Unit 3. Form and Event Handling

- Now, there is all work will done through online like if we want to buy any product or sell any product, even your admission also done through online only you have to submit your information. And e-payment etc….
- When you visit any website they asked to fill out a form—be it an order form, subscription form, membership form, financial form, survey, and the list goes on. Although forms may seem invasive, forms are the only practical way to collect information that is necessary to conduct business on the Internet.
- Forms are created using HTML form elements such as buttons and check boxes.
- Forms used by commercial web sites also interact by using JavaScript.
- A JavaScript is used for a variety of purposes, including data validation and for dynamically interacting with elements of a form.

3.1 Building Blocks of a Form

- A form is a section of an HTML document that contains elements such as radio buttons, text boxes, check 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.
- In a business, forms are used to gather order information from a customer. Forms are also used for online surveys. Teachers use forms for online tests. Information entered into a form is sent to the web server for processing when the user clicks a submit button.
- The program that processes the form is called a Common Gateway Interface (CGI) program.
- CGI programs are written in any of programming languages, including JSP, PHP, Perl, and ASP. CGI programs typically interact with non-web applications such as databases and other systems that are necessary to process the form.
- Once processing is completed, the CGI program usually creates another web page dynamically and sends the web page to the browser that sent the form.

HTML Input Types

Here are the different input types you can use in HTML:

- `<input type="button">`
- `<input type="checkbox">`
- `<input type="color">`
- `<input type="date">`
- `<input type="datetime-local">`
- `<input type="email">`
- `<input type="file">`
- `<input type="image">`
- `<input type="month">`
- `<input type="number">`
- `<input type="password">`
- `<input type="radio">`
- `<input type="range">`
- `<input type="reset">`
- `<input type="submit">`
- `<input type="text">`
- `<input type="time">`
- `<input type="week">`

Tip: The default value of the `type` attribute is "text".

- ### Input Type -Text

`<input type="text">` defines a **single-line text input field**:

Example

```
<form>
        <label for="fname">First name:</label><br>
         <input type="text" id="fname" name="fname"><br>
         <label for="lname">Last name:</label><br>
         <input type="text" id="lname" name="lname">
</form>
```

This is how the HTML code above will be displayed in a browser:

First name:

Last name:

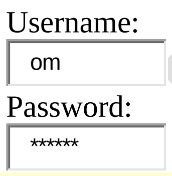
- ## Input Type-Password

<input type="password"> defines a **password field**:

Example

```html
<form>
        <label for="username">Username:</label><br>
        <input type="text" id="username" name="username"><br>
        <label for="pwd">Password:</label><br>
        <input type="password" id="pwd" name="pwd">
</form>
```

This is how the HTML code above will be displayed in a browser:



Username:
om
Password:
******

The characters in a password field are masked (shown as asterisks or circles).

- ## Input Type-Submit

<input type="submit"> defines a button for **submitting** form data to a **form-handler**.

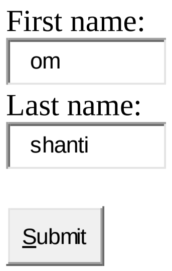
The form-handler is typically a server page with a script for processing input data.

The form-handler is specified in the form's action attribute:

Example

```html
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```

This is how the HTML code above will be displayed in a browser:



First name:
om
Last name:
shanti

If you omit the submit button's value attribute, the button will get a default text:

Example

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit">
</form>
```

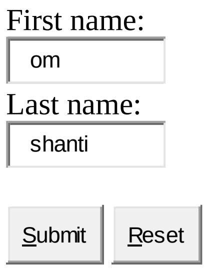
- **Input Type-Reset**

`<input type="reset">` defines a **reset button** that will reset all form values to their default values:

Example

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
  <input type="reset">
</form>
```

This is how the HTML code above will be displayed in a browser:



First name:
om
Last name:
shanti

If you change the input values and then click the "Reset" button, the form-data will be reset to the default values.

- **Input Type-Radio**

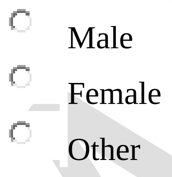
  `<input type="radio">` defines a **radio button**.

  Radio buttons let a user select ONLY ONE of a limited number of choices:

Example

```
<form>
  <input type="radio" id="male" name="gender" value="male">
  <label for="male">Male</label><br>
  <input type="radio" id="female" name="gender" value="female">
  <label for="female">Female</label><br>
  <input type="radio" id="other" name="gender" value="other">
  <label for="other">Other</label>
</form>
```

This is how the HTML code above will be displayed in a browser:


Male
Female
Other

- **Input Type-Checkbox**

  `<input type="checkbox">` defines a **checkbox**.

  Checkboxes let a user select ZERO or MORE options of a limited number of choices.

Example

```
<form>
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1">  a Bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2">  a Car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> a Boat</label>
</form>
```

This is how the HTML code above will be displayed in a browser:

☐ a Bike
☐ a Car
☐ a Boat

- **Input Type-Button**

<input type="button"> defines a **button**:

Example

