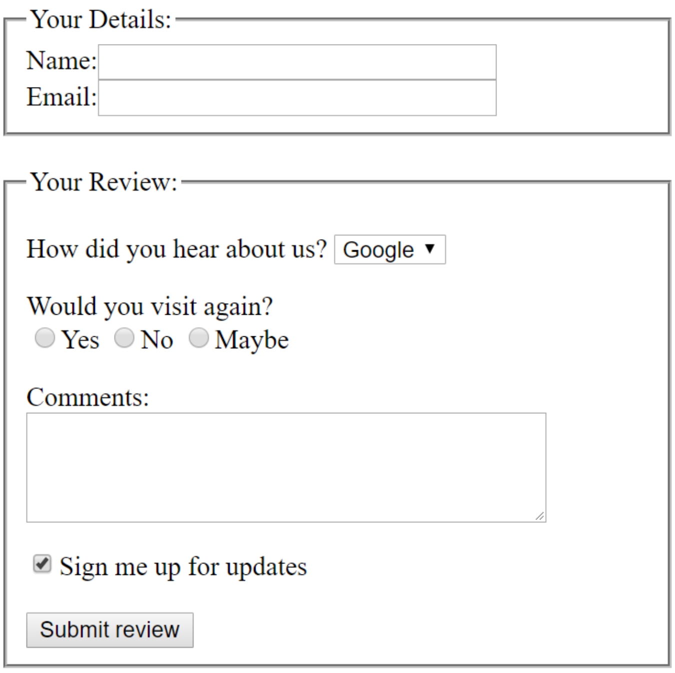
Temasek Junior College
2024 JC2 H2 Computing
Web Applications 3 - HTML Forms

Section	4	Computer Networks
Unit	4.2	Web Applications
Objectives	4.2.3	Use HTML, CSS (for clients) and Python (for the server) to create a web application that is able to: - accept user input (text and image file uploads) - process the input on the local server - store and retrieve data using an SQL database - display the output (as formatted text/images/table)

Are you able to create this form?



Your Details:

Name:

Email:

Your Review:

How did you hear about us? Google ▼

Would you visit again?
○ Yes ○ No ○ Maybe

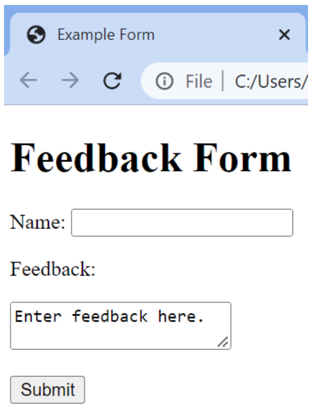
Comments:

[✓] Sign me up for updates

Submit review

1 How does a HTML Form Work?

An important advantage of webpages over conventional documents is that webpages can collect inputs from users via forms such as the one shown below.



Feedback Form

Name:

Feedback:

Enter feedback here.

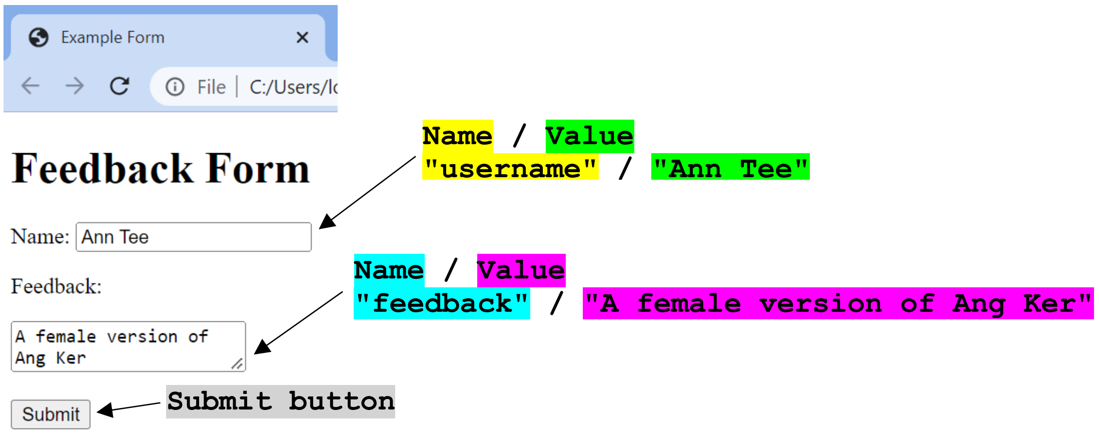
Such forms can be created in HTML using the **`<form>`…`</form>`**, **`<input>`**, **`<textarea>`…`</textarea>`** and **`<option>`…`</option>`** tags.
The HTML script of the above form is as follows:

```
<!doctype html>
<html>
  <head>
    <title>Example Form</title>
  </head>
  <body>
    <form action="https://www.example.com">
      <h1>Feedback Form</h1>
      <p>Name: <input name="username" type="text" value=""></p>
      <p>Feedback:</p>
      <textarea name="feedback">Enter feedback here.</textarea>
      <p><input type="submit"></p>
    </form>
  </body>
</html>
```

Before we dive into the creation of forms using HTML, it is important to first understand how does a HTML form work.

