JavaScript

JAVASCRIPT

- **JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more.**
- **JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Mozilla, Firefox, Netscape, Opera.**

WHAT IS JAVASCRIPT?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language (a scripting language is a lightweight programming language)
- A JavaScript consists of lines of executable computer code
- A JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Why Study JavaScript?

- JavaScript is one of the **3 languages** all web developers **must** learn:
    - **HTML** to define the content of web pages
    - **CSS** to specify the layout of web pages
    - **JavaScript** to program the behavior of web pages

Are Java and JavaScript the Same?

- NO!
- Java and JavaScript are two completely different languages in both concept and design!
- Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

JavaScript vs. PHP

- similarities:
  - both are interpreted, not compiled
  - both are relaxed about syntax, rules, and types
  - both are case-sensitive
  - both have built-in regular expressions for powerful text processing

JavaScript vs. PHP

- differences:
  - JS is more object-oriented
  - JS focuses on user interfaces and interacting with a document; PHP is geared toward HTML output and file/form processing
  - JS code runs on the client's browser; PHP code runs on the web server

JS <3

JavaScript Capabilities

- Improve the user interface of a website
- Make your site easier to navigate
- Easily create pop-up alert, windows
- Replace images on a page without reload the page
- Form validation
- Many others …

How to Put a JavaScript Into an HTML Page?

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

Embedding JavaScript

```
<html>
<head>
<title>First JavaScript Program</title>
</head>
<body>
<script language="JavaScript"
   src="your_source_file.js"></script>
</body>
</html>
```

Hide JavaScript from incompatible browsers

```
<script language="JavaScript">
<!- begin hiding JavaScript
// single-line comment, /* … */ multiple-line
   comment
End hiding JavaScript -->
</script>
<noscript>
Your browser does not support JavaScript.
</noscript>
```

Ending Statements With a Semicolon?

- With traditional programming languages, like C++ and Java, each code statement has to end with a semicolon (;).
- Many programmers continue this habit when writing JavaScript, but in general, semicolons are **optional**! However, semicolons are required if you want to put more than one statement on a single line.

