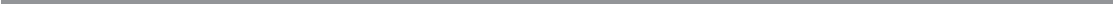


Ralph Steyer

Building web applications with Vue.js

MVVM patterns for conventional and single-page websites

 Springer



Building web applications with Vue.js

Ralph Steyer

Building web applications with Vue.js

MVVM patterns for conventional
and single-page websites

Ralph Steyer
Bodenheim, Germany

ISBN 978-3-658-37595-9 ISBN 978-3-658-37596-6 (eBook)
<https://doi.org/10.1007/978-3-658-37596-6>

© The Editor(s) (if applicable) and The Author(s), under exclusive licence to Springer Fachmedien Wiesbaden GmbH, part of Springer Nature 2022

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Fachmedien Wiesbaden GmbH part of Springer Nature.

The registered company address is: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Preface

Of course, the Internet is still on everyone's lips. Digitization in general is one of the most used "buzzwords" – especially by politicians, media people, and decision makers. But the times of static websites are largely over. While it was common a few years ago that at least simple websites of private persons or smaller associations still used pure HTML code (Hypertext Markup Language), such antiquated websites can be found less and less nowadays. Usually at least content management systems (CMS) like WordPress, Joomla!, Typo, or Drupal are used or such websites are at least spiced up with stylesheets and/or JavaScript.

But if you are not satisfied with these 08/15 solutions of the common CMS or if they do not (cannot) meet the required requirements, the only option is the real programming of websites or even Web applications. On the one hand, this path usually begins with programming on the Web server side, including downstream database systems, but on the other hand, modern websites or Web applications must also be programmed in the client (the browser).

And in the browser, only JavaScript has been available for years as a universally available technology for programming. Together with HTML and CSS (Cascading Style Sheets), JavaScript forms the triad of modern websites and especially of client-side Web programming.

Now JavaScript has been around for a very long time on the Web, but for most years it was completely underestimated and dismissed as a primitive beginner's language. Only in the last few years have we realized what a treasure JavaScript is for efficient and powerful programming and that this language is anything but a beginner's language – even if you can learn it quickly as a beginner. Quite the contrary. But you have to be able to use it professionally, because in contrast to all-round carefree programming worlds from the .NET and Java environment, which have "softened" programmers over the years by hardly allowing them to make mistakes, you can't expect such protective mechanisms in JavaScript. Efficient and secure programming with JavaScript requires skills from the programmer instead of transferring them to IDEs (Integrated Development Environment) and runtime environments. But this also makes JavaScript much leaner and more efficient than its now hopelessly overloaded competitors. Professional, efficient, and secure programming with JavaScript therefore requires immense programming experience. Already in the

client, but even more so on the server side, where JavaScript has meanwhile also begun its triumphant march. JavaScript is like a scalpel. In the hands of an experienced surgeon, you can perform miracles with it. In the hands of a layman or someone not working carefully, it can do immense damage. Where JavaScript manages the balancing act of still being easy to use by beginners for simple tasks. In my JavaScript trainings, I hear again and again that many participants have already used JavaScript. Mostly they copied existing scripts and adapted them if necessary or wrote very simple scripts themselves and built them into HTML pages. And that usually works, although almost always the addition of the participants was that they did not really know why it works.

The many frameworks that have established themselves in the Web environment in recent years use some fundamentally different approaches. But most frameworks often try to extend JavaScript by things that are not possible with the core version. This makes it easier to work with JavaScript and also provides possibilities that are not or not easily available with pure JavaScript.

The various frameworks take very different approaches to ultimately end up with the same result in the code that reaches the user, namely a conglomerate of HTML, CSS, and JavaScript. Of course, always connected with resources such as images, videos, audios, and the like.

Now, Vue.js is an increasingly popular framework on the Web that takes a very specific approach. It is a reactive, client-side JavaScript Web framework that is essentially used to create so-called single-screen Web applications or single-page Web applications (only one Web page in the browser that updates parts as needed and does not reload a new Web page) according to a Model-View-Controller pattern (MVC). Strictly speaking, a Model-View-Controller pattern (MVVC) is used. But you can definitely create Web applications and Web pages with it.