We shall use the HTML script above as an example.
(You may match the colors of shown below with the highlighted portions of the script above.)

1. A user fills in a form before submitting the data to the server.


Example Form

File | C:/Users/l

Feedback Form

Name: Ann Tee

Feedback:

A female version of Ang Ker

Submit

2. The name of each form control and the user-provided or user-selected value(s) are sent to the server when the **`Submit`** button is clicked.

   In the above example, the names and values are as follows.

Name	Value
`"username"`	**`"Ann Tee"`**
`"feedback"`	**`"A female version of Ang Ker"`**

3. The server then processes the information using some programming language. It may also store the information in a database.

4. The server (creates and) returns a (new) webpage to the user on a browser.

   In the above example, the server will return the webpage with URL **`https://www.example.com`** to the user to read on a browser.

   This is in accordance to the **`<form>`** start tag where the value of the **`action`** attribute is set to **`https://www.example.com`**.

```
<form action="https://www.example.com">
```

2 Sending a Form

Forms can be sent using one of the two methods: **`GET`** or **`POST`**.

2.1 The `GET` Method

The **`GET`** method is used to request data from a specified resource.

When the **`method`** attribute of the **`<form>`** start tag is not specified, it defaults to **`GET`**.

The **`GET`** method is ideal for:

- short forms used to retrieve or request for information, such as search boxes
- instances which you are just retrieving data from the server, not sending information that should be added to or deleted from a database.

With the **`GET`** method, the values collected by the form are added to the end of the URL specified in the **`action`** attribute of the **`<form>`** start tag. Hence the URL of the webpage returned to the user when the form is submitted will contain the values collected by the form.

Using the example from the previous section, the **`<form>`** start tag was written as

```
<form action="https://www.example.com">
```

Hence, the URL of the webpage returned after the form is submitted will be

```
https://www.example.com/?username=Ann+Tee&feedback=A+female+version+
of+Ang+Ker
```

2.2 The `POST` Method

To use the **`POST`** method, the value of the **`method`** attribute of the **`<form>`** start tag needs to be specified as **`"post"`**.

Using the example from the previous section, the **`<form>`** start tag should then be written as

```
<form action="https://www.example.com" method="post">
```

With the **`POST`** method, the values are sent in what are known as the hypertext transfer protocol (HTTP) headers in the HTTP requests. The anatomy of such requests shall be kept to a separate discussion on HTTP. It suffices here to know that with the **`POST`** method, the data collected by the form is stored in the request body of the HTTP request and will not be visible to users.

It follows that the URL of the webpage returned after the form is being submitted will not contain the values collected by the form and is thus the same as that specified in the action attribute of the **`<form>`** start tag.

Using the example from the previous section, the URL of the page returned to the browser after the form is submitted will be

```
https://www.example.com
```

As a rule of thumb, you should use the **`POST`** method if your form:

- allows users to upload a file.
- is very long.
- contains sensitive data, e.g. passwords.
- adds information to or deletes information from a database.

2.3 Comparing the `GET` and `POST` Methods

The following table gives a comparison between the **`GET`** and **`POST`** methods.

Process	**`GET`**	**`POST`**
Back / Reload	Harmless	Data will be resubmitted (the browser should alert the user that the data will be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data; use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the **`GET`** method adds the data to the URL and the total length of the URL is limited to 2048 characters	No restrictions
Restrictions on data type	Only ASCII characters allowed as the return URL which contains the data collected only accepts ASCII characters	No restrictions, binary data is also allowed
Security	**`GET`** is less secure compared to **`POST`** because data sent is part of the URL of the returned webpage Never use **`GET`** when sending passwords or other sensitive information	**`POST`** is a little safer than **`GET`** because the parameters are not stored in the browser history or in server logs
Visibility	Data is visible to everyone due to its display in the return URL	Data is not displayed in the return URL

3 Creating HTML Forms

3.1 The `<form>…</form>` Tags

Each form is contained in a separate set of **`<form>…</form>`** tags. This is accompanied with an **`action`** attribute in the **`<form>`** start tag that is set to the URL of the webpage that will be (created and) returned after the user submits the data.

In 9569 H2 Computing, the data will be processed by a server script written in Python upon submission. This shall be kept to a separate discussion.

Within the **`<form>…</form>`** tags, the following tags can be used:

- **`<input>`**,
- **`<textarea>…</textarea>`**
- **`<option>…</option>`**

Each **`<input>`**, **`<textarea>`** and **`<option>`** tag represents an **input control** and may have a unique name attribute to help the server retrieve these inputs.

In addition, other HTML tags may also be used where appropriate.

As discussed in the previous section, forms may be sent using the **`GET`** or the **`POST`** method. Hence we shall adopt the following generic structures when creating forms.

Structure using the `GET` Method

```
<!doctype html>
<html>
    <head>
        <title>Insert Title of Form Here</title>
    </head>
    <body>
        <form action="http://127.0.0.1:5000">
            Insert required form elements here with relevant tags
        </form>
    </body>
</html>
```

Structure using the `POST` Method

```
<!doctype html>
<html>
    <head>
        <title>Insert Title of Form Here</title>
    </head>
    <body>
        <form action="http://127.0.0.1:5000" method="post">
            Insert required form elements here with relevant tags
        </form>
    </body>
</html>
```

3.2 The `<input>` Tag

Depending on what the **`type`** attribute is set to, an **`<input>`** tag can be used to represent

- a single-line text box
- a radio button
- a checkbox
- a click button

For a text box, the **`value`** attribute determines the text box's initial contents.

For a radio button, checkbox and a click button, the **`value`** attribute is used as the label.

Note that the **`<input>`** tag does not have an end tag.

3.3 The `<textarea>`… `</textarea>` Tag

The **`<textarea>`**… **`</textarea>`** tags represents a multi-line text box.

The text contained within the **`<textarea>`**…**`</textarea>`** tags will be used as the text box's initial content.

3.4 The `<option>` Tag

The **`<option>`**…**`</option>`** tags define an option in a selection list.

While the **`<option>`** start tag can be used without any attributes, the **`value`** attribute is usually needed to indicate what is sent to the server upon form submission.

The **`<option>`** start tag also contains a **`selected`** attribute. When its value is not specified, the first option of the selection list will be shown when the page loads.

If the user does not select an option, then the first item will be sent to the server as the value.

Exercise 1
For each of the tags below, determine whether it is a tag for a normal or a void element.

Tag	**Description**	**Element Type**
`<form>`	User-submittable form	Normal / Void
`<input>`	Input control	Normal / Void
`<textarea>`	Text input area	Normal / Void
`<option>`	An option in a selection list	Normal / Void

3.5 Form Elements

This sub-section will discuss the different elements that can be included in a form. They include:

- Single-Line Text Input <input type="text">
- Password Input <input type="password">
- Radio Buttons <input type="radio">
- Checkboxes <input type="checkbox">
- File Upload <input type="file">
- Submit Button <input type="submit">
- Image Button <input type="image">
- Date Input <input type="date">
- Dropdown List <select>…</select>
- Multiple Select Box <select>…</select>

3.5.1 Single-Line Text Input

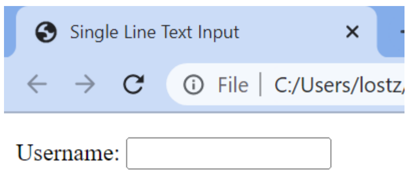
The **`<input>`** tag can be used to create a single-line text box for input. A possible version of the HTML script is shown below.

```
<!doctype html>
<html>
    <head>
        <title>Single Line Text Input</title>
    </head>
    <body>
        <form action="http://127.0.0.1:5000">
            <p>Username:<input type="text" name="username"
            size="15" maxlength="30"></p>
        </form>
    </body>
</html>
```

In the above version, we see that four attributes have been included with the **`<input>`** tag. Here's what each of them means.

