**The design and operation of web browsers**

Web browsers have been a powerful tool used to run web applications. These have evolved over the years and each time programmers are finding ways to optimize these browsers making them execute code faster for better user experience.

Web browsers can either be single-process browsers or multi-process browsers. Unlike single-process browsers that were used previously, modern browsers that operate using multi processes offer a much better user experience in terms of better stability, higher performance with fast execution, and higher level of security. Modern browsers do all these by creating multiple processes for each tab that is opened and these processes can create more than one thread allowing the browser to perform more than one task concurrently. This makes the usage of these browsers much faster and incase one process crushes, it does not affect other processes running in other tabs. Modern browser components that are high-risk are isolated in sandboxes making them inaccessible by vicious or malicious code.

Web browsers have a JavaScript engine embedded within to execute JavaScript code. Previously used browsers like internet explorer used the “Chakra engine” for JavaScript code execution. However, an open-source JavaScript engine named “V8” was invented in September 2008 by “Lars Bak”. Other notable engines include the “SpiderMonkey” and the “JavaScriptCore”.

A JavaScript engine works in such a way that once a JavaScript file is received by the parser located within the engine, it is checked for any syntax errors. An Abstract Syntax Tree (AST) is then created and tokens are laid out and sent over to the interpreter. The interpreter then interprets these tokens line by line and sends them over to the profiler to organize. After this happens, the AST is then deleted to free up memory space. The profiler has the ability to watch how many times statements are hit and marks them as **warm** and as these occurrences increase marks them as **hot**. Statements that are marked as hot are sent to the compiler ahead of time and are compiled at this point. From the profiler, the remaining (cold) instructions are sent over to the compiler that translates them from instructions to a lower-level language (binary form) that is readable to the machine and it is at this point that execution happens.

The JavaScript engine optimizes code through **function inlining.** This concept can be described as replacing a function call with a block of code performing the same action and by reducing function calls, the call stack is kept short improving the performance of code.