JAVASCRIPT

What is JavaScript?

- JavaScript is a scripting language that enables web developers/designers to build more functional and interactive websites.
- Common uses of JavaScript include:
- Alert messages
- Popup windows
- Dynamic dropdown menus
- Form validation
- Displaying date/time

JavaScript usually runs on the *client-side* (the browser's side), as opposed to *server-side* (on the web server). One benefit of doing this is performance. On the client side, JavaScript is loaded into the browser and can run as soon as it is called.

JavaScript Syntax

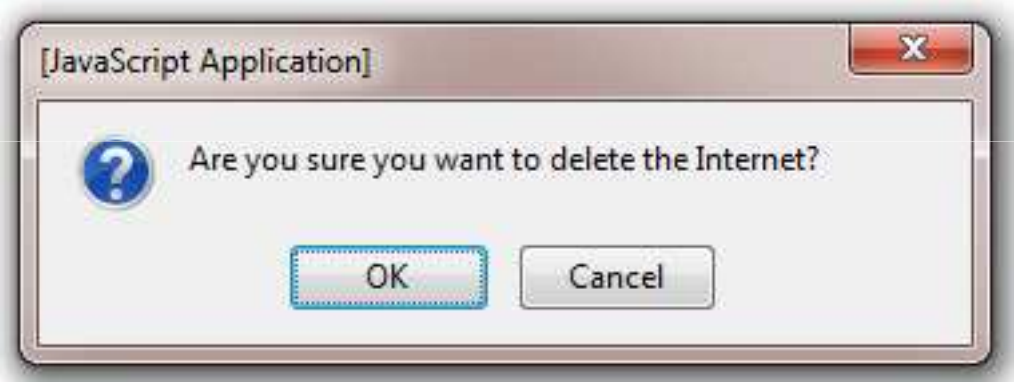
- <script type="text/javascript">
- <!-- document.write(“Welcome to javascript. This is not Java"); -->
- </script>

JavaScript Popup Boxes

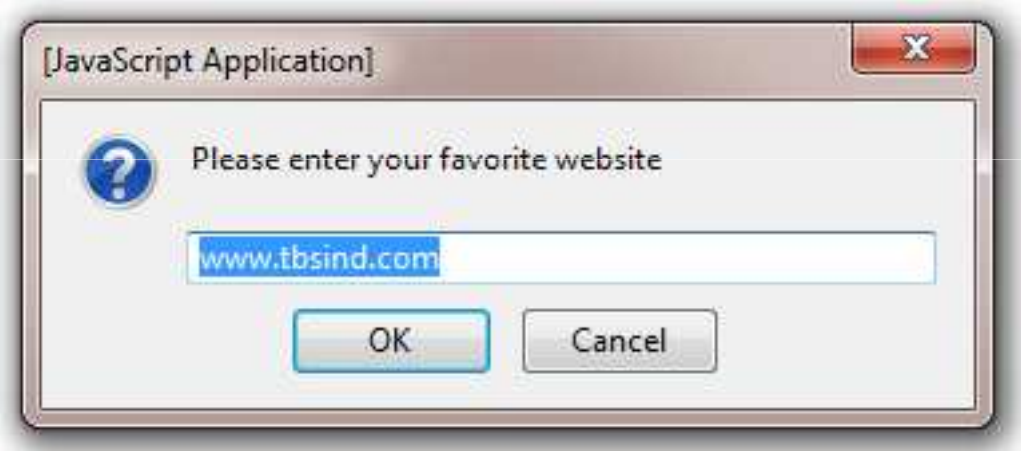
- JavaScript has three different types of popup box available for you to use. Here they are:

- 1) Alert :- Displays a message to the user

- 2) Asks the user to confirm something.



