The basics of JAVA SCRIPT

What is JavaScript?

1. JavaScript was designed to add interactivity to HTML pages.

2. JavaScript is a scripting language.

3. A scripting language is a lightweight programming language.

4. It is usually embedded directly into HTML pages.

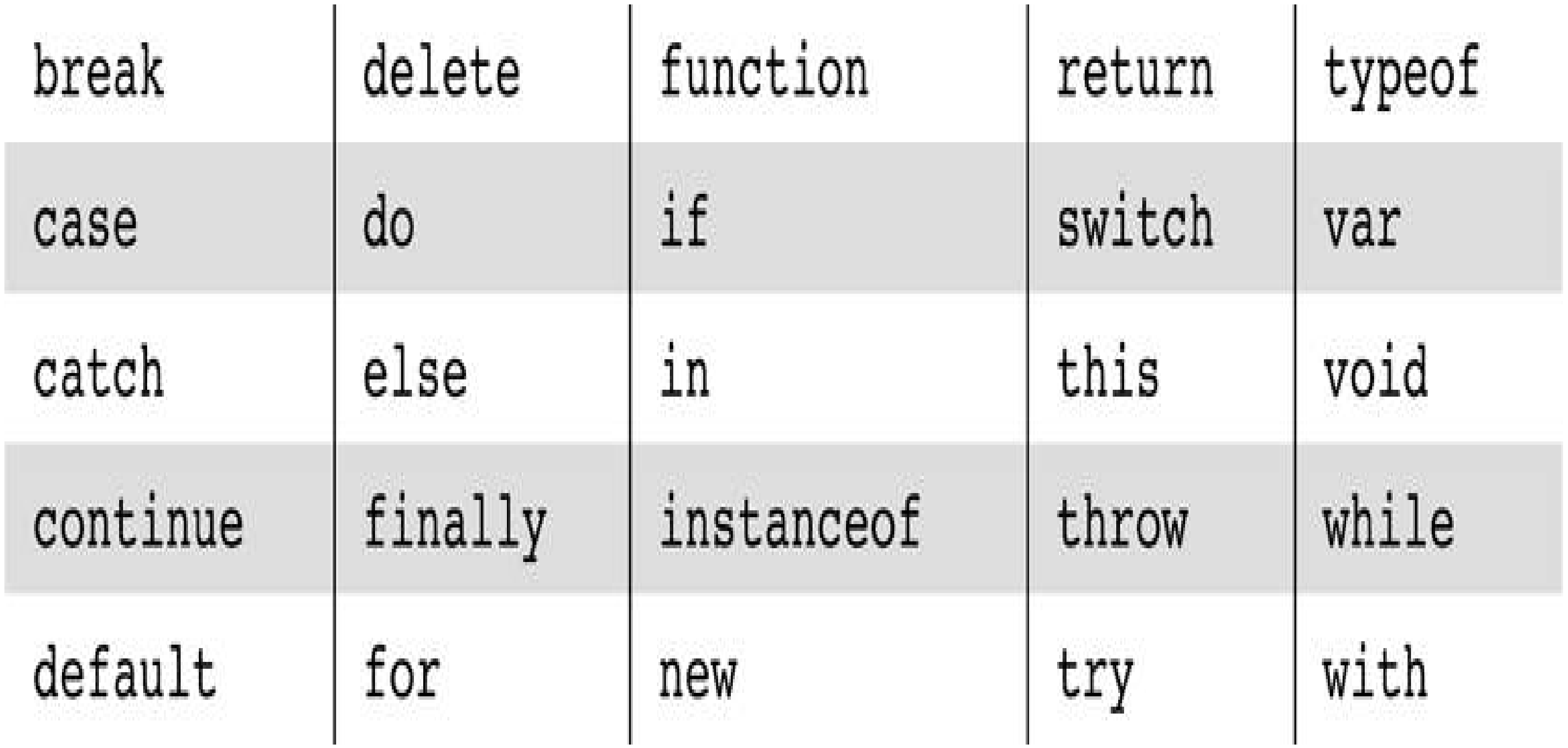
5. JavaScript is an interpreted language (Scripts are executed without preliminary compilations)

Object orientation and Java Script

- ➢ JavaScript Objects

1. In java script objects are collections of properties
2. Each property is either a data property or a function(method) property
3. JavaScript uses non-object types called primitive

JavaScript reserved words



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

JavaScript:

The first Web scripting language, developed by Netscape in 1995 syntactic similarities to Java/C++, but simpler, more flexible in some respects, limited in others

JScript:

Microsoft version of JavaScript, introduced in 1996 same core language, but some browser-specific differences fortunately, IE, Netscape, Firefox, etc. can (mostly) handle both JavaScript & JScript

VBScript:

client-side scripting version of Microsoft Visual Basic

document.write displays text in the page

- Text to be displayed can include HTML tags

- The tags are interpreted by the browser as normal when the text is displayed

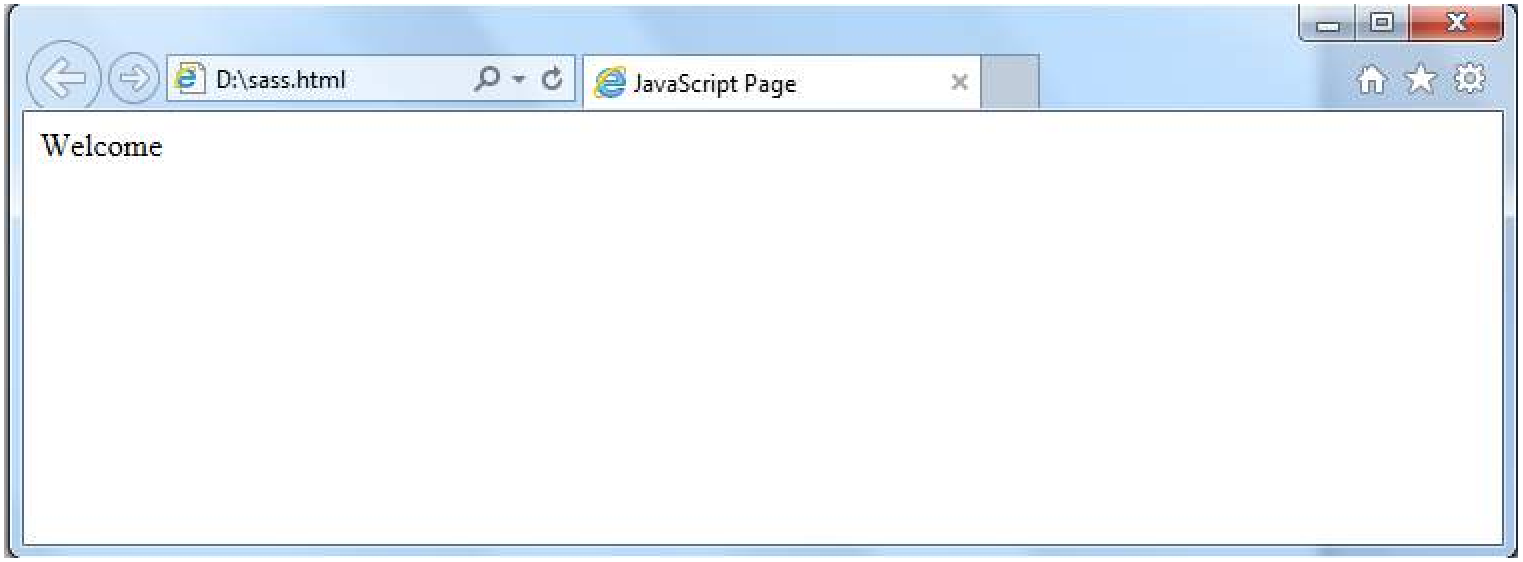
- As in C++/Java, statements end with ;

- JavaScript comments similar to C++/Java

  // starts a single line comment
  /*…*/ enclose multi-line comments

- JavaScript code can be embedded in a Web page using <script> tags

```
<html>
<head>
  <title>JavaScript Page</title>
</head>
<body>

  <script type="text/javascript">
  document.write("Welcome");
  </script>

</body>
</html>
```



```
Welcome
```

Data Types:

➢ Primary Data Types

The primary (primitive) data types are:

- String
- Number
- Boolean

➢Other primitive types

- **Null:** Indicates no value

Declaring variables:

- It is dynamically typed
- Variable can be used for anything
- Variable can have a value of any primitive data type

- Type of the value of a particular appearance of a variable in a program can be determined by the **interpreter**.

- A variable can be declared either by assigning it a value, in which case the interpreter implicitly declares it to be a variable

- JavaScript has only three primitive data types:

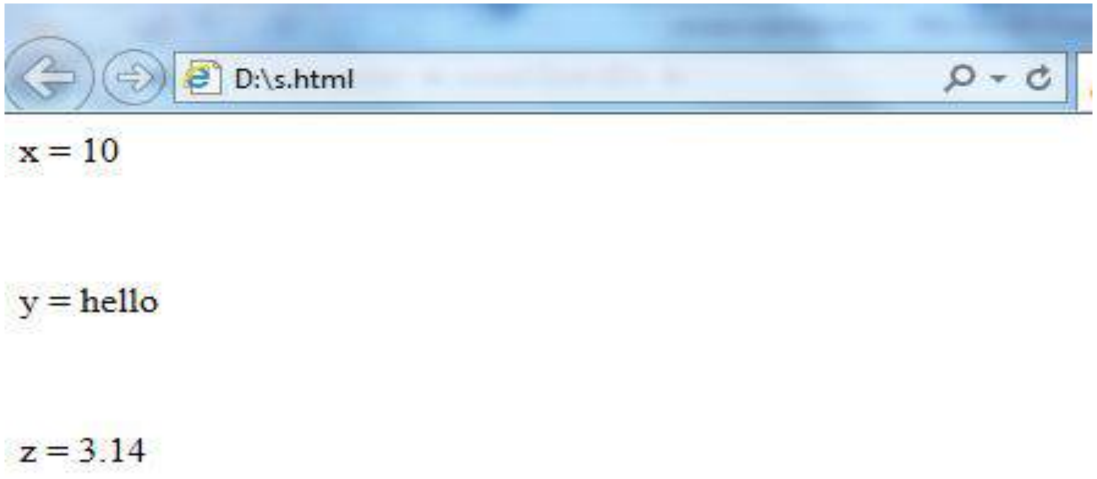
  String : `“hello"   'how do you do?’`
  Number: `12      3.14159`
  Boolean : `true    false`

- Assignments are as in C++/Java
  message = "hello“; pi = 3.14159;
- Variable names are sequences of letters, digits, and underscores that start with a letter or an underscore

