HTML5-Tutorium: JavaScript: Hello World 06

Wechseln zu:Navigation, Suche

Dieser Artikel erfüllt die GlossarWiki-Qualitätsanforderungen **nur teilweise**:

Korrektheit: 3 (zu größeren Teilen überprüft)	**Umfang**: 4 (unwichtige Fakten fehlen)	**Quellenangaben**: 3 (wichtige Quellen vorhanden)	**Quellenarten**: 5 (ausgezeichnet)	**Konformität**: 3 (gut)

Vorlesung WebProg

Inhalt | Teil 1 | Teil 2 | Teil 3 | Teil 4 | Teil 5 | Teil 6

Musterlösung: `index6.html` (Git-Repository)

Inhaltsverzeichnis

1 Anwendungsfälle (Use Cases)
2 Erstellen eines neuen Projektes
3 SCSS
- 3.1 SCSS-Konfiguration
- 3.2 Weitere SCSS-Konstrukte

4 Quellen

1 Anwendungsfälle (Use Cases)

Gegenüber dem dritten Teil des Tutoriums ändern sich die die Anwendungsfälle nicht. Die Anwendung leistet also genau dasselbe wie zuvor.

In diesem Teil des Tutoriums geht es darum, CSS durch SCSS zu ersetzen, um den CSS-Code „DRY zu machen“.

2 Erstellen eines neuen Projektes

Erstellen Sie ein neues Projekt `HelloWorld06` als Fork von `HelloWorld05` (siehe Hello-World-02-Tutorium).

```
git clone https://gitlab.multimedia.hs-augsburg.de/ACCOUNT/HelloWorld05
HelloWorld06
//git clone https://gitlab.multimedia.hs-
augsburg.de/kowa/WK_HelloWorld05d HelloWorld06

## Änderungen zur Projektidentifikation vornehmen (z.B. Titel anpassen)
git commit -m "HelloWorld05 fork created"

# Neues Repository in Gitlab anlegen:
git remote -v
git remote remove origin
git remote add origin https://gitlab.multimedia.hs-
augsburg.de/ACCOUNT/HelloWorld06
git remote -v
git push --set-upstream origin master
```

3 SCSS

Anstelle von CSS-Dateien verwenden Sie künftig SCSS-Dateien. Das ist zunächst einmal nicht weiter schwer, das jede CSS-Datei automatisch auch eine SCSS-Datei ist. Bei der Verwendung von CSS können Sie allerdings das Prinzip „Don't repeat yourself“ (DRY) nicht beachten. Sie müssen für verschiedene Elemente ständig Informationen wiederholen (gewünschter Font, Hintergrundfarbe, Größenangaben etc.). Deshalb gibt es mehrere Projekte wie z. B. LESS, Sass/SCSS, Stylus, die den CSS-Standard erweitern, um dieses Problem zu vermeiden. Sass verwendet allerdings eine ganz andere Syntax als CSS. Allerdings unterstützt das Sass-Projekt auch die Sprache SCSS, die CSS einfach um weitere syntaktische Elemente ergänzt. Zum Beispiel kann man in SCSS Konstanten definieren:

```
$background-color: #C5EFFC;
```

Und nun kann man in eine SCSS-Datei anstelle von `#C5EFFC` stets `$background-color` schreiben, um die Hintergrundfarbe von bestimmten Elementen zu definieren. Wenn man nun diese Farbe ändern will, muss das nicht mehr bei allen Elementen einzeln passieren. Es reicht, die Farbe in der Konstantendefinition abzuändern. Der SCSS-Code ist nun DRY.

Leider unterstützt kein Browser SCSS direkt. Aber für was gibt es Transpiler wie Webpack, die eine SCSS-Datei automatisch in eine CSS-Datei übersetzen können. Im fünften Teil des Hello-World-Tutorium haben wir im der Version 5d den SASS-Loader verwendet, um CSS-Dateien zu komprimieren. Damit haben wir aber gleichzeitig auch die Möglichkeit erhalten mit SCSS-Dateien an Stelle von CSS-Dateien zu verwenden.

