Unit 3
Form and Event Handling

Marks: 10 (R-2, U-4, A-4)

Course Outcome: Create event-based web forms using javascript.

Unit Outcome:

1. Write JavaScript to design a form to accept input values for the given problem.
2. Use JavaScript to implement form events to solve the given problems.
3. Develop JavaScript to dynamically assign specified attribute value to the given form control.
4. Use the given intrinsic functions with specified parameters.

Topics and Sub-topics:

3.1: Building blocks of a Form, properties and methods of form, button, text, text area, checkbox, radio button, select element.

3.2: Form events - Mouse Events, Key Events

3.3: Form objects and elements

3.4: Changing attribute value dynamically

3.5: Changing option list dynamically

3.6: Evaluating checkbox selection

3.7: Changing a label dynamically

3.8: Manipulating form elements

3.9 Intrinsic JavaScript functions, disabling elements, read only elements

Forms are one of the most common web page elements used with JavaScript.

JavaScript is commonly used with following two reasons:

- To add functionality that makes forms easier for users to fill out
- To validate or process the data that a user enters before that data is submitted to a server-side script.

JavaScript form object represents HTML form. HTML forms are a very powerful tool for interacting with users.

An HTML form is used to collect user input. The user input can then be sent to a server for processing. JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

3.1: Building blocks of a Form, properties and methods of form, button, text, text area, checkbox, radio button, select element:

A form is a section of an HTML document that contains elements such as radio buttons, text boxes and option lists. HTML form elements are also known as controls.

Elements are used as an efficient way for a user to enter information into a form. Typical form control objects also called "widgets" includes the following:

- ✓ Text box for entering a line of text.
- ✓ Push button for selecting an action.
- ✓ Radio buttons for making one selection among a group of options.
- ✓ Check boxes for selecting or deselecting a single, independent option.

The <form> element can contain one or more of the following form elements:

- <input>
- <textarea>
- <button>
- <select>
- <option>
- <fieldset>
- <label>

The attributes of the form are:

Attribute	Value	Description
action	*URL*	Specifies where to send the form-data when a form is submitted
method	get post	Specifies the HTTP method to use when sending form-data
name	*text*	Specifies the name of a form
id	Any id	Unique identifier
onSubmit	Function name	Event fired up when the form is submitted and before the execution of the action.

Syntax:

```
<form name = "myform" id = "myform" action = "page.html" onSubmit = "test()">
-----objects----
</form>
```

HTML Form Elements:

Sr. No	HTML Element	Type Property	Event Handler	Description and Events
1	<input type = "button"> or <button type = "button">	"button"	onclick	A push buttons.
2	<input type = "checkbox">	"checkbox"	onchange	A toggle button without radio button behavior.
3	<input type = "file">	"file"	onchange	An input held for entering the name of a file to upload to the web server, value property is read only.
4	<input type = "hidden"	"hidden"	none	Data submitted with the form but not visible to the user.
5	<option>	none	none	A single item within a select object, event handlers are

				an the select object and not on individual option objects.
6	<input type = “password”>	“password”	onchange	An input field for password entry where typed characters are not visible.
7	<input type = “radio”>	“radio”	onchange	A toggle button with radio button behavior where only one item is selected at a time.
8	<input type = “reset”> or <button type = “reset”>	“reset”	onclick	A push button that resets a form.
9	<select>	“select-one”	onchange	A list or drop-down menu from which one item may be selected.
10	<select multiple>	“select-multiple”	onchange	A list from which multiple items are selected.
11	<input type = “submit”> or <button type = “submit”>	“submit”	onclick	A push button that submits a form.
12	<input type = “text”>	“text”	onchange	A single line text entry field.
13	<textarea>	“textarea”	onchange	A multi-line text entry field.
14	<label>			defines a label
15	<fieldset>			tag is used to group related elements in a form. tag draws a box around the related elements.

<input> tag with its parameters:

1. name: Can be used so that the value of the element can be processed.
2. type: Can be used to specify the type of input.
3. id: Identification name of element.
4. value: Can be used to specify the initial value. It is required when type is set to checkbox or radio. It should not be used when type is set to file.
5. checked: Can be used when type is set to checkbox or radio to set the initial state of a checkbox or radio button to be selected.
6. maxlength: Can be used to specify the maximum number of characters allowed in a textbox.
7. src: Can be used when type is set to image to specify the location of an image file.
8. alt: Can be used when type is set to image to specify the alternative text of the image, which should be a short description.

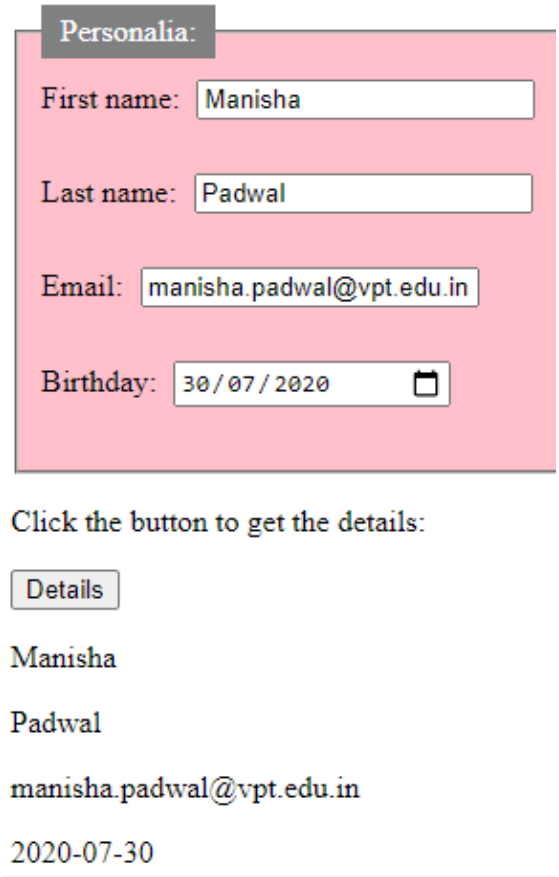
Code: To accept first name, last name, email and birthdate. After clicking on button, details will be displayed as an output.

```
<html>
<head>
<style>
fieldset
{
  background-color: pink;
}

legend
{
  background-color: gray;
  color: white;
  padding: 5px 10px;
}

input
{
  margin: 5px;
```

```
}
</style>
</head>
<body>
<form action=" ">
 <fieldset>
  <legend>Personalia:</legend>
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email"><br><br>
  <label for="birthday">Birthday:</label>
  <input type="date" id="birthday" name="birthday"><br><br>
 </fieldset>
</form>
<p>Click the button to get the details:</p>
<button onclick="myFunction()">Details</button>
<BR>
<p id="demo"></p>
<p id="demo1"></p>
<p id="demo2"></p>
<p id="demo3"></p>
<script>
function myFunction()
{
  var y = document.getElementById("fname").value;
  document.getElementById("demo").innerHTML = y;
  var x = document.getElementById("lname").value;
  document.getElementById("demo1").innerHTML = x;
  var z = document.getElementById("email").value;
  document.getElementById("demo2").innerHTML = z;
  var w = document.getElementById("birthday").value;
  document.getElementById("demo3").innerHTML = w;
```

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

Output:



Properties and Methods:

- The Form object represents a <form> element in an HTML document. The elements property is an HTML collection that provides convenient access to all elements of the form. The submit () and reset () methods allow a form to be submitted or reset under program control.
- Each form in a document is represented as an element of the documents.forms[] array. The elements of a form are collected in the array like object Form.elements.

Properties of Form:

Sr. No.	Properties	Description
1	action	Read/Write string that reflects the action attribute of the form.
2	elements[ ]	An array containing all of the elements of the form. Use it to loop through form easily.
3	encoding	Read/Write string that specifies how the form data is encoded.
4	length	The number of elements in the form.
5	method	Read/Write string that specifies how the method the form is submitted.
6	name	The name of the form.
7	target	The name of the target frame or window form is to be submitted to.

Methods of Form:

Sr. No.	Methods	Description
1	reset()	Resets the form
2	submit()	Submits a form

Methods of Events:

Sr. No.	Methods	Description
1	onReset	Code is executed when the form is reset.
2	onSubmit	Code is executed when form is submitted.

Code: Assign values to the text boxes after clicking on a button.

```
<html>    <head>
<script type="text/javascript">
function assign()
{
document.forms.book.title.value="CSS_book";
document.forms.book.author.value="Manisha Padwal";
}
</script>
</head>
<body>
```

```
<form id="book">
Title of Book:<input id="title"> <br> <br>
Author of Book:<input id="author"> <br> <br>
<input type="button" id="btn" value="Assign Values" onclick="assign()">
</form>
</body>
</html>
```

Output:

Title of Book: CSS_book

Author of Book: Manisha Padwal

Assign Values

Forms and the elements they contain can be selected from a document using

1. getElementById() method
2. getElementsByName() method
3. getElementsByTagName() method
4. innerHTML property

getElementById() method :

- The getElementById() method returns the element that has the ID attribute with the specified value.
- This method is one of the most common methods in the HTML DOM, and is used almost every time you want to manipulate, or get info from, an element on your document.

