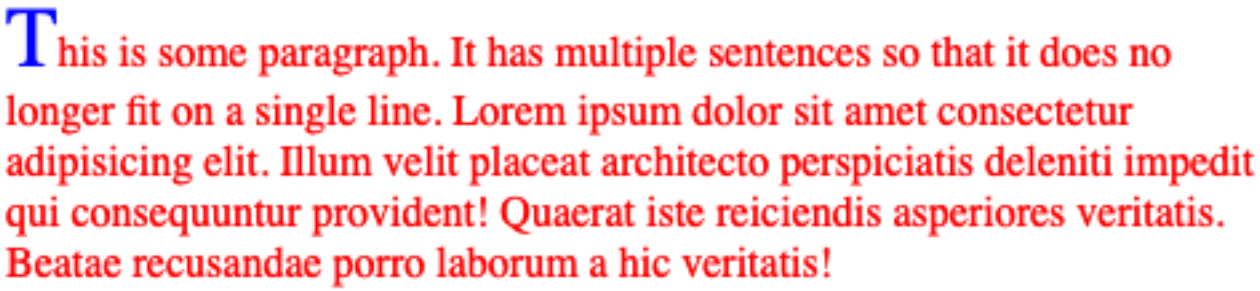
Some can be denoted by a single colon, but this is legacy notation and you should no longer use it.

https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements

::first-letter

::first-letter (:first-letter)
The **::first-letter** CSS pseudo-element applies styles to the first letter of the first line of a block-level element, but only when not preceded by other content (such as images or inline tables).

```
<p>This is some paragraph. It has multiple sentences so that it does no longer fit on a single line. Lorem ipsum dolor sit amet consectetur adipisicing elit. Illum velit placeat architecto perspiciatis deleniti impedit qui consequuntur provident! Quaerat iste reiciendis asperiores veritatis. Beatae recusandae porro laborum a hic veritatis!</p>
```




```
p {
    color: red;
}

p::first-letter {
    color: blue;
    font-size: 2em;
}
```

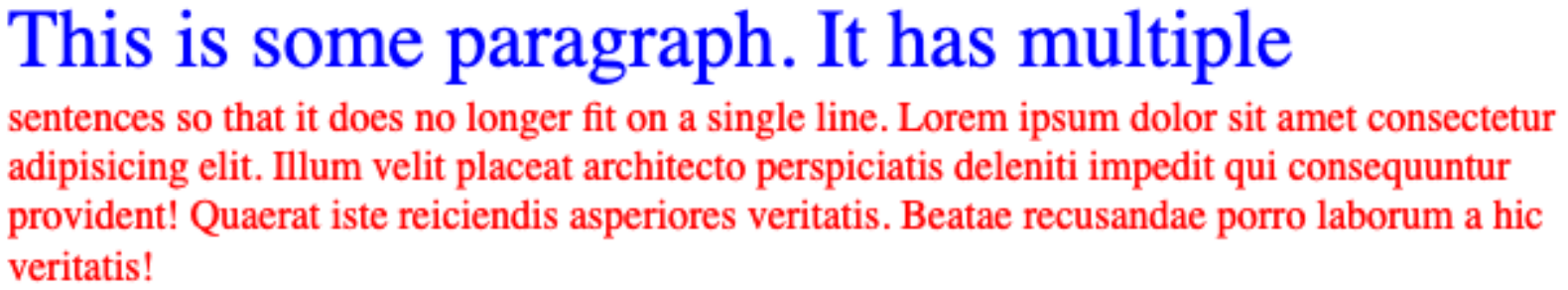
https://developer.mozilla.org/en-US/docs/Web/CSS/::first-letter

::first-line

::first-line (:first-line)

The **::first-line** CSS pseudo-element applies styles to the first line of a block-level element.

```
<p>This is some paragraph. It has multiple sentences so that it does no longer fit on a single line. Lorem ipsum dolor sit amet consectetur adipisicing elit. Illum velit placeat architecto perspiciatis deleniti impedit qui consequuntur provident! Quaerat iste reiciendis asperiores veritatis. Beatae recusandae porro laborum a hic veritatis!</p>
```




```
p {
    color: red;
}

p::first-line {
    color: blue;
    font-size: 2em;
}
```

https://developer.mozilla.org/en-US/docs/Web/CSS/::first-line

::after / ::before

In CSS, **::after** creates a pseudo-element that is the last child of the selected element. It is often used to add cosmetic content to an element with the content property. It is inline by default.

https://developer.mozilla.org/en-US/docs/Web/CSS/::after

In CSS, **::before** creates a pseudo-element that is the first child of the selected element. It is often used to add cosmetic content to an element with the content property. It is inline by default.

https://developer.mozilla.org/en-US/docs/Web/CSS/::before

::after / ::before

```
<p>This is some paragraph. It has multiple sentences so that it does no longer fit on a single line. Lorem ipsum dolor sit amet consectetur adipisicing elit. Illum velit placeat architecto perspiciatis deleniti impedit qui consequuntur provident! Quaerat iste reiciendis asperiores veritatis. Beatae recusandae porro laborum a hic veritatis!</p>
```

