CS2204 Fundamentals of Internet Applications Development

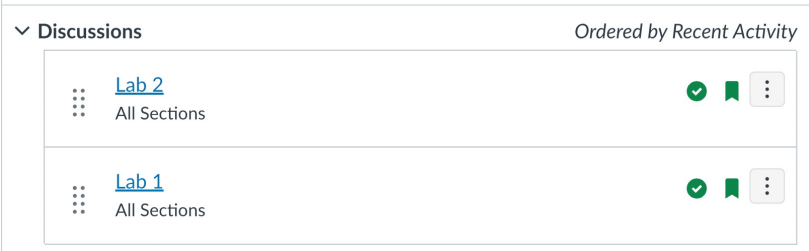
Lecture 3 HTML – Part 3 & CSS – Part 1

Computer Science, City University of Hong Kong

Semester B 2024-25

About Help Seeking

It is highly encouraged that you post the questions on Canvas Discussion or contact the student helper before reaching out to the instructor



Help-Seeking

Student Helper: BUDIANTO Audrey Gandyna (abudianto2-c@my.cityu.edu.hk)

We will also arrange group Zoom Q&A sessions as needed.

Before you reach out for help, you are strongly encouraged to:

- Ask the professors/lab tutors/teaching assistants during class (lecture/lab) if you have any questions
- Post a message on the discussion board (Discussions on the left menu on the Canvas course page)
- Seek help from our dedicated CS student helpers at the the programming clinic at scheduled time slots
- Email the course leader if you have specific concerns/feedback/suggestions.
  Course Leader: Prof. Yuhan Luo (yuhanluo@cityu.edu.hk)

About Bonus points for in-class participation

Only those who answered the questions **correctly** can receive their points up to 0.5 per lecture

- If you have concerns about missing bonus points, please contact the student helper Audrey Gandyna (abudianto2-c@my.cityu.edu.hk) to double check

Computer Programming Clinic

Introduction

Timetable

Map to Clinic

Contact

Internal Use

TimeTable for Semester B 2024-25

Six clinic sessions will be offered every week from Week4 to Week 13 *(excluding Holidays)*

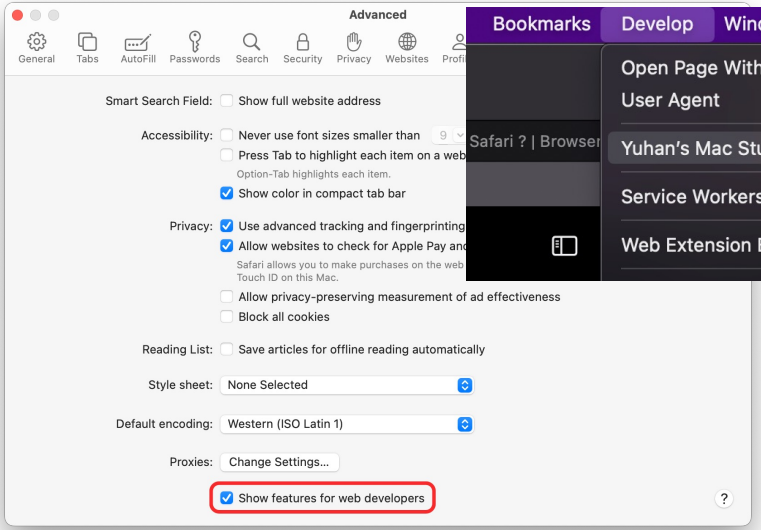
The sessions are offered in CSLab (MMW Building 2/F).
Students can casually walk-in during the scheduled time and ask questions. No booking is needed.

Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
08:00-09:59							
10:00-11:59				MMW2410 Helper on duty: **DAI Sijia**	MMW2410 Helper on duty: **DAI Sijia**		
12:00-13:59							
14:00-15:59	MMW2410 Helper on duty: **LIU Lijie**	MMW2410 Helper on duty: **LIU Lijie**	MMW2410 Helper on duty: **SHAO Xinhui**		MMW2410 Helper on duty: **SHAO Xinhui**		
16:00-17:59							
18:00-19:59							

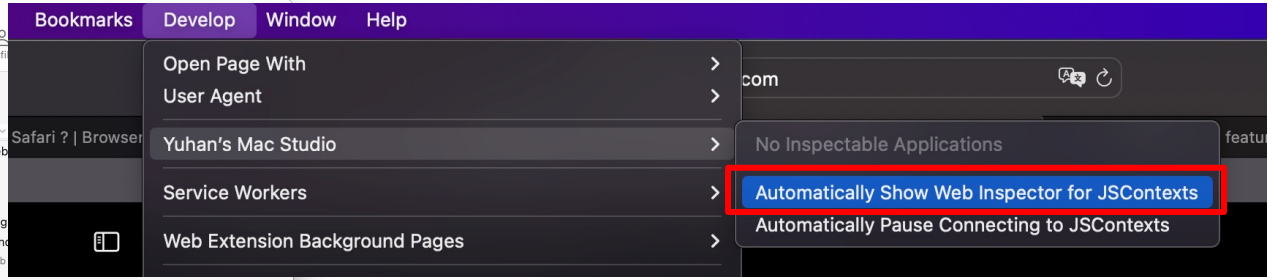
I would be grateful if you could bring the clinic to the students' attention. Thanks.
(official site: https://courses.cs.cityu.edu.hk/clinic)

“Inspect” in Safari on MacOS (1)

Due to the security concerns and design choice by Apple, “inspect” on Safari is less accessible. But you can follow the official guide provided by Apple to inspect webpages on Safari

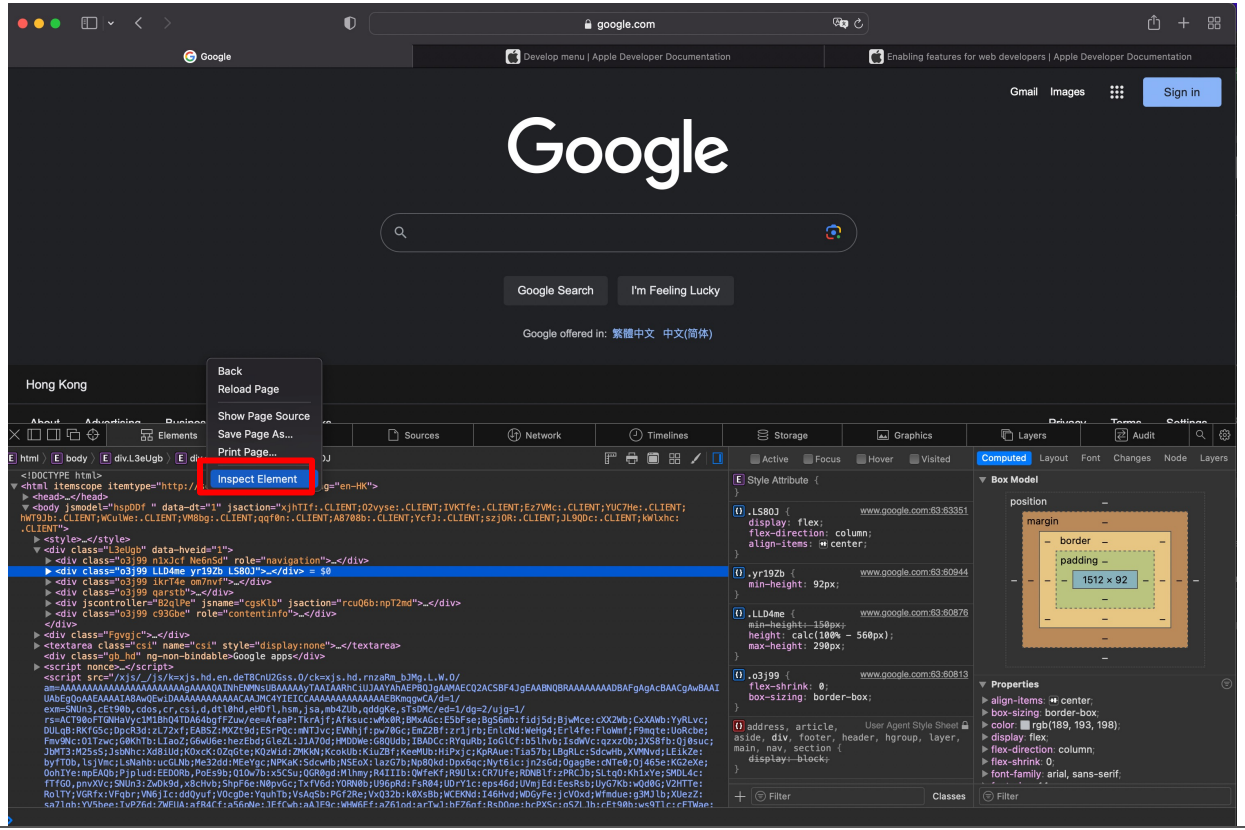


Enable developer mode



Find “inspect” under developer menu

“Inspect” in Safari on MacOS (2)



Then you can right click to find “Inspect Element”

Post-lab Quiz 1 Review

Q1. Which of the followings about SSH is **WRONG**?

	Answer	Respondents	Percentage
✓	Telnet is a SSH application.	34	69%
✗	SSH allows us to access a computer remotely.	4	8%
✗	SSH stands for Secure Shell.	4	8%
✗	MobaXterm is an SSH application	7	14%

Post-lab Quiz 1 Review

Q2. Refer to internet protocols we introduced in the lecture and lab, and match the following items.

Question	Protocol suite that provides the foundation for internet communication.	Protocol used for accessing and managing email on the mail server.	Protocol used for sending email from the sender's client program to the mail server.	Protocol used for transmitting hypertext documents over the internet.
TCP/IP	✓ 34	0	1	5
IMAP	0	✓ 35	2	3
SMTP	0	4	✓ 36	1
HTTP	5	3	1	✓ 32

A quick review from the lecture 02

Webpage structure and source code organization

- "inspect" source code in a browser

Commonly used HTML tags

`<h1>, <h2>, … <h5>, <p>, <ul>, <ol>, <table>, <a>, <image>, <br>, <hr>, <div>`

File path: absolute URL and relative URL

Web accessibility design

Question time

1. List **at least three** commonly used tags within `<head>...</head>`section. Which of them can be used **only** within `<head>...</head>`?

2. Most of tags work in pairs, but some do not (i.e., empty tags). Give **three examples.**

Question time

3. We learned several tags for creating a list, such as `<ul>`, `<ol>`, and `<dl>`. To create an ordered list sorted by alphabet lower case (e.g., a, b, c ...) and set it to start with “c,” what should we do?

Agenda

Form and additional tags

Multimedia

- Video & audio
- Canvas and animation

CSS Introduction

Agenda

Form and additional tags

Multimedia

- Video & audio
- Canvas and animation

CSS Introduction

HTML Form

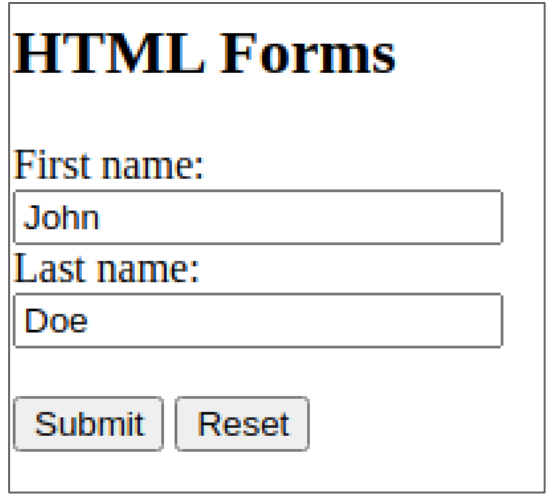
Form is important because it is the basic (standard) structure that **allows user's input** (e.g., a website login form) to be sent back to:

- Web server
- Other pages

Two basics parts of form:

- HTML Tags to define a form and its component (e.g., text input)
- Processing script (e.g., JavaScript in the server or other pages)

Form Structure



method - the way to communicate with the server using HTTP, Get or Post

action - the URL of the server script or target page that will receive and process the form

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
      <title>HTML Forms</title>
  </head>

  <body>
    <h2>HTML Forms</h2>

    <form method="get" action="Lec03-03-HTML-form-script" >
      <label for="fname">First Name:</label> <br>
      <input type="text" id="fname" name="ffname" value="John">
<br>
      <label for="lname">Last Name:</label> <br>
      <input type="text" id="lname" name="llname" value="Doe"> <br>

      <br>

      <input type="submit">
      <input type="reset">
    </form>

  </body>
</html>
```

<form> tag also works in pair

Labels and text input area, other types of inputs will be covered later

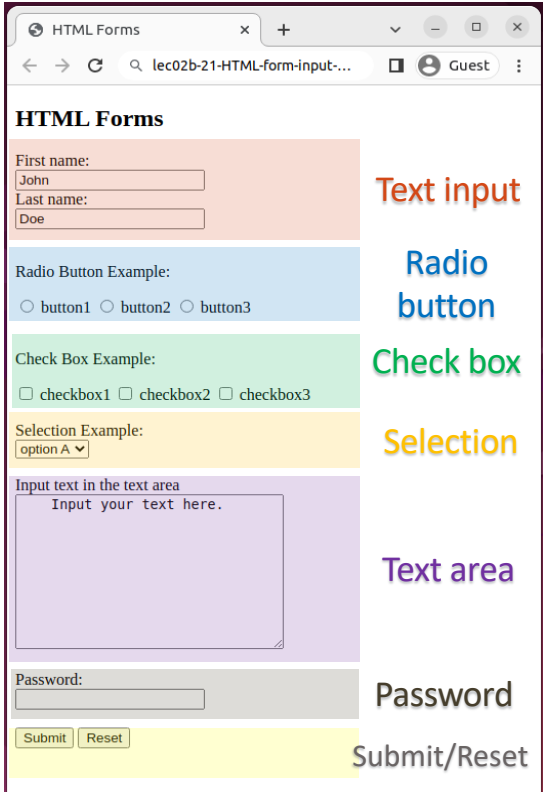
Submit & Reset buttons, when clicked the form will be processed by browser

input type inside the form tag

Code Example: lec03-01-HTML-form-basic-structure.html

Basic Form Input Element Types

```
<body>

<h2>HTML Forms</h2>

<form  method="get" action="Lec03-03-HTML-form-script" >
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>

  <p>Radio Button Example:</p>
  <input type="radio" id="button1" name="radio_button" value="button1">
  <label for="button1">button1</label>
  <input type="radio" id="button2" name="radio_button" value="button2">
  <label for="button2">button2</label>
  <input type="radio" id="button3" name="radio_button" value="button3">
  <label for="button3">button3</label><br><br>

  <p>Check Box Example:</p>
  <input type="checkbox" id="checkbox1" name="check_box" value="checkbox1">
  <label for="checkbox1"> checkbox1</label>
  <input type="checkbox" id="checkbox2" name="check_box" value="checkbox2">
  <label for="checkbox2"> checkbox2</label>
  <input type="checkbox" id="checkbox3" name="check_box" value="checkbox3">
  <label for="checkbox3"> checkbox3</label><br><br>

  <label for="selectExample">Selection Example:</label><br>
  <select id="selectExample" name="selections">
    <option value="optionA">option A</option>
    <option value="optionB">option B</option>
    <option value="optionC">option C</option>
    <option value="optionD">option D</option>
  </select><br><br>

  <label for="textinput">Input text in the text area</label><br>
  <textarea id="textinput" name="message" rows="10" cols="30">
    Input your text here.
  </textarea>

