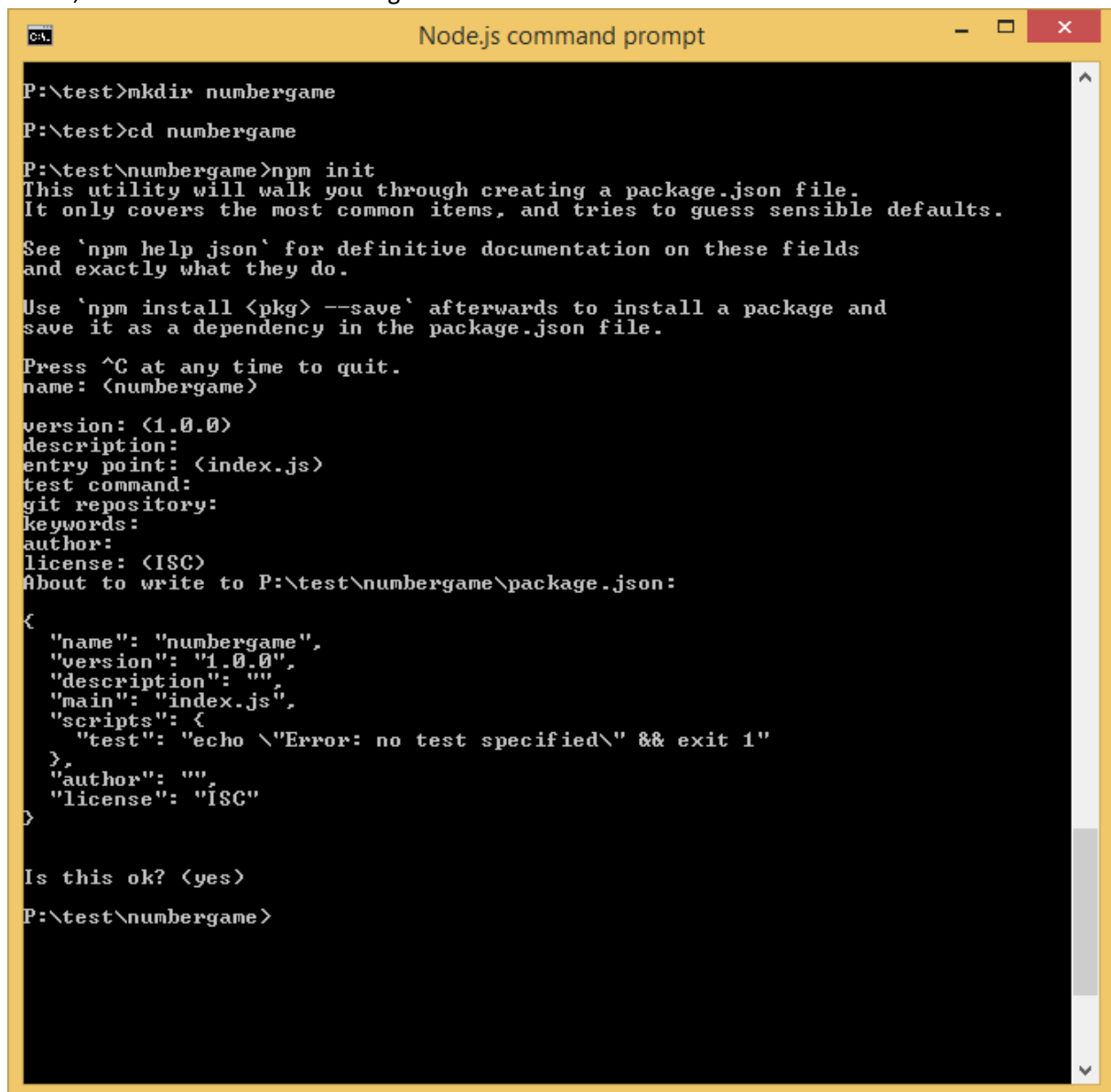


Zahlen Ratespiel

Übungsbeispiel für AngularJS

1. Anlegen eines neuen Projektes mit npm

Legen Sie einen neuen Ordner an, den Sie „numbergame“ nennen. Öffnen Sie eine Kommando Zeile und legen sie mittels „npm init“ ein neues Projekt an. NPM wird sie um ein paar Informationen bitten, die sie Enter einfach bestätigen können.



```
Node.js command prompt

P:\test>mkdir numbergame
P:\test>cd numbergame
P:\test\numbergame>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help json' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg> --save' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (numbergame)

version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to P:\test\numbergame\package.json:
{
  "name": "numbergame",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this ok? (yes)
P:\test\numbergame>
```