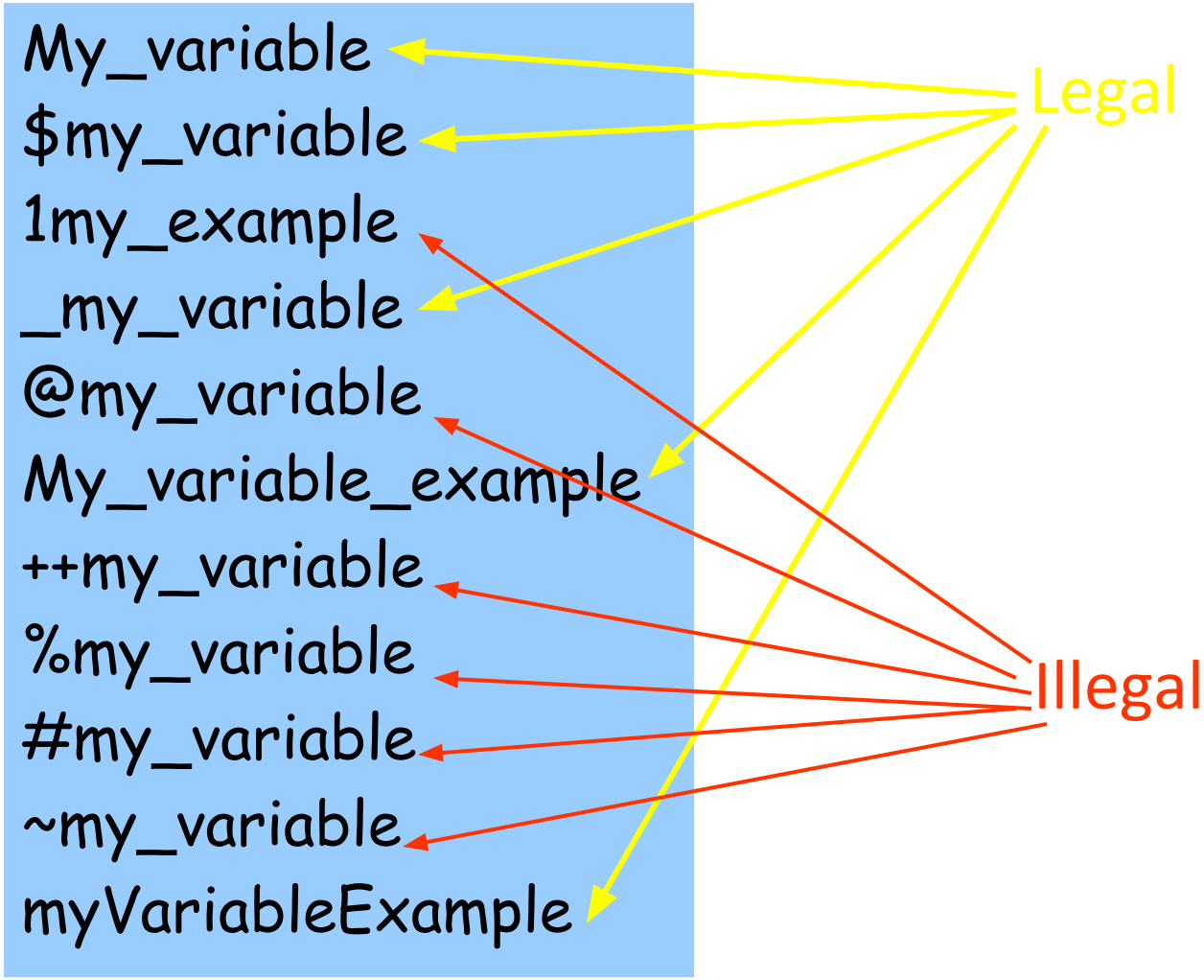
JavaScript Variables

- Variables are used to store data.
- A variable is a "container" for information you want to store. A variable's value can change during the script. You can refer to a variable by name to see its value or to change its value.
- Rules for variable names:
  - Variable names are case sensitive
  - They must begin with a letter or the underscore character
    - strname – STRNAME (not same)

Variables

- JavaScript allows you to declare and use variables to store values.
- How to assign a name to a variable?
  - Include uppercase and lowercase letters
  - Digits from 0 through 9
  - The underscore _ and the dollar sign $
  - No space and punctuation characters
  - Case-sensitive
  - No reserved words or keywords

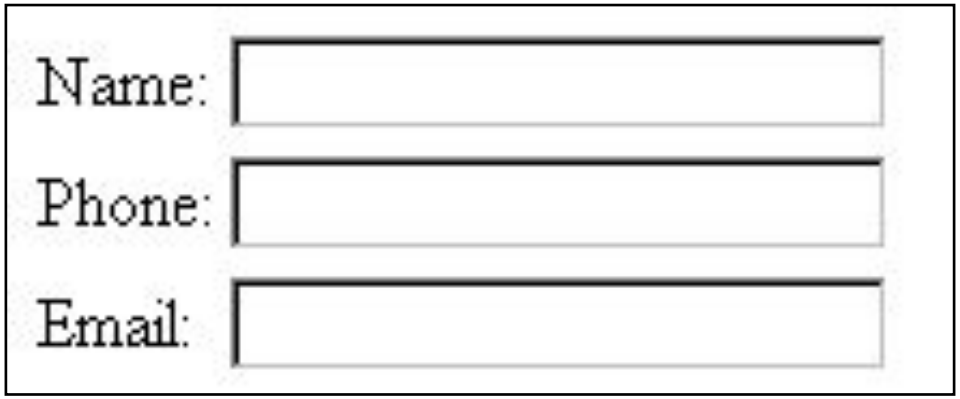
Which one is legal?



HTML Forms and JavaScript

- JavaScript is very good at processing user input in the web browser
- HTML `<form>` elements receive input
- Forms and form elements have unique names
  - Each unique element can be identified
  - Uses JavaScript Document Object Model (DOM)

