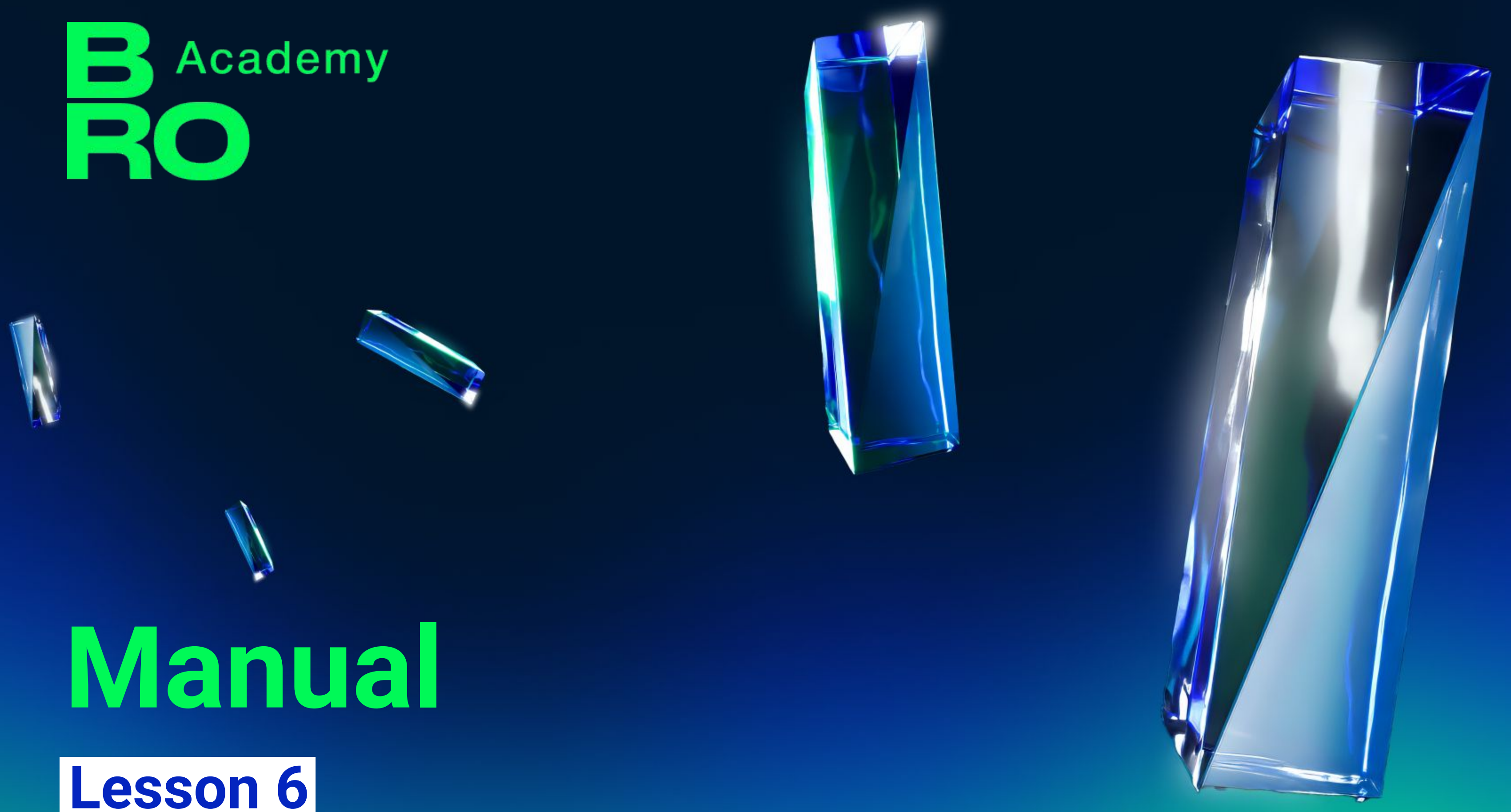
BRO Academy



Manual

Lesson 6

`<form>` Attributes

`action` – The **URL** where the form data will be sent

`method` – The HTTP method used for sending form data

- **GET** – Used to retrieve data
- **POST** – Used to send data, especially sensitive information

```html
<form action="submit.php" method="post">
```

`<form>`

Form controls live inside a <form> element. This element should always carry the action attribute and will usually have a method and id attribute too.

`action`

Every <form> element requires an action attribute. Its value is the URL for the page on the server that will receive the information in the form when it is submitted.

`method`

Forms can be sent using one of two methods: get or post.

With the **get** method, the values from the form are added to the end of the URL specified in the action attribute. The **get method** is ideal for:

- short forms (such as search boxes)
- when you are just retrieving data from the web server (not sending information that should be added to or deleted from a database)

With the **post** method the values are sent in what are known as HTTP headers. 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

If the method attribute is not used, the form data will be sent using the get method.

Example

```
<form
      action="subscribe.php"
      method="get">

</form>
```

`<input>`

– Used to create interactive form controls for user data input.

– The default type is `text`

– One of the most powerful and complex HTML elements due to the vast number of input types and attributes.

The <input> element is used to create several different form controls. The value of the type attribute determines what kind of input they will be creating.

```
<input type="text">
```

INPUT ATTRIBUTES

`type`	`all`	Specifies the type of input
`disabled`	`all`	Disables the input field, making it unavailable for interaction. Disabled fields are not included in the form data when the form is submitted.
`name`	`all`	Name of the form element, used for identification when submitting data to the server
`value`	`all except image`	Specifies the initial value of the input field, which is displayed when the page loads. If the user changes this value, the new value is sent when the form is submitted.
`autocomplete`	`all except checkbox, radio, and buttons`	Controls whether the browser should offer suggestions based on previously entered values for the same input field. Values can be on or off.
`required`	`all except hidden, range, color, and buttons`	Makes the input field mandatory. If the field is left empty, the browser will not allow the form to be submitted and will display an error message.
`readonly`	`all except hidden, range, color, checkbox, radio, and buttons`	Sets the input field as read-only. The user can see the value but cannot change it. This is useful for displaying data that should not be modified.

`checked`	`checkbox, radio`	Indicates whether a checkbox or radio button is selected
`max`	`date, month, week, time, datetime-local, number, range`	Maximum value for input types that support numeric or date values
`maxlength`	`text, search, url, tel, email, password`	Limits the maximum number of characters that can be entered into the input field.
`min`	`date, month, week, time, datetime-local, number, range`	Minimum value for input types that support numeric or date values
`minlength`	`text, search, url, tel, email, password`	Minimum number of characters that can be entered into the input field.
`pattern`	`text, search, url, tel, email, password`	Defines a regular expression that the input value must match in order to be considered valid. This is useful for enforcing specific formats, such as phone numbers or postal codes.
`placeholder`	`text, search, url, tel, email, password, number`	Displays a short hint inside the input field, indicating what kind of information is expected. The placeholder text disappears when the user starts typing.

INPUT TYPES

input type="text"

Text input fields allow users to enter a single line of text. This input type is commonly used for short textual data, such as names, usernames, or simple search queries.

Relevant Attributes:

- name
- value
- placeholder
- required
- readonly
- disabled
- maxlength / minlength
- autocomplete
- autofocus
- form
- pattern

Example

```
<input
       type="text"
       name="username"
       maxlength="30"
>
```

input type=”search”

Search input fields are designed specifically for entering search queries. This input type is optimized for handling search functionality within a webpage or application, providing an interface that often includes a clear button to reset the input field. The appearance and behavior of input type="search" can vary slightly depending on the browser, but it generally enhances user experience by being tailored for search tasks.

Relevant Attributes:

- name
- value
- placeholder
- required
- readonly
- disabled
- maxlength / minlength
- autocomplete
- autofocus
- form

Example

```
<input
      type="text"
      name="username"
      maxlength="30"
>
```

input type="number"

The <input type="number"> element allows users to enter a number within a specified range or any numerical value. This input type is particularly useful when collecting data that requires a numerical response, such as age, quantity, or measurements.

Relevant Attributes:

- `name`
- `value`
- `placeholder`
- `required`
- `min / max`
- `step`
  Specifies the interval between allowed values. For instance, step="0.01" would allow values like 0.01, 0.02, 0.03, etc. This is particularly useful for decimal values or setting specific increments (e.g., step="5" for increments of 5).
- `readonly`
- `disabled`
- `autocomplete`
- `autofocus`
- `form`

The input type="number" field automatically provides users with an increment and decrement control, allowing them to adjust the value by clicking up or down arrows. This can enhance usability by making it easier to enter precise values.

Unlike input type="text", the input type="number" field restricts input to numbers, reducing the likelihood of user errors and ensuring data consistency.

Example

```
<input
      type="number"
      name="age"
      placeholder="Your age"
>
```

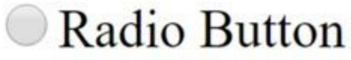
input type="radio"

Radio buttons allow users to pick just one of a number of options.

Relevant Attributes:

- **name**
  When a question provides users with options for answers in the form of radio buttons, the value of the name attribute should be the same for all of the radio buttons used to answer that question.
- **value**
  The value of each of the buttons in a group should be different (so that the server knows which option the user has selected).
- **checked**
  Only one radio button in a group should use this attribute.
- **required**
- **disabled**
- **form**

Please note: Once a radio button has been selected it cannot be deselected. The user can only select a different option. If you are only allowing the user one option and want them to be able to deselect it (for example if they are indicating they agree to terms and conditions), you should use a checkbox instead.

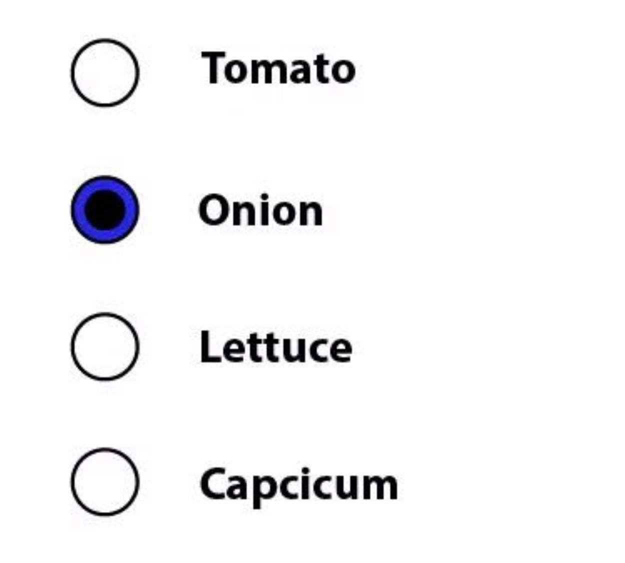


Example

```
<input
        type="radio"
        name="genre"
        value="rock"
        checked="checked"
/> Rock
<input
        type="radio"
        name="genre"
        value="pop"
/> Pop
<input
        type="radio"
        name="genre"
        value="jazz"
/> Jazz
```

Radio Button Usage Rules

```
<input type="radio" name="weight" value="weight0" class="radio
visually-hidden" id=”weight0”>
```



Required Attributes:

`name` – Required Attributes

`value` – Must be unique for each radio button

input type=”checkbox”

Checkboxes allow users to select (and unselect) one or more options in answer to a question.

Relevant Attributes:

- name
- value
  The value attribute specifies the value that is sent to the server if the checkbox is selected. If the checkbox is not checked, nothing is sent for that checkbox. Each checkbox should have a unique value to differentiate between the options selected.
- checked
- required
  Ensures that at least one checkbox must be selected before the form can be submitted. This attribute is useful when a response is mandatory, such as agreeing to terms and conditions.
- disabled
- form

Example

```
<input
      type="checkbox"
      name="service"
      value="itunes"
      checked
/> iTunes
<input
      type="checkbox"
      name="service"
      value="lastfm"
/> Last.fm
<input
      type="checkbox"
      name="service"
      value="spotify"
/> Spotify
```

Checkbox Usage Rules

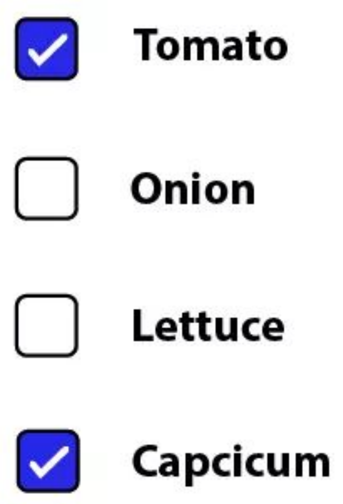
```
<input type="checkbox" name="fruits" value="apple" class="radio
visually-hidden" id="checkbox-apple">
```

Required Attributes:

`value` – Must be unique for each checkbox

Optional:

`name` – Should be the same for all checkboxes in a group



input type=”email”

The <input type="email"> element is designed to handle email address input, ensuring that the data entered follows the standard email format. This input type is particularly useful when collecting user email addresses for registrations, subscriptions, or contact forms.

Email Format Validation: The input type="email" field automatically validates that the entered text matches the general format of an email address (e.g., example@domain.com). However, it does not guarantee that the email address exists or is deliverable.

Error Handling: If the user enters an email address that doesn't match the expected format, the browser will display a validation message and prevent form submission until a valid email is provided.

`Relevant Attributes:`

- `name`
- `value`
- `placeholder`
- `required`
- `readonly`
- `disabled`
- `multiple`
  The multiple attribute allows users to enter more than one email address in the same input field, separated by commas. This is useful in scenarios where you want to allow multiple recipients for an email.
- `autocomplete`
- `autofocus`
- `form`
- `pattern`

Example

```
<input
      type="email"
      name="email"
      placeholder="Your email"
>
```

When the type attribute has a value of password it creates a text box that acts just like a single-line text input, except the characters are blocked out. They are hidden in this way so that if someone is looking over the user's shoulder, they cannot see sensitive data such as passwords.

