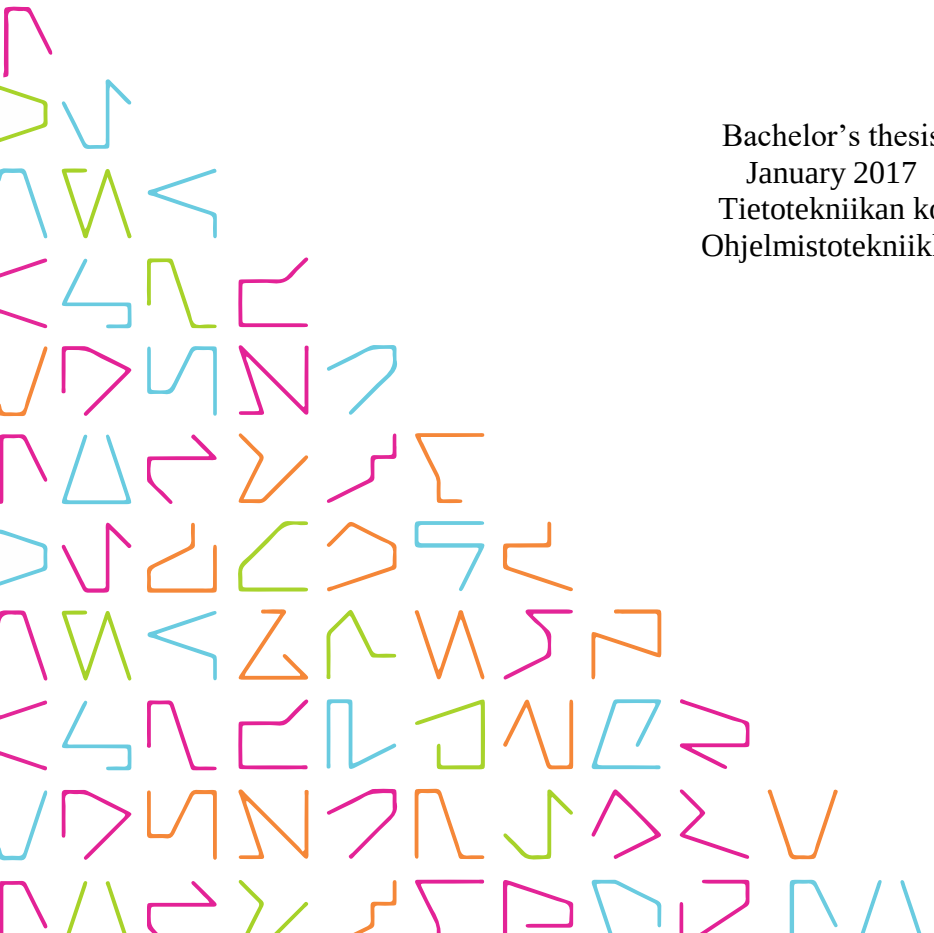
TAMPEREEN
AMMATTIKORKEAKOULU

DOCUMENTATION OF A UI-LIBRARY USED IN WEB DEVELOPMENT

Lauri Annala

Bachelor's thesis
January 2017
Tietotekniikan ko
Ohjelmistotekniikka



TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan ko
Ohjelmistotekniikka

Opinnäytetyö 31 sivua, joista liitteitä 0 sivua
Toukokuu 2017

LAURI ANNALA:
Web-sovelluskehityksessä käytettävän käyttöliittymäkirjaston dokumentaatio

Opinnäytetyö käsittelee tekijän toteuttamaa käyttöliittymäkirjaston dokumentaatiosivustoa. Työn aiheena oli toteuttaa Web-sovellus, jonka tehtävä on dokumentoida käyttöliittymäkirjastoa. Työn tuloksena syntyi sovellus, jota päivitetään käyttöliittymäkirjaston kehityksen yhteydessä. Sovellus esittelee kirjaston komponentteja, tyylejä ja muita ominaisuuksia, sekä kirjaston käyttöön liittyviä seikkoja. Tämä opinnäytetyöraportti esittelee sovellusta, sen kehittämisprosessia ja eri teknologioita joita sovelluksessa ja käyttöliittymäkirjastossa on käytetty.

Opinnäytetyöraportissa esitellään käyttöliittymäkirjastossa ja dokumentaatiosivustossa käytetyt teknologiat. Raportti esittelee dokumentaatiosivuston rakenteen ja arkkitehtuurin, sekä selittää kehitystyöhön liittyvät prosessit, työkalut ja työskentelytavat. Raportti myös käsittelee työn aikana hyväksi todettuja ohjelmiston kehittämismenetelmiä ja työkaluja.

