CSS Preprocessor's

LESS

By Mojtaba Madadyar

http://www.Madadyar.ir

CSS preprocessors

- A CSS preprocessor is basically a **scripting language that extends CSS** and then **compiles it into regular CSS**.
- Here are some other advantages.
  - Cleaner code with reusable pieces and variables
  - Saves you time
  - Easier to maintain code
  - Calculations and logic
  - More organized and easy to setup

LESS vs SASS

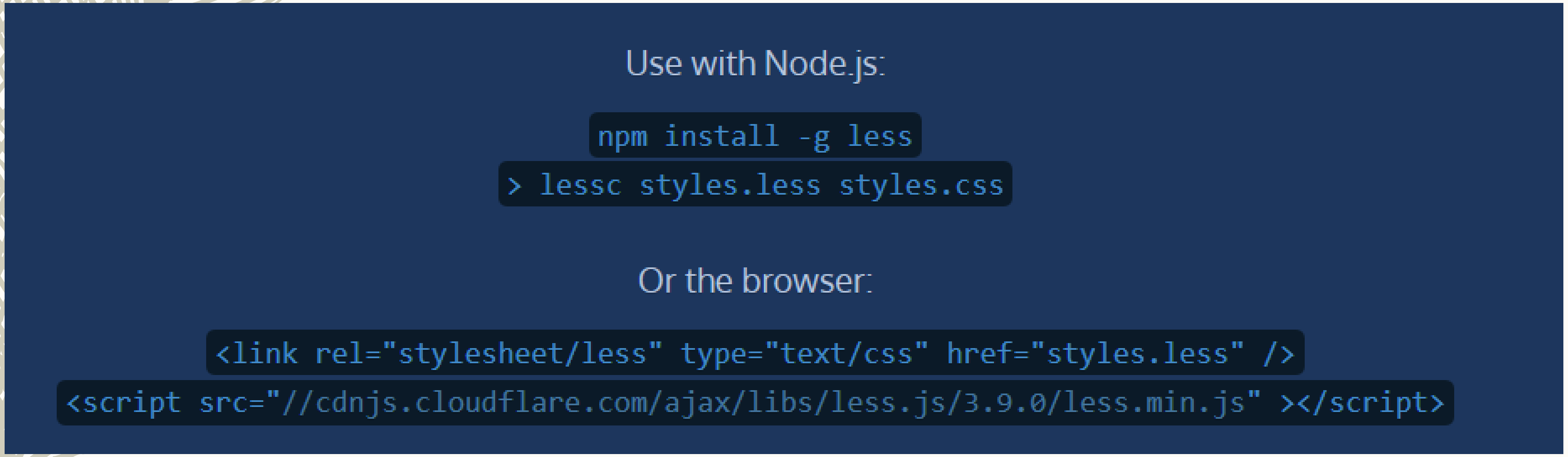
- **Sass** and **LESS** are both very powerful CSS extensions.
- You can think of them as more of a programming language designed to make CSS more maintainable, themeable, and extendable.
- Both **Sass** and **LESS** are backward compatible so you can easily convert your existing CSS files just by renaming the .css file extension to .less or .scss, respectively.
- LESS is JavaScript based and Sass is Ruby based.

Why LESS?

- LESS was designed by **Alexis Sellier** in 2009. LESS is an open-source.
  - The first version of LESS was written in Ruby; in the later versions, the use of Ruby was replaced by **JavaScript**.
- LESS supports creating cleaner, cross-browser friendly CSS faster and easier.
- LESS is designed in JavaScript and also created to be used in *live*, which compiles faster than other CSS pre-processors.
- LESS keeps your code in modular way which is really important by making it readable and easily changeable.
- Faster maintenance can be achieved by the use of LESS *variables*.

Install LESS

- **System Requirements for LESS**
  - **Operating System** – Cross-platform
  - **Browser Support** – IE (Internet Explorer 8+), Firefox, Google Chrome, Safari.



Use with Node.js:

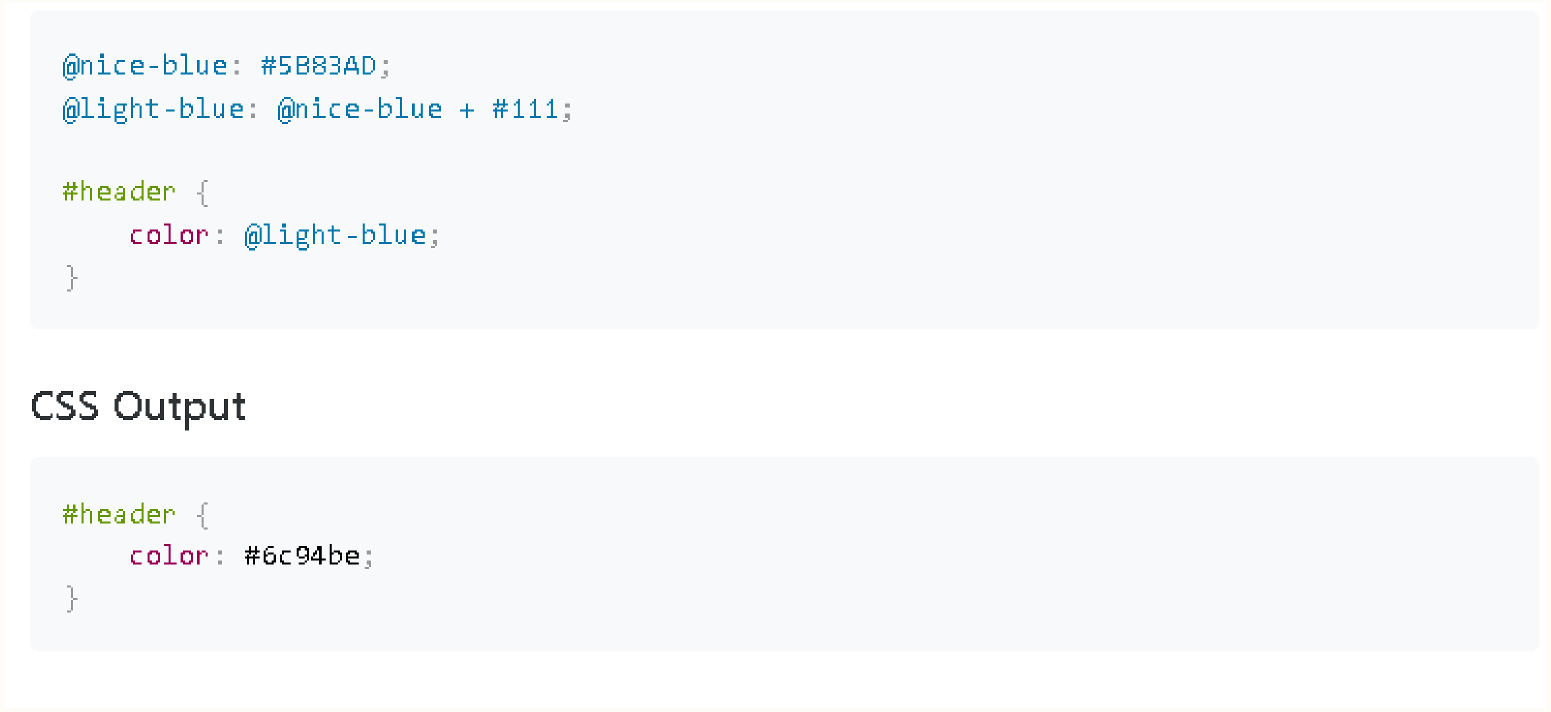
```
npm install -g less
> lessc styles.less styles.css
```

Or the browser:

```
<link rel="stylesheet/less" type="text/css" href="styles.less" />
<script src="//cdnjs.cloudflare.com/ajax/libs/less.js/3.9.0/less.min.js" ></script>
```

Variables

– With preprocessors, you have the advantage over traditional CSS because you can use variables. You can store things like colors, fonts, or pretty much any value you want to reuse later. See examples below.



