



Ordering Information:

[Internet & World Wide Web How to Program, 2/e](#)

[The Complete Internet & World Wide Web Training Course, 2/e](#)

[Wireless Internet & Mobile Business How to Program 1/e](#)

[The Complete Wireless Internet & Mobile Business Training Course](#)

- View the complete **Table of Contents:** [Internet & World Wide Web How to Program, 2/e](#) & [Wireless Internet & Mobile Business How to Program, 1/e](#)
- Read the **Preface:** [Internet & World Wide Web How to Program, 2/e](#) & [Wireless Internet & Mobile Business How to Program, 1/e](#)
- Download the **Code Examples:** [Internet & World Wide Web How to Program, 2/e](#) & [Wireless Internet & Mobile Business How to Program, 1/e](#)

To view all the Deitel products and services available, visit the Deitel Kiosk on InformIT at [www.informIT.com/deitel](http://www.informIT.com/deitel).

To follow the Deitel publishing program, sign-up now for the *DEITEL™ BUZZ ONLINE* e-mail newsletter at [www.deitel.com/newsletter/subscribeinformIT.html](http://www.deitel.com/newsletter/subscribeinformIT.html). To learn more about our [Internet and or Wireless Internet programming courses](#) or any other Deitel instructor-led corporate training courses that can be delivered at your location, visit [www.deitel.com/training](http://www.deitel.com/training) or contact our Director of Corporate Training Programs at (978) 461-5880 or e-mail: [christi.kelsey@deitel.com](mailto:christi.kelsey@deitel.com).

*Note from the Authors:* This article is an excerpt from Chapter 23, Section 23.12, of *Internet and World Wide Web How to Program, 2/e*. This is the second article in a two-part series which presents an introduction to WAP and WML. The first article introduced WAP and WML with a sample code example. Here in Part 2, we duplicate the example and incorporate the use of WMLScript. Readers should be familiar with basic markup concepts (e.g., HTML, XML), Internet Information Services (IIS) or Apache Web and basic programming concepts. The code examples included in this article show readers examples using the Deitel™ signature *LIVE-CODE™ Approach*, which presents all concepts in the context of complete working programs followed by the screen shots of the actual inputs and outputs.

## 23.12 WMLScript Programming

We now begin our introduction to the WMLScript scripting language. The relationship between WML and WMLScript is similar to that between HTML and JavaScript. However, one key difference is that WMLScript is placed in a separate document and cannot be embedded inside a WML document.



### Common Programming Error 23.4

*WMLScript is case sensitive. Failure to use the proper case is an error.*



### Software Engineering Observation 23.1

*The Openwave browser caches each deck loaded from the server. The cache is an area of a device in which the browser saves Web pages to facilitate the rapid retrieval of the pages. The Openwave browser searches for a document in the device's cache before going to the Web to access the document. When developing applications, be sure to clear the cache every time a page is changed by selecting **Clear Cache** from the **Edit** menu in the simulator.*

Our first WMLScript example provides the text “**Welcome to WMLScript Programming!**” to a WML document. Openwave's browser contains a *WMLScript interpreter* for the execution of WMLScript commands. Our first script is shown in Fig. 23.6; the associated WML document and output are shown in Fig. 23.7.



### Common Programming Error 23.5

*Placing any WMLScript code outside a function definition is an error.*

Lines 4–12 define function **welcome**. Keyword **extern** denotes that the function is externally accessible to other WML and WMLScript documents. The omission of this keyword restricts the function's visibility (or scope) to the WMLScript file in which it is defined. Functions that do not use **extern** are called *utility* or *helper* functions. These functions often contain logic that is specific to the WMLScript file. Other external documents cannot not call these functions directly.

