Creative Coding Cheat Sheet

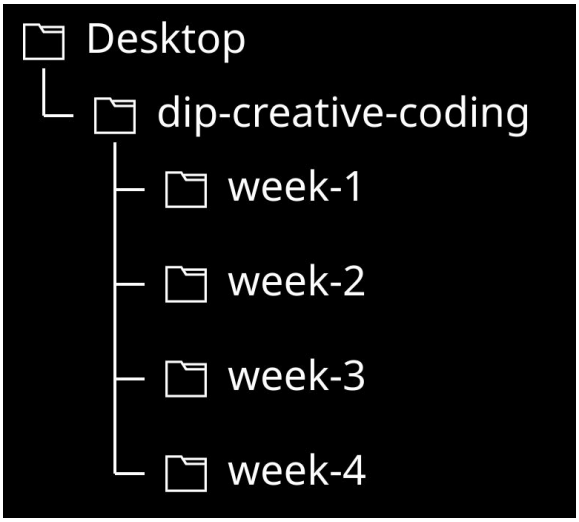
A quick overview of creative coding with HTML, CSS, JavaScript, and p5.js

Key technical resources

Coding examples for this unit on GitHub: https://github.com/IrtizaNasar/CCI-Diploma22-CreativeCoding
Getting started with Visual Studio Code: https://code.visualstudio.com/docs/introvideos/basics
Learn about Chrome developer tools: https://developer.chrome.com/docs/devtools
HTML Reference: www.w3schools.com/html
JavaScript reference: www.w3schools.com/js
CSS reference: www.w3schools.com/js
P5.js reference: https://p5js.org/reference
Critically informed approach to coding p5.js: https://aesthetic-programming.net/

Setting up your coding environment

We strongly suggest organising your code in the following way. Create a folder on your desktop called **dip-creative-coding.** And within that folder create directories named **week-1**, **week-2** etc. This will help you keep track of all the files and examples that you create each week. This structure will help us quickly locate where files are stored so we can easily help you.



