6.1 Introduction

HTML or XHTML is one of the easiest and popular computer languages around. It is used to give specific instructions to a browser. The browser interprets these instructions then displays text and items on the monitor screen. HTML uses a series of (small) instructions called tags that are sent to the browser. The browser is equipped with an internal program called an interpreter. It is the interpreter that receives HTML tags, analyzes them, and tells the browser what to do.

HTML was primarily created to display text. Therefore, it has greatly accomplished its intended purpose. To complement HTML's shortcomings, new languages were developed to handle other assignments. One of these languages is called JavaScript.

In this introductory chapter, we look at what JavaScript is, what it can do for us, and what we need in order to use it. With these foundations in place, we will see throughout the rest of the book how JavaScript can helps to create powerful web applications for the web site. In this chapter we really going to start learning to program; we will be learning the basics of a programming language called JavaScript. JavaScript is a lightweight programming language, often referred to as a **scripting language**, but is capable of understanding many of the basic concepts of programming.

6.2 Origin of Java Script

JavaScript originated with Netscape and it is invented by Brendan Eich. The Netscape company wanted a scripting language that could interface with the server-side components and created one called "**LiveScript**." The language Eich created was named "LiveScript," to reflect its dynamic nature, but was quickly renamed JavaScript. Netscape and Sun jointly announced the new language on December 4, 1995, calling it a "complement" to both HTML and Java.

Not to be out-engineered, Microsoft countered Netscape's effort with the release of Internet Explorer and its own scripting language VBScript derived from the company's popular Visual Basic. Later, it also released its own version of a JavaScript-like language: JScript.

The competition between browsers and languages impacted the early adoption of JavaScript within many companies, especially as the difficult challenge of maintaining cross-browser compatible pages increased not to mention confusion about the name.

In an effort to cut through the compatibility issues, Netscape submitted the JavaScript specification to the European Computer Manufacturer's Association (ECMA) International in 1996, to reissue it as a standardized work. Engineers from Sun, Microsoft, Netscape, and other companies holding a stake in the language were invited to participate, and the result was the release of the first **ECMAScript** specification **ECMA-262** in June 1997. Since that time, most companies that support a version of JavaScript (or JScript or ECMAScript) have agreed to, at a minimum, support ECMA-262.

Who invented JavaScript?

JavaScript originated with Netscape and it is invented by Brendan Eich

What's relationship between JavaScript and ECMAScript?

ECMAScript is yet another name for JavaScript (other names include LiveScript).

6.3 What is Java Script?

JavaScript is a cross-platform, object-based scripting language invented specifically for use in web browsers to make websites more dynamic and attractive. HTML or XHTML is capable of outputting more-or-less static pages. Once we load web page, the page doesn't change much until we click a link to go to a new page. Adding JavaScript to our code allows us to change how the document looks completely, from changing text, to changing colors, to changing the options available in a drop-down list and much, much more!.

Script is nothing but a small program which is generally very easy to learn and use. Though JavaScript can run on both client-side and server-side, it is very popular among client-side scripting languages. Client side means the script runs on client browser and server side means the script runs in server.

JavaScript is mainly used as a client side scripting language. This means that all the action occurs on the client's side of things. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it's up to the browser to do something with it. JavaScript is supported by many popular Web browsers, including all remotely recent versions of Microsoft Internet Explorer and Netscape Navigator, but sometimes we have to turn on the capabilities in the browser preferences.

DEFINITION: JAVASCRIPT

JavaScript is a cross-platform, object-based scripting language invented specifically for use in web browsers to make websites more dynamic and attractive. JavaScript is mainly used as a client side scripting language.

More About JavaScript:

- JavaScript is a object-based scripting language. A script is a segment of code that manipulates the browser and its contents.
- JavaScript is a programming language that can be written into HTML documents to increase interactivity with the user.
- JavaScript is created by Netscape
- JavaScript is easy to implement, as a complier is not needed for execution.
- JavaScript requires an interpreter for execution.
- JavaScript provides the facility of event driven programming.
- JavaScript is architecture neutral language and is independent of the hardware on which it works. It is a language that is understood by any JavaScript enabled browser.
- JavaScript is a case-sensitive language.
- JavaScript as an open language implies that it can be used by anyone; no license is required to use JavaScript.
- JavaScript has the ability to function both as an object oriented language as well as procedural language.
- Using JavaScript we can create objects, attach methods and properties. Though JavaScript can run on both client-side and server-side, it is very popular among client-side scripting languages.

Why JavaScript?

- Sometimes we need the functionality such as **HTML** Data Form validation. Then in web development we have two kinds of methods to valid the entire data form. One is to validate

it on server side. It is a fact that when we use server side scripting language our web server need to work much more. And experience says that we need to use client side scripting instead of server side for validations like email address validations, some text boxes are empty or not and much more. When we use JavaScript to validate some data on client side this will make our browsing fast and suppose instead of client side we use server side then the performance of this validation will be slow.

- JavaScript plays a vital role for adding interactive feature to HTML web pages. JavaScript can respond to several events that occur on web-pages and thereby help in designing dynamic and interactive web-sites. Millions of web-pages and server applications make use of JavaScript all over the world, there is no doubt that almost all major web-browsers support JavaScript.

What can JavaScript do?

- **Display information based on the time of the day**: JavaScript can check the computer's clock and pull the appropriate data based on the clock information.
- **Detect the visitor's browser** : JavaScript can be used to detect the visitor's browser, and load another page specifically designed for that browser.
- **Control Browsers:** JavaScript can open pages in customized windows, where we specify if the browser's buttons, menu line, status line or whatever should be present.
- **Validate forms data:** JavaScript can be used to validate form data at the client-side saving both the precious server resources and time.
- **Create Cookies**: JavaScript can store information on the visitor's computer and retrieve it automatically next time the user visits the page.
- **Add interactivity to the website:** JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on a button.
- **Change page contents dynamically:** JavaScript can randomly display content without the involvement of server programs. It can read and change the content of an XHTML elements or move them around pages.

6.4 Features of Java Script

A Great Programming Tool for HTML:

- JavaScript is powerful scripting languages that help HTML designers to effectively and interactively design websites and web pages in a very simple and efficient way.

Handles Dynamic Behaviour:

- JavaScript is such a powerful scripting language which has features to achieve dynamic behaviour in web pages. Using the features available in JavaScript, the designer can decide to have dynamically placed text at run time.

Browser Detection:

- One of the powerful features of JavaScript is its ability to detect client browser. Browser detection feature of JavaScript helps to achieve independent platforms. JavaScript can detect the type of browser the visitor is using and programmatically switch the page to show customised pages designed for different browsers. Thus by making use of browser detection feature of JavaScript, the designer gets better control over the browser.

