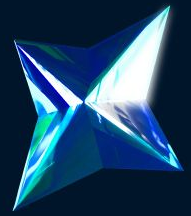
BRO Academy



Forms and Their Styling

Lesson 6

Lesson plan

1
HTML Form Elements

2
Styling Form Elements

3
Form Validation

4
Other Interactive Tags

Registration

Full Name
E.g: John Smith

Username
johnWC98

Email
johnsmith@hotmail.com

Phone Number
012-345-6789

Password
••••••••

Confirm Password
••••••••

Gender

Male Female Prefer not to say

Register

Send us a Message

CodingNepal

Enter your email

Enter your phone

Enter your website

Write your message

Send Message Sending your message...

Price Range

Use slider or enter min and max price

Min 2500 - Max 7500

Subscribe to the newsletter

your@email.com

Subscribe

Start typing in here to take over and try it out

5423 | Jondoe Donjoe

12 / 12 | 121 | Submit

Upload File

Click To Upload

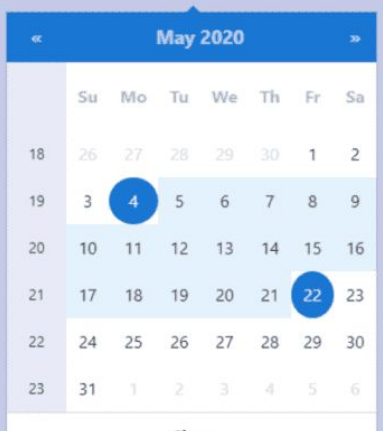
Uploaded Documents

Pdf document.folder.pdf

Pdf Resume.pdf

Docx Ravi Web.docx

Start Date 04-05-2020 End Date 22-05-2020



INGREDIENTS

- [x] Beef
- [ ] Pork
- [ ] Chicken
- [ ] Vegetables

NUTRITION

- [ ] Gluten Free

Your email address: test@example.org

Interactive form demo

Test message *

Optional files

Browse... No files selected.

Send to me

Demo source code

```
<!-- source HTML for the demo -->
<form
  action="https://api.mailslurp.com/forms"
  method="post"
  enctype="multipart/form-data">

  <!-- destination specified with hidden `_to` fi
  <input
    name="_to"
    type="hidden"
    value="test@example.org">

  <!-- use any named inputs, selects, textareas e
  <label>Test message</label>
  <textarea name="message"><textarea>

  <!-- files will be sent as attachments -->
  <label>Optional files</label>
  <input multiple name="files" type="file">

  <button type="submit">Send to me</button>

</form>
```

`<form>` Attributes

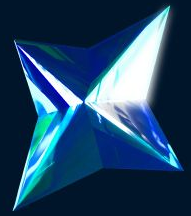
`action` – The **URL** where the form data will be sent

`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">
```

`<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.

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

input attributes

– customize **<input>**

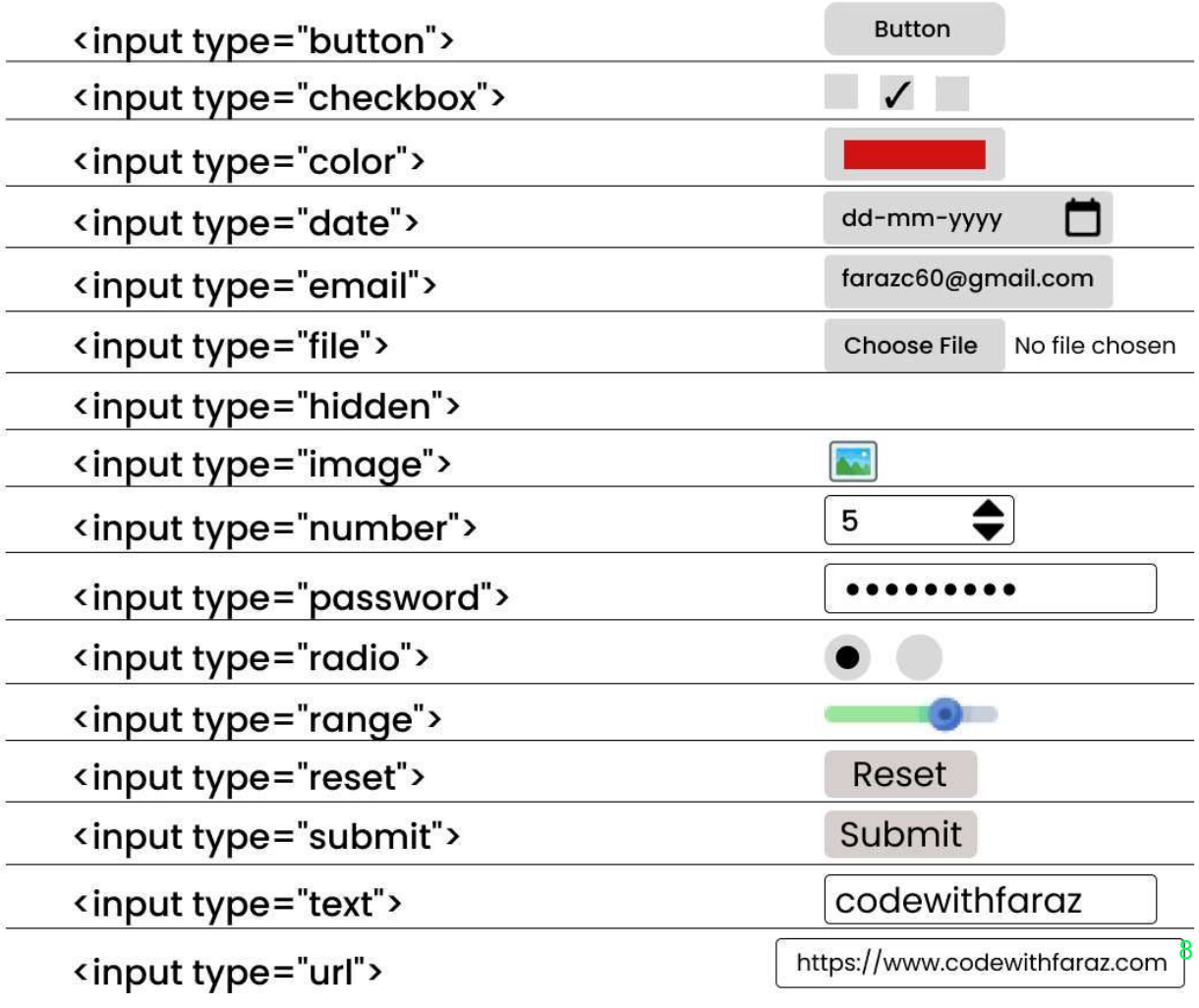
- These attributes add features and behaviors to the **<input>** elements.
- Attribute examples include **size, value, maxlength, required,** and many more.