`type`	specifies the type of the input to be collected (in this case text input)
`name`	specifies the name used to identify the input collected (in this case `"username"`)
`size`	the width of the element being displayed (in this case a length equivalent to 15 characters visible at any one time)
`maxlength`	the maximum number of characters that can be collected (in this case 30 characters)

The script will generate the following:



Username:

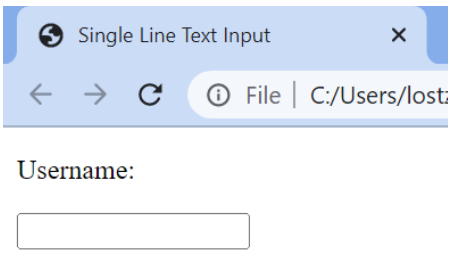
Notice that the string **`Username:`** is displayed in the same line as the input field. This is because the **`<input>`** tag and the string **`Username:`** are contained within the same paragraph tags **`<p>`**…**`</p>`** as follows.

```
<form action="http://127.0.0.1:5000">
     <p>Username:<input type="text" name="username" size="15"
     maxlength="30"></p>
</form>
```

By making a minor change to the script, we can have the string **`Username:`** to appear on top of the input field. This is achieved by placing the **`<input>`** tag outside of the paragraph tags **`<p>`**…**`</p>`** as follows.

```
<form action="http://127.0.0.1:5000">
     <p>Username:</p>
     <input type="text" name="username" size="15" maxlength="30">
</form>
```

The resulting output will be as such:



Username:

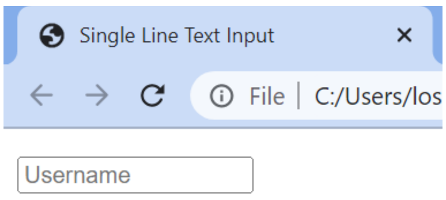
Sometimes, we might not want to label the input box. Instead a placeholder line contained within the input box that can be overwritten might be desired.

To do so, we can include in the **`<input>`** tag the **`placeholder`** attribute and set its value to the desired text string to be used as the placeholder.

```
<form action="http://127.0.0.1:5000">
     <p><input type="text" name="username" size="15" maxlength="30"
     placeholder="Username"></p>
</form>
```

Notice that the text string **`Username:`** has been removed from the script. Instead, the text string **`Username`** is now set as the value of the **`placeholder`** attribute of the **`<input>`** tag.

The resulting output will be as such:



Notice that all the above three versions use the **`GET`** method. Depending on the type of data being submitted, the **`POST`** method can also be used.

For example, a form created to collect search items for a search engine can be created using the **`GET`** method whereas a form created to collect sensitive data such as credit card number should preferably be created using the **`POST`** method.

3.5.2 Password Input

We have just seen how a single-line text input field can be created. For such an input field, the data entered is visibly displayed on the screen.

What happens if we do not want the data entered to be viewed on the screen?

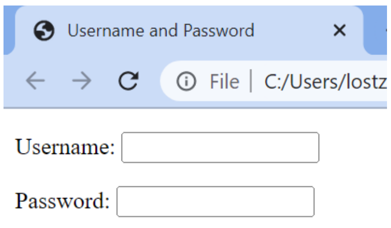
We will then need to set the **`type`** attribute of the **`<input>`** tag to **`password`** instead.

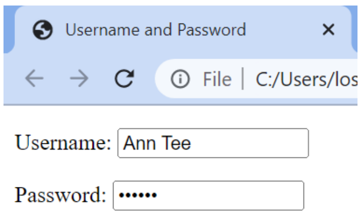
A possible script for the **`<input>`** tag used to create a password input field is as follows.

```
<input type="password" name="username" size="15" maxlength="30">
```

Exercise 2
Write HTML script to create a form that collects the username and password
(a) with labels for each field (placeholders are not required)
(b) without labels but with placeholders for each field
The data need not be submitted at this juncture.

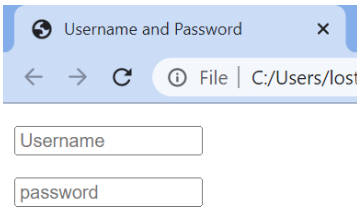
Sample Output for (a)

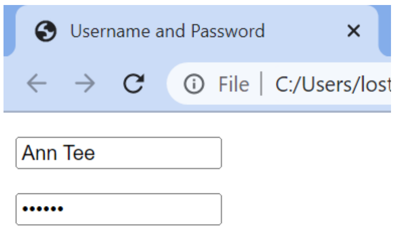

Username and Password
File | C:/Users/lostz
Username:
Password:


Username and Password
File | C:/Users/los
Username: Ann Tee
Password: ••••••

without data entered with data entered

Sample Output for (b)


Username and Password
File | C:/Users/lost
Username
password


Username and Password
File | C:/Users/lost
Ann Tee
••••••

without data entered with data entered

Notice that the **`POST`** method should be used in **Exercise 2**.

This is to ensure that the password collected will not be displayed as part of the URL for the webpage returned after the form is submitted.

Although the password is not visibly displayed on the screen, this does not mean that the password is sent securely to the server.

For full security, the server needs to be set up to communicate with the users' browser using Secure Sockets Layer (SSL). This shall be left to a separate discussion in network security.

3.5.3 Radio Buttons

Radio buttons are usually used for a set of related options where only one option can be selected out of the available options.

Once an option has been selected, it cannot be deselected. The user can only change the selection to another available option.

To create radio buttons, the **`type`** attribute of the **`<input>`** tag should be set to the value **`"radio"`**. A possible version of the HTML script is shown below:

```
<!doctype html>
<html>
    <head><title>Radio Options</title></head>
    <body>
        <form action="http://127.0.0.1:5000" method="post">
            <p>Select your favourite sport:</p>
            <input type="radio" name="sport" value="football"
            checked="checked">Football<
            <input type="radio" name="sport"
            value="basketball">Basketball
            <input type="radio" name="sport"
            value="badminton">Badminton
        </form>
    </body>
</html>
```

Let us take a closer look at the **`<input>`** tags in the script.

```
<input type="radio" name="sport" value="football"
checked="checked">Football
<input type="radio" name="sport" value="basketball">Basketball
<input type="radio" name="sport" value="badminton">Badminton
```

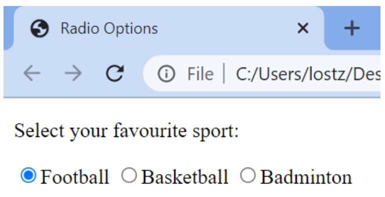
Notice that all three **`<input>`** tags share the same value for the **`name`** attribute: **`"sport"`**.

As only one option can be selected, the data assigned to the **`value`** attribute of the eventual option selected will then be identified by the name **`"sport"`** when the form is submitted.

Having the same **`name`** attribute also allows the radio buttons to be “grouped” as a set of related options. The ensures that only one of the available options can be chosen.

Note that the label for each option i.e. the text strings **`Football`**, **`Basketball`** and **`Badminton`** are placed at the end of their respective **`<input>`** tag. The label for each radio button will hence appear on the right of the button. For the label to appear on the left of the radio button, it should be placed before the **`<input>`** tag.

When the form is generated, it will look as follows:



Select your favourite sport:

● Football ○ Basketball ○ Badminton

Observe that the option **`Football`** is selected by default. This is because the **`checked`** attribute has been set to the value **`"checked"`** in the **`<input>`** tag created for **`Football`**. When no options are made, **`Football`** will be the default option sent to the server.

If it is desired not to have a default option to begin with, the **`checked`** attribute need not be set for all the options.

Observe also that the options appear in the same line. To have each option on a separate line, place each **`<input>`** tag within a set of paragraph tags **`<p>`**…**`</p>`**. Alternatively, use the **`<br>`** tag.

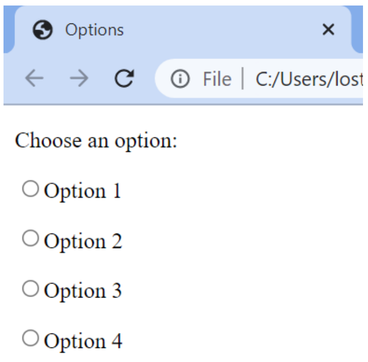
In the above script, the form is created using the **`POST`** method. Depending on the requirements, the form can also be created using the **`GET`** method.

Exercise 3

Write HTML script to create a form that provides four options on separate lines. The options are **`Option 1`**, **`Option 2`**, **`Option 3`** and **`Option 4`**.

A default option need not be selected and the data need not be submitted at this juncture.

Sample Output



Choose an option:

○ Option 1

○ Option 2

○ Option 3

○ Option 4

3.5.4 Checkboxes

Checkboxes are usually used for a set of related options where more than one option can be selected out of the available options.

Checkboxes also offer the flexibility to deselect an option after it has been selected. This would also mean that there can be zero options selected (if the form is designed to allow so).

To create checkboxes, the **`type`** attribute of the **`<input>`** tag should be set to the value **`"checkbox"`**. A possible version of the HTML script is shown below:

```
<!doctype html>
<html>
    <head><title>Checkbox Options</title></head>
    <body>
        <form action="http://127.0.0.1:5000" method="post">
            <p>Select the sports that you know how to play:</p>
            <input type="checkbox" name="sport"
            value="football">Football
            <input type="checkbox" name="sport"
            value="basketball">Basketball
            <input type="checkbox" name="sport"
            value="badminton">Badminton
        </form>
    </body>
</html>
```

Let us take a closer look at the **`<input>`** tags in the script.

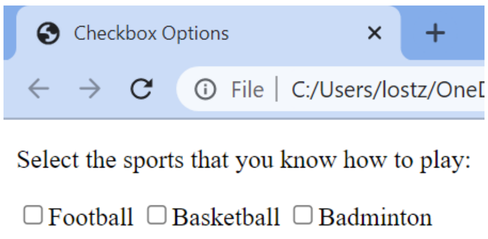
```
<input type="checkbox" name="sport" value="football">Football
<input type="checkbox" name="sport" value="basketball">Basketball
<input type="checkbox" name="sport" value="badminton">Badminton
```

Notice that all three **`<input>`** tags share the same value for the **`name`** attribute: **`"sport"`**.

This allows for the values of all the options selected to be sent to the server as an array with the name **`"sport"`** when the form is submitted.

As with radio buttons, the positioning of each checkbox label can be adjusted to the front or back of the checkbox by placing the label either before or after the **`<input>`** tag.

When the form is generated, it will look as follows:



Select the sports that you know how to play:

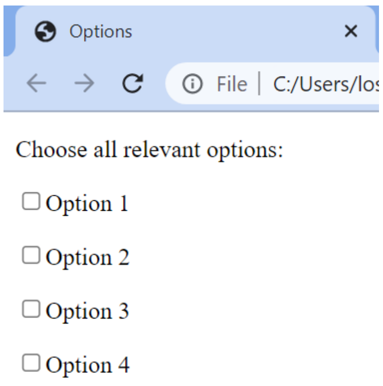
☐ Football ☐ Basketball ☐ Badminton

Observe also that the options appear in the same line. To have each option on a separate line, place each **`<input>`** tag within a set of paragraph tags **`<p>`**…**`</p>`**. Alternatively, use the **`<br>`** tag.

In the above script, the form is created using the **`POST`** method. Depending on the requirements, the form can also be created using the **`GET`** method.

Exercise 4
Repeat **Exercise 3** using checkboxes for the options instead.

Sample Output



Choose all relevant options:

☐ Option 1

☐ Option 2

☐ Option 3

☐ Option 4

3.5.5 File Upload

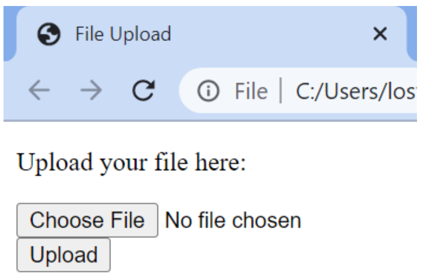
To upload files, we must use the **POST** method. We cannot send files using the **GET** method. Why?

This type of input may appear differently on different browsers: some creates a box that looks like a text input box with a **"Browse"** button. On the Chrome and Safari browsers, a **"Choose File"** button with a text **"No file chosen"** or the name of the file chosen by the user.

To create a file upload feature, the **type** attribute of the **<input>** tag should be set to the value **"file"**. A possible version of the HTML script is shown below:

```
<!doctype html>
<html>
    <head><title>File Upload</title></head>
    <body>
        <form action="http://127.0.0.1:5000" method="post">
            <p>Upload your file here:</p>
            <input type="file" name="userfile">
            <br>
            <input type="submit" value="Upload">
        </form>
    </body>
</html>
```

When the form is generated, it will look as follows:



Upload your file here:

No file chosen

Observe that there are two buttons in the output generated. The first button **"Choose File"** is the file upload feature created by the tag:

```
<input type="file" name="userfile">
```

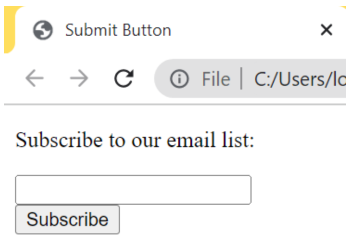
The second button **"Upload"** is a submit button. Upon clicking the **"Upload"** button, the selected file will be sent to the server. More details on the submit button will be discussed in the next sub-section.

3.5.6 Submit Button

A submit button is a clickable button used to send form data to a web server for processing. When a user clicks on the submit button, the form data is sent to the server using either the **`GET`** or **`POST`** method.