Avainsanat: dokumentaatio, web-kehitys, käyttöliittymäkirjasto

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software technology

Bachelor's thesis 31 pages, appendices 0 pages
August 2015

Lauri Annala:
Documentation of a UI-library used in web development

The thesis deals with documentation site developed by the author. The subject of the study was to implement a Web application which provides documentation for a UI-library. The work resulted in the application, which will be updated in parallel with the continuous development of the user interface library. The application presents the library's components, styles, and other features, as well as aspects of the operation of the UI-library. This thesis report presents the application and the different technologies that have been used in the application and the UI-library.

This thesis report describes the different technologies used in the library and the documentation web application. It goes through the design and architecture of the application and also explains the development process, tools and best practices in web development. Thesis also describes good software development practices and tools that were used during this work.

Key words: documentation, web development, user-interface library

TABLE OF CONTENTS

ABBREVIATIONS AND TERMS

UI-library	User interface library
MVC	Model-View-Controller paradigm
Web development	Development of websites or web applications
Framework	Provides functionalities and solutions for particular area
API	Application Programming Interface - Set of functions and procedures that allow access to a set of services, data or systems
IDE	Integrated Development Environment - Software application that provides comprehensive solution for writing, maintaining and debugging software
Intellisense	Visual feature in text editors and IDE's that help you learn about the code, used frameworks and libraries
IIS	Internet Information Services – Microsoft's web server software
Lint, linting	Various toolsets that flag suspicious and possible bugs in source code
TSLint	Specific linting tool for Typescript programming language
Route, routing	Web-page structure is based on routing, a specific route is defined in the page's URL

1 PREFACE

Thesis describes the creation process and resulting product of the documentation created for a UI-library used in web-development. The documentation is implemented as a web-application that is also using the UI-library that it provides documentation for. Documentation application uses several 3rd party libraries and frameworks, most of them already included in the UI-library. This thesis aims to describe and explain the usage of these frameworks, libraries and technologies. Thesis also provides an insight on how this documentation application was designed and developed and how it is aimed to be used as a tool for developers, designers and product owners that are using or are interested in the UI-library.

2 DESIGN GOALS & AUDIENCE

2.1 Design

The documentation application is aimed for developers, designers and product owners. It should be easy and fast to use by providing code examples of the correct usage of the UI-library's components, styles and other features. This goal is achieved by providing a live style guide, showcasing example usage of the library and providing source code to implement similar use-cases. The provided code example is also the source code that is used in the demonstration of the component. With this approach it is very convenient for the user to see the actual result of the code in a web-application. Documentation for the style library is also included in the application. All variables and custom overrides are showcased in the style-section of the documentation application. Style library uses a custom Bootstrap library that has modified values and definitions.

2.2 Goals

The documentation application should help the users of the UI-library to use it correctly. It should provide examples of the correct usage of the components and be a quick reference of the styles and variables provided by the UI-library. Documentation also provides brief description, guidelines and other resources for the people and organizations interested in using the UI-library. The documentation should be easy to approach, having all the basic information the user needs to set up a specific component or style. It should also provide all the detailed information for the user. All this is achieved by providing descriptions and API-documentation for the components of the UI-library, and by having a demonstrations and source codes available for the user to inspect.

2.3 Audience

Audience, the users of this documentation are developers, designers and product owners that are using or are interested in the UI-library. They should have basic knowledge of the languages, main frameworks and tools the UI-library depends on. Main frameworks are the libraries that are utilized in the UI-library to provide layouts, styles and functionalities. AngularJS framework, Typescript language and Bootstrap library. Users should

have basic skills in HTML, CSS and Javascript. The technologies, frameworks and libraries used in the UI-library, as well as the documentation application, are briefly introduced and described in the corresponding technologies- sections of this thesis report.

3 TECHNOLOGIES

This section describes the technologies that are used in the UI-library. The basic web technologies: Javascript, CSS and HTML are described in detail. UI-library also uses more advanced technologies like AngularJS and Typescript which are all also discussed in this section.

3.1 JAVASCRIPT & ES6

3.1.1 Javascript

Javascript has grown to be one of the most widely used programming language of the world. The web has grown to be enormous and the language of the web is Javascript. Every browser can interpret Javascript is one of the main reasons the language is used so widely. As a programming-language it is a high-level, dynamic, untyped and interpreted language.

High-level means that the language aims to abstract the deep functionality of the computer, like memory management of the machine. Reason for this is to provide language features that are easier and faster to use in applications where managing the low-level functionality of a computer doesn't matter that much.

Dynamic means that the language executes some of its features and behaviors at runtime, when code is executed at the browser. In contrast to static languages that would execute it's features and behaviors when the source code is being compiled, when the code is transpiled into machine code or to some other lower-level language.