Naming Form Elements in HTML



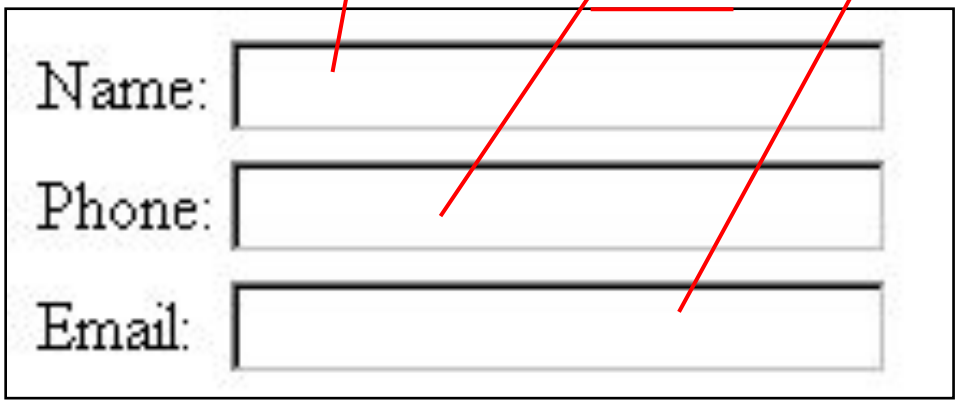

```
<form name="addressform">
Name:  <input name="yourname"><br />
Phone: <input name="phone"><br />
Email: <input name="email"><br />
</form>
```

Forms and JavaScript

*document.**formname.elementname.**value*

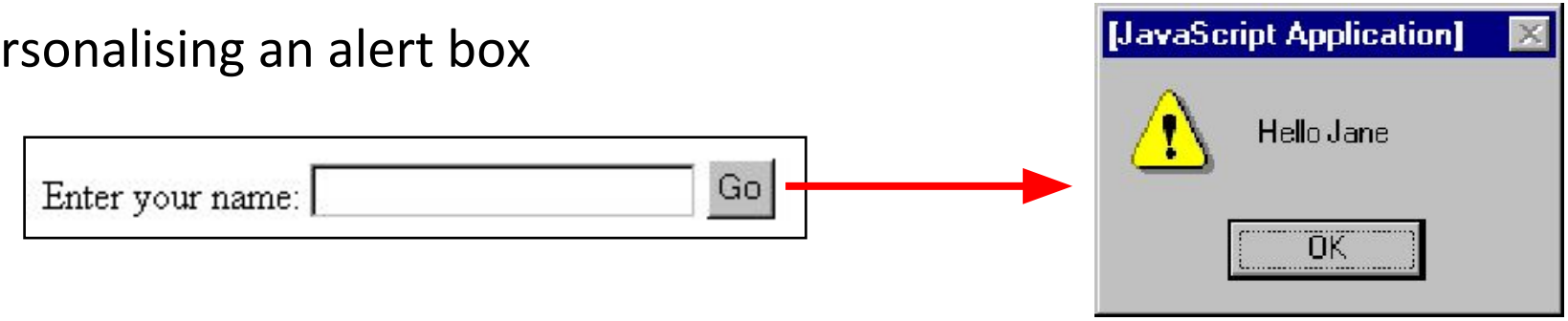
Thus:

```
document.addressform.yourname.value
document.addressform.phone.value
document.addressform.email.value
```



Using Form Data

Personalising an alert box




```
<form name="alertform">
Enter your name:
<input type="text" name="yourname">
<input type="button" value= "Go"
  onClick="window.alert('Hello ' + →
  document.alertform.yourname.value);">
</form>
```

JavaScript Operators

Arithmetic Operators

(İşleçler, iki ya da daha fazla değer üzerinde işlem yapılmasını sağlar. JavaScript içinde aritmetik ve hesaplama işleçleri olmak üzere iki tür işleç kullanılır)

Operator	Description	Example	Result
+	Addition	x=2 y=2 x+y	4
-	Subtraction	x=5 y=2 x-y	3
*	Multiplication	x=5 y=4 x*y	20
/	Division	15/5 5/2	3 2,5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

JavaScript Operators – 2

Assignment Operators

(Atama deyimi (=), bir değişkene bir değerin atanmasını sağlar. Değişkenlere türlerine ve tanımlamalarına uygun olan herhangi bir değer atanabilir.)

Operator	**Example**	**Is The Same As**
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

JavaScript Operators - 3

Comparison Operators

(Karşılaştırma işleci, iki ya da daha çok değeri birbiriyle karşılaştırarak True ya da False olarak mantıksal bir değer döndürür.)

Operator	Description	Example
==	is equal to	5==8 returns false
===	is equal to (checks for both value and type)	x=5 y="5" x==y returns true x===y returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

JavaScript Operators - 4

Logical Operators

(İkili işleçler birden çok karşılaştırma işlemini tek bir koşul ifadesi olarak birleştirirler.)

