

Command-driven, event-driven, and web-based software

Web pages appear to users as *graphical, interactive applications*. Their graphical and interactive features are supported by *browsers* such as Internet Explorer and Firefox.

The JavaScript language may be used within web pages, even though it is not part of the HTML language, the native language of web pages and browsers. The JavaScript language is similar to the Java language but simpler and less powerful. The environment for testing JavaScript is the browser itself. Another programming environment is *Alice*.

I. Networked and Web computing

Computers communicate by using standard communication rules called *protocols*, such as the Internet Protocol (IP) and the Hypertext Transfer Protocol (HTTP). Under the Internet Protocol, each device communicating on the Internet must have a unique number called an *IP address*, which is used in all communication between devices. All communication is in the form of *packets*, which are relatively small items of data containing sender IP address, receiver IP address, content communicated, and link information to allow that content to be combined with content from other packets to make up a larger message or data file.

The Hypertext Transfer Protocol governs web page access. Each web page has a Universal Resource Locator (URL), or web address, part of which is a *domain name*, such as *framingham.edu*. Under HTTP, when you click on a Web link, your computer sends a message to a *domain name server* that converts the domain name to the IP address of a *web-server* computer. A message is then sent to the web server to request the web page, and the server replies by sending your computer a series of packets containing the HTML file that you have requested. Your browser then displays a formatted version of the HTML file.

Event-driven applications

Browsers and most other applications are *interactive*, alternating input and output. For example, the user may scroll or may click on hyperlinks, and many web pages have buttons or input forms. An interactive application is called *event driven*. The events are input occurrences such as mouse clicks, keypresses, motions of the mouse, the passage of time, and the arrival of data from communication ports.

Events are initiated externally to the system, as is true in client-server computing. But event-driven applications run on desktop or laptop computers whether connected to the Internet or not. Most event-driven applications have graphical user interfaces (GUIs).

Most applications, including word processors, spreadsheets, and Internet browsers, are event driven. For example, the event of clicking on a hyperlink causes the browser to request and display the HTML code found at the URL linked to by the hyperlink.

Event-driven programs use *event handlers*: code that specifies the application's response to a particular event, such as a user click on a hyperlink or button. For each type of event that is to be responded to, an event handler specifies the response.

Languages used with computers

Languages associated with computers include

- *user interfaces* such as those of Windows and Microsoft's and other office suites;
- *command-line interfaces*, such as the Windows Command Prompt;
- spreadsheet *formulas*, including algebraic expressions, functions, and conditional formulas such as ones with IF and AND;
- database *queries*;
- *machine and assembler* languages for processors;
- *markup languages* such as HTML.

II. Command-line environments

Java programs may be developed and executed in command-line environments such as the Windows Command Prompt, available via *Start / All Programs / Accessories* or *Start / Run / cmd*. The command-line window's standard format has white type against a black background. In a command-line environment, all actions are initiated by the user typing a *command*, at the *prompt*, which is normally a series of characters giving the current directory path, followed by a greater-than symbol, as in:
C:/java>

Valid commands in a Windows system include *help*, *dir*, *cd*, *mkdir*, *path=*, *del*, *copy*, and *move*. The *help* command provides guidance in using the command line. The UNIX operating system has a powerful command-line language as well.

Current directory

Any command may have *arguments* or parameters that are often the names of files in a particular directory, named in the prompt, which is the *current directory* (C:/Java in the example above). To make a subdirectory of the current directory be the current directory (to "go into" the subdirectory), type *cd* and the subdirectory name.

To step out of a subdirectory into the parent directory, type *cd* and a period, then press the *Enter* key. (All commands must be followed by *Enter*.)

To see a list of files and subdirectories in the current directory, type *dir*.

To create a new subdirectory of the current directory, type *mkdir* followed by the

Executable files

Other valid commands are the names (minus extensions) of files with the extensions *.exe* and *.bat*. The latter are executable files. Files with the *.exe* extension are in machine language. Files with the *.bat* extension are batch files, which are text files in the language of the command line. Typing the name of a batch file causes the commands in the file to be executed in the command-line environment as if the user had typed them at the prompt.

Repeating commands used earlier

The time from when you invoke the command line, to when you close it with the command *exit*, is called a *session*. To use a command that you used earlier in a session, without retyping it, scroll through the command history of your session with the up or down cursor keys. You may also edit a command using the left and right cursor keys.

The *path* command

In order to execute *.exe* or *.bat* files that are not in the current directory, these files may be made available by use of the *path* command. For example, if the Java compiler is in the directory C:\java, then typing
path=%path%;c:\java
will make the compiler available in the current directory.

Other commands

To delete a file, type *del* followed by the file name.

To make a copy of a file, type *copy*, the file name, and the name of the new file.

