Ihr Webentwickler
Inhaber: Gunnar von Spreckelsen
www.ihrwebentwickler.de

service@ihrwebentwickler.de
Tel. 01525 3479709

Frontend-Methodik BEM
1. Ausgabe Februar 2014

BEM steht für eine interessante Denkweise bei der Produktion von Frontend-Schnittstellen, diese Methode wird im folgenden kurz beschrieben.

Inhalts-Verzeichnis

1.) Begrifflichkeit Element und Block

BEM steht als Abkürzung für “Block”, “Element” und “Modifier” und beruht auf der Philosophie, die klassische objektorientierte Programmierung bis zu einem gewissen Rahmen auf den Bereich der Frontend-Entwicklung zu übertragen. Dabei wird ein Webdesign in unterschiedliche Blöcke zerlegt. Eine klassische Webseite würde z.B. in die Bereiche Header, Content und Footer aufgeteilt. Der Block „Header“ kann dann weitere Blöcke wie beispielsweise Logo oder Navigation enthalten. In die objektorientierte Sprache übertragen stellt dann ein Block eine unabhängige Einheit dar. Elemente sind Bestandteil eines Blockes und zeichnen sich dadurch aus, in Abhängigkeit zu einem Block zu stehen und eine eigene Funktion zu besitzen.

2.) BEM-Beispiel mit JSON

Die Methodik wird anhand eines JSON-Objektes klarer:

```
var page = {
            block: 'page',
            content: {
                block: 'head',
                content: [
                    {
                        block: 'menu'
                    },
                    {
                        content: {
                            block: 'logo'
                        },
```

```
                    elem: 'column'
                }]
            }
        }
```

Der Haupt-Block **page** enthält den Block head. Im Block **head** ist wiederum ein Block **menu** und **logo** enthalten. Das Element **column** ist funktionell abhängig vom block **logo.**

3.) Blöcke zur Definition des Webseiten-Aufbaus

Blöcke spiegeln den Aufbau einer Webseite wieder, damit alle Projektbeteiligten eine gemeinsame Sprache sprechen können. Wenn der Projektmanager zum Beispiel vom Header spricht, dann verstehen der Designer und Entwickler den Hintergrund, dass der Kopfbereich der Seite weiterentwickelt werden soll. Der Block bildet immer eine eindeutige unabhängige Einheit. Eine Einheit kann je nach Arrangement der Webseite aus mehreren Ausgaben bestehen. Die Ausgabe einer Blogseite enthält z.B. den Block blogentry. Der Block blogentry wird ebenfalls mehrmals mit den Einträgen ausgegeben, dennoch bildet er eine logische eindeutige unabhängige Einheit.

4.) Namens-Konvention für CSS und Templates

Die Wahl der Blocknamen muss berücksichtigen, dass auch in großen Projekten mit vielen unterschiedlichen Formatierungen und Funktionalitäten eindeutige systemweite Namen verwendet werden.

Für unabhängiges CSS gilt:

- Ein Block spiegelt sich im CSS für die Rolle im gleichen Namen wieder

- HTML-Elemente sind zu vermeiden, da sie nicht kontextunabhängig sind

- kaskadierende Selektoren über unterschiedliche Blöcke sind zu vermeiden

Beispiele:

- .logo ist eindeutig und eine gültige Bezeichnung.

- .header table td ist nicht gültig, da die gemischte Form mit HTML-Elementen nicht kontextunabhängig ist.

- .logo .first_colum über zwei Blöcke ist als Mischform ebenfalls nicht eindeutig.

Die Namensgebung einer CSS-Rolle ergibt sich aus dem Namen des Blockes, bei der Verwendung von Elementen folgen eindeutige Trennzeichen plus dem Namen des Elementes.

Im Beispiel .logo__fullimage bildet logo den Block bzw. die CSS-Klasse, die beiden Unterstriche dienen als Trennzeichen zum Element fullimage.

In Templates wie z.B. in beschreibenden XSLT-Transformationen sind ebenfalls die eindeutigen Namen für die Blöcke und Elemente zu verwenden:

```
<xsl:template match="b:menu">
  <ul class="menu">
    <xsl:apply-templates/>
  </ul>
</xsl:template>
```

5.) Block-Wiederholungen im Detail

Ein Block, der eine einzelne Einheit bildet, kann beliebig oft wiederholt werden. Ein Block, der Bestandteil eines anderen Blockes ist, bildet deshalb keine einzige Einheit zu seinem Vater-Element, von dem er abhängig ist.

Beispiel:

Bei .header .menu ist menu keine einzelne Einheit, da dieser Block von header abhängig ist.

Eindeutige Blöcke können innerhalb der Seite beliebig oft verwendet werden, aus diesem Grund ist die Verwendung von ID-Selektoren nicht gestattet. Aus Sicht der JavaScript-Umgebung wird die Funktionalität eindeutig auf alle vorhandenen Klassen-Selektoren angewendet. Damit ist die abgebildete Funktionalität auf diese Klassen-Selektoren eindeutig definiert. Diese Abbildung wird mit Block-Konsistenz bezeichnet, es muss gewährleistet sein, dass alle Bestandteile der Umgebung (Design und Funktionalität) in ihrem Verhalten eindeutig durch den Zugriff auf eindeutige Blöcke funktionieren.

6.) Modifier

Es wird langsam Zeit, auf die Modifier zu sprechen zu kommen, die Teil der Namensgebung von BEM sind. Die Verwendung und das dahinterstehende Prinzip sind schnell erklärt: In Template-, CSS- oder Javascript-Umgebungen wie JSON-Objekten werden oft wiederverwendete Zusatz- Blöcke verwendet. Diese Blöcke werden Modifier genannt, weil sie zwar ein eigenständiges Verhalten bzw. eine eindeutige Funktionalität zur Verfügung stellen, aber schlichtweg mehrmals innerhalb einer Umgebung verwendet werden.

Hier folgt ein vereinfachtes klassisches Beispiel einer CSS-Formatierung für aktive Menüitems:

```
.active {
 background: red;
}
```

```
<ul>
    <li class="active">aktives Menü</li>
    <li>nicht-aktives Menü</li>
</ul>
```

Der Modifier active wird mehrmals z.B. in unterschiedlichen Menüs verwendet und stellt anderseits seine eindeutige Funktionalität zur Formatierung von aktiven Menüs zu Verfügung, er verändert/ modifiziert die Umgebung seiner Verwendung.

7.) Dateisystem-Repräsentation

Die Darstellung der Blöcke über die Namensgebung von Verzeichnissen und Dateien erfolgt nach dem Schema, jeweilige Unterverzeichnisse mit dem Blocknamen zu verwenden und alle einzelnen Dateien nach dem Blocknamen zu benennen. Elemente und Modifier werden dort nach ihren Namen in Unterverzeichnissen gespeichert.

```
menu/
  menu_item.css
  menu_item.js
  menu_item.xls
_state/
  state.js
```

Durch den vorgestellten Unterstrich bei _state ist direkter ersichtlich, dass es sich hierbei um ein Element handelt.

8.) Quellen

http://www.smashingmagazine.com/front-end-methodology-bem-file-system-representation/