FM
(Beginner Learning Path) HTML

COMPLETE INTRO TO WEB DEV
Brian Holt ← Instructor

HTML, CSS, JS are the three necessary components to creating a website from scratch.

HTML → HyperText MarkUp Language
JUST Content on a webpage
Also the structure of the content

Analogy: Materials used to build the car
Frame, Door, Steering Wheel, Tires, etc
By themselves, the materials are inert: they don't do anything on their own or look particularly.

CSS → Cascading Style Sheets
Style of a website (the look and feel of the website)
Since styling is separate from the content, styling can be reused

Analogy: The color of the car, all the styling details, is it left drive or right
Basically how the car looks and feels

JS → Javascript (Programming Language)
A website made with just HTML and CSS is completely valid
Interactivity of the website Example: Blog Pages
Insert content

HTML } Not technically
CSS } programming languages
More like a word doc
Javascript is the primary language that can run inside a browser

Analogy: HTML and CSS → Non-interactive website
Pages in a book (No interactivity other than just turning the pages and reading the content)

Going from a book to an interactive tablet experience
→ Javascript

HTML

Base building block in HTML is tag
Tags describe what's inside it
Every tag has an opening and a closing tag
While some tags open and close at the same point
<tag></tag>

<h1></h1> Heading tag
Biggest title on the page

→ Grouping of text
 Paragraph tag ← Paragraph of text together inside of a p tag
Smaller font size

However, later we can go and use CSS, the styling component, to make things smaller / bigger

What we are basically communicating here by what kind of tag we are choosing is what the meaning of this particular bit of text is

-ic
the actual meaning behind something, not just how it looks

Heading tags: h1 → h6
(Headings, subheadings)
Self closing tags / Void tags ⇒ <input />
→ optional

Just the opening tag ← <input> is also a valid
without the closing tag self closing tag

Tags can be nested as well (Infinite nesting is possible)

 → Division tag

The most recently opened tag must be the first one closed

Analogy: It's like writing a word doc but instead of highlighting something (make it italics), we are going to put a tag around it that says this is emphasised text

A good kind of information taxonomy ⟹ $\underbrace{\text{Title of a whole page}}_{\text{an h1}}$

There is an implied info hierarchy by making the heading smaller and smaller from h1 to h6.

The white space between the tags is not significant, the amount of white space is not important in HTML.

<a $\overbrace{\text{href=" "}}^{\text{Attribute}}$></a> Anchor tags

Links (to somewhere else)

href indentifies where it goes

But only the content inbetween the tags is displayed to the user, not the actual link which is where we are taken to

They basically hold stuff together so we can move / style / organize it as a group

 Division tags → Generic divider between something

Think of them as $\underbrace{\text{cardboard boxes}}_{\text{is more defined by what's in it makes a difference than what the cardboard box itself}}$

Basically we are identifying this part of something is distinct from something else.

Analogy: Filing cabinet
Dividers between various types of files

 will start making more sense once we start getting into CSS

 Span tags
Ziploc bag
Smaller idea of a div
Used to emphasize different parts of text
Denote that this part of text is different than the rest of the text
Span itself does not do anything, it allows CSS/Javascript to make the individual span do things

→ Used when the order is important

<ol></ol> Ordered list
<ul></ul> Unordered list
} List of information

<li></li> List items

If we switch the order of two of the items in the list, will the list still be correct?
True ⇒ No order (ul)

→ Used to signal a user what the button will do

<button></button> Button tag → Javascript is required for the button to react to something
It does not do anything

Analogy: Doorbell that someone installed on a house and did not actually hook up to anything

<img /> Image tags
Images belongs in the content (they live in HTML) more than in style

Images can be used with CSS (sometimes) ← Background Images

Analogy: Content images → Newspaper
Big image of whatever the top story is

<image src="" alt="" />

source (Address of where the image lives)

alt text (If someone is blind or if it is hard to see for them) → The screen reader will tell them what you have put in here

The image tag is valid without the alt attribute but adding alt is the ethical thing to do.

Google and Bing will index it, it will crawl our website, it'll find all those alt texts. Search engines cannot see images like humans can, since alt text tells them what it is, they can index it and show it when someone searches for that.

Sometimes we might want to accept information from the users → The particular part where we accept user input

User input

Browser inputs → Inputs that the browser provides for you (Device type, Screen size, language, location, etc.)

Many types of pre built components that allows us to accept info from a user

Same { <input /> <input type="text" /> ← Default

<input type="color"/>
<input type="file"/> A server is required to upload it to but this allows the user to select which file they want to
<input type="number"/>
<input type="datetime-local"/>
<input type="radio"> Radio Buttons ⊙
<input type="checkbox">
<textarea></textarea>
Long form text responses
Drag to make the text area bigger / smaller

Select and option

* Sometimes we might want to limit a user to a certain group of options to select from
* User interactive input that a user can select an option from a drop down menu

```
<select>
    <option></option>
        ⋮
    <option></option>
</select>
```

attribute ⇒ value

<form></form> Form tag
Group of inputs, text areas, selects related to gathering data from a user
html tags

<input placeholder="First Name"/>

First Name

Again the form itself does not do anything
Javascript and server needs to kind of hook up a form

This is just the part of the HTML that will accept all the information from the user.

Tables are really useful for something that we

could use spreadsheet for ⇒ Tabular data

```
<table> Creates a table for displaying data in rows and columns
    <thead> Semantic grouping for header
        <tr> Creates a table row
            <td> Names </td> Table data cell of the row created (usually column 1)
            <td> Points </ td > (usually column 2)
        </tr>
    </thead>
    <tbody> Semantic grouping for body
        <tr>
            <td> Horimiya </td>
            <td> 99 pts </td>
        </tr>
        <tr>
            <td> Suzume </td>
            <td> 98 pts </td>
        </tr>
    </tbody>
</table>
```

Names	Points
Horimiya	99 pts
Suzume	98 pts

→ We have created basic spreadsheet using the table tag

Technically a div can be used for anything (100% of our tags on our page could be a div)

Analogy: A div is a cardboard box and we can make various diff cardboard boxes into our cardboard fort of a website

But it is imp to use the thing that matches it the closest. Some sort of assistive tech (screen reader) can go and find the appropriate semantic tag we have used on the website and tell the blind person.

This is called writing semantic markup / semantic HTML
By just following these kinds of best practices, all of the assistive tech will just work for free and we don't have to do a lot of work to make our site work for everybody.

Choosing tags that match our meaning ※

<marquee> </marquee> Old tag
Scrolling text Depricated (Removed from HTML)

Comments
Never gets shown to the user
It's left there for future coders and for our future self to be like that's why I did this thing

Can be multiline as well
<!-- --> Comment tag ※
Combining them all into one gliff → Ligature

Imp to leave future notes to someone (this is why I did this)
Later if you are gonna be unclear why I did this, it's better to leave a comment
This was done this way, for this reason

Do not leave comments for the sake of it
If you are gonna comment, make it useful
Balance between commenting way too obvious things and leaving no comments
Make sure that you are leaving an appropriate amount of comments
Comments can be visible to anyone who views the page source / inspects the code in their browser
It is not intended to keep confidential / internal / private info secure

Hard Returns
HTML does not respect the whitespace inbetween the tags

```
<p>
    This is some text.

    This is some more text.
</p>
```

This is some text. This is some more text.

Correct way to split the text / paragraph (since white space is not considered by HTML)
Using 2 paragraph tags instead of using one paragraph tag adding a white space between the paragraphs trying to separate them

Wrong Way (If for some reason, we have to separate it using just one paragraph tag)

LINE BREAK

 Break tag can be used (Sometimes this might be useful to us if we have some constraints)
Will make a break in the text
Technically this works
Modern HTML programmers do not use break

<p> tag has a semantic meaning

 tag has no semantic meaning → invisible to the screen readers
It is just a visual line break
When using <br> to separate paragraphs, we are faking the visual separation without communicating the actual structure. We are telling the browser and screen readers, search engines etc, that it is just one para but it's actually two distinct thoughts

→ Uncommon in today's website

 tags can still be used when we need a line break within a single block of content

Address
Poetry
Names and designation

Every time they release new versions of HTML and CSS and Javascript, they add things to the language and to the new capabilities but they don't take away old things. Because they want websites from 1995 to work. Websites never break. But the negative thing is that we are left with a bunch of really old stuff that still technically works and at times, it becomes less clear of what's the correct way of doing things.

Attributes

Tags can have multiple different capabilities built into them

We can modify the behavior of a particular tag to be in a specific way

What kind of attribute it is

<typeoftag key="value">

Attribute

What we are setting the attribute to be

The attribute changes what type of tag is going to be

What kind of tag it is will determine what kind of attributes that could have

If we add an attribute to a type of tag that does not understand the concept of that particular attribute the browser just ignores it

HTML → If the browser does not understand something, it just ignores it and moves on

href and type are specific to a few tags

Almost everything can have a class attribute

Classes are the way that we kind of denote what type of tag it is to us

Documentation to our self

If we give classes useful names, it will be easier later to come back and do all our CSS → We can style it based on our class names

The classes can have multiple class names in a tag

It can be separated using white spaces

And we use dashes here for the spaces of the class name
They themselves are inert

Classes are mostly important for CSS and its kind of important for ourself to know what type of HTML we are looking at

A class is reusable

Classes are much more useful than IDs
IDs are important though
99.9% → We should use classes
Not IDs
IDs should be used very sparingly
Classes are not unique

ID is unique, they are designed to not be reusable, they are unique to the page

When you are gonna have only one of that type of tag on our page guaranteed, we are not going to reuse this on the page → Acceptable reason that we might wanna use an ID

We can use id tag to link a part of our page using the a tag directly

Organizing HTML
Storing all the like things together inside a di
Functionally there is no difference between article and div
Bit of info
Everywhere that we can use a div, we could use an article
more specific thing
and vice versa

It is not much as an article in a website
→ It could be like a tweet
Unit of info

Article is like a cardboard box that has a label on it that says this is this type of box

<section></section>

Analogy: If we were designing a house in HTML

Kitchen ← Section

Bathroom

Bedroom

Larger macro unit of a website

Article tag and Section tag → Semantic tag

Classes is a methodology of documentation for ourself

We write code once and maintain it forever

Optimize code for the forever part, not just for the writing part of it

Analogy: If we are building an house and if we come up with one blueprint, we can reuse that blueprint over and over again

(By kind of defining it once)

If we write code using classes and we create a structure, we can reuse it over and over again

Defining reusable components for ourselves to use later

A set of HTML defined and reused it over and over again

Every tweet looks structurally the same

They just have different content inside of them

They reuse the same structures and classes over and over again

Navbar

<nav></nav> → A type of div

more specific thing for navigation ← user can get around the website

has unordered list of things

<a href="/"> → It'll take us to whatever the root domain is

<a href="/about"> → Relative to the root domain

means it goes all the way back to the domain and start the domain from there

https:// → Protocol

Without the slash, it will be relative to where we are right now

Replaces the last slash with the current inputed one

In general, if we have a list of things and they are the same things (list of links, list of blog posts, list of favorite pets) → Unordered list can be used

But it's totatally upto us to just use divs inside of divs (Valid)

We get a lot of preference here in HTML (Whatever makes sense to us)

Classes are not case sensitive

We can name classes in anyway that makes sense to us

Naming things is hard

Why do we not need the file extension HTML in the path?

Sometimes we do

It depends on whatever our web server is doing

At times Next.js is used to generate websites

The URL that it generates don't have .html at the end

If we are not using Next.js (in other cases), we might

need that .html because it's referring to a file on the disk

It depends on whatever is giving the HTML file to the client at the end (whether it expects the .html / not) → Whatever our web server expects that's what we are gonna put in there

Article HTML tag can be used to represent a self contained piece of content (social media post)

A div acts like a container (cardbox) that encapsulates and groups related HTML elements together, helping to locally divide and organize content

→ self contained unit of info

<article>

<section>

Functionally similar
Diff levels of semantic meaning

→ generic
→ thematic grouping of content

Naming HTML Classes

Case insensitive
Conventionally kept lowercase
Use dashes to separate words
Descriptive and meaningful

Head and meta tags

Structuring Part of HTML

Meta tags ⇒ Title of the document (Name and logo on top of the tab)
Meta info about the website

It's all the information about the doc that does not necessarily get displayed

Convention: Root of our project ⟸ It is called index.html

Does not have to be
We could call it something else
But again, the main page

The very first thing in every HTML doc is we have to tell the browser this is the flavour of HTML I'm giving you

There are other million types of doctypes that we never have to care about

If we were really going to make a blog website, do we need to use a content management system?

It depends
The course website created solely for this course
CMS was not used
It's written in markdown → Flavour of writing text
The content and text written in markdown gets turned into the website using Next.js
There are so many correct ways to do this

Lorem ipsum generator / VS code's Lorem text expansion feature ('Lorem n' and tab)
No. of words we might need

→ Generates placeholder text