香港中文大學
The Chinese University of Hong Kong

INPUTS AND FORMS

CSCI2720 2022–23 Term 1

Building Web Applications

Dr. Chuck-jee Chau
chuckjee@cse.cuhk.edu.hk

OUTLINE

- The use of forms
- Form control elements
- Submitting a form
- Form CSS
- Form validation

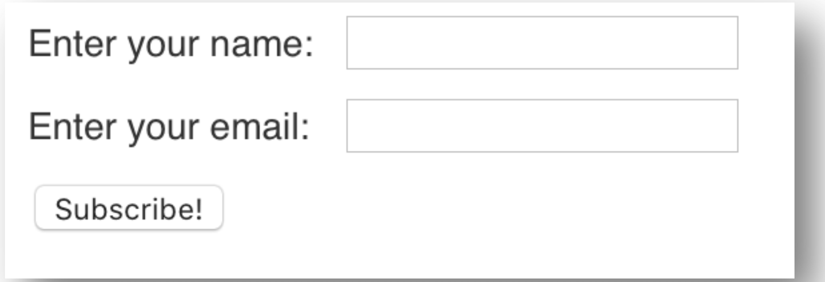
THE USE OF FORMS ON WEB

- HTML not only presents information, it also allows user to type or ***provide feedback***, for submission back to the server or 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:

```
<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>
```



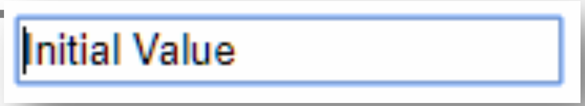
See: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form

FORM CONTROL ELEMENTS

https://codepen.io/chuckjee/pen/abLMJbE

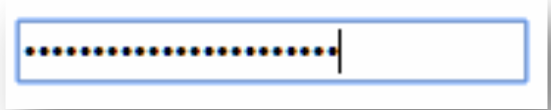
- Text input fields
  - `<input type="text">` for single line input
  - `<input type="password">` for passwords
  - `<textarea>` for multiple lines

```
<input type="text"
       name="LoginName"
       value="Initial Value">
```



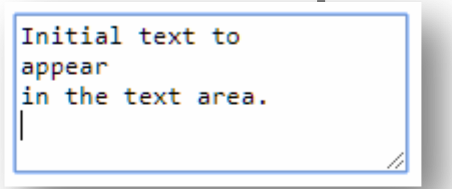
Initial Value

```
<input type="password"
       name="Pass">
```



••••••••••••••••••••••

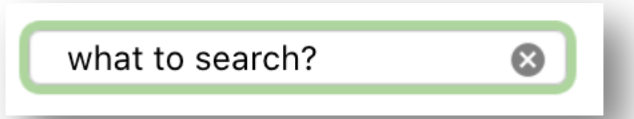
```
<textarea name="name"
          cols="25" rows="5">
Initial text to
appear
in the text area.
</textarea>
```

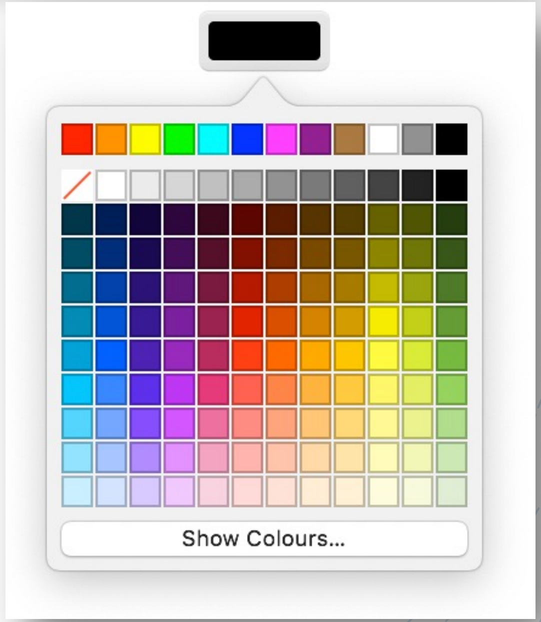


```
Initial text to
appear
in the text area.
```

FORM CONTROL ELEMENTS

- New input controls with validation or special effects
  - `<input type="email">` will ensure the 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 URL with correct syntax
  - `<input type="color">` will show a color picker
- More on `<input>`: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input

