
Preface

The best way to become acquainted with a subject is to write a book about it.

—Benjamin Disraeli

Each problem that I solved became a rule, which served afterwards to solve other problems.

—René Descartes

What is the Purpose of this Book?

There are many students and professionals in science and engineering, other than those specifically interested in fields such as computer science or computer engineering, who need to know how to solve computational problems on computers. There are basically two approaches to meeting the needs of such people. One is to rely on software applications such as spreadsheets, using built-in functions and perhaps user-defined macros, without requiring any explicit understanding of the principles on which programming languages are based.

A second approach is to learn a “traditional” programming language, for previous generations Fortran or Pascal, and more recently C, C++, or Java. These languages are important for certain kinds of work, but they may be viewed, possibly with good reason, as irrelevant by many students and professionals.

From a student’s point of view, there is no painless solution to this dilemma, but in this book I assume that learning to solve computational problems in an online environment using HTML and PHP will at least appear to be a more relevant approach. HTML, **H**yper **T**ext **M**arkup **L**anguage is universally used as the foundation for online applications. HTML documents can be used as “data input forms” for PHP—originated by Rasmus Lerdorf in the mid 1990s as “Personal Home Page Tools,” but long since expanded as a comprehensive programming language.

What separates PHP from an online development language such as JavaScript is its support for accessing externally stored data files, which greatly extends the range of science and engineering problems that can be addressed.

In some ways, an HTML/PHP environment is more difficult to learn than traditional and more “mature” (some might prefer “obsolete”) text-based programming languages. C, for example, is a fairly small language with an unambiguous set of syntax rules and a primitive text-based input/output interface. You can view the limitations of C as either a blessing or a curse, depending on your needs. A major advantage of C is that programs written in ANSI Standard C¹ should work equally well on any computer that has a C compiler, making the language inherently platform-independent.

HTML, on the other hand, is an immature and developing programming languages (if we can agree to call HTML a “programming language”) that functions within a constantly changing Web environment. It lacks a uniformly accepted set of syntax rules. There are dialects of HTML that will work only on particular computing platforms and the possibility exists for language “extensions” that may be even more platform-dependent. PHP is still an evolving language whose standards are set and maintained by a global user group—essentially by volunteers—but it adheres to broadly understood programming language concepts and has well-defined syntax rules.

Fortunately, it is possible to work with some core subsets of HTML which, along with PHP, can be used to solve some of the same kinds of computational problems that would be appropriate for a more traditional “scientific” programming language such as C or C++. My motivation for writing this book and its predecessors was to learn how to use HTML and PHP to create my own online applications, and I now use this environment for many tasks that I previously would have undertaken in Fortran or C. Based on my own experience I have concluded that, although it might not be accurate to define PHP as a “scientific” computing language, it is nonetheless entirely reasonable to use HTML/PHP as a framework for learning basic programming skills and creating a wide range of useful and robust science and engineering applications.

Although this book is intended for “scientists and engineers,” as suggested by its title, the content is not technically complex. The examples and exercises do not require extensive science, engineering, or mathematics background and only rarely is mathematics beyond basic algebra needed. So, I believe this book could serve as a beginning programming text for undergraduates and even for high school students.

¹ANSI = American National Standards Institute, a voluntary standardization system in the United States.

Learning by Example

It is well known that people learn new skills in different ways. Personally, I learn best by having a specific goal and then studying examples that are related to that goal. Once I understand those examples, I can incorporate them into my own work. I have used that learning model in this book, which contains many complete examples that can serve as starting points for your own work. (See the second quotation at the beginning of this preface).

This model works particularly well in an online environment. The amount of online information about HTML and PHP, including code samples, is so vast that it is tempting to conclude that nobody writes original code anymore. If you have trouble “learning by example,” you will have trouble learning these languages, not just from this book, but in general because that is how most of the available information is presented.

It is an inescapable fact that a great deal of the source code behind Web pages involves nothing more (or less) than creative cutting, pasting, and tweaking of existing code. Aside from the issues of plagiarism and intellectual dishonesty that must be dealt with in an academic environment, there is also the practical matter of an effective learning strategy. You cannot learn to solve your own computational problems just by trying to paste together someone else’s work. (Believe me, I’ve tried!) Until you develop your own independent skills, you will constantly be frustrated because you will never find *exactly* what you need to copy and you will be unable to synthesize what you need from what is available.

So, while you should expect to find yourself constantly recycling your own code based on what you learn from this book, you need to make sure that you really *learn how to use* these languages and don’t just *learn to copy*!

If you are reading this book, you almost certainly are not and do not aspire to be a professional programmer. For a casual programmer from a scientific or technical background, it can be very time consuming to cut through the clutter of online information about these languages when the applications are not directly applicable to the needs of scientists and engineers. In my own work, what I need over and over again is some sample code that will jog my memory about how to approach recurring programming problems—how to pass information from an HTML document to a PHP application, how to extract information from a data file, how to display data-based graphics, etc. Throughout the book, I have tried to give examples that serve this need, including an entire chapter devoted to PHP graphics.

The Origin and Uses of this Book

In 2007, Springer published my book, *An Introduction to HTML and JavaScript for Scientists and Engineers*. This was followed in 2008 by *An Introduction to PHP for Scientists and Engineers: Beyond JavaScript* and, in 2011, by *Guide to HTML*,

JavaScript and PHP. Those books followed the sequence in which I learned to use HTML, JavaScript, and PHP in my own work. (See the first quotation at the beginning of this preface.) Although I still use JavaScript for some applications, I now rely mostly on an HTML/PHP environment in which an HTML document serves as the input interface to a separate PHP application that performs the required calculations and, as appropriate, generates graphics.

This book easily provides enough material for a one- or two-semester introductory programming course for science and engineering students because the possibilities for PHP-based applications are limitless. Because of the book's very specific focus on science and engineering applications, I believe the book is also particularly well suited for developing a working knowledge of HTML and PHP on your own if you are a student or professional in any technical field.

Acknowledgements

I am indebted to Wayne Wheeler, Senior Editor for Computer Science at Springer, and Simon Rees, Associate Editor for Computer Science at Springer for encouraging and supporting this project.

Eagleville, USA

David R. Brooks



<http://www.springer.com/978-3-319-56972-7>

Programming in HTML and PHP
Coding for Scientists and Engineers
Brooks, D.R.
2017, XI, 293 p. 2 illus., Softcover
ISBN: 978-3-319-56972-7