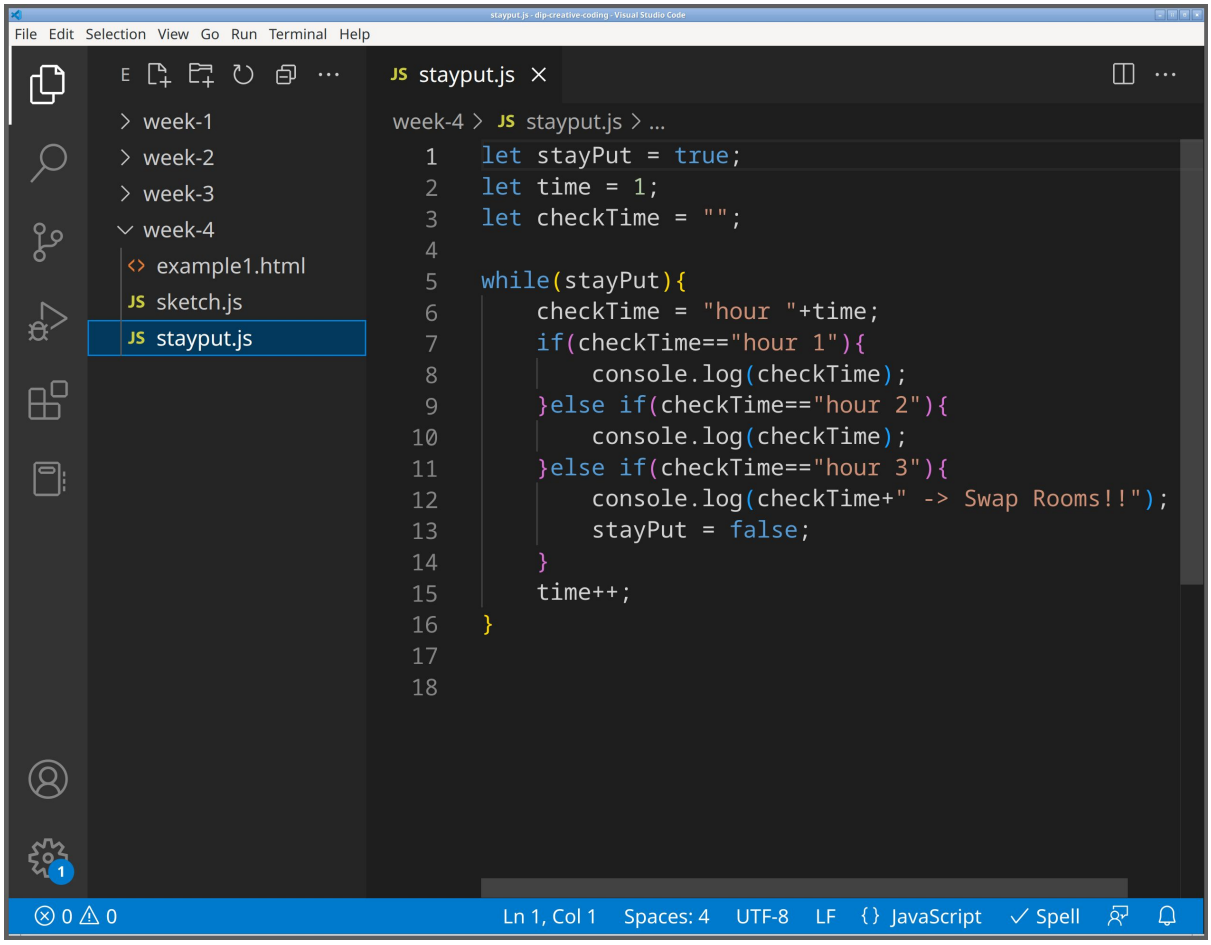
Default browser: Chrome

We recommend that you set the default browser on your computer to Chrome for a number of reasons. First, Chrome is the most widely used browser on the planet so if your code works in chrome then it is very likely to work on a majority of users machines. Second, if everyone uses chrome then it is much easier for us to help you, because we will all be working with the same menu system.

Using Visual Studio Code (VS Code)

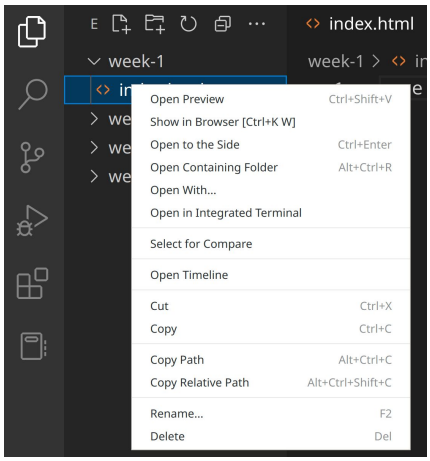
VS Code is a source code editor that we use for editing programming code. There is **a lot** of functionality in VScode that takes getting used to. So if you are feeling lost, we recommend watching this short video: https://code.visualstudio.com/docs/introvideos/basics. The image below, in columns from left to right, shows the key sections of VS code:

- Icons - including a ‘doc’ icon at the top for showing and hiding the file Explorer.
- The file Explorer - hiding the explorer creates more space to write code.
- The code edit area - each line of code is identified by a number which can you with debugging.



Viewing your HTML files in a browser

There are a number of ways to view HTML files (such as index.html or example.html) in a browser. You can either:



Open the file within Chrome
Just navigate to the file by using **cmd+o** to navigate to, and then open the file.

Open a browser from within VS Code:
MacOS: control+click the file. **Windows/Linux:** Right Click the file. Then select **Show in Browser** from the little menu. **Note:** There seems to be a bug where this does not work in all systems.

Copy the file path from within VS Code

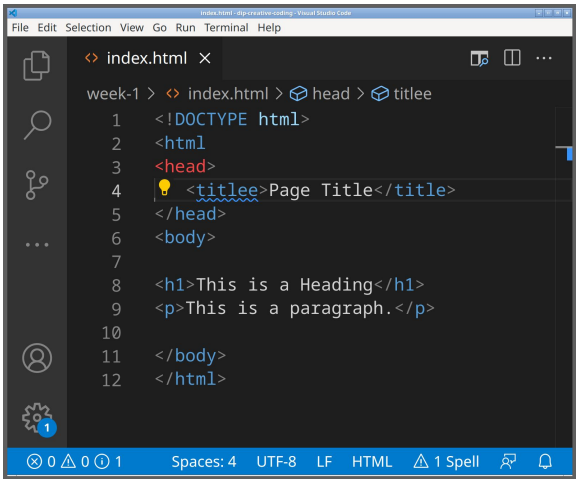
MacOS: Control+Click the file. **Windows/Linux:** Right Click the file. Then select **Copy Path** from the little menu and paste the file path into Chromes address bar.

