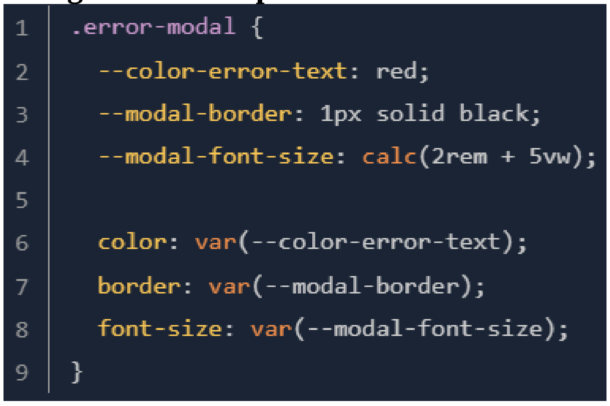
CUSTOM PROPERTIES

Custom properties(also known as CSS Variables) are defined entities to be reused throughout a document. They are useful for efficienty update specific value (the ease of changing multiple instances of a value at once), keep colors consistent, creating themes, and improving the reusability of CSS code.

A. Using Costum Properties



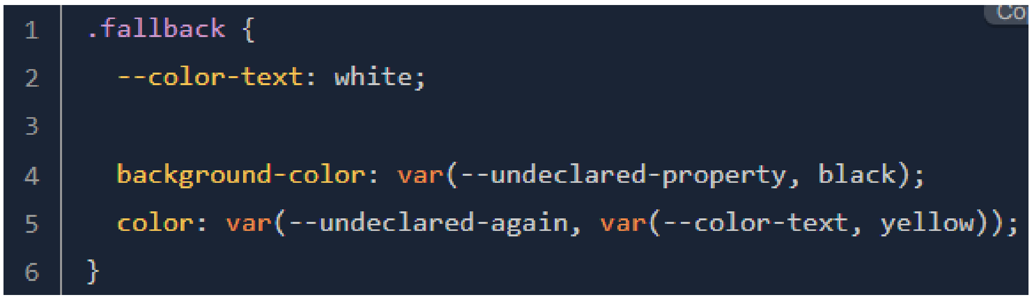
```css
.error-modal {
  --color-error-text: red;
  --modal-border: 1px solid black;
  --modal-font-size: calc(2rem + 5vw);

  color: var(--color-error-text);
  border: var(--modal-border);
  font-size: var(--modal-font-size);
}
```

The first 3rd rows we declare custom properties

The last 3rd rows we access custom properties

B. Fallback values

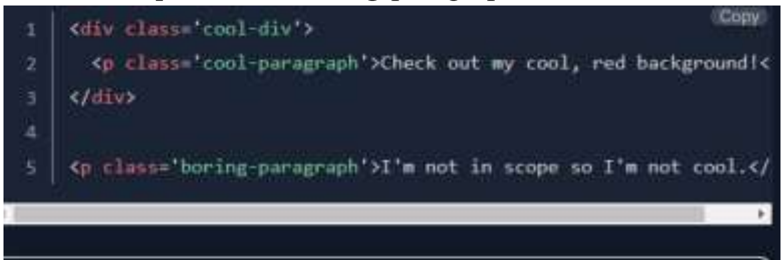
The var() function actually accepts two parameters. The second parameter is an optional fallback value.



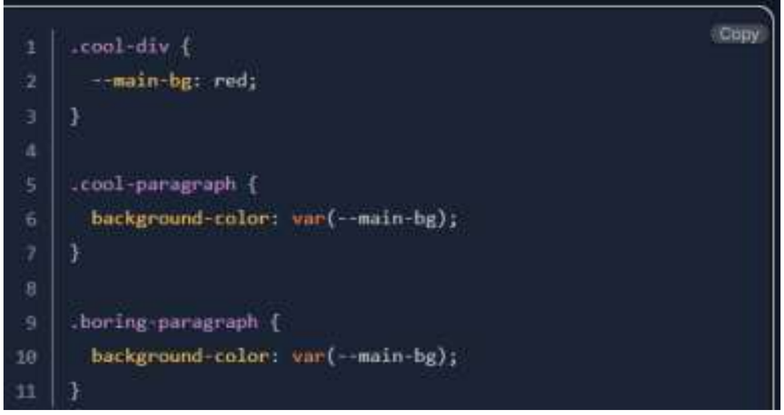
```css
.fallback {
  --color-text: white;

  background-color: var(--undeclared-property, black);
  color: var(--undeclared-again, var(--color-text, yellow));
}
```