- 3) **Prompt :-** Prompts the user for information



syntax for specifying popup boxes

- Alert ---- alert("Message");
- Confirm ------ confirm("Message");
- Prompt ------- prompt("Message","Default response");

Adding JavaScript to an HTML Element

- <input type="button" **onClick="alert('Hey, Welcome to TBS');"** value="Click Me!" />
- **JavaScript "On Double Click"**
- <input type="button" **onDblClick="alert('Hey, Welcome to TBS');"** value="Double Click Me!" />

Linking to an external JavaScript file

- <script type="text/javascript"
  src="external_javascript.js"></script>

JavaScript Operators

- JavaScript operators are used to perform an operation. There are different types of operators for different uses.
- **Arithmetic Operators**

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (remainder of a division)
++	Increment
--	Decrement

- **Assignment Operators**

Operator	Description
=	Assign
+=	Add and assign. For example, x+=y is the same as x=x+y.
-=	Subtract and assign. For example, x-=y is the same as x=x-y.
*=	Multiply and assign. For example, x*=y is the same as x=x*y.
/=	Divide and assign. For example, x/=y is the same as x=x/y.
%=	Modulus and assign. For example, x%=y is the same as x=x%y.

- **Comparison Operators**

Operator	Description
==	Is equal to
===	Is identical (is equal to and is of the same type)
!=	Is not equal to
!==	Is not identical
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

- **Logical/boolean Operators**

Operator	Description
&&	and
||	or
!	not

- **String Operators**

Operator	Description
=	Assignment
+	Concatenate (join two strings together)
+=	Concatenate and assign

JavaScript Variables

- // declaring one javascript variable
- var firstName;
- // declaring multiple javascript variables
- var firstName, lastName;
- // declaring and assigning one javascript variable
- var firstName = Vaibhav';
- // declaring and assigning multiple javascript variables
- var firstName = Vaibhav ', lastName = Deshpande';

Using JavaScript variables

- <script language="javascript" type="text/javascript" >
- <!-- var firstName = prompt("What's your first name?", ""); -->
- <!-- document.write(firstName); -->
- </script>

Rules for JavaScript Variables

- Can contain any letter of the alphabet, digits 0-9, and the underscore character.
- No spaces
- No punctuation characters (eg comma, full stop, etc)
- The first character of a variable name cannot be a digit.
- JavaScript variables' names are case-sensitive. For example *firstName* and *FirstName* are two different variables.

JavaScript Functions

- In JavaScript, you will use functions a lot. A function (also known as a *method*) is a self-contained piece of code that performs a particular "function". You can recognise a function by its format - it's a piece of descriptive text, followed by open and close brackets.
- A function doesn't actually do anything until it is called. Once it is called, it takes any arguments, then performs it's function.

Writing a function in JavaScript

- <script type="text/javascript">
- <!--
- function displayMessage(firstName) {
- alert("Hello " + firstName + ", hope you like JavaScript functions!")
- }
- //-->
- </script>
- Call the function:
- <form>
- First name:
- <input type="input" name="yourName" />
- <input
- type="button"
- onclick="displayMessage(form.yourName.value)"
- value="Display Message" />
- </form>

There are 18 event handlers that you can use to link your HTML elements to a piece of JavaScript.

Event Handler	Event that it handles
onBlur	User has left the focus of the object. For example, they clicked away from a text field that was previously selected.
onChange	User has changed the object, then attempts to leave that field (i.e. clicks elsewhere).
onClick	User clicked on the object.
onDblClick	User clicked twice on the object.
onFocus	User brought the focus to the object (i.e. clicked on it/tabbed to it)
onKeydown	A key was pressed over an element.
onKeyup	A key was released over an element.
onKeypress	A key was pressed over an element then released.
onLoad	The object has loaded.
onMousedown	The cursor moved over the object and mouse/pointing device was pressed down.
onMouseup	The mouse/pointing device was released after being pressed down.
onMouseover	The cursor moved over the object (i.e. user hovers the mouse over the object).
onMousemove	The cursor moved while hovering over an object.
onMouseout	The cursor moved off the object
onReset	User has reset a form.
onSelect	User selected some or all of the contents of the object. For example, the user selected some text within a text field.
onSubmit	User submitted a form.
onUnload	User left the window (i.e. user closes the browser window).

