JavaScript

The Scripting Language of the Web

Need for Scripting

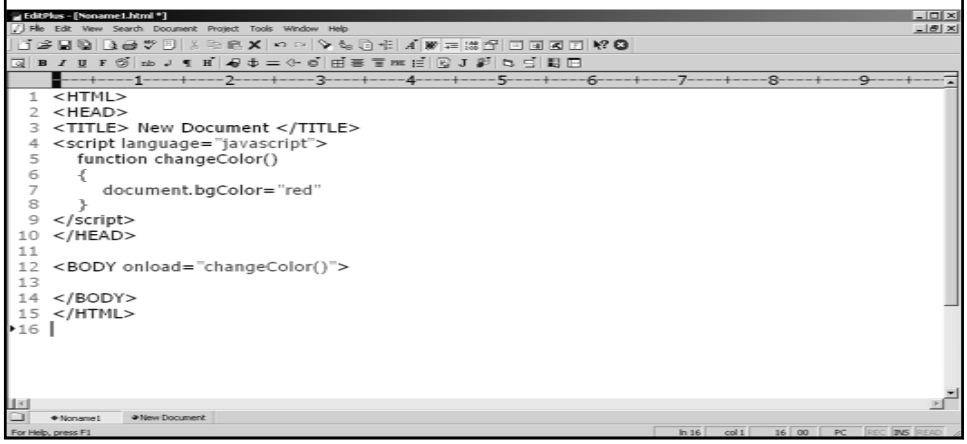
- HTML
  - Merely a mark-up language.
  - Creates Client-Side User Interface.
  - Cannot handle Client-Side activities to respond to a user.

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
</HEAD>

<BODY>
   <b>This is a HTML document</b>
   <form>
      Enter Name<input type="text">
      Enter Password<input type="password">
   </form>
</BODY>
</HTML>
```

Need for Scripting

- Thus the advent of Scripting Languages.
- To Make HTML document dynamic scripting was introduced
- Popular Scripting Languages
  - Perl, REXX, JavaScript, VBScript, Tcl/Tk



The History of JavaScript

JavaScript developed by Netscape in 1995 as a method validating forms and providing interactive content to HTML.

European Computer Manufacturers Association (ECMA) came to help and put forward ECMAScript which is now an ISO Standard.

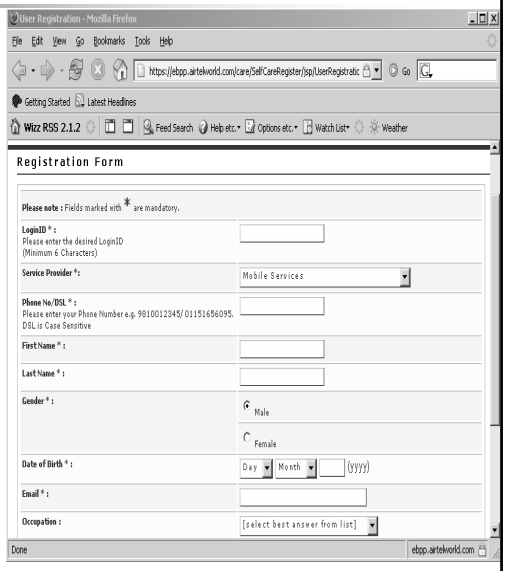
At the time, Microsoft and Netscape introduced their v4 browsers which lead to the unholiness referred to as DHTML.

Characteristics of JavaScript

- JavaScript syntax similar to C or Pascal
- JavaScript is Event-Driven
- JavaScript is platform-independent
- JavaScript enables quick development
- JavaScript is easy to learn and use.

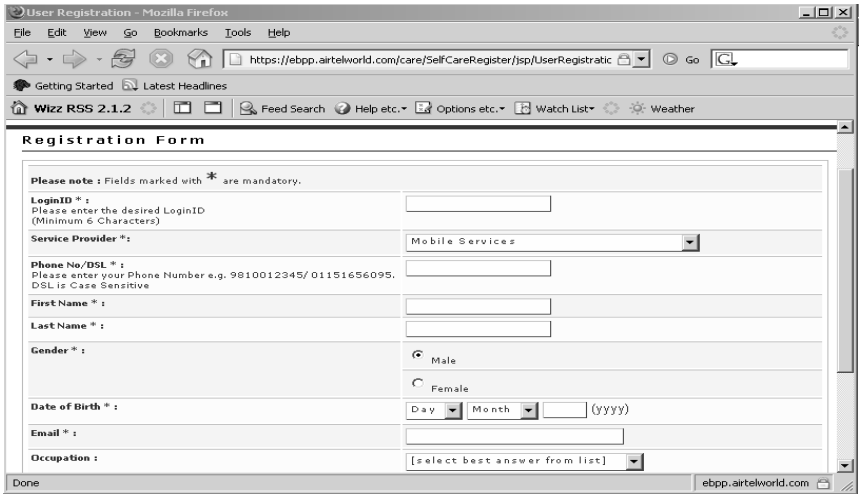
What is JavaScript?

- JavaScript is *the* scripting language of the Web!
- JavaScript is used in millions of Web pages to improve the design, validate forms, detect the visitor's browser, create/use cookies, and much more.
- JavaScript is supported by all major browsers, like Netscape and Internet Explorer



What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages



What is JavaScript?

- JavaScript is a scripting language (a scripting language is a lightweight programming language)
- A JavaScript consists of lines of executable computer code
- A JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  <script language="javascript">
5     function changeColor()
6     {
7
8        document.write("Hello World")
9        document.bgColor="red"
10    }
11 </script>
12 </HEAD>
13
14 <BODY onload="changeColor()">
15
16 </BODY>
17 </HTML>
```