Saves Time:

- JavaScript also has the feature of validating data submitted at the client level. This helps in saving the processing time of the server because JavaScript initially creates the validation on the client side.

DOM:

- Client side JavaScript is embedded inside XHTML This embedded JavaScript is used along with DOM (***Document Object Model***) for control over the browser by means of objects.

Popular Scripting language:

- JavaScript has simple rules and procedures that make it easier to use and learn for programmers. This has made JavaScript a popular client-side scripting language.

Interpreted Language:

- It is an interpreted language, meaning that it can be used or executed with ease without pre-compilation.

6.5 Advantages and Disadvantages

Advantages:

- **Less server interaction**: We can validate user input before sending the page to the server. This saves server traffic, which means fewer loads on the server.
- **Immediate feedback to the visitors**: The users don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity**: We can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces**: We can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to the site visitors.
- **Event-driven programming**: JavaScript recognizes when a form 'button' is pressed. This event can have the suitable JavaScript code attached, which will execute when the 'button pressed' event occurs.
- **GUI Features**: Many 'GUI features', such as alerts, prompts, confirm boxes and other GUI elements are handled by client side JavaScript.

Disadvantages:

We cannot treat JavaScript as a fully 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 cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocess capabilities.

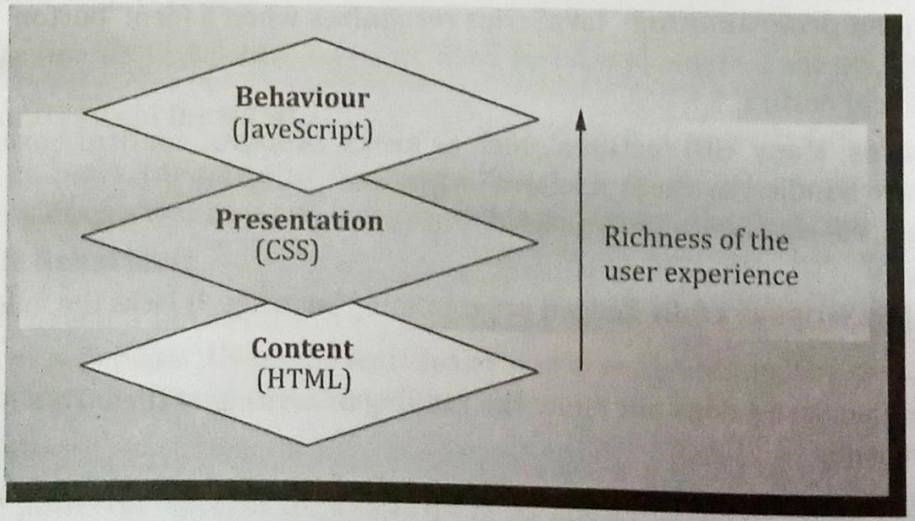
6.6 JAVA VS JAVASCRIPT

Java and Java Script share two very similar names, but they are completely different languages that possess few commonalties. They differ both in their purpose and the applications they can run.

JavaScript	Java
JavaScript is high level object based scripting language.	Java is high level object oriented language.
Java Script does not permit programmers to create stand alone applications.	Java allows creating stand alone applications.
JavaScript is simpler language and easy to learn.	Java is complex language when compared to JavaScript.
JavaScript does not have any compiler	Java has compiler.
JavaScript is interpreted language and it is interpreted by the browser.	Java is compiled and interpreted language.
JavaScript is text-based. We write it to an XHTML document and it is run through a browser. We can alter it after it runs and run it again and again.	Once the Java is compiled, we cannot modify the byte code. We can go back to the original text and alter it, but then we need to compile again.
Prototype based object model	Class based object model.
Lightweight language	Strong type language

6.7 THREE LAYERS OF WEB

When a Web page is designed on the client (browser) side, it might start out as a simple XHTML static page. Later the designer might want to add style to the content to give the viewer a more visually attractive layout. Last, JavaScript code is added to give the viewer the ability to interact with the page, make the page do something. A complete Web page, then, can be visualized as three separate layers as shown below.



When building a web site, we work through these layers from the bottom up:

1. We start by producing the **content** in HTML or XHTML format. This is the base layer, which any visitor using any kind of browser should be able to view.

2. With that done, we can focus on making the site look better, by adding a layer of **presentation** information using CSS. The site will now look good to users able to display CSS styles.
3. Lastly, we can use JavaScript to introduce an added layer of interactivity and dynamic **behavior**, which will make the site easier to use in browsers equipped with JavaScript.

XHTML + CSS + JavaScript = Dynamic Interactive Beautiful Website

6.8 How JavaScript Works?

We know that all the web pages for a particular website is stored in a central place called server. We can access the web page using an URL typed in the browser. Let us see the role of JavaScript in the life cycle of a web page.

Step 1: We type the URL in the browser. The browser sends the http request to Server to get the web page we have requested.

Step 2: Server checks the request and returns the XHTML page to the browser.

Step 3: The Browser gets the XHTML page and displays content. The entire XHTML page is now with browser. The JavaScript code also embedded in the XHTML.

Step 4: Assume that user gets Feedback Form. The user will type the values in the text box and submits the form. We need to validate the form whether user has typed proper email id or not. If JavaScript code is available in the XHTML page to validate the form, then JavaScript code will be executed before sending the form details to server. In this way, we can perform some kind of client side validations using JavaScript. (If there is no JavaScript code, then form details will be sent to server and server has to validate the correctness of the from fields; It consumes more time).

How to enable JavaScript in Internet Explorer?

1. Go to ***Tools*** from the top menu
2. Select ***Internet*** Options
3. Click on the ***Security*** tab
4. Click ***Custom*** Level
5. Scroll down until you see the ***Scripting*** section
6. Ensure that the ***Active Scripting*** option is set at Enabled
7. Click ***OK***.

How do web browsers run JavaScript code?

The Web browser has a special software called JavaScript Interpreter and it's job is to run JavaScript code written inside the web page.

How do we tell web browser to run the JavaScript code?

The JavaScript code will be executed only when something happens in the web page such as the page being loaded or user clicking a button.

6.9 Object Orientation and JavaScript

OOP is a programming technique designed to simplify complicated programming concepts. In essence, object-oriented programming revolves around the idea of user- and system-defined chunks of data, and controlled means of accessing and modifying those chunks. Object-oriented programming consists of **Objects, Methods** and **Properties**. An object is basically a black box which stores

some information. Some of the information in the object may actually be directly accessible; other information may require using a method to access it. The directly accessible bits of information in the object are its properties. The difference between data accessed via properties and data accessed via methods is that with properties.

Objects

The web page document is an object. Any table, form, button, image, or link on the page is also an object. Each object has certain properties (information about the object). For example, the background color of the document is written **document.bgcolor**. We can change the color of the page to red by writing the line: document.bgcolor="red". The contents (or value) of a textbox named "password" in a form named "**myform**" is **document.myform.password.value**.

