HTML Booklet

By James Davies

Table of Contents

Intro

Hyper Text Markup Language (HTML) is a Mark Up language, so it has no functionality on its own.

HTML is what is used to build up web pages and therefore websites.

HTML files end with '.html' and web document files always start with '<!DOCTYPE html>'.

The website home page must always be 'index.html' or any other file name as long as 'index.html' redirects to the home page.

All HTML files begin with the <html> tag and end with the </html> closing tag. The next section will explain about what tags are.

Section 1 – Understanding tags

HTML is built up of these things called 'tags'. An example of a tag could be `<p>`. That tag is called the paragraph tag. It is used to create a big piece of text, whereas `<span>` is a short piece of text.

When you are finished with the text in a tag, you need to close it. You do it like this:

```
<span>text inside tag</span>
```

You just use a '/' before the name of the tag.

You can also format it across rows:

```
<p>
This is paragraph text.
</p>
```

Here is a list of basic tags:

HTML	Name	Description
<p>	Paragraph text	Large amount of body text
	Span text	Small amount of body text
<h1> to <h6>	Headings	Heading text
 	Line Break	Starts a new line. Does not need closing tag

For example,

```
<html>
<h1>Welcome To My Website!</h1>
<span>By Bob</span>
<p>
     Hello and welcome to my corner of the web!
     My name is Bob and I like coding.
</p>
</html>
```

Would look like:

Welcome To My Website!

By Bob

Hello and welcome to my corner of the web!

My name is Bob and I like coding.

Section 2 – Using Attributes

In HTML, there is a way of customising and labelling your tags. They are called attributes.

An example of an attribute could be if you want to centre align a piece of text. You would use the 'align' attribute and then the 'centre' value. That would look like this:

```
<span align="center">Centred Text</span>
```

Warning!

HTML is written in American English so words like grey and centre are written as gray and center.

That align="center" attribute will make the text centred:

Centred Text

There is another tag called an Anchor Tag (`<a>`) which is used as a hyperlink (like this). We use the href attribute and the value being the URL. The Anchor Tag hyperlink is formatted like this:

```
<a href="https://example.com/">Click Here</a>
```

This HTML creates a hyperlink for the url "https://example.com/".

Another tag is the Image Tag (`<img>`) is used for displaying images from another source. It is the same as the Anchor Tag but instead of using the href attribute we use the src (source) attribute. The Image Tag also doesn't need a closing tag as it doesn't have any text in it. The Image Tag is formatted like this:

```
<img src="image.png">
```

This will display the image file 'image.png'.

Here is a table of some basic attributes:

Name/HTML Attribute	Description	Example
Height, Width	Specifies the height or width of an element	<button width="100px" height="20px">

Href	Specifies the hyperlink location	<a href="https://example.com/>
Src	Specifies the location of a file	<img src="image.png">
Style	Specifies the styling for an element (see Section 3)	<p style="font-family: Arial;">
OnClick	Runs a JavaScript function when clicked on	<button onClick="run()">

Section 3 – Cascading Style Sheets

Cascading Style Sheets (CSS) is a file type used to add styling to an HTML Web page.

Without styling a page would look a bit like this:

Welcome to Styli!

Welcome to this webpage. Enjoy all the styling tools on this site.

Copyright KC Publishing 2025 | All rights reserved

But with styling it can look like this:

Welcome to Styli!

Welcome to this webpage. Enjoy all the styling tools on this site.

Copyright KC Publishing 2025 | All rights reserved

Adding styling makes a webpage look much better.

A CSS file is formatted in most cases in this way:

```
button {
      width: 100px;
      height: 100px;
      padding: 10px;
      border: 1px black;
      border-radius: 5px;
}
```

This styling will make the button 100px by 100px, have a padding of 10px (padding is the distance between the text and the element border), a black border with a width of 1px and finally the border-radius make it have round corners.

The button now looks like this:

In CSS you first specify the name of element (e.g. h1). You then add a curly bracket (‘{‘) then you add your styles and then finish it off with a closing curly bracket (‘}’).

To access your CSS file, in the HTML file at the top create `<head></head>` tags and in side have a `<link rel=“stylesheet” src=“your-css-file-name.css”>`. It should look like this:

```
<html>
<head>
     <link rel=“stylesheet” src=“your-css-file-name.css”>
</head>
</html>
```

You must replace ‘your-css-file-name.css’ with the name of you css file. You must make sure that it is in the same folder/directory and folder/directory level.