To rename a file or to move it to a subdirectory, type *move*, the file name and the new file name or the subdirectory name.

To edit any text file, type *notepad* and the file name. Note that in Notepad, to save a file under a new name that does not have the extension *.txt*, you must use quotation marks around the file name in the Notepad *Save As* dialog.

Uses of the command line

Three applications for the command-line interface is to run executable Java *application* programs (as opposed to Web *applets*), to *compile* Java source code into executable programs, and to copy, move, and delete files within the Windows or Unix file systems.

III. JavaScript

JavaScript is a *procedural language* that has elements of all the above languages. It is expressed in text form and specifies the operations to be carried out by the computer; it is procedural in that it supports executable commands. JavaScript is used within web pages to input data from the user visiting a web site, to perform computations on this data, and to format the results of computations.

JavaScript is a programming language with the power to support the above tasks and more. Users may understand computer software better if they have an introduction to the basic concepts of software development and programming.

We all encounter situations when computer systems don't work correctly, so we have to solve problems posed by these faults, either by correcting them or working around them. Thus every user must occasionally do what software-development and information-technology professionals do all the time: *debug* systems. Solving these problems requires *analytical* thinking as well as precise thinking about the kinds of processes computer systems carry out, including *algorithms* and *interactive processes*.

Some understanding of *design and implementation* of systems may be helpful in *using* these systems. Designs are implemented in software development using *programming languages*. One such language is JavaScript. Its advantages include the fact that JavaScript may be embedded in HTML code, in which case a Web browser will execute the JavaScript code when displaying the HTML.

A simple JavaScript example

The HTML file below contains JavaScript code within the `<script>` and `</script>` tag delimiters. To include JavaScript within an HTML file, the argument, `language="JavaScript"`, is used after the `<script>` tag name.

```
<html>  <!-- hello.htm -->
  <body>63.120 says Hello!
    <script language="JavaScript">
      alert("hello");
    </script>
  </body>
</html>
```

In the JavaScript code above, the *alert* procedure causes a dialog box to be displayed with the

message, "hello," inside it and an "OK" button to close the dialog, called an "alert box." The browser's built-in event handler causes the dialog box to disappear when the user presses the "OK" button.

A simple button

You can create a button in a browser screen and write an event handler that defines what to do (display "Hi") when the user presses a button labeled "Hello":

```
<html> <!--button.htm-->
<body>
  <input type=button value = "Hello"
    onClick = 'alert("Hi")'>
</body></html>
```

The `<input>` tag defines an input object of the button type. The event handler code, `onClick = 'alert("Hi")'`, specifies the event type, a mouse click, and the response to it.

Counting button clicks

The JavaScript code in the HTML below displays "Yes," "No," "Stats," and "Reset" buttons for the user to click, counts the number of times the user has clicked the "Yes" or "No" button, and displays these values.

```
<html>  <!-- count-yes.htm -->
<head><title>yes-no counter</title></head>
<body>
<script language="JavaScript">
  var num_yes=0, num_no=0;  // Variables
</script>
  <td><input type=button value = "Yes"
    onClick = 'num_yes = num_yes + 1'></td>
  <td><input type=button value = "No"
    onClick = 'num_no = num_no + 1'></td>
  <td><input type=button value = "Stats"
    onClick = 'alert("Yes: "+num_yes +
      "No: "+num_no)'></td>
  <td><input type=button value = "Reset"
    onClick = 'num_yes = num_no = 0'></td>
</body>
</html>
```

Four event handlers, one for each button, specify the responses to the four different button-clicking input events. When user presses the "Yes" or "No" button, the variable `num_yes` or `num_no` is incremented, using the JavaScript *assignment statement*, `num_yes = num_yes + 1`.

In JavaScript, the equal sign is an assignment operator. The same operator, in pseudocode and flowchart language, is written ←.

In *count_yes.htm*, only the declarations of the variables *num_yes* and *num_no* are enclosed by the `<script>` tag; the JavaScript statements are passed to the browser in the event-handling HTML code.

Text input/output

The code below enables the user to input a name, which is stored in the variable *user_name*; the browser then displays the name with a greeting.

```
<! Input-echo.htm>
<head><title>Input echo</title></head>
<body>
  <form name="Input"><table>
    <!-- Display prompt and get input:-->
    <td>Enter your user name:
    <!-- Generate input-box:-->
    <input type="text" name="user_name"
      value="" size = 15> </td>
    <!-- Get button press, show message:-->
    <td><input type="button" value="Done"
      onClick =
        'alert("Hello " + user_name.value)'+
      </td>
  </table></form>
</body>
```

For text input, an HTML *form* is used. The HTML event handler uses JavaScript to display an alert box. This HTML file uses JavaScript only to pass a procedure invocation to the event handler, so it contains no `<script>` tag. In the *alert* statement above, the “+” sign is used to *concatenate* the text strings to its left and right, i.e., to express a value that is the two strings put side by side.

