

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!

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

About the Author



Ralph Steyer has a degree in mathematics and works as a freelance trainer, author, and programmer. You can find his website at <http://www.rjs.de> and his blog at <http://blog.rjs.de>. His professional focus is on Web development and programming in Java and .NET.

- Here is another short abstract of the professional career and experiences:
- Studied until 1990 in Frankfurt/Main at the Johann Wolfgang Goethe University.
- After graduation, programmer at a large insurance company in Wiesbaden for actuarial PC. programs.
- After almost 4 years, internal change to the conception of mainframe databases
- Freelancer since 1996. Division of work into different areas of activity – specialist author, specialist journalist, IT lecturer, and programmer/consultant.
- Numerous book publications, video productions, and online trainings in the IT sector as well as technical articles in computer magazines.
- Speaker at various IT conferences.
- Lecturer at the Rhine-Main University of Applied Sciences in Wiesbaden and the TH Bingen.

List of Figures

Fig. 1.1	The official website of Vue.js	4
Fig. 1.2	Information about Vue.js.	5
Fig. 1.3	Integrating Vue.js	6
Fig. 1.4	Information and resources about Node.js	8
Fig. 1.5	Installing Vue-CLI with npm	12
Fig. 1.6	Vue-CLI was successfully installed with npm	13
Fig. 2.1	Vue.js has output the text	18
Fig. 2.2	Vue.js updates the date string	19
Fig. 2.3	Data binding with Vue.js.	21
Fig. 2.4	In the background, Vue.js does its magic	21
Fig. 2.5	The examples were spiced up a bit with CSS – here the third example	22
Fig. 3.1	The emulation of the old version in Internet Explorer shows the “destruction” of the DOM tree.	29
Fig. 3.2	The DOM tree is clean and stable.	30
Fig. 3.3	Arrays and objects	32
Fig. 3.4	The methods and properties that Array provides are also offered via the object.	33
Fig. 3.5	Here the methods and properties of the class Array are missing	33
Fig. 3.6	Worst conceivable mixing of responsibilities	39
Fig. 3.7	Triggering functionality from HTML.	39
Fig. 4.1	Vue.js shows the permanently updated date string.	49
Fig. 4.2	Vue.js now displays the permanently updated date string in a form field	49
Fig. 4.3	Moustache syntax does not interpret HTML, but the v-html directive does	51
Fig. 4.4	The directives and the Moustache syntax in combination	52
Fig. 4.5	The v-html directive versus the v-bind directive in combination	53
Fig. 4.6	The button is there and deactivated.	55
Fig. 4.7	The attribute node is no longer there at all	55
Fig. 4.8	The component is rendered to normal HTML and CSS	59
Fig. 4.9	The contents are passed to the component via props	61
Fig. 4.10	From the view to the model and back again	63

Fig. 4.11	The homepage	68
Fig. 4.12	A help page	68
Fig. 4.13	The about page	69
Fig. 4.14	An error page	69
Fig. 5.1	The list was generated by Vue.js from the array	73
Fig. 5.2	The list was still designed with CSS	74
Fig. 5.3	Now the index is also used	75
Fig. 5.4	All “inner” elements are output among each other, but even in array form	77
Fig. 5.5	Only one “record” is output in array form	78
Fig. 5.6	Now we have individual values in the list	79
Fig. 5.7	Now the content of a “record” is output as a list	80
Fig. 5.8	Only a part of the data structure is output individually	81
Fig. 5.9	Now the data structure is output completely individually	82
Fig. 5.10	The display is now edited a bit more	83
Fig. 5.11	Each individual list item is located separately in an ul container	85
Fig. 5.12	Related list items are combined in only one ul container	86
Fig. 5.13	Numerical access to array elements	86
Fig. 5.14	Specifying a range of values	88
Fig. 5.15	Specification of index and value range are possible, but rarely useful	88
Fig. 5.16	Specifying index or element in connection with a JavaScript call	89
Fig. 5.17	Using a parent element from the v-for directive	90
Fig. 5.18	The view is bound to the array	94
Fig. 5.19	The array has been changed and the view automatically updated	94
Fig. 5.20	After loading the web page, the table shows the original sorting of the array	95
Fig. 5.21	The table has been reordered and the user can see the interactive column headers	97
Fig. 5.22	The view after loading the web page	101
Fig. 5.23	The view and output in the console after changing the array	102
Fig. 6.1	The heading of the if-branch is to be seen	106
Fig. 6.2	One heading is displayed – the one from the else branch	107
Fig. 6.3	The one heading displayed – the one from the if-else branch	108
Fig. 6.4	The heading of the else-branch is to be seen	108
Fig. 6.5	The empty nodes can be seen and should no longer be displayed in the modification	110
Fig. 6.6	The empty nodes are now hidden	110
Fig. 7.1	The website before localisation	118
Fig. 7.2	The localization was successful	121
Fig. 7.3	Localization only with user permission	121
Fig. 7.4	The web page before evaluation	123
Fig. 7.5	The event object was evaluated	124
Fig. 7.6	The web page with the bound input field and the two properties	129