```
<input
    type="text"
    name="bro"
    required
    minlength="2"
    maxlength="8"
/>
```

input types

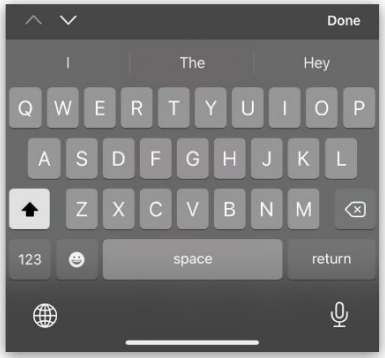
Why is it important to use the proper type?

- User Experience
- Validation
- Accessibility
- Styling and Behavior
- Improved Semantics

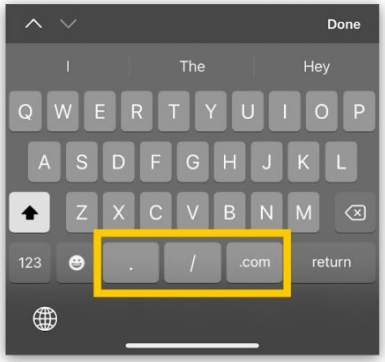


<input inputmode="..">

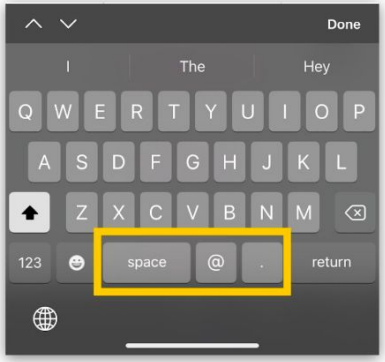
"none"



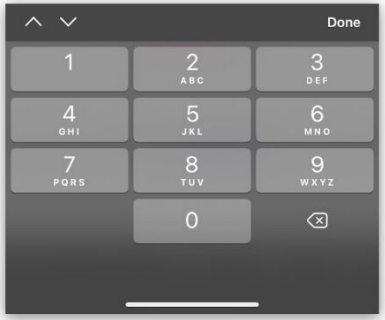
"text"



"url"

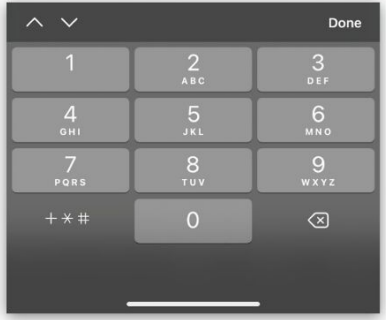


"email"

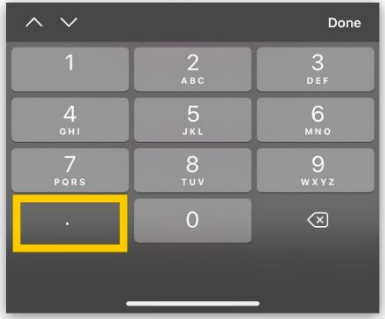


"numeric"

"search"



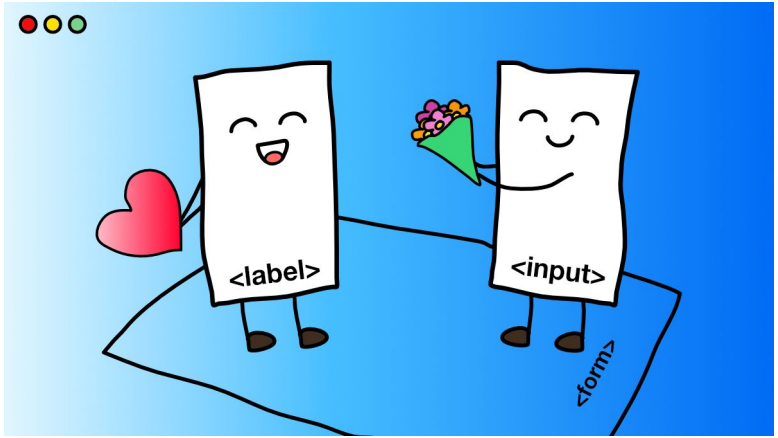
"tel"



"decimal"

label

– description for a form element



- 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="fruit" value="apple" id="apple">
<label for="apple"> Apple </label>
```

```
<label>
    Apple
    <input type="radio" name="fruit" value="apple">
</label
```

Checkbox Usage Rules

– used to let a user select **one or more** options of a limited number of choices.

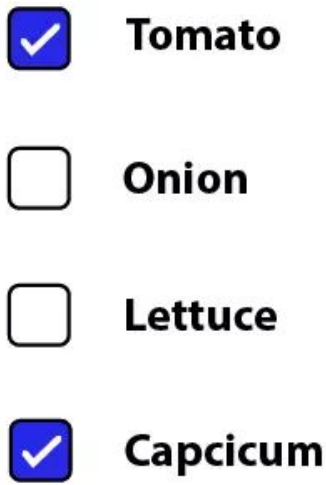
```
<input type="checkbox" name="vegetables" value="tomato"
class="input-checkbox" id=”checkbox-tomato”>
```

Required Attributes:

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

Optional:

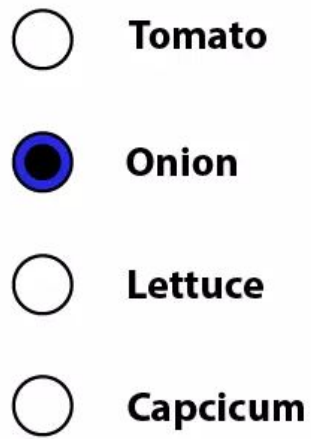
value – Must be unique for each checkbox in a group



Radio Button Usage Rules

– used to let a user select **one** option of a limited number of choices.

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



Required Attributes:

name – Should be the same for all radio buttons in a group

value – Must be unique for each radio button

textarea

- **Allows multi-line text input**

- By default, it is resizable. This can be disabled using CSS

textarea

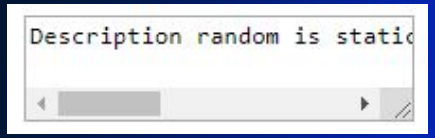
```html
<div class="form-row">
    <textarea name="basic-textarea" id="basic-textarea"
    class="form-control" cols="30" rows="10"></textarea>
    <label for="basic-textarea">Basic Textarea</label>
</div>
```

Attributes:

cols – specifies the width of the textarea in terms of the number of text columns

rows – specifies the number of text lines it can display at a time (height)

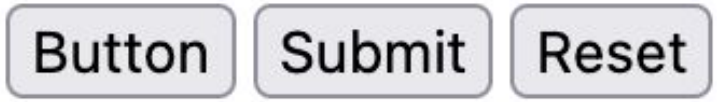
instead of using this attributes, you might set the width and height in your CSS



if you want to prevent users from resizing:

```css
textarea {
    resize: none;
}
```

button

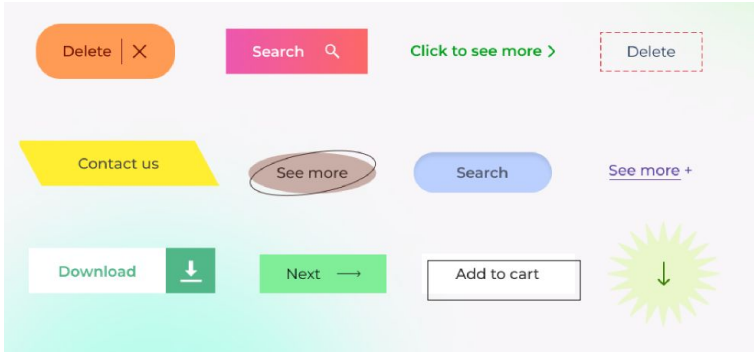


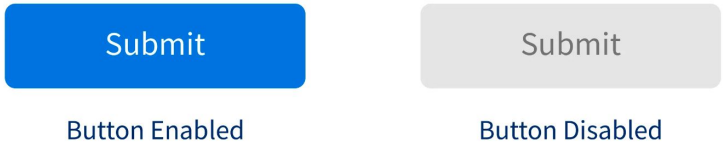
`button` – A generic button for any action