- variables names are case-sensitive

- *you don't have to declare variables, will be created the first time used, but it’s better if you use var statements (because of scoping issues)*
  var message, pi=3.14159;
- *Variables are loosely typed, can be assigned different types of values*

```
<html>
<head>
  <title>Data Types and Variables</title>
</head>
<body>
  <script type="text/javascript">
    var x, y,z;
    x=10;
    y="hello";
    z=3.14;
    document.write("<p>x = " + x + "</p></br>");
    document.write("<p>x = " + y + "</p></br>");
    document.write("<p>z = " + z + "</p></br>");
  </script>
</body>
</html>
```



```
x = 10


y = hello


z = 3.14
```

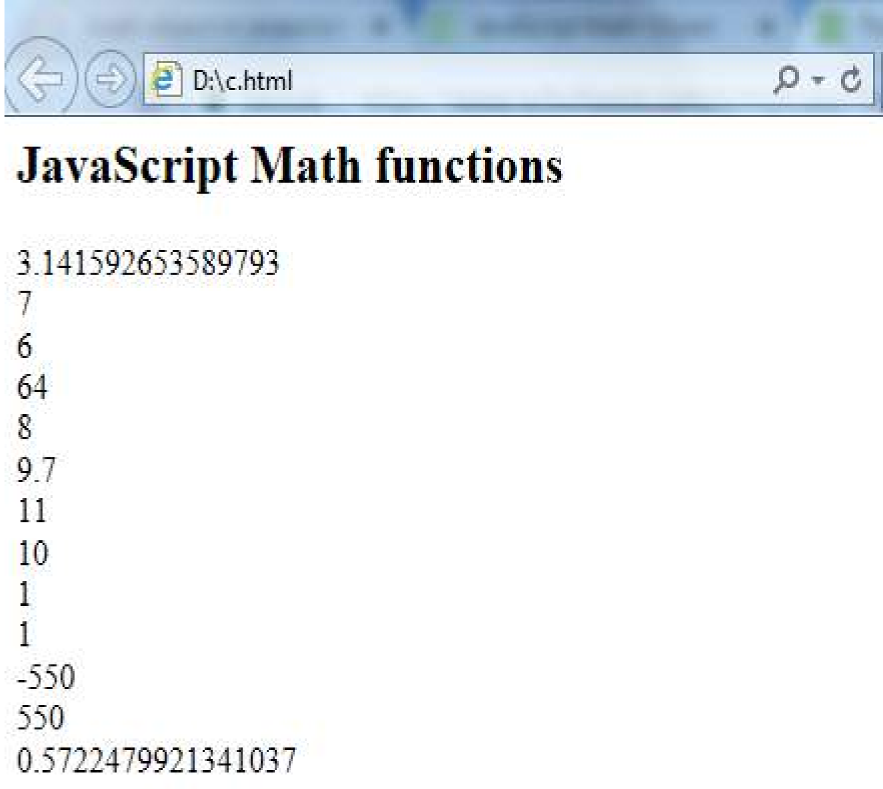
The Math Object

The built-in Math object contains many functions and constants

- **Math.sqrt**
- **Math.pow**
- **Math.abs**
- **Math.max**
- **Math.min**
- **Math.floor**
- **Math.ceil**
- **Math.round**
- **Math.PI**
- **Math.log**

- **Math.random**

```
<html>
<body>
<h2>JavaScript Math functions</h2>
<script>
document.write(Math.PI+"</br>");
document.write(Math.round(6.6)+"</br>");
document.write(Math.round(6.1)+"</br>");
document.write(Math.pow(8, 2)+"</br>");
document.write(Math.sqrt(64)+"</br>");
document.write(Math.abs(-9.7)+"</br>");
document.write(Math.ceil(10.4)+"</br>");
document.write(Math.floor(10.7)+"</br>");

document.write(Math.sin(90 * Math.PI / 180) +"</br>");
document.write(Math.cos(0 * Math.PI / 180)+"</br>");
document.write(Math.min(200, 550, -550, -20)+"</br>");
document.write(Math.max(200, 550, -550, -20)+"</br>");
document.write(Math.random()+"</br>");
</script>
</body>
</html>
```


D:\c.html

JavaScript Math functions

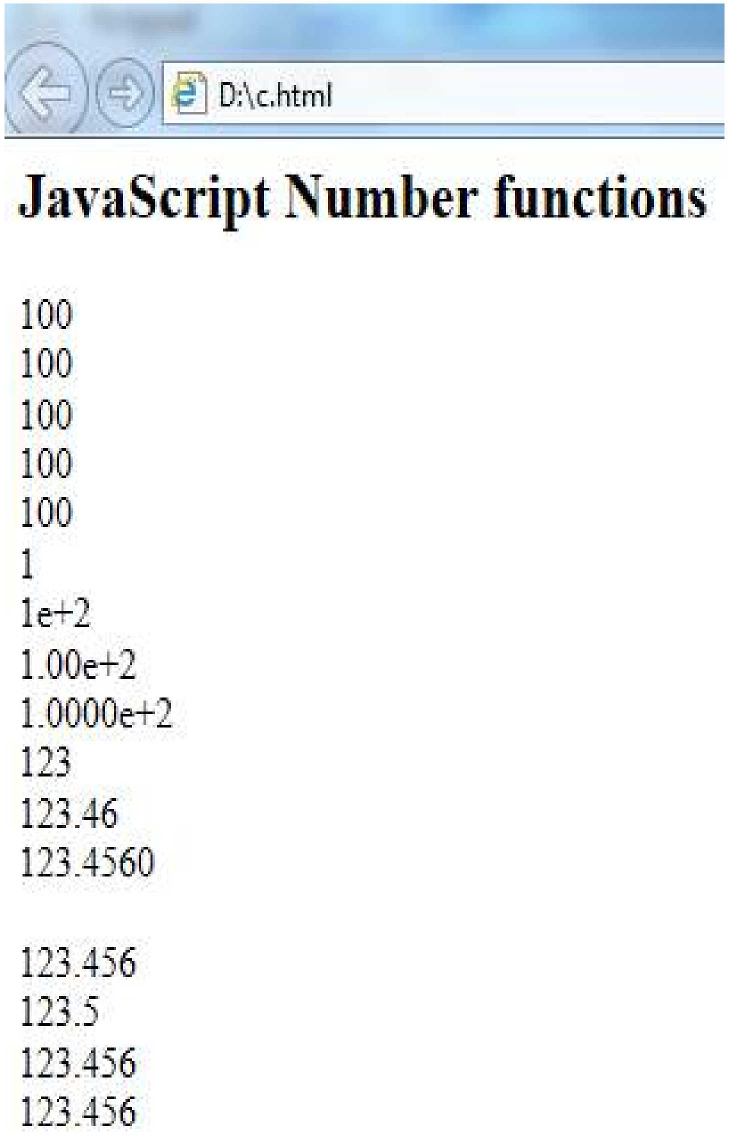
```
3.141592653589793
7
6
64
8
9.7
11
10
1
1
-550
550
0.5722479921341037
```

The number object

JavaScript number methods are methods that can be used on numbers:

Method	Description
toString()	Returns a number as a string
toExponential()	Returns a string, with a number rounded and written using exponential notation.
toFixed()	Returns a string, with a number rounded and written with a specified number of decimals.
toPrecision()	Returns a string, with a number written with a specified length
valueOf()	Returns a number as a number

```
<html>
<body>
<h2>JavaScript Math functions</h2>
<script>
var x=100;
document.write(x.toString()+"</br>");
document.write((100).toString()+"</br>");
document.write((99+1).toString()+"</br>");
document.write(("1"+"0"+"0").toString()+"</br>");
document.write(("1"+0+0).toString()+"</br>");
document.write((1+0+0).toString()+"</br>");
var y=123.456
document.write( x.toExponential()+"</br>");
document.write( x.toExponential(2)+"</br>");
document.write( x.toExponential(4)+"</br>");
document.write( y.toFixed(0)+"</br>");
document.write( y.toFixed(2)+"</br>");
document.write( y.toFixed(4)+"</br>"+"</br>");
document.write( y.toPrecision()+"</br>");
document.write( y.toPrecision(4)+"</br>");
document.write( y.toPrecision(6)+"</br>");
document.write( y.valueOf()+"</br>");
</script>
</body>
</html>
```



D:\c.html

JavaScript Number functions

```
100
100
100
100
100
1
1e+2
1.00e+2
1.0000e+2
123
123.46
123.4560

123.456
123.5
123.456
123.456
```

typeof Operator

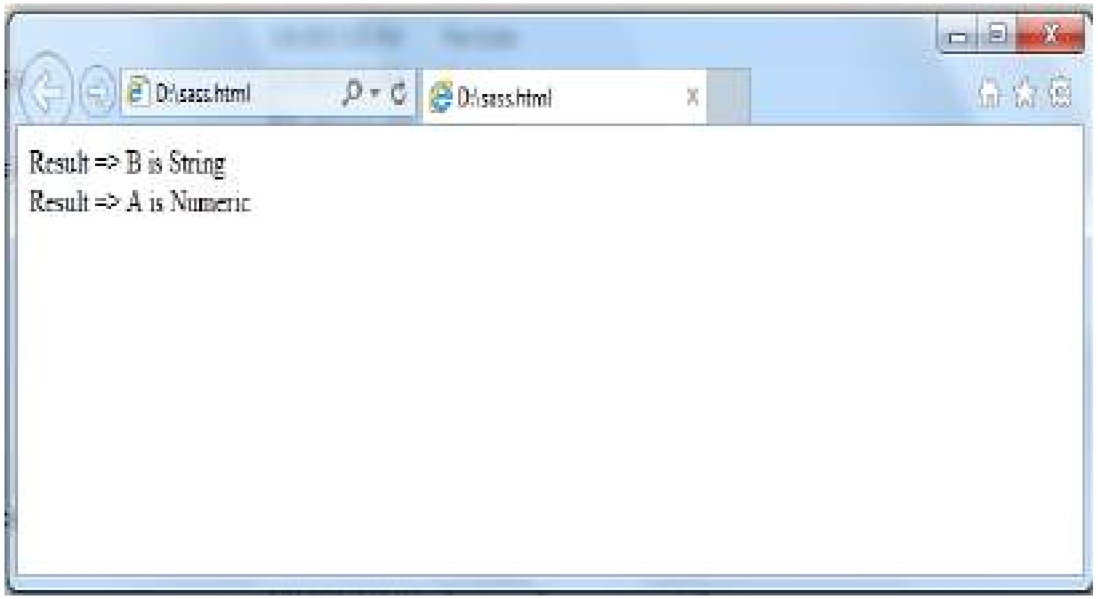
The **typeof** operator is a unary operator that is placed before its single operand, which can be of any type.

Its value is a string indicating the data type of the operand.

