Tatu Kalermo

Building a Design System

Metropolia University of Applied Sciences

Bachelor of Engineering

Software Engineering

Bachelor's Thesis

10.5.2021

Abstract

Author:	Tatu Kalermo
Title:	Building a Design System
Number of Pages:	21 pages
Date:	10 May 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Software Engineering
Supervisors:	Janne Salonen, Head of Major

The goal of this thesis was to showcase creation of a Design System using Storybook and explore the importance of Design Systems.

Starting point of the thesis was a design library made by a designer in Figma. In order to achieve the goal of transferring the Design System from Figma to Storybook, research about different component libraries and Storybook was carried out.

Transferring process was started with creating smaller pieces such as design tokens and gradually moving on from primitives to the components. All of the steps were documented either via git or a text document.

As results a start of the Storybook Design System and instructions of using it were created. The project continues in development as not all of the components have been transferred yet.

The result was received well and presented in a demo for the whole company.

Keywords: React, Design System, Storybook

Tiivistelmä

Tekijä:	Tatu Kalermo
Otsikko:	Design Systemin rakentaminen
Sivumäärä:	21 sivua
Aika:	10.5.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Software Engineering
Ohjaajat:	Janne Salonen, Osaamisaluepäällikkö

Opinnäytetyön tavoitteena oli näyttää Design Systemin luonti käyttäen Storybook työkalua sekä keskustella Design Systemien tärkeydestä.

Lähtökohtana opinnäytetyölle oli suunnittelijan kehittämä Figma-kirjasto. Tavoitteeseen pääsemiseksi Figma-kirjaston siirrosta Storybookiin tehtiin tutkimusta eri komponenttikirjastoista sekä Storybookista.

Siirtoprosessi alkoi pienemmistä palasista, kuten design tokeneista, ja siirtyi hiljalleen alkeellisista osista komponentteihin. Kaikki vaiheet dokumentoitiin joko gitin tai tekstidokumentin avulla.

Tuloksina syntyivät Storybook Design Systemin alku sekä ohjeet sen käytöstä. Projektin kehitys jatkuu työn jälkeen, koska kaikkia komponentteja ei saatu siirrettyä työn aikana.

Projektin tulokset otettiin hyvin vastaan ja työn tulos esitettiin demona koko yritykselle.

Avainsanat: React, Design System, Storybook

Contents

List of Abbreviations

DevOps	Set of practices that combine software development and IT operations. It provides continuous delivery and automation of the processes. The goal is to shorten development life cycle and bring different parts of the development closer together.
JS	JavaScript. Programming language. One of the core technologies of the World Wide Web.
React	A JavaScript library for building user interfaces. Can be used as a base in the development of single-page applications.
CSF	Component Story Format. Open standard based on ES6 modules.
JSX	JavaScript XML. Syntax extension to JavaScript. Allows usage of HTML in React.
MDX	Markdown and JSX. Format that lets you write JSX in Markdown documents.
CSS	Cascade Style Sheets. Style sheet language used for describing the presentation of a document written in a markup language such as HTML.
HTML	HyperText Markup Language. Markup language for documents designed to be displayed in a web browser.
CSS-in-JS	Styling technique where JavaScript is used to style components.
WCAG	Web Content Accessibility Guidelines. Explains how to make web content accessible for people with disabilities. [15]
JSON	JavaScript Object Notation. Lightweight data-interchange format.

NPM	Node Package Manager. Package manager for JavaScript.

1 Introduction

Why do we need Design Systems and why are they a hot topic today?

This thesis demonstrates the implementation of a Design System from a developer's point of view as well as discusses the importance of Design Systems and Storybook as a tool. The thesis was instructed by Eficode as an internal project for a potential customer use in the future.

Eficode Oy is a Finnish software company known for their consulting and DevOps services. They employ over 300 people in seven different countries most of them being in Finland. Eficode offers many courses through their academy about different tools that are related to DevOps, design, and programming. [1] They have been part of developing Robot Framework which is a generic open source automation framework. [2]

Idea of the Design System had been on Eficode's radar for a while. They wanted to have an internal Design System that could be later used in customer cases as an example. A designer who had previous experience working with Design Systems was found suitable for the job. He wanted to take the task of building the internal Design System and he needed a developer to develop components for it. After discussions and research, the language for the storybook was decided to be React and Material UI was decided to be used as a base for the components.

