Introduction

In January 2002, *Advanced Web Sites Made Easy* was published. The book introduced the world to HTML/OS, an exciting Web technology that simplifies the construction of advanced Web sites. Readers learned how to build a two-line login page, a ten-line text editor, a two-line database report, a five-line shopping cart, and dozens of other functional Web site elements. The book debunked the systems-integration approach to building Web sites.

Readers learned that systems integration is fundamentally complex, because it's based on the idea of salvaging old technologies. In short, systems integration is about integrating PC-centric programming languages and database engines that were engineered years before the Web existed. While this may benefit PC software companies, the approach puts the entire Web industry in shackles. Pure Web technologies, such as HTML/OS and H_2O, deliver 250 times less complexity.

Using the HTML/OS macro language, readers of *Advanced Web Sites Made Easy* learned how to build e-commerce sites, database-driven sites, and highly functional Web sites. After the PC-based technologies were dropped, advanced Web construction became easy.

Today, thousands of Web sites in a wide variety of disciplines use and depend on the HTML/OS technology. E-commerce sites such as `www.thepowerstore.com`, sell vitamins. Sites such as `cwp.tamu.edu` monitor weather conditions for cotton farmers. There are big event sites such as the New York City Marathon (`www.nyrrc.org`), where tens of thousands of people can monitor the progress of friends and family members as they run in one of the world's most popular marathons. The Web site revolution created by HTML/OS is advancing fast. With the recent release of Aestiva H_2O, a version of HTML/OS that's freely available, the Web technology has become easier than ever to use.

From Web Site to Web Program

This book uses the same technology discussed in *Advanced Web Sites Made Easy,* but looks at HTML/OS from the software maker's perspective—from the perspective of building multiuser software programs. In other words, this book shows you how to use HTML/OS to build Web software. The book is also useful to users of H_2O, a freely available version of HTML/OS that lacks the database engine. Both, contrasted in the note, "H_2O versus HTML/OS," may be used to build Web software.

H_2O versus HTML/OS

H_2O includes the same feature set as HTML/OS does, but has limited database capabilities. Applications written for H_2O work on HTML/OS and HTML/OS applications, lacking databases with thousands of records work, on H_2O.
Throughout this book, we refer mainly to HTML/OS, but, if you are an H_2O user, feel confident that code provided is equally applicable to HTML/OS.

What do we mean by Web software? We mean software that runs over a browser. Why is Web software important? First, Web software is acessible from browsers and potentially accessible from over the World Wide Web whereas PC software is not.

Second, Web software is easier to use because everyone knows how to use browsers. In addition, Web software is easier to learn, more reliable, and easier to maintain than PC software because there's no software to place on PCs, and most Web servers use Linux or Unix, because these are more reliable operating systems. Web software, in short, is more advanced than PC software. Also, as you will learn in this book, sophisticated Web software can be easier to build than PC software.

The notion that building Web software can be easy is not yet accepted among programmers. Ask the most seasoned PC programmers whether they can build a Web-based application, and they'll tell you it's a daunting task that requires extensive skills and expertise.

Few people know how to build Web software. Why? Because programmers look to the PC world for advice. But the PC world instructs programmers to salvage their outdated PC technologies using systems-integration techniques. Unfortunately, that advice almost always leads to failure. Of course, you won't read about those failures in magazines. But just look around. How many PC software companies do you see with browser-based versions of their products? Even the PC industry insiders can't "Webify" their products. The PC industry knows how to sell software; but its track record regarding the Web is dismal at best.

In this book, we show you the "Web way" of building Web software. This book debunks the entire PC establishment. It says, "Drop the PC tools, and you'll not only be able to build Web software, but that software will be easier to build and better than PC-centric applications." This book is based on the techniques used at Aestiva, the maker of more than a dozen Web-based products.

In this book, we guide you through the process of constructing a Web application, from beginning to end. We show you how to become a successful Web software product developer. We show you the "Web way" of building products.

Is This Book for You?

This book was written for everyone interested in learning how to build Web-based software applications. You may be an experienced PC-product developer who wants to build a Web version of an existing product. Or you may be a Web designer interested in broadening your business. You may be a Web designer, educator, computer scientist, or software engineer. Or you may be an amateur programmer interested in gaining more understanding of this new and exciting arena.

Indeed, this book is useful to anyone interested in Web programming or moving beyond simple HTML page development. To benefit from this book you should have experience with the following:

- Writing HTML tags
- Writing small computer programs or macros

If you have such experience, or you feel confident you can learn HTML tagging and macro programming (appendixes are provided at the end of this book for individuals lacking experience in these areas), this book is for you.

This book is refreshingly devoid of unnecessary learning barriers. In part this is because HTML/OS is devoid of mind-bending complexities. It's also because this book places emphasis on the practical, not the theoretical. This book was written for people at all experience and skill levels.

Your Free Copy of HTML/OS

To use this book you will want to download a copy of Aestiva H_2O. This is a free version of HTML/OS. Downloads are available from the `http://h2o.aestiva.com` Web site. Versions are available for Microsoft Windows, Mac OS X, Linux, Sun Solaris, and BSD. The `http://h2o.aestiva.com` Web site includes instructions for installing H_2O on a local PC or in a Web site hosting account.

How to Read This Book

If you are not familiar with HTML/OS programming, read Part I from beginning to end. Part II serves as a reference to the various building blocks that make up most commercial software applications. Part III puts it all together. You learn how to bundle up products for sale and protect your software from being stolen. We then guide you through the construction of a sample commercial-grade application, from beginning to end, to ensure that you understand the whole process.

The chapters in Part II and Part III of this book can be read in any order. They serve as independent references on the different components required by product makers. If you are building a commercial product, you can use the chapters to guide you in your development process.

This book is self-sufficient in that you do not need to read *Advanced Web Sites Made Easy* to use it. Although *Advanced Web Sites Made Easy* is recommended reading, this book recaps the fundamentals needed to use HTML/OS.

How This Book Is Organized

This book is designed to get you up and running at lightning speed. At the same time, it gives you a foundation in HTML/OS development. Part I of *Advanced Web Software Made Easy* provides the foundation. First, we look at HTML/OS and the applications included with it. Then we review the HTML/OS macro language. After that, we review the 4GL (Fourth Generation Language) database environment built into HTML/OS.

Part II is a reference of Web components. Although the Web development skills needed to build products are similar to those used when building advanced Web sites, the set of components you build and use are different. The chapters in Part II emphasize the components needed when building commercially salable products. It's an invaluable reference that covers myriad topics, ranging from the construction of licensing pages to content management.

Part III describes the construction of a Web-software product from beginning to end. It looks at the construction of a hypothetical product called WebResumes. The product is an employee-accessible company directory typically managed by the human resource department of a company. A version of this product has been deployed at a major news organization.

The appendixes include a special section on registration programs and other tools used in the course of selling products, a crash course on HTML (in case you're not yet conversant in it), and a reference on the HTML/OS tags used in this book.

Building Web Software

The packaging, copy-protection, and installation of software programs is a day-to-day occurrence in the PC world. But these capabilities, so commonplace in the PC world, are not to be found on the Web. The reasons for this vary. For one, the PC world is dominated by a monopoly, whereas the Web is a public trust. The Web is not, nor should it be, dominated by a single company. As a result, these most basic needs are overlooked. The PC industry, which includes commercial and open-source organizations, is of little help.

The open-source community shuns the idea of building intellectual property and protecting it. Instead, it touts a world in which Web developers are called upon to hold people's hands as they install, uninstall, and configure their Web-based systems.

Commercial vendors are not much better. They give priority to their PC-centric systems, a very anti-Web concept. The open-source community and PC-centric vendors rely on systems integration, an approach that's at odds with software packaging and the run-anywhere needs of Web products.

The result of this PC-centric thinking is devastating. Most programmers attempting to build salable Web software packages fail. Whether they choose open-source tools, or commercially available systems-integration tools, the unexpected happens. They are burdened by endless development and pre-sales and post-sales problems, resulting in the need for excessive compromise. Rarely does the final product fulfill the exciting promise of the original business proposition.

In the next section of this introduction you learn why HTML/OS doesn't suffer the same problems as systems integration and why it's an excellent choice. You learn that HTML/OS takes us back to the days when multiuser programs were easily developed and placed on

Unix systems and delivered to customers via terminals. Only this time, applications run on almost any server, and they are delivered via Web browsers, not terminals.

Most important, the HTML/OS approach is proven historically. Unlike recent Application Service Provider (ASP) business models, which are a throwback to the failed timesharing businesses of the 1980s, multiuser business models are accepted and proven. Tens of thousands of software millionaires made their fortunes based on this tried-and-true business model.

