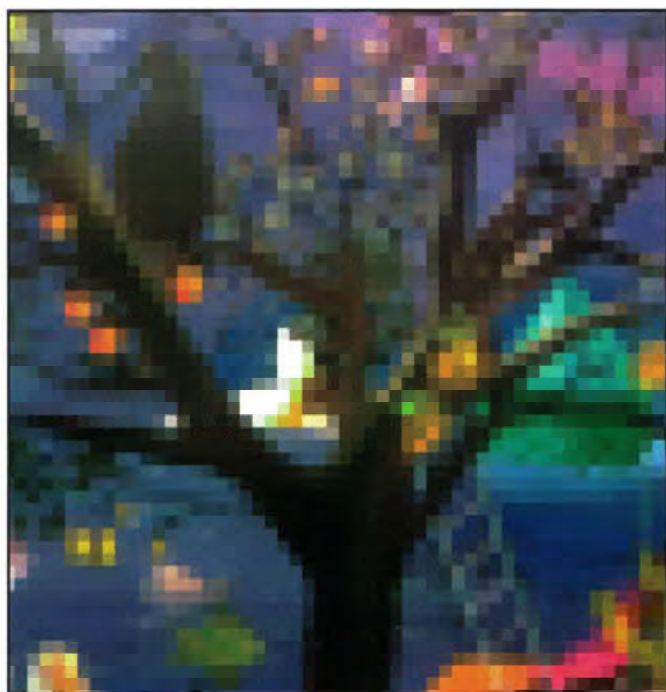


RAGGETT ON HTML 4

S E C O N D E D I T I O N



DAVE RAGGETT

JENNY LAM

IAN ALEXANDER

MICHAEL KMIEC



ADDISON-WESLEY

A USER'S GUIDE
TO HTML 4

Raggett
on
HTML 4

Raggett on HTML 4

second edition

**Dave Raggett
Jenny Lam
Ian Alexander
Michael Kmieć**



Addison-Wesley

Harlow, England • Reading, Massachusetts • Menlo Park, California
New York • Don Mills, Ontario • Amsterdam • Bonn • Sydney • Singapore
Tokyo • Madrid • San Juan • Milan • Mexico City • Seoul • Taipei

© Addison Wesley Longman 1998

Addison Wesley Longman Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the World.

The rights of Dave Raggett, Jenny Lam, Ian Alexander and Michael Kmiec to be identified as authors of this Work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a licence permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London W1P 9HE.

The programs in this book have been included for their instructional value. They have been tested with care but are not guaranteed for any particular purpose. The publisher does not offer any warranties or representations nor does it accept any liabilities with respect to the programs.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Addison Wesley Longman Limited has made every attempt to supply trademark information about manufacturers and their products mentioned in this book. A list of the trademark designations and their owners appears on page xvi.

Cover designed by odB Design and Communication, Reading, UK
and printed by Riverside, Reading, UK
Cover illustration by Jenny Lam
Typeset by 43
Printed and bound in the United States of America

First printed 1997

ISBN 0-201-17805-2

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

PREFACE

This book provides a detailed and lively description of the Web publishing language HTML 4, written for both the novice and the advanced user. We hope that it is intelligently presented and enjoyable to read.

HTML 4 is the latest version of the Web's publishing language: it follows on from HTML 3.2, which was a much simpler specification. HTML 4 comes from the World Wide Web Consortium ('W3C'), an international organization founded in 1994 to further the evolution of the World Wide Web. Both Netscape and Microsoft are members of W3C: they are publicly committed to the implementation of HTML 4 on their browsers. HTML 4 is on its way to becoming a truly cross-industry standard useful to a wide and international audience.

New features for HTML 4

Many entirely new features of HTML 4 are explained in this book:

- Tables are more sophisticated.
- Forms have been modified to help visually impaired readers who use speech generation software to read Web pages.
- Style sheets are fully supported by HTML 4; their use is now an integral part of the process of designing a Web site.
- With a view to making the Web truly international, HTML 4 encourages publishing in a variety of human languages.
- Frames have now been formally incorporated into the specification.
- The new `OBJECT` tag at last provides a standard way of incorporating many kinds of multimedia application.