To create a submit button, the **`type`** attribute of the **`<input>`** tag should be set to the value **`"submit"`**. A possible version of the HTML script is shown below:

```
<!doctype html>
<html>
    <head><title>Submit Button</title></head>
    <body>
        <form action="http://127.0.0.1:5000" method="post">
            <p>Subscribe to our email list:</p>
            <input type="text" name="email">
            <br>
            <input type="submit" name="subscribe"
            value="Subscribe">
        </form>
    </body>
</html>
```



Subscribe to our email list:

Notice that in the previous sub-section, the label of the submit button was **`"Upload"`**. In the above example, the label of the submit button has been changed to **`"Subscribe"`**. The label is controlled using the **`value`** attribute, which is used to specify the words that appear on the submit button. When the value is not specified, the default value of the button is **`"Submit"`** on some browsers, including the Chrome browser.

3.5.7 Image Button

Instead of a conventional submit button, the submit button can be created using an image. This results in an image button.

To create an image button, the **`type`** attribute of the **`<input>`** tag should be set to the value **`"image"`**. In addition the source **`"src"`** attribute must be set to the path of the image that is to be used.

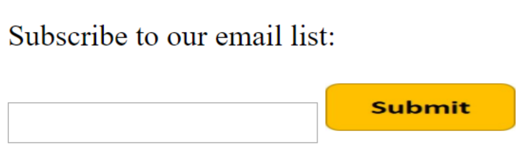
A possible version of the HTML script is shown below:

```
<!doctype html>
<html>
    <head><title>Image Button</title></head>
    <body>
        <form action="http://127.0.0.1:5000" method="post">
            <p>Subscribe to our email list:</p>
            <input type="text" name="email">
            <input type="image" src="image/submit.jpg"
            width="100" height="25" alt="Submit form">
        </form>
    </body>
</html>
```

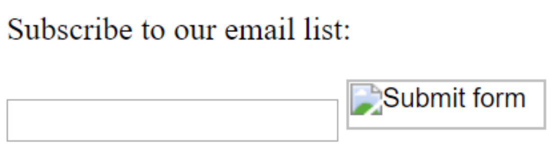
Notice from the script that there is a **`width`** attribute, a **`height`** attribute and an **`alt`** attribute.

The **`width`** and **`height`** attribute allows resizing of the image to create an appropriately sized button.

The **`alt`** attribute allows a text to be specified and displayed when the image used for the button cannot be loaded.



Subscribe to our email list:



Subscribe to our email list:

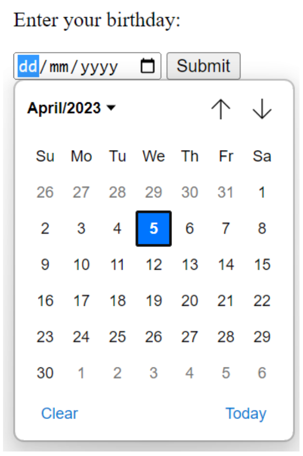
3.5.8 Date Input

A date input allows for typewritten dates to be entered. In addition, some browsers will also provide a calendar to select the required date.

To create a date input, the **`type`** attribute of the **`<input>`** tag should be set to the value **`"date"`**. A possible version of the HTML script is shown below:

```
<!doctype html>
<html>
    <head><title>Date Entry</title></head>
    <body>
        <form action="http://127.0.0.1:5000" method="post">
            <p>Enter your birthday:</p>
            <input type="date" name="birthday">
            <input type="submit" value="submit">
        </form>
    </body>
</html>
```

When the form is generated, it will look as follows:


Enter your birthday:

dd/mm/yyyy Submit

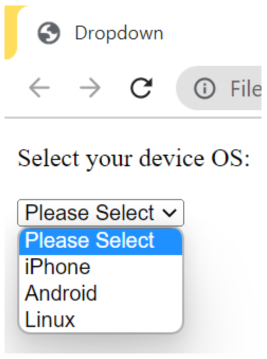
3.5.9 Dropdown List

A dropdown list allows you to select an option from a list of options

To create a date input, the **`type`** attribute of the **`<input>`** tag should be set to the value **`"date"`**. A possible version of the HTML script is shown below:

```
<!doctype html>
<html>
    <head><title>Dropdown</title></head>
    <body>
        <form action="http://127.0.0.1:5000" method="post">
            <p>Select your device OS:</p>
            <select name="devices">
                <option value="" selected="selected">Please
                Select</option>
                <option value="Windows">iPhone</option>
                <option value="macOS">Android</option>
                <option value="Linux">Linux</option>
            </select>
        </form>
    </body>
</html>
```

When the form is generated, it will look as follows:



Select your device OS:

Please Select
Please Select
iPhone
Android
Linux

Observe that the **`"Please select"`** label in the dropdown list is created using the following tag:

```
<option value="" selected="selected">Please Select</option>
```

This is not an actual option. As such the **`value`** attribute is set to "".

