Fachhochschule Bielefeld
Lehrgebiet Softwaretechnik und Programmierung
Prof. Dr. Jörg Brunsmann

Übungen zur Vorlesung

Spezielle Gebiete zum Software Engineering

Sommersemester 2015

Aufgabenblatt 3: Moderne Entwicklungswerkzeuge

Aufgabe 3.1: Anwendung von yeoman-Generatoren (5 Punkte)

Installieren Sie das Werkzeug *yeoman (*http://yeoman.io/) und installieren Sie den Generator „reveal“, den Sie unter der URL

```
https://github.com/slara/generator-reveal
```

finden. Reveal.js ist ein Framework zur Erstellung von Präsentationen in HTML. Erzeugen Sie mit Hilfe des Generators eine Präsentation, in der Sie Ihr Verständnis über die Verwendung von modernen Entwicklungswerkzeugen darstellen. Stellen Sie zum Beispiel anhand des Lebenszyklus einer Anwendung (z.B. Projektstart, Implementierung, Test, Deployment) dar, in welche Phase die einzelnen Werkzeuge fallen und welchen Zweck sie erfüllen.

Aufgabe 3.2: Anwendung von gulp.js (3 Punkte)

Machen Sie sich mit der API von gulp.js vertraut, die unter

```
https://github.com/gulpjs/gulp/blob/master/docs/API.md
```

zu finden ist. Installieren Sie relevante gulp-Plugins und schreiben Sie ein *gulpfile.js*-File, das folgende Aufgaben bearbeitet:

- Kopieren der Datei *index.html* in das Verzeichnis “copy”
- Kompression aller Bilder und Vektoren mit *gulp-imagemin*
- Minifizierung von HTML-Dateien mit *gulp-minify-html*
- Überprüfung von Javascript-Dateien mit *gulp-jshint*
- Minifizierung von Javascript-Dateien mit *gulp-uglify* und Zusammenfassen von Javascript-Dateien mit *gulp-concat*
- Transpilierung von ES6-Dateien mit *gulp-babel*
- Konvertierung der Sass-Dateien in CSS-Dateien mit *gulp-sass*
- Anpassen der Prefixes in CSS-Dateien mit *gulp-autoprefixer*
- Minifizierung der CSS-Dateien mit *gulp-minify-css*
- Hinzufügen eines Copyright-Texts mit *gulp-header*
- Kopieren aller Dateien in einen Ordner „dist“ für den Upload
- Überwachung von [*es6|*.js|*.scss]-Dateien auf mögliche Änderungen
- Ausführen der Aufgaben in einem 'default'-Task

Aufgabe 3.3: Anwendung von mocha.js bzw. Jasmine (2 Punkte)

Gegeben sei folgender Javascript-Code:

```
var Converter = (function() {

  function Converter() {}

      Converter.prototype.rgbToHex = function(red, green, blue) {
         var redHex = red.toString(16);
         var greenHex = green.toString(16);
         var blueHex = blue.toString(16);
         return pad(redHex) + pad(greenHex) + pad(blueHex);
       };

       function pad(hex) {
         return (hex.length === 1 ? "0" + hex : hex);
       }

      Converter.prototype.hexToRgb = function(hex) {
         var red = parseInt(hex.substring(0, 2), 16);
         var green = parseInt(hex.substring(2, 4), 16);
         var blue = parseInt(hex.substring(4, 6), 16);
         return [red, green, blue];
       };

      return Converter;
})();
```

Der obige Javascript-Code wandelt RGB-Farbwerte von Dezimal nach Hexadezimal und umgekehrt und wird wie folgt verwendet:

```
var c = new Converter();
c.rgbToHex(244, 255, 244);
c.hexToRgb("FFEEFF");
```

Installieren Sie mocha.js (http://mochajs.org/) oder Jasmine (http://jasmine.github.io/) und schreiben Sie Unit-Tests, die die zwei relevanten API-Funktionen *rgbToHex()* und *hexToRgb()* funktional testen.