The *typeof* operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	"function"
Undefined	"undefined"
Null	"object"

```
<html>
  <body>

    <script type="text/javascript">
        var a = 10;
        var b = "String";
        var linebreak = "<br />";
        var result = (typeof b == "string" ? "B is String" : "B is Numeric");
        document.write("Result => ");
        document.write(result);
        document.write(linebreak);
        var result = (typeof a == "string" ? "A is String" : "A is Numeric");
        document.write("Result => ");
        document.write(result);
        document.write(linebreak);
     </script>

  </body>
</html>
```



```
Result => B is String
Result => A is Numeric
```

String methods

Method	Description
charAt()	Returns the character at the specified index (position)
charCodeAt()	Returns the Unicode of the character at the specified index
concat()	Joins two or more strings, and returns a new joined strings
fromCharCode()	Converts Unicode values to characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
localeCompare()	Compares two strings in the current locale
match()	Searches a string for a match against a regular expression, and returns the matches
replace()	Searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced
search()	Searches a string for a specified value, or regular expression, and returns the position of the match
slice()	Extracts a part of a string and returns a new string

split()	Splits a string into an array of substrings
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLocaleLowerCase()	Converts a string to lowercase letters, according to the host's locale
toLocaleUpperCase()	Converts a string to uppercase letters, according to the host's locale
toLowerCase()	Converts a string to lowercase letters
toString()	Returns the value of a String object
toUpperCase()	Converts a string to uppercase letters
trim()	Removes whitespace from both ends of a string
valueOf()	Returns the primitive value of a String object

The string concatenation operator:

The string concatenation operator denoted by a plus sign(+)

```
var a=“hello”;

document.write(a+”  fst”);
```

Output: hello fst

```
var you=prompt(“enter your name”,” ”);
var age=prompt(“Enter your age in months”, “ “);
var result=you+age;

var result=you.concat(age);

var result=you.concat(age,”thanks”);



var result=you.substr(3,7);
var result=you.toLowerCase();

var result=you.toUpperCase();

var text=“World wide web”;
var words=text.split(“ “);
for(var i=0;i<words.lenght;i++)
document.writeln(words[i]);

var result=you.indexOf(“ABCD”);
```

Search done form 0 to string.length-1, returns -1 if search unsuccessful.

Date Object:

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

Implicit type conversion

The java script interpreter performs several different type conversions
These conversions are called ***coercions***

“august ”+1977 expression evaluates to august 1977
7* “ 3” expression evaluates to 21

Explicit type conversion

Primarily between strings and numbers
Strings that contains numbers can be converted to numbers with the string constructor

```
var str_value=String(value);
(or)
var num=6;
var str_value=num.toString();
```

Strings can be explicitely converted to numbers by number constructor

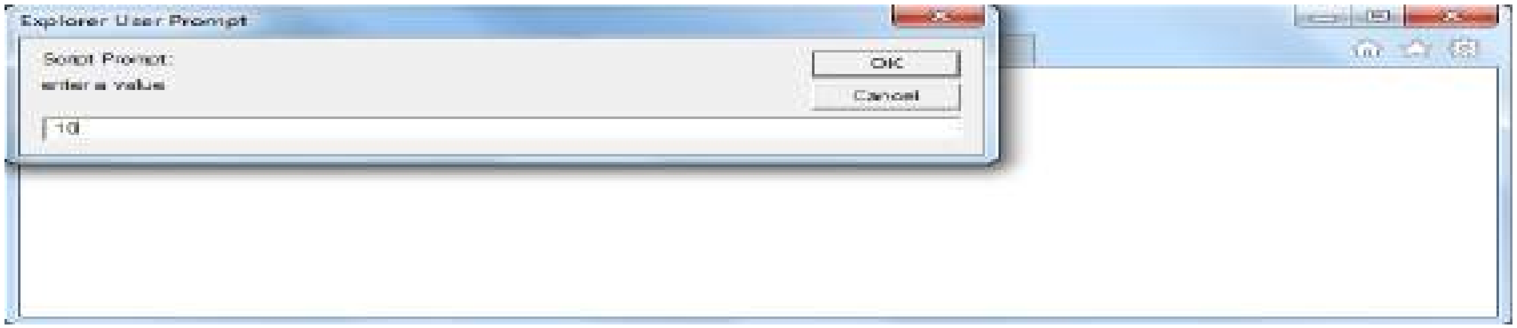
```
var number=Number(astring);
```

Screen output and keyboard input

```
document.write(“welcome”);

alert(“The a value is:”+a+”\n”);

a=prompt(“enter a value”);
```

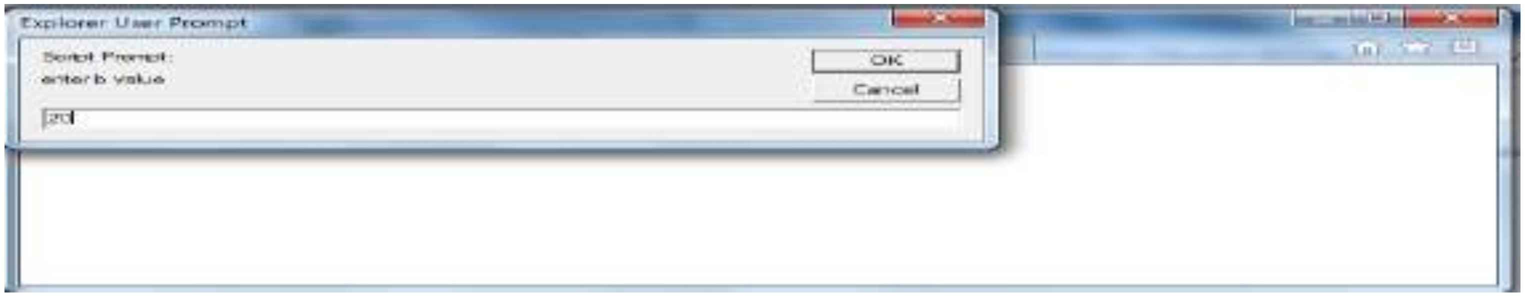
User input:

```
<html>
<head>
  <title>Data Types and Variables</title>
</head>

<body>
  <script type="text/javascript">
    var a, b;
    a=prompt("enter a value"," ");
    b=prompt("enter b value"," ");
    document.write("<p>a = " + a+ "</p>");
    document.write("<p>b = " + b + "</p>");
  </script>
</body>
</html>
```



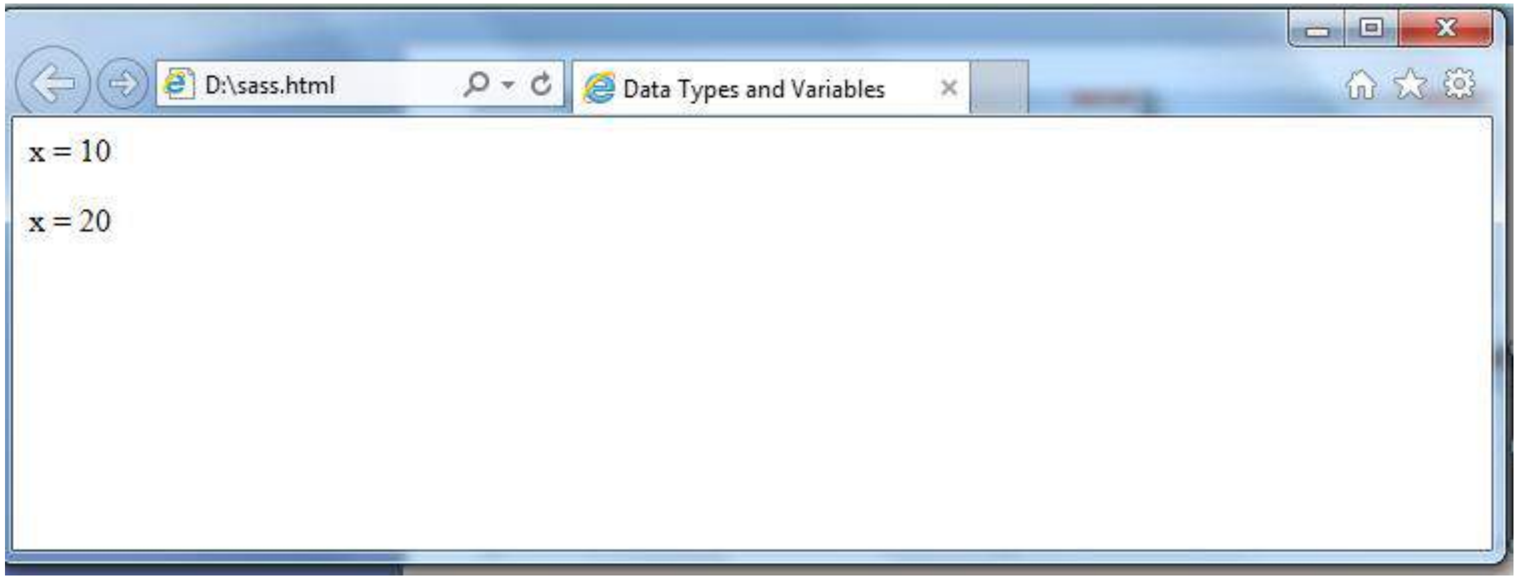
Script Prompt:
enter a value

10



Script Prompt:
enter b value

20

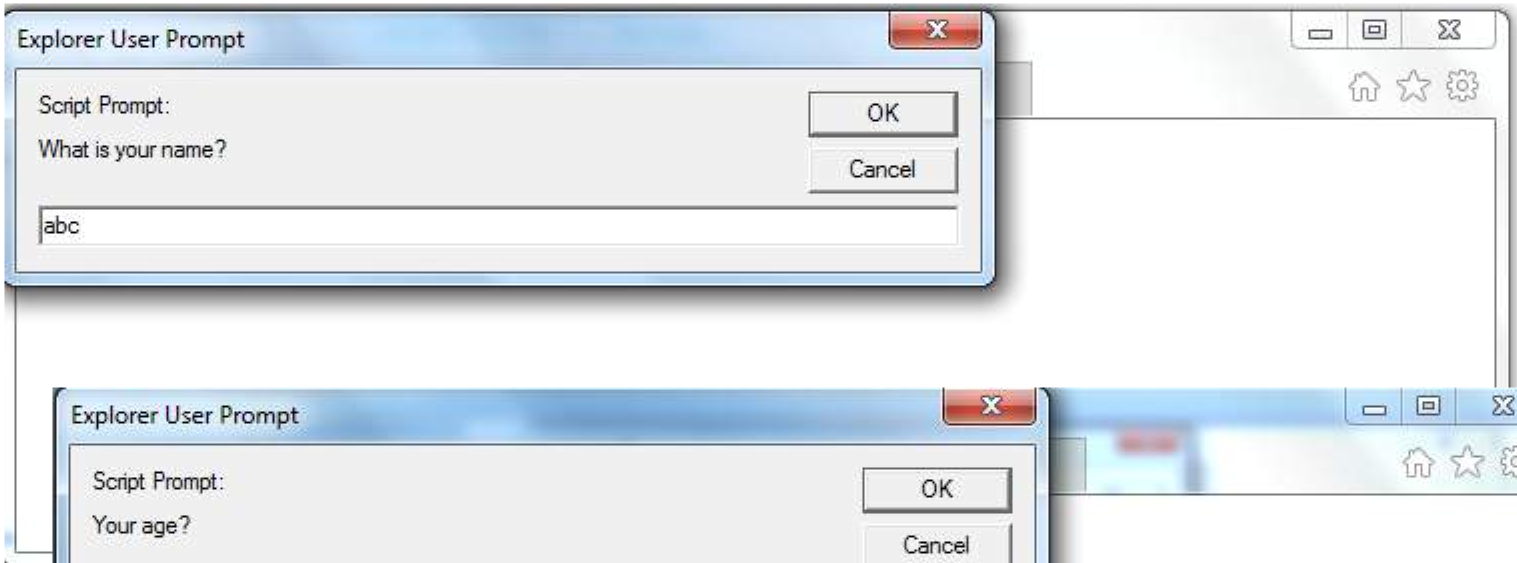


```
x = 10

x = 20
```

Control statements

- Control expressions
- Operator precedence
- Selection statements
    - if-else
- The switch statement

