Comparative Analysis of CSS Frameworks

Katja Helenius, Kiia Mäkirinta & Viivi Uhari

ABSTRACT

This paper presents a comparative analysis of three widely used CSS frameworks - Bootstrap, Tailwind CSS and Foundation. The comparison aims to discover the optimal use cases for each framework by focusing on their documentation, ease of use, responsiveness and customisation capabilities. To conduct the analysis, three identical websites were built with each framework and the development process was thoroughly documented. This paper showcases the findings of the developers, identifying the strengths and weaknesses of the frameworks and offering guidance on the kind of projects they would be most compatible with.

1 INTRODUCTION

Cascading Style Sheets (CSS) is a web development technology used to set the style and layout of websites. CSS offers the opportunity to create websites with unique styles, but sometimes writing CSS from the ground up for large projects can be time-consuming. CSS frameworks provide pre-built styles and components that simplify the development of websites. There are various CSS frameworks available and they all vary in the selection of components, customisation options and ease of use. The aim of our project was to conduct a comparative analysis of three different CSS frameworks to discover which kinds of projects they would be best suited for.

2 RELATED WORK

The popularity of CSS frameworks is easy to understand, considering the many benefits they provide. Foremost, CSS frameworks make designing and developing faster and more convenient (Goree, Doosti, Crandall & Su, 2021). In addition, CSS frameworks provide tools for responsive design, an essential aspect of modern website development (Goree et al., 2021). Such tools include pre-designed components and grids (Kaluvakuri & Vadiyala, 2016). CSS frameworks vary in their component selection as well as in their features such as the support for responsive design. In addition, the customisation capabilities of CSS frameworks are important (Salonen, 2023). This is because the adoption of CSS frameworks has made web applications more homogeneous (Goree et al., 2021; Salonen, 2023).

One of the most popular CSS frameworks is Bootstrap (Mohd, Thompson, Carmine & Reuter, 2022). It was first introduced in 2011, and it is particularly well-suited for beginners (Bankov, 2023). Bootstrap features an extensive library of pre-designed components and styles, a comprehensive documentation, and strong community support (Bankov, 2023). Other advantages include its mobile-first approach and intuitive grid system, both of which support responsive design (Bankov, 2023). Additionally, it enables high development speed (Mohd et al., 2022). Bootstrap has however limited customisation options often requiring numerous style overrides (Mohd et al., 2022).

Another popular CSS framework is Foundation. Also released in 2011, Foundation shares many similarities with Bootstrap. It employs a mobile-first approach, includes a flexible 12-column grid, and offers a variety of components (Agarwal, Goswami & Kaur, 2022). Notably, Foundation was the first framework to support responsive design and emphasizes functionality over visual styling (Mohd et al., 2022). Kopperi (2018) explains Foundation to provide more flexibility for customisation than Bootstrap.

Tailwind is a newer, more flexible CSS framework created in 2017 that relies on low-level utility classes (Bankov, 2023). These utility classes allow developers to make artistic decisions about layout and structure, rather than offering pre-made components, which also makes it more suitable for experts (Bankov, 2023). In addition, it is a lightweight framework, for example compared to Bootstrap (Bankov, 2023).

3 RESEARCH APPROACH

The goal of our study was to compare three popular CSS frameworks, Bootstrap (v5.3), Foundation (v6.9), and Tailwind (v3.4), to investigate differences between them. These three frameworks were chosen because of their widespread use and different features. The aim was to understand the pros and cons of each framework, analyse the development process from a developer's point of view and identify which types of web projects each framework is best suited for.

For the implementation part of our project we built three separate versions of the same website using HTML, with each site styled by one of the selected CSS frameworks. The rough UI design was created in Figma, where we designed a website surrounding the theme of preservation of glaciers.

Each web page consisted of commonly used UI elements such as cards, images, buttons, text inputs, and navigation, which we grouped into larger components. The components we ended up developing were a navigation bar, footer, grid, modal, table and form. Each member of our group implemented two components. We designed layouts to fit a wide range of screen sizes, with a maximum width of 1600 px and a minimum width of 360 px.