Vue.js is both easy to learn and very extensible and customizable. To be able to learn Vue.js successfully, a good knowledge of HTML and JavaScript is sufficient, as well as CSS if possible, which I would also like to assume in the book. The developers of Vue.js call the framework “progressive.” This essentially means that it can be used as much for small improvements to individual details of the website as for larger projects. It also supports the creation of reusable components. Another advantage of Vue.js is that it does not require a complex installation and can even be used entirely without installation “from the cloud” (from a CDN content delivery network) if required.

So, follow me into the fascinating world of Vue.js!

Bodenheim, Deutschland
Spring/Summer 2019

Ralph Steyer
<http://www.rjs.de>

Contents

1	Introduction: Before the Real Thing Starts	1
1.1	What Do We Cover in the Introductory Chapter?	1
1.2	The Aim of the Book	1
1.3	What Should You Already Be Able to Do?	3
1.4	What Do You Need to Work with the Book?	3
1.4.1	The Vue.js Framework	4
1.5	The Features of Vue.js	9
1.5.1	Directives	9
1.5.2	The Virtual DOM	9
1.5.3	Data Binding and Reactivity	10
1.5.4	Creation of Components	10
1.5.5	An Own Event System	11
1.5.6	Animation and Transition Effects	11
1.5.7	Calculated Properties	11
1.5.8	Templates	11
1.5.9	Watcher	12
1.5.10	Routing	12
1.5.11	Vue CLI	12
1.6	Summary	13
2	First Examples: Just Test Vue.js Once	15
2.1	What Do We Cover in the Chapter?	15
2.2	The Basic Framework and a First Example	15
2.3	Dynamics for the Example	18
2.3.1	Real Response and v-model	20
2.4	Summary	23
3	Behind the Scenes: How and Why Does Vue.js Work?	25
3.1	What Do We Cover in This Chapter?	25
3.2	The Principle of Fault Tolerance and the DOM Concept	25
3.2.1	The DOM Concept From a Particular Point of View	26

3.3	Arrays, Objects and JSON.	31
3.3.1	Hash Lists	34
3.3.2	The JavaScript Object Notation	35
3.3.3	Callbacks and Function References	36
3.4	MVC and MVVC	38
3.4.1	Design Patterns	40
3.4.2	The MVC Pattern	40
3.4.3	MVVC.	41
3.5	Summary	41
4	Vue.js in Depth: The Vue Instance, Vue Templates, and Data Binding	43
4.1	What Do We Cover in the Chapter?	43
4.2	The <i>Vue Instance</i>	44
4.2.1	Responding to the Creation of the <i>Vue Object</i> : The Life Cycle	44
4.3	Basic Information About Vue.js Templates	46
4.3.1	The <i>Template Attribute</i>	46
4.3.2	Under the Template Hood	47
4.3.3	Different Types of Data Binding in Templates	47
4.3.4	Using JavaScript Expressions for Data Binding	56
4.4	More on Directives	56
4.4.1	Arguments.	57
4.4.2	Dynamic Arguments	57
4.4.3	Restrictions for Dynamic Argument Values	57
4.4.4	Modifiers for Attributes.	58
4.5	Components.	58
4.5.1	Watch Out!	60
4.5.2	Global Versus Local Registration	60
4.5.3	Data Transfer.	60
4.5.4	The Way Back: Slots.	62
4.5.5	Asynchronous Data Transmission.	64
4.5.6	Single File Components	65
4.6	Which Side Would You Like to Have? Routing	66
4.6.1	MVVC/MVC and Routing	66
4.6.2	The Concrete Implementation in Vue.js	67
4.7	Summary	69
5	Working with Arrays: Iterations with the v-for Directive	71
5.1	What Do We Cover in the Chapter?	71
5.2	The v-for Directive	71
5.2.1	Static Display of Values From an Array	72
5.2.2	Access to the Index of the Current Element	74

