Animation and Framework Research

How do I use animations in CSS/JavaScript?/What frameworks will help me with animations and responsiveness?

Personal Project – Personal Showcase Portfolio

Contents

Animations in CSS/JS

Animations in CSS and JS are quite good. Most animations can be made with these languages, but CSS is mostly for easy animations like scaling and fading to black when hovering over something for example. JavaScript is mostly used for more complex animations, like bouncing and fading in/out (Lewis, P. & Thorogood, S., 2019).

However, most of these animations require more coding than when you are using a library. JavaScript libraries come into play here. With most libraries you can make a beautiful animation within a couple of lines of code. ScrollRevealJS, AnimeJS and GSAP are examples of how easy animations are made because of libraries (Priya, 2023).

But since I'm going to use a framework, I can also look into animations within frameworks. At first, I decided to weigh the pros and cons of the frameworks to each other, to see which framework is the best for me to use. I also decided to research the pros and cons of Vanilla JS and Frameworks in general.

Vanilla (plain) JS (No frameworks)

Pros

- Vanilla JS will never get outdated. It will keep updating forever, compared to frameworks, which could lose their support.
- You are not stuck to the specific rules of a framework. Besides, adding a framework later on is still possible.
- Bugs are easier to find, because it will be in your written code and it can't be an error from your framework.
- Very fast render time.
- Extensive documentation online.

Cons

- Vanilla JS does not have as much functions as a framework on itself. For this you have to import libraries, which slow down your render time.
- Vanilla JS can be written in multiple ways, which makes outsourcing code harder.
- No reusable components.

Frameworks in general

Pros

- Most frameworks have reusable components, which make writing the same code over and over again way easier.
- Most frameworks have built in libraries, which increase the amount of functions you can use in programming.
- Frameworks are almost always open-source.
- Frameworks get use by huge companies, like Google, Netflix and Facebook.
- Good for SEO (Search Engine Optimalisation).
- Easier with outsourcing code, since you have to stay to the conventions of the framework.
- Frameworks often have easier ways to make animations.

Cons

- Frameworks have a habit of getting updated a lot, leaving documentation deprecated.
- Slower with rendering content than Vanilla JS.
- If you use a framework, it's hard to change to another framework, if you desire to do so.
- Frameworks get outdated relatively fast. Think about Jquery for example, which was immensly popular for a while, but after JavaScript introduced more fuctions, it became redundant for most cases.

React.JS

Pros

- Learner-friendly for people with experience in Vanilla JS.
- React has a library to make beautiful user interfaces.
- Easy to test, because of the built-in test server.
- Virtual DOM decreases loading time, making everything faster compared to other frameworks.
- React has built-in animation libraries.

Cons

- React updates very, very frequently, which add new functions and removes others.
- These new updates often have bad documentation.
- React is frontend only, combine this with nodeJS, for example, to make a backend as well.

Angular

Pros

- Clean and efficient code is easy to make.
- Angular has multiple interface libraries, which make creating UI's easier.
- Angular has routing, which makes going from one page to another easy.

Cons

- Documentation is often unclear.
- Angular is seen as an older framework, with a massive learning curve.
- Angular can get very slow with interactive components.
- Integration with third parties can be difficult.
- Unclear code for animations.

Vue

Pros

- Vue is a small program. A compromised program is only 18kB to 20kB. This is good for SEO.
- Easy learning curve.
- Easy integration with other websites and apps.
- Detailed documentation that tells you what to do.
- Vue has 2 way communication, taht makes HTML block handeling go faster. This makes big blocks of code render faster. It can also do 1 way communication, which makes other components render more quick.
- Virtual DOM increases render speed, compared to other frameworks.
- Built in animation commands.

Cons

- A big part of the community is Chinese, because Vue gets used by Alibaba and Xiaomi for example.
- The 2 way communication does not work every time.
- Vue is relatively young, which means that some documentation isn't there yet.
- Vue is really flexible, what means that outsourcing code can again be more difficult.
- Vue has relatively little libraries, compared to React or Angular, which reduces functions.

Svelte

Pros

- No extra elements are needed when using Svelte. This means no overhead.
- Svelte around 30% faster than other frameworks.
- Svelte transforms itself in Vanilla JS when it gets uploaded to the browser.
- A lot of possibilities for animations and other effects.
- Server-side renderering make device specifications not affect load times.
- A lot of results with little code.
- Does not use a DOM, since Svelte actually is a compiler in itself.
- Built in animation functions.

Cons

- Svelte still has a couple of issues with routing, that do not make pages go from one to another smoothly.
- Svelte has no significant support, like React with Meta and Vue with Google.
- Svelte is pretty new and does not have a lot of documentation online.

