HTML and CSS

COMP1048: Databases and Interfaces (2024-2025)

Matthew Pike and Yuan Yao

Overview

This Lecture

- Introduction to using HTML and CSS for web development.
- Key tools and technologies for effective web development.
- Best practices for organising and structuring code.
- By the end of this lecture, you will be able to create a basic web page using HTML and CSS.

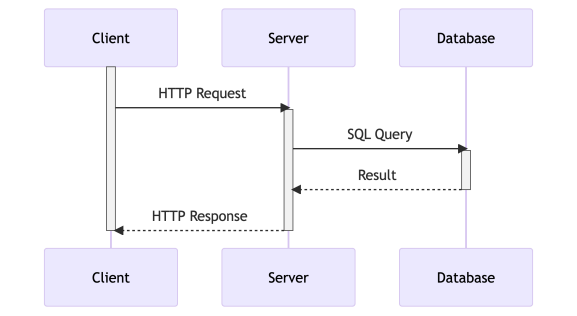
Client-Server Model

Hypertext Transfer Protocol (HTTP)

- HTTP is the protocol used to transfer data between a client and a server.
- HTTP is stateless and text-based:
  - The server does not retain memory of the client for subsequent requests.
- HTTP follows a request-response model:
  - The client sends a request to the server.
  - The server sends a response back to the client.
- Common HTTP status codes:
  - 200: OK
  - 404: Not Found
  - 500: Internal Server Error
  - 503: Service Unavailable (often seen with Moodle)
  - 418: I'm a teapot (an April Fools' joke)

Client-Server Model

- The client refers to the user's web browser.
- The server is the computer system that hosts the website's content and functionality.



This diagram illustrates the client-server model, showing the request-response cycle and interactions with a database.

HTML

What Makes a Web Page?

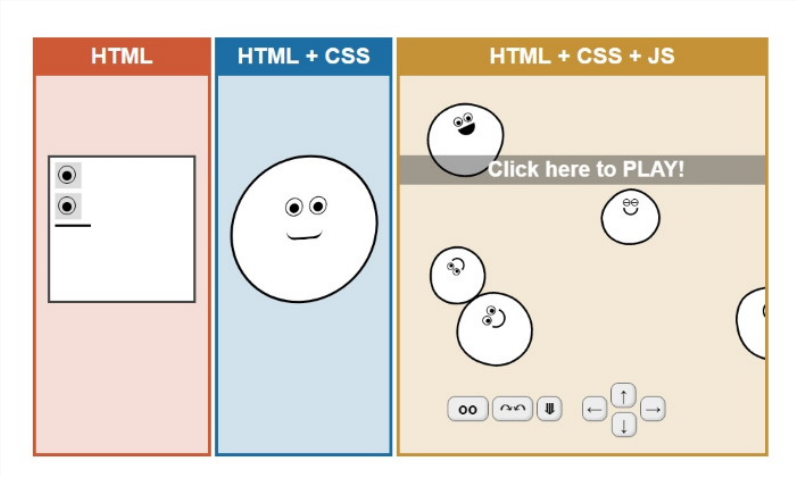


Figure 1: Image Source: html-css-js.com

What is HTML?

COMP1048 uses HTML5 (only)

We will exclusively use HTML5 in this module. The HTML5 specification is maintained by the World Wide Web Consortium (W3C): http://www.w3.org/TR/html5/

- Hypertext Markup Language (HTML) is the standard markup language for creating web pages.
  - HTML annotates text to provide structure to documents. It is not a programming language.
- HTML elements are used to structure content, including headings, paragraphs, lists, tables, and forms. Common elements include `<h1>`-`<h6>`, `<p>`, `<ul>`, `<ol>`, `<li>`, `<table>`, and `<form>`. These will be discussed in detail later.
- HTML5 introduces semantic elements like `<header>`, `<footer>`, and `<nav>`, as well as built-in multimedia support with `<video>` and `<audio>`.

HTML Tags

- HTML tags are the fundamental building blocks for creating web pages.
- Tags are case-insensitive and enclosed in angle brackets: `<tag>` for opening and `</tag>` for closing.
- Tags typically appear in pairs, consisting of an opening and a corresponding closing tag: `<p>Hello World!</p>`.

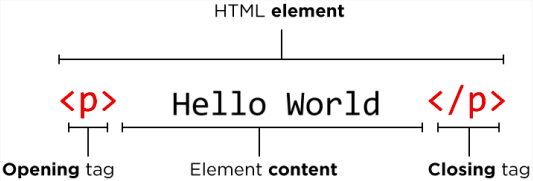


Figure 2: An annotated example of HTML tags, showing opening and closing tags, content, and the element

Example: Hello World

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Hello World</title>
</head>
<body>
    Hello World!
</body>
</html>
```

Figure 3: Rendered HTML page for a simple Hello World example.

HTML Tags Hierarchy

- `<!DOCTYPE html>` declares the document type and the version of HTML.
- `<html>` is the root element of the HTML document, enclosing all content except `<!DOCTYPE html>`.
- `<head>` contains metadata for the document, which is not displayed on the webpage.
  - `<title>` defines the document's title, shown in the browser's title bar.
  - `<meta>` specifies metadata, such as `charset="utf-8"`, indicating UTF-8 encoding.
- `<body>` holds the content displayed on the webpage.
  - The body element contains all the visible content rendered in the browser.

Nested HTML Tags

Web Browsers and Invalid HTML Code

Web browsers are tolerant of invalid HTML and will attempt to display content even with errors. However, always ensure your HTML is valid.

- HTML elements can be nested within one another:
  - The outer element is the parent element.
  - The inner element is the child element.
- It is crucial to match opening and closing tags correctly:
  - Correct: `<p> Hello <b>COMP</b>1048! </p>`
  - Incorrect: `<p> Hello <b>COMP1048. </p></b>`
- Use validation tools to check your HTML code:
  - W3C Markup Validation Service - https://validator.w3.org/

HTML Attributes

- HTML elements can include attributes that provide additional details about the element.
- Attributes are placed in the opening tag and follow a name/value format: `name="value"`.
- Common attributes include:
  - `id`: Assigns a unique identifier to an HTML element. Each `id` must be unique within the document.
  - `class`: Assigns one or more class names to an element. The same class name can be used across multiple elements.
- Examples:
  - `<p class="center">Hello World!</p>`
  - `<p id="intro">Hello World!</p>`
  - `<p class="center bold">Hello World!</p>`
- We will explore how to use CSS to style HTML elements using `class` and `id` attributes later in the lecture.

Headings, Paragraphs, and Line Breaks

- HTML headings are defined using the `<h1>` to `<h6>` tags:
  - `<h1>` represents the most important heading.
  - `<h2>`, `<h3>`, `<h4>`, and `<h5>` indicate progressively less important headings.
  - `<h6>` is the least significant heading.
- HTML paragraphs are defined with the `<p>` tag.
  - The browser automatically adds margin before and after paragraphs.
- HTML line breaks are created with the `<br/>` tag.
- Lists are created using the `<ul>` or `<ol>` tags:
  - `<ul>` defines an unordered list, displayed with bullet points.
  - `<ol>` defines an ordered list, displayed with numbers.
  - `<li>` is used for each item in both unordered and ordered lists.

Example: Headings, Paragraphs and Line Breaks

```
<!-- Header skipped for brevity -->
    <h1> Welcome! </h1>
    <p> This is <br /> my website. </p>
    <h2>Hobbies</h2>
    <ul>
        <li>Coding</li>
        <li>Playing Football</li>
    </ul>
    <h2> Football Teams </h2>
    <ol>
        <li>Swansea City</li>
        <li>Swansea City</li>
    </ol>
```

Figure 4: Headings, paragraphs, line breaks, and lists example.

Hyperlinks

- HTML links are defined using the `<a>` tag.
  - The `<a>` tag creates a hyperlink, allowing navigation from one page to another.
  - The primary attribute of the `<a>` tag is `href`, which specifies the link's destination.
  - The text between the opening and closing `<a>` tags is displayed as the link.
- The `href` attribute is flexible, supporting links to web pages, files, email addresses, or any URL.
- Example:
  - `<a href="http://www.nottingham.edu.cn">Visit our homepage</a>`

Example: Hyperlinks

```
<!-- Header skipped for brevity -->
<p>Links to each Nottingham Campus:</p>
<ol>
    <li><a href="https://nottingham.ac.uk">
        UNUK
    </a></li>
    <li><a href="https://nottingham.edu.cn">
        UNNC
    </a></li>
    <li><a href="https://nottingham.edu.my">
        UNNM
    </a></li>
</ol>
```

Links to each Nottingham Campus:

1. UNUK
2. UNNC
3. UNNM

Figure 5: Lists and Hyperlinks example.

Images

Closing Tags

Closing tags can usually be omitted if the element does not contain other tags or content. A forward slash (`/`) at the end of the opening tag is shorthand for this—`<br />` is equivalent to `<br></br>`. Although, strictly speaking, the closing tag is not required for self-closing tags, meaning `<br>` is also valid.

- HTML images are added using the `<img>` tag.
- The `src` attribute (mandatory) specifies the image's path.
- The `alt` attribute provides alternative text for the image if it fails to load.
  - This is crucial for **accessibility**, as screen readers use the `alt` text for users with visual impairments.
- The `width` and `height` attributes are commonly used to set the image's dimensions.

Example: Images

```
<img
    src="images/missing.png"
    alt="A missing image"
    width="200"
    height="200"
/>

<img
    src="https://tinyurl.com/uonlogo"
    alt="University of Nottingham Logo"
/>
```

Figure 6: Images example.

Tables

Do Not Use Tables for Layout

Tables should not be used for layout purposes. Use CSS for layout instead. This is a common mistake made by beginners.

- HTML tables are created using the `<table>` tag.
  - `<tr>` defines a table row, `<th>` is used for table headers, and `<td>` represents a table data cell.
- A table can be divided into header (`<thead>`), body (`<tbody>`), and footer (`<tfoot>`) sections.
- A caption can be added with the `<caption>` tag.
  - The caption improves accessibility by providing a description of the table's content, which can be read by screen readers for users with visual impairments.
- The `colspan` and `rowspan` attributes are used to merge cells across columns or rows.

Example: Tables

```
<table border="1">
<caption> DBI Class Schedule </caption>
<thead>
    <tr> <th>Week</th> <th>Topic</th> </tr>
</thead>
<tbody>
    <tr><td>1</td> <td>Introduction to DB</td> </tr>
    <tr> <td colspan="2"> <b> Holiday </b> </td> </tr>
</tbody>
<tfoot>
    <tr> <td> Updated </td>
    <td>08 November 2022</td> </tr>
</tfoot>
</table>
```

DBI Class Schedule

Week	Topic
1	Introduction to DB
Holiday	
Updated	08 November 2022

Figure 7: Tables example. Note the use of `colspan` attribute in row 3.

Using `<div>` and `<span>`

- The `<div>` tag defines a block-level element used to group other elements, typically for organising content into logical sections.
- The `<span>` tag is an inline element used for grouping within other elements, often to apply styles to text within a paragraph.
- The `class` and `id` attributes are used to apply styles (via CSS) to `<div>` and `<span>` elements.

Forms

- The `<form>` tag creates an interactive form for user input.
- Submitting the form sends the data to the server, with the destination specified in the `action` attribute.
- Data is sent as name/value pairs, with the `name` attribute designating the name for each piece of data.
- The `method` attribute specifies the data submission method, either `GET` or `POST`.
- The `<input>` tag creates an input field, where the `type` attribute defines the kind of input (e.g., `text`, `password`, `submit`).
- The `<label>` tag provides a label for an `<input>` element. The `for` attribute of the `<label>` should match the `id` attribute of the `<input>` to associate them.

Example: Forms

```html
<form action="example.com/process_form" method="post">
    <label for="frmEmail">Email:</label>
    <br />
    <input type="text" id="frmEmail"
        name="email"
        value="test@email.com">
    <br/>
    <label for="frmPswd">Password:</label>
    <br />
    <input type="password" id="frmPswd"
        name="password" value="password">
    <br/><br/>
    <input type="submit" value="Submit">
</form>
```

Figure 8: The rendered HTML page for the forms example.

GET vs POST

- The `GET` method retrieves information from a server at a specified URI.
  - `GET` should only be used to retrieve data and is not suitable for submitting sensitive information, as parameters are visible in the URL.
  - `GET` requests have limitations on the amount of data that can be sent.
- The `POST` method sends data to the server, such as customer information or file uploads.
  - `POST` can send more data than `GET` and is more secure, as parameters are not exposed in the URL and are not stored in browser history or web server logs.
  - Data in `POST` requests are included in the body of the HTTP request, with headers like `Content-Type: application/x-www-form-urlencoded` or `multipart/form-data`.

Semantic HTML

- Semantic HTML uses elements that clearly describe their meaning in the context of a web page.
  - Examples include `<header>`, `<footer>`, `<nav>`, `<article>`, `<section>`, and `<aside>`.
- There are three main benefits to using semantic HTML:
  - **Accessibility**: Screen readers and search engines can better understand the content.
  - **SEO**: Search engines can better index and rank the content.
  - **Readability**: The code is easier to read and maintain.

```html
<header>
    <h1>My Website</h1>
</header>
<nav>
    <ul>
        <li><a href="home.html">Home</a></li>
        <li><a href="about.html">About</a></li>
        <li><a href="contact.html">Contact</a></li>
    </ul>
</nav>
<section>
    <h2>About Us</h2>
    <p>Learn more about our company...</p>
</section>
<footer>
    <p>&copy; 2022 My Website</p>
</footer>
```

Accessibility in Web Development

What is Web Accessibility?

- Web accessibility ensures that websites and web applications are usable by everyone, including people with disabilities.
- It involves creating content that can be accessed and interacted with by all users, regardless of their abilities or disabilities.
- Tools for web accessibility include screen readers, keyboard navigation, and voice recognition software.

Key Accessibility Practices:

- **Alternative Text**: Use the `alt` attribute for images to provide descriptive text for screen readers.
- **Keyboard Navigation**: Ensure your site is navigable using only a keyboard (e.g., using `tab`, `shift+tab` for navigation).
- **Color Contrast**: Ensure sufficient contrast between text and background for readability by users with visual impairments.
- **Forms**: Use `<label>` elements with the `for` attribute to properly associate labels with form inputs.

Why is Accessibility Important?

- Accessibility improves the user experience for all users, not just those with disabilities.
- It's required by laws in many regions - https://www.w3.org/WAI/policies
- Accessible sites have a broader reach and increase audience engagement.

Tools for Evaluating Accessibility

- **WAVE**: Web accessibility evaluation tool to check accessibility issues in your pages.
  - WAVE Web Accessibility Evaluation Tool
- **Lighthouse**: A Chrome tool for auditing accessibility, performance, SEO, and more.
  - Lighthouse Accessibility Audits

HTML Resources

Practical Hints and Tips for developing HTML

- Ensure that your HTML is valid.
  - Validate your HTML using the W3C Markup Validation Service - validator.w3.org.
- Utilise a text editor with syntax highlighting.
  - Visual Studio Code is an excellent option.
- Use a *good* web browser.
  - Google Chrome
  - Firefox
- Familiarise yourself with browser developer tools.
  - Chrome DevTools - developers.google.com/web/tools/chrome-devtools
  - Firefox Developer Tools - developer.mozilla.org/en-US/docs/Tools
- Use the Mozilla Developer Network (MDN) for reference:
  - developer.mozilla.org/en-US/docs/Web/HTML/

CSS: Cascading Style Sheets

What is CSS?

- CSS allows us to control the appearance of HTML elements in the browser, overriding the default styles provided by the browser.
- In essence:
  - HTML defines the structure of web pages.
  - CSS controls the style and presentation of web pages.
- CSS offers a wide range of features, including:
  - Basic text formatting and layout control.
  - Complex multi-column layouts and responsive design.
  - Animations and transitions.
  - Font styling and typographic details.

Recall: What Makes a Web Page?

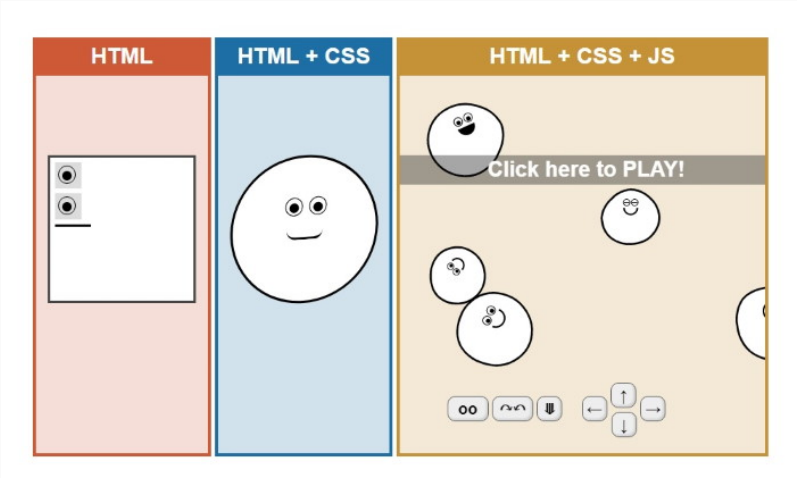


Figure 9: Image Source: html-css-js.com

CSS Syntax

CSS uses American English spelling, so `color` is used instead of `colour`.

- CSS is a rule-based language.
- The *selector* targets the HTML element to be styled.
- The declaration block, enclosed in curly braces `{}`, contains one or more declarations, each separated by a semicolon `;`.
- Each declaration consists of a CSS property name and a value, separated by a colon `:`.

Syntax:

```
selector {
    property: value;
    property: value;
}
```

Example:

```
p {
    color: red;
    font-weight: bold;
}
```

CSS Selectors

- CSS selectors identify the HTML elements to which styles will be applied.
- **Element selectors** target elements by their tag name.
  - Example: `p` will select all `<p>` elements.
- **Class selectors** select elements with a specific `class` attribute, prefixed by a `.`.
  - Example: `.center` will select all elements with `class="center"`.
- **ID selectors** target a unique element by its `id` attribute, prefixed by a `#`.
  - Example: `#intro` will select the element with `id="intro"`.

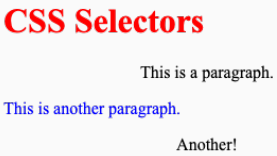
Example: CSS Selectors

HTML

```html
<h1> CSS Selectors </h1>
<p class="center">
    This is a paragraph.
</p>
<p id="myPara">
    This is another paragraph.
</p>
<p class="center"> Another! </p>
```

CSS

```css
h1 {color: red;}
.center {text-align: center;}
#myPara {color: blue;}
```



CSS Property Values

Not an Exhaustive List

This is not an exhaustive list of CSS properties. Please refer to the MDN CSS Reference for a comprehensive list.

- CSS property values can be defined in various formats:
  - **Keywords**: `center`, `left`, `right`, `top`, `bottom`, etc.
  - **Length units**:
    - Pixels: `10px`, `20px`, `30px`, etc.
    - Percentages: `10%`, `20%`, `30%`, etc.
    - Centimeters: `10cm`, `20cm`, `30cm`, etc.
  - **Colors**:
    - Named colors: `red`, `blue`, `green`, etc.
    - Hexadecimal codes: `#ff0000`, `#00ff00`, `#0000ff`, etc.
    - RGB values: `rgb(255, 0, 0)`, `rgb(0, 255, 0)`, `rgb(0, 0, 255)`, etc.

Adding CSS to HTML

- There are three methods to apply CSS to HTML:
  - **Inline CSS**: Embed CSS directly in an HTML element using the `style` attribute. (Avoid this method - it leads to code duplication and is hard to maintain.)
  - **Internal CSS**: Place CSS within the `<head>` of the HTML document using the `<style>` tag. (Not recommended.)
  - **External CSS**: Create a separate CSS file and link to it in the `<head>` section of the HTML document using the `<link>` tag. (Recommended.)

- It is best practice to use external CSS files to separate content from styling:
  - This approach simplifies code maintenance.
  - It allows reuse of the same CSS file across multiple HTML documents.
  - Link an external CSS file with the `<link>` tag in the `<head>` section of the HTML.
  - Example: `<link rel="stylesheet" href="styles.css">`

Cascade Order of CSS Rules

- The order in which CSS rules are applied is known as the *cascade order*.
- Cascade order is influenced by:
  - **Specificity**: Selectors with greater specificity take priority.
  - **Source order**: The order in which rules appear in the CSS file.
  - **Last rule**: When rules have equal specificity, the latter rule is applied.

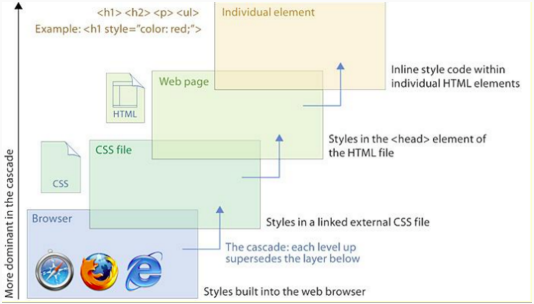


Figure 11: Image Source: Basic Design Principles for Creating Web Sites, by Patrick J. Lynch and Sarah Horton

The Box Model

- The box model is a conceptual framework that explains how HTML elements are placed and laid out on a page.
- It consists of:
  - **Content**: The actual content of the element (e.g., text or images).
  - **Padding**: The space around the content, which is transparent.
  - **Border**: A border surrounding both the padding and the content.
  - **Margin**: The space outside the border, which is transparent.

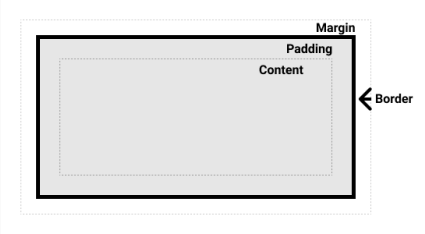


Figure 12: Image Source: MDN: The Box Model

How Does CSS Work?

1. The browser interprets the HTML document and builds a tree structure known as the Document Object Model (DOM).
2. It retrieves additional resources linked to the HTML document, such as images and CSS files.
3. The browser processes the CSS files to define the style rules for each node in the DOM tree.
4. Finally, the browser renders the HTML document, applying the style rules to display the content as intended.

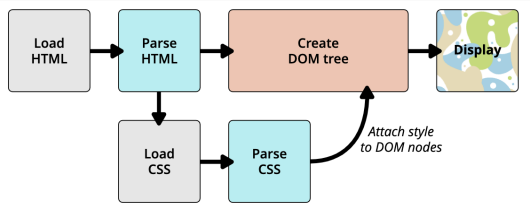


Figure 13: Image Source: MDN Web Docs: How CSS Works

The Document Object Model (DOM)

- The Document Object Model (DOM) is the browser's in-memory representation of the HTML document, used for page rendering.
- The DOM is structured as a tree of objects, with each object representing a part of the document.
- JavaScript can modify the DOM, which we will explore in a later lecture.
- The DOM is accessible through the browser's developer tools:
  - In Chrome: Press `F12` and select the `Elements` tab.
  - In Firefox: Press `F12` and go to the `Inspector` tab.
  - In Safari: Press `Option+Command+i` and click on the `Elements` tab.

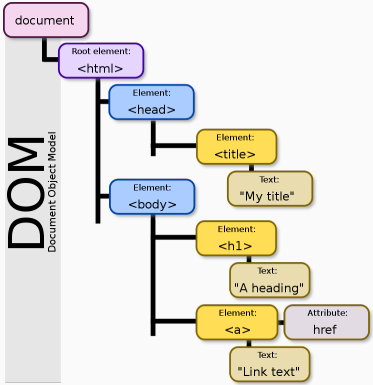


Figure 14: Image Source: Wikipedia: DOM

Answers to Common Questions

- It is not expected for you to memorise all HTML and CSS tags and properties, as this is impractical.
- You should aim to:
  - Understand and interpret HTML and CSS code and its function.
  - Create basic HTML and CSS code using the tags and properties we have covered.
- The goal is to build simple web pages with HTML and CSS and use developer tools to inspect and troubleshoot your code.

Resources

- HTML Validator: validator.w3.org
- CSS Validator: jigsaw.w3.org/css-validator
- MDN HTML Tutorial: developer.mozilla.org/en-US/docs/Learn/HTML
- MDN CSS Tutorial: developer.mozilla.org/en-US/docs/Learn/CSS
- CSS Zen Garden: csszengarden.com

References and Further Reading

- MDN HTML Web Docs: developer.mozilla.org/en-US/docs/Web/HTML
- MDN CSS Web Docs: developer.mozilla.org/en-US/docs/Web/CSS
- HTML Specification: html.spec.whatwg.org/multipage
- CSS Specification: w3.org/Style/CSS/specs.en.html