LESS-CSS

Workshop

pagina GmbH, Tübingen, 18. Mai 2017

Less-Einführung

Was ist das?

Less erweitert CSS mit dynamischem Verhalten wie Variablen, Mixins, Berechnungen und Funktionen.

Less-Einführung

Warum für parsX?

- 2 parsX-Module mit CSS als Layoutsprache
- CSS hat einige Nachteile:
  - Kein Variablen-Konzept
  - Große Stylesheets = Wartungs-Alptraum
  - Komplexe Selektoren werden unübersichtlich
  - etc.
- EPUB-Format schränkt viele CSS 3-Funktionen ein

Less-Einführung

Vorteile für parsX

- Variablen erleichtern CSS-Pflege
- Verschachtelung vereinfacht Selektoren
- PrintCSS ohne Less wäre undenkbar

Less-Einführung

Warum Less?

- Less ist ein *CSS pre-processor* unter vielen
- oXygen-Unterstützung für Less-Stylesheets (ab 17.0)
- Einfache Implementierung in parsX
- Less arbeitet mit der gleichen Syntax wie CSS

  → *Jede CSS-Datei ist automatisch eine Less-Datei*

Less-Einführung

Hinweis

- Aktuelle Less-Version: 2.7.2
- Von oXygen und parsX unterstützte Version: 1.7.5
- *Für neuere Versionen gibt es leider (noch) keinen Java-Compiler*
- *Reicht aber für alle hier genannten Funktionen aus*

Grundfunktionen

Variablen

Variablen definieren häufig gebrauchte Werte an einer zentralen Stelle.
Somit muss man bei globalen Änderungen nur noch eine einzige Zeile ändern.

Less

```
@schmuckfarbe: #305983;

h1, h2 { color: @schmuckfarbe; }
```

CSS

```
h1, h2 {
  color: #305983;
}
```

Less

```
@rahmen: 1px solid #152638;

div.einschub { border: @rahmen; }
```

CSS

```
div.einschub {
  border: 1px solid #152638;
}
```

Grundfunktionen

Mixins 1/3

Mit *Mixins* kann man alle Werte einer Klasse an eine andere Klasse übergeben. Dafür muss man nur den Namen der Klasse angeben.

Less

```
// normaler Klassen-Selektor
.kleindruck {
  font-family: "DejaVuSerif", serif;
  font-size: 0.8em;
}


span.ziffer { .kleindruck; }

div.einschub.typ8 {
  .kleindruck;
  margin-left: 1em;
}
```

CSS

```
.kleindruck {
  font-family: "DejaVuSerif", serif;
  font-size: 0.8em;
}
span.ziffer {
  font-family: "DejaVuSerif", serif;
  font-size: 0.8em;
}
div.einschub.typ8 {
  font-family: "DejaVuSerif", serif;
  font-size: 0.8em;
  margin-left: 1em;
}
```

Grundfunktionen

Mixins 2/3

Ist der *Mixin* keine echte CSS-Klasse sondern dient nur zum Speichern eines *Code-Templates*, dann kann er durch ein Klammerpaar nach dem Selektor in der Ausgabe unterdrückt werden.

Less

```
// Mixin wird nicht ausgegeben
.kleindruck() {
  font-family: "DejaVuSerif", serif;
  font-size: 0.8em;
}

div.einschub.typ8 {
  .kleindruck;
  margin-left: 1em;
}
```

CSS

```
div.einschub.typ8 {
  font-family: "DejaVuSerif", serif;
  font-size: 0.8em;
  margin-left: 1em;
}
```

Grundfunktionen

Mixins 3/3

Mixins können sich auch wie Funktionen verhalten, also Parameter akzeptieren.

Less

```
// Mixin Funktion
.haengenderEinzug(@einzug: 1em) {
  text-indent: -@einzug;
  margin-left: @einzug;
}

div.inhalt li { .haengenderEinzug; }

div.einschub.literatur {
  .haengenderEinzug(30px);
}
```

CSS

```
div.inhalt li {
  text-indent: -1em;
  margin-left: 1em;
}
div.einschub.literatur {
  text-indent: -30px;
  margin-left: 30px;
}
```

Grundfunktionen

Verschachtelung 1/3

Anstatt lange Selektoren (wiederholt) zu schreiben um Vererbungen zu erzeugen, kann man in LESS Selektoren ineinander verschachteln.

Less

```
div.backlink {
  float: right;
  display: inline;

  a {
    color: @linkfarbe;
  }
}
```

CSS

```
div.backlink {
  float: right;
  display: inline;
}

div.backlink a {
  color: #0358b0;
}
```

Grundfunktionen

Verschachtelung 2/3

Mit dem `&`-Zeichen kann man verschachtelte Selektoren an das Elternelement direkt anhängen.

Less

```
div.backlink {
  float: right;
  display: inline;
  a {
    color: @linkfarbe;
    &:hover {
      color: @schmuckfarbe;
    }
  }
}
```

CSS

```
div.backlink {
  float: right;
  display: inline;
}
div.backlink a {
  color: #0358b0;
}
div.backlink a:hover {
  color: #305983;
}
```

Grundfunktionen

Verschachtelung 3/3

Less

```
/* KWIC-Index */
.index.ref_kwic {
  .ix-text {
    font-weight: bold;
  }
  &.referenzen_kapitel .ix-text {
    font-weight: normal;
  }
  .ix-referenz {
    font-weight: normal;
    .inline.reg-referenz,
    .ix-verweis {
      display: block;
      margin-left: 2em;
      text-indent: -1em;
      &.relevanz-hoch a {
```