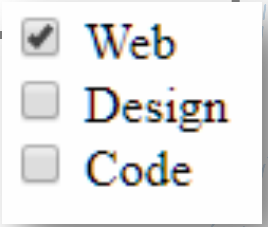




FORM CONTROL ELEMENTS

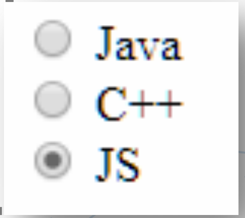
- List of option items
  - `<input type="checkbox">` is a box to be chosen
  - `<input type="radio">` is similar to checkbox, but grouped as a set with the **name** attribute and allow only one option

```
<input type="checkbox"
 name="web" checked> Web<br>
<input type="checkbox"
 name="design"> Design<br>
<input type="checkbox"
 name="code"> Code
```



☑ Web
☐ Design
☐ Code

```
<input type="radio" name="lang"
 value="Java"> Java<br>
<input type="radio" name="lang"
 value="C++"> C++<br>
<input type="radio" name="lang"
 value="JS" checked> JS
```



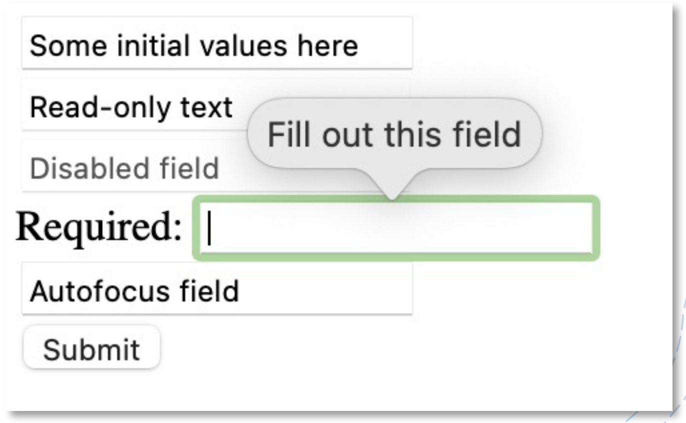
○ Java
○ C++
● JS

https://codepen.io/chuckjee/pen/zYEbNgE

INPUT ATTRIBUTES

```
<form>
  <input type="text" value="Some initial values here"><br>
  <input type="text" value="Read-only text" readonly><br>
  <input type="text" value="Disabled field" disabled><br>
  Required: <input type="text" required><br>
  <input type="text" value="Autofocus field" autofocus><br>
  <button type="submit">Submit</button>
</form>
```

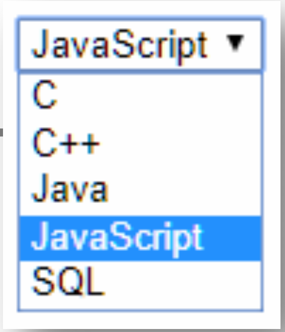
- Some attributes can help fine-tuning input controls
  - `value`: initial values
  - `readonly`: the field is read-only
  - `disabled`: the field is not available
  - `required`: the field must be filled out
  - `autofocus`: the field gets focus when page loads
- *See: https://www.w3schools.com/html/html_form_attributes.asp*



FORM CONTROL ELEMENTS

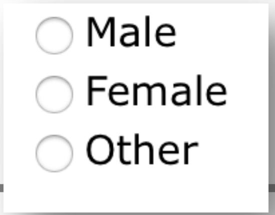
- List of option items
  - **`<select>`** and **`<option>`** can make a combobox (selectable list)
    - It is possible for allowing multiple selections

```
<select name="language">
  <option value="C">C</option>
  <option value="C++">C++</option>
  <option value="Java">Java</option>
  <option value="JavaScript" selected>
                JavaScript</option>
  <option value="SQL">SQL</option>
</select>
```



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
  - Browsers allow easier selection of the control
  - Screen-readers understand the relationship correctly for user to focus on the input element



See: https://www.w3schools.com/html/html_forms.asp -- The <label> Element

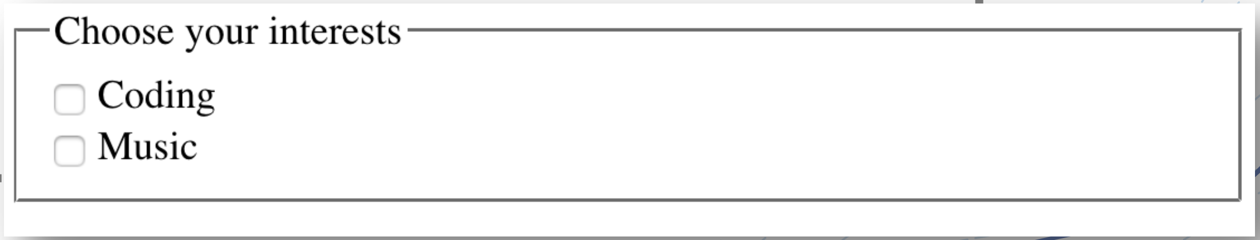
```
<form>
  <input type="radio" id="male" name="gender" value="male">
  <label for="male">Male</label><br>
  <input type="radio" id="female" name="gender" value="female">
  <label for="female">Female</label><br>
  <input type="radio" id="other" name="gender" value="other">
  <label for="other">Other</label>
</form>
```

FORM GROUPS

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

See: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/checkbox -- *Handling multiple checkboxes*

```
<fieldset>
<legend>Choose your interests</legend>
<div>
  <input type="checkbox" id="coding" name="interest" value="coding">
  <label for="coding">Coding</label>
</div>
<div>
  <input type="checkbox" id="music" name="interest" value="music">
  <label for="music">Music</label>
</div>
</fieldset>
```

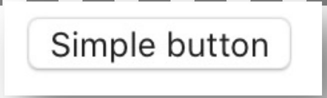

Choose your interests

☐ Coding

☐ Music

FORM BUTTONS AND ACTIONS

```
<button type="button">Simple button</button>
```



- Simple buttons
  - `<button type="button">` is a simple clickable button
    - Note: the default type setting is `"submit"`, so you must define clearly if you don't want the *submit* action!
- Submit and reset
  - `<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 reload
  - `<button type="reset">` clears and restores all input controls in form

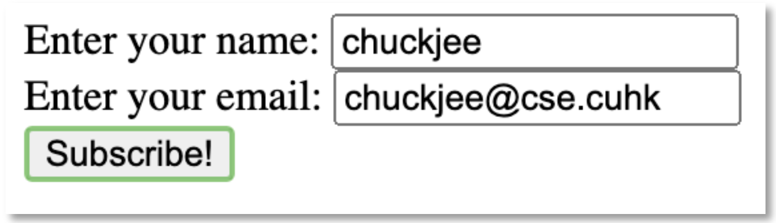
SUBMITTING A FORM

- 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
- The URL towards the script to process data is specified in the `<form action="">` attribute
- For GET/POST submission, the **`name`** attribute of form control elements must be set properly

GET VS. POST

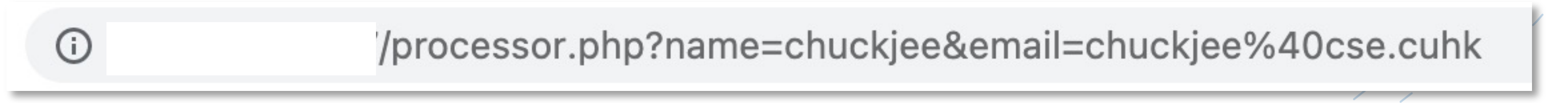
- Using the GET method…

```
<form action="processor.php" 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>
```