Methods

Most objects have a certain collection of things that they can do. Different objects can do different things, just as a door can open and close, while a light can turn on and off. A new document is opened with the method **document.open()**. We can write "Hello World" into a document by typing **document.write("Hello World") . open()** and **write()** are both ***methods*** of the object: document.

Events

Events are how we trigger our functions to run. The easiest example is a button, whose definition includes the words **onClick="run_my_function()"**. The onClick event, as its name implies, will run the function when the user clicks on the button. Other events include OnMouseOver, OnMouseOut, OnFocus, OnBlur, OnLoad, and OnUnload.

6.10 JavaScript Syntax

- JavaScript syntax refers to a set of rules that determine how the language will be written (by the programmer) and interpreted (by the browser). The JavaScript syntax is loosely based on the Java syntax.
- A JavaScript consists of JavaScript statements that are placed within the <script>... </script> XHTML tags in a web page.
- We can place the <script> tag containing the JavaScript anywhere within the web page but it is preferred way to keep it within the <head> tags.
- 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 type="text/javascript">
JavaScript code
</script>
```

- **type attribute**: This attribute indicates the scripting language in use and its value should be set to "***text/javascript***".

Which tag is used to add JavaScript?

<script> tag is used to include JavaScript

What are the attributes of script tag?

type and src

What is the value of type attribute for JavaScript? Or What is the MIME type of JavaScript?

text/javascript

6.11 Where to put Scripts?

We can place the java script anywhere on the page that we wish; there are some rules and conventions. We can place Java scripts in any of the following locations:

- Between the XHTML document's head tags.
- Within the XHTML document's body (i.e. between the body tags).
- In an external file (and link to it from the XHTML document).

In the Head Section

- As the head loads before the body, it is guaranteed that it is available when needed. Hence, it reduces the scope of errors.

```
<html>
    <head>
        <script type="text/javascript">
        The script tag has been placed inside the head section.
        </script>
    </head>
</html>
```

- If a programmer wants to execute JavaScript when called, or when an event is triggered, then JavaScript is placed in the HEAD section. JavaScript is placed in the HEAD Section because scripts gets loaded first.

In the Body Section

- Place the script tag in the body section if we want to generate certain content on the page when it loads.

```
<html>
<body>
    <script type="text/javascript">
        document.write ("Here script tag has been placed inside the body section ");
    </script>
</body>
</html>
```

- When a programmer wants to execute JavaScript when the page loads then JavaScript should be placed in the BODY section.

In an External File

- When we want to use the same script on different pages, we can use the external file. The advantage of using external file is that we don't need to rewrite it. External JavaScript file is saved with a '.js' extension.

```
<html>
<head>
    <script type="text/javascript" src="common.js"></script>
</head>
<body>
</body>
</html>
```

- In the above example, the JavaScript code is written in a file called "**common.js**". The file is linked to an XHTML page using "**src**" attribute of the script tag.

We can place the script tag at both places, i.e., inside the head section as well as the body section.

What are the different ways of placing JavaScript code in XHTML?

We can place Java scripts in any of the following locations:

- Between the XHTML document's head tags.
- Between the XHTML document's body tags.
- In both head and body tags
- In an external file.

What are the advantages of placing java script code in external file?

Even though it's nice and easy to just type some JavaScript straight into the XHTML code, it's preferable to include the JavaScript in an external file. This approach provides several advantages:

- It maintains the separation between content and behaviour (XHTML and JavaScript).
- It makes it easier to maintain the web pages.
- It allows easily reuse the same JavaScript programs on different pages of the web site.

6.12 First JavaScript Program

- The first thing we are going to learn is how to make JavaScript write something to the XHTML document. Open notepad and type the following code.

FirstJavaScript.html

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5        <head>
6              <title>My First JavaScript Program</title>
7              <script type="text/javascript">
8                document.write("<h1> Welcome to Java Script </h1>");
9              </script>
10       </head>
11 </html>
```

- This script will write the words, Welcome to Java Script. Now, this is a very simple script, and we can use XHTML to write words on the web page, but the basis of this script will be very important later on, so it is important to learn this one. Let's look at each element of the script.
- First, we are telling the web browser that it is about to see and interpret JavaScript with this tag:

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

- The next part of our script is:

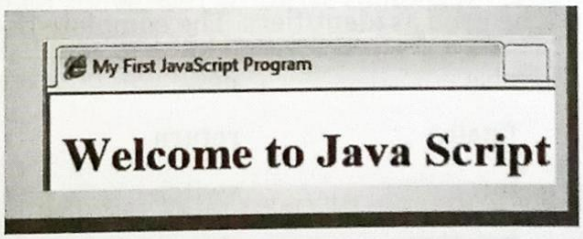
```
document.write
```

- The document is an object.Write is the method (or function) that we want to use on the document. It is presented with a period between the two words: document.write. The write method is actually presented as write() with what we want written inside the parentheses.
- The text that we want to written is called a string in JavaScript, and it is therefore enclosed in quotation marks. It is presented as this:

```
("<h1> Welcome to Java Script </h1>");
```

- Note that we have also added a little XHTML into the mix with the use of tags that will force the text to be printed as heading. We've written the object, the method, and the string so that now the script has everything it needs to run. We've also closed script tag.

Output is:



What is document?

The web page document is an object. This object refers to the web page.

What is write()?

The write() is a method of document object.

What happens when browser does not support JavaScript?

If a user's browser is not JavaScript enabled, the script can't run. Often, nothing will happen, but it is possible that the user will get error reports. This can be prevented by using the <noscript> JavaScript tag. It looks like this:

```
<noscript>
<p>Your Browser Does Not Support JavaScript<p>
</noscript>
```

Is JavaScript case sensitive?

JavaScript is a case-sensitive language. So identifiers Time, TIme and TIME will have different meanings in JavaScript.

6.13 JavaScript Comments

Standard Java - style comment lines and blocks are supported in JavaScript also. In general, any text inside a comment block is ignored by the interpreter. Single - line comments are preceded by a double forward - slash (//), and multiline comments are enclosed by the symbols /* and */. Comments can appear anywhere in your code, including at the top level outside of any functions.