Untyped means that in the language there are no type declarations. Typed languages like C use keywords for typing, like `int` for integer numbers and `string` for text. Note that there are a few definitions for an untyped language. One definition of untyped is that the language doesn't tag values and treats everything as bits. Javascript doesn't suit for this definition. Low-level languages like Assembly are considered untyped in this sense. The other definition is from Programming Language Theory: "In this domain, untyped just means everything belongs to a single type. Why? Because a language will only generate a program when it can prove that the types align.

This means in an untyped language:

- A program is always generated
- Therefore types always match up
- Therefore there must only be one type

In contrast to a typed language:

- A program might not be generated
- Because types might not match up
- Because a program can contain multiple types

…in PLT, untyped just means dynamically typed and typed just means statically typed. JavaScript is definitely untyped in this category." – Brian McKenna, Stack Overflow.

Interpreted means that the language doesn't need to compiled before executing. The browser interpreter that executes the code translates each statement into subroutines that are already compiled into machine code.

Javascript is a multi-paradigm language supporting object-oriented programming with its prototype-based first-class-functions. Javascipt also supports scripting, imperative, functional and event-driven programming styles. Scripting is a programming style where a common task in specified environment is executed automatically. One example of a scripted application would be a program that installs an application to a server machine. Imperative programming style is common in low-level language and machine code. Code is focusing on how the program executes and it is based on statements and functions. In functional programming style code is built on functions that avoid changing the state and mutable data. It is a declarative paradigm that uses expressions or declarations instead of statements. Event-driven programming focuses on the behavior and flow of the program determined by events. In Javascript, this style is commonly used when the program handles user input.

Javascipt's API provides support for text, arrays, dates and regular expressions. API doesn't support I/O, networking or storage or graphics. Host system, commonly the browser is responsible of these unsupported features.

3.1.2 ECMAScript6

ES6, ECMAScript6 or ECMAScript 2015 is the latest version of ECMAScript standard. It is not yet supported by modern browsers, but tools like BabelJS and Typescript can compile ES6 to ES5, which is supported by browsers. ES6 is very large update to the language overall. Notable features are arrow and lambda functions, template strings, modules, module loaders, promises and new classes that are easier to use than classic Javascript prototypes.

3.2 HTML

Hyper Text Markup Language is the standard markup language used in web pages and applications. Browsers can render a multimedia web page by interpreting this markup content. HTML describes the webpage semantically and provides its structure. HTML can also be embedded in Javascript-files that define the behavior and content of the webpage. HTML can also include CSS stylesheets that define the look and layout of the webpage.

3.3 CSS & LESS

3.3.1 CSS

Cascading Style Sheets is a style sheet language that defines the presentation of a markup language document such as an HTML document. CSS is commonly used in describing styles and layout of a web page. CSS rulesets can define for example the color, paddings and margins of an element with style properties. The elements that specific rulesets are applied are defined with selectors like classes or id:s. In larger projects using the plain old CSS can become increasingly cluttered and hard to maintain. CSS preprocessor languages like SASS and LESS have been designed to achieve better maintainability and add new language features like mixins and nesting of rules. Latest technologies have introduced new ways to achieve maintainability and separation of components. For example, with Vue.js library you can define isolated CSS rules for each component. The documented UI-library uses LESS as a CSS preprocessor.

3.3.2 LESS

LESS is a dynamic superset of CSS that can be compiled into CSS which then can be interpreted by browsers. LESS provides several new features to style sheets: variables, nesting, operators, functions and mixins which can contain several property definitions as a ruleset. Variables are a simple but important update to CSS that can increase maintainability greatly when specific values can be reused in several components. If it is decided to alter the values later, the refactoring can be done to a single variable instead of changing the values separately in several places. Nested rulesets offer convenience for writing structured style rules, but deep nesting can also reduce readability. Operations and functions are also available to stylesheets when using LESS.

3.4 ANGULARJS

AngularJS is a Javascript framework for building single page applications and dynamic web pages. By using Angular the developers can extend HTML vocabulary to build dynamic view content which HTML originally was not designed for. AngularJS makes developing dynamic web applications easier and faster.

In the moment of writing, Angular has already released their new version of Angular 2. However, the documented UI-library and the web application still use Angular 1. Angular 2 simplified and improved Angular and it is entirely separate new framework.

Transition from Angular 1 to Angular 2 is a high priority, since Angular 1 is much older and inferior compared to newest Javascript frameworks like Angular 2, ReactJS or VueJS. Angular 2 introduces many improvements and provides a fully-fleshed, modern JS framework. It is tempting to compare Angular 2 with other popular Javascript libraries like VueJS or ReactJS. Angular 2 is fundamentally different in a sense that it provides a full MVC-framework as opposed to Vue.JS or React which both provide only a View-library. These other libraries, VueJS and React, can be extended with additional libraries to achieve the same sort of full MVC solution similar to Angular 2. This difference might be one important aspect when choosing the best framework for specific purpose, Angular 2 provides a full solution with pre-defined conventions and best-practices. In the other

hand, VueJS and ReactJS can provide a flexible View-layer solution for your existing project, or you can more freely develop a custom solution using these libraries. In short, Angular 2 provides full, instantly ready-to-use MVC framework with less freedom, VueJS and ReactJS provide flexible View-layer libraries which can be integrated to existing projects and solutions with specific custom needs.