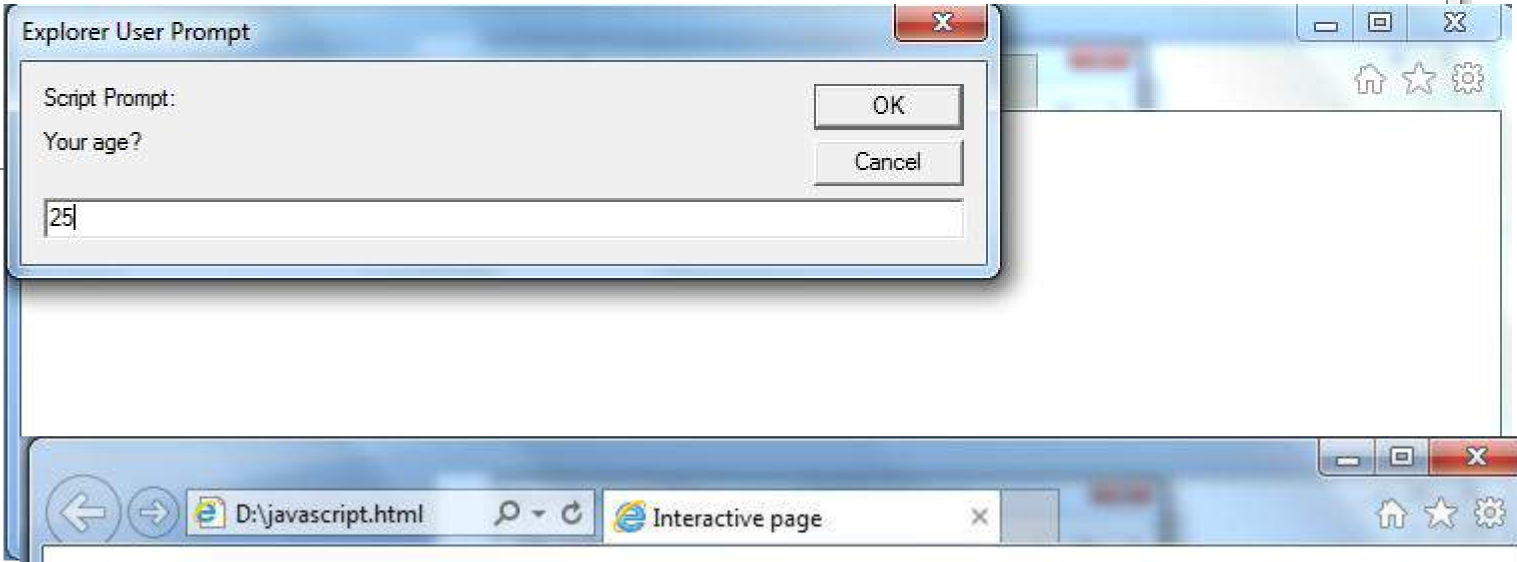
```
<html>
<head>
  <title>Interactive page</title>
</head>
<body>
<script type="text/javascript">
 var userName = prompt("What is your name?", “ ” );

 var userAge = prompt("Your age?", “ “);
 var userAge = parseFloat(userAge);

    document.write("Hello " + userName + ".")
    if (userAge < 18) {
      document.write("  Do your parents know " +"you are online?");
    }
    else {
      document.write("  Welcome friend!");
    }
</script>
</body>
</html>
```


Explorer User Prompt

Script Prompt:

What is your name?

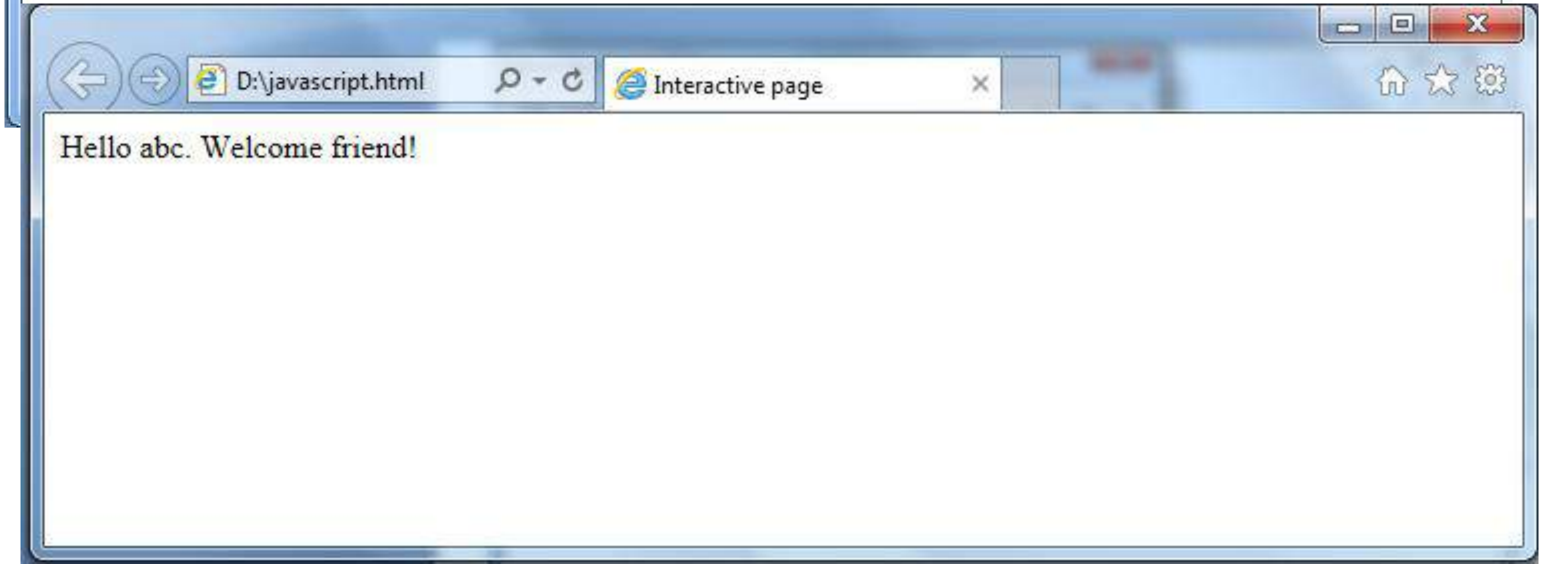
abc


Explorer User Prompt

Script Prompt:

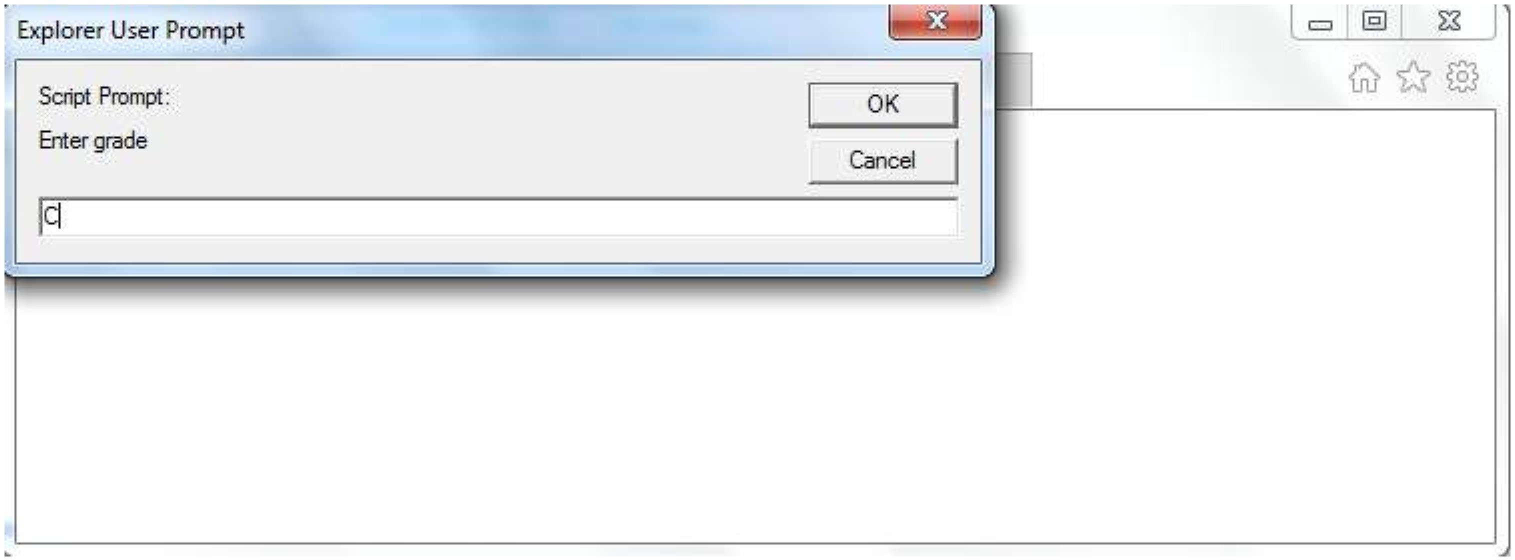
Your age?

25



Hello abc. Welcome friend!

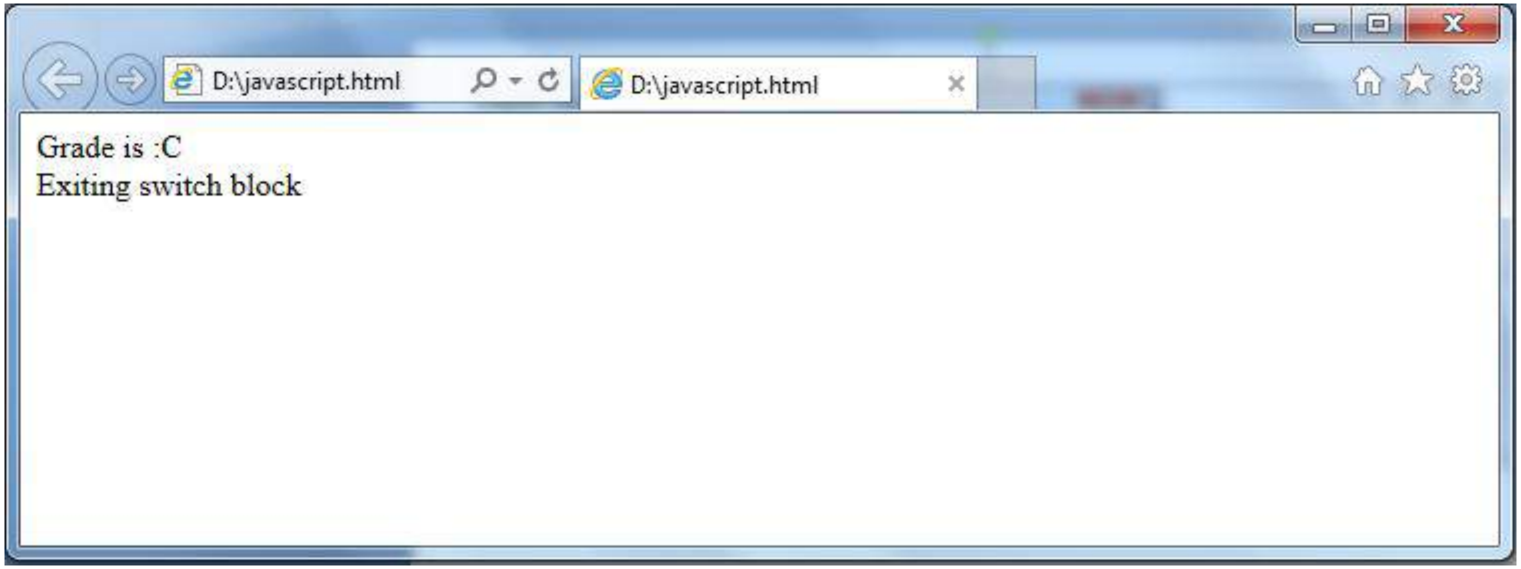
```
<html>
  <body>
        <script type="text/javascript">
         var grade=prompt("Enter grade","");
         switch (grade)
           {
              case 'A': document.write("Grade is :A <br />");
              break;
              case 'B': document.write("Grade is :B<br />");
              break;
              case 'C': document.write("Grade is :C<br />");
              break;
              case 'D': document.write("Grade is :D<br />");
              break;
              case 'F': document.write("Grade is :E<br />");
              break;
              default:  document.write("Unknown grade<br />")
           }
           document.write("Exiting switch block");
</script>
   </body>
</html>
```



Script Prompt:

Enter grade

C



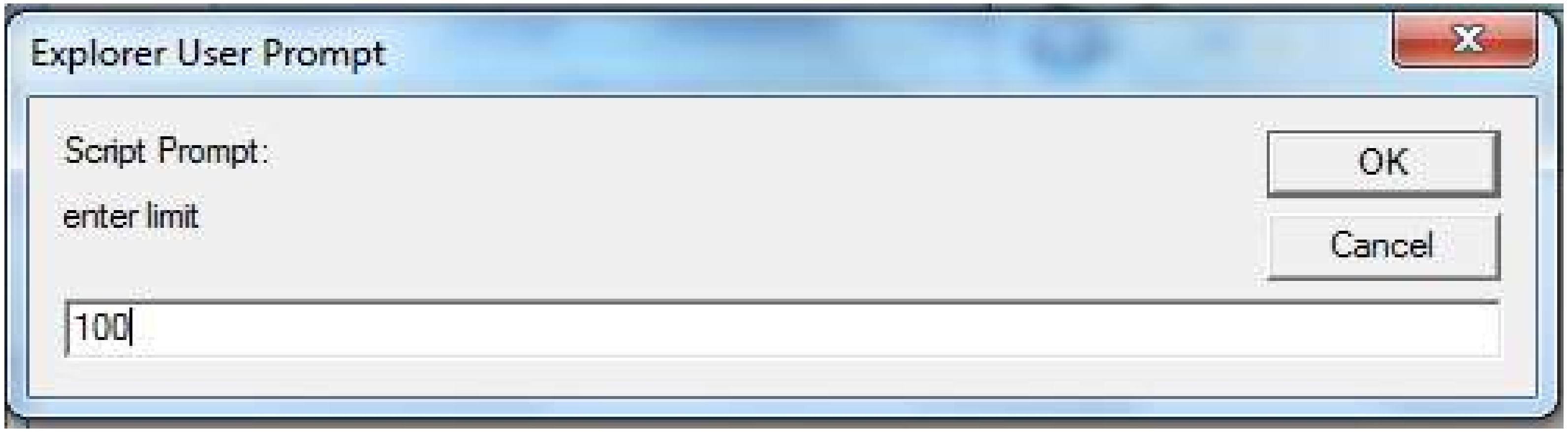
```
Grade is :C
Exiting switch block
```

Loop statements

- While loop
- For loop

```
<script language="javascript">

var done=false;
var msg;
while(done==false)
{
msg=prompt("Enter a string"," ");
document.writeln("<p>"+msg+"</p>");
if(msg.toLowerCase()=="quit")
{
document.writeln("<P> Thanks, Goodbye</p>");
done=true;
}
}
</script>
```

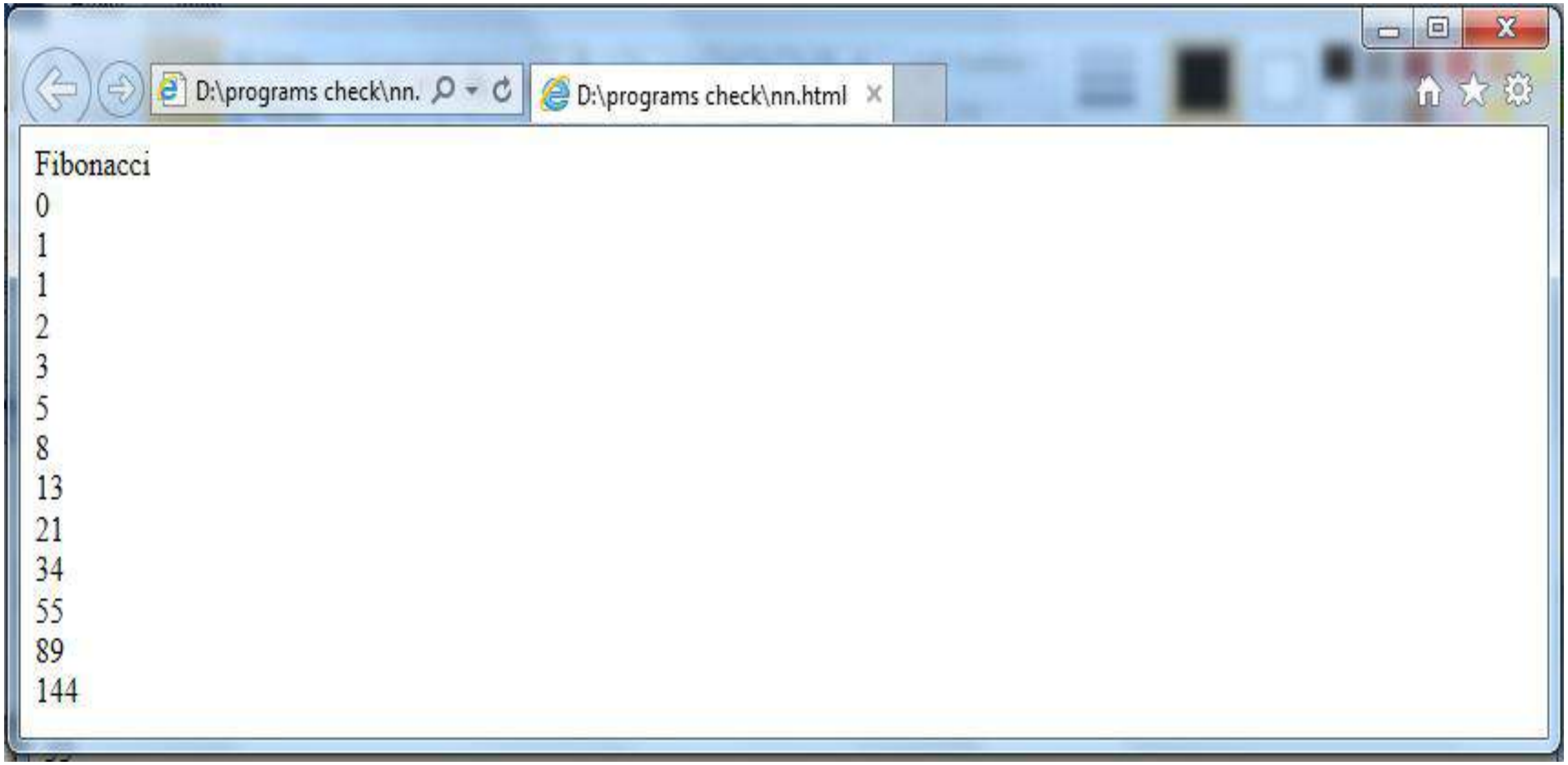
Assignment



Script Prompt:

enter limit

100



```
Fibonacci
0
1
1
2
3
5
8
13
21
34
55
89
144
```

```
<html>
<body>
<script type="text/javascript">
var a=0,b=1,c;
document.write("Fibonacci");
var limit = prompt("enter limit", "");
document.write("<br/>");
document.write(a);
document.write("<br/>");
document.write(b);
document.write("<br/>");

while (b<=limit)
{
c=a+b;
document.write(c);
document.write("<br/>");
a=b;
b=c;
}
</script>
</body>
</html>
```

operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Arrays

An array is an ordered set of data elements which can be accessed through a single variable name

The index is the position of the element in the array starting from 0 to arraylength-1

Basic array functions:

1. Creating arrays
2. Adding elements to an array
3. Accessing array elements
4. Searching an Array
5. Removing array elements

Creating Arrays:

var days=[“Monday” , ”Tuesday” , ”Wednesday” , “Thursday” , “Friday” , “Saturday”];

var days=new Array(“Monday” , ”Tuesday” , ”Wednesday” , “Thursday” , “Friday” , “Saturday”);

var days=new Array(6);

Java script can hold mixed data types:

var days=[“Monday” , ”Tuesday” , 34 , “Thursday” , 56.78 , “Saturday”];

Adding elements to an Array:

```
var days[3]=“Monday”;
var data=[“Monday” , ”Tuesday” , ”Wednesday” , “Thursday” ,
“Friday” , “Saturday”];
data[5]=“Thursday”;
data[3]=56;
```

Accessing Array Elements

```
var days=[“Monday” , ”Tuesday” , ”Wednesday” , “Thursday” ,
“Friday” ];
var len=days.length;
for(var count=0; count<len; count++)
{
document.write(data[count]+” ,“);
}
```

Searching an Array

```
for(var count=0; count<len;count++)
{
if(data[count]=="Tuesday")
{
document.write(data[count]+" , ");
break;
}
```

Removing array elements

```
//ask user to remove what to remove

var rem=prompt(“which item shall I remove?”, “ “);
var tmp=new Array(data.length-1);
var count1=0;
for(var count=0;count<len;count++)
{
if(data[count]==rem)
{
}
else
tmp[count1]=data[count];
count1++;
}
}
```

Sorting array elements

```
<html>
<body>
<script type="text/javascript">
var days = ["sunday", "monday", "tuesday",
"wednesday","thursday","friday"];
days.sort();
var len=days.length;
for(var count=0; count<len; count++)
{
document.write(days[count]+” ,“);
}

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

Functions

- A function is a group of reusable code which can be called anywhere in your program.

- This eliminates the need of writing the same code again and again.

- It helps programmers in writing modular codes.

- Functions allow a programmer to divide a big program into a number of small and manageable functions

Function Definition

The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax

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

 function functionname(parameter-list)

 { statements }

 </script>
```

Calling a Function

```
<html>
<head>
 <script type="text/javascript">
 function display() {
document.write (“welcome"); }
 </script>
</head>
<body>
<p>
Click the following button to call the function</p
<form>
<input type="button" onclick=“display()" value=" Hello">
 </form>
 <p>Use different text in write method and then try...</p>
 </body>
 </html>
```

Function Parameters

```
<html>
 <head>

 <script type="text/javascript">
 function display(name, age) {
document.write (name + " is " + age + " years old."); }
 </script>

 </head>
 <body>
 <p>Click the following button to call the function</p>
<form>
 <input type="button" onclick="display('abc',20)" value="hello">
 </form>
 </body>
 </html>
```

The return Statement

- A JavaScript function can have an optional **return** statement.
- This is required if you want to return a value from a function.
- This statement should be the last statement in a function.

```
<html>
<head>
 <script type="text/javascript">
 function concatenate(first, last) {
var full; full = first + last;
 return full;  }
function secondFunction() {
var result;
result = concatenate(‘ABC', ‘DEF');
document.write (result ); }
</script>
</head>
<body>
 <p>
Click the following button to call the function</p>
<form>
<input type="button" onclick="secondFunction()" value="Call
Function">
 <form>
 </body>
 </html>
```

Local Variables and global variables

A variable that is declared inside a function definition is called local and has scope to that function only

```
<script>
function Show() {
// A local variable is declared in this function.
var a = "Hello World !";

alert("Value of 'a' inside the function " + a); //Hello
World !
}
 </script>
```

A variable that is declared outside of a function definition is called a global variable and its scope is throughout your program means its value is accessible and modifiable throughout your program.

```
<script>
var b;
function Show() {
// A Local variable is declared in this function.
var a = "Hello World !";
alert("Value of 'a' inside the function :" + a); //Hello World !
 //b will have global scope
b = "Hello JavaScript !";
Display();
}
 alert("Value of 'a' outside the function : " + a); // error
 function Display() {
//Since b variable has global scope
alert("Value of 'b' outside the function : " + b); //Hello JavaScript !
}
</script>
```

Constructors in Javascript:

JavaScript constructors are special functions that create and initialize the properties of newly created objects.

JavaScript has a predefined reference variable “this” .
It is a reference to the newly created objects.
 This is used to construct and initialize the properties of the object.

```
function car(new_make, new_model, new_year)
{
this.make=new_make;
this.model=new_model;
this.year=new_year;
}
```

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>

<script>
function car(model, make,year) {
   this.newmodel = model;
   this.newmake = make;
   this.year = year;
}

var car1 = new car("ABC", "XYZ",2000);
var car2 = new car("ABC1", "XYZ1",2010);

document.write("Car1 model is " + car1.newmodel + "<br/>"+"Car2
model is " + car2.newmodel);
</script>
</body>
</html>
```

<u>Output:</u>

```
Car1 model is ABC
Car2 model is ABC1
```

Pattern matching by using regular expressions

• JavaScript has powerful pattern matching capabilities based on regular expressions.

•There are two approaches
1. Based on the methods of RegExp object
2. Based on the methods of String object

• We will cover only string methods for pattern matching.

•The simplest patter-matching method is search, which takes a pattern as a parameter.

•The search method returns the position in the String object at which the pattern matched.

•If no match it returns -1

Example:

```
var str="Rabbits are furry"
var position=str.search(/bits/);
if(position>=0)
  document.write(" 'bits' appears in position", position, "<br/>");
else
   document.write(" 'bits' does not appear in str<br/>");
```

Output:

```
'bits' appears in position 3
```

Character and character-class patterns

- \|( )[ ]{ } ^$* +?.   ---- > Metacharacters

- Metacharacters can themselves be matched by being immediately preceded by a forwardslash.

- A period matches any character other than newline.
  Ex: /snow./   matches   “snowy”, “snowd”, “snowe”

- To match a period in a string, the period must be preceded by a backslash in the pattern.

- /3\.4/ matches 3.4

- /3.4/ matches 3.4 and 374 etc

•[abc] matches ‘a’, ’b’ or ‘c’

•[a-h] matches any lowercase letter from ‘a’ to ‘h’

•^ circumflex character inverts the specified set.

•[^aeiou] matches the first letter other than ‘a’, ‘e’, ‘I’, ‘o’ or ‘u’

•Some character classes are pre-defined and named and can be specified by their names.

Predefined character classes

Construct	**Description**
.	**Any character (may or may not match line terminators)**
\d	**A digit: [0-9]**
\D	**A non-digit: [^0-9]**
\s	**A whitespace character: [\t\n\x0B\f\r]**
\S	**A non-whitespace character: [^\s]**
\w	**A word character: [a-zA-Z_0-9]**
\W	**A non-word character: [^\w]**

/\d\.\d\d/ :Matches a digit, followed by a period followed by two digits

/\D\d\D/ :Matches a single digit

/\w\w\w/ :Matches three adjacent word characters

To repeat a pattern, a numeric qualifier delimited by braces, is attached

/xy{4}z/ matches xyyyyz

*: means zero or more repetitions
+: means one or more repetitions
?: means one or none repetition

/x*y+z?/ : Matches strings begin with any number of x's followed by one or more y's, possibly followed by z.

/\d+\.\d*/ : string of one or more digits followed by a decimal point and possibly by more digits

/[A-Za-z]\w*/ : matches a letter, followed by zero or more letters, digits, or underscores

Anchors:

/^pearl/ :matches "pearls are pretty"
/gold$/ :matches I like gold

<u>Program to validate email id and password using ‘pattern matching’</u>

```
 <body>
<script>
var email=prompt("enter u r email-id:");
var pwd=prompt("enter password:");
var p=email.search(/^[a-zA-Z][a-zA-Z0-9]*@[a-zA-Z]+\.[a-zA-Z]+$/);
var q=pwd.search(/^[a-zA-Z0-9]+$/);
var st1,st2;
if(p==-1)
st1="Sorry, Email is Invalid";
else
st1="Great, your email id is valid";
if(q!=-1)
{
   if(pwd.length<6)
        st2="password must be minimum six characters..";
    else
         st2="Your password is also valid";
}
else
  st2="please enter only alphanumeric for password";
alert(st1+"\n"+st2);
</script>
</body>
```

JavaScript Objects:

- ❖Java script is not an object oriented language
- ❖It tries to be an object oriented language
- ❖Objects is a thing
- ❖A class is a description of something
- ❖An object is an instance of a class
- ❖Object exists in computer memory and which does the work
- ❖An object is described by class
- ❖A class is specialized through inheritance
- ❖A class usually contains both items and processing capabilities

The Javascript execution environment

- **Imperative and structured :**

  - ❖ JavaScript supports all the structured programming syntax in C (e.g., if statements, while loops, switch statements, etc.)

  - ❖ One syntactic difference from C is automatic semicolon insertion, in which the semicolons that terminate statements can be omitted

- **Dynamic typing**

- **Object based**

- **Functional**

The Document Object Model

1. The DOM developed by W3C

2. DOM Level3 is the current version

3. It allows a specification that would allow java programs and JavaScript scripts that deal with HTML documents to be portable among different browsers

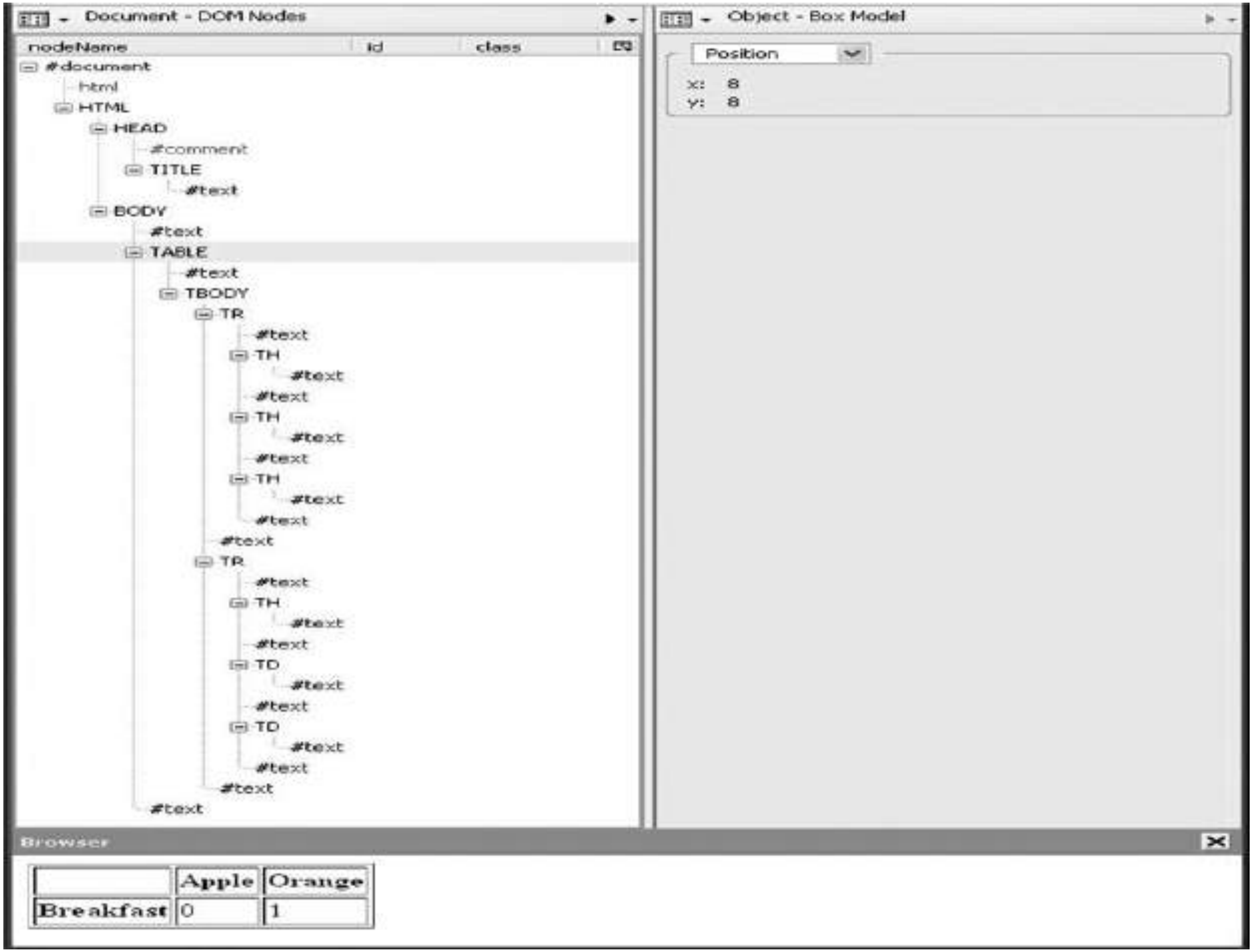
4. The DOM is an application programming interface(API)

5. It defines an interface between HTML documents and application programs

6. Documents in DOM is like tree structure

- IE8+, FX#, C10+ provide a way of viewing DOM structure
- After displaying a document with IE9 ,

  select Tools/DeveloperTools
- The lower-left area of the resulting display will show an elided version of the DOM structure
- By clicking all of the elided icons the whole structure will be displayed
- IE9 Developer tools are helpful for developing and analyzing HTML documents.



Document - DOM Nodes