Drag a file

From the VS Code Explorer (or your desktop) directly into Chrome's address bar.

Debugging

Debugging errors in HTML and JavaScript can be tricky so follow these tips:



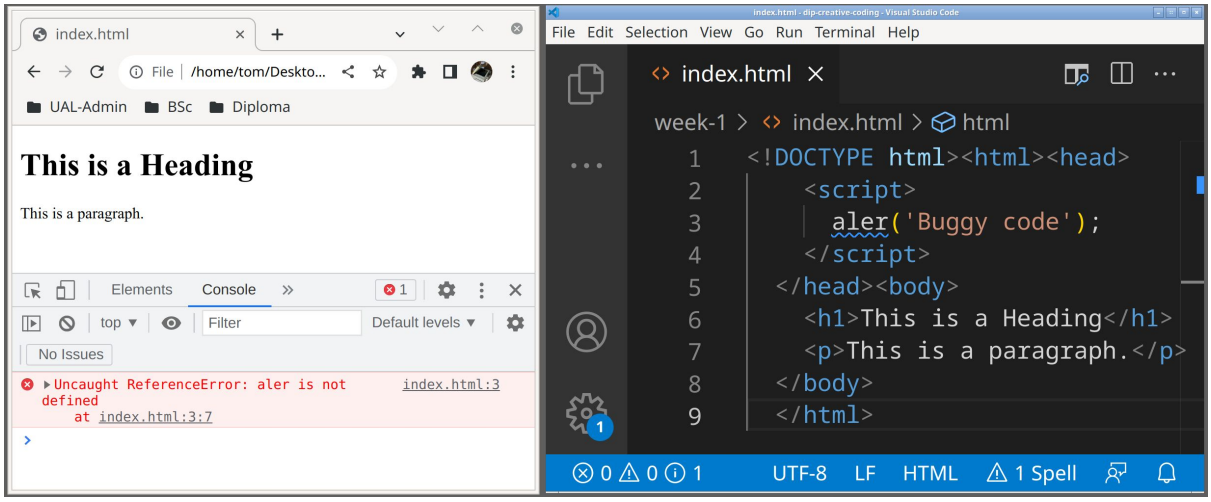
A. Learn to 'read' VS Code

For example, the image to the right shows **<head** highlighted in red which indicates that the preceding **<html** tag is missing a **>** character i.e. it should be **<html>**. Also, the yellow light bulb icon, and wavy blue line, under the **<titlee>** tag, indicates a misspelling i.e. the tag should read **<title>**.

B. Learn to read the browser console

The browser console is a space to test JavaScript or debug errors in your code. The image below indicates an error in line 3 of index.html. Indeed, the VS COde editor also indicates a misspelling (i.e **aler**) of the **alert** command. The console can be accessed via the Chrome menu **More tools -> Developer tools**, or the key combination:

- Mac OS: Control+Shift+J
- Windows/Linux: Command + Option + J

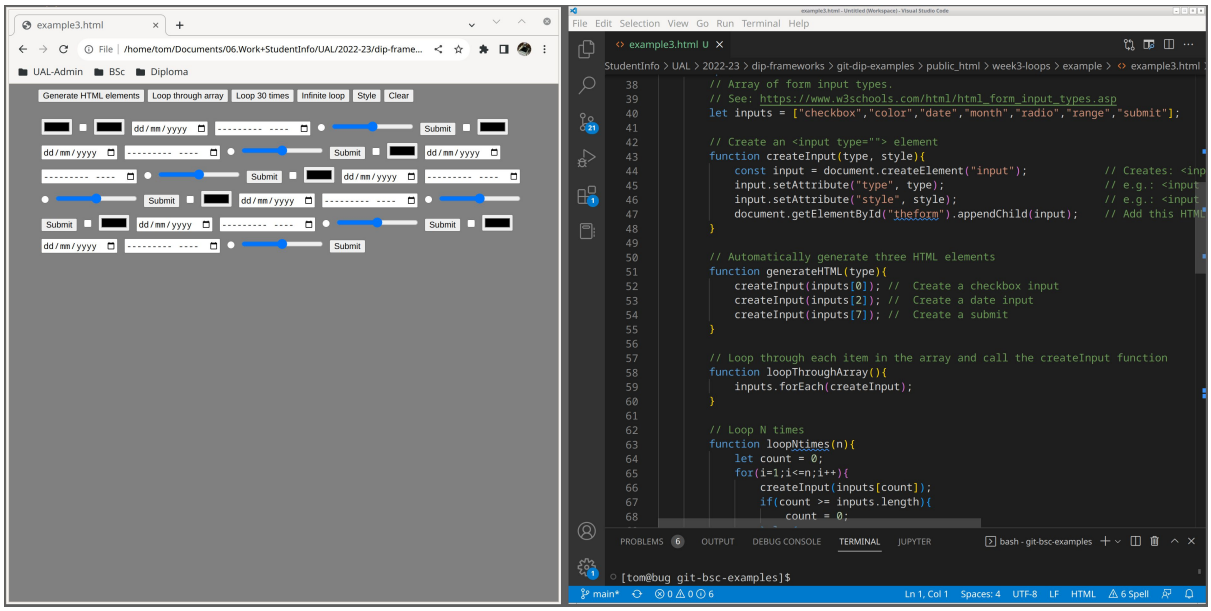


C. The built-in debugger within VS Code

If you are setting out in coding then we **DO NOT** recommend using VS Codes internal debugging tool for now. Though do read about this tool in their documentation if you want to know more.

Make life easy: Arrange VS code and your browser

Arrange your coding environment so it is easy to edit code and view it in the browser. We suggest arranging VS code studio on one side of your screen and your browser on the other - as per the image below. A less cluttered screen makes it much easier to debug!



HTML + JavaScript + CSS

HTML

HTML describes the structure of a webpage. An HTML document contains ‘tags’ that tell a browser how to display content such as <html> <body> and <p>.

www.w3schools.com/html is a great reference that explains different kinds of HTML tag. For example, the code to the right is an entire HTML page which you could save into a file (i.e example.html) and view in a browser.

```
<!DOCTYPE html>
<html>
<head>
   <title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

It is important to remember that most HTML tags open (i.e. <h1>) and close (i.e </h2>) so a browser can choose how to display their content of text (or other tags).

HTML Comments

HTML comments allow you to annotate or write comments about your code without them displaying in the browser - unless someone **views the source** of your code.

Note: In this example, I saved space by putting the <html>, <head>, <title>, and <body> tags on the same line.

While browsers don’t demand that tags are written on seperate lines, we suggest you write your HTML more clearly than this!! Generally, tags should be indented and placed on separate lines so your code is easy to read.

```
<!DOCTYPE html><html><head>
<title>Page Title</title>
</head><body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

<!--This is a comment.
Comments are not displayed
in the browser-->

</body></html>
```

Cascading Style Sheets (CSS)

CSS is the language used to style an HTML document. CSS describes how HTML tags (elements) should be displayed. Do follow the tutorials at www.w3schools.com/css. The example to the right:

```
<!DOCTYPE html>
<html>
 <head>
   <title>CSS Example</title>
   <style>
     h1, p{
       border:10px solid rgb(33,99,102);
     }
     .other{
       border:1px solid rgb(33,99,102);
```

1. Defines some CSS, in the HTML header, which draws a thick 10 pixel line around **all** <h1> tags and <p> tags.
2. Shows how inline CSS within the **style** attribute, of the second <p> tag, overrides the CSS in the header.
3. Defines a 1px line around all tags with the class attribute ‘other’.

```
     }
    </style>
 </head>
 <body>

   <h1>A styled heading</h1>
   <h1>Another styled heading</h1>
   <p>I have a border</p>
   <p style="border:1px dotted #f59b42;">
        I have a dotted border.
   </p>
   <p class="other">Defined by class</p>

 </body>
</html>
```

JavaScript

JavaScript is a programming language that can be ‘embedded’ or ‘linked’ within an HTML page. Crucially, JavaScript can access, rewrite, or add-to existing HTML on a web page. This ‘reworking’ of HTML by JavaScript is often referred to as dynamic HTML.

Embedded JavaScript

The example to the right shows how JavaScript can be embed in an HTML page. The JavaScript code placed within a <script> tag in the <head> of the HTML document to the right will run before the body tag is rendered or loaded by the browser. The second <script> tag will load after the body content has loaded.

