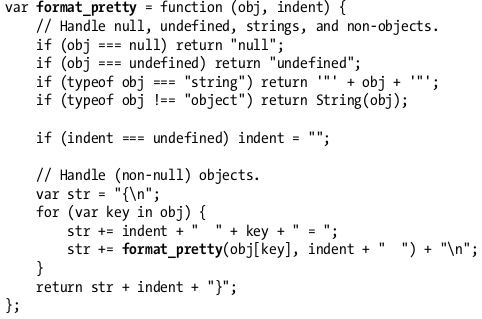
**Chapter 10 Extracting Unrelated Subproblems**

* Before writing the code, think of your high-level goal.
* Is it working directly to the goal?
* If enough lines are solving an unrelated subproblem, extract the code separate function.
* If your function does a lot of problems that are unrelated, separate the unrelated code so the high-level code will be more readable and the reader can focus on the high-level goal. And the unrelated code can be reused later.
* One benefit of extracting unrelated problem is it makes calling code simpler.



* General-purpose is great because it’s completely decoupled from the rest of the project. Serves as user library.
* The more project you can isolate libraries, the better, because the rest of the code will be smaller and easier to think about.
* Top-down programming is a style where the highest-level modules and functions are designed first and lower-level functions are implemented as needed to support them.
* Bottom-up programming tries to anticipate and solve all subproblems first and then build the higher-level components using pieces.
* If an interface isn’t clean, you can make your own “wrapper” functions.
* Never settle for an interface that’s less than ideal.
* On the other hand, small piece of functions may hurt the readability since the reader has to get track of functions.
* So you should only extract codes if necessary.