Are Java and JavaScript the Same?

- NO!
- Java and JavaScript are two completely different languages in both concept and design!
- Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What can a JavaScript Do?

JavaScript gives HTML designers a programming tool
HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages

JavaScript can put dynamic text into an HTML page
A JavaScript statement can write a variable text into an HTML page

JavaScript can react to events
A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element

JavaScript can read and write HTML elements
A JavaScript can read and change the content of an HTML element

JavaScript can be used to validate data
A JavaScript can be used to validate form data before it is submitted to a server, this will save the server from extra processing

JavaScript can be used to detect the visitor's browser
A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser

JavaScript How To ...?

The HTML <script> tag is used to insert a JavaScript into an HTML page.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7     <script language="javascript">
8        document.write("Hello World!")
9     </script>
10 </BODY>
11 </HTML>
```

Ending Statements With a Semicolon (;)

With traditional programming languages, like C, C++ and Java, each code statement has to end with a semicolon (;).

But in general, semicolons are **optional**! Semicolons are required to put more than one statement on a single line.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7     <script language="javascript">
8        var name = "Pratian";
9        document.write(name);
10       var secondName = "somename"
11       document.write(secondName)
12    </script>
13 </BODY>
14 </HTML>
```

How to Handle Older Browsers?

Browsers that do not support JavaScript will display the script as page content. To prevent them from doing this, we may use the HTML comment tag.

The two forward slashes at the end of comment line (//) are a JavaScript comment symbol. This prevents the JavaScript compiler from compiling the line.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7     <script language="javascript">
8        <!--
9        var name = "Pratian";
10       document.write(name);
11       var secondName = "somename"
12       document.write(secondName)
13       //-->
14    </script>
15 </BODY>
16 </HTML>
```

JavaScript Where To ...?

JavaScripts in the body section will be executed WHILE the page loads.

JavaScripts in the head section will be executed when CALLED.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  <script language="javascript">
5  function someFunction()
6  {
7     document.write("Hello World!")
8  }
9  </script>
10 </HEAD>
11
12 <BODY onload="someFunction()">
13    <script language="javascript">
14       document.write("Hello World!")
15    </script>
16 </BODY>
17 </HTML>
```

Scripts in the head section

Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  <script language="javascript">
5  function someFunction()
6  {
7     document.write("Hello World!")
8  }
9  </script>
10 </HEAD>
11
12 <BODY onload="someFunction()">
13 </BODY>
14 </HTML>
```

Scripts in the body section

Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7    <script language="javascript">
8      document.write("Hello World!")
9  </script>
10 </BODY>
11 </HTML>
```

Scripts in both the body and head section:

You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  <script language="javascript">
5  function someFunction()
6  {
7     document.write("Hello World!")
8  }
9  </script>
10 </HEAD>
11
12 <BODY onload="someFunction()">
13    <script language="javascript">
14       document.write("Hello World!")
15    </script>
16 </BODY>
17 </HTML>
```

Using an External JavaScript file

To reuse a JavaScript code across different HTML files, it is written in an external file and saved with a .js file extension.

Note: The external script cannot contain the <script> tag!

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script src="abc.js">
8
9  </script>
10 </BODY>
11 </HTML>
```

JavaScript Lexical Structure

- Keywords
  - They have a special meaning in JavaScript.
  - Part of the language syntax itself.
  - Also called as Reserved Words.

break	case	continue	default
delete	do	else	export
for	function	if	import
in	new	return	switch
this	typeof	var	void
while	with		

Operators in JavaScript

Assignment	=
Arithmetic	+ - * / %
Logical	&& || !
Comparison	== != < <= > >=
Compound Assignment	+= -= *= /= %=
String	+ (for concatenation)
Increment & Decrement	++ --
Typeof operator	Used to identify the type of variable

Operator Precedence

Precedence	Operator
1	**Parenthesis or array subscript**
2	**!, -, ++, --**
3	*** / %**
4	**+ -**
5	**< <= > >=**
6	**== !=**
7	**&&**
8	**||**
9	**?:**
10	**= += -= *= /= %=**

JavaScript Guidelines

- Some important things to know when scripting with JavaScript.

JavaScript is Case Sensitive

JavaScript is case sensitive, myFunction and myfunction are not same, similarly variables myName and myname are not same.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <script language="javascript">
6     function myFunction()
7     {
8        var myName = "somename"
9        var myname = "noname"
10       document.write(myName)
11    }
12    function myfunction()
13    {
14       var myName = "somename"
15       var myname = "noname"
16       document.write(myName)
17    }
18 </script>
19 <BODY>
20 </BODY>
21 </HTML>
```

Symbols

Opening symbols, like (, {, [, “, ', must always have a matching closing symbol, like ‘, “,], },).

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <script language="javascript">
6     function myFunction()
7     {
8        var myName = "somename"
9        document.write(myName)
10    }
11 </script>
12 <BODY>
13 </BODY>
14 </HTML>
```

White Space

JavaScript ignores extra spaces. You can add white space to your script to make it more readable.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <script language="javascript">
6     function myFunction()
7     {
8        var myName="somename"
9        var myName  =  "somename"
10    }
11 </script>
12 <BODY>
13 </BODY>
14 </HTML>
```

Break up a Code Line

You can break up a code line within a **text string** with a backslash.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8  document.write("Hello \
9      world!")
10 document.write \
11 ("Hello World!")
12 </script>
13 </BODY>
14 </HTML>
```

Works

Error

Comments

Start a comment with two slashes "//":

Using /* and */ to create a multi-line comment:

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     var name = "some name" // this is a single line comment
9     document.write(name)
10    /* this is
11    a multi line
12    comment
13    */
14 </script>
15 </BODY>
16 </HTML>
```

JavaScript Variables

- A variable is a "container" for information you want to store. A variable's value can change during the script. You can refer to a variable by name to see its value or to change its value.
- Rules for variable names:
  - Variable names are case sensitive
  - They must begin with a letter or the underscore character
  - Can not be a key word
- **IMPORTANT!** JavaScript is case-sensitive! A variable named strname is not the same as a variable named STRNAME!

Declare and assign a value to a Variable

Variables are created using a var statement.

It can also be created without using the var statement.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     var name = "some name" // declaring variable using var
9     name1 = "some name" // declaring variable without using var
10    document.write(name)
11 </script>
12 </BODY>
13 </HTML>
```

Lifetime of Variables

- When you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.
- If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

Variables - Example

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     var name = "some name" // declaring variable using var
9     name1 = "some name" // declaring variable without using var
10    var temp
11    temp = "abc"
12    document.write(name)
13    document.write(temp)
14 </script>
15 </BODY>
16 </HTML>
```

Conditional Statements

- Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.
- In JavaScript we have the following conditional statements:
  - if statement
  - if...else statement
  - if...else if....else statement
  - switch statement

Conditional Statements

- if statement - use this statement if you want to execute some code only if a specified condition is true
- if...else statement - use this statement if you want to execute some code if the condition is true and another code if the condition is false
- if...else if....else statement - use this statement if you want to select one of many blocks of code to be executed
- switch statement - use this statement if you want to select one of many blocks of code to be executed

If Statement

- You should use the if statement if you want to execute some code only if a specified condition is true.
- **Syntax**

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

- Note that <u>if</u> is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

If Statement – Example 1

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     //display good morning if time is less than 10
9     var d = new Date()
10    var time = d.getHours()
11    if(time<10)
12    {
13       document.write("Good Morning!")
14    }
15 </script>
16 </BODY>
17 </HTML>
```

If Statement – Example 2

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     //display lunch time if time is 12
9     var d = new Date()
10    var time = d.getHours()
11    if(time==12)
12    {
13       document.write("Lunch Time!")
14    }
15 </script>
16 </BODY>
17 </HTML>
```

If...else Statement

- If you want to execute some code if a condition is true and another code if the condition is not true, use the if....else statement.

```
if (condition)
 {
      code to be executed if condition is true
 }
else
 {
      code to be executed if condition is not true
 }
```

If...else Statement : Example

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     var d = new Date()
9     var time = d.getHours()
10    if(time<10)
11    {
12       document.write("Lunch Time!")
13    }
14    else
15    {
16       document.write("Good Day!")
17    }
18 </script>
19 </BODY>
20 </HTML>
```

If...else if...else Statement

- You should use the if....else if...else statement if you want to select one of many sets of lines to execute.

```
if (condition1)
{
   code to be executed if condition1 is true
}
else if (condition2)
{
  code to be executed if condition2 is true
}
else
{
  code to be executed if condition1 and condition2 are not true
}
```

If...else if...else Statement : Example

```
<script language="javascript">
   var d = new Date()
   var time = d.getHours()
   if(time<10)
   {
      document.write("Good Morning!")
   }
   else if (time>10 && time<14)
   {
      document.write("Good Afternoon!")
   }
   else
   {
      document.write("Good Day!")
   }
</script>
```

The JavaScript Switch Statement

- You should use the switch statement if you want to select one of many blocks of code to be executed.

```
switch(n)
{
case 1:     execute code block 1
            break
case 2:     execute code block 2
            break
default:    code to be executed if n is different from
            case 1 and 2
}
```

switch(n)…case : How it works?