About the authors



Dave Raggett



Jenny Lam



Ian Alexander



Michael Kmiec

Dave Raggett

Dave Raggett is the lead architect for HTML 4 on assignment from Hewlett-Packard Laboratories. He works with the World Wide Web Consortium where he continues to pursue a major role in HTML's development.

Jenny Lam

Jenny Lam is a technical writer who specializes in explaining science and technology at layman level. In an age of cyberspace and virtual men, Jenny remains down-to-earth by combining writing with her career as a full-time housewife and mother.

Ian Alexander

Ian Alexander is a systems engineer specializing in requirements engineering. He divides his time between writing, training, research and consultancy. When not otherwise engaged he collects natural patterns for his Web site.

Michael Kmiec

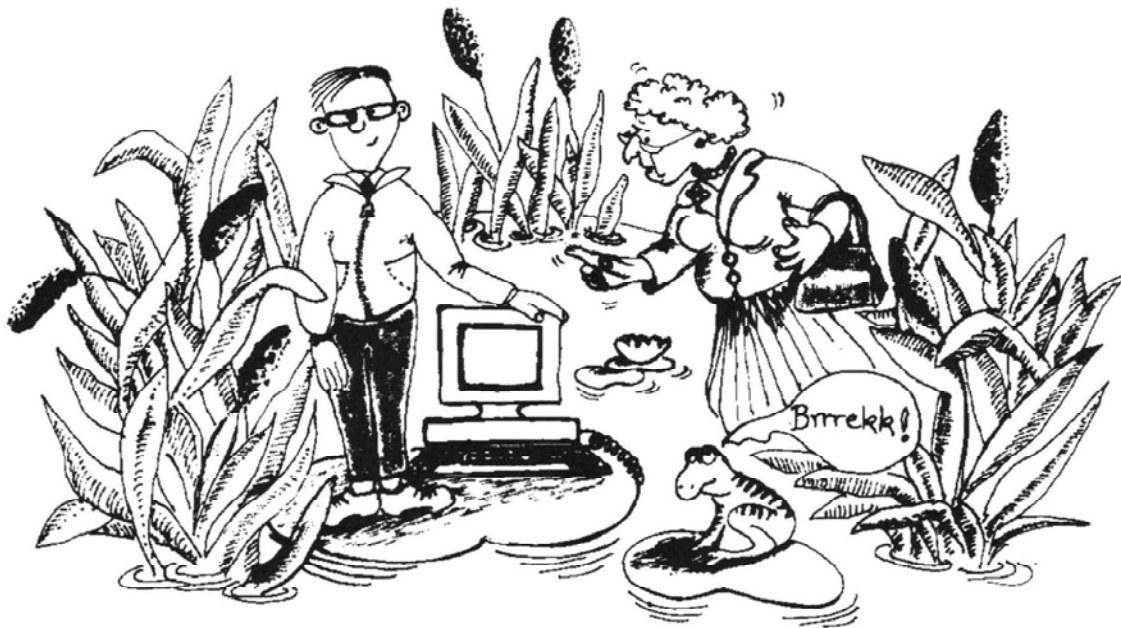
Michael Kmiec is an applications developer who uses C, C++, Java and a bit of Tao to make things work.

Overall structure

The book starts by explaining the basic concepts of the Web for the layman. It then goes on to give an inside story of how HTML came about, embarking on an interesting historical account of the language. There follows a chapter which tells you how to compose your first HTML document and also introduces you to the technical terms and concepts needed to understand the later and more technical chapters in the book.

