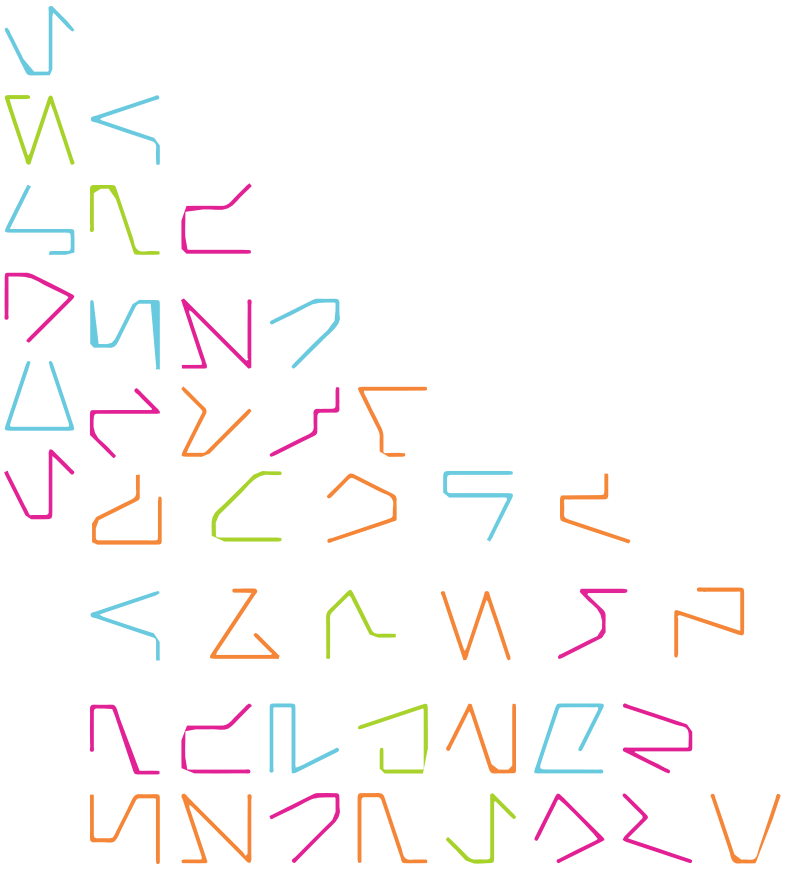
DOCUMENTATION OF A GOALS COMPONENT USED IN WEB DEVELOPMENT



ABSTRACT

Documentation of a UI-library used in web development

The subject of the study was to implement a Web application which provides documentation for Goals component. 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 in web development.

TABLE OF CONTENTS

ABBREVIATIONS AND TERMS

UI-library	User interface library
Web development	Development of websites or web applications
CSS	Cascading Style Sheet
HTML	Hyper Text Markup Language
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
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 Goals component 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, most of them already included in the UI-library. This thesis aims to describe and explain the usage of these 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 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.

2.1 JAVASCRIPT & ES6

2.1.1 Javascript

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the
scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles.

JS(ECMA Script) is a like verb(anything) you do on a website, tweet, play video, etc. It's a file, a file in which we can write instructions to the computer.
It's a client side scripting language. In developers tool -> console, is the space where we can write JS and also in a HTML file.
HTML file is a static file. JS brings the dynamics in the web page.
JS without any libraries is called as Vanilla JS. It is a case sensitive language. It is a scripting language. It reduces the load on the server as some
operations are done on the client side. It is used as default scripting lang in HTML. Code is interpreted/Just In Time(JIT) compilation.

In JS we get a big bucket. The bucket has all the elements, like buttons, input types, radio buttons, etc. This bucket in JS is known as DOCUMENT. DOCUMENT is a
ready made object in JS. Using this DOCUMENT we can access all the elements on the page. The DOCUMENT has many methods defined which help us reach to the particularelement.
EG: document.getElementById(); document.getElementsByTagName(), etc. We can access the elements using their attributes like id, class, etc.

2.1.2 ECMAScript6

ES6 refers to version 6 of the ECMA Script programming language. ECMA Script is the standardized name for JavaScript, and version 6 is the next version.
Web browser support for the full language is not yet complete, though major portions are supported. Major web browsers support some features of ES6. However, it is
possible to use software known as a transpiler to convert ES6 code into ES5, which is better supported on most browsers.