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

JavaScript Can Change HTML Content

- One of many JavaScript HTML methods is **getElementById()**.
- This example uses the method to "find" an HTML element (with id="demo") and changes the element content (**innerHTML**) to "Hello JavaScript":

```
<!DOCTYPE html>
<html>
<body>

<h1>What Can JavaScript Do?</h1>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick="document.getElementById('demo').innerHTML =
'Hello JavaScript!'">Click Me!</button>

</body>
</html>
```

JavaScript Can Change HTML Attributes

- This example changes an HTML image by changing the src (source) attribute of an <img> tag:

```
<button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Turn
on the light</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Turn
off the light</button>
```

JavaScript Can Change HTML Styles (CSS)

- Changing the style of an HTML element, is a variant of changing an HTML attribute:

```
<p id="demo">JavaScript can change the style of an HTML element.</p>

<button type="button"
onclick="document.getElementById('demo').style.fontSize='35px'">Click Me!
</button>
```

What Can JavaScript Do?

JavaScript can change the style of an HTML element.

Click Me!

What Can JavaScript Do?

JavaScript can change the style of an HTML element.

Click Me!

JavaScript Can Hide HTML Elements

- Hiding HTML elements can be done by changing the display style:

```
<p id="demo">JavaScript can hide HTML elements.</p>

<button type="button"
onclick="document.getElementById('demo').style.display='none'">Click Me!
</button>
```

What Can JavaScript Do?

JavaScript can hide HTML elements.

Click Me!

What Can JavaScript Do?

Click Me!

JavaScript Functions and Events

- A JavaScript **function** is a block of JavaScript code, that can be executed when "asked" for.
- For example, a function can be executed when an **event** occurs, like when the user clicks a button.

```
<head>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>

<body>

<h1>JavaScript in Head</h1>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

</body>
```

External JavaScript

- Scripts can also be placed in external files:
- External scripts are practical when the same code is used in many different web pages.
- JavaScript files have the file extension **.js**.
- To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

```
<h1>External JavaScript</h1>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<p><strong>Note:</strong> myFunction is stored in an external file called
"myScript.js".</p>

<script src="myScript.js"></script>
```

External JavaScript Advantages

- Placing JavaScripts in external files has some advantages:
  - It separates HTML and code
  - It makes HTML and JavaScript easier to read and maintain
  - Cached JavaScript files can speed up page loads

JavaScript Output

- JavaScript does NOT have any built-in print or display functions.
- JavaScript Display Possibilities
  - JavaScript can "display" data in different ways:
    - Writing into an alert box, using **window.alert()**.
    - Writing into the HTML output using **document.write()**.
    - Writing into an HTML element, using **innerHTML**.
    - Writing into the browser console, using **console.log()**.

JavaScript: Object-Based Language

- There are three object categories in JavaScript: Native Objects, Host Objects, and User-Defined Objects.
  - Native objects: defined by JavaScript.
    - String, Number, Array, Image, Date, Math, etc.
  - Host objects : supplied and always available to JavaScript by the browser environment.
    - window, document, forms, etc.
  - User-defined objects : defined by the author/programmer
- Initially, we will use host objects created by the browser and their methods and properties

JavaScript Objects

- You define (and create) a JavaScript object with an object literal:

```
<p>Creating a JavaScript Object.</p>

<p id="demo"></p>

<script>
var person = {
    firstName : "John",
    lastName  : "Doe",
    age       : 50,
    eyeColor  : "blue"
};

document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>
```

Creating a JavaScript Object.

John is 50 years old.

What can JavaScript Do?

- Event handlers can be used to handle, and verify, user input, user actions, and browser actions:
  - Things that should be done every time a page loads
  - Things that should be done when the page is closed
  - Action that should be performed when a user clicks a button
  - Content that should be verified when a user inputs data
- Many different methods can be used to let JavaScript work with events:
  - HTML event attributes can execute JavaScript code directly
  - HTML event attributes can call JavaScript functions
  - You can assign your own event handler functions to HTML elements
  - You can prevent events from being sent or being handled

JavaScript Data Types

- In JavaScript there are 5 different data types that can contain values:
  - string
  - number
  - boolean
  - object
  - function
- There are 3 types of objects:
  - Object
  - Date
  - Array
- And 2 data types that cannot contain values:
  - null
  - undefined

JavaScript Popup Boxes

- Alert Box
  - An alert box is often used if you want to make sure information comes through to the user.
  - When an alert box pops up, the user will have to click "OK" to proceed.

```
<script>
alert(“Image is too large!")
</script>
```

