Department of Computer Science
CS 298/299 Proposal
Summer 1822

# JestScript - A Whimsical Dive into Humorous Programming

John von Neumann

**Advisor**
Dr. Donald Knuth, sample@sjsu.edu

**Committee Members**
Dr. Edsger W. Dijkstra, sample@sjsu.edu
Dr. Claude Shannon, sample@sjsu.edu

## Introduction

In the realm of code seriousness, JestScript emerges as a playful endeavor, infusing the traditionally stoic world of programming with humor. Motivated by the desire to bring joy to coding, JestScript presents a unique take on syntax, error messages, and debugging experiences.

**Keywords:** *programming language, enjoyable coding, programming language design, user experience, light-weight programming language*

## Problem Definition and Motivation

The motivation behind JestScript lies in challenging the stereotypical perception of coding as a dry and serious task. With a plethora of programming languages prioritizing functionality over fun, JestScript aims to provide developers with a light-hearted alternative, fostering a more enjoyable coding atmosphere.

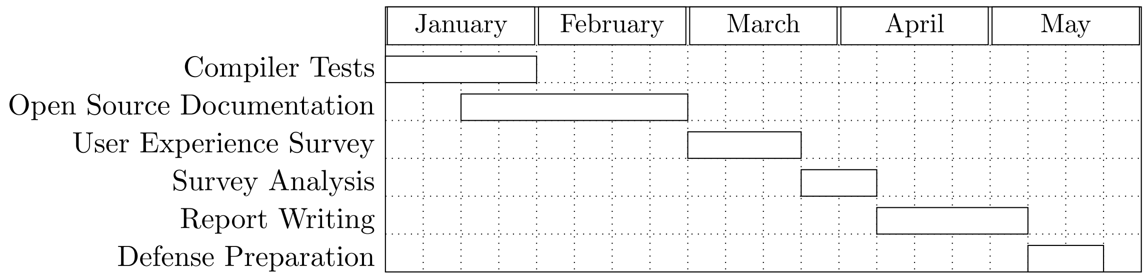
## Results Achieved in CS 297

- JestScript Compiler: The JestScript compiler that translates JestScript code into executable instructions for the target platform
- JestScript Syntax Guide: A detailed guide documenting the JestScript syntax, language rules, and conventions for developers.

## Expected Deliverables in CS 298/299

- Open-Source Repository: The public repository on a platform like GitHub containing the JestScript source code, documentation, and issue tracker.
- User Feedback Analysis Report: A report summarizing user feedback from testing phases, outlining areas of improvement and potential future enhancements.

## Timeline and Milestones for CS 298/299



## References

[1] C. A. R. Hoare, “An axiomatic basis for computer programming,” *Communications of the ACM*, vol. 12, no. 10, pp. 576–580, 1969.

[2] P. J. Landin, “The next 700 programming languages,” *Communications of the ACM*, vol. 9, pp. 157–166, Mar. 1966.

[3] R. Milner, “A theory of type polymorphism in programming,” *Journal of Computer and System Sciences*, vol. 17, pp. 348–375, Aug. 1978.

[4] G. Plotkin, “Call-by-name, call-by-value, and the $\lambda$-calculus,” *Theoretical Computer Science*, vol. 1, pp. 125–159, 1975.

[5] J. C. Reynolds, “Towards a theory of type structure,” in *Colloque sur la Programmation, Paris, France*, vol. 19 of *Lecture Notes in Computer Science*, pp. 408–425, Springer-Verlag, 1974.

[6] C. A. R. Hoare, “An axiomatic basis for computer programming,” *Communications of the ACM*, vol. 12, pp. 576–580 and 583, October 1969.

[7] G. Morrisett, D. Walker, K. Crary, and N. Glew, “From System-F to typed assembly language,” *ACM Transactions on Programming Languages and Systems*, vol. 21, pp. 527–568, May 1999.

[8] L. Cardelli, “A semantics of multiple inheritance,” in *Semantics of Data Types* (G. Kahn, D. MacQueen, and G. Plotkin, eds.), vol. 173 of *Lecture Notes in Computer Science*, pp. 51–67, Springer-Verlag, 1984. Full version in *Information and Computation*, 76(2/3):138–164, 1988.

[9] L. Damas and R. Milner, “Principal type schemes for functional programs,” in *ACM Symposium on Principles of Programming Languages (POPL), Albuquerque, New Mexico*, pp. 207–212, 1982.

[10] W. A. Howard, “The formulas-as-types notion of construction,” in *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism* (J. P. Seldin and J. R. Hindley, eds.), pp. 479–490, Academic Press, 1980. Reprint of 1969 article.

[11] P. J. Landin, “The mechanical evaluation of expressions,” *Computer Journal*, vol. 6, pp. 308–320, Jan. 1964.

[12] E. Moggi, “Computational lambda-calculus and monads,” in *IEEE Symposium on Logic in Computer Science (LICS), Asilomar, California*, pp. 14–23, June 1989. Full version, titled *Notions of Computation and Monads*, in Information and Computation, 93(1), pp. 55–92, 1991.