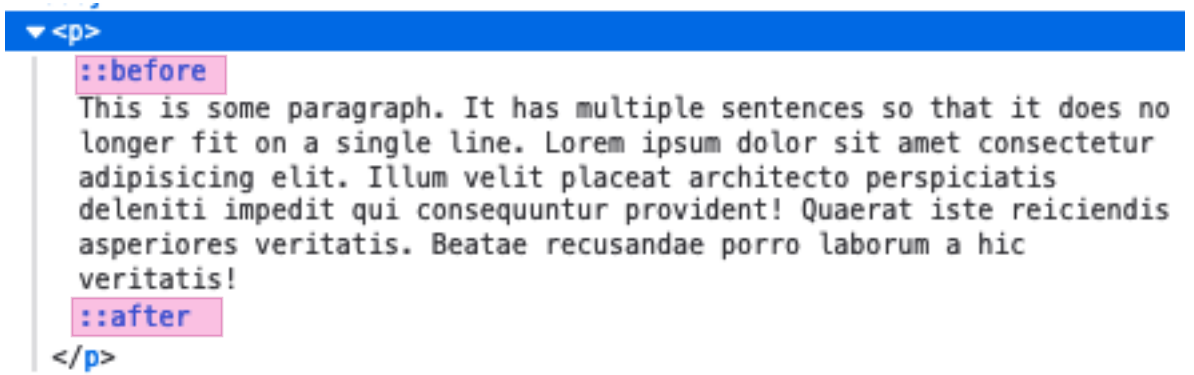
```
p::before {
    content: "before";
}

p::after {
    content: "after";
}
```

beforeThis is some paragraph. It has multiple sentences so that it does no longer fit on a single line. Lorem ipsum dolor sit amet consectetur adipisicing elit. Illum velit placeat architecto perspiciatis deleniti impedit qui consequuntur provident! Quaerat iste reiciendis asperiores veritatis. Beatae recusandae porro laborum a hic veritatis!after

::after / ::before

```css
p::before {
    content: "";
    width: 0.6rem;
    height: 0.6rem;
    display: inline-block;
    background-color: red;
}

p::after {
    content: "";
    width: 0.6rem;
    height: 0.6rem;
    display: inline-block;
    background-color: blue;
}
```

■This is some paragraph. It has multiple sentences so that it does no longer fit on a single line. Lorem ipsum dolor sit amet consectetur adipisicing elit. Illum velit placeat architecto perspiciatis deleniti impedit qui consequuntur provident! Quaerat iste reiciendis asperiores veritatis. Beatae recusandae porro laborum a hic veritatis!■



Part of the *p*-element. So CSS that is applied to the *p*-element is also applied to ::before/::after.

::after / ::before

```
p::before {
    content: "";
    width: 0.6rem;
    height: 0.6rem;
    display: inline-block;
    background-color: red;
}
```

Mandatory, without the *content* property, no element will be created.

Setting the dimension

By default, pseudo-elements are *inline*. Those can't be given a height/width

howest hogeschool

Universal selector (*)

The CSS **universal selector** (*) matches elements of any type.

There are some performance concerns with the universal selector since it iterates over all existing html tags, and not just those that occur in your html.

```
body > * {
   background-color: red;
}
```

gets rewritten to

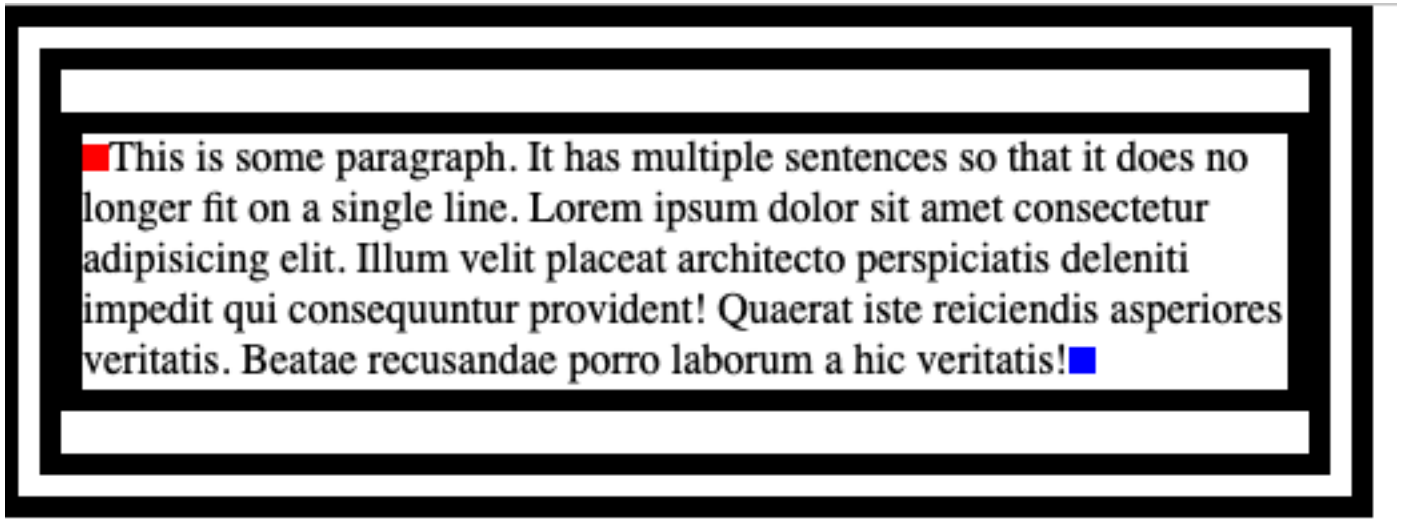
```
body > a {
   background-color: red;
}

body > address {
   background-color: red;
}

body > article {
   background-color: red;
}
. . .
```

https://developer.mozilla.org/en-US/docs/Web/CSS/Universal_selectors

Universal selector with ::before and ::after

The universal selector (*) will not select the ::before and ::after elements.

