{less}

OUR TEAM

Raquel

Oscar

Ronald

¿PREPROCESSOR?

- It is a tool that allows you to write CSS pseudocode to later be compiled
- This CSS pseudocode has some common elements of any programming language
- It could be said that with a preprocessor we have a programming language whose mission is to generate CSS code.

{less}

- Less (Leaner Style Sheets) is a backwards-compatible language extension for CSS.
- Written entirely with javascript
- Syntax quite similar to CSS
- Very popular, used on twitter

FEATURES

VARIABLES

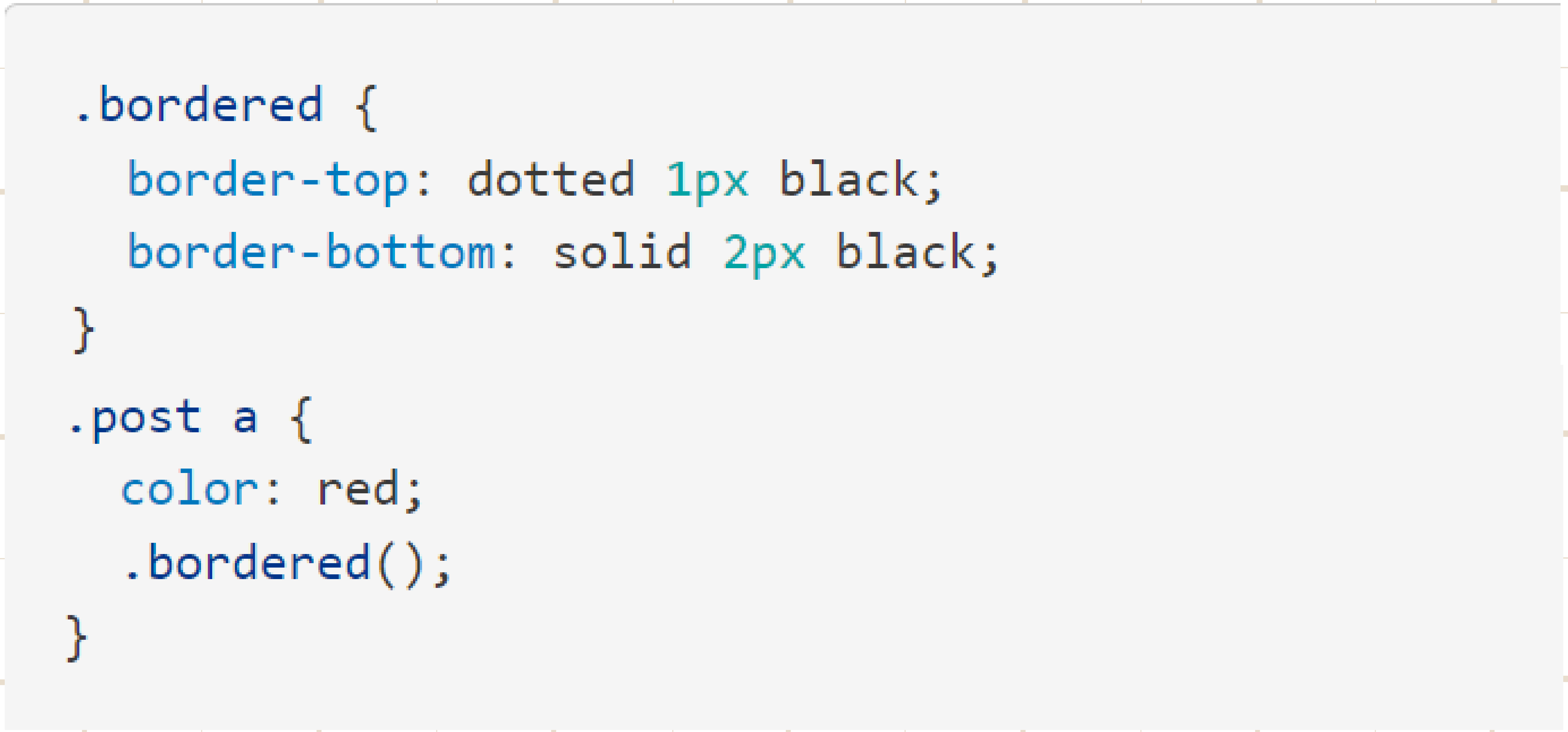
Make your code easier to maintain by giving you a way to control those values from a single location

```
@width: 10px;
@height: @width + 10px;

#header {
  width: @width;
  height: @height;
}
```

```
#header {
  width: 10px;
  height: 20px;
}
```