3.5 TYPESCRIPT

Typescript is a superset language to ECMAScript 6 that compiles to Javascript which then can be executed in browsers. Typescript introduces types to Javascript which can enhance the workflow in larger projects and provide IDE features such as advanced autocompletion and refactoring in Javascript. Typescript is a viable choice to neglect time-consuming issues that arise from dynamic languages. Types can prevent the application to end up in faulty behavior. Features and architecture can be described, documented and shared to other developers better with types. Typescript's tooling can save a lot of time by preventing common mistakes caused by human error. For example the Typescript compiler can be very helpful when you define your types, classes and interfaces. Developers can get error reports before running the code and tools like intellisense present the APIs for you conveniently. When using Typescript you can interactively inspect library interfaces from your editor and take benefit from auto-completion.

4 TECHNOLOGIES USED IN THE DEVELOPMENT

This section describes the technologies that are used in the development of the UI-library. These technologies are included in the UI-library's source code. The distributed UI-library package includes the libraries, documentation application and their dependencies.

4.1 NodeJS & NPM

NodeJS is a Javascript runtime environment used widely in web applications. NodeJS also includes its package-manager, npm (node package manager) which is the largest ecosystem of open source libraries and packages. NodeJS is designed for scalable network applications. It's asynchronous, event-driven and has non-blocking I/O model. In our application, NodeJS is used to build the application and manage its packages and dependencies. NodeJS is also used as a platform of the development server. NodeJS also supports ES6 features.

4.2 GULP

Gulp is a task runner or a build tool used in NodeJS applications. It can help with automating common tasks like setting up a development server or building production or development software from source code. In front-end development, some common tasks for gulp would be to automatically reload the browser when there are changes to source code during development. Compiling Typescript to Javascript or LESS/SASS to CSS.
In our application, gulp builds the library, minifies and creates sourcemaps for it, builds the documentation and executes tests, it also provides a local development server that serves the documentation web application. Development server supports live reload on code changes as well. Build system has separate tasks for production like minifying, and tasks for development like building source maps for better debugging experience.

4.3 LODASH

Lodash is a utility library that has different methods like forEach, forOwn, debounce, map and many more. In our application some of these methods are used in the build system and the UI-library typescript source files. While ES6 is increasing its popularity, the relevance of Lodash could be questioned since many of its features are now available in the

newest version of Javascript. However, our documentation application's Angular application doesn't use ES6 at this point. Some lodash array methods could be replaced by their ES6 equivalents in the NodeJS build system because it already has ES6 set up. Lodash still has many unique features that are not covered in the ES6 core, like array functions working on Javascript objects as well, and having several not so common, but very useful utilities, like debounce and throttle. Lodash also has very good optimizations in the library that can improve the overall application performance.

5 DESCRIPTION OF THE DOCUMENTED UI-LIBRARY

The UI-library offers custom bootstrap styles and css variables in its style library. It also provides custom UI-components using AngularJS-framework. Style-library includes bootstrap custom overrides, css variable definitions like paddings, margins and screen-width breakpoints used in the stylesheets. Angular-library includes UI-components like datepicker, dropdown-panel, list and many more. The UI-library package also includes the documentation application. In development, source files include all typescript and less files that produce the core library as Javascript and CSS. Source files also include the build system, unit tests and source files needed to build the documentation application.

The UI-library greatly increases the efficiency of the development. Using a separate library in the product supports separation of concerns in the architecture. This will increase modularity, maintainability and it will also enhance development workflows in larger projects. Components with clear API's promote coherent usage across the platform which will increase code readability and maintainability. The UI-library has a dedicated team that maintains the package and continuously adds new features and fixes bugs in the library. The team is challenged to understand the use cases for the library and the users should provide feedback so the team can improve the library based on it.

6 BRIEF EXPLANATION OF THE DEVELOPMENT PROCESS OF THE DOCUMENTATION APPLICATION

Important goal during the development of the documentation application was to achieve easy maintainability. The software architecture was based on this requirement. Application doesn't use any database and all the data is determined in the build process. The benefit of this is that the documentation application is included in the UI-library package. The application can be served in server as static files requiring minimal amount of configuration and effort. The application can also be maintained and developed easily alongside with the UI-library. This application aims to support the developer, so local development server with live reload is also included in the software. The main libraries the application uses are already included in the UI-library and accessible there. But some libraries are specific for the documentation app, like Angular-scroll and HighlightJS.

