Conclusion And Future Work

8.1 The research contribution achieved

Considering all the constraints of embedded device, we designed and developed a lightweight interpreter for JavaScript Engine which takes very less static and runtime memory of the mobile browser. It executes the scripts and closure property of the JavaScript in predefined time with asynchronous manner and zero memory leaks.

8.2 Evaluation methodology conducted.

From the analysis it has been concluded that the static memory of object can be optimized by replacing all function pointers with a dummy function pointer to access all the properties of the objects. It has been observed that optimizing the AST node build logic of identifier, dot operator, function calls and if-then-else, will reduce the static memory allocation and improve the efficiency of interpreter. Interpreting the scripts with non-recursive stack algorithm proves that it can interpret all the complex structures of JavaScript syntax. Also it has been verified that the run time stack can be used for addressing the critical closure property algorithms without any memory leaks. These algorithms can be extended for other dynamic languages in future and can be ported in various feature phones and other embedded devices.

8.3 Main results and achievements

By using a dummy function pointer, we can save more than 13Kbytes of initial memory of built in objects, which will result in huge amount of reduction in runtime memory for JavaScript programs. After implementing AST node optimization for identifier, dot operator, function call and if then else it has been found that runtime memory size is reduced by 50% in each web site of Alexa site. The design of a non-recursive AST based stack algorithm has been designed to interpret the JavaScript in a predefined time period and in asynchronous manner. The structure of Runtime stack algorithm solves the closure property with zero memory leaks.
8.4 Limitations in current research and a clear plan for future work

Looking at the memory constraint in the embedded device many times we compromise the speed of execution. To improve the performance of the script execution it is required to implement a cache memory and to implement its algorithm for looking up the identifier. To avoid the repeated execution of same AST node inside the for loop, in future we need to introduce Just-in-Time compiler. Also it has been observed that many functions are defined and not called and those function may contains more than one inner function. To generate the AST node of such functions will take considerable time and wastage of memory. To avoid such issues in future during the build of AST node of function, we will write algorithm to avoid generating the AST node of inner function.