Introduction to

JavaScript Programming

The “Nothing but a Browser” Approach

Eric S. Roberts
Stanford University and Reed College
To Lauren, with love
To the Student

Welcome! By picking up this book, you have taken a step into the world of computer science. Twenty-five centuries ago, the Chinese philosopher Lao-tzu observed that the longest journey begins with a single step. This book can be your beginning.

Computing is a profoundly empowering technology. Jobs are plentiful, salaries are high, and there are far too few people with the necessary skills to fill those jobs. Computation, moreover, is transforming almost every discipline, not just in science and engineering but also across the entire university curriculum. No matter what field of study you choose, understanding how to use computing effectively will be of enormous value.

To get you started on your journey into the wonders of computer science, this book teaches you how to write programs in a language called JavaScript, which is a central technology for web-based applications. The programs you create as you go through this book will run in any web browser, including the browser on your phone. More importantly, however, the book will teach you the fundamental principles of programming. While it uses JavaScript to illustrate these principles, they will carry over into any other language that you learn.

As with any skill that is worth knowing, programming will not necessarily come easily. Many students find computers overwhelming and imagine that computer science is beyond their reach. Learning the basics of programming, however, does not require advanced mathematics or a detailed understanding of electronics. What matters is whether you can progress from the statement of a problem to its solution.

To do so, you must be able to think logically and to express your logic in a form that the computer can understand. Perhaps most importantly, you must be able to see the task through to its completion without getting discouraged by difficulties and setbacks. If you stick with the process, you will discover that reaching the solution is so exhilarating that it more than makes up for any frustrations along the way.

I wish you an exciting and empowering journey.

Eric Roberts
Stanford University
January 2019
To the Instructor

This book is designed for use in the first programming course in a typical college or university curriculum. It covers the material in a traditional first course in computer science, generically referred to as CS1. The book assumes no prior programming experience and is appropriate both for prospective computer science majors and for students in other disciplines who are interested in learning the fundamentals of programming.

The book uses a programming language called JavaScript, which is one of the most widely used languages in industry and has become the standard language for writing interactive web applications. Because of its popularity, JavaScript is built into every major web browser, which means that any device with a browser can run JavaScript programs without any additional software. The focus of this text, however, is on the fundamental concepts of programming rather than the language. It does not cover all of JavaScript and avoids several aspects of the language that students are likely to misuse. The subset of JavaScript employed in this book provides students with more than enough power to write exciting programs that run in any web browser.

The case for using JavaScript in CS1

As a discipline, computing is always changing. These changes can be profoundly exciting, particularly for students who have grown up in the digital world. For educators, the breakneck pace of development creates challenges that invariably extend even into the earliest levels of the curriculum. As languages, tools, and paradigms shift, teaching strategies must evolve to remain current.

Two decades ago, the Java programming language, with its support for applets, looked like the wave of the future and seemed likely to become the standard for web-based programming. That didn’t happen. By the early 2000s, Java applets had been abandoned in favor of the ubiquitous blend of JavaScript, HTML, and CSS that serves as the foundation for web applications today.

Once it was clear that JavaScript had triumphed in the race to become the language of the web, several of my colleagues and I began to think about using it in Stanford’s introductory programming course. Much of our original motivation for choosing JavaScript came from our excitement about a “Nothing but a Browser” model in which students—even those with no access to computers other than a smart phone—could work entirely in the web environment. When we began teaching the JavaScript version of our introductory course, we found that the browser-based model worked exactly as we envisioned. Students no longer have to install and use a separate development environment. All they need is a web
browser, which every student has. They are also able to add JavaScript content to their own web pages, which increases the students’ incentive to learn the material and encourages them to show off their work.

Beyond the advantages of the web-based model, we were delighted to discover just how well JavaScript works as a teaching language. Students can master the essentials of JavaScript much more quickly than their predecessors learned Java. As Douglas Crockford, who developed the widely used JavaScript Object Notation described in Chapter 9, observes, JavaScript is “a beautiful, elegant, highly expressive language,” even though it is often used in the context of other web technologies that share little of that elegance. If you focus on the JavaScript language itself, its beauty and elegance make teaching JavaScript much simpler than teaching other languages, with far fewer confusing details to explain.

