JavaScript

What is JavaScript?

- HTML is undoubtedly the most widely used technology on the World Wide Web.
- It is the backbone on which all Web pages are based.
- However, HTML has limitations, so over time, developers have created a wide variety of other technologies to enhance and extend the basic capabilities of standard HTML.
- One of these widely used companion technologies is JavaScript.
- JavaScript provides Web developers with tools to add dynamic functionality and interactivity to static Web pages.

What is JavaScript?

- JavaScript is a widely used scripting language originally developed by Netscape for use within HTML Web pages.
- The language is becoming an international standard, approved by the European standards body ECMA (ECMA-262) in 1997 and later by the ISO in 1998.
- Client-side JavaScript is used widely and supported well by major browsers including Internet Explorer, Mozilla, Opera etc.
- JavaScript programs are not written in Java which is actually another programming language.

What is Scripting language

- Scripting language is a language that is interpreted by another program at runtime rather than compiled by the computer's processor as other programming languages such as C and C++.

- Scripting languages, which can be embedded within HTML, commonly are used to add functionality to a Web page, such as different menu styles or graphic displays.

What is Scripting language

- There are two main categories of scripting language - either client or server-based.
- These are designed to describe attributes and functions that can be interpreted by browsers to produce a Web page.
- Client-side scripting languages, affects the data that the end user sees in a browser window.
- Server-side scripting languages is used to manipulate the data, usually in a database, on the server.
- Some examples of client side scripting language are JavaScript, VBScript.
- Some examples of server side scripting language are PHP, Perl.

Difference Between Scripting language and Programming Language

- Both types of languages must be converted from a human-readable form to a machine-readable form.
- For programming languages, the process is performed before the program runs by a specialized piece of software called a compiler.
- The programmer controls this conversion process.

Difference Between Scripting language and Programming Language

- With a scripting language, however, there is no need for the programmer to explicitly initiate the code-conversion process. It happens automatically when the source code is processed by the target program.
- When HTML document contains embedded JavaScript code, that code is interpreted by the browser and converted into its machine-readable form when the page is loaded.

Embedding JavaScript in a HTML Web Page

- A JavaScript program usually consists of various code parts in a web page.
- Where and how to place code in a Web page depend on the purpose of the code.
- Generally, Java script code is placed within the <script> HTML tags in a web page.
- Any number of script elements can be placed in head, body, block and inline elements.

Embedding JavaScript in a HTML Web Page

- Place a script in the HTML head element unless it generates document content.
- The script tag takes two important attributes:
  1. **Language:** This attribute specifies what scripting language you are using. Typically, its value will be JavaScript.
  2. **Type:** This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript“.

Embedding JavaScript in a HTML Web Page

JavaScript Syntax:

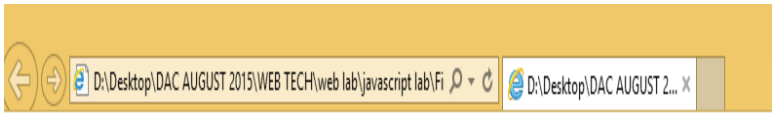
```
<script language="javascript" type="text/javascript">

    JavaScript code

</script>
```

First JavaScript Example:

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



Hello World!

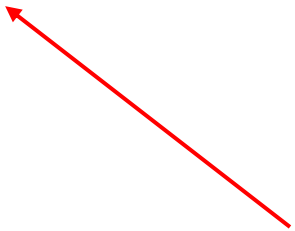
In Javascript a string literal (Character value) is given inside double quotes. The write method of the document object inserts the given string into the HTML code of the document where the <script> element is located

JavaScript Example:

```
<html>
<head><title>My Page</title></head>
<body>
<script language="JavaScript">

document.write('<h1>This is my first →
JavaScript Page</h1>');

</script>
</body>
</html>
```



HTML written inside JavaScript

Embedding JavaScript in a HTML Web Page

- There is a flexibility given to include JavaScript code anywhere in an HTML document.
- There are following most preferred ways to include JavaScript in your HTML file.