ECMA Script6's covers the following new features:

- Support for constants
- Block Scope
- Arrow Functions
- String Templating
- Enhanced Object Properties
- Classes
- De-structuring Assignment

1. Support for constants: Constants are values that can be defined only once per scope. A re-definition within the same scope triggers an error.

2. Block Scope: With ES6, variables declared using "let", follow block scoping rules just like in Java, C++, etc. Before this update, variables in JavaScript
were function scoped. That is, when you needed a new scope for a variable, you had to declare it within a function.

3. Arrow Functions: ES6 introduces arrow functions to JavaScript. These are similar to traditional functions, but have a simpler syntax.

```
Eg: var x = a => a + 1;
        x(4) // returns 5
```

4. String Templating: String templating refers to interpolating variables and expressions into strings using a syntax like perl or the shell. A string template is enclosed in back-tick

characters (`). By contrast single quotes (') or double quotes (") indicate normal strings. Expressions inside the template are

marked out between ${ }.

Eg: var name = "joe";

```
var x = `hello ${name}`
// returns "hello joe"
```

5. Enhanced Object Properties: ES6 brings a simplified object creation syntax.

Eg: var x = "hello world", y = 25

```
var a = { x, y }
// is equivalent to the ES5:
{x: x, y: y}
```

6. Classes: JavaScript gets a formal class definition syntax. It serves to enhance code clarity.

Eg:

```
class Circle {
        constructor(radius) {
                this.radius = radius
        }
}
```

7. De-structuring Assignment: The destructuring assignment syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.

2.2 HTML

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. HTML can embed programs

written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.

2.3 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.
CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility,
provide more flexibility and control in the specification of presentation characteristics.

2.4 REACTJS

ReactJS is JavaScript library used for building reusable UI components.
React is a library for building composable user interfaces. It encourages the creation of reusable UI components, which present data that changes over time. Lots
of people use React as the V in MVC. React abstracts away the DOM from you, offering a simpler programming model and better performance. React can also render on
the server using Node, and it can power native apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is easier
to reason about than traditional data binding.

React Features:

1. JSX: JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.
2. Components: React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale
projects.
3. Unidirectional data flow and Flux: React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your
data unidirectional.

React Advantages:

Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.
Can be used on client and server side as well as with other frameworks.
Component and data patterns improve readability, which helps to maintain larger apps.

React Limitations:

Covers only the view layer of the app, hence you still need to choose other technologies to get a complete tooling set for development.
Uses inline templating and JSX, which might seem awkward to some developers.

React JSX:

React uses JSX for templating instead of regular JavaScript. It is not necessary to use it, however, following are some pros that come with it.
1. It is faster because it performs optimization while compiling code to JavaScript.
2. It is also type-safe and most of the errors can be caught during compilation.
3. It makes it easier and faster to write templates, if you are familiar with HTML.

3 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.

3.1 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 DESCRIPTION OF THE DOCUMENTED UI-LIBRARY

The UI-library offers custom bootstrap styles and css variables in its style library. Style-library includes bootstrap custom overrides, css variable definitions like paddings, margins and screen- width breakpoints used in the stylesheets. The UI-library package also includes the documentation application. In development, source files include all files that produce the core library as Javascript and CSS. Source files also include the build system 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.

5 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 HighlightJS.

6 CONTENT STRUCTURE

Documentation application is a static package that builds its data from source files. The create-react-app tool was launched by Facebook and is a recommended way of setting up a new project. To create a new app/project using this tool, all we need to do is run the command "create-react-app" followed by the app name.

>>create-react-app goals_vanguard

After running the above command, a new folder called "goals_vanguard" will get created and that would have all of our application code.

Project Layout

```
├── README.md
├── node_modules
├── .gitignore
├── build
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
├── src
    ├── components
        ├── goals.js
        └── sideNav.js
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── serviceWorker.js
├── package.json
└── package-lock.json
```

Build represents the path to our final production build. This folder would actually be created after we run the npm build. All the "dependencies" and "devDependencies" required by the React app in node_modules. These are as specified or seen in the package.json file. When the run command is given as ls -1, it'll show almost 800 sub-directories. This directory gets added to .gitignore so it does not really get uploaded/published.

All of the dynamic components will be located in the src. To ensure that, at the client side, only the most recent version is downloaded and not the cached copy, Webpack will generally have the updated files a unique file name in the final build. We can also see files like App.js which is kind of our main JS component and the corresponding styles go in App.css. index.js is the entry point for our App and it triggers the registerServiceWorker.js. And also a 'components' directory here to add new components and their associated files, as that improves the organization of our structure.

