Modern Programming Languages-JavaScript
Lecture 38

The <script> Tag

The <script> tag (<script>….</script>) in all major browsers interprets contents as JavaScript unless one of the following occurs

Inclusion of language attribute

<script language="VBS">….</script>
Inclusion of type attribute
<script type="text/javascript">….</script>

The type attribute is W3C recommended, it makes the language more common and in many ways more useful

<script> tag is used to delimit the script code from the HTML

- **The script tag causes the browser's JavaScript interpreter to be invoked, the script run and any output produced**
- **The browser is considered the "host" environment**
- **There are other hosts for JavaScript and its variants**

Location of Code

JavaScript may be placed at three places
In the <head> element

Place scripts to be called or when event is triggered here
Ensures script is loaded before called

```
<html>
  <head>
    <script type="text/javascript">
                    //script statements
    </script>
  </head>
```

Location of Code

In the <body> element
Place scripts to be executed when the page loads here
Script generates some or all of the contents of the page

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

Location of Code

External to the HTML file

```
<head>
  <script src="myfile.js">
  </script>
</head>
```

Could be in <head> or <body>
External script should not contain <script> tag

- You can use as many **<script>** tags as you like in both the **<head>** and **<body>** and they are executed sequentially.

-
```
<h1>Ready start</h1>
<script language="Javascript" type="text/javascript">
    alert("First Script Ran");
</script>
<h2>Running...</h2>
<script language="Javascript" type="text/javascript">
    alert("Second Script Ran");
</script>
<h2>Keep running</h2>
<script language="Javascript" type="text/javascript">
    alert("Third Script Ran");
</script>
</h1>Stop!</h1>
</body>
```

Statements

- A script is made up of individual statements
- Javascript statements are terminated by returns or semi-colons same as

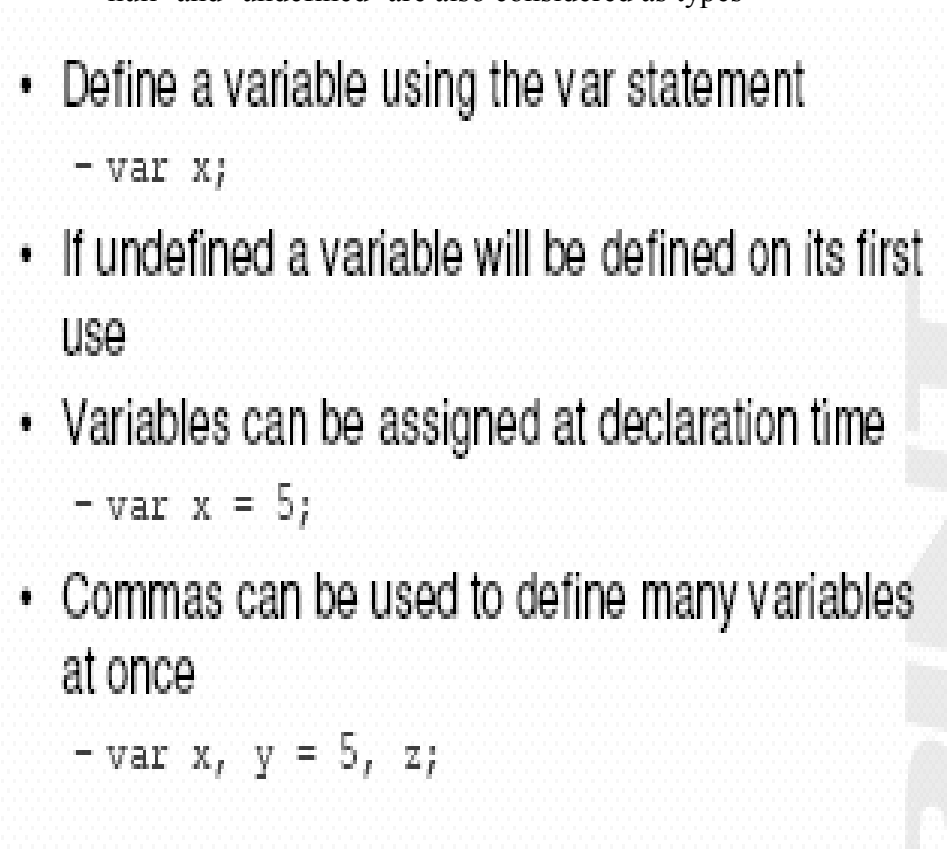
- So, prefer to use semi-colons

```
x=x+1    alert(x) //throws an error while
x=x+1;   alert(x); //does not
```

- Every variable has a data type that indicates what kind of data the variable holds
- Basic data types in JavaScript
- Strings

Strings may include special escaped characters

- Numbers (integers, floats)
- Booleans (true, false)
- 'null' and 'undefined' are also considered as types



- JavaScript is a weekly typed language meaning that the contents of a variable can change from one type to another
- Example

x = "hello"; x = 5; x=false;

While week typing seems beneficial to a programmer it can lead to problems

Arrays

- An ordered set of values grouped together with a single identifier
- Defining Arrays

- var myArray = [1,5,1968,3];
- var myArray2 = [“fakhar”,true,3,-47.2];
- var myArray3 = new Array ();
- var myArray4 = new Array (10);

Arrays

- Arrays in JavaScript are 0 based
- We access arrays by index values

- var myArray = [1,5,1968,3];
- myArray[3] is ‘3’

Difference

Array types (composite type as well) are reference types, so problem of aliasing is there

```
var firstArray = [“Mars”, “Jupiter”, “Saturn” ];
var secondArray = firstArray;
secondArray[0] = “Neptune”;
alert(firstArray); // it has been changed
```

Operators

- Basic Arithmetic
  + , - , / , * , %
- Increment decrement
  ++, - -
- Comparison
  < , > , >= , <= , != , == , === (type equality)
- Logical
  &&, ||, !
- Bitwise Operators
  & , | , ^
- String Operator

- + (used for concatenation)
- document.write("JavaScript" + "is" + "great!");

Type Conversion

- **Converting one type of data to another is both useful and a source of numerous errors in JavaScript**
- var x = "10" – 2 ; //result is 8
- var x = "2" – "2" ; //result is 0
- var x = "2" + "2" ; //result is "22"

Control Statements

- **If**
- **Switch**
- **While**
- **Do-while**
- **For**
- **Continue**
- **Break**
- **For (in)**

Labels and Flow Control

```
outerloop:
  for (var i=0; i < 3; i++)
  {
        document.write("Outerloop: "+i+"<br />");
        for (var j = 0; j < 5; j++)
          {
            if ( j == 3)
              break outerloop;
            document.write("Innerloop: "+j+"<br />");
          }
  }
  document.write("All loops done"+"<br>");
```

```
function name(parameter list)
 {
  function statement(s)
  return;
 }
```

Local Functions

```
function testFunction()
    {

    function inner1() {  document.write("testFunction-inner1<br />");  }

    function inner2() {  document.write("testFunction-inner2<br />");  }

     document.write("Entering testFunction<br />");
     inner1();
     inner2();
     document.write("Leaving testFunction<br />");
    }

    document.write("About to call testFunction<br />");
    testFunction();
    document.write("Returned from testFunction<br />");

  /* Call inner 1 or inner2 here and error */
    inner1();
```