CSS

```
/* KWIC-Index */
.index.ref_kwic.referenzen_kapitel .i
  font-weight: normal;
}
.index.ref_kwic .ix-referenz {
  font-weight: normal;
}
.index.ref_kwic .ix-referenz .inline
.index.ref_kwic .ix-referenz .ix-verw
  display: block;
  margin-left: 2em;
  text-indent: -1em;
}
```

```
.index.ref_kwic .ix-text {
  font-weight: bold;
}
```

Syntax-Infos

Kommentare

Less unterscheidet zwei Arten von Kommentaren

Less

```
/* Das ist ein normaler CSS
   Kommentar über mehrere Zeilen */
div.einschub.typ9 {
   color: black;
  // Nicht-persistenter Less-Kommenta
}

/* Einschub Typ10: Kleindruck */
div.einschub.typ10 {
  .kleindruck;
  clear: both;  // Hack wegen Bug #81
}
```

CSS

```
/* Das ist ein normaler CSS
   Kommentar über mehrere Zeilen */
div.einschub.typ9 {
  color: black;
}

/* Einschub Typ10: Kleindruck */
div.einschub.typ10 {
  font-family: "DejaVuSerif", serif;
  font-size: 0.8em;
  clear: both;
}
```

Syntax-Infos

Variablen-Geltungsbereich

Der Geltungsbereich von Variablen (»*Scope*«) ist abgeleitet von JavaScript und anderen Programmiersprachen. Zuerst wird versucht, die Variable in lokalem Kontext aufzulösen, danach im Übergeordneten.

Less

```less
@farbe: red;

body.kapitel {
  @farbe: white;
  h1 {
    color: @farbe;    // white
  }
}
```

Less

```less
@farbe: red;

body.kapitel {
  h1 {
    color: @farbe;   // white
  }
  @farbe: white;
}
```

Syntax-Infos

Escapen von Sonderzeichen

Manche Zeichen(folgen), wie z.B. den Doppelpunkt, kann man nicht in Variablen verwenden. Man muss sie zuvor escapen: `~"zeichenfolge"`

Less

```less
// Fehler weil der String : enthält
@komplex: div.einschub:not(.typ5);


// Funktioniert durch Escaping mit ~""
@komplex: ~"div.einschub:not(.typ5)";
```

Erweiterte Funktionen

Funktionen

Less enthält eine Reihe voreingebauter Funktionen die CSS noch lange nicht bieten wird. Diese sind vor allem im PrintCSS-Kontext nützlich.

- Textersetzung mittels `replace()` *
- Mathematische Rundungsfunktionen wie `ceil()`, `floor()`, etc. *
- Mathematische Funktionen wie `sqrt()`, `pow()`, `mod()`, etc. *
- Trigonometrische Funktionen wie `sin()`, `tan()`, `cos()`, etc. *
- *... und mit Less 2 noch viele weitere...*

→ ***Achtung:*** *Man darf dabei nicht vergessen, dass man nicht auf XML-Content zugreifen kann. Auch nicht mit CSS-Funktionen wie* `attr()`

Erweiterte Funktionen

Farbfunktionen

Ein Großteil der Less-Funktionen sind dem Thema »Farbe« gewidmet.

- Steuerung der Sättigung mittels `saturate()` + `desaturate()` *
- Steuerung der Helligkeit mittels `lighten()` + `darken()` *
- Mischen von zwei Farben mittels `mix()`, `tint()` + `shade()` *
- Überlagern von zwei Farben mit `multiply()`, `overlay()`, `difference()`, `average()`, etc.* ähnlich wie in Photoshop

→ *Weitere Infos dazu in der Less-Doku: Color Operations*

Erweiterte Funktionen

Parent-Selektor 1/2

Der *Parent*-Selektor `&` kann sehr vielfältig eingesetzt werden:

Less

```
.einschub { &.typ1 { color: black; }
.u-block { &1 { font-size: 2em; } }
.einschub { & > p { margin: 0; } }
```

CSS

```
.einschub.typ1 { color: black; }
.u-block1 { font-size: 2em; }
.einschub > p { margin: 0; }
```

```
.fett {
  font-weight: bold;
  & .kursiv { font-style: italic; }
  .kursiv & { color: red; }
  & & { font-weight: 900; }
}
```

```
.fett { font-weight: bold; }
.fett .kursiv { font-style: italic;
.kursiv .fett { color: red; }
.fett .fett { font-weight: 900; }
```

Erweiterte Funktionen

Parent-Selektor 2/2

Kombinatorische Explosion! 💥

Less

```
p, a, ul, li {
  // Genereller Abstand nach oben
  margin-top: 1em;
  & + & {
    // Aber KEIN Abstand bei
    // aufeinanderfolgenden Elementer
    margin-top: 0;
  }
}
```

CSS

```
p, a, ul, li {
  margin-top: 1em;
}
p + p, p + a, p + ul, p + li,
a + p, a + a, a + ul, a + li,
ul + p, ul + a, ul + ul, ul + li,
li + p, li + a, li + ul, li + li {
  margin-top: 0;
}
```

Erweiterte Funktionen

Variablen

Variablen können auch zum Speichern komplexer Selektoren verwendet werden.
Dann muss die Variable aber mit `@{var}` aufgerufen werden.

Less

```
@typ: div.liste_geordnet ol.listType_

@{typ}_upper-A {
  list-style: upper-latin;
}
@{typ}_lower-i {
  list-style: lower-roman;
}
```

CSS

```
div.liste_geordnet ol.listType_upper-
  list-style: upper-latin;
}
div.liste_geordnet ol.listType_lower-
  list-style: lower-roman;
}
```

Last but not least

- Less Features und Beispiele
- Funktionsreferenz
  - parsX-Doku

</parsX-Workshops>

Vielen Dank für Ihre Aufmerksamkeit!