Tracing code tips

I suggest you read this again later after you have learned more JavaScript.

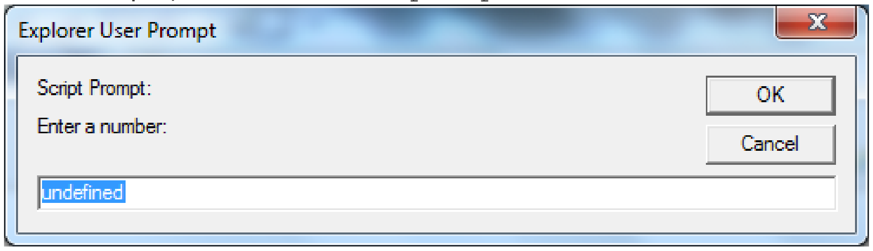
1. **Program execution** - Code execution occurs line by line, from top to bottom, except when a **control structure**[1] is encountered, such as a loop, if-then or function. Let's look at the function. The function changes this order of execution. If there is a function in the script, its code will not be executed until the interpreter encounters the line of code containing the call to the function. So, the code of the function, if listed before the call to the function, is initially skipped over.

2. The **prompt statement -** Before tracing some code, let's take a look at another JavaScript command called the prompt. (It was mentioned in the **JavaScript Notes** document and will appear in some of the sample code.) The prompt statement executes a pop-up window that prompts the user for an input value (number, string, etc.). It has similar form to that of the alert or the document.write. The form is:

   ```
   prompt("text",[default value]);
   ```

   where the text in the quotation marks will appear in the pop up window. The default value is optional. The square brackets indicate an optional component to the command. If a default value is given it will appear as the user's input value in the input box.

   For example, the statement: `prompt("Enter a number: ");` appears as:

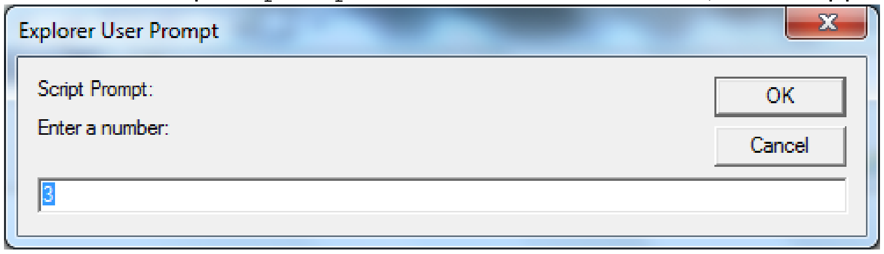


Script Prompt:
Enter a number:
undefined

   In the example above, the user will replace "undefined" with his chosen number, say 8 and then click OK or press <enter>. Furthermore the 8 may be "*captured*", and stored in variable x with the assignment statement:

   ```
   x = prompt("Enter a number: ");
   ```

   Another example: `prompt("Enter a number: ", 3);` appears as:



Script Prompt:
Enter a number:
3

   The default value 3, appears in advance in the pop-up window and will be used if the user doesn't input any other value in its place. A zero is used if the user clicks Cancel.

3. **Tracing program instructions**. It is helpful to systematically follow or *trace* the code execution, using pencil and paper, until it becomes familiar to you.

[1] A control structure is a structure that does not execute in the ordinary manner. That is, it does not necessarily execute line by line -- top to bottom, left to right. It is said that the "control" of execution of code changes. For example, a line(s) of code might be skipped over. Or some lines of code may be repeated until a condition is met.

A suggested method. (There is no set way to do this, and you may find a better method.) Make a table to trace the code, labeling each column with a different variable used in the program. It is convenient to list them in the order of their appearance in the program. List a column or separate area on your page for output too. In completing the trace table, list in its column the value assigned to that variable. Write these values as they occur and change, left to right and top to bottom.

Use a systematic approach when the code is too complex to follow in your head or giving you problems. Think: *"I are the interpreter.", "I am the machine."*

Code sample to trace:

```
1       <head>
2       <script>
3       // a function:

4       function myFunc(num) {
5          return (num/2);
6       }

7       var x = 8;
8       var y = 3;
9       var z;
10      x = 2;    // 2 overwrites 8
11      x = 6;    // 6 overwrites 2
12      z = x*y;
13      document.write("my answer is : " + z);
14      var k = myFunc(14);
15      alert(k);

16      </script>
17      </head>
```

This function code executes in line 14, where the function is called with *"myFunc(14);"*

x	**y**	**z= x*y**	**k = myFunc(num)**	**Output on page**
~~8~~	3			
~~2~~				
6		~~6*3~~ 18		my answer is: 18
	myFun(14)		~~myFunc(14)~~ 7	In a popup window appears: 7

myFunc(num)

return (14/2)
return 7

The code for the above can be found in this same folder in the *tracing code example* Folder and is called **trace_ex.htm**.