MIXINS

A way of including ("mixing in") a bunch of properties from one rule-set into another rule-set



```
.bordered {
  border-top: dotted 1px black;
  border-bottom: solid 2px black;
}
.post a {
  color: red;
  .bordered();
}
```

NESTING

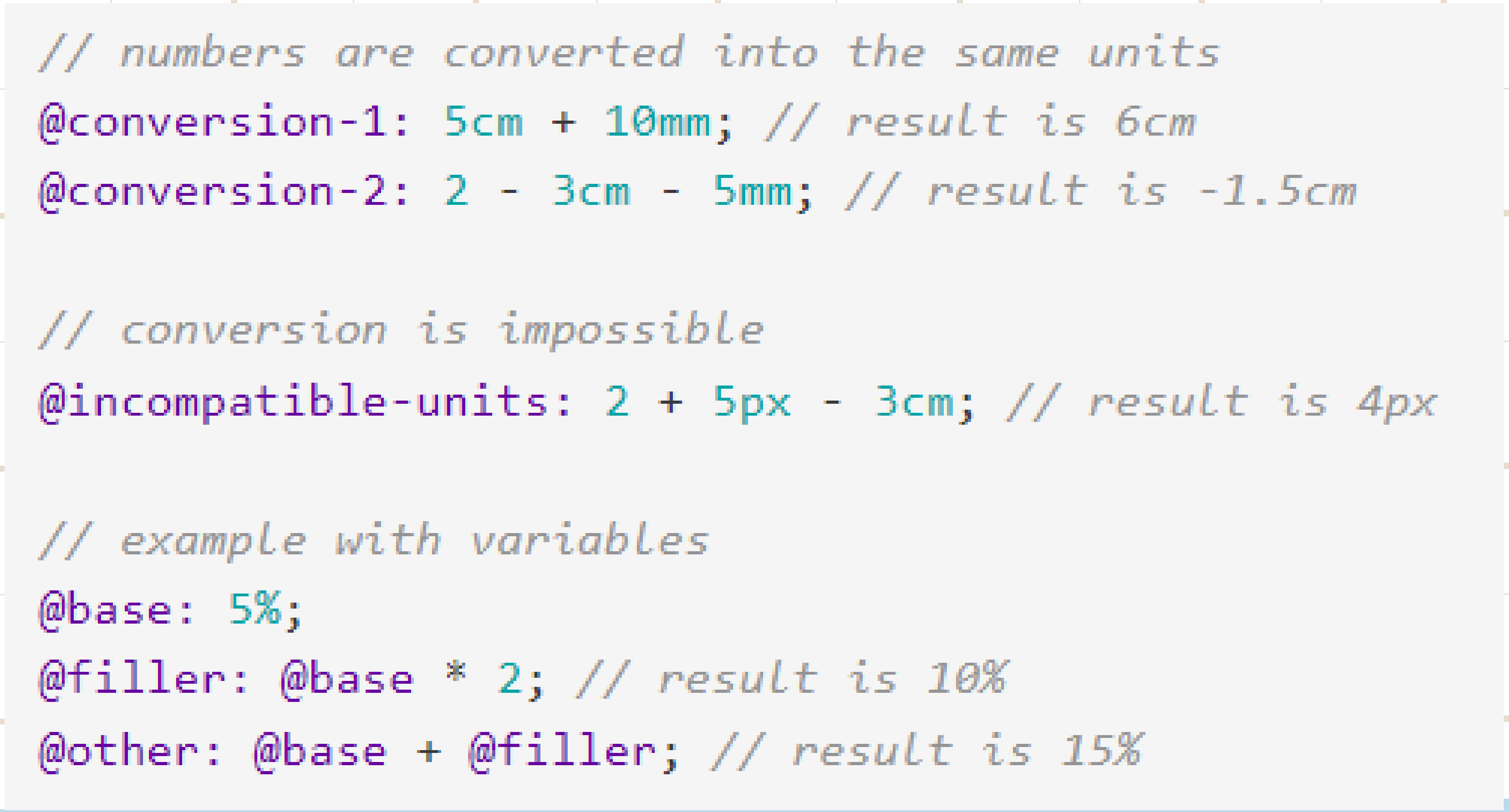
Less gives you the ability to use nesting instead of, or in combination with cascading

```less
#header {
  color: black;
  .navigation {
    font-size: 12px;
  }
  .logo {
    width: 300px;
  }
}
```

```css
#header {
  color: black;
}
#header .navigation {
  font-size: 12px;
}
#header .logo {
  width: 300px;
}
```