The relative simplicity of JavaScript is reflected in the length of this textbook, which is 30 percent shorter than my Java book for the same course. Despite being shorter, the JavaScript version covers at least 25 percent more material. In part, the economy of presentation reflects the fact that JavaScript programs are shorter than the same programs written in Java. Of even greater importance, however, is the fact that the parts of those programs that disappear in the transition from Java to JavaScript are precisely the parts that would have required the most explanation.

**Pedagogical approach**

*Introduction to JavaScript Programming* relies on three proven strategies to maximize students’ understanding. First, it introduces the concepts of programming using a JavaScript version of Rich Pattis’s wonderful Karel the Robot microworld, which has gently welcomed Stanford students to the world of programming for almost 40 years. Although it is possible to skip the Karel chapter, we have found that the increased conceptual understanding students derive from that introduction more than repays the time. Second, the book adopts the programming guidelines recommended by Douglas Crockford in *JavaScript: The Good Parts*, which outlines how to write elegant, well-structured programs in JavaScript. Third, the book presents topics in an order that defers the most challenging topics until students have the necessary background. For example, the detailed discussion of how JavaScript works together with other web technologies like HTML and CSS appears in Chapter 12, after students have already completed chapters on data structures and inheritance.
Pedagogical features

The text uses the following features to enhance student learning:

• **Chapter openers.** Every chapter begins with a photograph and short biography of someone who has had a major impact on computing. These biographies, and the diverse backgrounds of the people included, emphasize the human side of computing.

• **Key terms.** Each new term in the text appears in boldface italic and is followed immediately by a definition.

• **Syntax boxes.** Each new syntactic form is summarized in a highlighted box that appears in the margin of that page.

• **Syntax coloring.** The major program examples appear in numbered figures that use syntax coloring to ensure that students can easily differentiate comments, keywords, and string constants from the rest of the code.

• **Source code.** The source code for all the sample programs is available on the web site for the book. The programs, moreover, all run in any browser.

• **Chapter summary.** Each chapter includes an extensive summary that lists the key ideas introduced in the chapter.

• **Review questions.** Each chapter summary is followed by review questions that provide a self-test of student understanding. Answers to the review questions appear on the web site.

• **Programming exercises.** Each chapter ends with an extensive set of exercises that test whether students understand the chapter material while giving them opportunities to create exciting applications.

Resources for the instructor

The following resources are available to instructors who adopt this textbook on the Pearson Instructor Resource Center (http://www.pearsonhighered.com/irc):

• **Guide to the instructor.** Teachers who adopt the text receive an instructor’s guide that includes additional examples, recommendations for projects and programming assignments, hints on pedagogical strategy, and suggestions for possible variations in the order of presentation.

• **Lecture slides.** The web-based repository for the book includes a set of slides in Microsoft® PowerPoint for each of the 12 chapters. These slides illustrate the key points in each chapter and include detailed animations of the program examples.

• **Solutions to the exercises.** Adopters have access to a repository containing solutions to the programming exercises.
**Resources for students**

The following resources are available at www.pearsonhighered.com/cs-resources:

- **Program sources.** The web site includes the source code for all sample programs presented in the text. The web pages for those programs also allow students to run the programs directly in the browser.

- **Program animations.** For several of the most important sample programs, the web site makes it possible for students to step through the execution of the program so they can see it in operation.

- **Answers to the review questions.** The review questions serve as a self-test that allows students to check their understanding. The answers to these review questions are available on the web site.

**System requirements**

The most important advantage of using JavaScript in an introductory course is that programs run in any browser that implements HTML 5 and ECMA 6—both of which are supported by all major browsers today. None of the JavaScript programs require features from more recent versions of JavaScript and in fact use only ECMA 6 extensions that are typically supported even in older browsers.