JavaScript Popup Boxes - 2

- Confirm Box
  - A confirm box is often used if you want the user to verify or accept something.
  - When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
  - If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

JavaScript Popup Boxes - 2

- var r = confirm("Press a button");
  if (r == true) {
  x = "You pressed OK!";
  } else {
  x = "You pressed Cancel!";
  }

JavaScript Popup Boxes - 3

- Prompt Box
  - A prompt box is often used if you want the user to input a value before entering a page.
  - When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
  - If the user clicks "OK“, the box returns the input value. If the user clicks "Cancel“, the box returns null.

Prompt Box Example

Syntax:

window.prompt("*sometext*","*defaultText*");

Example:

```
var person = prompt ("Please enter your name", "Harry Potter");
    if (person != null) {
       document.getElementById("demo").innerHTML = "Hello " +
    person + "! How are you today?";
    }
```

Three methods

```
<script language="JavaScript">
alert("This is an Alert method");
confirm("Are you OK?");
prompt("What is your name?");
prompt("How old are you?","20");
</script>
```

JS Examples -1

```
<script>
x=3
y=20*x+12
alert(y)
</script>
```

Examples -2

```
<script>
s1=12
s2=28
sum=s1+s2
document.write(“the sum is: "+sum)
</script>
```

Conditional Statements

- Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- **if statement** - use this statement if you want to execute some code only if a specified condition is true
- **if...else statement** - use this statement if you want to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement if you want to select one of many blocks of code to be executed
- **switch statement** - use this statement if you want to select one of many blocks of code to be executed

Conditional Statements - 2

if (*condition*)
{
code to be executed if condition is true
}

---

if (*condition*)
{
code to be executed if condition is true
}
else
{
code to be executed if condition is not true
}

Dynamic Pages

- A script can adapt the content based on explicit input from the user or other information
  - System clock: Time of day
  - Hidden input
  - Cookies
- User input can be collected by invoking the prompt method of a window object
  - This will display a dialog box that prompts user for input

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig. 7.6: welcome5.html -->
<!-- Using Prompt Boxes      -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Using Prompt and Alert Boxes</title>

      <script type = "text/javascript">
         <!--
         var name; // string entered by the user

         // read the name from the prompt box as a string
         name = window.prompt( "Please enter your name", "GalAnt" );

         document.writeln( "<h1>Hello, " + name +
            ", welcome to JavaScript programming!</h1>" );
         // -->
      </script>
```

- JavaScript is a ***loosely typed*** language. Variables take on any data type depending on the value assigned.
- Value returned by the `prompt` method of the `window` object is assigned to the variable name
- "+" symbol can be used for text concatenation as well as arithmetic operator

