**Abstract**

JavaScript is one of the most used programming languages in term of website application development. Along with the increasing complexity of the web application, more and more JavaScript Frameworks (JFS) are created to support the programmer in website development. Nowadays, developers have too many options (JSFs) to choose when the start of a new web project which leads to a challenge for the developer to choose a right JavaScript framework to use. There are several reasons affect the decision of a developer when they choose JSFs. JSFs performances are one of the most important reason because it will affect directly to the web application performance. This research will survey the performance aspect of three most wide adopted JavaScript frameworks AngularJS, ReacJS and Vue. In this research, three Todo web applications of those three JSF have been developing then the experiment is run on three different browsers to test JSFs performance over three type of Document Object Model (DOM) operations CREATE, UPDATE and DELETE. The result data of the experiment have been processed and analysed to display the performance of the three JSFs on each browser and operations. The experiment shows that VueJS give the best performance on overall and all JSFs produce a better performance when running on Chrome.

**Introduction**

The way web developers develop web application has been changing dramatically in recent year because of the emergence of JavaScript Framework. A JavaScript framework is an application framework written in JavaScript. The JSFs have been created to facilitate the job of web developer when the complexity of developing website application increased. These JSFs aims provide a predefine structure which developers can follow to organise and structure their code (specially need when working with a large team and intensive collaboration). JSFs also provide build-in tools, methods, libraries that developers can use to speed up the developing process. By using JSFs, web developers have a ready-to-go foundation where they will build their website on top of that. It’s clear about the advantage of using JSFs. However, developers also need to consider the performance of the JSFs when using it, because the performance of JSFs affects directly into the performance of web application. Therefore, it is important to have a deeper insight into performance aspect of modern JSFs, so the developer can have more information when they select the JSFs for next web project.

In this research, three most popular JSFs AngularJS, ReactJS and VueJS (according to a survey of StateOfJS over 20,000 developers [1] and rating of developers on GitHub [2]) are chosen for the survey. The research focus to evaluate the performance of Create, Update and Delete HTML DOM operations of the chosen frameworks. Also, the “performance” term in this research comprises of 3 different aspects “Time Duration”, “CPU Usage” and “Memory Usages”. The goal of this research and experiment are to provide deeper insight into DOM operations performance of the selected JSFs and to answer the following questions:

* In overall, what JSF archive the best performance?
* Which DOM operations of the selected JSFs require most/less time and system resource (CPU, Memory)?
* Are there any particular web browsers that the selected JSFs achieve better performance when running on it?

The rest of the research paper is organised as follows: Sections II describe literature review and related works. The experiment design and execution are given in section III. Section IV is the data analysis and discussion of this research. Finally, conclusion and future work are described in section V.

*There are three reasons that I think this question is important. \\*

*Firstly, Performance is the most importance factor of web application nowadays. If your website responses too slow (more than 2 seconds) very likely that you are losing your users to another website that provides the same functionality but have higher performance. If your website uses one of those frameworks, this research can give your a clear picture of your how well your framework performs in compare with the others. \\*

*Secondly, The answer for this research can help software companies and developers have more information to consider when they select a JavaScript framework to develop a web application. This also helps them have better insight into the performance aspect of those JSFs so they can choose the right tool for the right job.\\*

*Thirdly, human tends to stick with what they are familiar. The developer is human; they tend to prefer their familiar JSF. This research gives evident and information about the performance of Angular, ReactJS and VueJS which can motivate developers to adopt new better JSF.*

**Background**

**AngularJS**

**ReactJS**

**VueJS**

**Related Works**

In [3] Andreas el al. does the comparative survey on many aspects of six JavaScript's frameworks jQuery, ExtJS, MonoTools, Prototype, YUI under seven different browsers. The author measure complexity and maintainability of JSFs using various metrics such as size metrics (number of statements, comment lines, lines of code etc.), complexity (cyclomatic complexity, depth and branch) metrics and maintainability (Halstead, MI, program volume) metrics. The test has conducted on seven different browsers and three operation systems (Linux, Window and MacOS). This research gives a very good insight into quality aspects of those frameworks. However, on the JavaScript framework performance aspect, the authors have only done the performance evaluation on “Time execution duration” of the selected JSFs which is not enough. In addition, those frameworks are losing their popularity due to out of date technology and design patterns. In my research, the “performance” evaluate also included “CPU usage” and “Memory Usage” which are also important factors of performance evaluation and the target JSFs have been carefull selected to match with currently popular and adopted JSFs.