Notice that a **`selected`** attribute with value set to **`selected`** is used. If this is not done, the first option (**`iphone`** in this case) will be shown when the page loads.

When using a dropdown list, if no option is selected, the first item in the dropdown list will be sent to the server when the form is submitted.

The dropdown list has the same functionality as a set of radio buttons. The difference being a set of radio buttons allow users to see all the available options at a glance as compared to a dropdown list which may sometimes require scrolling (especially when the list is long).

3.5.10 Multiple Selection Box

A multiple selection box allows for the selection of more than one option. It is created using a **`<select>`**…**`</select>`** tag by setting the value of the **`multiple`** attribute to **`multiple`**.

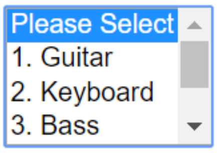
A **`size`** attribute is used to specify the number of options to display at once.

When creating a multiple selection box, it is a good practice to tell the user they can use the control/command key on the PC/Mac while selecting the options.

A possible version of the HTML script is shown below:

```
<!doctype html>
<html>
    <head><title>Multiple Selection</title></head>
    <body>
        <form action="http://127.0.0.1:5000" method="post">
            <p>Select the musical instruments which you know
            <br>
            (You can select more than one option by holding down
            control on a PC or command key on a Mac while selecting
            different options)</p>
            <select name="instruments" size="4"
            multiple="multiple">
                <option value="" selected="selected">Please
                Select</option>
                <option value="Guitar">Guitar</option>
                <option value="Keyboard">Keyboard</option>
                <option value="Bass">Bass</option>
                <option value="Piano">Piano</option>
                <option value="Flute">Bass</option>
            </select>
        </form>
    </body>
</html>
```

When the form is generated, it will look as follows:



Notice that the size of 4 includes the dummy option **`"Please select"`**.

3.5.11 Large Text Input

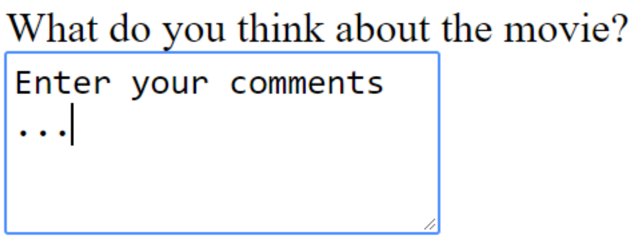
What happens when you enter multiple lines of text?

The **`<textarea>`**…**`</textarea>`** tag can be used to do so.

A possible version of the HTML script is shown below:

```
<!doctype html>
<html>
    <head><title>Multiple Selection</title></head>
    <body>
        <form action="http://127.0.0.1:5000" method="post">
            <p>What do you think about the movie?</p>
            <textarea name="comments" cols="20" rows="4">Enter
            your comments:</textarea>
        </form>
    </body>
</html>
```

A text input box of 20 characters in width and 4 rows in height is generated.



What do you think about the movie?

```
Enter your comments
...
```

3.6 Grouping Elements

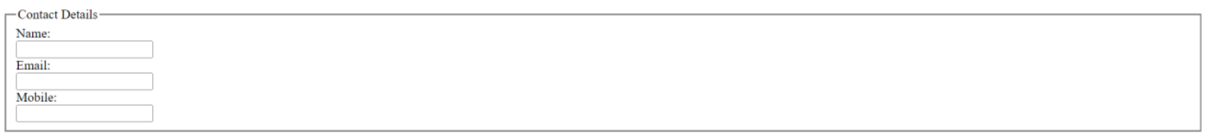
Different form elements can be grouped together for easy classification using the **`<fieldset>`**…**`</fieldset>`** tag.

The **`<legend>`**…**`</legend>`** tag can be used to include a header for the grouped elements.

A possible version of the HTML script is shown below:

```
<!doctype html>
<html>
    <head><title>Grouped Elements</title></head>
    <body>
        <form action="http://127.0.0.1:5000" method="post">
            <fieldset>
                <legend>Contact Details</legend>
                Name:<br>
                <input type="text" name="name"><br>
                Email:<br>
                <input type="text" name="email"><br>
                Mobile:<br>
                <input type="text" name="mobile">
            </fieldset>
        </form>
    </body>
</html>
```

The following form will be generated:


Contact Details
Name:
Email:
Mobile:

<u>Exercise 5</u>
Create the form shown in the first page of this set of notes.