HTML

What is HTML?

Hyper Text Markup Language
Not a programming language (no logic or math), rather is a 'markup' language.
A web browser is an application that parses HTML and renders it; is also a javascript interpreter.

Elements and tags

HTML is composed of elements, represented using tags.
Tags are keywords that instruct the parser (the browser) how to render certain content.
They are enclosed by angle brackets `<, >` and can have their own attributes apart from the main keyword.
Not case sensitive.
Examples: head, img, p, a

DOCTYPE declaration

`<!DOCTYPE html>` is the first line required in every HTML document.
It needs to specified to ensure that all browsers render the page in the same way.

More info: https://stackoverflow.com/a/6076470

Common Tags

- `<html>` : Represents the root of the HTML document; contains all other HTML elements.
- `<head>` : Used to contain metadata or information related to the document. Usually holds the `<title>`, `<meta>` and `<link>` tags.
- `<title>` :
  Title of the website. Displays it on the browser/tab header.
- `<body>` :
  Contains all the visible contents in a webpage.

Containers

- `<p>` : Used to enclose paragraphs with.
- `<div>` : Most common container, used to encapsulate a **block** of content.
- `<span>` : Similar to a div, but **inline**.

Headings

- `<h1> till <h6>` : Different sizes/levels of headings.

Hyperlinks

- `<a>` : Used to link to other webpages.

Empty (self-closing) tags

- `<br />` : Line break.
- `<img />` : Insert an image.

Forms

- `<form>` : Container for a form (a group of inputs/labels/buttons).
- `<input>` : Tag to specify some kind of input.

  > type="text": textbox
  > type="password": hidden text input for sensitive info
  > type="radio": radio buttons (single-choice)
  > type="checkbox": checkboxes (multi-choice)
  > type="submit": renders a submit button

- `<label>` : Add label text to any `<input>` element.
- `<button>` : Add a button.

  > type="submit": button to submit the form; same as `<input type="submit">`

Detour: Identifying/selecting specific elements

Obviously, there can be multiple elements having the same tag but with different contents/purpose.
Every tag can be assigned an `id` attribute which can be used to uniquely identify them.
IDs are user-defined and are supposed to be unique.

In input elements within a form, the attribute `name` is also required to properly generate the form data.

Lists

- `<ul>` : Container for an unordered (bulleted) list.
- `<ol>` : Container for an ordered (numbered) list.
- `<li>` : Each list item.

Tables

- `<table>` : Container for a table.
- `<tr>` : Container for one row of the table.
- `<th>` : Specify row/column header.
- `<td>` : Specify column data.

Comments

- `<!-- -->` : Anything between these tags are considered as comments and aren't parsed by the browser.

More info: https://www.washington.edu/accesscomputing/webd2/student/unit2/common_tags.html

HTML Entities

Whitespace characters

Whitespace is ignored by HTML parsers.
i.e. if you hit the space bar multiple times within a document, only one of them will actually be rendered by the browser.
To overcome this, we have: `&nbsp;` (non-breaking space).

Angle brackets

Words enclosed in angle brackets get parsed as tags by default.
To type angle brackets, we can use the entities: `&lt;` (<) and `&gt` (>).

More info: https://www.w3schools.com/html/html_entities.asp

CSS

Styling

The `<style>` tag can be used to define style rules for each element.

Selectors

Styles can be applied to whole tags, specific elements, or user-defined classes.

- Whole tags:

```
div {
    background-color: blue;
}
```

- Specific elements:

```
#someDiv {
    background-color: red;
}
```

- User-defined classes (best-practice):

```
.someClass {
    background-color: green;
}
```

Using classes is the best practice to organize your styles, because they act as a single source of truth for multiple elements and can be tweaked later/applied to more elements easily.

More info: https://blog.hubspot.com/website/css-tutorial

Class precedence and `!important` override

Multiple classes can be applied to the same element. A general rule of thumb is that the priority increases from left to right.

```
<div class="class0 class1 class2">
```

The `!important` keyword can be used next to an attribute in a class to force it to be important even if its priority in the list is low.

```
.class0 {
    background-color: green !important;
}
```

More info: https://css-tricks.com/precedence-css-order-css-matters/

Properties

Examples:

- background-color
- color
- font-family
- font-size
- font-weight
- width
- height

Stylesheets

Separating out styles for a document into its own dedicated file is considered to be best practice. This can be done by putting the contents of all `<style>` tags into a separate file (without the tags) with a ".css" extension and then linking it in the main HTML document.

To link external stylesheets add this snippet within the HTML doc's `<head>` tag:

```
<link rel="stylesheet" href="styles.css">
```

More info: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/Getting_started

Nomenclature and more best practices

Name your base HTML document `index.html`, because most web servers look for this file by default.
Make sure to name style classes and HTML element IDs meaningfully for improved readability and easier debugging.