Using Buttons

In this lesson, we'll cover the usage of buttons in HTML forms.
Let's begin!

WE'LL COVER THE FOLLOWING

- Listing 4-2: Using the `changeBy()` JavaScript method
- Listing 4-3: Adding a reset button
- Listing 4-4: Adding image to represent a button
- Listing 4-5: Composing an image and a text into a button
- Listing 4-6: Making use of the `formaction` attribute

In the previous form examples, you only used one kind of button, the submit button, which was represented by an `<input>` tag with type set to `"submit"`. You have other options to use buttons in your forms, here you will review them.

In the Exercise-04-08 folder within the source code download of the chapter, you'll find a project that includes a number of HTML files.

When you work with forms, you do not have to send the form back to the server, you may run a JavaScript function to respond to the event when the user submits the form.

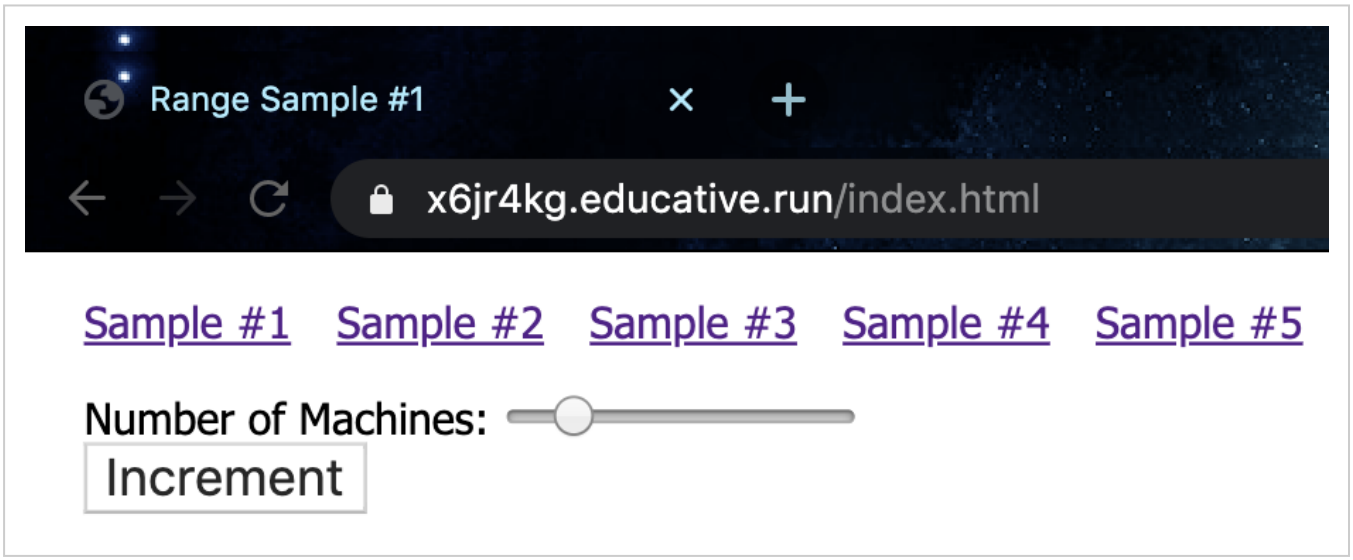
Listing 4-2 demonstrates this; it shows that every time the user clicks the submit button, the `changeBy()` JavaScript method is called with "1" passed as the function argument.

Listing 4-2: Using the `changeBy()` JavaScript method

```
function changeBy(byVal) {
  var range = document.getElementById('noMachines');
```

```
  var range = document.getElementById('noMachines');
  var newVal = parseInt(range.value) + byVal;
  if (newVal < range.min) newVal = range.min;
  if (newVal > range.max) newVal = range.max;
  range.value = newVal;
}
```

This form uses a range control where the type of `<input>` is set to “range”, with the initial value of two (`value="2`"). By clicking the submit button, it increments the control’s current value. After clicking the button twice, the control has a value of four, as shown below:



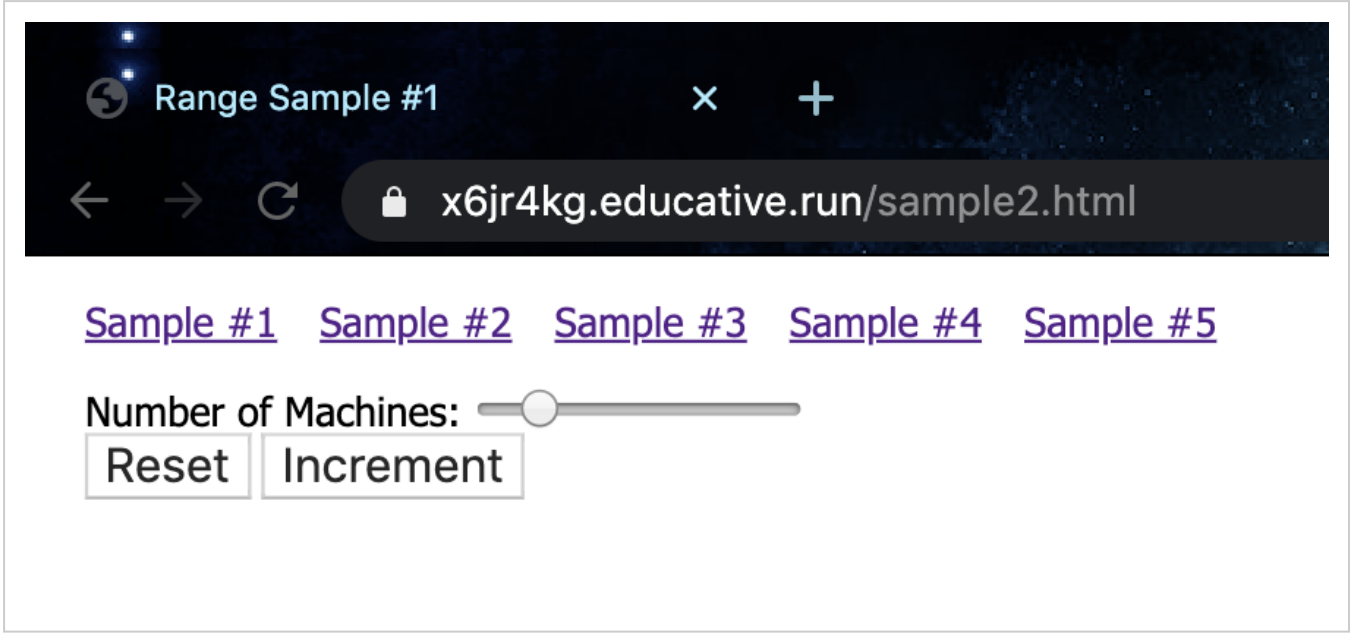
Form using a JavaScript action

When you use `<input>` with the type of `“reset”`, clicking this button resets the form data to its initial state, including the value of each control. Listing 4-3 extends Listing 4-2 with a reset button. When you display the page, after two increments it reaches the value of four, and then clicking Reset will set the range back to its initial value, 2.

Listing 4-3: Adding a reset button

```
function changeBy(byVal) {
  var range = document.getElementById('noMachines');
  var newVal = parseInt(range.value) + byVal;
  if (newVal < range.min) newVal = range.min;
  if (newVal > range.max) newVal = range.max;
  range.value = newVal;
}
```

Sample2.html produces the output shown below:

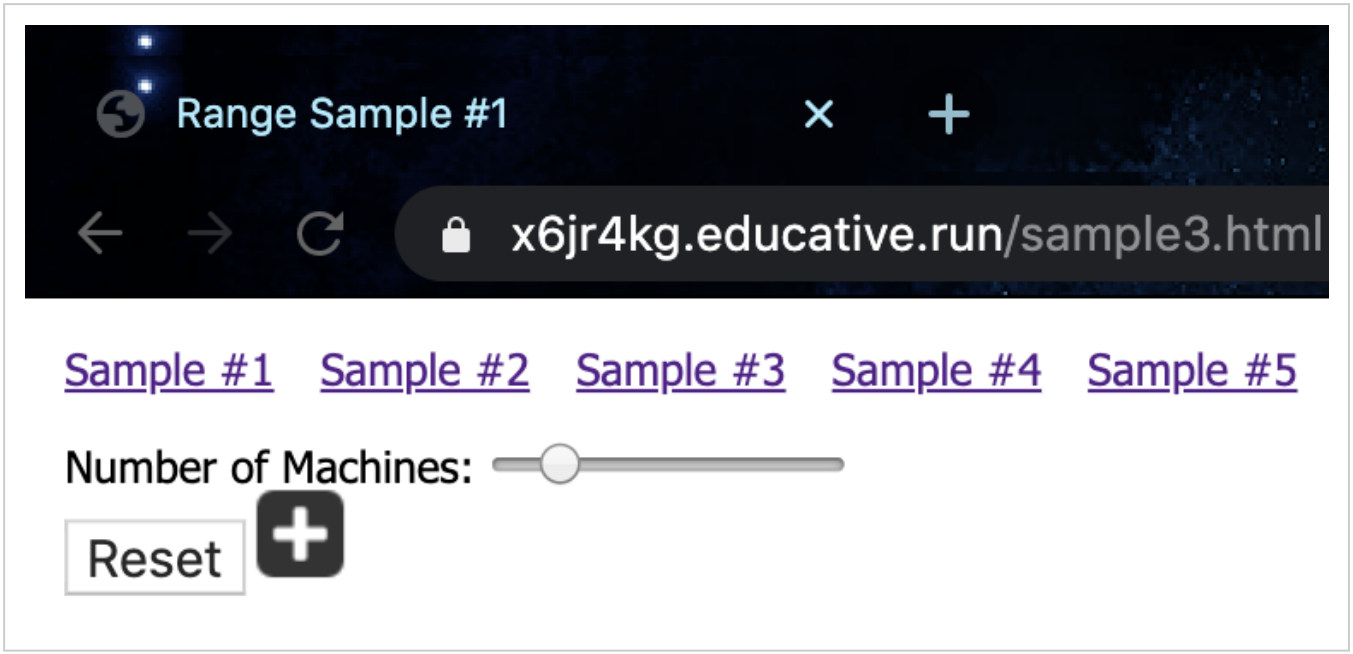


You can use images instead of buttons. Use the “image” type of `<input>`, as Listing 4-4 shows. In this case, you need to specify the source of the image in the src attribute. You may use the width and height attributes, too. The image below shows the submit button represented by an image.

Listing 4-4: Adding image to represent a button

```
function changeBy(byVal) {
  var range = document.getElementById('noMachines');
  var newVal = parseInt(range.value) + byVal;
  if (newVal < range.min) newVal = range.min;
  if (newVal > range.max) newVal = range.max;
  range.value = newVal;
}
```

Sample3.html produces the output shown below:



Now, the submit button is represented by an image

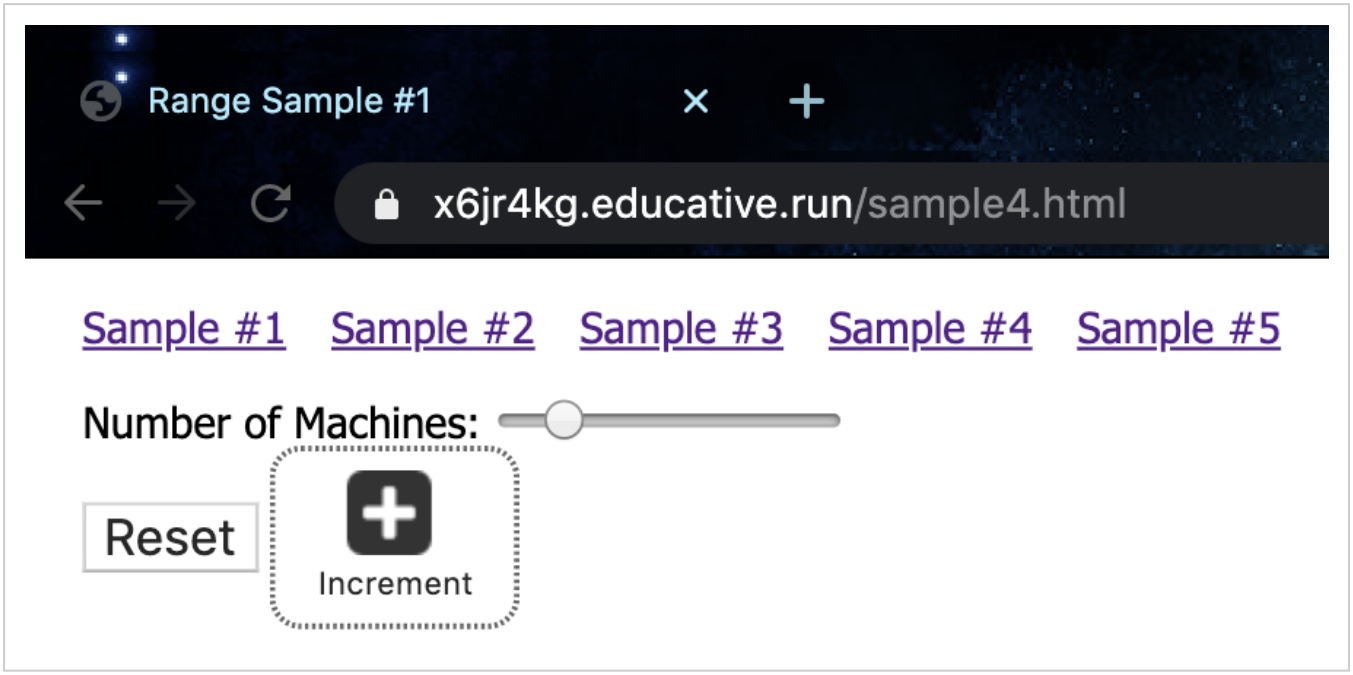
As the image above shows, the submit button is represented by an image. You are not constrained to apply a simple button or an image, you can create a compound markup to represent a clickable area with the `<button>` element.

Listing 4-5 composes an image and a text into a button. For a little styling help, see the button rule in style.css. The result is shown in the image below.

