

B Academy
RO

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

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

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.

`<form>`

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 types

<code><input type="checkbox"></code>	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
<code><input type="color"></code>	<input type="color"/>
<code><input type="date"></code>	<input type="date"/> dd-mm-yyyy
<code><input type="email"></code>	<input type="email"/> farazc60@gmail.com
<code><input type="file"></code>	<input type="file"/> Choose File No file chosen
<code><input type="hidden"></code>	
<code><input type="image"></code>	<input type="image"/>
<code><input type="number"></code>	<input type="number"/> 5
<code><input type="password"></code>	<input type="password"/>
<code><input type="radio"></code>	<input type="radio"/> <input type="radio"/>
<code><input type="range"></code>	<input type="range"/>
<code><input type="reset"></code>	<input type="reset"/> Reset
<code><input type="submit"></code>	<input type="submit"/> Submit
<code><input type="text"></code>	<input type="text"/> codewithfaraz
<code><input type="url"></code>	<input type="url"/> https://www.codewithfaraz.com

When the type attribute has a value of text, it creates a singleline text input.

input type="text"

name

When users enter information into a form, the server needs to know which form control each piece of data was entered into.

(For example, in a login form, the server needs to know what has been entered as the username and what has been given as the password.)

Therefore, each form control requires a name attribute.

The value of this attribute identifies the form control and is sent along with the information they enter to the server.

size

The size attribute should not be used on new forms. It was used in older forms to indicate the width of the text input (measured by the number of characters that would be seen).

For example, a value of 3 would create a box wide enough to display three characters (although a user could enter more characters if they desired).

In any new forms you write, CSS should be used to control the width of form elements.

The size attribute is only mentioned here because you may come across it when looking at older code.

Example

```
<input  
    type="text"  
    name="username"  
    size="15"  
    maxlength="30"  
>
```

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.

input type="number"

min and max

These attributes define the minimum and maximum values that the user can enter. For example, if you want the user to enter an age between 18 and 65, you would set `min="18"` and `max="65"`. These attributes help ensure that the data collected fits within your desired range.

step

The `step` attribute specifies the interval between allowed numbers. For example, if `step="5"`, users can enter values like 5, 10, 15, etc. If the step is not specified, the default is 1, which means users can enter any integer within the specified range.

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"
>
```

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

input type="radio"

name

The name attribute is sent to the server with the value of the option the user selects. 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 attribute indicates the value that is sent to the server for the selected option. 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

The checked attribute can be used to indicate which value (if any) should be selected when the page loads. The value of this attribute is checked. Only one radio button in a group should use this attribute.

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
```


input type="checkbox"

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

name

The name attribute is sent to the server with the value of the option(s) the user selects. When a question provides users with options for answers in the form of checkboxes, the value of the name attribute should be the same for all of the buttons that answer that question.

value

The value attribute indicates the value sent to the server if this checkbox is checked.

checked

The checked attribute indicates that this box should be checked when the page loads. If used, its value should be checked.

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
```

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.

input type="email"

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.

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.

Using input type="email" improves accessibility, as screen readers and other assistive technologies can identify the field as one that requires an email address, providing better context to users.

Example

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

input type="password"

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.

name

The name attribute indicates the name of the password input, which is sent to the server with the password the user enters.

size, maxlength

It can also carry the size and maxlength attributes like the single-line text input.

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.

Example

```
<input  
  type="password"  
  name="password"  
  size="15"  
  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.

URL Format Validation: 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.

Error Handling: 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.

Custom Validation: Although the input type="url" provides basic validation, you can use the pattern attribute to enforce more specific URL formats, ensuring that the data collected meets your requirements.

Example

```
<input
  type="url"
  name="url"
  id="url"
  placeholder="https://example.com"
  pattern="https://.*"
  size="30"
  required
>
```

input type="tel"

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.

Flexibility: 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.

Custom Validation: Although the input type="tel" does not automatically validate phone numbers, you can use the pattern attribute to enforce specific formats. For example, you could require that users include an area code or format the number according to local standards.

Mobile Optimization: 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.

Example

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

input type="color"

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.

Hexadecimal Format: The input type="color" element uses hexadecimal color codes. When you set the value attribute, make sure to use a valid hex code (e.g., #0000ff for blue).

User Interface: 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.

Fallback Behavior: If the browser does not support the color picker, it may fall back to a basic text input where users can manually enter a hex color code.

Custom Color Palettes: 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.

Example

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

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.

input type="hidden"

Invisible to Users: As the name suggests, hidden inputs are not displayed on the page and cannot be interacted with by the user. They are purely for storing data that needs to be sent along with the form submission.

Common Use Cases: Hidden inputs are often used to store data such as:

- 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.

Dynamic Data Handling: 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.

Not Accessible: 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.

Security Considerations: 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.

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.

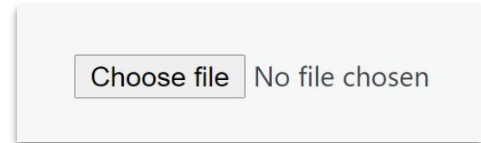
When you are allowing users to upload files, the method attribute on the <form> element must have a value of post. (You cannot send files using the HTTP get method.)

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.

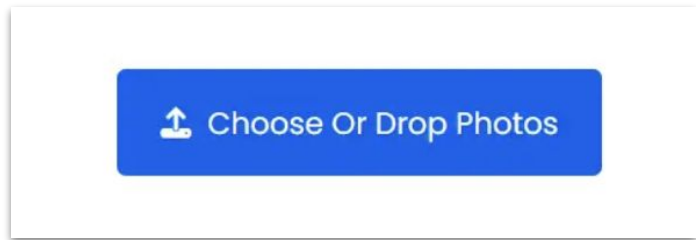
Example

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


Styling `<input type="file">`



- Styled using pseudo-elements.
- Icons can be added using pseudo-elements.



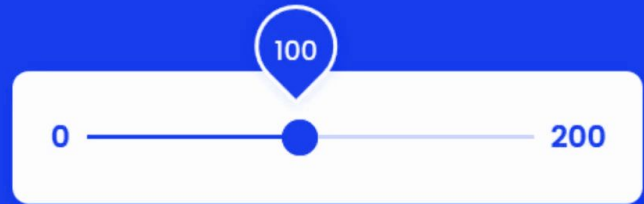
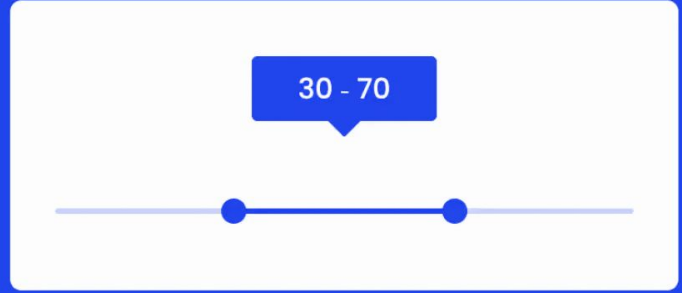
```
input[type="file"]::file-selector-button {}  
input[type="file"]::before {}
```

input

range

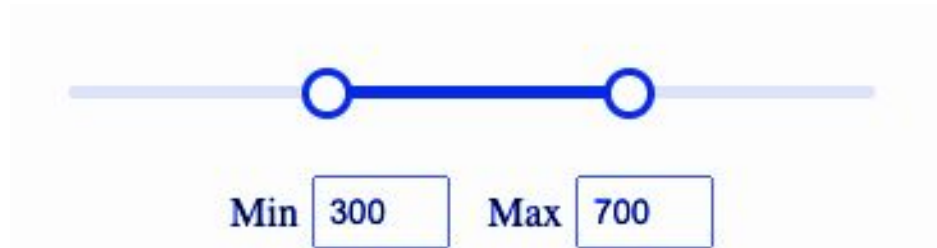
- Has a single slider
- Difficult to style
- usually requires a plugin for more complex controls

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>
```

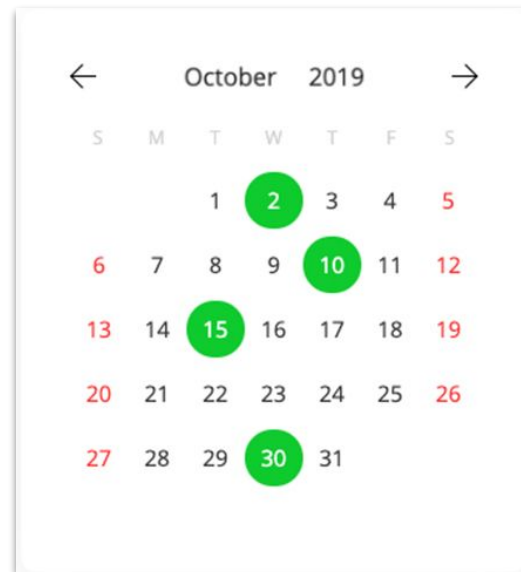
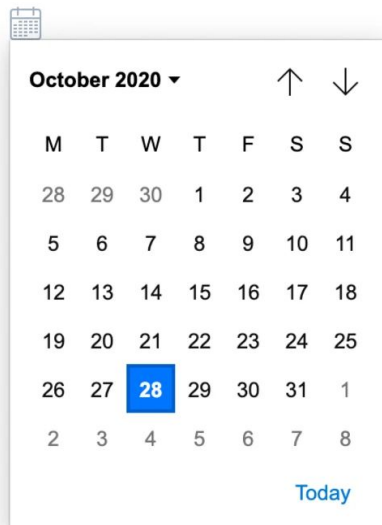


styling datepicker

`<input type="date">`

— Cannot be styled natively.

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



input type="date"

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.

value

The `value` attribute sets the initial date value in the input field when the page loads. The value should be in the format YYYY-MM-DD (e.g., 2024-07-31 for July 31, 2024).

min and max

These attributes define the minimum and maximum dates that the user can select. The values should also be in the format YYYY-MM-DD. This is useful for restricting date selection within a specific range.

Example

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

input attributes

type	all	Specifies the type of input
disabled	all	Indicates that the input is disabled and cannot be interacted with
name	all	Name of the form element, used for identification when submitting data to the server
value	all except image	Initial value of the form control
autocomplete	all except checkbox, radio, and buttons	Suggests autofill options based on saved data in the browser
required	all except hidden, range, color, and buttons	Indicates that the field must be filled out
readonly	all except hidden, range, color, checkbox, radio, and buttons	The input value cannot be changed by the user

input attributes

checked	checkbox, radio	Indicates whether a checkbox or radio button is selected
max	date, month, week, time, datetime-local, number, range	Maximum value
maxlength	text, search, url, tel, email, password	Maximum number of characters
min	date, month, week, time, datetime-local, number, range	Minimum value
minlength	text, search, url, tel, email, password	Minimum number of characters
pattern	text, search, url, tel, email, password	Pattern that the input value must match to be considered valid
placeholder	text, search, url, tel, email, password, number	Text displayed in the input when it has no value

label

- description for form element

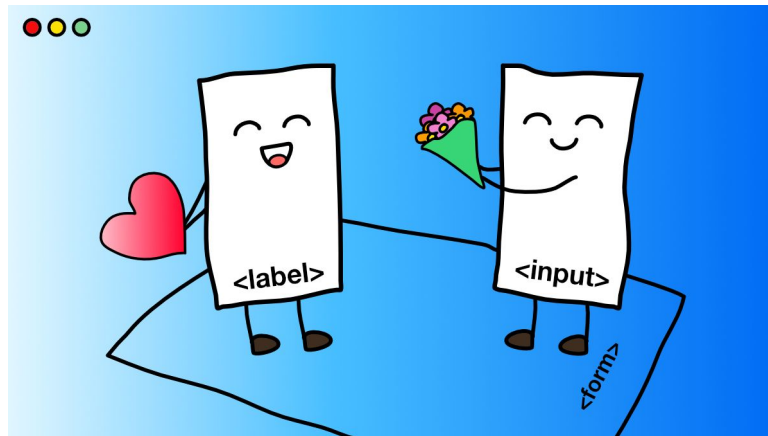
- A label for form elements, linked using the **id** attribute
- Clicking on the **label** focuses on the associated form element
- !Every form element should have a corresponding label
- Can be used for styling elements that are otherwise difficult to style

Label

input

Name

john wick



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

`<label>`

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>
```

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



Tomato



Onion



Lettuce



Capcicum

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



Tomato



Onion



Lettuce



Capcicum

Accessible **input** hiding

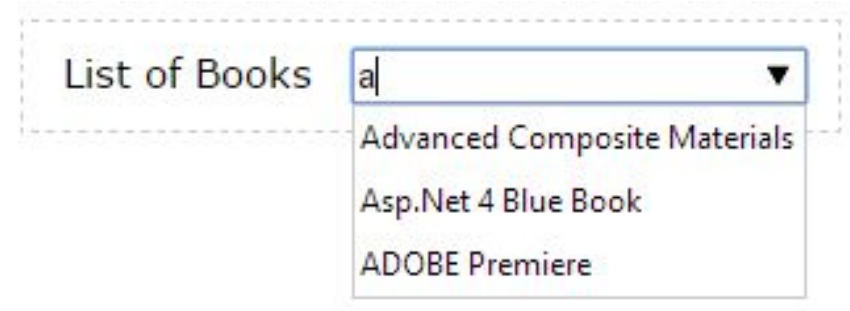
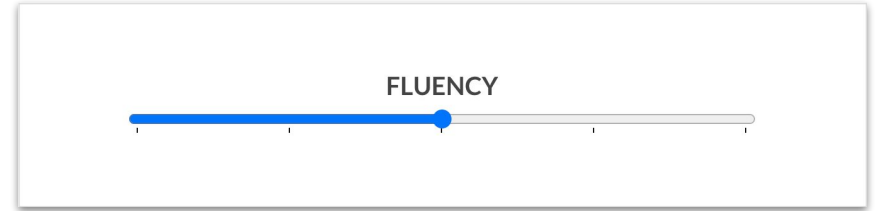
```
<input type="radio" class="visually-hidden">
```

```
.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;  
}
```

datalist

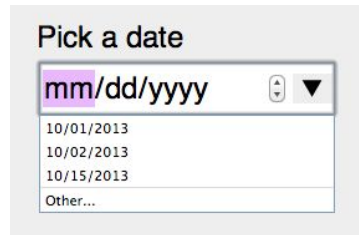
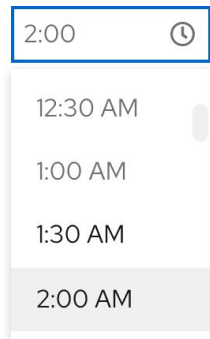
- Provides a list of options for an **input** element.
- You can select an option from the list instead of entering it manually
- Works with specific input types such as:

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



datalist

```
<input list="options" name="input_name">
<datalist id="options">
  <option value="option 1">
  <option value="option 2">
  <option value="option 3">
</datalist>
```



The `<textarea>` element is used to create a multi-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.

`<textarea>`

Example

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

A drop down list box (also known as a select box) allows users to select one option from a drop down list.

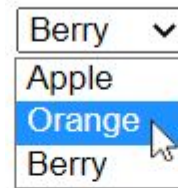
The `<select>` element is used to create a drop down list box. It contains two or more `<option>` elements.

name

The name attribute indicates the name of the form control being sent to the server, along with the value the user selected.

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.



`<select>`

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. In the example you can see that three of the four options are shown.

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>
```

стилизация select

1. **Скачиваем** файлы со страницы плагина и добавляем их в свой проект. .js – в папку **js**, .css – в папку **css**
2. Тег script подключаем **перед** закрывающим тегом **body**, указывая путь к файлам, которые вы только что скачали
3. Подключаем стили в теге head **до** своего **файла стилей**
4. После подключенных тегов script создаем **свой тег** script и **добавляем** туда код из примера
5. **Задаем стили** по тем классам, которые были заданы для select

1 jQuery Nice Select

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

[Download](#)[View on GitHub](#)

```
<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>
```

2

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

3