```
nodeName                     id          class
#document
  html
  HTML
    HEAD
      #comment
      TITLE
        #text
    BODY
      #text
      TABLE
        #text
        TBODY
          TR
            #text
            TH
              #text
            #text
            TH
              #text
            #text
            TH
              #text
            #text
          #text
          TR
            #text
            TH
              #text
            #text
            TD
              #text
            #text
            TD
              #text
            #text
          #text
      #text
```

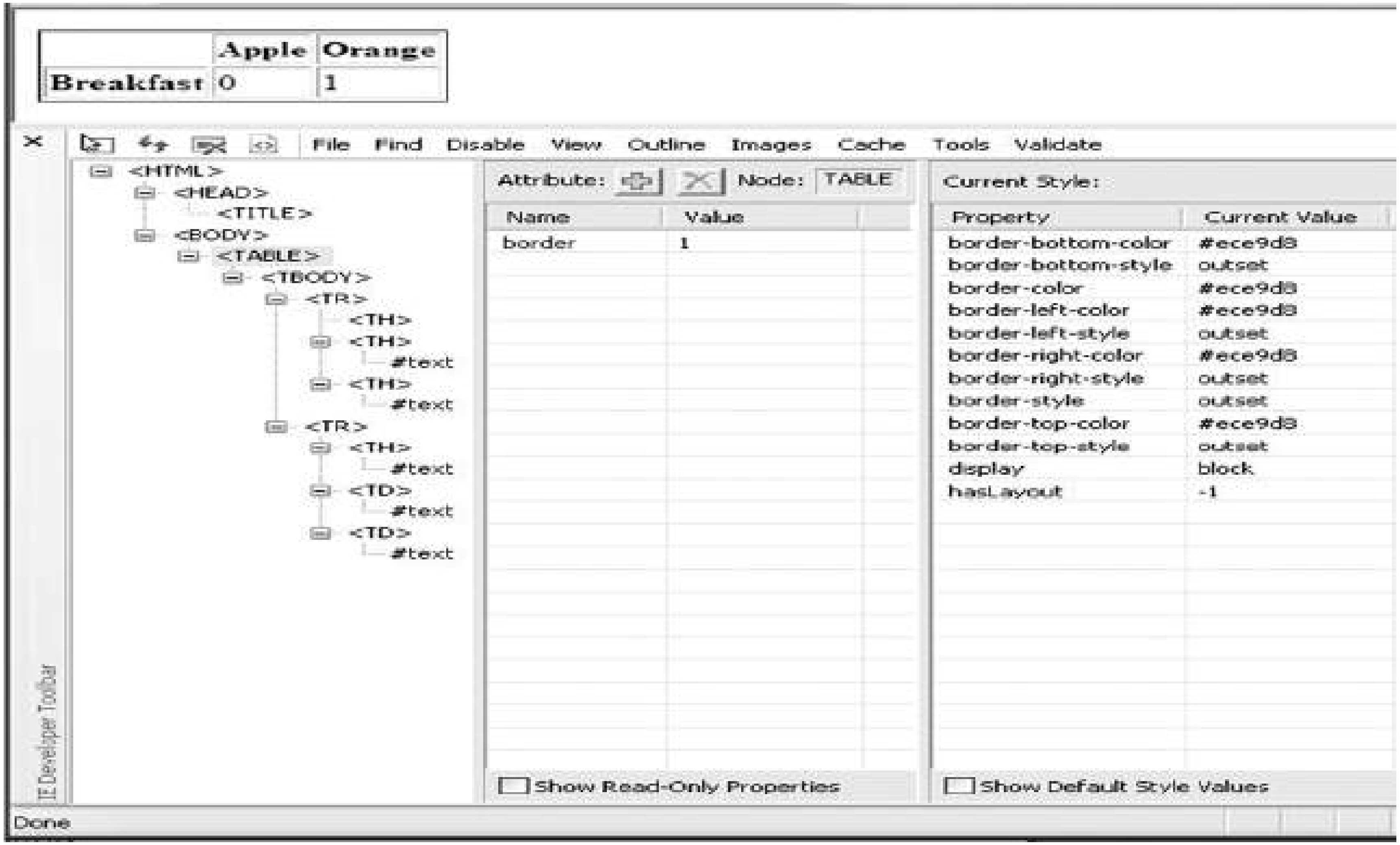
Object - Box Model

Position

```
x:  8
y:  8
```

Browser

	Apple	Orange
Breakfast	0	1



	Apple	Orange
Breakfast	0	1

```
<HTML>
  <HEAD>
    <TITLE>
  <BODY>
    <TABLE>
      <TBODY>
        <TR>
          <TH>
          <TH>
            #text
          <TH>
            #text
        <TR>
          <TH>
            #text
          <TD>
            #text
          <TD>
            #text
```

Name	Value
border	1

Property	Current Value
border-bottom-color	#ece9d8
border-bottom-style	outset
border-color	#ece9d8
border-left-color	#ece9d8
border-left-style	outset
border-right-color	#ece9d8
border-right-style	outset
border-style	outset
border-top-color	#ece9d8
border-top-style	outset
display	block
hasLayout	-1

Element access in Javascript

The elements of an HTML document have corresponding objects that are visible to an embedded JavaScript script

The addresses of these objects are required for making dynamic changes

There are several ways the object can be addressed in JavaScript

Approach 1-> DOM 0 model:
Use the forms and the elements arrays of the Document object, which is referenced through the document property of the window object

```
<html>
<head>
<title>Access to form elements </title>
</head>
<body>
<form>
<input type="button"  name="turnItOn"/>
</form>
</body>
</html>
```

We refer to the address of the JavaScript object that is associated with HTML element as the DOM address of the element

The DOM address of the button is:

Var dom=document.forms[0].elements[0];

Disadvantage:

If new button is added before the turnItOn button in the form, the DOM address shown wrong.

Approach 2-> Use Element names:

The element and its enclosing elements must use name attributes

```
<html>
<head>
<title>Access to form elements </title>
</head>
<body>
<form name=“myform” >
<input type=“button”  name=“turnItOn”/>
</form>
</body>
</html>
```

Using name attributes the button’s DOM address is:

```
Var dom=document.myform.turnItOn;
```

Disadvantage:

The XHTML 1.1 standard does not allow the name attribute in the form element

Approach 3-> Use JavaScript method getElementById:

It is defined in DOM1

Using this the button's DOM address is:

Var dom=document.getElementById(turnItOn);

To access the arrays, the DOM address of the form object must be first obtained

```
<form id="vehicleGroup">
 <input type="checkbox" name="vehicles" value="car"/>car
 <input type="checkbox" name="vehicles" value="truck"/>truck
 <input type="checkbox" name="vehicles" value="bike"/>bike
 <input type="checkbox" name="vehicles" value="lorry"/>lorry
</form>

var numChecked=0;
var dom = document.getElementById("vehicleGroup");
for(index=0; index< dom.vehicles.length; index++)

if(dom.vehicles[index].checked)
numChecked++;
```

Events and event handling

An event is a notification that something specific has occurred.

An event is an object that is implicitly created by the browser and the java script system in response to something having happened.

An event handler is a script that is executed in response to the appearance of an event.

Event can be handled in 2 ways:

1. By assigning tag attribute
2. By assigning handler address to object properties.

Event	Tag Attribute
blur	onblur
change	onchange
click	onclick
dblclick	ondblclick
focus	onfocus
keydown	onkeydown
keypress	onkeypress
keyup	onkeyup
load	onload
mousedown	onmousedown
mousemove	onmousemove
mouseout	onmouseout
mouseover	onmouseover
mouseup	onmouseup
reset	onreset
select	onselect
submit	onsubmit
unload	onunload

Attribute	Tag	Description
onblur	`<a>`	The link loses the input focus
	`<button>`	The button loses the input focus
	`<input>`	The input element loses the input focus
	`<textarea>`	The text area loses the input focus
	`<select>`	The selection element loses the input focus
onchange	`<input>`	The input element is changed and loses the input focus
	`<textarea>`	The text area is changed and loses the input focus
	`<select>`	The selection element is changed and loses the input focus
onclick	`<a>`	The user clicks on the link
	`<input>`	The input element is clicked
ondblclick	Most elements	The user double clicks the left mouse button
onfocus	`<a>`	The link acquires the input focus
	`<input>`	The input element receives the input focus
	`<textarea>`	A text area receives the input focus
	`<select>`	A selection element receives the input focus

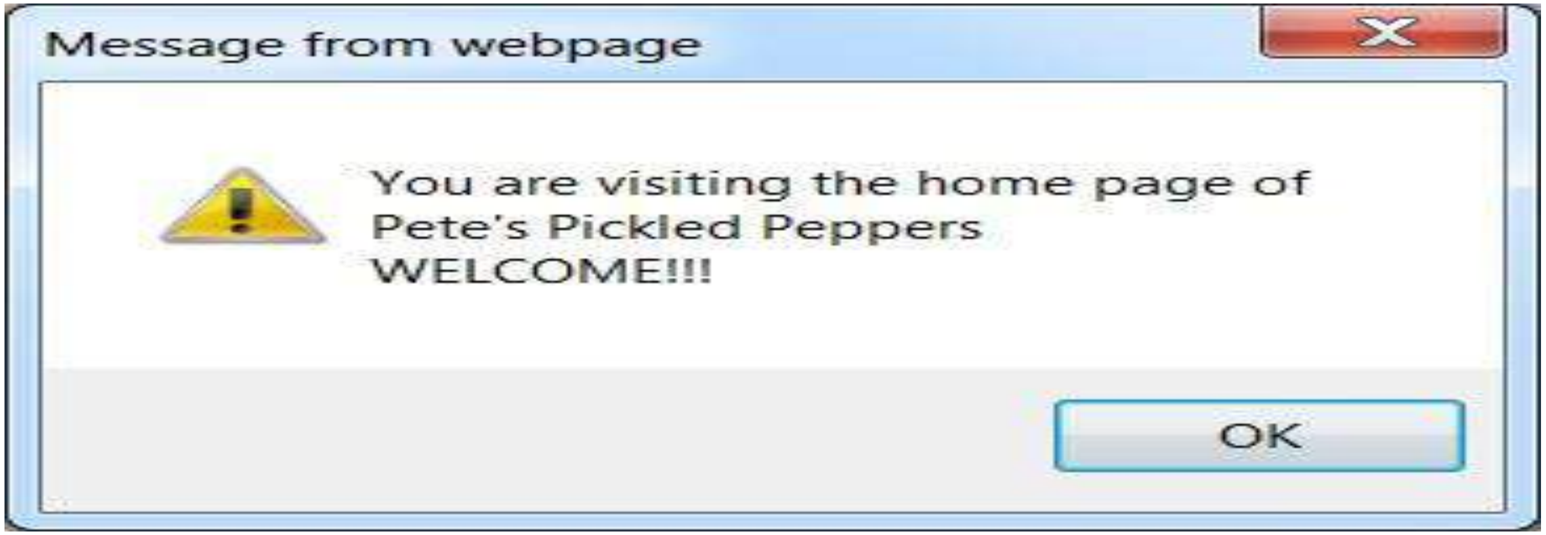
Handling events from body elements

The events most often created by body elements are load and unload.

Giving an alert message when the body of the document has been loaded.

