

Extracted from:

# Rails, Angular, Postgres, and Bootstrap, Second Edition

Powerful, Effective, Efficient, Full-Stack Web Development

This PDF file contains pages extracted from *Rails, Angular, Postgres, and Bootstrap, Second Edition*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit <http://www.pragprog.com>.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printed versions; the content is otherwise identical.

Copyright © 2017 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The Pragmatic Bookshelf

Raleigh, North Carolina

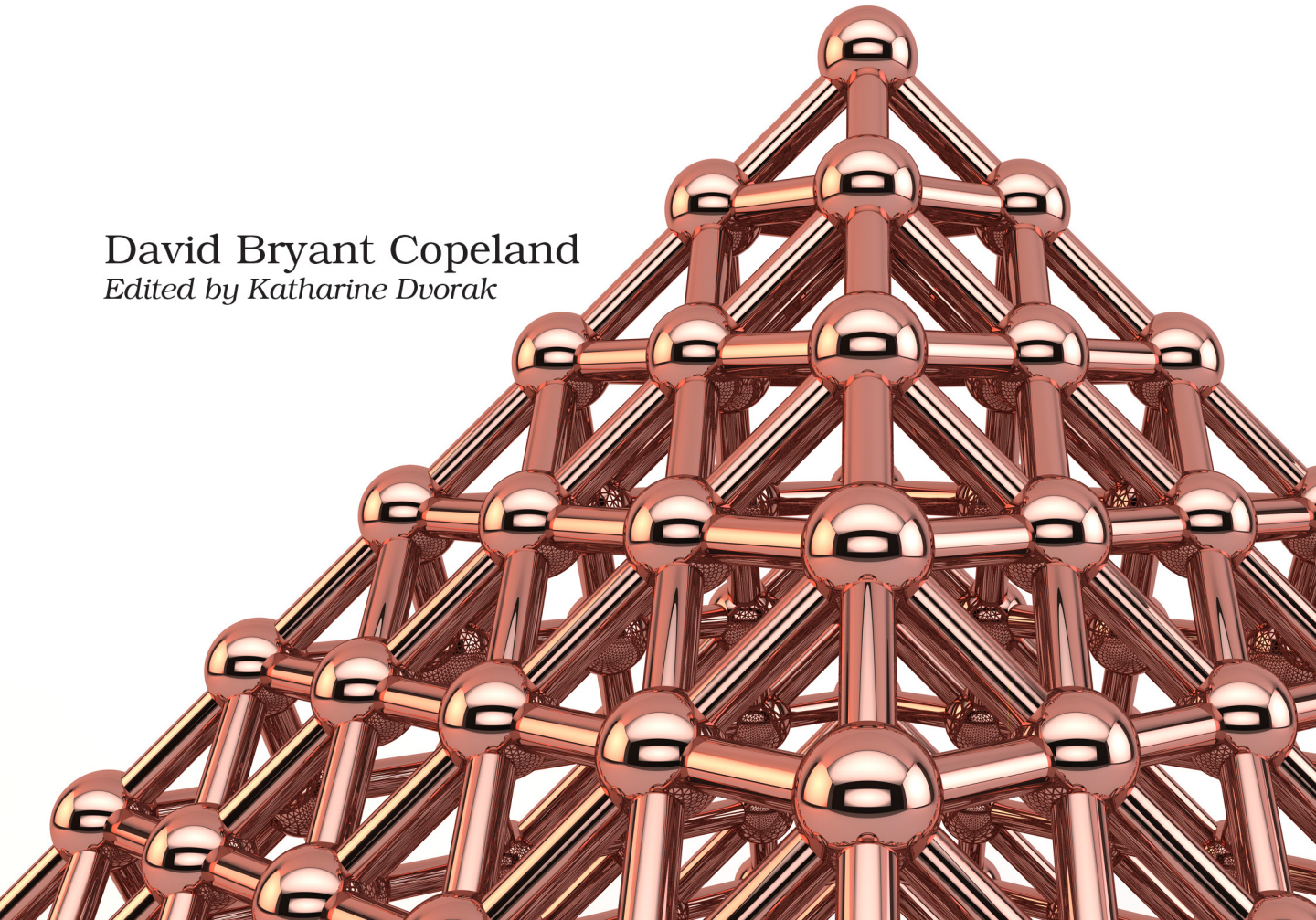
The  
Pragmatic  
Programmers

# Rails, Angular, Postgres, and Bootstrap

## Second Edition

Powerful, Effective, Efficient,  
Full-Stack Web Development

David Bryant Copeland  
*Edited by Katharine Dvorak*



# Rails, Angular, Postgres, and Bootstrap, Second Edition

Powerful, Effective, Efficient, Full-Stack Web Development

David Bryant Copeland

The Pragmatic Bookshelf

Raleigh, North Carolina



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at <https://pragprog.com>.

The team that produced this book includes:

Publisher: Andy Hunt

VP of Operations: Janet Furlow

Executive Editor: Susannah Davidson Pfalzer

Development Editor: Katharine Dvorak

Indexing: Potomac Indexing, LLC

Copy Editor: Liz Welch

Layout: Gilson Graphics

For sales, volume licensing, and support, please contact [support@pragprog.com](mailto:support@pragprog.com).