`submit` – Submits the form data to the server

`reset` – Resets the form data to its initial state

The **`disabled`**, attribute can be used to prevent the form from being submitted

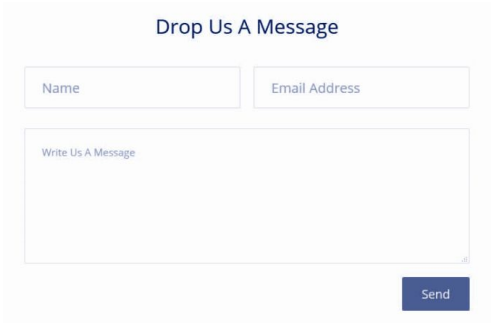




Button Enabled

Button Disabled

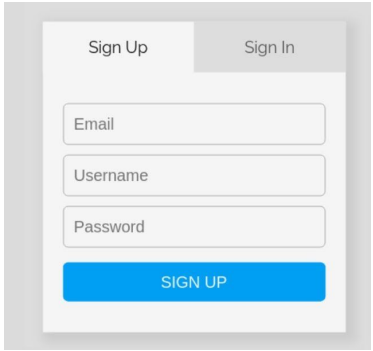

2


Drop Us A Message
Name
Email Address
Write Us A Message
Send


Captcha:
N H T T 1 8 7
Reset
Enter Captcha


What is your rate?
Please share your opinion about the product


Form
Pizza
Pasta
Cookies
I love chocolate
Submit
Reset form


Sign Up
Sign In
Email
Username
Password
SIGN UP


Message...


00 : 00 : 00 : 000
Pause
Start
Reset


170
1800

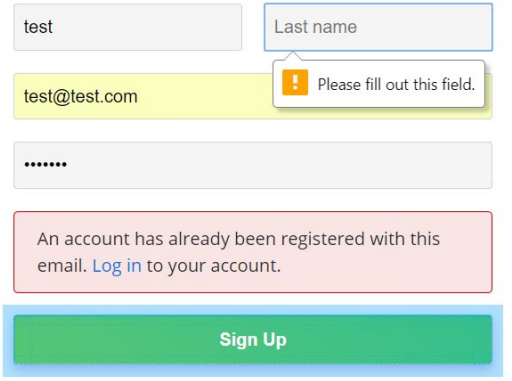

Enter your mail
Subscribe

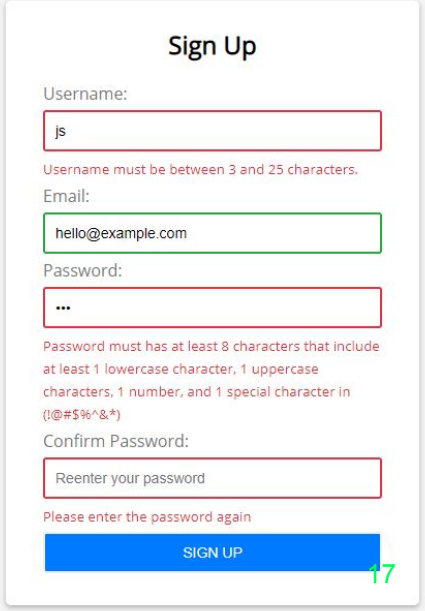
form validation

- Validates form data according to specific rules

- Methods for validation include:
  - **html**
  - **js**

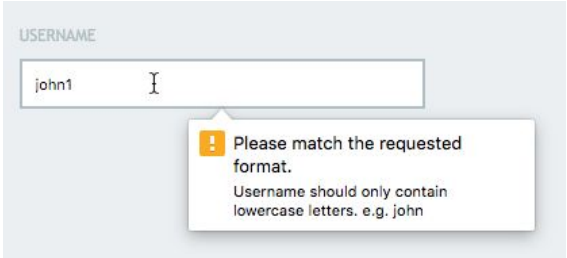
1. Use the appropriate **input type**
2. Add the **required** attribute where necessary
3. Use the **pattern** attribute for custom validation rules
4. Style inputs and validation messages using pseudo-classes like **:invalid**, **:valid**



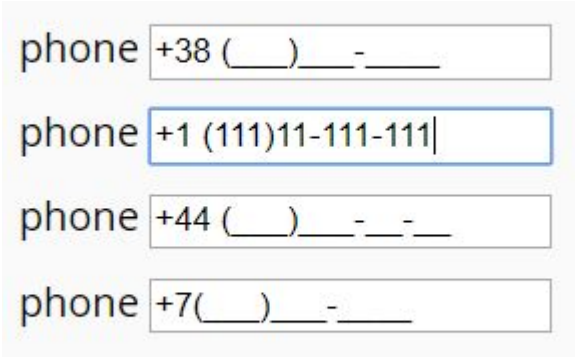


input attribute pattern

— helps ensure data is entered in the correct format
— provides a prompt or error message if the data doesn't match the pattern
— ask ChatGPT to help you create the right pattern



`pattern="[A-Za-z]{5}"`
This pattern requires 5 characters, allowing both uppercase and lowercase Latin letters.




```
<input type="email" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$">
```

appearance

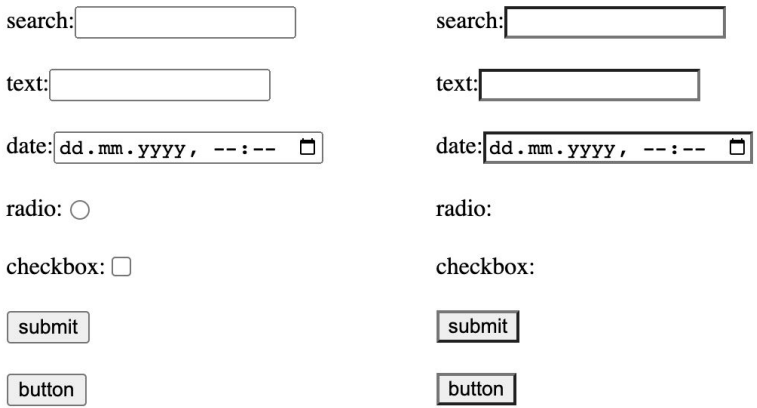
appearance: none – resets the appearance of an element to a consistent style across all browsers and operating systems

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

```
appearance: searchfield;
```

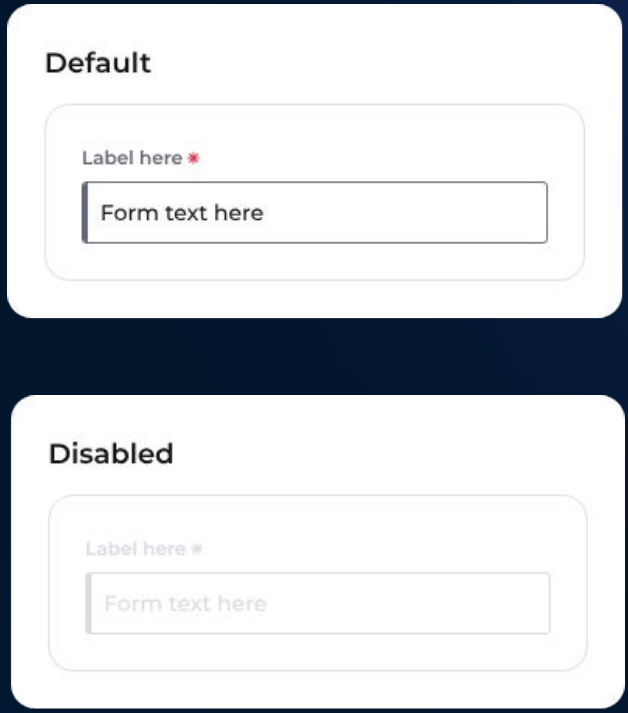
The default styling applied by the browser:

```
appearance: auto;
```



