**Forms (Input)**

Whenever we input information that will then be sent somewhere else, we are inputting into a form.

1. **How a Form Works**

Computers need an *HTTP request* to be instructed to receive and handle the incoming information.

The <form> element is a great tool for collecting information, but we need need to send that information somehwere else for processing. We need to supply the <form> element with both:

* The location of where the <form>’s information goes, using the action attribute.
* What HTTP request to make, using the method attribute and assign it a HTTP verb (request). As a convention, the assigned verb should be in CAPLOCKS.
* The <form> element can also contain child elements, including a header or a paragraph for details.

<form action="/example.html" method="POST">  
  <h1>Creating a form</h1>  
  <p>Looks like you want to learn how to create an HTML form. Well, the best way to learn is to play around with it.</p>  
</form>

1. **Text Input Field**

If we need to create an input field in our <form>. We do this using <input> element (self closing)

* The <input> element has a type attribute which determines how it renders on the web page and what kind of data it can accept. There are different values for the type attribute.

type=“text” 🡪 Renders a text field that user can type into. (also default value)

* We also **have to include a name attribute** for the <input>. Without the name attribute, information in the <input> won’t be sent when the <form> is submitted.
* A value attribute, which can be set to a default value (pre-filled text field when rendered)

A picture containing table

Description automatically generated<form action="/example.html" method="POST">  
  <input type="text" name="first-text-field"> 🡪   
</form>

After users type into the <input> element, the value of the value attribute becomes what is typed into the text field. The value of the value attribute **is paired with** the value of the name attribute and **sent as text** when the form is submitted **to action**.

Table

Description automatically generated🡪 value = “important details”

“first-text-field” = value

1. **Adding a label**

Previously, we created an <input> element but we didn’t include anything to explain to the users what the <input> is used for. As a result, we need to add a <label> element.

The <label> element has an opening and closing tag and displays text that is written between.

To associate a <label> and an <input>, the <input> needs an id attribute. We then assign the for attribute of the <label> element with the value of the id attribute of <input>.

Table

Description automatically generated<form action="/example.html" method="POST">  
  <label for="meal">What do you want to eat?</label>  
  <br> 🡪  
  <input type="text" name="food" id="meal">  
</form>

Another benefit for using the <label> element is when this element is clicked, the corresponding <input> is highlighted/selected.

1. **Password Input**

For this input field, we can use the type=”password” attribute. Using this, the input text will be replaced with (\*) or (.).

Rectangle

Description automatically generated with low confidence<form>  
  <label for="user-password">Password: </label>  
  <input type="password" id="user-password" name="user-password">  
</form>

1. **Number Input**

Sometimes we want to get user input as a number, in which case we will use the type=”number”

By setting type="number" for an <input> we can restrict what users type into the input field to just numbers (and a few special characters like -, +, and .).

We can also provide a step attribute which creates arrows inside the input field that user can increase or decrease the input by the value of the step.

Graphical user interface, application

Description automatically generated with medium confidence<form>  
  <label for="years"> Years of experience: </label>  
  <input id="years" name="years" type="number" step="1">  
</form>

1. **Range Input**

We can limit what numbers our users could type in using a different type value. We could use **“range”**, which **creates a slider**.

To set the minimum and maximum values of the slider we assign values to the min and max attribute of the <input>.

Text

Description automatically generatedWe could also control how smooth and fluid the slider works by assigning the step attribute a value. Smaller step values will make the slider more fluidly, whereas larger step values will make the slider move more noticeably.

<form>  
  <label for=“volume”> Volume Control</label>  
  <input id=“volume” name=“volume” type=“range” min=“0” max=“100” step=“1”>  
</form>

1. **Checkbox Input (Multiple options)**

In a <form> we can use the <input> element for each choice and set type=“checkbox”.