Fig. 7.7	The web page shows the bound static property as well as the calculated property.	130
Fig. 7.8	The calculated property can also deal sensibly with non-numerical values	131
Fig. 7.9	The website before localisation	134
Fig. 7.10	Localization only with user permission	135
Fig. 7.11	The localization was successful	135
Fig. 7.12	The checkbox had the value false when calling the watcher function	136
Fig. 7.13	The user can search for an employee	136
Fig. 7.14	The Ajax request was sent away with a letter as transfer value	136
Fig. 7.15	The employee has been found.	137
Fig. 7.16	When the input field is cleared, an alternative message is displayed	137
Fig. 8.1	The class active is assigned.	146
Fig. 8.2	The class active is not assigned.	146
Fig. 8.3	No class is assigned	148
Fig. 8.4	Only class active is assigned.	149
Fig. 8.5	Both classes are assigned	150
Fig. 8.6	Only the second class is assigned	150
Fig. 8.7	Formatting with inline styles	153
Fig. 9.1	The web page with the form after loading	159
Fig. 9.2	The text fields	161
Fig. 9.3	The text fields have changed.	162
Fig. 9.4	The checkbox is deselected.	162
Fig. 9.5	Three checkboxes are selected	163
Fig. 9.6	The order in the array is changed	163
Fig. 9.7	Only two checkboxes were selected – first the last checkbox	163
Fig. 9.8	The first radio button was selected	163
Fig. 9.9	The list fields were also selected.	164
Fig. 9.10	The list is dynamic	165
Fig. 9.11	Text is now used instead of value	166
Fig. 9.12	The todo list after loading	167
Fig. 9.13	The user enters a task	167
Fig. 9.14	The task was taken over	168
Fig. 9.15	The entry “Shopping” has been deleted again	169
Fig. 9.16	The user enters a task and the Local Storage is empty so far	175
Fig. 9.17	The task was taken over and stored in the Local Storage at the same time	175
Fig. 9.18	It is easy to see that the local storage and the task list are completely synchronized.	176
Fig. 9.19	You may also have to deal with such entries in the task list.	180
Fig. 10.1	An array and a button appear after loading the web page.	186
Fig. 10.2	Only the even numbers are now displayed	187
Fig. 10.3	Dynamic filtering	191
Fig. 11.1	The web page before the animated transition	196

Fig. 11.2 The text has disappeared after the animated transition. 196

Fig. 12.1 Vue.js functionality is used in the WordPress site 204

Fig. 12.2 In the source code of the WordPress site, Vue.js and its own
functionality are noted 205

List of Tables

Table 4.1	Lifecycle hooks	45
Table A.1	Sources on the Internet around the book and Vue.js	211



Introduction: Before the Real Thing Starts

1

1.1 What Do We Cover in the Introductory Chapter?

Before we really get started, this introductory chapter will clarify a few things that will make your subsequent work with this book and Vue.js easier. In particular, you'll learn what you should bring along as prerequisites and where you can get Vue.js. And it briefly discusses what Vue.js brings to the table in terms of features.

1.2 The Aim of the Book

This book is designed to **get you started** using the Vue.js framework. Either in the form of self-study or as accompanying material in appropriate courses. It teaches the elementary basics of creating and maintaining web applications with Vue.js. This includes topics such as the following:

- The environment – HTML/JavaScript/CSS and the web
- Basic creation of applications with Vue.js
- The JavaScript basis – especially arrays and JSON, function references and the DOM concept
- The MVVC concept as a special variant of the MVC concept
- The *Vue instance* and how to work with it
- Event handling
- Watcher
- Calculated properties
- Components and their life cycle

- The Double Curly Syntax and Data Binding
- Directives
- Templates
- Modularity and the extension of Vue.js

For the Vue.js framework, version 2.x (as of early 2019) is used in the book.

► **Definition**

The term “*framework*” is not so clear in its form. According to Wikipedia, it is understood to mean that:

A framework is a programming framework used in software engineering, especially in object-oriented software development and component-based development approaches. In a more general sense, a framework also refers to an order framework.