```
<!DOCTYPE html>
<!-- load.html
     A document for load.js
     -->
<html lang = "en">
  <head>
    <title> load.html </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript"  src = "load.js" >
    </script>
  </head>
  <body onload="load_greeting();">
  <!-- use the onload attribute of <body> to specify the event
handler.  -->
    <p />
  </body>
</html>
```

```
// load.js
//   An example to illustrate the load event

// The onload event handler

function load_greeting () {
  alert("You are visiting the home page of \n" +
       "Pete's Pickled Peppers \n" + "WELCOME!!!");
}
```



You are visiting the home page of
Pete's Pickled Peppers
WELCOME!!!

Handling events from button elements

click()

By assigning tag attribute

```
<!DOCTYPE html>
<html lang = "en">
  <head>
    <title> radio_click2.html </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript"  src = "radio_click2.js" >
    </script>
  </head>
  <body>
    <h4> Cessna single-engine airplane descriptions </h4>
    <form id = "myForm"  action = "">
      <p>
        <label> <input type = "radio"  name = "planeButton” value = "152"
         onclick=“planechoice(152)”/>   Model 152         </label>  <br />
        <label> <input type = "radio"  name = "planeButton"  value = "172"
        onclick=“planechoice(172)”/>     Model 172 (Skyhawk) </label>  <br />
        <label> <input type = "radio"  name = "planeButton“ value =182
         onclick=“planechoice(182)” />  Model 182 (Skylane) </label>    <br />
        <label> <input type = "radio"  name = "planeButton"  value=210
        onclick=“planechoice(210)” /> Model 210 (Centurian) </label>   </p>
    </form>
</body>
</html>
```

```
// radio_click2.js

function planeChoice (plane) {
switch (plane) {
   case "152":  alert("A small two-place airplane for flight training");
                    break;
   case "172":  alert("The smaller of two four-place airplanes");
                    break;
   case "182“:  alert("The larger of two four-place airplanes");
                    break;
   case "210“:  alert("A six-place high-performance airplane");
                     break;
   default: alert("Error in JavaScript function planeChoice");
     break;
  }
}
```

By assigning address of the handler to the object properties

```
<!DOCTYPE html>
<html lang = "en">
  <head>
    <title> radio_click2.html </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript"  src = "radio_click2.js" >
    </script>
  </head>
  <body>
    <h4> Cessna single-engine airplane descriptions </h4>
    <form id = "myForm"  action = "">
      <p>
        <label> <input type = "radio"  name = "planeButton" value = "152"
        />   Model 152         </label>  <br />
        <label> <input type = "radio"  name = "planeButton"  value = "172"
        />     Model 172 (Skyhawk) </label>  <br />
        <label> <input type = "radio"  name = "planeButton" value =182
         />  Model 182 (Skylane) </label>    <br />
        <label> <input type = "radio"  name = "planeButton"  value=210
         /> Model 210 (Centurian) </label>   </p>
    </form>
<!– script for registrering the event handler-->
<script type = "text/javascript"  src = "radio_click2r.js" >    </script>   </body>
</html>
```

```
// radio_click2.js

function planeChoice (plane) {
var dom=document.getElementById(“myForm”);
For (var index=0; index< dom.planeButton.length; index++) {
 if ( dom.planeButton[index].checked) {
                  plane = dom.planeButton[index].value;
                  break;
                  }
}
switch (plane) {
    case "152":  alert("A small two-place airplane for flight training");
                         break;
    case "172":  alert("The smaller of two four-place airplanes");
                         break;
    case "182“:  alert("The larger of two four-place airplanes");
                         break;
    case "210“:  alert("A six-place high-performance airplane");
                          break;
    default: alert("Error in JavaScript function planeChoice");
      break;
  }
}
```

```
// radio_click2r.js

var dom= document.getElementById(“myForm”);
dom. elements[0].onclick = planeChioce;
dom. elements[1].onclick = planeChioce;
dom. elements[2].onclick = planeChioce;
dom. elements[3].onclick = planeChioce;
```

Handling events from text box and password elements

Four different events are:
blur
focus
change
select

The Focus Event:
The textbox is focused to blur it

Handling events from text box and password elements

```
<!DOCTYPE html>
<html lang = "en">
  <head>
    <title> nochange.html </title>
    <meta charset = "utf-8" />
    <style type = "text/css">
      td, th, table {border: thin solid black;}
    </style>
<!-- Script for the event handlers -->
    <script type = "text/javascript"  src = "nochange.js" >
    </script>
  </head>
  <body>
    <form action = "">
      <h3> Coffee Order Form </h3>
      <table>
        <tr><th> Product Name </th><th> Price </th> <th> Quantity </th></tr>
        <tr><th> French Vanilla (1 lb.) </th><td> $3.49 </td><td>
        <input type = "text"  id = "french"  size ="2" /> </td></tr>
        <tr><th> Hazlenut Cream (1 lb.) </th><td> $3.95 </td><td>
        <input type = "text"  id = "hazlenut"  size = "2" /> </td></tr>
        <tr><th> Columbian (1 lb.) </th><td> $4.59 </td><td>
        <input type = "text"  id = "columbian"  size = "2" /></td></tr>
      </table>
```

```
<p>
      <input type = "button"  value = "Total Cost“  onclick = "computeCost();" />
      <input type = "text"  size = "5"  id = "cost“     onfocus = "this.blur();" />
    </p>
    <p>
      <input type = "submit"  value = "Submit Order" />
      <input type = "reset"  value = "Clear Order Form" />
    </p>
  </form>
 </body>
</html>
```

```
// nochange.js
 function computeCost() {
  var french = document.getElementById("french").value;
  var hazlenut = document.getElementById("hazlenut").value;
  var columbian = document.getElementById("columbian").value;

// Compute the cost

  document.getElementById("cost").value =   totalCost = french * 3.49 + hazlenut * 3.95
   + columbian * 4.59;
}  //* end of computeCost
```

Validating form input: password checking

```
<!DOCTYPE html>
<html lang ="en">
 <head>
  <title> Illustrate password checking> </title>
  <meta charset = "utf-8" />
  <script type = "text/javascript"  src = "pswd_chk.js" >
  </script>
 </head>
 <body>
  <h3> Password Input </h3>
  <form id = "myForm"  action = "" >
   <p>
   <label> Your password <input type = "password" id = "initial"  size = "10" />
   </label>
   <br /><br />
    <label> Verify password <input type = "password"  id = "second“ size = "10" />
   </label>
   <br /><br />
   <input type = "reset"  name = "reset" />
   <input type = "submit"  name = "submit" />
   </p>  </form>
<script type = "text/javascript"  src = "pswd_chkr.js“></script>
 </body> </html>
```

```
// pswd_chk.js
function chkPasswords() {
  var init = document.getElementById("initial");
  var sec = document.getElementById("second");
  if (init.value == "") {
    alert("You did not enter a password \n" +
          "Please enter one now");
    return false;
  }
  if (init.value != sec.value) {
    alert("The two passwords you entered are not the same \n" +
          "Please re-enter both now");
    return false;
  } else
    return true;
}
// pswd_chkr.js
// Register the event handlers for pswd_chk.html

document.getelementById(“second”).onblur=chkPasswords;
document.getelementById(“myform”).onsubmit=chkPasswords;
```

Validating form input: Name and phone number

```
<!DOCTYPE html>
<html lang = "en">
 <head>
  <title> Illustrate form input validation> </title>
  <meta charset = "utf-8" />
  <script type = "text/javascript"  src = "validator.js" >
  </script>
 </head>
 <body>
  <h3> Customer Information </h3>
  <form action = "">
   <p> <label>
     <input type = "text"  id = "custName" />Name (last name, first name, middle initial)
    </label>
    <br /><br />
    <label>
     <input type = "text"  id = "phone" />Phone number (ddd-ddd-dddd)     </label>
    <br /><br />
    <input type = "reset"  id = "reset" />
    <input type = "submit"  id = "submit" /> </p>
  </form>
</body></html>
```

```
// validator.js
function chkName() {
  var myName = document.getElementById("custName");
  var pos = myName.value.search(
            /[A-Z][a-z]+, [A-Z][a-z]+, [A-Z]+$/);
  if (pos != 0) {
    alert("The name you entered (" + myName.value + ") is not in the correct form. \n" +
          "The correct form is: " + "last-name, first-name, middle-initial \n" +
          "Please go back and fix your name");
    return false;
  } else
    return true;
}
function chkPhone() {
  var myPhone = document.getElementById("phone");
  var pos = myPhone.value.search(/^\d{3}-\d{3}-\d{4}$/);
  if (pos != 0) {
    alert("The phone number you entered (" + myPhone.value + ") is not in the correct form. \n"
+ "The correct form is: ddd-ddd-dddd \n" +"Please go back and fix your phone number");
    return false;
  } else
    return true;
}
```