Syntax:

```
document.getElementById(elementID);
```

Code: Following code displays after clicking on button, text color is changed as red.

```
<html>
<body>

<p id="demo">Click the button to change the color of this paragraph.</p>

<button onclick="myFunction()">change color</button>

<script>
function myFunction()
{
  var x = document.getElementById("demo");
  x.style.color = "red";
}
</script>
</body>
</html>
```

Output:

Click the button to change the color of this paragraph.

change color

Click the button to change the color of this paragraph.

change color

2. getElementsByName() method

The getElementsByName() method returns a collection of all elements in the document with the specified name (the value of the name attribute).

Syntax:

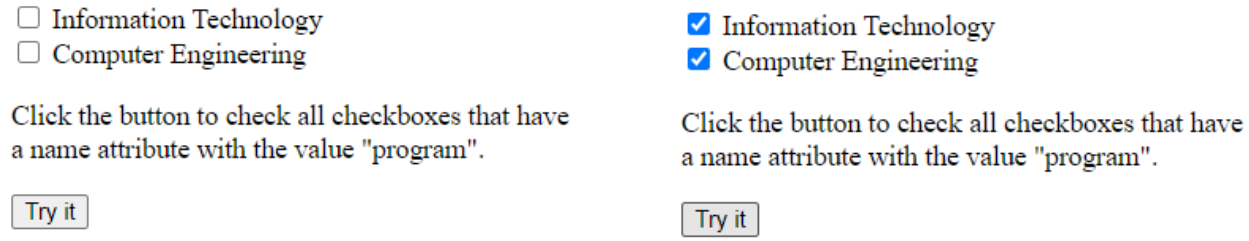
```
document.getElementsByName(name);
```

Code: Check all <input> elements with type="checkbox" in the document that have a name attribute with the value "animal":

```
<html>
<body>
 <input name="program" type="checkbox" value="IF">
Information Technology <br>
<input name="program" type="checkbox" value="CO">
Computer Engineering
<br>
<p>Click the button to check all checkboxes that have a name attribute with the value
"program".</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
  var x = document.getElementsByName("program");
  var i;
  for (i = 0; i < x.length; i++)
   {
    if (x[i].type == "checkbox")
    {
      x[i].checked = true;
    }
  }
}
</script>
</body>
```

```
</html>
```

Output:



3. getElementsByTagName() method

The getElementsByTagName() method returns a collection of all elements in the document with the specified tag name.

Syntax:

```
document.getElementsByTagName(tagname);
```

Code: Following code illustrates the use of getElementsByTagName() to count how many LI elements are present in unordered list.
Find out how many <li> elements there are in the document.

```
<html>
<body>
<p>An unordered list:</p>
<ul>
  <li>Information Technology</li>
  <li>Computer Engineering</li>
  <li>Chemical Engineering</li>
</ul>
<p>Click the button to find out how many li elements there are in this document. </p>
<button onclick="myFunction()">Click</button>
<p id="demo"></p>
<script>
function myFunction()
{
```

```
  var x = document.getElementsByTagName("LI");
  document.getElementById("demo").innerHTML = x.length;
}
</script>
</body>
</html>
```

Output:

An unordered list:

- Information Technology
- Computer Engineering
- Chemical Engineering

Click the button to find out how many li elements there are in this document.

Click

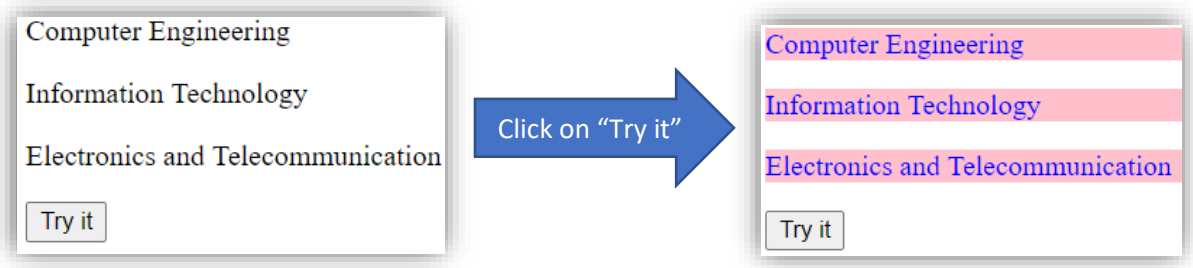
3

Code: Change the background color of all <p> elements in the document as pink and text color as blue.

```
<html>
<body>
<p>Computer Engineering</p>
<p>Information Technology</p>
<p>Electronics and Telecommunication</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
  var x = document.getElementsByTagName("P");
  var i;
  for (i = 0; i < x.length; i++) {
    x[i].style.backgroundColor = "pink";
    x[i].style.color = "blue";
```

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

Output:



4) innerHTML property

The easiest way to modify the content of an HTML element is by using the innerHTML property.

To change the content of an HTML element, use this syntax:

```
document.getElementById(id).innerHTML = new HTML;
```

Code: to change the text by using innerHTML property.

```
<html>
<body>
<script type="text/javascript">
function changeText()
{
 document.getElementById('js').innerHTML = 'Fifth Semester Javascript!!!!';
}
</script>
<p>Welcome to <b id='js'>JAVASCRIPT</b> </p>
<input type='button' onclick='changeText()' value='Change Text'/>
</body>
</html>
```

Output:

Welcome to **JAVASCRIPT**

Change Text

Welcome to **Fifth Semester Javascript!!!!**

Change Text

Code: Script for count the number of <p> tag and <H2> tag.

```
<html>
<head>
<style>
div
{
  border: 1px solid black;
  margin: 5px;
}
</style>
</head>
<body>
<div id="myDIV">
  <p>Information Technology</p>
  <p>Computer Engg.</p>
  <p>Electronics and Telecommunication</p>
  <p>Chemical Engg.</p>
</div>
<div id="myh">
<H2>Vidyalankar Polytechnic</H2>
<H2>Vidyalankar Institute of Technology </H2>
</div>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<p id="demo1"></p>
<script>
function myFunction()
{
  var x = document.getElementById("myDIV").getElementsByTagName("P");
  document.getElementById("demo").innerHTML = x.length;
```

```
  var y = document.getElementById("myh").getElementsByTagName("H2");
  document.getElementById("demo1").innerHTML = y.length;
}
</script>
</body>
</html>
```

Output:

Information Technology

Computer Engg.

Electronics and Telecommunication

Chemical Engg.

Vidyalankar Polytechnic

Vidyalankar Institute of Technology

Try it

4

2

Code:

```
<script type="text/javascript">
function changeText()
{
 var userInput = document.getElementById('userInput').value;
 document.getElementById('vp').innerHTML = userInput;
}
</script>
<p>Welcome  <b id='vp'>JavaScript</b> </p>
<input type='text' id='userInput' value='Enter Text Here' />
<input type='button' onclick='changeText()' value='Change Text'/>
```

Output:

Welcome **JavaScript**

Enter Text Here | Change Text

Welcome **Java Programming**

Java Programming | Change Text

Code:

```
<html>
<body>
Name: <input type="text" id="userInputName" value=""> <br> <br>
Password: <input type="password" id="userInputPwd" value=""> <br> <br>
<input type="button" onclick="changeText()" value="Change Text">
<p>Name is <b id="vp">JavaScript</b> </p>
<p>Password is <b id="vp1">JavaScript</b> </p>
<script>
function changeText()
{
var userInputName = document.getElementById("userInputName").value;
document.getElementById("vp").innerHTML = userInputName;
var userInputPwd = document.getElementById("userInputPwd").value;
document.getElementById("vp1").innerHTML = userInputPwd;
}
</script>
</body>
</html>
```

Output:

Name:

Password:

Change Text

Name is **JavaScript**

Password is **JavaScript**

Name: VP

Password: ••••••••••••••••

Change Text

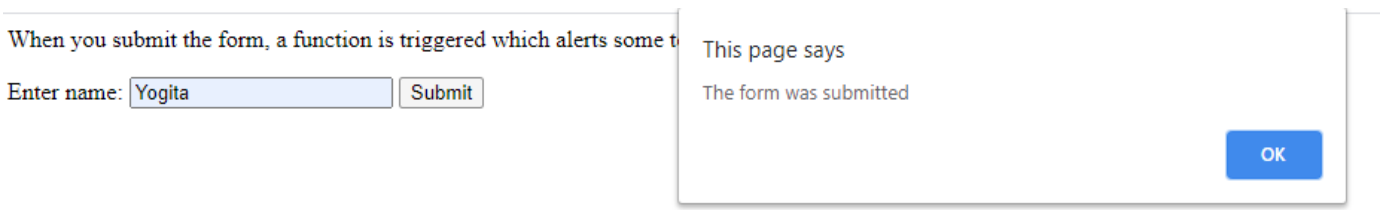
Name is **VP**

Password is **informationTech**

Code: Execute a JavaScript when a form is submitted.

```
<html>
<body>
<p>When you submit the form, a function is triggered which alerts some text.</p>
<form action="" onsubmit="myFunction()">
  Enter name: <input type="text" name="fname">
  <input type="submit" value="Submit">
</form>
<script>
function myFunction()
{
  alert("The form was submitted");
}
</script>
</body>
</html>
```

Output:



<input> Element of form:

<input> tag defines the start of an input field where the user can enter data.

Syntax:

```
<input type="value">
```

Attributes of <input> Tag:

<u>Name:</u> Name assigns a name to the input field. The name of the input field is used to send the information to the server.

<u>Value:</u> This attribute sets the value for the input field.

<u>Type:</u> type attributes indicates the type of input element that has to be given in following table.

Value	Description
button	Defines a clickable button (mostly used with a JavaScript to activate a script)
checkbox	Defines a checkbox
color	Defines a color picker
date	Defines a date control (year, month, day (no time))
email	Defines a field for an e-mail address
hidden	Defines a hidden input field
image	Defines an image as the submit button
password	Defines a password field
radio	Defines a radio button
reset	Defines a reset button
submit	Defines a submit button
text	Default. Defines a single-line text field

Button:

Button is created by using following code:

```
<form method = "GET" action = "">
<input type = "button" name = "MyButton" value = "Click" onclick = "msg()">
<form>
```

There are several types of button, which are specified by the type attribute:

1. Button which corresponds to the graphic component.
2. Submit, which is associated to the form and which starts the loading of the file assigned to the action attribute.
3. Image button in which an image loaded from a file.

A Button object also represents an HTML <button> element which is specified as follows:

```
<button name = "btn" value = "MyButton" onclick = "msg()">
```

Inside a <button> element you can put content, like text or images. But this is not the case with the buttons created with <input> tag.

Attribute	Value	Description
name	*name*	Specifies a name for the button
type	button reset submit	Specifies the type of button

value	*text*	Specifies an initial value for the button

Event handling with Button:

A push button that activates a JavaScript when it is clicked:

Syntax:

```
<input type="button" value="aaa" onclick="function_name()">
```

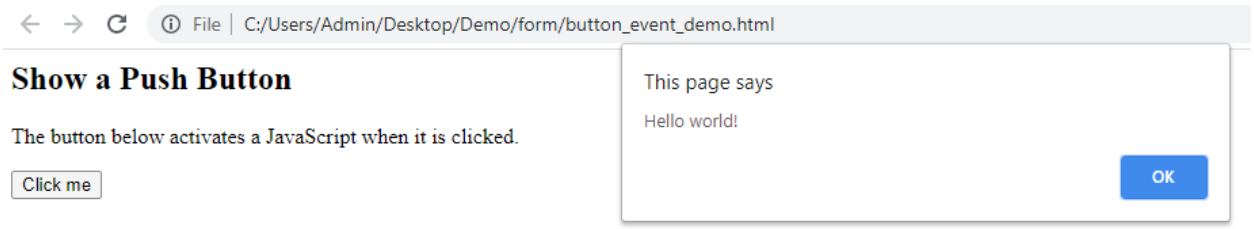
OR

```
<button onclick="Function_name()"> Click me </button>
```

Code:

```
<html>
<body>
<h2>Show a Push Button</h2>
<p>The button below activates a JavaScript when it is clicked. </p>
<form>
  <input type="button" value="Click me" onclick="msg()">
</form>
<script>
function msg()
{
  alert("Hello world!");
}
</script>
</body>
</html>
```

Output:



Code:

```
<html>
<body>
<h3>The onclick Event using Button tag. </h3>
<button onclick="myFunction()">Click me</button>
<p id="demo"></p>
<script>
function myFunction()
{
  document.getElementById("demo").innerHTML = "Welcome to JavaScript";
}
</script>
</body>
</html>
```

Output:

The onclick Event using Button tag.

Click me

Welcome to JavaScript

Code:

```
<html>
<body>
<p id="demo">Click me.</p>
<script>
document.getElementById("demo").onclick = function()
{
myFunction()
```

```
};
function myFunction()
{
  document.getElementById("demo").innerHTML = "YOU CLICKED ME!";
}
</script>
</body>
</html>
```

Output:

Click me. YOU CLICKED ME!

Text:

Input “text” is an object to enter a single line of text whose content will be part of form data.

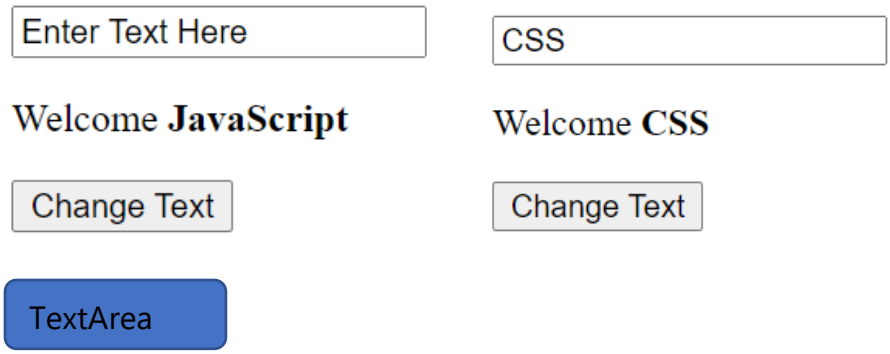
In html a text is created by following code:

<input type=”text” name=”textname” id=”textid” value=” assign_value” />

Code:

```
<script type="text/javascript">
function changeText()
{
 var userInput = document.getElementById('userInput').value;
 document.getElementById('vp').innerHTML = userInput;
}
</script>

<input type='text' id='userInput' value='Enter Text Here' />
  <p>Welcome  <b id='vp'>JavaScript</b> </p>
<input type='button' onclick='changeText()' value='Change Text'/>
</script>
```

Output:



TextArea

The Textarea object represents an HTML <textarea> element.

The <textarea> tag indicates a form field where the user can enter a large amount of text.

You can access a <textarea> element by using getElementById():

Attributes of TextArea tag:

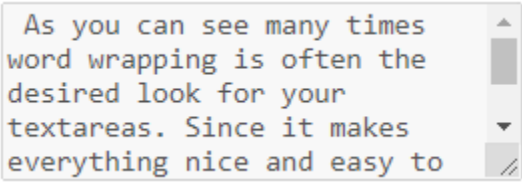
Property	Description
cols	Sets or returns the value of the cols attribute of a text area
name	Sets or returns the value of the name attribute of a text area
rows	Sets or returns the value of the rows attribute of a text area
value	Sets or returns the contents of a text area
wrap	Sets or returns the value of the wrap attribute of a text area. • Soft: Soft" forces the words to wrap once inside the textarea but once the form is submitted, the words will no longer appear as such, and line breaks and spacing are not maintained. • Hard :"Hard" wraps the words inside the text box and places line breaks at the end of each line so that when the form is submitted the text will transfer as it appears in the field, including line breaks and spacing. • "Off": sets a textarea to ignore all wrapping and places the text into one ongoing line. *textareaObject*.wrap = soft/hard/off
readonly	Setting a "yes" or "no" value for the readonly attribute determines whether or not a viewer has permission to manipulate the text inside the text field.
disabled	Disabling the textarea altogether prevents the surfer from highlighting, copying, or modifying the field in any way. To accomplish this, set the *disabled* property to "yes".

Code: to demonstrate the <textarea> and its attributes.

```
<html>
```

```
<body>

<textarea cols="30" rows="5" wrap="hard" readonly="yes" disabled="yes">
 As you can see many times word wrapping is often the desired look for your textareas.
Since it makes everything nice and easy to read and preserves line breaks.
</textarea>
</body>

</html>
```

Output:

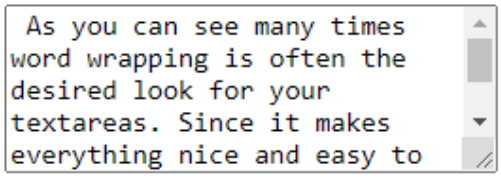


In above code, disabled="yes" that is textarea is disabled (cannot perform any highlighting, copying, or modifying) .

Without using “disabled” attribute of <textarea>

```
<textarea cols="30" rows="5" wrap="hard" readonly="yes">
```

Output will be like:



Code: to display the content from textarea.

```
<html>
<body>
Address:<br>
<textarea id="myTextarea" cols=32 Rows=5>
</textarea>
<p>Click the button to get the content of the text area.</p>
<button type="button" onclick="myFunction()">Display Address</button>
<p id="demo"></p>
```

```
<script>
function myFunction()
{
  var x = document.getElementById("myTextarea").value;
  document.getElementById("demo").innerHTML = x;
}
</script>
</body>
</html>
```

Output:

Address:

```
Vidyalankar Polyetchnic,
Antop Hill,
Wadala,

Mumbai-400037
```

Click the button to get the content of the text area.

