JavaScript

Onkar Deshpande

Table of Contents

Module 1. Introduction to JavaScript

- Overview
  - Introduction
  - What can JavaScript do?
  - Where to place it
  - The first script
  - Capital letters

Introduction

- JavaScript is a scripting language that will allow you to add real programming to your WebPages.
- JavaScript was designed to add interactivity to HTML pages
- You can create small application type processes with JavaScript, like a calculator or a primitive game of some sort.
- JavaScript is used in millions of Web pages to add functionality, validate forms, detect browsers, and much more.
- JavaScript is usually embedded directly into HTML pages

What can a JavaScript do?

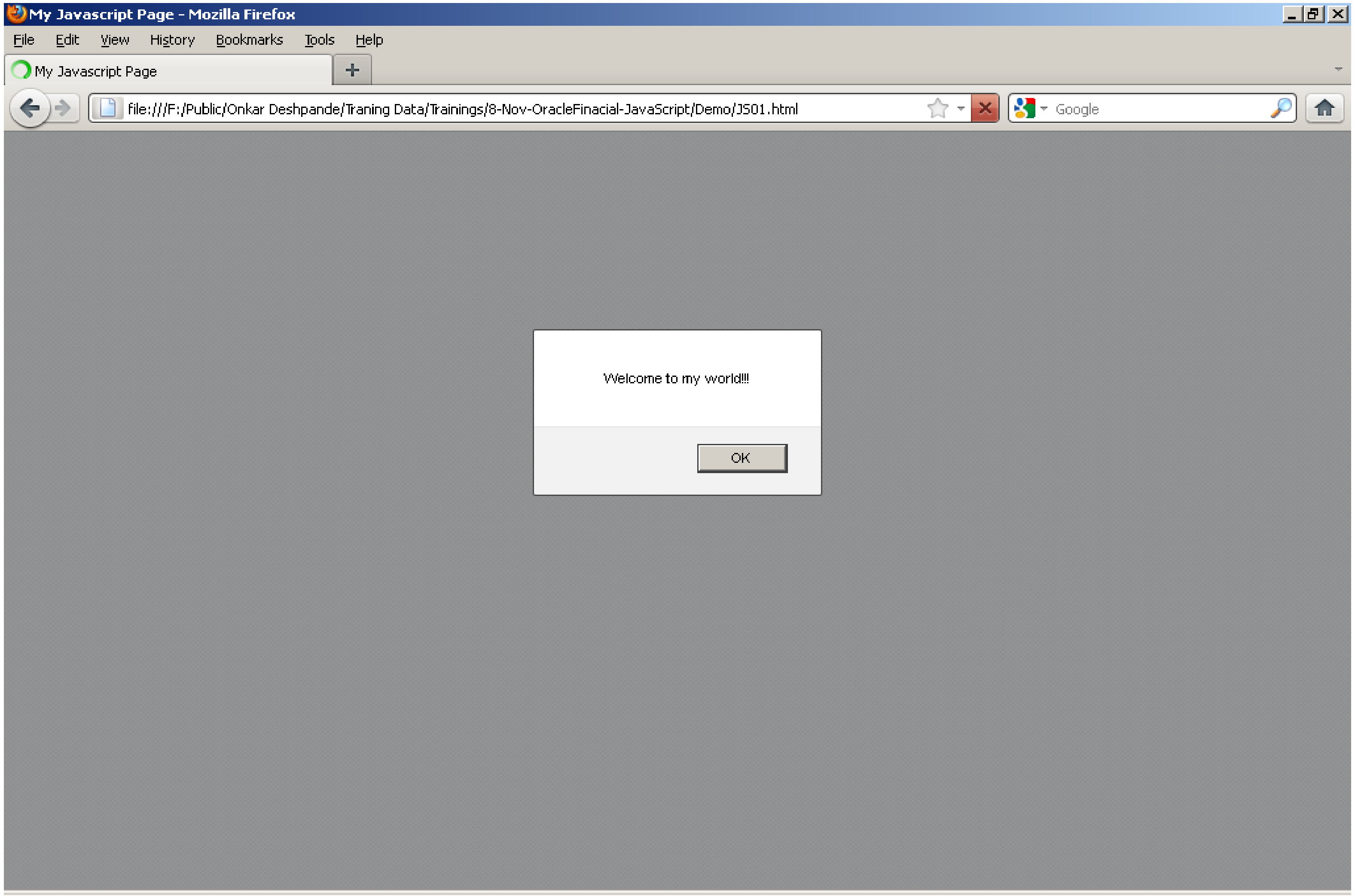
- JavaScript gives HTML designers a programming tool
- JavaScript can put dynamic text into an HTML page
- JavaScript can react to events
- JavaScript can read and write HTML elements
- JavaScript can be used to validate data
- JavaScript can be used to detect the visitor's browser
- JavaScript can be used to create cookies

Where to place it

- Since JavaScript isn't HTML, you will need to let the browser know in advance when you enter JavaScript to an HTML page. This is done using the <script> tag.

```
<html>
        <head>
                <title>My JavaScript Page</title>
        </head>
        <body>
                <script type="text/JavaScript">
                        alert("Welcome to my world!!!");
                </script>
        </body>
</html>
```

Output


My Javascript Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
My Javascript Page
file:///F:/Public/Onkar Deshpande/Traning Data/Trainings/8-Nov-OracleFinacial-JavaScript/Demo/JS01.html
Google
OK

Welcome to my world!!!

Handling Browsers with no JavaScript Support

- Browsers that do not support JavaScript, will display JavaScript as page content.
- To prevent them from doing this, the HTML comment tag should be used to "hide" the JavaScript.

```
<html>
    <body>
     <script type="text/JavaScript">
            <!--
                  document.write("Hello World!");
            //-->
     </script>
    </body>
</html>
```

JavaScript Statements

- JavaScript is a sequence of statements to be executed by the browser.
- JavaScript is Case Sensitive
- A JavaScript statement is a command to a browser.

```
document.write("Hello World");
```

Example

```
<script type="text/JavaScript">
    document.write("<h1>This is a heading</h1>");
    document.write("<p>This is a paragraph.</p>");
    document.write("<p>This is another paragraph.</p>");
</script>
```

Writing Comments

- Comments can be added to explain the JavaScript, or to make the code more readable.
- Single line comments start with //.
- Multi line comments start with /* and end with */.

```
<script type="text/JavaScript">
        /*
        The code below will write
        one heading
        */
        document.write("<h1>This is a heading</h1>");
        ……..
</script>
```

JavaScript Variables

- Variables are "containers" for storing information

- **Rules for JavaScript variable names:**
  - Variable names are **case sensitive** (y and Y are two different variables)
  - Variable names must begin with a letter or the underscore character

Capital letters

Example 1	**Example 2**
`<html>`	`<html>`
`<head>`	`<head>`
`<title>My Page</title>`	`<title>My Page</title>`
`</head>`	`</head>`
`<body>`	`<body>`
`<script>`	`<script>`
`myvalue=2;`	`myvalue=2;`
`myvalue=5; //Override`	`MyValue=5;`
`result=myvalue+myvalue;`	`result=myvalue+MyValue;`
`document.write(result);`	`document.write(result);`
`</script>`	`</script>`
`</body>`	`</body>`
`</html>`	`</html>`
Output :-	Output :-
Example 1 would be 10 (5+5).	Example 2 would be 7 (2+5).

Declaring/Creating JavaScript Variables

- In JavaScript, variables are declared with **var statement.**

```
    var num;   // declares empty variable


  var num=5;
  var carname="Volvo";
```

JavaScript Operators

- **Arithmetic Operators**

  + - * / % ++ --

- **Assignment Operators**

  = += -= *= /= %=

Comparison and Logical Operators

- Comparison and Logical operators are used to test for true or false.
- **Comparision operators**

  == != < > <= >=

- **Logical operators**

  &&(logical and) ||(logical or) !(not)

- **Conditional operator(?:)**

  greeting=(visitor=="PRES")?"Dear President ":"Dear ";

Module 2. Control Statements

- Overview
  - Conditional Statements
    - if , if … else, switch
  - Iterative statements
    - while, do… while, for, for … in

Conditional Statements

- JavaScript allows use of following Conditional constructs
  - if
  - if … else
  - switch

if condition

- **if Statement**
  - Syntax:

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

- **Example:**

  Write JavaScript code to display “Good Morning” greeting if time is less than 10.