OPERATIONS

Arithmetical operations +, -, *, / can operate on any number, color or variable.



```
// numbers are converted into the same units
@conversion-1: 5cm + 10mm; // result is 6cm
@conversion-2: 2 - 3cm - 5mm; // result is -1.5cm

// conversion is impossible
@incompatible-units: 2 + 5px - 3cm; // result is 4px

// example with variables
@base: 5%;
@filler: @base * 2; // result is 10%
@other: @base + @filler; // result is 15%
```

ESCAPING

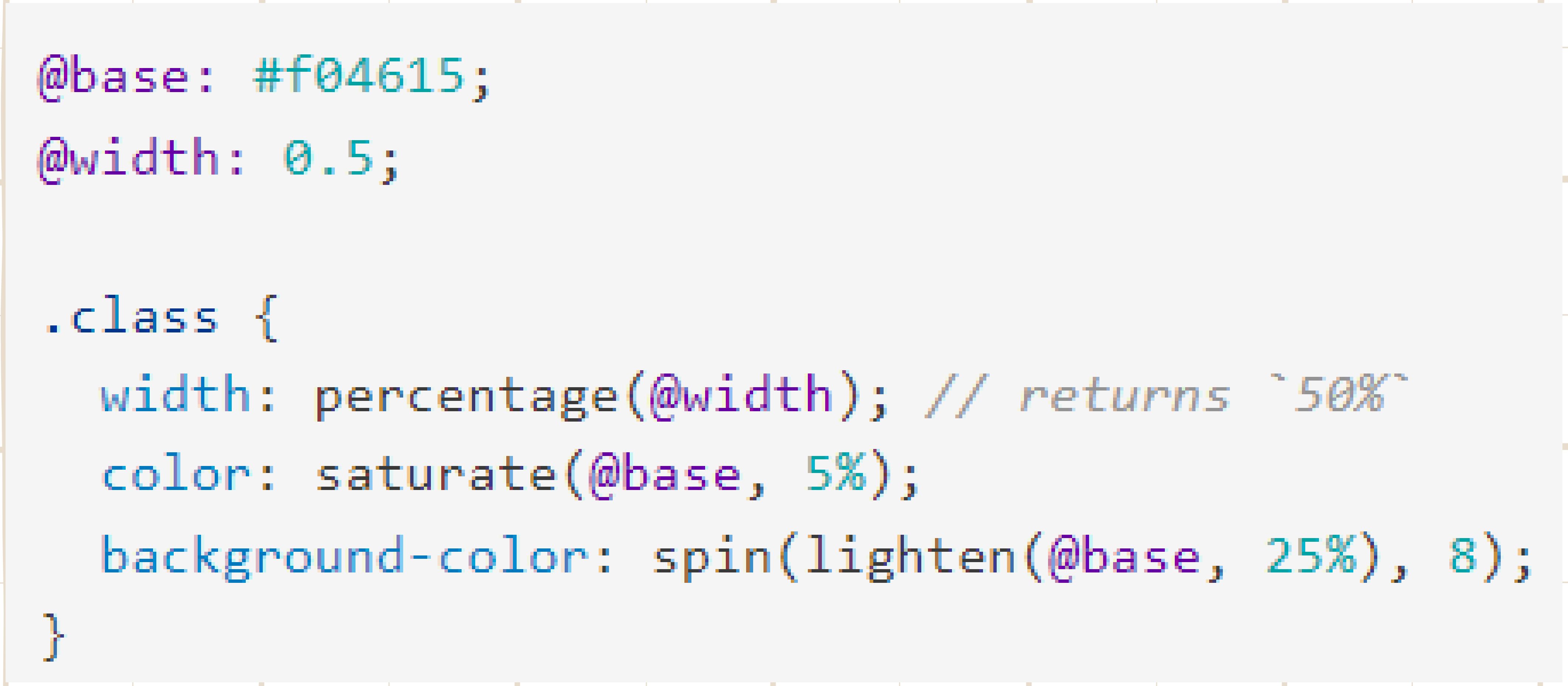
Arithmetical operations +, -, *, / can operate on any number, color or variable.

```
@min768: ~"(min-width: 768px)";
.element {
  @media @min768 {
    font-size: 1.2rem;
  }
}
```

```
@media (min-width: 768px) {
  .element {
    font-size: 1.2rem;
  }
}
```