In many ways, HTML/OS unleashes a new and exciting area of computing, because until now, it was impractical to build highly salable, multiuser, Web-based software programs. Now anyone with some knowledge of HTML and some macro programming experience can build salable Web-based software programs.

The opportunities created by HTML/OS are widespread, multi-industry, and international. (HTML/OS supports all languages.) Ask yourself a simple question: If forced to choose between two business software packages, each with the same features except that one program was PC-based and the other was Web-based, which would you choose?

The answer most often is the Web-based program. The Web-based program is accessible from home or office. No software needs to be set up on each PC. No software needs to be upgraded or maintained on each PC. The software will be more reliable, easier to upgrade, and easier to back up. You'll be able to access it from your Macintosh, Windows PC, or Linux box. The Web wins the prize.

Indeed, HTML/OS foretells the day when Web software exceeds the capabilities of multiuser PC-based applications in terms of features, Web accessibility, ease of development, ease of deployment, and ease of maintenance. If cell phones eventually replace most land-line telephones, it's not difficult to envision the day when multiuser, Web-based business software will replace its PC-tied ancestors.

The HTML/OS Solution

In an earlier section, we mentioned the problems encountered by product developers. In this section, we provide the rules needed to avoid these problems. Thankfully, they are easily solved; but you need to change your perspective. The programmer who applies PC product-development strategies to Web-based programming is in for a rude awakening. The need to run products on Web servers changes the fundamentals. This simple fact is the root cause of so much pain and failure. We discuss these issues in detail in the following sections.

Know the Rules

The PC is not the Web, and the Web is not the PC. Unlike the computer world of 20 years ago, where anyone could write, compile, and sell a program, today things are different. Many developers, after more than a year of programming, have had to scrap their work and their business. Sometimes the products became too expensive. At other times, development costs skyrocketed out of control as the developers rushed to support new platforms and satisfy requests for new features.

This section exposes the decisions that can lead to failure and shows you why the HTML/OS development environment is well suited to Web product development. After you know the rules of the Web, you'll be able to make smart decisions and enjoy the benefits that come with building and selling Web-based products.

Rule #1: Build Cross-Platform Products

Unlike PC computing environments, the Web's computing environment is multiplatform. Many Webmasters prefer Unix and Linux because of their superior reliability and security. Unix and Linux are also more popular as the servers behind virtual servers, which are the servers behind rented hosting accounts. They are also the operating systems behind a new generation of hand-held devices and cell phones that merge the capabilities of cellular technologies with the Web. In total, this makes the Web space fundamentally different from PC environments. Microsoft isn't more popular than Linux. And the trend is toward greater variety, not less.

Why is this important? Because building a product for one platform can spell disaster. The math is simple. Let's say you build a product that sells for $1,500. Now let's say it costs you $1,500 to get three people to buy your product. If your product is cross-platform, meaning it can run on Unix, Linux, or a Windows server, you gain all three customers. If it's not, you gain only one customer. The cross-platform product earns $3,000. The single-platform product has no profit at all.

It is also important not to fool yourself into thinking you can build a product for one platform and add versions for other platforms later. After a product has been launched, you'll immediately be faced with requests to improve it. Such improvements will be ongoing. Once a product has been launched, there's no time to go back and develop a product for another platform. You can't catch up. The platform-compatibility decision is one that must be made in the beginning.

The HTML/OS environment discussed in this book solves this problem because its engine is hardware independent. Your product is cross-platform from the very beginning. With HTML/OS, you build once and deploy using the HTML/OS engine for the particular hardware you want to run on. The HTML/OS engine shields you from the particulars of each kind of hardware, so that you can sell your product to people running Windows, Unix, Linux, or Mac OS X. Your product will also run in virtual hosting accounts.

Rule #2: The Webmaster Is King

Unlike off-the-Web computing environments, Web software runs on a Web server and thus requires a Webmaster. That Webmaster may be your buyer, a person in an IT department, or a hosting organization. Webmasters protect their Web servers the way a lioness protects her cubs. They have no interest in setting up or maintaining new systems. And most avoid extra tasks that might add to their already busy schedules.