2. Installieren von AngularJS, Bootstrap und ExpressJS

Installieren sie mittels npm Angularjs (Paketname „angular“) und Bootstrap (Paketname „bootstrap“) und ExpressJS (Paketname „express“)

```
Node.js command prompt

P:\test\numbergame>npm install angular
npm WARN package.json numbergame@1.0.0 No description
npm WARN package.json numbergame@1.0.0 No repository field.
npm WARN package.json numbergame@1.0.0 No README data
angular@1.4.7 node_modules\angular

P:\test\numbergame>npm install bootstrap
npm WARN package.json numbergame@1.0.0 No description
npm WARN package.json numbergame@1.0.0 No repository field.
npm WARN package.json numbergame@1.0.0 No README data
bootstrap@3.3.5 node_modules\bootstrap

P:\test\numbergame>npm install express
npm WARN package.json numbergame@1.0.0 No description
npm WARN package.json numbergame@1.0.0 No repository field.
npm WARN package.json numbergame@1.0.0 No README data
express@4.13.3 node_modules\express
├── escape-html@1.0.2
├── merge-descriptors@1.0.0
├── cookie@0.1.3
├── array-flatten@1.1.1
├── cookie-signature@1.0.6
├── content-type@1.0.1
├── content-disposition@0.5.0
├── range-parser@1.0.2
├── fresh@0.3.0
├── vary@1.0.1
├── etag@1.7.0
├── serve-static@1.10.0
├── methods@1.1.1
├── path-to-regexp@0.1.7
├── utils-merge@1.0.0
├── parseurl@1.3.0
├── depd@1.0.1
├── qs@4.0.0
├── finalhandler@0.4.0 <unpipe@1.0.0>
├── on-finished@2.3.0 <ee-first@1.1.1>
├── debug@2.2.0 <ms@0.7.1>
├── proxy-addr@1.0.8 <forwarded@0.1.0, ipaddr.js@1.0.1>
├── type-is@1.6.9 <media-typer@0.3.0, mime-types@2.1.7>
├── accepts@1.2.13 <negotiator@0.5.3, mime-types@2.1.7>
└── send@0.13.0 <destroy@1.0.3, statuses@1.2.1, ms@0.7.1, mime@1.3.4, http-errors@1.3.1>

P:\test\numbergame>
```

3. Konfigurieren des Express WebServers

Um unsere Webseite auszuliefern benötigen wir einen lokalen Webserver. Legen sie dafür eine Datei mit dem Namen server.js an und fügen Sie folgenden Inhalt ein:

server.js angularjs\assignments\numbergame

```
1 var express = require('express');
2
3 var app = express();
4 app.use(express.static('./'));
5
6 var server = app.listen(3000, function () {
7   var host = server.address().address;
8   var port = server.address().port;
9   console.log('Example app listening at http://%s:%s', host, port);
10 });
```

Starten Sie anschließend den webserver, indem Sie im Kommandofenster „node server.js“ aufrufen:

```
P:\test\numbergame>node server.js
Example app listening at http://:::3000
```