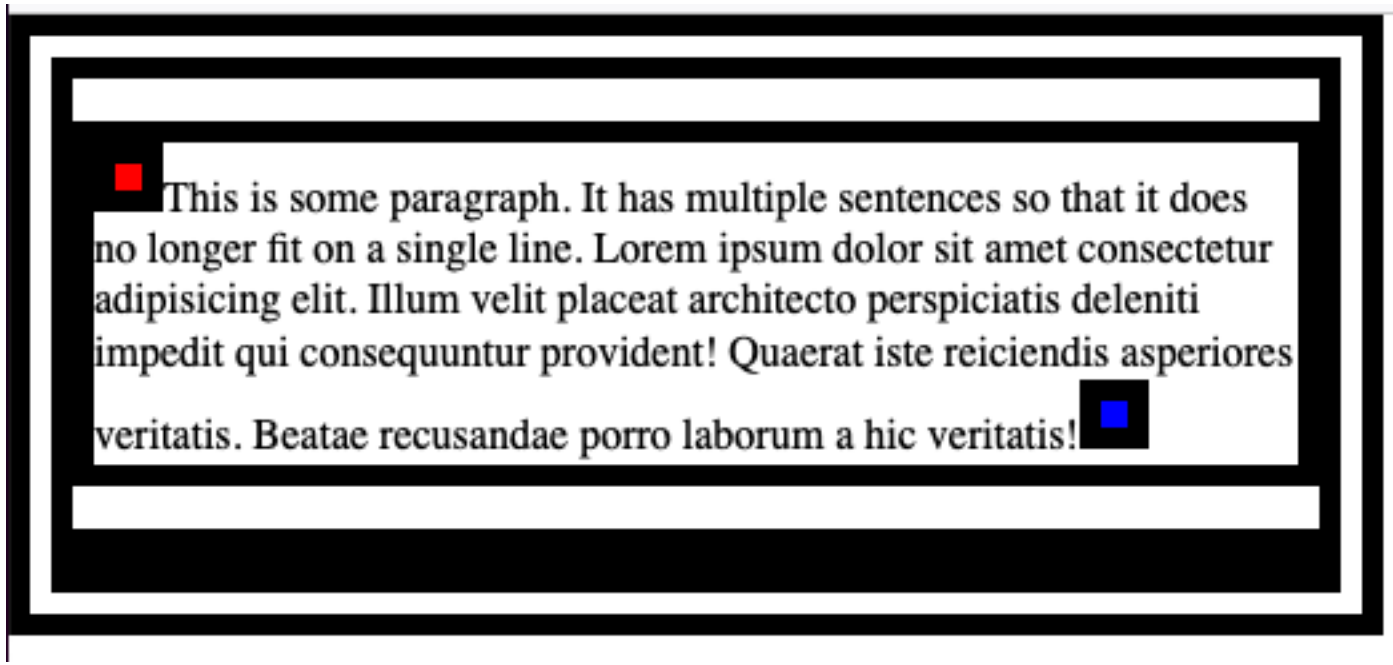
```
* {
  border: 0.5rem solid black;
}
```



Universal selector with ::before and ::after

The universal selector (*) will not select the ::before and ::after elements.

```
*,
*::after,
*::before {
    border: 0.5rem solid black;
}
```



Use case

```
*,
*::after,
*::before {
    box-sizing: border-box;
}
```

box-sizing

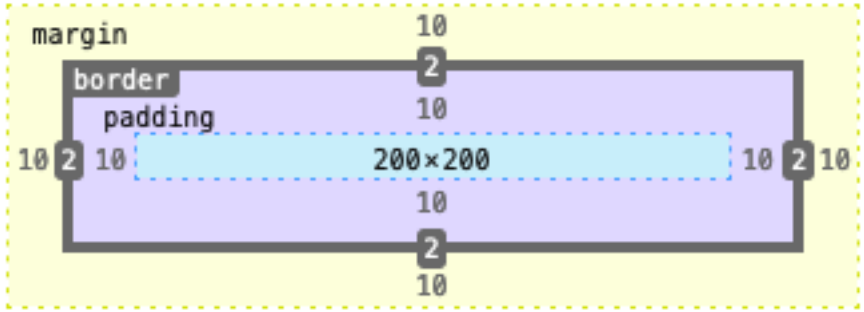
The **box-sizing** CSS property sets how the total width and height of an element is calculated.

- content-box gives you the default CSS box-sizing behavior. If you set an element's width to 100 pixels, then the element's content box will be 100 pixels wide, and the width of any border or padding will be added to the final rendered width, making the element wider than 100px.
- border-box tells the browser to account for any border and padding in the values you specify for an element's width and height.

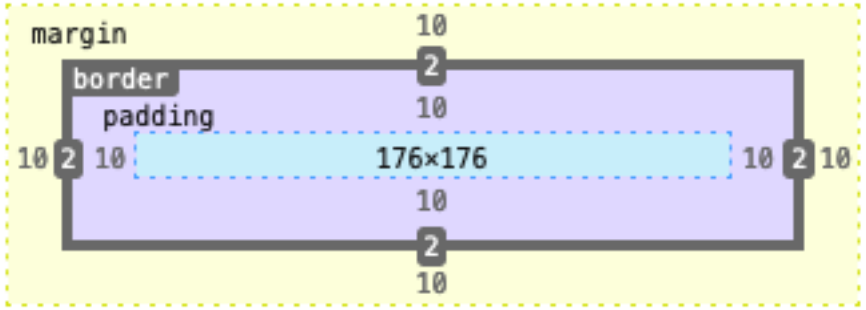
https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing

Use case

```
div {
    width: 200px;
    height: 200px;
    padding: 10px;
    margin: 10px;
    border: 2px solid black;
    background-color: red;
}
```



content-box



border-box

howest
hogeschool

CSS Variables

CSS Variables / Custom properties

Custom properties (--*): CSS variables
Property names that are prefixed with --, like --example-name, represent *custom properties* that contain a value that can be used in other declarations using the var() function.

```
:root {
    --main-text-color: rgb(96 96 96);
    --font-stack: Arial, Helvetica, sans-serif;
}

body {
    font-family: var(--font-stack);
    color: var(--main-text-color);
}
```

Define variables (use dash dash syntax)

Reference them using var keyword

https://developer.mozilla.org/en-US/docs/Web/CSS/--*

Scope of CSS variables

CSS variables inherit to their child elements

```
#div1 {
    --success: green;
}
```

```
#div2 {
    --success: orange;
}
```

```
p.success {
    color: var(--success)
}
```

```
<div id="div1">
    <p class="success">Succes for task 1!</p>
</div>
```

```
<div id="div2">
    <p class="success">Success for task2!</p>
</div>
```

Succes for task 1!

Success for task2!

The root pseudo class

:root matches the root of the HTML tree.

These variables will be accessible everywhere

```
:root {
    --main-text-color: rgb(96 96 96);
    --font-stack: Arial, Helvetica, sans-serif;
}
```

```
body {
    font-family: var(--font-stack);
    color: var(--main-text-color);
}
```

Defining color schemes

```
body.normal-color-scheme {
   --primary-color: #b32c1a;
   --primary-color-light: #ef4232;
   --neutral-color: #ffffff;
   --complement-color: #135659;
   --complement-color-light: #1cb1c5;
}

body.alternative-color-scheme {
   --complement-color: #b32c1a;
   --complement-color-light: #ef4232;
   --neutral-color: #ffffff;
   --primary-color: #135659;
   --primary-color-light: #1cb1c5;
}
```

```
<body class="normal-color-scheme">
   ...
</body>
```

Use normal color scheme

```
<body class=”alternative-color-scheme">
   ...
</body>
```

Use alternative color scheme

Don’t hardcode your colors in your css, always use the variable (*var(--…)*)

Prefers-color scheme

prefers-color-scheme
The **prefers-color-scheme** CSS media feature is used to detect if the user has requested a light or dark color theme.
The user might indicate this preference through an operating system setting (e.g. light or dark mode) or a user agent setting.

light
Indicates that user has notified that they prefer an interface that has a light theme, or has not expressed an active preference.

dark
Indicates that user has notified that they prefer an interface that has a dark theme.

https://developer.mozilla.org/en-US/docs/Web/CSS/@media/prefers-color-scheme

Prefers-color scheme

```
@media (prefers-color-scheme: dark) {
   :root {
      --primary-color: #b32c1a;
      --primary-color-light: #ef4232;
      --neutral-color: #fff;
      --complement-color: #135659;
      --complement-color-light: #1cb1c5;
    }
}
```

Applied when user's preference is dark (setting in os/browser)

```
@media (prefers-color-scheme: light) {
   :root {
      --primary-color: #b32c1a;
      --primary-color-light: #ef4232;
      --neutral-color: #fff;
      --complement-color: #135659;
      --complement-color-light: #1cb1c5;
    }
}
```

Applied when user's preference is light (setting in os/browser)

https://developer.mozilla.org/en-US/docs/Web/CSS/@media/prefers-color-scheme

@property

The **@property** CSS allows developers to explicitly define their CSS custom properties, allowing for property type checking and constraining, setting default values, and defining whether a custom property can inherit values or not.

```css
@property --property-name {
  syntax: "<color>";
  inherits: false;
  initial-value: #c0ffee;
}
```

type of value

is **value of property** inherited

what is the value when not specified

https://developer.mozilla.org/en-US/docs/Web/CSS/@property

howest hogeschool

howest
hogeschool

CSS Nesting

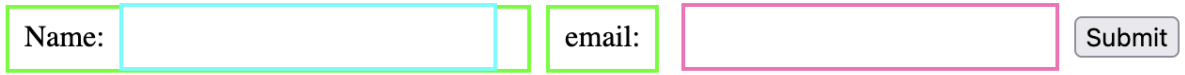
CSS nesting

The **CSS nesting** module defines a syntax for nesting selectors, providing the ability to nest one style rule inside another, with the selector of the child rule relative to the selector of the parent rule.


```
<form>

 <label>Name:
  <input type="text" id="name">
 </label>

 <label for="email">email:</label>
 <input type="text" id="email">

 <button type="submit">Submit</button>
</form>
```

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_nesting

CSS nesting




```
<form>

 <label>Name:
   <input type="text" id="name">
 </label>

  <label for="email">email:</label>
  <input type="text" id="email">

  <button type="submit">Submit</button>
</form>
```

```
input {
  border: 0.2rem solid hotpink;
}

label {
  border: 0.2rem solid lime;

  input {
    border: 0.2rem solid aqua;
  }
}
```

Styles for all *input* elements

Styles for all *label* elements

Styles for all *input* elements inside a *label*

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_nesting

howest hogeschool

Descendant selector

```
input {
  border: 0.2rem solid hotpink;
}

label {
  border: 0.2rem solid lime;

  input {
    border: 0.2rem solid aqua;
  }
}
```

```
input {
  border: 0.2rem solid hotpink;
}

label {
  border: 0.2rem solid lime;
}

label input {
  border: 0.2rem solid aqua;
}
```

