Justin Raymond

80 Home Ave, Apt 1
Middletown CT, 06459
978-621-7567
jraymond@wesleyan.edu
jrraymond.github.io

Education

2015–2016	**M.A. in Computer Science**, *Wesleyan University*, Middletown, CT, GPA 4.11. Research static complexity analysis of functional languages.
2010–2014	**B.A. in Computer Science, Biology**, *Wesleyan University*, Middletown, CT, GPA 3.74. Computer Science courses: Design of Programming Languages, Computer Checked Programs and Proofs, Automated Theorem Proving, Automata Theory and Formal Languages, Algorithms and Complexity, Data Structures, Computer Graphics, Computer Structure and Organization, Genomics and Bioinformatics, Evolutionary Bioinformatics, Tutorial in Type Theory.

Work Experience

2014–present	**Co-founder, programmer**, Joomah. Develop web application using Python/Django. Joomah is a startup which connects employers with jobseekers in Africa.
2013–2014, 2015–present	**Teacher's Assistant**, *Computer Science Department*, Wesleyan University. Ran help sessions for computer science students. Required knowledge of Python, Java, C0 and SML.
Sep 2015–Dec 2015	**Peer Tutor**, *Computer Science Department*, Wesleyan University. Tutor students taking Design of Programming Languages. Required use of OCaml.
Aug 2013 – Aug 2014	**Programmer**, *Instructional Media Services*, Wesleyan University. Worked on cmdr (wesleyan.github.io/cmdr/), an open source audio/visual control system. Required use of Ruby, CouchDB, Javascript, and HTML5/CSS3.
Mar 2013 – Dec 2014	**Project Leader**, *Wesleyan University*. Led a student forum on mobile app development. Developed an Andriod and iOS app using Apache Cordova (github.com/WesAppGroup).

Skills

Languages	Haskell, OCaml, SML, Agda, C, C++, Python, Javascript, Java, HTML, CSS, Ruby
Frameworks	Django, Flask, Yesod, Servant, Ruby on Rails

Operating Systems	Linux, OSX
Version control	git

Projects

otto	otto is an auto-grading library in OCaml. otto is extremely robust. Tests are run in a seperate process. Code that crashes the process or loops infinitely without allocating will not hang the auto-grader. Source code: github.com/jrraymond/otto
MAAX	An NES AI written in Haskell. MAAX is an implementation of the NeuroEvolution of Augmenting Topologies (NEAT) algorithm. Source: github.com/mdietz94/MAAX
ray-tracer	A program that generates photorealistic images. Features include antialiasing, soft-shadows, reflections, refractions, depth of field, glossy reflections, imports wavefront scene description files, and animated gifs. Distributed using Cloud Haskell. Source and example images: github.com/jrraymond/ray-tracer
Units as Types	Compile time checking of units of measure in Agda. Source: https://github.com/jrraymond/UnitsAsTypes
Godel's T	An implementation of Godel's T in Agda. Godels' T combines function types with natural numbers. Iteration is provided via primitive recursion. Consequently only total function may be defined. Includes proofs of progress (a well-typed term is either a value or takes a step of evaluation) and preservation (if a well-type term takes a step of evaluation then its type is preserved). Source: https://github.com/jrraymond/GodelsT
GHC	Contributions to Data.List and Data.Vector documentation.