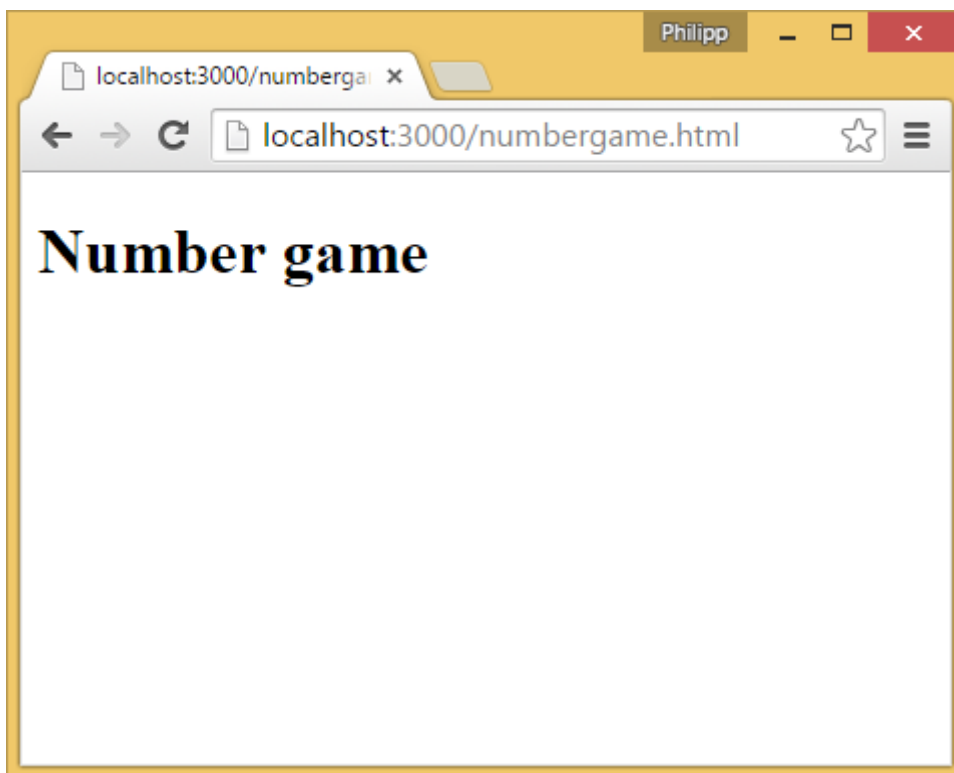
Legen sie nun die Hauptseite unserer Applikation an, und testen Sie ob sie diese im Web-browser abfragen können. Dafür legen Sie eine Datei, numbergame.html, im Hauptordner (numbergame) an und befüllen Sie, sie mit folgendem Inhalt:

numbergame.html angularjs\assignments\numbergame

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4   </head>
5
6   <body>
7     <h1>Number game</h1>
8   </body>
9 </html>
```

Öffnen Sie danach ihren Webbrowser und navigieren Sie zur Seite

<http://localhost:3000/numbergame.html>



Gratulation, Sie können erfolgreich auf den Webserver zugreifen und die HTML Datei für die Übung laden.

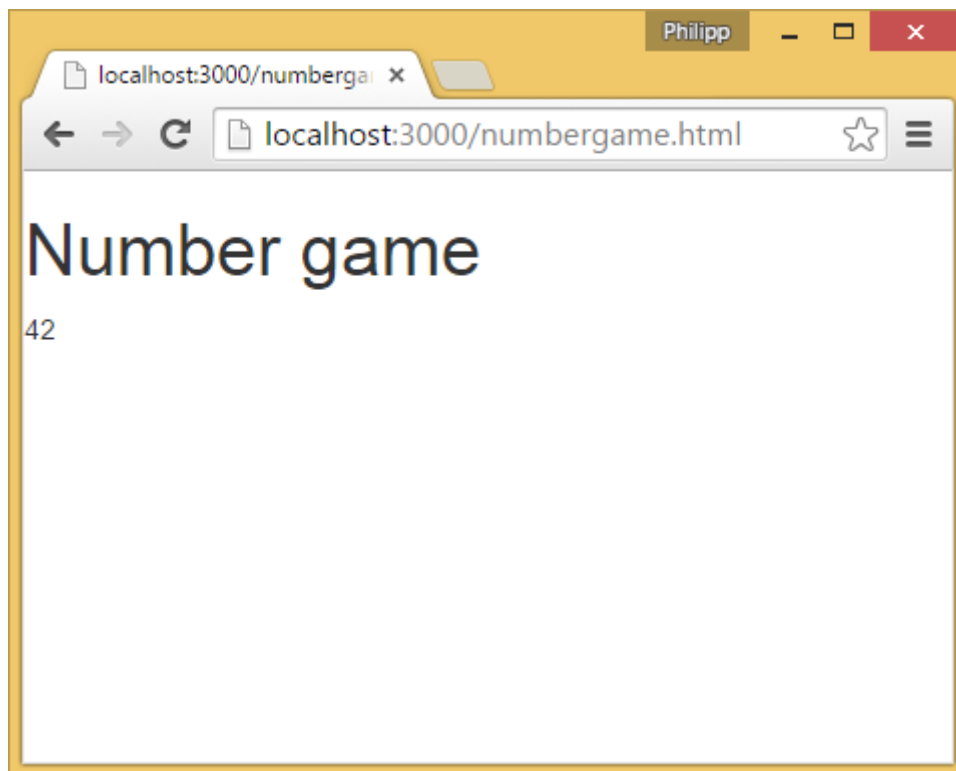
4. Einrichten von AngularJS und Bootstrap

Öffnen Sie die numbergame.html Datei und ändern Sie den Inhalt des <head> Elementes, so dass dieser AngularJS und Bootstrap referenziert. Weiters fügen wir dem <html> element ein ng-app attribut hinzu und erweitern das <body> mit einem AngularJS Ausdruck um zu überprüfen ob AngularJS korrekt eingerichtet ist:

numbergame.html angularjs\assignments\numbergame