We cover:

- Simple mark-up of paragraphs and headings including how to apply a style to give text a color on the screen or display text in a special font and size.
- Character mark-up including subscripts, superscripts, bold text, emphasized text and other effects.
- Lists: simple bulleted lists, numbered lists, lists within lists, lists with customized bullets.
- Hypertext links, including a layman's explanation of what they are and how to construct them, how to link to a particular place in the document rather than to the beginning.
- Style sheets: an introduction to using the CSS style sheet language to control color and font, background texture and position on the 'page'. The CSS language is very easy to use: if you have some experience of desk top publishing then simple use of CSS should be no problem.
- Graphics: inserting photographs, illustrations and other artwork into Web pages. There is a discussion of graphics formats and notes on which are applicable under what circumstances: GIF, JPEG and PNG.
- Tables, including design principles, merging table cells, automatic alignment, fancy borders and more.
- Frames: how to structure a Web site to keep individual documents simple and maintainable, how to provide decent navigation tools, how to combine text with graphics, and, not least, how to support older browsers.
- Multimedia on the Web: the use of the new standard OBJECT tag to insert animation clips, music and sound bites, and much else besides.



An explanation of the characters in the cartoons in this book.

Adrian Anorak

Adrian is always in his head, he lives in the virtual world of the Internet, which consists of a world-wide web of interlinked nodes, represented here by a lily pond.

Prunella

Prunella is Adrian's aunt. She treats Adrian as the small boy she embarrassingly once knew, and probably still thinks he is.

Lily Pond

Virtual pond in Prunella's garden, representing the waters of Cyberspace. The water-lily leaves are nodes on the web.

Fred Frog

Fred lives in the pond, hopping from leaf to leaf. He is of course a virtual frog, unlike Adrian who only imagines he is virtual (well, maybe . . .)
Fred is naturally derisive (Brrrekk!)

- Scripts to support a range of simple animation and navigation purposes. We deal with the HTML involved in client-side scripting far enough to help readers understand what they can achieve, and when they will need to make the investment in server-side programming.
- Appendices consisting of convenient reference material, an entertaining and informative glossary, and a look behind the scenes at the Cyberia Café.

Thanks to . . .

Our thanks to Fred Volans for livening up the book with her wonderful cartoons.

We would also like to thank Karen Mosman and Nicky McGregor for their valued efforts in diplomacy and negotiation, also Kathryn Esplin who did the copy-editing, and Louise Wilson who coordinated the production of this book.

Thanks also go to Grandpa and Grandma Lam for distracting children, guinea-pigs, rabbits and birds at critical moments, a service which cannot be underestimated in its value. And to Sabina for constant support and the best coffee in the world.



of the Internet,
epresented

oy she

of Cyberspace.

ourse a virtual
aybe . . .)

CONTENTS

Preface	v
1 Introduction to the World Wide Web	1
1.1 Simple use of the Web	2
1.2 The Web in the context of the Internet	6
1.3 Basic components of the Web	7
1.4 A universally understood publishing language: HTML	9
1.5 The HTTP protocol	9
1.6 More than just text and pictures on the Web	10
1.7 HTML and its relationship to SGML	13
1.8 The Web for people with disabilities	14
1.9 Math and HTML	15
1.10 HTML and style sheets	15
2 A history of HTML	17
3 Simple tutorial and introduction to HTML 4	35
3.1 Introduction	36
3.2 Step-by-step guide: Entering text and displaying it on a browser	36
3.3 More on HTML syntax	40
3.4 A more formal explanation of HTML terms and concepts	44
3.5 Including symbols and accented characters in documents	47
3.6 Non-breaking space characters, hyphens and spaces	48
3.7 Formal structure of an HTML 4 document	48
3.8 The HTML 4-compliant document	49
4 Marking up paragraphs and headings	50
4.1 Introduction	51
4.2 Paragraph mark-up	52
4.3 Preformatted text	60
4.4 Quotations with the <code>Q</code> and <code>BLOCKQUOTE</code> elements	61
4.5 The <code>DIV</code> element to give a section or chapter a uniform style	62
4.6 <code>ADDRESS</code>	62

