香 港 中 文 大 學
The Chinese University of Hong Kong

CSCI2720 - Building Web Applications

Lecture 6: Forms

Dr Colin Tsang

Outline

- The use of forms
- Form control elements
- Form submission
- Form CSS
- Form validation

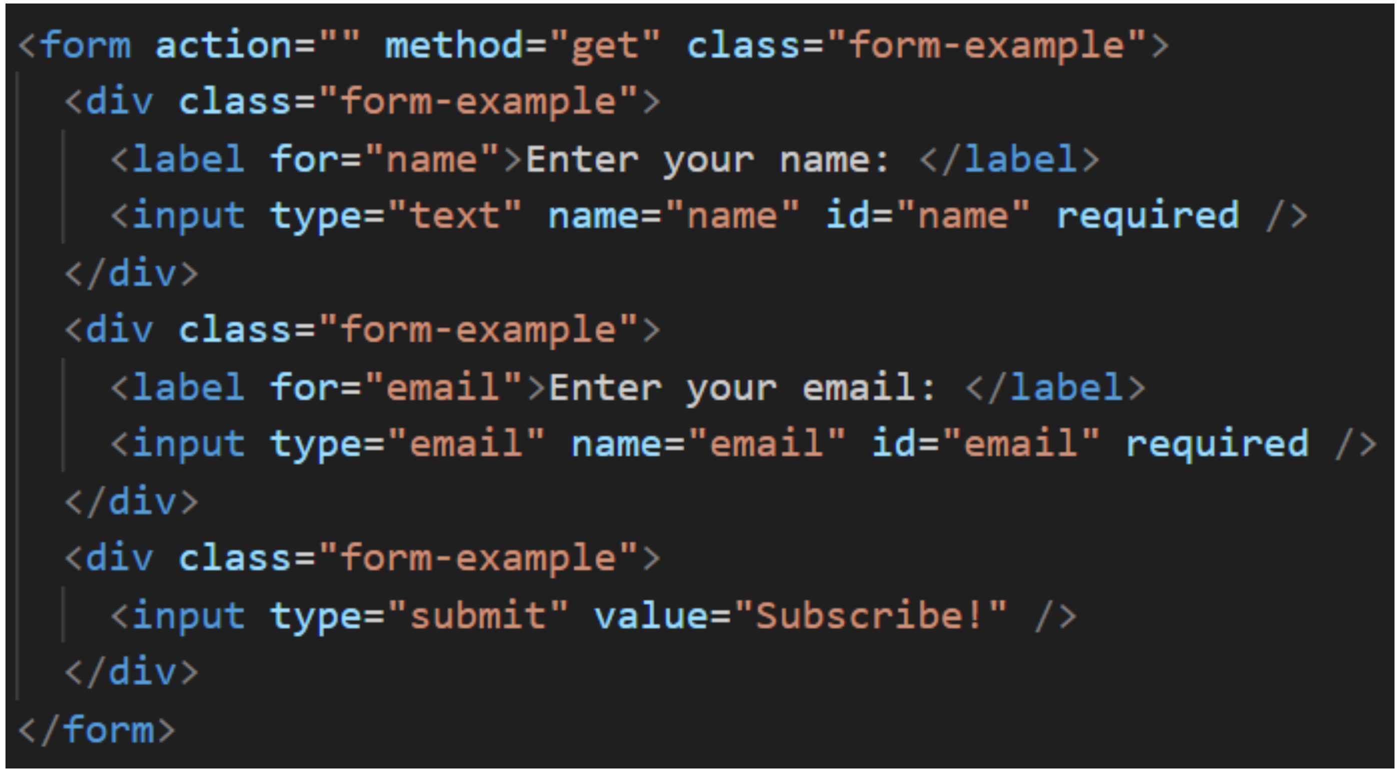
The use of forms on web

- HTML can allow user to provide input.
- The user input can:
  - Submission back to the server
  - Interaction with scripts

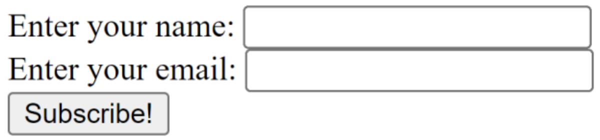
- The web form (or HTML form) has elements for user interface building, easily skinnable with CSS.
- Everything should be enclosed in the **<form>** element.