The build system is implemented using NodeJS and gulp plugins. The build system is responsible of making a tar-file that includes the UI-library and the documentation that uses that library. Documentation application uses the same external libraries as the UI-library.

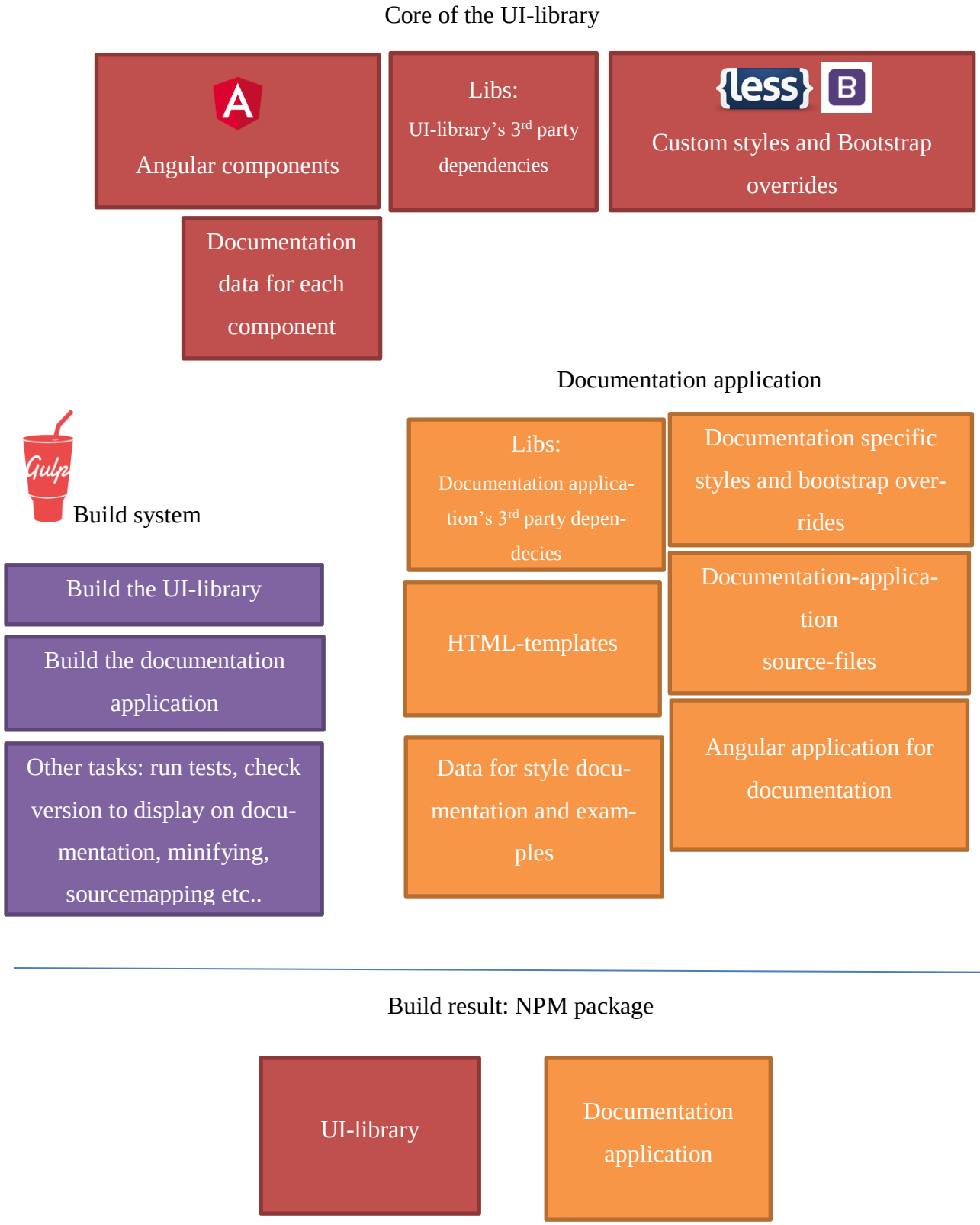
7 ARCHITECTURE

Documentation application is a static package that builds its data from source files. This data is injected into templates. The UI-library and its dependencies are documented and many parts of them are also included in the documentation applications page styles, layout and behavior. If certain data from the source is not directly available in the libraries, it is initialized to the applications controller through the injected templates.

The documentation application essentially consumes the documented UI-library and describes the use-cases to the viewer of the web page. The goal of the architecture is to use the documented library as simply and efficiently as possible. The application uses very few additional libraries and these are mainly used in the view layer based on the special needs of the documentation.