User interface pseudo-class selectors

- `:disabled` – element that is disabled (using the `disabled attribute`)
- `:enabled` – elements that do not have the `disabled attribute`
- `:default` – the element that is the default among a group of similar elements



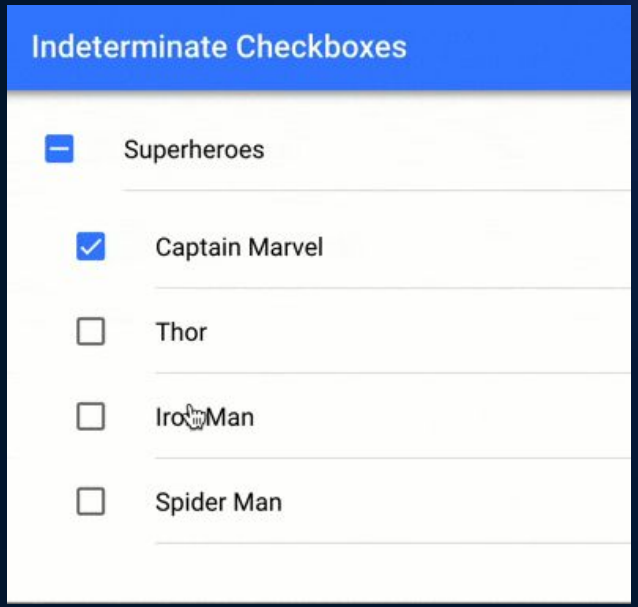
User interface pseudo-class selectors

- `:checked` – `radio or checkbox` elements that have the `checked` attribute or have been selected by the user

  demo

- `:indeterminate` – `radio or checkbox` elements that are in an intermediate state, without the checked attribute or not selected by the user

  demo



User interface pseudo-class selectors

- `:in-range` – highlights an element when the user's input falls within the specified range

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

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

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

User interface pseudo-class selectors

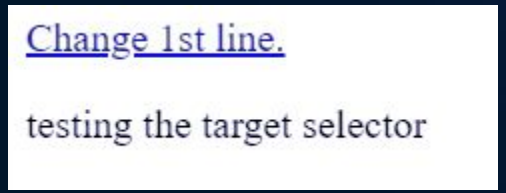
- `:read-only` – targets elements with the readonly attribute (indicating that the user cannot change the element's value)
- `:read-write` – targets elements that do not have the readonly attribute

User interface pseudo-class selectors

- `:target` – applies styles to elements that are referenced by a fragment identifier (ID) in a URL.

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

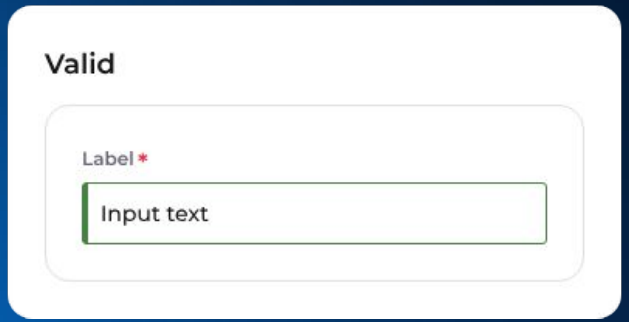
p:target {
    background-color: red;
    color: white;
}
```



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.



styling valid & invalid

Data entered correctly:

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

Data entered incorrectly:

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

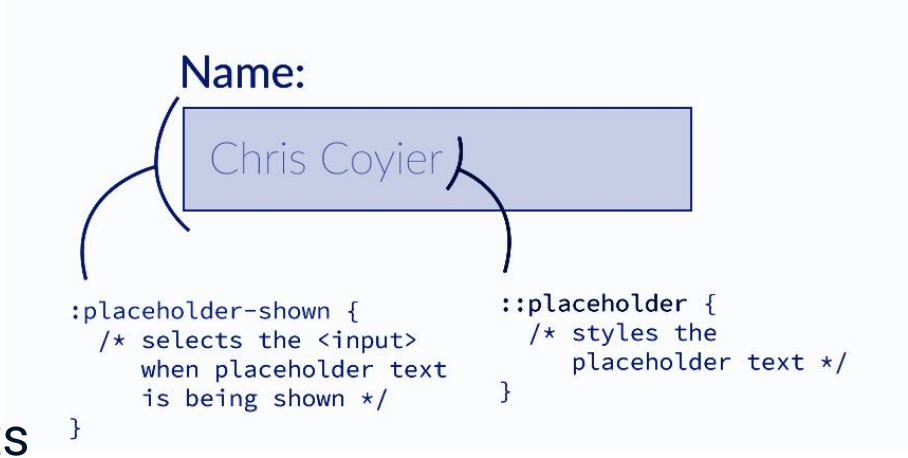
Note: This works even if the input is empty, because an empty value is also considered invalid.

Pseudo-elements

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

Pseudo-class

`:placeholder-shown` – represents any **`input`** or **`textarea`** element that is currently displaying placeholder text.



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)

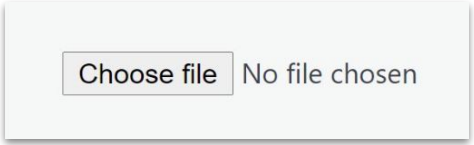
styling valid & invalid

Considering whether the input is focused:

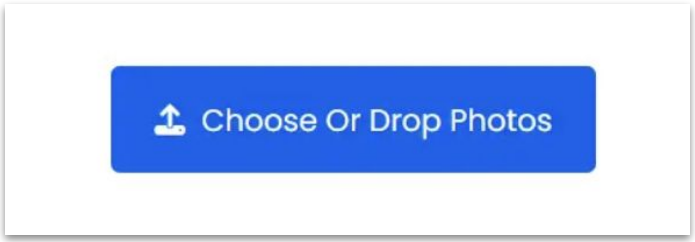
```
input:invalid:not(:placeholder-shown):not(:focus)
{border-color: red}
```

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.

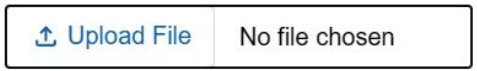
styling file upload input



- Styled using pseudo-elements.
- Icons can be added using pseudo-element or by adding wrapper for input

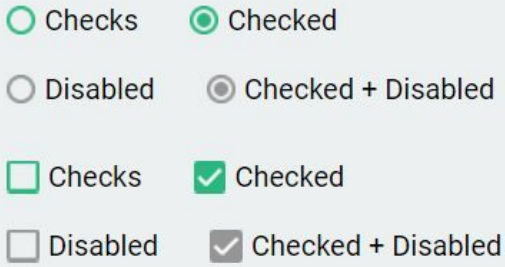


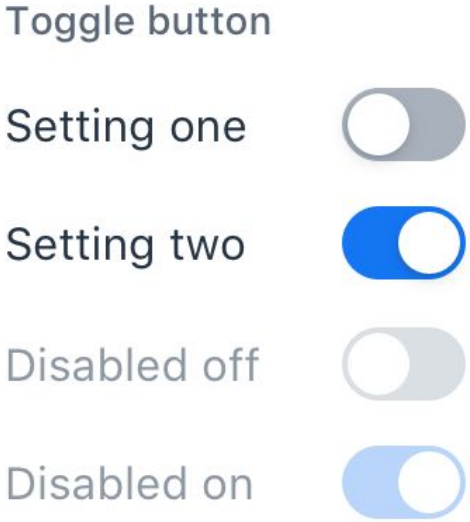
`::file-selector-button` – Styles the button for `<input type="file">`




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

multiple - attribute allows to add more than one file
accept - attribute allows to specify acceptable file types





`<input>`

radio & checkbox

- radio and checkbox elements cannot be easily styled directly
  (though it is possible)
- Typically, the **`input`** element is hidden, and the associated **`label`** is styled instead
- Always consider the **`checked`** and **`disabled`** states for styling

```
<input type="checkbox" name="name1" value="value1">
<input type="checkbox" name="name2" value="value2" checked>
<input type="radio" name="name" value="value01">
<input type="radio" name="name" value="value02" checked>
```

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 {}
```

Accessible input hiding

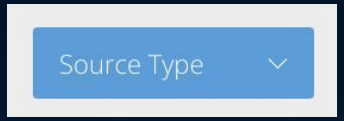
```
<input type="radio"class="visually-hidden">
```

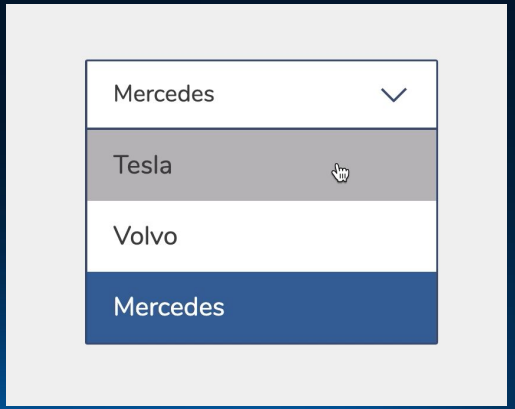
~~display: none;~~

```
.visually-hidden {
  clip-path: inset(50%);
  height: 1px;
  width: 1px;
  position: absolute;
  white-space: nowrap;
  overflow: hidden;
}
```

select

- Dropdown menu allowing the selection of one or multiple options
- The dropdown button can be partially styled
- The dropdown cannot be natively styled





select

```
<select name="fruit">
  <option value="apple">Apple</option>
  <option value="orange">Orange</option>
  <option value="berry">Berry</option>
</select>
```

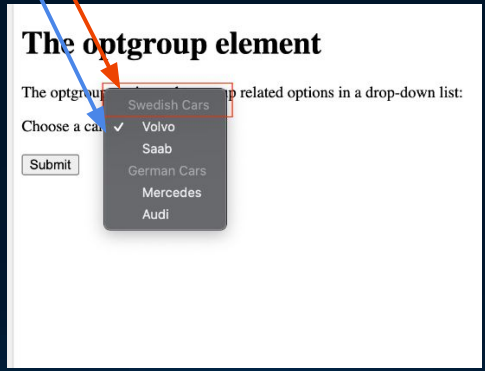
option – an element in a dropdown list

`value` – the value of the element that will be sent to the server; a required attribute

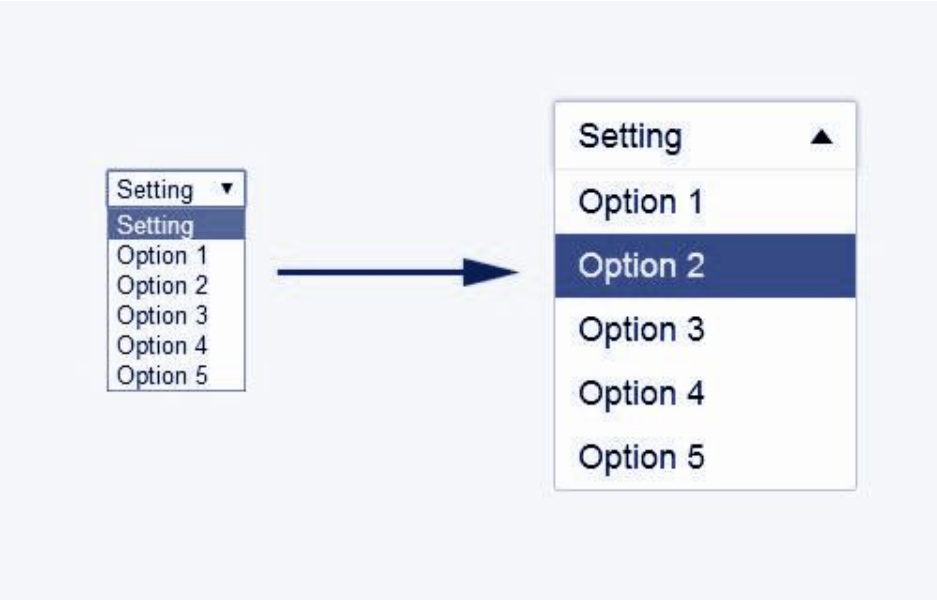
optgroup – an element used to group items in a dropdown list

`label` – an attribute that sets the title for a group (the title is not clickable)

```
<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select name="cars" id="cars">
    <optgroup label="Swedish Cars">
      <option value="volvo">Volvo</option>
      <option value="saab">Saab</option>
    </optgroup>
    <optgroup label="German Cars">
      <option value="mercedes">Mercedes</option>
      <option value="audi">Audi</option>
    </optgroup>
  </select>
  <br><br>
  <input type="submit" value="Submit">
</form>
```



styling `select`



Using JavaScript libraries.

They create a separate element next to the **`<select>`** for styling purposes, while hiding the original **`<select>`** element.
When an option is selected, the choice is passed to the hidden **`<select>`**, which is then submitted with the form.

For example, **jQuery Nice Select**.

