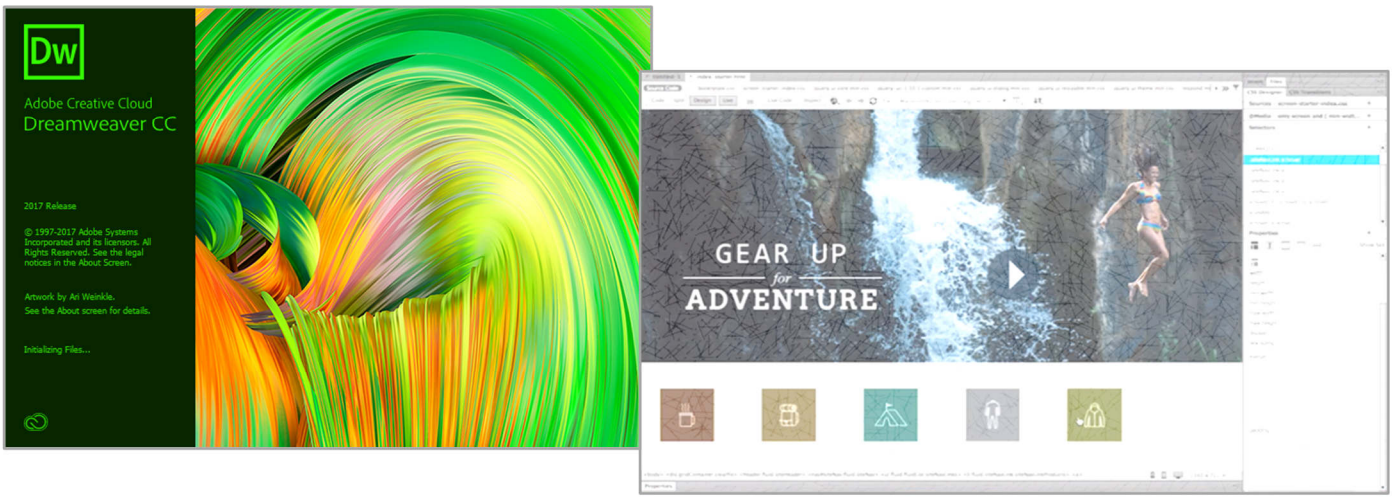
Dreamweaver CC 2017

JavaScript

JavaScript is one of the programming languages that make things happen in a web page. It is a fantastic way for students to get to grips with some of the basics of programming, whilst opening the door to advanced possibilities.

These tasks have been created in Dreamweaver but can easily be adapted for use in any other web designer or text editor.



In the last task, we used an *alert* to display a word in a box. This worked, but alert boxes are really not very popular (you don't see them very often on the web these days and when you do, they are usually associated with something trashy).

We will now look at some ways of changing the text displayed so that we can avoid alerts.

Task 1 – document.write

a. Create a new blank page and save as 'sectionC.html'.

b. Type the code shown on the right between the <body> and </body> tags, then view the page in *Live view* or your browser. The JavaScript should execute, simply writing the text to the page.

```
<body>
<script>
    document.write("A new way")
</script>
</body>
```

c. There's not a lot of point in doing this. It's much more interesting if we put this action into a button.

Try the code on the right and see what happens when you click the button in *Live view*.

```
<body>
<h1>document.write</h1>
<button onclick="document.write('A new way')">Click me</button>
</body>
```

Things to notice

1. The HTML tags <button> and </button> are used to create the button.
2. The 'onclick' *event* is used to make something happen when you click the button. (Easy this, isn't it?)
3. We've used a set of single quotation marks around the text we want to display. This is to differentiate them from the double quotation marks around the whole JavaScript statement. You could swap these around, but not mix them up (try it, if you like). Different styles of quotation marks should not be interlaced (e.g. "..'.. ".. ').
4. The 'document.write' *method* clears the screen before it writes the new text. This is a little limiting.

Something to try: What happens if you remove the word 'document' from the 'document.write' method? We are telling the code to write to the document. In this case, the browser doesn't have much choice so it understands the instruction. In other situations, it might be confused if this is not made clear. Similarly, you may see the code 'window.alert' being used rather than just 'alert'.

Task 2 – innerHTML

Try the code below. Make sure that you are accurate and don't mix up the quotation marks. You can place this code after the code above, still within the body tags. Remember to *refresh* each time you want to reset the page.

```
<h1>innerHTML</h1>
<p id="question">What are we doing?</p>
<button onclick="getElementById('question').innerHTML = 'Changing text'">Click me too</button>
```

See if you can work out what all this code is doing before looking at the explanation on the next page.