```
<input type="button" onclick="alert('Hello World!')" value="Click Me!">
```

This is how the HTML code above will be displayed in a browser:

- **Input Type-Color**

The <input type="color"> is used for input fields that should contain a color.

Depending on browser support, a color picker can show up in the input field.

Example

```
<form>
  <label for="favcolor">Select your favorite color:</label>
  <input type="color" id="favcolor" name="favcolor">
</form>
```

- **Input Type-Date**

The <input type="date"> is used for input fields that should contain a date.

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
  <label for="birthday">Birthday:</label>
  <input type="date" id="birthday" name="birthday">
</form>
```

You can also use the min and max attributes to add restrictions to dates:

Example

```
<form>
  <label for="datemax">Enter a date before 1980-01-01:</label>
  <input type="date" id="datemax" name="datemax" max="1979-12-31"><br><br>
  <label for="datemin">Enter a date after 2000-01-01:</label>
  <input type="date" id="datemin" name="datemin" min="2000-01-02">
</form>
```

- ## Input Type-Datetime-local

The `<input type="datetime-local">` specifies a date and time input field, with no time zone.

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
  <label for="birthdaytime">Birthday (date and time):</label>
  <input type="datetime-local" id="birthdaytime" name="birthdaytime">
</form>
```

- ## Input Type-Email

The `<input type="email">` is used for input fields that should contain an e-mail address.

Depending on browser support, the e-mail address can be automatically validated when submitted.

Some smartphones recognize the email type, and add ".com" to the keyboard to match email input.

Example

```
<form>
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email">
</form>
```

- ## Input Type-File

The `<input type="file">` defines a file-select field and a "Browse" button for file uploads.

Example

```
<form>
  <label for="myfile">Select a file:</label>
  <input type="file" id="myfile" name="myfile">
</form>
```

- ## Input Type-Month

The `<input type="month">` allows the user to select a month and year.

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
  <label for="bdaymonth">Birthday (month and year):</label>
  <input type="month" id="bdaymonth" name="bdaymonth">
</form>
```

- ## Input Type-Number

The `<input type="number">` defines a **numeric** input field.

You can also set restrictions on what numbers are accepted.

The following example displays a numeric input field, where you can enter a value from 1 to 5:

Example

```
<form>
  <label for="quantity">Quantity (between 1 and 5):</label>
  <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>
```

Input Restrictions

Here is a list of some common input restrictions:

Attribute	Description

checked	Specifies that an input field should be pre-selected when the page loads (for type="checkbox" or type="radio")
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

The following example displays a numeric input field, where you can enter a value from 0 to 100, in steps of 10. The default value is 30:

Example

```
<form>
  <label for="quantity">Quantity:</label>
  <input type="number" id="quantity" name="quantity" min="0" max="100" step="1
0" value="30">
</form>
```

- ## Input Type-Range

The `<input type="range">` defines a control for entering a number whose exact value is not important (like a slider control). Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the `min`, `max`, and `step` attributes:

Example

```
<form>
  <label for="vol">Volume (between 0 and 50):</label>
  <input type="range" id="vol" name="vol" min="0" max="50">
</form>
```

- ## Input Type-Time

The `<input type="time">` allows the user to select a time (no time zone).

Depending on browser support, a time picker can show up in the input field.

Example

```
<form>
  <label for="appt">Select a time:</label>
  <input type="time" id="appt" name="appt">
</form>
```

- **Input Type-Week**

The <input type="week"> allows the user to select a week and year.

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
  <label for="week">Select a week:</label>
  <input type="week" id="week" name="week">
</form>
```

Events and Event Handlers

An event is something that happens when user interact with the web page, such as when he clicked a link or button, entered text into an input box or textarea, made selection in a select box, pressed key on the keyboard, moved the mouse pointer, submits a form, etc. In some cases, the Browser itself can trigger the events, such as the page load and unload events.

When an event occur, you can use a JavaScript event handler (or an event listener) to detect them and perform specific task or set of tasks. By convention, the names for event handlers always begin with the word "on", so an event handler for the click event is called onclick, similarly an event handler for the load event is called onload, event handler for the blur event is called onblur, and so on.

There are several ways to assign an event handler. The simplest way is to add them directly to the start tag of the HTML elements using the special event-handler attributes. For example, to assign a click handler for a button element, we can use onclick attribute, like this:

Example

```
<button type="button" onclick="alert('Hello World!')">Click Me</button>
```

However, to keep the JavaScript seperate from HTML, you can set up the event handler in an external JavaScript file or within the <script> and </script> tags, like this:

Example

```
<button type="button" id="myBtn">Click Me</button>
<script>
```

```
    function sayHello() {
        alert('Hello World!');
    }
    document.getElementById("myBtn").onclick = sayHello;
</script>
```

Note: Since HTML attributes are case-insensitive so onclick may also be written as onClick, OnClick or ONCLICK. But its *value* is case-sensitive.

In general, the events can be categorized into four main groups — mouse events, keyboard events, form events and document/window events.

- **Mouse Events:**

A mouse event is triggered when the user click some element, move the mouse pointer over an element, etc. Here're some most important mouse events and their event handler.

1. The Click Event (onclick)

The click event occurs when a user clicks on an element on a web page. Often, these are form elements and links. You can handle a click event with an onclick event handler.

The following example will show you an alert message when you click on the elements.

Example