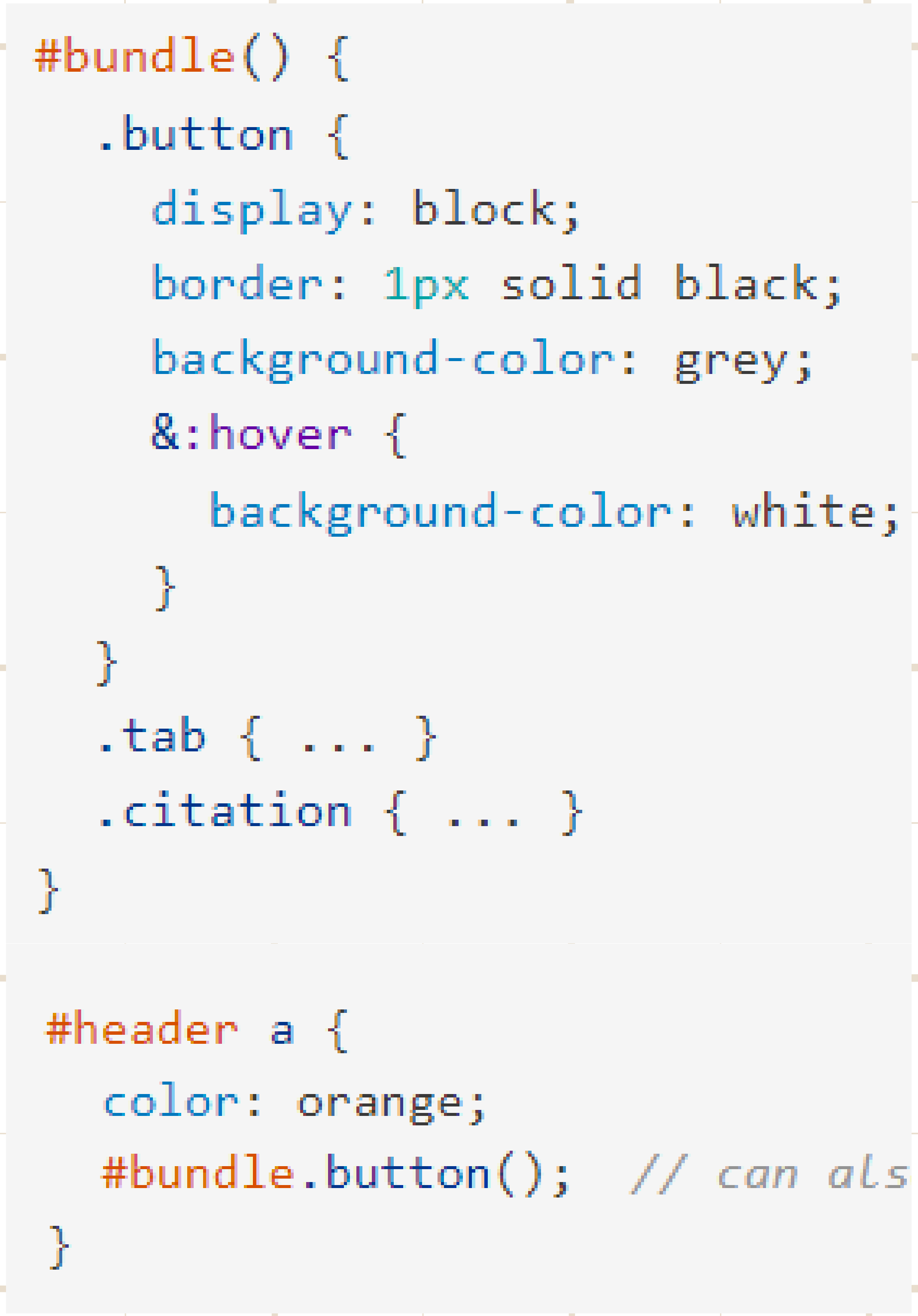
FUNCTIONS

Less provides a variety of functions which transform colors, manipulate strings and do maths.



```less
@base: #f04615;
@width: 0.5;

.class {
  width: percentage(@width); // returns `50%`
  color: saturate(@base, 5%);
  background-color: spin(lighten(@base, 25%), 8);
}
```