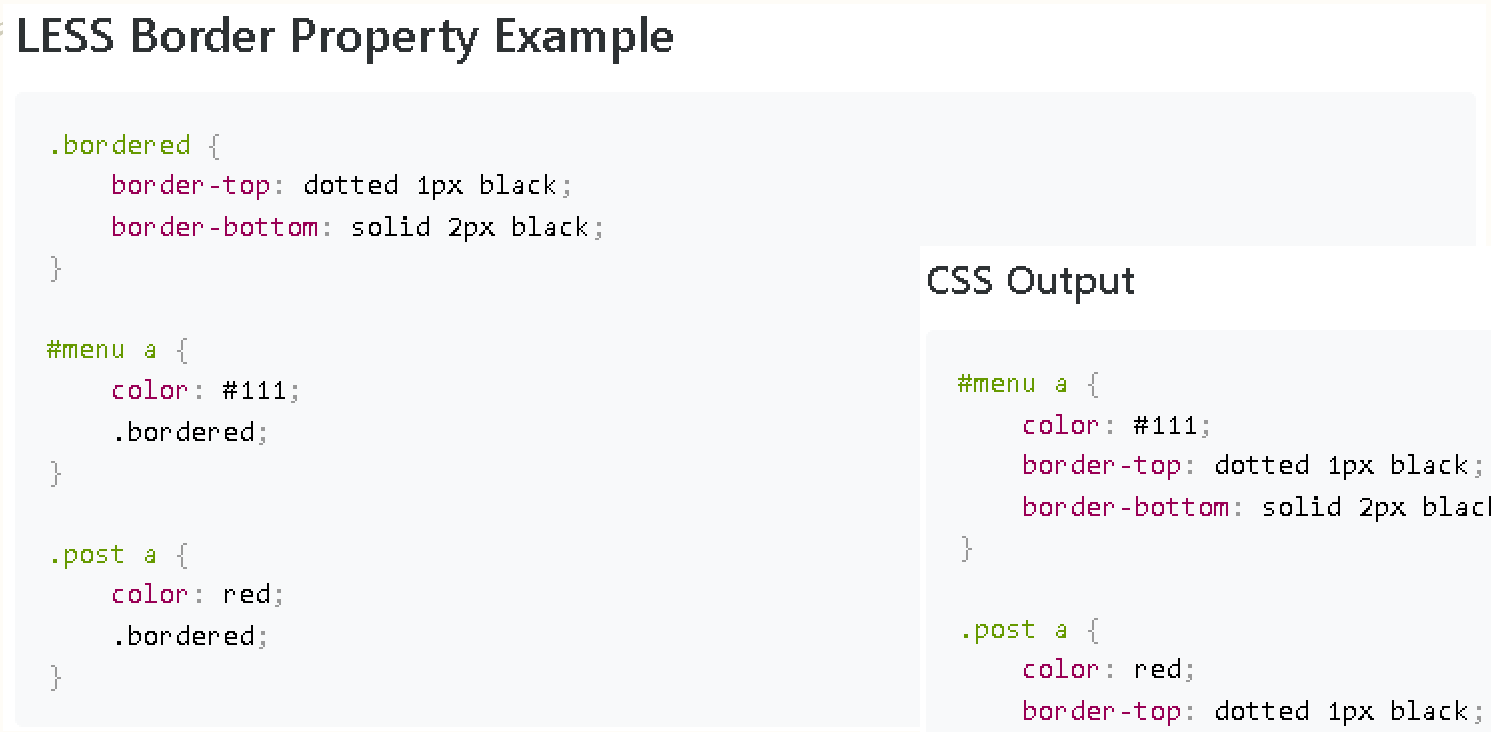
```
@nice-blue: #5B83AD;
@light-blue: @nice-blue + #111;

#header {
    color: @light-blue;
}
```

CSS Output

```
#header {
    color: #6c94be;
}
```

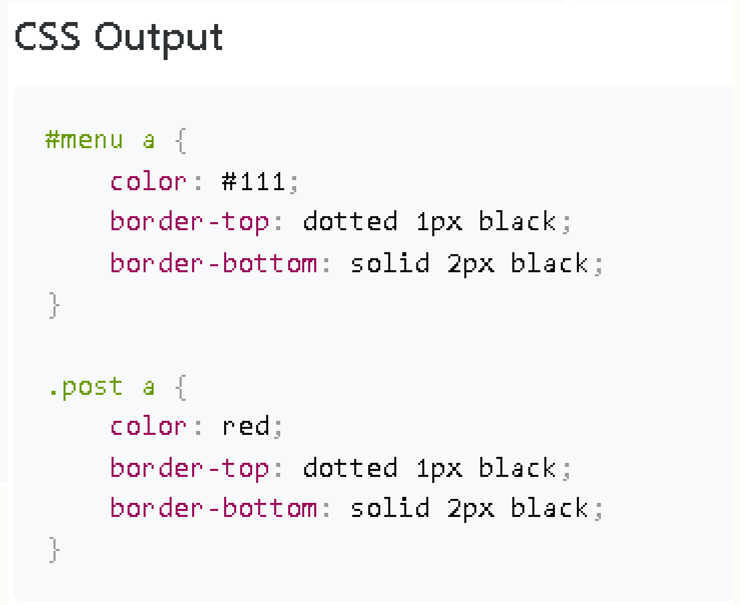
Mixins

– Mixins are used to include a bunch of properties or group declarations together.



LESS Border Property Example

```
.bordered {
    border-top: dotted 1px black;
    border-bottom: solid 2px black;
}

#menu a {
    color: #111;
    .bordered;
}

.post a {
    color: red;
    .bordered;
}
```



CSS Output

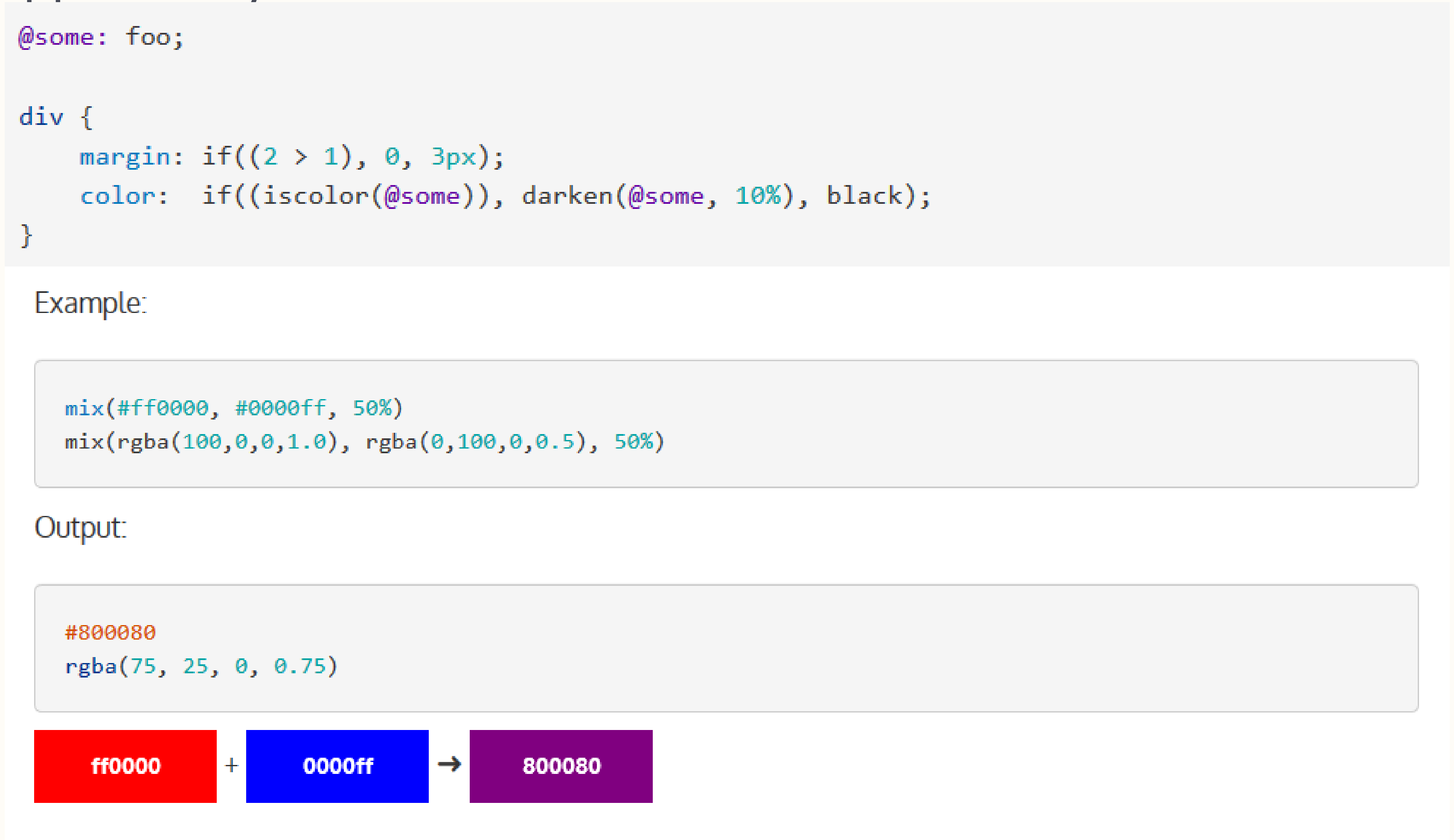
```
#menu a {
    color: #111;
    border-top: dotted 1px black;
    border-bottom: solid 2px black;
}

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

Mixin with Parameter

```
#circle(@size: 25px){
  background-color: #4CAF50;
  border-radius: 100%;

  width: @size;
  height: @size;
}

#small-circle{
  #circle
}

#big-circle{
  #circle(100px)
}
```

Functions

– Built-in functions supported by Less.



```
@some: foo;

div {
    margin: if((2 > 1), 0, 3px);
    color:  if((iscolor(@some)), darken(@some, 10%), black);
}
```

Example:

```
mix(#ff0000, #0000ff, 50%)
mix(rgba(100,0,0,1.0), rgba(0,100,0,0.5), 50%)
```

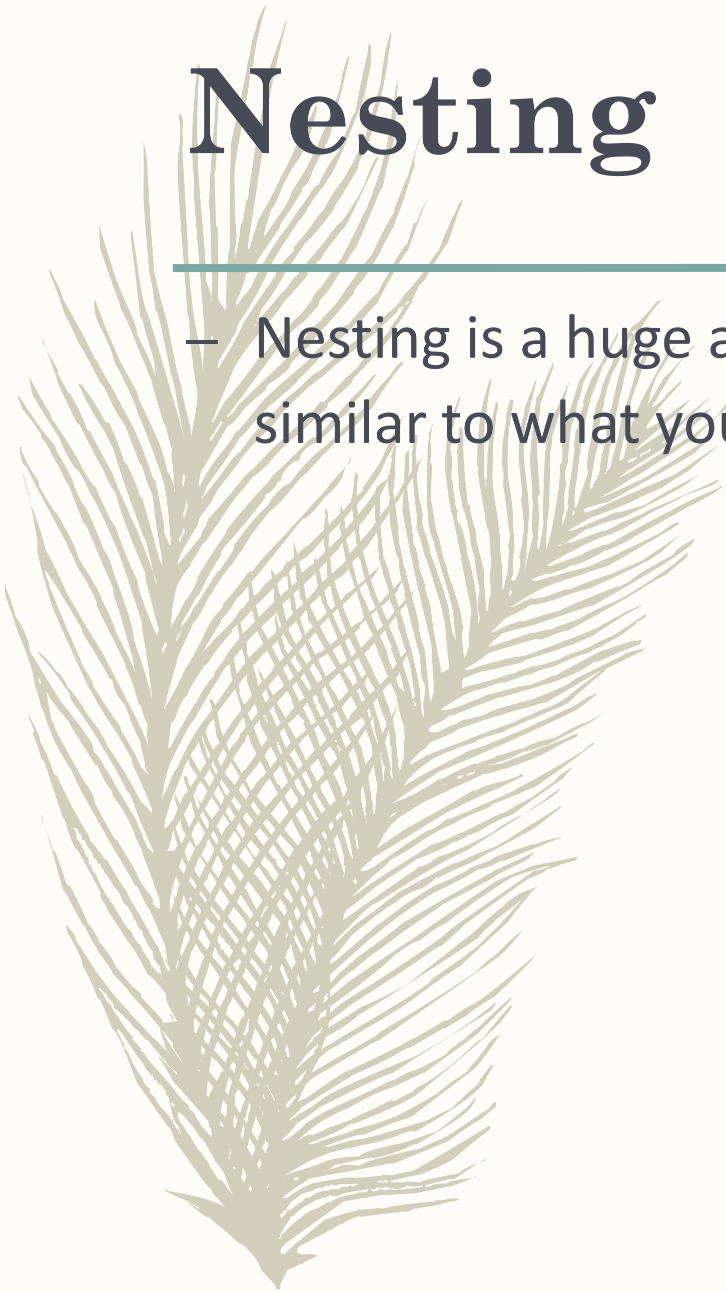
Output:

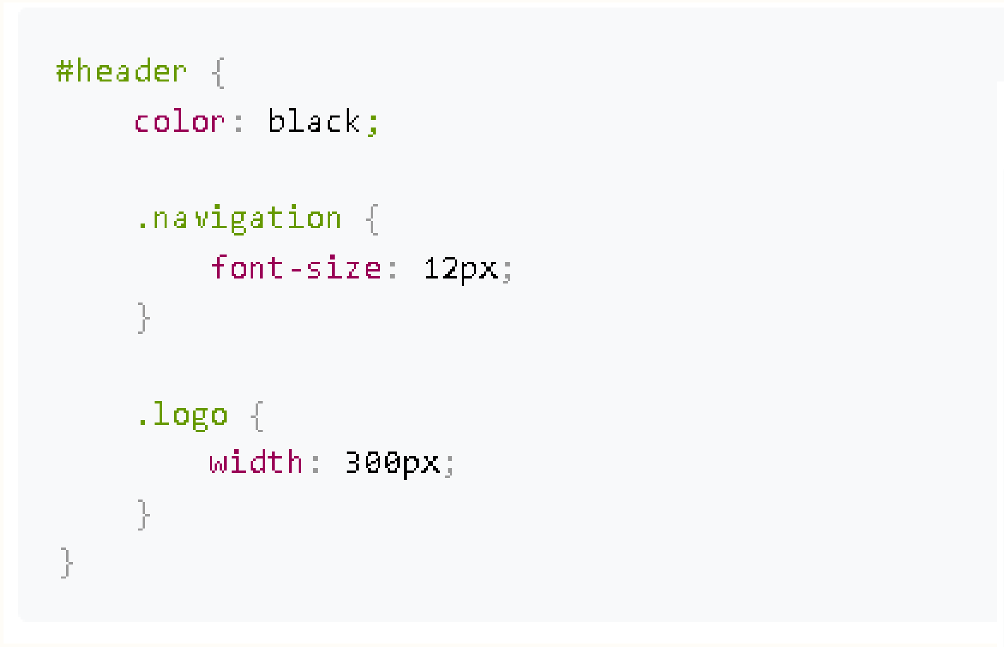
```
#800080
rgba(75, 25, 0, 0.75)
```

ff0000 + 0000ff → 800080

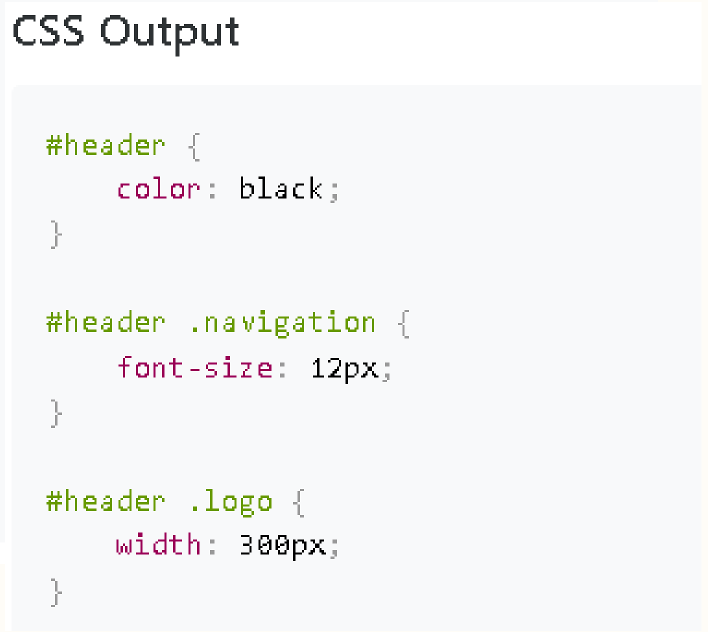
Nesting

– Nesting is a huge advantage over CSS because it creates a visual hierarchy, similar to what you are used to with HTML.





```
#header {
    color: black;

    .navigation {
        font-size: 12px;
    }

    .logo {
        width: 300px;
    }
}
```

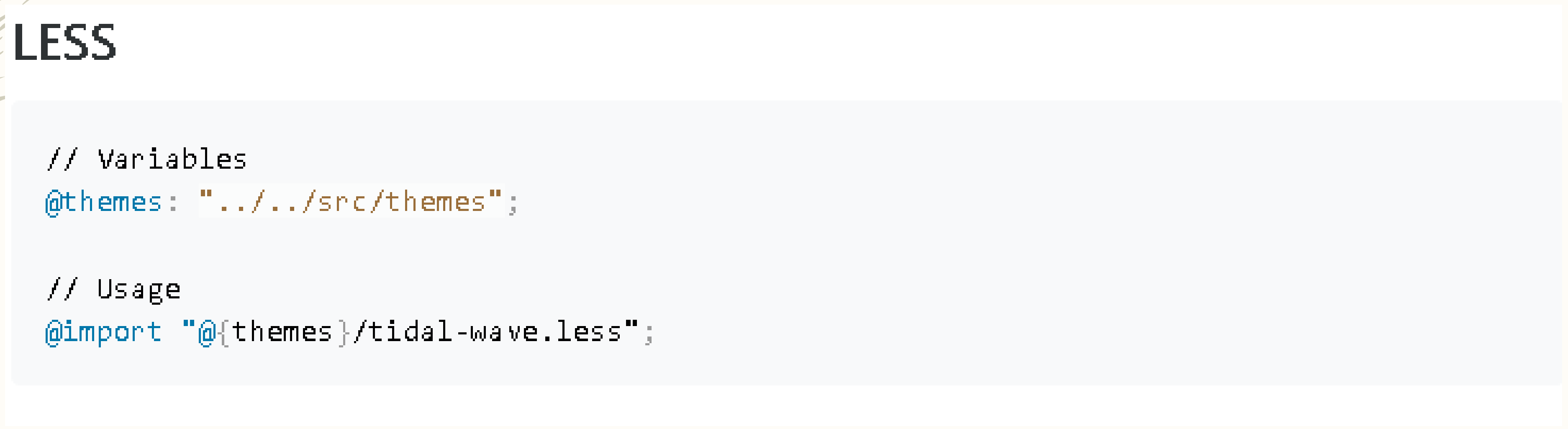


CSS Output

```
#header {
    color: black;
}

#header .navigation {
    font-size: 12px;
}

#header .logo {
    width: 300px;
}
```

Import

- The standard CSS @import allows you to split into multiple files. The problem with this is that it creates additional HTTP requests.
- **Less** work a little different. Instead of creating another HTTP request, they combine the files into one. Similar to concatenation.



LESS

```less
// Variables
@themes: "../../src/themes";

// Usage
@import "@{themes}/tidal-wave.less";
```

Extend

– This is another powerful feature. It lets you share properties from one selector to another.

```
nav ul {
    &:extend(.inline);
    background: blue;
}

.inline {
    color: red;
}
```

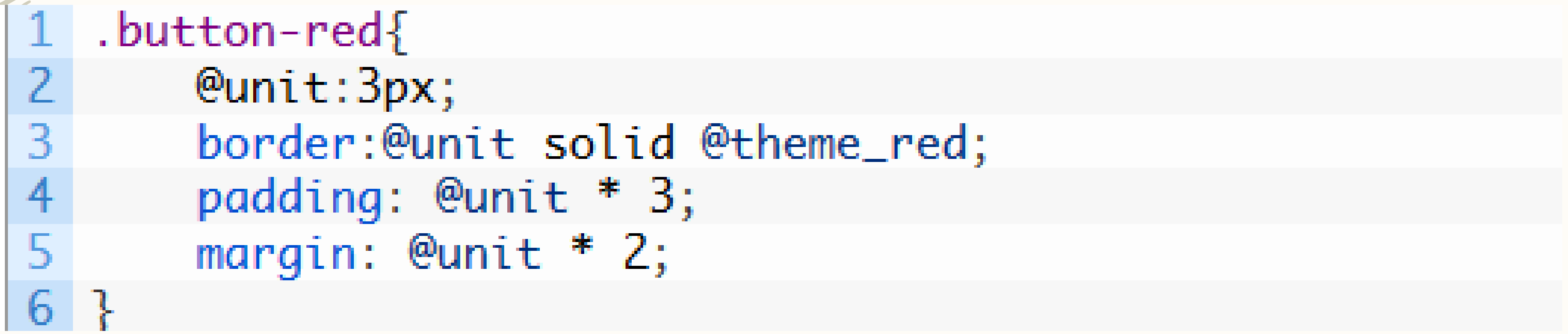
Referencing parent selectors with &

CSS Output

```
nav ul {
    background: blue;
}

.inline, nav ul {
    color: red;
}
```

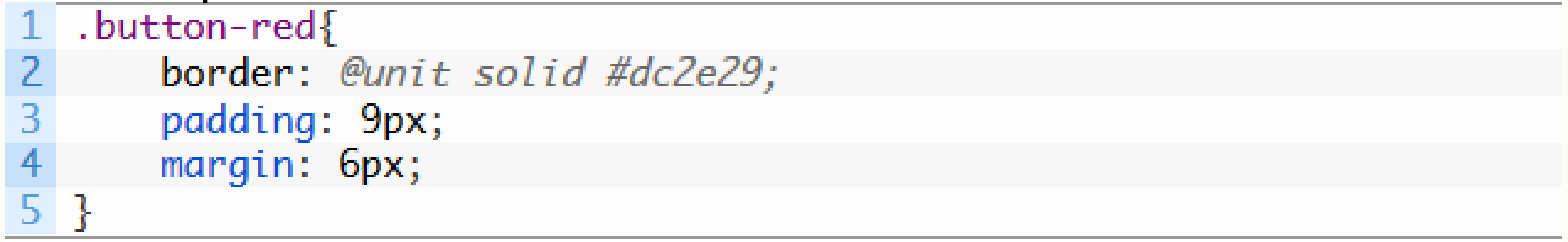
Operations

– Unlike with CSS, you can do all sorts of operations in LESS.