In [4] Selakovic el al has an empirical study about the performance issue and how a developer can optimise their code improve performance. More than sixteen popular client-side and server-side JavaScript projects have been surveyed by the author. There three main contributions of Selakovic with his research.

* Identified root-cause of the issues that can affect the performance selected JS projects inside its code base.
* Provide valuable suggestion for JS developers to find and fixed the performance issues in existing in their project.
* Reveal various recurrent optimisation pattern which can always apply to avoid performance issue in any JavaScript project such as “avoid for-loop”, “implicit string conversion”, “use instanceof” etc.

Selakovic empirical study provides some good idea which is adopted when I design the experiment method in this research such as:

* Using the genuine virtual machine to execute performance experimental that can reduce the side effect of background process interference while running the test.
* N Warm-Up run test to make sure everything is ready and stable before running the real test to collecting data.

In [5] Kumar el al provides an introduction to two most used JavaScript framework AngularJS and ReactJS. The author gave their comparation rating between two frameworks over 14 criteria such as DOM rendering technic, development architecture, learning curve, data binding technic, mobile compatibility, compile size and community. By reading Kumar research, the reader is provided general information about strong and weak points of AngularJS and ReactJS. The performance aspect between two JSFs is given in favour of ReactJS due to React’s unique feature “Virtual DOM Rendering”. However, this research does not provide experiment data and support evidence to justify their rating decision which reduces the credibility of the research in reader’s eyes.

In [6] of Sotiris el al provide another interesting research that is about performance evaluation framework for DOM selectors of JavaScript Libraries. The research conducts a huge test-suite which cover 263 type of JavaScript selectors, run the test-suite on six popular JavaScript libraries (jQuery, DoJo, Prototype, ExtJS, YUI and MooTools). The main contribution of Sotiris is a complete framework (comprise of methodology and a set of tools) that web developer can use to evaluate the performance overhead by JavaScript library DOM selectors in their web application. Figure out the selectors that cause performance issue and optimise them. The methodology conduct in [6] that consists of three main parts: design test case and environment, run experiment a collect data, interpret the collected data and discuss the finding result.

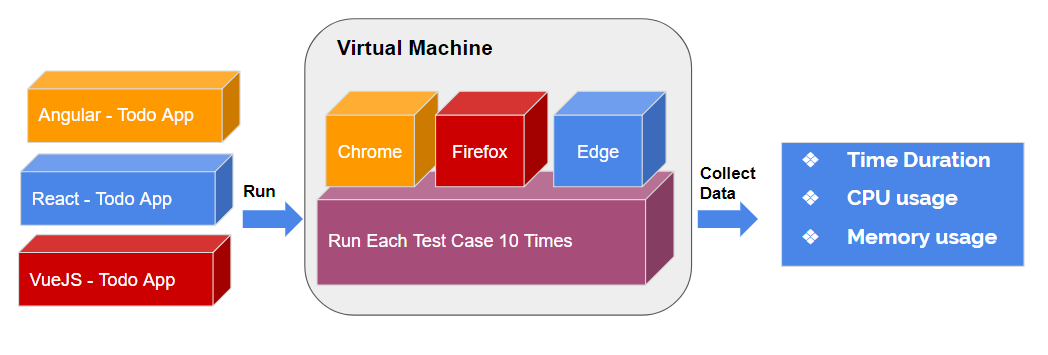
**Problem Definition**

**Research Methodology and Research Design**

This section describes an overview of experiment steps, experiment applications, test cases design and experiment environment specification and tools used to collect data.

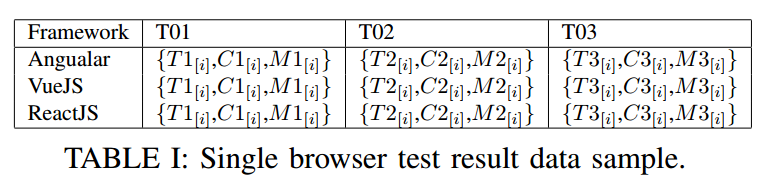
**Experiment Steps**

There are three main steps are used in the experiment to collect data in this study as shown in the figure below that:

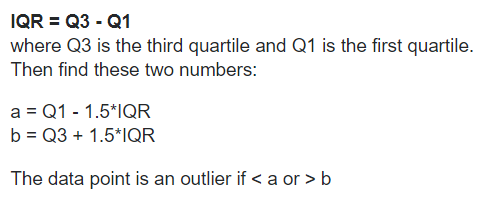


* **Step 1 – Develop Experiment Application:** Developed three Todo Application with in each targeted JSFs AngularJS TodoApp, React TodoApp and VueJS TodoApp. The TodoApp specification is given in detail in next section “Experiment Application.”
* **Step 2 – Deploy and Run Test Cases:** Deploy Todo Applications to the test environment (a genuine virtual machine) and for each application on each browser (Chrome, Firefox, Microsoft Edge) do following tasks:
  + Run the warm-up test cases 3 times. The purpose of warm up test-suite to stabilise the environment and make sure that application is ready.
  + Run each test cases (Create/Update/Delete) in the test suite ran ten times to collect the needed data. The detail of test cases and its execution steps are described in the section “Test Cases Design”.

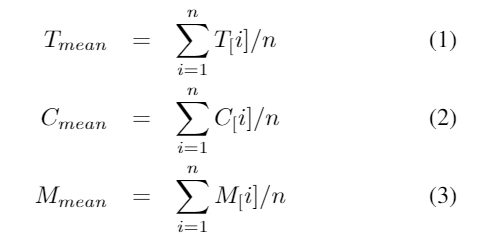
The data collected are organised into three 3-dementional matrixes. Each of the matrix storage data collected while running test-suite in one browser as described in table 1.



* **Step 3 – Remove Outliers and Process Data:**  In this step, we do two following things.
  + The first one is to filter or eliminate the outlier data. Although, we already try to remove outside factors that can affect the data by using a new separate virtual machine to run the tests. However, outlier data can still occur, so we need to remove them if any before moving to the next step. The outlier in this research is removed by using interquartile range formula as described below.



* + After removing the outlier, the next step is to process the data. For each data-set (group by test-case and data-type) in the matrixes, we calculate the mean value by using equations:



**Experiment Applications Specification**

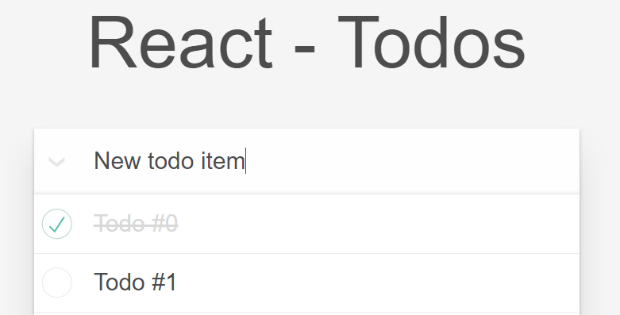
In this research, Todo application has been chosen as the reference application and three TodoApps implemented using all three selected JSFs. There are several reasons to choose TodoApp for the experiment:

* Todo application is simple and easy to develop.
* We can build an entire application on the front end. It does not need next a backend to persist data.
* The most important is that we can test all kind of Html DOM operations with Todo application (Create - Add New Todo; Update – Mark Complete Item; Delete – Remove Todo Item)

The Todo application interface is very simple with three main components that are a to input textbox, todo item and a todo item list.

* Todo input textbox: This text box allows the user to enter todo item description and press enter to add new Todo item to Todo List.
* Todo Item: a todo item consists of a check box to mark the item as completed, item description section and a red “X” button to remote this item.
* Todo List: This component is where all Todo items are displayed in chronological order (except deleted items).

The figure below shows an example user interface of ReacJS Todo application. The UI is the same for the other two applications.



**Environment Specification**

A genuine virtual machine has been setup to use as the experimental environment in this research. Below is the specification of the machine:

* Operation System: Window 10 Pro
* Processor: 4 virtual cores 2.60 GHZ
* Memory (RAM): 8192 MB.
* Intel HD Graphics 3000 shared.
* Installed software:
  + Chrome 69.0.3452
  + Firefox 60.0.1
  + Microsoft Edge 42.17134.1.0
  + Process Explorer 16.21
  + NodeJS v6.14.2

We also turn off some unused window background service and uninstall all pre-installed plugin of three test browser to avoid unexpected affect from them.

**Test Cases Designs**

There are three test cases have been defined to run in the experiment applications in this study. Those test cases are designed to test and evaluate the performance of 3 DOM manipulation operations.

* Test Case T01: Create 1000 Todo Items. This test case uses to evaluate the performance of the JSFs on creating HTML DOM operation.
  + Initial Step: Start application with empty todo item list.
  + Execution Step: Create 1000 new Todo item with description “Todo Item #[Index]”
* Test Case T02: Update 1000 Todo Items. This test case uses to evaluate the performance of the selected JSFs on update existing HTML DOM element operation.
  + Initial Step: Start the application and add 1000 todo item with description “Todo Item #[Index]”
  + Execution Step: Mark 1000 toto items as complete (click on the check box).
* Test Case T03: Delete 1000 Todo Items. This test case used to evaluate the performance of the target JSFs on deleting HTML DOM operation.
  + Initial Step: Start the application and add 1000 todo item with description “Todo Item #[Index]”
  + Execution Step: Click on the red “X” button on each item to delete 1000 todo items.

Each test case is run ten times, and three different data types would be collected:

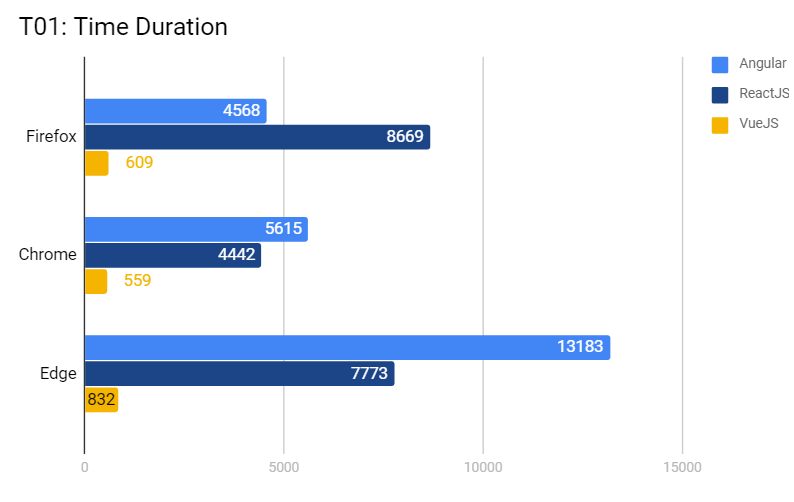
* Execution Time Duration: the time needs for an application run and finishes the “Execution Step” of the currently running test case.
* CPU Usage: is defined as the peak CPU consumption in “Execution Step.”
* Memory Usage: is defined as the maximum amount of Memory that application needs in “Execution Step”.

**Result & Discussion**

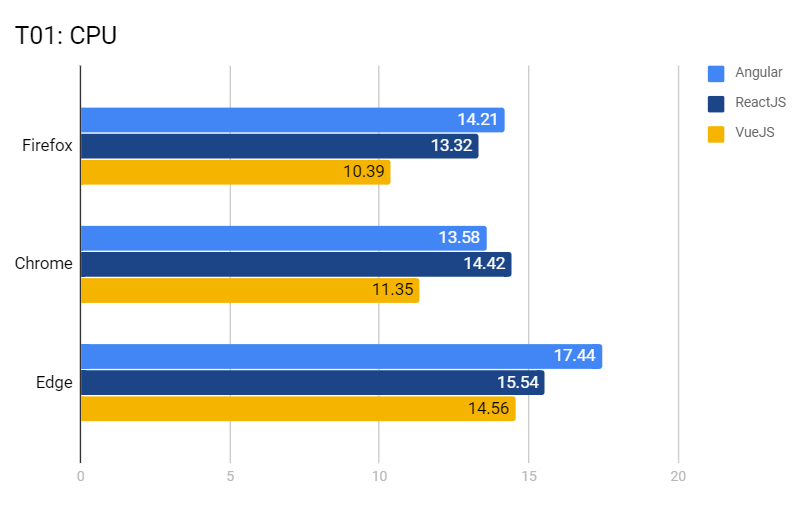
**Test Case T01 - Create 1000 Todo Items**

In this test case, we evaluate the performance of target JSFs on create HTML DOM. The final processed data of this test case can find in the appendix section \hyperlink{fig:T01-Data}{Test Case T01 Data}. In overall, VueJS outperform Angular and ReactJS in all performance categories (Time, CPU and Memory). Chrome browser shows slightly better support for DOM creation operation than Firefox and Ms-Edge. The detailed analysis with charts for each category will be given below.

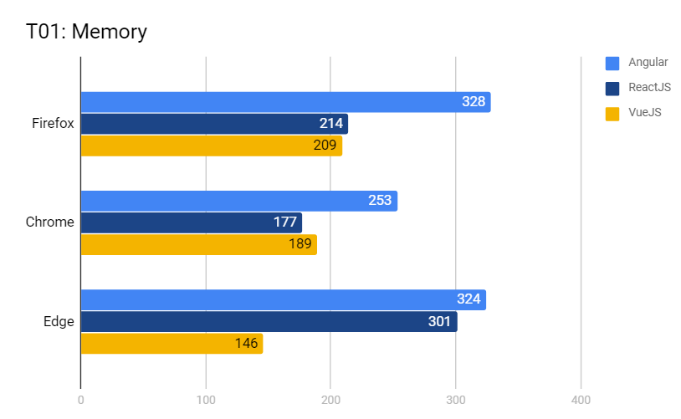
* T01 - Time Duration: The result of this aspect shows that VueJS have significantly better performance in compare to Angular and ReactJS. The average time for VueJS complete the task is less than one second in all test browser while Angular, ReactJS took much longer.



* T01 – CPU Usage: VueJS also score the lowest CPU consumption with around 10% -14.56% while Angular and ReactJS use 17.44% and 15.54% of CPU respectively to finish the first test case. Firefox needs less CPU in this test than the others.

****

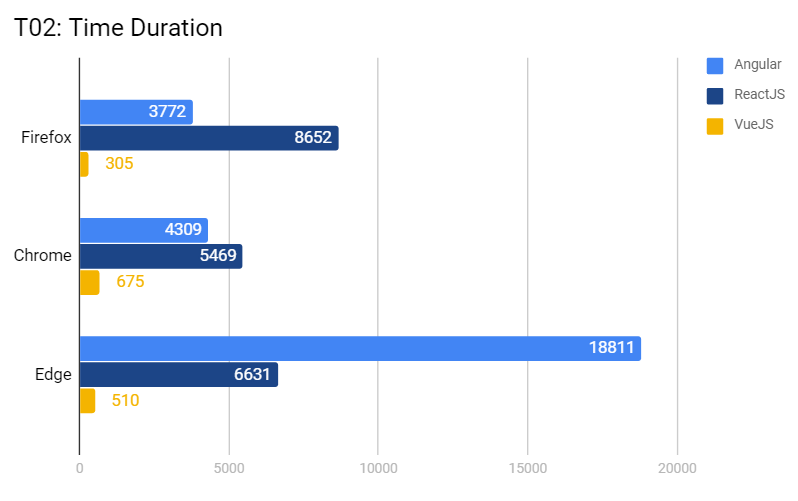
* T01 – Memory Usage: Angular use the most memory in test case T01 at around 253-324MB. The best performance in this category is VueJS at around 146-209 MB. Chrome browser shows better memory management that Firefox and Edge.

****

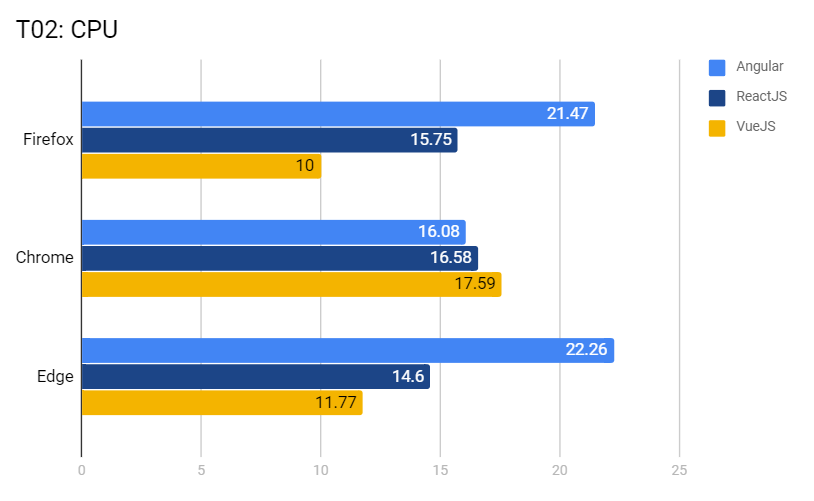
**Test Case T02 - Update 1000 Todo Items**

In this test case, we evaluate the performance of target JSFs on update HTML DOM. The final processed data of this test case can find in the appendix section \hyperlink{fig:T02-Data}{Test Case T02 Data}. In general, the “update” operation took less time to execute than “create” operation except in Angular application on MS-Edge browser when it took more time to update than create. That VueJS give better performance (cross-browser) in all categories in this test case. The second place is ReactJS and Angular have the worst performance.

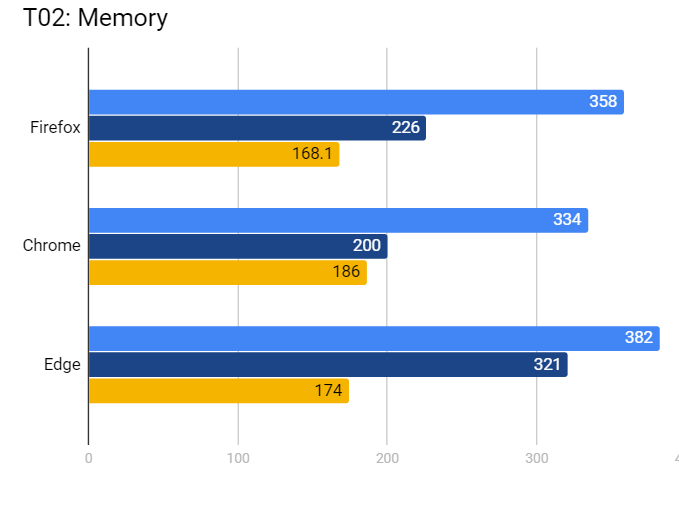
* T02 – Time Duration: The result show that VueJS run a least 10 times faster than Angular and ReactJS. The average time duration of VueJS, ReactJS and Angular are 475ms, 6910ms and 8964ms respectively. Chrome browser run faster than Firefox and Edge in this test case.