Example:

```
// This is a single line comment.
/*
* This is a multi
*  line comment
*/
/*
      This is also
      a valid multiline comment
*/
```

6.14 JavaScript Keywords

- ECMA - 262 describes a set of keywords that have specific uses, such as indicating the beginning or end of control statements or performing specific operations. By rule, keywords are reserved and cannot be used as identifiers. The complete list of keywords is as follows:

break	else	new	var
case	finally	return	void
catch	for	switch	while
continue	function	this	with
default	if	throw	
delete	in	try	
do	instanceof	typeof	

- The specification also describes a set of reserved words that cannot be used as identifiers. Though reserved words don' t have any specific usage in the language, they are reserved for future use as keywords. The some of the reserved words are shown below.

abstract	enum	int	short
boolean	export	interface	static
byteextends	long	super	
charfinal	native	synchronized	
class	float	package	throws
const	goto	private	transient

6.15 Primitive Data Types

Every variable has a data type. Data type indicates what kind of data the variable holds. In JavaScript, the type of data variable hold can be grouped into two categories:

1. Primitive types
2. Composite types

Primitive types in JavaScript

JavaScript supports five primitive data types:

1. Numbers
2. Strings
3. Booleans
4. Undefined
5. null

From these basic data types, more complex data type (also known as composite types) is derived. Although JavaScript supports five primitive data types, we actually use only numbers, strings, and Booleans to store data. The undefined and null types arise under certain circumstances in programming.

Composite types in JavaScript

From primitive types mentioned above, we can derive composite types. A composite type can consist of numbers, strings, Booleans, undefined and null values. The three types of composite types available in JavaScript are objects, arrays, and functions.

6.15.1 Numbers

- As the name implies, a **number type** refers to numerical values. Numbers can be divided into two groups:
  1. **Floating-point** - are fractional numbers such as 2.45, –3.58, and .97
  2. **Integers** - are whole numbers such as 245, –480, and 3.
- Unlike programming languages such as C++, and Java, the number type in JavaScript include both floating-point and integer values. Integers, in JavaScript, can span in the range of -2^{53} to 2^{53} and floating-point values can be as high as $\pm 1.7976 * 10^{308}$ and as low as $\pm 2.2250 * 10^{-308}$.

6.15.2 Strings

- A string type refers to one or more characters of text. In JavaScript, a string is enclosed inside single or double quotes. "Scripting Language", 'XHTML', 'JavaScript', and "2011", are all examples of strings.
- Why is "2011" a string? As we stated above, a variable of string type is surrounded by double or single quotes. Note, 2011, is an example of **numerical type** because of the absence of the quotes.
- When working with strings, it is important to understand that strings must be either surrounded with single or double quotes. The following, for example, represents an invalid string:

```
"This is not a valid string.'
```

- In the above example, we are not being consistent when creating the string: we start with double quotation marks but end with only single quote. If we need to use an apostrophe (or single quote) in the string, then, we may want to surround it with double quotation marks to make it a valid string. For example, the below one is not a valid string

```
'they're here!'
```

- We have unbalanced quotes. Our string contains a single quote while we start and end our string with a single quote. We can make that a valid string by using double quotes around the string:

```
"they're here!"
```

- On the other hand, if we have to use double quotes in a string, then, we may want to avoid using double quotes around the string. As another example, the following is an invalid string:

```
""JavaScript" is easy."
```

- because again here we have unbalanced quotes. Instead, we can create the string as:

```
'"JavaScript" is easy.'
```

6.15.3 Booleans

- In JavaScript, Boolean logic allows the program to make decisions. A Boolean logic statement consists of a condition that evaluates to either true or false. We can think of the **condition** as a question that has one of two possible answers:
  1. yes or no
  2. ON or OFF
  3. positive or negative
  4. true or false
- When working with Boolean logic, keep in mind there is no "maybe" or third possibility. Thus a Boolean type a can be used anytime there is only one possible outcome out of two possible outcomes.

6.15.4 Undefined type

- The ***undefined type*** refers to those variables or object properties that are either undefined or do not exist. When a variable, for example, is declared without assigning a value to it, it is of undefined type. So

```
var MyAge;
```

- declares a variable called My***Age***. Its type is undefined; it can be confirmed by simply outputting its type using the typeof operator:

```
document.write("MyAge is of type : "+typeof(MyAge));
```

- So our output will show that MyAge is of type : ***undefined***;
- So as long as no value is assigned to a variable, its type is undefined. Once a value is assigned, it will no longer be of type undefined but rather one of other types supported in JavaScript: Numbers, Strings, Boolean, or Null.

6.15.5 Null type

- The ***null*** type indicates an empty value. When a variable is assigned the value ***null***, its type becomes ***null***. A null type variable is a place holder that represents nothing. The following shows an example of a variable of type ***null***:

```
var MyAge = null;
```

- Since null is an empty object, its type is object. We can confirm this by using the typeof operator:

```
document.write("MyAge is of type :"+typeof(MyAge));
```

- That will output: MyAge is of type : object

What are the primitive types of JavaScript?

JavaScript supports five primitive data types: Numbers, Strings,Booleans, Undefined, and null

What are the composite types of JavaScript?

Arrays, functions, and Objects are examples of composite types

What does javascript null mean?

The null value is a unique value representing no value or no object. It implies no object,or null string,no valid boolean value,no number and no array object.

6.16 Declaring Variables

A variable is a named element used to store and retrieve information. With variables, we can store information in one part of the program, and access information from that variable in another part of the program.Before we use a variable, we should declare (create) it. The process of creating a variable is also known as **declaring a variable**. When we declare a variable, we tell the computer to reserve some memory for the program to store some data.

Declaring variable using var command

- To declare a variable in JavaScript, use the **var** command. For instance, the following command creates a variable called age.

```
var age;
```

- With the above command, computer will reserve some memory and allow us to store to or retrieve from that memory with the name we chose, ***age***. Remember at this point our variable ***age*** is null or has a garbage value.
- We can declare more than one variable by separating each with a comma and still use one var command, as:

```
var state, city, zip, country;
```

- When you declare a variable, keep the following JavaScript restrictions in mind to a variable name:
  (1) A variable name cannot be one of the ***reserved words*** in JavaScript.
  (2) The first letter of a variable name must be a letter or an underscore (_). age, year, _month are all valid variable names. However, 11month, 9_ are not valid variable names.
  (3) A variable name cannot contain any space characters. If a variable consists of more than one word, separate each word with a underscore (i.e., first_name, last_name) or make the first letter of each word a capital letter (i.e., FirstName, LastName).
- Remember that variable names are case-sensitive. In JavaScript, a variable named first_name is considered different from first_Name or FIRST_name.

Declaring variable without var command

- As noted above, to explicitly declare (create) a variable in JavaScript we use the **var** command. In JavaScript, however, we can also declare variables implicitly by using the assignment operator to assign a value to the new variable. For example,

```
age = 100;
```

declares a variable called age and assigns the integer value 100 to this variable.

- The following shows some more examples of variables being created when a value is assigned to a variable:

```
<script type="text/javascript">
        sum = x = y = 0;    //declares three variables and each is set to 0.
</script>
```

- Alternatively, we may declare and assign the values using the var command:

```
<script type="text/javascript">
        var sum, x, y;      // declares multiple variable with the var command.
        sum = x = y = 0;    // sets each variable to 0
</script>
```

- What approach should we take to create variables? Well, as a matter of good programming practice, we should **explicitly** declare variables with the **var** command - before using. Explicitly declaring a variable helps the browser efficiently and accurately process and manage the variables.

