



**Course Code: SE-3003**

**Course : Web Engineering Lab**

**Spring 2023, Lab Manual – 03**

**LLO 01: Design and Implement a simple web application.**

## **Contents:**

- Intro to Web Engineering
- Technologies
- Tools
- HTML Tags basic and advanced
- Intro to CSS

## **Introduction to Web Engineering**

Web Engineering is the application of systematic and quantifiable approaches (concepts methods, techniques tools) to cost - effective requirements analysis, design, implementation, testing, operation, and maintenance of **high quality Web applications**.

## **Technologies to be studied**

- HTML
- CSS
- JavaScript
- Bootstrap
- JQuery
- PHP
- MySQL [Database]
- Laravel [PHP FRAMEWORK]

## **Tools – IDEs**

- Visual Studio Code
- Adobe Dreamweaver
- Visual Studio

## **3.1 The <form> Element**

The HTML <form> element is used to create an HTML form for user input:

*<form>*

...

*form elements*

...

*</form>*

The <form> element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc. All the different form elements are covered in this chapter: HTML Form Elements.

The <input> Element. The HTML <input> element is the most used form element. An <input> element can be displayed in many ways, depending on the type attribute.

**Here are some examples:**

Type	Description
<code>&lt;input type="text"&gt;</code>	Displays a single-line text input field
<code>&lt;input type="radio"&gt;</code>	Displays a radio button (for selecting one of many choices)
<code>&lt;input type="checkbox"&gt;</code>	Displays a checkbox (for selecting zero or more of many choices)
<code>&lt;input type="submit"&gt;</code>	Displays a submit button (for submitting the form)
<code>&lt;input type="button"&gt;</code>	Displays a clickable button

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

- `<input>`
- `<label>`
- `<select>`
- `<textarea>`
- `<button>`
- `<fieldset>`
- `<legend>`
- `<datalist>`
- `<output>`
- `<option>`
- `<optgroup>`

## Example: **A form with input fields for text:**

`<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>`

### Text input fields

First name:

Last name:

## The `<label>` Element:

Notice the use of the `<label>` element in the example above.

The `<label>` tag defines a label for many form elements.

The `<label>` element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element. The `<label>` element also help users who have difficulty clicking on very small regions (such as

radio buttons or checkboxes) - because when the user clicks the text within the <label> element, it toggles the radio button/checkbox.

The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

## Radio Buttons

The <input type="radio"> defines a radio button.

Radio buttons let a user select ONE of a limited number of choices.

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>Radio Buttons</h2>
```

```
<p>Choose your favorite Web language:</p>
```

```
<form>
```

```
  <input type="radio" id="html" name="fav_language" value="HTML">
```

```
  <label for="html">HTML</label><br>
```

```
  <input type="radio" id="css" name="fav_language" value="CSS">
```

```
  <label for="css">CSS</label><br>
```

```
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
```

```
  <label for="javascript">JavaScript</label>
```

```
</form>
```

```
</body>
```

```
</html>
```

```

<!DOCTYPE html>
<html>
<body>

<h2>Radio Buttons</h2>

<p>Choose your favorite Web language:</p>

<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>

</body>
</html>

```

## Radio Buttons

Choose your favorite Web language:

- ☐ HTML
- ☐ CSS
- ☐ JavaScript

# Checkboxes

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

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

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>Checkboxes</h2>
```

```
<p>The <strong>input type="checkbox"</strong> defines a checkbox:</p>
```

```
<form action="/action_page.php">
```

```
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
```

```
  <label for="vehicle1"> I have a bike</label><br>
```

```
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
```

```
  <label for="vehicle2"> I have a car</label><br>
```

```
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
```

```
  <label for="vehicle3"> I have a boat</label><br><br>
```

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

</form>

</body>

</html>
```

```
<!DOCTYPE html>
<html>
<body>

<h2>Checkboxes</h2>
<p>The <strong>input type="checkbox"</strong> defines a checkbox:</p>

<form action="/action_page.php">
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label><br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

## Checkboxes

The `input type="checkbox"` defines a checkbox:

- ☐ I have a bike
- ☐ I have a car
- ☐ I have a boat

Submit

# The Submit Button

The `<input type="submit">` defines a button for submitting the form data to a form-handler. The form-handler is typically a file on the server with a script for processing input data.

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

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

## 3.2 The Method Attribute

The method attribute specifies the HTTP method to be used when submitting the form data.

The form-data can be sent as URL variables (with `method="get"`) or as HTTP post transaction (with `method="post"`).