```
<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>
<script>
  $(document).ready(function() {
    $('select').niceSelect();
  });
</script>
</body>
```

4

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

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

5

User interface **pseudo-class** selectors

– These allow you to style various aspects of elements associated with the **form** tag.

● **: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**

● **:intermediate** – 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

User interface **pseudo-class** selectors

- **: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

# User interface **pseudo-class** selectors

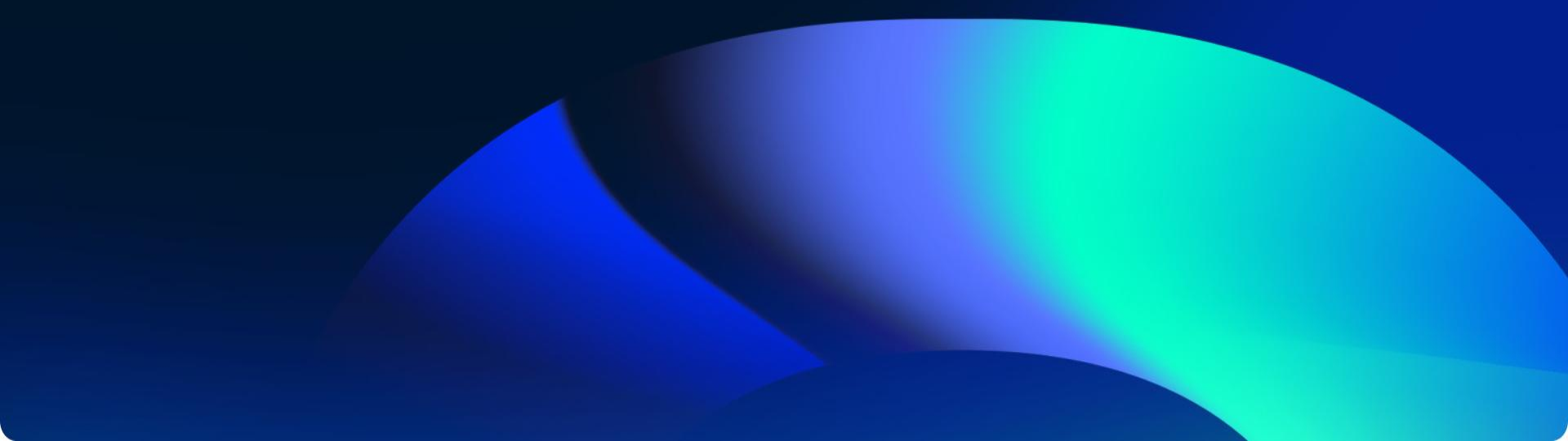
- **:required** – targets elements that have the **required attribute** (indicating that the field is mandatory)
- **:optional** – targets elements that **do not have** the required attribute
- **:read-only** – targets elements with the **readonly attribute** (indicating that the user cannot change the 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**.

```
Link
```

```
<h2 id="one"> Target element </h2>
```

# Pseudo-elements

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



# styling valid & invalid

Data entered correctly:

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

valid

Data entered incorrectly:

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

input

**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.

# 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

press Reset

# progress

Project Task 2

67% completed

↑ 13.5%

– 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.

Loading data...

26%

Installing application

In progress | About 6 minute remaining

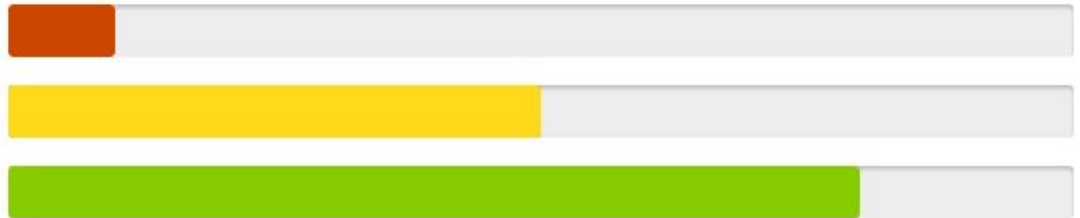
Upload

In progress | 152 / 256 GB

# 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;**

search:

text:

date:

radio: ☐

checkbox: ☐

search:

text:

date:

radio: ☐

checkbox: ☐