Part One - HTML Basics

01 – Concepts Related to Webpages

1. **URL (Uniform Resource Locator)**: The address we enter in the browser.
2. **Webpage**: Each individual page displayed by the browser.
3. **Website**: A collection of multiple webpages forming a single site.
4. **Web Standards**:

02 – A Taste of HTML

1. Step 1: Right-click => New => Text Document => Enter the following content and save.

```
<details>
    <summary>The secret to mastering programming:</summary>
    Subscribe to Codex and Start Coding Today!
</details>
```

2. Step 2: Change the file extension to `.html`, then double-click to open it.

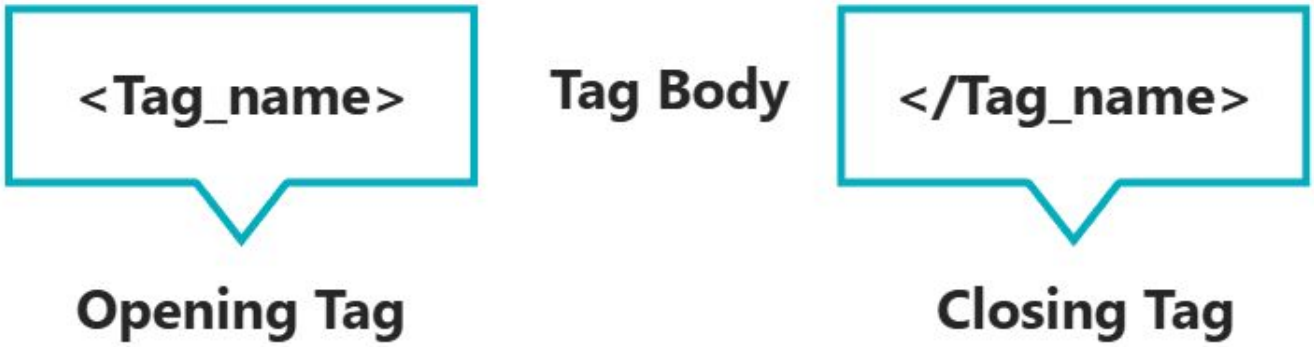
> The file extension `.htm` can also be used here, but the more standard `.html` is recommended.

3. Programmers write **source code**, which is handed over to the browser for rendering.
4. To view a webpage's **source code** using the browser, follow these steps:

> Right-click on a blank area of the webpage => Select **View Page Source**.

03 – HTML Tags

1. **Tags**, also known as **elements**, are the basic building blocks of HTML.
2. Tags are categorized into: **paired tags** and **self-closing tags** (the vast majority are paired tags).
3. Tag names are **case-insensitive**, but lowercase is recommended as it is more standard.
4. **Paired tags**:



Example Code:

```
<details>
    <summary>The secret to mastering programming:</summary>
    Subscribe to Codex and Start Coding Today!
</details>
```

5. **Self-closing Tags**:

Note: The / in self-closing tags can be omitted

Example Code:

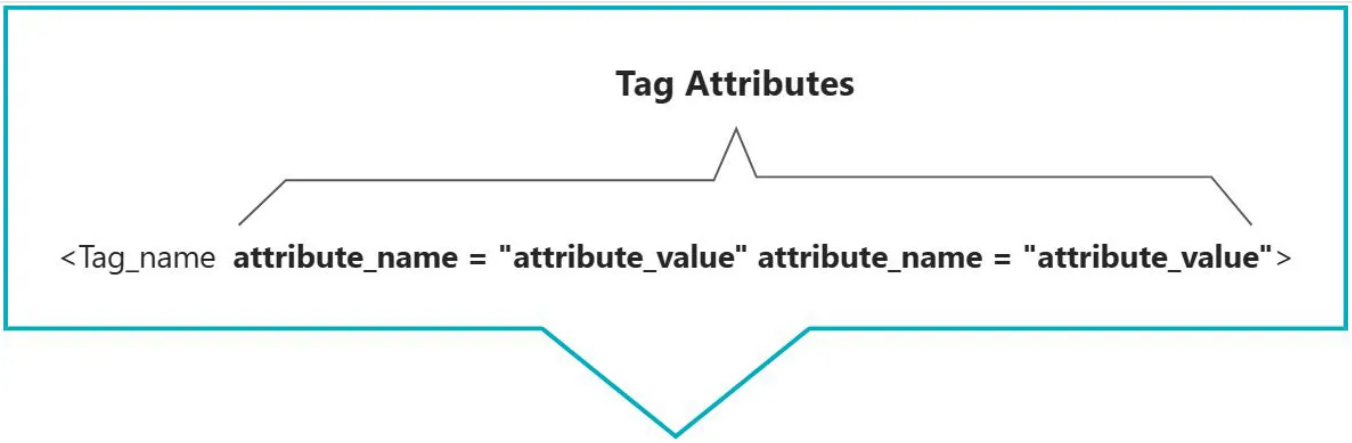
```
<input>
```

6. The relationships between tags: **parallel relationship** and **nested relationship**. Indentation can be done using the **Tab key**:

```
<details>
  <summary>The secret to mastering programming:</summary>
  Subscribe to Codex and Start Coding Today!
  <input>
</details>
<input>
```

04 – HTML Tag Attributes

1. Used to provide **additional information** to tags.
2. Can be written in the **opening tag** or **self-closing tag**, as follows:




```
// Example Code:
<progress value="45" max="100">
  45%
</progress>;
<input type="password" />;
```

3. Some special attributes do not have an attribute name, only an attribute value, such as:

```
<input disabled />
```

4. Points to Note:

> 1. Different tags have different attributes; there are also some common attributes (which can be written in any tag, to be summarized in detail later).
> 2. Attribute names and values should not be written arbitrarily, as they are defined by W3C.
> 3. Attribute names and values are case-insensitive, but lowercase is recommended.
> 4. Double quotes can be written as single quotes, or even omitted, but double quotes are still recommended.
> 5. Do not use the same attribute name multiple times within a tag, as the later ones will take effect. For example:

```
<input type="text" type="password" />
```

05 – Basic Structure of HTML

---

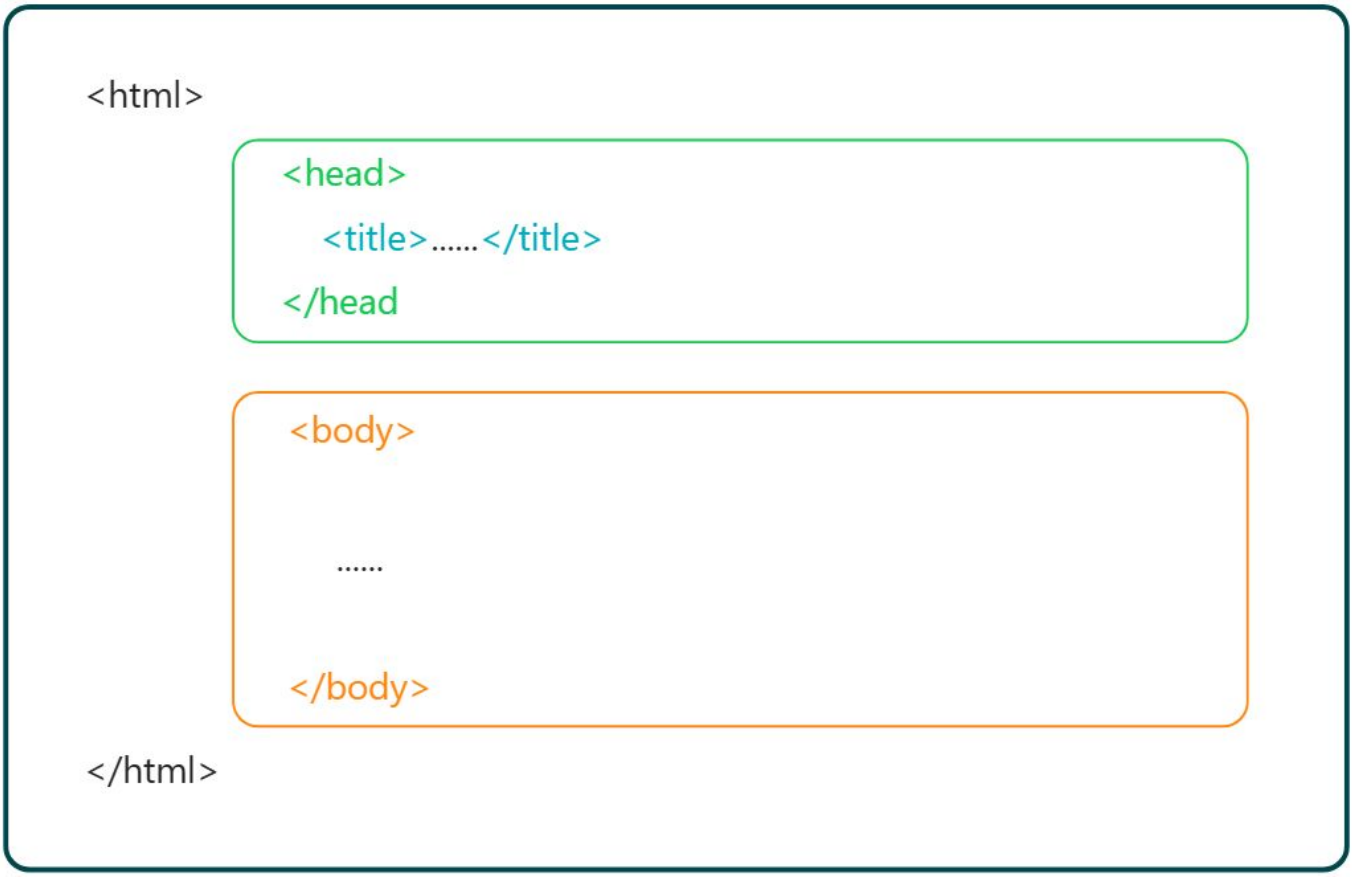
1. How to view the specific code of a section in a webpage? — Right-click and select "**Inspect**".
2. Difference between "Inspect" and "View Page Source":

> **View Page Source** shows: The source code written by the programmer.
>
> **Inspect** shows: The source code after being "processed" by the browser.
>
> **Note**: In daily development, **"Inspect"** is used the most.

3. The **basic structure** of a webpage is as follows:

> 1. The content you want to display on the webpage is written inside the `<body>` tag.
> 2. The content inside the `<head>` tag does not appear on the webpage.
> 3. The `<title>` tag inside the `<head>` tag can specify the title of the webpage.

4. Illustration:



5. Code:

```
<html>
  <head>
    <title>Website Title</title>
  </head>
  <body>
    ......
  </body>
</html>
```

06 – VSCode

1. Quick Introduction
2. Two methods to open a folder in VSCode
3. Adjust the font
4. Set a theme
5. Install the Live Server extension
    a. Makes it more convenient to open webpages

b. Provides a way to view webpages that how they would look after deployment

c. Automatically reloads the page when changes are made to the code.

d. Configure VSCode's auto-save settings based on your preferences

Note 1: Ensure you open a folder in VSCode; otherwise, the Live Server extension will not work properly!

Note 2: The opened webpage must follow the standard HTML structure; otherwise, auto-refresh will not function!

07 – Doctype

---

1. **Purpose**: To inform the browser about the version of the HTML.
2. **Syntax**:

- **Old Syntax**: The syntax depends on the version of HTML being used, with multiple variations.

For details, refer to the W3C website – Document Declaration (for understanding only, no need to memorize!).

- **New Syntax**: Everything has become simpler! The W3C recommends using the HTML5 syntax.

```
<!DOCTYPE html>
```

3. The doctype declaration must be placed on the very first line of the webpage and should be outside the `<html>` tag.

08 – HTML Character Encoding

---

1. **Computer Operations on Data**:

- During storage, data is **encoded**.
- During retrieval, data is **decoded**.

2. **Encoding and decoding follow certain standards — called character sets.**
3. **Common character sets** (for reference):

- **ASCII**: Includes uppercase letters, lowercase letters, numbers, and some symbols, totaling 128 characters.
- **ISO 8859-1**: Expands on ASCII by including some Greek characters, totaling 256 characters.
- **GB2312**: Further expands to include 6,763 commonly used Chinese characters and 682 symbols.
- **GBK**: Includes more than 20,000 Chinese characters and symbols, supporting Traditional Chinese.
- **UTF-8**: Contains all characters and symbols of all languages in the world — widely used.

4. **Principles of Usage**:

Principle 1: Ensure the appropriate character encoding is used during storage.
Otherwise: Data cannot be stored, and it may be lost!

Principle 2: Use the same encoding for retrieval as was used for storage.
Otherwise: Data corruption or garbled text may occur!

Example

Below text includes: English, Hindi, Japanese and Chinese.

I love you

मुझे तुमसे प्यार है

愛してます

我爱你

If ISO8859-1 encoding is used for storage, issues will arise the moment data is stored because ISO8859-1 only supports English!

To ensure all input can be stored and read properly, **UTF-8 encoding** is now almost universally adopted.

When writing HTML files, it's standard practice to use **UTF-8 encoding** for consistency.

5. Summary:

- When writing code, always use UTF-8 encoding (the safest choice).
- To ensure the browser renders HTML files correctly, you can specify the character encoding using the `<meta>` tag with the `charset` attribute.

```
<head>
<meta charset="UTF-8"/>
</head>
```

09 – `lang` Attribute

---

1. The `lang` attribute in HTML is used to declare the primary language of a document or specific elements. This helps browsers, translation tools, and search engines process the content appropriately.
2. Usage:
    a. defined in the `<html>` tag to specify the default language of the page

```
<html lang="en">
```

        This indicates the default language is English.

    b. specify a different language for individual elements

```
<p lang="fr">Bonjour, tout le monde !</p>
```

        This marks the content as French.

3. Language Codes

Use ISO639-1 two-letter codes. Examples include:

- `en` : English
- `zh` : Chinese
- `es` : Spanish
- `ja` : Japanese

Regional variations can also be specified:

- `en-US` : American English
- `en-GB` : British English
- `zh-CN` : Simplified Chinese
- `zh-TW` : Traditional Chinese

4. Why is it Important?

- **Accessibility**
  - Screen readers use the `lang` attribute to apply proper pronunciation rules.
- **SEO Benefits**
  - Search engines can better identify the target audience based on the declared language.
- **Translation Tools**
  - Online translation services use the `lang` attribute for accurate translations.

10 – Standard Structure

---

1. The standard structure of HTML is as follows.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>I am a title</title>
  </head>
  <body>
  </body>
</html>
```

2. In VSCode, type `!` and press Enter/Tab to quickly generate the standard structure.

In the generated structure, there are one `<meta>` tags that we don't need for now, so they can be deleted.

3. Configure the built-in Emmet plugin in VS Code to customize the default `lang` and `charset` attributes of the generated structure.

11 – Typography Tags

Tag Name	Tag Meaning	Type
h1 ~ h6	Heading	Paired Tag
p	Paragraph	Paired Tag
div	No specific meaning, used for overall layout	Paired Tag

1. h1~ h6 should not be nested with each other.
2. The p tag is quite special! It cannot contain: h1~ h6, p, or div tags (for now, just remember this; the rules will be explained later).

12 – Semantic Tags

1. **Definition of Semantic**:

- A word is **semantic** when it has a clear, defined meaning.
- In web development, **semantic HTML** elements describe the type of content they enclose and their role in the webpage structure.

2. **Semantic vs. Non-Semantic Tags**:

- **Semantic tags** provide clear, human-readable descriptions of their role (e.g., `<header>`, `<footer>`, `<h1>`, `<p>`).
- **Non-semantic tags** (e.g., `<div>`, `<span>`) are generic containers that don't convey any specific meaning, making them harder for browsers, search engines, and screen readers to interpret.

- **Examples**:
  - `<header>` : Represents the header of a layout.
  - `<footer>` : Represents the footer of a layout.
  - `<h1>` to `<h6>` : Represent headings, indicating the content's hierarchy.
  - `<p>` : Represents a paragraph, grouping text into logical blocks.
  - `<nav>` : Represents navigation links.
  - `<main>` : Represents the main content of the webpage.
  - `<section>` : Groups content into sections.
  - `<article>` : Represents independent, self-contained content (e.g., a publication).
  - `<aside>` : Represents secondary content (e.g., sidebar).

3. **Benefits of Semantic HTML**:

- **Intuitive & Readable Code**: Semantic elements make the code easier to understand and maintain.
- **Improved Accessibility**: Helps screen readers and assistive technologies interpret content more accurately.
- **Better SEO**: Enhances search engine optimization by providing clear content structure and meaning.

13 – Block & Inline Elements

---

1. Block level elements: take up an entire line
2. Inline elements: do not take up an entire line
3. Rules

1. Rule 1: Block level elements can contain both block level elements and inline elements (almost everything can go inside them)
2. Rule 2: inline elements can contain other inline elements, but can not contain block level elements.
3. Special rule: h1 ~ h6 can not be nested inside each other
4. special rule: p tag can not contain block level elements

14 – Text Elements

1. **Used for wrapping:** words, phrases, etc.
2. **Usually written inside Typography tags.**
3. **Typography tags are more macro-level** (used for large sections of text), while **text tags are more micro-level** (used for words and phrases).
4. **Text tags are typically inline elements.**
   Real-life analogy: A `div` is like a large packaging bag, while a `span` is like a small packaging bag.

Commonly Used Text Tags

Tag Name	Tag Semantics	Type
`em`	Content that needs emphasis	Paired
`strong`	Very important content (stronger tone than `em`)	Paired
`span`	No semantics, used as a general-purpose container for phrases	Paired

Less Commonly Used

Tag Name	Tag Semantics	Type
`cite`	Title of a work (book, song, movie, TV show, painting, sculpture)	Paired
`dfn`	Special term or specific noun	Paired
`del`	Deleted text	Paired
`ins`	Inserted text	Paired
`sub`	Subscript text	Paired
`sup`	Superscript text	Paired
`code`	A block of code	Paired
`samp`	Content extracted from normal context, e.g., device output	Paired

`kbd`	Keyboard text, representing text input via keyboard, often used in computer-related manuals	Paired
`abbr`	Abbreviation, preferably with the `title` attribute	Paired
`bdo`	Change text direction, requires `dir` attribute, values: `ltr` (default), `rtl`	Paired
`var`	Marks a variable, can be used with the `code` tag	Paired
`small`	Subtext, such as copyright or legal text; rarely used	Paired
`b`	Keywords in abstracts or product names in reviews; rarely used	Paired
`i`	Originally for: thoughts, speech, etc. Now often used for: displaying font icons (covered later)	Paired
`u`	Text with contrast to normal content, e.g., misspelled words or inappropriate descriptions; rarely used	Paired
q	Short quotation; rarely used	Paired
`blockquote`	Long quotation; rarely used	Paired
`address`	Address information	Paired

Notes:

1. These less commonly used text tags don't need to be overthought during coding (use them as needed; it's fine not to use them).
2. `blockquote` and `address` are block-level elements, while other text tags are inline elements.
3. Some tags with weak semantics are rarely used, such as: `small`, `b`, `u`, `q`, `blockquote`.
4. There are too many HTML tags! Just remember the important, semantically strong tags, such as: `h1~h6`, `p`, `div`, `em`, `strong`, `span`.

15 – <img> tag

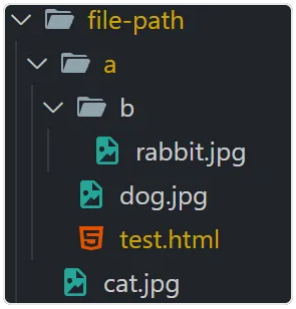
Tag Name	Tag Semantics	Common Attributes	Type
`img`	Image	**`src`** : Image path (also known as image address) — specifies the exact location of the image. **`alt`** : Image description. **`width`** : Image width, measured in pixels, e.g., `200px` or `200` . **`height`** : Image height, measured in pixels, e.g., `200px` or `200` .	Paired Tag

1. **Pixels (px)** are a unit of measurement. We will cover this in more detail when we study CSS.
2. Avoid modifying both the width and height of an image at the same time, as this may distort its proportions.
3. For now, consider the `<img>` tag as an **inline element**. (When we study CSS, we'll learn about a new classification of elements. Currently, we only know about **block** and **inline** elements.)
4. The purpose of the `alt` attribute:
    - **Primary purpose:** Search engines use the `alt` attribute to understand the content of the image.
    - When an image cannot be displayed, some browsers will show the value of the `alt` attribute instead.
    - Screen readers for visually impaired users will read the value of the `alt` attribute aloud.

16 – File Path

1. **Relative Path**: Establishes a path with the current location as the reference point.

Existing structure	symbols	meanings	examples

	./	current folder	<img src="./dog.jpg">
	/	**child folder**	<img src="./b/rabbit.jpg">
	../	parent folder	<img src="../cat.jpg">

- `./` in relative paths can be omitted.
- Relative paths depend on the current location. If the file location changes later, the paths in the file must also be updated.

2. **Absolute Path**: Establishes a path with the root location as the reference point.

- Local Absolute Path:
  Example: `E:/a/b/c/cat.jpg` (rarely used).
- Web Absolute Path:
  Example: `http://www.abc.com/images/logo.png`.

- Using **local absolute paths** can be cumbersome, as paths need to be updated when switching devices, so they are rarely used.
- Using **web absolute paths** is convenient but has potential issues: if the server enables hotlink protection, images might fail to load.