The project had initially three goals determined for it and they were:

- Design library in Figma for designers.
- React Storybook for developers and designers.
- Documentation that will help anyone setup a Design System with storybook.

2 Design System

2.1 What is a Design System?

A Design System is multiple things. It’s not just a style guide or a library of branded components. Design System should guide both designers and developers bringing everyone on the same page. It should create a solid unified foundation for the company’s designs and components. Typically Design Systems have three elements. Style guide, component library and other guidelines (e.g. accessibility).

Style guide focuses on the looks such as colours, fonts, and illustrations.

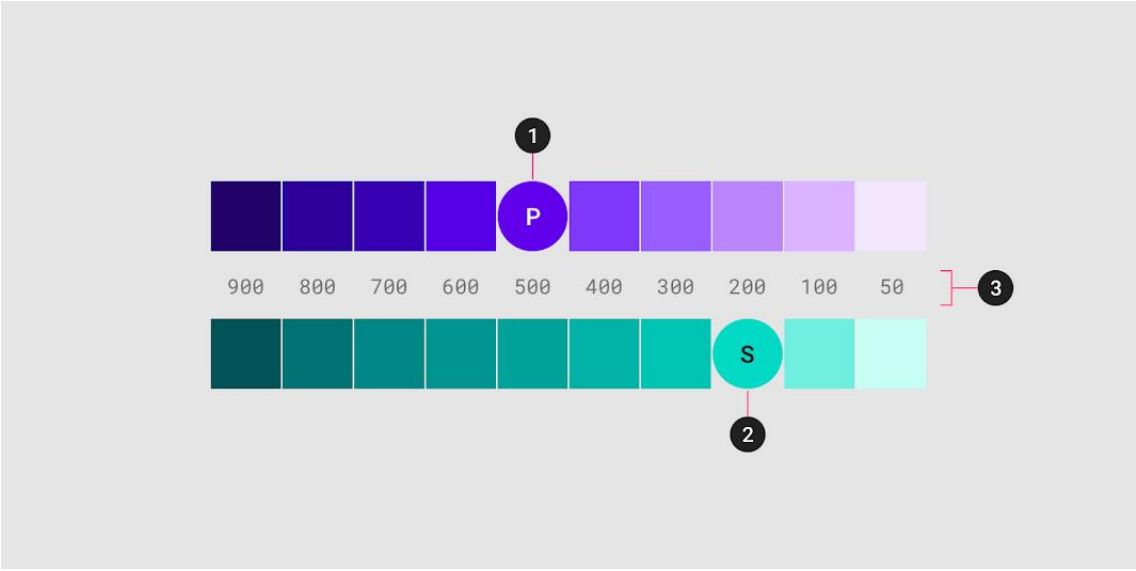

Figure 1. Material Design sample primary and secondary palette [5]

Component library shows the user how to design or develop components as well as how to use them. Depending on the size of the Design System it may have many other guidelines as well. These days many companies working on

the web are focusing on accessibility. Therefore, the Design Systems have the best practices and principles to make sure everything is accessible. [3;4]

2.2 Benefits

A Design System has many benefits for the company and individual teams using it. It helps to create a coherent brand for the company and its products. Having a clear style guide will boost consistency of the products which will make the brand more recognizable. To users components will look and feel like the brand. Consistency leads to higher quality all around. For example, on a website all the buttons will behave the same way and there will be no surprises to users.

The benefits are not only limited to customers and the brand. Teams working with the Design System avoid unnecessary misunderstandings by having a single source of truth which they can refer to when talking with co-workers. Avoiding mistakes coming from misunderstandings speeds up the development process which will lead to a faster cooperation between designers and developers. When developers don’t have to spend time wondering about the design of the components they can focus more on the usability and functionality. [3]

Faster cooperation is not the only thing that will speed up when using a Design System. Developers’ implementation speeds will rise immensely when they can use ready-made components straight from the Design System. Starting up a project is much easier with customising already fully accessible and responsive components. Sometimes simple components don’t even need to be customized to use them which saves a lot of time in development.

2.3 Design + Development

One of the main purposes of a Design System is to help build a bridge between designers and developers. Breaking barriers between different areas of software development has been a hot topic over the last few years. With the advancement of the Agile software development teams have grown more cross-functional bit by bit.