We intentionally used only simple HTML and avoided using any additional web development frameworks like React or Next.js. Any interactive features needed were added using plain JavaScript to focus on the CSS frameworks themselves.

During the project, we analysed each CSS framework mainly based on three factors: how clear and helpful the documentation was, how customisable the features were and how well the framework supported responsive design. These factors helped us to compare the frameworks to each other and what they were best suited for.

4 RESULTS

The created websites are displayed in Figures 1-4. They mostly look the same but differ in specific aspects such as breakpoints, icons and dropdown menus.

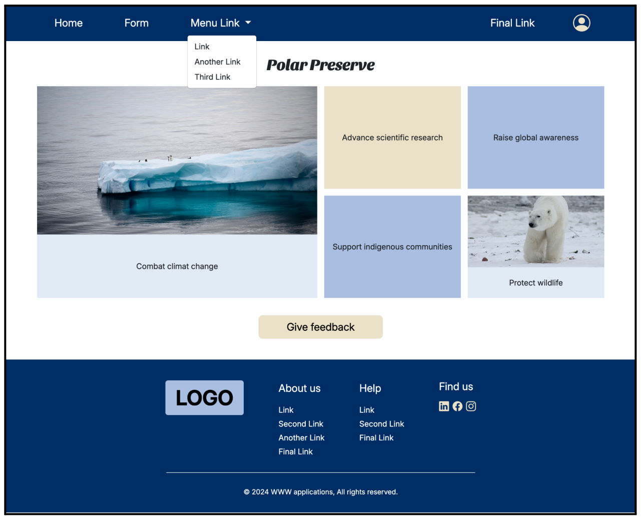
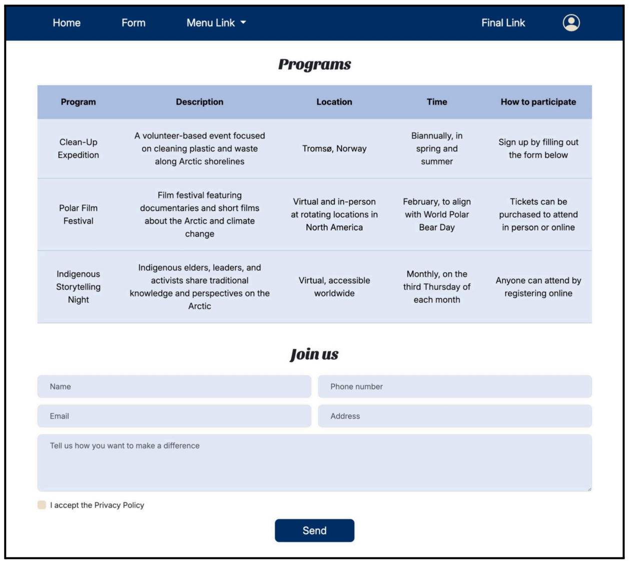


Figure 1. Bootstrap navigation bar, grid and footer in large screen width



Programs

Program	Description	Location	Time	How to participate
Clean-Up Expedition	A volunteer-based event focused on cleaning plastic and waste along Arctic shorelines	Tromsø, Norway	Biannually, in spring and summer	Sign up by filling out the form below
Polar Film Festival	Film festival featuring documentaries and short films about the Arctic and climate change	Virtual and in-person at rotating locations in North America	February, to align with World Polar Bear Day	Tickets can be purchased to attend in person or online
Indigenous Storytelling Night	Indigenous elders, leaders, and activists share traditional knowledge and perspectives on the Arctic	Virtual, accessible worldwide	Monthly, on the third Thursday of each month	Anyone can attend by registering online

Join us

Name
Phone number
Email
Address
Tell us how you want to make a difference
I accept the Privacy Policy
Send

