1 Haben Sie alles, was Sie benötigen?

»Angular's best feature is its community.«

Dave Geddes
(Organisator der Konferenz ng-conf)

Bevor wir beginnen, möchten wir sicherstellen, dass Sie für die Entwicklung mit Angular bestmöglich gewappnet sind. Darum widmen wir uns zunächst der Einrichtung aller erforderlichen Werkzeuge. Wir geben Ihnen in diesem Kapitel außerdem Tipps zur Konfiguration der Programme mit.

Falls Sie bereits über eine integrierte Entwicklungsumgebung (IDE) für die Webentwicklung verfügen und mit Begriffen wie *Node.js* und *NPM* vertraut sind, können Sie dieses Kapitel überspringen.

1.1 Visual Studio Code

Unser empfohlener Editor für Angular

Visual Studio Code (VS Code)[1] ist eine quelloffene Entwicklungsumgebung unter der MIT-Lizenz. Der Editor ist unter den Betriebssystemen Windows, macOS und Linux lauffähig. VS Code lässt sich leicht durch neue Pakete erweitern und verändern. Weiterhin verfügt der Editor nicht nur über eine moderne Oberfläche, sondern unterstützt auch zahlreiche Programmiersprachen. Da die TypeScript-Integration sehr ausgereift ist, verwenden wir diesen Editor für die Entwicklung von Angular-Anwendungen. VS Code bringt weiterhin eine sehr gute automatische Codevervollständigung und Codedokumentation mit sich. Außerdem besitzt der Editor von Haus aus eine *Git*-Integration.[2] Somit lassen sich Änderungen am Projektcode über die grafische Oberfläche des Editors gut nachvollziehen.

[1] https://ng-buch.de/x/3 – Visual Studio Code

[2] Git ist ein Tool zur Versionsverwaltung von Quellcode.

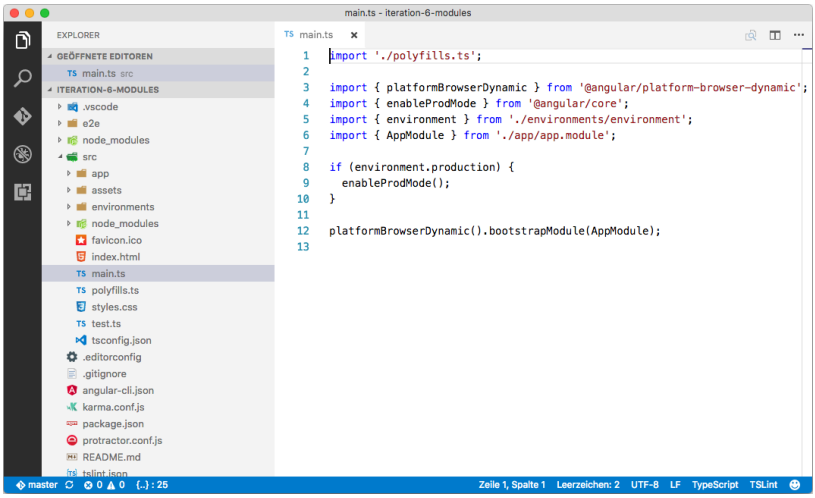

Abb. 1–1
Die Oberfläche von Visual Studio Code

Erweiterungen für VS Code

Wir empfehlen zusätzlich noch die Installation einiger Erweiterungen (*Extensions*). Mit Erweiterungen lassen sich Funktionalitäten des Editors optimal ausnutzen und die Produktivität bei der Entwicklung mit Angular kann gesteigert werden. Erweiterungen für VS Code können über den *Marketplace* von Visual Studio bezogen werden.[3] Die Installation der Plugins erfolgt am einfachsten über den in den Editor integrierten *Extensions Browser* (Abbildung 1–2). Hier können wir den