Although the password is hidden on the screen, this does not mean that the data in a password control is sent securely to the server. You should never use these for sending sensitive data such as credit card numbers.
While autocomplete can be convenient, it's often disabled for password fields to prevent browsers from storing and automatically filling in passwords, which can be a security risk. Developers should consider the security implications of enabling or disabling autocomplete.

input type=”password”

Relevant Attributes:

- `name`
- `value`
- `placeholder`
- `required`
- `readonly`
- `disabled`
- `maxlength / minlength`
- `autocomplete`
- `autofocus`
- `form`
- `pattern`
  Using the pattern attribute, you can enforce specific password rules, such as requiring at least one number, one special character, or a minimum length. However, it's crucial to ensure these requirements are clear to the user.

Example

```
<input
       type="password"
       name="password"
       maxlength="30"
/>
```

input type="url"

The <input type="url"> element is designed to handle URL input, ensuring that the data entered follows the standard format for web addresses. This input type is particularly useful when collecting website URLs, links to online resources, or any other kind of web address.

The input type="url" field automatically validates that the entered text matches the general format of a URL, such as https://www.example.com. However, it does not verify whether the URL actually exists or is reachable.

If the user enters a URL that doesn't match the expected format, the browser will display a validation message and prevent form submission until a valid URL is provided.

Relevant Attributes:

- name
- value
- placeholder
- required
- readonly
- disabled
- autocomplete
  Controls whether the browser should offer to complete the input based on previously entered URLs.
- autofocus
- form
- pattern

URL Address

Example

```
<input
       type="url"
       name="url"
       id="url"
       placeholder="https://example.com"
       pattern="https://.*"
       required
>
```

The <input type="tel"> element is designed to handle telephone number input, allowing users to enter phone numbers in a variety of formats. This input type is useful for collecting contact information in forms where users need to provide their phone numbers. Unlike input types like email or url, the input type="tel" does not enforce specific validation rules or formats. This flexibility allows users to enter phone numbers in various formats, which can be beneficial depending on your audience's location and preferences.

Many mobile browsers will display a number pad when users focus on an input type="tel" field, making it easier for them to enter phone numbers. This enhances the user experience on smartphones and tablets.

input type="tel"

Relevant Attributes:

- `name`
- `value`
- `placeholder`
  Provides a short hint that describes the expected format of the input. For example, the placeholder might display a sample format like (123) 456-7890 or +1-234-567-890 to guide users in entering their phone number correctly.
- `required`
- `readonly`
- `disabled`
- `autocomplete`
  Controls whether the browser should offer to complete the input based on previously entered phone numbers.
- `autofocus`
- `form`
- `pattern`

Example

```
<input
      type="tel"
      id="phone"
      name="phone"
      pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}"
      required
>
```

The <input type="color"> element allows users to select a color using a color picker interface provided by the browser. This input type is useful when you want users to choose a color, such as for customizing themes, selecting a color for a drawing tool, or choosing a background color.

When users click on the color input, most browsers will open a color picker dialog that allows them to choose a color visually. This picker usually provides a palette, sliders, or other tools to help users select the exact shade they want.

While the default color picker interface is provided by the browser and cannot be styled or customized directly, you can create a custom color picker interface using JavaScript and CSS if you need more control over the design or functionality.

input type=”color”

Relevant Attributes:

- `name`
- `value`
  The value should be a valid hexadecimal color code (e.g., #ff0000 for red). If not set, the color picker usually defaults to black (#000000).
- `required`
- `readonly`
- `disabled`
- `autofocus`
- `form`

Example

```
<input
      type="color"
      id="head"
      name="colorpicker"
      value="#e66465"
>
```

input type="hidden"

The <input type="hidden"> element is used to store data that you want to send to the server but do not want the user to see or interact with. This input type is typically used to include information in the form submission that is not visible or editable by the user, such as session tokens, form identifiers, or other metadata.

Common Use Cases:

- Form identifiers to distinguish between multiple forms on a page.
- Session tokens or CSRF (Cross-Site Request Forgery) tokens for security purposes.
- Information about the user or environment that doesn’t need to be visible or editable.
- Flags or status indicators that are used by server-side scripts during form processing.

Hidden inputs can be dynamically updated using JavaScript. For example, you might use JavaScript to set or change the value of a hidden input based on user interactions elsewhere on the page.

Because hidden inputs are not visible, they are not accessible via screen readers or other assistive technologies. If the data needs to be accessible, consider using a different input type or providing the data through an accessible method.

Although hidden inputs are not visible on the page, their values can still be viewed and manipulated by users through browser developer tools. Therefore, they should not be used to store sensitive information, as they do not provide any security on their own.

Relevant Attributes:

- `name`
- `value`

  Since users cannot interact with hidden inputs, this value is typically set programmatically or during the initial page load.
- `form`

Example

```
<input
        type="hidden"
        id="postId"
        name="postId"
        value="34657"
>
```

input type=”file”

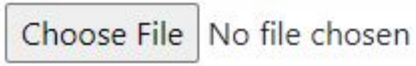
If you want to allow users to upload a file (for example an image, video, mp3, or a PDF), you will need to use a file input box.
This type of input creates a box that looks like a text input followed by a browse button. When the user clicks on the browse button, a window opens up that allows them to select a file from their computer to be uploaded to the website.
When you are allowing users to upload files, the method attribute on the <form> element must have a value of post. (You cannot send files using the HTTP get method.)
When a user clicks on the browse button, the presentation of the window that allows them to browse for the file they want to upload will match the windows of the user's operating system. You cannot control the appearance of these windows.

Relevant Attributes:

- `name`
- `value`
  is read-only, and attempting to set it via HTML or JavaScript does not work due to security restrictions. The value of a file input is the path to the file(s) selected by the user, but this is not accessible via the value attribute.
- `accept`
  specifies the types of files that the server accepts, usually by specifying MIME types or file extensions. For example, accept="image/*" allows all image types, while accept=".pdf,.docx" restricts uploads to PDF and Word documents.
- `multiple`
  allows users to select more than one file at a time. Without this attribute, users can only select a single file.
- `capture`
  is used on mobile devices to specify whether to use the device’s camera or microphone to capture a new file, such as a photo, video, or audio recording, directly from the input field. For example, capture="camera" will directly open the camera app on devices that support this feature.
- `required`
- `disabled`
- `form`

Example

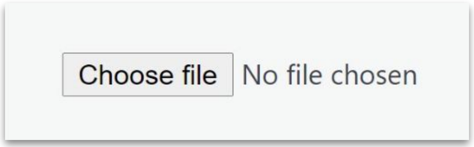
```
<input
      type="file"
      name="user-song"
>
```

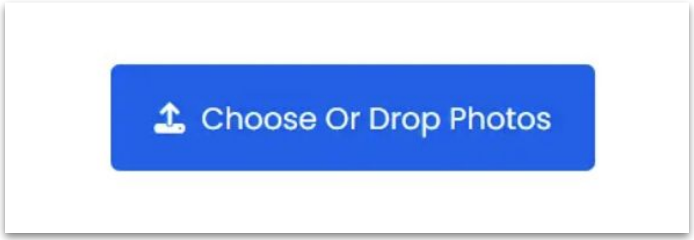


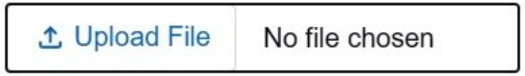
Styling file upload input

– Styled using pseudo-elements.

– Icons can be added using pseudo-elements.







```
input[type="file"]::file-selector-button {}
input[type="file"]::before {}
```

Range input fields allow users to select a numeric value by dragging a slider. This input type is commonly used for settings where users can adjust a value within a predefined range, such as volume controls, brightness settings, or setting a numerical preference.

Since the range input type only provides a slider, it is common to pair it with a <span> or <output> element to display the selected value dynamically as the user interacts with the slider.

While the default appearance of the range slider is consistent with the browser's design, it can be customized using CSS, particularly with pseudo-elements like **::-webkit-slider-thumb** and **::-webkit-slider-runnable-track** in WebKit-based browsers.

input type="range"

Relevant Attributes:

- name
- value
  This value should be a number within the defined range set by the min and max attributes.
- min and max
  These attributes define the minimum and maximum values that the user can select with the slider.
- step
  Specifies the intervals between allowed values on the slider.
- required
- readonly
- disabled
- autocomplete
- autofocus
- form

Example

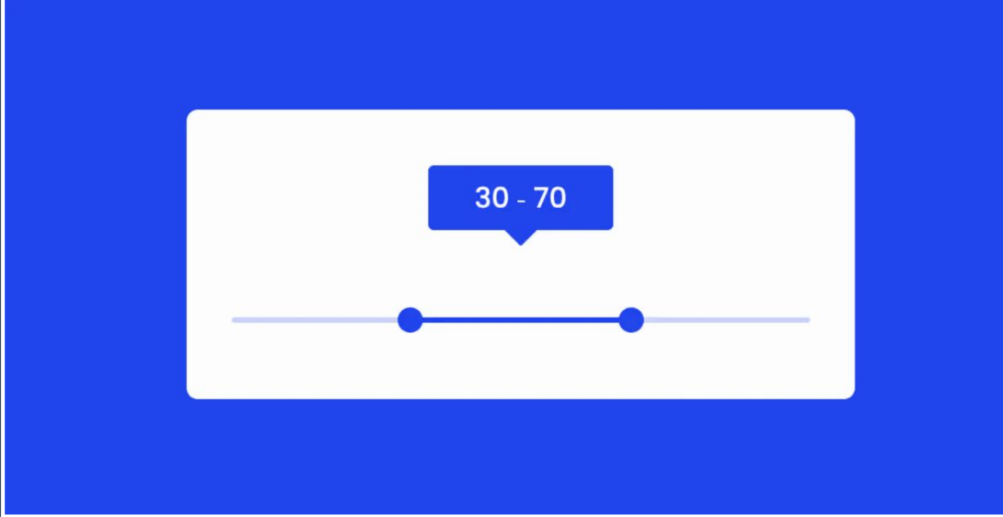
```
<input
      type="range"
      name="price-range"
      min="0"
      max="1000"
>
```

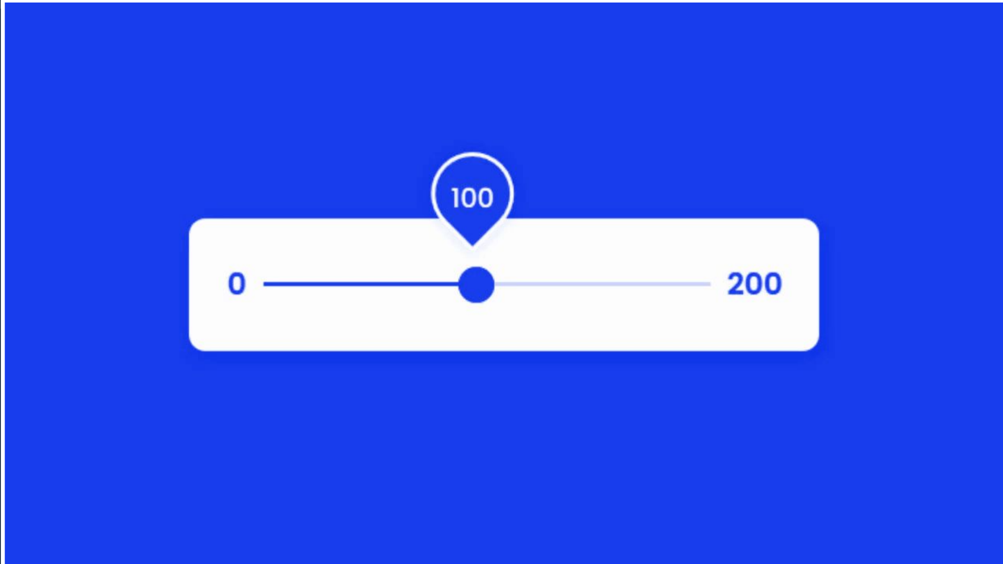
input range

– element allows users to select a value from a specified range using a slider control. This input type is useful for settings where users can adjust a value within a set range, such as volume control or setting a rating.

- Has a single slider
- Difficult to style
- usually requires a plugin for more complex controls

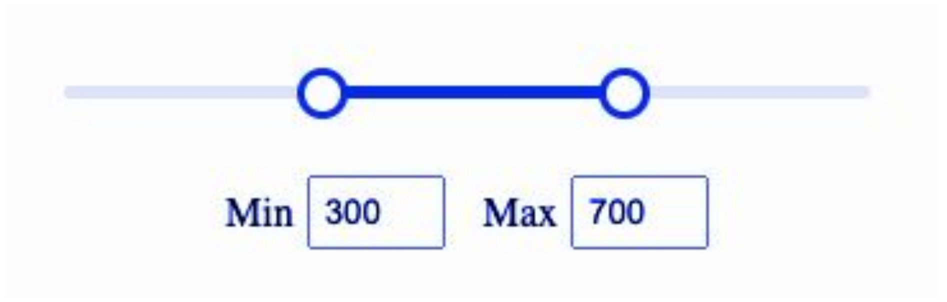
For styling, consider using a JS plugin like `AlRangeSlider`.





input range in HTML

```
<div class=”range”>
    <button class=”range-button is-min></button>
    <button class=”range-button is-max></button>
    <div class=”range-line”></div>
    <div class=”range-line is-selected”></div>
    <div class=”range-container>
        <div class=”range-min”> Min
            <input>
        </div>
        <div class=”range-max”> Max
            <input>
        </div>
    </div>
</div>
```



Min 300 Max 700

The <input type="date"> element allows users to select a date using a date picker provided by the browser. This input type is useful for forms where users need to enter a specific date, such as a birthdate, appointment date, or event date.

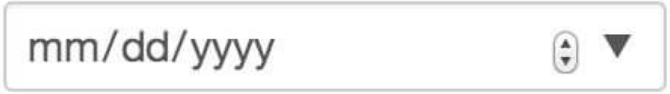
The appearance of the date picker interface can vary depending on the browser and operating system. Most modern browsers provide a calendar widget, making it easier for users to select a date without manually entering it.

While the browser provides basic validation to ensure the date is in the correct format, additional validation may be necessary on the server side, especially when working with date ranges or specific application logic.

input type="date"

Relevant Attributes:

- `name`
- `value`
  The value should be in the format YYYY-MM-DD (e.g., 2024-07-31 for July 31, 2024). This value can be pre-set or left empty for the user to select.
- `min / max`
  define the minimum and maximum dates that the user can select. The values should be in the format YYYY-MM-DD. For example, min="2022-01-01" and max="2024-12-31" restrict the selectable dates to within this range.
- `required`
- `readonly`
- `disabled`
- `autofocus`
- `form`



Example

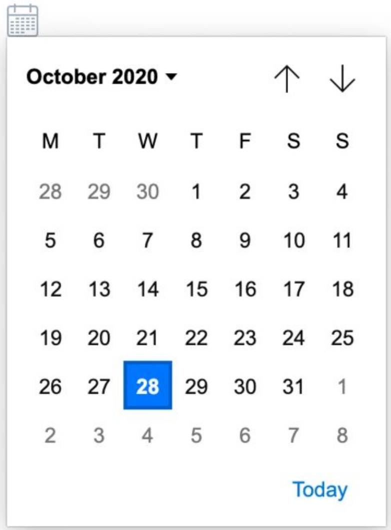
```
<input
      type="date"
      name="birth-date"
>
```

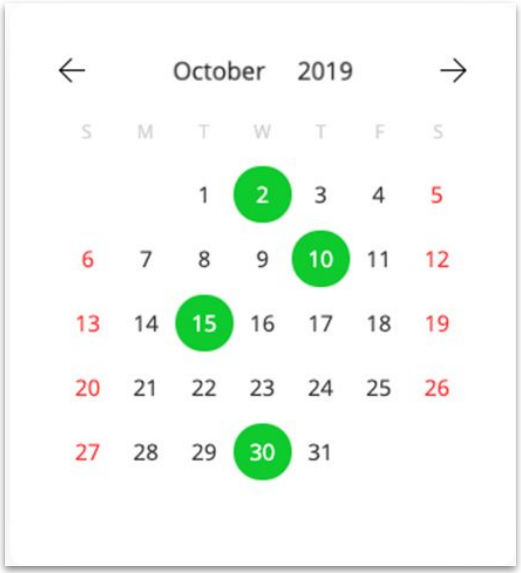
styling datepicker📅

```
<input type="date">
```

— Cannot be styled natively.

— Use JS plugins like [Air datepicker] for custom styling





Month input fields allow users to select a specific month and year, without requiring a day selection. This input type is useful in forms where only the month and year are needed, such as for credit card expiration dates, billing cycles, or general date ranges.

The month picker interface typically allows users to select a month and year from a dropdown or calendar-like widget. The exact appearance can vary depending on the browser and operating system.

This input type is particularly useful when day-specific data is not required, such as tracking monthly trends, scheduling recurring events, or specifying expiration dates.

`input type=”month”`

`Relevant Attributes:`

- `name`
- `value`
  The value should be in the format YYYY-MM (e.g., 2024-07 for July 2024). This value can be pre-set or left empty for the user to select.
- `min / max`
  These attributes define the minimum and maximum month-year values that the user can select. The values should be in the format YYYY-MM. For example, min="2022-01" and max="2024-12" restrict the selectable months to within this range.
- `required`
- `readonly`
- `disabled`
- `autofocus`
- `form`

Example

```
<input
       type="month"
       name="vacation-month"
>
```

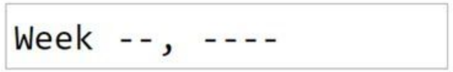
Week input fields allow users to select a specific week of the year, typically represented by the year and the week number. This input type is useful in forms where week-based scheduling or data entry is required, such as for weekly planning, reporting, or tracking purposes.

This input type is particularly useful for applications involving weekly schedules, recurring events, or tracking weekly progress, where the day of the week is less important than the week itself.d

`input type=”week”`

`Relevant Attributes:`

- `name`
- `value`
  The value should be in the format YYYY-Www (e.g., 2024-W31 for the 31st week of 2024). This value can be pre-set or left empty for the user to select.
- `min / max`
  The values should be in the format YYYY-Www. For example, min="2022-W01" and max="2024-W52" restrict the selectable weeks to within this range.
- `required`
- `readonly`
- `disabled`
- `maxlength`
- `autofocus`
- `form`


Week --, ----

Example

```
<input
        type="week"
        name="vacation-week"
>
```

Time input fields allow users to select a time of day, typically including hours and minutes, and optionally seconds, depending on the browser's implementation. This input type is useful in forms where users need to specify a particular time, such as setting an appointment, choosing a time for an event, or specifying a deadline.

This input type is ideal for applications that require specific time entries, such as booking systems, event scheduling, or alarm settings.

input type=”time”

Relevant Attributes:

- `name`
- `value`
  The value should be in the format HH:MM or HH:MM:SS (e.g., 14:30 for 2:30 PM or 14:30:00 for 2:30 PM with seconds). This value can be pre-set or left empty for the user to select.
- `min / max`
  The values should be in the format HH:MM or HH:MM:SS. For example, min="09:00" and max="17:00" restrict the selectable time to between 9:00 AM and 5:00 PM.
- `step`
  For instance, step="300" would allow users to select times in 5-minute increments. The value is in seconds, so 300 seconds equals 5 minutes. If step is not set, the default is 1 minute.
- `required`
- `readonly`
- `disabled`
- `autofocus`
- `form`

Example

```
<input
      type="time"
      name="appointment-time"
>
```

label

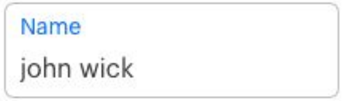
– description for form element

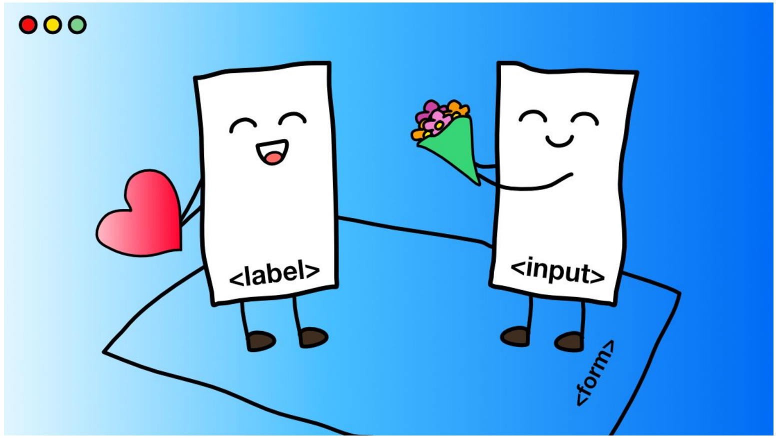
– A label for form elements, linked using the **id** attribute

– Clicking on the **label focus**es on the associated form element

– !Every form element should have a corresponding label

– Can be used for styling elements that are otherwise difficult to style





```
<input type="radio" name="weight" value="weight0" id="weight0">
<label for="weight0"> 0%</label>
```

<label>

When introducing form controls, the code was kept simple by indicating the purpose of each one in text next to it. However, each form control should have its own <label> element as this makes the form accessible to vision-impaired users.

The <label> element can be used in two ways. It can:

1. Wrap around both the text description and the form input (as shown on the first line of the example to your right).
2. Be kept separate from the form control and use the for attribute to indicate which form control it is a label for (as shown with the radio buttons).

for

The for attribute states which form control the label belongs to. Note how the radio buttons use the id attribute. The value of the id attribute uniquely identifies an element from all other elements on a page. (The id attribute is covered on page 183.) The value of the for attribute matches that of the id attribute on the form control it is labelling. This technique using the for and id attributes can be used on any form control. When a <label> element is used with a checkbox or radio button, users can click on either the form control or the label to select. The expanded clickable area makes the form easier to use. The position of the label is very important. If users do not know where to enter information or what information to enter, they are less likely to use the form correctly.

Example

```
<input id="female" type="radio" name="gender"
value="f">
<label for="female">Female</label>
<input id="male" type="radio" name="gender"
value="m">
<label for="male">Male</label>
```

Accessible input hiding

```
<input type="radio" class="visually-hidden">
```

```
.visually-hidden:not(:focus):not(:active) {
  clip: rect(0 0 0 0);
  clip-path: inset(50%);
  height: 1px;
  overflow: hidden;
  position: absolute;
  white-space: nowrap;
  width: 1px;
}
```

Styling Radio & Checkbox

```
<input type="radio" name="weight" value="weight0" class="radio
visually-hidden" id=”weight0”>

<label for="weight0">0%</label>
```

Styles for Unchecked Elements:

```
.radio:not(:checked) + label {}
```

Styles for Checked Elements:

```
.radio:checked + label {}
```

<datalist>

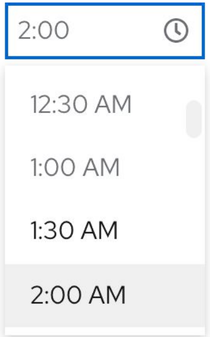
The <datalist> element is used to provide a list of predefined options for an <input> element. This allows users to either type their own input or choose from a list of suggestions. The <datalist> is typically used in combination with an <input> element to enhance form usability, offering users a list of valid choices while still allowing them to enter custom data.

The <datalist> element itself contains a set of <option> elements, each representing a potential choice. These options are displayed as suggestions when the user interacts with the associated <input> field.

To link a <datalist> to an <input>, the input's list attribute must be set to the id of the <datalist>. When the user focuses on or starts typing in the input field, the suggestions from the <datalist> appear.

Works with specific input types such as:

`text, search, range, color, number, date, time, tel, email`



Example

```
<label for="browsers">Choose your browser: </label>
<input list="browsers" id="browser" name="browser">
<datalist id="browsers">
  <option value="Chrome">
  <option value="Firefox">
  <option value="Safari">
  <option value="Opera">
  <option value="Edge">
</datalist>
```

FORM ELEMENTS

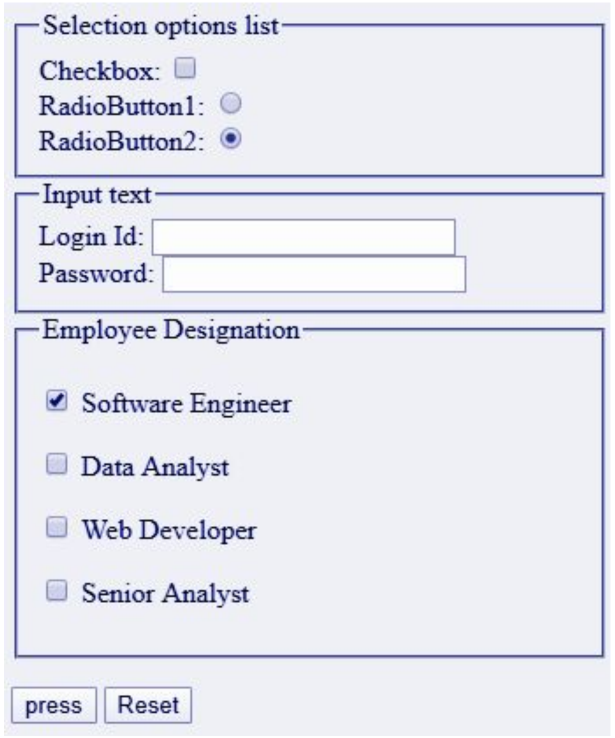
Grouping Elements

`<fieldset> –`

You can group related form controls together inside the <fieldset> element. This is particularly helpful for longer forms.
Most browsers will show the fieldset with a line around the edge to show how they are related. The appearance of these lines can be adjusted using CSS.

`<legend> –`

The <legend> element can come directly after the opening <fieldset> tag and contains a caption which helps identify the purpose of that group of form controls.


Selection options list
Checkbox: ☐
RadioButton1: ○
RadioButton2: ●

Input text
Login Id:
Password:

Employee Designation
✓ Software Engineer
☐ Data Analyst
☐ Web Developer
☐ Senior Analyst

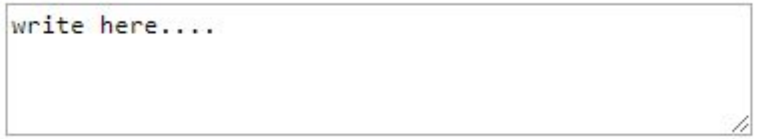
<textarea>

The <textarea> element is used to create a mutli-line text input. Unlike other input elements this is not an empty element. It should therefore have an opening and a closing tag.
Any text that appears between the opening <textarea> and closing </textarea> tags will appear in the text box when the page loads.
If the user does not delete any text between these tags, this message will get sent to the server along with whatever the user has typed. (Some sites use JavaScript to clear this information when the user clicks in the text area.)

If you are creating a new form, you should use CSS to control the width and height of a <textarea>. However, if you are looking at older code, you may see the cols and rows attributes used with this element.

cols
The cols attribute indicates how wide the text area should be (measured in numbers of characters).

rows
The rows attribute indicates how many rows the text area should take up vertically.



```
write here....
```

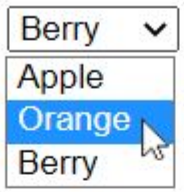
Example

```
<textarea
        name="comments"
        cols="20"
        rows="4">
                Enter your comments...
</textarea>
```

<select>

A drop down list box (also known as a select box) allows users to select one option from a drop down list.
The <select> element is used to create a drop down list box. It contains two or more <option> elements.