Combinators

Combinator	Denoted by
Descendant	a single space (“ “)
Child	>
Next-sibling	+
Subsequent-sibling	~

```
p {
  color: lime;

  ~img {
    border: 0.2rem solid hotpink;
  }
}
```

```
p {
  color: lime;
}

p ~ img {
  border: 0.2rem solid hotpink;
}
```

Compound selector

A **compound selector** is a sequence of simple selectors that are not separated by a combinator. A compound selector represents a set of simultaneous conditions on a single element. A given element is said to match a compound selector when the element matches all the simple selectors in the compound selector.

```
a#selected {
  border: 0.2rem solid hotpink;
}
```

```
a {
  #selected {
    border: 0.2rem solid hotpink;
  }
}
```

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_selectors/Selector_structure#compound_selector

Compound selector

A **compound selector** is a sequence of simple selectors that are not separated by a combinator. A compound selector represents a set of simultaneous conditions on a single element. A given element is said to match a compound selector when the element matches all the simple selectors in the compound selector.

```
a #selected {
  border: 0.2rem solid hotpink;
}
```

```
a {
  #selected {
    border: 0.2rem solid hotpink;
  }
}
```

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_selectors/Selector_structure#compound_selector

& nesting selector

The CSS **& nesting selector** explicitly states the relationship between parent and child rules when using CSS nesting. It makes the nested child rule selectors *relative to the parent element*. Without the & nesting selector, the child rule selector selects child elements. The child rule selectors have the same specificity weight as if they were within :is().

```
a {
  &#selected {
    border: 0.2rem solid hotpink;
  }
}
```

```
a#selected {
  border: 0.2rem solid hotpink;
}
```

https://developer.mozilla.org/en-US/docs/Web/CSS/Nesting_selector

& nesting selector

```
a {
  &#selected {
    border: 0.2rem solid hotpink;
  }
}
```

```
a#selected {
  border: 0.2rem solid hotpink;
}
```

```
a {
  & #selected {
    border: 0.2rem solid hotpink;
  }
}
```

```
a #selected {
  border: 0.2rem solid hotpink;
}
```

Appending the & nesting selector (not recommended)

```
a {
  border: 0.2rem solid lime;

  #selected & {
    border: 0.2rem solid hotpink;
  }
}
```

```
a {
  border: 0.2rem solid lime;
}

#selected a {
  border: 0.2rem solid hotpink;
}
```

howest
hogeschool

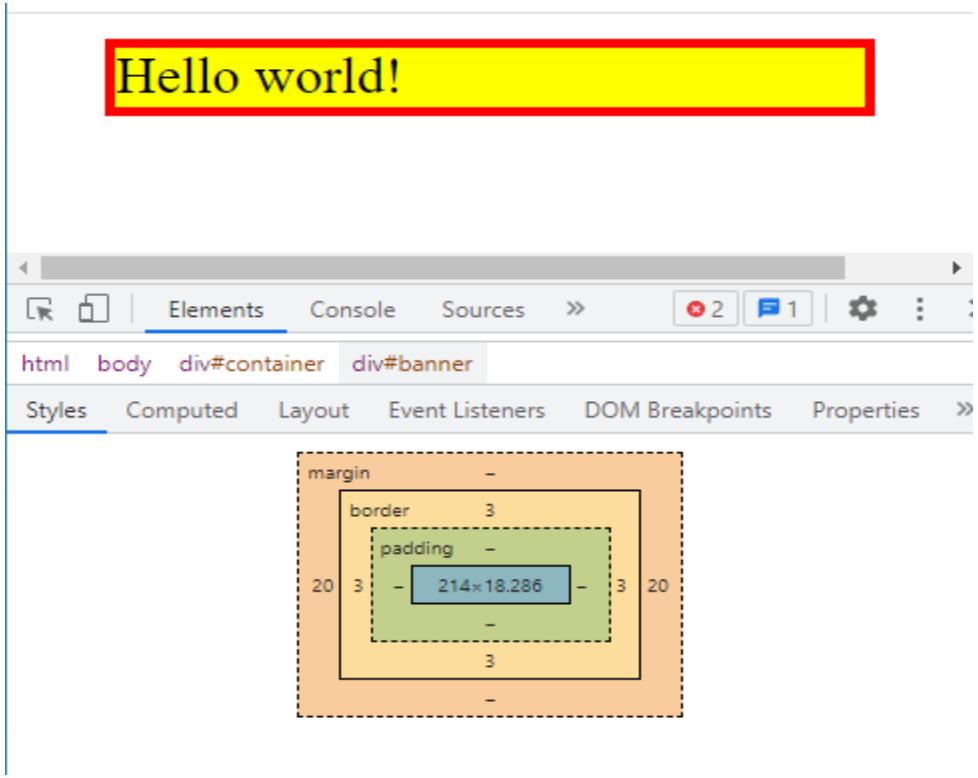
Calc

CSS calc

calc can be used to performed calculations in your CSS

```
#banner {
    width: calc(100% - 40px);
    margin: auto;
    border: 3px solid red;
    background-color: yellow;

    box-sizing: border-box;
}
```

```
<div id="container">
    <div id="banner">
        Hello world!
    </div>
</div>
```