Long gone are the days of graphic guidelines that developers must have tried to follow without cooperation with designers. A Design System is the next level towards a more modern software development and brand designing. Companies have noticed the value of bringing their design teams closer to developers and the Design Systems have helped that greatly.

2.4 Component library

One possibility when building a Design System is to use a component library as a base of your Design System. Building your own components from the scratch takes time and resources to accomplish. A faster and cheaper way is to use ready-made components from a known library and customize them for your own liking.

Bigger companies usually build their own components from scratch but for smaller companies or personal projects using a component library saves a lot of hassle especially considering accessibility and responsivity. Most of the popular component libraries have fully accessible and responsive components. As a downside customizing some of the components might be a little tedious and it differs from library to library.

Theming is a big part of the component libraries. It allows overriding of the default styles and therefore enables customization of the components with your own styles. Themes allow you to customize all design aspects of your project and are powerful tools to learn when using component libraries. In Material UI

this is done by using a function from the library to create a theme constant and then wrapping the component or the whole application with the ThemeProvider component. This is demonstrated in the listing 1 by changing the default spacing and primary colour.

```
const theme = createMuiTheme({
  spacing: 4,
  palette: {
    primary: {
      main: '#ffd100',
    },
  },
});

export default function CustomStyles() {
  return (
    <ThemeProvider theme={theme}>
      <CustomCheckbox />
    </ThemeProvider>
  );
}
```

Listing 1. Example of creating a custom theme in Material UI

3 Storybook

3.1 Purpose

Storybook is an open source tool for developing UI components and pages in isolation as it is stated on the official Storybook website. What does it mean in practice? The problems Storybook is trying to solve are quite similar to the ones that Design Systems solve but Storybook adds extra features that Design Systems don’t have. [6]

Let’s imagine a large project that has several front-end developers working simultaneously on different parts of the project. They work on their own parts and everything works as planned. Now two of the developers both have to implement a menu button for their applications. They both implement the button according to the design they’ve been given. They both also have to put their buttons into the application to test if it works and how it looks like. When they both do this, they must maintain the consistent design of the brand. See the problem? They are both using their time to implement the same thing which might not even look the same hence not maintaining the consistency that is required. This is where the Storybook comes into the play. [7]

With Storybook you can more easily develop components together as a team and maintain that consistency with your designs. When put into Storybook the components can also be tested and viewed in isolation without a complicated application surrounding it. If our example’s developers were using Storybook, they could’ve implemented the button in there and tested different states in isolation without worrying about breaking other things. One of the example’s developers could’ve also implemented the button and the other could’ve reused that code in his application with ease. Another major help with Storybook is the documentation. The components have the documentation with them in stories. [8]

3.2 Stories

Stories capture the components' rendered states. They are the main building blocks of Storybooks. Components usually have multiple stories because they have multiple states that need to be rendered. [9]

The important thing is to have all of the files that contain the stories end with .stories.js or .stories.ts. Below in listing 2 is an example of creating a story named Primary for a component named Button and in figure 2 a view of the story rendered inside Storybook.

```
import React from 'react';

import { Button } from './Button';

export const Primary = () => <Button primary>Button</Button>;
```