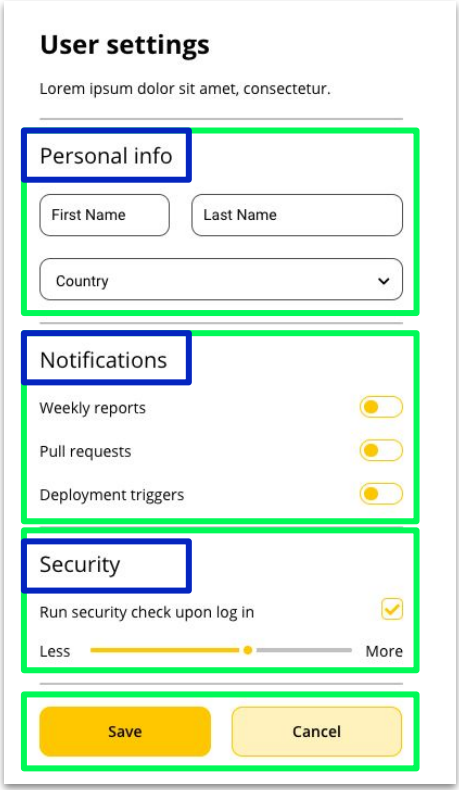
demo

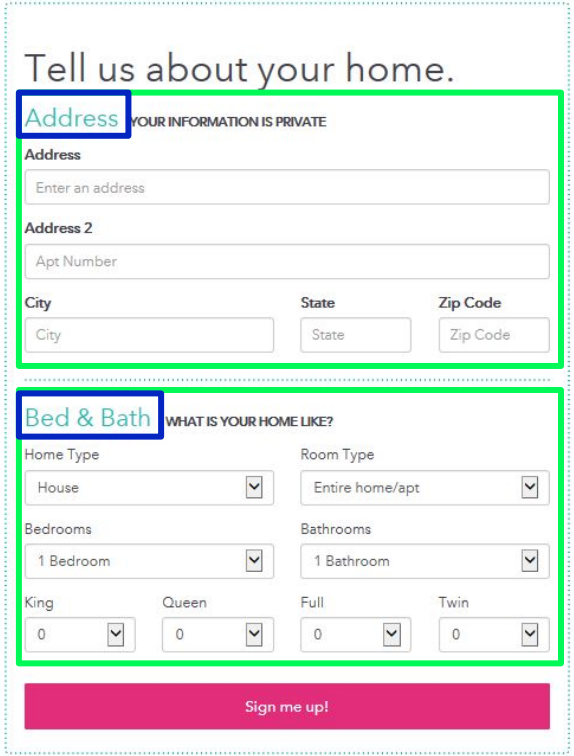
Grouping Elements

- **`<fieldset>`** – Groups related elements within a form

- **`<legend>`** – A caption for the fieldset
  – The caption is offset upwards, which can make it difficult to align with other elements
  – It does not respond to **`flex`** layout properties applied to the **`fieldset`**


Funkastic
EVENT
Registration
Friday 15th & Saturday 16th February @ 8pm
YOUR DETAILS
Name
Country
Phone
State
Email
Mem No.
TICKETS
Date
Fri
Sat
Delivery
Email
Collect
Quantity
Promotional Code
Deadline for Registration is Friday 4th January
eTickets will be issued to your email address two weeks before the event.
Gates will open three hours before the event gets started.
Please remember to bring identification if collecting your ticket from our events desk.
LET'S GET THIS
PARTY
STARTED
SUBMIT

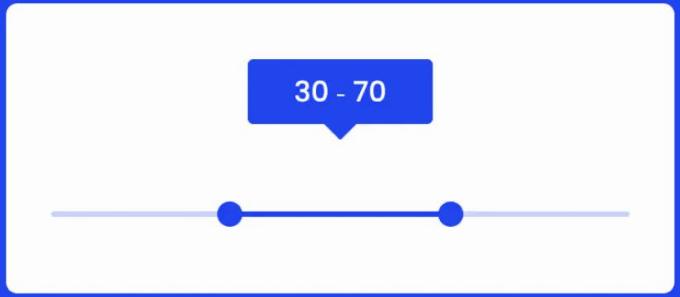

User settings
Lorem ipsum dolor sit amet, consectetur.
Personal info
First Name
Last Name
Country
Notifications
Weekly reports
Pull requests
Deployment triggers
Security
Run security check upon log in
Less
More
Save
Cancel

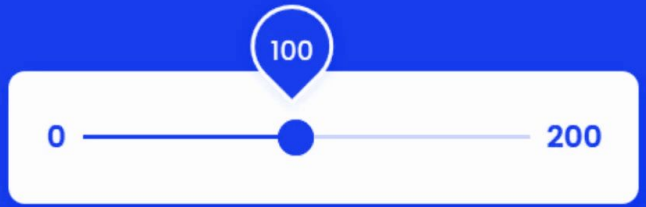

Tell us about your home.
Address
YOUR INFORMATION IS PRIVATE
Address
Enter an address
Address 2
Apt Number
City
State
Zip Code
City
State
Zip Code
Bed & Bath
WHAT IS YOUR HOME LIKE?
Home Type
House
Room Type
Entire home/apt
Bedrooms
1 Bedroom
Bathrooms
1 Bathroom
King
Queen
Full
Twin
0
0
0
0
Sign me up!

input range

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

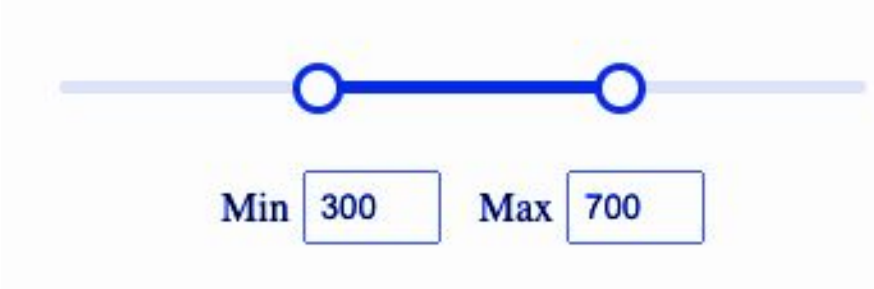
For styling, consider using a JS plugin like `AlRangeSlider`.





