RECORDS PRESENTS

Gods of the Web

Erim Emmanuel Otioh

Gods of the Web

Road from Web Basics to Fullstack JavaScript

By

ERIM EMMANUEL OTIOH

PREFACE

This book is an initiation approach to General Computer Programming with foundations on the Web. With the dynamic nature of Web technologies, having a base/foundation on Web technologies is the safest start for all beginners. The boldest step you have ever taken in attacking the limiting friction to starting your Programming Career is reading this Preface which you are doing now. The last statement is to emphasize that in Programming, every bit is important, so you must pay attention to everything no matter how little.

This book is set to pick you up from a no Programmer through the basics of the Web to advance Web technologies and land you as a Fullstack Software Developer with great strength in JavaScript and a great skill in PHP for backend. The book will also in future editions introduce a new Web Framework/Library known as Flame.

As a take-off I'm giving you a gist on the roots of everything you will learn in this book beginning with the Hyper Text Mark-Up Language (HTML). Since HTML's introduction in 1993, web-programming technologies have been in flux, with web programmers using different versions of HTML for different browsers. The constant change made it difficult for authors to write quality textbooks about the subject. Consequently, most of the books were trade books, not textbooks. With HTML5's approval as a "stable recommendation" in 2014, web programmers and browsers

appear to have embraced it fully. With the huge demand for web programmers in the workforce, there has been a significant demand for web-programming courses for quite a while. Now that web programming has coalesced around HTML5, there is a need for better textbooks about web programming.

PREREQUISITE

This book is not an introduction to Computer Science or Software Engineering, in general. We'll assume that you have some working understanding of Problem Solving and Algorithm Development. If you're brand new to Computer Science, I suggest consuming a material or two on Problem Solving before coming back to take the plunge into your software development journey. Specifically, you are not required to have any understanding of coding any programming language, because this book would offer that to you, but if you already have knowledge or experience in the field, kudus for the advantage.

RESOURCES

It’s dangerous to go alone! Well, not really, but that doesn’t mean you have to. Here are some resources you may find useful as you work through the book: The GitHub repository for this book contains all of the code samples we’ll be discussing. If you get stumped, or want more context, try looking here first.
https://github.com/godsoftheweb/

You may also want to work hand in hand with W3School vast resources on Web technologies that applies to this studies.
https://www.w3schools.com/

CHAPTER 1

INTRODUCTION TO WEB PROGRAMMING

Introduction

Have you ever perused the Web and wondered how its web pages are made? If so, this book is for you. Actually, even if you haven't thought about how web pages are made, this book can still be for you. All you need is a logical mind and an interest in creating things and solving problem. This book takes you on a journey where you learn to create informative, attractive, and interactive web pages. So climb on board and enjoy the ride! To make this book accessible to readers with little background in computers, we start slowly and build upon what comes earlier in the book.

If you already know how to program, you'll probably want to skip some of the programming basics when we get to JavaScript. But rest assured that unless you already know HTML, CSS, and JavaScript, the vast majority of this book's content should be new to you. Let's start with a brief description of the Web, which is short for World Wide Web. Most people say "Web" instead of "World Wide Web," and we'll follow that convention. The Web is a collection of documents, called web pages, that are shared (for the most part) by computer users throughout the world. Different types of web pages do different things, but at a minimum, they all display

content on computer screens. By "content," we mean text, pictures, and user input mechanisms like text boxes and buttons.