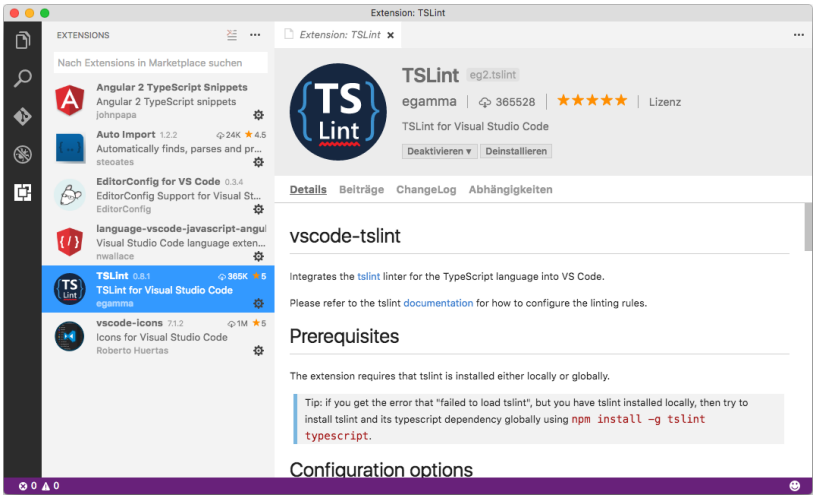

Abb. 1–2
Erweiterungen in VS Code

[3] https://ng-buch.de/x/4 – »Extensions for the Visual Studio family of products«

Erweiterung	Kurzbeschreibung
Auto Import[4]	Generiert automatisch Import-Anweisungen für TypeScript
EditorConfig for VS Code[5]	Verarbeitet Informationen einer `.editorconfig`-Datei und konfiguriert entsprechend den Editor. Somit können editorenübergreifende Einstellungen für die Anzahl von Leerzeichen, verwendete Zeichencodierung etc. geschaffen werden.
TSLint[6]	Ist eine Integration des Tools TSLint[7]. Verstöße gegen festgelegte Codestil-Richtlinien (festgelegt in der Datei `tslint.json`) werden grafisch im Editor dargestellt.
angular2-inline[8]	Darstellung von Syntax-Highlighting bei Verwendung von Inline-Templates bzw. -CSS innerhalb von Komponenten in Angular
vscode-icons[9]	Verbessertes Iconset für Dateien und Ordner im Dateibaum

Tab. 1–1 *Empfohlene Erweiterungen für Visual Studio Code*

Marketplace nach Plugins durchsuchen und die installierten Erweiterungen verwalten.

Tabelle 1–1 zeigt eine Liste von Erweiterungen, die wir für die Entwicklung mit Angular empfehlen. Alle Erweiterungen lassen sich mit folgenden Befehlen installieren:

```
$ code --install-extension steoates.autoimport
$ code --install-extension EditorConfig.EditorConfig
$ code --install-extension eg2.tslint
$ code --install-extension natewallace.angular2-inline
$ code --install-extension robertohuertasm.vscode-icons
```

[4] https://ng-buch.de/x/5 – Visual Studio Code: Auto Import
[5] https://ng-buch.de/x/6 – Visual Studio Code: EditorConfig for VS Code
[6] https://ng-buch.de/x/7 – Visual Studio Code: TSLint
[7] https://ng-buch.de/x/8 – TSLint
[8] https://ng-buch.de/x/9 – Visual Studio Code: angular2-inline
[9] https://ng-buch.de/x/10 – Visual Studio Code: vscode-icons

1.2 Google Chrome mit Augury

Zur Darstellung der Angular-Anwendung und für das Debugging nutzen wir Google Chrome[10]. Wir setzen auf diesen Browser, weil er bereits ein umfangreiches Set an Debugging-Tools mitbringt. Diese *Chrome Developer Tools* schauen wir uns im Powertipp ab Seite 157 genauer an.

Mit der Erweiterung *Augury*[11] steht uns außerdem ein umfangreiches Debugging-Tool für Angular-Anwendungen zur Verfügung. Augury ist allerdings nur als Chrome-Erweiterung erhältlich. Wir werden im Powertipp auf Seite 203 mehr über dieses Tool erfahren und lernen, wie wir es effizient einsetzen können.

1.3 Paketverwaltung mit Node.js und NPM

JavaScript ohne Browser

Node.js[12] ist eine Laufzeitumgebung zur Ausführung von JavaScript auf dem Server. Es basiert auf der Google V8 Engine[13], die auch in Google Chrome zum Einsatz kommt. Mit Node.js können serverbasierte Dienste mit JavaScript implementiert werden. Das hat den Vorteil, dass JavaScript für die Entwicklung von Backends *und* Frontends eingesetzt werden kann. Das Anwendungsspektrum ist nicht auf Webserver und REST-Schnittstellen begrenzt, sondern es können viele weitere skalierende Szenarien abgebildet werden. Seine Stärke zeigt Node.js bei der Arbeit mit asynchronen Operationen, die ein elementares Paradigma bei der Entwicklung mit dieser Laufzeitumgebung sind. Node.js wird von vielen Tools verwendet, die die Webentwicklung für den Programmierer komfortabler gestalten. Automatisierungen mit Grunt oder Gulp, CSS-Präprozessoren wie LESS oder SASS, Tests mit Karma oder Protractor und noch vieles mehr – alle basieren auf Node.js. Wir verwenden Node.js in diesem Buch nur zum Betrieb jener Tools, die wir für die Entwicklung mit Angular benötigen. Das REST-Backend, welches wir im Kapitel zu HTTP ab Seite 169 vorstellen, basiert auch auf Node.js.