Assigning values to variables

- After we **declare** a variable, we can assign a value to a variable. Assigning a value to a variable means storing a value to a variable. To assign a value, use the equal sign:

```
var age;
age = 55;
```

- The first line declares a variable called age. The second line stores the number 55 to our variable age. If we want, we could also combine those two lines into one, as :

```
age = 55;
```

- Consider the following example:

```
<script type="text/javascript">
        var name, age;
        name = "Snigdha";
        age = 5;
        document.write (name + " is " + age);
</script>
```

- On the first line, we indicate that it is a JavaScript code. The second line declares two variables. On the third line (name = "Snigdha";), we assign a string value to the variable name. On the fourth line, we assign a numerical value to the variable age. In the line 5, we print the name followed by string " is " and age.

```
Snigdha is 5
```

How to declare a variable in JavaScript?

The variable is declared in JavaScript using var command.

Example: var myAge=10;

6.17 VARIABLE SCOPE IN JAVASCRIPT

The availability or accessibility of a variable within an executing program is referred to as **variable scope.** A variable is available for use if its data can be accessed or used. If a variable is available only in certain part of the program, it is said to have local scope. If, on the other hand, a variable is accessible through out the program, it is said to have global scope.

Local scope

- Variables, for instance, declared in a function have a local scope. Essentially what happens is when the function starts to execute, any variables declared inside the function are created

for local use. The variables can be used until the function ends. Once the function returns (or its execution stops), the variables inside the function are destroyed.

```
<script type="text/javascript">
function testVariableScope()
{
        var scope = "local";
        document.write (scope);
}
 testVariableScope();
 document.write(scope);
</script>
```

Accessible

Not Accessible

Global scope

- Variables declared, for example, outside of a function have global scope. In this case, a function can access not only its locally declared variables but also those variables that are declared outside of the function - global variables. Consider the following example:

```
<script   type="text/javascript">
    var  scopeGlobal = "global";   // global variable
    function testScope ()
    {
        var scopeLocal= "local";
        document.write (scopeLocal);
        document.write (scopeGlobal);
    }
    testScope();
    document.write (scopeGlobal);
</script>
```

Accessible

What is local variable?

A variable which is defined or declared inside the function is termed as local variable.

What is global variable?

The variable is which is defined outside of all the functions and inside the <script> tag is termed as global variable.

6.18 Operators in JavaScript

An operator is a method implemented to replace, modify or combine values represented by variables. With operators, we can evaluate expressions-JavaScript commands that assign values to variables. In the following expression,

```
totalAmountDue = totalBeforeTax + (totalBeforeTax * .05);
```

=, +, * are operators. An expression, as shown above, always has an assignment operator (= sign). An expression can also contain other operators (like +, *, -, /, etc.).

6.18.1 Arithmetic Operators

- Arithmetic operators are used to perform simple mathematical calculations. Arithmetic operators can be divided into two categories:
  (1) **binary** - the operators that work on two elements in an expression.
  (2) **unary** - the operators that work on only one element or variable.
- The below table lists binary arithmetic operators used in JavaScirpt.

Operator	Name	Description	Example

+	Addition operator	Adds two values together. In JavaScript, this operators is also used to combine two elements.	var webPages = 100;
			var images = 25;
			var TotalFiles = webPages + images;
			var message = "Hello " + "Sam";

-	Subtraction operator	Subtracts one value from another.	var Amount = 100;
			var discount = 3.50;
			var totalAmountDue = Amount - discount;

/	Division operator	Divides one value by another	var totalCost = 300;
			var numOfItems = 7;
			var averagePrice = totalCost / numOfItems

*	Multiplication operator	Multiples two values together	var pricePerItem = 5.50;
			var quantityPurchased = 40;
			var totalCost = pricePerItem * quantityPurchased;

%	Modulus operator	Determines the remainder after dividing one value by another	var result = 5 % 2;

- The below table summarizes unary operators.

Operator	Name	Description	Example
++	Increment operator	Increases a value by 1	a = 10; b = a++;
--	Decrement operator	Decreases a value by 1	a = 10; b = a--;
-	Negation operator	Changes the sign of a value	a = -10; b = -a;

6.18.2 Assignment operators

- Expressions use assignment operator (= sign) to assign values. JavaScript supports other assignment operators that we can use to assign values to a variable. Those other assignment operators can assign a value in single operation instead of two. If we have worked in a programming language like C, C++, and Java, then they should be familiar to us. The following table illustrate the same.

Operator	Description	Example	Equivalent to
=	Assigns the value of a variable on the right hand side (of the =) to the variable on the left of = operator.	a = b;	a = b;
+=	Adds two variables and assigns the result to a variable on the left (of +=).	a += b;	a = a + b;
-=	Subtracts two variables and assigns the result to a variable on the left (of -=).	a -= b;	a = a - b;
/=	Divides the variable on the left (of /=) by the variable on the right (of /=) and assigns the result to the variable on the left.	a /= b;	a = a / b;
*=	Multiples the variable on the left (of *=) by the variable on the right (of *=) and assigns the result to the variable on the left.	a *= b;	a = a * b;
%=	Divides the variable on the left (of %=) by the variable on the right (of %=) and assigns the remainder to the variable on the left.	a %= b;	a = a % b;

6.18.3 Relational or Comparison operators

- Comparison operators are also called relational operators. These operators are used to construct and test conditions. To use the operator, we need left-hand-side and right-hand-side. The sides represent the values we want to compare and the comparison operator is placed between the values. The following table summarizes the same:

Operator	Description
==	Returns true if both sides are equal. Note the mathematical equal sign (=) is interpreted as an assignment operator in JavaScript. So, in JavaScript, use two equal symbols (==) when we want to find out if one variable is equal to another.
!=	Returns true if variables are not equal
>	Returns true if the variable on the left is greater than the variable on the right
<	Returns true if the variable on the left is less than the variable on the right
>=	Returns true if the variable on the left is greater than or equal to the value of the variable on the right
<=	Returns true if the variable on the left is less than or equal to the value of the variable on the right

6.18.4 Logical Operators

- The logical operators are used to connect two or more Boolean expressions. Examples of logical operators include the AND operator (&&), OR operator (||), and the NOT operator (!). A condition joined with the AND operator is true only when all of the Boolean expressions are true. For example, in the following

  (4 > 2) && (10 < 15) -- True

  (4 < 5) && (3 < 2) -- False

- In the below table, we list the logical operators available in JavaScript.

Operator	Description
&&	Returns true only when all expressions are true.
||	Returns true only when at least one expression is true.
!	Returns true if an expression is false, and false if an expression is true.