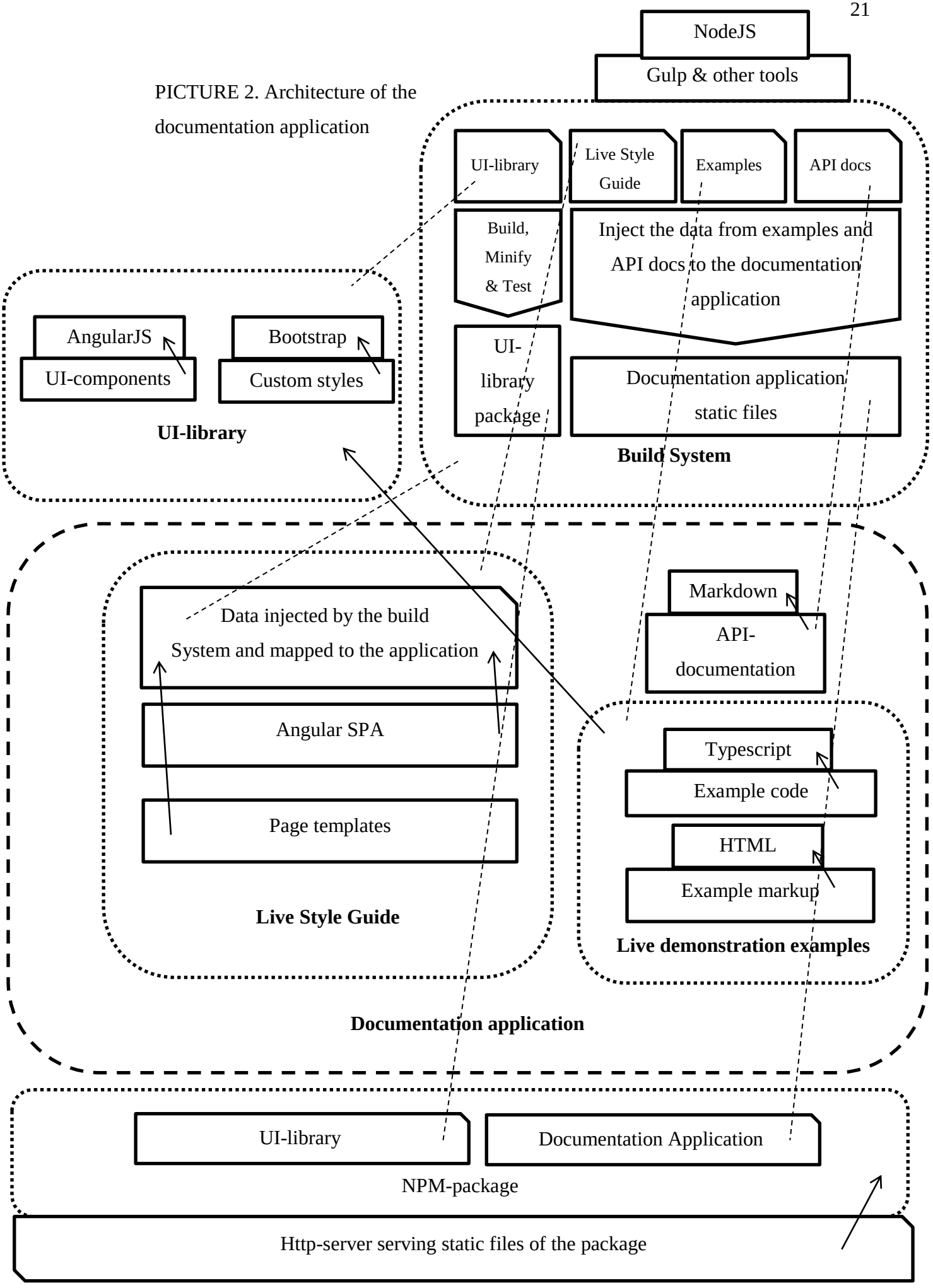
The build process is the core part of the architecture because the used data is based on static source files that are handled in the build. The application doesn't have any back end that would server the data dynamically. This is because the application does not have any interaction with the user outside of the navigation and the documented components. When components are required to use some sort of business data it is mocked in the demonstration source code.

The application is a single-page-application that is served statically from the server. The application doesn't have any specific requirements for the server and it can be served straight from the filesystem if it is desired. In practice, the files are either served locally from the NodeJS development server or in production from an IIS server.



PICTURE 1. Technologies of the documentation application

PICTURE 2. Architecture of the documentation application



8 DEVELOPMENT TOOLS

8.1 Gulp & TSLint

When writing code in a team, people usually adapt to certain style guides that the team should follow. Linting tools help with this by automatically proofing the source code. Lint is a software tool that reports any suspicious usage in source code. Linting tools are available for different programming languages. Users and teams can define their own rulesets that will cause linting to fail. Style Guides often define different coding styles that should be followed and a linting tool can help to accommodating to these rules.
If there are some linting errors in the code, eg. unused variables or empty constructor, author should notice and fix the problem before these changes would end up to version control. In continuous integration, where the UI-library is automatically built by gulp, these linting errors will fail the build. Gulp provides easy tools to run tslint automatically each time there are code changes.

