Approaches towards the Comparison and Utilization of JavaScript Animation Libraries*

Nikita Vozisov[1][0000−0002−2558−2946], Ilya Gosudarev[2][0000−0003−4236−5991], and Irina Gotskaya[3][0000−0003−3074−8936]

[1] ITMO University, Saint Petersburg, Russia nikitavozisov2108@gmail.com
[2] ITMO University, Saint Petersburg, Russia goss@itmo.ru
[3] ITMO University, Saint Petersburg, Russia iringot@yandex.ru

Abstract. A lot of JavaScript libraries for animations exist at the present time. They all have the same functionality and it is always a problem, when it is necessary to decide, which tool should be used for a project. This article aims to provide information about most widely spread ones and gives libraries comparison with pure JavaScript and pure CSS solutions according to the following criteria: minimum, maximum and average number of frames per second. In addition, it pays attention to other important factors, which should be kept in mind while choosing library for a project: library demand and its support. Also memory consumption is taken into account. First section describes current situation with animation on the web: why animation is used and which ways of creating animation exist. Section two explains the choice of libraries reviewed in next sections. In section three an explanation of the choice of criteria of libraries comparison is presented. Section four describes performance test of animation 1000 and 3000 DOM nodes and provides results obtained and contains their description. As a conclusion, the results of the experiment are reviewed and recommendations are given about library choice according to the comparison results.

Keywords: animation · JavaScript · FPS · Anime.js · Popmotion · GSAP · developer tools.

1 Introduction

During the last two decades, modern web has always been moving towards better user experience, design and interactivity. The stage if evolution, at which site developers created web pages in a static way, has expired. Currently, there are many ways to make a website better in these terms. One of them is animation, which is used to improve interactivity.

User, most likely, will prefer a site with a better user experience and design. Thus, to be competitive in the modern World Wide Web, companies, teams and

isolated developers currently must develop visually attractive pages [1]. Therefore, website developers need to choose a way how to add an animation to their application. A large number of techniques and technologies exist which help to create animations, including:

- Pure CSS
- Pure JavaScript, changing CSS properties of DOM element
- SVG animation
- Canvas HTML element
- Animation libraries
- And some others

The simplest approach is to use animation libraries, because they solve different performance problems, which developers otherwise would have to solve on their own. In addition, they support the vast majority of browsers, so no fallback or workarounds need to be provided. Therefore, next step is library choice.

2 Animation library selection

JavaScript being one of the most widely used programming languages [5, 6], it has grown a significant ecosystem of libraries. Hence, it is a hard problem to solve, which library to choose. According to articles [2, 3] and github search for animation libraries [4], it was decided to consider the following libraries:

- Anime.js [7]. Lightweight JavaScript animation library with a simple, yet powerful API. It is able to change CSS properties, animate SVG, DOM attributes and JavaScript objects [8]. This library allows to bind multiple animation properties, create timelines and more. To create an animation an anime() function must be called and provided with an options object with properties, which have to be animated.
- Popmotion [9]. As a part of Popmotion, in this article Popmotion pure is considered [10] – a functional, flexible JavaScript library for animation. It is more similar to Anime.js, but it requires styler() – a special library function to be called with DOM element. Thus, all the DOM nodes should be received before they can be animated. After all the styled nodes obtained, tween() function can be applied to create animation.
- GSAP [11]. This library is a little bit harder to install – additionally transpiler packages must be installed: @babel/preset-env and @babel/core. Despite it, as stated on GSAP website it solves countless browser inconsistencies at high speed. To create an animation, timeline is supposed to be created and animation is supposed to be added to this timeline. It supports CSS selectors.

The authors gathered statistics about all the libraries (see table 1).

Table 1. Libraries parameter list

Parameter	Anime.js [9]	Popmotion [11]	GSAP [13]
Github stars (thousand)	32.8	17.3	9.8
Contributors	44	53	2
Open/closed issues	96/392	154/367	2/278
Size, Kb	14	11.7	36

3 Selection of comparison criteria

One of the most important parameters [15] for any graphics, and animation in particular, is FPS (frames per second) [14]. Basically, FPS defines how smooth animation will be, so the higher FPS, the smoother the animation. To conduct fair test, the following FPS metrics are considered: minimum FPS, maximum FPS and average FPS. In addition, a web browser's activity has to be considered: how much time it spends on style recalculation, reflow, painting and layer compositing. The influence of these stages on animation performance has to be explored. The more time browser spends on earlier stages, the more time it will spend on later stages. Hence, the most performant library does less style recalculations, for example.

Memory consumption is also an essential parameter to focus attention on. The high usage of memory may lead to animation delays, especially on low-end devices.