The use of forms on web

- A web form can look like this:



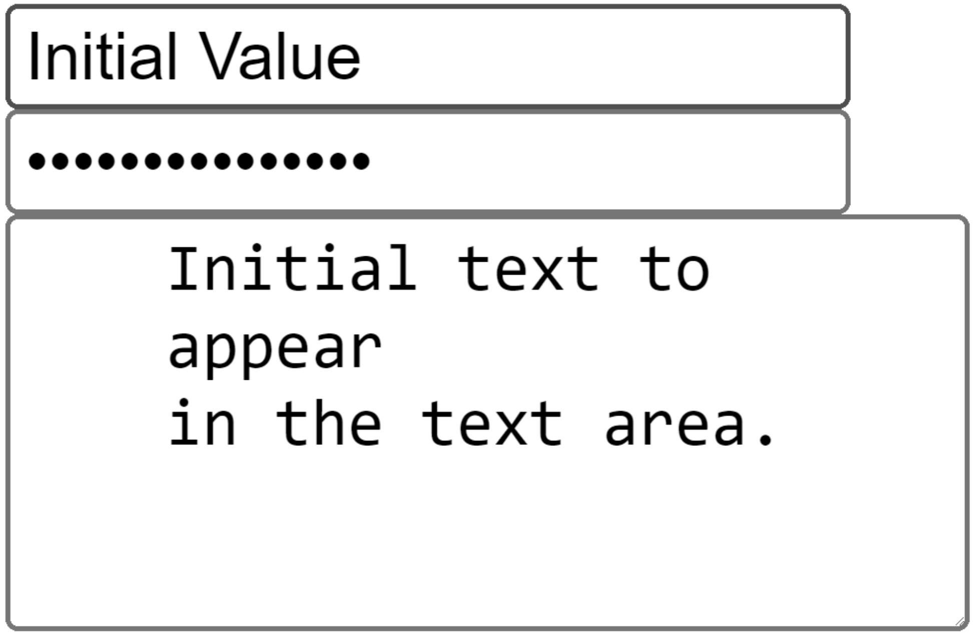
```html
<form action="" method="get" class="form-example">
  <div class="form-example">
    <label for="name">Enter your name: </label>
    <input type="text" name="name" id="name" required />
  </div>
  <div class="form-example">
    <label for="email">Enter your email: </label>
    <input type="email" name="email" id="email" required />
  </div>
  <div class="form-example">
    <input type="submit" value="Subscribe!" />
  </div>
</form>
```



- See: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form

Form control elements

- Text input fields:
  - **<input type="text">** for single line input
  - **<input type="password">** for passwords input
  - **<textarea>** for multiple lines



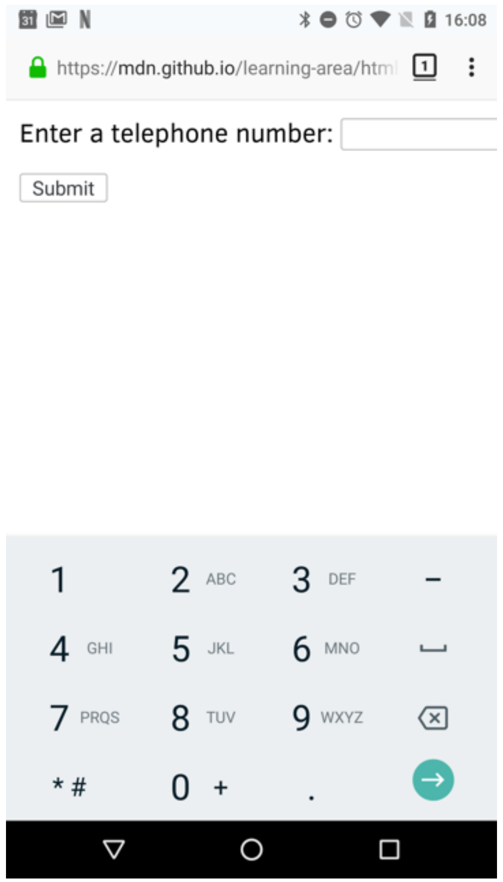
Initial Value

•••••••••••••••

```
     Initial text to
     appear
     in the text area.
```

Form control elements

- New input controls with validation or special effects
  - **<input type="email">** will ensure input is an email address
  - **<input type="search">** will provide a cross to cancel search
  - **<input type="tel">** will invoke a numpad input on mobile devices
  - **<input type="url">** will ensure the input is a URRL with correct syntax
  - **<input type="color">** will show a color picker
  - **<input type="date">** will show a date picker