C. Scope

The scope of a custom property is determined by the selector. This scope includes the selector the custom property was declared for as well as any descendants of that selector. In the example below, boring-paragraph won’t have red background:

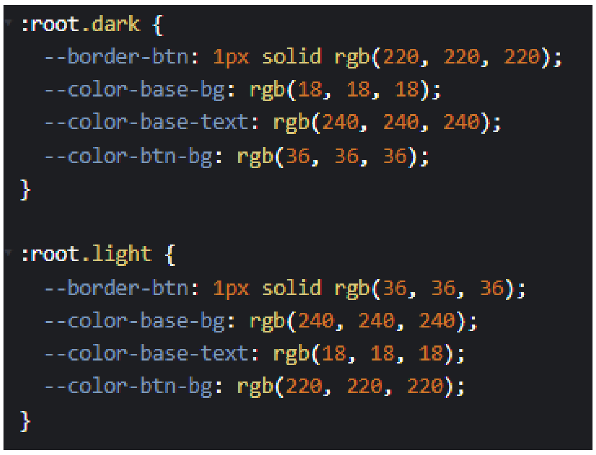


```html
<div class='cool-div'>
  <p class='cool-paragraph'>Check out my cool, red background!<
</div>

<p class='boring-paragraph'>I'm not in scope so I'm not cool.</
```



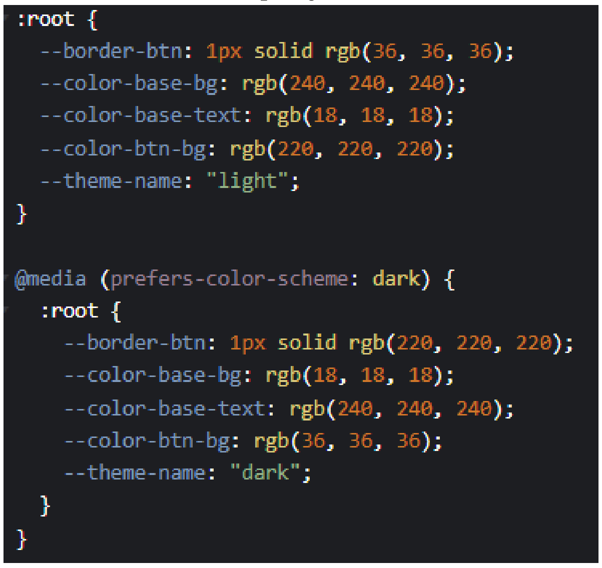
```css
.cool-div {
  --main-bg: red;
}

.cool-paragraph {
  background-color: var(--main-bg);
}

.boring-paragraph {
  background-color: var(--main-bg);
}
```

1) The :root selector
   By declaring our custom property on the ‘:root’ selector, we can access it on any other selector within our CSS file, since any other selector is a descendant of the :root selector.
   A. Theme with :root
      The ' :root ' selector gives us one way to add themes to our pages:



```
:root.dark {
  --border-btn: 1px solid rgb(220, 220, 220);
  --color-base-bg: rgb(18, 18, 18);
  --color-base-text: rgb(240, 240, 240);
  --color-btn-bg: rgb(36, 36, 36);
}

:root.light {
  --border-btn: 1px solid rgb(36, 36, 36);
  --color-base-bg: rgb(240, 240, 240);
  --color-base-text: rgb(18, 18, 18);
  --color-btn-bg: rgb(220, 220, 220);
}
```