8.2 Version control & Semantic Versioning

Git was used as a version control system. As a branching model we adopted widely used Vincent Driessen's model. It uses two main git branches: develop and master. Develop-branch has new features for the next releases, feature and bugfix branches are usually merged to develop branch. Master branch represents the features ready for production. Release branches are checked out from master branch with release version tag, eg: "1.0.1". Version numbers are based in semantic versioning.

8.3 Source mapping

When the source code is built, Typescript is converted to Javascript and LESS is converted to CSS. The source code is minified and other procedures are executed like concatenating the source code. The result might be impossible to read and to track down bugs. It is preferable to keep the client-side code readable and debuggable while developing the software. Source maps help with this as they map the code back to readable and debuggable form. Developer tools like Google Chrome dev tools will automatically use the generated source maps to map the code. The application uses gulp to make source maps and as result, the source code readable again and it is possible to debug the original Typescript source code and inspect the LESS stylesheets.

9 IMPLEMENTATION OF THE APPLICATION

9.1 Concept, inspiration & requirements

Documentation application is included in the UI-library source code and build result package. Application doesn't have detailed requirements for the back-end or the server. Build package can be served statically from the server when making sure to allow certain mime-types like fonts and animations to be served from the server. Application is very easy to be setup as a local development server with node because the build system offers an automatic command to set it up: `gulp sync`. Static files can be served in simple NodeJS http-server or IIS-server with minimal configuration. Page layout and general architecture of application was originally inspired by UI-Bootstrap's documentation application. There are many differences, one large difference are the tools that are used. UI-Bootstrap uses grunt in contrary to our project using gulp. Grunt in configuration based build-system and gulp is more like a streamed task-runner. Somewhat similar design and architecture is used in how the documentation is created and maintained. Documentation consists a series of descriptions and guidelines as a readme-file (md), a live example of the usage of component that consists of markdown(html) and typescript. These files are included in the UI-library's source code, exactly in the corresponding component's `docs`-folder. The main goal of the application is to provide a living style guide that is constantly updated along with the UI-library.

9.2 Developing the build system

Build system consists of several gulp tasks for different purposes. Tasks can build the source code as development or production build, run unit tests and build the documentation application. Documentation build system takes all the sources of the documentation, transpiles the Typescript- and LESS- files and builds them into the html-templates using Lodash and Angular's template cache. Angular loads different pages, like components and styles pages from template cache depending on the client's current route.

9.3 Developing the documentation page: layout, html & css

Documentation application has its own stylesheet but it is not supposed to override or alter the behavior of the UI-library. Documentation applications own stylesheet are used strictly for defining the presentation of the documentation. AngularJS components have markdown, html and typescript source-files and they create the examples and documentation for each component. Build system takes care of any necessary navigation, categorizing and component specific layouts.

9.4 Developing the documentation page: Angular, Javascript, Lodash & Templates

Documentation application has it's own Javascript source code using Angular that takes care of navigation and routing. Some of the application logic is based on the data that is injected to the Angular application through templates. This mechanism relies of the build process where the required data is determined and templates are formed based on that data.

9.5 Developing the documentation page: examples and making the documentation

Documentation aims to provide enough information for the user to know how the component or style should be used. User should also have clear knowledge of how the component looks and how it works. Detailed API-documentation, such as specific attributes or optional behavior should be available and demonstrated in live demo. Demonstration's source code should be easy to understand and it should offer all the source code needed, typescript, less and markup.

10 DEVELOPMENT PROCESS, DEPLOYMENT & BEST PRACTICES

10.1 Agile & Scrum

The development of the documentation library was done in parallel with the development of the UI-library. The most crucial part of any team work is communication. Our team had the traditional daily scrum meetings and retrospectives.

Scrum meetings emphasize the importance of continuous communication. In a daily scrum meeting each team member explains what they have done yesterday, what they are going to do today and if they have any problems. The meeting is supposed to be fast and brief, lasting around 15 minutes. This time is not dedicated to solve problems. After a scrum daily meeting every member of the team gains good understanding of what work has been done and what work remains. When a team member announces their commitment to a certain task the commitment is made for the team and the team will know the next day if the task is completed. This creates a certain sense of commitment for each member of the team.