```
23
24      </head>
25
26      <body>
27      <p>Click Refresh (or Reload) to run this script again.</p>
28      </body>
29  </html>
```

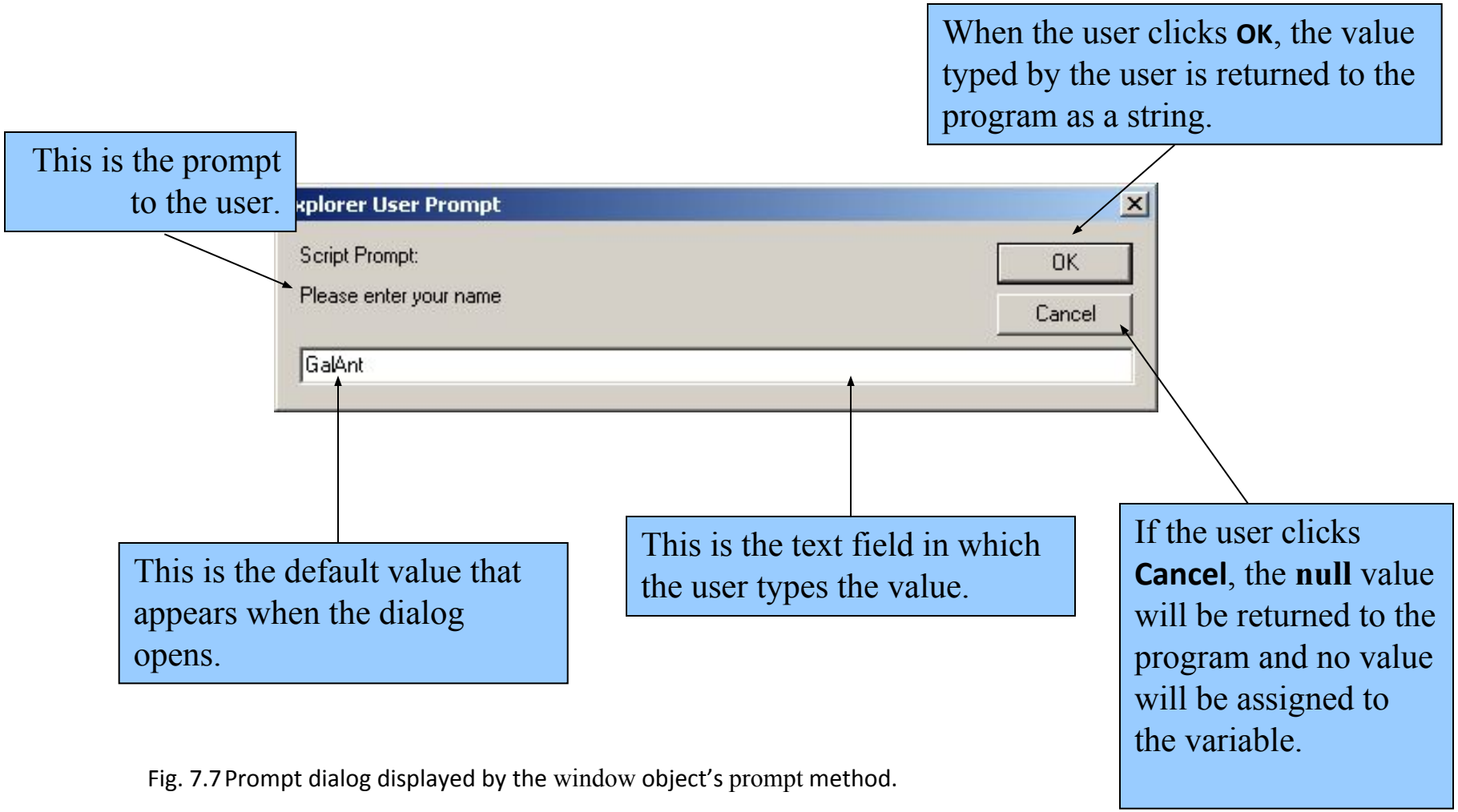


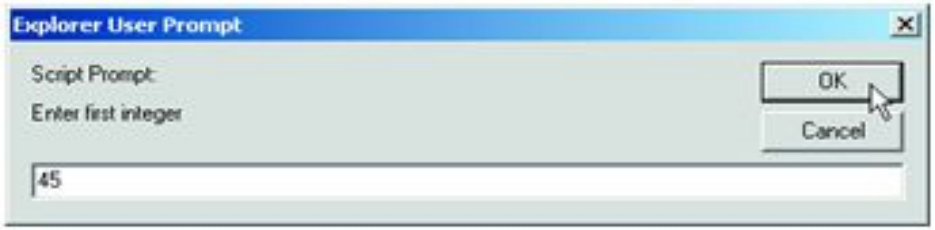
Fig. 7.7 Prompt dialog displayed by the window object's prompt method.

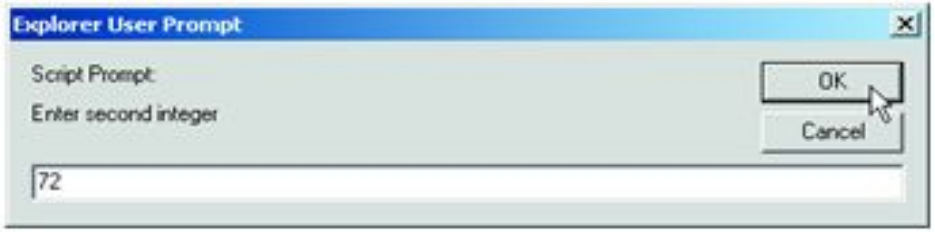
7.3.2 Adding Integers

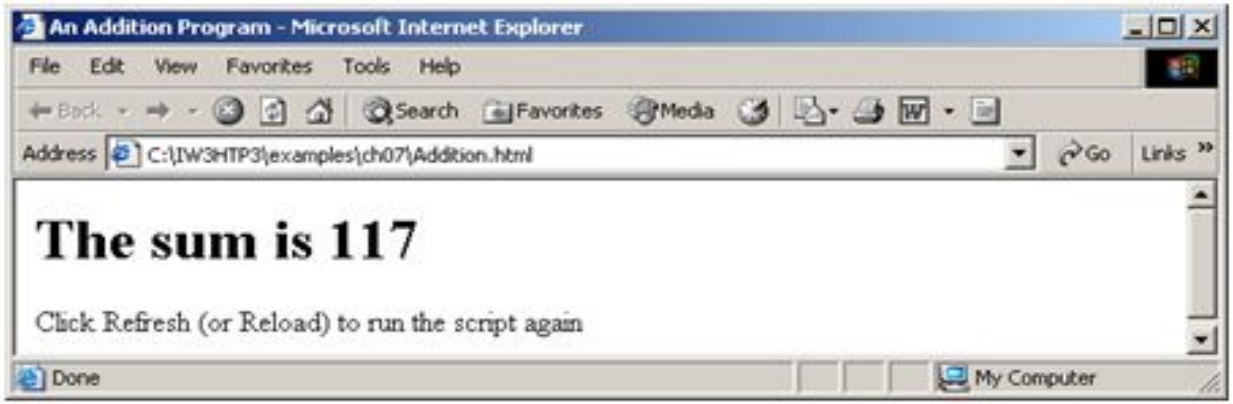
- Prompt user for two integers and calculate the sum (Fig. 7.8)
- `NaN` (not a number)
- `parseInt`
  - Converts its string argument to an integer

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 7.8: Addition.html -->
6  <!-- Addition Program        -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>An Addition Program</title>
11
12       <script type = "text/javascript">
13          <!--
14          var firstNumber,   // first string entered by user
15              secondNumber,  // second string entered by user
16              number1,       // first number to add
17              number2,       // second number to add
18              sum;           // sum of number1 and number2
19
20          // read in first number from user as a string
21          firstNumber =
22             window.prompt( "Enter first integer", "0" );
23
```

```
24          // read in second number from user as a string
25          secondNumber =
26             window.prompt( "Enter second integer", "0" );
27
28          // convert numbers from strings to integers
29          number1 = parseInt( firstNumber );
30          number2 = parseInt( secondNumber );
31
32          // add the numbers
33          sum = number1 + number2;
34
35          // display the results
36          document.writeln( "<h1>The sum is " + sum + "</h1>" );
37          // -->
38       </script>
39
40    </head>
41    <body>
42       <p>Click Refresh (or Reload) to run the script again</p>
43    </body>
44 </html>
```