The function of the drop down list box is similar to that of the radio buttons (in that only one
option can be selected). There are two key factors in choosing which to use:
1. If users need to see all options at a glance, radio buttons are
better suited.
2. If there is a very long list of options (such as a list of countries), drop down list boxes work better.



Example

```
<select name="devices">
       <option value="ipod">iPod</option>
       <option value="radio">Radio</option>
       <option value="computer">Computer</option>
</select>
```

<select multiple>

size

You can turn a drop down select box into a box that shows more than one option by adding the size attribute. Its value should be the number of options you want to show at once.

multiple

You can allow users to select multiple options from this list by adding the multiple attribute with a value of multiple.
It is a good idea to tell users if they can select more than one option at a time. It is also helpful to indicate that on a PC they should hold down the control key while selecting multiple options and on a Mac they should use the command key while selecting options.

Example

```
<select
       name="instruments"
       size="3"
       multiple>
       <option value="guitar" selected>
              Guitar
       </option>
       <option value="drums">Drums</option>
       <option value="keyboard"
selected">Keyboard</option>
       <option value="bass">Bass</option>

</select>
```

`<option>`

The <option> element is used to specify the options that the user can select from. The words between the opening <option> and closing </option> tags will be shown to the user in the drop down box.

`value`

The <option> element uses the value attribute to indicate the value that is sent to the server along with the name of the control if this option is selected.

`selected`

The selected attribute can be used to indicate the option that should be selected when the page loads. The value of this attribute should be selected. If this attribute is not used, the first option 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 for this control.

Example

```
<select name="devices">
      <option value="ipod">iPod</option>
      <option value="radio">Radio</option>
      <option value="computer">Computer</option>
</select>
```

Styling select

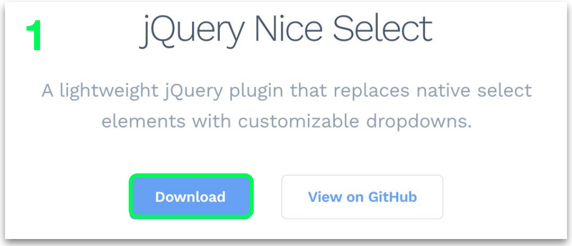
1. **Download** the files from the plugin's page and add them to your project. Place the .js files in the **js folder** and the .css files in the **css folder**.

2. Link the script file by adding a <script> tag **before** the closing **</body>** tag, specifying the path to the files you just downloaded.

3. Include the styles in the <head> tag, **before** your own **stylesheet**.

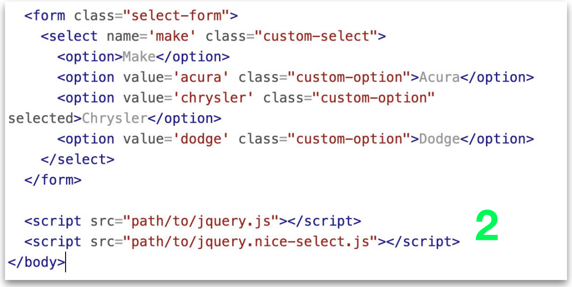
4. After linking the script files, create **your own** <script> tag and **add the code** from the example.

5. **Apply styles** using the classes that were assigned to the select element.



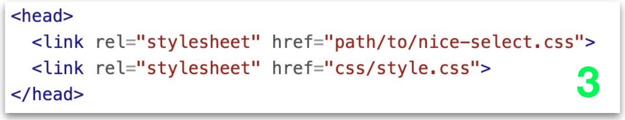
jQuery Nice Select

A lightweight jQuery plugin that replaces native select elements with customizable dropdowns.

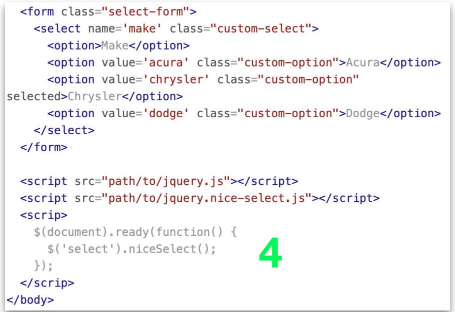


```
  <form class="select-form">
    <select name='make' class="custom-select">
      <option>Make</option>
      <option value='acura' class="custom-option">Acura</option>
      <option value='chrysler' class="custom-option"
selected>Chrysler</option>
      <option value='dodge' class="custom-option">Dodge</option>
    </select>
  </form>

  <script src="path/to/jquery.js"></script>
  <script src="path/to/jquery.nice-select.js"></script>
</body>
```

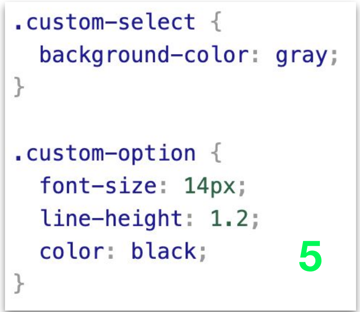


```
<head>
  <link rel="stylesheet" href="path/to/nice-select.css">
  <link rel="stylesheet" href="css/style.css">
</head>
```



```
  <form class="select-form">
    <select name='make' class="custom-select">
      <option>Make</option>
      <option value='acura' class="custom-option">Acura</option>
      <option value='chrysler' class="custom-option"
selected>Chrysler</option>
      <option value='dodge' class="custom-option">Dodge</option>
    </select>
  </form>

  <script src="path/to/jquery.js"></script>
  <script src="path/to/jquery.nice-select.js"></script>
  <scrip>
    $(document).ready(function() {
      $('select').niceSelect();
    });
  </scrip>
</body>
```



```
.custom-select {
  background-color: gray;
}

.custom-option {
  font-size: 14px;
  line-height: 1.2;
  color: black;
}
```

User interface pseudo-class selectors

- `:disabled` – element that is disabled (using the `disabled attribute`)
- `:enabled` – elements that do not have the `disabled attribute`
- `:checked` – `radio or checkbox` elements that have the `checked` attribute or have been selected by the user
- `:indeterminate` – `radio or checkbox` elements that are in an intermediate state, without the checked attribute or not selected by the user
- `:default` – the element that is the default among a group of similar elements

- `:valid` – highlights an element when the entered data meets the specified patterns
- `:invalid` – highlights an element when the entered data does not meet the specified patterns.
- `:in-range` – highlights an element when the user's input falls within the specified range

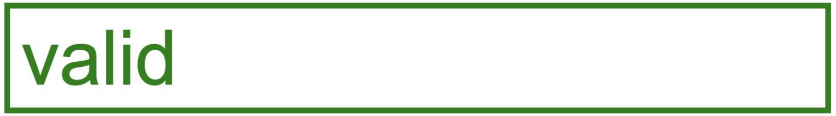
```
input:in-range { }
<input min="1" max="10">
```

- `:out-of-range` – highlights an element when the user's input falls outside the specified range

styling `valid & invalid`

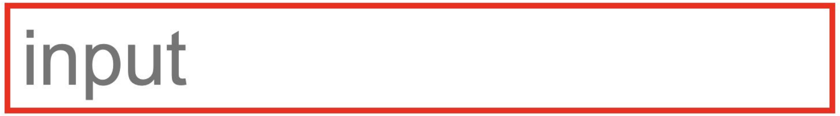
Data entered correctly:

```
input:valid {border-color: green;}
```



Data entered incorrectly:

```
input:invalid {border-color: red;}
```



Note: This works even if the input is empty, because an empty value is also considered invalid.

styling valid & invalid

Considering whether data has been entered into the input:

```
input:invalid:not(:placeholder-shown) {border-color: red;}
input:valid:not(:placeholder-shown) {border-color: green;}
```

This checks whether the placeholder is visible (it's not visible if data has been entered in the field)

styling valid & invalid

Considering whether the input is focused:

```
input:invalid:not(:placeholder-shown):not(:focus)
{border-color: red}
```

This checks whether the placeholder is visible and whether the field is in focus. If the field is focused, the user might still be entering data, so it may be too early to validate.