Listing 2. Creating a story with React

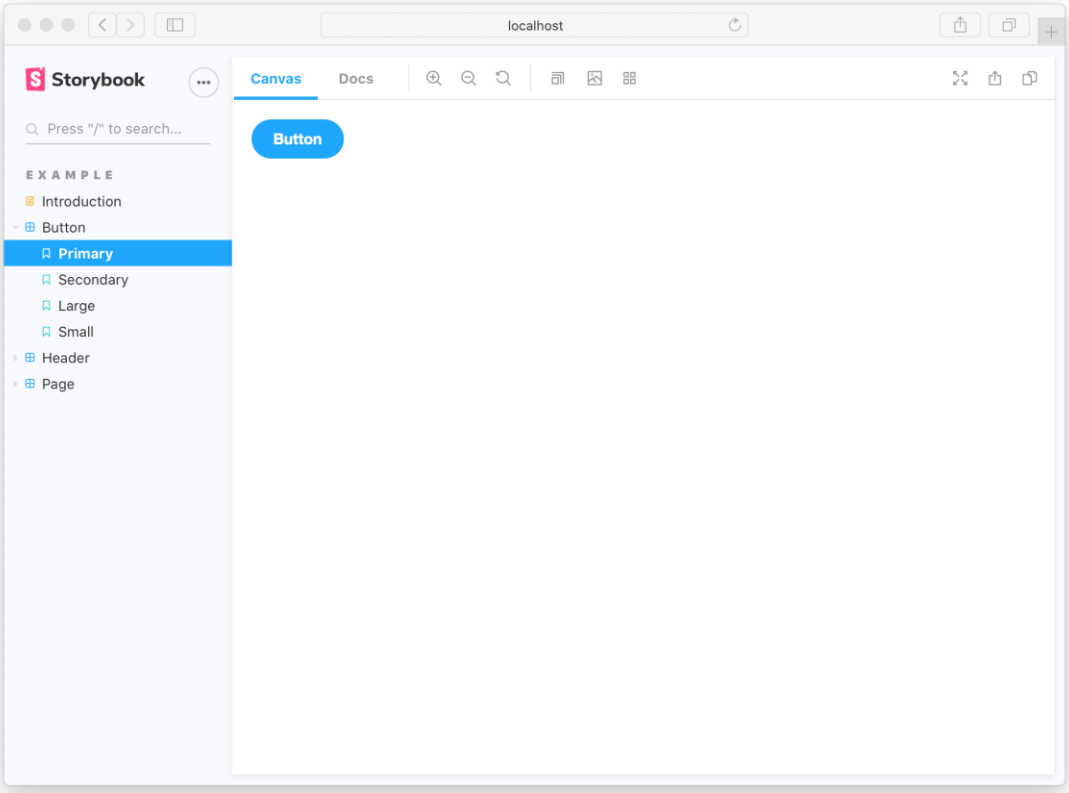

Figure 2. View of Primary button story in Storybook

Stories are written with CSF which is very similar to JavaScript because it's based on ES6. That makes it easy to use for developers and designers as most of them have used JS. All of the components are rendered like they would be on a website.

3.3 Addons

Addons are the major strength of Storybook. They extend its behaviour and make it a powerful tool. There are many third-party addons as well as addons developed by the Storybook team themself. The main ones are the essential addons that come with Storybook by default when installed. One important addon that is not installed by default is the Accessibility addon. These days all of the Storybook projects should include it for an accessibility testing. Otherwise the project manages fine with the essential addons and others could be installed if custom things are wanted in Storybook.

Docs is the addon that adds documentation to the stories. When writing component stories Storybook creates documentation for your story by default. This documentation can be customized to create component specific guidelines and usage information. This is done by using MDX to write markdown documents with JSX in them. This allows the user to write documentation and stories simultaneously like seen below in figure 3.

Figure 3. Documentation in Storybook [10]

Controls is an addon that gives the user a graphical UI to interact with a component’s arguments without any coding. Storybook auto-generates controls based on props in React. This could be used for showcasing a component’s functionality for non-developers or developers wanting to explore the component without messing with the code.

Actions addon is used to show data received by event handlers. It’s very useful to showcase or to test clickable items and component’s that make things happen. Viewport addon gives the user a toolbar item that lets them change the viewport of the iframe their story is rendered in. Useful in cases you want to see your component in mobile view for example. Backgrounds addon works similarly but it gives an ability to change the background of the iframe. Different custom backgrounds can be created inside a file called .storybook/preview.js.

Accessibility addon is used to test the component compliance with web accessibility standards. As accessibility is beginning to grow more and more important in today’s web developing this addon is a necessity. The addon tests components with the usual accessibility tests such as a picture having a proper alternative text. After testing it tells you if the component has passed the tests or where the violations are. It’s very useful for simple accessibility fixes for the components and highly recommended as it’s not part of the default installation.

All of these addons make Storybook a suitable tool for making Design Systems. Documentation is very important when building a Design System and these addons let you document both with text and UI elements. They’re not only explaining how the components work via a huge text document but showing the component in usage and how the component behaves in reality. This is what makes the Storybook work for designers, developers and marketing as everyone sees and feels the brand come alive. [11]

4 Implementation

4.1 Tech stack and setup

The project started with deciding a tech stack. React was chosen as a JavaScript framework because of its popularity and the developer's familiarity with the framework. Together with React Next.js was decided to be used in the project as well. Next.js enables server-side rendering and fast refresh that may come in handy when the project expands.