The default HTTP method when submitting form data is GET.

```
<form action="/action_page.php" method="get">
```

```
<form action="/action_page.php" method="post">
```

#### Notes on GET:

- Appends the form data to the URL, in name/value pairs
- NEVER use GET to send sensitive data! (the submitted form data is visible in the URL!)
- The length of a URL is limited (2048 characters)
- Useful for form submissions where a user wants to bookmark the result
- GET is good for non-secure data, like query strings in Google

#### Notes on POST:

- Appends the form data inside the body of the HTTP request (the submitted form data is not shown in the URL)
- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked

## 3.3 HTML Input Attributes

### The value Attribute

The input value attribute specifies an initial value for an input field:

```
<input type="text" id="fname" name="fname" value="John"><br>
```

### The readonly Attribute

A read-only input field cannot be modified (however, a user can tab to it, highlight it, and copy the text from it). The value of a read-only input field will be sent when submitting the form!

```
<input type="text" id="fname" name="fname" value="John" readonly><br>
```

### The disabled Attribute

The input disabled attribute specifies that an input field should be disabled. A disabled input field is unusable and un-clickable. The value of a disabled input field will not be sent when submitting the form!

```
<input type="text" id="fname" name="fname" value="John" disabled><br>
```

## The size Attribute

The input size attribute specifies the visible width, in characters, of an input field.

The default value for size is 20.

**Note:** The size attribute works with the following input types: text, search, tel, url, email, and password.

```
<input type="text" id="fname" name="fname" size="50"><br>
```

## The maxlength Attribute

The input maxlength attribute specifies the maximum number of characters allowed in an input field.

**Note:** When a maxlength is set, the input field will not accept more than the specified number of characters. However, this attribute does not provide any feedback. So, if you want to alert the user, you must write JavaScript code.

```
<input type="text" id="pin" name="pin" maxlength="4" size="4">
```

## The min and max Attributes

The input min and max attributes specify the minimum and maximum values for an input field. The min and max attributes work with the following input types: number, range, date, datetime-local, month, time and week.

**Tip:** Use the max and min attributes together to create a range of legal values.

```
<input type="date" id="datemax" name="datemax" max="1979-12-31"><br><br>
```

```
<input type="date" id="datemin" name="datemin" min="2000-01-02"><br><br>
```



## The multiple Attribute

The input multiple attribute specifies that the user is allowed to enter more than one value in an input field. The multiple attribute works with the following input types: email, and file.

```
<input type="file" id="files" name="files" multiple>
```

## The pattern Attribute

The input pattern attribute specifies a regular expression that the input field's value is checked against, when the form is submitted. The pattern attribute works with the following input types: text, date, search, url, tel, email, and password.

**Tip:** Use the global title attribute to describe the pattern to help the user.

```
<form>
  <label for="country_code">Country code:</label>
  <input type="text" id="country_code" name="country_code"
    pattern="[A-Za-z]{3}" title="Three letter country code">
</form>
```

## The placeholder Attribute

The input placeholder attribute specifies a short hint that describes the expected value of an input field (a sample value or a short description of the expected format). The short hint is displayed in the input field before the user enters a value. The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

```
<form>
  <label for="phone">Enter a phone number:</label>
  <input type="tel" id="phone" name="phone"
    placeholder="123-45-678"
    pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```

## The required Attribute

The input required attribute specifies that an input field must be filled out before submitting the form.

The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

```
<form>
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required>
</form>
```

## 3.4 CSS Layout

### Overflow:

The CSS overflow property controls what happens to content that is too big to fit into an area.

The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The overflow property has the following values:

- visible - Default. The overflow is not clipped. The content renders outside the element's box
- hidden - The overflow is clipped, and the rest of the content will be invisible
- scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
- auto - Similar to scroll, but it adds scrollbars only when necessary

```
div {
  width: 200px;
  height: 65px;
  background-color: coral;
  overflow: visible;
}
```

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

## CSS z-index Property

The z-index property specifies the stack order of an element. An element with greater stack order is always in front of an element with a lower stack order.

**Note:** z-index only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky) and flex items (elements that are direct children of display:flex elements). **Note:** If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

```
img {  
  position: absolute;  
  left: 0px;  
  top: 0px;  
  z-index: -1;  
}
```

## CSS The Position Property

The `position` property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky). The position property specifies the type of positioning method used for an element.

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

## Static

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.static {
  position: static;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: static;</h2>

<p>An element with position: static; is not positioned in any special
way; it is always positioned according to the normal flow of the page:
</p>

<div class="static">
This div element has position: static;
</div>

...

