Preface

Back in the late 1990s, when I first began to professionally develop web applications, we usually developed page by page. Every page was complete, which means self-contained and separated. There was little shared code, maybe only for the connections to data sources and some default application settings. Modifying such an application often meant repetitive changes to all affected pages. For small applications and simple services, this was fine and may still be the cheapest way. Why use an elaborate approach and extensive frameworks, if a simple CGI-script is sufficient to fulfill the requirements?

In time, when web applications grew in size and complexity, this approach was no longer feasible. With growing size of an application, repetitive code is difficult to maintain consistently, and changes become increasingly troublesome and expensive. Developing teams also grew and developers specialized. We began using more and more templates to separate programming logic and view generation. And with improved browser compliance Cascading Style Sheets (CSS) were used to separate layout and design issues from HTML.

Various servers and server-side frameworks competed to provide solutions for common aspects of web development, like routing, authentication and authorization, connection pooling of data sources, templating and reusability of parts of the user interface, and user input sanitation. Applying the Model-View-Controller (MVC) pattern emerged as the new standard. Still, at that time, web application HTTP requests were usually responded to by full HTML pages generated on the server. Caching was the big thing to improve response times and take load of the involved servers.

Client-side JavaScript is a language, which has been part of web development since the early browser versions. It has mostly been used to add dynamic features to otherwise static webpages and to give instant feedback to user input and actions. Long before the XMLHttpRequest and AJAX, data were loaded in a so called hidden HTML frame of 1 pixel height and processed with JavaScript to avoid the repeated loading of full web pages to respond to user interaction. In time, when PCs got faster CPUs and more memory, it became reasonable to have the client hold more data and do some serious work. Faster JavaScript engines and the hype surrounding AJAX brought another push for client-side JavaScript.

More recently, generating and modifying user interfaces completely with client-side JavaScript has become fashionable. Web servers are no longer responsible for generating HTML pages, but mainly for securely and speedily delivering assets and data following authorized requests. The view and controller parts of the application are separated from the server-side and moved to the client. In this setup, the model part has both server and client-side aspects.

Having developed client-side JavaScript features and applications for years, mostly with only rudimentary external library support, I joined a team of developers working with UI5 to build custom shop floor applications. None of us had done much work with UI5 and so we had no experienced lead developer to guide us. The motivation to write this book stems from the experiences gained in these projects, especially the process of finding common ways of handling typical issues. Exploring the depths of UI5 ourselves, we repeatedly found the need to refactor the whole code even at an early stage after discovering how to proceed on a better path than we previously thought.

1. Why this book?

There are only few books about UI5 on the market. I can recommend the "SAPUI5: The Comprehensive Guide to UI5" for a detailed overview and to understand the general structure of UI5. But it is of limited help to quickly find a good way to solve whatever task is at hand.

Working with UI5, the API Reference is an indispensable source of information, but it is just a reference for the hundreds of objects and their numerous functions. The general documentation included in the UI5 SDK is helpful to get started, but lacks practical details and examples. On the other hand, the code samples lack explanation.

UI5 provides a vast number of namespaces and objects with numerous functions. This is confusing and does not help developers beginning to use the toolkit to learn what is relevant to solve the task at hand. Also, as there are many ways to get something done, it is easy to get things wrong and end up with an inflexible code base, difficult to change and test. When developers become hesitant to touch working code and customers get nervous that modifications take too long and cause seemingly unrelated errors to pop up, we know that we are in serious trouble.

I decided to write this book after realizing that UI5 can actually be very convenient to build data-centric web user interfaces. The main goal is to teach you, the reader, how to use UI5 efficiently to build high quality user interfaces. The book does not cover all of UI5, or even most of it. Instead the focus is on practical solutions for specific tasks, more like a user guide than a technical documentation.

2. Who this book is for

This book is mainly for web application developers. To follow the examples in the book requires some familiarity with JavaScript. Certain parts of the book will be difficult to follow without some understanding of asynchronous HTTP requests. Deep knowledge of the Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) is not required, however.

If you are interested in finding out how UI5 works, or if you have to use UI5 in your project, this book is for you. No prior knowledge of UI5 is required, but I would like to think that even developers familiar with UI5 will find plenty of interesting approaches to common problems.

3. Structure of the book

Chapter 1, *Introduction* lets the reader get started with UI5 development. It introduces basic conventions and terminology of UI5 and also explains the conventions followed throughout the book.

In Chapter 2, *Building a basic responsive Web Application*, we build the template application to be used as basis for all further examples in the book. This chapter is demanding because many aspects are introduced, but only briefly explained. It is designed to allow the reader to reproduce the steps and get into the process of developing with UI5.

The next chapters are divided into parts. The first part covers the basic tools and user interface elements provided by UI5. It begins with Chapter 3, *Views and Pages*, giving an overview of the available page layouts and explaining their main features. Then we move on to Chapter 4, *Navigation and Routing* and present ways to restrict access for unauthorized users in Chapter 5, *User Permissions*. Handling user actions and giving feedback are the main topics of Chapter 6, *Events* *and Messages*. The following chapters are concerned with data and how to present and edit them. Chapter 7, *Models* shows how to bind data to user interface elements. How to present, sort and filter lists of data is covered by Chapter 8, *Lists and Tables*. Chapter 9, *Forms and Input Validation* completes this part.

The second part is about solutions to common problems of UI5 development and particularly how to separate domain or business logic from the user interface. It begins with Chapter 10, *Entity specifications: Mapping*, showing a way to organize and provide domain metadata. Then we look at the handling of requests to get the domain data in Chapter 11, *Request Handling*, followed by the presentation of basic objects for these data in Chapter 12, *Custom data objects*. Chapter 13, *Validation* is about validating user input and form data. After this excursion, we finally return to the user interface in Chapter 14, *Form Control Generation*.

The third part is about building an example application. We are building an example order application throughout Chapter 15, *Beginning application development*, Chapter 16, *Edit Order page* and Chapter 17, *Edit Address view*. After that, we look into One Page Acceptance testing in Chapter 18, *Write user interface tests*. We conclude this part with an introduction of how to automate necessary project related tasks, like building a distribution package, generating documentation and maintaining a coding standard in Chapter 19, *Automate development tasks with Grunt*.

4. Online resources for the book

You can download zip archives containing all of the source code listed in the book from the author's website [https://cahein.de/ui5guide/]. The example code is also available at GitHub [https://github.com/cahein/ui5guide/tree/edition_1].

On the website you can also post whatever feedback you have regarding the book. Likewise, if you find errata or other problems with the book, please let me know.

5. Notes on the e-book version

A big problem for anyone creating an e-book are the numerous ways to view them. Many readers use dedicated e-reader devices of different brands and models, while many also use a tablet or desktop computer. The variety of reader/viewer applications further complicates the task. Instead of creating versions for particular devices and distribution channels, this e-book complies with the EPUB® 3.1 specification [https://www.idpf.org/epub/31/spec/epub-spec.html] by the International Digital Publishing Forum.

The code listings in the book are preformatted text displayed in a monospace font. Furthermore, to make the code easier to follow, comments have been added to some of the lines. The drawback of this is that the lines have a fixed width. To minimize the required width, we use a narrow monospace font, which has been embedded in the e-book. If you, nonetheless, find lines being cut off, you have to either decrease the font size or, if possible, increase the visible width of the application window.