Benennen Sie die Dateien `src/css/head.css` und `src/css/body.css` in `src/css/head.scss` und `src/css/body.scss` um. Ersetzen Sie außerdem in der Datei `src/js/main.js` den Import-Befehl

```
import '/css/body.css';
```

durch

```
import '/css/body.scss';
```

Zu guter Letzt müssen sie in `webpack.config.js` im Entry-Objekt folgende Ersetzung vornehmen:

```
head: absolutePath('src/css/head.css')
```

wird zu

```
head: absolutePath('src/css/head.scss')
```

Wenn Sie nun mittels `npm run dev` oder `npm run prod` Ihren Code übersetzen, sollte noch alle laufen, da der SASS-Loader die neuen SCCS-Dateien ebenfalls in CSS-Dateien übersetzt.

Ersetzen Sie in Ihrem Projekt die Datei `src/css/main.css` durch drei SCSS-Dateien: `_config.scss`, `initial.scss` und app.scss.

3.1 SCSS-Konfiguration

In der Datei `_config.scss` werden alle von den anderen SCSS-Dateien benötigten Konstanten definiert. Das heißt, das DRY-Prinzip „Konstanten gehören nicht in den Code, sondern in Konfigurationsdateien“ wird ab sofort beachtet. Wenn man künftig am Layout einen Zeichensatz, eine Farbe, einen Hintergrund oder Ähnliches ändern will, reicht es, die entsprechenden Konstanten anzupassen und schon erzeugt Webpack automatisch aktualisiere CSS-Dateien.

Der Underscore _ am Beginn des Dateinamens deutet an, dass diese SCSS-Datei nur von anderen SCSS-Dateien aber nicht von HTML- oder JavaScript-Dateien importiert werden. Durch SCSS-Import-Befehle ist es möglich, auch SCSS-Dateien zu modularisieren.

`src/css/_config.scss`

```
@use 'sass:math';

$background-color:  #C5EFFC;

$font-family-sans:  Verdana, Helvetica, sans-serif;
$font-family-serif: "Times New Roman", Times, serif;

$font-size-base:    3vw;
$font-size-factor:  1.25;  // cmp. https://type-scale.com/

$font-size-h5:      math.pow($font-size-factor,1)*$font-size-base;
$font-size-h4:      math.pow($font-size-factor,2)*$font-size-base;
$font-size-h3:      math.pow($font-size-factor,3)*$font-size-base;
$font-size-h2:      math.pow($font-size-factor,4)*$font-size-base;
$font-size-h1:      math.pow($font-size-factor,5)*$font-size-base;

$font-size-p:       math.pow($font-size-factor,0)*$font-size-base;
$font-size-p-large: math.pow($font-size-factor,2)*$font-size-base;

$font-size-small:   math.pow($font-size-factor,-1)*$font-size-base;
```

Konstanten starten in SCSS mit einem Dollarzeichen. Sobald eine Konstante definiert wurde, kann sie in allen weiteren SCSS-Konstrukten verwendet werden. In SCSS stehen einen Operatoren zur Verfügung wie `+`, `-`, `*` und `/`, um Maße zu berechnen. Es gibt weitere Operatoren zum Vergleichen von Zahlen und Maßen sowie zur Berechnung von Strings und booleschen Werten: https://sass-lang.com/documentation/operators.[1]

Die Datei `_config.scss` kann nun mit Hilfe des SCSS-Befehls `@import 'config';` (ohne Underscore und ohne Endung) in die anderen beiden SCSS-Dateien eingefügt werden. Webpack (genauer gesagt der SASS-Loader von Webpack) kennt diesen Befehl und führt im beim Erstellen der zugehörigen CSS-Dateien auch aus.

Der Import-Befehl hat erst dann einen Effekt, wenn in den anderen Dateien auf diese Konstanten zugegriffen wird. Aber er schadet auch nicht, wenn auf keine der dort definierten Konstanten zugegriffen wird. Der Import-Befehl selbst taucht ebenso wenig in den generierten CSS-Dateien auf, wie die Konstantendefinitionen. Der Transpiler ersetzt Konstanten in CSS-Definitionen durch ihren jeweiligen Wert.

