Abstract

Considering the limitation of memory, time and battery power in the embedded devices and various challenges to execute JavaScript in a mobile browser, this thesis aims to provide an optimized JavaScript engine, a light weight interpreter for a feature phone. In object oriented programming languages the methods are resolved during runtime with the help of function pointers. As function pointers consume huge amount of static memory, we attempt to eliminate those function pointers in order to reduce the memory requirement drastically. Programming languages are built with syntax. To reduce the code size, we optimize the syntax of AST node like identifier, dot operators, if-then-else and function call. The script should be interpreted at a predefined time within the constrained memory. Though a recursive interpreter is simpler to implement, it involves lot of memory space and the execution may be non-deterministic. We define an algorithm for the interpreter, which executes the script in non-recursive manner using a finite state machine and our own execution stack. In our work, we define different states and the mechanism for each JavaScript expression to achieve efficient execution. During the execution of script, the closure behavior of JavaScript causes memory leaks in various web browsers. We design a runtime stack algorithm and reference counter to resolve this issue.

The major contributions of the thesis are: to provide alternate solutions to reduce the static memory by optimizing the object behavior and optimizing the AST node of script, design of a non-recursive AST based stack algorithm to interpret the JavaScript in a predefined time period with asynchronous manner and runtime stack algorithm to handle closure property of JavaScript.