```
1 <!DOCTYPE html>
2 <html ng-app>
3   <head>
4     <script src="../../node_modules/angular/angular.js"></script>
5     <link rel="stylesheet" href="../../node_modules/bootstrap/dist/css/bootstrap.css">
6   </head>
7
8   <body>
9     <h1>Number game</h1>
10
11     <div>{{40 + 2}}</div>
12   </body>
13 </html>
```

Nach dem Speichern der Änderungen, aktualisieren wir den Inhalt des Webbrowsers (F5) und finden folgende Seite vor.



Super! Da der AngularJS Ausdruck `{{40 + 2}}` korrekt ausgewertet wird, können wir erkennen das AngularJS korrekt funktioniert.

5. Eigener Controller

In diesem Schritt legen wir das Applikationsmodul und den Controller für unser Beispiel an. Erstellen sie dafür eine neue Datei mit dem Namen „numbergame.js“ neben der dazugehörigen html Datei (numbergame.html) an und befüllen Sie sie mit folgendem Inhalt:

numbergame.js angularjs\assignments\numbergame

```
1 var app = angular.module('numbergameApp', []);
2 app.controller('numbergameController', ['$scope', function($scope) {
3 }]);
```

Damit die Datei geladen wird, fügen wir im <head> Element eine Referenz auf die numbergame.js Datei hinzu. Des Weiteren ändern wir das ng-app Attribut auf den Namen unseres Applikationsmoduls („numbergameApp“) und fügen beim <body> Element ein ng-controller Attribut mit dem Namen unseres Controllers („numbergameController“) hinzu:

numbergame.html angularjs\assignments\numbergame

```
1 <!DOCTYPE html>
2 <html ng-app="numbergameApp">
3   <head>
4     <script src="../../node_modules/angular/angular.js"></script>
5     <link rel="stylesheet" href="../../node_modules/bootstrap/dist/css/bootstrap.css">
6     <script src="numbergame.js"></script>
7   </head>
8
9   <body ng-controller="numbergameController">
10     <h1>Number game</h1>
11
12     <div>{{40 + 2}}</div>
13   </body>
14 </html>
```

Diesen Schritt überprüfen Sie am besten durch Aktualisieren der Webseite (wie bereits in Schritt 4), öffnen mit F12 die Entwickler-tools Ihres Browsers und überprüfen ob Sie eventuell in der Konsole Fehlermeldungen entdecken können.

6. Grundstruktur der Applikation

Im nächsten Schritt legen die die Grundlegenden UI Elemente an. Ganz oben wollen wir dem Benutzer mit einer Überschrift (<h1>) erklären wo er sich gerade befindet und Ihm darunter das Spiel mit einer kleinen Anleitung erläutern.

Darunter platzieren wir die Kernelemente des Spiels: das Eingabefeld für den Rate versuch und einen Button um einen Versuch auszulösen. Diese Elemente werden durch das <input> und <button> Element abgebildet. Die anderen <div>s und das <form> Element dienen der visuellen Struktur. Um den Inhalt etwas vom Rand des Fensters zu trennen, verleihen wir dem <body> Element die Bootstrap container Klasse:

numbergame.html angularjs\assignments\numbergame

```
1 <!DOCTYPE html>
2 <html ng-app="numbergameApp">
3   <head>
4     <script src="../../node_modules/angular/angular.js"></script>
5     <link rel="stylesheet" href="../../node_modules/bootstrap/dist/css/bootstrap.css">
6     <script src="numbergame.js"></script>
7     <style>
8       .top-buffer { margin-top:20px; }
9     </style>
10  </head>
```

