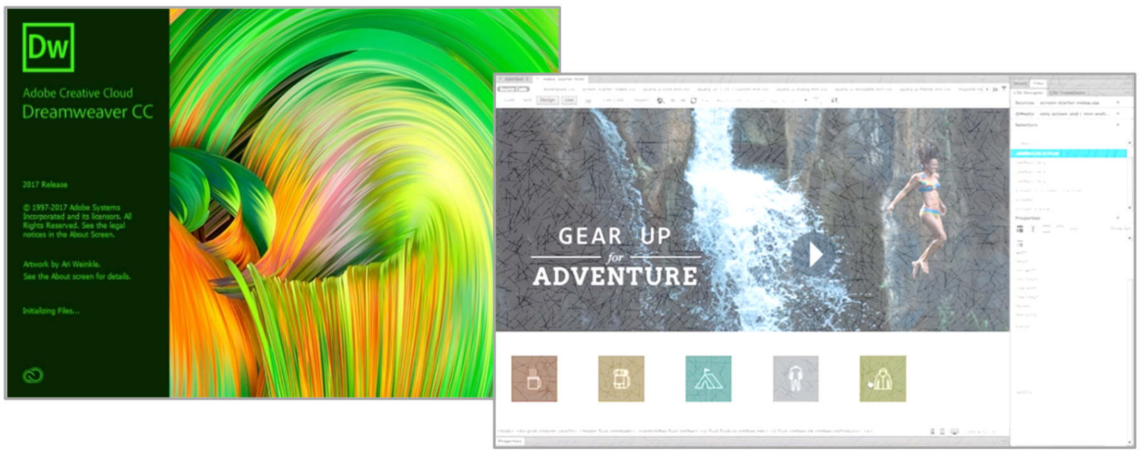
Dreamweaver CC 2017

The Basics

These basic resources aim to keep things simple and avoid HTML and CSS completely, whilst helping familiarise students with what can be a daunting interface. The final websites will not demonstrate best practice at this stage, but will help users become more comfortable with the application and learn how to use many of the visual tools.

This version is for Dreamweaver CC 2017. There are also versions available for the earlier Dreamweaver CC 2015 and CS6.



Hyperlinks are the things that tie the web together. Clicking on a hyperlink usually takes you to a new place on the internet; sometimes in the same website, sometimes in a completely different location.

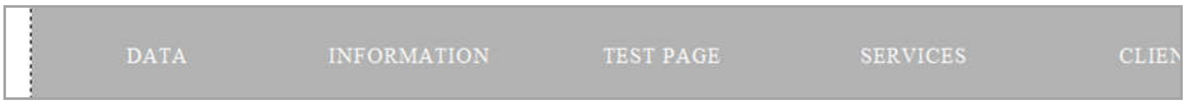
We will start by creating hyperlinks from your homepage to each of the other pages in your website. These are called *Internal hyperlinks* (or *relative hyperlinks*) because they will be linking to a page in the same website rather than a location out in the web.

Creating Your Hyperlinks

a. If it is not already open, bring up your homepage by double-clicking on 'index.html' in the list of files on the right.

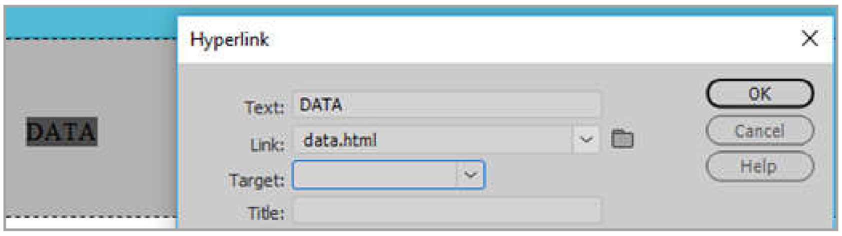
b. We are going to start by editing three of the links in the navigation menu along the top of the page. These will link to each of our three other pages. It's worth leaving the other links in place until later, just in case you decide to use them.

The first link, which currently shows the word 'ABOUT', will become the link to our Data page. Select the 'ABOUT' text and replace it with the word "DATA". Similarly, change the second link to "INFORMATION" and the third to "TEST PAGE". Leave the last three links as they are for now. You can delete them at the end if you don't need them.



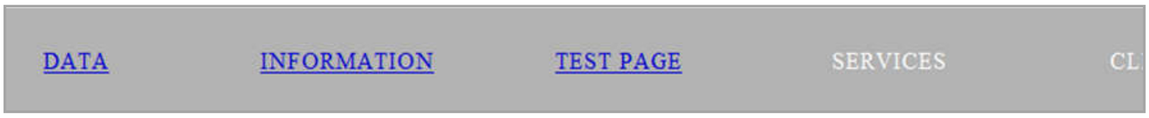
c. Now the text is in place, we need to make it do something. When someone clicks on the word 'DATA', we want the web browser to move to the Data page. We need to make this text a hyperlink. Select the text and click 'Insert / Hyperlink'.

d. In the window that opens, click on the folder icon alongside the *Link* box and select the file 'data.html' (the link will point to this page). Click 'OK'. You should notice that the text becomes blue and underlined. We can style this later.



Text: DATA
Link: data.html
Target:
Title:

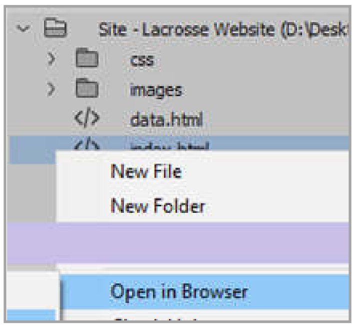
e. Repeat this process, making two more links that point to your other pages. If you make a mistake, you may undo the change. Alternatively, right-click on a link and select 'Change Link' or 'Remove Link'. Save your work.



