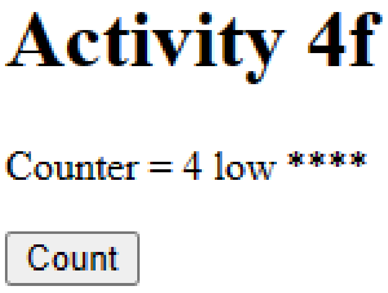
Teddywaddy Code Club

Activity 4f

Coding Basics



Activity 4f

Counter = 4 low ****

Coding Basics

This set of activities is a small introduction to some common coding statements, not specifically JavaScript.

All coding languages have statements like variables, loops and decisions.

Most languages have very similar statements, so learning to code is applicable to any language.

Open VS Code and create a folder or create the folder first and then open that folder with VS Code.

To learn more about JavaScript and many other web technologies, go to

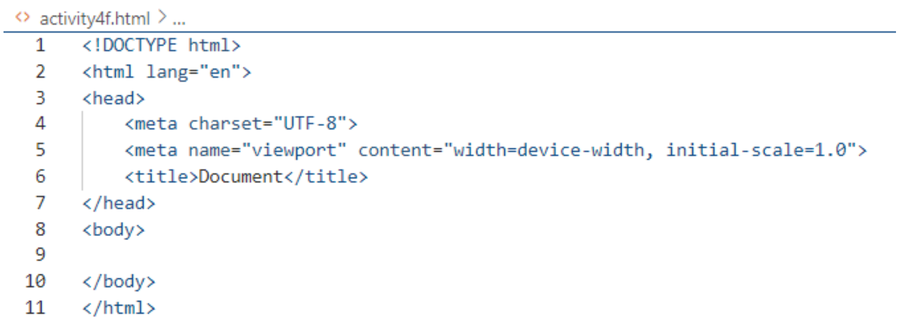
https://www.w3schools.com/

Create the HTML File

In VS Code, create a new file called activity4f.html.

Use the VS Code ! abbreviation to insert template html code.

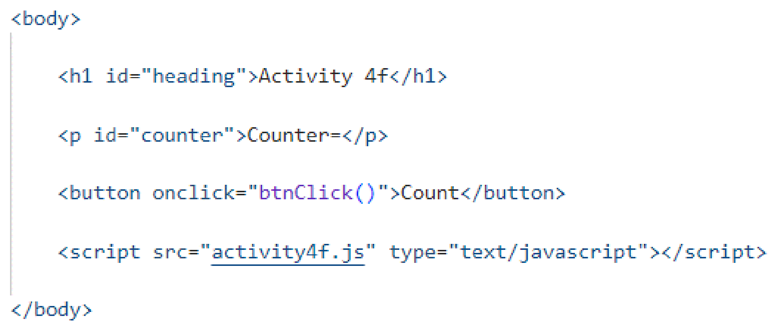
Type ! and then hit tab or enter.