5.3	Access to More Complex Structures	75
5.3.1	Nested v-for Directives	79
5.3.2	Addressing Individual Entries Directly.	85
5.4	Specific Applications of the v-for Directive	87
5.4.1	The v-for Directive with a Range of Values	87
5.4.2	Access to the Parent Elements	89
5.4.3	Key and Index in One Object	91
5.4.4	The Key Attribute for Binding the Id	91
5.4.5	Calling Callbacks	92
5.5	Observing Changes in Arrays	93
5.5.1	Mutating Methods.	93
5.5.2	Sorting Arrays and Working with Methods	95
5.5.3	Generating New Arrays.	100
5.6	Summary	103
6	Conditional Rendering: The v-if Directive – Making Decisions	105
6.1	What Do We Cover in the Chapter?	105
6.2	The v-if, v-else and v-else-if Directives	105
6.3	The v-show Directive	109
6.4	When v-if and When v-show?	109
6.5	A Special Combination: The Directive v-for with v-if or v-show.	110
6.5.1	A Wrapper with v-if is Better	111
6.6	Summary	112
7	Events, Methods, Observers and Calculated Properties: Calculated Results and Reactions	113
7.1	What Do We Cover in the Chapter?	113
7.2	Basic Considerations on the Distribution of Tasks	113
7.3	Methods of a Vue Object and the Methods Property.	114
7.4	The Event Handling in Vue.js	115
7.4.1	Background to Event Handling.	115
7.4.2	The Concrete Example of v-on.	118
7.4.3	Evaluating the Event object	121
7.4.4	Event Modifier	124
7.4.5	Other Modifiers.	125
7.4.6	User-Defined Events	126
7.5	The <i>Computed Property</i>	128
7.6	When Methods and When Calculated Properties?	131
7.7	Watcher (Observer)	132
7.7.1	Observing the Geolocation with a Watcher	132
7.7.2	Ajax with a Watcher	135
7.8	Summary	142

8	Dynamic Layouts with Data Binding: Making Stylesheets Dynamic	143
8.1	What Do We Cover in the Chapter?	143
8.2	Data Binding and the <i>v-bind Directive</i> for Conditional Classes	143
8.2.1	Switching CSS Classes	144
8.2.2	The Array Notation for Multiple Properties	151
8.2.3	Logic in the HTML File	152
8.3	Data Binding and the <i>v-bind Directive</i> for Inline Styles	152
8.4	Abbreviations (Shorthands)	154
8.4.1	The <i>v-bind</i> Abbreviation	154
8.4.2	Abbreviation <i>v-on</i>	154
8.5	Summary	154
9	Forms and Form Data Binding: Interaction with the User	155
9.1	What Do We Cover in the Chapter?	155
9.2	Basics of Using Forms on the Web	156
9.2.1	The Contained Form Elements	156
9.3	Basic Use of Form Binding in <i>Vue.js</i>	157
9.3.1	<i>Vue</i> Instance First	157
9.4	Some Concrete Examples	158
9.4.1	A Simple Form with Different Form Elements	158
9.5	Dynamic Options	164
9.6	A Task List as a Practical Example	166
9.6.1	A First Simple Version of a <i>Todo</i> List	166
9.6.2	A Permanent Task List	171
9.6.3	Persistence the Second: <i>Server-Side</i>	176
9.7	More on Value Bindings for Forms	182
9.7.1	The Modifiers	183
9.8	Summary	183
10	Filtering Techniques: Selected Data Only	185
10.1	What Do We Cover in the Chapter?	185
10.2	Basics of Filters for JavaScript Arrays	185
10.2.1	The Arrow Notation	188
10.3	Filters in <i>Vue.js</i>	188
10.3.1	Local Filters	189
10.3.2	Global Filters by Extending the <i>Vue</i> Instance	189
10.3.3	Dynamic Filtering	190
10.3.4	Chaining Filters	192
10.3.5	Transfer to Parameters	192
10.4	Summary	192

11	Transitions and Animations: Moving Things	193
11.1	What Do We Cover in the Chapter?	193
11.2	Transitions with Transition	194
11.3	The Transition Classes	197
11.4	CSS Animations	198
11.5	Special Situations	199
11.5.1	Using Transitions and Animations Together.....	199
11.5.2	Explicit Transition Times: The Duration Specification	199
11.5.3	JavaScript Hooks.....	199
11.5.4	Animation of Data.....	201
11.6	Summary	201
12	Outlook: What Else Is There in Vue.js?	203
12.1	What Do We Cover in the Chapter?	203
12.2	Use Vue.js in CMS or in Combination with Other Frameworks	203
12.3	Server-Side Rendering.....	205
12.4	Mixins	206
12.5	User-Defined Directives	206
12.6	Plugins.....	207
12.6.1	Using a Plugin.....	208
12.6.2	Writing a Plugin	208
12.7	Summary	209
	Appendix	211
	Index	215