

# 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

---

## i

### 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 programming language, previously Fortran or Pascal, and more recently C, C++, or Java. These languages are important for certain kinds of work, such as computer science or scientific computing, 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**,<sup>1</sup> **JavaScript**, and **PHP** will at least appear to be a more relevant approach. HTML and JavaScript are universally used for developing self-contained online applications. The use of PHP for accessing externally stored data files, a capability that is not available directly through JavaScript, greatly extends the range of science and engineering problems that can be addressed. A working knowledge of these programming languages is a valuable skill for any scientist or engineer. The fact that these are Web-based languages may make such skills more immediately marketable than comparable skills developed with older text-based languages.

---

<sup>1</sup> See Glossary for definitions of terms appearing in bold font.

In some ways, the HTML/JavaScript/PHP environment is more difficult to learn than a traditional text-based programming language such as C. C is a mature (some might prefer “obsolete”), 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 C** should work equally well on any computer that has a C compiler, making the language inherently **platform-independent**.

HTML, JavaScript, and PHP, on the other hand, are immature and unstable programming languages (if we can agree to call HTML a “language”) that function within a constantly changing Web environment. There are dialects of HTML and JavaScript that will work only on particular computing platforms and the possibility exists for language “extensions” that may be even more platform-dependent. PHP is more platform-independent, but it is still an evolving language whose standards are set and maintained by a user group – essentially by volunteers. While it is true that there are extensions to languages such as C and other older languages that are platform-dependent, the platform dependence of languages used in the online environment is a major implementation issue rather than an occasional minor inconvenience.

As one indication of the teaching and learning challenges these environments provide, just three popular paperback HTML and JavaScript reference books occupy nearly 6 in. of space (15 cm in deference to a metric audience) on my office bookshelf! A great deal of the material in those books is devoted to explaining the often subtle differences among various versions of HTML and JavaScript.

Fortunately, it is possible to work with some core subsets of HTML and JavaScript which, with PHP, can be used to solve some of the same kinds of computational problems that would be appropriate for a more traditional language such as C or C++. My initial motivation for writing this book was to learn how to use HTML, JavaScript, and PHP to create my own online applications, and I now use this environment for many tasks that I previously would have undertaken in C. Based on this experience, I have concluded that, despite the fact that these languages cannot fairly be defined as “scientific computing” languages, it is nonetheless entirely reasonable to use them to learn basic programming skills, and to create 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.

---

## ii

### 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 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, JavaScript, and PHP, including code samples, is so vast that it is only a slight exaggeration to state 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 select items from a pull-down list, how to extract information from a data file, how to pass information from an HTML document to a PHP application, 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.

---

### iii

## The Origin and Uses of This Book

In 2007, Springer published *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*. Those two 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.) When the time came to consider a second edition of the *HTML and JavaScript* book, it seemed a better idea to undertake a rewrite that would combine both books into a single volume. This book is the result. I have, hopefully, clarified some of the explanations. There are more examples and exercises and I have added some new material that my students and I have found useful, including a brief introduction to using “pseudo-code” as an approach to organizing solutions to computing problems (see Appendix 4).

I have used both of the original books as texts in an 11-week (one quarter) introductory programming course for biomedical engineering graduate students at Drexel University. I found that a course restricted just to HTML and JavaScript is a little “thin” for this audience. Adding a brief introduction to PHP solves the problem. This book easily provides enough material for a one-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, JavaScript, and PHP on your own if you are a student or professional in any technical field.

---

**iv****Acknowledgments**

I am indebted to several classes of graduate students from Drexel University’s School of Biomedical Engineering, Science & Health Systems, who have provided feedback on the material in this book and its predecessors. I am also once again indebted to my wife, Susan Caughlan, for her patient and always helpful proofreading of my manuscripts.

Institute for Earth Science Research and Education

David R. Brooks





<http://www.springer.com/978-0-85729-448-7>

Guide to HTML, JavaScript and PHP  
For Scientists and Engineers

Brooks, D.R.

2011, XIII, 415 p., Hardcover

ISBN: 978-0-85729-448-7