Figure 2. Bootstrap navigation bar, table and form in large screen width

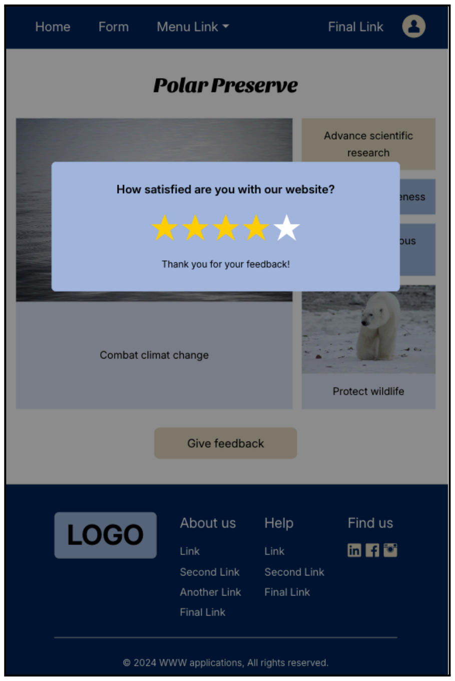


Figure 3. Foundation navigation bar, grid, modal and footer in medium screen width

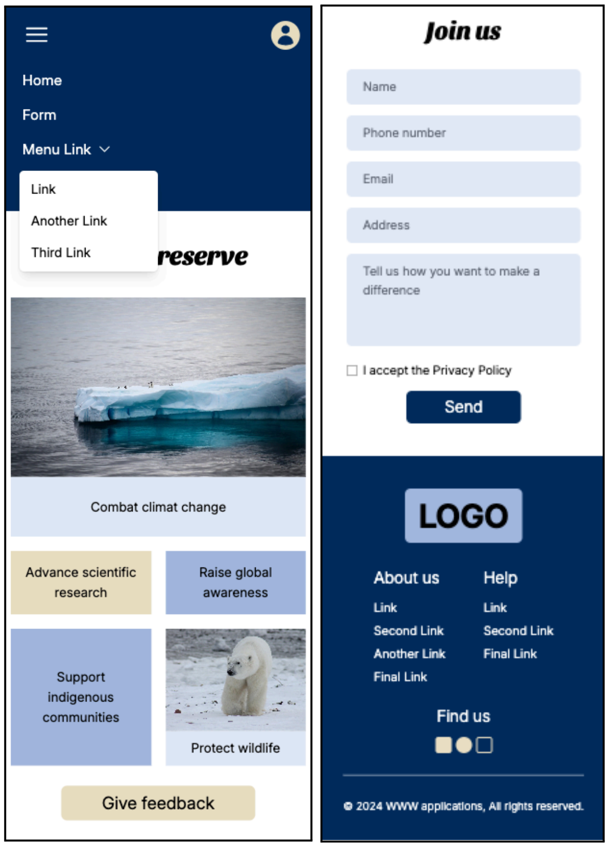


Figure 4. Tailwind navigation bar and grid; and form and footer in small screen width

5 ANALYSIS

5.1 Documentation and Ease of Use

A comprehensive and well structured documentation allows developers to quickly get started with a new framework and discover the features it offers (Salonen, 2023). In our analysis, the documentation for the frameworks is vital since all of our developers had to learn to utilise multiple unfamiliar frameworks in a short amount of time. In addition to documentation, the overall ease of use of the frameworks is analysed. For the analysis, the developers recorded their experiences in understanding the syntax of the frameworks, searching for solutions and solving errors.

The first framework, Bootstrap, provides a wide array of ready-made components that can be copy pasted into the code from the framework's documentation. This makes it simple to pick up the framework and quickly utilise the features it offers. Moreover, the documentation was found to be well structured and easy to navigate. While a few drawbacks, such as the documentation missing some utility classes, were discovered, the ready-made components, clear documentation and example code made the framework easy to learn and utilise.

