

7. JavaScript: Intro. to Scripting

JavaScript was originally developed by Netscape.

Microsoft's version is JScript.

Microsoft and Netscape have cooperated in the standardization of JavaScript/JScript by the ECMA known as ECMAScript.

We refer to JavaScript and JScript generically as JavaScript.

We start with JavaScript code in the header – this is executed before the body.

(We later look at inline scripting, JavaScript code in the body.)

Enclose the code in `<script> ... </script>` with attribute `type` with value `"text/javascript"`.

Single-line comments are introduced with `//`.

Multi-line comments are enclosed in `/* ... */`.

The browser contains a complete set of objects (with associated data and methods) that allow programmers to access and manipulate every element in an HTML document.

The `document` object represents the document currently displayed.

Method `writeln` writes a line of HTML text in the document and positions the output cursor at the beginning of the next line.

Method `write` is similar but doesn't advance the output cursor to the next line.

These methods write HTML text; the browser must still render the text.

Example:

(Here the <body> ... </body> tags are not needed.)

```
<html>

<head>
<title>First Java Script Program</title>

<script type = "text/javascript">
  document.write( "<H2>Hello " );
  document.writeln( "world!</H2>" );
</script>

</head>

<body>
</body>

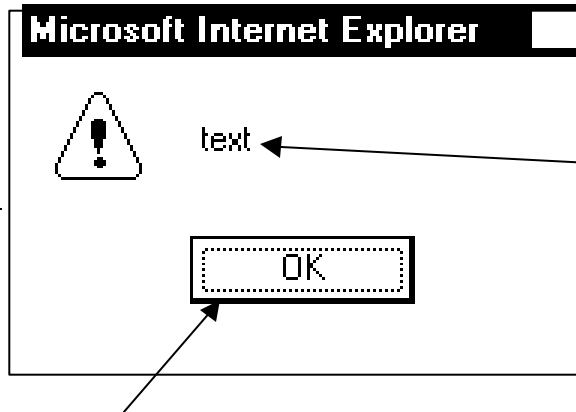
</html>
```

Hello world!

Dialog Boxes

Dialog boxes are manipulated with methods of the browser's window object.

```
window.alert("text"); – an alert dialog
```



The box is automatically sized to accommodate *text*.

Allow the user to dismiss the dialog box.