What does all this code do?

`<h1>innerHTML</h1>`	*Create a heading using HTML*
`<p id="question">What are we doing?</p>`	*Create a paragraph and give it the ID 'question'*
`<button`	*Open a button tag*
`onclick="`	*Set up an onclick event which activates when the button is clicked*
`getElementById('question')`	*When clicked, find the element with the ID 'question'…*
`.innerHTML =`	*… and replace the text in this element…*
`'Changing text'`	*… with this new text*
`">`	*Close the onclick event and the first button tag*
`Click me too`	*Place this text on the button*
`</button>`	*Close the button*

Task 3 – Practicing our Outputs

Try the tasks below. You can place all the solutions in your 'sectionC' page, or if you prefer, create new pages such as 'sectionC3a' etc. You should copy and paste code frequently, partly to save time, but also to minimise errors.

What if the code doesn't work?

Unfortunately, there aren't any simple tools in Dreamweaver to find the errors in JavaScript. There is a process called *linting*, but for the JavaScript we are using it's easier to just check through each line carefully if the program isn't working. Here are some tips.

- Make sure that each quotation mark is part of a pair and that these are not confusing the browser.
- Check that the IDs referred to are identical to the ones you have created. IDs are case sensitive, so the ID 'question' is different to the ID 'Question'.
- If you are producing several solutions on one page, all your IDs must be different.
- And most of all… DON'T PANIC. Even great programmers spend half their time working out what went wrong.

Note: *Many programmers use the developer tools in Chrome to debug their JavaScript. If you're feeling brave you could look at these, but they are also difficult to use for a beginner.*

a. Create a button that, when clicked, opens up an alert box.

Button text = "Click for an Alert"

Alert text = "The button was clicked"

The code on the right should get you most of the way.

Strings are pieces of text. Strings are usually enclosed in double quotes. The following are all examples of strings:

"This is a string" ***"And another"*** ***"2"***

Strings are used in association with variables e.g.

	Variable name	***Value***
var string_1 = "This is a string"	*string_1*	*This is a string*
var string_2 = "And another"	*string_2*	*And another*
var string_3 = "2"	*string_3*	*2*

Task 1 – Displaying our String

Create a new page and save as 'sectionE1.html'. Add the code on the right to your page and view the result.

Note: *You should start by copying and pasting the code from the previous activity.*

```
<body>
<p id="Answer"></p>
<script>
var string_1 = "This is a string"
document.getElementById("Answer").innerHTML = string_1;
</script>
```

Task 2 – Concatenation

We can use the '+' sign to join two strings together. This is called 'concatenation'. Concatenation is different to adding numeric values together.

a. Adjust the script so that it now looks like that shown on the right.

Write down the text displayed **exactly** as it appears in your browser.

```
var string_1 = "This is a string"
var string_2 = "And another"
var string_3 = "2"
var joinString = string_2 + string_3
document.getElementById("Answer").innerHTML = joinString;
```

_____

b. Change the line so that it now reads: var joinString = string_2 + " " + string_3;

How have we changed the presentation of the text? _____

Task 3 – Do you get it?

Analyse the script shown and write down the text that you think it will display.

_____

After making your prediction, type in the code and see if you were correct. Save the file as 'sectionE3.html'.

```
<script>
var valueX = "2";
var valueY = "4";
var total = valueX + valueY;
document.getElementById("Answer").innerHTML = total;
</script>
```

Functions are pieces of code that perform a particular task. The function will be executed when an event takes place. The event may be triggered by the user loading the page, clicking a button or placing the mouse over an image etc.

In our first case, the JavaScript function will be placed in the head section of the HTML. It is then ignored by the browser until required. We will later go on to create an external script file which will keep our JavaScript separate from the HTML.

Task 1 – Calling a Function

Save a new page as *sectionH.html* then enter and test the code below. A JavaScript function called 'addSeven' has been added between the </title> and </head> tags (remember that JavaScript names are case sensitive). All the *onclick* event now has to do is call the function using the code 'addSeven()'.

Open the head section

Title tags

Start of JavaScript
Create a function called 'addSeven'
Open the squiggly brackets
Find the text entered in the input box
Add 7 to it
Place answer in our output paragraph
Close the squiggly brackets
Stop using JavaScript

Close the head section

Open the body section
Ask for a number
Add an input box
Add a button which calls the function
Create a paragraph for the output
Close the body section

```
<head>
<meta charset="utf-8">
<title>Section H</title>

<script>
function addSeven()
{
 var startNo = document.getElementById('inputBox').value;
 var endNo = Number(startNo) + 7;
 document.getElementById('outputText').innerHTML = endNo;
}
</script>

</head>

<body>
<p>Please enter a number and I'll add 7 to it.</p>
<input id="inputBox">
<button type="button" onclick="addSeven()">Enter</button>
<p id="outputText"></p>
</body>
```

Note: *Number() is a built-in JavaScript global function which converts strings and other objects into numbers.*

Task 2 - Questions (Using a printout? Save paper – type your answers into a document.)

a. What does this code do? _____

_____

b. What is the name of the input box? _____

c. What code is used to invoke (call) the function? _____

K. String Properties and Methods

Although JavaScript can be used for all sorts of things in a webpage, we are leaning towards the validation of inputs because this enables us to test out lots of ideas without building complicated pages.

Forms are used to collect information from the user e.g. names, addresses and emails. These forms are found all over the internet, but they have been a source of a huge number of problems because devious people can use them to gain access to online databases. It is therefore necessary to place strict controls over the data that is allowed through. This is called data validation.

As far as the validation of form data is concerned, you may want to refuse information for any of the following reasons:

- There are too many or too few characters (e.g. for usernames and passwords);
- A number that is either too high or too low has been entered;
- The inclusion of unwanted characters (e.g. you may not want brackets in a telephone number field);
- An email address may clearly not be real.
- There may be code included that is designed to infiltrate the website and database.

The string properties and methods can be used to help build all these validation rules. We introduced the '**toLowerCase()**' method at the end of the last set of activities. There is a similar '**toUpperCase()**' method. We'll now look at some others.

Task 1 – length Property

The *length* property can be used to find the number of characters in a string. You can then use this to set a minimum and maximum length for the data. Place the code below into the head section of a page named '*sectionK1.html*' and test it.

```
<script>
var String1 = prompt("Please enter a word","");
var LengthString1 = String1.length;
alert("Your word was " + LengthString1 + " letters long");
</script>
```

Note: *We can place this script in the header even though it means that the code will run immediately. This is fine as we are not using any HTML further down the page. Prompts and alerts are pure JavaScript.*

Task 2 – charAt Method

The *charAt* method tells us the character that occupies a certain position in a string. Place the code below into a page named '*sectionK2.html*' and test it.

```
<script>
var String1 = prompt("Please enter a word of at least 2 letters","");
var Letter2 = String1.charAt(1);
alert("The second letter is " + Letter2);
</script>
```

The number (or index) of the character you want to select is in the brackets. The index starts at 0, so (0) would identify the first letter, (1) the second, (2) the third etc. What happens if you only enter a 1 letter word?

The *for loop* has a slightly different structure to a *while loop*. Although there are lots of variations, we are usually aiming to run the loop a certain number of times. The structure of the loop is like this:

```
for ( statement 1;  statement 2;  statement 3 ) {
carry out the actions in these squiggly brackets
}
```

Here is an example of a *for* loop:

```
for (  loopNo = 1 ;  loopNo <= 10  ;  loopNo ++  ) {
Calculate loopNo * 7
}
```

This example will show the 7 times table.

Statement 1
This is the situation the first time the loop runs through. In our case we are setting the variable 'loopNo' to 1 so that the first calculation will be 1x7.

Statement 2
This is the condition for running the loop. We will continue looping while the variable 'loopNo' is less than or equal to 10.

Statement 3
This statement is executed at the end of each loop. In our case we will add 1 to the variable 'loopNo' each time round. We will therefore calculate 2x7, 3x7, 4x7 etc.

Note: *You can actually choose a 'for' loop or a 'while' loop for most tasks. 'for' loops are generally selected when you want a fixed number of iterations (cycles), although this can easily be achieved with a 'while' loop.*

Task 1

a. Create a new page saved as '*sectionN1a.html*' and add the function below to your header (remember the script tags). Add an input box to your HTML where the user enters the times table they would like to calculate. Also, add a button to invoke the function and an output paragraph for the display. The HTML code
 adds a line break between each result.

```
function timesTable() {

var multiple = document.getElementById("MultipleBox").value;
var loopNo = 0;
var outputString = ""

 for ( loopNo=1 ; loopNo<=10 ; loopNo++ ) {

 outputString = outputString + loopNo * multiple + "<br>";
 }

 document.getElementById("outputText").innerHTML = outputString;
}
```