In spite of animation performance being an all-in-all parameter it is worth taking into consideration the development process. It is important to pay attention to how many contributors develop a project. If there are only a few contributors, the project can be closed at any time. Contrariwise, a project with many developers typically produces better product. Animation library demand is also a significant factor for consideration. If library is demanded, then there is huge probability that the most important problems are solved and that there exist a lot of tutorials and best practices. Furthermore, it is recommended to study how many issues have been registered and how many of them have been solved. This factor shows how active a project is. If the issues are not being solved, a project can become suspended. The final list of parameters is:

- Minimum, maximum and average FPS
- Memory usage
- Time spent on style recalculations, reflow, painting and layer compositing
- Number of contributors
- Animation library popularity
- Open and closed issues

4 Animation libraries comparison

To compare the selected libraries Google Chrome developer tools and Mozilla Firefox Developer Edition developer tools were used. Libraries were additionally compared with solutions, which used only CSS and only JavaScript in the

following way – 1000 and 3000 DOM nodes have been created and saved log produced by corresponding developer tools. All DOM nodes are similar – it is 100 pixels radius ball, which moves back and forth for 500 pixels distance.

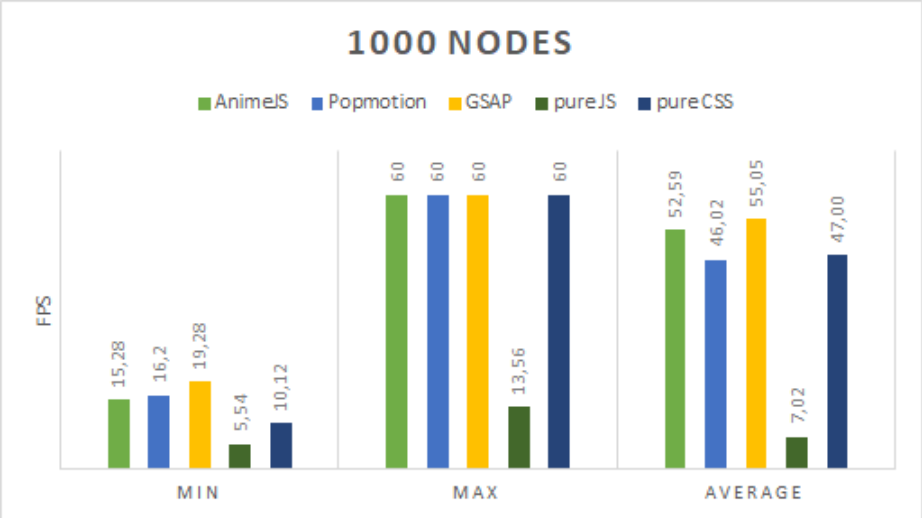

Fig. 1. One thousand nodes test results

As the performance test shows, the most performant library is GSAP. It has shown the highest average FPS during the test. In addition, it has the highest minimum FPS. As it can be seen, all solutions, except pure JavaScript, have maximum 60 FPS at 1000 nodes and satisfying FPS on average. Actually, 1000 nodes performance test (see Fig. 1) shows a big difference in average FPS. So it was decided to conduct 3000 nodes performance test (see Fig. 2) and here are the results.

All the libraries have maximum 60 FPS at 1000 DOM nodes and, as expected, less FPS with 3000 DOM nodes. It is important to notice, that GSAP has roughly 2.45 times FPS decrease, while in the case of Popmotion this number is 4.45 times at 3000 DOM nodes test. Thus, Popmotion is the slowest library among the considered ones.

The best results are demonstrated by the pure JS and pure CSS solutions – they consume the lowest amount of memory. The fastest library, GSAP, takes more memory than other libraries. The library, which requires the least memory, is Anime.js

The time browser has spent on processing of these animations is considered next. Figure 3 clarifies what are browser's stages of rendering elements on a page.