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 type="number">
        </div>
        <div class="range-max"> Max
            <input type="number">
        </div>
    </div>
</div>
```

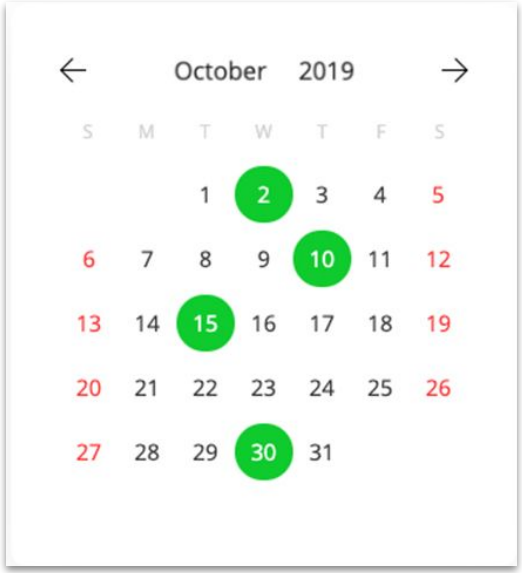


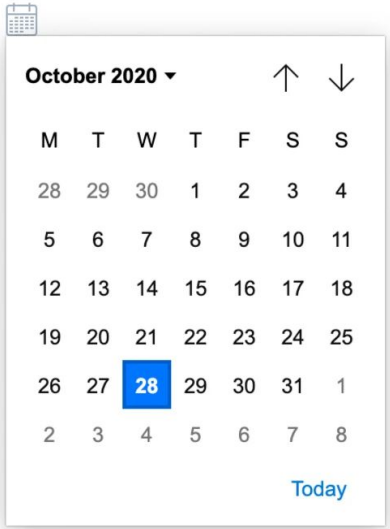
styling datepicker

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

— Cannot be styled natively.

— Use JS plugins like **Air datepicker** for custom styling

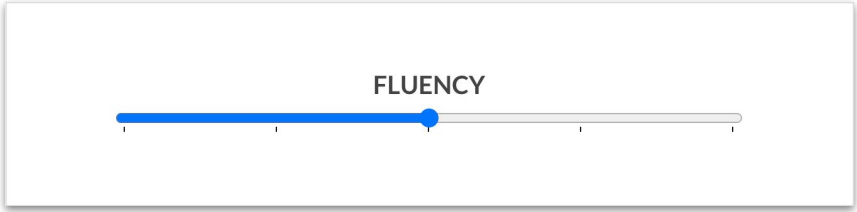


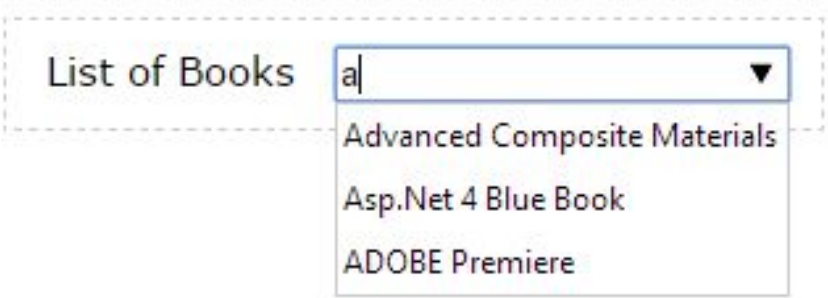


datalist

— 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:

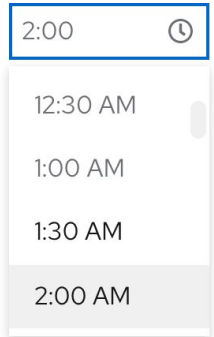
text, search, range, color, number, date, time, tel, email

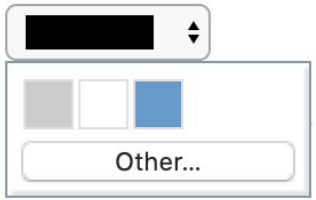


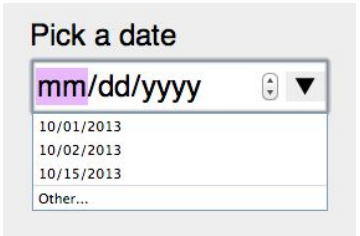


datalist

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







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.

progress

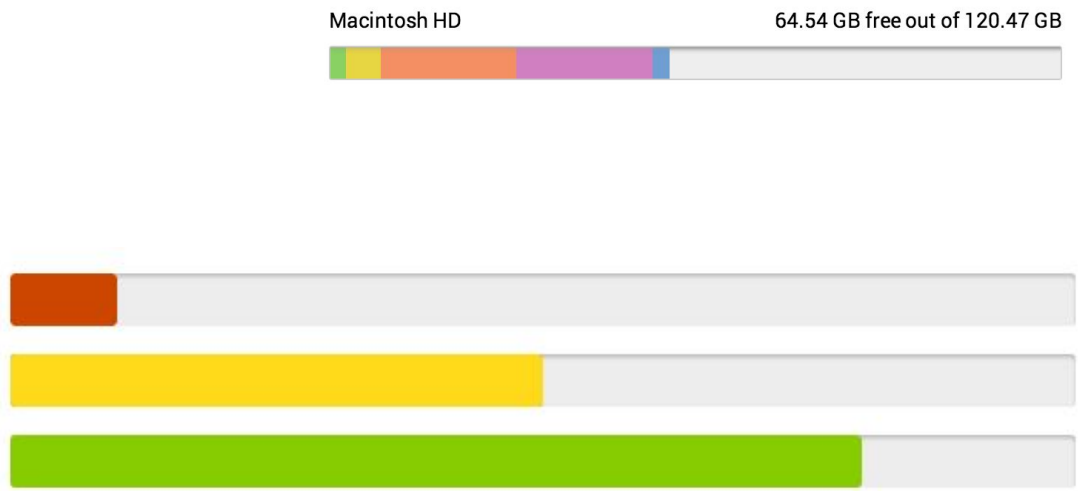
```
<label for="progress-bar">Downloading</label>
<progress
  id="progress-bar"
  value="57"
  max="100"
  title="percentage">
</progress>
```

max – the maximum value (a number greater than 0)
value – the current value (a number between 0 and **max**)

meter

– displays a scalar measurement within a known range

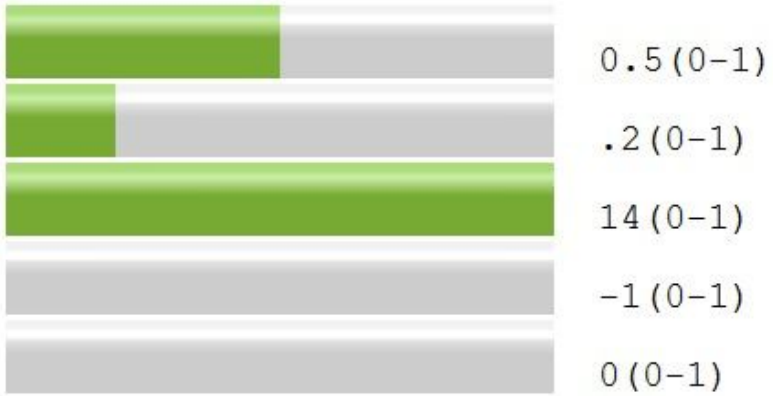
- Password strength
- Battery level
- Sound volume



meter

```
<label for="vote"></label>
<meter
    id='vote'
    value='75'
    min='0'
    max='100'>
</meter>
```

value — current value on the scale
min — minimum value
max — maximum value
low — lower threshold value
high — upper threshold value
optimum — optimal threshold value



styling progress

```
progress {
  appearance: none;
}

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

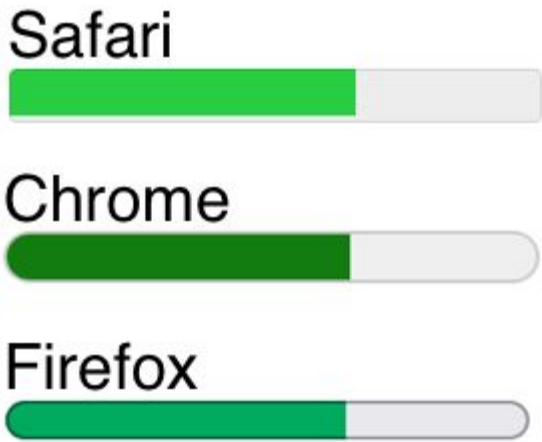
progress::-webkit-progress-value {
   background-color: green;
}
```

meter