howest hogeschool

CSS calc: some attention points

- Allowed operators: + - * /

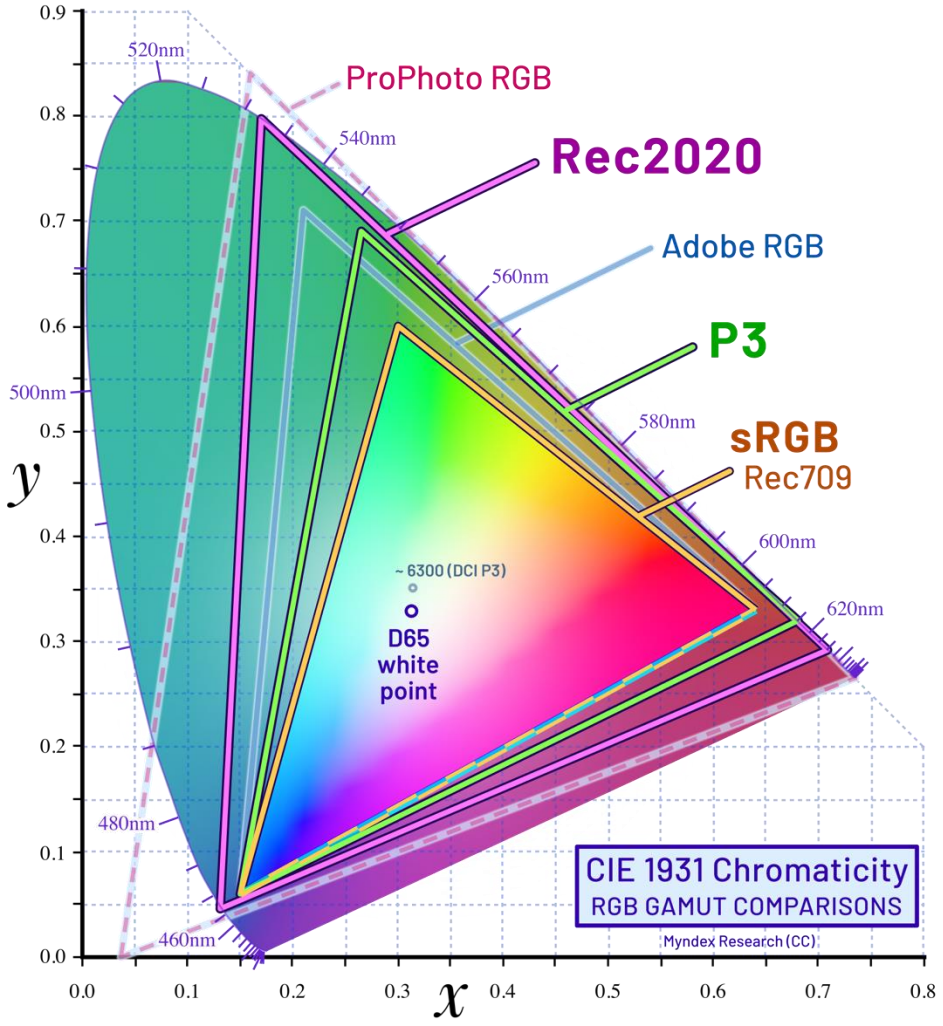
- Include whitespace between the operands and operator to avoid incorrect interpretations
  - calc(50%-8px) Interpreted as 50% followed by a negative length (**invalid**)
  - calc(50% - 8px) Interpreted as 50% followed by a subtraction and length (**valid**)

- Avoid division by zero

howest
hogeschool

Colors

Color space



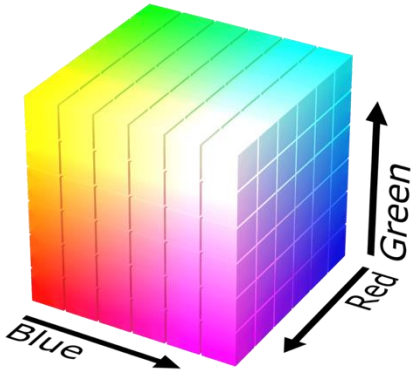
A color space is a specific organization of colors. Some color space have more colors than others.

CSS color models

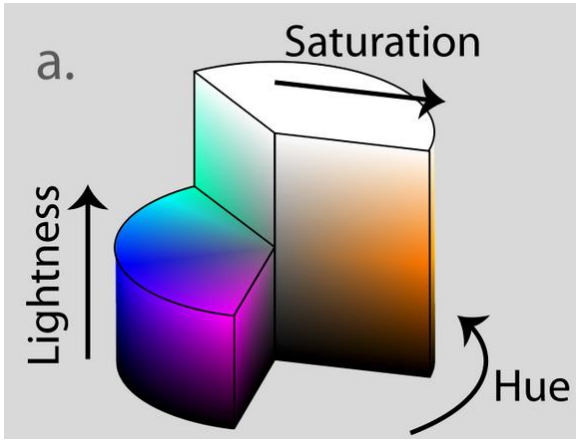
A color model is a way to represent colors in a color space.

Color model	Color space
Keyword (blue, red, …)	sRGB
rgb(), rgba()	sRGB
hsl(), hsla()	sRGB
hwb()	sRGB
oklch()	Display P3
oklab()	Display P3

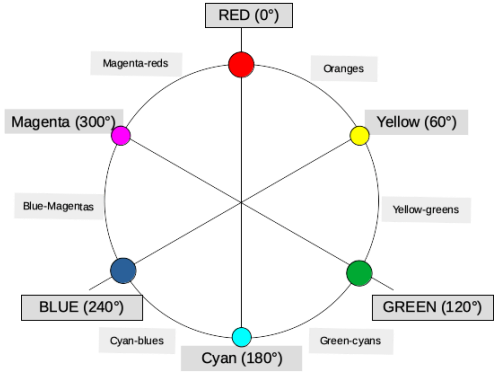
RGB vs HSL (sRGB colors)