Listing 4-5: Composing an image and a text into a button

```
function changeBy(byVal) {
  var range = document.getElementById('noMachines');
  var newVal = parseInt(range.value) + byVal;
  if (newVal < range.min) newVal = range.min;
  if (newVal > range.max) newVal = range.max;
  range.value = newVal;
}
```

Sample4.html produces the output shown below:



A submit <button> is used

Fifth, you can use more submit buttons that each having their own action. Earlier you learned that a `<form>` has only one action attribute that describes what to do when the submit button is clicked. How can a single attribute describe multiple actions so that each submit button can have its own? It cannot.

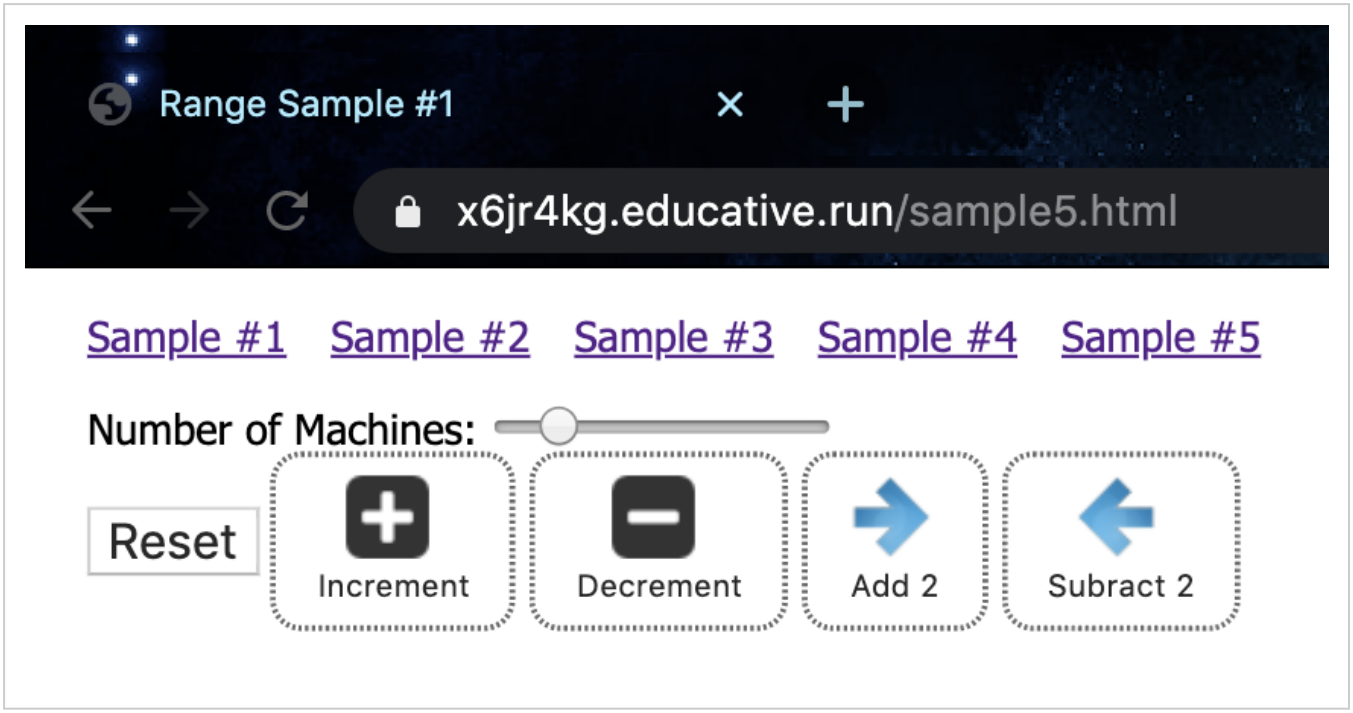
Submit buttons may have a `formaction` attribute that overrides the action attribute of the form.

Listing 4-6 demonstrates this feature with three new buttons (image below). Observe that all new `<button>` elements have a `formaction` attribute.

Listing 4-6: Making use of the `formaction` attribute

```
function changeBy(byVal) {
  var range = document.getElementById('noMachines');
  var newVal = parseInt(range.value) + byVal;
  if (newVal < range.min) newVal = range.min;
  if (newVal > range.max) newVal = range.max;
  range.value = newVal;
}
```

Sample5.html produces the output shown below:



Four submit buttons on the same form

> 📜**NOTE**: Besides the `formaction` attribute, `<input>` offers you the `formmethod` attribute to override the method attribute of the `<form>` element. For example, if the form's method suggests a GET verb, in the `formmethod` that belongs to a submit button you can change it to POST.

By now, you have learned many important things about forms. However, one pivotal thing is still missing: validation. It's time to get acquainted with this concept.