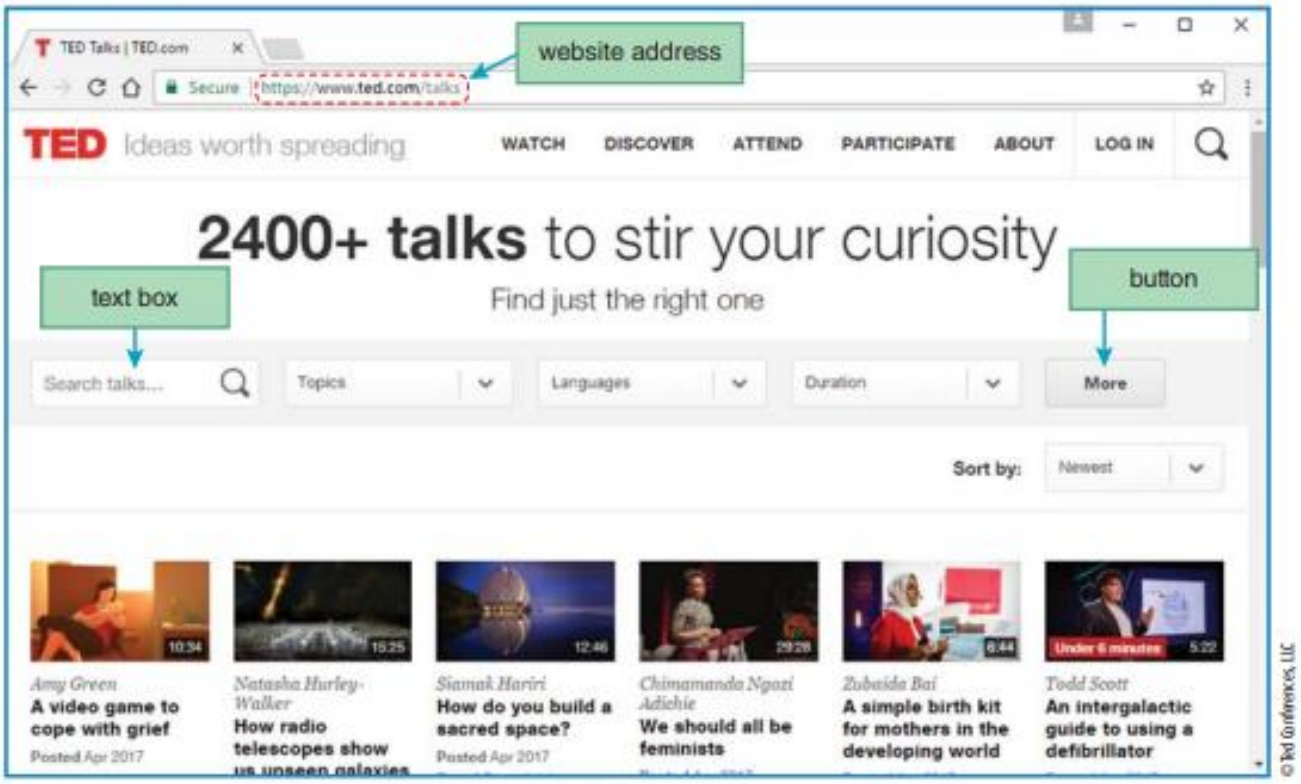


FIGURE 1.1 A typical web page

FIGURE 1.1 shows a typical web page. Note the web page's text, pictures, text boxes, and buttons. Also note the web page's address shown in the figure's address bar. The web page address is the location where the web page resides on the Internet. Speaking of the Internet, what is it? It's a collection of several billion computers connected throughout the world. Each web page is stored on one of those computers.

Next, we'll focus on the appearance of displayed content. Then on to organizational constructs, pictures, sound clips, and video clips. Finally, we

will focus on implementing user input controls. For example, in Figure 1.1, note the text boxes and the buttons.