6.18.5 Operator precedence

- The order in which operators are evaluated in an expression is referred to as **operator precedence**. It is particularly noticeable in algebra when solving equations. In algebra, for example, division and multiplication have higher precedence over addition and subtraction. In the following simple arithmetic equation:

  6 + 8 + 2 * 2

- In the above example, multiplication is performed first. Thus the result will be 6 + 8 + 4 = 18, not 32 if we added first and then multiplied by 2!. If we wanted to force a particular precedence order, we may use parenthesis because expressions grouped with parentheses are evaluated first. If we wanted to perform addition before multiplication in the previous example, we may write our expression as:

  (6 + 8 + 2) * 2

Associativity in JavaScript

- Expression evaluation is also influenced by the operator associativity. Associativity means the direction (right to left or left to right) in which entire expression is evaluated. If two or more operators with the same level of precedence appear in an expression, which will be evaluated first? The operator associativity answers this question.

Operator precedence and associativity in JavaScript			
Operator	**Operator Use**	**Operator Associativity**	**Operator Precedence**
()	Method/function call, grouping	Left to right	Highest — 1
[]	Array access	Left to right	1
.	Object property access	Left to right	1
++	Increment	Right to left	2

Operator	Operator Use	Operator Associativity	Operator Precedence
--	Decrement	Right to left	2
-~	Negation	Right to left	2
!	Logical NOT	Right to left	2
~	Bitwise NOT	Right to left	2
delete	Removes array value or object property	Right to left	2
new	Creates an object	Right to left	2
typeof	Returns data type	Right to left	2
void	Specifies no value to return	Right to left	2
/	Division	Left to right	3
*	Multiplication	Left to right	3
%	Modulus	Left to right	3
+	Plus	Left to right	4
+	String Concatenation	Left to right	4
-	Subtraction	Left to right	4
>>	Bitwise right-shift	Left to right	5
<<	Bitwise left-shift	Left to right	5
>, >=	Greater than, greater than or equal to	Left to right	6
<, <=	Less than, less than or equal to	Left to right	6
==	Equality	Left to right	7
!=	Inequality	Left to right	7
===	Identity operator — equal to (and same data type)	Left to right	7
!==	Non-identity operator — not equal to (or don't have the same data type)	Left to right	7
&	Bitwise AND	Left to right	8
^	Bitwise XOR	Left to right	9
|	Bitwise OR	Left to right	10
&&	Logical AND	Left to right	11
||	Logical OR	Left to right	12
?:	Conditional branch	Left to right	13
=	Assignment	Right to left	14
*=, /=, %=, +=,, -=, <<=, >>=, >>>=, &=, ^=, |=	Assignment according to the preceding operator	Right to left	14
,	Multiple evaluation	Left to right	Lowest: 15

Methods to show output

(1) document.write() or document.writeln()
(2) window.alert()
(3) window.status

Methods for taking Input from User

(1) window.prompt()
(2) window.confirm()

6.27.1 document.write() and document.writeln()

- JavaScript provide two methods for displaying text on a web page:
  (1) document.write ("SomeText");
  (2) document.writeln ("SomeText");
- Note we used a semicolon above in each of the two statements. Any JavaScript statement (a complete JavaScript command) must end with a semicolon to mark the end of the statement.
- If we have worked with a object-oriented language, we will immediately recognize that **document** is an object and both **write** and **writeln** are methods. The term "method" in context of object-oriented language means an action or behaviour. As noted earlier, both of these actions print text. But what text do they print? Any text that is enclosed inside the double quotation marks ("") or single quotation mark ('').
- **What is the difference between these two methods?** The "write" method prints the text to the browser without attaching a carriage return at the end of the text while the "writeln" method adds a carriage return to the end of text.
- Consider the following example that uses the "write" method

```
<script type="text/javascript">
        document.write ("<pre>Line 1");
        document.write ("Line 2</pre>");
</script>
```

Which prints: Line1Line 2

- Consider the following example that uses the "writeln" method

```
<script type="text/javascript">
        document.writeln ("<pre>Line 1");
        document.writeln ("Line 2</pre>");
</script>
```

- **The output is:**

```
Line 1
Line 2
```

- We used the <pre> tag to show the difference between the two methods. We can also use XHTML tags to format text inside the write or writeln method. For instance,

```
document.write ("Web Programming By <b>Skyward</b>");
```

will print:

```
Web Programming By Skyward
```

6.27.2 Window.alert()

- The **alert()** method is part of Window object. The **alert()** method show the message in a small message box . The message box contains the string and an "OK" button on it. If the user presses the "OK" button the message box disappears. The message to be displayed has to be passed as parameter to the **alert()** method .
- The window's ***alert()*** method is used to send a warning to the user or alert him or her to do something. The alert box is also commonly used for debugging to find out the results of a calculation, if the program is executing in an expected order, and so on.
- The message for the alert dialog box is any valid expression, variable, or a string of text enclosed in matching quotes, and sent as a single argument to the ***alert()*** method. XHTML tags **are not rendered** within the message string but we can use the escape sequences, \n and \t.

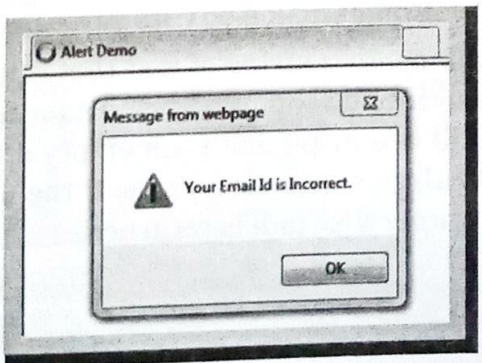
Example:

alert() method demo

Alert.html

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5        <head>
6              <title>Alert Demo</title>
7              <script type="text/javascript">
8                    window.alert("Your Email Id is Incorrect.")
9              </script>
10       </head>
11 </html>
```

Output



6.27.3 Window.status

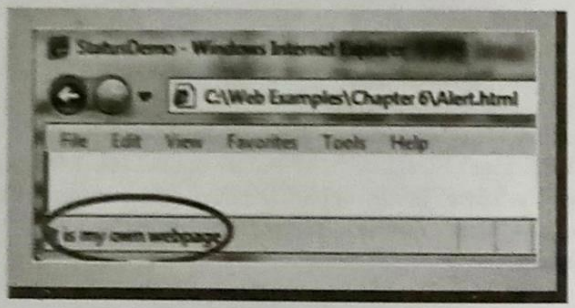
- The status is not a method and it is property of window object. The **window.status** property show the message in status bar. The message to be displayed has to be assigned to the window.status property.

Example:

Window-Status Demo

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6          <title>StatusDemo</title>
7          <script type="text/javascript">
8              window.status="It is my own webpage";
9          </script>
10     </head>
11 </html>
```

Output



6.27.4 window.prompt()

- A prompt() method asks the user for some small amount of information such as a password, completion of a form input, or personal information, such as nickname or title.
- JavaScript does not provide a simple method for accepting user input, the prompt dialog box and XHTML forms are used. The prompt dialog box pops up with a simple textfield box. After the user enters text into the prompt dialog box, its value is returned.
- This method takes two arguments: a string of text that is normally displayed as a question to the user, prompting the user to do something, and another string of text that is the initial default setting for the box. If this argument is an empty string, nothing is displayed in the box. The prompt() method always returns a value. If the user clicks the OK button, all the text in the box is returned; otherwise null is returned.