This is how it works:

- First we have a single expression *n* (most often a variable), that is evaluated once.
- 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 associated with that case is executed.
- Use **break** to prevent the code from running into the next case automatically.

switch(n)…case : Example

```
7  <script language="javascript">
8     var d = new Date()
9     var day = d.getDay()
10    switch(day)
11    {
12       case 0   :  document.write("Sunday")
13                   break;
14       case 1   :  document.write("Monday")
15                   break;
16       case 2   :  document.write("Tuesday")
17                   break;
18       case 3   :  document.write("Wednesday")
19                   break;
20       case 4   :  document.write("Thursday")
21                   break;
22       case 5   :  document.write("Friday")
23                   break;
24       case 6   :  document.write("Saturday")
25                   break;
26
27    }
28 </script>
```

JavaScript Loops

- Loops in JavaScript are used to execute the same block of code a specified number of times or while a specified condition is true.

JavaScript Loops

- If we want the same block of code to run over and over again in a row, instead of adding several almost equal lines in a script we can use loops to perform a task like this.
- Different kinds of loops in JavaScript:
  - For
  - While
  - Do…while
  - For..in

The for Loop

- The for loop is used when you know in advance how many times the script should run.
- Syntax

```
for(var=startvalue;var<=endvalue;var=var+increment)
{
            code to be executed
}
```

The for Loop: Example

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     var i = 0;
9     for(i=0 ; i<=10 ; ++i)
10    {
11       document.write("The no is "+ i +"<br>")
12    }
13 </script>
14 </BODY>
15 </HTML>
```

The while loop

- The while loop is used when you want the loop to execute and continue executing while the specified condition is true.
- **Note:** The <= could be any comparing statement.

```
while (var<=endvalue)
{
      code to be executed
}
```

The while loop: Example

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE> New Document </TITLE>
 4 </HEAD>
 5
 6 <BODY>
 7 <script language="javascript">
 8    var i =0;
 9    while(i<=10)
10    {
11       document.write("The no is "+ i +"<br>")
12       i = i+1
13    }
14 </script>
15 </BODY>
16 </HTML>
```

The do...while Loop

- This loop will always be executed once, even if the condition is false, because the code are executed before the condition is tested.

```
do
{
      code to be executed
} while (var<=endvalue)
```

The do...while Loop: Example

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     var i =0;
9     do
10    {
11       document.write("The no is "+ i +"<br>")
12       i = i+1
13    }while(i<=10)
14 </script>
15 </BODY>
16 </HTML>
```

The for…in Loop

- The for…in loop is used to iterate through the elements of an array or through the properties of an object.
- The code in the body of the loop is executed once for each element.

```
for(variable in object)
{
      code to be executed
}
```

The for…in Loop:Example

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE> New Document </TITLE>
 4 </HEAD>
 5
 6 <BODY>
 7 <script language="javascript">
 8    var i =0;
 9    var myCars = new Array("Zen","Alto","Getz")
10    for(i in myCars)
11    {
12       document.write(myCars[ i] +"<br>")
13    }
14 </script>
15 </BODY>
16 </HTML>
```

JavaScript break and continue Statements

- There are two special statements that can be used inside loops: break and continue.
- **Break**
  - The break command will break the loop and continue executing the code that follows after the loop (if any).
- **Continue**
  - The continue command will break the current loop and continue with the next value.

break Statement : Example

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     var i =0;
9     for(i=0 ; i<=10 ;++i)
10    {
11       if(i==4){break}
12       document.write("The no is "+i+"<br>")
13    }
14 </script>
15 </BODY>
16 </HTML>
```

continue Statement : Example

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     var i =0;
9     for(i=0 ; i<=10 ;++i)
10    {
11       if(i==4){continue}
12       document.write("The no is "+i+"<br>")
13    }
14 </script>
15 </BODY>
16 </HTML>
```

JavaScript Popup Boxes

- In JavaScript we can create three kind of popup boxes:
  - Alert box,
  - Confirm box,
  - Prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     alert("Hello World!")
9  </script>
10 </BODY>
11 </HTML>
```

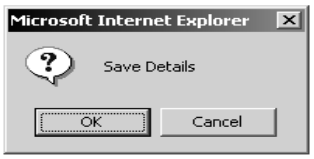
Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     confirm("Save Details")
9  </script>
10 </BODY>
11 </HTML>
```



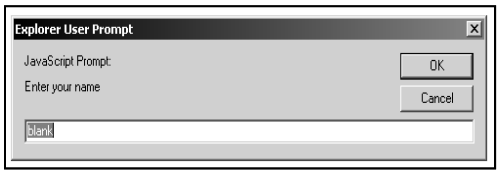
Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8     var name = prompt("Enter your name","blank")
9  </script>
10 </BODY>
11 </HTML>
```



JavaScript Functions

- A function is a reusable code-block that will be executed by an event, or when the function is called.

```
function functionName(arguments)
{
        statements to execute
        ......
        return (value)
}
```

JavaScript Functions: Why?

- To keep the browser from executing a script as soon as the page is loaded, you can write your script as a function.
- A function contains some code that will be executed only by an event or by a call to that function.
- You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).
- Functions are defined at the beginning of a page, in the <head> section.

How to Define a Function?

- The syntax for creating a function is:

```
function functionname(var1,var2,...,varX)
{
    some code
}
```

- var1, var2, etc are variables or values passed into the function. The { and the } defines the start and end of the function.

  Note: A function with no parameters must include the parentheses () after the function name:

```
function functionname()
{
 some code
}
```

How to Define a Function?

- **Note:** Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs!
- Also note that you must call a function with the exact same capitals as in the function name.

JavaScript Functions: Example

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  <script language="javascript">
5     function displayMessage()
6     {
7        alert("Hello World")
8     }
9  </script>
10 </HEAD>
11
12 <BODY>
13 <input type="button" value="Click" onclick="displayMessage()">
14 </BODY>
15 </HTML>
```

The return Statement

The return statement is used to specify the value that is returned from the function.

So, functions that is going to return a value must use the return statement.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  <script language="javascript">
5     function product(a,b)
6     {
7        return a*b
8     }
9     function display()
10    {
11       var x = product(2,7)
12       document.write(x)
13    }
14 </script>
15 </HEAD>
16
17 <BODY>
18 <input type="button" value="Click" onclick="display()">
19 </BODY>
20 </HTML>
```

JavaScript Events

- Events are actions that can be detected by JavaScript.

Events

- By using JavaScript, we have the ability to create dynamic web pages.
- Examples of events:
  - A mouse click
  - A web page or an image loading
  - Mousing over a hot spot on the web page
  - Selecting an input box in an HTML form
  - Submitting an HTML form
  - A keystroke
- **Note:** Events are normally used in combination with functions, and the function will not be executed before the event occurs!

Common Events

Some common form Events

onClick -- the form element is clicked

onDblClick -- the form element is clicked twice in close succession

onMouseDown -- the mouse button is pressed while over the form element

onMouseOver -- the mouse is moved over the form element

onMouseOut -- the mouse is moved away from the form element

onMouseUp -- the mouse button is released while over the form element

onMouseMove -- the mouse is moved

Event Example 1

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  <script language="javascript">
5     function display()
6     {
7        document.write("Using onClick Event")
8     }
9  </script>
10 </HEAD>
11
12 <BODY>
13 <input type="button" value="Click" onclick="display()">
14 </BODY>
15 </HTML>
```

Event Example 2

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  <script language="javascript">
5     function display()
6     {
7        document.write("Using onClick Event")
8     }
9  </script>
10 </HEAD>
11
12 <BODY>
13 <input type="button" value="Click" onmouseover="display()">
14 </BODY>
15 </HTML>
```

Event Example 3

```
1 <HTML>
2 <HEAD>
3 <TITLE> New Document </TITLE>
4 </HEAD>
5 <BODY>
6 <h1 onmouseover="style.color='red'"
7    onmouseout="style.color='blue'">Hello World!</h1>
8 </BODY>
9 </HTML>
```

onload and onUnload Event

The onload and onUnload events are triggered when the user enters or leaves the page.

The onload event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

```
1 <HTML>
2 <HEAD>
3 <TITLE> New Document </TITLE>
4 </HEAD>
5
6 <BODY onload="alert('Using onload event')">
7 </BODY>
8 </HTML>
```

onFocus, onBlur and onChange

The onFocus, onBlur and onChange events are often used in combination with validation of form fields.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <input type="text" onfocus="alert('In Focus')"
8   onchange="alert('Changed')" onblur="alert('Blur')">
9  </BODY>
10 </HTML>
```

onSubmit

The onSubmit event is used to validate ALL form fields before submitting it.

