SASS Tutorial

SASS (Syntactically Awesome StyleSheet) ist eine OpenSource Skriptsprache und stellt eine Erweiterung für CSS dar. Vor allem in großen Projekten werden Stylesheets immer umfangreicher und komplexer. Mit seinen vielen Features hilft SASS eine klare Struktur einzuhalten. Erreicht wird dies durch Wiederverwendung von Code-Passagen und die Nutzung von Kontrollstrukturen, ähnlich wie in eine Programmiersprache. Darüber hinaus lassen sich in SASS bestimmte Styles oftmals einfacher implementieren als in CSS.

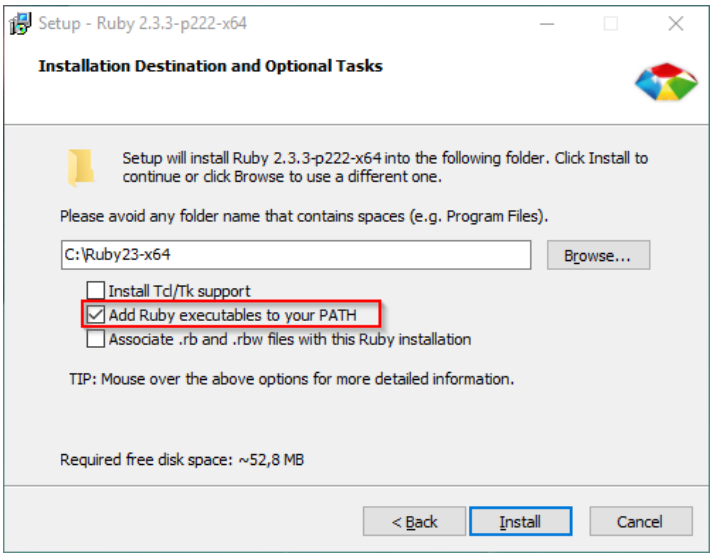
Aktuelle Browser können kein SASS interpretieren, weshalb die SASS-Dateien vorher nach CSS umgewandelt werden. Dies übernimmt ein Präprozessor. Im HTML Dokument wird dann auf diese CSS-Datei verwiesen.

Installation

Ruby

Für die Verwendung von SASS muss Ruby installiert werden:

Windows: https://rubyinstaller.org/



Linux:

```
$ sudo apt-get install ruby
```

Mac: bereits installiert

Sass

Ist Ruby installiert, kann SASS über die Kommandozeile installiert werden:

```
gem install sass
```

Um zu testen ob SASS erfolgreich installiert wurde, kann die Version von SASS abgefragt werden:

```
sass -v
```

Erstellt nun in einem Verzeichnis eine .scss Datei (z.B. master.scss). Navigiert in das Verzeichnis und gebt folgendes ein:

```
sass --watch master.scss:master.css
```

Nach jeder Änderung der .scss Datei wird nun automatisch eine .css Datei erzeugt.

Im Folgenden soll nun eine einfache HTML Website mit Hilfe von SASS gestaltet werden. Dabei werden einige Features von SASS vorgestellt. Achtet darauf, dass in der HTML Datei das Stylesheet eingebunden wird.

Variablen

Variablen eigenen sich gut, um Informationen wiederzuverwenden. In diesem Beispiel soll zunächst die Hintergrundfarbe des gesamten Dokuments verändert werden. Wir benutzen dafür eine Variable, welche den Farbwert speichert. Variablen werden mit dem $ Zeichen eingeleitet:

```
$MyColor: #d6d6d6;
body{background-color: $MyColor}
```

Verschachtelungen

HTML Dokumente sind hierarchisch aufgebaut. Im Gegensatz zu CSS kann SASS diese Struktur nachbilden. Wir nutzen nun die Verschachtelung in SASS, um die Überschrift (h1) in der Klasse main zu formatieren:

```
.main{
  h1{
    color: red;
    font-size: 30px;
    text-transform: uppercase;
    text-decoration: underline;
  }
}
```

Mixins

Mixins sind nützlich, wenn Code mehrmals verwendet werden soll. Mit @mixin, gefolgt von einem Namen wird solches eingeleitet. In diesem Abschnitt können nun beliebig viele Deklarationen gemacht werden. In unserem Beispiel wird der Abschnitt (p – Tag) über ein Mixin formatiert. Dazu muss zusätzlich zur Definition des Mixins, dieses mit @include eingebunden werden.:

```
@mixin paraStyle {
  text-align: center;
  font-size: 20px;
  font-family: sans-serif;
  color: complement(green);
}
.main{
  p{
    @include paraStyle;
  }
}
```

Möchte man Mixins variabel halten, können auch Parameter übergeben werden (ähnlich wie bei einer Funktion). In diesem Beispiel können Schriftgröße und Farbe als Parameter übergeben werden:

```
@mixin paraStyle($size, $color) {
  text-align: center;
  font-size: $size;
  font-family: sans-serif;
  color: complement($color);
}
.main{
  p{
    @include paraStyle(22px, green);
  }
}
```

Vererbung

Im nächsten Schritt soll die Liste (ul -Tag) genau die gleiche Formatierung erhalten wie der vorherige Abschnitt. Man könnte nun ebenfalls das Mixin mit den gleichen Parametern einbinden. Für den Fall, dass später die Parameter des Abschnittes geändert werden, müssen ebenfalls die Parameter der Liste angepasst werden. Daher bietet sich das Vererbungsfeature von SASS an. Mit @extend gefolgt von dem Element, von welchem geerbt werden soll, werden alle seine Eigenschaften übernommen:

```
.main{
ul{
    @extend p;
    text-align: left;
  }
}
```

Modularität

SASS bietet die Möglichkeit den Code auf mehrere Dateien aufzuteilen. Dadurch bleiben einzelne Dateien klein und es ist einfacher diese zu verwalten. Oftmals macht es Sinn die Mixins auszulagern. Wir erstellen deshalb eine neue Datei mit dem Namen mixin.scss und verschieben das Mixin paraStyle dort hin. Vom anderen Skript verwiesen wir nun mit @import “mixin“; auf diese Datei. Der Präprozessor fügt während des Konvertierungsvorgangs alle Dateien zusammen, sodass am Schluss eine CSS-Datei herauskommt.

master.scss:

```
@import "mixin";
```

mixin.scss:

```
@mixin paraStyle($size, $color) {
  text-align: center;
  font-size: $size;
  font-family: sans-serif;
  color: complement($color);
}
```

In diesem Tutorial wurden nur grundlegende Features behandelt. SASS ist natürlich deutlich umfangreiche und bietet viele weitere nützliche Features. Wenn ihr mehr über SASS erfahren wollt, besucht folgende Seite: http://sass-lang.com/ Dort findet ihr unter anderem eine ausführliche Dokumentation.