The Beauty of Simplicity

Jeremy Gibbons

(published as [2])

As an admirer of the "artistic flare, nuanced style, and technical prowess that separates good code from great code" explored by Robert Green and Henry Ledgard [1], I was disappointed by the authors' emphasis on "alignment, naming, use of white space, use of context, syntax highlighting, and IDE choice." As effective as these aspects of beautiful code may be, they are at best only skin deep.

Beauty may indeed be in the eye of the beholder, but there is a more compelling beauty in the deeper semantic properties of code than layout and naming. I also include judicious use of abstraction, deftly balancing precision and generality; elegant structuring of class hierarchies, carefully trading between breadth and depth; artful ordering of parameter lists, neatly supporting common cases of partial application; and efficient reuse of library code, leveraging existing definitions with minimum effort. These are subjective characteristics, beyond the reach of objective scientific analysis—matters of taste not of fact—so represent aspects of the art rather than the science of software.

Formalizing such semantic properties is more difficult than establishing uniform coding conventions; we programmers spend our professional lifetimes honing our writing skills, not unlike novelists and journalists. Indeed, the great American essayist Ralph Waldo Emerson (1803–1882) anticipated the art in the science of software like this: "We ascribe beauty to that which is simple; which has no superfluous parts; which exactly answers its end; which stands related to all things; which is the mean of many extremes." It is to this standard I aspire.

References

[1] Robert Green and Henry Ledgard, "Coding Guidelines: Finding the Art in the Science". *Communications of the ACM*, 54(12):57–63, December 2011. DOI 10.1145/2043174.2043191.

[2] Jeremy Gibbons, "The Beauty of Simplicity". *Communications of the ACM*, 55(4):6, April 2012. DOI 10.1145/2133806.2133808.