Example

```
<script type="text/JavaScript">
       //Write a "Good morning" greeting if
      //the time is less than 10

      var d=new Date();
      var time=d.getHours();

      if (time<10)
     {
       document.write("<b>Good morning</b>");
     }
 </script>
```

if … else

- **if … else Statement**
  - Syntax:

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

switch

- **switch statement**
  - Syntax:

```
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
}
```

Example

- **Script to display day's name (Note that Sunday=0, Monday=1, Tuesday=2, etc.)**

```
<script type="text/JavaScript">
   var d=new Date();
   theDay=d.getDay();
   switch (theDay) {
   case 5:
    document.write("Finally Friday");
    break;
   case 6:
    document.write("Super Saturday");
    break;
   case 0:
    document.write("Sleepy Sunday");
    break;
   default:
    document.write("I'm looking forward to this weekend!");
   }
</script>
```

Iterative constructs

- JavaScript allows use of following Iterative constructs
  - for
  - while

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
}
```

Example

```
<script type="text/JavaScript">
 var i=0;
 for (i=0;i<=5;i++)
 {
   document.write("The number is " + i);
   document.write("<br />");
 }
</script>
```

while Loop

- The while loop loops through a block of code while a specified condition is true.
- **Syntax:**

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

Example

```
<script type="text/JavaScript">
       var i=0;
       while (i<=5)
 {
         document.write("The number is " + i);
     document.write("<br />");
     i++;
 }
</script>
```

do … while

- do … while loop will execute the block of code atleast ONCE.
- **Syntax:**

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

for … in Loop Statements

- The for...in statement loops through the elements of an array or through the properties of an object.
- **Syntax:**

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

Example

```
<script type="text/JavaScript">
    var x;
    var names = new Array();
     names [0] = “Sameer";
     names [1] = “Swati";
     names [2] = “Dolly";

    for (x in names )
  {
       document.write(names [x] + "<br />");
  }
</script>
```

Module 3. Popup Box

- Overview
  - Alert Box
  - Prompt Box
  - Confirm Box

Alert Box

- JavaScript has three kind of popup boxes:

  Alert box, Confirm box, and 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.
  - **Syntax:**

```
alert("sometext");
```

Example of Alert Box

```
<html>
<head>
        <script type="text/JavaScript">
              function show_alert()
              {
              alert("I am an alert box!");
              }
        </script>
</head>
<body>
       <input type="button" onclick="show_alert()"
value="Show
      alert box" />
    </body>
</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.
- **Syntax**

```
confirm("sometext");
```

Example of Confirm Box

```
<script type="text/JavaScript">
function show_confirm()
{
 var r=confirm("Press a button");
 if (r==true)
 {
  document.write("You pressed OK!");
 }
else
 {
  document.write("You pressed Cancel!");
 }
}
</script> </head>
<body>
  <input type="button" onclick="show_confirm()"
value="Show
         confirm box" />
       </body>
```

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.
- **Syntax**

```
prompt("sometext","defaultvalue");
```

Example of Prompt Box

```
<script type="text/JavaScript">
    function show_prompt()
    {
    var name=prompt("Please enter your name","Harry Potter");
    if (name!=null && name!="")
     {
     document.write("Hello " + name + "! How are you today?");
     }
    }
    </script>
    </head>
    <body>
      <input type="button" onclick="show_prompt()"
     value="Show prompt box" />
    </body>
```

Module 4. Functions, Events & Exception handling

- Overview
  - Creating functions in JavaScript
  - Functions returning values
  - Events
  - Exception-handling in JavaScript
  - try, catch, throw

Creating Functions in JavaScript

- To keep the browser from executing a script when the page loads, you can put your script into a function.
- A function contains code that will be executed by an event or by a call to the function.<h6> defines the smallest heading.

- **Syntax:**

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

Example

```
<head>
<script type="text/JavaScript">
   function showMessage()
   {
   alert(“Welcome to JavaScript Functions");
   }
</script></head>
<body>
 <form>
   <input type="button" value="Click"
   onclick="showMessage()" />
</form> </body>
```

Functions returning value (return statement)

```
<script type="text/JavaScript">
function myFunction()
{
    return (“Welcome to JS");
}
</script>
</head>
<body>
<script type="text/JavaScript">
    document.write(myFunction())
</script>
```

JavaScript Events

- By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.
- Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button.

List of events recognized by JavaScript

Event	Detected when	HTML tags
onfocus=""	Form field gets focus	select, text, textarea
onblur=""	Form field looses focus	select, text, textarea
onchange=""	Content of a field changes	select, text, textarea
onselect=""	Text is selected	text, textarea
onmouseover=""	Mouse moves over a link	A
onmouseout=""	Mouse moves out of a link	A
onclick=""	Mouse clicks an object	A, button, checkbox, radio, reset, submit
onload=""	Page is finished loading	body, frameset
onunload=""	Browser opens new document	body, frameset
onSubmit=""	Submit button is clicked	form

Exception Handling (try… catch)

- try...catch statement allows you to test a block of code for errors.
  - **Syntax**

```
try
 {
      //Run some code here
 }
catch(err)
 {
     //Handle errors here
 }
```

Exception handling Example

```
    var txt="";
    function message(){
    try  {
                    adddlert("Welcome guest!");
           }catch(err)
           {
                    txt="There was an error on this page.\n\n";
                    txt+="Click OK to continue viewing this page,\n";
                    txt+="or Cancel to return to the home page.\n\n";
                     if(!confirm(txt)){
           document.location.href="http://www.w3schools.com/";
           }
      }
}
    </script></head>
<body>
    <input type="button" value="View message“
    onclick="message()"/>
</body>
```

throw

- throw statement allows you to create an exception.
- Syntax:

```
throw (exception)
```

Throw Example

```
<script type="text/JavaScript">
var x=prompt("Enter a number greater than 10:","");
try{
     if(x<10)
          throw("Err2");
    else if(isNaN(x)
          throw "Err3";
    }
catch(er)
{
    if(er=="Err2")
          alert("Error! The value is too low");
    if(er=="Err3")
          alert("Error! The value is not a number");
}
</script> </body>
```

Module 5. JavaScript Objects

- Overview
  - Built-In JavaScript Objects
    - String, Array, Math, Date
  - Browser Objects
    - Window, Document, Navigator
  - Creating User-defined Objects
  - Cookies
  - Form Validation

JavaScript Objects

- JavaScript is an Object Oriented Programming (OOP) language.
- An OOP language allows you to define your own objects and make your own variable types.
- An object is just a special kind of data. An object has properties and methods.
  - **Property:** Value associated with an object.
  - **Methods:** Actions that can be performed on object.

Example

```
<script type="text/JavaScript">
   var str="Hello world!";

   //using  length property of String object
   document.write(str.length);

   //using toUpperCase() on String object
   document.write(str.toUpperCase());
</script>
```

Built-In JavaScript objects

- String
- Date
- Math
- Boolean

String Object methods

Method	Description
charAt()	Returns the character at the specified index
concat()	Joins two or more strings, and returns a copy of the joined strings
toLowerCase()	Converts a string to lowercase letters
toUpperCase()	Converts a string to uppercase letters
split()	Splits a string into an array of substrings

Date Object

- Date object is used to work with date. and time

- Creating Date Object

  new Date()

  - **Example:**

```
<script type="text/JavaScript">
        var d=new Date();
        document.write(d);
  </script>
```

Date Object methods

Method	Description
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year (four digits)
getHours()	Returns the hour (from 0-23)
setMinutes()	Set the minutes (from 0-59)

Comparing dates

```
<script type="text/JavaScript">
    var myDate=new Date();

    //set date to 14-Jan-2010
    myDate.setFullYear(2010,0,14);
    var today = new Date();

    if (myDate>today)
        alert("Today is before 14th January 2010");
    else
        alert("Today is after 14th January 2010");
</script>
```