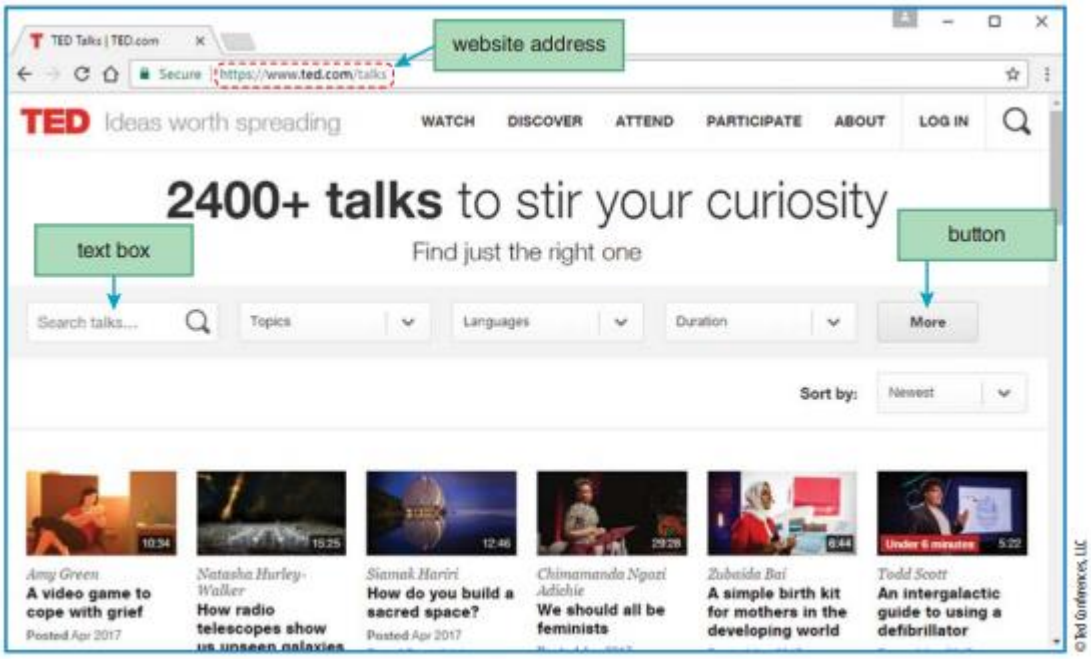


FIGURE 1.1 **A typical web page**

Those are controls. You'll learn about those controls, plus more controls, in the last several chapters. In this first chapter, we stick with the basics, so you can get up and running quickly. Specifically, we start with some overarching concepts that explain the process of web page development and dissemination. Then, we introduce the basic constructs that you'll use to describe and display a web page's content. Next, we provide a cursory overview of Cascading Style Sheets (CSS), which you'll use to display a web page's content in a pleasing, formatted manner.

Creating a Website

A website is a collection of related web pages that are normally stored on a single web server computer. A web server is a computer system that enables users to access web pages stored on the web server's computer. The term "web server" can refer to the web page-accessing software that runs on the computer, or it can refer to the computer itself. To create a website, you'll need these things: (1) a text editor, (2) an upload/publishing tool, (3) a web hosting service, and (4) a browser. We'll describe them in the upcoming paragraphs.

Text Editors

There are many different text editors, with varying degrees of functionality. Microsoft's Notepad is free and provides no special web functionality. To use it, the web developer simply enters text, and the text appears as is. Although it's possible to use a plain text editor such as Notepad, most web developers use a fancier type of text editor.

Like already said, there are many text editor options but for the purpose of this course I recommend Visual Studio Code so that you may be in Synch with future examples. You can download it from:
https://code.visualstudio.com/docs/?dv=win

Web Page Uploads

After you enter your web page text on your local computer with your favorite IDE, you'll probably want to publish it. Publishing means that you upload your web page to a web server computer so other users can access it on the Web. Some IDEs, like Dreamweaver, provide built-in uploading capabilities, but other IDEs, like Visual Studio, do not. For IDEs that do not provide built-in uploading capabilities, you'll need to use a separate file upload tool. There are lots of file upload tools. For this course we strongly recommend a very popular FTP known as FileZilla.