Due to the project's nature and resources a component library was decided to be used as a base for the components. Using readymade components straight from a library saves time in implementation. Different factors were considered when choosing the right component library. Research began from looking at the most popular React component libraries. Some of them were ruled out by talking with co-workers of their experiences. Semantic UI for example had bad experiences amongst the teams and was avoided because of them. The first question was whether the biggest libraries are too bloated and big in sizes. Counter point to that is that the smaller libraries simply don't have enough components to cover all the necessary ones.

After gathering information about different libraries, the biggest factor that made the decision clear was accessibility. The goal of the project was to make fully accessible components and the chosen library had to have accessible components as a base. The chosen library was Material UI which is one of the most popular libraries these days. It has WCAG compliant components and a vast selection of components that cover the project's necessary components [12]. As seen in figure 4 Material UI was one of the three libraries that passed the accessibility of the web UI frameworks test. Further testing done by the developer of the project proved this as well.

Library	Focus Ring	Contrast Ratio	WAI-ARIA
Reakit *1.0.0-beta.2*	★★★★	★★★★	★★★★
Lightning Design System *2.7.4*	★★★★	★★★★	★★★★
Material UI *4.3.2*	★★★★	★★★★	★★★★
Carbon *10*	★★★☆	★★★★	★★★★
Foundation *6.5.0*	★★★☆	★★★★	★★★★
Clarity *0.13*	★★★☆	★★★★	★★★★
Instructure UI *5.32.0*	★★★★	★★☆☆	★★★★
Elastic UI *5.0.1*	★★★★	★★★★	★★☆☆
Grommet *2.7.11*	★★★★	★★★★	★★☆☆
Atlaskit	★★★★	★★★★	★★☆☆
Polaris *3.0.0*	★★★★	★★★★	★★☆☆
Fabric *9.6.1*	★★★★	★★★★	★★☆☆
Blueprint *3.8.0*	★★★★	★★☆☆	★★★★
Bootstrap *4.1.3*	★★★☆	★★☆☆	★★★★
Evergreen *4.5.0*	★★★☆	★★☆☆	★★★★
ng-lightning *2.0.1*	★★★★	★★★☆	★★☆☆

Figure 4. The state of accessible web UI frameworks

Styled components was chosen as a CSS framework for the project because it was familiar to the team and it complemented Material UI's CSS-in-JS styling technique. [13]

Setting up React Storybook is quite seamless. Using Create React App for initializing a React app with the command line has never been easier. To have a React Storybook set up you only need two commands as seen in listing 3.

```
npx create-react-app design-system
npx sb init
```

Listing 3. Setting up React app with Storybook

4.2 Primitives

Creating a Design System starts from building primitives. Primitives include fonts, spacings, colours, grid and other building blocks used in UI. The designer creates primitives in Figma. The developer's job is to transfer the designs into Storybook.

This is done by first creating design tokens. What are they? Design tokens are variables made out of primitives that are stored the way they could be used anywhere. In this project tokens were saved on a JSON file. Tokens' main idea is that if you want to change all of your buttons primary colour to a slightly different colour, for example, you wouldn't need to go through every button and change the colour one by one. You could simply change the token's value and all of the buttons' colour would change. Other plus is that every value can be found in the same place. If someone is wondering what the primary colour 100 value actually is, he can just go look at the hex code from the tokens. Putting every value in a token lays a good foundation for later when the values are needed in the components. [14]

After the tokens are done it is time to create stories for the primitives. All the colours, fonts and other primitives should be displayed clearly and nicely in the Storybook. This is where the MDX comes to the rescue.

