Lesson 10: Building Forms

- use forms to: acquire user input
  process requests

Initializing a Form

- <form> = to add a form to a page
  identifies where on the page control elements will appear

  → (eg. <form action = "/login" method = "post">
  ...
  </form>)

- form attributes: action
  method] most common

- action attribute = contains the URL to which info included w/in the form will be sent for processing by the server

  method attribute = HTTP method browsers should use to submit form data

Text Fields & Textareas

- used to collect text/string-based data: text fields
  textareas

- <input> = to obtain text from users → single line text field
  self-contained

  - format: <input type = "type" name = "name">

  - attributes: type = to define what type of info is going to be captured w/in the control
    name = name of the control
    submitted w/ input data to server

  - new html5 type attribute values:

color	date	datetime
email	month	number
range	search	tel
time	url	week

  ] default fallback: text value

  → (eg. <input type = "date" name = "birthday">)

  → (eg. input type = "tel" name = "phone-number">)

- <textarea> = to capture text-based data → large text spanning multiple lines

  - format: <textarea name = "name"> text that appears in textarea box </textarea>

text that appears...

- sizing attributes: **cols** = width in terms of avg character width
  rows = height in terms of # of lines of visible text] < css width & height properties

Multiple Choice Inputs & Menus

- **radio buttons** = to select one option from a small list of options

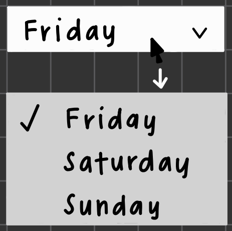
→
```
eg. <input type="radio" name="sex" value="female" checked> female
    <input type="radio" name="sex" value="male"> male
```

  - attributes: **value** = to distinguish choices
    checked = to preselect a button for users

- **checkboxes** = to select one or more options
  same format as radio buttons except for the type attribute value

→
```
eg. <input type="checkbox" name="
    <input type="checkbox" name="
    <input type="checkbox" name="
```

- **drop-down lists** = to select one option from a long list of options
  <select> = to wrap all the menu options
- **<option>** = to mark up each menu option



→
```
eg. <select name="day">
        <option value="sunday" selected> Sunday </option>
        <option value="Monday"> Monday </option>
          ⋮
        <option value="saturday"> Saturday </option>
    </select>
```

  - **selected attribute** = to preselect an option for users

- **multiple attribute** = to choose more than one option from a drop-down list
  options aren't displayed like a drop-down list
  user must hold down shift key + click to select

Friday
Saturday
Sunday

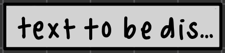
→
```
eg. <select name="day" multiple>
        <option value="sunday" selected> Sunday </option>
        <option value="Monday"> Monday </option>
          ⋮
        <option value="saturday"> Saturday </option>
    </select>
```

Form Buttons

- used to process form data: submit input
  submit button

- submit input = a submit button made using <input>
  self-contained → can't wrap other content

  - format: <input type="submit" name="submit" value="text that appears w/in the button">



- submit button = a submit button made using <button>
  more control over structure & design > submit input
  can wrap other content
  default type = submit

  - format: <button name="submit">
    text to be displayed w/in button
    </button>

Other Inputs

- hidden input = to pass data to the server w/o displaying it to users
  info isn't displayed on page, but can be accessed by page source code
  for non-sensitive/secure info that's not pertinet to user but is helpful when processing the form

  → (eg. <input type="hidden" name="tracking-code" value="abc-123">)

- file input = to allow users to add a file to a form
  difficult to style file type input w/ css → browser default input style is used

  - format: <input type="file" name="name">

Organizing Form Elements

- organize forms & guide users how to properly complete them w/: labels
  fieldsets
  legends

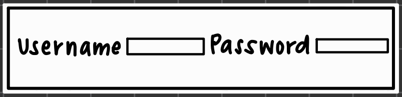
- <label> = captions/headings for form controls
  to include text that describes input/control

  - connect label to form control: for attribute value = form control id attribute value
    wrap form control w/in element

  → (eg. <label for="username"> Username </label>
    <input type="text" name="username" id="username">)

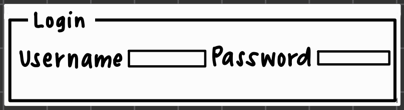
  → (eg. <label>
    <input type="radio" name="sex" value="female" checked> female
    </label>
    <label>
    <input type="radio" name="sex" value="male"> male
    </label>)

- <fieldset> = to group form controls & labels into organized sections
  block-level element
  includes a default border outline



→
```
eg. <fieldset>
      <label>
         Username
         <input type="text" name="username">
      </label>
      <label>
         Password
         <input type="text" name="password">
      </label>
    </fieldset>
```

- <legend> = caption/heading for <fieldset> element
  to describe form controls that fall w/in fieldset
  must come right after opening <fieldset> tag
  appears in top-left of fieldset border



→
```
eg. <fieldset>
      <legend> Login </legend>
      <label>
         Username
         <input type="text" name="username">
      </label>
      <label>
         Password
         <input type="text" name="password">
      </label>
    </fieldset>
```

Form & Input Attributes

- disabled attribute = turns off an element/control so it's not available for interaction/input
  disabled elements don't send data to server for form processing
  boolean attribute

- precedence: display overwrites hidden

→
```
eg. <label>
       Username
       <input type="text" name="username" disabled>
    </label>
```

- placeholder attribute = hint/tip w/in form control of an <input> or <textarea> element
  disappears once control is clicked in/gains focus
  shows form input format

Email Address
name@domain.com

→
```
eg. <label>
        Email Address
        <input type="email" name="email-address"  placeholder="name@domain.com">
    </label>
```

- placeholder vs. value : value text stays in place unless user manually deletes it
- specific use : placeholder → providing suggestions
  value → pre-populating data

- required attribute = enforces an element/form control must contain a value upon being submitted to the server
  error message displays if element/control doesn't have a value → requests user to complete required field
  also checks if given input is a valid value of that control's type
  boolean attribute

Email Address
Submit
Email Address
Submit
Please fill out this field.

→
```
eg. <label>
        Email Address
        <input type="email" name="email-address" required>
    </label>
```

- styling pseudo-classes : :optional :required for invalid elements & form controls

- additional attributes :

accept	autocomplete	autofocus
formaction	formenctype	formmethod
formvalidate	formtarget	max
maxlength	min	pattern
readonly	selectiondirection	step