```
.button-red{
    @unit:3px;
    border:@unit solid @theme_red;
    padding: @unit * 3;
    margin: @unit * 2;
}
```

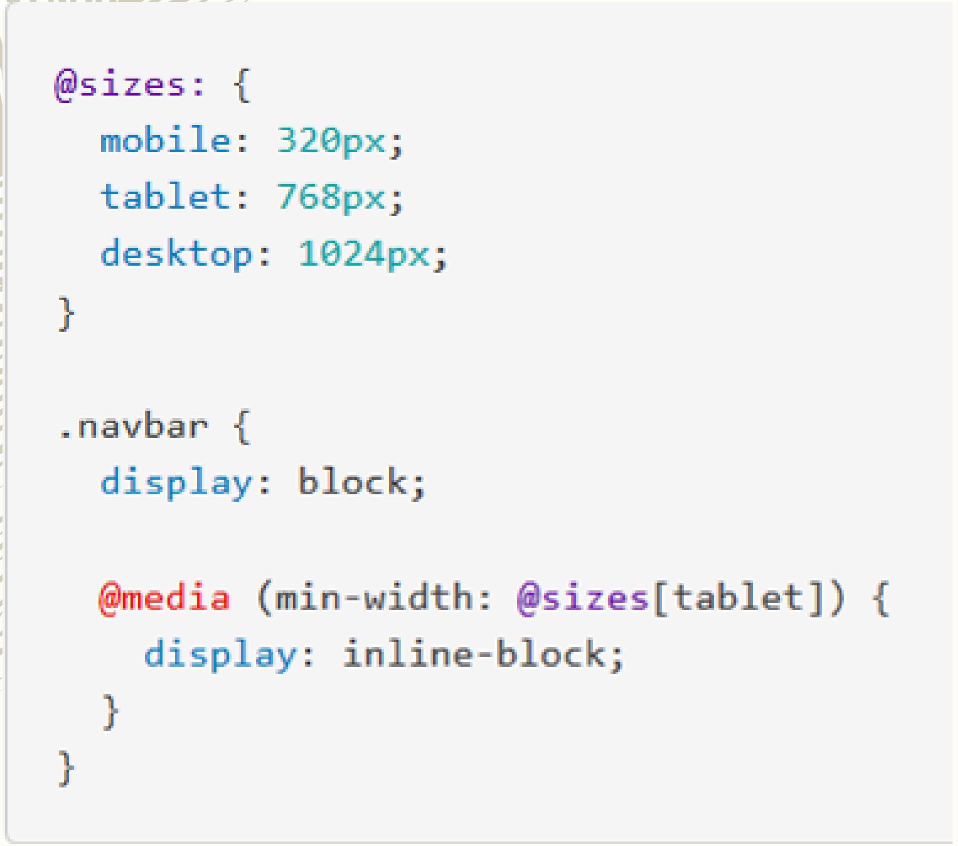
CSS Output



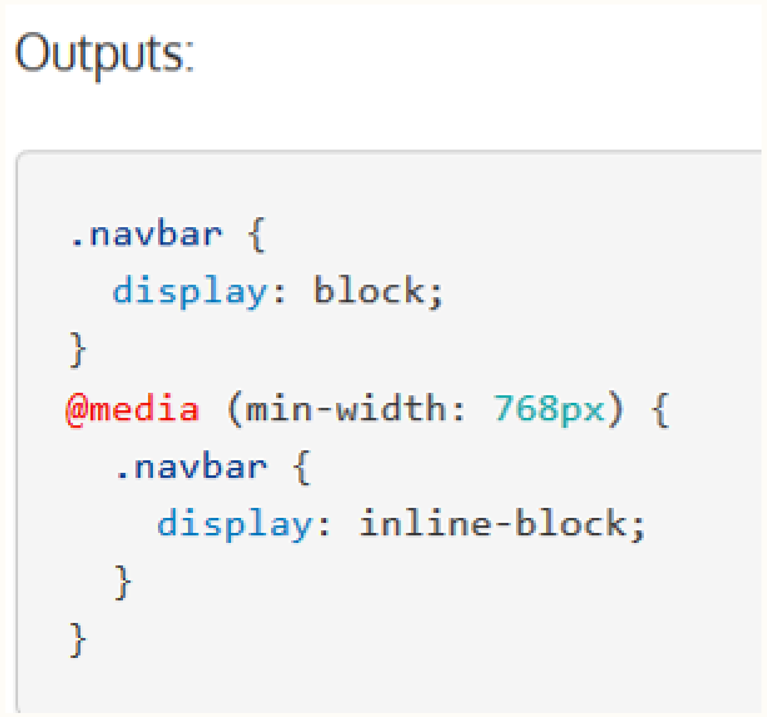
```
.button-red{
    border: @unit solid #dc2e29;
    padding: 9px;
    margin: 6px;
}
```

Maps

- Use rulesets and mixins as maps of values.
  - By combining namespacing with the lookup [] syntax, you can turn your rulesets / mixins into maps.



```
@sizes: {
  mobile: 320px;
  tablet: 768px;
  desktop: 1024px;
}

.navbar {
  display: block;

  @media (min-width: @sizes[tablet]) {
    display: inline-block;
  }
}
```



Outputs:

```
.navbar {
  display: block;
}
@media (min-width: 768px) {
  .navbar {
    display: inline-block;
  }
}
```

References

– http://lesscss.org/