  <br><br>
  <label for="pwd">Password:</label><br>
  <input type="password" id="pwd" name="pwd"><br><br>

  <input type="submit" value="Submit">
  <input type="reset" value="Reset">
</form>

</body>
```



Code Example: lec03-02-HTML-form-input-element-basic.html

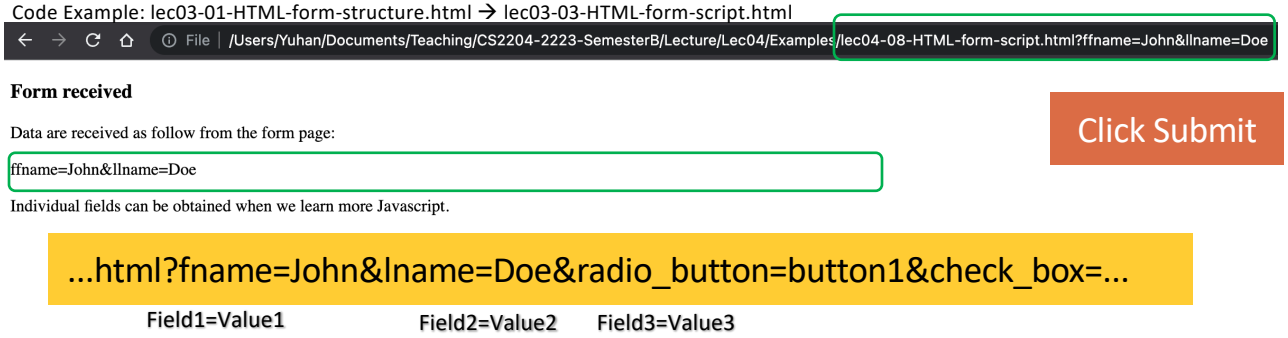
Form processing: overview

Sending the data back to the server by the browser (GET method)?

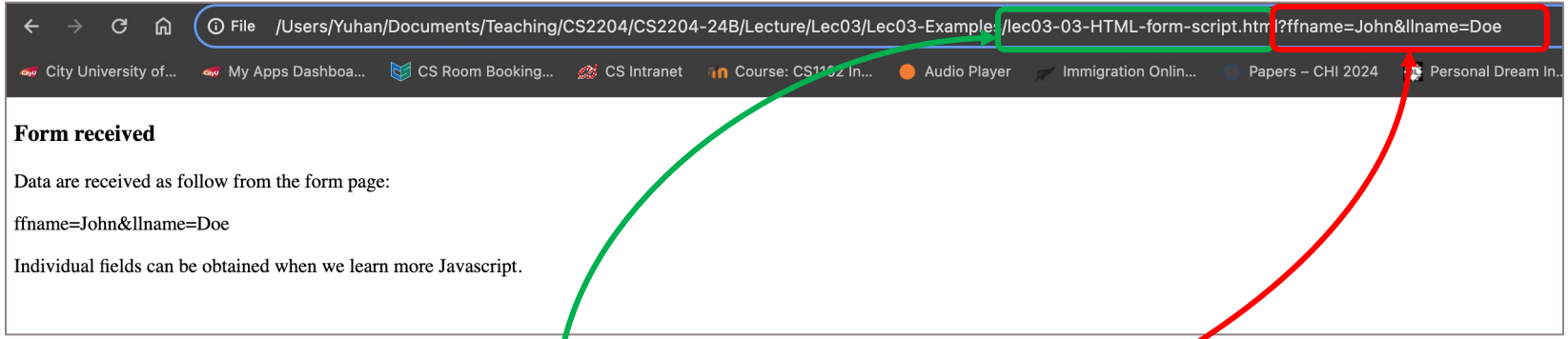
- When the **Submit button** is clicked, the address bar of the browser is changed. The URL is appended with a question mark followed by a string
- This is called Query String. All these are done automatically by the browser

The query string consists of one or more name-value **pair(s)**

- Names come from the name attributes in the input tags
- Values are from the user's input
- Based on this format, the server side/target page scripts and extracts the values for further processing



Form processing: send the data

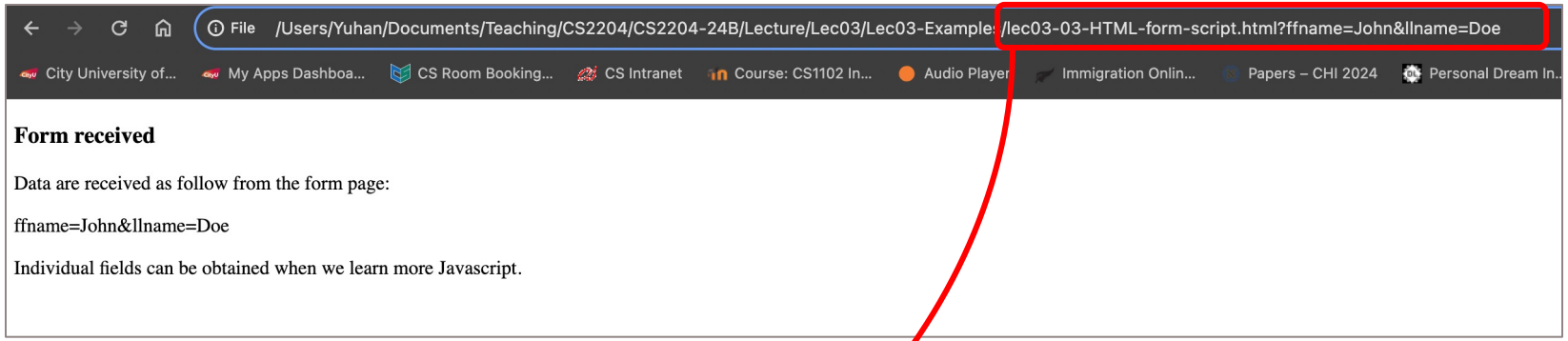



```
<form method="get" action="Lec03-03-HTML-form-script">
  <label for="fname">First Name:</label> <br>
  <input type="text" id="fname" name="ffname" value="John">
<br>
  <label for="lname">Last Name:</label> <br>
  <input type="text" id="lname" name="llname" value="Doe"> <br>

  <br>

  <input type="submit">
  <input type="reset">
</form>
```

When the form is submitted (e.g., by clicking the submit button). **GET** request appends all required data in the URL, which are saved based on their values in the `'name'` attribute and separated by `?`

Code Example: lec03-01-HTML-form-basic-structure.html

Form Processing: reading the data




```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Forms</title>
</head>
<body>
<!-- Page Content -->
    <div id="pageContent">
    <h3>Form received</h3>
    <p>Data are received as follow from the form page:</p>
    <p>
    <script type="text/javascript">
    document.write(document.URL.split("?")[1]);
    </script>
    </p>
    <p>Individual fields can be obtained when we learn more
Javascript.</p>
    </div>

</body>
</html>
```

JavaScript specified in the `'action'` attribute will then process the data

In this example, it spilts the data from the URL by `?` and display the information with a function called `document.write` (details to be covered later during JavaScript lectures)

Think about it: what are the **disadvantages** of the GET request?

Code Example: lec03-03-HTML-form-script.html

Question time

1. Within `<input>`, how does id differ from name?

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
      <title>HTML Forms</title>
  </head>

  <body>
    <h2>HTML Forms</h2>

    <form method="get" action="Lec03-03-HTML-form-script" >
      <label for="fname">First Name:</label> <br>
      <input type="text" id="fname" name="ffname" value="John">
<br>
      <label for="lname">Last Name:</label> <br>
      <input type="text" id="lname" name="llname" value="Doe"> <br>

      <br>

      <input type="submit">
      <input type="reset">
    </form>

  </body>
</html>
```

2. What are the **disadvantages** of the GET request?

Notes on the GET method

- Can be cached
- Remain in the browser history
- Requests can be bookmarked

Another method for sending data

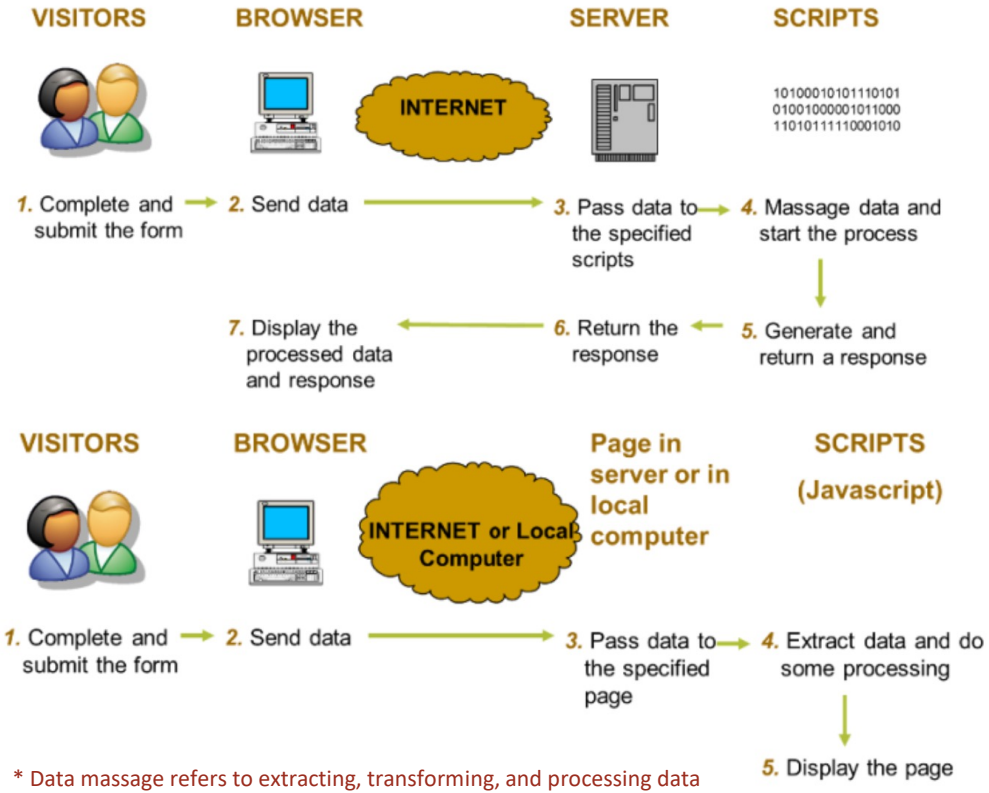
'POST'

- Send data to the **server** or create or update a resource
- Cannot be cached or remain browser history
- Cannot be bookmarked
- No restriction on data length

```
POST /test/demo_form.php HTTP/1.1
Host: w3schools.com

name1=value1&name2=value2
```

Post request is often processed by php script, which is outside the scope of this course.

How does a form send data to other pages?



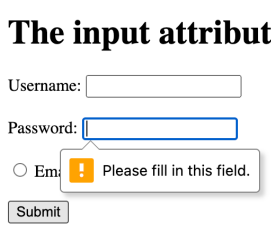
* Data massage refers to extracting, transforming, and processing data

Form elements and their attributes

`<input>` (an empty tag)

- *type,* specify the input type, can be *textbox*, *radio button*, *checkbox*, etc
- *maxlength*, e.g., `<input type="text" name="uname" maxlength="30">`
- *checked*, e.g., `<input type="radio" name="rbut1" checked="checked">`
- *required*, e.g., `<input type="password" name="pwd" required="required">`

```
<!DOCTYPE html>
<html>
<body>

<h1>The input attributes</h1>

<form action="">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" maxlength="10"><br><br>
  <label for="username">Password:</label>
  <input type="text" id="password" name="password" required="required"><br>
<br>

  <input type="radio" id="contactChoice1" name="contact" value="email" />
  <label for="contactChoice1">Email</label>

  <input type="radio" id="contactChoice2" name="contact" value="phone"
checked="checked"/>
  <label for="contactChoice2">Phone</label>

  <input type="radio" id="contactChoice3" name="contact" value="mail" />
  <label for="contactChoice3">Mail</label>

  <br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```



Here we focus on the form elements rather than the processing steps, so we didn't specify a method or action. Thus, the form will not be processed

Code Example: lec03-04-HTML-form-input-attributes.html

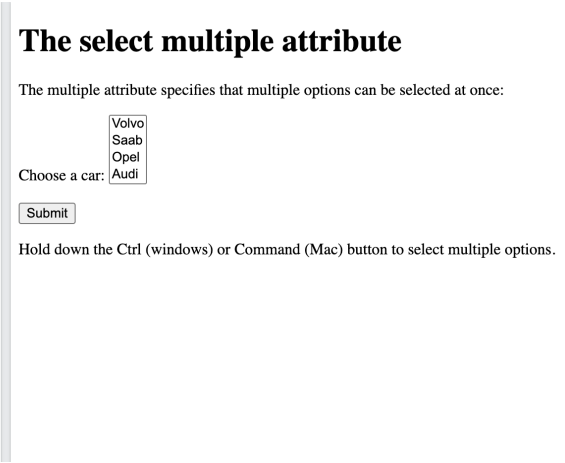
Form elements and their attributes (2)

`<select> … </select>`

- *multiple*, e.g., `<select id="selection" name="ssel" multiple="multiple">`

On mac, press "command" for multiple selection; on windows, press "ctrl"

```
<!DOCTYPE html>
<html>
<body>

<h1>The select multiple attribute</h1>

<p>The multiple attribute specifies that multiple options can be selected at
once:</p>

<form action="">
  <label for="cars">Choose a car:</label>
  <select name="cars" id="cars" multiple="multiple">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="opel">Opel</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit" value="Submit">
</form>

<p>Hold down the Ctrl (windows) or Command (Mac) button to select multiple
options.</p>

</body>
</html>
```



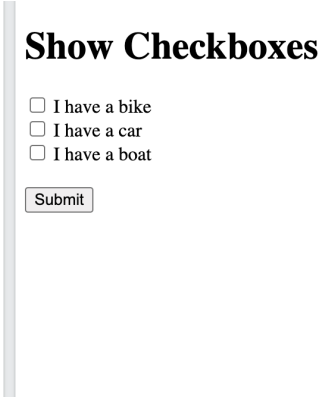
Code Example: lec04-05-HTML-form-select-attributes.html

Form elements and their attributes (3)

'Checkbox'

- If *uncheck* is needed, you can use checkbox
- In radio button/selection box, once a button is checked, it **cannot** be unchecked

```
<!DOCTYPE html>
<html>
<body>

<h1>Show Checkboxes</h1>

<form action="">
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label><br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```



Think about it: When to use radio buttons vs checkbox?

Code Example: lec04-06-HTML-form-checkbox-attributes.html

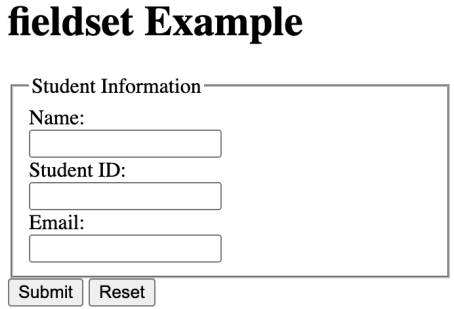
Form elements and their attributes (4)

<fieldset> can group form controls together

- Useful for long forms
- Most browsers show a line around the edge of the `fieldset` element

<legend> explains the purpose of the `fieldset` grouping

- Immediately after the opening <fieldset> tag




```
<form method="get" action="lec04-08-HTML-form-script.html">
    <fieldset>
        <legend>Student Information</legend>

        <label for="name">Name:</label><br>
        <input type="text" id="name" name="nname">

        <br>
        <label for="sid">Student ID: </label> <br>
        <input type="number" id="sid" name="ssid">

        <br>
        <label for="em">Email: </label> <br>
        <input type="email" id="em" name="eem">

        <br>
        <input type="password" name="psw">

    </fieldset>
    <input type="submit">
    <input type="reset">
</form>
```

Code Example: lec03-08-HTML-form-group.html

Think about it: how to validate user input

For example, you have a textbox asking user to enter their email, but some of them may just enter random letters such as “xxx, yyy” that are obviously not an email address

How do we check such issues and send an error message?

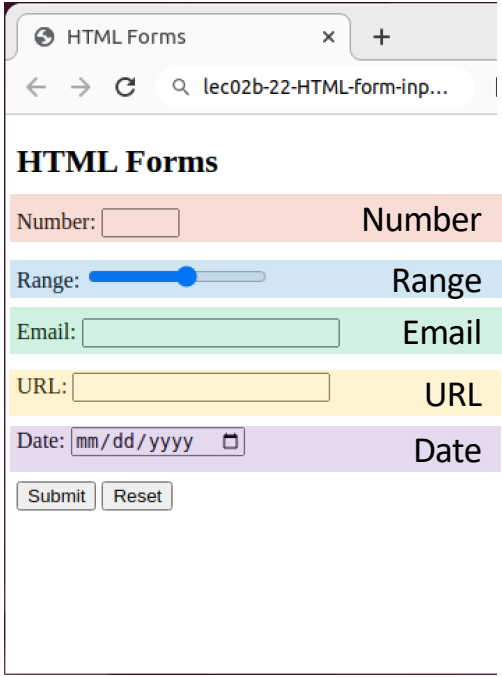
Technically, we need to write a script (e.g., JavaScript) to check the input string. e.g., whether the string includes “`@`” “ **.** `com`” etc or using regular expression to validate the string

However, this is a tedious step and we may have the issues again with other types of input, such as a textbox for telephone number, date, zipcode, etc

Is there a more straightforward approach to validate user input?

New Form Input Element Types

```html
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Forms</title>
      </head>
<body>

<h2>HTML Forms</h2>

<form  method="get" action="lec02b-23-HTML-form-script.html">
    <label for="number">Number:</label>
    <input type="number" min="1" max="10" id="number" name="number"/><br><br>

    <label for="range">Range:</label>
    <input type="range" min="1" max="10" id="range" name="range"/><br><br>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" value="" /><br><br>

    <label for="url">URL:</label>
    <input type="url" id="url" name="url" value="" /><br><br>

    <label for="date">Date:</label>
    <input type="date" id="date" name="date" value="" /><br><br>

    <input type="submit" value="Submit">
    <input type="reset" value="Reset">
</form>

</body>
</html>
```



Code Example: lec03-07-HTML-form-input-element-new.html

Additional Tags (1)

In a `form`, `input type ='reset'` or `'submit'` are special buttons

In HTML, a **button** can be created by using the tag `<button>` which is not tied to a `form` or an `input`

```
<!DOCTYPE html>
<html>
<head>

</head>
<body>

<h1>HTML Button</h1>

<button class="button button1">Button 1</button>
<button class="button button2">Button 2</button>

</body>
</html>
```

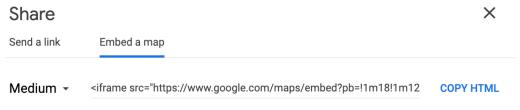
Similar to id, class attribute allows CSS/JavaScript to select the target HTML elements and apply specific styles/actions. But different from id, class value does not need to be unique and can include multiple names for more sophisticated style management (details to be covered later in CSS).

Code Example: lec03-09-HTML-button.html

Additional Tags (2)

`<iframe>` defines a window to show embedded information, short for *inline frame*

- `width` and `height` attributes define the window size
- `src` attribute: URL of the page to be shown in the frame
  - e.g., Google Maps



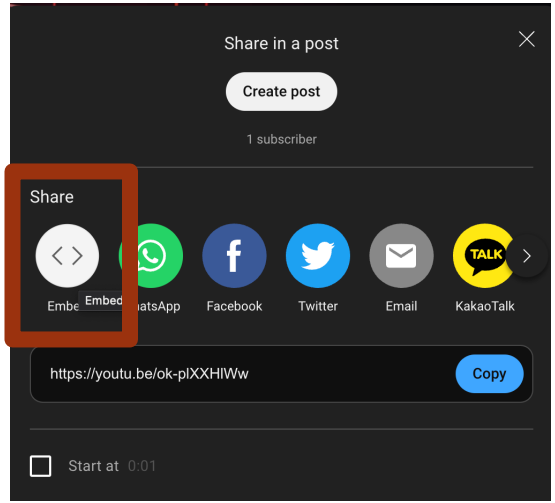

```
    <iframe
        src="https://www.google.com/maps/embed?
pb=!1m18!1m12!1m3!1d3690.4487292240233!2d114.17022935101134!3d22.336679285232595!2m
3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x3404073400f3ef35%3A0xeb61704ffb0b
a959!2sCity%20University%20of%20Hong%20Kong!5e0!3m2!1sen!2shk!4v1663814852321!5m2!1
sen!2shk"
        width="500"
        height="400"
    >

    </iframe>
```

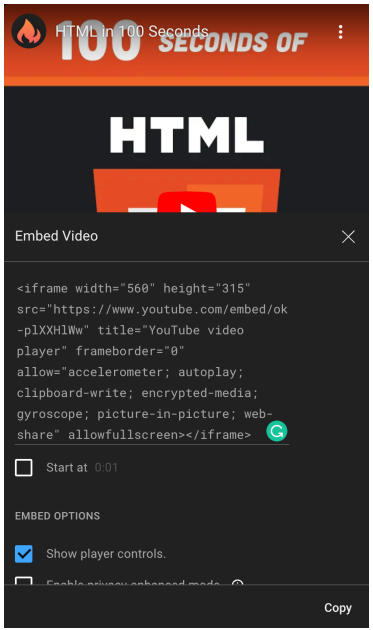
iframe Example

Code Example: lec03-10-HTML-iframe.html

Additional Tags (3)

Some services also included basic CSS style in their <iframe> code for convenient embedding (e.g., Youtube)





Form Good Practice

Basic considerations:

- Use names for input tags is **essential**
- Use label to mark different fields
- Use fieldset & legend to **group** elements…

Note that `<input>` can be used outside `<form>`, a convenient way to get user input for interaction within a page, but data are not sent out

For new input types, the format checking is provided by browser without using JavaScript. However, **not all browsers support all features**

Agenda

Form and additional tags

Multimedia

- Video & audio
- Canvas and animation

CSS Introduction

Multimedia

Multimedia handling is the strength of **HTML5**

- Use of multimedia in HTML5 in fact involves CSS3 and new JavaScript API (Application Programming Interface)
- The development of HTML5 is advanced by WHATWG (Web Hypertext Application Technology Working Group)

Embedding video/audio

- Used to be messy due to different plug-ins and media formats
- Now simplified as HTML element <video> & <audio>
- But with limited the supported formats
- Even more powerful when we learn JavaScript

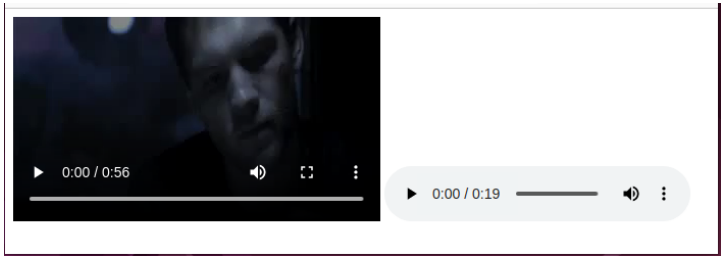
Drawing - Canvas: a drawing area with specified shape, position, and size, must work together with JavaScript