Testing Your Hyperlinks

We will now check that the hyperlinks work by testing them in a browser.

a. Right-click on *index.html* in the files panel and select 'Open in Browser' (it's near the bottom of the list). Choose one of the browsers on your system (Internet Explorer, Safari, Chrome etc.).



b. Using the browser, click on the 'Data' link. You should be taken to your *Data* page.

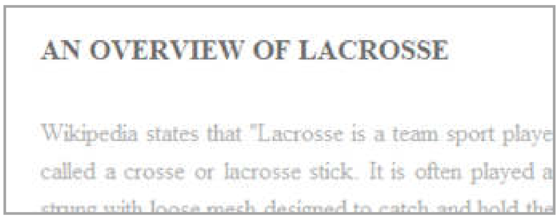
c. Use the *Back* button in your browser to return to your homepage and check the other links. Return to Dreamweaver when you have finished checking.

Once you have had a look at the *Page Properties* and set any default styles that you like, it's time to start looking at the text in more detail. This involves both the content and the style of the text.

As always, we are just trying out ideas and learning new skills at this point. Don't worry too much about the overall appearance. Once you understand a little more about how Dreamweaver and webpages work, you can go back and change anything that you think will improve your design.

Editing Text

Open your homepage and start editing the text. We have used the title "Lacrosse" at the top of the page and the heading "An Overview of Lacrosse" in our information section. You should make up titles relevant to your subject.



AN OVERVIEW OF LACROSSE

Wikipedia states that "Lacrosse is a team sport playe
called a crosse or lacrosse stick. It is often played a
strung with loose mesh designed to catch and hold the

Copying and Pasting Text

We have borrowed some text from the Wikipedia page *http://en.wikipedia.org/wiki/Lacrosse*. You can make up your own text or copy some from a legal source. You can perform a normal copy and paste, but this may produce unappealing results if formatting styles are also copied. Should this be the case, try a *Paste Special* as below:

a. Select the text you want to copy and (if using Windows) right-click and select 'Copy' (other platforms use similar methods).

b. Return to Dreamweaver and place your cursor in the area that you want the text to appear. Click 'Edit / Paste Special'.

c. Generally, you are better off selecting 'Text only' otherwise things can get messy. However, you may have a reason to use a different method (you can always 'Undo' and try again). Click 'OK'.

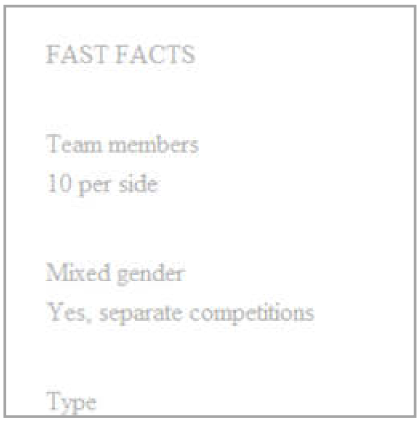
d. Delete any unwanted text.

Paragraphs and Line Breaks

Type the heading "FAST FACTS" and press the 'Enter' key. You should find that a new paragraph is created.

Decide on some fast facts. When you want to start a line immediately below the previous one, hold down the *Shift* key and press 'Enter'. This is called a *Line Break*.

Note: *It doesn't matter where you type your fast facts. You can move them later.*



FAST FACTS

Team members
10 per side

Mixed gender
Yes, separate competitions

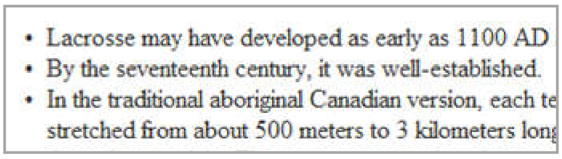
Type

Lists

a. Think of an idea for a list. Do you think the list would be better with bullet points or numbers?

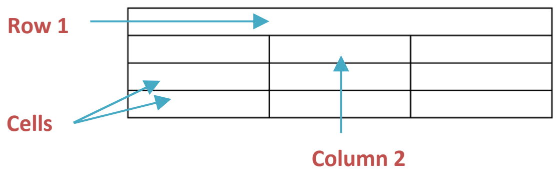
b. Click on 'Insert / Ordered List' to set up a numbered list, or use 'Unordered List' if you prefer bullet points.

c. Use the 'Enter' key each time you want to create a new list item. Press it twice to end the list.



- Lacrosse may have developed as early as 1100 AD
- By the seventeenth century, it was well-established.
- In the traditional aboriginal Canadian version, each te
  stretched from about 500 meters to 3 kilometers long

Tables are basically a collection of boxes, called *Cells*, that you can enter text or other content into. Tables have *Rows* and *Columns*. For example, the table on the right has 4 rows and 3 columns. Although there were probably 12 cells to begin with, the 3 cells in the top row have been *merged*, meaning that there are now only 10 cells altogether.



Creating and Editing a Table

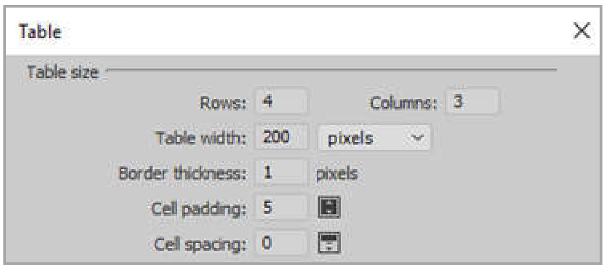
a. