**what you set out to do?**

In this research, I started by choosing the topic which I have most interested that is JavaScript Frameworks(JSF). After having the topic, I started to narrow down the scope of the topic since there are too many (more than 20) of JSFs out there. Then I choose the three most popular [1] and widely use [2] by the developer to investigate that are Angular, VueJS and ReactJS. Finally, I do a survey on existing research about those JSF to find the gap that I can fill in with my research. In result, I found the topic about the performance of those three JSF on HTML Document Object Model (or HTML DOM) manipulation. I choose this topic because it doesn’t any formal research. Besides that, I choose to evaluate the performance of DOM manipulation aspect but not another because all of the web applications, more or less, is involved with DOM manipulation.

After defining my research question “Angular, ReactJS, VueJS: Cross Browser Performance Survey in Intensive DOM Manipulations Application”. Look closer to my research question I realise that it still has two things need to clarify before I start my research.

What is “Performance”? At first, I only thing that a performance of a JSF is the execution time. It is true but is it not enough? Finally, I come up with the definition of “performance” in my research is include three attributes “Execution Duration” (execution time), “CPU Usage” and “Memory Usage”.

What DOM “manipulations” this research will cover? The answer to this question will affect how I design the experiment test cases. In this research, I would like to cover all DOM manipulation operation: “Create, Delete, Update.”

**how you met these goals?**

To answer my research question, I need to answer the following question:

* How will I go to survey those JSF specifically on HTML DOM operation?
* What data do I need to answer my research question?
* How am I going to analysis the data?

The best way to survey a JSF that creates an application with that framework. My plan to survey those three frameworks are created the same application with each JSF and then run the same test case on each application over 3 three browsers to collect the data. I decide to create To-do application because its logic is simple, and I can cover all defined DOM manipulations (Create, Delete, Update).

The next step is to define the test cases that I can collect data to answer my research question. As mentioned in the “Performance” definition above for every time run a test case I will collect three information:

* The total run time of the test case from start to finish
* The peak Memory usage of the browser while running the test case.
* The peak CPU usage of the browser while running the test case

**Test Case Design**

About the test case design, at first, I planned to run a single test case that covers all three operations Create, Delete, Update like this:

* Step 1: Create 1000 todo-items.
* Step 2: Mark 1000 todo-items as complete.
* Step 3: Delete 1000 todo-items.

However, after running several times over three todo-applications, I found that there is framework fast at creating operation (step 1) but slow at update or delete operation. Therefor, a test case combines three operations can lead to misleading in analysis phase because we cannot differentiate which operation perform fast or slow. In the end, I decided to separate the test case into three small test cases and record them separately. That is triple the effort and data, but I can get a better insight and know how JSF perform in each operation when doing data analysis. Below is my final set of the test case:

* T01: Create 1000 todo-items
* T02: Update 1000 todo-items (Create 1000 todo-items first then start record data)
* T03: Delete 1000 todo-items (Create 1000 todo-items first then start record data)

**What are the obstacles you meet while implementing the research and how I dealt with them?**

There are several obstacles/difficulties while I am doing the research. I will go over each the obstacle and solution of mine for it. The solution that I use may not perfect, but it is the best solution that I can figure out at this time with my knowledge and understanding.

**Leaning three different JSF**

To develop experiment Todo applications, this is the first obstacle I run into. I have experience with developing a web application in Angular JS, but zero-experience for the other two frameworks VueJS and ReactJS. Although, all of them are JavaScript, the architecture, developing principal and library API are different for each JSF. I can not just spend a month to study everything about the JSF. My purpose is not to master that JSF but create a simple Todo application using the JSF.

My approach to this problem by following steps.

* I go over the get started tutorial of each JSF to learning about the essential knowledge and configuration of the framework (VueJS for example). You can find that information on the official website of those JSF. I also watch some YouTube tutorial video about the framework; this can be very useful because you get to see other develop an application from scratch by using that JSF.
* After getting basic knowledge about the JSF, I started to build a very simple “HelloWorld” applications. At this stage, I already know how to make an application with the framework and run successfully it.
* The last step was to modify the “HelloWorld” application to “Todo list” application and only google or search for information when I got stuck.

By following above approach, I manage to study a framework and develop a Todo application on two days because I don’t waste my time on study any irrelevant with Todo application.

**Test case execution difficulty.**

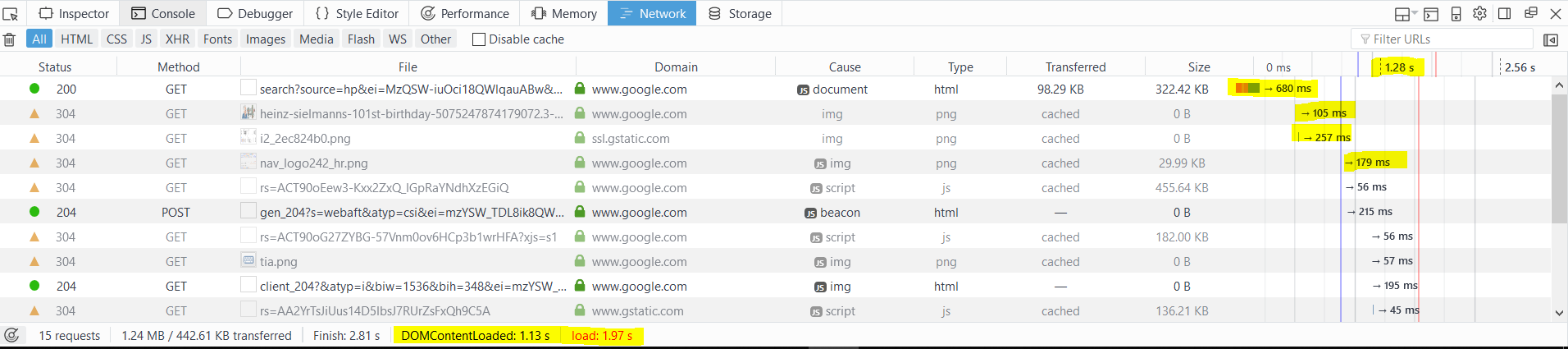
The problem is that impossible to execute the test cases manually because the time it takes to execute a test case, for example, create 1000 Todo items is about 20 minutes (this just an estimate). Manually run test case also involve human error and many other factors that lead to an incorrect result.

The solution to this obstacle is developing automation script to run the test case. This automation script develops in JavaScript with two main components.

* TestRunner: this component will take input from a set of test case’s instruction (TestSuite) then it will interact with the browser to execute the instruction step by step. The TestRunner also help me to record the execution time so that I don’t have to do it manually.
* TestSuite: this is the definition of each test cases with detail instruction so that TestRunner will know how to execute the test case. In general, a TestSuite will have two steps:
  + Preparation step: this step contain task need to do before the actual action of the test case is executed and the time of preparation step does not count in the execution time of test case. For example: Create 1000 todo items in “T02- Update 1000 todo-items”.
  + Execute step: this step contains the main task or activity that we want to examine in the current test. For example: Mark 1000 todo-items as a completed item in test T02.

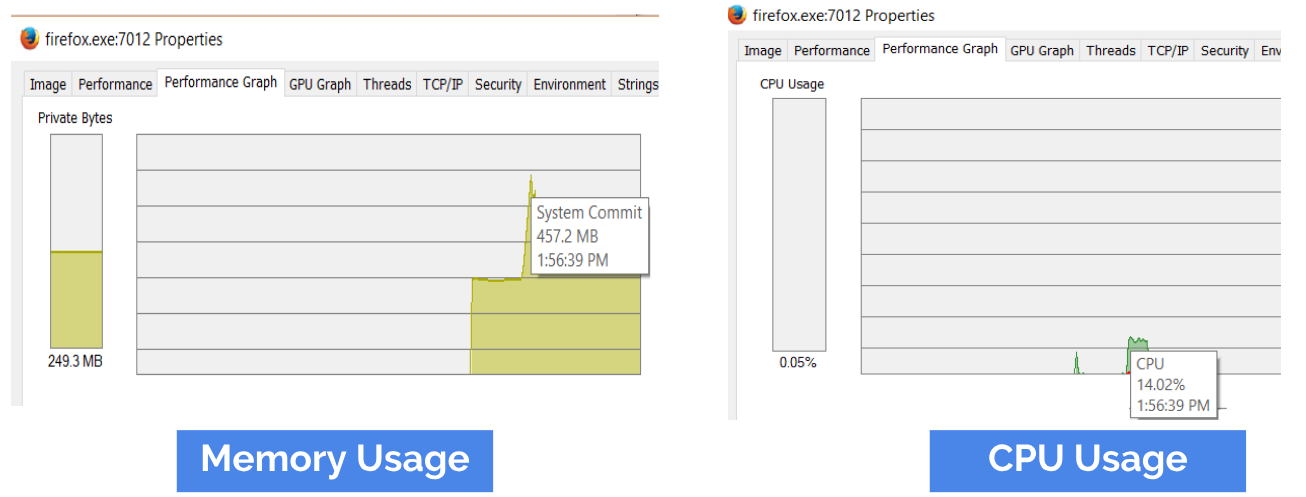
**Collect data difficulty.**

In this research, there is 3 type of data I need to collect in each run of the test case “Execution Time Duration”, “CPU Usage” and “Memory Usage”. However, the build-in feature of the browsers is not provided with enough information data that I need for this research. For example:

* All of the browsers don’t provide CPU information.
* The memory usage information is provided, but it hard understands. The memory usage calculation various from browser to the browser because each browser has different measurement methods.
* The “Time Execution Duration” is provided per request and event point (see yellow highlight in the screenshot below) make it difficult if not impossible to get exact execution time for each test.

To overcome this obstacle, I have to find a solution that does not depend on the browser and can produce consistent and accurate data.

* The solution for collect “Execution Time Duration” data is using the TestRunner to calculate the duration. By using the TestRunner, I can use config to start the timer exactly at the test step I want to measure and stop the timer right after the test finish.
* The “CPU usage” and “Memory usage” data are collected by using third-party tool call “ProcessExplorer” [3]. This tool will watch and summary information about CPU and Memory of the browser. So, we can get the usages data accurately and consistently. The screen-shoot below show the information provide by “ProcessExplorer” tool.



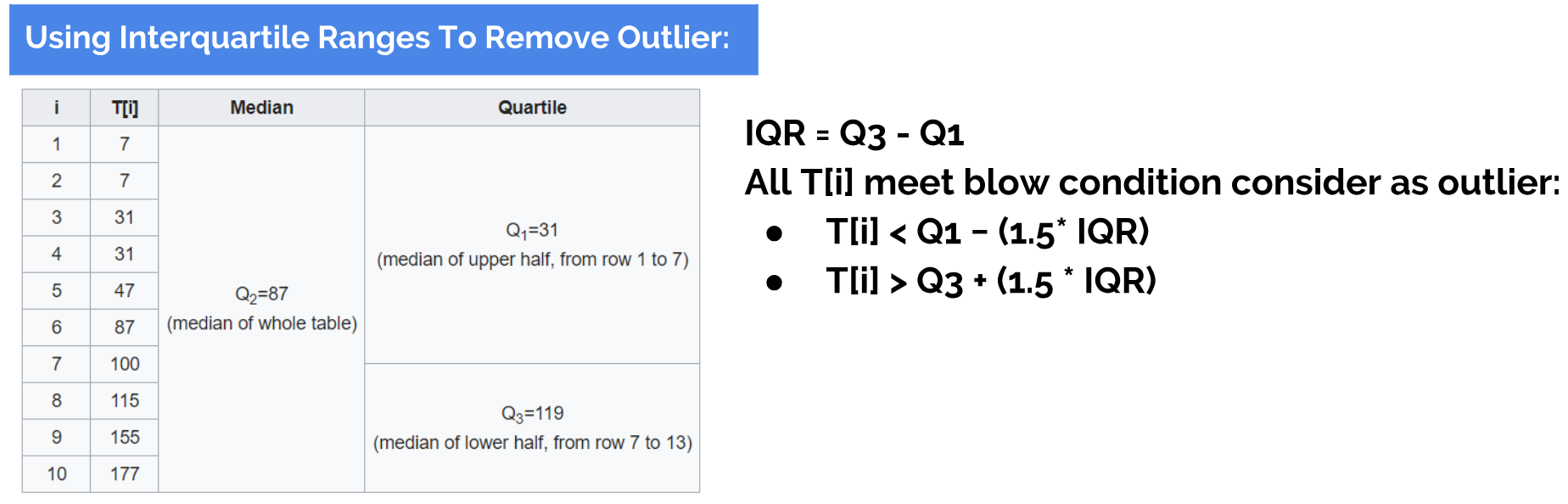
**Interferences by other application**s.