Array Object

- Array object is used to store multiple values in a single variable

- **Creating/ Accessing array**

```
// regular array (add an optional integer)
var names=new Array();
names[0]="Swati";
names[1]="Varsha";
names[2]=“Dolly";
```

Example

```
<body>
  <script type="text/JavaScript">
        var names=new Array();
        names[0]="Swati";
        names[1]="Varsha";
        names[2]=“Dolly";
        for (i=0;i<names.length;i++)
     {
     document.write(names[i] + "<br />");
     }
   </script>
</body>
```

Array Object Methods

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
join()	Joins all elements of an array into a string
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
sort()	Sorts the elements of an array

Example

- Join 3 arrays into single array

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

var teamLeads = ["Janes", “Roschelle"];
var developers = ["Smith", “Jacob", “Raman"];
var testers = [“Lovleen", “Iram"];
var project_resource = teamLeads.concat(developers , testers
   );
document.write(project_resource );

</script>
```

Example

Example: Numerically sorting of data

```
<script type="text/JavaScript">
    function sortNumber(a, b)
    {
    return a - b;
    }
    var n = ["10", "5", "40", "25", "100", "1"];
    document.write(n.sort(sortNumber));
</script>
```

Math Object

- The Math object allows you to perform mathematical tasks.
- The Math object includes several mathematical constants and methods.

- **Using Math Object's properties/methods**

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

Math Object Methods

Method	Description
abs(num)	Returns the absolute value of num
ceil(num)	Returns num, rounded upwards to the nearest integer
max(n1,n2,n3,n4)	Returns the number with the highest value
pow(x,y)	Returns the value of x to the power of y
sqrt(num)	Returns the square root of num

Window Object

- The window object represents an open window in a browser.
- If a document contain frames (<frame> or <iframe> tags), the browser creates one window object for the HTML document, and one additional window object for each frame.

Window Object Properties/Methods

Property/ Method	Description
status	Sets the text in the statusbar of a window
parent	Returns the parent window of the current window
alert()	Displays an alert box with a message and an OK button
close()	Closes the current window
print()	Prints the content of the current window

Navigator Object

- The Navigator object contains all information about the visitor's browser.

Navigator Object Properties/Methods

Property/ Method	Description
appName	Returns the name of the browser
appVersion	Returns the version information of the browser
cookieEnabled	Determines whether cookies are enabled in the browser
javaEnabled()	Closes the current window

Example

- **To display Browser name and version**

```
<script type="text/JavaScript">
    var browser=navigator.appName;
    var b_version=navigator.appVersion;
    var version=parseFloat(b_version);

    document.write("Browser name: "+ browser);
    document.write("<br />");
    document.write("Browser version: "+ version);
</script>
```

Document Object

- Each HTML document loaded into a browser window becomes a Document object.
- The Document object provides access to all HTML elements in a page, from within a script.

Document Object Properties/Methods

Property/ Method	Description
images[]	Returns an array of all the images in the document
cookie	Returns all name/value pairs of cookies in the document
getElementById ()	Accesses the first element with the specified id
write()	writes HTML expressions or JavaScript code to a document
writeln()	Same as write(), but adds a newline character after each statement

Example

- **Change text, URL and target of a link**

```
<head>
<script type="text/JavaScript">
        function changeLink(){
        document.getElementById('myAnchor').innerHTML="W3Schools";
        document.getElementById('myAnchor').href="http://www.w3schoo
        ls.com";
        document.getElementById('myAnchor').target="_blank";
}
</script></head>
        <body>
   <a id="myAnchor" href="http://www.microsoft.com">Microsoft</a>
   <input type="button" onclick="changeLink()" value="Change link">
</body>
```

What is a Cookie

- A cookie is a variable that is stored on the visitor's computer.
- Each time the same computer requests a page with a browser, it will send the cookie too.
- With JavaScript, you can both create and retrieve cookie values.

Creating and Setting a Cookie

```
function setCookie(c_name,value,expiredays)
  {
  var exdate=new Date();
  exdate.setDate(exdate.getDate()+expiredays);
  document.cookie=c_name+ "=" +escape(value)+
  ((expiredays==null) ? "" :
  ";expires="+exdate.toGMTString());
  }
```

Example: Retrieving Cookie information