Conclusion

Vanilla JS has a lot of pros compared to frameworks, but vice versa it's the same. Vanilla JS is easier with noticing mistakes and it never gets deprecated (Darveau, F. P.). The reason why I'm using a framework is mostly because of reusable components, which increase efficiancy. With frameworks you're stuck to the rules of the framework, so I will not create code that has a lot of different approaches in it. This is good for if I decide to make changes in the future.

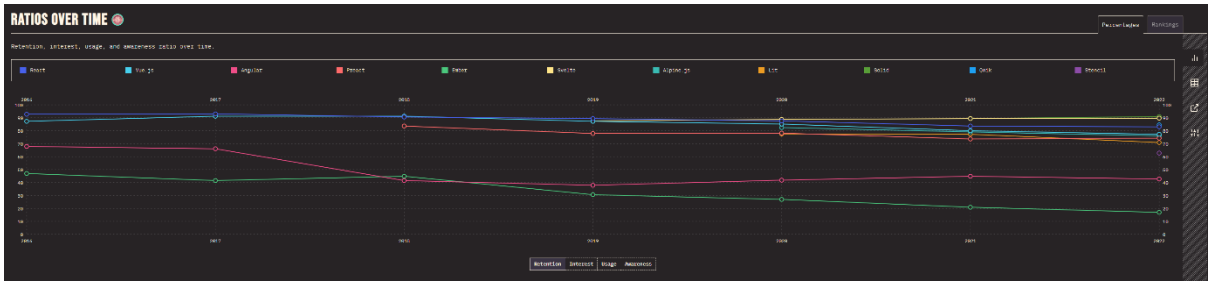

Figure 1: User statisfaction research for JavaScript frameworks (Greif & Benitte, 2023)

So I chose to use a framework and that framework is React. React is still one of the most used and most respected framework to date. Svelte en Vue also piqued my interest as well. Angular already showed in most tests and user experiences, that it was not a framework I would like to use. Long loading times mean that Angular is very slow compared to other frameworks (Levlin, n.d.). And most importantly, no clear documentation (InterviewBit, 2022). Lastly, it has crumbled in popularity over the last couple of years, as shown in the graph below (Greif & Benitte, 2020).

Svelte was very interesting to me, however, since Svelte is so new and does not have a large community and documentation, I opted to not choose for Svelte. This is a framework that I will be watching in the future, because it sound promising. So it was between React and Vue, and since I have experience in React, I chose react for this project. React and Vue are very similar in frameworks and they fill up eachothers cons. If these two frameworks get mixed into one, the perfect framework could be made. But for now, React is my choice for the framework I'm using for my portfolio.

Animations are still often done with libaries, since these libraries are specifically made to make animations easier, while frameworks are made to make general coding easier. So I will use libraries like GSAP and ScollRevealJS for example, to make animations way easier to program.

References

- [DavidDev]. (2015, 13 januari). *React-router URLs don't work when refreshing or writing manually* ([Peter Mortensen], Red.). Stack Overflow. Used on 28 February 2023, from https://stackoverflow.com/questions/27928372/react-router-urls-dont-work-when-refreshing-or-writing-manually
- Darveau, F. P. (2022, 24 maart). *You SHOULD Learn Vanilla JavaScript Before JS Frameworks*. Snipcart. Used on 28 February 2023, from https://snipcart.com/blog/learn-vanilla-javascript-before-using-js-frameworks
- Greif, S. & Benitte, R. (2023). *State of JS 2022: Front-end Frameworks*. Stateofjs. Used on 28 February 2023, from https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/
- Infotech, T. (2022, 28 september). *Vanilla JS Vs React JS: What To Choose For Your Development?* Tagline Infotech. Used on 28 February 2023, from
- https://taglineinfotech.com/react-js-vs-vanilla-js/
- InterviewBit. (2022, 2 juni). *Angular Vs React: Difference Between Angular and React*. InterviewBit. Used on 28 February 2023, from
- https://www.interviewbit.com/blog/angular-vs-react/
- Levlin, M. (z.d.). *DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte*. Åbo Akademi University. Used on 28 February 2023, from
- https://www.doria.fi/bitstream/handle/10024/177433/levlin_mattias.pdf?sequence=2&isAllowed=y
- Lewis, P., & Thorogood, S. (2019, August 3). *CSS versus JavaScript animations*. web.dev. Retrieved March 7, 2023, from https://web.dev/css-vs-javascript/
- Priya. (2023, January 1). *10+ Best JavaScript Animation Libraries to Use in 2023*. CodeinWP. https://www.codeinwp.com/blog/best-javascript-animation-libraries/

- Woke, G. (2022, 14 september). *The top 3 JavaScript front-end frameworks in 2022*. Pieces. Geraadpleegd op 28 september 2022, van https://code.pieces.app/blog/the-top-3-javascript-front-end-frameworks-in-2022