In the example the function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <script language="javascript">
6     function checkForm()
7     {
8        //validation done
9        return true;
10    }
11 </script>
12 <BODY>
13 <form onsubmit="return checkForm()">
14 Enter Name<input type="text"><br>
15 Enter Password<input type="password"><br>
16 <input type="submit">
17 </form>
18 </BODY>
19 </HTML>
```

Events and event handlers I

Event	Applies to	Occurs when	Handler
Load	Document body	User loads the page in a browser	onLoad
Unload	Document body	User exits the page	onUnload
Error	Images, window	Error on loading an image or a window	onError
Abort	Images	User aborts the loading of an image	onAbort

Events and event handlers II

Event	Applies to	Occurs when	Handler
KeyDown	Documents, images, links, text areas	User depresses a key	onKeyDown
KeyUp	Documents, images, links, text areas	User releases a key	onKeyUp
KeyPress	Documents, images, links, text areas	User presses or holds down a key	onKeyPress
Change	Text fields, text areas, select lists	User changes the value of an element	onChange

Events and event handlers III

Event	Applies to	Occurs when	Handler
MouseDown	Documents, buttons, links	User depresses a mouse button	onMouseDown
MouseUp	Documents, buttons, links	User releases a mouse button	onMouseUp
Click	Buttons, radio buttons, checkboxes, submit buttons, reset buttons, links	User clicks a form element or link	onClick

Events and event handlers IV

Event	Applies to	Occurs when	Handler
MouseOver	Links	User moves cursor over a link	onMouseOver
MouseOut	Areas, links	User moves cursor out of an image map or link	onMouseOut
Select	Text fields, text areas	User selects form element's input field	onSelect

Events and event handlers V

Event	Applies to	Occurs when	Handler
Move	Windows	User or script moves a window	onMove
Resize	Windows	User or script resizes a window	onResize
DragDrop	Windows	User drops an object onto the browser window	onDragDrop

Events and event handlers VI

Event	Applies to	Occurs when	Handler
Focus	Windows and all form elements	User gives element input focus	onFocus
Blur	Windows and all form elements	User moves focus to some other element	onBlur
Reset	Forms	User clicks a Reset button	onReset
Submit	Forms	User clicks a Submit button	onSubmit

Try…Catch statement

- The try…catch statement allows to test a block code for errors. The try block contains the code to be run, and the catch block contains the code the code to be executed if an error occurs.

```
try
{
 //run some code
}
catch(err)
{
 // handle errors if any
 }
```

Try…Catch Example

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <script language="javascript">
6  var txt
7  function display()
8  {
9          Alert("Hello World!")
10 }
11 </script>
12 <BODY>
13 <input type="button" value="Display"
14     onclick="display()">
15 </BODY>
16 </HTML>
```

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <script language="javascript">
6  var txt
7  function display()
8  {
9     try
10    {
11       Alert("Hello World!")
12    }catch(err)
13    {
14       txt="There was an error on this page.\n\n"
15       txt+="Error description:"+err.description+"\n\n"
16       alert(txt)
17    }
18 }
19 </script>
20 <BODY>
21 <input type="button" value="Display"
22     onclick="display()">
23 </BODY>
24 </HTML>
```

Throw statement

- Throw statement allows to create an exception.Together with the try…catch statement, it can be used to control program flow and generate accurate error message.

```
throw(exception)
```

Throw Example

```
<BODY>
<script language="javascript">
var x = prompt("Enter a number between 1 and 10")
try
{
   if(x>10)
      throw "Err1"
   else if(x<=0)
      throw "Err2"
   else if(isNaN(x))
      throw "Err3"
}
catch (er)
{
   if(er=="Err1")
      alert("Error! The value is too high")
   if(er=="Err2")
      alert("Error! The value is too low")
   if(er=="Err3")
      alert("Error! The value is not a number")
}
</script>
</BODY>
```

Onerror Event

- The onerror event is fired whenever there is a script error in the page.
- To use the onerror event, a function to handle the errors is created.

```
onerror = handleErr
Function handleErr(msg,url,l)
{
    //handle error
    return true or false
}
```

Onerror Example

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <script language="javascript">
6  onerror=someFunction
7  function someFunction(msg,url,l)
8  {
9     alert(" Error Msg :"+msg+"\n URL :"+url+"\n Line :"+l)
10    return true
11 }
12 </script>
13 <BODY>
14 <script language="javascript">
15    Alert("Hello World")
16 </script>
17 </BODY>
18 </HTML>
```

JavaScript Objects

- JavaScript is an Object Oriented Programming(OOP) language. An OOP language allows you to define your own objects and make your own variables type.

JavaScript Objects

- Array
- Boolean
- Date
- Math
- String
- HTML DOM

Array Object

- The Array object is used to store a set of values in a single variable name. Each value is an element of the array and has an associated index number.
- You create an instance of the Array object with the "new" keyword.

Array Object - Example

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
</HEAD>
<BODY>
<script language="javascript">
   var carName1 = new Array(3)
   carName1[0]="Zen"
   carName1[1]="Alto"
   carName1[2]="Getz"
   var carName2 = new Array("Zen","Alto","Getz")
   for(i=0 ; i<3 ; ++i)
   {
      document.write(carName1[i])
      document.write(carName2[i])
   }
</script>
</BODY>
</HTML>
```

Array Object Methods

Method	Description
concat()	Joins two or more arrays and returns the result
join()	Puts all the elements of an array into a string. The elements are separated by a specified delimiter
pop()	Removes and returns the last element of an array
push()	Adds one or more elements to the end of an array and returns the new length
reverse()	Reverses the order of the elements in an array
sort()	Sorts the elements of an array

The concat() method

- The concat() method is used to join two or more arrays.This method does not change the existing arrays, it only returns a copy of the joined arrays.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7     var carName1 = new Array(3)
8     carName1[0]="Maruti"
9     carName1[1]="Maruti"
10    carName1[2]="Hyundai"
11    var carName2 = new Array("Zen","Alto","Getz")
12    document.write("<h1>"+carName1.concat(carName2)+"</h1>")
13 </script>
14 </BODY>
15 </HTML>
```