Display Address

Vidyalankar Polyetchnic, Antop Hill, Wadala, Mumbai-400037

Methods of TextArea:

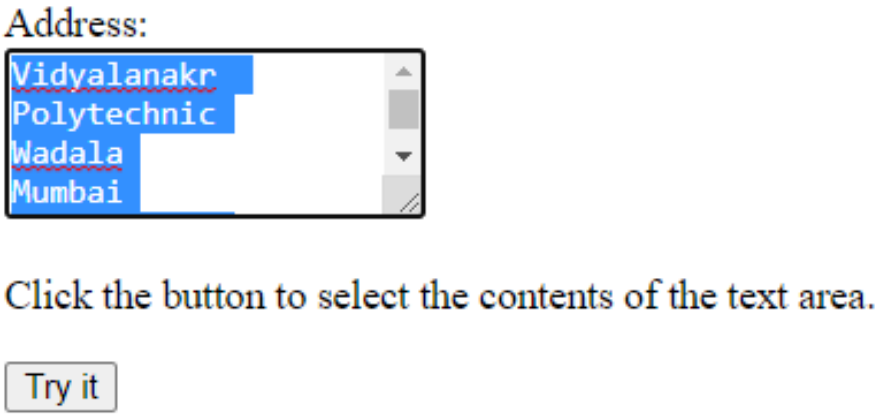
Method	Description
select()	Selects the entire contents of a text area

Code: Select the contents of a text area.

```
<html>
<body>
Address:<br>
<textarea id="myTextarea" cols=32 Rows=5>
</textarea>
<p>Click the button to get the content of the text area.</p>
<button type="button" onclick="myFunction()">Display Address</button>
<p id="demo"></p>
<script>
function myFunction()
{
```

```
  var x = document.getElementById("myTextarea").value;
  document.getElementById("demo").innerHTML = x;
}
</script>
</body>
</html>
```

Output:



Checkbox:

<input> elements of type checkbox are rendered by default as boxes that are checked (ticked) when activated. A checkbox allows you to select single values for submission in a form (or not).

Syntax for creating checkbox is:

```
<input type="checkbox" id="myCheck" onclick="myFunction()">
```

A checkbox can have only two states:

1. Checked
2. Unchecked

Code:

```
<html>
<body>
<div>
Program:
<br>
```

```
    <input type="checkbox" name="program" id="it" value="IT">
    <label for="it">Information Tech</label> <br>

    <input type="checkbox" name="program" id="co" value="CO" checked>
    <label for="co">Computer Engg</label> <br>

    <input type="checkbox" name="program" id="ej"  value="EJ">
    <label for="ej">Electronics</label> <br>
     <button onclick="validate();">Validate</button>
</div>

 <div id="status">
</div>
<script>
function validate()
{
    var elements = document.getElementsByName("program");
    var statusText = " ";
    for (var index=0;index < elements.length;index++)
    {
statusText = statusText +
elements[index].value+"="+elements[index].checked+"<br>";
    }
    document.getElementById("status").innerHTML = statusText;
}
</script>
</body>
</html>
```

Output:

Program:
- [x] Information Tech
- [x] Computer Engg
- [ ] Electronics

Validate

IT=true
CO=true
EJ=false

Radio Button:

The radiobutton allows the user to choose one of a predefined set of options. You can define groups with the name property of the radio buttons.

Radio buttons with the same name belong to the same group. Radio buttons with different names belongs to the different groups. At most one radio button can be checked in a group.

Syntax:

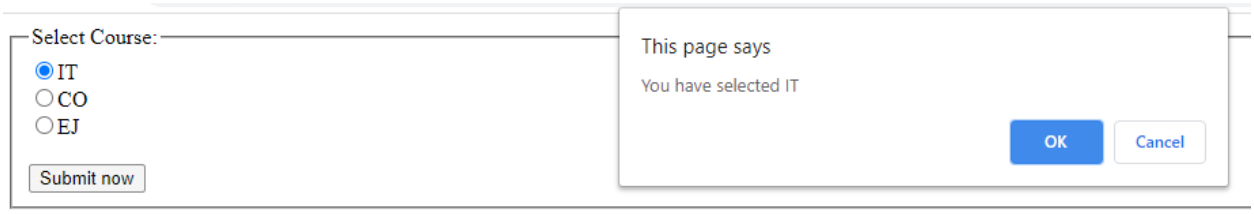
```
<input type="radio" id="male" name="gender" value="male">
```

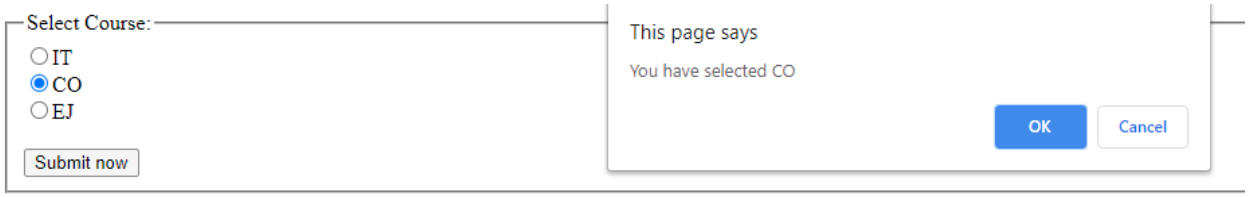
Code:

```
<html>
<body>
<form method="post" action=" " onsubmit="return ValidateForm();">
  <fieldset>
  <legend>Select Course:</legend>
  <input type="radio" name="br" value="IT" checked>IT<br>
  <input type="radio" name="br" value="CO">CO<br>
  <input type="radio" name="br" value="EJ">EJ<br>
    <br>
  <input type="submit" value="Submit now">
  </fieldset>
</form>
<script type="text/javascript">
function ValidateForm()
```

```
{
    var obj = document.getElementsByName("br");
    for(var i = 0; i < obj.length; i++)
    {
        if(obj[i].checked == true)
        {
            if(confirm("You have selected " + obj[i].value))
                return true;
            else
                return false;
        }
    }
}
</script>
</body>
</html>
```

Output:





Select:

Form SELECT elements (<select>) within your form can be accessed and manipulated in JavaScript via the corresponding Select object.

To access a SELECT element in JavaScript, use the syntax:

document.myform.selectname //where myform and selectname are names of your form/element.

document.myform.elements[i] //where i is the position of the select element within form

document.getElementById("selectid") //where "selectid" is the ID of the SELECT element on the page.

Option Object

The Option object represents an HTML <option> element.

Access an Option Object

You can access an <option> element by using getElementById().

Option Object Properties

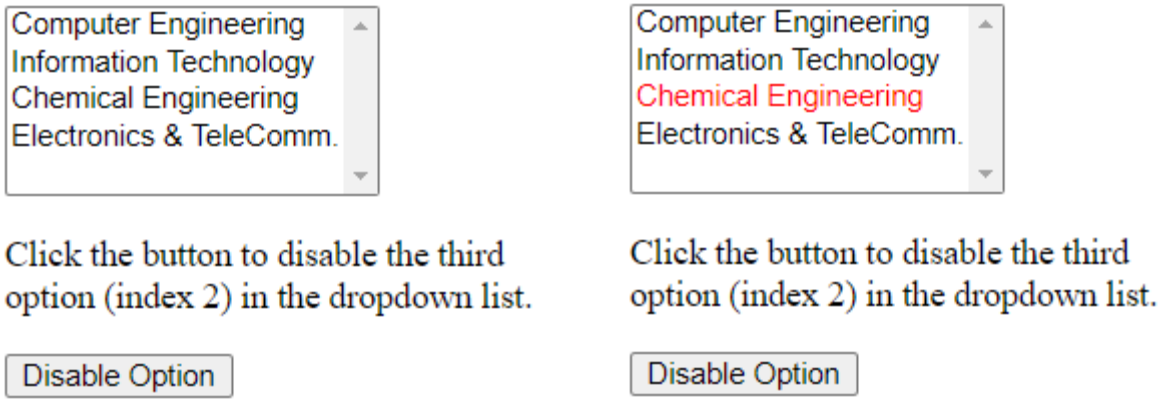
Property	Description
defaultSelected	Returns the default value of the selected attribute
disabled	Sets or returns whether an option is disabled, or not
form	Returns a reference to the form that contains the option
index	Sets or returns the index position of an option in a drop-down list
label	Sets or returns the value of the label attribute of an option in a drop-down list
selected	Sets or returns the selected state of an option
text	Sets or returns the text of an option

value	Sets or returns the value of an option to be sent to the server

Code: Disable the third option (index 2) in a drop-down list and apply color as red to disabled index.

```
<html>
<body>
<select id="programs" size="5">
  <option>Computer Engineering</option>
  <option>Information Technology</option>
  <option>Chemical Engineering</option>
  <option>Electronics & TeleComm.</option>
  </select>

<p>Click the button to disable the third option (index 2) in the dropdown list.</p>

<button onclick="myFunction()">Disable Option</button>

<script>
function myFunction()
{
  var x = document.getElementById("programs").options[2].disabled = true;
  document.getElementById("programs").options[2].style.color = "red";
 }
</script>
</body>
</html>
```