5	Character emphasis	65
5.1	Introduction	66
5.2	Font-style elements	67
5.3	Phrase-style tags	70
5.4	Special text-level elements	73
5.5	Combining tags	78
5.6	Elements rarely used but still in the HTML 4 specification	78
6	Marking up lists	80
6.1	Introduction	81
6.2	Bulleted lists	82
6.3	Numbered lists	84
6.4	Definition lists	86
7	Hypertext links	89
7.1	Introduction	90
7.2	Simple examples of hypertext links	91
7.3	Links to a specific point in a document: NAME and ID	97
7.4	Hypertext links using clickable graphics instead of text	103
7.5	Link semantics: REL and REV	105
7.6	Using the TITLE attribute	106
7.7	Where to go from here	106
8	Graphics on the Web	107
8.1	Introduction	109
8.2	How IMG works	110
8.3	Telling the browser how big your image is for faster loading	117
8.4	Borders around images	117
8.5	Background textures for your Web pages	119
8.6	Placing a texture or color behind an image	120
8.7	Horizontal rules with HR	121
8.8	Preparing graphics for the Web	122
8.9	Thinking about the blind and visually impaired	129
9	Introduction to style sheets for Web documents	130
9.1	Introduction	131
9.2	How CSS style sheets work	134
9.3	Incorporating stylistic information into a document	136

65	9.4	How style sheet properties interact with each other	140
66	9.5	Constructing named styles for HTML elements	141
67	9.6	Built-in styles for drop-down initial characters and specially formatted first lines	144
70	9.7	Worth noting: SPAN and DIV in the context of style sheets	147
73	9.8	Fonts for the Web	148
78	9.9	Style sheet properties for controlling font	151
78	9.10	Setting the color of the text	154
80	9.11	Specifying the background color	155
81	9.12	Specifying a background image for an element	156
82	9.13	Underlined, boxed text, blinking text and other character effects	157
84	9.14	Text spacing, indentation and alignment	158
86	9.15	Margins and white space around text and pictures	160
	9.16	Borders around elements	163
	9.17	List style	163
89	10	Tables	165
90	10.1	Introduction	167
91	10.2	Simple tables	168
97	10.3	Aligning text in individual table cells	170
103	10.4	Table headers	174
105	10.5	Table captions	175
106	10.6	Controlling the table as a whole	176
106	10.7	Cells that span rows or columns	177
107	10.8	Designating header, body and footer groups	178
109	10.9	Organizing by columns	179
110	10.10	Inheritance	182
117	10.11	Frames, borders, rules	182
117	10.12	Putting other elements inside tables	189
119	10.13	Speeding up display	194
120	10.14	Summary	196
121	11	Forms	197
122	11.1	Introduction	198
129	11.2	Why would you want information from users?	199
130	11.3	Your first form	201
131	11.4	HTML's form elements	202
134	11.5	Submitting forms	213
136	11.6	Form design	214
	11.7	HTML 4 extensions to forms	218
	11.8	Summary	227

12	The document head	228	
12.1	Introduction	229	
12.2	The TITLE element	230	
12.3	The ISINDEX element	230	
12.4	The LINK element	231	
12.5	The META element	237	
12.6	The BASE element	243	
12.7	The NEXTID element	243	
12.8	The SCRIPT element	244	
12.9	The STYLE element	244	
13	Frames	246	
13.1	What frames are	247	
13.2	Basic frames	254	
13.3	Using frames elegantly	271	Appendix
14	Image maps	281	Appendix
14.1	Introduction	282	
14.2	Server-side image maps	282	Appendix
14.3	Client-side image maps	285	
14.4	Summary	287	Appendix
15	The OBJECT element	288	Appendix
15.1	Introduction	289	
15.2	The PARAM element to provide parameters for the object	296	Appendix
15.3	Summary	298	Appendix
16	Client-side scripting	300	Appendix
16.1	Introduction	301	
16.2	The SCRIPT element	301	Appendix
16.3	The SRC attribute to reference a URL for an externally loaded script	302	Appendix
16.4	LANGUAGE to specify scripting language	302	
16.5	TYPE to specify scripting language as Internet media type	302	Appendix
16.6	The NOSCRIPT element	303	
16.7	Using HTML comments	304	Appendix
16.8	Example of simple <SCRIPT>	304	
16.9	Event handling	306	
16.10	Going further with forms	312	
16.11	Summary	318	