Example:

window.prompt() demo

```
1  <?xml version="1.0" encoding="ISO 8859-1"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6          <title>Prompt Demo</title>
7          <script type="text/javascript">
8            var name=window.prompt("What is your name?", "");
9            alert("Welcome to My Webpage !... " + name);
10           var age=window.prompt("Tell me your age.", "Your age: ");
11           if (age == null)     // If user clicks the Cancel button
12               window.alert("Not sharing your age with me");
13           }
14           else{
15              window.alert(age + " is young");
16           }
17         </script>
18     </head>
19 </html>
```

Explanation:

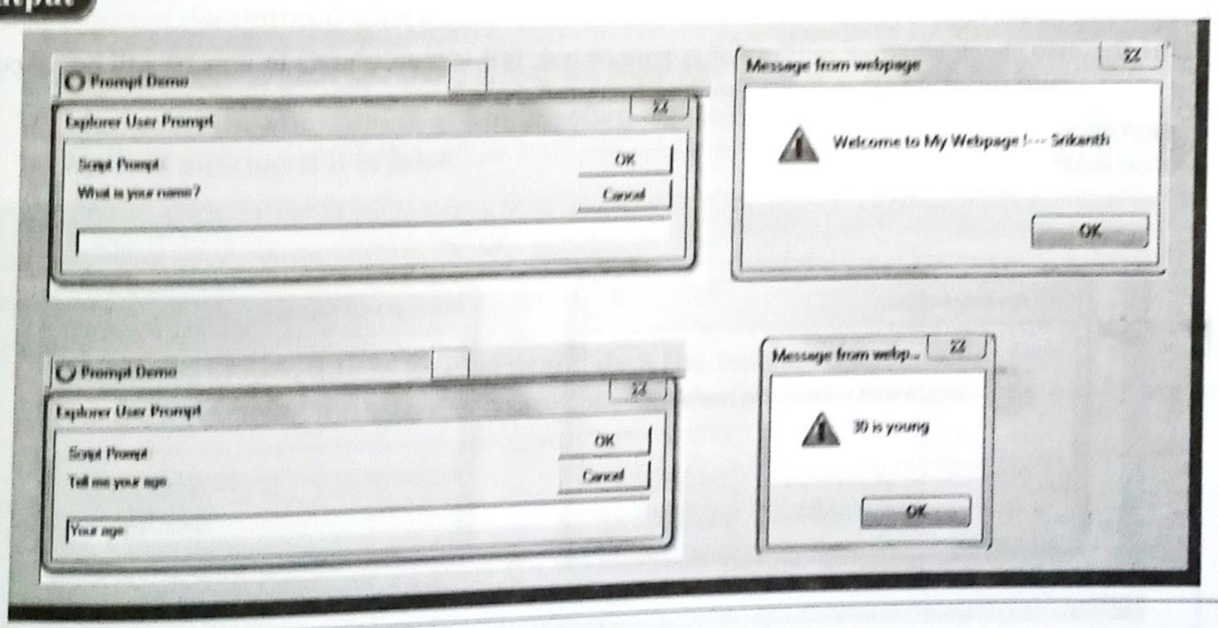
Line 8: The user types his or her name in the prompt textbox, the response is returned and assigned to the variable ***name***. The ***alert()*** method displays that value on the screen.

Line 10: The variable called ***age*** will be assigned whatever the user types into the prompt box. This time a second argument, "***Your age:*** ", is sent to the ***prompt()*** method. When the prompt box appears on the screen, ***Your Age***: will appear inside the box where the user will type his or her response.

Line 11: If the user clicks the Cancel button, the value returned by the ***prompt()*** method is ***null***. This if statement tests to see if the value of ***age*** is ***null***.

Line 15: If the return value was ***null***, this line is printed in the alert dialog box.

Output



27.5 window.confirm()

- The confirm dialog box is used to confirm a user's answer to a question. The user must agree before the action is completed. We will often see this when placing an online order or deleting a file where a yes or no confirmation determines what will happen next. A question mark will appear in the box with an OK button and a Cancel button. If the user clicks the OK button, true is returned; if he or she clicks the Cancel button, false is returned. This method takes only one argument, the question that we are going to ask the user.

Example:

window.confirm() demo

ConfirmJ.html

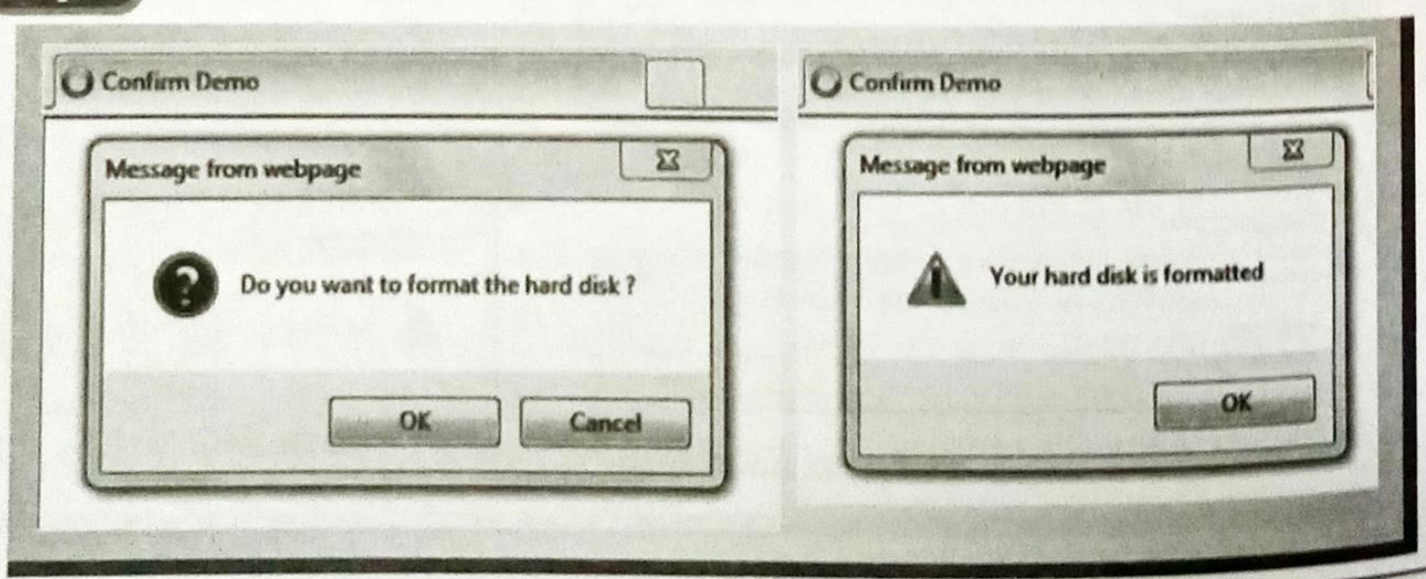
```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5        <head>
6              <title>Confirm Demo</title>
7              <script type="text/javascript">
8                var canFormat=window.confirm(" Do you want to format the hard disk ?");
9                if(canFormat==true)
10                 window.alert(" Your hard disk is formatted");
11               else
12                 window.alert(" Your hard disk is not formatted");
13             </script>
14       </head>
15 </html>
```