A color consists of a red, green and blue component

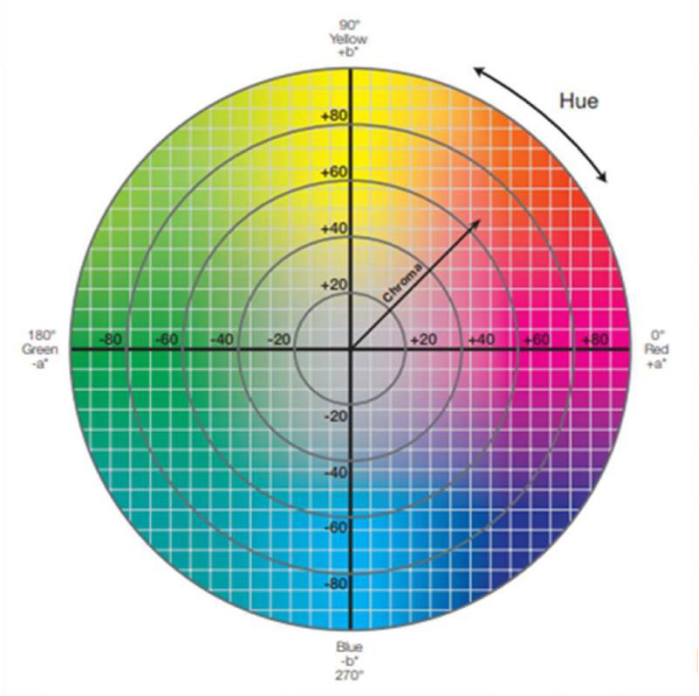


A color consists of a hue (angle), saturation and lightness



hsl is usefull if you need lighter and darker variants of the same color

OKLAB vs OKLCH (Oklab colors)



oklab: (Lightness, distance along a access, distance along b access)

https://developer.mozilla.org/en-US/docs/Web/CSS/color_value/oklab

oklch: (Lightness, chroma, hue)

https://developer.mozilla.org/en-US/docs/Web/CSS/color_value/oklch

More info:

https://www.smashingmagazine.com/2023/08/oklch-color-spaces-gamuts-css/

https://oklch.com/

Relative colors

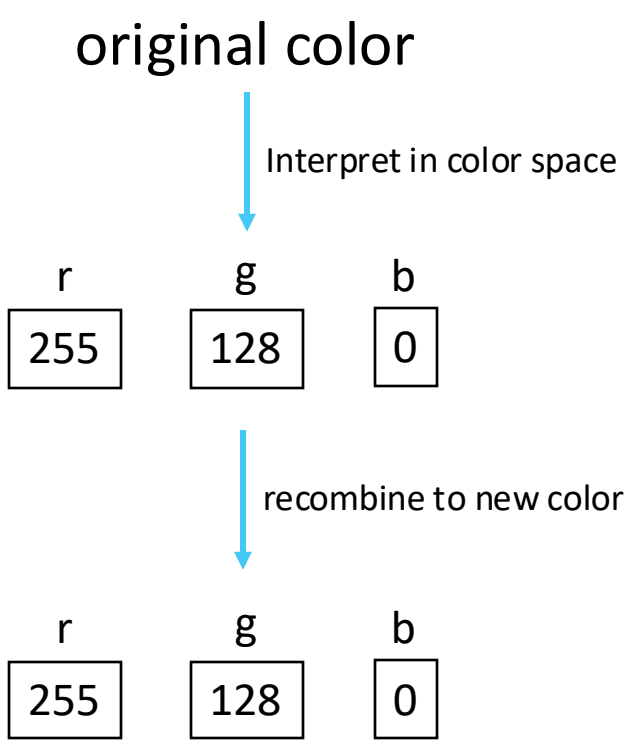
The CSS colors module defines **relative color syntax**, which allows a CSS <color> value to be defined relative to another color. This is a powerful feature that enables easy creation of complements to existing colors — such as lighter, darker, saturated, semi-transparent, or inverted variants — enabling more effective color palette creation.

```
:root {
  --primary-color: rgb(255 128 0);
}

#first {
  background-color: rgb(from var(--primary-color) r g b);
}
```

space — original color — new color

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_colors/Relative_colors

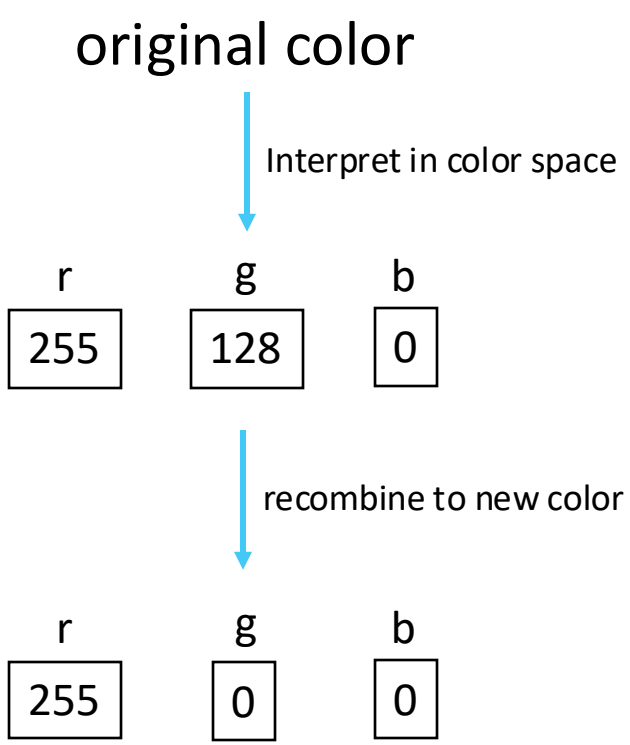
rgb(from var(--primary-color) r g b);