activity4f.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

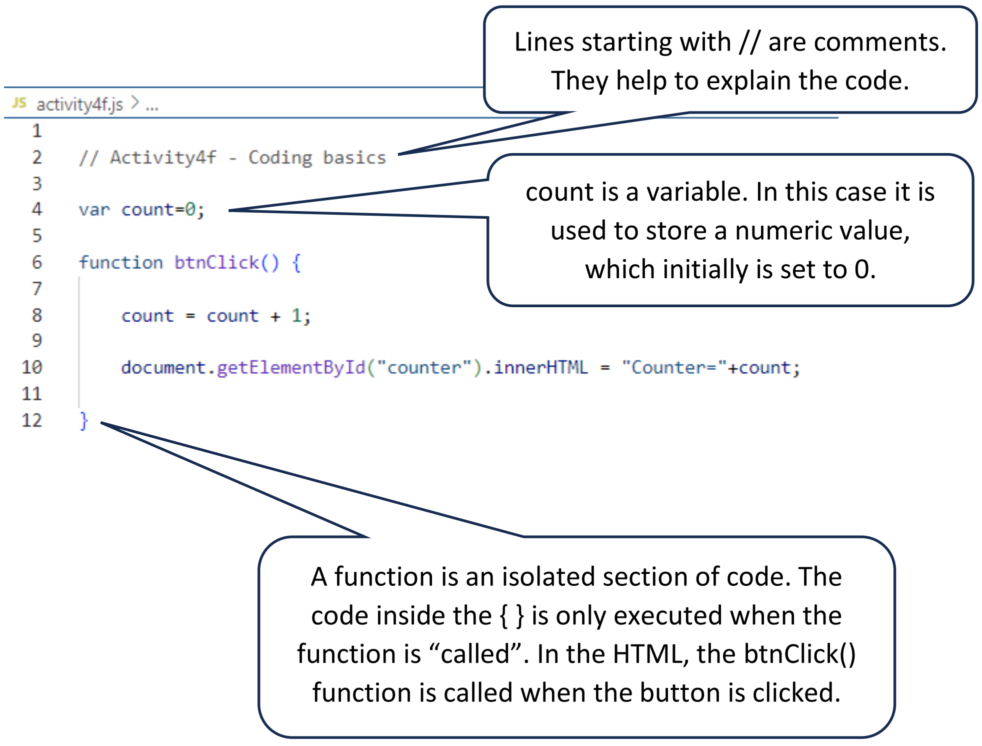
Add some HTML code to the body, including the <script> tag to link to the JavaScript file, which will be created next.



```
<body>

    <h1 id="heading">Activity 4f</h1>

    <p id="counter">Counter=</p>

    <button onclick="btnClick()">Count</button>

    <script src="activity4f.js" type="text/javascript"></script>

</body>
```

Create the JavaScript File

In VS Code, create a new file called activity4f.js and enter the following code.



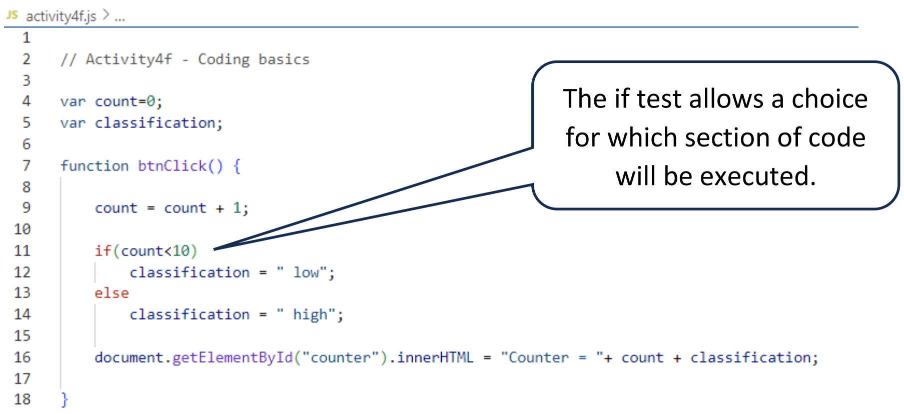
```

// Activity4f - Coding basics

var count=0;

function btnClick() {

    count = count + 1;

    document.getElementById("counter").innerHTML = "Counter="+count;

}
```

Lines starting with // are comments. They help to explain the code.

count is a variable. In this case it is used to store a numeric value, which initially is set to 0.