JavaScript If Statements

- <script type="text/javascript">
- <!-- var myColor = "Blue";
- **if (myColor == "Blue")**
- **{ document.write("Just like the sky!");** } //-->
- </script>

JavaScript If Else statement

- <script type="text/javascript">
- <!-- var myColor = "Red";
- if (myColor == "Blue")
- { document.write("Just like the sky!"); }
- **else { document.write("Didn't pick blue?");** } //-->
- </script>

If Else If statement

- <script type="text/javascript">
- <!-- var myColor = "Red";
- if (myColor == "Blue") { document.write("Just like the sky!"); }
- **else if (myColor = “green") { document.write("Just like grass!");** }
- else { document.write("Suit yourself then…"); } //-->
- </script>

JavaScript Switch Statement

```
<script type="text/javascript">
<!--
var myColor = "Red";

switch (myColor)
{
case "Blue":
        document.write("Just like the sky!");
        break
case "Red":
        document.write("Just like shiraz!");
        break
default:
        document.write("Suit yourself then…");
}
//-->
</script>
```

JavaScript While Loop

- <script type="text/javascript">
- <!--
- var myBankBalance = 0;

- while (myBankBalance <= 10) {
- document.write("My bank balance is $" + myBankBalance + "
- ");
- myBankBalance ++;
- }
- //-->
- </script>

JavaScript For Loop

- <script type="text/javascript">
- <!--
- var myBankBalance = 0;

- for (myBankBalance = 0; myBankBalance <= 10; myBankBalance++)
- {
- document.write("My bank balance is $" + myBankBalance + "
- ");
- }
- //-->
- </script>

JavaScript Try... Catch

- **Code without a Try... Catch statement:**
- <script type="text/javascript">
- <!--
- document.write("My bank balance is $" + myBankBalance);
- //-->
- </script>

- The above code will result in an error. This is because the variable "myBankBalance" hasn't been declared yet.

Code with a Try... Catch statement:

```
<script type="text/javascript">
<!--
try
{
document.write("My bank balance is $" + myBankBalance);
}
catch(err)
{
document.write("Sorry, an error has occurred");
}
//-->
</script>
```

JavaScript Escape Characters

- **Without an escape character:**
- <script type="text/javascript">
- <!--
- document.write("They call it an "escape" character");
- //-->
- </script>

- The above code won't work as intended because, as soon as the browser encounters the first double quote, it will think that the string has finished. Further, it will result in an error because it will be expecting the closing bracket.

- Code with an escape character:

- <script type="text/javascript">
- <!--
- document.write("They call it an \"escape\" character");
- //-->
- </script>

- The resulting output:
- They call it an "escape" character

JavaScript Void(0)

- you could use the JavaScript void() function and pass a parameter of 0 (zero).
- **Example of void(0):**
- We have a link that should only do something (i.e. display a message) upon two clicks (i.e. a double click). If you click once, nothing should happen. We can specify the double click code by using JavaScript's "ondblclick" method. To prevent the page reloading upon a single click, we can use "JavaScript:void(0);" within the anchor link.
- Double Click Me!

JavaScript Date and Time

- JavaScript provides the following date and time functions. Note that UTC stands for *Universal Coordinated Time* which refers to the time as set by the World Time Standard. Previously referred to as Greenwich Mean Time or GMT.
- **Displaying the Current Date & Time**

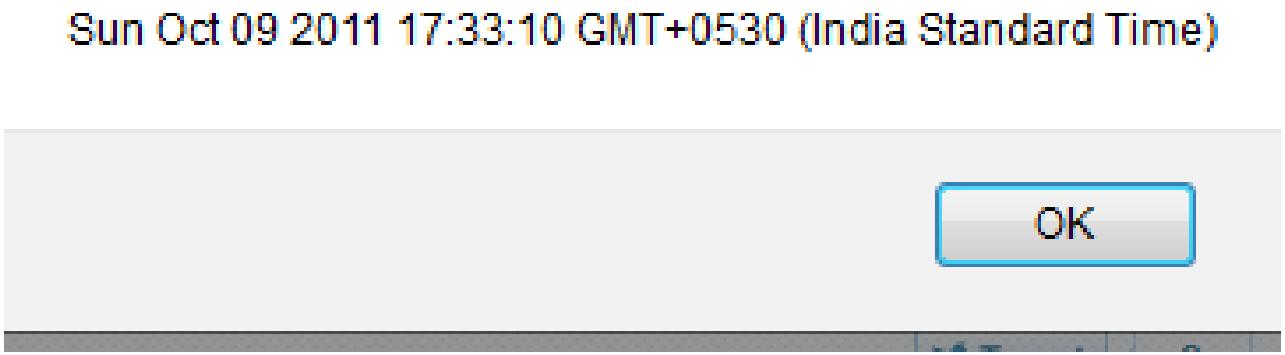
- <script language="javascript">
- var today = new Date();
- document.write(today);
- </script>

Date/Time functions

- The built-in object for date and time in JavaScript is Date. It handles all functionality: creation, modification and, partially, date formatting.

new Date()

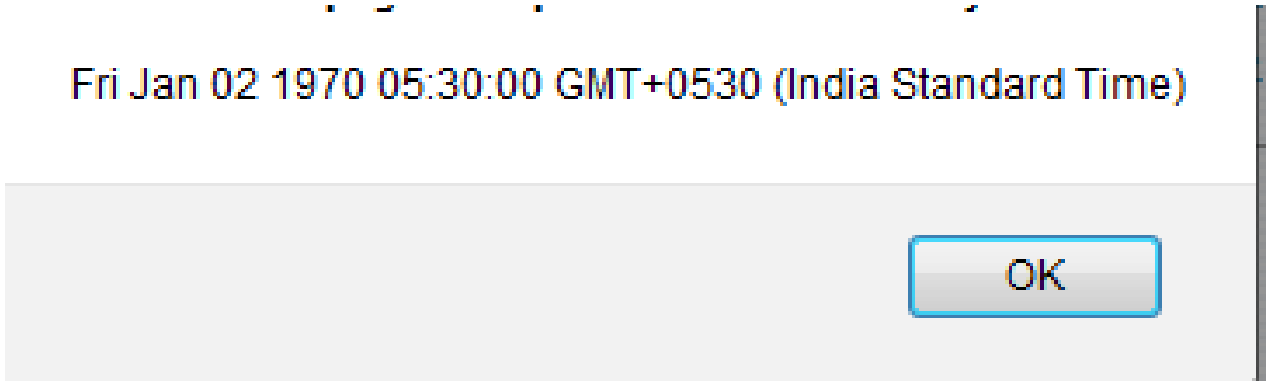
- Creates a Date object with current date and time:
- var now = new Date()
- alert(now)



new Date(milliseconds)

- Creates a Date given the number of milliseconds passed from January 1, 1970 00:00:00, GMT+0. Note, millisecond is 1/1000 of second

- var Jan02_1970 = new Date(3600*24*1000)
- alert(Jan02_1970)



- new Date(datestring)
- new Date(year, month [, day, hours, minutes, seconds, ms])
- The date can be created given it's components in the current time zone. For this format there should be at least two first arguments, next ones are optional. Note that year should have 4-digits, and month starts with 0:
- new Date(2011, 0, 1, 2, 3, 4, 567) // 1 Jan 2011, 02:03:04.567 in local timezone
- To create a date given it's components in *UTC* time zone, use static method Date.UTC():

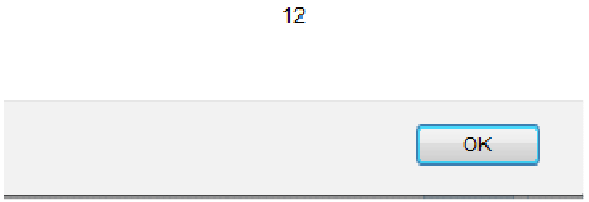
- Date.UTC(2011, 0, 1, 2, 3, 4, 567) // 1 Jan 2011, 02:03:04.567 in UTC

Getter methods

- All access to date/time components is carried out through methods.
- getFullYear()
- Get 4-digit year
- getMonth()
- Get month, from 0 to 11.
- getDay()
- Get the number of day in a week, from 0(Sunday) to 6(Saturday)
- getDate()
- Get day, from 1 to 31
- getDay()
- getHours(), getMinutes(), getSeconds(), getMilliseconds()
- **These methods work in local time zone.**
- There are UTC variants of these methods: getUTCFullYear(), getUTCMonth(), getUTCDay() etc.

- If your time zone is not UTC, the code below should output different hours:

- var d = new Date()
- alert(d.getHours())
- alert(d.getUTCHours())



Setter methods

- The following methods allow to set date components:
- setFullYear(year [, month, date])
- setMonth(month [, date])
- setDate(date)
- setHours(hour [, min, sec, ms])
- setMinutes(min [, sec, ms])
- setSeconds(sec [, ms])
- setMilliseconds(ms)
- setTime(milliseconds) (sets whole date)
- All of them except setTime() are available in UTC variants: setUTCHours().