In the scrum retrospective the team discusses the recently concluded sprint. Sprint stands for a period of time in which the team has agreed to complete certain set of tasks. In scrum retrospective team discusses on what went well during the latest spring cycle, what went wrong and what could be done differently to improve the efficiency and quality of the next sprint. In sprint retrospectives the team can provide feedback and discuss different ways to improve efficiency and quality of their work.

10.2 Development & Deployment

During development of the documentation application and the UI-library there were certain best practices the team members followed. In addition to scrum daily meetings and retrospectives the team members also did code reviews to new features. Before new changes were added to the source code, other members reviewed the new code added for a new feature. This guaranteed to keep the established code quality and improved the common understanding of the source code when members could question and reason about the source code. Some small oversights and mistakes could already be caught in the code review process.

When code is pushed to the repository the continuous integration system could spot regression bugs and errors when the developer was implementing a new feature. This continuous integration process guaranteed that these errors could be spotted and addressed early.

10.3 Perspective & Insights

The team should be willing to deeply understand what the real customer needs from their product. It should be clear to understand who the customer is in the first place. It is a common misconception for a developer to solely implement features for their manager or product owner. But understanding the needs of the actual users of the product is the most crucial and challenging skill to acquire. In the perspective of the documentation application the customer is only indirectly the user of the business product, the application that uses the library. The primary customer of the documentation application are the developers and product owners that are using it on their product or are interested in doing so. It is crucial to truly understand who is the real customer and what their needs are.

11 PRODUCTION

11.1 Minifying & concatenating

For production there are various tasks to make the resulting package as lightweight as possible. These tasks include source code minification which removes all the unnecessary characters from the source code. This means that everything that doesn't change the logic or functionality of the program and only exists for human readability, eg. white space characters, new line characters and comments will be removed.

```
var x = 10;

function createFunction1() {
    var x = 20;
    return new Function("return x;"); // this |x| refers global |x|
}

function createFunction2() {
    var x = 20;
    function f() {
        return x; // this |x| refers local |x| above
    }
    return f;
}

var f1 = createFunction1();
console.log(f1());          // 10
var f2 = createFunction2();
console.log(f2());          // 20
```

PICTURE 3. Javascript code before minification

```
var x=10;function createFunction1(){var x=20;return new Function("return x;")}
function createFunction2(){var x=20;function f(){return x}
return f}
var f1=createFunction1();console.log(f1());var f2=createFunction2();console.log(f2())
```

PICTURE 4. Javascript code after minification

Instead of including multiple source files in html, all the different library source files can be concatenated in one file. In our application, 3rd party libraries are concatenated to one file, eg: libs.js and all application source files to its own concatenated file.

11.2 Continuous Integration

When developer start to implement a new feature or a bug fix and pushes new code to the version control repository, the continuous integration will trigger certain tasks. Certain set of automatic tests are run in the continuous integration system. The benefit is fast feedback and early detection of regressions and erroneous behavior of the software. When new features are approved and pushed to the main repositories like the main release or development branch, a new version of the documentation is also build and copied to the official documentation server. This way the documentation is always up to date and doesn’t require any manual intervention on procedures unless there are some problems in the underlying system.

12 CONCLUSION

In the future, the main challenges are to unify the documentation application with other guidelines used in the organization. Also, Angular 2 has recently been released and in the future the UI-library will be updated to use that. Same applies when bootstrap is updated to version 4. In web-development new attractive technologies are released and updated each year and updating the dependencies for the documentation application and the UI-library is one part of maintaining the software. Switching to new libraries needs to be carefully studied first. In the future the documentation application could be extended to have live editing features that would enable the user to edit the demonstration examples directly in the browser.

SOURCES

Stack Overflow: Is Javascipt an untyped language. Read 24.10.2016
http://stackoverflow.com/questions/964910/is-javascript-an-untyped-language

Luke Hoban: ES6 features. Read 24.10.2016
https://github.com/lukehoban/es6features

Zell Liew: Gulp for Beginners, Read 10.11.2016
https://css-tricks.com/gulp-for-beginners/

Ryan Seddon: Introduction to Javascript Source Maps. Read 17.11.2016
https://www.html5rocks.com/en/tutorials/developertools/sourcemaps/

Vincent Driessen: A successful git branching model. Read 17.11.2016
http://nvie.com/posts/a-successful-git-branching-model/

Kitson Kelly: Typescript, WTF?! Read 17.11.2016
https://davidwalsh.name/typescript

Derick Bailey: Does ES6 mean the end of Underscore/Lodash? Read 17.11.2016
https://derickbailey.com/2016/09/12/does-es6-mean-the-end-of-underscore-lodash/

John Fawcett: Generating Documentation for TypeScript Projects. Read 23.11.2016
https://blog.cloudflare.com/generating-documentation-for-typescript-projects/

Mountain Goat Software – Scrum. Read 15.4.2017
https://www.mountaingoatsoftware.com/agile/scrum/