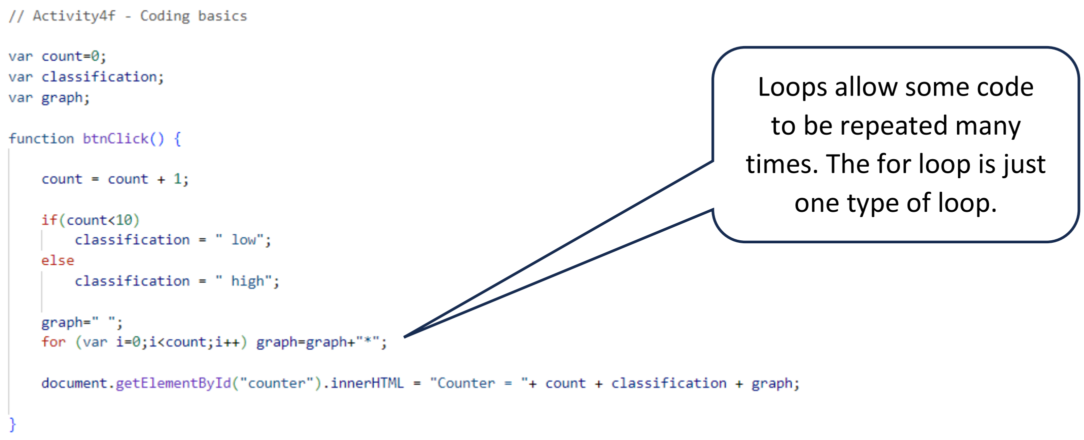
A function is an isolated section of code. The code inside the { } is only executed when the function is "called". In the HTML, the btnClick() function is called when the button is clicked.

Now add some additional code that will classify the count into low or high.


activity4f.js > ...
The if test allows a choice for which section of code will be executed.

```

// Activity4f - Coding basics

var count=0;
var classification;

function btnClick() {

    count = count + 1;

    if(count<10)
        classification = " low";
    else
        classification = " high";

    document.getElementById("counter").innerHTML = "Counter = "+ count + classification;

}
```

Mode code to give a graphical representation to count.


Loops allow some code to be repeated many times. The for loop is just one type of loop.

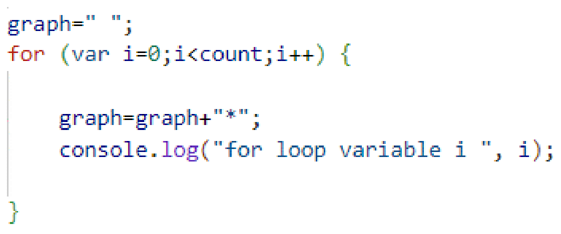
```
// Activity4f - Coding basics

var count=0;
var classification;
var graph;

function btnClick() {

    count = count + 1;

    if(count<10)
        classification = " low";
    else
        classification = " high";

    graph=" ";
    for (var i=0;i<count;i++) graph=graph+"*";

    document.getElementById("counter").innerHTML = "Counter = "+ count + classification + graph;

}
```

A for loop repeats the execution of some code a certain number of times. In this case the number of times is equal to the value of variable count.

i is the variable used to do the counting of the repetitions and i++ means add 1 to i.

Most coding statements can contain more than one line of code by using the { } braces, just like the function above.

Change the for loop as follows and check the console as well as the web page.



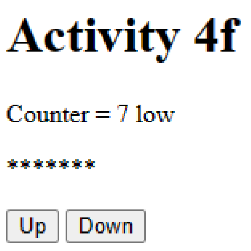
```
graph=" ";
for (var i=0;i<count;i++) {

    graph=graph+"*";
    console.log("for loop variable i ", i);

}
```

Using functions

There are many, many coding statements to learn about, however equally important are overall code design and structure. So rather than learning more statements, this section will now look at how the previous code can be refined.

Firstly, think about what is required. The new task is that we should have a button for counting down as well as up. This will mean more JavaScript coding is required, but it also raises some questions about the HTML and CSS layout too. Should the buttons be beside each other or above and below? Where should the asterisk bar be placed?

Target Layout



Activity 4f

Counter = 7 low

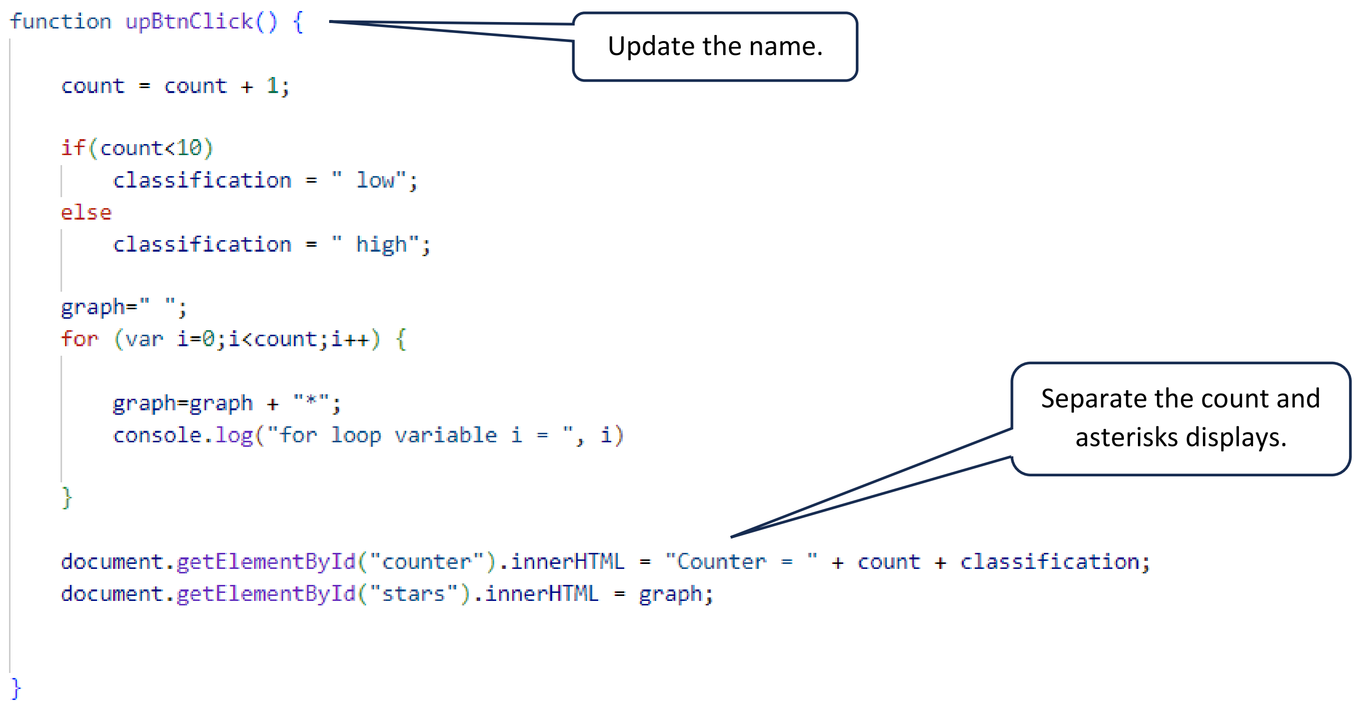
****

Update the HTML code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Activity 4f</title>
</head>
<body>

    <h1 id="heading">Activity 4f</h1>

    <p id="counter">Counter =</p>
    <p id="stars">-</p>

    <button onclick="upBtnClick()">Up</button>
    <button onclick="downBtnClick()">Down</button>