Example:

```
<html>

<head>
<title>Using a dialog box</title>

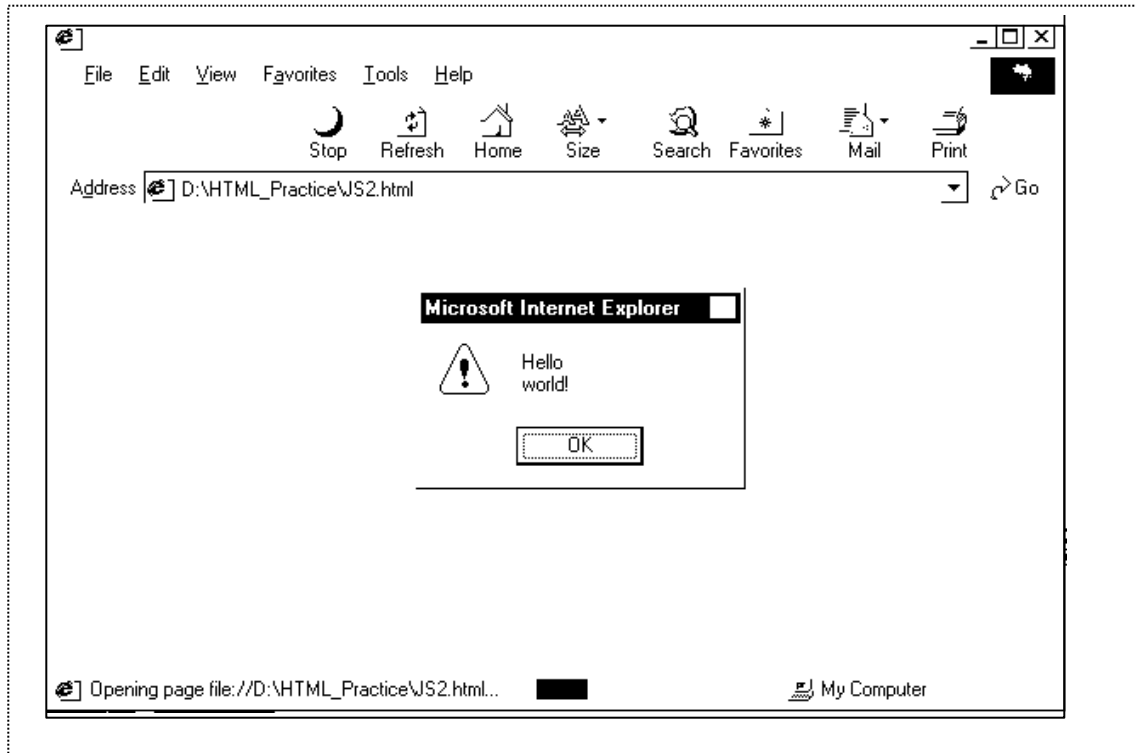
<script type = "text/javascript">
  window.alert( "Hello\nworld!" );
</script>

</head>

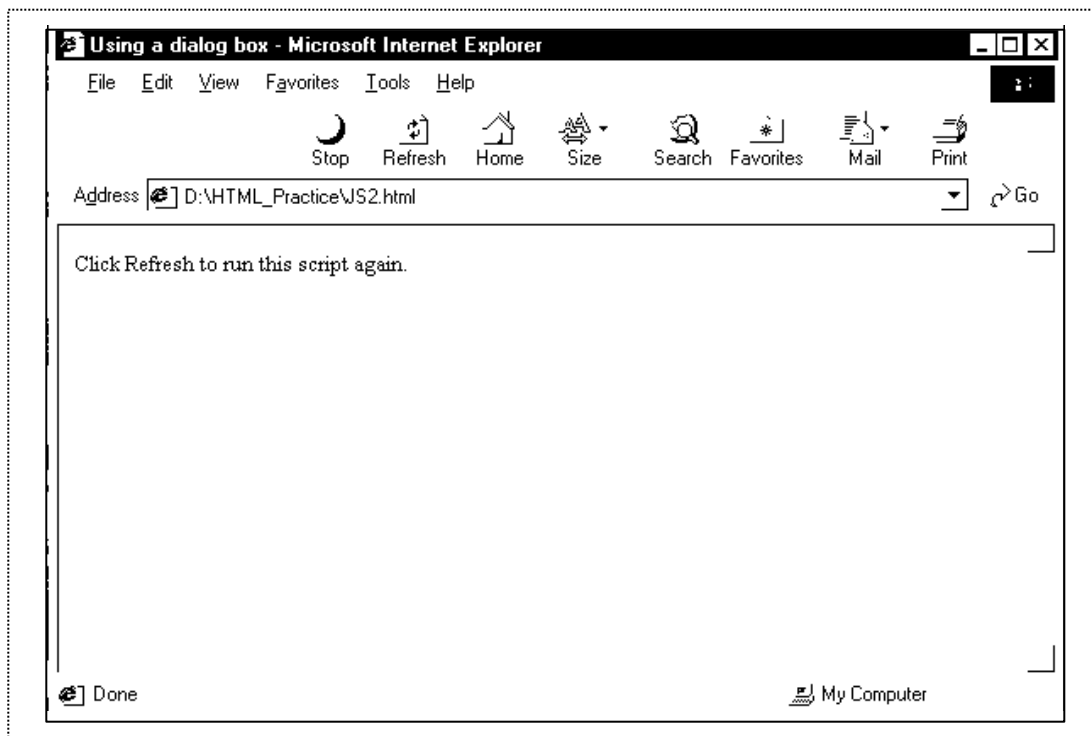
<body>
<p>Click Refresh to run this
  script again.</p>
</body>

</html>
```

When the page is first loaded:



After "OK" is clicked:



An *identifier* is a sequence of alphanumeric characters, underscores, and \$'s not beginning with a digit.