****

* T02 – CPU Usage: All JSF consume more CPU resource for update operation than that in create operation. VueJS consume the less CPU resource than other JSFs on Firefox (10%) and Edge (11.77%) but slight higher in Chrome (17.59). Angular framework need the most CPU resource on at about 20% average in comparation with React and VueJS are 15.6% and 13.1% respectively.

****

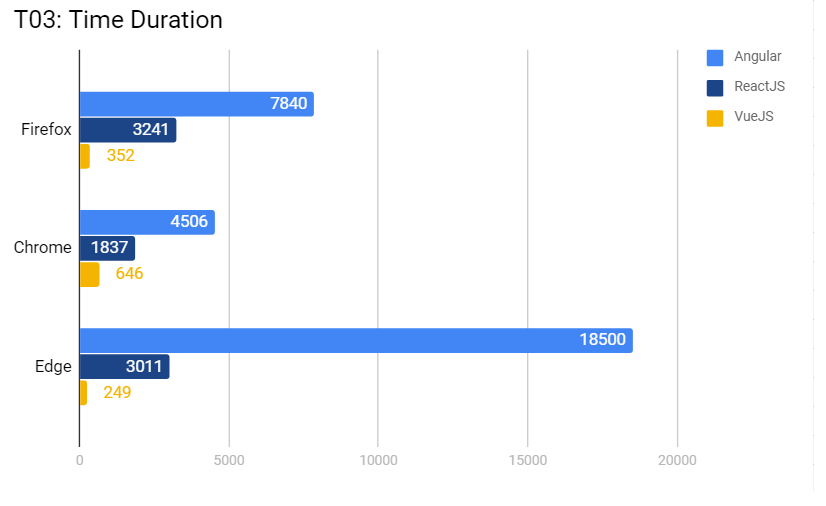
* T02 – Memory Usage: Memory usage show the same pattern with CPU when it uses more memory in DOM update operation than in DOM create operation. Angular have the poorest performance in this category with 353MB on average. The best performance in Memory usage is VueJS with 176MB on average (50% less than Angular).

****

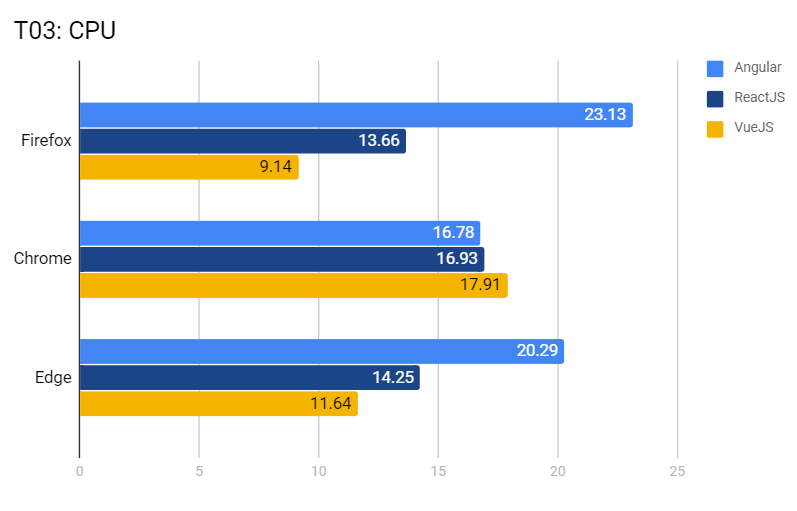
**Test Case T03 - Delete 1000 Todo Items**

In this test case, we evaluate the performance of target JSFs on delete HTML DOM. The final processed data of this test case can find in the appendix section \hyperlink{fig:T03-Data}{Test Case T03 Data}. In T03 result, VueJS is the best performance and angular have the worst performance. In addition, VueJS and ReactJS need less time to completed in T03 the test than other two test cases while Angular need more time than those previous one. All JSF have better performance when run on Chrome than Firefox or MS-Edge.