Another problem when running the test experiment that is the interferences of other background application. For example: some background service applications can occupy CPU resource while the browser is running the test. The browser running the test has installed some plugins or add-ins, and those plugins can also affect the performance of the current test.

My solution to this problem is that I setup a new clean virtual machine. In this virtual machine, I don’t install any application except three test browser and the “ProcessExplorer”. I also remove all plugins, add-ins in test browsers if any (in some browser may have some pre-install plugins).

**Eliminate outlier**

Although, I have tried my best to remove factors can affect the correctness of the collected data while running the test cases. However, outliers still can happen for some unknow reason and how to eliminate outliers also need to consider.

My solution is using Interquartile Range to remove outliner [4] as formula defined in the image below.

**how you executed the design?**

My plan to execute the experiment include four steps:

**Step 1 Preparation:** Below is tasks have been done in this step.

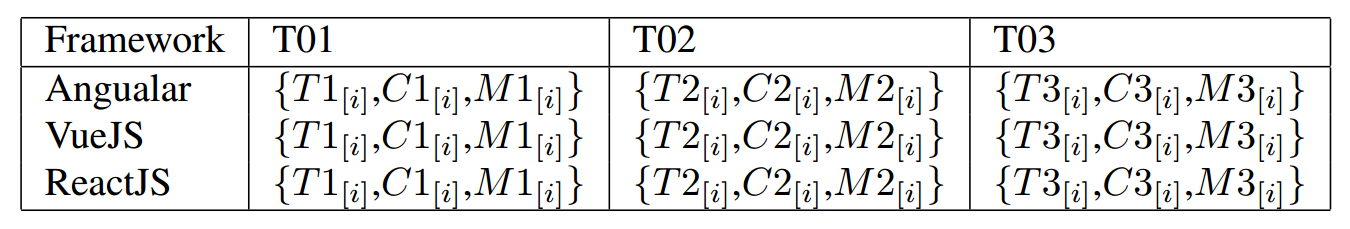
* I develop 3 Todo application with 3 JSFs (Angular, VueJS and ReactJS).
* Define what data I want to collect and how I am going to collect those data.
* Define the test cases which produce valuable data can help me address the research question and Develop TestRunner script to run those test case.
* Create a genuine virtual machine to eliminate as much as possible the interference of other software while executing the experiment. Install three experiment browsers (Chrome, Firefox and Edged) and Todo applications into the virtual machine.

**Step 2 Execution:** This step happens inside the virtual machine where I run the test cases to collect data.

For each browser and Todo app pair (For example: [Chrome, AngularTodoApp]; [Firefox, AngularTodoApp], [Edge, VueJSTodoApp] etc.) run following test case:

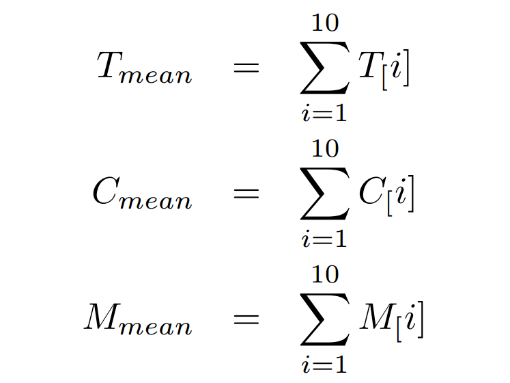
* **T01** run 10 time and for each run collect the tuple [TimeDuration, CPU, Memory].
* **T02** run ten times and for each run collect the tuple [TimeDuration, CPU, Memory].
* **T03** run ten times and for each run collect the tuple [TimeDuration, CPU, Memory].

At the end of this step, I have three (each for one browser) 3-dimensional matrix as screen-shoot below which are ready to be refined and analysis in the next step.