By convention, a variable name begins with a lower case letter and any embedded words begin with capitals – e.g.,

```
firstNumber
```

A *declaration*

begins with `var`,

contains a comma-separated list of variable names, and is terminated with a “;”.

You do not specify a type – JavaScript is loosely typed.

A JavaScript program may contain zero or more declarations.

Example:

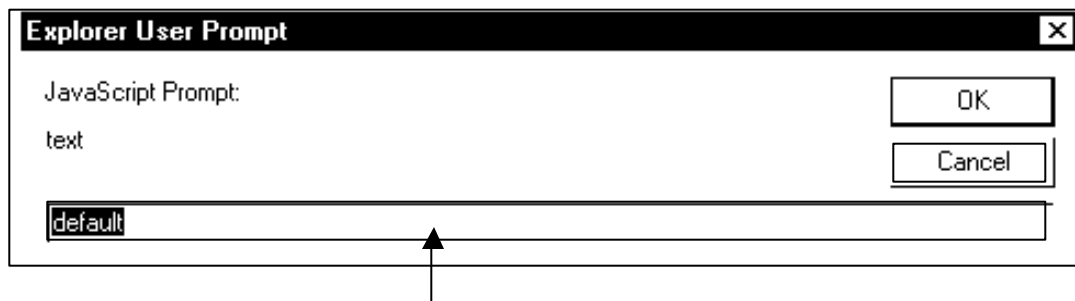
```
var  first, second,  
    third;  
var  fourth;
```

(Declarations are often optional, but their inclusion is always good style.)

The prompt dialog is another predefined dialog box of the window object.

```
vname = window.prompt( "text" , "default" );
```

Here *default* is optional



The value entered here is assigned to *vname* when **OK** is clicked or the **Enter** key is struck.

An undefined value is returned if **Cancel** is clicked or the **Esc** key is struck.

The confirm dialog is similar, but there is no text box for entering a value; instead, there's a '?' bubble.

If **OK** is clicked or the **Enter** key is struck, a true value is returned.

If **Cancel** is clicked or the **Esc** key is struck, a false value is returned.

Example:

```
if ( window.confirm( "Do it?" ) ) { ...
```

In all the dialog methods can be called as if they were functions; window is the default object in this context – e.g.,

```
if ( confirm( "Do it?" ) ) { ...
```

Javascript's arithmetic and relational operators are just like those of C++.

Javascript has built in functions for converting text to numerical values:

```
parseInt, parseFloat
```

The '+' operator is overloaded – it also is the binary string concatenation operator.

For example, suppose the value of x is 2. Then

```
document.write("<p>The value is " + x + "</p>");
```

sends the string

```
<p>The value is 2</p>
```

to the browser.

Example

```
<script type = "text/javascript">
  var firstNumber, secondNumber,
      number1, number2,
      sum;

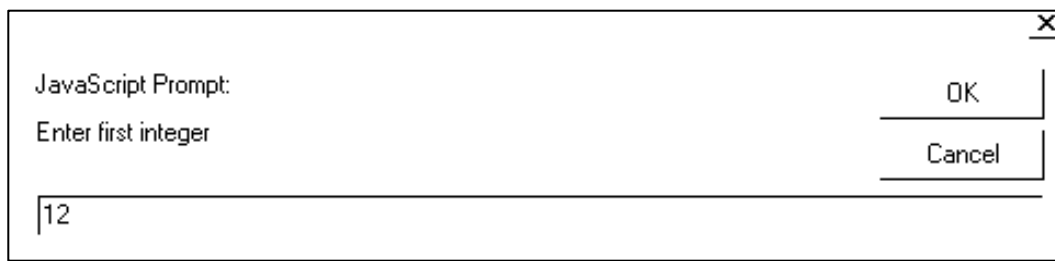
  firstNumber =
    window.prompt("Enter first integer", "0");
  secondNumber =
    window.prompt("Enter second integer", "0");

  number1 = parseInt( firstNumber );
  number2 = parseInt( secondNumber );

  sum = number1 + number2;

  document.writeln( "<H2>The sum is " +
                    sum + "</H2>" );
</script>
```

We enter 12 in the first prompt dialog box:



After clicking **OK**, the next prompt dialog box appears, we enter 23 in it, then we click **OK** again.