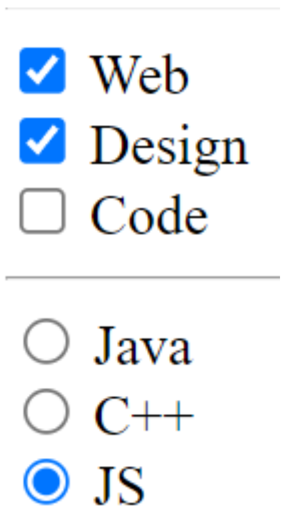


Enter a telephone number:

- More on **<input>**: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input

Form control elements

- List of option items
  - **<input type=“chatbox”>** is a box to be chosen
  - **<input type=“radio”>** is similar to checkbox, but grouped as a set with the name attribute and *allow only one* option

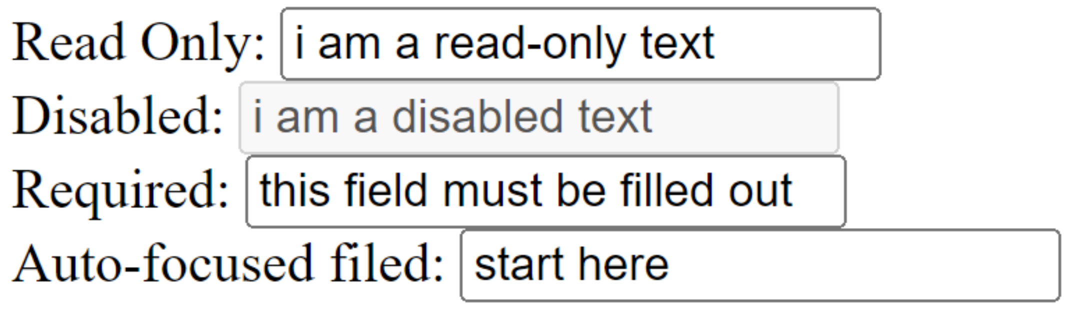


[✓] Web
[✓] Design
[ ] Code

( ) Java
( ) C++
(●) JS

Input attributes

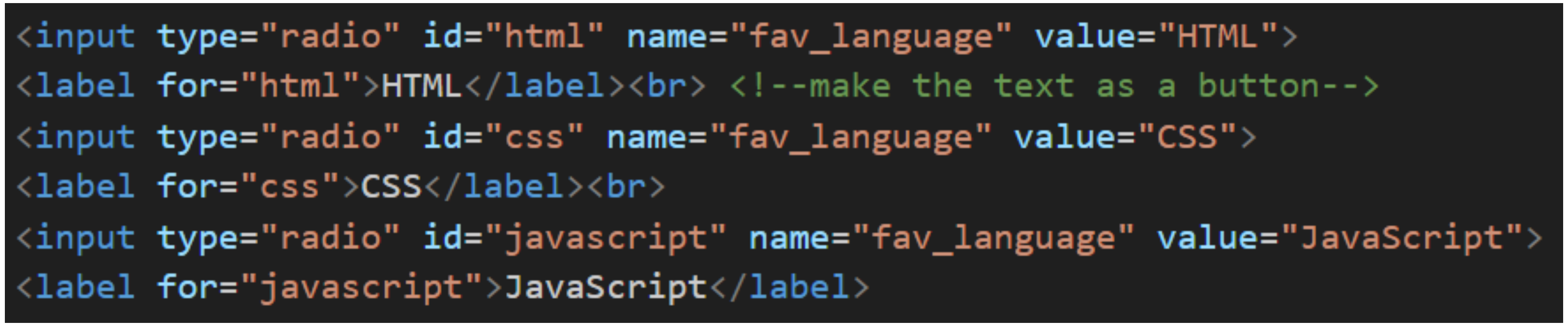
- Some attributes can help fine-tuning input controls
  - Value: initial values
  - Readonly: the field is read only
  - Disabled: the field is not available
  - Required: the filed must be filled out
  - Autofocus: the field gets focus when page loads