You can also add CSS into your html straight away in two different ways:

1. You can add `<style></style>` tags inside your head tags (`<head></head>`)
   Inside the `<style></style>` tags, you can have your own little mini CSS file.
2. In each individual element, you can add a style attribute and then add CSS styles to that in the same formatting and a normal CSS file. This is a slow way but always works and is sometimes used for debugging.

Here is a list of CSS styles:

CSS/Name	Format	Description
width / height	Width: width;	Specifies the width/height of an element. Also, a HTML Attribute
padding	Padding: length;	Specifies the distance between the text/img/etc and the border of the element
border	Border: width colour;	Creates a visible border round the perimeter of the element
border-radius	Border-radius: radius;	Adds rounded corners to the border of an element
margin	Margin: length;	Creates space between the border of the element and the border of the parent element.

Section 4 – Classes and IDs

When using CSS, you might have more than one button/h1/etc…, but you might want to style them differently or only style one of them. That is when classes and IDs come in.

Classes

Classes are a way of labelling an element or a group of elements so that instead of the CSS styling all of that type of element but just that class.

To make something a class in HTML, you need to add a class attribute to the element:

```
<button class="big-button">
```

This will add that element to the class.

To access the class in CSS, instead of using the general term 'button' we use:

```
.big-button {
      // CSS for class 'big-button' goes here
}
```

You just need a '.' And then the class name (.className). The rest of the formatting is the same as general CSS.

IDs

IDs are just like classes but are more commonly used as identifiers for linking HTML to JavaScript. They are also generally only used to specify only one element.

To create an ID for an HTML element, we use the 'id' attribute:

```
<button id="small-button">Button Text</button>
```

And then to access it in CSS, you must do the same for a class but instead of a dot (.) you use a hashtag (#):

```
#small-button {
      // CSS for class 'small-button' goes here
}
```

Section 5 – Absolute Positioning

In CSS/HTML, you can sometimes feel limited to elements just being placed below the other element and you can't change that. But you can. Using absolute positioning, you can use CSS to tell an element where to go on a page.

To use absolute positioning, in the CSS of an element, use the 'position: absolute;' style and then use the 'top', 'bottom', 'left' or 'right' to position your element:

```
button .topLeft {
      position: absolute;
      top: 5px;
      left: 5px;
}
```

This will make all the buttons in the 'topLeft' class appear 5px away from the top left corner.

Warning!

When using absolute positioning you cannot use aligning, so you have to use something like 'left: 45%; right: 45%;' or something else to center align.

Absolute positioning is useful if you want small icons and menu buttons in top corners, or useful for page footers.

Section 6 – Using iFrames

iFrames are a way of embedding a web page into another. You use `<iframe>` tags and have the embedded page URL as the src attribute.

```
<html>

<iframe width=“100%” height=“100%”
src=“https://example.com/”></iframe>

</html>
```

This will make a fullscreen iFrame for the URL ‘https://example.com/’.

You can also use the `<embed><embed>` tags.

The next section will cover dynamic iFrames.

Section 7 – JavaScript with HTML

Something even more cool with HTML is its compatibility with JavaScript (JS). JavaScript is a scripting language used to make the page dynamic and moving.

First you need to create a JS file (ending in .js). Then to connect it with the HTML file by writing this in the <head></head> tags:

```
<head>
      <script src="your-js-file-name.js"></script>
</head>
```

Remember to replace 'your-js-file-name.js' with the actual file name for your JS file.

You can also add JS to an HTML page just by using the <script></script> tags. A bit like the <style></style> tags but for a mini JS file instead of CSS.

To connect an element for the JS to control, in the JS file you must write:

```
const VARIABLE-NAME = document.getElementById("ID-OF-ELEMENT");
```

You must replace 'VARIABLE-NAME' with the name that you wish to call it in the JS file and replace 'ID-OF-ELEMENT' with the HTML ID of the element that you wish to control.

For example, we can have a <input> element which will make a small area that the user can type in. The ID will be 'userInputBox'. We first must write out the HTML for this <input> element (input does not need a closing tag):

```
<html>
<head>
      <script>
            userInput = document.getElementById("userInputBox").value;
      </script>
</head>
<body>
      <input id="userInputBox">
```

```
</body>
</html>
```

We can then add a button acting as a submit button:

```
<input id="userInputBox">
<button onClick="submit()">Submit</button>
```

Notice that we have used the 'onClick' attribute. This triggers a JS function. In JS all functions end with '()' this is the argument box but that isn't important for this.

We can now change the JS so that when the button is clicked, the webpage will do the alert() function to give a little popup and say what's in the user input box:

```
function submit() {
     var userInput = document.GetElementById("userInputBox").value;
     alert(userInput);
};
```

What this JS code is doing, is it's defining the submit() function (which is run when the button is clicked). The function contains getting the .value of 'userInputBox' which will be what the user has put in the box. Then when the function is run, the alert(userInput) will alert the variable, 'userInput' which was defined when we did the first line in the function.

In JS after every line, you have to write a semi-colon ';' except after the '{' of starting the contents of a function.

Section 8 – Changing Values with JS

Using JS you can configure it to change certain values of text or attributes. For example, you can have a user enter a URL into an <input> text box and then click a button which will change the 'src' attribute for an iFrame element.

Here is the basic HTML layout with some CSS styling:

```
<!DOCTYPE html>
<html>
<head>
	<style>
		#mainIframe {
			width: 100%;
			height: 500px;
		}
		#urlInput {
			width: 300px;
			height: 30px;
			margin: 10px;
			border-radius: 5px;
		}
		#goButton {
			height: 36px;
			width: 50px;
			border-radius: 5px;
		}
	</style>
</head>
<body align="center">
	<iframe id="mainIframe" src=""></iframe>
```

```
      <br>
      <input id="urlInput" type="text" placeholder="Enter URL">
      <button onClick="go()" id="goButton">Go</button>
</body>
</html>
```

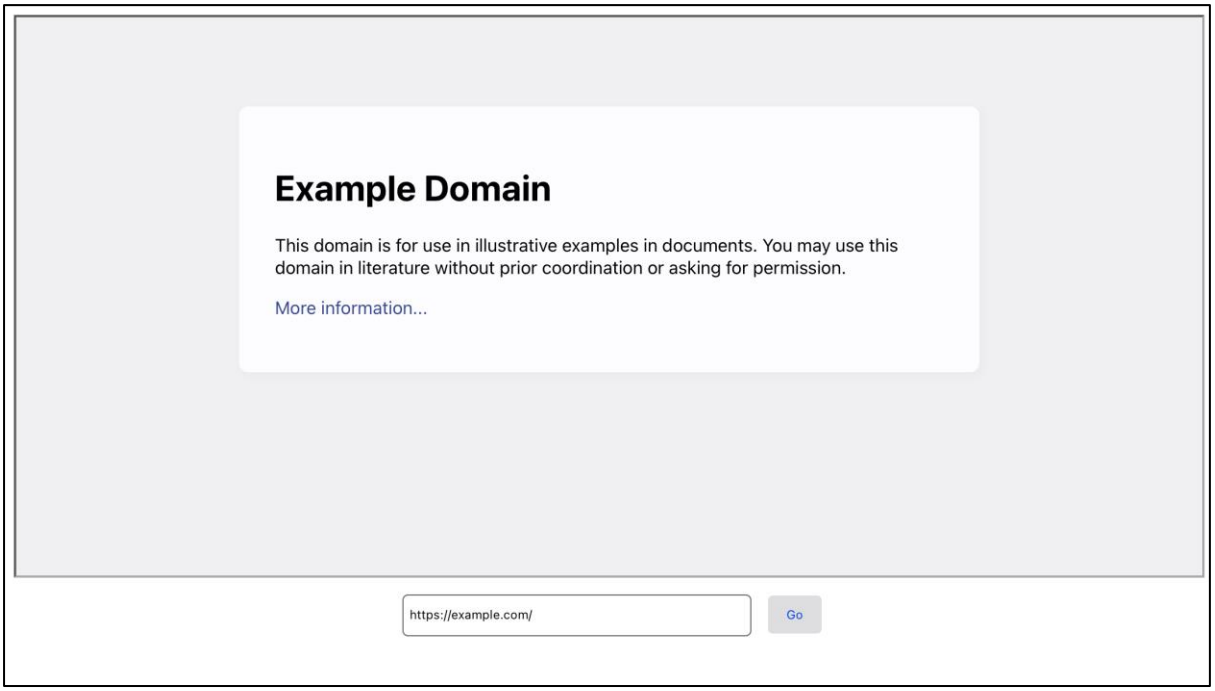
Then we add some JS to complete with functionality:

```
<html>
<head>
      <style>
            #mainIframe {
                  width: 100%;
                  height: 500px;
            }
            #urlInput {
                  width: 300px;
                  height: 30px;
                  margin: 10px;
                  border-radius: 5px;
            }
            #goButton {
                  height: 36px;
                  width: 50px;
                  border-radius: 5px;
            }
      </style>
</head>
<body align="center">
      <iframe id="mainIframe" src=""></iframe>
      <br>
```

```
    <input id="urlInput" type="text" placeholder="Enter URL">
    <button onClick="go()" id="goButton">Go</button>

    <script>
        function go() {
            const urlInput =
document.getElementById("urlInput");
            const iframe =
document.getElementById("mainIframe");
            iframe.src = urlInput.value;
        }
    </script>
</body>
</html>
```

This should look like this:



When the ‘Go’ button is clicked on, it runs the ‘go()’ function, that will get the iFrame and the URL Input and set the ‘src’ attribute of the iFrame to the .value of URL Input.

Section 9 – Dynamic CSS

Even though JS is the main way of making a webpage dynamic, you can also do a bit through CSS.

In CSS you can put a colon ':' then an event, like this:

```
button:hover {
      // CSS for when the pointer is hovering over the button
}
```

The following CSS will create a simple button style for modern buttons:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .buttonStyle {
      height: 30px;
      padding: 5px;
      color: black;
      background-color: lightgray;
      border-radius: 5px;
      border: 1px solid black;
      font-size: 14px;
      cursor: pointer;
    }
    .buttonStyle:hover {
      border-bottom: 2px solid green;
      background-color: #ebebeb;
    }
  </style>
```

```
</head>
<body>
  <button class="buttonStyle">Button Text</button>
</body>
</html>
```

This would make a button look like this:

And when the pointer is hovering over it:

Here is a list of Dynamic CSS events:

CSS	Description
:hover	When the mouse pointer or touchscreen is hovering over an element

:focus	When you are using keyboard shortcuts or clicking on a input field to select it
:active	When the mouse is down on an element and being clicked on
:checked	When a radio or checkbox is checked

Section 10 – More Tags

There are some other useful tags in HTML for designing webpages:

- ** **

  Everything inside this tag is centre aligned

- **<noscript>**

  This is only shown if JavaScript is disabled or not available on the user's browser

- **<b>**

  Defines bold text

- **<code>**

  Defines some computer code

- **<address>**

  Defines a contact card

- **<mark>**

  Defines marked/highlighted text

Section 11 – Keeping HTML Tidy

HTML can seem a bit overwhelming if it is all messy and untidy. To categorise and basically tidy up the whole HTML file we use something called divisions. The HTML tag for divisions is `<div>` there are a few others like

- <section>
- <main>
- <article>
- <blockquote>

You can add IDs and Classes to divisions to access them in JS and CSS. You can also nest them inside each other to make in-section sections.

Divisions are usually placed with a comment just above them to inform what the `<div>` is for. Comments are made using the `<!----->` tag. Just place what you want to say inside the `<!--` and the `-->`:

```
<!-- Navigation Container -->
<div>
     <!-- Navigation Menu-->
</div>
```

Comments are also regually used for other things like in-file labelling bits of code or just to increase the readability of an HTML file.

Section 12 – Text Formatting

There are lots of ways of formatting text in HTML. Here is a selection of basic formatting tags.

Bold, Italic and Underline

- **Bold Text**
  To create bold text, use either, `<b>`, `<strong>` or change the font-weight in CSS. `<b>` is the best option though.
- *Italic Text*
  To create italic text, use the `<i>` tag or the `<em>` tag.
- Underlined Text
  To create underlined text, use the `<u>` tag or change 'text-decoration' in CSS.

Superscript and Subscript

- S^{uperscript}
  To create superscript text, use the `<sup>` tag.
- S_{ubscript}
  To create subscript text, use the `<sub>` tag.

Lists

- Unordered List
  To create an unordered list, put some `<li>` tags inside a `<ul>` tag:

```
<ul>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
</ul>
```

- Ordered List
  To create an ordered list, do the same as an unordered list but swap the `<ul>` tags for `<ol>` tags.

Headings

- Heading 1-6
  To create headings 1-6, use the `<h1>` to `<h6>` tags.

Section 13 – Forms and Inputs

When creating a web page, you are almost certainly going to use some sort of input, whether it's using the `<input>` tag or not.

Inputs can be used in forms as well. In this section we will be creating parts of a form.

Text Input

Text input is simply defined as `<input>` or `<input type=text>` as the default 'type' is 'text'.

Checkbox

A checkbox is defined as `<input type="checkbox">` then you can have a `<label></label>` to add text next to it.

Radio Buttons

Radio buttons are defined as `<input type="radio">` and can also have labels. Radio buttons must be in radio button groups.

Dropdown

Dropdown are placed in a `<select></select>` tags. The options are `<option></option>` tags:

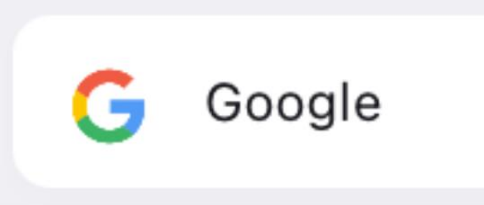
```
<select>
     <option>Item 1</option>
     <option>Item 2</option>
     <option>Item 3</option>
     <option>Item 4</option>
</select>
```

Submit Buttons

All forms need a submit button. Just simply use `<button onClick="submit()">`, then in the JS file, check to see if the inputs ate filled out then do the final action.

Section 14 – Page Title and Favicon

In most webpages, there is a small icon and a page title:



This can be configured in the HTML.

Page Title

To change the page title, use the `<title></title>` tag and the text inside them will be the page title. For example, if a wanted to set the page title to 'My Website' you would put this code in the `<head></head>` tag:

```
<title>My Website</title>
```

Then the page title should be changed when the HTML loads.

Page Favicon

The page favicon can be changed using the `<link>` tag. The attributes for this tag are as follows:

```
<link rel="icon" src="IMAGE-FILE/URL">
```

Remember to replace 'IMAGE-FILE/URL' with the file or url of the desired image.

The image file should usually be scale 1:1 and file type '.png' or '.ico'.

Section 15 – Conclusion to HTML

HTML is made up of tags, like .

HTML can be connected to CSS to add styling and make the page look better.

HTML can be connected to JavaScript to make the page dynamic.

HTML can let CSS can use dynamic events to make things change with user interaction.

HTML can control the page favicon and page titles.

HTML can use user actions and edit attributes with functions.

Index

Tags

HTML Tag	Name	Description
<p>	Paragraph	A paragraph sized amount of body text.
	Span	A small one lined sized amount of text.
<a>	Anchor	Used for creating hyperlinks.
<img>	Image	Displays images.
<button>	Button	A JS/HTML linked button.
	Division	A section of HTML.
<head>	Head	Stores all the page config.
<input>	Input	Allows the user to enter data to the webpage.
<b>	Bold	Bold text formatting.
<i>	Italic	Italic text formatting.
<u>	Underline	Underline text formatting.
<sup>	Superscript	Superscript text formatting.
<sub>	Subscript	Subscript text formatting.
<script>	Script JS	Used to create mini JS file.
<style>	Style CSS	Used to create mini CSS file.
<link>	Link	Used to run another file on top of a HTML file (e.g., CSS, Favicons).

Attributes

HTML Attribute	Name	Description
Href	Hypertext Reference	Used to create hyperlinks.
Src	Source	Used to access other files.
Width/height	Width / Height	Specifies the dimensions of an element.
Style	Style CSS	Contains CSS.
OnClick	OnClick JS	Runs the specified JS function when element is clicked on.
Id	Id	Used to locate an element in both JS and CSS. Usually for one element.
Class	Class	Used to locate an element in both JS and CSS. Usually for a group of elements.

Type	Element Type	Used to specify the type of an element.
Align	Align	Aligns the text and elements to the given alignment.

CSS Styles

CSS Style	Name	Description
Color	Text Colour	Specifies the colour of text.
Background-color	Background Colour	Specifies the colour of background.
Padding	Padding	Specifies the distance between.
Border	Border	Adds a border around the perimeter of the element.
Border-radius	Border Radius	Specifies how rounded the corners are.
Position	Position	Changes the type of position relativity.
Top/bottom/right/left	Top/bottom/right/left positioning	Specifies the distances from the side (top/bottom/left/right).