Foundation offers both official components and components created by a community of developers. For this analysis, only official components provided by Foundation were utilised to maintain cohesion in class names and styling. These components were styled both with utility classes and CSS, which could be learned from the examples in the frameworks documentation. Overall, the documentation for Foundation was the poorest out of the three frameworks, with odd categorisation, insufficient information and difficulties in finding information. Moreover, the ease of use of the framework was affected by confusion about the balance between utility classes and CSS, since both were required to properly style components with Foundation.

Tailwind CSS is used through a wide array of utility classes. For this analysis, only free to use components made by Tailwind UI were utilised.

Overall, the Tailwind CSS was found to require a longer learning time than Bootstrap or Foundation. For developers with no experience with the framework, the documentation felt insufficient, with the most friction caused by the lack of code examples. In contrast, a developer with more knowledge of the framework found the documentation to be easy to search and find information from. The ease of use of Tailwind was slightly decreased due to the repetition that emerged from the use of utility classes with pure HTML.

5.2 Responsiveness
As responsiveness is one of the most critical features in modern web services, we wanted to focus on it as one of our main factors to analyse.

Bootstrap's responsiveness is based on its 12-column grid system. The default breakpoints provided by Bootstrap were sufficient as there were 6 breakpoints, from extra small to extra extra large. Breakpoints in Bootstrap are also customisable, but we did not find a need for this since the provided breakpoints covered all screen sizes so well. The responsiveness came handy for example when developing the grid on the home page, including the cards with photos. As proposed in our initial Figma design, the layout of cards varied between each screen size. This took relatively little effort to implement with Bootstrap, since it was easy to define the amount of columns for each card in each screen size. Also, the media-queries worked seamlessly by allowing specific styles added on specific screen sizes. The documentation for Bootstrap’s responsive features was well executed. For example, components like the navbar included clear instructions on how to make them responsive in the first lines of their documentation.

Foundation provides only three default breakpoints, which were not enough on many occasions. Simply, the gaps between small, medium and large screen sizes were too large. Foundation does however offer a possibility to customise its breakpoints. Another problem with Foundation was that not all utility classes are responsive. For example, you could change the direction of a flexbox based on the screen width, but you could not change the alignment of components within that flexbox.

Tailwind, which relies on the utility-first approach, offered responsiveness by utility class prefixes, such as sm:, md: and lg:. It was quite a straightforward way to add styles for different screen sizes without needing media queries or separate CSS files. Tailwind offered five breakpoints by default, from small to extra extra large. There could have been an additional breakpoint for extra small mobile screens, but again, the breakpoints in Tailwind can be customised.

One thing all three frameworks had in common regarding responsiveness was their mobile-first approach. They prioritise styling for small screens first and progressively move on to style larger screens as needed.

5.3 Customisation Options
Salonen (2023) explains that a component can be customised by adjusting its functionality or styling, with our focus being on the latter. The frameworks required customisation regarding primary and secondary colours, font families, and spacing which we had defined in our Figma design.

Bootstrap offered a few options for customisation, such as using a package manager or distribution files. Overall, the information about customisation was scattered in the documentation and understanding it as a whole required studying multiple pages. Finally, we decided to customise our application by using npm because it had the most support in the documentation. With this setup, customisation was done by overriding Bootstrap’s Sass variables and adding our own and compiling our custom Sass code into CSS. As Mohd et al. (2022) explain, we needed to override a lot of variables, for example to customise our buttons, and compile the custom file multiple times, which was frustrating. However, Bootstrap was the only framework that had support for defining the background colour of an unchecked checkbox, which would require a lot of styles overrides in pure CSS.

Foundation had a similar process for customisation of overriding Sass variables and compiling the custom settings into CSS. Understanding the customisation process based on the documentation was however slightly more difficult than with Bootstrap. The documentation primarily focused on using npm for customisation, which made it an obvious choice for our project as well. For our use cases, Foundation offered fewer Sass variables than Bootstrap, requiring us to write more CSS. In addition, the custom file in Foundation included all possible Sass variables, whereas with Bootstrap, the custom file only contained the overridden and added variables. As a result, it was more difficult to track the defined styles when using Foundation.