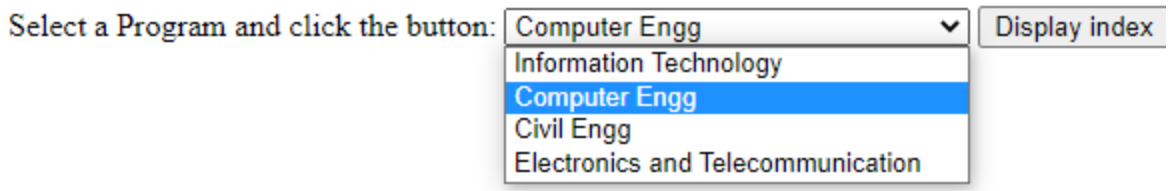
Output:

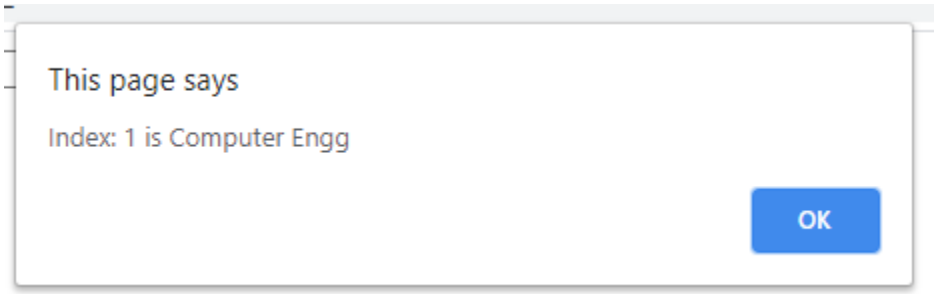


Code: Display index and text associated with that index.

```
<html>
<body>
Select a Program and click the button:
<select id="mySelect">
  <option>Information Technology</option>
  <option>Computer Engg</option>
  <option>Civil Engg</option>
  <option>Electronics and Telecommunication</option>
</select>
<button type="button" onclick="myFunction()">
Display index</button>
<script>
function myFunction()
{
  var x = document.getElementById("mySelect").selectedIndex;
  var y = document.getElementById("mySelect").options;
alert("Index: " + y[x].index + " is " + y[x].text);
}
</script>
</body>
</html>
```

Output:





3. 2 Form Events:

The form property within the document object contains an array of all forms defined within the document.

Each element within the array is a form object, the index number associated with the form object defines the order in which the form appears on the webpage.

The change in the state of an object is known as an Event. In html, there are various events which represents that some activity is performed by the user or by the browser. When javascript code is included in HTML, javascript react over these events and allow the execution. This process of reacting over the events is called Event Handling. Thus, javascript handles the HTML events via Event Handlers.

For example, when a user clicks over the browser, add javascript code, which will execute the task to be performed on the event.

Table: Event handlers for Form Elements.

Object	Event Handler
button	*onClick, onBlur, onFocus*
checkbox	*onClick, onBlur, onFocus.*

FileUpLoad	*onClick, onBlur, onFocus*
hidden	*none*
password	*onBlur, onFocus, onSelect.*
radio	*onClick, onBlur, onFocus*
reset	*onReset.*
select	*onFocus, onBlur, onChange.*
submit	*onSubmit*
text	*onClick, onBlur, onFocus , onChange*
textarea	*onClick, onBlur, onFocus , onChange*

The main utility of a button object is to trigger an event, say an *onClick()* event, but a button object has no default action.

The are several types of buttons associated with a form:

- submit
- reset
- button

These events are fired when some click related activity is registered.

Form events:

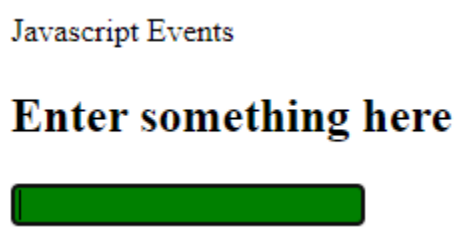
Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form

blur	onblur	When the focus is away from a form element (The onblur event occurs when an object loses focus.)
change	onchange	When the user modifies or changes the value of a form element

Code: onfocus event

```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onfocus="focusevent()"/>
<script>
 function focusevent()
 {
  document.getElementById("input1").style.background=" green";
 }
</script>
</body>
</html>
```

Output:

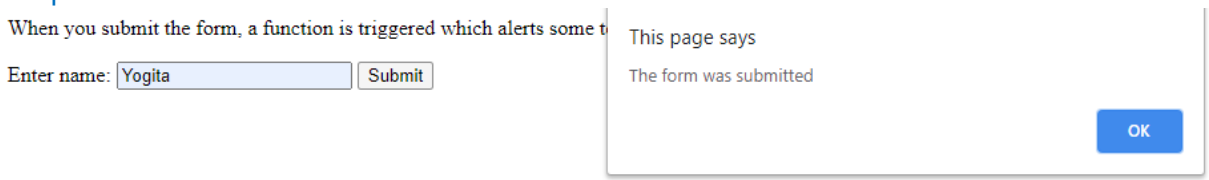


Code: onsubmit event

```
<html>
<body>

<p>When you submit the form, a function is triggered which alerts some text.</p>

<form action="" onsubmit="myFunction()">
```

```
  Enter name: <input type="text" name="fname">
  <input type="submit" value="Submit">
</form>

<script>
function myFunction()
{
  alert("The form was submitted");
}
</script>
</body>
</html>
```

Output:



Code: onblur event

Execute a JavaScript when a user leaves an input field, as soon as user leaves the input field, content in text box is appeared as in uppercase and color is blue.

```
<html>
<body>

Enter your name: <input type="text" id="fname" onblur="myFunction()">

<p>When you leave the input field, a function is triggered which transforms the input
text to upper case.</p>

<script>
function myFunction()
{
  var x = document.getElementById("fname");
  x.value = x.value.toUpperCase();
  document.getElementById("fname").style.color="blue";
}
```

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

Output:

Enter your name: YOGITA KHANDAGALE

When you leave the input field, a function is triggered which transforms the input text to upper case.

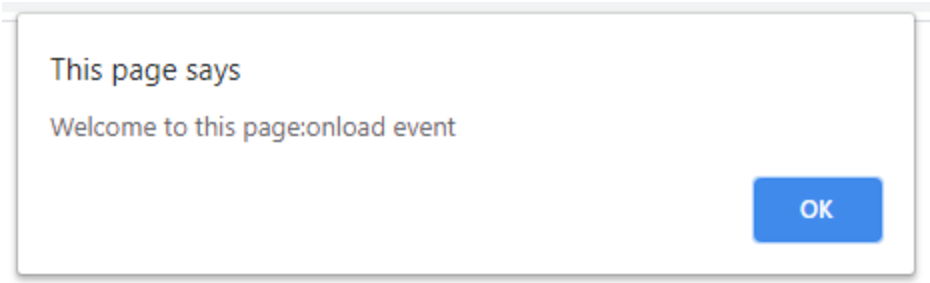
Window/Document events:

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

Code: onload event

```
<html>
<head>
    <script type="text/javascript">
      function message() {
          alert("Welcome to this page:onload event");
      }
    </script>
</head>
<body onload="message()">
When page loaded alert is displayed.
</body>
</html>
```

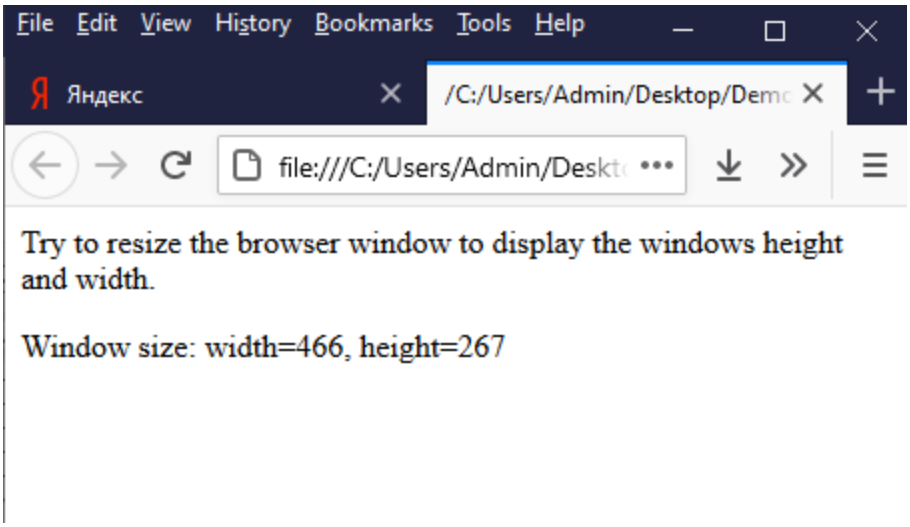
Output:



Code: onresize event

```
<html>
<body onresize="myFunction()">

<p>Try to resize the browser window to display the windows height and width.</p>

<p id="demo"></p>

<script>
function myFunction()
{
  var w = window.outerWidth;
  var h = window.outerHeight;
  var txt = "Window size: width=" + w + ", height=" + h;
  document.getElementById("demo").innerHTML = txt;
}
</script>
</body>
</html>
```

Output:



Code: onlclick event

```
<html>
 <head><title>Javascript Form Events : onClick Event</title>
 <script>
   function Japan()
 {
     alert("konnichiwa");
   }
     function India()
 {
     alert("namaste");
   }
   function Germany()
  {
     alert("Guten Tag");
   }
  </script>
</head>
<body>
 <p>Hello in Different Countries</p>
 <form>
  <input type="button" value="Japan"  onclick ="Japan()" />
```

```
    <input type="button" value="India" onclick ="India()" />

    <input type="button" value="Germany" onclick ="Germany()" />
  </form>
</body>
</html>
```

Output:

Mouse Events:

Attribute	Value	Description
onclick	*script*	Fires on a mouse click on the element
ondblclick	*script*	Fires on a mouse double-click on the element
onmousedown	*script*	Fires when a mouse button is pressed down on an element
onmousemove	*script*	Fires when the mouse pointer is moving while it is over an element
onmouseout	*script*	Fires when the mouse pointer moves out of an element
onmouseover	*script*	Fires when the mouse pointer moves over an element
onmouseup	*script*	Fires when a mouse button is released over an element
onwheel	*script*	Fires when the mouse wheel rolls up or down over an element
oncontextmenu	*script*	oncontextmenu event occurs when the user right-clicks on an element to open the context menu.

Code: onmouseout and onmouseover event

```
<html>
<html>
<body>
<img onmouseover="bigImg(this)" onmouseout="normalImg(this)" border="1" src="aa.jpg" width="64" height="64">
<script>
```

```
function bigImg(x)
{
  x.style.height ="120px";
  x.style.width ="120px";
}
function normalImg(x)
{
  x.style.height = "64px";
  x.style.width = "64px";
}
</script>
</body>  </html>
```

Output:

Code: onwheel event

When you roll the mouse over the paragraph either up or down, paragraph text will be increased to 35 pixels.

```
<html>
<body>

<div id="aa" onwheel="myFunction()">
This example demonstrates how to assign an "onwheel" event event to a DIV element.
Roll the mouse wheel over me - either up or down!</div>

<script>
function myFunction()
```

```
 {
  document.getElementById("aa").style.fontSize = "35px";
}
</script>
</body>
</html>
```

Output:

This example demonstrates how to assign an "onwheel" event event to a DIV element. Roll the mouse wheel over me - either up or down!

This example demonstrates how to assign an "onwheel" event event to a DIV element. Roll the mouse wheel over me - either up or down!

Code: ondblclick event

```
<html>
<body>
<h2>
<p id="demo" ondblclick="color_change()">Double-click me to change my text color.</p> </h2>

<script>
function color_change()
{
  document.getElementById("demo").style.color = "red";
  document.getElementById("demo").style.backgroundColor  = "yellow";
}
</script>

</body>
</html>
```

Output:

Double-click me to change my text color.

Double-click me to change my text color.

Code: onmousedown and onmouseup event

```
<html>
<body>

<p id="p1" onmousedown="mouseDown()" onmouseup="mouseUp()">
Click the text! The mouseDown() function is triggered when the mouse button is pressed down over this paragraph. The function sets the color of the text to red. The mouseUp() function is triggered when the mouse button is released. The mouseUp() function sets the color of the text to blue.
</p>
<script>
function mouseDown()
{
  document.getElementById("p1").style.color = "red";
   }
function mouseUp()
{
  document.getElementById("p1").style.color = "blue";
}
</script>
</body>
</html>
```

Output:

Click the text! The mouseDown() function is triggered when the mouse button is pressed down over this paragraph. The function sets the color of the text to red. The mouseUp() function is triggered when the mouse button is released. The mouseUp() function sets the color of the text to blue.

Click the text! The mouseDown() function is triggered when the mouse button is pressed down over this paragraph. The function sets the color of the text to red. The mouseUp() function is triggered when the mouse button is released. The mouseUp() function sets the color of the text to blue.

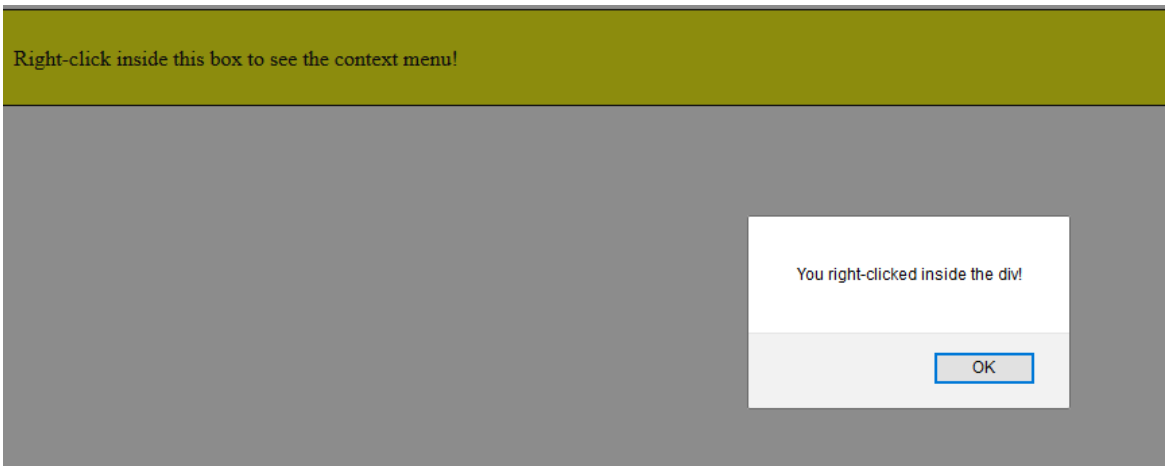
Code: oncontextmenu event

Execute a JavaScript when the user right-clicks on a element with a context menu:

```
<html>
```

```
<head>
<style>
div {
  background: yellow;
  border: 1px solid black;
  padding: 10px;
}
</style>
</head>
<body>
<div oncontextmenu="myFunction()" contextmenu="mymenu">
<p>Right-click inside this box to see the context menu!
</div>
<script>
function myFunction()
 {
  alert("You right-clicked inside the div!");
}
</script>
</body>
</html>>
```

Output:



Code: onmousemove event

Execute a JavaScript when moving the mouse pointer over a element and display the x and y coordinates.

```
<html>
<head>
<style>
div {
  width: 200px;
  height: 100px;
  border: 1px solid black;
}
</style>
</head>
<body>
<p>This example demonstrates how to assign an "onmousemove" event to a div element.</p>
<div onmousemove="myFunction(event)"></div>
<p>Mouse over the rectangle above, and get the coordinates of your mouse pointer.</p>
<p id="demo"></p>
<script>
function myFunction(e)
{
  var x = e.clientX;
  var y = e.clientY;
  var coor = "Coordinates: (" + x + "," + y + ")";
  document.getElementById("demo").innerHTML = coor;
}
</script>
</body>
</html>
```

Output:

This example demonstrates how to assign an "onmousemove" event to a div element.

Mouse over the rectangle above, and get the coordinates of your mouse pointer.

Coordinates: (64,91)

KeyEvent:

These event types belong to the KeyboardEvent Object:

Event	Description
onkeydown	The event occurs when the user is pressing a key
onkeypress	The event occurs when the user presses a key
onkeyup	The event occurs when the user releases a key

1) The **onkeydown** event occurs when the user is pressing a key (on the keyboard).

Syntax: <element onkeydown="myScript”>

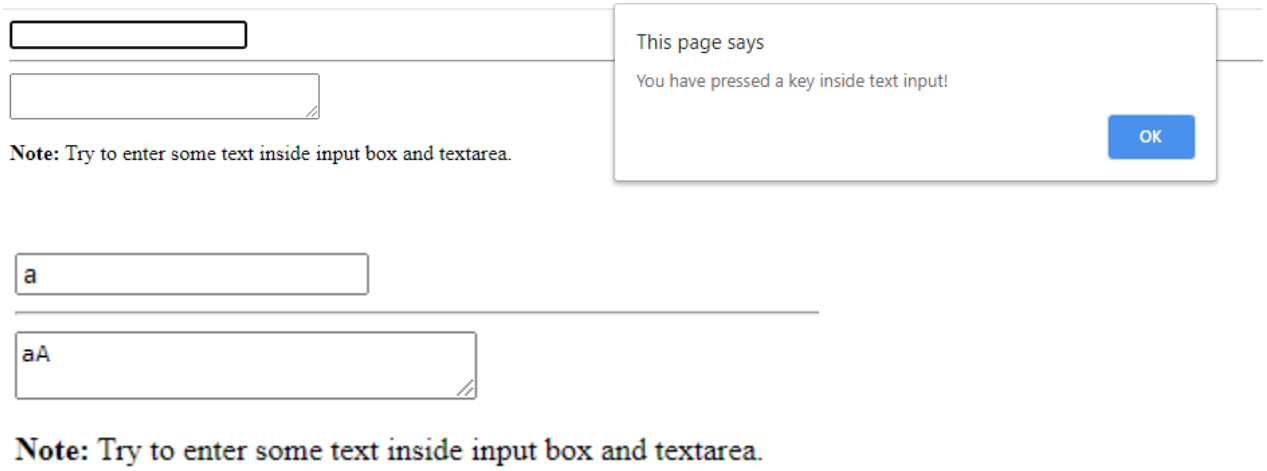
The keydown event occurs when the user presses down a key on the keyboard. You can handle the keydown event with the onkeydown event handler. The following example will show you an alert message when the keydown event occurs.