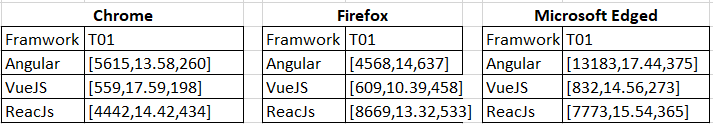
**Step 3 Refine Data and Process Data:**

In this step, I refined the data by removing the outlier in each set of the collected data. The formula to remove outlier described above in section Obstacles and Solution. After removing outlier, I calculate the mean value of each dataset group by type and test case by using following equation:



**Step 4 Data Analysis or How I carried out the analysis:**

After step 3, I get three 2-dementinal matrices one for each environment where values in the matrix calculated by the formula described in step 3. The screen-shot below is the example of data matrices after process the first test case T01 (tuple have format [Tmean, CPUmean, Memorymean]):

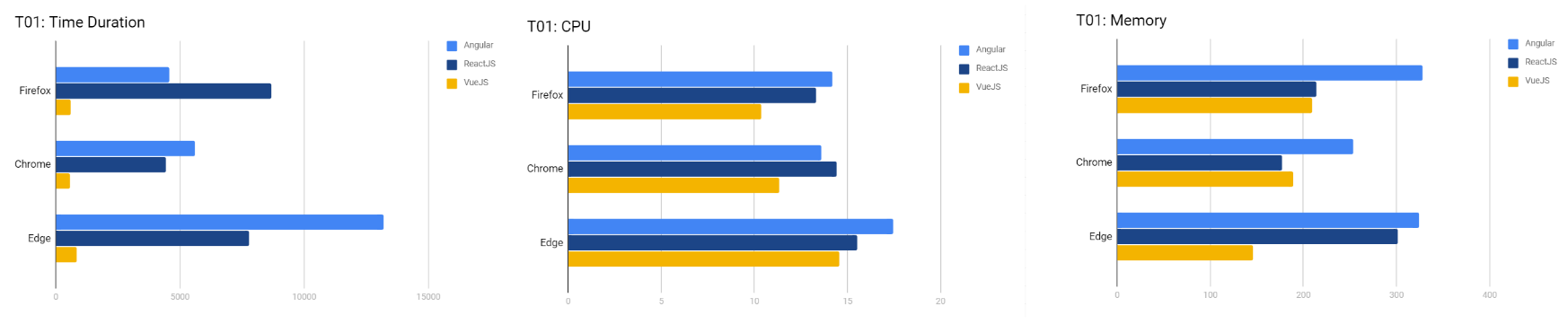


For each test case, I plot those value group by data types (Time Duration, CPU usage, Memory Usage), browsers and JSF separably then observe the information from the charts to answer the research question.

For example: This is how I analysed the first test case T01.

The picture below is plot chart of test case T01 for creating DOM operation, from the charts we can answer the following question:

* Which JSF has the fastest execution time in DOM Creation operation?
  + By looking at the first chart, we can answer that is VueJS.
* Which JSF consume the less CPU resource in DOM Creation operation?
  + By looking at the second chart, we can answer that is VueJS.
* Which JSF consume the less Memory resource in DOM Creation operation?
  + By looking at the third chart, although VueJS consume a bit higher memory than ReacJS in Chrome, in general, we can answer that is VueJS.
* In overall, the first test case T01 VueJS have the best performance across all criteria, next is ReactJS, and AngularJS performance is the worst among them.
* In addition, All the JSFs have better performance when running on Chrome than that on Firefox and Edge.

****

The same process is used to analysis the test case T02 (for DOM update operation) and test case T03 (for DOM delete operation).

**[1]** [**https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282**](https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282)

**[2]** **https://stateofjs.com/2017/front-end/results/**

**[3]** [**https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer**](https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer)

**[4]** [**http://www.purplemath.com/modules/boxwhisk3.htm**](http://www.purplemath.com/modules/boxwhisk3.htm)

The mini-research project, which is a reflection and exegesis of the process (worth 20%), we are looking for this:

You will be writing this up as a short reflection, where you clearly identify what you set out to do, how you met these goals, how you dealt with obstacles, how you executed the design and how you carried out the analysis. I note that you don't need to replicate the paper (above) in here. This is your opportunity to provide a rich (less formal) guide to what happened. Think of this as the explanatory narrative for the paper, where you provide the voice over that explains behind the scenes.

Don't just say "X happened": we want to know why things happened, what happened, what effect it had, and how you dealt with it. Feel free to throw in more detail.  
  
There's no formal page limit but it shouldn't be that long. It should be at least two pages long, as a guide.

Marking: 25% for meeting the proposal goals (or adapting them appropriately), 50% for effectively executing the planned design or providing clear details of adaptation, 25% for correctly carrying out analysis and justifying what you did.