# Front End Development Notes

[Front End Development Notes 1](#_Toc36547532)

[HTML5 3](#_Toc36547533)

[HTML 4](#_Toc36547534)

[Typical starting structure 5](#_Toc36547535)

[head Element 6](#_Toc36547536)

[Basic Tags 7](#_Toc36547537)

[body Element 7](#_Toc36547538)

[header Element 7](#_Toc36547539)

[main Element 7](#_Toc36547540)

[footer Element 7](#_Toc36547541)

[Text Tags 8](#_Toc36547542)

[h1 - h6 Element 8](#_Toc36547543)

[p Element 8](#_Toc36547544)

[strong element 8](#_Toc36547545)

[b element 8](#_Toc36547546)

[em Element 8](#_Toc36547547)

[i element 8](#_Toc36547548)

[link 9](#_Toc36547549)

[Tables 10](#_Toc36547550)

[Video & Audio 12](#_Toc36547551)

[CSS3 14](#_Toc36547552)

[CSS 15](#_Toc36547553)

[File Organisation 15](#_Toc36547554)

[Syntax 16](#_Toc36547555)

[Selectors 16](#_Toc36547556)

[Properties 16](#_Toc36547557)

[Specificity 17](#_Toc36547558)

[JAVASCRIPT 18](#_Toc36547559)

[Media Element Player - mediaelement js 19](#_Toc36547560)

# HTML5

A close up of a sign

Description automatically generated

## HTML

HTML = Hypertext Markup Language

An HTML document provides the content layer and structure of a web page.

Hypertext is any text that can be displayed on a computer screen and contains links to other text or hypertext documents.

A Markup language provides meaning to text in a document using instructions that describe how text should be structured, formatted and laid out.

Every HTML page starts with a Doctype. It is an instruction to the browser about what version of HTML the page is written in.

Everything following this should be contained in the HTML tags. This lets the browser know that everything between these tags is HTML code.

The head element contains information about the page, such as title, description and links to CSS/js files etc. Most head elements are not viewable on the webpage.

Metadata is information about data. A metadata tag provides metadata about the HTML page.

The body element is where anything that you want to show in the browser goes.

## Typical starting structure

#### <!DOCTYPE html>

#### <html lang=en>

#### <head>

#### <meta charset=”UTF-8”>

#### <title>Page Title</title>

#### <meta name="description" content="add description here up to 160 char"/>

#### <meta name="keywords" content="add keywords here">

#### <meta name="author" content="add author here">

#### <meta name="viewport" content="width=device-width, initial-scale=1.0">

#### <link src=”css/main.css” rel=”stylesheet”>

#### </head>

#### <body>

#### <header>

#### </header>

#### <main>

#### </main>

#### <footer>

#### </footer>

#### <script src="http://code.jquery.com/jquery-1.11.3.min.js"></script>

#### <script src="js/main.js" type="text/javascript"></script>

#### </body>

#### </html>

## head Element

The head element contains machine readable information (metadata) about the document.

<title> - The documents title. This is displayed on the browsers title bar or page's tab.

<base> - Specifies the base URL for all relative URL's in a document.

<link> - Used to link to an external resource such as a CSS file, or an external source such as Bootstrap or google fonts.

<style> - CSS information

<meta> - Metadata that is not already represented by other head elements.

charset - Declares the page's character encoding. It must contain a standard IANA MIME name for character encoding. Developers are encouraged to use UTF-8.

Viewport - gives hints about the initial size of the viewport. Without this setting mobile devices will render pages as typical desktop widths. Include the below on all documents.

#### <meta name="viewport" content="width=device-width, initial-scale=1.0">

<script> - Used to embed or reference executable code such as JavaScript.

src links to external document.

type specifies file type. For JavaScript use text/javascript.

<noscript> - Defines a section of HTML to be inserted if a script type on the page is unsupported or if scripting is turned off in the browser.

<template> Mechanism for holding HTML that is not to be rendered as soon as the page is loaded but may be created later during a runtime using JavaScript.

## Basic Tags

### body Element

Contains the data that will be displayed on the document. There can only be one body element in a document.

### header Element

Represents a container for introductory content and/or a set of navigational links. You can have several header elements in one document.

A header element cannot be placed with a footer, address or another header element.

### main Element

Specifies the main content of a document. That content should be unique to the document and not contain content that is used across multiple documents, such as header, footer, nav etc.

There must only be one Main element on each document.

### footer Element

Defines the footer section of a document. Typically contains copyright, contact and nav elements. You can have multiple footer elements on a document.

## Text Tags

### h1 - h6 Element

The <h1> to <H6> tags are used to define headings. H1 is the most important and H6 is the least important.

The choice of heading should be based on the importance of the text rather than the required style or size. This is for accessibility to allow screen readers to identify the importance correctly.

Example:

H1 - Overall page title

H2 - Page Subtitle

H3 -Titles of main sections

H4 - titles of the subsections

### p Element

The p tag defines a paragraph of text. The browser will ignore any line breaks in the paragraph unless adding a line break tag.

### strong element

Defines important text. Will display as **bold** in all browsers

### b element

Used to mark keywords. Will also display text as **bold**.

### em Element

Used to emphasise text. Will display in *italics*.

### i element

Used to mark text as "alternative text". Common examples include book titles, technical terms or foreign words. Also displays in *italics*.

## link

The a tag is used to link to various locations such as links within the website, to external documents as well as locations within the existing document.

href is used to define the target, for example:

<a href=”URL”>Link</a> to link to a URL address

<a href=”mailto:email@address>e-mail<a> to open an e-mail client with the e-mail address pre-loaded.

<a name=”name”></a> sets the destination for an internal link.

<a href=”#name”>internal link</a> to link to the internal link

target specifies where to open the linked document.

\_blank - opens in a new window or tab

\_self - opens in the same frame as it was clicked. This is the default if nothing is entered.

\_parent - opens in the parent frame

\_top - opens the document in the full body of the window

framename - opens the document in a named frame.

## Tables

Anything that you might put into a spreadsheet could be put into a table. Tables should only be used for tabular data and not layout as there are other options for this, except in HTML e-mails.

Tables are often populated with data from a database.

Example code:

#### <table>

#### <thead>

#### <tr>

#### <td>Name</td>

#### <td>Occupation</td>

#### </tr>

#### </thead>

#### <tfoot colspan=2>

#### <tr>

#### <td>Table of names and occupations</td>

#### </tr>

#### </tfoot>

#### <tbody>

#### <tr>

#### <th scope=”row”>David</th>

#### <td>Accountant</td>

#### </tr>

#### <tr>

#### <th scope=”row”>Paul</th>

#### <td>IT</td>

#### </tr>

#### </tbody>

#### </table>

<table> - Defines the HTML table.

<thead> - Defines the column headers. Not necessary but good for accessibility. Also allows the table to be scrolled independently of the table body which is good for large amounts of data. Will be styled differently to the body.

<tbody> - Wraps all table data, do not include head of foot within this.

<tfoot> - Used for any notes about the table or data such as sum, average etc. Should be located between the head and body but will show at the bottom of the table.

<tr> - A row of the table

<th> - Table header, not the same as thead. Can be used as a header for rows or columns using the source=”row” attribute.

<td> Table cell. No content should be outside of a cell. The number of cells in each row determines the number of columns.

<caption> - Adds a caption /title for the table.

Colspan - Allows a cell to span more than one column.

## Video & Audio

<video> - Tag used for videos in HTML. Tag can be self-closing if source tag not used.

<audio> - Tag used for audio in HTML. Tag can be self-closing if source tag not used.

<source> - tag used to specify the source of the video or audio, used when there is more then one source. More than one source is necessary for different file types supported by different browsers.

src - Source of video or audio.

type - video/filetype or audio/filetype such as mp4, mp3 etc.

controls - adding controls attribute to video / audio tag will add controls to the player.

autoplay - adding autoplay attribute to video / audio tag will play the file when the webpage loads.

File sizes and formats

File sizes are important to reduce page load times.

VLC media player can compress files.

Can use H.264 codec, AAC or mp3 for audio and mp4 as the video container for Safari / Chrome / Internet Explorer 9+ and mobile devices.

Use Theora video codec, Vorbis audio codec and Ogg video as the videon container for other browsers.

Captions

Create a VTT file containing:

WEBVTT FILE

0 🡪 Index of the caption

Start time 🡪 end time (format as 00:00:00:000 H:M:S:MS)

[Caption] 🡪 Whatever you want to appear over the video.

Use [https://quuz.org/webvtt](https://quuz.org/webvtt/) validator to check that the file is valid as it is easy to make mistakes.

Add the caption file between the video tags using the track element which is self-closing.

Attributes to use are:

label - description / label of the caption

kind - subtitle / description etc.

srclang - language, eg. "en"

src - location of VTT file

default - adds captions by default without clicking on CC button.

# CSS3

A close up of a sign

Description automatically generated

## CSS

CSS = Cascading Style Sheet.

CSS provides the presentation layer and creates the visual style of websites using colours, typography, layout, etc.

There are 3 different ways to write CSS:

* Inline - CSS is written in the HTML file in line with the tag, e.g. <div style="color:black">
* Internal - added within <style> tags in the head section of the document. Reference HTML elements using tags, id, class etc.
* External - This is the most common way of writing CSS. Link to an external CSS file within the head of the HTML file using <link href="*stylesheet location*" ref="stylesheet">. If using more than one external stylesheet the main stylesheet should be the last of all CSS files so that it overrides any previous files.

Browsers set their own default styles, called the user agent stylesheet. Some styles are inconsistent between browsers so can make web pages appear different in different browsers. A CSS reset can be used to remove the browsers default styles. Use the following two links to achieve this:

#### <link href="css/reset.local.css" rel="stylesheet" type="text/css">

#### <link href="css/normalize.css" rel="stylesheet" type="text/css">

### File Organisation

When working on large projects or in teams it is often smarter to use a modular approach to CSS. Splitting files across a team can increase overall productivity. Files could be split as follows:

* Typography.css - Font-faces, weights, styles for body and headings.
* Forms.css - Styles for form controls and labels.
* Lists.css - list-specific styles.
* Tables.css - table-specific styles.

## Syntax

CSS syntax is a selector followed by a declaration.

Selector - Points to the HTML element that you want to style. This could be achieved by using the tag name, id, class etc.

Declaration - A declaration contains a CSS property name and value separated by a colon. The declaration always ends with a semi-colon. Multiple declarations can be made within a block. The declaration block is surrounded with curly braces {}.

### Selectors

Tags are referenced as just the tag name with nothing preceding.

Classes are preceded with a full stop (.). Classes can be used on multiple elements.

ID's are preceded with a hash (#). ID's can only be used on one element.

The more specific you are with a selector the higher priority it will be given, e.g.

div p - Targets all p elements within a div element. This will take a higher priority than just a p tag. It will also mean that only p tags within a div will be targeted.

div > p - Targets all p elements that are a direct child of a div element. A direct child is not placed within another child.

\* selects all elements. This is not usually recommended as it can be slow.

div#id - Finds all divs with a certain id. Can also be used with classes.

:nth-child(i) - Finds the child at the index i, of the element before the colon. Can also have first-child and last-child. Can also use (an + b) as the index. n is a counter starting at 0 and b is an offset value, for example, 3n + 2 = 2, 5 and 8th child. (3 \* 0 + 2 = 2, 3 \* 1 + 2 = 5, 3 \* 2 + 2 = 8 etc).

:nth-of-type(i) - Similar to nth-child but looks for the specific element before the colon at an index i. Can also use first-of-type and last-of-type. Can also use (an + b).

:hover - Specified what happens when the mouse hovers over the element.

::before - Insert something before the element. Use content property to specify what is to be inserted.

::after - As above but is inserted after the element.

:checked - Can be used to style an input that is checked. Could be used for a CSS only dropdown.

### Properties

Properties define the style of the element.

A good reference for properties is <https://cssreference.io/>

Each property is entered on a new line, separated with a semi-colon.

Adding !important to the end of a property will override all other properties.

## Specificity

Specificity can be thought of as a score or rank that determines which style is applied to an element.

The universal selector (\*) has a low specificity while ID(#) has a high specificity.

There are four categories which define the specificity level of a selector:

* Inline styles - included in the tag in the HTML file.
* A - ID's specified to the element with a #.
* B - Class attributes and Pseudo classes. This includes .classes, [attributes], and Pseudo classes such as :hover.
* C - Elements and Pseudo elements. This includes element tags such as h1 or div and Pseudo classes such as ::before and ::after.

A, B & C combine to give a specificity value.

* A - Count the number of ID's
* B - Count the number of class selectors, attributes and pseudo classes.
* C - Count the number of Elements and pseudo elements.

A, B, C Example

Id selectors have a specificity value of 1, 0, 0 #foo

Class selectors have a specificity of 0, 1, 0 .className

Class and pseudo have a specificity of 0, 2, 0 .className:hover

Element selectors have a specificity of 0, 0, 1 h1

If two selectors are of equal value, the cascade kicks in and the rule that appears last will be applied.

Try to choose low specificity to make it easier to re-use CSS.

# JAVASCRIPT



## Media Element Player - mediaelement js

Media Element is a 3rd party library that allows you to add useful controls to video and audio on a web page.

It allows the controls to have a consistent look across all browsers. The standard video and audio controls will look different on each browser.

Adding Media Element

Files can be downloaded from here: <https://www.mediaelementjs.com/>

Four files are needed, all located in the Build folder:

1. Mediaelement-and-player-min.js
2. Mediaelementplayer.css
3. Mejs-controls.svg
4. Jquery.js ?

Link to these files from the HTML file above the main CSS file.

If not customising add class=”mejs\_\_player” to the video or audio tag.

Customising

Remove or don't add the mejs\_\_player class to the video tag.

Add the following script above the closing body tag of the HTML file.

#### <script>

#### $(.video, audio).mediaelementplayer({

#### Features: ["playpause", "tracks", any plugins etc

#### startLanguage: "en"

#### ]});

#### </script>

Configuration options can be found here: <https://github.com/mediaelement/mediaelement/blob/master/docs/api.md>

Additional options can be added as plugins. Plugins add features such as skip back, speed etc. Plugins can be found here: <https://github.com/mediaelement/mediaelement-plugins>.

Plugins are stored in the dist folder.

In the HTML file, load the plugin js file after the media player main js file. Add the plugin CSS file before the main CSS file.

Add the plugin name to the into the configuration script as shown above.