For international rights, please contact [rights@pragprog.com](mailto:rights@pragprog.com).

Copyright © 2017 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

Printed in the United States of America.

ISBN-13: 978-1-68050-220-6

Encoded using the finest acid-free high-entropy binary digits.

Book version: P1.0—June 2017

## Changing Our Search to Use Typeahead

Given everything you've done up to this point, changing the search from one where you must click a button to one where the search happens as you type will actually be fairly straightforward. Because Angular has allowed us to separate our concerns, you have all the code you need in place. You'll just need to connect it to the user interface in a different way.

Currently, when the user modifies the contents of the text field, Angular updates the value of keywords in our CustomerSearchComponent class. You can't directly see it, but it does it as the user types. If you can hook into that behavior, you can perform your search as the user is typing.

Angular provides a way to do this by binding to the `ngModelChange` property. The code you have now is

```
<input bindon-ngModel="keywords" ... >
```

which is equivalent to this:

```
<input bind-ngModel="keywords"
      on-ngModelChange="keywords=$event" ... >
```

Recall that `on-` creates a one-way binding from the view to our code (you used this for the click event earlier in [Respond to Click Events, on page ?](#)). As part of sending the event back to our code for `ngModelChange`, Angular sets the global variable `$event`. This means that instead of assigning it to `keywords`, as happens by default, you can send `$event` to our search function:

```
<input type="text" id="keywords" name="keywords" \
      placeholder="First Name, Last Name, or Email Address" \
      class="form-control input-lg" \
      bind-ngModel="keywords" \
      on-ngModelChange="search($event)"> \
```

Note that because you’ve replaced Angular’s default behavior, you need to set `this.keywords` inside `search` yourself. Right after you do that, however, you can use the updated value to perform the search just as before (though you’re only going to do a search for three or more characters so you don’t do too broad a search):

6. angular/50-actual-typeahead/shine/app/javascript/packs/customers.js

```

> search: function($event) {
    var self = this;
    self.keywords = $event;
    if (self.keywords.length < 3) {
    >     return;
    > }
    self.http.get(
      "/customers.json?keywords=" + self.keywords
    ).subscribe(
      function(response) {
        self.customers = response.json().customers;
      },
      function(response) {
        window.alert(response);
      }
    );
  }
}

```

Finally, let’s remove the “Find Customers” button since it’s no longer needed. Removing this means you can remove the `span` surrounding the button as well as the `div` you used to make the button group. Our search form now looks like so:

```

<section class="search-form"> \
  <form> \
    <label for="keywords" class="sr-only">Keywords</label> \
    <input type="text" id="keywords" name="keywords" \
      placeholder="First Name, Last Name, or Email Address" \
      bind-ngModel="keywords" \
      on-ngModelChange="search($event)" \
      class="form-control input-lg"> \
    </form> \
  </section> \

```

Now, reload the page and type in “pat.” You’ll see some search results like those shown in the [figure on page 7](#).

Customer Search

pat

Results

Pat Bogisich

mathilde4

wilton.altenwerth4@marvin.biz

JOINED 2017-05-25T13:51:31.994Z

Pat Considine

haskell2

rachelle.keeling2@jacobson.name

JOINED 2017-05-25T13:51:31.990Z

Patricia Hegmann

dallin66

gabriel66@streich.co

JOINED 2017-05-25T13:51:31.997Z

Pat Hermann

unique1

harley\_jast1@lebsack.org

JOINED 2017-05-25T13:51:31.989Z

Pat Jones

patty\_valentin

pat123@somewhere.net

JOINED 2017-05-25T13:51:32.000Z

Patricia Koelpin

desiree.goodwin99

leonardo99@hane.info

JOINED 2017-05-25T13:51:31.995Z

Pat Reichert

kamryn0

madilyn0@collinsmoriette.com

JOINED 2017-05-25T13:51:31.984Z

Pat Ritchie

friedrich.buckridge3

eileen3@heathcotelittel.org

JOINED 2017-05-25T13:51:31.992Z

If you keep typing out “patricia,” the results automatically reduce to only those that match, as shown in the next figure.

Customer Search

patricia

Results

Patricia Hegmann

dallin66

gabriel66@streich.co

JOINED 2017-05-25T13:51:31.997Z

Patricia Koelpin

desiree.goodwin99

leonardo99@hane.info

JOINED 2017-05-25T13:51:31.995Z

The typeahead works! The entire feature required little code (once you installed and configured Angular—a one-time cost), and instead of implementing typeahead with a special-purpose library, you have set up a framework for implementing any user interface you might need. Because of how Angular works, you aren't wrestling with how to attach our JavaScript to our DOM elements or how to interact with the back end. Because of how Rails works, our back end is almost identical to the original back end.

In other words, by using what Rails gives us, and using what Angular gives us, you were able to create a fairly sophisticated feature quickly and without a lot of code. And it's fast, thanks to Postgres's sophisticated indexing and ordering features.