Das Angular-Tooling setzt auf Node.js.

NPM-Pakete

Die Plattform Node.js bietet eine Vielzahl von Paketen, die sich jeder Entwickler zunutze machen kann. Um dieser Menge Herr zu wer-

[10] https://ng-buch.de/x/11 – Google Chrome

[11] https://ng-buch.de/x/12 – Angular Augury

[12] https://ng-buch.de/x/13 – Node.js

[13] https://ng-buch.de/x/14 – Google V8

den, ist der hauseigene Paketmanager Node Package Manager (NPM)[14] das richtige Werkzeug. Damit kann auf die Online-Registry aller Node.js-Module zugegriffen werden. Wer möchte, kann mit der Webseite http://npmjs.org nach den passenden Paketen suchen.

Pakete lassen sich sowohl *lokal* als auch *global* installieren. Die lokalen Pakete werden je Projekt installiert. Dazu werden sie auch im jeweiligen Verzeichnis gespeichert. Damit wird erreicht, dass ein Paket in verschiedenen Versionen parallel auf dem System existieren kann.

Globale Pakete werden von NPM in einem zentralen Verzeichnis[15] auf dem Computer gespeichert. Darin befinden sich meist CLI-Pakete (CLI steht für Command Line Interface), die von der Konsole aufgerufen werden können. Bekannte Beispiele dafür sind: `@angular/cli`, `typescript`, `webpack`, `gulp`, `grunt-cli`. All diese Pakete sind dazu da, andere Dateien auszuführen, zu verarbeiten oder umzuwandeln.

Node.js und NPM installieren

Node.js bietet auf der Projektwebseite Installationspakete für die verbreitetsten Betriebssysteme zum Download an. Einige Linux-Distributionen führen Node.js auch in den offiziellen Paketquellen, allerdings zum Teil nicht in aktueller Version. Wir empfehlen die Verwendung der offiziellen Installationspakete[16] bzw. Repositorys von Node.js. Hier sollten Sie die *LTS*-Variante wählen, denn sie wird breitflächig von den meisten Paketen unterstützt.

Installation prüfen

Nach der Installation prüfen wir auf der Kommandozeile, ob `node` und `npm` richtig installiert sind, indem wir die Versionsnummer ausgeben:

```
$ node -v
$ npm -v
```

Listing 1–1
Versionsnummer von Node.js und NPM ausgeben

Achten Sie darauf, dass Node.js und NPM stets aktuell sind, denn manche Tools funktionieren mit alten Versionen nicht.

NPM-Pakete installieren

Stehen `node` und `npm` ordnungsgemäß bereit, so können wir NPM zur Installation von Paketen verwenden. Dabei ist zu unterscheiden, ob ein Paket *lokal* oder *global* installiert werden soll.

[14] https://ng-buch.de/x/15 – npm

[15] Aktuellen globalen Pfad ausgeben: `npm config get prefix`. Unter Windows ist dies normalerweise `%AppData%\Roaming\npm`.

[16] https://ng-buch.de/x/16 – Node.js Downloads

Lokale Installation

Installieren wir NPM-Pakete *lokal*, wird im aktuellen Verzeichnis ein Unterordner mit der Bezeichnung `node_modules` erstellt. Darin befinden sich die installierten Pakete. Diese Variante empfiehlt sich zur Installation von Abhängigkeiten oder Befehlen, die wir innerhalb des aktuellen Projekts benötigen. Das gilt unabhängig davon, ob wir Angular oder eine andere Technologie einsetzen. Im Hauptverzeichnis eines Projekts existiert häufig eine Datei mit dem Namen `package.json`, in der alle NPM-Abhängigkeiten verzeichnet sind. Darauf gehen wir auf Seite 53 noch ausführlicher ein, wenn wir unser Beispielprojekt anlegen.