```
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;
}
```



Quality Criteria for HTML Course

❤️Mandatory for passing the course

💛Required for the highest grade

💚Optional

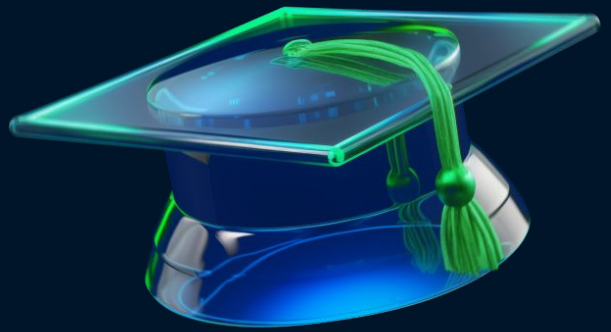
❤️ 1.19. Each form element should have a <label>.

💛 1.20. Resizing <textarea> should not break the layout.

❤️ 1.21. Basic HTML form validation should be implemented.

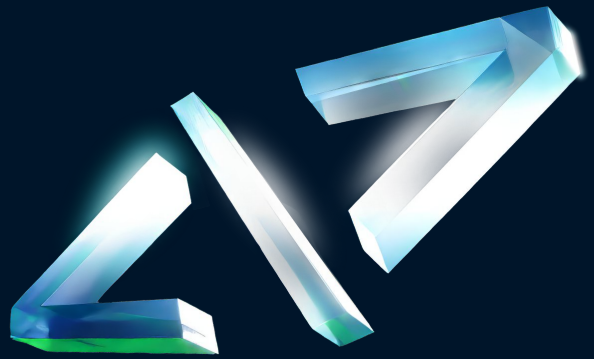
Summary

1. The importance of forms on websites.
2. Components of forms.
3. How to solve styling issues with form elements.
4. How to validate form data.
5. Interactive tags outside of forms.

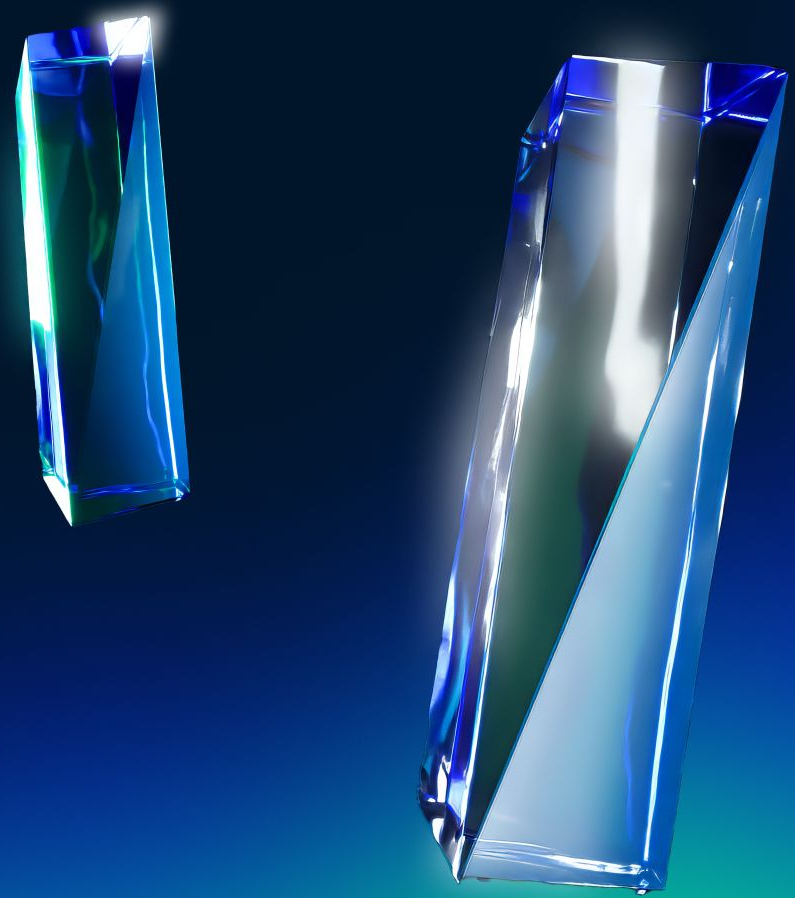


Homework

1. Complete the Lesson about forms elements
2. Markup and Style Forms in Binabox:
    - Specify Button Types
    - Validate Your Forms



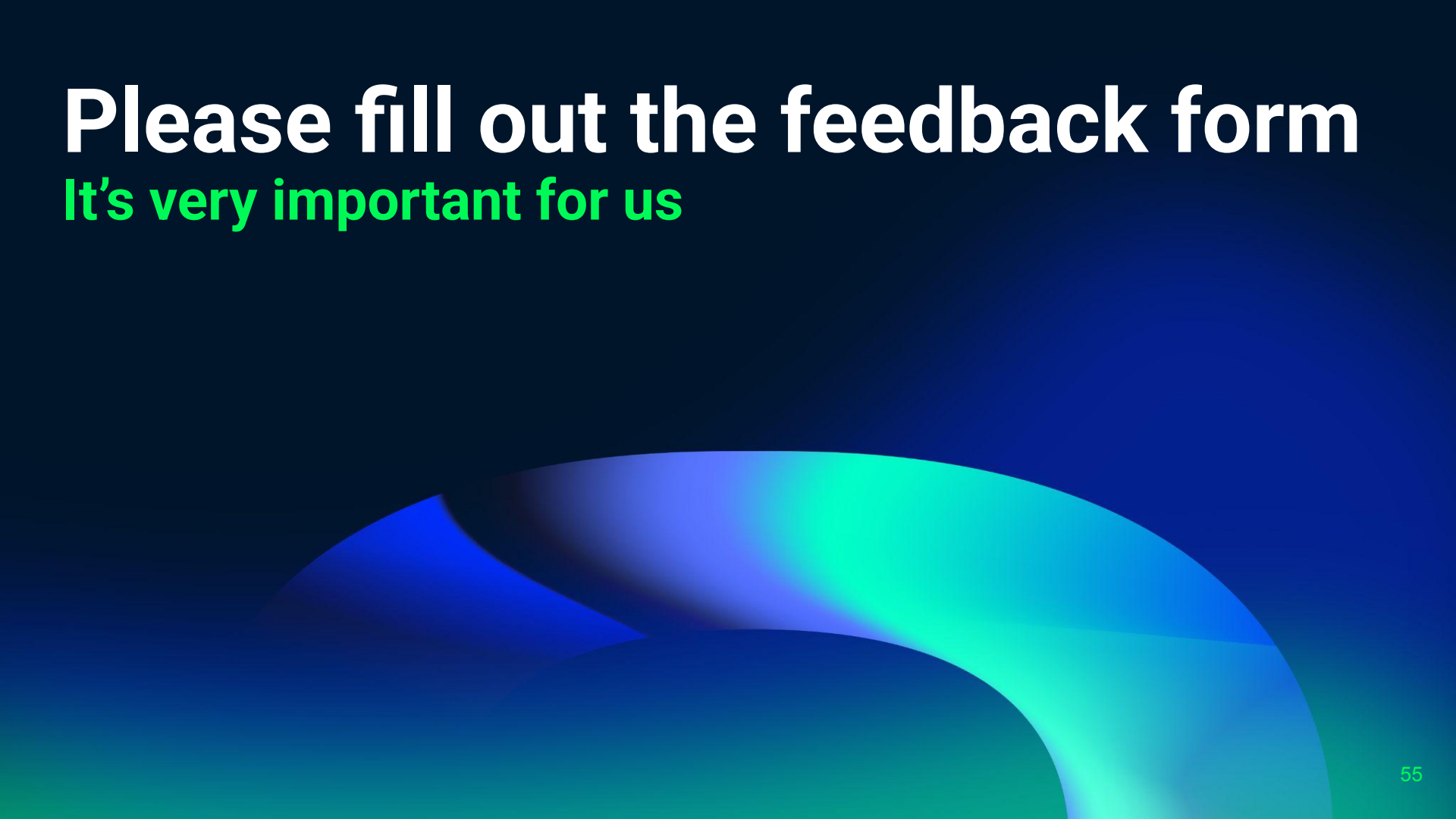
BRO Academy



QUESTIONS?

Please fill out the feedback form

It’s very important for us



THANK YOU!

Have a good evening!