Web Hosting Service

For a file upload tool such as FileZilla to work, you need to have a web server computer on which to store the uploaded files. For the uploaded files to be accessible as web pages on the Web, your web server computer needs to have a web hosting service in place. The web developer usually doesn't have to worry about the web hosting service software. If the web developer is part of a medium- to large-sized organization, then the organization's information technology (IT) department will install and maintain the web hosting service. On the other hand, if the web developer is part of a very small organization or not part of an organization at all, the developer will need to set up the web hosting service or rely on a generic web-hosting company (e.g., GoDaddy .com) to do so. Regardless of who's in charge of the web hosting service, all web hosting services need to have a mechanism for receiving uploaded files from a file upload tool. Typically,

that mechanism is an FTP (file transfer protocol) server, which is a program that runs on the web server computer.

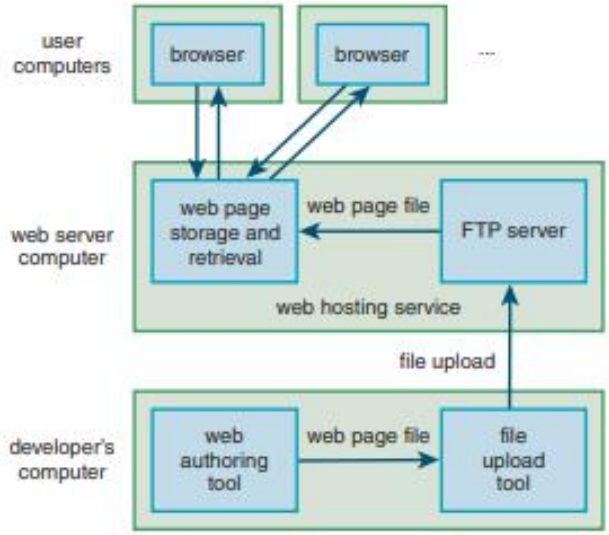


FIGURE 1.2 **Website file processing**

FIGURE 1.2 is a pictorial description of how the FTP server fits in with the rest of the website creation process.

HTML Tags

Now we're going to describe how to implement elements for a web page. To implement an element for a web page, you'll need to use tags. For example, if you want to implement a strong element (in order to put emphasis on the element's content and display using boldface), surround the content with tags. Here's how to implement a strong element for the word "very": Caiden was very happy when her tooth finally came out.

The use of tags is the key characteristic of a markup language. Why is it called "markup"? A markup language "marks up" a document by

surrounding parts of its content with tags. Web pages are implemented with HTML, which stands for Hypertext Markup Language. You already know what “markup” means. The “hyper” in “Hypertext” refers to HTML’s ability to implement hyperlinks. A hyperlink (or link for short) is where you click on text or an image in order to jump to another web page. So HTML is a language that supports markup tags for formatting and the ability to jump to other web pages.

To simplify the web page creation process, the WYSIWYG option (for web authoring tools) lets you hide the HTML tags. Unfortunately, in hiding the HTML tags, the developer loses some control, and the resulting web pages tend to be sloppy. Even if your web authoring tool generates clean HTML code when in WYSIWIG mode, we strongly recommend that you enter all your tags explicitly, at least for now. That will help you learn HTML details so you’ll be able to understand and debug subtle code issues. Most, but not all, HTML tags come in pairs with a start tag and an end tag. For example, the following code uses an
<h1> start tag and ends with </h> end tag

```
<h1> Today's Weather </h1>
```

Note that each tag is surrounded by angled brackets.

Tag in general, refer to it as the h1 element. There are two types of elements—container elements and void elements. A container element (usually called simply a “container”) has a start tag and an end tag, and it contains content between its two tags. For example, the h1 element is a container. On the other hand, a void element has just one tag, and its content is stored within the tag. We’ll see an example of that when we get to the meta element