Code: onkeydown event

```
<html>
<body>
  <input type="text" onkeydown="alert('You have pressed a key inside text input!')">
  <hr>
  <textarea cols="30" onkeydown="alert('You have pressed a key inside textarea!')"></textarea>
```

```
 <p><strong>Note:</strong> Try to enter some text inside input box and textarea.</p>
</body>
</html>
```

Output:



2) The **onkeyup** event occurs when the user releases a key (on the keyboard).

Syntax: <element onkeyup="myScript">

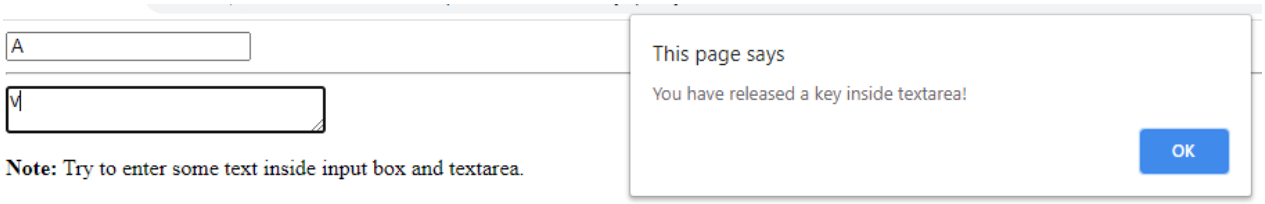
You can handle the keyup event with the onkeyup event handler.

The following example will show you an alert message when the keyup event occurs.

Code: onkeyup event

```
<html>
<body>
  <input type="text" onkeyup="alert('You have released a key inside text input!')">
  <hr>
  <textarea cols="30" onkeyup="alert('You have released a key inside textarea!')"></textarea>
 <p><strong>Note:</strong> Try to enter some text inside input box and textarea.</p>
</body>
</html>
```

Output:



3) The **onkeypress** event occurs when the user presses a key (on the keyboard).

Syntax: <*element* onkeypress="*myScript*">

The keypress event occurs when a user presses down a key on the keyboard that has a character value associated with it. For example, keys like Ctrl, Shift, Alt, Esc, Arrow keys, etc. will not generate a keypress event, but will generate a keydown and keyup event.

You can handle the keypress event with the onkeypress event handler.

The following example will show you an alert message when the keypress event occurs.

Code: onkeypress event

```
<html>
<body>
<p>Press a key inside the text field to set a red background color.</p>

<input type="text" id="demo" onkeypress="myFunction()">

<script>
function myFunction()
{ document.getElementById("demo").style.backgroundColor = "red";
}
</script>
</body>
</html>
```

Output:

Press a key inside the text field to set a red background color.

Press a key inside the text field to set a red background color.

V

In JavaScript, using the addEventListener() method, you can handle an event:

Syntax: object.addEventListener("name_of_event", myScript);

Code: addEventListener( )

```
<html>
<body>
<p>Press a key inside the text field to set a red background color.</p>

<input type="text" id="demo">
<script>
document.getElementById("demo").addEventListener("keypress", myFunction);

function myFunction()
{
  document.getElementById("demo").style.backgroundColor = "red";
}
</script>
</body>
</html>
```

Output:

Press a key inside the text field to set a red background color.

A

3.3 Changing an attribute value dynamically:

The change in any attribute value can be reflected to the user by highlighting the value or text by some color.

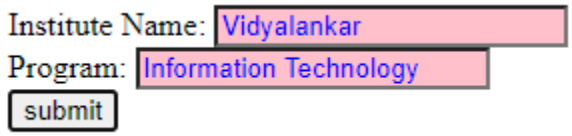
The onchange event is associated with many elements <input>, <select> of a form object and helpful to make call to a function where the change of attribute value code is written.

In following example onchange event is used with two textboxes and whenever user will make chage in value of these textboxes, text color and background of text boxes will change.

Code: onchange event to change text color as blue and background color is pink.

```
<html>
<head>
<script type="text/javascript">
function highlight(x)
{
x.style.color="blue";
x.style.backgroundColor="pink";
}
</script>
</head>
<body>
<form name="myform" action=" " method="post">
Institute Name:
<input type="text" name="iname" onchange="highlight(this)"/>
<BR>
Program:
<input type="text"  name="infotech" onchange="highlight(this)"/>
<br>
<input type="submit" value="submit" name="submit">
</form>
</body>
</html>
```

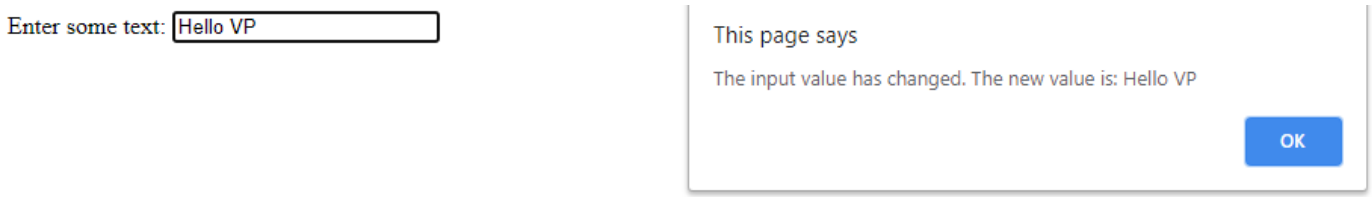
Output:



Code: onchange event to change the text in text box.

```
<html>
<body>
Enter some text:
<input type="text" name="txt" value="Hello" onchange="myFunction(this.value)">

<script>
function myFunction(val)
{
  alert("The input value has changed. The new value is: " + val);
}
</script>
</body>
</html>
```

Output:



"with" keyword

The with keyword is used as a kind of shorthand for referencing an object's properties or methods.

The object specified as an argument to with becomes the default object for the duration of the block that follows. The properties and methods for the object can be used without naming the object.

Syntax:

with (object)

{

Properties used without the object name and dot

}

Example:

Without using “with” keyword	Use of “with” keyword

<html>
<body>
 JavaScript Math functions without using "with"
<script>
var r = 10;
{
  a = (Math.PI) * r * r;
  x = r * Math.cos(Math.PI);
  y = r * Math.sin(Math.PI / 2);
  z=Math.sqrt(16);
}
document.write("a="+a+"
");
document.write("x="+x+"
");
document.write("y="+y+"
");
document.write("z="+z+"
");
</script>
</body>
</html>
 	
Output:
JavaScript Math functions without using "with"
a=314.1592653589793
x=-10
y=10
z=4
	<html>
	<body>
	 JavaScript Math functions without using "with"
	<script>
	var r = 10;
	with (Math)
	{
	  a = PI * r * r;
	  x = r * cos(PI);
	  y = r * sin(PI / 2);
	  z=sqrt(16);
	}
	document.write("a="+a+"
	");
	document.write("x="+x+"
	");
	document.write("y="+y+"
	");
	document.write("z="+z+"
	");
	</script>
	</body>
	</html>
	
	Output:
	JavaScript Math functions using "with"
	a=314.1592653589793
	x=-10
	y=10
	z=4

3.4 Changing option list dynamically:

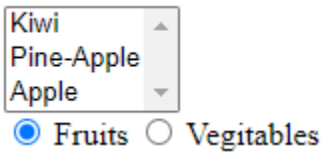
Code: Following example provides two radio buttons to the user one is for fruits and another is for vegetables.

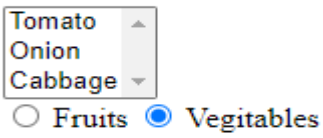
When user will select the fruits radio button, the option list should present only the fruits names to user and when user will select the vegetable radio button, the option list should present only the vegetable names to user so that user can select an appropriate element of interest.

```
<html>
<body>
<html>
<script type="text/javascript">
function modifyList(x)
{
with(document.forms.myform)
{
if(x ==1)
{
optionList[0].text="Kiwi";
optionList[0].value=1;
optionList[1].text="Pine-Apple ";
optionList[1].value=2;
optionList[2].text="Apple";
optionList[2].value=3;
}

if(x ==2)
{
optionList[0].text="Tomato";
optionList[0].value=1;
optionList[1].text="Onion ";
optionList[1].value=2;
optionList[2].text="Cabbage ";
optionList[2].value=3;
}
}
}
```

```
</script>
</head>
</body>
<form name="myform" action=" " method="post">
<select name="optionList" size="3">
<option value=1>Kiwi
<option value=1>Pine-Apple
<option value=1>Apple
</select>
<br>
<input type="radio" name="grp1" value=1 checked="true"
onclick="modifyList(this.value)"> Fruits

<input type="radio" name="grp1" value=2 onclick="modifyList(this.value)">
Vegitables
</form>
</body>
</html>
```

Output:





Code: Following example provides four list elements as name of branches. When you select a branch from list, selected branch will be displayed as output.

```
<html>
<body>
<p>Select Program from list:</p>
<select id="mySelect" onchange="myFunction()">
  <option value="CO">Computer Engg</option>
  <option value="IF">Information Technology</option>
  <option value="EJ">Electronics and Tele</option>
  <option value="CE">Chemical Engg</option>
</select>
```

```
<p id="demo"></p>
<script>
function myFunction()
{
  var x = document.getElementById("mySelect").value;
  document.getElementById("demo").innerHTML = "You selected: " + x;
}
</script>
</body>
</html>
```

Output:

Select Program from list:

Information Technology

You selected: IF

3.5 Evaluating check box selections

A checkbox is created by using the input element with the type="checkbox" attribute-value pair.