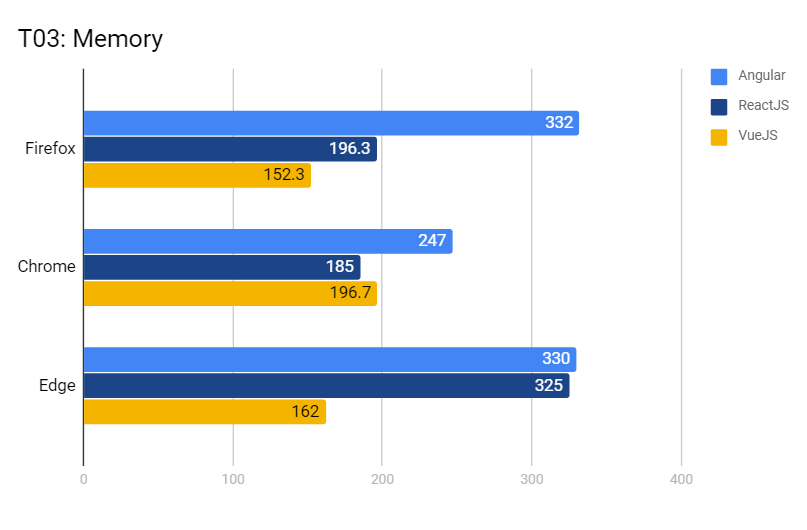
* T03 – Time Duration: The result shows that VueJS is the fastest framework with 415ms on average while ReactJS and Angular are significant slower with 2696ms and 10270ms respectively.

****

* T03 – CPU Usage: VueJS consume the less CPU resource at 12.8% on average. ReactJS need a bit higher at 14.9% on average, and Angular consume the most CPU resource at around 20.2% on average.

****

* T03 – Memory Usage: In this result, the best memory usage performance is VueJS at 170MB on average, second place is ReacJS at around 235MB and the worse is Angular at 303MB on average.



In summary, after the analysed result of three test case over three performance aspect Time Duration, CPU Usage and Memory Usage. We have following conclusion for the research questions of this study:

* What JSF archive the best performance?
  + It is clear that VueJS archive the best performance in all test cases and across performance category. In contrast, Angular has the worst performance
* Which DOM operations require most/less time and system resource (CPU, Memory)?
  + “DOM creation” operation need more system resource and time to execute than Update and Delete operations. “DOM Delete” operation need the less resource and time.
* Are there any particular web browsers that the selected JSFs achieve better performance when running on it?
  + From the test result, we can say that all selected JSFs have better performance when running on Chrome browser. In addition, we can see that all JSFs framework have the worst performance on MS-Edge and MS-Edge does not optimise when handling a large number of DOM elements.

**Conclusion**

In this study, we have surveyed the DOM manipulation operation performance of three most popular JavaScript framework Angular, ReactJS and VueJS. The performance is appraised on three category Time Execution Duration, CPU Usage and Memory Usage by using a set of three test cases. Each of the test cases is designed to evaluate one DOM manipulation operations as follow Create, Delete and Update. The result of this study shows that VueJS have the best performance across performance categories and operations. The result is a surprise because VueJS is less popular in compare to Angular and ReactJS. We also found that “Create” operation required more time and system resource than that in “Update” and “Delete”. On the browser supported side, we found that all JavaScript frameworks have better performance when it runs on Chrome browser than the other two Microsoft Edge and Firefox.

The developers can use this study to gain more deep insight in to the performance aspect of Angular, ReacJS and VueJS when they want to select a JavaScript framework for their next project. In the future, we also want to expand our research to add more JSFs. We also want to survey on other aspects of a JSFs such as architecture, learning curve, community and social support because of these aspects are also important to consider when choosing a JavaScript framework.

**[1]**