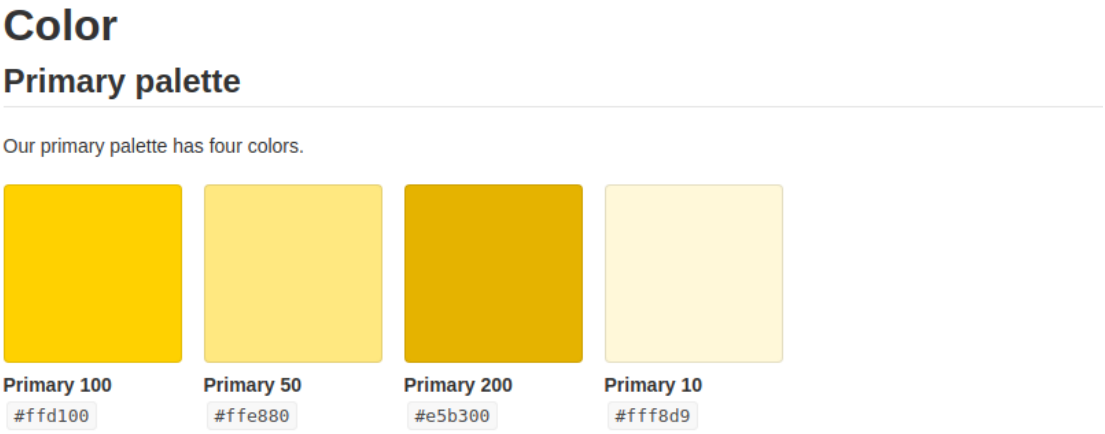

Figure 5. Eficode primary palette

As seen in figure 5 with MDX it is possible to have a square react component showcasing a colour mixed with text that tells information of it and groups it up with other proper colours. This displays the primitives like in Figma, but they are easier to skim through and allow you to get extra information that you wouldn't necessarily otherwise get.

4.3 Components

Components are where the fun begins because they have to be implemented and functional. When doing the components, it is very important to communicate with a designer to make sure the components look and behave like they are meant to in the design. Transferring a component from Figma to Storybook has three steps.

First step is inspecting the component in Figma. It includes getting familiar with the design and taking a look at the styles. Convenient thing in Figma is that you can see the CSS from there and have a glimpse of how it could look like. From experience it is better to get to know things thoroughly before diving into the implementation. No one likes to back-pedal when they realize they've implemented the component not according to the design.

Second step is finding a corresponding component from the library. This means going to the library's web page and looking at the documentation. In this project's case the library is Material UI. The key is to find a way to customize the default components to look like the custom design. With Material UI this happens via theming. In the project theme.js was created for this purpose. Having all of the overrides in the same file makes editing them easier.

Third step is the actual implementation of the component with React. With a ready-made component coming from a library this means importing a component and customizing it. Things to remember are not breaking the accessibility and keeping the components as simple as possible.

```
import { Button, ThemeProvider } from '@material-ui/core';
import theme from '../utils/theme';
import { createMuiTheme, withStyles } from '@material-ui/core/styles';
import { tokens } from '../tokens/Tokens.json';
export function PrimaryButton(props) {
 const PrimaryButton = withStyles({
   root: {
     background: `${tokens.color.primary100}`,
     border: 0,
     padding: '8px 24px',
     minWidth: '84px',
     width: `${props.width}`,
     '&:hover': {
       background: '#F1AE03',
     },
     '&:disabled': {
       background: '#F5F5F6',
     },
   },
 })(Button);
 return (
   <ThemeProvider theme={theme}>
     <PrimaryButton disableRipple disabled={props.disabled}>
       {props.children}
     </PrimaryButton>
   </ThemeProvider>
 );
}
```

Listing 4. Button.js PrimaryButton component implementation

Customizing the component in Material UI can be done with withStyles method that takes a base component from the Material UI and customizes it with the CSS provided. In listing 4 the PrimaryButton component is customized this way. Notice that the CSS inside the withStyles is not a typical CSS code like it would be written in a normal .css file. For example, normally you would write minimum

width property in CSS as “min-width: 84px” but in CSS-in-JS you write it as “minWidth: ‘84px’”. This little syntax difference is hard to get used to at first but after some time comes naturally like the normal one. Material UI components also have default props that can be used to remove default effects or in the PrimaryButton case the ripple effect. The “disableRipple” prop sets the property to true which removes the normal ripple effect coming with the Button.

4.4 Creating stories

Putting the components to the Storybook has been made very easy for developers and it shows. Only thing that is required is to have the story file ending with .stories.js or .stories.mdx if mdx is wanted. To make things simpler a good naming convention is to name your stories with the same name as your component. In the project Button component was created in Button.js. Corresponding story file for the Button was named Button.stories.js to make it clear which component’s story it is.

```
import React from 'react';

import {
 PrimaryButton,
} from '../components/Button';

export default {
 title: 'Components/Button',
 component: PrimaryButton,
 argTypes: {
   children: { control: 'text' },
   disabled: { control: 'boolean' },
 },
};

export const Primary = (args) => <PrimaryButton {...args}></PrimaryButton>;
Primary.args = {
 children: 'Text',
 disabled: false,
};
```

Listing 5. Button.stories.js Primary Button story

The Primary Button story is created by first importing a PrimaryButton component from Button.js as seen in listing 5 and then using the imported component to export a constant that the Storybook will use to render the story as seen in figure 6.

