

**SCALESCRIPT - A SCALA (SUBSET) TO JAVASCRIPT COMPILER**, [T. Griswold](#)<sup>1</sup>, M. Stees<sup>2</sup>, B. Howard<sup>\*3</sup>, Willamette University<sup>1</sup>, Salem, OR 97301, Monmouth College<sup>2</sup>, Monmouth, IL 61462, DePauw University<sup>3</sup>, Department of Computer Science, Greencastle, IN 46135, [bhoward@depauw.edu](mailto:bhoward@depauw.edu)

ScalEScript is a compiler from a subset of Scala, a hybrid functional and object-oriented programming language, to JavaScript, a widely available scripting language. The goal of this project was to create an environment to facilitate the teaching of functional programming in Scala. In particular, we wanted to create a website that a student could open on nearly any computer or mobile device, type in some valid Scala code, and be able to execute it without installing any software. For this to work, we needed to translate the code entered by the student into something that could be run in the browser, specifically a client side scripting language that would be supported on most platforms. Therefore, we decided that we would create a Scala to JavaScript compiler, which could be executed through any JavaScript compatible browser; being written in our subset of Scala, ultimately it should be able to compile itself to produce a compiler running entirely within the browser.

The first challenge we faced was deciding on a reasonable subset of Scala to support in our compiler. In choosing our subset, we examined several years of Professor Howard's class materials that were used in teaching the introductory functional language course, finding all of the language constructs that were used in example code and assignments. Some of the key constructs in our subset include variables, functions, classes, blocks, standard collections (map, list, array, set), iterative constructs (for, while, and do-while), branching statements (if and if-else), and pattern matching. Some of the features that we chose to leave out of the subset were advanced library-design features, constructs related to Java compatibility, and type inference. Another difficulty that we faced was finding a suitable translation between our subset and JavaScript code. JavaScript is a loosely typed programming language and its object-orientation is based on prototypes instead of classes, so a one-to-one translation for some of the Scala constructs was not possible.

Although we accomplished a great deal over the summer, we were not able to complete our goal in its entirety. We developed a compiler that consists of four stages, namely the Parser which takes in Scala code and builds an abstract syntax tree (AST), ASTConverter which annotates the AST with additional information, TypeVerifier which performs type checking, and CodeGenerator which generates JavaScript code. Our compiler currently supports much of the desired subset with the exception of pattern matching, for loops, exception handling, and a few other features. Future work on this project could include implementing the rest of the subset and generating the JavaScript version of the compiler, developing a website with a simple interface to enter Scala code to be compiled and executed, and the addition of multimedia support in the form of graphics and animation.

This project made possible by NSF-REU grant No. CNS-1156893