**Form attributes**: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form>

**The Input Element**

The most versatile and widely used form control is the input element. From text inputs to radios and checkboxes, many of the form controls you see and use are input elements. The big difference is the type of input, which lets the browser know what control type to display as well as how it should behave.

## Type

The type attribute is where you always want to start when creating an input element. With the development of HTML5, the number of types available almost tripled. Some of these will be rarely used, but they all serve their purpose.

The most common, and the default value of the type attribute, is text. Text inputs are simple boxes you can enter any Unicode characters you want in for the value. They also do not have a maximum or minimum length by default, so you could provide it with an entire paragraph of text if need be, though we'll see an element later on that is better suited for large blocks of text like comments or email message bodies.

The other input types are as follows:

| **Value** | **Description** |
| --- | --- |
| button | A push button with no default behavior. |
| checkbox | A check box. You must use the value attribute to define the value submitted by this item. Use the checked attribute to indicate whether this item is selected. You can also use the indeterminate attribute to indicate that the checkbox is in an indeterminate state (on most platforms, this draws a horizontal line across the checkbox). |
| color | A control for specifying a color. A color picker's UI has no required features other than accepting simple colors as text. |
| date | A control for entering a date (year, month, and day, with no time). |
| datetime-local | A control for entering a date and time, with no time zone. |
| email | A field for editing an e-mail address. The input value is validated to contain either the empty string or a single valid e-mail address before submitting. The :valid and :invalid CSS pseudo-classes are applied as appropriate. |
| file | A control that lets the user select a file. Use the accept attribute to define the types of files that the control can select. |
| hidden | A control that is not displayed, but whose value is submitted to the server. |
| image | A graphical submit button. You must use the src attribute to define the source of the image and the alt attribute to define alternative text. You can use the height and width attributes to define the size of the image in pixels. |
| month | A control for entering a month and year, with no time zone. |
| number | A control for entering a floating point number. |
| password | A single-line text field whose value is obscured. Use the maxlength attribute to specify the maximum length of the value that can be entered. |
| radio | A radio button. You must use the value attribute to define the value submitted by this item. Use the checked attribute to indicate whether this item is selected by default. Radio buttons that have the same value for the name attribute are in the same "radio button group"; only one radio button in a group can be selected at one time. |
| range | A control for entering a number whose exact value is not important. This type control uses the following default values if the corresponding attributes are not specified:  min: 0  max: 100  value: min + (max-min)/2, or min if max is less than min  step: 1 |
| reset | A button that resets the contents of the form to default values. |
| search | A single-line text field for entering search strings; line-breaks are automatically removed from the input value. |
| submit | A button that submits the form. |
| tel | A control for entering a telephone number; line-breaks are automatically removed from the input value, but no other syntax is enforced. You can use attributes such as pattern and maxlength to restrict values entered in the control. The :valid and :invalid CSS pseudo-classes are applied as appropriate. |
| text | A single-line text field; line-breaks are automatically removed from the input value. |
| time | A control for entering a time value with no time zone. |
| url | A field for editing a URL. The input value is validated to contain either the empty string or a valid absolute URL before submitting. Line-breaks and leading or trailing whitespace are automatically removed from the input value. You can use attributes such as pattern and maxlength to restrict values entered in the control. The :valid and :invalid CSS pseudo-classes are applied as appropriate. |
| week | A control for entering a date consisting of a week-year number and a week number with no time zone. |

**Checked**

If you'd like to set a checkbox to its checked state by default or set one of the radio inputs to checked, a boolean attribute of checked can be added. This is recommended with radios since the form could conceivably be submitted with no option selected without it. Note, however, that Firefox will remember the state of checkboxes and radios that were previously changed by the user, overriding the checked attributes on them. To disable this behavior on either individual inputs or on the entire form, add the attribute autocomplete and set it to off.

## Inputmode

Now that a large portion of all Web traffic is from a mobile or tablet device, the inputmode attribute is especially useful. Have you ever filled out a form on your phone and wondered why the phone number input doesn't change the keyboard to a number pad? This attribute will allow you to fix that. The value of inputmode will tell the browser that it's suggested that the keyboard layout be changed to fit a specific format. The common ones are email, tel, numeric, and url, but there are others that don't directly correspond to an input type.

| **Value** | **Description** |
| --- | --- |
| verbatim | Alphanumeric, non-prose content such as usernames and passwords. |
| latin | Latin-script input in the user's preferred language with typing aids such as text prediction enabled. For human-to-computer communication such as search boxes. |
| latin-name | As latin, but for human names. |
| latin-prose | As latin, but with more aggressive typing aids. For human-to-human communication such as instant messaging for email. |
| full-width-latin | As latin-prose, but for the user's secondary languages. |
| kana | Kana or romaji input, typically hiragana input, using full-width characters, with support for converting to kanji. Intended for Japanese text input. |
| katakana | Katakana input, using full-width characters, with support for converting to kanji. Intended for Japanese text input. |
| numeric | Numeric input, including keys for the digits 0 to 9, the user's preferred thousands separator character, and the character for indicating negative numbers. Intended for numeric codes, e.g. credit card numbers. For actual numbers, prefer using <input type="number"> |
| tel | Telephone input, including asterisk and pound key. Use <input type="tel"> if possible instead. |
| email | Email input. Use <input type="email"> if possible instead. |
| url | URL input. Use <input type="url"> if possible instead. |

## Pattern

This is a regular expression that can be added to the text-style inputs like search, url, and email. If you know Javascript-style regular expressions, you can write these. Just remember to remove the surrounding forward slashes. If you have no idea what any of this means, don't worry. Regular expressions are both a powerful and complicated type of filtering mechanism to quickly locate matches or test validity of strings. The server side of forms should also be validating the user's values, since a form can technically be submitted even if the data input wouldn't pass the regular expression's validation. Just remember that once you're a Javascript expert, you can add an extra layer of validation by way of the pattern element and your new found expertise with regular expressions.

## Readonly

This is a boolean attribute very similar to the disabled attribute. The difference is that it will not work on some inputs, like checkboxes and radios.

## Required

When the browser being used has built-in validation, this boolean attribute tells the browser not to submit the form until the input has a value. This also gives us a handy pseudo-class in CSS to style required fields and mark them in an identifiable manner.

## Autofocus

When an input has this boolean attribute, the browser will put the user's focus on that input when the page first loads. This attribute can only be placed on one form control per page, though, since there's no way to have the user focus on two inputs at once.

Top of Form

Bottom of Form

## Select Element

Now that we've got inputs under our belts, the rest of the form controls will be quick to learn. The select element is a pretty basic element. Selects are used when you want to create an expandable drop-down list of options for the user to choose from, or a boxed list of options where you can select one or more options. It has two possible child elements, the option and optgroup elements. The select element uses the name attribute just like the other form elements. Rather than have the value on the select element, however, the value comes from the option element within it that is selected.

## Option Element

The option element is what makes the select element work. Just like an unordered list is nothing without its list items, a selectelement is useless without its option elements. Each of these represents a possible value for the select. Option elements will use the value attribute as the value that is sent with the select's name on form submission. If the value attribute is omitted, the text within the option element will be used instead. Other than the disabled attribute, this is really the only attribute to be concerned with on option elements.

Many select boxes make use of a placeholder option. This normally says something like "Choose one" and has a value attribute of an empty string, "". This works like the placeholder attribute on text boxes, where the user sees helpful text but the value of the select is empty until the user does something with the control.

## Optgroup Element

When you feel the need to group options together with a group label, the optgroup element is what you would use. This has one required attribute, label, that is used to display text at the top of the group of options. To give you an example of how the optgroup element is used, imagine you have a list of movies that a user can purchase from your site. Some are DVD format, others are Blu-Ray. Instead of using two selects when you really only need one, you can use two optgroup elements to separate the options by label.

### Multiple

The select element also has a multiple boolean attribute that allows the user to select more than one option from the select box. The select is normally turned into a scrollable rectangle that displays multiple options at once. The user then holds either Ctrl or Command on their keyboard as they select options. If you use this attribute, the size attribute can also be used to control how many rows of text will be displayed at a time. You can just as easily control the size of the element with CSS, however, and that will usually lead to a more accurate way of controlling height across browsers.

## Textarea element

This element provides the user with a multiline version of a text input. Unlike inputs, which ignore carriage return and newline characters, textarea elements retain them and use them to format the text. Unlike the other form controls, though, the value of the textarea is not represented with a value attribute. Instead, the value would be the text that is contained within the opening and closing tags of the element. If you wanted to pre-fill the textarea with a sentence, for example, you would add that sentence as the text inside of the element.

### Rows and Cols

Notice those two new attributes? These are only available on the textarea element. Like the size attribute on select elements, these two control the horizontal and vertical size of the element, measured in the number of characters that will be displayed per row and how many rows to display. Again, the width and height can be fine-tuned with CSS, but it's a good idea to include these attributes in case the CSS doesn't kick in. Also of note is that the row count is not a limit to the number of rows you can enter in but how many rows are visible at a time. If it exceeds this height, scrollbars appear within the textarea.

Because of the ability to set rows and columns on textareas, the default font used for them is called a monospace font. Fonts in this category have a pre-set width that each character takes up. That way, the browser is able to determine when to break the text to a new row.

**A form that prevents iOS devices from modifying the email input:**

<form action="" method="post">

<fieldset>

<dl>

<dt>

<label for="first\_name">First Name</label>

</dt>

<dd>

<input type="text" name="first\_name" id="first\_name" />

</dd>

<dt>

<label for="last\_name">Last Name</label>

</dt>

<dd>

<input type="text" name="last\_name" id="last\_name" />

</dd>

<dt>

<label for="email">Email Address</label>

</dt>

<dd>

<input type="email" name="email" id="email" **autocomplete="off" autocorrect="off" autocapitalize="none"** />

</dd>

<dt>

<label for="phone">Phone Number</label>

</dt>

<dd>

<input type="tel" name="phone" id="phone" />

</dd>

<dt>

<label for="phone\_type">Phone Type</label>

</dt>

<dd>

<select name="phone\_type" id="phone\_type">

<option value="home" selected>Home</option>

<option value="business">Business</option>

<option value="mobile">Mobile</option>

</select>

</dd>

</dl>

<input type="submit" value="Send" />

</fieldset

</form>