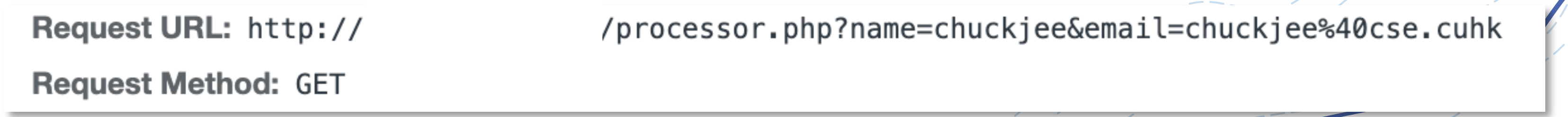


Enter your name: chuckjee

Enter your email: chuckjee@cse.cuhk



/processor.php?name=chuckjee&email=chuckjee%40cse.cuhk



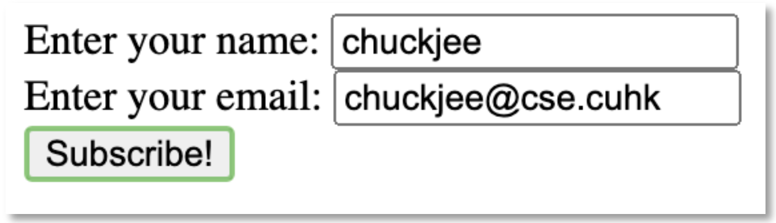
Request URL: http:// /processor.php?name=chuckjee&email=chuckjee%40cse.cuhk

Request Method: GET

GET VS. POST

- Using the POST method…

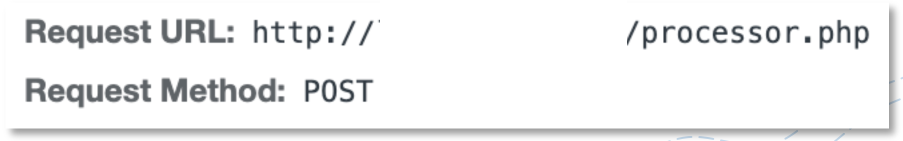
```
<form action="processor.php" method="post" class="form-example">
  <div class="form-example">
    <label for="name">Enter your name: </label>
    <input type="text" name="name" id="name" required>
  </div>
  <div class="form-example">
    <label for="email">Enter your email: </label>
    <input type="email"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



Enter your name: chuckjee

Enter your email: chuckjee@cse.cuhk

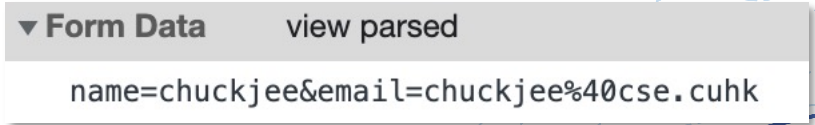
Header



Request URL: http:// /processor.php

Request Method: POST

Payload (body)



▾ Form Data view parsed

name=chuckjee&email=chuckjee%40cse.cuhk

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!)

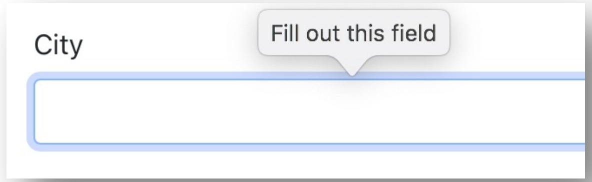
        ➔ more flexibility for developers for displaying helpful messages
- For JavaScript, values in form controls are usually captured using the **`id`** attributes
- Therefore, you may see the **`id`** or **`name`** attributes in examples depending on their purpose…

FORM CSS

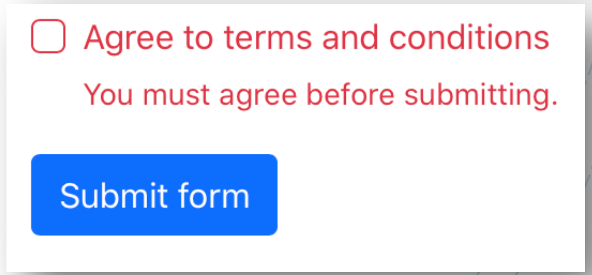
- Form controls can be applied with usual CSS properties, e.g., **`width`**, **`padding`**, etc.
- To style particular types of form controls, the *attribute selector* can help, e.g.:
  - `input[type=text] input[type=button]`
- To style particular states of form controls, some *pseudo-classes* are available,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

FORM VALIDATION

- HTML form should be validated before being processed by server-/client-side scripts
  - To make sure the user put down ***correct data***
  - To make sure incorrect items do not crash scripts
  - To ***lighten workload*** of the processing scripts
- Easily handled with JavaScript or even CSS
  - When user has finished (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



Fill out this field



You must agree before submitting.

READ FURTHER…

w3schools.com HTML Forms

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

MDN Web forms

- *https://developer.mozilla.org/en-US/docs/Learn/Forms*