Dynamic languages such as JavaScript are more difficult to com pile than statically typed ones. Since no concrete type information is available, traditional compilers need to emit generic code that can handle all possible type combinations at runtime. We present an al ternative compilation technique for dynamically-typed languages that identifies frequently executed loop traces at run-time and then generates machine code on the fly that is specialized for the ac tual dynamic types occurring on each path through the loop. Our method provides cheap inter-procedural type specialization, and an elegant and efficient way of incrementally compiling lazily discov ered alternative paths through nested loops. We have implemented a dynamic compiler for JavaScript based on our technique and we have measured speedups of 10x and more for certain benchmark programs.

We present a trace-based compilation technique for dynamic languages that reconciles speed of compilation with excellent per formance of the generated machine code. Our system uses a mixed mode execution approach: the system starts running JavaScript in a fast-starting bytecode interpreter. As the program runs, the system identifies hot (frequently executed) bytecode sequences, records them, and compiles them to fast native code. We call such a se quence of instructions a trace.