228	17 Internationalization of the Web	319
229	17.1 Introduction	320
230	17.2 The LANG attribute to specify the language for an element	324
230		
231		
237		
243		
243		
244		
244		
246	18 Designing your Web project	329
247	18.1 Introduction	330
254	18.2 Specifying what you want	330
271	18.3 Hypertext design	331
	18.4 Writing the HTML	347
	18.5 Testing and debugging	352
	18.6 Rollout and maintenance	358
	Postscript	360
	Appendix A Examples of tags	361
281	Appendix B Alphabetical list of HTML 4 tags	370
282	Appendix C Characters	379
282	Appendix D How to name colors	385
285	Appendix E URL suffixes	388
287	Appendix F Country codes	389
	Appendix G Language codes	391
288	Appendix H Tags that did NOT make it into HTML 4	393
289	Appendix I Color on the Web	395
296	Appendix J Applet	401
298	Appendix K The Cyberia Café	407
	Glossary	417
300	Index	427
301		
301		
302		
302		
302		
303		
304		
304		
306		
312		
318		

Trademark notice

Adobe Acrobat[®], GIF89A Export[™] and PhotoShop[™] are trademarks of Adobe Systems Inc.
HyperCard[®], Macintosh[®], Newton[®] and QuickTime[®] are registered trademarks of Apple Computer Inc.

Toolbook[®] is a trademark of Assymetrix Inc.

UNIX[®] is a trademark of AT&T

RenderWare[®] is a trademark of Criterion

JPEG Transmogrifier[®] is a trademark of inTouch Technologies

Internet Explorer[®], Microsoft Assist[™], PaintBrush[™], Windows[™] and Word[™] are trademarks of Microsoft Corporation

Netscape Navigator[®] is a trademark of Netscape Communications Corporation

Home Space Builder[®] is a trademark of Paragraph

HoTMetal[®] is a trademark of SoftQuad

Enhanced Mosaic[®] is a trademark of Spyglass

Java[®] and HotJava[®] are trademarks of Sun Microsystems Inc.

WebSpace Navigator[®] is a trademark of TGS and SGI

Walkthrough Pro[®] is a trademark of Virtus

WebEdit PRO[®] is a trademark of Luckman Interactive Inc.

WebExpress[®] is a trademark of MicroVision Development Inc.

11

FORMS



Included in this chapter is information on:

- Handling forms in HTML
- Using FORM elements
- Submitting forms to the server
- Designing forms, with practical tips

Summary

To remedy various shortcomings of HTML 3.2, the following new features are included in HTML 4 forms:

- The **TABINDEX** attribute is introduced to allow explicit specification of the tabbing order when traversing from one form control to another.
- The **ACCESSKEY** attribute provides for specifying direct keyboard access to form fields.
- The new attribute **DISABLED** allows you to make a form-control initially insensitive.
- The attribute **TITLE** is added to most existing elements to be a placeholder for additional information that will be rendered when the user requests help.

- The **LABEL** element associates a label with a particular form control.
- The **FIELDSET** element groups related fields together, and, in association with a **LEGEND**, can name the group.

These two new elements allow better rendering and better interactivity. Speech-based browsers can better describe the form.

- The attribute **ONCHANGE**, in association with support for scripting languages, allows form providers to include some code checking on entered data.
- The **INPUT** element has a new attribute **ACCEPT**, that allows you to specify a list of media types or type patterns for the input.
- The **FORM** element's new attribute **ACCEPT-CHARSET** is modeled on the HTTP Accept-Charset header (see Chapter 18), and can be used to specify a list of character sets acceptable to the server.

11.1 Introduction

Server-side scripts

This chapter does not cover the creation of server-side scripts to handle incoming HTML forms data, although it briefly explains their use.

A form consists of a group of fields to fill in, and these appear on the screen when a user loads the HTML document in question. Unlike other HTML elements such as headers and paragraphs, forms are designed to enable users to send information *back* to the server, rather than just retrieve information in the usual way. When you submit the contents of a form, it is sent to the server for processing by special software. If you are not a programmer, then you need to find someone who can make your forms work for you on your server. The server side of things is outside the scope of HTML, and of this book.

A paper form contains a title, some printed instructions, and various boxes for you to fill in. Sometimes you need to write your name or various numbers in these boxes; sometimes you can just insert a single letter (like 'M' for Married) in a form-field, and sometimes all you need to do is to check the box to indicate that you have selected this item. All of these mechanisms are available in HTML forms in the shape of fields, checkboxes, radio buttons, selection menus, and so on.