    <script src="activity4f.js" type="text/javascript"></script>

</body>
</html>
```

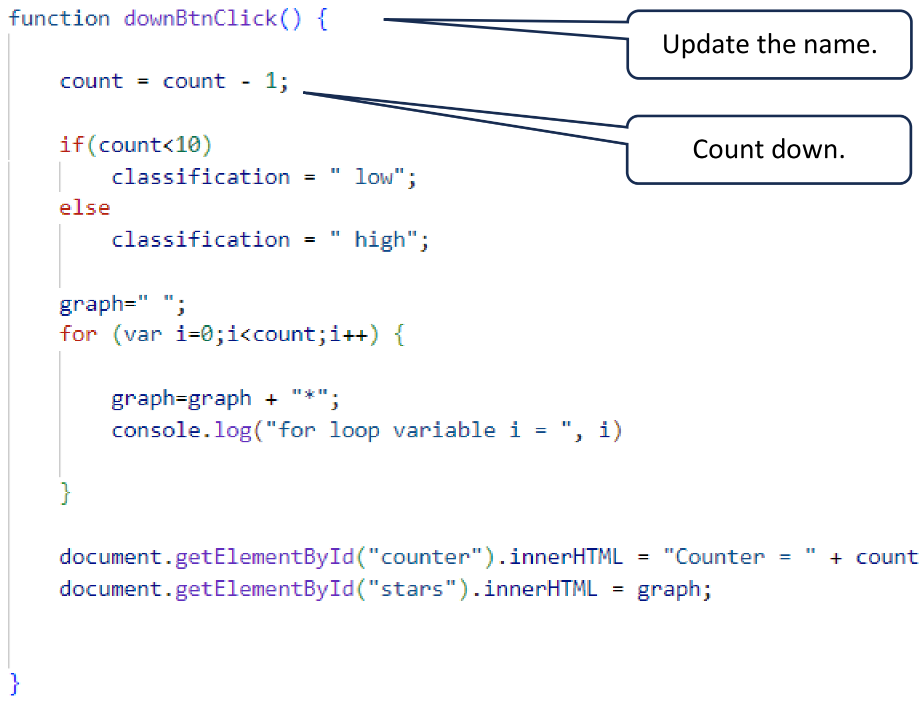
Update the JavaScript code

Firstly, get the function updated to work properly for the up button clicks.



```
function upBtnClick() {

    count = count + 1;

    if(count<10)
        classification = " low";
    else
        classification = " high";

    graph=" ";
    for (var i=0;i<count;i++) {

        graph=graph + "*";
        console.log("for loop variable i = ", i)

    }

    document.getElementById("counter").innerHTML = "Counter = " + count + classification;
    document.getElementById("stars").innerHTML = graph;


}
```

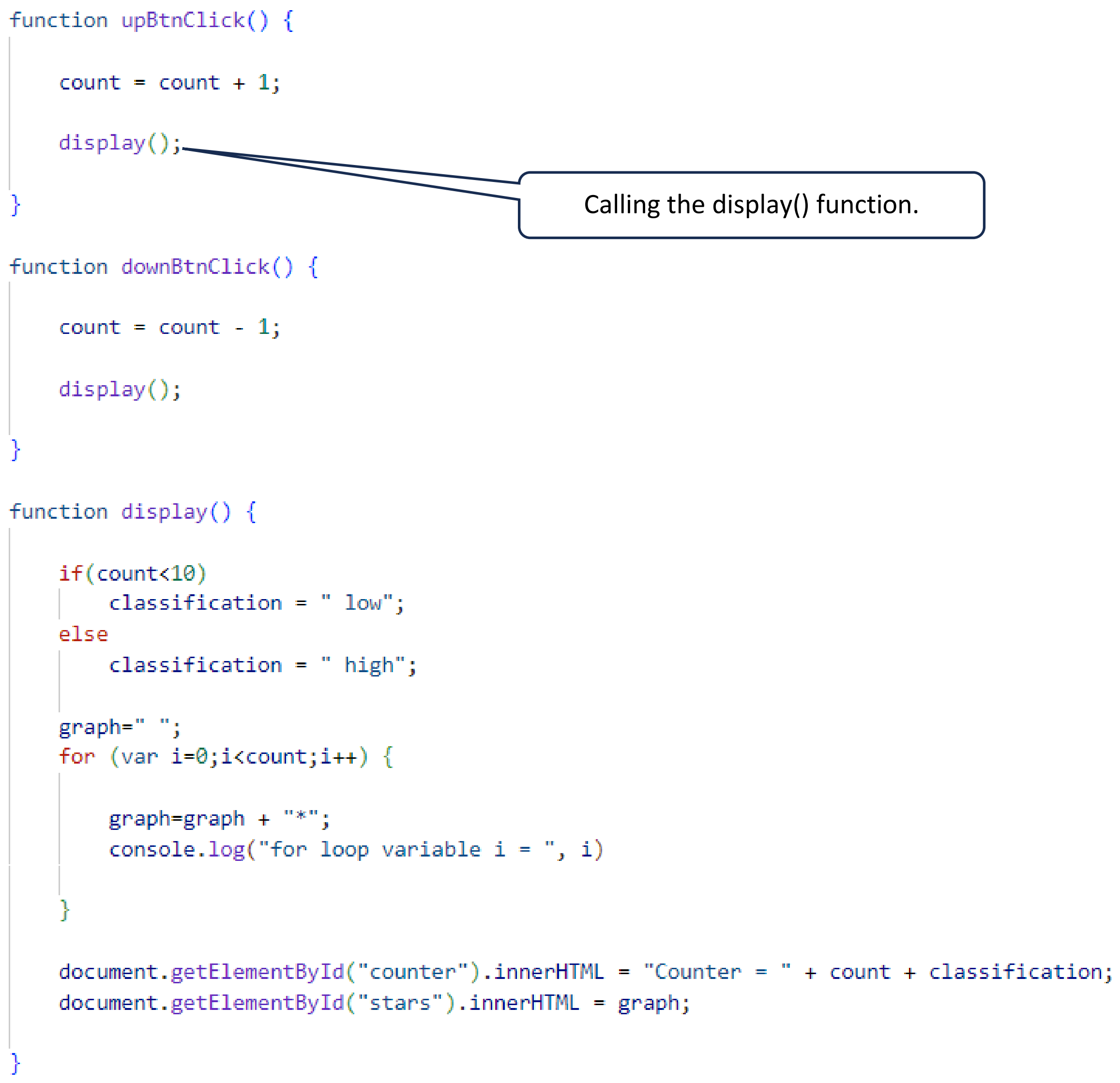
Now another function needs to be created for the down button clicks. Make a copy of the down function and then there are only a couple of very small changes required.



```
function downBtnClick() {

    count = count - 1;

    if(count<10)
        classification = " low";
    else
        classification = " high";

    graph=" ";
    for (var i=0;i<count;i++) {

        graph=graph + "*";
        console.log("for loop variable i = ", i)

    }

    document.getElementById("counter").innerHTML = "Counter = " + count + classification;
    document.getElementById("stars").innerHTML = graph;


}
```

Now, because there is so much code that is duplicated, even more refinement can be done.

All the common code can be placed in a function that is used twice, rather than having the code twice.



```
function upBtnClick() {

    count = count + 1;

    display();

}

function downBtnClick() {

    count = count - 1;

    display();

}

function display() {

    if(count<10)
        classification = " low";
    else
        classification = " high";

    graph=" ";
    for (var i=0;i<count;i++) {

        graph=graph + "*";
        console.log("for loop variable i = ", i)

    }

    document.getElementById("counter").innerHTML = "Counter = " + count + classification;
    document.getElementById("stars").innerHTML = graph;