Generell gilt, dass eine lokale Installation der globalen vorzuziehen ist. Stellen wir uns vor, dass auf unserem System mehrere Softwareprojekte entwickelt werden. Jedes Projekt setzt NPM-Pakete in verschiedenen Versionen ein. Wenn nun alle Pakete global installiert sind, kann es zu Versionskonflikten, also unerwartetem Verhalten unserer Projekte kommen. Aus diesem Grund bevorzugen wir die lokale Installation.

Listing 1–2
NPM-Pakete installieren

```
$ npm install <paketname>
```

Bei der globalen Installation ist das entsprechende Paket aus allen Node-Anwendungen heraus erreichbar. Diese Variante bietet sich dann an, wenn die Pakete ausführbare Kommandozeilenbefehle beinhalten.

Globale Installation

Die Befehle sind bei einer globalen Installation aus jedem Arbeitspfad heraus aufrufbar. Ein häufiger Anwendungsfall für globale Pakete ist, Tools für die Kommandozeile bereitzustellen. Später in diesem Buch werden wir die Angular CLI kennenlernen (ab Seite 21). Sie vereinfacht die Erstellung von Angular-Projekten und steigert die Produktivität des Entwicklers.

Listing 1–3
NPM-Pakete global installieren

```
$ npm install -g <paketname>
```

Zusammenfassung

Unsere Arbeitsumgebung ist nun eingerichtet, und wir sind startklar, um mit Angular zu beginnen. Die vorgestellten Tools greifen uns bei der Arbeit mit Angular unter die Arme, sodass wir viele Dinge nicht von Hand erledigen müssen. Vor allem ein robuster und featurereicher Editor kann uns viel Tipparbeit abnehmen. *Los geht's!*

1.4 Codebeispiele in diesem Buch

Dieses Buch enthält viele Beispiele, um die Funktionen der Angular-Plattform zu demonstrieren. Die dazugehörigen Projekte haben wir Ihnen zentral zur Verfügung gestellt.

Hosting auf GitHub

Unter

https://ng-buch.de/app

erhalten Sie Zugriff auf eine Online-Demo des Beispielprojekts *BookMonkey 2*. Alle Projekte sind in der Entwicklerplattform GitHub[17] gehostet. Wenn Sie mit Git[18] arbeiten, können Sie jedes GitHub-Repository direkt über folgende Kurzlinks klonen und verwenden.

Listing 1–4
Beispielprojekt als Komplettpaket klonen

```
$ git clone https://ng-buch.de/app-code.git
```

Listing 1–5
Beispielprojekt in verschiedenen Stadien klonen

```
$ git clone https://ng-buch.de/it1-comp.git
$ git clone https://ng-buch.de/it1-evt.git
$ git clone https://ng-buch.de/it1-prop.git
$ git clone https://ng-buch.de/it2-di.git
$ git clone https://ng-buch.de/it2-nav.git
$ git clone https://ng-buch.de/it3-http.git
$ git clone https://ng-buch.de/it3-rxjs.git
$ git clone https://ng-buch.de/it4-forms.git
$ git clone https://ng-buch.de/it4-reactive.git
$ git clone https://ng-buch.de/it4-validators.git
$ git clone https://ng-buch.de/it5-directives.git
$ git clone https://ng-buch.de/it5-pipes.git
$ git clone https://ng-buch.de/it6-guards.git
$ git clone https://ng-buch.de/it6-lazy.git
$ git clone https://ng-buch.de/it6-modules.git
$ git clone https://ng-buch.de/it6-resolver.git
$ git clone https://ng-buch.de/it7-i18n.git
```

Listing 1–6
Alle weiteren Codebeispiele klonen

```
$ git clone https://ng-buch.de/start.git
$ git clone https://ng-buch.de/bm-native.git
$ git clone https://ng-buch.de/bm-rdx.git
$ git clone https://ng-buch.de/two-way.git
```

[17] https://ng-buch.de/x/17 – GitHub

[18] https://ng-buch.de/x/18 – Git

Des Weiteren ist ein Download als ZIP-Archiv möglich. Rufen Sie dafür einfach einen der vielen QR-Code-Links auf, z. B. diesen:

Demo und Quelltext:
https://ng-buch.de/app-code

... und laden Sie das ZIP-Archiv entsprechend der Abbildung 1–3 herunter.