❑ **Script in <head> </head> section:**

- Code for defining functions or creating data structures are placed in <script> elements

❑ **Script in <body> </body> section**:

- Code for generating HTML to be displayed as part of the page are placed in <script> elements inside the <body> where the generated code will be inserted.
- Code for actions triggered by events are given as values of event attributes of HTML tags.
- The code is usually a function call or a few short statements.
- The onfocus, onblur, onclick, onmouseover, and onmouseout are examples of event attributes.

Embedding JavaScript in a HTML Web Page

- **JavaScript Code in External File:**

- You can use the SRC attribute of the <SCRIPT> tag to call JavaScript code from an external text file.
- This is useful if you have a lot of code or you want to run it from several pages, because any number of pages can call the same external JavaScript file.
- This external text file should be saved with .js extension.
- The text file itself contains no HTML tags.

Embedding JavaScript in a HTML Web Page

- The external JavaScript code file can be attached to a Web page by using the following tag in <head>..</head> section:

```
<script type="text/javascript" src="filename.js"> </script>
```

How JavaScript Code works?

- It is easy for a Web browser to detect whether a particular Web page contains embedded JavaScript code.
- <script> tag is used to mark the beginning of a JavaScript section and the </script> tag to indicate the end of that section.
- The Web browser will interpret everything between these two tags as JavaScript source code rather than standard HTML text.
- The browser will then convert the script into its equivalent machine-readable form called *binary code.*
- *This binary code will* then be executed, and its output (if any) will be inserted into the HTML text stream and displayed as if it had been typed into the original HTML document by a person.

JavaScript objects & JavaScript methods

- JavaScript is essentially made up of a number of invisible entities called **objects** that contain a well-defined set of capabilities.

- For JavaScript programmers to make use of these capabilities, they must call upon the services of one or more specialized functions known as **methods** within those objects.

How to Use JavaScript objects & methods

- The programmer invokes the services of JavaScript methods by entering the name of the object, followed by a period (the . character), followed by the method name.
- Method names are always followed by a parameter list, even though the list is sometimes empty.
- The parameter list simply provides the method with the information it needs to perform its function correctly.
- The syntax of the parameter list consists of an opening parenthesis, zero or more parameter items, and a closing parenthesis.

How to Use JavaScript objects & methods

Example:

To invoke the write method of the JavaScript object called "document," following line of code will be written:

```
document.write("A string of text.");
```

JavaScript Properties

- Properties are object attributes.
- Object properties are defined by using the object's name, a period, and the property name.
- **Example:** Background color is expressed by:

document.bgcolor

Here, document is the object and bgcolor is the property.

Writing JavaScript code

- JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs.
- Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if your statements are each placed on a separate line.
- JavaScript is a case-sensitive language .

Comments in JavaScript

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also treats HTML comment opening sequence <!-- as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

JavaScript Data Types

- There are three primitive data types in JavaScript:
  1. Numbers : 1234, 122.50 etc.
  2. Strings : "text string" etc.
  3. Boolean: True or False.

JavaScript Keywords

- JavaScript Keywords are reserved words in JavaScript .
- JavaScript cannot be used as JavaScript variables, functions, methods or any object names.
- The following are reserved words in JavaScript

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

JavaScript Operators

- JavaScript has various types of operators. Some of them are given below:
  - Arithmetic operators
  - Comparison operators
  - Assignment operators
  - Logical operators
  - String operators
  - Special operators

JavaScript Operators - Arithmetic Operators

Operator	Description	Example	Result
+	Addition	2+2	4
-	Subtraction	5-2	3
*	Multiplication	4*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus. Returns the remainder after the first division Example 1: 5/2 results in 2, remainder 1 Example 2: 10/8 results in 1, remainder 2	5%2 10%8	1 2
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

JavaScript Operators - Comparison Operators

Operator	Description	Example
==	is equal to	5==8 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 - Assignment Operators

- The basic assignment operator is equal (=), which assigns the value of its right operand to its left operand. That is, x = y assigns the value of y to x.

Assignment Shorthand Operators

Operator	Example	Is The Same As
+=	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 - Logical Operators

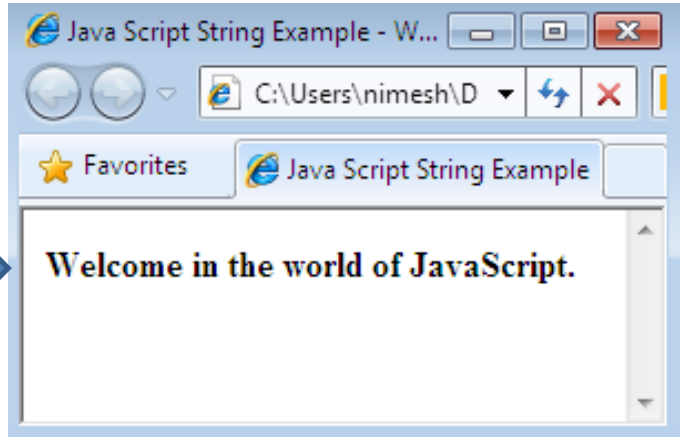
Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 x != y returns true

String Operator

- A string is most often a text, for example "Hello World!".
- The **+** operator can also be used to add string variables or text values together.
- To add a space between the two strings, insert a space into one of the strings or insert a space into the expression.

String Operator - Example

```
<HTML>
<HEAD> <TITLE> JavaScript String Example
    </TITLE>
</HEAD>
<BODY>
<b> <script type="text/javascript">
txt1="Welcome in the world of ";
txt2="JavaScript.";
txt3=txt1+txt2 ;
document. write(txt3);
</script> </b>
</BODY>
</HTML>
```



JavaScript Variables

- A variable is a user-defined name for a memory location whose value can change over time.

Rules of JavaScript Variables

- Do not use any of the JavaScript reserved keyword as variable name.
- JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or the underscore character.

  Example: *123test is an invalid variable name but _123test is a valid one.*
- JavaScript variable names are case sensitive.
- Variable names cannot contain spaces to separate characters.
- Capitalize the first letter of every word except the first

  Example: *calculateSum* or userName

Declaring a Variable

- Before you use a variable in a JavaScript program, you must declare it.
- Variables are declared with the **var** keyword and the variable name.

```
var variableName
```

- You don't declare the *types* of variables in JavaScript
- To assign values to variables, add an equal sign and the value:

Example:

var userName = "Priti"

JavaScript Variable Scope

- The scope of a variable is the region of your program in which it is defined.
- JavaScript variable will have only two scopes.

1. **Global Variables:** Variables declared outside a function are global .These variables will be accessible from anywhere on the page.

2. **Local Variables:** Variables declared within a function are local to that function . These variables will be accessible only within that function.

• 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()**.

example

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

  <h1>My First Web Page</h1>

  <script>
  window.alert(5 + 6);
   document.write(5 + 6);
   document.getElementById(“a").innerHTML = 5 + 6;
   console.log(5 + 6);
  </script>

  </body>
  </html>
  ```
  ```

example2

- <!DOCTYPE html>
- <html>
- <body>

- JavaScript code (or just JavaScript) is a list of JavaScript statements.

- <p id=“raman">

- <script>
- var x = 5;
- var y = 6;
- var z = x + y;
- document.getElementById(“raman).innerHTML = z;
- </script>

- </body>
- </html>

example3

- <!DOCTYPE html>
- <html>
- <body>

- <h1>JavaScript Operators</h1>

- The + operator concatenates (adds) strings.

- <p id=“raman">

- <script>
- var txt1 = “Priti";
- var txt2 = “Bhardwaj";
- document.getElementById(“raman").innerHTML = txt1 + " " + txt2;
- </script>

- </body>
- </html>

example4

- <!DOCTYPE html>
- <html>
- <body>

- <h1>JavaScript Operators</h1>

- Adding a number and a string, returns a string.

- <p id=“raman">

- <script>
- var x = 5 + 5;
- var y = "5" + 5;
- var z = "Hello" + 5;
- document.getElementById(“raman").innerHTML =
- x + "
" + y + "
" + z;
- </script>

- </body>
- </html>

What will be the output

- var x = 16 + 4 + "Volvo"; 20Volvo
- var x = "Volvo" + 16 + 4; Volvo164

example

- <!DOCTYPE html>
- <html>
- <body>

-

- <script>
- var carName1 = "Volvo XC60";
- var carName2 = 'Volvo XC60';
- var answer1 = "It's alright";
- var answer2 = "He is called 'Johnny'";
- var answer3 = 'He is called "Johnny"';

- document.getElementById("demo").innerHTML =
- carName1 + "
" +
- carName2 + "
" +
- answer1 + "
" +
- answer2 + "
" +
- answer3;
- </script>

- </body>
- </html>

output

- Volvo XC60
  Volvo XC60
  It's alright
  He is called 'Johnny'
  He is called "Johnny"

Conditional Statements in JavaScript

- Every programming language possesses the ability to make decisions.
- Every language gives programmers the ability to evaluate a specific condition and then perform different actions depending on the results of that evaluation.

Conditional Statements in JavaScript

- The conditional statement begins with the keyword if, and then a condition is specified within a pair of parentheses.
- The condition is followed by a statement block that consists of an opening brace ({), one or more JavaScript statements, and then a closing brace (}).
- The JavaScript if statement also supports an optional else clause, which defines the action to take if the specified condition is not true.
- The else keyword appears immediately after the statement block of the if clause and is accompanied by a statement block of its own.

Conditional Statements in JavaScript

Syntax:

```
if (condition)
{
statements1;
}
else
{
statements2;
}
```

Conditional Statements in JavaScript

```
if (expression 1){
   Statement(s) to be executed if expression 1 is true
}else if (expression 2){
   Statement(s) to be executed if expression 2 is true
}else if (expression 3){
   Statement(s) to be executed if expression 3 is true
}else{
   Statement(s) to be executed if no expression is true
}
```

switch statement

- The basic syntax of the switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression.

- The interpreter checks each case against the value of the expression until a match is found.

- If nothing matches, a default condition will be used.

switch statement

```
switch (expression)
{
   case condition 1: statement(s)
                     break;
   case condition 2: statement(s)
                     break;
     ...
   case condition n: statement(s)
                     break;
   default: statement(s)
}
```

JavaScript Looping

- A **program loop** is a set of instructions that is executed repeatedly.
- In JavaScript the following looping statements can be used:

  - **while loop**
  - **do..while loop**
  - **for loop**

Loop Statements : while Loop

- The while statement will execute a block of code while a condition is true.

Syntax:

```
while (condition)
{
code to be executed
}
```

condition is a Boolean expression that can be either true or false

Loop Statements : do-while Loop

- The do-while statement will execute a block of code once, and then it will repeat the loop while a condition is true

Syntax:

```
Do
{
code to be executed
}
while (condition)
```

Loop Statements : For Loop

- The **For loop** allows you to create a group of commands to be executed a set number of times through the use of a **counter** that tracks the number of times the command block has been run.
- Set an initial value for the counter, and each time the command block is executed, the counter changes in value.
- When the counter reaches a value above or below a certain stopping value, the loop ends.

Loop Statements : For Loop

Syntax:

```
for (initialization; test condition; iteration statement)
{
Statements
}
```

- The loop initialization is used to represent the starting value of the counter. The initialization statement is executed before the loop begins.
- The test condition is used to test if the given condition is true or not. If condition is true then code given inside the loop will be executed otherwise loop will come out.
- The iteration statement is used to increase or decrease the counter.

example

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

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

<script>
var cars = ["BMW", "Volvo", "Saab", "Ford"];
var text = "";
var i;
for (i = 0; i < cars.length; i++) {
    text += cars[i] + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

JavaScript functions

- Functions are one of the fundamental building blocks in JavaScript
- A function is a JavaScript procedure, a set of statements, that performs a specific task.
- A function is a name block of code that will be executed when it is called.
- Block of code of a function can be reused anywhere in the program.
- There are two types of JavaScript functions.

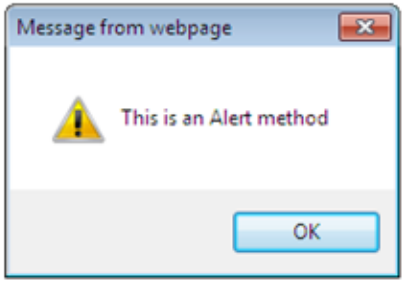
1. Built-in JavaScript functions
2. User-Defined JavaScript Functions

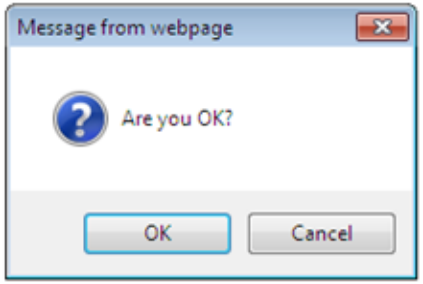
Built-in JavaScript functions

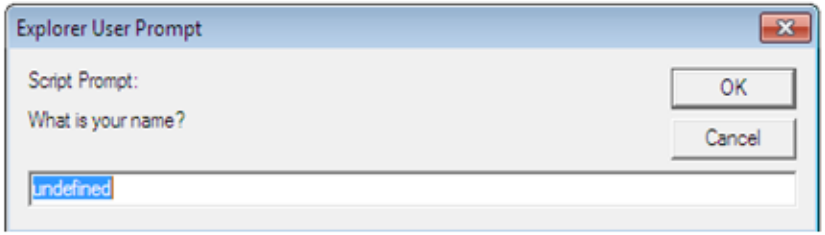
- **alert("message")** - to display a message box. An alert dialog box is mostly used to give a warning message to the users.
- **confirm("message")** - to display a confirmation box. A confirmation dialog box is mostly used to take user's consent on any option.
- **prompt("message")** - to display a prompt box and allow the user to enter a value

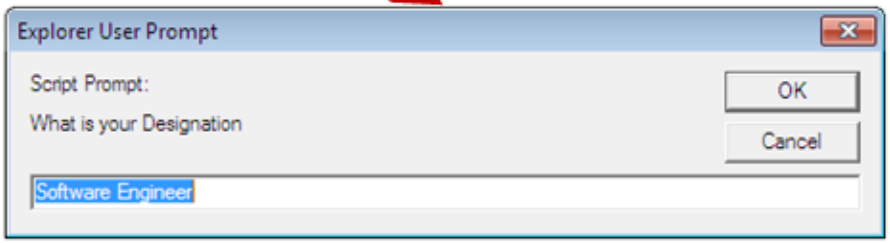
Built-in JavaScript functions : Example

```
<html>
<head>
<title>HTML and JavaScript</title>
</head>
<body>
<script type="text/javascript">
alert("This is an Alert method");
confirm("Are you OK?");
prompt("What is your name?");
prompt("What is your Designation", "Software Engineer");
</script>
</body>
</html>
```









User-Defined JavaScript Functions

- With user-defined functions, you can create your own functions and call it when you need it.
- A function is defined in the HEAD section of a web page to ensure that they are loaded first.
- A function definition has following basic parts:
  - The **function** keyword.
  - A **function name**.
  - A comma-separated **list of arguments** to the function in parentheses.
  - The statements in the function in curly braces.

User-Defined JavaScript Functions

Syntax for defining a function :

```
function functionName(argument 1, ..., argument N)
{
Statements
}
```

- The function may contain return *value*; statements
- Any variables declared within the function are local to it

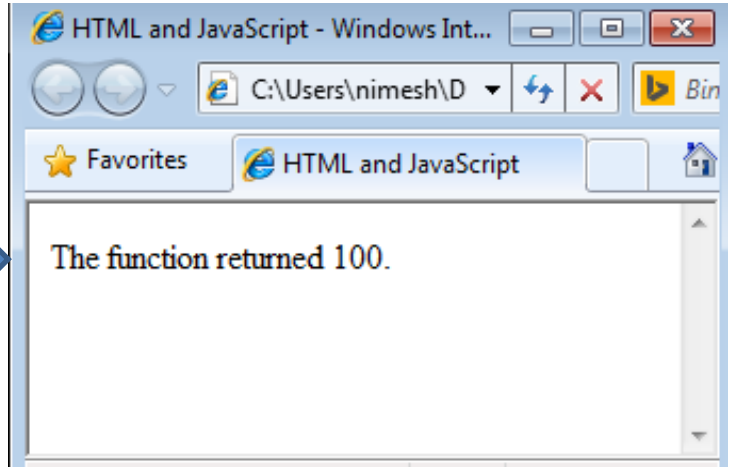
User-Defined JavaScript Functions

Syntax for calling a function:

```
functionName( arguement 1, ..., argument N)
```

User-Defined JavaScript Functions - Example

```
<html>
<head>
<title>HTML and JavaScript</title>
<script type="text/javascript">
function square(number)
{
return number * number
}
</script>
</head>
<body>
<SCRIPT>
document.write("The function returned ", square(10), ".")
</SCRIPT>
</body>
</html>
```



JavaScript Events

- Events associate an object with an action.
- An event as a system-level response to the occurrence of some specific condition.

  Example: clicking a button is an event, as is changing a text field or moving the mouse over a hyperlink.
- User actions trigger events.
- Some of these conditions are generated by the Web browser software, but most of them are caused by the user performing some action.
- Such actions might include moving the mouse, clicking on a button, or even selecting a block of text on the screen.

JavaScript Event handlers

- JavaScript applications are largely event-driven.
- *Events* are actions that occur usually as a result of something the user does.
- You can define *event handlers*, such as **onChange** *and* **onClick**, to make your script react to events.

Some Event Handler Function

onClick: occurs when an object is clicked

onChange: occurs when a user changes value of an element

onFocus: user gives input focus to window

onLoad: occurs when a page loads in a browser

onUnload: occurs just before the user exits a page

onMouseOver: occurs when you point to an object

onMouseOut: occurs when you point away from an object

onSubmit: occurs when you submit a form

Example 1

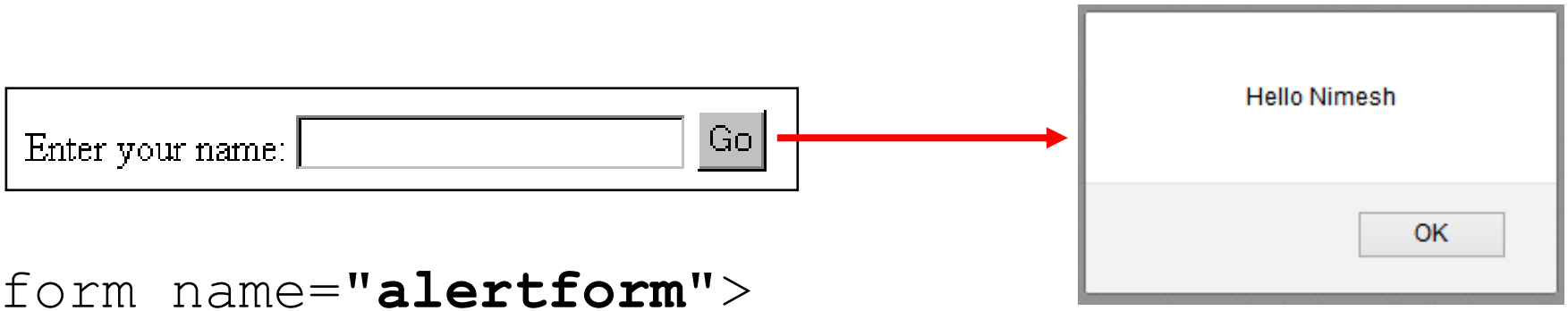
```
<html>
<head><title>My Page</title></head>
<body>
<body onLoad="alert(‘ WELCOME. Enjoy your visit. Please let me know if you detect any problems. Thank you.')">
</body>
</html>
```

Example 2

```
<html>
<head>
<script type="text/javascript">
function myFunc()
{ var a = 100;
alert("Value of variable a is : " + a );
} </script> </head>
<body>
<p>Click the following to see the result:</p>
<form>
<input type="button" value="Click Me" onclick="myFunc();" />
</form>
</body>
 </html>
```

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>
```