```

10     </head>
11
12     <body class="container" ng-controller="numbergameController">
13         <h1>Number game</h1>
14
15         <div>
16             Guess a number between 0 and 100
17         </div>
18
19         <div>
20             Number of tries: {{guessCount}}
21         </div>
22         <div class="top-buffer"></div>
23
24         <form class="form-inline">
25             <div class="form-group">
26                 <input class="form-control" type="number" ng-model="currentGuess">
27             </div>
28             <button class="btn btn-primary" ng-click="check()">Check</button>
29         </form>
30     </body>
31 </html>

```

Nach einem Aktualisieren der Seite können wir die Grundelemente wiedererkennen. Aber auch, dass uns gewisse Elemente fehlen, wie z.B. {{guessCount}}.

7. Spiel Logik

In Schritt 6 haben wir die HTML Struktur um die Bedienelemente für unser Spiel erweitert. Die Logik dafür fügen wir nun in den Controller ein. Es gibt hier zwei Bereiche: Einerseits den Initialisierungsteil, der beim Anlegen des Controllers ausgeführt wird, andererseits die check Methode die bei jedem Versuch ausgeführt wird und die eigentliche Spiel Logik beinhaltet. Im Initialisierungsteil verwenden wir das Math Objekt um einen neue Zufallszahl zwischen 0 und 100 anzulegen und setzen currentGuess, result und guessCount auf ihre entsprechenden Initialwerte.

Die check-Methode vergleicht die aktuelle Eingabe mit der anfangs generierten Zufallszahl und setzt die result Variable entsprechend:

numbergame.js angularjs\assignments\numbergame

```
1 var app = angular.module('numbergameApp', []);
2 app.controller('numbergameController', ['$scope', function($scope) {
3
4     $scope.theNumber = Math.round(Math.random() * 100);
5     $scope.currentGuess = null;
6     $scope.result = null;
7     $scope.guessCount = 0;
8
9     $scope.check = function() {
10         if ($scope.currentGuess > $scope.theNumber) {
11             $scope.result = 'toohigh';
12         } else if ($scope.currentGuess < $scope.theNumber) {
13             $scope.result = 'toolow';
14         } else if ($scope.currentGuess == $scope.theNumber) {
15             $scope.result = 'correct';
16         }
17
18         $scope.currentGuess = null;
19         $scope.guessCount++;
20     }
21 }]);
```

Nach dem Aktualisieren können wir das Spiel spielen und im Debugger unser Ergebnis sehen (ev. Breakpoint in der check Methode setzen). Nun fehlt und noch eine Darstellung des Ergebnisses in der HTML Seite.

8. Anzeigen des Ergebnisses

Schließlich führen wir noch zwei weitere <div>s in die Seite ein, die mittels ng-show nur dann angezeigt werden, wenn bestimmte Bedingungen erfüllt sind:

numbergame.html angularjs\assignments\numbergame

```
1 <!DOCTYPE html>
2 <html ng-app="numbergameApp">
3     <head>
4         <script src="../node_modules/angular/angular.js"></script>
5         <link rel="stylesheet" href="../node_modules/bootstrap/dist/css/bootstrap.css">
6         <script src="numbergame.js"></script>
7         <style>
8             .top-buffer { margin-top:20px; }
9         </style>
10    </head>
11
12    <body class="container" ng-controller="numbergameController">
13        <h1>Number game</h1>
14
15        <div>
16            Guess a number between 0 and 100
17        </div>
18
19        <div>
20            Number of tries: {{guessCount}}
21        </div>
22        <div class="top-buffer"></div>
```

```

23
24     <form class="form-inline">
25         <div class="form-group">
26             <input class="form-control" type="number" ng-model="currentGuess">
27         </div>
28         <button class="btn btn-primary" ng-click="check()">Check</button>
29     </form>
30 <div class="top-buffer"></div>
31
32 <div class="alert alert-danger" ng-show="result && result != 'correct'">
33     You entered a number that is
34     <span ng-show="result == 'toohigh'">too high! Pick a smaller number!</span>
35     <span ng-show="result == 'toolow'">too low! Pick a higher number!</span>
36 </div>
37
38 <div class="alert alert-success" ng-show="result && result == 'correct'">
39     Congratulations, you entered the correct number: {{theNumber}} !
40 </div>
41 </body>
42 </html>

```

Ein weiteres Aktualisieren des Webbrowsers und Sie können Ihr spiel genießen ☺