}
```

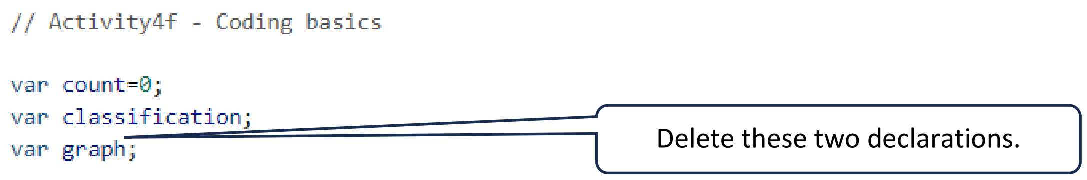
Much simpler and neater.

The use of functions is extremely useful in breaking down a large project into more focused code blocks – often matching the structure of the design.

One aspect of using functions is the concept of scope. Good coding should limit the scope of variables across functions. Scope means where variables are known. In the code above, the variables are all what are called global variables. Ideally, there should be a minimum of global variables.

In some cases, it is easy to limit the scope of a variables, like the *graph* and *classification* variables in the code above.

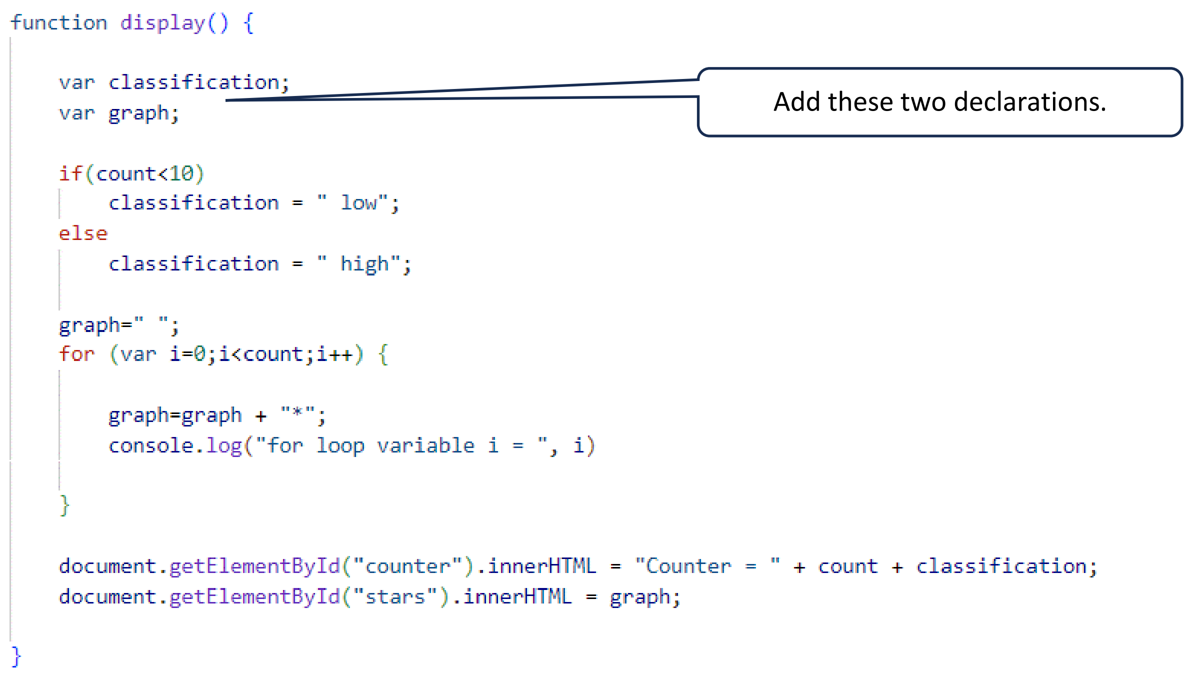
Delete the global declaration of the *graph* and *classification* variables.



```
// Activity4f - Coding basics

var count=0;
var classification;
var graph;
```

Delete these two declarations.

Add the graph and classification variables to the display() function.



```
function display() {

    var classification;
    var graph;

    if(count<10)
        classification = " low";
    else
        classification = " high";

    graph=" ";
    for (var i=0;i<count;i++) {

        graph=graph + "*";
        console.log("for loop variable i = ", i)

    }

    document.getElementById("counter").innerHTML = "Counter = " + count + classification;
    document.getElementById("stars").innerHTML = graph;

}
```

Add these two declarations.

These variables are now only known within the display() function. Try a console.log(graph) or console.log(classification) at various places throughout the code to see the effect of scope.

It would be desirable to limit the scope of the count variable as well, but it is used across all three functions and the value needs to be retained, so it is ok to leave it. It could be moved to the functions, but this would require more advanced function usage.