Generally, a framework is understood to be a real or conceptual structure designed to support or guide the creation of something that extends the structure itself into something meaningful. In IT, it is often understood to be a layered structure that specifies the types of programs that can or should be created and how they relate to each other. Some frameworks include programs, specify application programming interfaces (APIs), or provide programming tools that can be used with the framework. Usually, in IT, a framework will have at least one library along with a set of rules for its use. Sometimes there is an even broader form in which, in addition to certain libraries and syntax structures or languages, tools such as Visual Studio and/or SQL Server are explicitly fully integrated into the concept.

In the documentation, emphasis is placed on the basic application of the various techniques and simple examples, less on completeness of all possible statements, commands or parameters. In particular, only an introduction to the creation of a Vue.js application should and can be provided here. For various further topics, however, reference is made to the official documentation or additional sources, in order to provide you with an introduction there as well.

- The source codes of the book can be found sorted by chapters and projects created in them on the publisher’s web pages. The names of the current files or projects are given as notes or directly in the text before the respective examples and are repeated if necessary. However, I strongly recommend that you create all of the examples by hand. This is clearly better for understanding and learning than just copying or looking at them.

At some points in the book, tasks are formulated that you should solve at the point. For a few tasks (such as creating a specific program), explicit reference is made to the solution in the appendix if it is necessary and the explanations in the passage do not further explain or describe the solution to the task.

1.3 What Should You Already Be Able to Do?

This book is designed to help you get started with Vue.js and learn the basics of this framework from the ground up. However, working with such a framework is very rarely done without prior knowledge of the web and/or a programming language. Therefore, a good knowledge of HTML and basics in CSS should be assumed. And then there is JavaScript. For the understanding of the book, the basic syntactic principles (integration in web pages, data types including loose typing, variables, loops, decision structures, jump instructions, simple function declarations including calling functions, etc.) should also be assumed.

However, the really interesting and non-trivial things in JavaScript are the function references or the callback philosophy and the (largely) equivalence of objects and arrays and especially the JSON format (JavaScript Object Notation). Together with the DOM concept (Document Object Model), these are the absolute basics to understand how Vue.js works. True, you can also “work” with Vue.js without being well-versed in the techniques. However, I would go so far as to say that Vue.js is almost self-evident once you have a little understanding of the MVVC pattern of thinking and just these three key technologies really well. For this reason, the book also explains these concepts thoroughly right at the beginning.

- ▶ The framework Vue.js itself including the official documentation for the framework can be found on the web (Fig. 1.1) at <https://vuejs.org/> or <https://vuejs.org/v2/guide/>. For various further questions and topics, as mentioned, reference is made to it again and again.

1.4 What Do You Need to Work with the Book?

- A PC or similar is required as a basis for the book.
- The reference operating system is Windows 10 (previous versions like Windows 7 are also possible, but are not explicitly considered), but you can – as usual in web programming – also work with other operating systems like Linux or MacOS.
- Beyond that, any editor will do, but you can also use an IDE like Visual Studio or Visual Studio Code, Eclipse, etc.
- Otherwise, a local web server is highly recommended. If you want to make things as simple as possible, an all-round carefree package such as XAMPP is a good choice, although Internet Information Services (IIS) is also suitable, especially under Windows.

The XAMPP package is a collection of programs with the Apache web server at the center, which is supplemented by the MySQL database management system or, in new versions, its fork MariaDB (including phpMyAdmin for administration of the database management system) and PHP support, the FTP server FileZilla, and several other web