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="author" content="John Dean">
<meta name="description" content="Kansas City weather conditions">
<title>K.C. Weather</title>
<style>
 h1 {text-align: center;}
 hr {width: 75%;}
</style>
</head>
<body>
<h1>Kansas City Weather</h1>
<hr>
<p>
 It should be pleasant today with a high of 95 degrees.<br>
 With a humidity reading of 30%, it should feel like 102 degrees.
</p>
<div>
 Tomorrow's temperatures:<br>
 high 96, low 65
</div>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
 .
 .
 .
</head>
<body>
 . . .
</body>
</html>
```

The first construct, , tells the browser what type of document the web page is. Its html value (in) indicates that the document is an HTML document, and more specifically that the document uses the HTML5 standard for its syntax. Syntax refers to the words, grammar, and punctuation that make up a language.

After the doctype instruction comes the html element. It's a container, and it contains/ surrounds the rest of the web page. Its start tag includes lang="en", which tells the browser that the web page is written in English. The head and body elements are also containers. The head element surrounds elements that provide information associated with the web page as a whole. The body element surrounds elements that display content in the web page. Container elements must be properly nested, meaning that if you start a container inside another container, you must

end the inner container before you end the outer container. Because the body element starts inside the html element, the

title Element

Let's now dig a little deeper into the head element. The style element is more complicated, and we'll discuss it later in this chapter. The head element contains two types of elements—meta and title. In your web pages, you should position them in the order shown in our code snippet, meta and then title. But in the interest of clarity, we'll discuss the title element first. Remember that the head element surrounds elements associated with the web page as a whole. The web page's title pertains to the entire web page, so its title element goes within the head container. The title element's contained data (e.g., "K.C. Weather") specifies the label that appears in the browser window's title bar. Go back to Figure 1.3 and verify that "K.C. Weather" appears in the tab near the top of the window. With browsers that support tabbed windows, that tab is considered to be the browser's title bar

meta Element

Note the meta elements within the head container. The meta elements provide information about the web page. If you look up "meta" in a standard dictionary, you'll probably see a confusing definition. In everyday speech and in HTML, "meta" means "about itself." As in "Check out this video of two guys watching a how-to video about appreciating videos." "Dude, stop now. You're hurting my brain. That's so meta!" There are many

different types of meta elements—some you should always include, but most are just optional. We'll present a few of the more important ones soon, but first some details about the meta element's syntax. The meta element is a void element (not a container), so it does not have an end tag. Note how there are no end tags for the three meta elements in Figure 1.6. Many web programmers end their void elements with a space and a slash. For example: The space and slash are a bit outdated, and we recommend omitting them. If you decide to include them, be consistent. Don't worry about the meaning of charset and utf-8 for now; we'll discuss them shortly.

HTML Attributes

Before we formally introduce a few of the different types of meta elements, we first need to dis cuss attributes, which are used in all meta elements. Container elements provide information between their start and end tags. Void elements (including the meta element) have no end tags, so they can't provide information that way. Instead, they provide information using attributes. In the following example, charset is an attribute for a meta element: Most attributes have a value assigned to them. In this example, charset is assigned the value "utf-8". Although most attributes have a value assigned to them, some do not. Later on, we'll see some attributes that appear by themselves, without a value. You should always surround attribute values with quotes, thus forming a string. A string is a group of zero or more characters surrounded by a pair of double quotes (") or a pair of single quotes ('), with double quotes preferred.