```
<button type="button" onclick="alert('You have clicked a button!');">Click Me</button>
<a href="#" onclick="alert('You have clicked a link!');">Click Me</a>
```

2. The Contextmenu Event (oncontextmenu)

The contextmenu event occurs when a user clicks the right mouse button on an element to open a context menu. You can handle a contextmenu event with an oncontextmenu event handler.

The following example will show an alert message when you right-click on the elements.

Example

```
<button type="button" oncontextmenu="alert('You have right-clicked a button!');">Right Click
on Me</button>
<a href="#" oncontextmenu="alert('You have right-clicked a link!');">Right Click on Me</a>
```

3. The Mouseover Event (onmouseover)

The mouseover event occurs when a user moves the mouse pointer over an element.

You can handle the mouseover event with the onmouseover event handler. The following example will show you an alert message when you place mouse over the elements.

Example

```
<button type="button" onmouseover="alert('You have placed mouse pointer over a
button!');">Place Mouse Over Me</button>
<a href="#" onmouseover="alert('You have placed mouse pointer over a link!');">Place Mouse
Over Me</a>
```

4. The Mouseout Event (onmouseout)

The mouseout event occurs when a user moves the mouse pointer outside of an element.

You can handle the mouseout event with the onmouseout event handler. The following example will show you an alert message when the mouseout event occurs.

Example

```
    <button type="button" onmouseout="alert('You have moved out of the
button!');">Place Mouse Inside Me and Move Out</button>
    <a href="#" onmouseout="alert('You have moved out of the link!');">Place Mouse
Inside Me and Move Out</a>
```

• Keyboard Events

A keyboard event is fired when the user press or release a key on the keyboard. Here're some most important keyboard events and their event handler.

1. The Keydown Event (onkeydown)

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.

Example

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

2. The Keyup Event (onkeyup)

The keyup event occurs when the user releases a key on the keyboard.

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.

Example

```
<input type="text" onkeyup="alert('You have released a key inside text input!')">
<textarea onkeyup="alert('You have released a key inside textarea!')"></textarea>
```

3. The Keypress Event (onkeypress)

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.

Example

```
<input type="text" onkeypress="alert('You have pressed a key inside text input!')">
<textarea onkeypress="alert('You have pressed a key inside textarea!')"></textarea>
```

• Form Events

A form event is fired when a form control receive or loses focus or when the user modify a form control value such as by typing text in a text input, select any option in a select box etc. Here're some most important form events and their event handler.

1. The Focus Event (onfocus)

The focus event occurs when the user gives focus to an element on a web page.

You can handle the focus event with the onfocus event handler. The following example will highlight the background of text input in yellow color when it receives the focus.

Example

```
<script>
 function highlightInput(elm)
{
elm.style.background = "yellow";
 }
</script>
<input type="text" onfocus="highlightInput(this)">
<button type="button">Button</button>
```

Note: The value of this keyword inside an event handler refers to the element which has the handler on it (i.e. where the event is currently being delivered).

2. The Blur Event (onblur)

The blur event occurs when the user takes the focus away from a form element or a window.

You can handle the blur event with the onblur event handler. The following example will show you an alert message when the text input element loses focus.

Example

```
<input type="text" onblur="alert('Text input loses focus!')">
<button type="button">Submit</button>
```

To take the focus away from a form element first click inside of it then press the tab key on the keyboard, give focus on something else, or click outside of it.

3. The Change Event (onchange)

The change event occurs when a user changes the value of a form element.

You can handle the change event with the onchange event handler. The following example will show you an alert message when you change the option in the select box.

Example

```
<select onchange="alert('You have changed the selection!');">
   <option>Select</option>
 <option>Male</option>
<option>Female</option>
```

```
</select>
```

4.The Submit Event (onsubmit)

The submit event only occurs when the user submits a form on a web page.

You can handle the submit event with the onsubmit event handler. The following example will show you an alert message while submitting the form to the server.

Example

```
<form action="action.php" method="post" onsubmit="alert('Form data will be submitted to the server!');">
<label>First Name:</label>
<input type="text" name="first-name" required>
<input type="submit" value="Submit">
</form>
```

• Document/Window Events

Events are also triggered in situations when the page has loaded or when user resize the browser window, etc. Here're some most important document/window events and their event handler.

1. The Load Event (onload)

The load event occurs when a web page has finished loading in the web browser.

You can handle the load event with the onload event handler. The following example will show you an alert message as soon as the page finishes loading.

Example

```
<body onload="window.alert('Page is loaded successfully!');">
 <h1>This is a heading</h1>
<p>This is paragraph of text.</p>
</body>
```

2. The Unload Event (onunload)

The unload event occurs when a user leaves the current web page.

You can handle the unload event with the onunload event handler. The following example will show you an alert message when you try to leave the page.

Example

```
<body onunload="alert('Are you sure you want to leave this page?');">
  <h1>This is a heading</h1>
 <p>This is paragraph of text.</p>
</body>
```