Read Only:	i am a read-only text
Disabled:	i am a disabled text
Required:	this field must be filled out
Auto-focused filed:	start here

Form labels

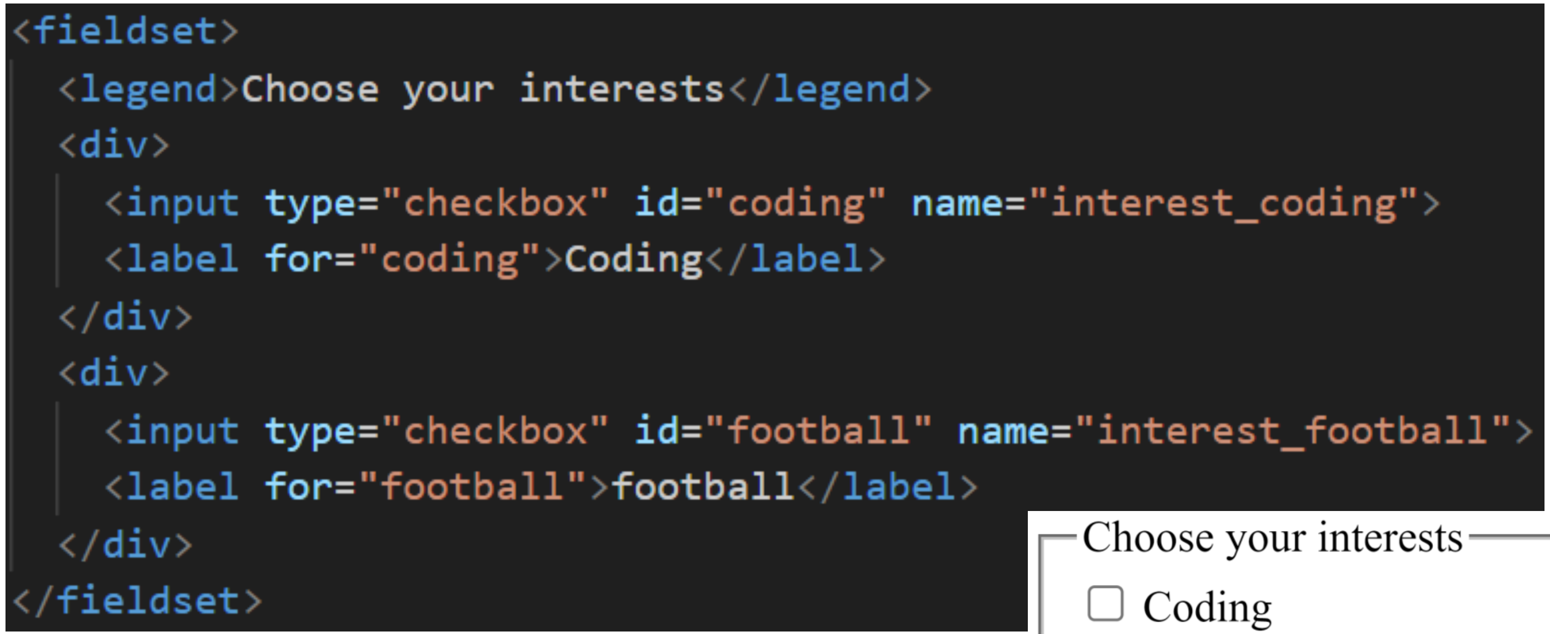
- **<label>** can be used to define any caption for form elements, such as radio buttons
- They should be carefully associated to the control elements using attribute *for*, so that
  - The attribute *for* corresponds to the attribute *id* for the **<input>** element.
  - Browsers allow easier selection of the control
  - Screen-readers understand the relationship correctly for user to focus on the input element



