

Chapter Four

Client-Side Scripting Language

Client-side JavaScript:

- Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.
- The JavaScript code is executed when the user submits the form, and only if all the entries are valid they would be submitted to the Web Server.

Advantages of JavaScript:

The merits of using JavaScript are:

- **Less server interaction:** You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors:** They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity:** You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces:** You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Limitations with JavaScript:

- We can not treat JavaScript as a full fledged programming language. It lacks the following important features:
- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript can not be used for Networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multi-process capabilities.

Java script syntax

- A JavaScript consists of JavaScript statements that are placed within the `<script>... </script>` HTML tags in a web page.
- You can place the `<script>` tag containing your JavaScript anywhere within you web page but it is preferred way to keep it within the `<head>` tags.

Cont'd...

- The `<script>` tag alert the browser program to begin interpreting all the text between these tags as a script. So simple syntax of your JavaScript will be as follows:

`<script ...>`

JavaScript code

`</script>`

Cont'd...

- The script tag takes two important attributes:
 - **language:** This attribute specifies what scripting language you are using. Typically, its value will be *javascript*. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
 - **type:** This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "*text/javascript*".

Cont'd...

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

Cont'd...

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

Cont'd...

Semicolons are Optional:

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

```
<!--
```

```
var1 = 10
```

```
var2 = 20
```

```
//-->
```

```
</script>
```

Cont'd...

But when formatted in a single line as follows, the semicolons are required:

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

```
<!--
```

```
var1 = 10; var2 = 20;
```

```
//-->
```

```
</script>
```

JavaScript Placement in HTML File

- Script in `<head>...</head>` section.
- Script in `<body>...</body>` section.
- Script in `<body>...</body>` and `<head>...</head>` sections.
- Script in an external file and then include in `<head>...</head>` section.

Cont'd...

1. JavaScript in <head>...</head> section:

```
<html>
  <head>
    <script type="text/javascript">
      <!-- function sayHello()
      {
        alert("Hello World");
      } //-->
    </script>
  </head>
  <body>
    <input type="button" onclick="sayHello()" value="Say Hello" />
  </body>
</html>
```

Cont'd...

2.JavaScript in <body>...</body> section:

```
<html>
<head></head>
<body>
  <script type="text/javascript">
    <!-- document.write("Hello World");
  //-->
</script>
<p>This is web page body </p>
</body>
</html>
```

Cont'd...

3. JavaScript in External File :

```
<html>
```

```
<head>
```

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

```
<body> ..... </body>
```

```
</html>
```

Programming Basics

- Programmers use variables to store values. A variable can hold several types of data. In JavaScript you don't have to declare a variable's data type before using it. Any variable can hold any JavaScript data type, including:
 - String data
 - Numbers
 - Boolean values (T/F)

JavaScript Variables

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

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

```
    var money;
```

```
    var name;
```

```
</script>
```

JavaScript if...else Statements

- JavaScript supports conditional statements which are used to perform different actions based on different conditions. Here we will explain **if..else** statement.
- JavaScript supports following forms of **if..else** statement:
 - if statement
 - if...else statement
 - if...else if... statement.

Cont'd...

- **if statement:**

-The **if** statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

```
    if (expression)
    {
        Statement(s) to be executed if expression
is true
    }
```

Cont'd...

- Example

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

```
<!--
```

```
var age = 20; if( age > 18 )  
  { document.write("<b>Qualifies for  
    driving</b>"); } //
```

```
-->
```

```
</script>
```

Cont'd...

- **if...else statement:**

The **if...else** statement is the next form of control statement that allows JavaScript to execute statements in more controlled way.

Cont'd...

- **Syntax:**

if (expression)

{

Statement(s) to be executed if expression is true

}

else

{

Statement(s) to be executed if expression is false

}

Cont'd...

- Example

```
<script type="text/javascript">
<!-- var age = 15;
if( age > 18 ){ document.write("<b>Qualifies for
    driving</b>");
}
else
{
document.write("<b>Does not qualify for driving</b>");
}
//-->
</script>
```

Cont'd...

- **if...else if... statement:**

The **if...else if...** statement is the one level advance form of control statement that allows JavaScript to make correct decision out of several conditions.

Switch statement

- You can use multiple *if...else if* statements, as in the previous chapter, to perform a multiway branch. However, this is not always the best solution, especially when all of the branches depend on the value of a single variable.

Cont'd...

- Syntax

switch (expression)

{

case condition 1: statement(s)

break;

case condition 2: statement(s)

break;

... case condition n: statement(s)

break;

default: statement(s)

}

Cont'd...

- ```
<script type="text/javascript">
 <!-- var grade='A'; document.write("Entering
 switch block
");
switch (grade)
{
case 'A': document.write("Good job
");
 break;
case 'B': document.write("Pretty good
");
 break;
```

## Cont'd...

---

```
case 'C': document.write("Passed
"); break;
case 'D': document.write("Not so good
");
break;
case 'F': document.write("Failed
");
break;
default: document.write("Unknown grade
")
document.write("Exiting switch block"); //-->
</script>
```

# JavaScript Functions

---

- A function is a group of reusable code which can be called anywhere in your programme. This eliminates the need of writing same code again and again.
- JavaScript allows us to write our own functions as well.

## Cont'd...

---

### **Function Definition:**

- Before we use a function we need to define that function. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name,

## Cont'd...

---

- `<script type="text/javascript">`  
  `<!--`  
    function functionname(parameter-list)  
    {  
      statements  
    }  
  `//-->`  
`</script>`

## Cont'd...

---

- **Example:**

A simple function that takes no parameters called sayHello is defined here:

```
<script type="text/javascript">
 <!-- function sayHello()
 {
 alert("Hello there");
 }
 //-->
</script>
```

## Cont'd...

---

### **Calling a Function:**

To invoke a function somewhere later in the script, you would simple need to write the name of that function as follows:

```
<script type="text/javascript">
<!--
sayHello();
//-->
</script>
```

## Cont'd...

---

### **Function Parameters:**

A function can take multiple parameters separated by comma.

```
<script type="text/javascript">
<!-- function sayHello(name, age)
{
 alert(name + " is " + age + " years old.");
}
//-->
</script>
```

## Cont'd...

---

```
<script
type="text/javascript">
 <!--
 sayHello('Zara', 7);
 //-->
</script>
```

# *Return* statement

---

- **Example:**

```
<script type="text/javascript">
<!--
 function concatenate(first, last)
{
 var full;
 full = first + last;
 return full;
}
//-->
</script>
```

## Cont'd...

---

- Now we can call this function as follows:

```
<script type="text/javascript">
<!--
var result;
result = concatenate('Zara', 'Ali');
alert(result);
//-->
</script>
```

# JavaScript Events

---

## **What is an Event ?**

JavaScript's interaction with HTML is handled through events that occur when the user or browser manipulates a page.

## Cont'd...

---

- **Example:**

```
<html>
<head>
<script type="text/javascript">
<!--
function sayHello()
{
 alert("Hello World");
}
//-->
</script>
</head>
<body>
<input type="button" onclick="sayHello();" value="Say Hello" />
</body>
</html>
```

Event	Value	Description
onchange	script	Script runs when the element changes
onsubmit	script	Script runs when the form is submitted
onreset	script	Script runs when the form is reset
onselect	script	Script runs when the element is selected
onblur	script	Script runs when the element loses focus
onfocus	script	Script runs when the element gets focus
onkeydown	script	Script runs when key is pressed
onkeypress	script	Script runs when key is pressed and released
onkeyup	script	Script runs when key is released
onclick	script	Script runs when a mouse click
ondblclick	script	Script runs when a mouse double-click
onmousedown	script	Script runs when mouse button is pressed
onmousemove	script	Script runs when mouse pointer moves
onmouseout	script	Script runs when mouse pointer moves out of an element
onmouseover	script	Script runs when mouse pointer moves over an element
onmouseup	script	Script runs when mouse button is released

# JavaScript Popup Boxes

---

There are three basic popup boxes as follows,

## **1. Alert box**

- An alert box is often used if you want to make sure information comes through to the user.
- User will have to click "OK" to proceed  
`alert("sometext");`

## Cont'd...

---

### **2. Confirm box**

- User will have to click either "OK" or "Cancel" to proceed
- A confirm box is often used if you want the user to verify or accept something.

`confirm("sometext");`

## Cont'd...

---

### **3. Prompt box**

- User will have to click either "OK" or "Cancel" to proceed after entering an input value
- A prompt box is often used if you want the user to input a value before entering a page.

```
prompt("sometext","defaultvalue");
```

---

Thank U!