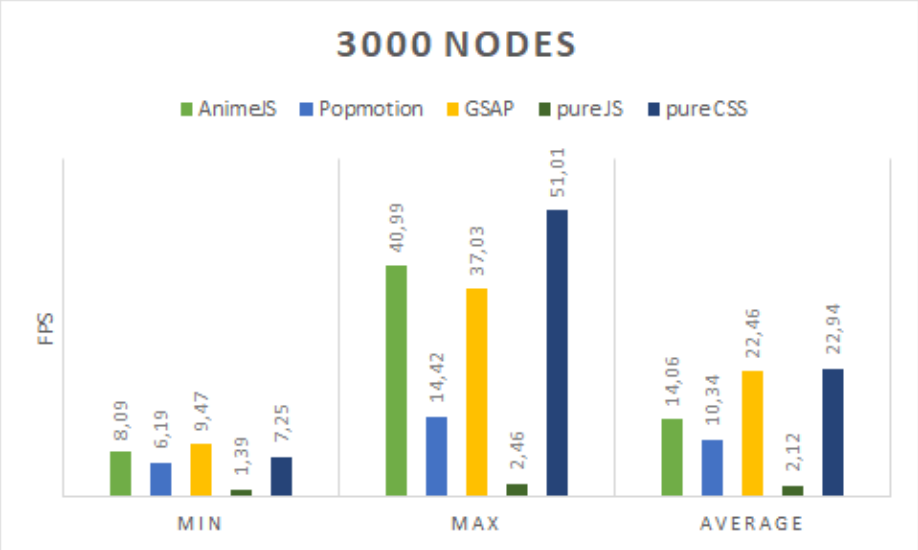

Fig. 2. Three thousand nodes test results

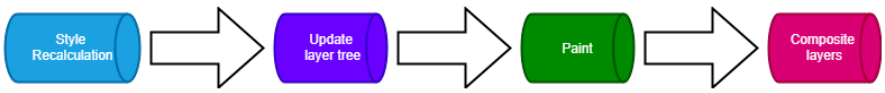

Fig. 3. Browsers rendering pipeline

- Style recalculation – browser determines which CSS rules should be applied to which DOM elements.
- Update layer tree – when a browser knows which CSS rules to apply to the current element, it calculates which size element will be. This step can take a lot of time due to different relations between nodes, for instance, parent DOM node can affect size of its children.
- Paint – painting gives page color for back-grounds, text, shadows and so on. Typically, a web page has multiple layers, so the painting comes over all the layers.
- Composite layers – as it was previously said, a web page may have multiple layers to be painted. At this step, browser determines the order of these layers, which means that browser needs to determine, which layers should be at the top of other layers. It is especially important for the layers which have overlapping.

  temize

  We conducted same tests again and measured time spent on every stage. Those tests have shown the following results (see table 2 and 3).

Table 2. 1000 DOM nodes test result

Time spent (ms) on	Anime.js	Popmotion	GSAP	pure JS	Pure CSS
Style recalculations	1667.8 (54.9 %)	1005.5 (41.9 %)	1013.7 (42.6 %)	81.2 (0.8 %)	4996.2 (61.5 %)
Update layer tree	423.0 (13.9 %)	533.4 (22.2 %)	588.6 (24.8 %)	21.5 (0.2 %)	1644.2 (20.2 %)
Paint	7.1 (0.2 %)	38.1 (1.6 %)	20.3 (0.9 %)	4.0 (0.0 %)	297.4 (3.7 %)
Composite layers	18.4 (0.6 %)	111.3 (4.6 %)	119.4 (5.0 %)	0.7 (0.0 %)	66.9 (0.8 %)

Table 3. 3000 DOM nodes test result

Time spent (ms) on	Anime.js	Popmotion	GSAP	pure JS	Pure CSS
Style recalculations	1697.5 (43.1 %)	1016.8 (38.1 %)	1056.4 (41.2 %)	401.7 (4.1 %)	5208.8 (61.2 %)
Update layer tree	500.8 (12.7 %)	537.7 (20.2 %)	543.4 (21.2 %)	0.0 (0 %)	1839.1 (21.6 %)
Paint	17.6 (0.4 %)	61.4 (2.3 %)	41.9 (1.6 %)	0.0 (0 %)	318.2 (3.7 %)
Composite layers	6.4 (0.2 %)	106.9 (4.0 %)	119.5 (4.7 %)	0.0 (0 %)	24.3 (0.3 %)

Although style recalculations, update layer tree, paint and composite layers are the most important parameters in terms of animation and smoothness of user interface, they do not have so much influence in our case. These results do not show any performance correlation. It is worth mentioning, that the slowest library spent in an idle state most of its test time (6102 ms), whereas GSAP and Anime.js spent 202 ms and 7 ms respectively. There is a big difference in FPS, but the idle state can not be considered as a factor, which influences performance.

The next comparison criterion is popularity. The most popular library is Anime.js, which has 32.8 thousand of github stars. Then follows Popmotion

having 17.3 thousand of github stars. The fastest library has only 9.8 thousand stars.
In terms of support Anime.js has the best numbers: 44 contributors, 96 open and 392 closed issues. At the same time, Popmotion has 53 contributors, 154 open and 367 closed issues, which is a bit worse than Anime.js. Finally yet importantly, GSAP has only two contributors, 2 open and 278 closed issues, what is not satisfactory, because the project depends on a small number of persons.
The last point we will consider in this article is memory usage. On the figure 4 the memory usage can be seen.

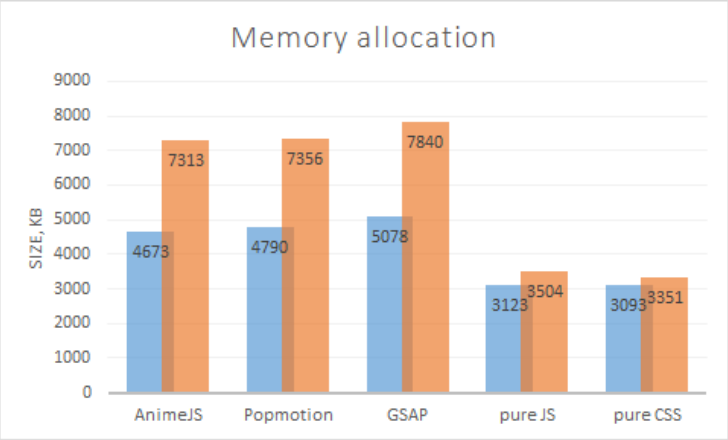

Fig. 4. Memory allocation

For every approach considered, there are two bars on the chart: the blue bar represents memory allocation in 1000 DOM nodes test and the orange one represents memory allocation in 3000 DOM nodes test. All the tests were conducted for 30 seconds.
The best results show pure JS and pure CSS solutions – they consume the lowest amount of memory. The fastest library, GSAP, takes more memory than other libraries. The library, which requires the least memory, is Anime.js

5 Conclusion

We reviewed some of the most popular animation libraries, conducted performance tests and gathered information about how popular and supported libraries are. The most performant library is GSAP, so if there is need of the best performance, it is recommended for use. In addition, GSAP is the only library, which has plugin system and different utilities. Moreover, it is fairly compatible – it supports modern and old browsers and can be used

alongside with jQuery, React, Vue and other frontend tools. In spite of its benefits it has the following disadvantages: the biggest library size and the highest memory consumption. It is recommended for this library to decrease bundle size and memory usage.
In addition, GSAP has a huge disadvantage – the project is supported by a very small group of contributors. Therefore, for production usage Anime.js is recommended – it is not so performant, however, it is the most demanded animation library, which has a lot of contributors, good documentation and well supported.
The last library – Popmotion – has shown the lowest FPS on average, but it has the lowest size – only 11.3 Kilobytes. This library can be used for very small projects, where bundle size is critical. If this library increases FPS shown, it can improve its popularity and total market usage.
The results of using non-libraries approaches are the following: pure JavaScript and CSS solutions have the lowest amount of memory usage. Pure JavaScript solution shows the smallest FPS on average, but this approach is more flexible than pure CSS. However, pure CSS has the highest average FPS in 3000 DOM nodes test, so it is recommended for use in case of simple animations.
As a result, GSAP has the best performance among other libraries, Anime.js has the best support, which is important for real projects, and Popmotion has the lowest size.

References

1. Zajceva E.N., Manvelov N.S: “Animacija v veb-dizajne: preimushhestva ispol’zovani-ja.” Nauka i obshhestvo v uslovijah globalizacii. 107-109 (2017).
2. Garrett, D: “Veb-dizajn. Jelementy opyta vzaimodejstvija”. Simvol-pljus. Russia Saint Petersburg (2018).
3. Richard Williams: “The Animator’s survival kit”. Faber & Faber, London, UK (2015).
4. Top JavaScript libraries for animations, https://www.sitepoint.com/our-top-9-animation-libraries/, last accessed 2019/10/19
5. Top JavaScript libraries for animations, https://hackernoon.com/10-javascript-animation-libraries-to-follow-ee193196776, last accessed 2019/10/19
6. Github search for animation libraries, https://github.com/search?l=JavaScript&o=desc&q=animation&s=stars&type=Repositories, last accessed 2019/10/19
7. TIOBE index of programming languages https://www.tiobe.com/tiobe-index/, last accessed 2019/10/19
8. A small place to discover languages at github, https://githut.info/, last accessed 2019/10/19
9. JavaScript animation engine https://animejs.com/, last accessed 2019/10/19
10. Lightweight JavaScript animation library with a simple, yet powerful API. https://github.com/juliangarnier/anime, last accessed 2019/10/19
11. Simple libraries for delightful interfaces, https://popmotion.io/, last accessed 2019/10/19
12. A functional, flexible JavaScript library, https://popmotion.io/pure/, last accessed 2019/10/19

13. Professional-grade JavaScript animation for the modern web, https://greensock.com/gsap/, last accessed 2019/10/19
14. FPS definition, https://techterms.com/definition/fps, last accessed 2019/10/19
15. What is Frame rate and Why is it important to PC gaming, https://store.hp.com/app/tech-takes/what-is-frame-rate, last accessed 2019/10/19