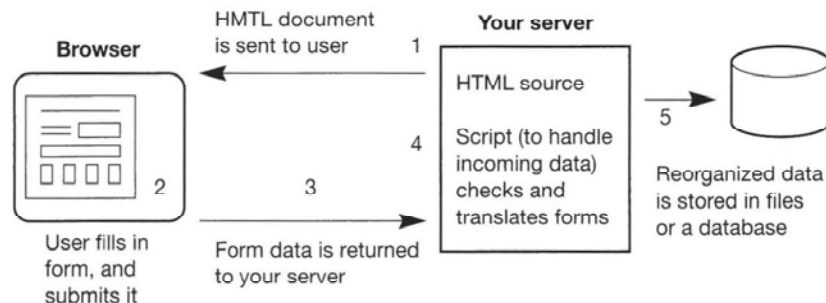
One advantage on-screen forms have over actual paper forms is that they are guaranteed to be legible, as all the words are keyed-in, rather than written in longhand.

Another and more significant advantage is speed: as soon as your readers have filled in their forms, they can press a button to send the form data straight to you, or at least, to your server. Everything they have inserted into the form is sent to your server for processing.

How forms are handled behind the scenes

As explained, the server has to run a specific program that can handle all incoming user information generated when the form is submitted. For example, the server might save the incoming data to a specific file or to a database. Or, if you are very ambitious, you might have an automatic reply service that enables your customers to perform certain functions such as booking a ticket for a forthcoming concert, for example. The program on your server could confirm bookings by sending your customers unique booking reference numbers, and then it would print the tickets, which would be soon mailed to you. You can see that many different levels of sophistication, and expense, are possible.

The diagram shows an example of what happens behind the scenes when a user completes a form.



In the example, once the data has been stored, the following sequence of events usually occurs:

1. A confirmation is quickly sent back to the user and a log of changes is recorded.
2. The system reserves the seats, prints and mails the tickets, and then records the user's address for the next mail list.

As you can see from this everyday example of an interactive form, HTML on the Web, is, in fact, a mere placid surface under which there is much activity by other systems.

11.2 Why would you want information from users?

HTML forms, then, together with server scripts, render your documents truly interactive: your users can talk back. There are many possible reasons why you might want them to do this:

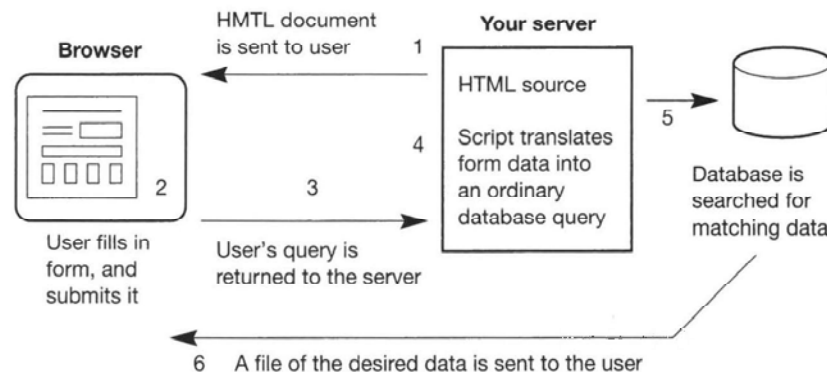
- To let users give you comments and feedback on your document or the products it describes.

- To enable users to find out more about your products or services.
- To enable prospective customers to discuss technical issues with you directly.
- To allow actual customers to order products or services from you.
- To find out who is interested in your document, ideas, products or services.
- To enable users to search for and retrieve information from other search engines.

Improved information sharing

The Internet was developed for information sharing primarily for Government and Academic purposes. Many databases full of technical information exist in relatively hard-to-access formats. But HTML forms make it quite easy for organizations with such data to display Web pages; these forms can include specific and helpful explanations of what is available, as well as the easy-to-use forms themselves that let users conduct searches and then retrieve the information they need.