Although it is possible to use the interactive applications on the web site to write simple programs without using any other tools, many students will find it easier to use a more powerful text editor to write and edit their programs. Many excellent editors are freely available on the web, and we have found it best to let students choose an editor that fits their own needs rather than to impose a particular choice.
Acknowledgments

Many people contributed in different ways to the development of this book. I am especially indebted to Douglas Crockford, whose book *JavaScript: The Good Parts* proved that it is possible to write “beautiful, elegant, highly expressive” programs in JavaScript and showed his readers how to do so.

I also want to thank colleagues at Stanford who have helped make this book possible. I am profoundly grateful to Jerry Cain, with whom I taught a pilot version of a JavaScript-based introductory course in the spring of 2016-17. We learned a great deal from that exercise and have integrated those insights and understandings into the final version of this book. In addition to being a wonderful co-teacher, Jerry has been a tireless reader of this text through many drafts. His amazingly thorough comments have made it much stronger. I owe considerable thanks as well to my colleagues Chris Piech, Keith Schwarz, and Marty Stepp for their extensive contributions to the book. And I also need to thank several generations of section leaders and the students in the pilot versions of the course, all of whom have helped make it so exciting to teach this material.

I am grateful to Tracy Johnson, Marcia Horton, Erin Ault, Meghan Jacoby, and the other members of the team at Pearson for their support on this book as well as its predecessors over the years. In addition, I deeply appreciate the comments from reviewers Paul Fodor, Ian Utting, Kristine Christensen, and Zerksis Umrigar, who offered excellent feedback and suggestions.

As always, my greatest thanks are due to my wife Lauren Rusk, who has again worked her magic as my developmental editor. Lauren’s expertise has added considerable clarity and polish to the text. Without her, nothing would ever come out as well as it should.
## Contents

### 1 A Gentle Introduction 1
1.1 Introducing Karel 2  
1.2 Teaching Karel to solve problems 4  
1.3 Control statements 10  
1.4 Stepwise refinement 17  
1.5 Algorithms in Karel’s world 23  
  - Summary 24  
  - Review questions 26  
  - Exercises 26

### 2 Introducing JavaScript 39
2.1 Data and types 40  
2.2 Numeric data 41  
2.3 Variables 44  
2.4 Functions 49  
2.5 String data 53  
2.6 Running JavaScript in the browser 56  
2.7 Testing and debugging 61  
2.8 Software maintenance 69  
  - Summary 71  
  - Review questions 73  
  - Exercises 76

### 3 Control Statements 79
3.1 Boolean data 80  
3.2 The if statement 85  
3.3 The switch statement 88  
3.4 The while statement 90  
3.5 The for statement 94  
3.6 Algorithmic programming 100  
3.7 Avoiding fuzzy standards of truth 109  
  - Summary 109  
  - Review questions 110  
  - Exercises 112
### Arrays

- **8.1** Introduction to arrays 274
- **8.2** Array operations 281
- **8.3** Using arrays for tabulation 288
- **8.4** Reading text from files 292
- **8.5** Multidimensional arrays 294
- **8.6** Image processing 295
  - Summary 304
  - Review questions 305
  - Exercises 306

### Objects

- **9.1** Objects in JavaScript 314
- **9.2** Using objects as maps 317
- **9.3** Representing points 321
- **9.4** Rational numbers 330
- **9.5** Linking objects together 338
  - Summary 342
  - Review questions 342
  - Exercises 343

### Designing Data Structures

- **10.1** Abstract data types 350
- **10.2** Implementing a token scanner 351
- **10.3** Efficiency and representation 358
- **10.4** Representing real-world data 370
  - Summary 376
  - Review questions 377
  - Exercises 378

### Inheritance

- **11.1** Class hierarchies 388
- **11.2** Defining an employee hierarchy 391
- **11.3** Extending graphical classes 395
- **11.4** Decomposition and inheritance 399
- **11.5** Alternatives to inheritance 404
  - Summary 405
  - Review questions 406
  - Exercises 407