```

### position: static;

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

This div element has position: static;

## Relative:

An element with **position: relative;** is positioned relative to its normal position.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: relative;</h2>

<p>An element with position: relative; is positioned relative to its normal position:</p>

<div class="relative">
This div element has position: relative;
</div>

</body>
</html>

```

### position: relative;

An element with position: relative; is positioned relative to its normal position:

This div element has position: relative;

## Fixed

An element with **position: fixed;** is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

```

<!DOCTYPE html>
<html>
<head>
<style>
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: fixed;</h2>

<div class="fixed">
This div element has position: fixed;
</div>

</body>
</html>

```

**position: fixed;**

This div element has position: fixed;

## Absolute:

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):

```

<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: absolute;</h2>

<p>An element with position: absolute; is positioned relative to the nearest positioned
ancestor (instead of positioned relative to the viewport, like fixed):</p>

<div class="relative">This div element has position: relative;
  <div class="absolute">This div element has position: absolute;</div>
</div>

</body>

```

**position: absolute;**

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):

This div element has position: relative;

This div element has position: absolute;

## Sticky:

An element with `position: sticky;` is positioned based on the user's scroll position.

```

<!DOCTYPE html>
<html>
<head>
<style>
div.sticky {
  position: -webkit-sticky;
  position: sticky;
  top: 0;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
</style>
</head>
<body>

<p>Try to <b>scroll</b> inside this frame to understand how sticky positioning works.</p>

<div class="sticky">I am sticky!</div>

<div style="padding-bottom:2000px">
<p>In this example, the sticky element sticks to the top of the page (top: 0), when
</div>

</body>
</html>

```

Try to **scroll** inside this frame to understand how sticky positioning works.

I am sticky!

In this example, the sticky element sticks to the top of the page (top: 0), when

## CSS Opacity / Transparency

The opacity property specifies the opacity/transparency of an element. The opacity property can take a value from 0.0 - 1.0. The lower the value, the more transparent:

```

img {
  opacity: 0.5;
}

```

## Combinatory:

A combinator is something that explains the relationship between the selectors. A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

### Descendant Selector

The descendant selector matches all elements that are descendants of a specified element. The following example selects all <p> elements inside <div> elements:

```

div p {
  background-color: yellow;
}

```

## Child Selector (>)

The child selector selects all elements that are the children of a specified element. The following example selects all <p> elements that are children of a <div> element:

```
div > p {  
  background-color: yellow;  
}
```

## Adjacent Sibling Selector (+)

The adjacent sibling selector is used to select an element that is directly after another specific element. Sibling elements must have the same parent element, and "adjacent" means "immediately following". The following example selects the first <p> element that are placed immediately after <div> elements:

```
div + p {  
  background-color: yellow;  
}
```

## General Sibling Selector (~)

The general sibling selector selects all elements that are next siblings of a specified element. The following example selects all <p> elements that are next siblings of <div> elements:

```
div ~ p {  
  background-color: yellow;  
}
```

Selector	Example	Example description
<u><i>element element</i></u>	div p	Selects all <p> elements inside <div> elements
<u><i>element&gt;element</i></u>	div > p	Selects all <p> elements where the parent is a <div> element
<u><i>element+element</i></u>	div + p	Selects the first <p> element that are placed immediately after <div> elements
<u><i>element1~element2</i></u>	p ~ ul	Selects every <ul> element that are preceded by a <p> element

# Lab Task

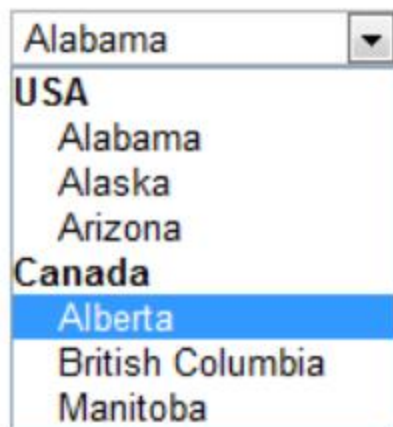
## Task-1

Create a Web Form in which all types of **INPUT TYPE** used [which we learn in this lab] using CSS to make it interactive.

## Task-2

Create a dropdown as like below; in which USA and Canada cannot be selected.





Alabama

USA

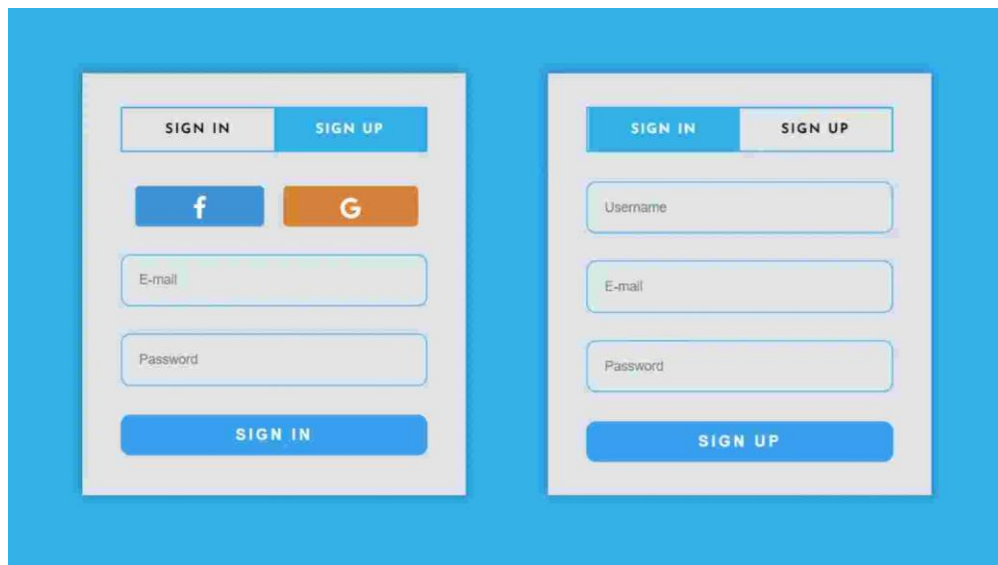
- Alabama
- Alaska
- Arizona

Canada

- Alberta
- British Columbia
- Manitoba

### Task-3

Create a Web Form in which Sign-in AND Sign-up shown as like below.



**Sign-in Form:**

- Buttons: SIGN IN, SIGN UP
- Social Media: f, G
- Input Fields: E-mail, Password
- Bottom Button: SIGN IN

**Sign-up Form:**

- Buttons: SIGN IN, SIGN UP
- Input Fields: Username, E-mail, Password
- Bottom Button: SIGN UP

### Task-4

Create a Web Form in which Contact US shown using the placeholder as like below.

## Connect With Us

We would love to respond to your queries and help you succeed.  
Feel free to get in touch with us.

### Send your request

Name

John Amendo

Phone

+1 412 520 3231

Email

johnamendogm@gmail.com

Subject

Product Demo

Message

Your Message

SEND

### Reach Us

Email

contactus@example.com

Phone

+1 012 345 6789

Address

#212, Ground floor, 7th cross  
Some layout, Some Road,  
Koromangla  
Bengaluru 560001

### Task-5

Create a Header with **NavBar** using all types of Combinatory which you learn in this Lab.

### Task-6

Using HTML FORM and CSS, Create the below cart;

My Cart

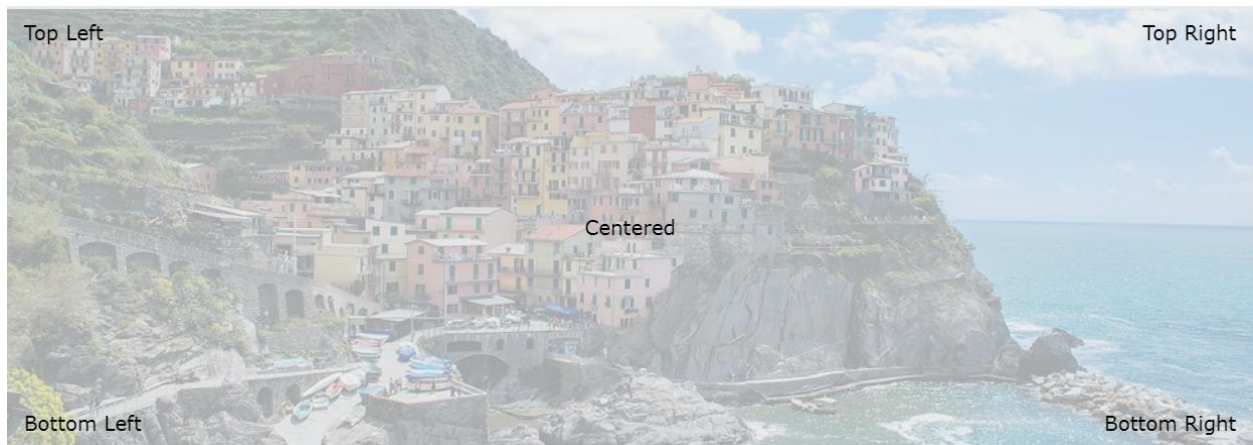
	product 1	\$10	<input type="text" value="2"/>	\$20	<span>X</span>
	product 3	\$30	<input type="text" value="6"/>	\$180	<span>X</span>
	product 4	\$40	<input type="text" value="1"/>	\$40	<span>X</span>
Total				\$240	
Total (including discount)				\$120	

Close

Checkout

### Task-7

Using position [all properties] places all text on the image according to the position right top, left top, bottom left, bottom right and on center, where image used opacity to look light.



### Task-8

Using CSS properties create the below box in which margin, padding, outlines and content will be used.