The join() method

- In this example we will create an array, and then put all the elements in a string.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7     var carName = new Array(3)
8     carName[0]="Maruti"
9     carName[1]="Maruti"
10    carName[2]="Hyundai"
11    document.write(carName.join()+"<br>")
12    document.write(carName.join(".."))
13 </script>
14 </BODY>
15 </HTML>
```

The pop() method

- In this example we will create an array, and then remove the last element of the array. Note that this will also change the length of the array.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7     var arr = new Array("One","Two","Three","Four","Five")
8     document.write(arr+"<br>")
9     document.write(arr.pop())
10    document.write(arr+"<br>")
11 </script>
12 </BODY>
13 </HTML>
```

The push() method

- In this example we will create an array, and then change the length of it by adding a element.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7  var arr = new Array("One","Two","Three","Four")
8  document.write(arr)
9  document.write(arr.push("Five")+"<br>")
10 document.write(arr)
11 </script>
12 </BODY>
13 </HTML>
```

The reverse() method

- In this example we will create an array, and then reverse the order of it.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7  var arr = new Array("One","Two","Three","Four","Five")
8  document.write(arr+"<br>")
9  arr.reverse()
10 document.write(arr)
11 </script>
12 </BODY>
13 </HTML>
```

The sort() method

- In this example we will create an array and sort it alphabetically.

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
</HEAD>
<BODY>
<script language="javascript">
var arr = new Array("Ram","Anil","Venkat","Mohan","Karan")
document.write(arr+"<br>")
document.write(arr.sort())
</script>
</BODY>
</HTML>
```

Boolean Object

- The Boolean object is an object wrapper for a Boolean value and it is used to convert a non-Boolean value to a Boolean value, either true or false.
- If the Boolean object has no initial value or if it is 0, null, "", false, or NaN, the initial value is false. Otherwise it is true (even with the string "false").

Boolean Object – Examples 1

- All the following lines of code create Boolean objects with an initial value of false.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7  var b1 = new Boolean()
8  var b2 = new Boolean(0)
9  var b3 = new Boolean(null)
10 var b4 = new Boolean(NaN)
11 var b5 = new Boolean(false)
12 </script>
13 </BODY>
14 </HTML>
```

Boolean Object – Examples 2

- All the following lines of code create Boolean objects with an initial value of true.

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7  var b1 = new Boolean(true)
8  var b2 = new Boolean("true")
9  var b3 = new Boolean("false")
10 var b4 = new Boolean("abcd")
11 document.write(b4)
12 </script>
13 </BODY>
14 </HTML>
```

JavaScript Date Object

- The Date object is used to work with dates and times.
- To create an instance of the Date object and assign it to a variable called "d", you do the following:

```
var d=new Date()
```

- After creating an instance of the Date object, you can access all the methods of the Date object from the "d" variable.
- To return the current day in a month (from 1-31) of a Date object, write the following:

```
d.getDate()
```

Date Object Methods

Method	Description
Date()	Returns today's date and time
getDate()	Returns the day of the month from a Date object (from 1-31)
getDay()	Returns the day of the week from a Date object (from 0-6)
getMonth()	Returns the month from a Date object (from 0-11)
getFullYear()	Returns the year, as a four-digit number, from a Date object
getYear()	Returns the year, as a two-digit or a four-digit number, from a Date object. Use getFullYear() instead !!

Date Object Methods

Method	Description
getHours()	Returns the hour of a Date object (from 0-23)
getMinutes()	Returns the minutes of a Date object (from 0-59)
getSeconds()	Returns the seconds of a Date object (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1, 1970
setDate()	Sets the day of the month in a Date object (from 1-31)
setTime()	Calculates a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970

Date Object Example1

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7  var d = new Date()
8  document.write(d.getDate())
9  document.write(d.getDay())
10 document.write(d.getMonth())
11 </script>
12 </BODY>
13 </HTML>
```

Date Object Example 2

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7  var d = new Date()
8  document.write(d.getHours())
9  document.write(d.getMinutes())
10 document.write(d.getSeconds())
11 </script>
12 </BODY>
13 </HTML>
```

JavaScript Math Object

- The built-in Math object includes mathematical constants and functions. You do not need to create the Math object before using it.

JavaScript Math Object : Methods

Method	Description
abs(x)	Returns the absolute value of a number
ceil(x)	Returns the value of a number rounded upwards to the nearest integer
pow(x,y)	Returns the value of x to the power of y
floor(x)	Returns the value of a number rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of a number
max(x,y)	Returns number with the highest value of x and y
min(x,y)	Returns the number with the lowest value of x and y
random()	Returns a random number between 0 and 1
round(x)	Rounds a number to the nearest integer
sqrt(x)	Returns the square root of a number

Math Object Example

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7  document.write(Math.abs(-5.65)+"<br>")
8  document.write(Math.log(2)+"<br>")
9  document.write(Math.max(4.56,8.23)+"<br>")
10 document.write(Math.pow(2,3)+"<br>")
11 </script>
12 </BODY>
13 </HTML>
```

JavaScript String Object

- The string object is used to manipulate a stored piece of text.