var(--primary-color)

rgb

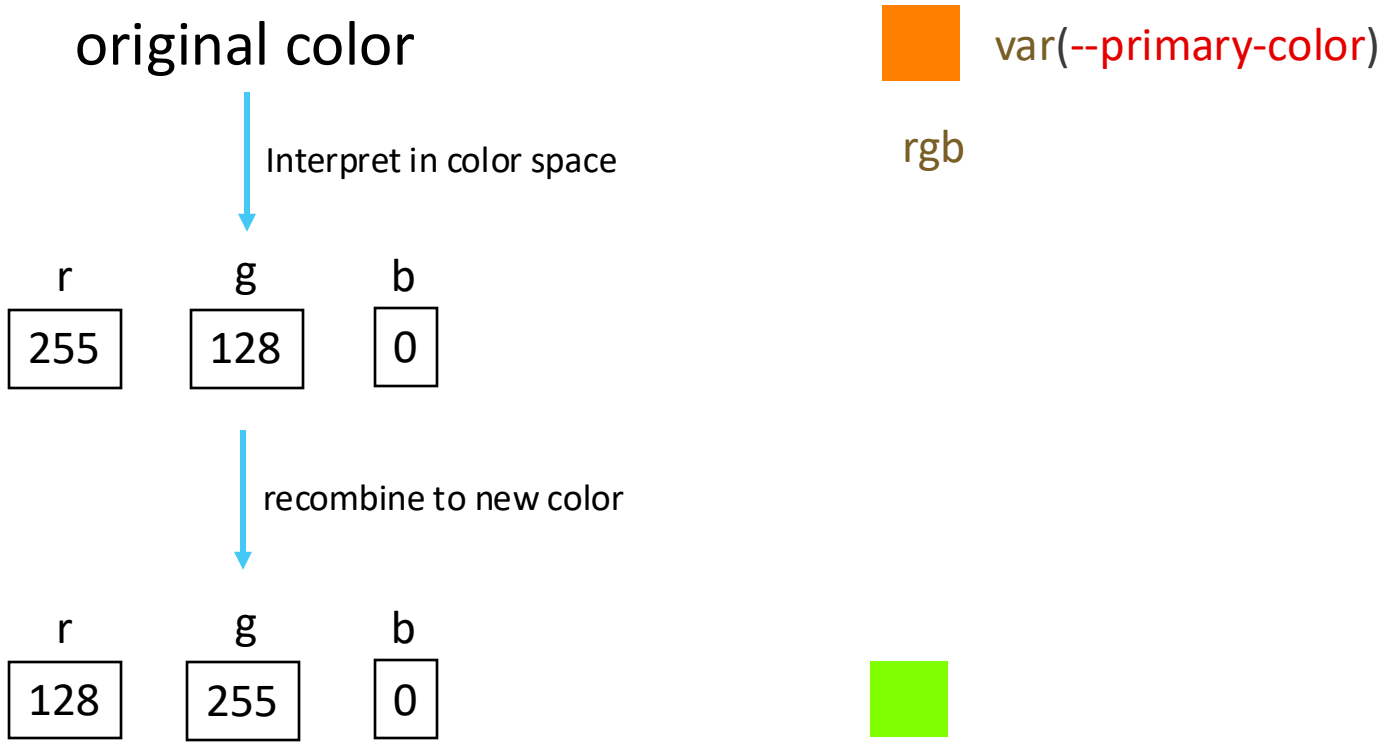
rgb(from var(--primary-color) r 0 b);



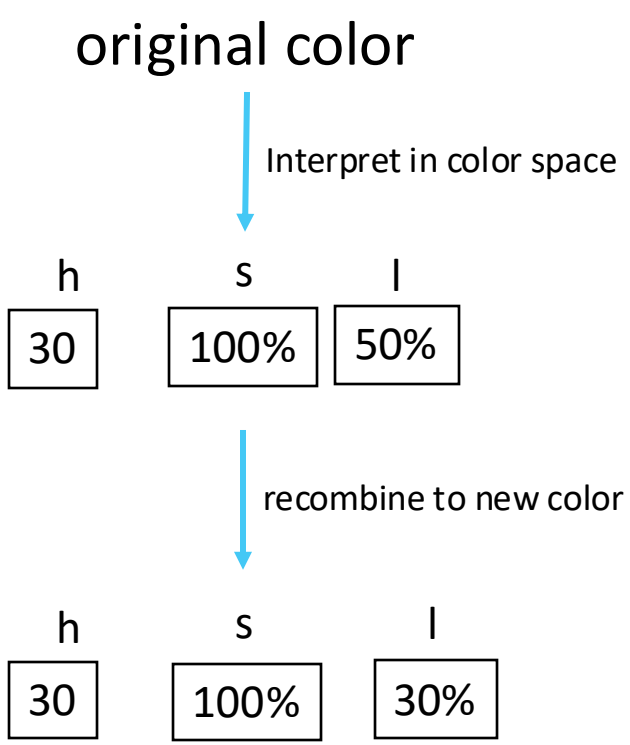
var(--primary-color)

rgb

rgb(from var(--primary-color) g r b);



hsl(from var(--primary-color) h s 30%);



var(--primary-color)

hsl

Use cases

```
rgb(from var(--primary-color) r g b / 70%);
```

Make color transparent

```
hsl(from var(--primary-color) h s calc( 0.6 * l));
```

Make color darker