The process of sharing information from an existing database is similar to the common use of HTML forms. The only significant difference is that your server needs an interface to the database; although this may require some programming, usually, it is not difficult to perform.



For example, several different authorities may hold data on Atlantic fish stocks. The full picture is available only by detailed study of many information sources. Suitable querying of the existing databases using HTML forms could improve understanding of the problems and help to secure international agreements on conservation of the remaining fish. Of course, the various authorities all need to provide their own forms. HTML offers a uniform and internationally agreed upon mechanism for this purpose, which should make things easier for everyone.

Savezip
The HTML part is explained in this chapter. Programs on the such as 'savezip' not part of HTML.

MAILTO: Using HTML without forms

The use of forms to set up a dialogue with your readers is a relatively advanced topic, and you will be able to create effective HTML documents for many purposes without ever using them. We suggest that you set up your Web pages without forms, and then carefully work out whether or not you actually need the more sophisticated interaction that forms can provide. If you only need to enable your readers to send comments, then you do not necessarily have to use forms. Instead, you can simply do the following:

- State your email address, preferably using an alias like 'Webmaster' (so that readers can write to you, or your successor in your HTML-writing job, independently), or
- Use the 'MAILTO:' keyword to connect readers' browsers to their mail tools directly.

Most browsers now handle the 'MAILTO:' mechanism, provided that their human users have remembered to fill in their email addresses and the directory paths of their mail tools. A typical use of this simple mechanism is:

```
<A HREF="MAILTO:pete@www.yourco.com">Mail to pete@www.yourco.com</A>
```

This causes a predefined mail form to appear; when filled in and submitted, an ordinary email message is sent to the address given in the HREF.

We think this is well worth knowing, as it may save you the trouble and expense of setting up the server mechanisms that you would otherwise need to handle forms. That said, email is no substitute for the power and convenience of a properly configured forms system.

11.3 Your first form

When it comes to forms handling, HTML is quite sophisticated. Luckily, you can still write a form in only a line or two. Do not worry about the details for now: the example here is just to show you that a little HTML can be very useful. The following snippet tells your server to run a script called 'savezip' when it receives the forms data, which, in this case, consists of a US postal zipcode. It might be handy for an anonymous survey, for example:

```
<FORM ACTION="savezip">
  <P>Zip or post code: <INPUT NAME="zipcode" SIZE="9">
  <P> <INPUT TYPE="submit">
```

This little piece of HTML displays an input field, with a standard Submit button:

Zip or post code:

Submit Query

Most of the other form constructs are no more complicated than this, but there are often several ways of achieving a result. For example, you can use lists to format columns of input fields, or tables, or decimal alignment: each method has its own advantages.

More sophisticated forms

Other constructs include a `FILE` attachment widget; images on `SUBMIT` and `RESET` buttons, and graphical `SELECTION` menus (using `SHAPES` within `OPTIONS`).

It is permissible to have more than one form in a document, though this may be confusing to authors and readers. Forms may not, however, be nested, unlike tables.

This chapter introduces the various form elements one by one, with worked examples of useful types of form. Other elements (such as `TABLES` and `UL` or `DL` lists), which you will often need when building forms, are incorporated in the examples, and are explained in their own chapters. Once the basic elements have been explained, the chapter concludes by showing how they can be combined in forms design to yield practical forms, ready to have scripts written for them.

11.4 HTML's form elements

Of the `FORM` element itself, there is little to write. You always put:

```
<FORM ACTION = "http://myserver.cobweb.com.doform">
... fields go here ...
... mixed with any body-content text and tables you like ...
<P> <INPUT TYPE=submit> <INPUT TYPE=reset>
</FORM>
```

The `form` tag itself can be qualified with an `HTTP METHOD` (`=post` or `=get`) attribute, to tell the browser which HyperText Transfer Protocol method the server supports. `get` is the default. For example:

```
<FORM METHOD=post ACTION="http://www.acme.co.uk/forms/register">
```

The `ACTION` attribute, as illustrated above, tells the browser where you would like the form submitted; the default is the URL of the document itself, which usually is not what you want.