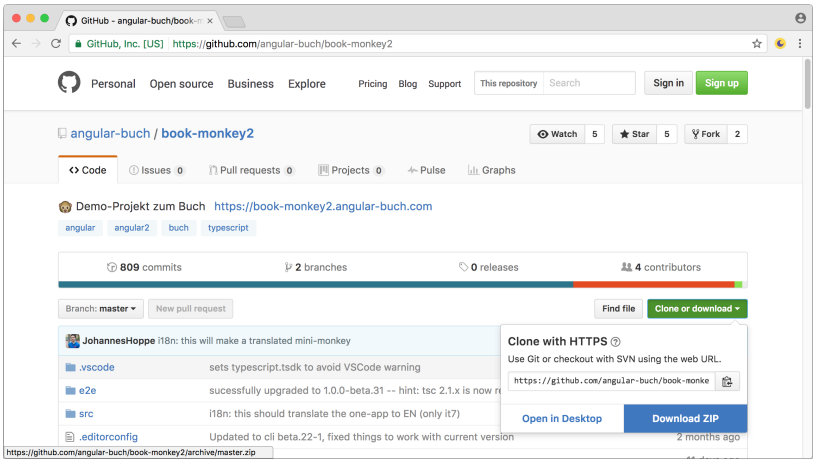

Abb. 1–3 *Codebeispiele dieses Buchs von GitHub als ZIP herunterladen*

2 Schnellstart

»2016: the year that setting up the tooling for a project takes longer than coding the actual product.«

Todd Motto
(Gründer von Ultimate Angular)

Am besten wird man mit einem neuen Framework vertraut, wenn man die Konzepte und Beispiele direkt selbst ausprobiert. Hierfür wollen wir eine sehr minimale Angular-Anwendung aufsetzen. Wir wollen an diesem Beispiel zunächst nur betrachten, wie eine solche Anwendung aufgebaut ist. Danach gehen wir im Beispielprojekt ausführlich auf alle Details des Angular-Frameworks ein.

Der Schnellstart ist nicht für den Produktivbetrieb geeignet.

Bitte beachten Sie, dass das hier vorgestellte Setup zwar gut zum Experimentieren mit Angular, aber aufgrund fehlender Optimierungen **nicht für einen produktiven Betrieb geeignet ist!**

2.1 Die erste Angular-Anwendung aufsetzen

Zum Beginn wollen wir eine erste prototypische Angular-Anwendung aufsetzen, um zu sehen, wie die Grundstruktur aufgebaut ist. Ganz bewusst wollen wir in diesem Kapitel alle Schritte manuell machen. Später werden wir die Angular CLI kennenlernen, die uns beim Aufsetzen eines Projekts einen Großteil der Arbeit abnimmt.

In der offiziellen Dokumentation gibt es einen ähnlichen Schnellstart.[1] Wir haben die originale Anleitung noch einmal gehörig gekürzt. Lediglich die folgenden sechs Schritte sind nun notwendig:

1. Das HTML-Grundgerüst anlegen (`index.html`)
2. Den Modul-Loader konfigurieren (`systemjs.config.js`)
3. Die Startdatei für das Bootstrapping anlegen (`main.ts`)
4. Das zentrale Angular-Modul anlegen (`app.module.ts`)
5. Eine erste Komponente anlegen (`app.component.ts`)
6. Den Webserver starten

[1] https://ng-buch.de/x/19 – Angular Docs: Quickstart

Ordner anlegen

Bevor wir beginnen, erstellen wir für unsere neue Anwendung einen leeren Ordner, z. B. 🗀`schnellstart`. Und dann geht's auch schon los!

2.2 HTML-Grundgerüst erstellen

Für unsere erste Anwendung benötigen wir ein HTML-Grundgerüst, denn die Anwendung soll im Browser ausgeführt werden. Wir legen hierzu die Datei `index.html` an, wie sie im Listing 2–1 abgebildet ist.

Listing 2–1 *Die Datei* index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Angular Schnellstart</title>

    <script
    ↪ src="https://unpkg.com/core-js/client/shim.min.js"></script>
    <script
    ↪ src="https://unpkg.com/zone.js@0.8.5?main=browser"></script>
    <script
    ↪ src="https://unpkg.com/systemjs@0.19.47/dist/system.src.js">
        ↪ </script>
    <script src="systemjs.config.js"></script>
    <script>
      System.import('./app/main')
    </script>
  </head>

  <body>
    <my-app>Loading...</my-app>
  </body>
</html>
```

Die ersten zwei Script-Tags binden Polyfills ein, welche für die Ausführung von Angular unverzichtbar sind. Die Polyfills müssen als Erstes bereit sein und stehen deswegen an erster Stelle.