Sie können nun beispielsweise in der Datei `head.scss` die Konstante `$background-color` verwenden. In der daraus erzeugten CSS-Datei steht weiterhin der Farbwert `#C5EFFC`;

`src/css/head.scss`

```
@import 'config';

body
{ background-color: $background-color; }

.hidden
{ display: none; }
```

Die Datei `body.scss` ist deutlich umfangreicher. Daher wird der zugehörige CSS-Code ja auch erst gemeinsam mit der Datei `web/js/main.js` geladen.

`src/css/body.scss`

```
@import 'config';

html
{ height:  100%;
  width:   100%;
  display: table;
}

body
{ display:          table-cell;
  text-align:       center;
  vertical-align:   middle;
  background-color: $background-color;
  font-size:        $font-size-base !important;
  font-family:      $font-family-sans;
}

h1
{ padding:   0;
  margin:    0;
  font-size: $font-size-h1;
}

p, label
{ font-family: $font-family-serif;
  font-size:   $font-size-p !important;
}

label, input, button
{ width:      10em;
  font-size:  100%;
  display:    inline-block;
  box-sizing: border-box;
  margin:     0.5ex;
}

label
{ text-align: right;
}

#section_form
{ text-align:   center;
  margin-left:  auto;
  margin-right: auto;

  h1
  { margin-bottom: 0.5ex; }
}
```

```
#section_hello
{ p
  { font-size: $font-size-p-large !important; }
}
```

In dieser SCSS-Datei werden folgende SCSS-Erweiterungen verwendet:

Mathematische Berechnungen: `@use 'sass:math';`
Import-Befehl: `@import 'config';`
Konstanten: `$font-family-sans`, `$font-size-base` etc.
Verschachtelte Anweisungen: z. B. `h1` innerhalb von `#section_form`

Die verschachtelten Anweisungen strukturieren den CSS-Code deutlich besser. Es ist nicht mehr notwendig einen Selektor wie `#section_form` mehrfach zu verwenden.

Sehen Sie sich den Unterschied an. In der Datei `src/css/body.css` der Musterlösungen vom dritten Teil des Tutoriums hatte ich Folgendes geschrieben:

```
#section_form
{ text-align:   center;
  margin-left:  auto;
  margin-right: auto;
}

#section_form > h1
{ margin-bottom: 0.5ex; }
```

Die SCSS-Variante ist deutlich strukturierter und prägnanter.

```
#section_form
{ text-align:   center;
  margin-left:  auto;
  margin-right: auto;

  h1
  { margin-bottom: 0.5ex; }
}
```

Insbesondere bei großen CSS-Dateien erweist sich diese kompaktere Schreibweise als vorteilhaft.

3.2 Weitere SCSS-Konstrukte

SCSS ist sehr nächtig. Es gibt neben der Import-Anweisung eine Use-Anweisung (`@use`)[2], die, SCSS-Konstrukten im Gegensatz zur Import-Anweisung einen Namespace zuweist. Damit ist es möglich mehrere SCSS-Module zu verwenden, auch wenn dieses gleichbenannte Elemente enthalten, ohne dass es zu Namenskollisionen kommt.

Darüber hinaus gibt es Mixins[3] und Funktionen[4]. Mit Mixins kann man ganze Styleblöcke definieren,

um sie mehrfach zu verwenden. Mixins werden mit `@include` eingebunden. Dabei kann man Argumente an die Mixins übergeben, mit denen die entsprechenden Konstanten innerhalb des Mixins initialisiert werden. Funktionen funktionieren wie Funktionen in anderen Programmiersprachen auch. Es stehen einem If-Anweisungen und Schleifen-Befehle zur Verfügung.

SCSS ist im Prinzip eine vollwertige Programmiersprache, um CSS-Dateien modular zu erstellen.

4 Quellen

https://sass-lang.com/documentation/operators
https://sass-lang.com/documentation/at-rules/use
https://sass-lang.com/documentation/at-rules/mixin
https://sass-lang.com/documentation/at-rules/function
Kowarschick (MMProg): Wolfgang Kowarschick; Vorlesung „Multimedia-Programmierung“; Hochschule: Hochschule Augsburg; Adresse: Augsburg; Web-Link; 2018; Quellengüte: 3 (Vorlesung)

Kategorien:
Multimedia-Programmierung/Tutorium
Praktikum:MMProg
HTML5-Tutorium: JavaScript: Hello World
HTML5-Beispiel
Kapitel:Multimedia-Programmierung:Beispiele