The second dialog box looks just like first but has as text “Enter second integer.”

Then the following is rendered in the window:

The sum is 35

It's critical to convert the text to integers.

Suppose we use the following script and again enter 12 and 23 into the dialog boxes.

```
<script type = "text/javascript">
  var firstNumber, secondNumber,
      sum;

  firstNumber =
    window.prompt("Enter first integer", "0");
  secondNumber =
    window.prompt("Enter second integer", "0");

  sum = firstNumber + secondNumber;

  document.writeln( "<H2>The sum is " +
                    sum + "</H2>" );
</script>
```

Then the rendering is

The sum is 1223

But JavaScript will convert strings to numbers (if possible) when strings are operands of other arithmetic operators. E.g.:

2 - "3" and "2" - "3" both evaluate to -1.

2 - "fred" results in a special NaN ("not a number") value.

The previous program can be written with fewer variables and more briefly:

```
<script type = "text/javascript">
  var
    number1 =
      parseInt( window.prompt(
                  "Enter first integer", "0")),
    number2 =
      parseInt( window.prompt(
                  "Enter second integer", "0"));

  document.writeln( "<h2>The sum is " +
                    (number1 + number2) + "</h2>" );
</script>
```

We can construct sophisticated HTML elements in a principled way.

Note that, when quotation marks appear within quoted text, we must use single quotes (` `), not double quotes (" ").

Example:

```

<script type = "text/javascript">
  var first  =  parseInt( window.prompt(
                        "Enter first integer:",  "0"  ) ),
      second =  parseInt( window.prompt(
                        "Enter second integer:", "0"  ) );

  document.writeln(
    "<table border = '1' width = '50%'">" );

  document.writeln(
    "<caption>Operation results</caption>" );

  document.writeln( "<tr><td>" + first + " + " +
                    second + " = " + (first + second) +
                    "</td></tr>" );

  document.writeln( "<tr><td>" + first + " - " +
                    second + " = " + (first - second) +
                    "</td></tr>" );

  document.writeln( "<tr><td>" + first + " * " +
                    second + " = " + (first * second) +
                    "</td></tr>" );

  document.write( "<tr><td>" + first + " / " +
                  second + " = " );
  if ( second == 0 )
    document.writeln( "Undefined" +
                      "</td></tr>" );
  else
    document.writeln( (first / second) +
                      "</td></tr>" );

  document.writeln( "</table>" );
</script>

```

When 100 is entered into the first prompt dialog box and 10 is entered into the second, the rendering is:

Operation results
$100 + 10 = 110$
$100 - 10 = 90$
$100 * 10 = 1000$
$100 / 10 = 10$

When 10 is entered into the first dialog box and 0 into the second, the rendering is:

Operation results
$10 + 0 = 10$
$10 - 0 = 10$
$10 * 0 = 0$
$10 / 0 = \text{Undefined}$

Separate JavaScript Files

The `script` element has an attribute `src` whose value is the name of the JavaScript file loaded with the HTML file.

A JavaScript file has extension `.js`.

The code in this file is exactly what would appear between the `<script>` and `</script>` tags.

Example

The following HTML file specifies a JavaScript file, `hello.js`.

```
<html>
<head>
<title>Example with a .js file</title>
<script type = "text/Javascript"
        src = "hello.js">
</script>
</head>
<body></body>
</html>
```

File `hello.js` is:

```
document.writeln("<p>Hello world!</p>");
```

The rendering is:

```
Hello world!
```