Explanation:

Line 8: It display the confirm dialog box. If user clicks on ok, then true value is assigned to the variable **canFormat**. If user clicks on cancel button, then it returns false and **canFormat** will be false value.

Line 9: We check whether canFormat is true or not. If it is true, alert() in line 10 will be executed. Otherwise alert() in line 12 will be executed.

Output



What are the different methods to show output?

(1) document.write() or document.writeln()

(2) window.alert():

(3) window.status

What are the different methods to get input from user?

(1) window.prompt() (2)window.confirm()

What is the use of document.write()?

It is used to display the text in the web page.

What is the difference between document.write() and writeln()?

The "write" method prints the text to the browser without attaching a carriage return at the end of the text while the "writeln" method adds a carriage return to the end of text.

What is alert() method?

The alert() method is part of Window object. The alert() method show the message in a small message box . The message box contains the string and an "OK" button on it. If the user presses the "OK" button the message box disappears. The message to be displayed has to be passed as parameter to the alert() method

What is the use of window.status?

The window.status property show the message in status bar.

What is prompt()?

A prompt() method asks the user for some small amount of information such as a password, completion of a form input, or personal information, such as nickname or title.

What is confirm()?

The confirm dialog box is used to confirm a user's answer to a question

6.28 Conditional Statements

Conditional constructs control the flow of a program. If a condition is true, the program will execute a block of statements and if the condition is false, flow will go to an alternate block of statements. There are three types of conditional statement:

- **if statements**: Which are used when we want the script to execute if a condition is true
- **if...else statements**: Which are used when we want to execute one set of code if a condition is true and another if it is false
- **switch statements**: Which are used when we want to select one block of code from many depending on a situation.

6.28.1 if Statements

- if statements allow code to be executed when the condition specified is true; if the condition is true then the code in the curly braces is executed. Here is the syntax for an if statement·

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

- **Example**: We might want to start the home page with the text "Good Morning" if the time is in the morning. We could achieve this using the following script.

```
<script type="text/JavaScript">
        date = new Date();
        time = date.getHours();
        if (time < 12)
                document.write('Good Morning');
        }
</script>
```

6.28.2 if else Statements

- When we have two possible situations and we want to react differently for each, we can use an if...else statement. This means: "If the conditions specified are met, run the first block of code; otherwise run the second block." The syntax is as follows:

```
if (condition)
{
        code to be executed if condition is true
}
else
{
        code to be executed if condition is false
}
```

- **Example**: We can write Good Morning if the time is before noon, and Good Afternoon if it is after noon.

```
<script type="text/JavaScript">
        date = new Date();
        time = date.getHours();
        if (time < 12) {
                document.write('Good Morning');
        }
        else {
             document.write('Good Afternoon');
        }
</script>
```

6.28.3 Switch Statements

- A switch statement allows dealing with several results of a condition. We have a single expression, which is usually a variable. This is evaluated immediately. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code will execute. Here is the syntax for a switch statement:

```
switch (expression)
{
    case option1:
        code to be executed if expression is what is written in option1
        break;
    case option2:
        code to be executed if expression is what is written in option2
        break;
    case option3:
        code to be executed if expression is what is written in option3
        break;
    default:
        code to be executed if expression is different from option1, option2,and option3
}
```

- **Example**: We get the day number from the date object and based on the day number it will display alert message about the timings of the office. (Remember that dat number starts from 0 to 6, 0 is Monday)

```
<script type="text/javascript">
        var date =new Date();
        var day_number=date.getDay();
        switch(day_number){
            case 0:
            case 1:
            case 2:
            case 3:   alert("Business hours Monday through Thursday are from 9am to 10pm");
                      break;
            case 4:   alert("Business hours on Friday are from 9am to 6pm");
                      break;
            case 5:   alert("Business hours on Saturday are from 11am to 3pm");
                      break;
            default:   alert("We are closed on Sundays and holidays");
        }
</script>
```

6.28.4 Conditional (or Ternary) Operator

- A conditional operator (also known as the ternary operator) assigns a value to a variable based upon a condition:

```
variablename=(condition)?value1:value2
```

- **Example**: Let us say that we have a variable called instruction and a variable called color. If the value of color is red, then we want to assign STOP to instruction; otherwise CONTINUE to be assigned to instruction.

```
var instruction=(color=="red")?"STOP":"CONTINUE";
```

6.29 Iterative or Loop Statements

Looping or Iterative statements are used to execute the same block of code a specified number of times:

- A while loop runs the same block of code while or until a condition is true.
- A do while loop runs once before the condition is checked. If the condition is true, it will continue to run until the condition is false. (The difference between the do and do while loop is that do while runs once whether or not the condition is met.)
- A for loop runs the same block of code a specified number of times.

6.29.1 While loop

- In a while loop, a code block is executed if a condition is true and executes as long as that condition remains true. The syntax is as follows:

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

- In the following example, we can see a while loop that shows the multiplication table for the number 4. This works based on a counter called i; every time the while script loops, the counter increments by one. So, the first time the script runs the counter is 1, and the loop writes out the line 1 × 4 = 4; the next time it loops around the counter is 2, so the loop writes out 2 × 4 = 8. This continues until the condition-that i is no longer less than 11-is true.

```
<script type="text/JavaScript">
        i = 1;
        while (i < 11) {
            document.write(i + " x 4 = " + (i * 4) + "<br />" );
            i ++;
        }
</script>
```

6.29.2 do...while loop

- A do ... while loop executes a block of code once and then checks a condition. For as long as the condition is true it continues to loop. So, whatever the condition, the loop runs at least once (as we can see the condition is after the instructions). Here is the syntax:

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