Graphical user interface, text, application

Description automatically generated

<form>  
  <p>Choose your pizza toppings:</p>  
  <label for=“cheese”>Extra cheese</label>  
  <input id=“cheese” name=“topping” type=“checkbox” value=“cheese”>  
  <br>  
  <label for=“pepperoni”>Pepperoni</label>  
  <input id=“pepperoni” name=“topping” type=“checkbox” value=“pepperoni”>  
  <br>  
  <label for=“anchovy”>Anchovy</label>  
  <input id=“anchovy” name=“topping” type=“checkbox” value=“anchovy”>  
</form>

NOTE:

* There are assigned values to the value attribute of the checkboxes. These values are not visible on the form itself, that’s why it is important that we use an associated <label> to identify the checkbox (value, if chosen, will be assigned to name)
* Each <input> has the same value for the name attribute. Using the same name for each checkbox groups the <input>s together. However, each <input> has a unique id to pair with a <label>.

1. **Radio Button Input (Only 1 selection)**

To create radio buttons, we instead use the type=“radio”. Other attributes including name, type, and value is similar to Checkbox.

Table

Description automatically generated with low confidence<form>  
  <p>What is sum of 1 + 1?</p>  
  <input type=“radio” id=“two” name=“answer” value=“2”>  
  <label for=“two”>2</label>  
  <br>  
  <input type=“radio” id=“eleven” name=“answer” value=“11”>  
  <label for=“eleven”>11</label>  
</form>

NOTE: Notice from the code, radio button values (like checkboxes) do not display their value. We have an associated <label> to represent the representation of the radio button. To group radio buttons together, we assign them the same name and only one radio button from that group can be selected.

1. **Dropdown List**

If we have many different choices, and only want 1 input, it would be hard to name all the labels and inputs using radio button. Therefore, we can instead use the dropdown list.

Graphical user interface, text, application, chat or text message

Description automatically generated<form>  
  <label for=“lunch”>What's for lunch?</label>  
  <select id=“lunch” name=“lunch”>  
    <option value=“pizza”>Pizza</option>  
    <option value=“curry”>Curry</option>  
    <option value=“salad”>Salad</option>  
    <option value=“ramen”>Ramen</option>  
    <option value=“tacos”>Tacos</option>  
  </select>  
</form>

NOTE: The text rendered is the text between the opening and closing <option> tags. However, it is the value of the value attribute that is used in <form> submission.

When the <form> is submitted, the information from this input field will be sent using the name of the <select> and the value of the chosen <option>. For instance, if a user selected Pizza from the dropdown list, the information would be sent as ”lunch=pizza”.

1. **Datalist Input**

Even if we have an organized dropdown list, if the list has a lot of options, it could be tedious for users to scroll through the entire list to locate one option. That’s where we use the <datalist> element.

The <datalist> is used with an <input type=“text”> element. The <input> creates a text field that users can type into and filter options from the <datalist>.

Graphical user interface, application

Description automatically generated<form>  
  <label for=“city”>Ideal city to visit?</label>  
  <input type=“text” list=“cities” id=“city” name=“city”>  
  
  <datalist id=“cities”>  
    <option value=“New York City”></option>  
    <option value=“Tokyo”></option>  
    <option value=“Barcelona”></option>  
    <option value=“Mexico City”></option>  
    <option value=“Melbourne”></option>  
    <option value=“Other”></option>    
  </datalist>  
</form>

NOTE: The <input> is associated to the <datalist> via the <input>‘s list attribute and the id of the <datalist>. (id of input is linked to for of label)

If none of the <option>s match, the user can still use what they typed in. When the form is submitted, the <input>‘s name and the value of the option selected, is sent as a pair.

1. **Textarea element**

An <input> element with type=“text” creates a single row input field for users to type in information. However, there are cases where users need to write in more. In such cases, we could use <textarea>.

The <textarea> element is used to create a bigger text field for users to write more text. We can add the attributes rows and cols to determine the amount of rows and columns for the <textarea>.

Text

Description automatically generated with medium confidence<form>  
  <label for=“blog”>New Blog Post: </label>  
  <br>  
  <textarea id=“blog” name=“blog” rows=“5” cols=“30”>  
  </textarea>  
</form>

We can also add optional default text between the textarea tags.

1. **Submit Form**

After adding all elements into the <form>, it’s time to submit. That’s the role of the submit button.

To make a submit button in a <form>, we’re going to use the <input> element and set type=“submit”.

We can add the attribute value and assign to it any text that shows up on the button. By default, the button will show “Submit”

A picture containing graphical user interface

Description automatically generated<form>  
  <input type=“submit” value=“Send”>  
</form>

**\*\*Include after submission, link to another internal pages.**