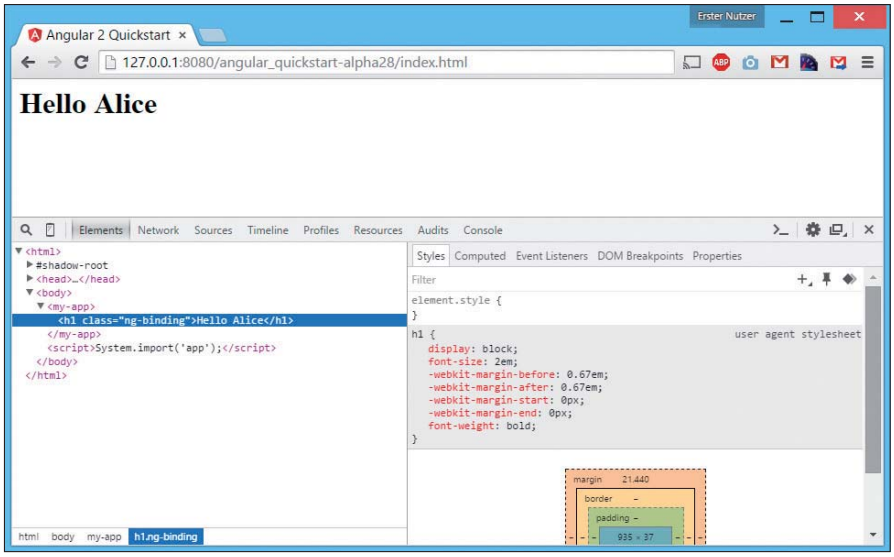
ANGULAR 2.0 UND MODULARER CODE

Modularer Code

Die Vorschauversion von AngularJS zeigt ein solides Management von Abhängigkeiten.

Angular 2.0 wird in naher Zukunft fertiggestellt sein. Es gibt es bereits regelmäßige Vorabversionen für interessierte Entwickler. Das Angular-Team hat sich entschieden, alte Zöpfe rigoros abzuschneiden und ein komplett überarbeitetes Framework zu entwickeln. Die neue Version bricht mit bestehenden Konzepten – was für viel Aufregung gesorgt hat. Die Template-Syntax ist neu, und man setzt nun Komponenten statt Controller ein. Auch der Einsatz von TypeScript rüttelt am Althergebrachten. In diesem Artikel soll auf eine weitere maßgebliche Änderung eingegangen werden. Diese betrifft das Laden von JavaScript-Dateien. Weitere Artikel zu den Neuerungen in Angular 2.0 folgen in den kommenden Ausgaben der **web & mobile developer**.



Der Output im Browser (Bild 1)

Hello World

Das hier gezeigte Beispiel nutzt eine Vorschauversion von Angular 2.0. Der gezeigte Code muss für spätere Versionen wahrscheinlich angepasst werden [1].

Listing 1: index.html

```
<html>
  <head>
    <title>Angular 2.0 Quickstart</title>
    <!-- Zeile 1 --> <script src="https://github.
    jspm.io/jmcriffey/bower-traceur-runtime@0.0.87/
    traceur-runtime.js"></script>
    <!-- Zeile 2 --> <script src="https://jspm.io/
    system@0.16.js"></script>
    <!-- Zeile 3 --> <script src="https://code.
    angularjs.org/2.0.0-alpha.28/angular2.dev.js">
    </script>
  </head>
  <body>
    <my-app></my-app>
    <!-- Zeile 4 --> <script>System.import('app');
    </script>
  </body
```

Auf der neuen Website unter Angular.io findet man einen kurzen 5-Minuten-Schnellstart in das neue Framework [2]. In dem Quickstart wird unter anderem beschrieben, wie man eine erste Komponente erstellt. Ebenso wird der Transpiler TypeScript vorgestellt, der die Datei *app.ts* in eine JavaScript-Datei namens *app.js* umwandelt. Listing 1 und Listing 2 zeigen das Schnellstart-Beispiel von Angular 2.0. Vier Zeilen sollen im Folgenden genauer betrachtet werden.

Das Beispiel baut auf einer Reihe von Frameworks auf, um diese Datei zu laden und auszuführen. Die eigentliche Funktion dieses Beispiels erschließt sich aber dennoch schnell. Das DOM-Element *<my-app>* wird mit einer Überschrift ergänzt, die den Text *Hello Alice* trägt (Bild 1).

Hinter den Zeilen *<!-- Zeile 1 -->* bis *<!-- Zeile 4 -->* aus Listing 1 verbirgt sich ein Strauß an Technologien – unter anderem Traceur, jspm, SystemJS, TypeScript und natürlich Angular 2.0. Durch die Auswahl dieser Frameworks ist es möglich, bereits mit heutigen Browsern eine Anwendung auf Grundlage von ECMAScript 6 zu entwickeln. Die verwendeten Tools sollen nun betrachtet werden. Alle gezeigten Kommandozeilen-Befehle setzen voraus, dass Node.js installiert ist.

ES6 Module Loader Polyfill

In der Webwelt steht der Begriff Polyfill für eine Software, die fehlende JavaScript-Funktionalitäten im Browser zur Verfügung stellt. In der Vergangenheit ging es bei Polyfills häufig darum, standardisierte Funktionen in alten Internet-Explorer-Versionen nachzurüsten. Es können aber auch mit Hilfe