Explorer User Prompt
Script Prompt:
Enter first integer
OK
Cancel
45


Explorer User Prompt
Script Prompt:
Enter second integer
OK
Cancel
72


An Addition Program - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch07\Addition.html
Go Links
The sum is 117
Click Refresh (or Reload) to run the script again
Done
My Computer

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 7.16: welcome6.html    -->
6  <!-- Using Relational Operators -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Using Relational Operators</title>
11
12       <script type = "text/javascript">
13          <!--
14          var name, // string entered by the user
15             now = new Date(),       // current date and time
16             hour = now.getHours(); // current hour (0-23)
17
18          // read the name from the prompt box as a string
19          name = window.prompt( "Please enter your name", "GalAnt" );
20
21          // determine whether it is morning
22          if ( hour < 12 )
23             document.write( "<h1>Good Morning, " );
24
```

"now" is a new instance of JavaScript native object Date. It can invoke all the methods of that object class

Note that conversion to integer type was not needed when the value was returned by the getHours method

```
25          // determine whether the time is PM
26          if ( hour >= 12 )
27          {
28             // convert to a 12 hour clock
29             hour = hour - 12;
30
31             // determine whether it is before 6 PM
32             if ( hour < 6 )
33                document.write( "<h1>Good Afternoon, " );
34
35             // determine whether it is after 6 PM
36             if ( hour >= 6 )
37                document.write( "<h1>Good Evening, " );
38          }
39
40          document.writeln( name +
41             ", welcome to JavaScript programming!</h1>" );
42          // -->
43       </script>
44
45    </head>
46
```

```
47    <body>
48       <p>Click Refresh (or Reload) to run this script again.</p>
49    </body>
50    </html>
```

(2 of 2)