JavaScript String Object

Method	Description
big()	Displays a string in a big font
blink()	Displays a blinking string
bold()	Displays a string in bold
charAt(n)	Returns the character at a specified position
indexOf(chr)	Returns the position of the first occurrence of a specified string value in a string
match(str)	Searches for a specified string value in a string
replace()	Replaces some characters with some other characters in a string
substr()	Extracts a specified number of characters in a string, from a start index
toLowerCase()	Displays a string in lowercase letters
toUpperCase()	Displays a string in uppercase letters

String Object Example

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7  var str = "Hello World!"
8  document.write(str.big())
9  document.write(str.bold())
10 document.write(str.fontcolor("red"))
11 document.write(str.fontsize(7))
12 </script>
13 </BODY>
14 </HTML>
```

Create Objects

- In addition to built in objects, we can create our own.
- An object is a special kind of data, with a collection of properties and methods.
- Two ways of creating objects
  - Direct Instance of an Object
  - Template of an Object

Direct Instance of Object

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7  Car = new Object()
8  Car.color = "Red"
9  Car.mpl = 15
10 Car.doors=4
11 document.write("The car color is "+Car.color +
12 " with "+Car.doors+" doors "+"and mileage of "+Car.mpl)
13 </script>
14 </BODY>
15 </HTML>
```

Template for an Object

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7  function Car(_color,_mpl,_doors)
8  {
9  this.color = _color
10 this.mpl = _mpl
11 this.doors = _doors
12 }
13 myCar = new Car("Red",15,4)
14 document.write("The car color is "+myCar.color +
15 " with "+myCar.doors+" doors "+"and mileage of "+myCar.mpl)
16 </script>
17 </BODY>
18 </HTML>
```

Methods in Object Template

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <BODY>
6  <script language="javascript">
7  function Car(_color,_mpl,_doors)
8  {
9  this.color = _color
10 this.mpl = _mpl
11 this.doors = _doors
12 this.showColor = function() { alert(this.color) }
13 }
14 myCar = new Car("Red",15,4)
15 document.write("The car color is "+myCar.color +
16 " with "+myCar.doors+" doors "+"and mileage of "+myCar.mpl)
17 myCar.showColor()
18 </script>
19 </BODY>
20 </HTML>
```

Navigator Object

- The Navigator Object contains information about the visitor's browser name, browser version and more.
- Navigator Properties
  - appName
  - appVersion
  - appCodeName
  - platform
  - cookieEnabled

Navigator Object Example 1

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8  document.write("Browser name "+navigator.appName+"<br>")
9  document.write("Browser version "+navigator.appVersion)
10 </script>
11 </BODY>
12 </HTML>
```

Navigator Object Example 2

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5
6  <BODY>
7  <script language="javascript">
8  document.write("Browser name: "+navigator.appName+"<br>")
9  document.write("Browser version: "+navigator.appVersion+"<br>")
10 document.write("CodeName:  "+navigator.appCodeName +"<br>")
11 document.write("Platform: "+navigator.platform +"<br>")
12 document.write("Cookie Enabled: "+navigator.cookieEnabled +"<br>")
13 </script>
14 </BODY>
15 </HTML>
```

Timing Events

- In JavaScript, it is possible to execute some code after a specified time interval.
- Timing Event methods
  - setTimeout()
  - clearTimeout()

Timing Event Example 1

```
1  <HTML>
2  <HEAD>
3  <TITLE> New Document </TITLE>
4  </HEAD>
5  <script language="javascript">
6  function timedMsg()
7  {
8     setTimeout("alert('5 seconds!')",5000)
9  }
10 </script>
11 <BODY>
12 <input type="button" value="Click Me!" onclick="timedMsg()">
13 </BODY>
14 </HTML>
```

Timing Event Example 2

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
</HEAD>
<script language="javascript">
var c =0
var t
function start1()
{
   document.getElementById('txt').value=c
   c = c+1
   t = setTimeout("start1()",1000)
}
function stop1()
{
   clearTimeout(t)
}
</script>
<BODY>
<input type="button" value="Start Count" onclick="start1()">
<input type="text" id="txt">
<input type="button" value="Stop Count" onclick="stop1()">
</BODY>
</HTML>
```

Thanks

Any Questions?