Animation: must use JavaScript in the past, now can use style (CSS3)

Multimedia: Video & Audio

HTML5 trades extendibility for ease of use. Only 3 video formats are supported: **mp4**, **WebM** and **ogg**.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Video Demo</title>
  </head>
  <body>
    <div id="container">
      <video src="../video/trailer.mp4" type="video/mp4"
             controls="controls"
             autoplay="autoplay">
      </video>
      <audio controls="controls">
        <source src="../video/75934537.mp3" />
        </audio>
    </div>
  </body>
</html>
```



Code Example: lec03-11-HTML-video.html

Common useful **attributes** to control the video:

- controls - show the control bar
- loop - repeat playing
- autoplay - play once fully loaded

Browser support: Chrome - mp4, WebM & ogg; Opera - mp4, WebM & ogg; Firefox - mp4, WebM & ogg; IE - mp4; Safari - mp4

Multimedia: Cross Browser Support (1)

Not all browsers support all the three formats. We need to consider providing multiple formats in order to support common browsers (although mp4 is supported by all)

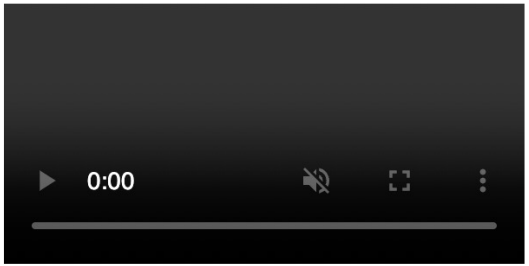
Fall back - the browser will process the multiple sources **one by one** until it recognize a supported format

- In old browsers, if no source format is supported, the remaining coding/HTML will be shown. This is called fallback code (the rationale is similar to alternate text for image).

```
<video controls="controls" autoplay>
     <source src="../video/wonders.ogg" type="video/ogg">
     <source src="../video/wonders.mp4" type="video/mp4">
     <h2>Your browser does not support this video format</h2>
</video>
```

Multimedia: Cross Browser Support (2)

Modern browsers generally recognize the <video> tags, thus the fallback code may not be displayed if the video form is not supported or the link is broken. Instead, it will show a video frame but nothing is playing.



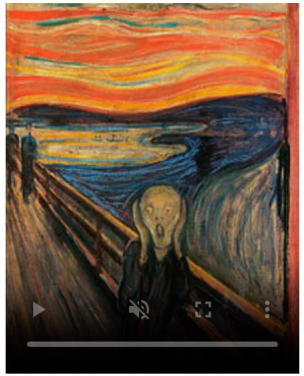
In this case, you may use another attribute `'onerror'` to display the error message within the **last source** tag

```
<video controls="controls" autoplay>
     <source src="../video/wonders.ogg" type="video/ogg">
     <source src="../video/wonders.mp4" type="video/mp4”
onerror="parentNode.parentElement.innerText = 'There is an issue loading or encoding the video'>
</video>
```

What would be a better alternative of fallback code, compared with pure text?

Multimedia: Cross Browser Support (3)

A video cover can be added in case there is an encoding issue with the `'poster'` attribute specifying the file source



In this case, you may remove the `'onerror'` attribute, depending on what you want the users to see when there is an issue loading the video (cover image vs error message)

```
<video controls="controls" autoplay poster=”source/scream.jpeg">
     <source src="../video/wonders.ogg" type="video/ogg">
     <source src="../video/wonders.mp4" type="video/mp4”>
</video>
```

Multimedia: Accessibility

Media **alternatives** (multiple formats for cross-browser support)

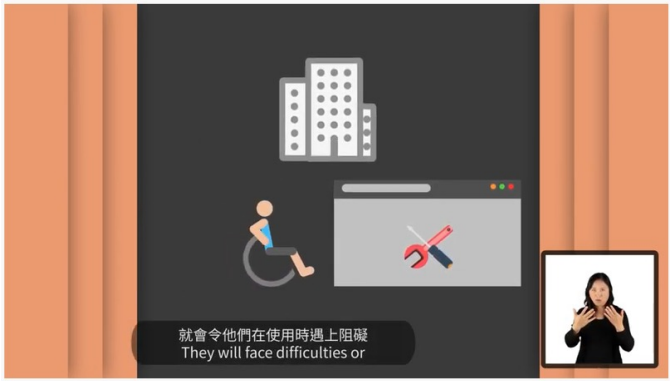
Extended audio **description** (pausing the video for the soundtrack to pass on all the audio info)

Audio control (play, pause, progress, volume)

Avoid irrelevant background noise

Add subtitles or sign language narration

Additional subtitles to show ambient sounds for a clearer picture for deaf people



Multimedia: Canvas

- `Width` and `height` attributes may be added for specifying the canvas size
- Each canvas element should have a unique id, and you can use that id to reference and manipulate the specific canvas element.

Canvas means a real-time drawing surface

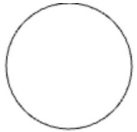
```
<canvas id="surface"> </canvas>
```

The tag only defines an area for drawing, actual action of drawing needs to use JavaScript (to be covered later)

```
<body>
    <h2>A basic square</h2>
    <canvas id="myCanvas1">your browser does not support the canvas tag </canvas>

    <h2>A basic circle</h2>
    <canvas id="myCanvas2">your browser does not support the canvas tag </canvas>

    <script type="text/javascript">

    //draw a rect
    var canvas1=document.getElementById('myCanvas1');
    var ctx=canvas1.getContext('2d');
    ctx.fillStyle='#79ce45';
    ctx.fillRect(0,0,100,100);

    //draw a circle
    const canvas2 = document.getElementById("myCanvas2");
    const ctx2 = canvas2.getContext("2d");

    ctx2.beginPath();
    ctx2.arc(100,50,50,0,2*Math.PI);
    ctx2.stroke();
    </script>
</body>
```

A basic square

A basic circle



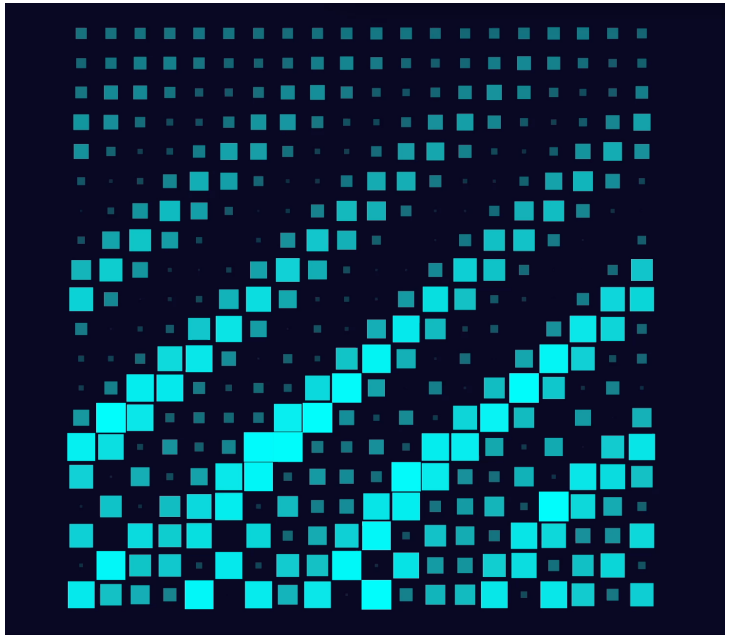
Complex drawing can be done, commonly used for graphing (e.g., http://stock360.hkej.com/quotePlus/1)

Code example: lec03-12-HTML-simple-canvas.html

Multimedia: Animation

Animation can be created with JavaScript or CSS3 with HTML5, but cannot be done by HTML alone

More details will be covered when we talk about CSS



Animation Example and source code:
https://codepen.io/yuanchuan/pen/yLjaERw

HTML

```
<css-doodle>
  <style>
    @grid: 20 / 60vmin;

    background: cyan;
    scale: 0;
    opacity: 0;
    animation: m 4s linear infinite;

    animation-delay: calc(
      -1s/@I * @i * @sin(@i)
    );

    @keyframes m {
      0%, 50%, 90% { scale: 1; opacity: 1 }
      25%, 75%, 100% { scale: 0; opacity: 0 }
    }
  </style>
</css-doodle>
```

CSS

```
html, body {
  padding: 0;
  height: 100%;
  display: grid;
  place-content: center;
  background: #0A0A27;
}
```

Agenda

Form and additional tags

Multimedia

- Video & audio
- Canvas and animation

CSS Introduction

CSS

Cascading Style Sheets (**CSS**) describes how the HTML elements should be **displayed** by specifying the fonts, colors, layout and placement of these HTML elements

Versions of CSS

CSS Level 1 (**CSS1**), 1996

- Simple styles for HTML elements, such as format text, set fonts, and set margins

CSS Level 2 (**CSS2**), 1998

- Same page, different style sheets for different media, such as visual browsers, hearing devices, printers, braille devices

CSS Level 2 Revision 1 (**CSS 2.1**), 2006

- Correct errors in CSS2 errata and add a small amount of new property values
- A stable, widely used version

CSS Level 3 (**CSS3**), 2000

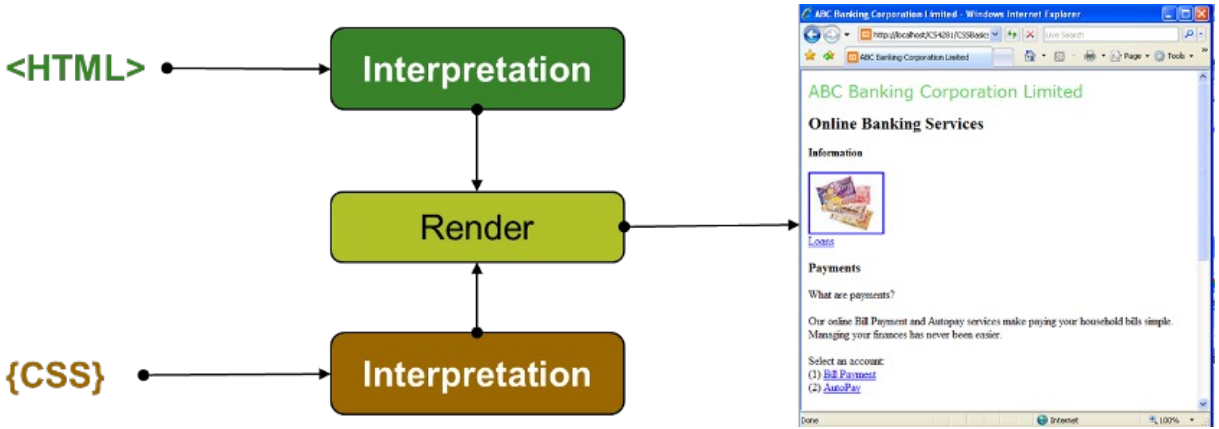
- Chosen to work with **HTML5** and in an active development
- Many new features: web fonts, animation, transform, etc.

How Does CSS work?

Can be either embedded in (put directly inside) the HTML file **OR** linked to it as a separate sheet

When a web page is requested, the server sends the HTML file first, followed by any files embedded to or linked, such as images and `.css` files

After getting the CSS code, the browser interprets it and applies the CSS style to the HTML, and then display the final page in the browser window



How to Set Styles for Elements?

The basic idea

1) find/select an element

2) apply styles on it

The direct way: inline style

```
<h1 style="color: blue; font-size: 20pt;"> A big and blue color heading</h1>
```

But the problems are …

How to Set Styles for Elements?

CSS rule: to select element more efficiently and set styles without mixing with HTML mark-ups

Each rule consists of:

```
selector {property1: value1; property2: value2; ...}
```

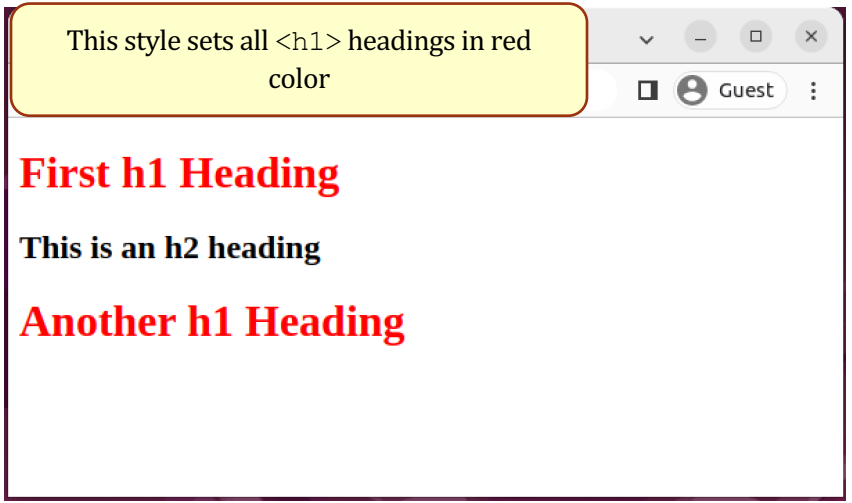
- A **selector** can be HTML tag(s), or class/id name(s)
- These rules are put in either the `<head>` section or an external file, **NOT** mixing with HTML

A Simple CSS Example

CSS can be embedded in the head section of the webpage, with the styles defined inside the `<style> </style>` tags

```
1   <!DOCTYPE html>
2   <html>
3     <head>
8       <title>CSS Simple Example</title>
9       <style>
10          h1 {
11              color: red;
12          }
13      </style>
14    </head>
15    <body>
16    <!-- Page content begins here -->
17
18      <h1>First h1 Heading</h1>
19
20      <h2>This is an h2 heading</h2>
21
22      <h1>Another h1 Heading</h1>
23
24    <!-- Page content ends here -->
25    </body>
26  </html>
```

This style sets all <h1> headings in red color

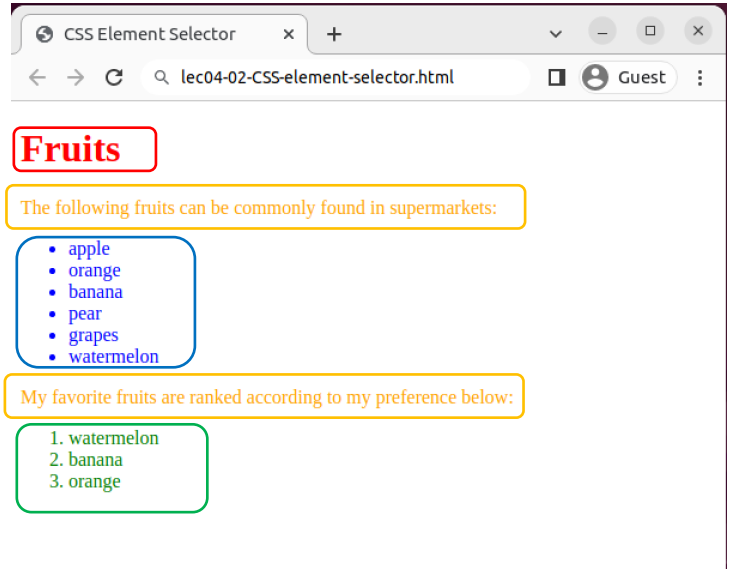


Code Example: lec03-13-CSS-simple-example.html

Element Selector

A CSS style can be applied to different HTML elements by using the corresponding HTML tag as **element selector**

```
<!DOCTYPE html>
<html>
  <head>
    <title>CSS Element Selector</title>
    <style>
      h1 {
        color: red;
      }
      ul {
        color: blue;
      }
      ol {
        color: green;
      }
      p {
        color: orange;
      }
    </style>
  </head>
  <body>
  <!-- Page content begins here -->
   <h1>Fruits</h1>
    <p>The following fruits can be commonly found in supermarkets:</p>
    <ul>
        <li>apple</li>
        <li>orange</li>
        <li>banana</li>
        <li>pear</li>
        <li>grapes</li>
        <li>watermelon</li>
    </ul>

    <p>My favorite fruits are ranked according to my preference below:</p>
    <ol>
        <li>watermelon</li>
        <li>banana</li>
        <li>orange</li>
    </ol>
  <!-- Page content ends here -->
  </body>
</html>
```



Code Example: lec03-14-CSS-element-selector.html

Additional HTML Tag:

The tag can be used to enclose a **part** of the text such that the style can be applied

```html
<head>
  <meta charset="utf-8">
  <title>Span Tag Style</title>
  <style>
    span {
      color: green;
      font-weight: bold;
    }
  </style>
</head>

<body>
  <!-- Page content begins here -->
  <h1>How can you become more <span>effective learner</span> ?</h1>
  <ul>
      <li>Take <span>initiative</span> in your learning</li>
      <li>Be prepared to the class (try to read the materials before hand)</li>
      <li>Attend and participate in all classes (both lecture & lab) </li>
      <li>Submit assignments on time</li>
      <li>Don't be afraid of asking questions (inside or outside classroom)</li>
      <li><span>Use online resources as much as possible</span></li>
      <span><li>Refrain from academic dishonest behaviour</li></span>
  </ul>
  <!-- Page content ends here -->
</body>
```

How can you become more effective learner ?

- Take **initiative** in your learning
- Be prepared to the class (try to read the materials before hand)
- Attend and participate in all classes (both lecture & lab)
- Submit assignments on time
- Don't be afraid of asking questions (inside or outside classroom)
- **Use online resources as much as possible**
- **Refrain from academic dishonest behaviour**

Code Example: lec03-15-HTML-span-tag.html

Back to the button example

How to make Button 1 green and Button 2 blue, and both buttons text white?

```
<!DOCTYPE html>
<html>
<head>

</head>
<body>

<h1>HTML Button</h1>

<button class="button button1">Button 1</button>
<button class="button button2">Button 2</button>

</body>
</html>
```

Button property: background-color

How to make Button 1 green and Button 2 blue, and both buttons text white?

Declaration block:
- Define style by ***name: value***
- Separate multiple declarations by ;

```
<!DOCTYPE html>
<html>
<head>
<style>
.button1 {
  background-color: #4CAF50; /* Green */
  color: white;
}

.button2 {
  background-color: #008CBA; /* Blue */
  color: white;
}
</style>
</head>
<body>

<h1>The button element - text and background with CSS</h1>

<button class="button button1">Green</button>
<button class="button button2">Blue</button>

</body>
</html>
```

The button element - text and background with CSS

Green Blue

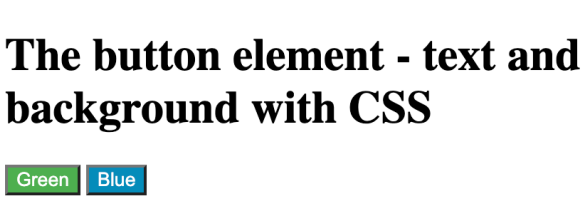
Is there another way to write the CSS part to achieve the same effect?

Code Example: lec03-15-CSS-button-color.html

CSS Comments

```
<!DOCTYPE html>
<html>
<head>
<style>
.button1 {
  background-color: #4CAF50; /* Green */
  color: white;
}

.button2 {
  background-color: #008CBA; /* Blue */
  color: white;
}
</style>
</head>
<body>

<h1>The button element - text and background with CSS</h1>

<button class="button button1">Green</button>
<button class="button button2">Blue</button>

</body>
</html>
```

CSS comment: /* … */



Code Example: lec03-16-CSS-button-color.html

CSS Basics: a quick summary

Describes how HTML elements are to be displayed on screen or in other media

Selector:

- Points to the HTML element you want to style
- HTML elements can be selected by tags, id, class (will cover mode details later)

Declaration block:

- Surrounded by curly braces.
- Declare a CSS property by name : value
- Contains multiple declarations separated by semicolons ;

Common CSS properties: *color*, *back-ground color*, *text-align*, *font-size*, *font-weight*, *margin*, *display*, *border* …

Wrap-up and critical thinking

1. How do you validate user input in HTML forms (e.g., ensure that the input of email includes necessary symbols such as @)?

2. How do you specify the destination URL in submitting a form (e.g., after the “submit” button is clicked)?

Wrap-up and critical thinking

3. The following code intends to create to shapes in two `canvas` elements, but what is wrong with the code?

```
<!DOCTYPE HTML>
<html>
<head>
    <title>Canvas Introduction</title>
</head>

<body>
    <h2>A basic square</h2>
    <canvas id="myCanvas">your browser does not support the canvas tag </canvas>

    <h2>A basic circle</h2>
    <canvas id="myCanvas">your browser does not support the canvas tag </canvas>

    <script type="text/javascript">

    //draw a rect
    var canvas=document.getElementById('myCanvas');
    var ctx=canvas.getContext('2d');
    ctx.fillStyle='#79ce45';
    ctx.fillRect(0,0,100,100);

    //draw a circle
    const canvas = document.getElementById("myCanvas");
    const ctx = canvas.getContext("2d");

    ctx.beginPath();
    ctx.arc(100,50,50,0,2*Math.PI);
    ctx.stroke();
    </script>
</body>
</html>
```

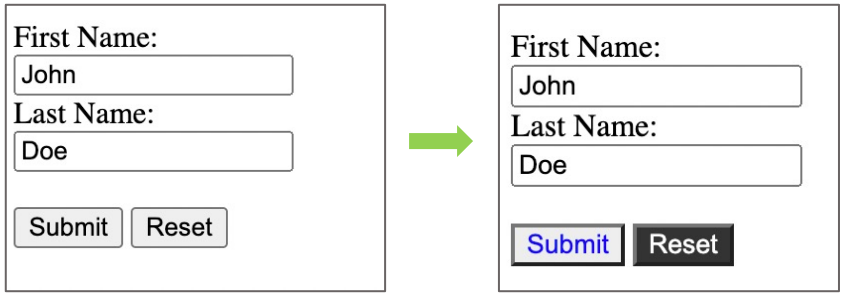
A basic square

A basic circle

Wrap-up and critical thinking

4. Add CSS style to the first form example presented earlier, so the two buttons have different effects (do not change the HTML code)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
      <title>HTML Forms</title>
  </head>

  <body>
    <h2>HTML Forms</h2>

    <form method="get" action="lec03-08-HTML-form-script.html" >
      <label for="fname">First Name:</label> <br>
      <input type="text" id="fname" name="ffname" value="John">
<br>
      <label for="lname">Last Name:</label> <br>
      <input type="text" id="lname" name="llname" value="Doe"> <br>

      <br>

      <input type="submit">
      <input type="reset">
    </form>

  </body>
</html>
```



Lecture summary

HTML form is commonly used for collecting user input

A form has multiple types of element types (e.g., `<input>, <select>`).

`<Input>` is a form element which can be textbox, radio button, checkbox, or a button ("`submit`" and "`reset`")

New form elements such as `number`, `email`, and `url` allow developers to check and validate user input in a convenient way

When a form is submitted, user input can be sent to another page via different methods such as `GET` or `POST`

With HTML5, we can easily insert audio, video, or canvas to a webpage