Customising Tailwind was a completely different experience. Since Tailwind needs to be installed to use it within a project, we had already initialised it using npm. When Tailwind is run through npm, the package manager automatically compiles the required CSS file. Instead of Sass variables, customising Tailwind is done by modifying its configuration file, where users can either override its default styles or extend them. We used both approaches but understanding Tailwind's customisation options and configuring it in a way that best fits a project would require extensive studying.

We also looked into the offerings of icons since using framework-provided icons ensures their easy integration and a consistent style between them and the rest of the components (Salonen, 2023). All three frameworks provide icon packages, but Tailwind’s collection did not include social media icons that we required for the footer. Bootstrap and Tailwind offer different methods for including icons in a project of which we used SVG HTML. Tailwind supports styling its icons with utility classes but with Bootstrap this was not possible. Foundation only instructs using its icons as icon fonts styled with CSS, but this lacked examples.

6 CONCLUSIONS

Based on the findings gathered while developing websites with Bootstrap, Tailwind CSS and Foundation, optimal use cases, clear strengths and weaknesses can be identified for each of the frameworks.

Bootstrap was determined to be a middle ground between the three chosen frameworks. The documentation was comprehensive and multiple code examples were available to help the developers get started. The customisation of the framework had both pros and cons and it was discovered to require a good build pipeline for a smooth development process. Overall, the findings indicate that Bootstrap would be best suited for quickly building projects that don’t require a lot of customisation.

Foundation's customisation process was similar to Bootstrap. However, complete styling required both pure CSS as well as the framework's utility classes, which caused confusion during the development process. Overall, Foundation had the most bugs as well as insufficient documentation. Based on the analysis, the framework is a good option if the goal is to utilise Foundation specific components without excessive customisation.

The styling for Tailwind CSS relies completely on utility classes since no free-to-use ready-made components exist for the framework. The extensive documentation and the wide array of customisation options had both its advantages and drawbacks during the development process. Based on the features of the framework, it would be the most useful for creating custom reusable components with another framework or a library, such as React.

To conclude, The comparison of Bootstrap, Tailwind CSS, and Foundation revealed notable advantages and limitations for each of the frameworks. The analysis reveals the importance of understanding the features of different CSS frameworks when selecting one for a project, since it will have an effect on both the development process and the look and feel of the final product.

REFERENCES

Agarwal, S., Goswami, A., & Kaur, A. (2022). Zurb Foundation Framework: A Review Paper. *International Journal of Multidisciplinary Research and Growth Evaluation, 3*(3), 26-31.

Bankov, B. (2023). Software solutions for responsive and accessible web systems. In *Дигитализация, големи данни, изкуствен интелект* (pp. 39-43). University of Economics. https://www.researchgate.net/publication/371947883_Software_solutions_for_responsive_and_accessible_web_systems

Goree, S., Doosti, B., Crandall, B., & Su, N., M. (2021). Investigating the Homogenization of Web Design: A Mixed-Methods Approach. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1-14). Association for Computing Machinery. https://doi.org/10.1145/3411764.3445156

Kaluvakuri, S., & Vadiyala, V. R. (2016). Harnessing the Potential of CSS: An Exhaustive Reference for Web Styling. *Engineering International, 4*(2), 95-110. https://doi.org/10.18034/ei.v4i2.682

Kopperi, A. (2018). *CSS-ohjelmistokehysten vertailu*. [Thesis, Häme University of Applied Sciences]. https://urn.fi/URN:NBN:fi:amk-2018100415689

Mohd, T., K., Thompson, J., Carmine, A., & Reuter, G. (2022). Comparative Analysis on Various CSS and JavaScript Frameworks. *Journal of Software, 17*(6), 282-291. https://doi.org/10.17706/jsw.17.6.282-291

Salonen, S. (2023). *Evaluation of UI Component Libraries in React Development*. [Master of Science Thesis, Tampere University]. Trepo. https://urn.fi/URN:NBN:fi:tuni-202304274742