The overall configuration for the React project is outlined in the package.json. Below is what that looks like:

```
{
 "name": "my-sample-app",
 "version": "0.0.1",
 "private": true,
 "dependencies": {
        "react": "^16.5.2",
     "react-dom": "^16.5.2"
 },
 "devDependencies": {
        "react-scripts": "1.0.7"
        },
        "scripts": {
        "start": "react-scripts start",
        "build": "react-scripts build",
        "test": "react-scripts test --env=jsdom",
        "eject": "react-scripts eject"
 }
}
```

Figure 1: package.json

The dependencies which are shared in the application can go to the assets directory. These can include mixins, images, etc. Thus, they would represent a single location for files external to our main project itself.

The component directory structure is the most important thing in any React app. While components can reside in src/components/my-component-name, it is recommended to have an index.js inside that directory. Thus, whenever someone imports the component using src/components/my-component-name, instead of importing the directory, this would actually import the index.js file.In components material-ui/core, react-chartjs-2, proptypes are imported and used.

From react-chartjs-2, pie chart and line graph are used in goals component. Below is what that looks like:

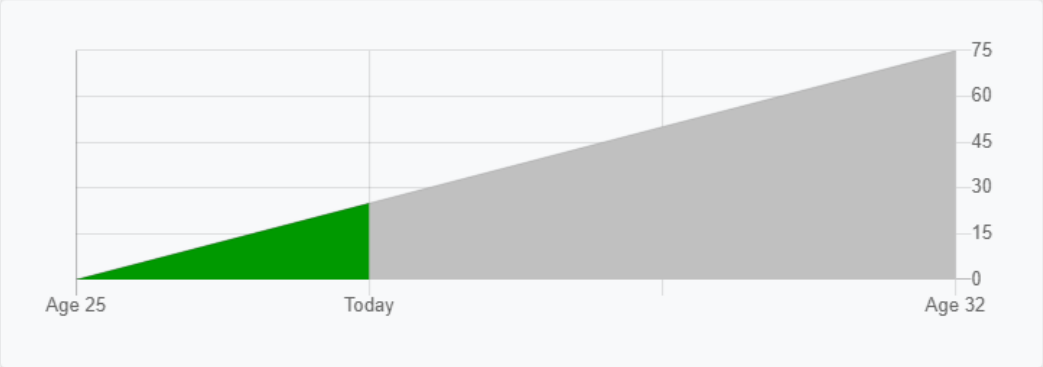


Figure 2: Line Graph chart

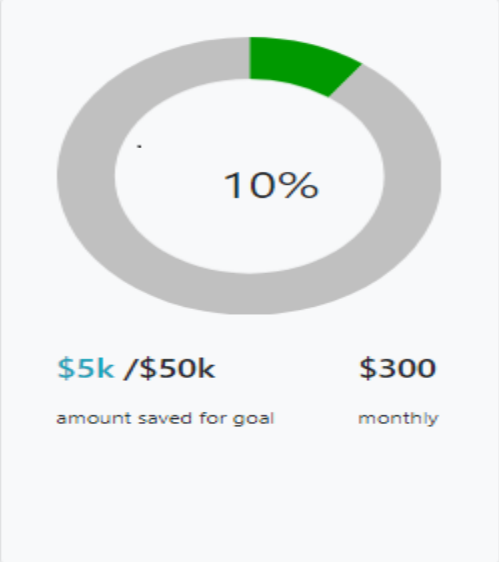


Figure 3: Doughnut Pie chart

7 IMPLEMENTATION OF THE APPLICATION

7.1 Concept & 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. 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. 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). 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.

8 CONCLUSION

In the future, the main challenges are to unify the documentation application with other guidelines used in the organization. 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

Chart.js: Referring properties for Doughnut chart
https://www.chartjs.org/docs/latest/charts/doughnut.html

Chart.js: Referring properties for Line graph chart
https://www.chartjs.org/docs/latest/charts/line.html

Tabs: SideNavbar from material ui
https://material-ui.com/api/tabs/

Stack Overflow: Setting border Width for doughnut chart
https://stackoverflow.com/questions/36339791/how-can-i-remove-the-white-border-from-chart-js-pie-chart/36339979