- `:required` – targets elements that have the required attribute (indicating that the field is mandatory)
- `:optional` – targets elements that do not have the required attribute
- `:read-only` – targets elements with the readonly attribute (indicating that the user cannot change the element's value)
- `:read-write` – targets elements that do not have the readonly attribute
- `:target` – applies styles to elements that are referenced by a fragment identifier (ID) in a URL

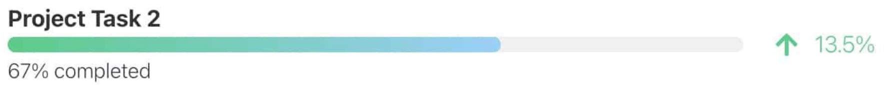
```
<a href="#one">Link</a>
<h2 id="one"> Target element </h2>
```

Pseudo-elements

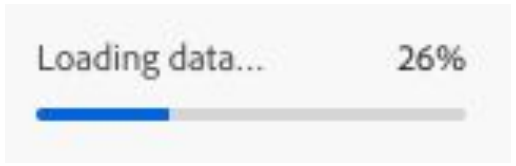
`::placeholder` – Styles the placeholder text inside an **`input`**, when nothing is entered into it

OTHER DYNAMIC ELEMENTS

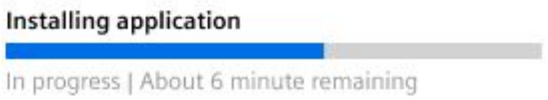
progress

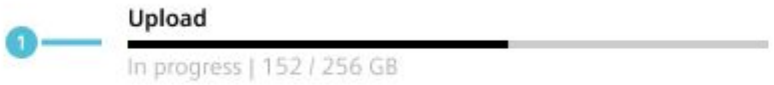


– displays the progress of long-running processes.



- Progress of filling out a registration form.
- Task completion progress.
- Progress of uploading a large file.
- Progress of course completion in online learning.
- Game progress as levels are completed.

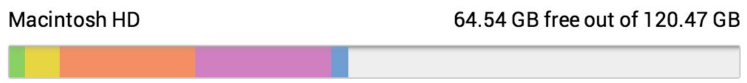


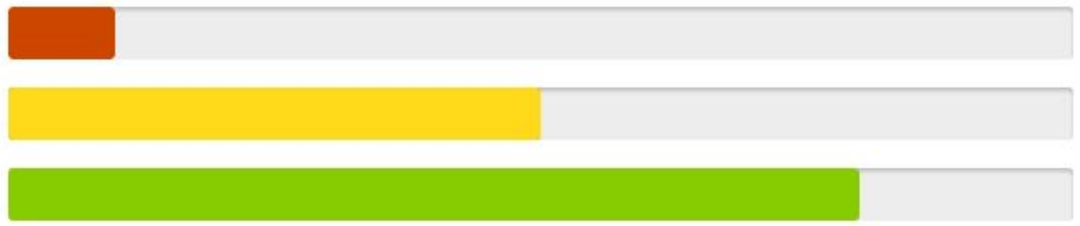


`meter`

– displays a scalar measurement within a known range

- Password strength
- Battery level
- Sound volume





appearance

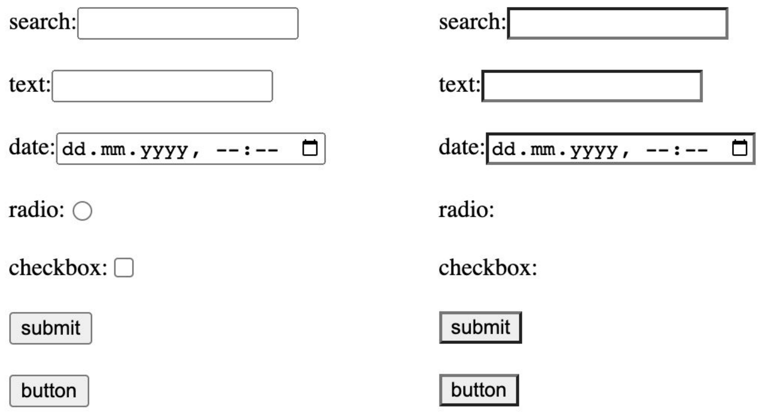
`appearance: none` – resets the appearance of an element to a consistent style across all browsers and operating systems

`appearance` other than **none** – used to apply specific styles to elements that do not have those styles by default

```
appearance: searchfield;
```

The default styling applied by the browser:

```
appearance: auto;
```

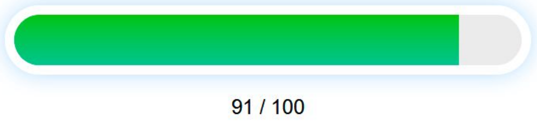


styling progress

```
progress {
  appearance: none;
}

progress::-webkit-progress-bar {
   background-color: grey;
}

progress::-webkit-progress-value {
   background-color: green;
}
```

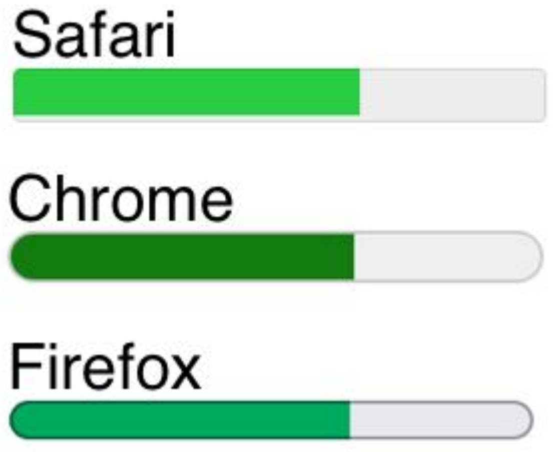


meter

```css
meter {
  appearance: none;
}

meter::-webkit-meter-bar {
  background: none;
    /* Required to get rid of the default background property */
  background-color: grey;
}

meter::-webkit-meter-optimum-value {
  background-color: green;
}
```


Safari
Chrome
Firefox