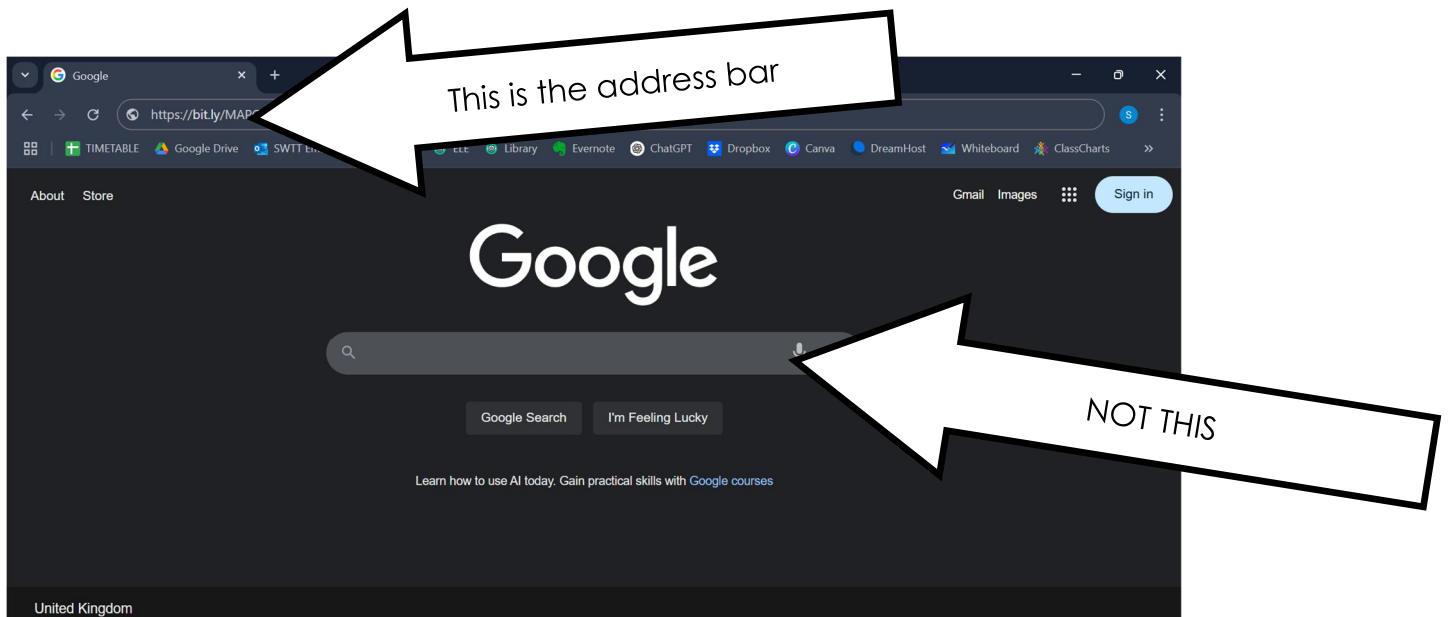


#1 HTML

Setting Up

We are going to use an online tool to create our web pages. Open a web browser and type into the **address bar** (not the Google Search box) the following address:

`https://www.programiz.com/html/online-compiler/`



Session 1: Hello Web!

Web sites are coded in a format called HTML. Text is marked with **tags** which indicate how the browser should interpret the page. There is a tag at the start and a tag at the end. Without any extra fancy additions, the output is pretty basic, as it was 25 years ago, but by the end of this coding club we will have moved on towards making your project look like a modern website.

The basic contents of a web page are this:

```
<html>
<head>
<title>Page title goes here</title>
</head>
<body>
  The content of the page goes here
</body>
</html>
```

So the body of the content is 'wrapped' in the tags `<body>` and `</body>`.

Tags can be placed inside each other. See how `<title>` is inside `<head>`.

Over the page are some of the key tags we will use this week to create our first web page!

Key tags

<code><h1></code>	Heading 1	Like in Word, the big title
<code><h2></code>	Heading 2	A secondary title
<code><p></code>	Paragraph (ie normal text)	Paragraph
<code></code>	Bold text	Bold text
<code><i></code>	Italic text	<i>Italic text</i>
<code><u></code>	Underlined text	<u>Underlined Text</u>

Notice that you can combine them, so

```
<strong><u>Bold and underlined</u></strong>
```

becomes **Bold and underlined**

Lists

One other really useful thing to learn today is **lists** or bullet points. You define them by first declaring the `<ul>` tag (the listen is ended as you have guessed by now with the `</ul>` tag) and each entry is marked with `<li>This is the entry</li>` which displays as

- This is the entry

Today's Challenge

Given what this document has shown, why not create a "this is me" page as your first web page.

Use the editor to create a page in your name.

- Give it a web page title of `<your name>`
- Use `<h1>` to give your name as it is the most important thing.
- The use `<h2>` and `<p>` to say something about yourself.
- Why not tell us about your interests, favourite foods, hobbies or favourite bands with an `<h2>` title and an list.

Mr Rundell gives out cool stickers to people who achieve good things. Will your first ever web page be the one which wins you a sticker of your choice because of what you have achieved?

Let's see what you can do!

Next: "Style it up" - using Cascading Style Sheets (CSS)

#2 Style Sheets

Session 2: Style it up

By using Cascading Style Sheets (CSS) you can format the layout of a webpage.

You can do a load of really clever things with CSS like this



You can hold your CSS in a separate file and link all your pages to it or you can build it *inline* with your existing html.

We are going to do it this second way at the moment as it is easier to manage it in this simplified way of working.

You put your CSS between the `<style></style>` tags, and it looks like this:

```
<style>

body {
  background-color: black;
  font-family: Century Gothic;
}

h1 {
  font-size: 28px;
  color: red;
}

p {
  color: white;
}

</style>
```

Each entry has

- curly brackets around it.
- the element we are changing (there are hundreds)
- a value we are setting
- a semicolon ; at the end of each value

Let's practice this in a way that you can see it happen immediately.

Go to Programiz and choose "HTML" and "Online Compiler"

You can even copy over the stuff you made in the last session if you saved it into this online edit, make changes to the CSS and see how you can really style the page to make it really individual.

Some common CSS Styles

Element	Property	Example Values	Description
Text	color	red, #3498db, rgb(0, 255, 0)	Sets the text color
Text	font-size	16px, 1.5em, 120%	Controls text size
Text	font-family	Arial, Georgia, monospace	Sets the font style
Background	background-color	lightblue, #f2f2f2	Changes background color
Borders	border	1px solid black, 2px dashed red	Adds a border around an element
Box/Layout	margin	10px, auto, 5px 10px	Space outside the element
Box/Layout	padding	10px, 0 20px	Space inside the element's border
Box/Layout	width / height	200px, 50%, auto	Sets the size of the element
Display	display	block, inline, none	How the element is displayed
Text alignment	text-align	left, center, right	Aligns text horizontally
Positioning	position	static, relative, absolute, fixed	Sets how the element is positioned
Visual Effects	opacity	1, 0.5, 0	Controls transparency
Visual Effects	box-shadow	2px 2px 5px grey	Adds shadow around the box
Links	text-decoration	none, underline, overline, line-through	Styles underlines and other link visuals

This is only a tiny selection of what you can do. Notice some values can be text, some are numbers or a mixture of both. Colours can be specified as the text ('red') or as a hexadecimal value #ff0000 or RGB value (rgb(255,0,0)).

Two things to remember:

- if the browser doesn't understand a command it will just ignore it. This might have unintended consequences. The spellings are also US spellings (**color**, for example)
- Each setting must end with a semicolon or it and the following line will both be ignored. Don't forget it!

Styling up your biography from the last session

Now you have some CSS skills, why not add them into the html you created last time. You can make your "About me" page look really fun, funky, colourful or classy! You can work on it using the try-it editor and then copy it back to the one on the MAP Coding Club web server.

Next: DIVs, Images and positioning.

#3 DIVs and Positioning

Images

The tag to bring an image into a web page is

```

```

The image may be in a location on the server, or may be the full URL (address) of an image elsewhere. The height and width values can be specified as a pixel (px) value or a percentage (%) of the available remaining space, enclosed in quotes like `width="100%"` or `width="100px"`

<DIV> Containers and an introduction to classes

So far we have just used the basic tags to arrange things. There is a block which is much more powerful called `<DIV>`. This enables a section of HTML to be put in its own container. It can be styled and positioned by our CSS. But how does the browser know that I want a certain style to apply to a certain `<DIV>`? The answer is I give it a name or as we call it a **class**. So, if I want a group of paragraphs to share the same font or colour I can create the html which says:

```
<html>
<head>
<style>
.title {
    font-family: "Arial";
    font-size: 18px;
    color: red;
}
.mytext {
    font-family: "Trebuchet MS";
    font-size: 14px;
    color: blue;
}
</style>
</head>

<body>
    <div class="title">
        <h1>My Heading</h1>
    </div>
    <div class="mytext">
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
    </div>
</body>
</html>
```

Notice that the name of the class is preceded by a full stop. Like `.title` This tells the browser it is a custom class.

`font-family` sets the font or group of fonts for text. The size is set by `font-size` and might be a specific size or a relative size, like bigger or smaller. Each style is ended with a semicolon ;

Any item can be assigned a class so you could style a single paragraph with:

```
<p class="mytext">Text here</p>
```

Positioning with Flexbox

So far, we've learned how to group things using `<div>` and change the way they look using CSS classes. But how can we control where things go on the page?

That's where something clever called `Flexbox` comes in.

Flexbox is short for **Flexible Box Layout**. It helps us lay out items **horizontally** or **vertically** inside a container, like lining up boxes next to each other in a neat row or column.

To use it, we create a container (a `<div>`) and tell it to behave like a flexbox. Then we can place items inside it, and they will arrange themselves like magic!

```
<html> <head>
<style>
.flex-container {
  display: flex;
  background-color: lightgrey;
  padding: 10px;
}

.flex-item {
  background-color: white;
  border: 1px solid black;
  padding: 20px;
  margin: 10px;
}
</style>
</head>

<body>
  <div class="flex-container">
    <div class="flex-item">One</div>
    <div class="flex-item">Two</div>
    <div class="flex-item">Three</div>
  </div>
</body> </html>
```

Next time: Some Basic JavaScript!

The result is this:



`display: flex;` turns the container into a **flexbox**.

Items inside line up **in a row**, with space and padding.

It's much easier than using floats or margins everywhere!

Other options include:

`justify-content:` aligns items horizontally (left, center, space-between, etc.)

`align-items:` aligns items vertically (top, center, bottom)

`flex-direction:` row or column

Try putting some of these commands in `flex-container` or `flex-item` and see how you can change things around really easily!

If your layout looks weird, check:

Did you set `display: flex;` on the container?

Did you close all your `<div>` tags properly?

Are your class names spelled exactly right?

#4 JavaScript

Putting some JavaScript in the Web Page

JavaScript is the language that makes the Web sing. And dance. It is embedded into the html documents you have created, or linked to them. At this stage we will embed them inside the HTML to make it easier to understand.

JavaScript is embedded inside the `<script></script>` tags. At the end of each command is a semicolon so the browser knows that this is what it should work on next.

One of the most basic commands is to create an alert which pops up as soon as the page is loaded. The command for this is `alert()`

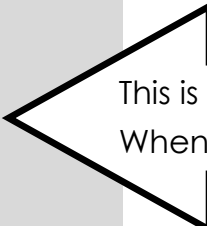
```
<html>
<script>
    alert('Yes!');
</script>
<body>
<h1>Do you see the message?</h1>
</body>
</html>
```



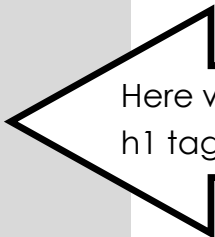
The brackets at the end show that this is a **function** and we can give information to this function to display on screen

But this isn't all! We can get the alert to happen when we click on something. This is called an **event**. Events are triggered by the user doing something.

```
<html>
<script>
    function h1_onClick() {
        alert("Now you do!");
    };
</script>
<body>
<h1 onclick="h1_onClick()">
Do you see the message?</h1>
</body>
</html>
```



This is the function called `h1_onClick()`. When it is triggered it will show the alert.



Here we have added a command to the `h1` tag to call the function `h1_onClick()`

Now the alert will only be shown when you click on the `<h1>` tag.

Getting Information from the Web Page

If I create an **inputbox** on the page and give it an `id`, I can take that information from the user and do something with it.

An input box is set up in html with

```
<input type="text" id="yourName" prompt="Enter your name">
```

We will also add a button to create an event which does something.

```
<html>
<script>
    function sayHello() {
        var yourName=document.getElementById('yourName').value
        alert('Hello, ' + yourName + '!');
    };
</script>
<body>
Enter your name:<input type='text' id='yourName'>
<button id='submitButton' onclick='sayHello()'>Click to say hello
</button>
</html>
```

There are three things to notice here:

- Names of **objects** as we call them should not have spaces in them, so we often use `camelCase` where we start in lower case and then capitalise other words to show what the variable is about. It is called `camelCase` because it has humps in it like a camel!
- We created a **variable**: one of the most important things in coding. Think of it as a box which can hold a value like a number or a *string* of text. I defined it when I used the word **var**. Notice the name of the variable is in `camelCase` as well.
- I have called a special function built into JavaScript to get the value of the contents of our inputbox. It is called

```
document.getElementById('name of object').value.
```

- Using this I can not only read a value, but I can set it as well. Change the `alert()` line to

```
document.getElementById('submitButton').innerText = 'Hello, ' +
yourName;
```

and run it again... See how we can change the text of the button!