The **ENCTYPE** attribute tells the browser which MIME contents type to use when encoding form data. Its default is "application/x-www-form-urlencoded" which we hope you never need to change.

The **SCRIPT** attribute lets you insert a URL for a script, which the browser can fetch and interpret. Scripts are introduced in Chapter 16.

The rest of this section presents the form elements one by one.

Simple text fields

The most general form element is **INPUT**. Its simplest use is as a simple text field. Like the other form elements, it is only allowed inside **FORM** tags. As text is the default type, you need only give the field name and size. Notice the use of ordinary formatting of text:

```
<H3> Please fill in the following details for your enquiry: </H3>
<P> Surname : <INPUT NAME="surname" SIZE="20">
```

Please fill in the following details for your enquiry:

Surname :

This shows the simplest kind of formatting, namely left-alignment hard up against the field prompt. This is less than ideal if there are several fields to consider.

When the form is submitted (the user presses the Submit button – see the section 'Submit button', below), the browser returns each field's name with its currently entered value.

An issue that you need to be aware of when designing forms is the browser window's probable width. It is nice to group related fields (for example, surname, initials, and so on), on a single line, but you have to remember that users will not necessarily see them displayed like that.

Text fields (like range and hidden fields) can be given default **VALUES**:

```
<INPUT NAME="surname" SIZE="20" VALUE="your surname">
```

These appear in the field when first displayed, or when the Reset button is pressed – see below. Two other attributes are necessary in most uses of text and password fields:

SIZE For fixed-pitch fonts (the default), the number of characters to display.
For variable-pitch fonts, the number of en units (defined as half the width of the widest character in the font) visible in the field.

MAXLENGTH The maximum number of text/password characters allowed.

Generic attributes of FORM elements

You can qualify any INPUT element, including TEXT, with several generic attributes. We will list them just once, here:

ID	Identifier unique in this document. Allows hypertext links to point to this field by referencing the identifier (this is described in Chapter 7, 'Hypertext links')
LANG	Language identifier
CLASS	To attach a style to this element
TYPE	TEXT (the default); other types include RADIO, CHECKBOX, PASSWORD and so on
NAME	Field name, returned with the field value data when submitted
VALUE	Initial value for the field

Multi-line text fields

Multi-line text was formerly handled with TEXT, but this unintentionally limited input to 1024 characters. Using TEXTAREA helps you over that hurdle, so that you can make multi-line fields as large as you need. Of course, if your server is bombarded with replies, this means you will need megabytes of free storage. The syntax is pleasantly straightforward:

```
<TEXTAREA NAME="request" ROWS=2 COLS=64>
  Please overtype this with your request.
</TEXTAREA>
```

The HTML can be read as saying:

'create a free-format input textarea called "request" with a viewport of two rows of 64 characters displayed on the browser screen. Prefill the text area with the words "Please overtype this with your request".'

It is important to understand from the above example that the input text is not limited to 2×64 characters. As the user continues typing, the text scrolls away. The specified rows and columns define what used to be called a viewport onto the text area: you are, in effect, looking through a small window into a much larger room, which can hold as much text as you please.

The result is a free-format input field like this:

You can see that the text between the `TEXTAREA` tags is used to initialize the field (compare this with the `VALUE` attribute for `INPUT TYPE=TEXT`). The scrollbars make it clear that the text is allowed to extend beyond the narrow limits of the visible text area.

A fixed-pitch font such as Courier is normally used in input areas for simplicity, but browsers can actually use proportional fonts like Times or Arial as long as they wrap text properly. Of course, width in characters does not have a definite meaning with proportional fonts.

The meaning of the size declared with `'COLS=64'` therefore depends on the type of font in use:

- **fixed-pitch font:**
declared number of columns = width in characters
- **proportional font:**
declared number of columns = width in en units

Horizontal and vertical scrollbars, as shown in the example above, are typically provided on computers with windowed environments. Text-only browsers may allow scrolling by means of the cursor keys on the keyboard.

You should always include a `NAME`, as with the `INPUT` types. Generic attributes have their usual effects.

Here is an example from the Web of a form that consists principally of a multi-line text area: