JAVA SCRIPTING

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**

What can a JavaScript Do?

- JavaScript is used in web pages for:
  - Dynamics : mouse clicks, pop up windows, and animations
  - Client-side execution : validating input, processing requests
- It avoids Client/Server communication and traffic
- JavaScript is executed on client-side
- JavaScript is simple, powerful, and interpretive language and requires only a web browser
- There have been a number of revisions
- Two types of JavaScript exists:
  - Client-side → code is sent to the client's browser for execution
  - Server-side → code stays on the server for execution

What can a JavaScript Do? …

- **JavaScript gives HTML designers a programming tool -** HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax!
- **JavaScript can put dynamic text into an HTML page -** A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page
- **JavaScript can react to events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element

What can a JavaScript Do?...

- **JavaScript can read and write HTML elements** - A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data** - A JavaScript can be used to validate form data before it is submitted to a server.
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

The Main Features of JavaScript (summary)

- Efficient Programming by the use of **flow control statements** such as for and if.
- Use of **predefined objects** (Documents, Math and Date
- **Use of events** such as mouse clicks or form input to prompt procedures
- **Time procedure**
- **Data input and output checks** via input/output dialog
- **Form Validation**
- Opening a new Page and managing frames and windows.

A Comparison of Java and JavaScript

	JavaScript	Java
Program Compilation	Not Necessary	Necessary
Class, Inheritance	Object-based. Uses no classes or inheritance. (Prototype-based model)	Object-Oriented. Applets consist of object classes with inheritance. (Class-based object model)
Coding	Code integrated with ,and embedded in HTML	Applets distinct from HTML. accessed from HTML pages
Variable Declaration	Variable data types not declared.	Variable data types must be declared.
Script Execution	Interpreted and executed by client	Bytecodes (compiled files) downloaded from server, executed on client
HTML Document Manipulation	Possible	Not Possible

JavaScript coding and Execution

- What you need for Java Script
  - **A text editor**
  - A JavaScript **Compatible web browser**

JavaScript	Nestcape Navigator	Internet Explorer
1.3	4.06	5.0 and above

Learning JavaScript

- Special syntax to learn
- Learn the basics and then use other people's (lots of free sites)
- Write it in a text editor, view results in browser
- You need to revise your HTML

Tips

- Check your statements are on one line
- Check your " and ' quotes match
- Take care with capitalisation
- Lay it out neatly - use tabs
- Remember → in the workbook denotes a continuing line
- Be patient

How to Embed JavaScript

- <html> <body>

  <script type="text/javascript"> ... </script>

  </body> </html>


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


```
<SCRIPT LANGUAGE="JavaScript">
JavaScript statements here
</SCRIPT>
```

Embedding JavaScript in XHTML

- `<script>` tag is used to embed JavaScript code in XHTML code of a web page
- The `<script>` tag can be used anywhere inside the code but it is usually embedded right before of after the `<head>` tag closes
- Any number of `<script>` tags can be embedded, but usually one tag is enough
- Nesting `<script>` tags is prohibited and generates errors
- HTML editors do not follow the `<script>` tag guidelines and embed the tag any where and any number of times

Development Environment

- JavaScript source code is written in an editor and run and tested in a browser, like XHTML
- AceHTML editor has a JavaScript template and also allows writing code manually
- Dreamweaver generates code automatically as the author adds JavaScript functionality
- Error in JavaScript code prevent the page from being rendered and thus debuggers are needed to find where the errors are
- JavaScript interpreters serve the purpose by showing where the error is
- Errors are reported one at a time and are usually easy to fix

JavaScript Statements

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

document.write('This is my first →
JavaScript Page');

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

Note the symbol for line continuation

JavaScript Statements

```
<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

JavaScript Statements

```
<html>
<head><title>My Page</title></head>
<body>
<p>
<a href="myfile.html">My Page</a>
<br />
<a href="myfile.html"
onMouseover="window.alert('Hello');">
My Page</A>
</p>
</body>
</html>
```

An Event

JavaScript written inside HTML

How to Notate Comments

- Use a double slash (//)
  - Web browsers interprets a **single line** proceeded by // As a comment

```
<SCRIPT LANGUAGE =“JavaScript”>
// Your comment here
</SCRIPT>
```

- Enclose comments between /* and */
  - Web browsers interprets **an area enclosed** by /* and */ as comments.
  - This notation is used when you have comments that span **multiple lines**

```
<SCRIPT LANGUAGE =“JavaScript”>
/*  more comment here
    more comment here */
</SCRIPT>
```

Displaying a Document

- Use **document.write() for Displaying text and graphics** in the browser window
  - If you specify a string in document.write(), then browser will display the specified string.

```
document.write("string here");
```

- You can specify HTML tags within documents.write()

```
document.write("<IMG SRC='Image/neko.gif' ALIGN='left'>
                JavaScript for displaying image here.
                <br>string here");
```

- When displaying **multiple data, separate items by a comma(,)** or a plus (+) sign

```
Num=20;
Document.write("The price is",Num, ".Thank you.");
```

Variables

- A variable is a symbolic name that stores a value and has the some characteristics
- Identifiers
  The name of the variable is its identifier
  It must begin with a letter, underscore, or $ sign
  It cannot begin with a number or other characters
  JavaScript is case-sensitive
  Examples: `test, Test, jam234, _best, $abc, a_1$4`
- Types
  Data types are implicit and are converted automatically
  The first use of a variable declares its types
  Types can be numbers (integer or real), logical (boolean), or string
  Examples: `3, 40, -10.5, true, false, "zeid", "9abc"`

Variables

- A variable can hold data such as numbers or characters
  - A variable name must with **a letter**,
  - **an underscore**("_")
  - or **a dollar**($)

```
<body>
<script language="javascript">
<!--
a=100;
 document.write(a);
abc=20-10;
 document.write(abc);
 _abc=30-5;
 document.write(_abc);
 $abc=40-2;
 document.write($abc);
 answer=100-10*2;
  document.write(answer);
//-->
</script>
</body>
```

Variables

- Scope
  The code block within which the variable is available
  Global variable is available everywhere
  Local variable is available only inside a code block
  Global variables are easy to manage but a bad habit
- Constants
  Read only variables defined by a `const` keyword
  Cannot change values or be re declared
  Examples: `const x=22`
- Literals
  Fixed values (hard-coded values) in JavaScript
  Nesting literals needs extra care
  Examples: `3.5, false, "Hello"`
- Data Type Conversion
  JavaScript converts data types automatically, but creates confusion
  Examples: `answer=true, answer=20`
- Escaping and Special Characters
  Backslash is the escaping character and is used to define special ones

Statements

- A statement uses an assignment operator, an equal sign
- The operator has two operands on each of its side and the value of the right operand is assigned to the left one
- Example : `x = y`
- Values of right operand must always be known, if not, and error is generated
- Statement must be only one line long and cannot be broken
- Semicolon (`;`) is used to separate statements
- JavaScript also provides comment statements
  - Inline Comment statement → `//one line comment`
  - Block Comment statement → `/* comment starts here`

    `comment ends here`

    `*/`

Expressions and Operators

- Expressions are a valid set of any variables that evaluates to a single value
  - Arithmetic Expressions evaluate to numbers
  - Logical Expressions evaluate to booleans (true or false)
  - String Expressions evaluate to strings
- JavaScript has a rich set of operators
  - Assignment Operators → `=, +=, -=, *=, /=`
  - Comparison Operators → `==, !=, >, >=, <, <=`
  - Arithmetic Operators → `+, -, *, /, %, ++, --`
  - Logical Operators → `&&, ||, !`

Control Structures

- Control structures control the code execution according to a certain criteria
- Conditional Statements
  - Executes if the specified condition statement is met
  - `if` and `switch` statements are the two types

`if` statements: structure 1:

```
if (condition)
{……}
```

structure 2:

```
if (condition)
{……}
else {……}
```

`switch` statement:

```
switch
(expression){
case condition1:
statements; break;
case condition2:
statements; break;
default:
statements;}
```

Control Structures

- Loop Statements
  - Executes repeatedly till a specific condition is met
  - `for, while,` and `do while` statements are used
  - `break` exits the loop all together
  - `continue` skips the current iteration

`for` statement:
```
for (ini value; end value; incr){
        statements
}
```

`while` statement:
```
while (condition) {
   statements
}
```

`do while` statement:
```
do {
   statements
}while (condition)
```

Code Execution

- JavaScript code shell looks like:

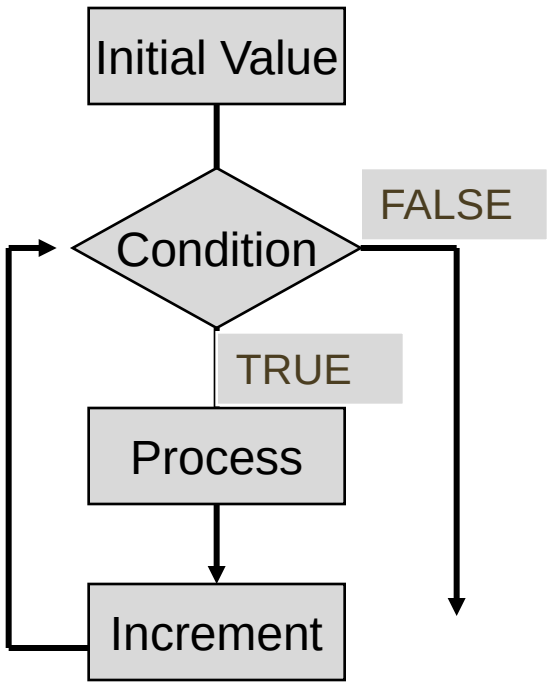
```
<script language="javascript">
function definition code
function definition code
function definition code
statements
function calls
statements
function calls
</script>
```

- JavaScript interpreter executes code top to bottom, left to right
- Function definitions are executed only when called

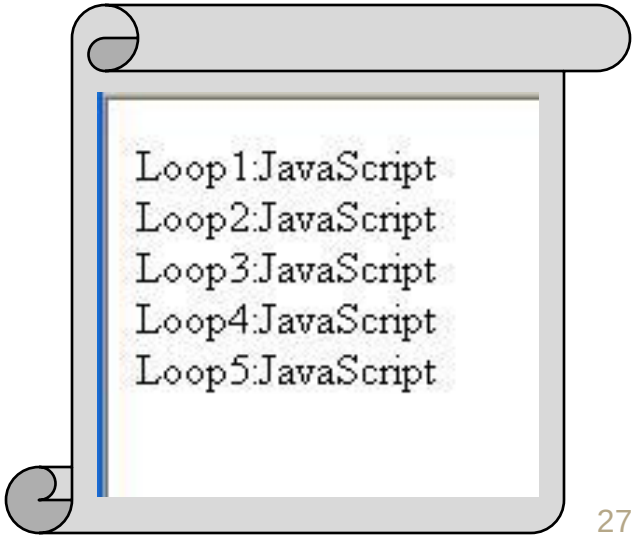
Loop - for

- Use a for loop statement when you want to **repeat statements** a fixed no. of. Times.

```
For (initial expression; terminating condition; increment expression){
        process;
        .
        .
}
```




```
for (i=1; i<6; i++){
document.write("Loop",i,":JavaScript<br>");
}
```

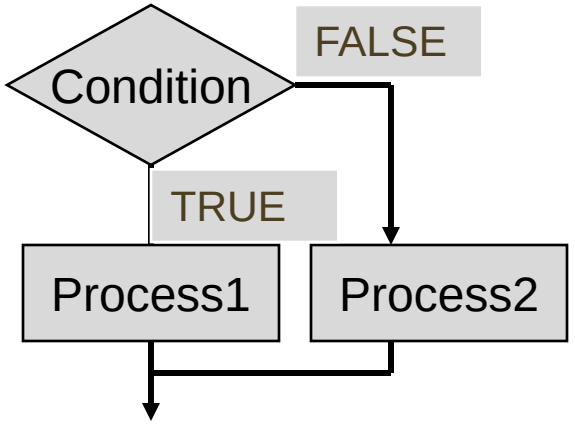


Conditional Branching

- Use the if statement to **perform separate statements according to a condition**

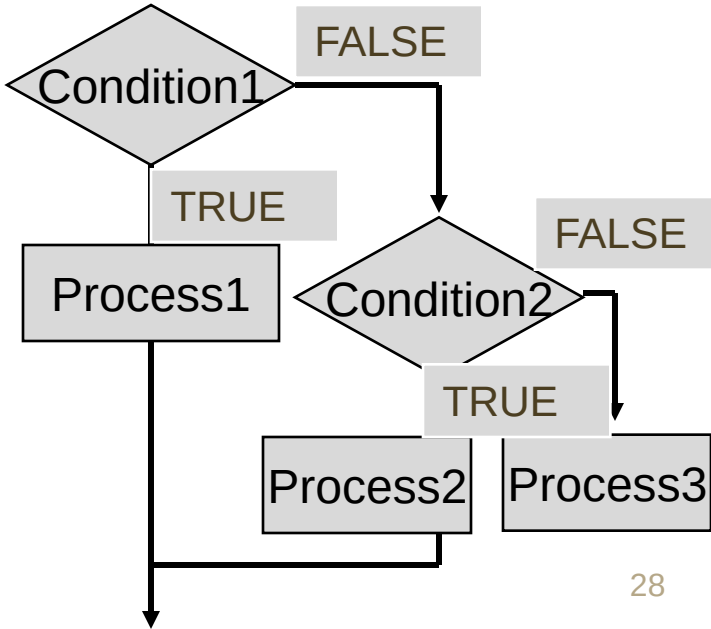
if

```
if (condition){
            statement for when condition1 is true;
} else {
            statement for when condition1 false
}
```



Else if

```
if (condition1){
            statement for when condition1 is true;
} else if (condition2){
            statement for when condition2 true;
} else {
            statements for when all condition are false;
}
```



Functions

- **A function groups together a set of statements under a named subroutine.** A function can be called by that name whenever its action is required.
- Reasons for use;
  - **Reuse script**
    - You can simply call the function as necessary and avoid rewriting the entire block of code again.
  - **Clarify your program**
    - Functions make the purpose of each section clear and thus makes your script coding more efficient.
  - **Easy maintenance**
    - You can simply change that part
- What is an argument
  - **Arguments are variables used in the functions.** The values in the variable are passed on by the function call
- What is a return value?
  - A return **value is value returned to the calling expression**. It can be omitted if a return value is not necessary.

Defining Functions

- How to define and call functions;

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
Function function_name (argument, argument,…) {
        my_statemetn;
        :
        return return_value;
        }
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
variable_name = function_name (argument, argument,…);
        </SCRIPT>
        </BODY>
        </HTML>
```

1 Function Definition

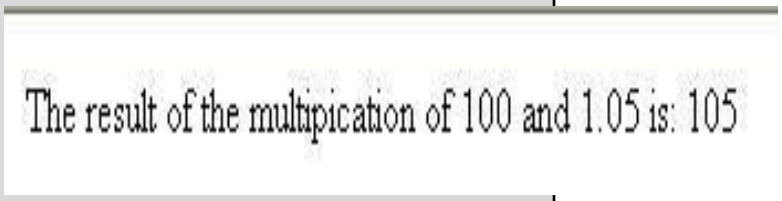
2 Calling a function

3 The returned value from the function is assigned to this variable

Function Example

- The function is **defined in the <HEAD>** section, and **called from the <BODY> part of the HTML** document.

```
<html><head>
<title>kansu.html </title>
 <script language="javascript">
 function kansu (i){
          result= i*1.05;
          return result;
          }
  </script>
 </head>
<body>
 The result of the multipication of 100 and 1.05 is:
 <script language="javascript">
<!--
    x=kansu(100);
   document.write(x);
//-->
</script>
</body></html>
```



Event Procedures / handlers

- What are events
  - **Events are actions that occur usually as a result of something** a user does such as clicking a mouse.
- Event Handlers
  - **Events handlers identify such events** and they can be placed within the HTML tags.

Event Handler	Occurs when…
onChange	User changes value of text, textarea or select element
onClick	User clicks on form element or link
onFocus	User gives form element focus
onLoad	User loads the page
onUnLoad	User unloads the page (exit)
onMouseOut	User moves mouse pointer off of link or anchor
onMouseOver	User moves mouse pointer over a link or anchor
onSelect	User selects form element's input field
onSubmit	User submits a form
onReset	User resets form fields

Event Procedure Example

<INPUT TYPE=“button” onClick=“some JavaScript code here or some function name here”>

<INPUT TYPE=“button” VALUE=“display message”

onClick=“alert(‘Welcome to my homepage’)”>

```
<html>
<head>
<title>event.html </title>
 <script language="javascript">
 function message(){
        alert("Welcome to my home page");
        }
   </script>
 </head>
<body>
<a href="http://www.flm.fujitsu.com/" onMouseOut="message()">
            Welcome to the home page
</a>
</body>
</html>
```

Using Objects

- What is an Object ?
  - An **object consists of a collection of data and processes** (methods)
- What is a Property?
  - **A property is equivalent of object data or a value**.
  - Javascript **defines properties as variables**
- What is a Method
  - A **method defines what takes to perform**.
  - In Javascript a method is a function call.
- Types of Predefined objects
  - **String Objects** (For working with text)
  - **Date Object** (for working with dates and times)
  - **Math Objects** (Mathematical constants and functions)
  - **Array object** (To store a set of values in a single variable)
  - **Number Object** (working with numbers)
  - RegExp (Provides simple regular **expression pattern** searches.

Example Script for Getting Dates and Time

```
<html>
<head>
<title>Date and Time </title>
 </head>
<body>
The program will display the current year, month, date hour, minute, and second.<br>
 <script language="javascript">
 <!--
 // Creating an Date object
 now = new Date();
 /* Getting and Displaying the year, month, date, hour, minute, and second*/
 document.write(now.getFullYear()+"Year");
 document.write(now.getMonth()+1,"Month",now.getDate(),"date");
 document.write(now.getHours(),"hour",now.getMinutes(),"minute");
 document.write(now.getSeconds(),"second");
//-->
 </script>
</body>
</html>
```

The program will display the current year, month, date hour, minute, and second.
2005Year8Month8date15hour15minute25second

Example Script for Closing a Window

```
<HTML>
<HEAD>
      <TITLE>new.html</TITLE>
</HEAD>
<BODY bgcolor="ffcc99" onload="setTimeout('window.close()',10000)">
      I am a cat!!<BR><BR>
      <IMG SRC = 'image/neko.gif'><BR><BR>
      <script language="javascript">
   <!--
   document.write("The last modified date/time:", document.lastModified,"<br>");
//--> </script>
      <form>
      <input type="button" value="close" onClick="window.close()">
      </form>
</BODY>
</HTML>
```

Example Script for Last Modified Date and Time

```
<html>
<head>
<title>The last modified date and time</title>
  </head>
<body>
 <script language="javascript">
 <!--
 document.write("The last modified date/time:", document.lastModified);
//-->
 </script>
</body>
</html>
```

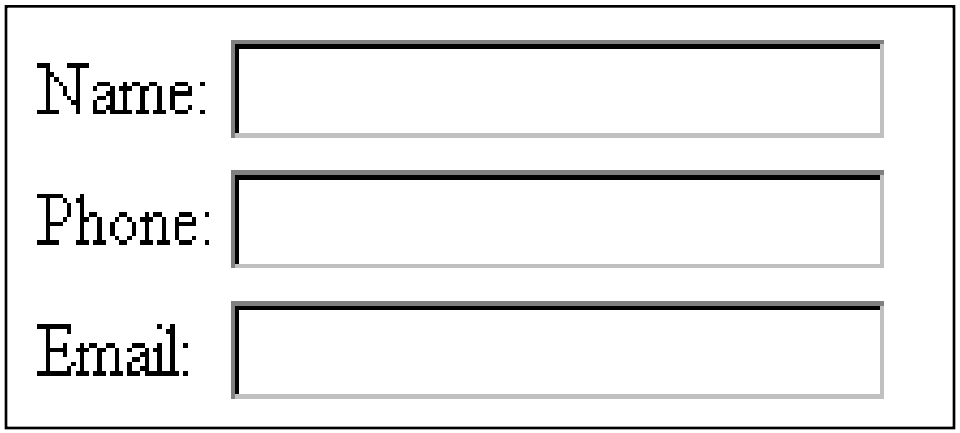
Input and Output

- Client-side JavaScript has limited input/output utilities due to security reasons
- The input functions available are:
  `prompt (message, default)` → takes an input and returns it to the JavaScript program
  `confirm (question)` → asks the user to confirm an input value and return a boolean value
- The output functions available are:
  `document.write (string)`
  `alert (string)`
  Both these functions are used to output results in a web page

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



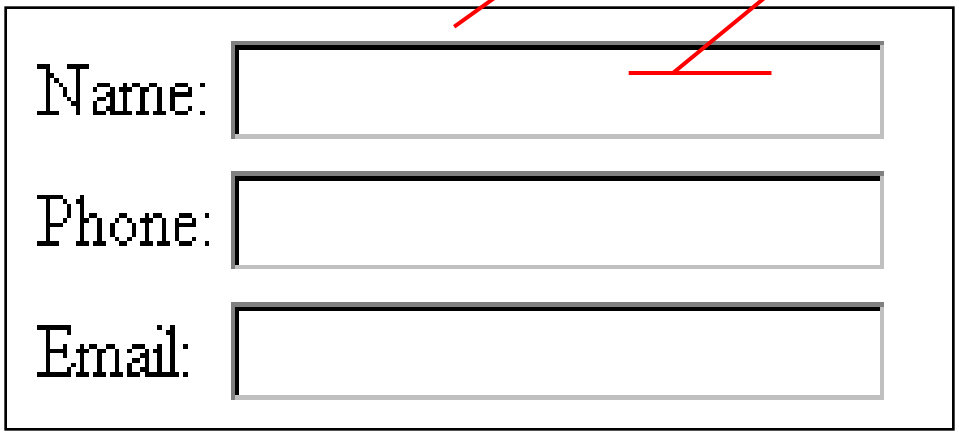

```
<form name="addressform">
Name:  <input name="yourname"><br
  />
Phone: <input name="phone"><br />
Email: <input name="email"><br />
</form>
```

Forms and JavaScript

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

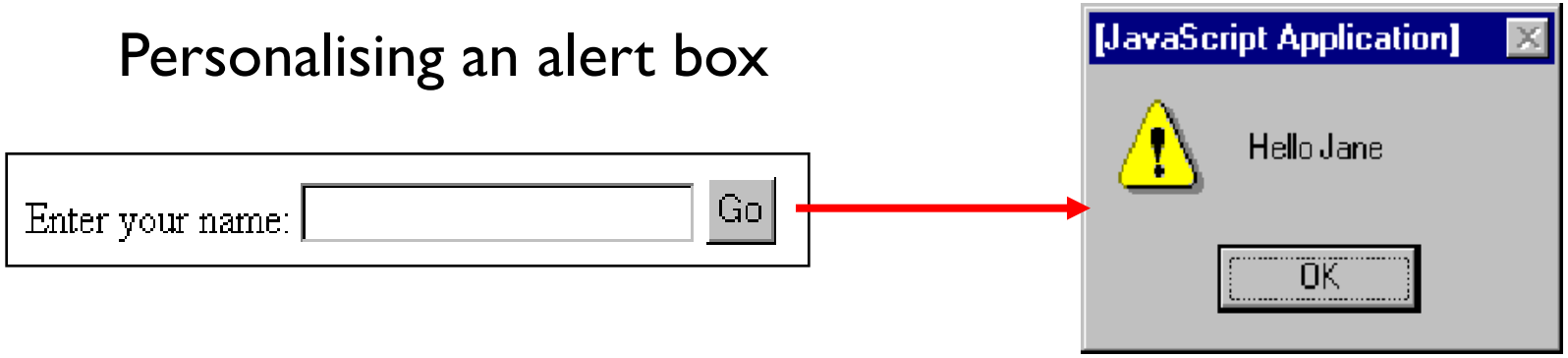
Thus:

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



Using Form Data

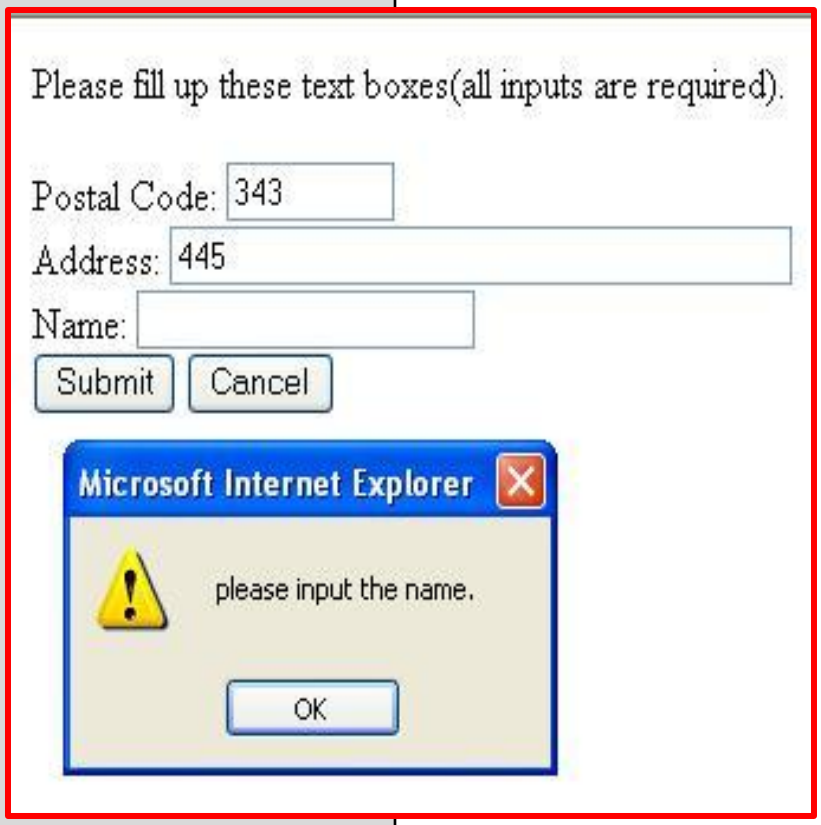
Personalising an alert box




```
<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>
```

Example Script for Form Validation

```
<html><head><title>Form Validation Checking</title>
 <script language="javascript">
 <!--
 //Calculate to check form input
 function checkForm() {
 if (document.fm.yubin.value==""){
 alert("please input the postal code.");
 return false;
 }
 if (document.fm.address.value==""){
 alert("please input the address.");
 return false;
 }
 if (document.fm.name.value==""){
 alert("please input the name.");
 return false;
 }
 return true;
 }
 :
 :
```



Example Script for Form Validation…

```
:
:
//-->
 </script> </head><body>
 Please fill up these text boxes(all inputs are required).<br>
<form action ="flm.cgi" name="fm" onSubmit="return checkForm()">
Postal Code:
<input type="text" Name="yubin" size="8"><br>
Address:
    <input type="text" Name="address" size="40"><br>
Name:
      <input type="text" Name="name" size="20"><br>
     <input type="submit" value="Submit">

     <input type="reset" value="Cancel">
  </form></body></html>
```

Summary

- JavaScript is a powerful language and makes a web page dynamic
- JavaScript and Java are fundamentally different in most ways
- JavaScript code is embedded in XHTML code
- JavaScript code is written and tested like XHTML code
- JavaScript begins with variables
- JavaScript uses statements to build code block
- JavaScript has a rich set of operators
- JavaScript has control structures to control code execution
- Code execution follows top to bottom, left to right rule
- Input and output is handled using basic functions