Auto-adjusting

- The important property of the Date object is that it accepts even incorrect components and auto-adjusts itself.
- For example, let's set 62 second:
- var d = new Date()
- d.setSeconds(62)
-
- alert(d)
- // outputs correct date, increased by 1 min 2 sec.

Using JavaScript To Control Forms

Allows the Web Developer to provide responsive, intelligent, and interactive forms that can modify themselves in-response to a users input. Can be accomplished by adding or removing options and questions based upon previous responses.

Form Validation

- Three problems dealing with the data that has been entered in a form
  - People leaving required fields blank
  - people supplying garbage information
  - people needing assistance when filling out a form
- JavaScript is used to validate form fields
  - verify that a field has been filled in
  - verify that a field contains correctly formatted data
  - can not necessarily verify that the data is correct

Form Validation

- <INPUT> elements with NAME and VALUE attributes will be tested
  - The VALUE attribute for the <INPUT> element is the actual text entered
- Other elements can supply data for a form
  - <SELECT> and <TEXTAREA>
  - Are not INPUT elements
- The information is sent to the address defined in the ACTION attribute of the <FORM> element.

Form Validation

- A required field is a one that MUST have at least some content before the form will be processed.
- This does not prevent garbage input such as @~@@!!.

Example Form

```
<HTML>
<HEAD>
<TITLE>Name and Course?</TITLE>
</HEAD>
<BODY>
<FORM ACTION="mailto:yourname@youraddress.com" METHOD="post"
  enctype="text/plain" >
<INPUT TYPE="TEXT" NAME="Name">Please enter your name
<BR>
<INPUT TYPE="RADIO" NAME="course" VALUE="php">PHP
<INPUT TYPE="RADIO" NAME="course" VALUE="asp">ASP
<BR>
<INPUT TYPE="SUBMIT">        <INPUT TYPE="RESET">
</FORM>
</BODY>
</HTML>
```

Form Modifications for Validation

- Additions to <FORM> tag
  - NAME attribute:
    - NAME="mainform"
  - onSubmit event handler:
    - onSubmit="validate();”
- The event onSubmit diverts the form data to the JavaScript function validate()
- The validate() function test the required form elements

Updated Form

```
</HEAD>
<BODY>
<FORM NAME="mainform" ACTION="mailto:yourname@youraddress.com"
   METHOD="post" enctype="text/plain" onsubmit="validate();">
<INPUT TYPE="TEXT" NAME="Name" >Please enter your name
    (required)
      <BR>
      <INPUT TYPE="RADIO" NAME="course" VALUE="php">PHP
    <INPUT TYPE="RADIO" NAME="course" VALUE="asp">ASP
<BR>
<INPUT TYPE="SUBMIT">
</FORM>
</BODY>
</HTML>
```

Example Script

```
<SCRIPT>
function validate()
{
mNv=mainform.Name.value;
if (mNv=='')
    {
        alert('Your name is a required field. Please try again.');
          event.returnValue=false;
    }
if (!(mainform.course[0].checked || mainform.course[1].checked))
    {
        alert(`Course is a required field. Please try again.');
          event.returnValue=false;
    }
}
</SCRIPT>
```

Testing for Empty Field

- The first section of the validate() function examines the Name field of the form mainform.
  - If the field is empty then the form is rejected when submitted
  - mNv=mainform.Name.value;
    if (mNv=='')
    {
        alert('Your name is a required field. Please try again.');
        event.returnValue=false;
    }
- Setting the returnValue to false rejects the form

Testing for Empty Fields

- We can detect whether a radio box is selected (checked) by looking at its checked property.
- mainform.course[0].checked is true if the first radio box is selected.
- If either PHP or ASP is selected, then (mainform.course[0].checked || mainform.course[1].checked) is true
- If neither is selected, then this expression is false.

The Test

- The second section checks that a radio button has been selected.
- If we find no radio button selected, we notify the user, ask them to try again, and reject the form.

```
if (!(mainform.course[0].checked || mainform.course[1].checked))
   {
     alert('Course is a required field. Please try again.');
     event.returnValue=false;
   }
```

Checking a Series of Buttons

```
<BODY>
<FORM NAME="mainform“ ACTION=“mailto:yourname@youraddress.com”
   METHOD="post" enctype=“text/plain” onsubmit="validate();">
<BR>
<INPUT TYPE="RADIO" NAME="IceCream" VALUE="Vanilla">Vanilla
<INPUT TYPE="RADIO" NAME="IceCream" VALUE="Choc">Chocolate
<INPUT TYPE="RADIO" NAME="IceCream" VALUE="Straw">Strawberry
<INPUT TYPE="RADIO" NAME="IceCream" VALUE="RaspR">Raspberry Ripple
<INPUT TYPE="RADIO" NAME="IceCream" VALUE="Mint">Mint
<BR>
<INPUT TYPE="SUBMIT"> <INPUT TYPE=“RESET">
</FORM>
</BODY>
</HTML>
```

The Script

```
<SCRIPT>
function validate()
{
   ICOK=false;
   for (ic=0;ic<5;ic++)
   {
         if (mainform.IceCream[ic].checked) ICOK=true;
   }
   if (!ICOK)
   {     alert('Your favorite Ice Cream flavor is a required field. Please
         try again.');
         event.returnValue=false;
   }
}
</SCRIPT>
</HEAD>
```

The Script

- This code creates an ICOK variable
- This is set to false, and if the viewer has selected a favorite ice cream flavor, then the ICOK variable becomes true.
- for (ic=0;ic<5;ic++) {
  if (mainform.IceCream[ic].checked) ICOK=true;
  }
- If ICOK is still false, then no favorite was selected, and the user is prompted to try again.
- if (!ICOK) {
  alert('Your favorite Ice Cream flavor is a required field. Please try again.');
  event.returnValue=false;
  }

Hints

- Radio Buttons and Check Boxes
  - Do NOT provide a default selection frivolously
    - Order forms (shipping method)
  - Default selections can bias opinion data
- The <SELECT> element suffers from a user bias towards the default option,
  - Use a default option, which is a dummy i.e. 'Please Choose One'

Forms

```
<BODY>
FORM NAME="mainform"
ACTION="http://somewebsite.edu/cgi-bin/post-query"
METHOD="POST" onsubmit="validate();">
<SELECT NAME="Salary">
<OPTION VALUE="0" SELECTED>Salary Range</OPTION>
<OPTION VALUE="1">Less Than $10,000</OPTION>
<OPTION VALUE="2">$10,000-$20,000</OPTION>
<OPTION VALUE="3">$20,000-$30,000</OPTION>
<OPTION VALUE="4">More than $30,000</OPTION>
</SELECT>
<INPUT TYPE="SUBMIT">
</FORM>
</BODY></HTML>
```

Forms

```
<HTML>
<HEAD>
<TITLE>Required Select Field</TITLE>
<SCRIPT>
function validate()
{
if (mainform.Salary.options[0].selected)
{
alert('Please choose a Salary Range.');
event.returnValue=false;
}
}
</SCRIPT>
</HEAD>
```

Forms

- The onsubmit event is intercepted and re-routed to the validate() function.
- if (mainform.Salary.options[0].selected) {
- This line asks if the first option is the selected option.
- Lists and arrays are zero-based, which means the first item is at index 0.
- If the code is true, i.e. mainform.Salary.options[0] is selected, then the viewer has not satisfactorily answered the question, and is prompted to try again.
- We must also cancel the onsubmit event, in order to prevent the incorrect information being submitted.

Regular Expressions

- Regular Expressions define templates for strings and numbers.
- A template is a pattern of characters.
- Regular expressions provide a complete syntax for defining any template
- Provides the ability to detect patterns
  - two letters and a number
  - a word with the letter 'w'
  - a word without an 'e'

```
function validate() {
    mNv=ColForm.First.value;
    if (mNv==''){
            alert('Your name is a required field. Please try again.');
            event.returnValue=false;
    }
    mNv=ColForm.email.value;
    if (mNv==''){
            alert('Your e-mail address is a required field. Please try again.');
            event.returnValue=false;
    }
    else {
            var str=ColForm.email.value;
            var filter=/^.+@.+\..{2,3}$/
            if (filter.test(str))
            event.returnValue=true;
    else{
            alert("Please input a valid email address!");
            event.returnValue=false;
    }
    }
}
```