HEXAWARE

JAVA SCRIPT

Learning material references

- HexaGuru+
  - HTML 5 and JavaScript
- Books
  - JavaScript Bible
    - John Wiley & sons (US)
  - JavaScript—A Beginner's Guide, Fourth Edition
    - McGraw-Hill/Osborne
- Web
  - http://www.w3schools.com/

Introduction to JavaScript

- JavaScript is a programming language that can be included on web pages to make them more interactive.
- Its is a lightweight programming language that is interpreted by the browser engine when the web page is loaded.
- You can use JavaScript to:
  - Change HTML content
  - Change HTML attributes
  - Change HTML styles
  - Detect visitors' browsers
  - Validate web form data

etc..

Introduction to JavaScript

- JavaScript and Java are completely different languages, both in concept and design.
- JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997.
- ECMA-262 is the official name. ECMAScript 5 (JavaScript 1.8.5 - July 2010) is the latest standard.
- JavaScript code is written into an HTML page.
- Netscape initiative
- Initially named as “Livescript”, later renamed to “Javascript”

Introduction to JavaScript

- HTML have limited functionality
  - Text, images, tables, frames
- JavaScript allows for interactivity
  - Browser/page manipulation
  - Reacting to user actions
- A type of programming language
  - Easy to learn
- Developed by Netscape
- Now a standard exists – www.ecma-international.org/publications/standards/ECMA-262.HTM

Introduction to JavaScript

- How Java Script Works?
  - JavaScript is most commonly used as a client side scripting language.
  - This means that JavaScript code is written into an HTML page (Embedded within HTML page).
  - When a user requests an HTML page with JavaScript in it, the script is sent to the browser.
  - When the browser loads the page, the browser has a built-in interpreter that reads the JavaScript code it finds in the page and runs it.
  - Executes on client (Fast, no connection needed once loaded)
  - Interpreted (not compiled)

Introduction to JavaScript

- Uses of JavaScript
  - Use it to add multimedia elements
    - With JavaScript you can show, hide, change, resize images, and create image rollovers. You can create scrolling text across the status bar.
  - Create pages dynamically
    - Based on the user's choices, the date, or other external data, JavaScript can produce pages that are customized to the user.
  - Interact with the user
    - It can do some processing of forms and can validate user input when the user submits the form.

Introduction to JavaScript

- Writing JavaScript
  - JavaScript code is typically embedded in the HTML, to be interpreted and run by the client's browser. Here are some tips to remember when writing JavaScript commands.
    - JavaScript code is case sensitive
    - White space between words and tabs are ignored
    - Line breaks are ignored except within a statement
    - JavaScript statements end with a semi- colon (;)

Introduction to JavaScript

- The SCRIPT Tag
  - The <SCRIPT> tag alerts a browser that JavaScript code follows. It is typically embedded in the HTML.

```
<SCRIPT language = "JavaScript">
    Statements
</SCRIPT>
```


Introduction to JavaScript

- Implementing JavaScript
  - There are three ways to add JavaScript commands to your Web Pages.
    - Embedding code
    - Inline code
    - External file

Introduction to JavaScript

- External File
  - You can use the SRC attribute of the <SCRIPT> tag to call JavaScript code from an external text file.
  - This is useful if you have a lot of code or you want to run it from several pages, because any number of pages can call the same external JavaScript file.
  - The text file itself contains no HTML tags.
  - It is call by the following tag:

```
<SCRIPT SRC="filename.js">

</SCRIPT>
```

Introduction to JavaScript

- Inline JavaScript
  - Using inline JavaScript allows you to easily work with HTML and JavaScript within the same page.
  - This is commonly used for temporarily testing out some ideas, and in situations where the script code is specific to that one page.

```
<script type="text/javascript">
    // JavaScript code here
</script>
```


Introduction to JavaScript

- Embedding code
  - The code will be embedded in the HTML tag itself instead of written in under the script tag.
    - click here

Data Types

- Numbers
  - 1, 3.14159, -99
- Logical (Boolean) values
  - true or false
- Strings
  - “hello”, ‘hello’
- null
  - special keyword denoting null value

String Literals

- A string literal is zero or more characters enclosed in double (") or single (') quotes. A string must be delimited by quotes of the same type; that is, either both single quotes or double quotes. The following are examples of string literals:
  - "Hello"
  - 'Hello'
  - "1234"
  - “one line \n another line”
  - "'Don\'t try that again!," I yelled.'

Introduction to JavaScript

- **Variables**
  - Variables are used to store data.
  - 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
    - strname – STRNAME (not same)

Variables

- JavaScript is a loosely typed language.

```
myVar = 33;
myVar = “Hello World”;
myVar = 33 + “Hello World”; // gets “33Hello World”
myVar = “Hello World” + 33; //  gets “Hello World33 ”
```


Introduction to JavaScript

- **Variables**
  - To declare variables, use the keyword var and the variable name:

    var userName
  - To assign values to variables, add an equal sign and the value:

    var userName = "Smith"

    var price = 100

Introduction to JavaScript

- **More Syntax**
  - ***var*** *Variablename = value;*
    - var x = 3;
    - var name = “ISU”;
    - var x, y = 0;
    - var x = null;
  - Note: Null is: 0 (number), false (Boolean)
  - There is no way to specify that a particular variable represents an integer, a string or a floating-point (real) number.

Variable Names

- A JavaScript identifier or name must start with a letter or underscore ("_"); subsequent characters can also be digits (0-9).
- Letters include the characters "A" through "Z" (uppercase) and the characters "a" through "z" (lowercase). JavaScript is case-sensitive.
- Some examples of legal names are:
  - Last_Name
  - status
  - _name

JavaScript Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Operator	Description	Example	Result
+	Addition	x=2 y=2 x+y	4
-	Subtraction	x=5 y=2 x-y	3
*	Multiplication	x=5 y=4 x*y	20
/	Division	15/5 5/2	3 2,5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

JavaScript Operators

Operator	Description	Example
==	is equal to	5==8 returns false
===	is equal to (checks for both value and type)	x=5 y="5" x==y returns true x===y returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

JavaScript Popup Boxes

- JavaScript has built-in functions for several predefined operations.
- Here are some functions which is used for Pop up boxes.
  - alert("message")
  - confirm("message")
  - prompt("message")

JavaScript Popup Boxes

- 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.

```
<script>
alert("Hello World!")
</script>
```


JavaScript Popup Boxes

- 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.

JavaScript Popup Boxes

- 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.

Introduction to JavaScript

- Objects
  - JavaScript supports programming with objects. Objects are a way of organizing the variables. The different screen elements such as Web pages, forms, text boxes, images, and buttons are treated as objects.
  - Properties and Methods
    - Every object has its own properties and methods.
    - Properties define the characteristics of an object
      - Ex. color, length, name, height, width
    - Methods are the actions that the object can perform or that can be performed on the object.
      - Ex. alert, confirm, write, open, close

JAVA Script DOM

- JavaScript is designed on a simple object-based paradigm.
- An object is a construct with properties (variables), methods or other objects.
- JavaScript uses an object model known as **Document Object Model(DOM)** to navigate through the HTML document in an hierarchy.
- DOM nodes accessed through the following methods,
- 1. getElementById() – Returns the element specified by the ID
- 2. getElementsByTagName() – Returns the list of elements with the specified tag name.
- 3. getElementsByName() – Returns the list of elements with the specified name.

Inner Text vs Inner HTML

For updating the contents of a div tag we have two options in IE

1. **innerHTML**
2. **innerText**

Text inserted using inner html will be considered as HTML tags and parsed rather innerText includes it as normal text.

Example:

```
var text="<b>This is a Text</b>"
document.getElementById("div1").innerHTML="text";
document.getElementById("div2").innerText="text";
```

Output:

Div1 value : **This is a Text**

Div2 value : This is a Text

Text made bold by the tag, since it is parsed as HTML

Tag is not parsed displayed as it is

Objects in a Page

- Every page has the following objects:
  - **navigator**: is used for browser detection. It can be used to get browser information such as appName, appCodeName, userAgent etc.
  - **window**: represents a window in browser. Window is the object of browser, It is not the object of javascript
  - **document**: contains properties based on the content of the document, such as title, background color, links, and forms.
  - **location**: has properties based on the current URL.
  - **history**: contains properties representing URLs the client has previously requested.
- Depending on its content, the document may contain other objects.
  - For instance, each form (defined by a FORM tag) in the document has a corresponding Form object.

Introduction to JavaScript

- Naming Objects
  - Objects are organized in a hierarchy. To refer to an object use :
    objectName
  - To refer to a property of an object use:
    objectName.propertyName
  - To refer to a method of an object use:
    objectName.methodName()

Statements

- Variable Declaration / Assignment
- Function Definition
- Conditionals
- Loops
- with statement
- Comments

Comments

- // Single Line
- var x // this part of the line is a comment
- /* Multiline Comment
  Line 2.....
  Line 3 */

JavaScript Basic Examples

```
<script>
document.write("Hello World!")
</script>
```
⇒ format text with HTML code - heading

```
<script>
alert("Hello World!")
</script>
```


Example

```
<script>
    x=“Hello World!”
    document.write(“welcome” + x)
</script>
```

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** - 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

Conditional Statements - 2

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

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

Conditional Statements Examples

```
<script>
    x=3
    if(x<0)
    {
        alert (“Negative”)
    }
    else
    {
        alert (“Positive”)
    }
</script>
```


Conditional Statements Examples

```
<script>
    c=confirm(“Do You Want to Continue”)
    if(c)
    {
        alert (“Welcome”)
    }
    else
    {
        alert (“Thank You”)
    }
</script>
```


Conditional Statements Examples

```
<script>
p=prompt(“Enter the Value", " ")
if(p=="06")
{
alert(“Welcome")
}
else
{
alert(“Thank you")
}
</script>
```

Loops

- Syntax

```
for ([initial expression]; [condition]; [update expression]) {
    statements
}
```

