CSS Preprocessors

A preprocessor is a program that converts one type of data to another. CSS Preprocessors generate CSS using a custom language syntax that typically includes features that don’t exist in pure CSS. This way, writing code is made easier and extended with more features and assets.

**SASS (Syntactically Awesome Style Sheets)**

Sass is a scripting language that when compiled, is converted to CSS. The designer and developer of this scripting language are Hampton Caitlin and Natalie Weizenbaum, respectively. Later, SassScript was created. It is an extension of Sass- a scripting language itself. The separation of code blocks and rules is possible by using indentation and newline characters. Several mechanisms observed in traditional programming languages like object-oriented languages are available extensions.

Variables

Sass allows the initialization of variables. Variables are declared using a dollar sign ($) and assigned with a colon (:). Numbers (including units), String (with or without quotes), Colors (names), and Boolean data types are supported.

Sass

$primary-color: #990a

$margin: 12px

.content-nav

border-color: $primary-color

padding: $margin

margin: $margin/2

Would compile to:

.content-nav {

border-color: #990a;

padding: 12px;

margin: 6px;

}

Nesting

Nesting inserts nested code within each other. This way, visual hierarchy can be easily noticed.

Sass

tr

font:

family: Verdana;

size: 1.6em;

Would compile to:

tr {

font-family: Verdana;

font-size: 1.6em;

}

Loops

The loops @for, @each, and @while, can be used to apply different styles to similar elements through iteration.

Sass

$counter: 4

@for $i from 1 through $counter

#name-#{$i}

background-color: green

height: 50px

Would compile to:

#name-1{

background-color: green;

height: 50px;

}

#name-2{

background-color: green;

height: 50px;

}

#name-3{

background-color: green;

height: 50px;

}

#name-4{

background-color: green;

height: 50px;

}

Arguments

Mixin supports arguments. This is a set of definitions that are compiled based on other parameters or rules.

Sass

@mixin bordered(width) {

border: $width solid #366ff;

&:hover {

border-color: #99ccff;

}

}

p {

@include bordered(10px);

}

Would compile to:

p {

border: 10px solid #366ff;

}

p:hover {

border-color: #99ccff;

}

Selector Inheritance

The @extend keyword is used when you want to share a general style among different elements without the need of using that block of code. This way, inheritance can be achieved.

Sass

.block {

margin: 12px 8px;

border: 1px;

background: #ddd

}

h1 {

@extend .block;

font-size: 16px;

text-transform: lowercase;

}

Would compile to:

.block, h1 {

margin: 12px 8px;

border: 1px;

background: #ddd

}

h1 {

font-size: 16px;

text-transform: lowercase;

}

CSS Frameworks

A CSS Framework is a package of predefined files and folders of code that aims for easier and more standard-compliant web design. Through frameworks, CSS functionalities can be reused, extended, and customized.

**Bootstrap**

Bootstrap is basically a free front-end framework that includes various HTML and CSS design templates and JavaScript plugins. This allows you to use existing templates for forms, buttons, navigation, tables, and many more including being able to create responsive designs. It was developed by Mark Otto and Jacob Thornton.

**Foundation**

Foundation, an internal style guide built and maintained by ZURB, is a free and responsive front-end framework that provides HTML and CSS design components, templates, code snippets and many more. It includes other interface elements such as buttons, navigation, and typography. It was built with Sass, making it easy to style and extend.

CSS Polyfills

A polyfill provides an extension of a browser’s technology that isn’t provided natively but developers expect it to be. It is a piece of code or plugin that flattens the API landscape.

**Modernizr**

Modernizr is a Javascript library that has the ability of detecting implemented features of HTML5 and CSS3 in different browsers. This allows developers to maximize new features that a browser offers and also have a fallback when browsers do not. Instead of looking into the properties of the browser, feature detection is able to recognize what a browser supports.

Feature detection is more reliable by creating tests for more than 250 features. These results are then stored in a JavaScript object called Modernizr. Classes to the HTML element are also added based on features that are and are not natively supported. Feature detection involves creating an element with a specific style instruction then retrieving that setting. If a web browser supports that feature, it will return something sensible since it understands the instruction. Otherwise, nothing or “undefined” is returned.