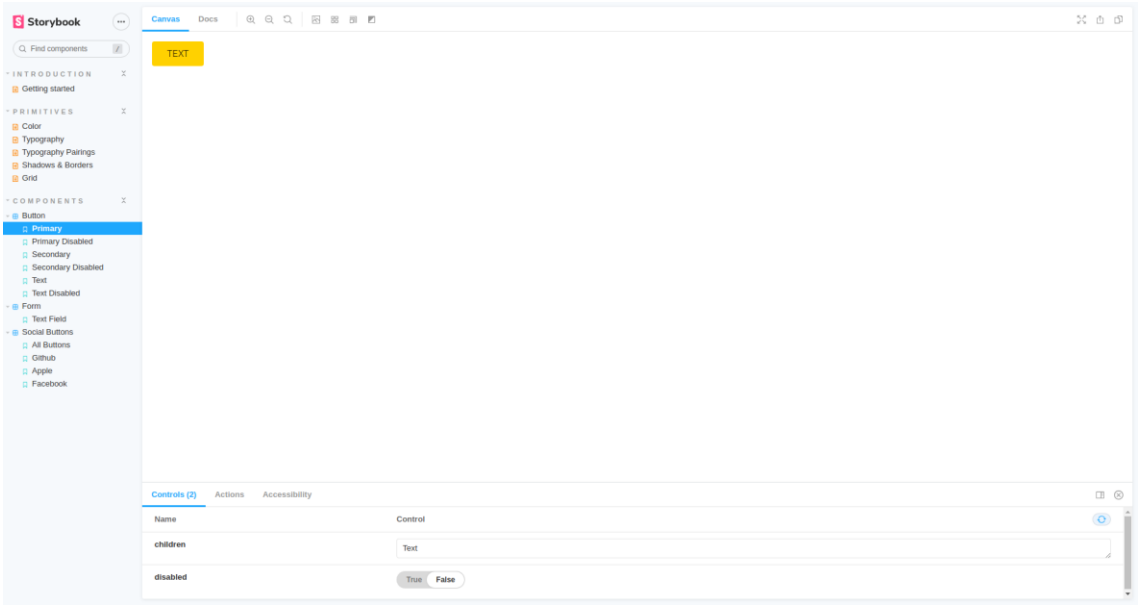

Figure 6. Primary Button story in Storybook

The controls are added to the story to test the props that the PrimaryButton takes in. It is done by first inside the default export describing the types of args that will be used, then giving the args as props to the PrimaryButton. In this example it allows the user of the Storybook to edit the text in the button and disable the button with the controls. Controls can be seen in use at the bottom of the figure 7 below.

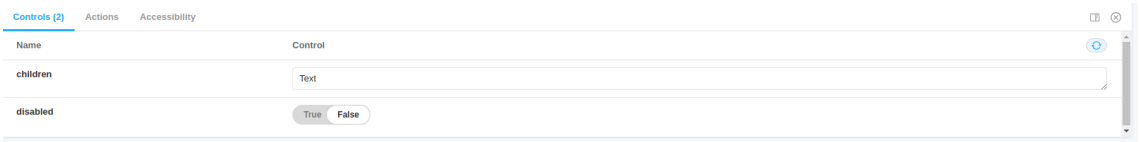

Figure 7. Controls addon in Primary Button story

4.5 Documentation

The project had more goals than just creating a functional Storybook and a Design System. Being an internal project one of the biggest goals of the project was to make documentation for learning purposes. Eficode as a firm highly values education of their employees and all the projects are seen as learning opportunities.

Everything was documented from the start. Code was documented on a Bitbucket repository using git to have a clean version control of the Design System. Bitbucket’s advantage is its flexibility and a JIRA integration. For the guides and all the other documentation, a text document was created in Google Docs. Step-by-step guides for both designers using Figma and for developers using Storybook were gathered in the document. The guides had detailed instructions about installing the necessary tools, setting up Storybook and building components.