Expressions

Data expressions are found in spreadsheet formulas and in JavaScript. They have *values*, including numeric (e.g., 2.5), character-string (e.g., “Bill”), and true/false (e.g., $a > 4$). Number, string, and truth value are *data types* recognized by JavaScript. A type is an interpretation of the meaning of bit patterns.

Expressions may be *literals*, *variables*, or *function calls*, or may be formed with arithmetic operators such as +, -, *, /, %. Truth-value (Boolean) expressions may be constructed using relational operators (==, !=, <, >, <=, >=) or logical operators: ! (not), && (and), and || (or).

Numeric operations

If a text string is input, then it cannot be used in numeric expressions unless it is first converted to a numeric data type. This is done with the JavaScript *parseInt* function. For example, if *x* is a text string, then *parseInt(x)* is a numeric value. See the example below, which inputs a number and displays the number, doubled.

```
<html> <! double-num.htm>
<head><title>Input doubler</title></head>
<body>
  <script language="JavaScript">
    var x;
  </script>
  <form name="Input"><table>
    <!-- Display prompt and get input:-->
    <td>Enter a number:
      <!-- Generate input-box:-->
      <input type="text" name="x"
        value="0" size = 6>
    </td>
    <!-- Wait for button-press and
      display message:-->
    <td><input type="button" value="Done"
      onClick = 'alert("Doubled: " +
        2 * parseInt(x.value))'+
      </td>
  </table></form>
</body>
</html>
```

Statements

Statements in JavaScript may be *executable* or may be *variable declarations*. They include *assignment* (using the ‘=’ operator), branch (*if*, *if ... else*), *loop* (*while*), and compound statements. Compound statements are surrounded by curly braces. Simple statements, such as assignments, are separated by semicolons.

Conditional statement (if)

The *if* statement, like the *if* function in Excel, makes it possible to branch in either of two directions depending on the value of a test expression. In the file below the event handler for the button labeled *Check* causes a JavaScript *if* statement to execute. The *if* statement tests the value of the expression, (*most_recent* == "A"), and if that value is true, displays a corresponding message about how the “A” button was pressed last; otherwise it displays the information that “B” was pressed last. The variable, *most_recent*, stores the name of the most recent button pressed.

```

<html><head>
/* remember.htm
Displays 2 buttons for user to click,
tells which was clicked most recently. */
<title>Yes-no with memory</title></head>
<body>
  <script language="JavaScript">
    var most_recent="A",
        message="Last you clicked was ";
  </script>
  <td><input type=button value = "A"
    onClick = 'most_recent="A"'></td>
  <td><input type=button value = "B"
    onClick = 'most_recent="B"'></td>
  <td><input type=button value = "Check"
    onClick =
      'if (most_recent == "A")
        alert(message+"A");
      else alert(message+"B")'>
  </td>
</body>
</html>

```

Loops (*while*)

Repetition can be performed with the *while* statement, as below. After *while* appears a pair of parentheses, containing a true-false expression whose value may change in the code below it. The code below that, called the *loop body*, is in braces.

```

<html> <! power.htm>
<head><title>Input exponentiator</title></head>
<body>
  <script language = "JavaScript">
    var i,y;
  </script>
  <form name="Input"><table>
    <! Display prompt and get input:>
    <td>Enter a base:
      <! Generate input-box:>
      <input type=text name=x1
        value="0" size = 8>
    </td>
    <td>Enter an exponent:
      <! Generate input-box:>
      <input type=text name=x2
        value="0" size = 8>
    </td>
    <td><input type=button value="Done"
      onClick = '
        i = parseInt(x2.value);
        y = 1;
        while(i > 0) {
          y = y * parseInt(x1.value);
          i = i - 1;
        }
        alert(x1.value + "^" + x2.value +
          " = " + y);
      '>
    </td>
  </table></form>
</body>
</html>

```

IV. Summary

Event-driven software responds to user choices, usually in graphical user interfaces (GUIs). Networked computing enables interaction with any other computer in the world. Command-line interfaces support file manipulation and support execution of Java programs. Procedural languages such as JavaScript support alert boxes, button controls, text and numeric input/output, numeric operations, branch statements, and loop statements.

References

- C. Horstmann. "Windows Shell Tutorial." http://www.cs.sjsu.edu/web_mater/cs46a/cs46ala/b/lab1/tutorial.html
- D. Keil. Slides on Web-based and event-driven software. <http://www.framingham.edu/faculty/dkeil/cs1j-0-intro.pdf>.
- L. Snyder. *Fluency with Information Technology*. Addison Wesley, 2006.