NAMESPACES



```
#bundle() {
  .button {
    display: block;
    border: 1px solid black;
    background-color: grey;
    &:hover {
      background-color: white;
    }
  }
  .tab { ... }
  .citation { ... }
}
```

```
#header a {
  color: orange;
  #bundle.button();  // can als
}
```

Sometimes, you may want to group your mixins, for organizational purposes

MAPS

You can also use mixins and rulesets as maps of values.

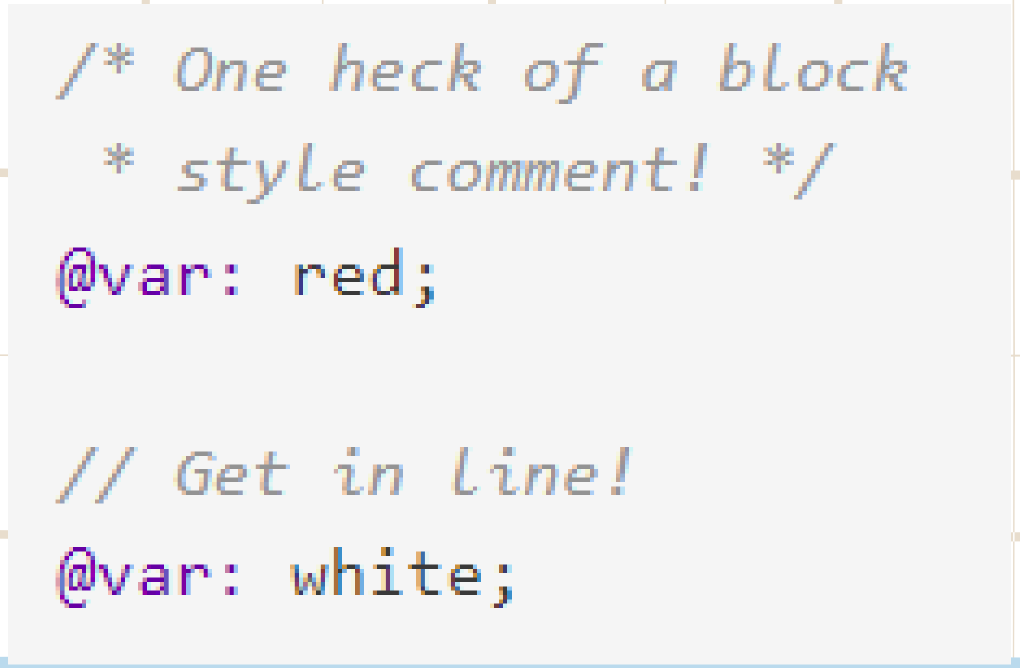
```
#colors() {
  primary: blue;
  secondary: green;
}

.button {
  color: #colors[primary];
  border: 1px solid #colors[secondary];
}
```

```
.button {
  color: blue;
  border: 1px solid green;
}
```

SCOPE

Variables and mixins are first looked for locally, and if they aren't found, it's inherited from the "parent" scope.

COMMENTS



```
/* One heck of a block
 * style comment! */
@var: red;

// Get in line!
@var: white;
```

IMPORTING

Importing works pretty much as expected. You can import a .less file, and all the variables in it will be available. The extension is optionally specified for .less files.

```
@import "library"; // library.less
@import "typo.css";
```