The main idea of the documentation was that anyone who reads the instructions will know how to setup a Storybook and be able to use it wherever they need it. As a bonus it’s spreading knowledge of why Storybook is needed and advantages of it.

When finished the documentation will be cleaned up and put on Confluence for the whole company to see. Learning new things as a developer and as a company is important. As new tools and opportunities rise in this modern high paced fast-growing world of programming getting to know things and testing if they work or not is very valuable for everyone. That is also the reason why documentation although hated by some is very important to do.

5 Conclusion

5.1 Goals

The project had three major goals. The first goal of creating a Design System in Figma was completed fully.

The second goal of creating a Storybook was partially completed. The Storybook was successfully setup and is working. All the primitives were transferred from the design to the Storybook. All of the components are not transferred to the Storybook yet as their implementation takes time. The second goal is a work in progress and its development will continue.

The third goal of creating documentation was fully completed as well. The documentation has all the necessary guides and instructions for starting a Storybook. Although the documentation is completed it may be refined and cleaned up in the future when the project continues.

The project was a success. First demo of the project was held remotely, and the project was presented in front of 200 employees through a Zoom meeting. The development still continues on and the project will stay open for now.

5.2 What’s next?

More plans for the project have already been discussed and will be discussed in the near future. Being an internal project, the project has no deadline or time goals. Employees will work on the project when they have time for it between the customer cases and when they are free. As the second goal was not fully completed its development will continue until all the components are transferred to the Storybook.

The next step after that is to bring DevOps to the project as well. The goal is to have a working pipeline for the Storybook and possibly a public web page hosted online for the Design System. One other possibility is the creation of a

NPM package, enabling the use of the components in any internal or external project. As a big DevOps company Eficode wants to make the project automated with necessary tools to make it a complete package for potential customer use in the future.

Possibility of training videos as an extension of the documentation has been discussed as well. It could be an opportunity to fully capture what it is like to work on a Design System and better the communication between the designers and developers.

Better communication means better results. That is what the Design System comes down to. Improving the look, consistency and speed by creating a single source of truth that makes the communication more fluent.

References

1 Eficode [online]. Available from: https://www.eficode.com/who-we-are [Accessed 27 March 2021]

2 Wikipedia. Eficode [online]. Available from: https://fi.wikipedia.org/wiki/Eficode [Edited 22 March 2021] [Accessed 27 March 2021]

3 Uxmisfit.com [online]. Available from: https://uxmisfit.com/2019/03/26/what-is-a-design-system-everything-you-need-to-know/ [Edited 26 March 2019] [Accessed 3 April 2021]

4 UX Collective [online]. Available from: https://uxdesign.cc/everything-you-need-to-know-about-design-systems-54b109851969 [Edited 22 May 2018] [Accessed 3 April 2021]

5 Material Design [online]. Available from: https://material.io/design/color/the-color-system.html#color-usage-and-palettes [Accessed 17 April 2021]

6 Storybook [online] Available from: https://storybook.js.org/ [Accessed 24 April 2021]

7 The software house [online] Available from: https://tsh.io/blog/storybook-js/ [Edited 23 April 2020] [Accessed 24 April]

8 Webdevstudios.com [online] Available from: https://webdevstudios.com/2020/04/09/storybookjs/ [Edited 9 April 2020] [Accessed 24 April 2021]

9 Storybook.js.org [online] Available from: https://storybook.js.org/docs/react/get-started/whats-a-story [Accessed 28 April 2021]

10 Storybook.js.org [online] Available from: https://storybook.js.org/docs/react/writing-docs/mdx [Accessed 29 April 2021]

11 Storybook.js.org [online] Available from: https://storybook.js.org/docs/react/essentials/introduction [Accessed 29 April 2021]

12 Darek Kay [online] Available from: https://darekkay.com/blog/accessible-ui-frameworks/ [Edited 19 December 2019] [Accessed 1 May 2021]

13 Wikipedia [online] Available from: https://en.wikipedia.org/wiki/CSS-in-JS [Accessed 6 May 2021]

14 CSS-Tricks [online] Available from: https://css-tricks.com/what-are-design-tokens/ [Edited 3 April 2019] [Accessed 6 May 2021]

15 W3.org [online] Available from: https://www.w3.org/WAI/standards-guidelines/wcag/ [Accessed 8 May 2021]