**UNDERSTANDING JAVASCRIPT**

JAVASCRIPT – SYNTAX

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

You can place the **<script>** tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it within the **<head>** tags.

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

<script ...>

JavaScript code

</script>

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".

So your JavaScript syntax will look as follows.

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

JavaScript code

</script>

**Your First JavaScript Code**

Let us take a sample example to print out "Hello World". We added an optional HTML comment that surrounds our JavaScript code. This is to save our code from a browser that does not support JavaScript. The comment ends with a "//-->". Here "//" signifies a comment in JavaScript, so we add that to prevent a browser from reading **15**

the end of the HTML comment as a piece of JavaScript code. Next, we call a function **document.write** which writes a string into our HTML document.

This function can be used to write text, HTML, or both. Take a look at the following code.

<html>

<body>

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

<!--

document.write ("Hello World!")

//-->

</script>

</body>

</html>

This code will produce the following result:

Hello World!

**Whitespace and Line Breaks**

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

**Semicolons are Optional**

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 each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

<script language="javascript" type="text/javascript"> **16**

<!--

var1 = 10

var2 = 20

//-->

</script>

But when formatted in a single line as follows, you must use semicolons:

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

<!--

var1 = 10; var2 = 20;

//-->

</script>

**Note: It is a good programming practice to use semicolons.**

**Case Sensitivity**

JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

**NOTE:** Care should be taken while writing variable and function names in JavaScript.

**Comments in JavaScript**

JavaScript supports both C-style and C++-style comments. Thus:

 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.

**17**

 JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this 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 //-->.

**Example**

The following example shows how to use comments in JavaScript.

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

<!--

// This is a comment. It is similar to comments in C++

/\*

\* This is a multiline comment in JavaScript

\* It is very similar to comments in C Programming

\*/

//-->

</script> **18**

3. JAVASCRIPT – ENABLING

All the modern browsers come with built-in support for JavaScript. Frequently, you may need to enable or disable this support manually. This chapter explains the procedure of enabling and disabling JavaScript support in your browsers: Internet Explorer, Firefox, chrome, and Opera.

**JavaScript in Internet Explorer**

Here are the steps to turn on or turn off JavaScript in Internet Explorer:

 Follow **Tools -> Internet Options** from the menu.

 Select **Security** tab from the dialog box.

 Click the **Custom Level** button.

 Scroll down till you find the **Scripting** option.

 Select *Enable* radio button under **Active scripting**.

 Finally click OK and come out.

To disable JavaScript support in your Internet Explorer, you need to select **Disable** radio button under **Active scripting**.

**JavaScript in Firefox**

Here are the steps to turn on or turn off JavaScript in Firefox:

 Open a new tab -> type **about: config** in the address bar.

 Then you will find the warning dialog. Select **I’ll be careful, I promise!**

 Then you will find the list of **configure options** in the browser.

 In the search bar, type **javascript.enabled**.

 There you will find the option to enable or disable javascript by right-clicking on the value of that option -> **select toggle**.

If javascript.enabled is true; it converts to false upon clicking **toogle**. If javascript is disabled; it gets enabled upon clicking toggle. **19**

**JavaScript in Chrome**

Here are the steps to turn on or turn off JavaScript in Chrome:

 Click the Chrome menu at the top right hand corner of your browser.

 Select **Settings**.

 Click **Show advanced settings** at the end of the page.

 Under the **Privacy** section, click the Content settings button.

 In the "Javascript" section, select "Do not allow any site to run JavaScript" or "Allow all sites to run JavaScript (recommended)".

**JavaScript in Opera**

Here are the steps to turn on or turn off JavaScript in Opera:

 Follow **Tools-> Preferences** from the menu.

 Select **Advanced** option from the dialog box.

 Select **Content** from the listed items.

 Select **Enable JavaScript** checkbox.

 Finally click OK and come out.

To disable JavaScript support in Opera, you should not select the **Enable JavaScript checkbox**.

**Warning for Non-JavaScript Browsers**

If you have to do something important using JavaScript, then you can display a warning message to the user using **<noscript>** tags.

You can add a **noscript** block immediately after the script block as follows:

<html>

<body>

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

<!-- **20**

document.write ("Hello World!")

//-->

</script>

<noscript>

Sorry...JavaScript is needed to go ahead.

</noscript>

</body>

</html>

Now, if the user's browser does not support JavaScript or JavaScript is not enabled, then the message from </noscript> will be displayed on the screen. **21**

4. JAVASCRIPT – PLACEMENT

There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows:

 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.

In the following section, we will see how we can place JavaScript in an HTML file in different ways.

**JavaScript in <head>...</head> Section**

If you want to have a script run on some event, such as when a user clicks somewhere, then you will place that script in the head as follows.

<html>

<head>

<script type="text/javascript">

<!--

function sayHello() {

alert("Hello World")

}

//-->

</script>

</head>

<body>

Click here for the result

<input type="button" onclick="sayHello()" value="Say Hello" /> **22**

</body>

</html>

This code will produce the following results:

Click here for the result

Say Hello

**JavaScript in <body>...</body> Section**

If you need a script to run as the page loads so that the script generates content in the page, then the script goes in the <body> portion of the document. In this case, you would not have any function defined using JavaScript. Take a look at the following code.

<html>

<head>

</head>

<body>

<script type="text/javascript">

<!--

document.write("Hello World")

//-->

</script>

<p>This is web page body </p>

</body>

</html>

This code will produce the following results:

Hello World **23**

This is web page body

**JavaScript in <body> and <head> Sections**

You can put your JavaScript code in <head> and <body> section altogether as follows.

<html>

<head>

<script type="text/javascript">

<!--

function sayHello() {

alert("Hello World")

}

//-->

</script>

</head>

<body>

<script type="text/javascript">

<!--

document.write("Hello World")

//-->

</script>

<input type="button" onclick="sayHello()" value="Say Hello" />

</body>

</html>

This code will produce the following result.

HelloWorld

Say Hello **24**

**JavaScript in External File**

As you begin to work more extensively with JavaScript, you will be likely to find that there are cases where you are reusing identical JavaScript code on multiple pages of a site.

You are not restricted to be maintaining identical code in multiple HTML files. The **script** tag provides a mechanism to allow you to store JavaScript in an external file and then include it into your HTML files.

Here is an example to show how you can include an external JavaScript file in your HTML code using **script** tag and its **src** attribute.

<html>

<head>

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

</head>

<body>

.......

</body>

</html>

To use JavaScript from an external file source, you need to write all your JavaScript source code in a simple text file with the extension ".js" and then include that file as shown above.

For example, you can keep the following content in **filename.js** file and then you can use **sayHello** function in your HTML file after including the filename.js file.

function sayHello() {

alert("Hello World")

} **25**

5. JAVASCRIPT – VARIABLES

**JavaScript Datatypes**

One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.

JavaScript allows you to work with three primitive data types:

 **Numbers**, e.g., 123, 120.50 etc.

 **Strings** of text, e.g. "This text string" etc.

 **Boolean**, e.g. true or false.

JavaScript also defines two trivial data types, **null** and **undefined**, each of which defines only a single value. In addition to these primitive data types, JavaScript supports a composite data type known as **object**. We will cover objects in detail in a separate chapter.

**Note:** Java does not make a distinction between integer values and floating-point values. All numbers in JavaScript are represented as floating-point values. JavaScript represents numbers using the 64-bit floating-point format defined by the IEEE 754 standard.

**JavaScript Variables**

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.

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; **26**

//-->

</script>

You can also declare multiple variables with the same **var** keyword as follows:

<script type="text/javascript">

<!--

var money, name;

//-->

</script>

Storing a value in a variable is called **variable initialization**. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

For instance, you might create a variable named **money** and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

<script type="text/javascript">

<!--

var name = "Ali";

var money;

money = 2000.50;

//-->

</script>

**Note:** Use the **var** keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

JavaScript is **untyped** language. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically. **27**

**JavaScript Variable Scope**

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

 **Global Variables:** A global variable has global scope which means it can be defined anywhere in your JavaScript code.

 **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Within the body of a function, a local variable takes precedence over a global variable with the same name. If you declare a local variable or function parameter with the same name as a global variable, you effectively hide the global variable. Take a look into the following example.

<script type="text/javascript">

<!--

var myVar = "global"; // Declare a global variable

function checkscope( ) {

var myVar = "local"; // Declare a local variable

document.write(myVar);

}

//-->

</script>

It will produce the following result:

Local

**JavaScript Variable Names**

While naming your variables in JavaScript, keep the following rules in mind. **28**

 You should not use any of the JavaScript reserved keywords as a variable name. These keywords are mentioned in the next section. For example, **break** or **boolean** variable names are not valid.

 JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, **123test** is an invalid variable name but ***\_*123test** is a valid one.

 JavaScript variable names are case-sensitive. For example, **Name** and **name** are two different variables.

**JavaScript Reserved Words**

|  |  |  |  |
| --- | --- | --- | --- |
| A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names. abstract  boolean  break  byte  case  catch  char  class  const  continue  debugger  default  delete  do  double | else  enum  export  extends  false  final  finally  float  for  function  goto  if  implements  import  in | Instanceof  int  interface  long  native  new  null  package  private  protected  public  return  short  static  super | switch  synchronized  this  throw  throws  transient  true  try  typeof  var  void  volatile  while  with |