- Example

```
for (var i = 0; i < 10; i++) {
    document.write(i);
 }
```


Introduction to JavaScript

- Functions
  - With functions, we can give a name to a whole block of code, allowing us to reference it from anywhere in your program.
  - With user-defined functions, we can name a block of code and call it when we need it.
  - You define a function in the HEAD section of a web page. It is defined with the function keyword, followed by the function name and any arguments.

```
function functionName(argument)
{
Statements;
}
```


Functions

- **Example**

```
// Takes an endValue and returns 1+2+3+ … + endValue
function summation (endVal) {
    var thesum=0;
    for ( var iter = 1; iter <= endVal; iter++ )
        thesum += iter;
    return ( thesum );
}
answer  = summation(5);
```


Arguments and Parameters

```
<SCRIPT LANGUAGE="JavaScript">
<!--  to hide script contents from old browsers
  function square(i) {
      document.write("The call passed “+ i + " to the function.“+"<BR>")
      return i * i
  }
  document.write("The function returned “+square(8)+".")
// end hiding contents from old browsers  -->
</SCRIPT>
```


Arguments and Parameters

FileName: argument.htm

```
<SCRIPT LANGUAGE="JavaScript">
<!--  to hide script contents from old browsers
  function square(i) {
      document.write("The call passed “+ i + " to the function.“+"<BR>")
      return i * i
  }
  document.write("The function returned “+square(8)+".")
// end hiding contents from old browsers  -->
</SCRIPT>
```


Event Handling

Event Handling

- JavaScript provides a moderate level of event detection to pass control to functions attached to built-in event handlers
- e.g.,
  - <INPUT type="button" VALUE="button1"
  - onClick="computeSomething()">
- Events are triggered in the browser primarily by user actions such as button click, page load, form submit.

Event Handlers

- The following event handlers are available in JavaScript:
  - onAbort
  - onBlur
  - onChange
  - onClick
  - onDragDrop
  - onError
  - onFocus
  - onKeyDown
  - onKeyPress
  - onKeyUp
  - onLoad
  - onMouseDown
  - onMouseMove
  - onMouseOut
  - onMouseOver
  - onMouseUp
  - onMove
  - onReset
  - onResize
  - onSelect
  - onSubmit
  - onUnload

Event Handlers (Cont...)

- **onAbort**
  - An abort event occurs when a user aborts the loading of an image (for example by clicking a link or clicking the Stop button)
- **onBlur**
  - A blur event occurs when a select, text, or textarea field on a form loses focus.
- **onChange**
  - A change event occurs when a select, text, or textarea field loses focus and its value has been modified.
- **onClick**
  - A click event occurs when an object on a form is clicked.

Event Handlers (Cont...)

- **onDragDrop**
  - A drapDrop event occurs when a user drops an object onto the browser window, such as dropping a file on the browser window
- **onError**
  - An error event occurs when the loading of a document or image causes an error
- **onFocus**
  - A focus event occurs when a field receives input focus by tabbing with the keyboard or clicking with the mouse.
- **onKeyDown, onKeyPress, onKeyUp**
  - A keyDown, keyPress, or keyUp event occurs when a user depresses a key, presses or holds down a key, or releases a key, respectively

Event Handlers (Cont...)

- **onLoad**
  - A load event occurs when Navigator finishes loading a window or all frames within a <FRAMESET>.
  - Examples
    - In the following example, the onLoad event handler displays a greeting message after a web page is loaded.
    - <BODY onLoad="window.alert('Welcome to my home page!')">
- **onMouseDown, onMouseMove, onMouseOut, onMouseOver, and onMouseUp**
  - A MouseDown, MouseMove, MouseOut, MouseOver, or MouseUp event occurs when a user depresses a mouse button, moves a cursor, moves a cursor out of a link or image map, moves a cursor over a link, releases a mouse button, respectively

Event Handlers (Cont...)

- **onMouseOver**
  - A mouseOver event occurs once each time the mouse pointer moves over an object from outside that object.
- Example

```
<A HREF="http://www.ilstu.edu/"
   onMouseOver="window.status='A Good Place …!';
   return true">
Click me</A>
```

Return true tells the browser not to perform its own event handling routine of displaying the link's URL in the status bar

Event Handlers (Cont...)

- **OnResize**
  - A Resize event occurs when a user or script resizes a window

```
<html>
<head>
<script language="JavaScript">
window.onresize= message;
function message() {
  alert("The window has been resized!");
}
</script>
</head>
<body>
Please resize the window.
</body>
</html>
```


Event Handlers (Cont...)

- **Another look at onClick:**

```
<body>
<form name="myForm">
<input type="button" name="myButton" value="ClickMe!" >
</form>
<script language="JavaScript">
document.myForm.myButton.onClick= message;

function message() {
  alert('Click event occured!');
}

</script>
</body>
```


Event Handlers (Cont...)

- **onSelect**
  - A select event occurs when a user selects some of the text within a text or textarea field.
- **onSubmit**
  - A submit event occurs when a user submits a form
- **onUnload**
  - An unload event occurs when you exit a document.

HEXAWARE

Innovative Services

Passionate Employees

Delighted Customers

Thank you

www.hexaware.com