      Ketika elemen root (<html> atau <body>) memiliki kelas dark, maka nilai-nilai custom property untuk tema gelap akan digunakan. Begitu juga sebaliknya untuk kelas light.
   B. Theme with media query



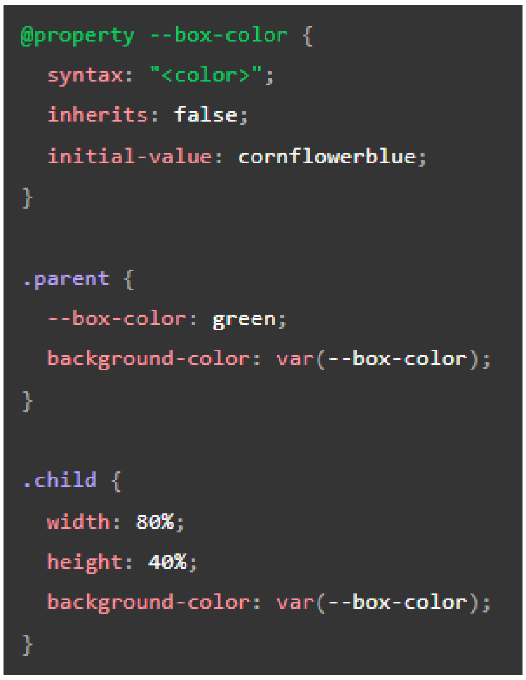
```
:root {
  --border-btn: 1px solid rgb(36, 36, 36);
  --color-base-bg: rgb(240, 240, 240);
  --color-base-text: rgb(18, 18, 18);
  --color-btn-bg: rgb(220, 220, 220);
  --theme-name: "light";
}

@media (prefers-color-scheme: dark) {
  :root {
    --border-btn: 1px solid rgb(220, 220, 220);
    --color-base-bg: rgb(18, 18, 18);
    --color-base-text: rgb(240, 240, 240);
    --color-btn-bg: rgb(36, 36, 36);
    --theme-name: "dark";
  }
}
```

      @media (prefers-color-scheme:) dapat digunakan untuk mendeteksi preferensi tema pengguna di sistem operasi atau browser mereka, sehingga tema website akan sama dengan tema sistem operasi/browser.
      - Media query membuat user tidak bisa mengubah tema web secara langsung
      - Hanya 'dark' & 'light'(default) yang merupakan valid values untuk media query

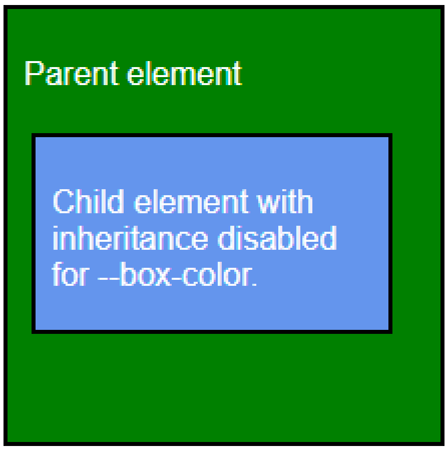
D. Additional Note

a) Inheritance of custom properties
   var(--variable-name) defaultly always inherits the value of its parent. For example, a parent has child1 & child2, if the parent's background is green, child1's background is blue, child2's background is notset, then child2's background will be green (following the parent's background).

b) @property
   @property lets you explicitly state whether the property inherits or not.



```css
@property --box-color {
  syntax: "<color>";
  inherits: false;
  initial-value: cornflowerblue;
}

.parent {
  --box-color: green;
  background-color: var(--box-color);
}

.child {
  width: 80%;
  height: 40%;
  background-color: var(--box-color);
}
```



Because “ inherits: false; “ and a value for the --box-color property is not declared within the .child scope, the initial value of ‘cornflowerblue’ is used instead of green that would have been inherited from the parent.

In the example above we also know how to use fallback value in @property.

c) Invalid custom properties

When value of a property is invalid(e.g. color: 16px) then the initial or inherited value of the property is used. https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties#invalid_custom_properties