

## Introduction to programming concepts and JavaScript

The computer *user* sometimes plays the *programmer* role, as when formatting a document with named styles, writing spreadsheet formulas, and writing database queries. *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 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 helps 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.

### Procedural languages

Languages presented so far in this course include

- *user interfaces* such as those of Windows and Microsoft's and other office suites;
- 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.

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. 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.

### 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.

A browser is 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.

### 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  $\leftarrow$ .

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 *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, 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>
```