```
<!DOCTYPE html>
<html>
<head>
  <title>Embeded JavaScript</title>
  <script>
    alert('Hello 1 - runs before body');
  </script>
</head>
<body>

  <h1>This is a Heading</h1>
  <p>This is a paragraph.</p>

  <script>
    alert('Hello 2 - runs after body');
  </script>

</body>
</html>
```

Linking external JavaScript + CSS files

You can link to separate JavaScript and CSS files within HTML rather than embedding

```
<!DOCTYPE html>
<html>
```

code. This example loads separate JavaScript (.js) and CSS (.css) files into the HTML file - their contents may look like this, where the grey text are code comments:

```
// I'm JavaScript
alert("I'm a script");
```

```
/*I'm CSS code*/
p{
   color:red;
}
```

```
<head>

 <title>JavaScripyt and HTML</title>
 <script src="script1.js"></script>
 <link rel="stylesheet" href="styles1.css" />
 <link rel="stylesheet" href="styles2.css" />

</head>
<body>

 <h1>This is a Heading</h1>
 <p>This is a paragraph.</p>

 <script src="script1.js"></script>

</body>
</html>
```

Note: The contents of external .js and .css files should not include any HTML tags such as <script> or <style>.

Core programming concepts

Variables

Variables can be thought of a containers to store data.

```
let a = "This is";        // A string
let b = `${a} a string`;  // String with 'a' inserted
let c = 1;                // Integer (a whole number)
let d = 1.4;              // Floating point
let e = true;             // Boolean (true or false)
```

Arrays

Declare an array in JavaScript and access its elements at a specific index.

```
let words = [];           // Declare the array
words[0] = "what";        // Define a word at '0'
words[1] = "are";         // Define a word at '1'
words[2] = "arrays";      // etc.
console.log(words[1]);    // Output index 1: are
console.log(words.length);// Output length of the array
```

For loops

A for loop enables

you to execute a block of code many times.

```
for(let i=0;i<101;i++){
    console.log("Number="+i);
}
```

Conditionals

Conditionals allow you to set a different path in code by specifying a condition.

```
let hungry = "Cake";
if (hungry === "Cake") {
   console.log("eat some food");
} else {
   console.log("go for a run");
}
```

Functions

A function allows you to run (or call) a snippet of code in multiple places. This saves you from writing the same code multiple times. In this example, we define a function called **printMyName()**

```
function printMyName(name) {      // Declare a function
  console.log("My name is: "+name);
} // End of the function

printMyName("Tom");
printMyName("Nicola");

// Will print to the console:
//    My name is Tom
//    My name is Sally
```

JavaScript Frameworks

A JavaScript Frameworks is basically pre-written code which allows you to get going quickly on a project. Examples include p5.js, paper.js, angular.js, react.js, and JQuery.

Create your own!

You can build a collection of useful JavaScript functions and save them in a separate file. If the JavaScript to the right was saved into a file called **MyFramework.js**

```
// This code is in MyFramework.js

// Generate HTML tag within tag identified by an id
function createTagAtID(id, tag, style, content){
    const thetag = document.createElement(tag);
    thetag.setAttribute("style", style);
    thetag.innerHTML = content;
    document.getElementById(id).appendChild(thetag);
}
```

and linked within an HTML file then those functions would be available to that script.

```
// Generate a random number between a & b
function random(a, b){
    return Math.floor(Math.random()*b)+a;
}
```

In the HTML example to the right, the code automatically generates a paragraph tag with random color text.

The ‘**createTagAtID**’ and random function can be written as many times as you want.

```
<!DOCTYPE html>
<html>
<head>
  <title>My JavaScript Framework</title>
  <script src="MyFramework.js"></script>
</head>
<body>

  <div id="myBox"></div>

  <script>
    let id = "myBox";
    let tag = "p";
    let r = random(0, 255);
    let g = random(10, 255);
    let b = random(0, 255);
    let style = `color: rgb(${r},${g},${b});`;
    let content = `a paragraph in ${id}`;

    // Create a p tag
    createTagAtID(id, tag, style, content);
    createTagAtID(id, tag, style, content);
  </script>

</body>
</html>
```

TODO: p5.js

Content coming soon: will explain what a canvas is and why.

Basic p5.js structure

Basic hello world example…

Difference between setup and draw

Introduce clear function()