WMLScript provides objects for performing common mathematical calculations, string manipulations, browser manipulations and other functions. These objects offer many basic capabilities that programmers need. WMLScript provides the **WMLBrowser** object for interacting with the browser. We call the **WMLBrowser** object's **setVar** method (lines 7–8) to create a *browser variable* named **welcome** and to assign it a string. Browser variables are global variables; they reside in the browser's memory and are accessible to any WML or WMLScript document residing in the browser's memory.

```

1 // Fig. 23.6: welcomeDoc.wmls
2 // Writing a line of text
3
4 extern function welcome()
5 {
6 // creating a browser variable and assigning it a value
7 WMLBrowser.setVar( "welcome",
8 "Welcome to WMLScript programming!" );
9

```

Fig. 23.6 WMLScript listing for **welcomeDoc.wmls** (part 1 of 2).

```

10 // refresh the display window
11 WMLBrowser.refresh();
12 }

```

Fig. 23.6 WMLScript listing for `welcomeDoc.wmls` (part 2 of 2).



### Common Programming Error 23.6

*Failure to terminate a WMLScript statement with a semicolon is an error.*

Line 11 calls the `WMLBrowser` object's `refresh` method to update (or refresh) the values of all browser variables. This allows the WML document that calls function `welcome` to use the browser variable's new value. In this instance, invoking the `refresh` method refreshes the browser, updating `welcome`'s value. If the variable is not refreshed, browser variable `welcome` will display an empty string when it is dereferenced in the WML document (line 19 of Fig. 23.7).



### Common Programming Error 23.7

*If a browser variable is created in a WMLScript document and control goes back to the card that referenced the function, the browser must be refreshed using the `refresh` method. If this is not done, the value of the variable will not be updated and displayed.*

The WML document in Fig. 23.7 invokes function `welcome` in `welcomeDoc.wmls` (Fig. 23.6). The result of function `welcome` is displayed by the WML document.

```

1 <?xml version = "1.0"?>
2 <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
3   "http://www.wapforum.org/DTD/wml12.dtd">
4
5 <!-- Fig. 23.7: fig23_7.wml -->
6 <!-- Printing a line of text -->
7
8 <wml>
9   <card id = "Line" title = "Line">
10
11     <onevent type = "onenterforward">
12
13       <!-- call function welcome -->
14       <go href = "welcomeDoc.wmls#welcome()" />
15
16     </onevent>
17
18     <p>
19       $welcome <!-- dereference browser variable welcome -->
20     </p>
21   </card>
22 </wml>

```

Fig. 23.7 WML document that calls function `welcome` (part 1 of 2). (Image of UP.SDK courtesy Openwave Systems Inc. Openwave, the Openwave logo, and UP.SDK are trademarks of Openwave Systems Inc. All rights reserved.)



**Fig. 23.7** WML document that calls function `welcome` (part 2 of 2). (Image of UP.SDK courtesy Openwave Systems Inc. Openwave, the Openwave logo, and UP.SDK are trademarks of Openwave Systems Inc. All rights reserved.)

Lines 11–16 contain the `onevent` element that invokes WMLScript function `welcome`. The `onevent` element is an *event element* that executes a *task element*, such as the `go` element (line 14), which is wrapped in its tags. Task elements such as `go`, `refresh` and `prev` perform certain actions when executed. A complete list of task elements can be found at [www.w3schools.com/wap/wml\\_reference.asp](http://www.w3schools.com/wap/wml_reference.asp). Attribute `type` is set to `"onenterforward"`. This executes the task element `go` when the `card` is loaded. Function `welcome` in `welcomeDoc.wmls` is invoked in line 14 by assigning to attribute `href` the WMLScript document name followed by a `#` sign and the function name (Fig. 23.6). WMLScript documents have the `.wmls` file extension and are referenced from within a WML document.



### Good Programming Practice 23.2

If the value of the `href` attribute does not include the name of the function, the first function declared using keyword `extern` is executed. Always include the function name when referencing a WMLScript file.

**Common Programming Error 23.8**

*Failure to enclose link addresses in quotes is a syntax error.*

---

**Common Programming Error 23.9**

*When referencing a function from an **href** attribute's value, failure to precede a function name with a pound sign (#) is a runtime error.*

---

Lines 18–20 mark up browser variable **welcome**'s value with **<p>** tags. The insertion of the *dollar sign* (\$) before the variable name retrieves the browser variable's value from the device's memory.