**Stack Trace Limit**

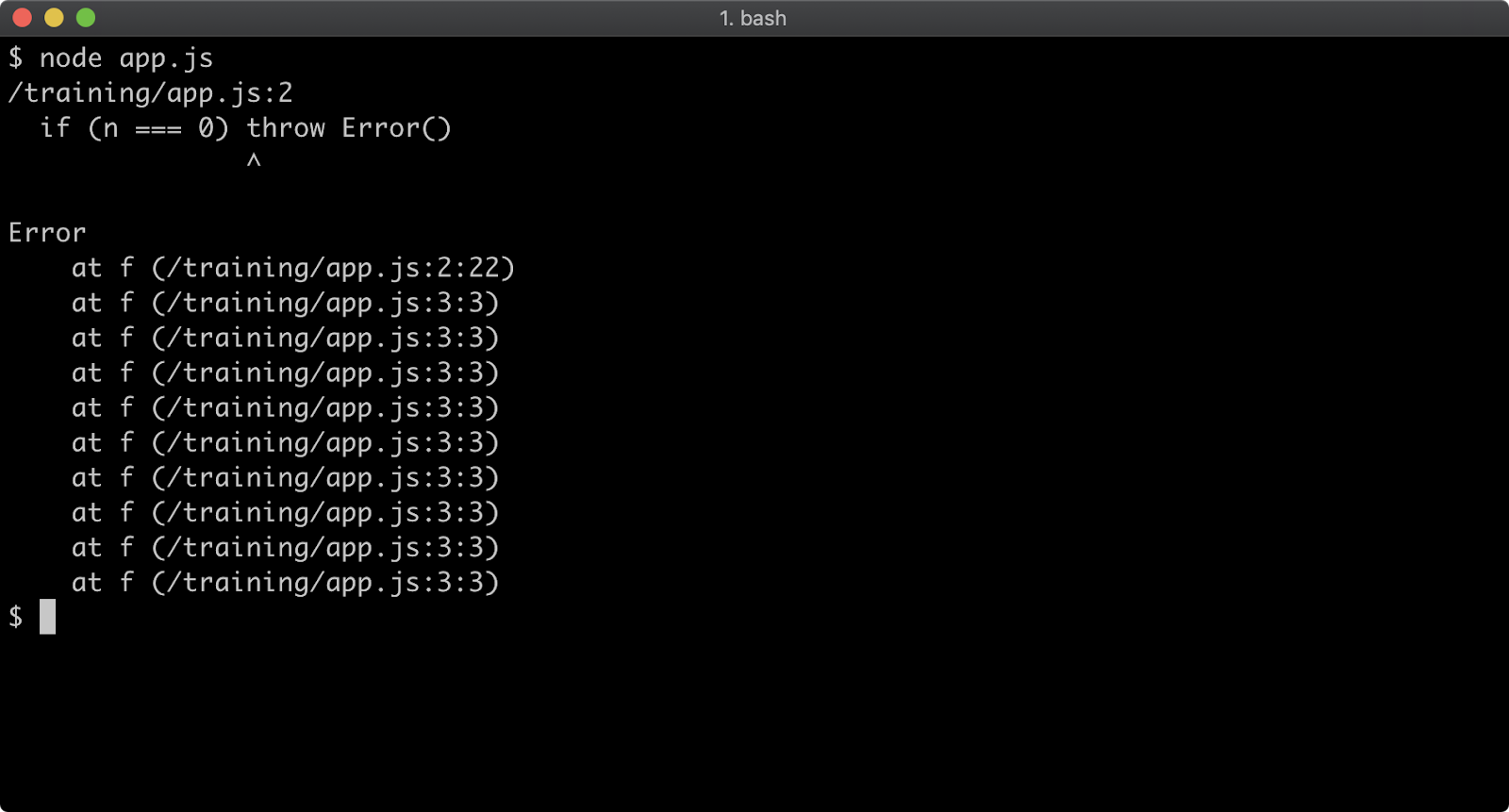
Stack traces are generated for any **Error** that occurs, so they're usually the first point of call when debugging a failure scenario. By default, a stack trace will contain the last ten stack frames (function call sites) at the point where the trace occurred. This is often fine, because the part of the stack you are interested in is often the last 3 or 4 call frames. However there are scenarios where seeing more call frames in a stack trace makes sense, like checking that the application flow through various functions is as expected.

The stack trace limit can be modified with the **--stack-trace-limit** flag. This flag is part of the JavaScript runtime engine, V8, and can be found in the output of the **--v8-options** flag.

Consider a program named **app.js** containing the following code:

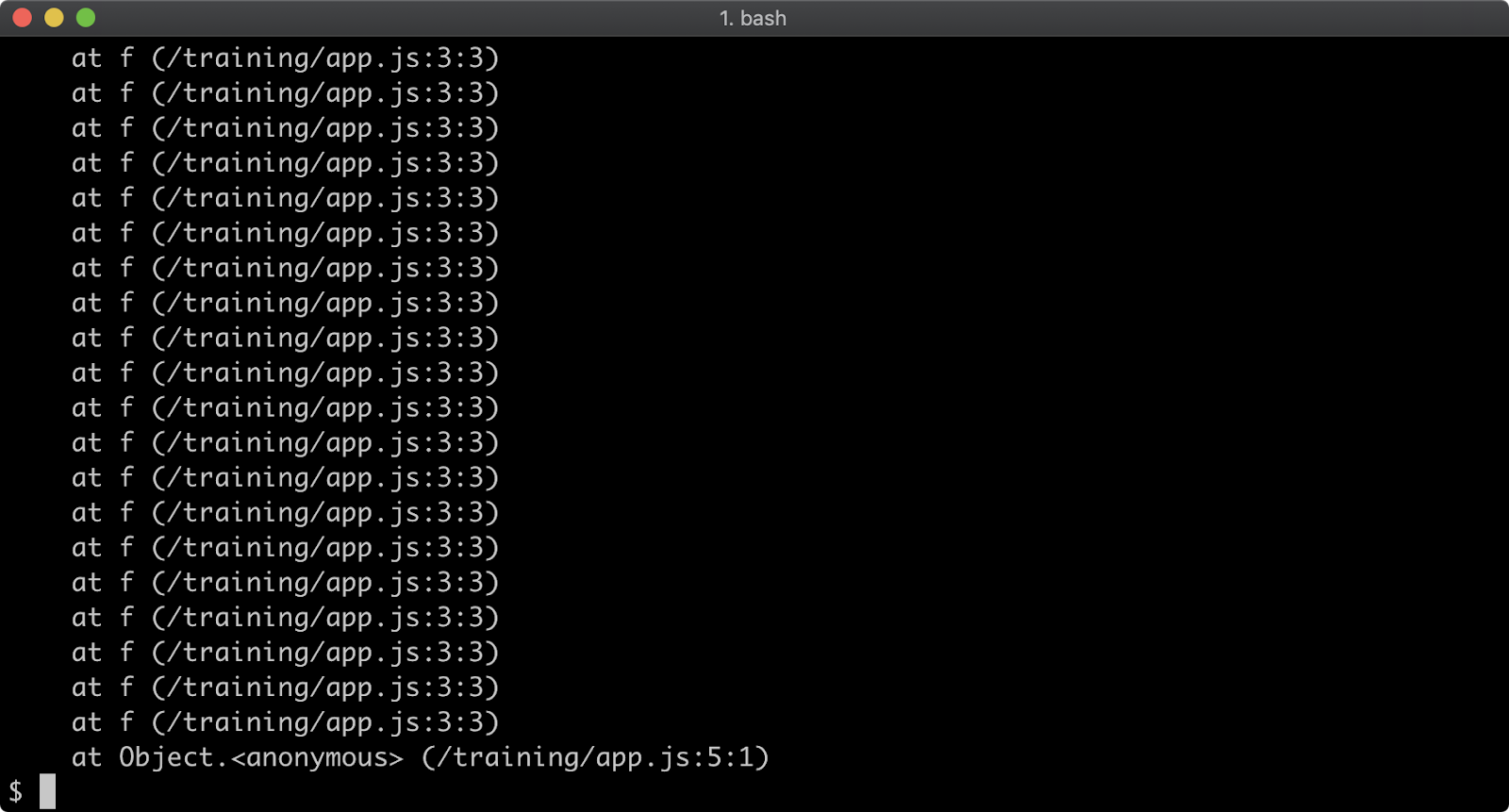
**function f (n = 99) {  
  if (n === 0) throw Error()  
  f(n - 1)  
}  
f()**

When executed, the function **f** will be called 100 times. On the 100th time, an **Error** is thrown and stack for the error will be output to the console.



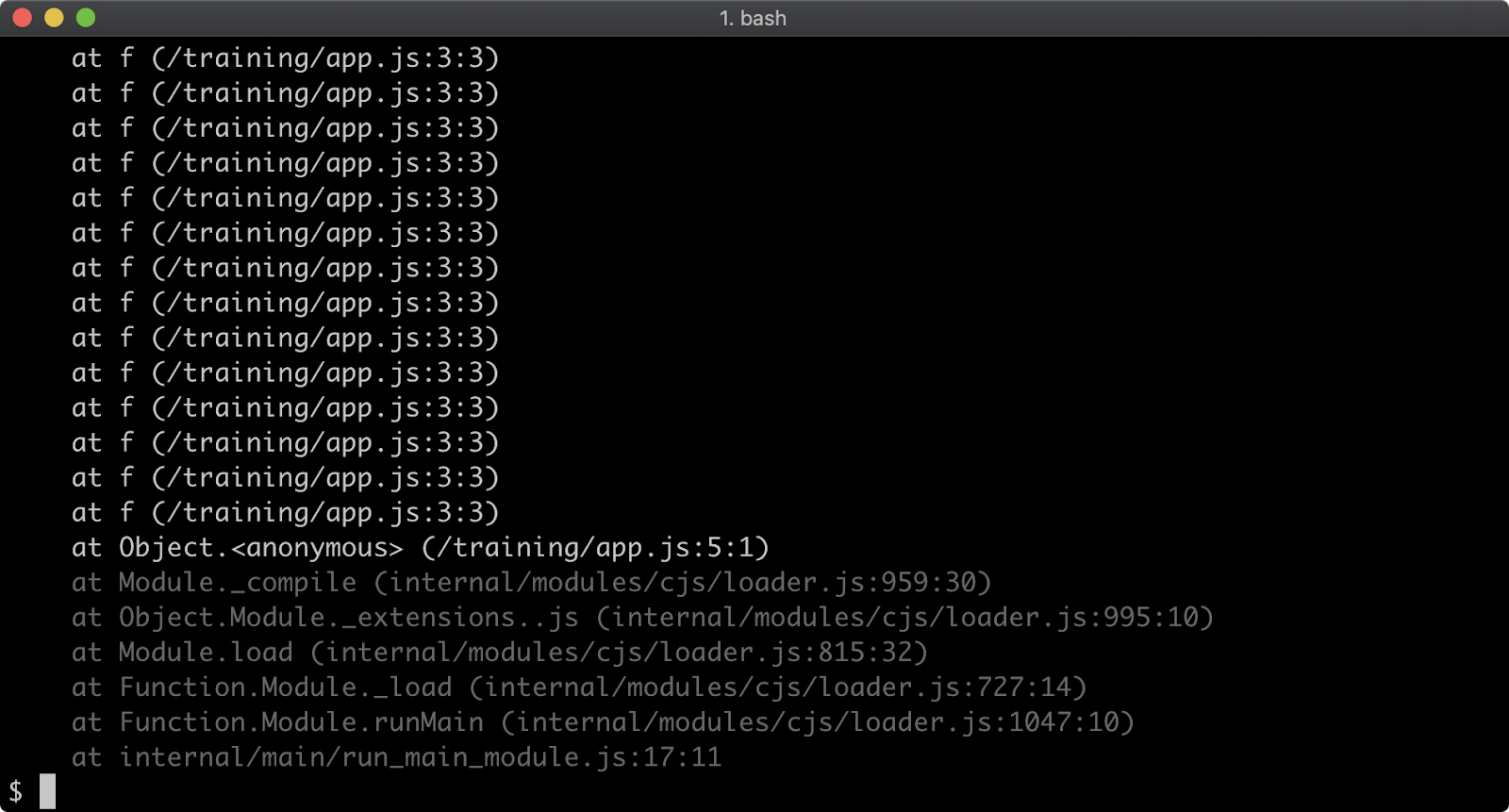
The stack trace output only shows the call to the **f** function, in order to see the very first call to **f** the stack trace limit must be set to 101. This can be achieved with the following:

**node --stack-trace-limit=101 app.js**



Setting stack trace limit to a number higher than the amount of call frames in the stack guarantees that the entire stack will be output:

**node --stack-trace-limit=99999 app.js**



Generally, the stack trace limit should stay at the default in production scenarios due to the overhead involved with retaining long stacks. It can nevertheless be useful for development purposes.