Performance Testing

The performance is one of the main requirements of our project (see Product Planning, 1.1.1 must haves). Octopeer must be unnoticeable when it comes to system performance. Therefore this is stated as one of the main requirements of our project. Hence we want to tell you the outcome of our small research.

Methods

As the performance of our extension is mainly something that is based on the feel of the webpage. Therefore the first part of this test is performed based on the first look: how fast does this website respond, or does it freeze completely? These questions are the basis for further investigation. This investigation is made possible with the tools Chrome provides us. The internal JavaScript profiler serves nice statistics based on the usage of a GitHub page. It enables us to dive into the code and see where the most time is consumed. This is the complete toolset that is needed to run these small experiments.

The measurements are done on an HP ZBook 15 (which is a high end machine) that has several processes running including an IDE. This is done in order to simulate the average developer. The tests are done on the latest version of our project (v1.0.0).

Results

Let's start off with the good part of the research. Most of the tracking features can be enabled and used without any noticeable performance drops. The profiler confirmed these observations. Most of the data that is submitted to the database, can be send in a matter of milliseconds (at most 100ms, average around 20ms).

However there are 2 main exceptions to this rule. Those are the mouse scroll event and the HTML page. These particular pieces of data cause the DOM to be rewritten. It is not a problem for small pull requests, yet for big pull requests it is. In this context "big" can be seen as a pull request with more than 20 commits and comments. All of these elements must be rewritten to contain data tags in order to know whether they scroll into view or out of view. The same must be done for when the HTML page gets posted to the database. The rewriting can take up to 6 seconds for a rather large pull requests. During this time the user is unable to interact with the page.

Conclusion

Since the performance for most of the options is unnoticable for the user, we have decided to enable them by default. The 2 exceptions found during the test have caused some discussion in our group. We came to the conclusion that it would not be fair to the user to enable these by default. The performance can be a reason the user uninstalls the extension. This is worse than missing some pieces of data. As a group we decided to change their default values to off and place a small note at the bottom of the options page.