Attributes are more common with void elements, but they can be used with container ele ments as well. Here's an example of a container element that uses an attribute: • • •

The lang attribute tells the browser that the element is using a particular language for its content. You can use the lang attribute for any element. Here we're using it for the html element, so it means that the entire web page uses French. You're not required to use the lang attribute, but we recommend that you do include it for the html element. For web pages written in English, use

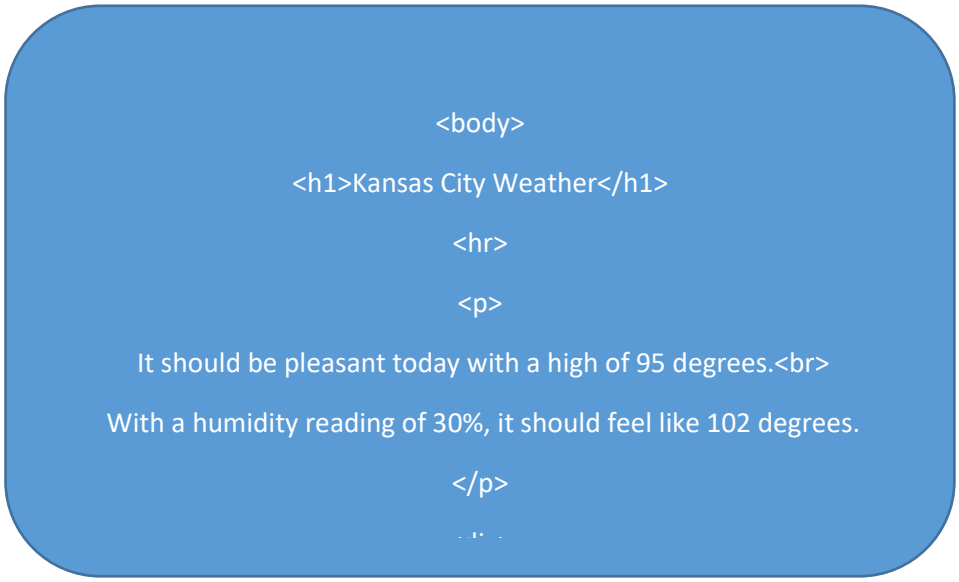
body Elements: hr, p, br, div

In the prior sections, we covered the elements that appear inside the weather page's head con tainer. Now let's cover the elements that appear inside the weather page's body container. Our example which shows the body container code, note the h1, hr, p, br, and div elements.

We've already talked about the h1 heading element, so now let's focus on the other elements.

The hr element is used to render a horizontal line. When a browser renders an element, it figures out how the element's code should be displayed. To keep things simple, you can think of "render" as a synonym for "display." Go back to Figure 1.3 and note the horizontal line on the weather page browser window. The "h" in hr stands for horizontal. The "r" in hr stands for rule, presumably because a rule is another name for a ruler, which can be used to make a straight line.

The hr element is a void element, so it uses just one tag, <hr>.



The p element is a container for a group of words that form a paragraph. Normally, browsers will render a p element's enclosed text with a blank line above the text and a blank line below it. Go back to Figure 1.3 and note the blank lines above and below the two-sentence paragraph. In Figure 1.7, note how we indented the text between the

start tag and the

end tag. Whenever you've got a p element whose enclosed text is greater than one line, you should put the start and end tags on separate lines and indent the enclosed text. That rule is an example of a coding-style convention rule (or style rule for short). Whenever you write a program, including an HTML program, it's important to follow standard coding-style con ventions, so your program is easy to read by you and also by future programmers who need to understand your program. Programmers get used to certain conventions, such as when to use uppercase versus lowercase, when to insert blank lines, and when to indent. You don't want to jar someone reading your program by using nonstandard conventions. For a complete list of coding-style conventions used in this book, see the two appendices: Appendix 1, HTML5 and CSS Coding-Style Conventions, and Appendix 2, JavaScript Coding-Style Conventions. You might want to skim the appendices now, but don't worry about the details. We'll cover those details as we proceed through the book. Even though it's a container, the HTML standard allows p's start tag to appear without its end tag. However, that's considered to be bad style, so don't do it. You should never omit end tags for container elements because then it's more difficult for the browser and for people reading your source code to figure out where the container element ends. A div element is also a container for a group of words, but it's more generic, so the words don't have to form sentences in a paragraph. div stands for division because a division can refer to a part of something that has been divided, and a div element

is indeed a part of a web page. Normally, the div element causes its enclosed text to have single line breaks above and below it. If a div element's enclosed text is greater than one line, then proper style suggests putting the

tags on separate lines and indenting the enclosed text.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="author" content="John Dean">
<title>Electric Power History</title>
</head>
<body>
<h2>Brief History of Electric Power</h2>
<hr>
```