# 13.1 Inputs

Having begun our journey into the world of forms, in this lesson you'll have a chance to get more familiar with the other commonly used form controls. We've split them into two groups:

- **Keyboard-driven inputs:** inputs requiring user interaction via the keyboard. This includes text, password, email, numbers, etc.
- **Mouse-driven inputs:** inputs controlled through the mouse. Examples include checkboxes, file uploads, submission buttons, etc.

At the end of the lesson, we'll also discuss how you can label and group all these elements to create even better code. You'll be able to test them out by crafting an intricate questionnaire all by yourself!

Key driven input

<form>
<input type="text" > ---- It takes input in one line
<textarea> </textarea> ------ IT can be maximize and multiple line inputs

<input type="date " > ---- to take input in form of date

<input type="url " >

<input type="number">
<input type="search">
<input type="tel " pattern="[0-7]{3} > ----- to enter telephone

<form>

# 13.3 Keyboard-driven Inputs

Keyboard-driven inputs allow the user to use keyboard keys to type in their answer. In the previous lesson, you tried out the text, password, and email input types. Now, let's go a bit further.

<textarea>
This element creates a text area, a multiple-line text input box. Unlike the others, `<textarea>` **is not an empty element**: it must have both an opening and a closing tag. The text added to the tag's contents will show up in the textbox as the initial value.

In the tag, you can specify how wide you want the text area to be (`cols`, which is measured by the number of characters) and how many vertical `rows` it should take up.

```
<textarea name="story"
          rows="4">
Tell us about your most embarrassing moment...
</textarea>
```

TRY IT OUT

Task: Go to the code editor via the *Try it Out* button above and try setting the `cols` attribute to 10, 40, and 500. Notice the difference?

**Note: While cols and rows do the job, it's recommended to actually use CSS for adjusting the width and height of a <textarea> box. That way, you'll make sure to have consistently-sized boxes in your whole website. You'll learn how to do it in the sprint to come.*

### Keyboard-driven input values

Before HTML5, forms for collecting dates of birth, email addresses, or URLs used simple text inputs. Now, new input types have been introduced to standardize the processing of certain data.

If you're collecting **dates**, you would use the date value. Browsers supporting HTML5 will automatically only allow inputs in a date format.

```
Departure date: <input type="date" name="birthday" >
```

TRY IT OUT

**URLs** work similarly. If you are asking for a website address, HTML5-supported browsers will check the user's input and confirm whether it's in the format of a URL.

```
<p>Please share a funny video:</p>
<input type="url" name="funnyvideo" >
```

The **number** value assures that all inserted inputs are numeric. You can also specify its range with the `min` and `max` attributes.

```
<input type="number" name="computerskills" min=”0” max=”45”>
```

TRY IT OUT

If you're asking visitors for a **contact phone number**, you can control their input through the tel value. It has a specific attribute to control telephone pattern-specific validation — you'll learn it in the following lesson.

```
<input type="tel" name="phone" >
```

TRY IT OUT

For search queries, you can use the search value. The newest web browsers also include some quirks in regards to this one. For example, Safari automatically adds a cross to the end of the box once the user starts typing, making it easier to clear the input.

```
<p>Search:</p>
<input type="search" name="search" >
<input type="submit" value="Search" >
```

TRY IT OUT

## Placeholder

For text inputs, you can also add a `placeholder` attribute. Its value will be displayed in the text box until the visitor clicks on the textbox. Here's what it would look like when added to the search bar:

```
<p>Search:</p>
<input type="search" name="search" placeholder=”What are you looking for today?”>
<input type="submit" value="Search" >
```

TRY IT OUT

**Mouse driven inputs:**

```
<form action =“link” method=“GET”>
   <input type=“radio” name = “Boolean”> TRUE
  <input type=“radio” name=“Boolean”> FALSE


</form>


<select name=“cars”   multiple>    ====== if not multiple then its good to select one
              <option> Volvo </option>
              <option> Audi</option>
              <option> BMW</option>
</select>

<select>
     <input type=“file”>
     <button> Simple Button </button>
```

# 13.5 Mouse-driven Inputs

Mouse-driven inputs are all those you can fill in with a simple click (or two) of the mouse. You've already added a submit button. Below, you'll find a more expansive list of input values you can use to jazz up the questionnaire for this lesson.

Radio button

One of the most common mouse-driven inputs, the radio value allows visitors to choose a single choice out of all the options provided.

```
<p>Select your favorite ice-cream flavour:
    <br>
    <input type="radio" name="icecream" value="Vanilla"> Vanilla
<input type="radio" name="icecream" value="choco" >
Chocolate
    <input type="radio" name="icecream" value="strawberry" >
Strawberry
</p>
```

TRY IT OUT

*Note: Once a user presses on a radio button, it cannot be deselected. The user can then only change their option. If you want to allow for deselection, choose a checkbox input instead.*

Checkbox

Checkboxes give the user the option to both select and deselect multiple options.

```
<p>Please select all countries you've visited:
    <br>
    <input type="checkbox" name="countries"
value="Italy" > Italy
<input type="checkbox" name="countries"
value="China"> China
<input type="checkbox" name="countries"
value="Puerto Rico" > Puerto Rico
</p>
```

TRY IT OUT

## Drop-downs

The **drop-down** option allows visitors to select one of the options provided in a list. Unlike most other inputs we've seen, this one uses two different tags: `<select>` and `<option>`.

The `<select>` element creates a drop-down list. Within it, you'll put multiple `<option>` elements to specify the user's options. Both `<select>` and `<option>` have content in it and thus require both an opening and a closing tag.

```
<p>How do you watch movies?</p>
<select name="devices">
 <option value="streaming">Streaming service</option>
 <option value="TV">On TV</option>
 <option value="cinema">At the cinema</option>
</select>
```

TRY IT OUT

**Note: Radio buttons are better suited for inputs with few options that should be easily seen at a glance. If you have a lot of options, e.g. a full list of countries, use the drop-down instead.*

## Multiple Select

If you want users to be able to select multiple choices from the drop-down list, you can add a `multiple` attribute in the `<select>` tag.

Task: Go to our code editor via the *Try it Out* button above and change our last code snippet so that it allows for multiple selection. Remember that multiple is a boolean data type, i.e. does not require a value. Does it work?

## File input

Are you asking users to upload files? Add a file input box to make that possible!

```
<p>Upload your code example:</p>
<input type="file" name="code-example" >
<br>
<input type="submit" value="Upload" >
```

TRY IT OUT

## Image as a button

Remember how you can add an image to show up as a submit button, instead of leaving its styling to the browser or adjusting it via CSS? You can use the `image` value to do so.

## Image as a button

Remember how you can add an image to show up as a submit button, instead of leaving its styling to the browser or adjusting it via CSS? You can use the `image` value to do so.

```
<p>Add any comments:</p>
<input type="text" name="comments" >
<input type="image" src="images/addcomment.jpg"
    width="100" height="20" alt=”comment picture”>
```

TRY IT OUT

**Note: You'll understand more about the `src`, `width`, and `height` attributes once you reach the HTML Multimedia section.*

## Button

What if you want to combine both text AND an image? Well, that's also possible with the help of the `<button>` element.

```
<button><img src="images/plussign.png" alt="add"
width="10px" height="10px" > Add</button>
```

## Button

What if you want to combine both text AND an image? Well, that's also possible with the help of the `<button>` element.

```
<button><img src="images/plussign.png" alt="add"
width="10px" height="10px" > Add</button>
```

TRY IT OUT

## Reset

If you want the user to be able to reset all form values to its initial state, add a **reset button**. This is particularly useful when your form has radio buttons, which don't allow for deselection. The code for it is pretty simple. You don't even need a name within the tag since it doesn't register anything with the server:

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

TRY IT OUT

**Note: The reset button must be inside a `<form>` element to work.*

# 13.7 Labelling and Grouping Inputs

Labelling Inputs

<label>

The code snippets for inputs in earlier sections were kept pretty simple, just to make sure you got the hang of the basics. However, in reality, every input added to the code should be labelled. This increases your website's accessibility in two ways:

1. Screen readers announce the labels for vision-impaired users.
2. When labels are used, the associated area can also activate the input. For example, instead of being limited to the small checkbox, you can click on the associated text as well (try this out with the code below). This is great for users with tablets or those who have problems clicking on a small checkbox.

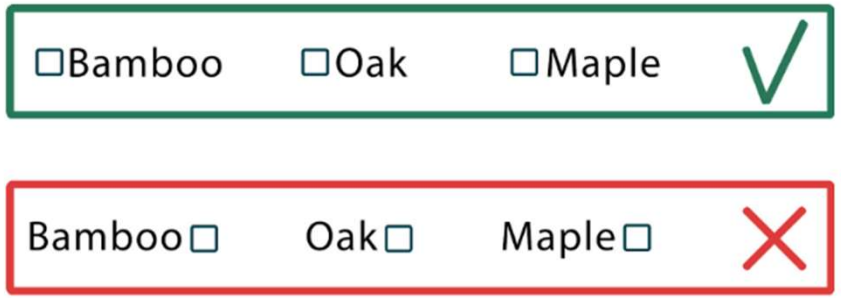
```
<p> How do you feel about tea? </p>
<label for="tea"> I love tea! </label>
<input type="radio" name=”teaopinion” id="tea">
<br>
<label for="coffee"> I prefer coffee. </label>
<input type="radio" name=”teaopinion” id="coffee">
```

for attribute

For the label to know which input you're associating it with, do not forget to add the `for` attribute to the `<label>` element and an `id` attribute to the `<input>` element. The values for the `for` and `id` attributes should match; this is how the browser will know they're related.

To avoid confusing the user, be careful with label placement. Here are the general best practices:

- With text inputs, text areas, select boxes, and file uploads, **place the label left of or above** the input.
- For individual radio buttons and checkboxes, place the label **to the right** of the input.
- You can also separate each input and its label with `<br>` or `<div>` tags. That way, each pair will appear in a separate line, as in the code snippet above.



Grouping Inputs

<fieldset>

If you have a longer form, or if you're not using a form at all (just separate inputs), you can group related elements together with a `<fieldset>` tag. Typically, the browser will render a box around the grouped inputs. You can then modify the box with CSS styling.

<legend>

To clarify the purpose of the grouping for the user, use the `<legend>` element after opening your `<fieldset>`. Its contents will appear on the top line of the box.

```
<fieldset>
<legend>Word Associations</legend>
<label>Happy:
<br> <input type="text" name="happy" ></label>
<br>
<label>Rain:
<br >
<input type="text" name="rain" ></label>
<br >
<label>Red:<br >
<input type="text" name="red" ></label>
</fieldset>
```

A couple perks of using the `<fieldset>` element:

1. You can disable all grouped elements at the same time.
2. If you add the `form` attribute to specific inputs, you can specify a web server of a different form to send data from the grouped inputs. For example, if a Marketing team processes information differently and some (but not all) inputs from your form are relevant to them, you could group those elements and send it to the Marketing's system separately.

> **Note: With JavaScript, you can add multiple separate requests on a single HTML button. If you'd want to do that with HTML, you'd have to insert a form inside a form.*