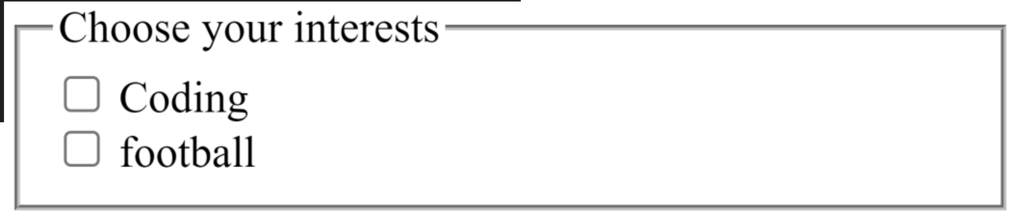
```html
<input type="radio" id="html" name="fav_language" value="HTML">
<label for="html">HTML</label><br> <!--make the text as a button-->
<input type="radio" id="css" name="fav_language" value="CSS">
<label for="css">CSS</label><br>
<input type="radio" id="javascript" name="fav_language" value="JavaScript">
<label for="javascript">JavaScript</label>
```

Form groups

- **<fieldset>** group items together, and allow using **<legend>** to show a group caption



```
<fieldset>
  <legend>Choose your interests</legend>
  <div>
    <input type="checkbox" id="coding" name="interest_coding">
    <label for="coding">Coding</label>
  </div>
  <div>
    <input type="checkbox" id="football" name="interest_football">
    <label for="football">football</label>
  </div>
</fieldset>
```



Form buttons and actions

- Simple button:
  - **<button type=“button”>** is a simple clickable button
  - The default type is *submit*
- Submit button:
  - **<button type=“submit”>** will by default run the form action
  - If the *action* attribute is defined in form, the form data is sent to the server scripts
  - Otherwise, the page will be reloaded
- Reset button:
  - **<button type=“reset”>** clears and restores all input controls in form

Form submission

- Traditional HTML form is for data submission to server-side scripts
  - e.g., to a PHP/ASP/JSP/Node.js script on the server
  - Data in the form will be sent as name-value pairs
  - Two possible methods:
    - **GET**: data is encoded into the URL as a query string
    - **POST**: data is embedded into an HTTP request body

- For **GET/POST** submission, the *name* attribute of form control elements must be set properly.

GET vs POST

GET	POST
Data only delivered ***inside the request URL*** in text form	Data can be encapsulated inside request body
Only limited amount of data (~2k) due to URL length	Data size is only ***limited by the body size*** (~1MB to 2GB) depending on the HTTP server
The request URL can be bookmarked, and is visible in the location bar ➔ ***security concern!***	Only URL can be saved but not the data in the body
The request URL would stay in the browser history, and can also be found on HTTP server log ➔ ***security concern!***	Only URL is recorded but not the data in the body

Submitting a form

- Nowadays, another approach is to pre-process the data on client-side with JavaScript, and to optionally return to server
  - Button click event -> processing, instead of using a form action
  - JS can help with form validation, or asynchronous submission (no refresh!)
    - More flexibility for developers for displaying helpful messages

- For JS, values in form controls are usually captures using *id* attributes
- Therefore, you may see the *id* or *name* attributes in examples depending on their purpose

Form CSS

- Form controls can be applied with usually CSS properties, e.g., width, padding, etc.
- To style particular types of form controls, the *attribute selector* can help:
  - **input[type="text"] input[type="button"]**
- To style particular states of form controls, some pseudo-classes are available:
  - **:hover :focus :active**
  - **:required :optional :enabled: disabled :read-only**
- https://developer.mozilla.org/en-US/docs/Learn/Forms/UI_pseudo-classes

Form validation

- HTML form should be validated before being processed by the server-side
  - To make sure the user put down correct data
  - To make sure incorrect item do not crash script
  - To lighten workload of the processing script

- Easily handled by JS
  - e.g., when user has finished a field (control elements lose focus), a check can be run, and warn the user if something is wrong
  - CSS has new pseudo-classes **:valid** and **:invalid** for easy styling

Further readings

- w3schools
- https://www.w3schools.com/html/html_forms.asp

- MDN web forms
- https://developer.mozilla.org/en-US/docs/Learn/Forms