A checkbox in a form has only two states (checked or un-checked). Checkboxes can be grouped together under a common name.

Code: Following example make use of five checkboxes to provide five options to the user regarding favorite color. After the selection of favorite colors, all selected color names are displayed as output.

```
<html>
 <head>
  <title>Print value of all checked CheckBoxes on Button click.</title>
  <script type="text/javascript">
      function printChecked()
        {
          var items=document.getElementsByName('check_print');
          var selectedItems="";
          for(var i=0; i<items.length; i++)
            {
```

```
                if(items[i].type=='checkbox' && items[i].checked==true)
                    selectedItems+=items[i].value+"<br>";
            }
            document.getElementById("y").innerHTML =selectedItems;
        }
    </script>
    </head>
    <body>
    <big>Select your favourite accessories: </big><br>
    <input type="checkbox" name="check_print" value="red">red<br>
    <input type="checkbox" name="check_print" value="Blue">Blue<br>
    <input type="checkbox" name="check_print" value="Green">Green<br>
    <input type="checkbox" name="check_print" value="Yellow">Yellow<br>
    <input type="checkbox" name="check_print" value="Orange">Orange<br>
    <p><input type="button" onclick='printChecked()' value="Click me"/></p>
      You Selected:
      <p id="y"></p>
    </body>
    </html>
```

Output:

Select your favourite accessories:
- [ ] red
- [x] Blue
- [x] Green
- [ ] Yellow
- [x] Orange

Click me

You Selected:

Blue
Green
Orange

3.6 Changing labels dynamically

What is a label?
The <label>tag is used to provide a usability improvement for mouse users i.e, if a user clicks on the text within the <label> element, it toggles the control.
Approach:

- Create a label element and assign an id to that element.
- Define a button that is used to call a function. It acts as a switch to change the text in the label element.
- Define a javaScript function, that will update the label text.
- Use the innerHTML property to change the text inside the label.
  The innerHTML property sets or returns the HTML content of an element.

Code: Given an HTML document and the task is to change the text and color of a label using JavaScript.

```
 <html>
<head>
</head>
<body style="text-align:center;">
<h1 style="color:green;">
Client-SideScripting
</h1>
<h4>
Click on the button to change the text of a label
</h4>
<label id = "aaa">
Welcome to Client-Side Scripting Course.
</label>
<br>
 <button onclick="change_L()">
      Click Here!
   </button>

   <script>
      function change_L()
      {
         document.getElementById('aaa').innerHTML
            = "CSS is a client-side scripting language.";
         document.getElementById('aaa').style.color
            = "red";
      }
```

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

Output:

3.7 Manipulating form elements

Javascript make it possible with help of hidden element which is similar to any html element except it does not appear on screen.

Code: Following example is displaying the text of hidden text box after clicking on submit button.

```
  <html>
  <head>
    <title>
      HTML Input Hidden value Property
    </title>
  </head>
   <body style="text-align:center;">
      <h1 style="color:green;">
      Vidyalankar Polyetechnic
    </h1>
        <h2>Input Hidden value Property</h2>
        <input type="hidden" id="it"
            value="Information Technology">
        <button onclick="disp_hidden_Text()">
      Submit
    </button>
     <p id="demo" style="color:green;font-size:35px;"></p>
      <script>
      function disp_hidden_Text()
```

```
        {
            var x = document.getElementById("it").value;
            document.getElementById("demo").innerHTML = x;
         }
      </script>
    </body>
     </html>
```

Output:

3.8 Intrinsic javascript functions

The HTML <input> src Attribute is used to *specify the URL of the image to be used as a submit Button*. This attribute is not used with <input type="image">

Syntax:

```
<input src="URL">
```

Attribute Values: It contains a single value URL which specifies the link of source image. There are two types of URL link which are listed below:

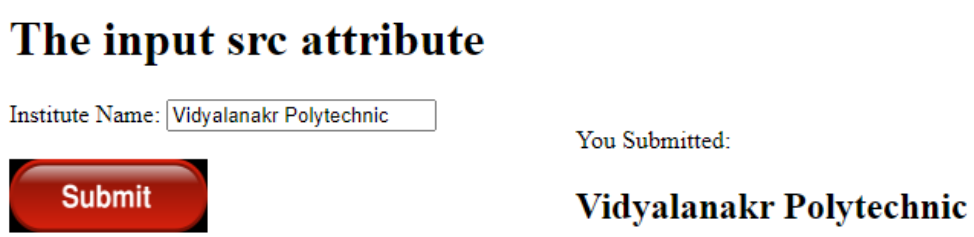
- Absolute URL: It points to another webpage.
- Relative URL: It points to other files of the same web page.

Code: Following example we have used one <img> tag to simulate the functionality of submit button. Before writing the code make sure one "submit.jpg" picture should save in your folder.

```
    <html>
    <body>
    <h1>The input src attribute</h1>

    <form action=" ">
```

```
  <label for="fname">Institute Name:</label>
  <input type="text" id="name" name="name"><br><br>
  <input type="image" src="submit.jpg" alt="Submit" width="130" height="48"
onclick="myFunction()">
</form>
<p id="demo"></p>
<script>
 function myFunction()
     {
          var x = document.getElementById("name").value;
          document.write("You Submitted:<h2>" +x +"</h2>");
    }
</script>
</body>
</html>
```

Output:



Disabling Elements:

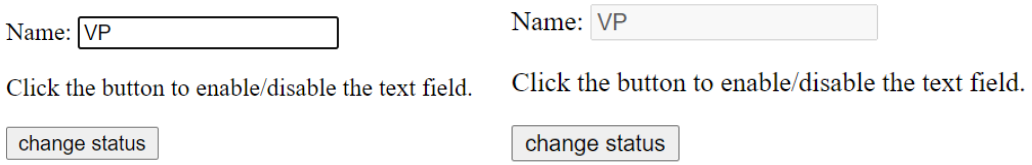
It is common to display a form with some elements disabled, which prevents the user from entering information into the element.

Code: Following example shows to enable and disable text field.

```
<html>
<body>
Name: <input type="text" id="myText">
<p>Click the button to enable/disable the text field.</p>
<button onclick="myFunction()">
change status
</button>
```

```
<script>
function myFunction()
{
var txt=document.getElementById("myText")
if ('disabled' in txt)
{
txt.disabled=!txt.disabled;
}
}
</script>
</body>
</html>
```

Output:

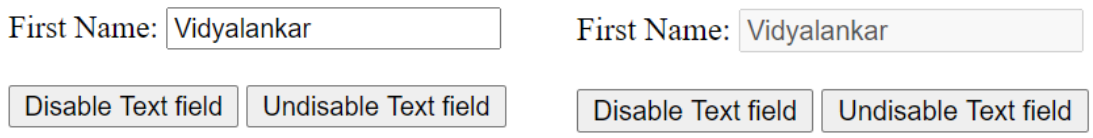


OR

```
<html>
<body>

First Name: <input type="text" id="myText"><br>
<br>
<button onclick="disableTxt()">Disable Text field</button>
<button onclick="undisableTxt()">Undisable Text field</button>

<script>
function disableTxt()
 {
  document.getElementById("myText").disabled = true;
}

function undisableTxt()
```

```
{
  document.getElementById("myText").disabled = false;
}
</script>
</body>
</html>
```

Output:



Read only elements:

The readOnly property sets or returns whether a text field is read-only, or not.
A read-only field cannot be modified. However, a user can tab to it, highlight it, and copy the text from it.
Set a text field to read-only:
document.getElementById("myText").readOnly = true;

Syntax:
To return the readOnly property: *textObject*.readOnly
To Set the readOnly property: *textObject*.readOnly = true|false

Code: Following example illustrate the use of read only property.
When user clicks on "Click here" button, text box is disabled.

```
<html>
<body>

Name: <input type="text" id="myText" value="VP">

<p>Click the button to set the text field to read-only.</p>

<p><strong>Tip:</strong> To see the effect, try to type something in the text field
before and after clicking the button.</p>
```

```
<button onclick="myFunction()">Click here </button>

<script>
function myFunction()
{
  document.getElementById("myText").readOnly = true;
}
</script>

</body>
</html>
```

Output:

Name: VP

Click the button to set the text field to read-only.

Tip: To see the effect, try to type something in the text field before and after clicking the button.

Click here