```
function getCookie(c_name)
    {
    if (document.cookie.length>0)
      {
       c_start=document.cookie.indexOf(c_name + "=");
       if (c_start!=-1)
        {
          c_start=c_start + c_name.length+1;
          c_end=document.cookie.indexOf(";",c_start);
           if (c_end==-1) c_end=document.cookie.length;
          return unescape(document.cookie.substring(c_start,c_end));
        }
      }
    return "";
}
```

Example: Retrieving Cookie information contd…

```
function checkCookie()
    {
    username=getCookie('username');
    if (username!=null && username!="")
      {
      alert('Welcome again '+username+'!');
      }
    else
      {
      username=prompt('Please enter your name:',"");
      if (username!=null && username!="")
        {
        setCookie('username',username,365);
        }
      }
    }
```

Example: Retrieving Cookie information contd…

```
<html>
    <head>
    <script type="text/JavaScript">
    .// functions for creating, and retrieving cookie
    -
    -
    </script>
    </head>
<body onload="checkCookie()">
</body>
</html>
```

Form Validation

- JavaScript can be used to validate data in HTML forms before sending off the content to a server.
- Form data that typically are checked by a JavaScript could be:
  - has the user left required fields empty?
  - has the user entered a valid e-mail address?
  - has the user entered a valid date?
  - has the user entered text in a numeric field?

Example: Required Fields

```
function validate_required(field,alerttxt)
  {
  with (field)
   {
   if (value==null||value=="")
     {
     alert(alerttxt);return false;
     }
   else
     {
     return true;
     }
   }
  }
```

Example: Required Fields contd…

```
function validate_form(thisform)
    {
    with (thisform)
     {
     if (validate_required(email,"Email must be filled out!")==false)
              email.focus();return false;
    }

    }
    </script>
    </head>

    <body>
    <form action="submit.htm" onsubmit="return
    validate_form(this)" method="post">
    Email: <input type="text" name="email" size="30">
    <input type="submit" value="Submit">
    </form>
    </body>
```

Example: Email Validation

```
<head><html>
<script type="text/JavaScript">
function validate_email(field,alerttxt)
{
with (field)
  {
  apos=value.indexOf("@");
  dotpos=value.lastIndexOf(".");
  if (apos<1||dotpos-apos<2)
    {alert(alerttxt);return false;}
  else {return true;}
  }
}
```

Example: Email Validation contd…

```
function validate_form(thisform)
{
with (thisform)
 {
 if (validate_email(email,"Not a valid e-mail address!")==false)
  {email.focus();return false;}
 }
}
</script>
</head>
<body>
<form action="submit.htm" onsubmit="return validate_form(this);"
   method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body></html>
```

Example: Display Clock on Web Page

```
<head>
<script type="text/JavaScript">
    function startTime()
    {
    var today=new Date();
    var h=today.getHours();
    var m=today.getMinutes();
    var s=today.getSeconds();
    // add a zero in front of numbers<10
    m=checkTime(m);
    s=checkTime(s);
    document.getElementById('txt').innerHTML=h+":"+m+":"+s;
    t=setTimeout('startTime()',500);
    }
```

Example: Display Clock on Web Page contd…

```
function checkTime(i)
{
    if (i<10)
    {
     i="0" + i;
    }
return i;
}
</script>
</head>

<body onload="startTime()">
<div id="txt"></div>
</body>
```

Creating User-defined Objects

- There are two ways to create a new object:
  - **Create a direct instance of an object**

```
employeeObj=new Object();
employeeObj.firstname="John";
employeeObj.lastname="Smith";
employeeObj.age=30;
```

The following code adds a method called eat() to the personObj:

```
personObj.display=display;  // display() is already
                                //defined under
head section
```

Creating User-defined Objects contd…

- **Create a template of an object**

```
function employee(firstname,lastname,age)
  {
  this.firstname=firstname;
  this.lastname=lastname;
  this.age=age;
  this.display=display;    // function
}
```

Creating new instances of the object

```
employee1=new employee("John",“”Smith",50);
employee2=new employee("Sam",“Speilsburg",48);
```