PROS

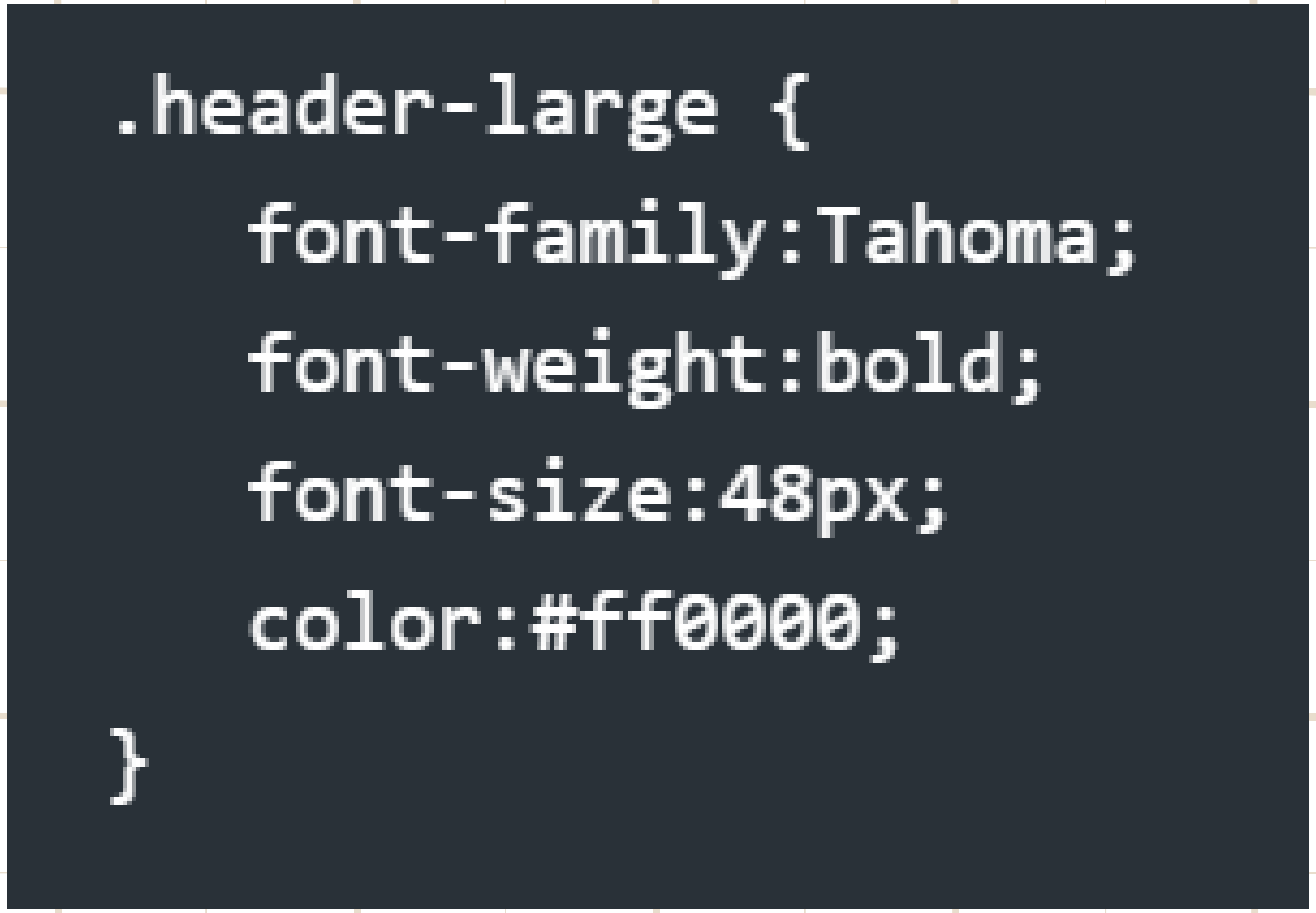
- Cross-browser compatibility
- Clean and organized code
- Simplified maintenance
- Faster and lighter
- File export
- JavaScript based

CONS

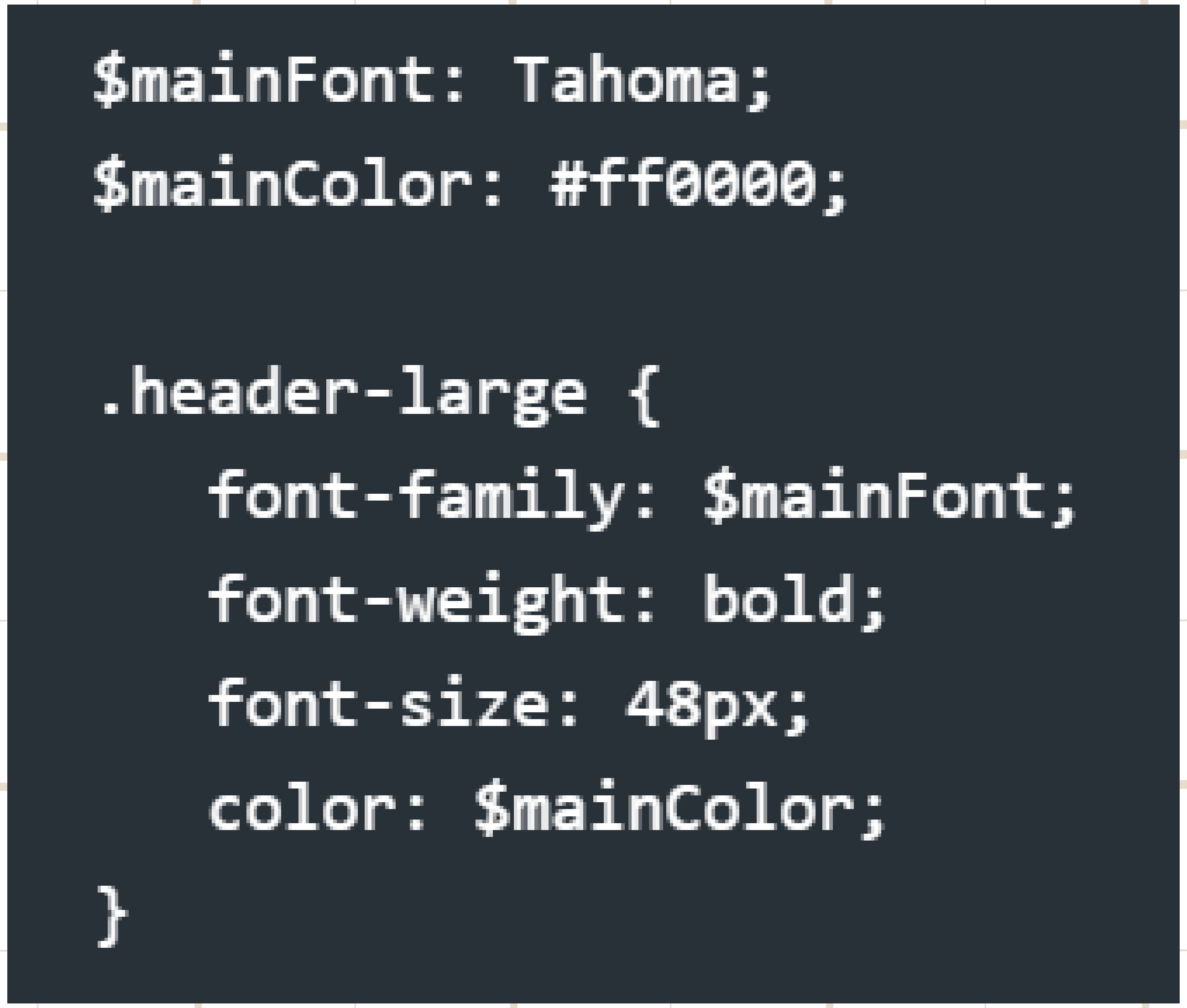
learning curve

less frameworks

WHY PRE PROCESSORS?



```
.header-large {
    font-family:Tahoma;
    font-weight:bold;
    font-size:48px;
    color:#ff0000;
}
```



```
$mainFont: Tahoma;
$mainColor: #ff0000;

.header-large {
    font-family: $mainFont;
    font-weight: bold;
    font-size: 48px;
    color: $mainColor;
}
```

LESS VS SASS

Sass requires Ruby pre-installed and is installed via command line.	With Less the installation is simpler, it only requires Node.js.
Sass hss a more extensive library of built-in functions	Less has Less variety of built-in functions
Both allow the use of variables (Sass with $, Less with @) and mixins (reusable set of styles)	JavaScript based

LESS VS SASS

Less:

- More readable code and easier debugging
- Smoother learning curve
- Documentation and active community

Sass:

- More advanced features such as maps, loops, conditionals, etc.
- Two syntaxes available: Sass (idempotent) and SCSS (like CSS)
- More third-party libraries
- Better integration with frameworks like Bootstrap

LESS VS SASS

Sass is more powerful and versatile, with more advanced features. **Less** is simpler and easier to learn, ideal for getting started with preprocessors.

The choice depends on your specific needs and personal preferences.

THANKS