Requiring that a Webmaster maintain an SQL server for your customer, reconfigure the server, or provide installation support services to your customer can and will kill sales. Consider a product built using standard systems-integration methods. Eventually, someone must maintain the external database and provide support for patches and upgrades. The cost will either be yours or your customer's. Either way it's a hidden cost that can exceed the cost of the software you want to sell. Again, let's say your product sells for $1,500, and you spend $1,500 to get three people to buy your product. If your product requires the Webmaster to support the product, you may gain only one customer. If you provide the support, you may gain three customers; but you will need to provide the support, resulting in future hidden costs. Either way, you lose.

How does HTML/OS solve the Webmaster support problem? First, the HTML/OS engine is a single engine, so no systems-integration support is required on the part of the Webmaster. No reconfiguration of the Web server is needed, either. Plus, the database engine included with HTML/OS is maintenance free. Webmasters need not take responsibility for your customer applications. They only need to maintain a functional Web server, a task they already perform. Products built with an HTML/OS engine have no ongoing support needs and hence no added support costs.

Rule #3: Simplicity Is a Necessity, Not a Convenience

PC computing is rooted in proprietary hardware systems. On the Web, the center of computing is the nonproprietary, multiplatform HTML document. The two require distinct

engineering systems, which when muddled, cause unnecessary complexity—often 100 times the complexity as measured by the number of lines of code needed to program a component. Imagine what this must mean to the developer responding to requests for new features.

Simplicity is more than a convenience. It means you'll need fewer staff members to complete a task. It means you can solve problems ten times as fast. It means you can build that many more features into your product. Simplicity is a necessity. You should always use a Web-centric approach when building for the Web.

The HTML/OS engine is one such approach. On average, it delivers 100 times less complexity. A commercial product can be developed with one developer. Changes, bug fixes, and new features can be added quickly. Your ability to respond to customer requests will be measured in days, not months or years.

Rule #4: Use Tools Friendly to Product Developers

Building Web products is different from building a Web site. Web sites do not need fast install tools. They do not need to be copy protected. They do not need to be bundled and installed across different platforms. These are just three of the differences between Web development and Webproduct development. There are many more.

Using a development environment lacking such tools puts you at an extreme disadvantage. The development of installing such tools alone can take longer than the development of the core software. Without the tool, you'll unexpectedly be buried in endless preparations as you close in on launch day. It is at this point most Web software businesses fail. Unfortunately, by the time the developer gets to this stage, it's already too late.

HTML/OS solves this problem by including a complete set of product-development tools with its engine.

Rule #5: Visual Tools Are a Dead End

Although the PC world espouses visual tools, sometimes called code generators or code wizards, reliance on visual tools and wizards denies the programmer an underlying understanding of how to program. PC tools allow you to build applications from beginning to end without touching a line of code.

The Web is different. Web pages cannot be programmed without having to go into the code. It's the same for Web programming. Visual tools hide the underlying complexities, so that when the time comes for you to dive in and edit the programming code, you're severely restricted. It's better to learn how to write a 20-line computer program than to use a wizard that generates thousands of lines of code over which you have no control. On the Web, visual programming tools do more harm than good.

When you program with HTML/OS, you do not use visual tools. Instead, you write macros. A five-line macro in HTML/OS might replace 500 lines of code generated by a wizard. Writing the five-line program might take you a bit longer than using a wizard, at least in the beginning. But changing the code will take you a lot less time, because you wrote the code and know how it works.

Book Conventions

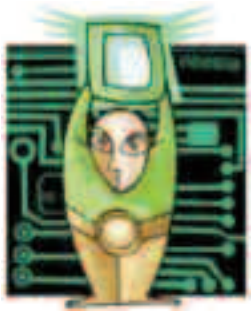
This book describes how to build Web-based programs containing HTML and HTML/OS programming. In this book, HTML tags are written in lowercase whereas HTML/OS instructions are written in uppercase, although HTML/OS tags can be written either way. HTML tags, HTML/OS tags, variables, and all programming content is written in courier type, as in this example:

```
IF ISGOODKEY != "TRUE" THEN GOTO "not_registered.html" /IF
```

What's to Come?

As you've seen, the world of Web software development is at odds with the PC world, but is full of promise and opportunity. Most important, this introduction has introduced you to HTML/OS, a development environment that simplifies Web software construction, which is why we titled this book *Advanced Web Software Made Easy*.

In Part I of this book you'll get on a copy of HTML/OS and learn how HTML/OS works. You'll review the tools available to you and begin experiencing for yourself the beauty of Web product construction.



PART I

Essentials