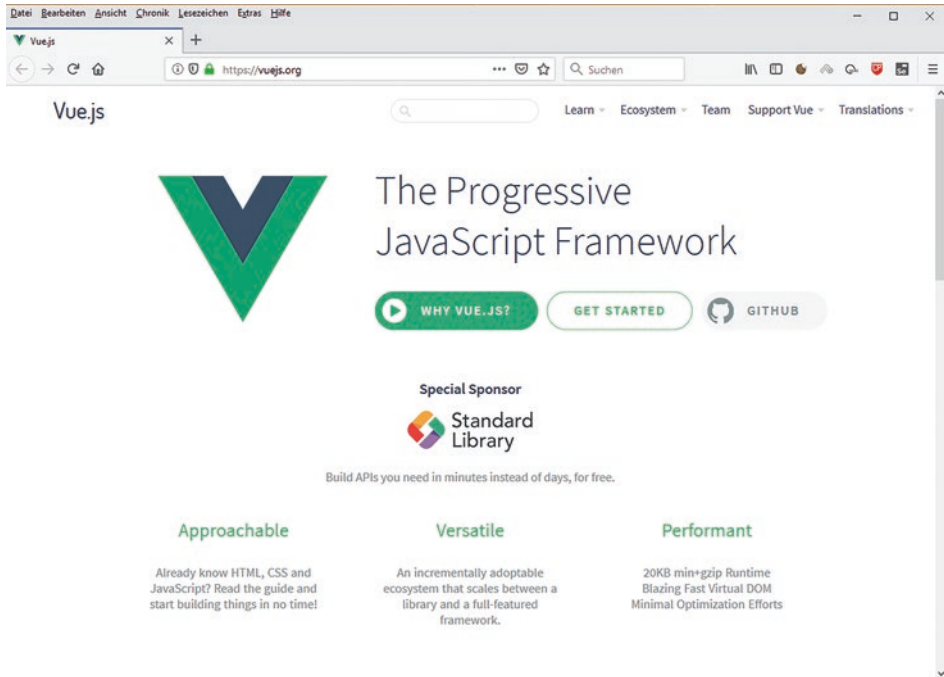


Fig. 1.1 The official website of Vue.js

technologies. XAMPP is available for various operating systems (<http://www.apache-friends.org/de/>).

You only need to install this package with a simple wizard and you will have a fully functional Apache web server in a basic configuration. Note, however, that XAMPP is configured by default for local testing purposes only. To keep things as simple as possible, all security settings are set low. Once the installation of XAMPP is complete, you can either start Apache manually or set it up so that Apache is integrated as a service or process in your operating system and can even be called automatically when the computer is started. XAMPP provides a comfortable and very easy to use control program for administration.

1.4.1 The Vue.js Framework

Of course, then you need Vue.js itself. There are already several good hints on the project's website on how to get started (Fig. 1.2).

Above all, you will find tips on how you can use Vue.js specifically in your website. This is remarkably easy and one of the highlights of this framework.

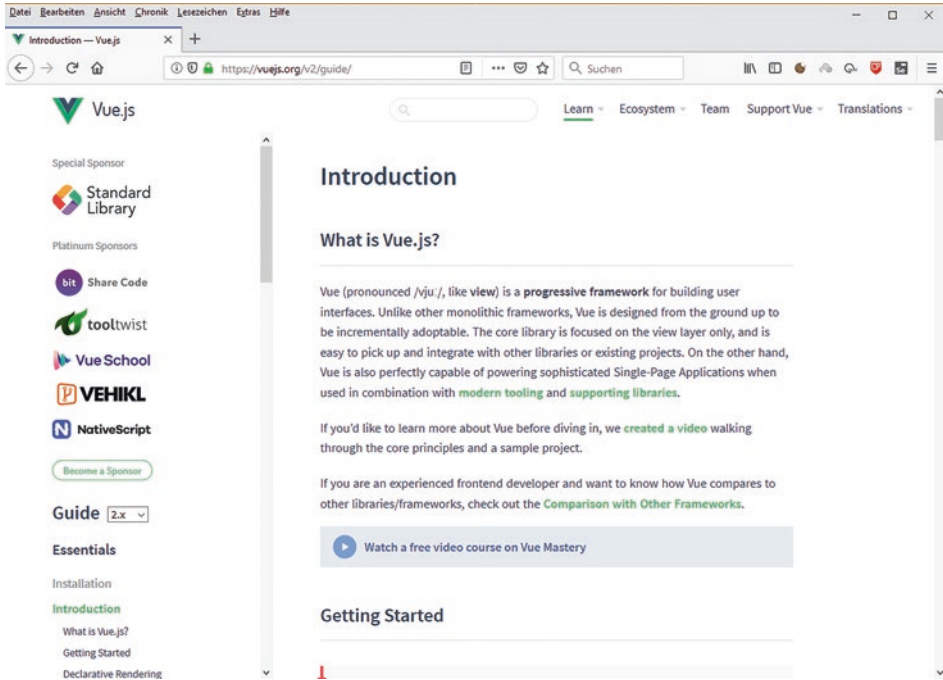


Fig. 1.2 Information about Vue.js

- ▶ You can join the team around Vue.js itself at <https://vuejs.org/v2/guide/join.html> (Join the Vue.js Community!).

Since Vue.js is essentially just a JavaScript library, you basically include the framework like any other external JavaScript file. You can include this file – also simple JavaScript – from a foreign or even your own web server.

Note – Vue.js comes in two flavors:

- A minimized version for practical use.
- A non-minimized version for development. The code of this variant is easier to read and provides warnings for common errors. Therefore, it is more suitable at development time, and when publishing, just swap the link.

1.4.1.1 Integrating a CDN

In particular, Vue.js is already provided by the project for inclusion in this way. You just need to specify the link given on the project's web page in the *script tag*. This is something like <https://cdn.jsdelivr.net/npm/vue/dist/vue.js>, but of course this can change and the exact details can be found on the Vue.js project's web page in each case (Fig. 1.3).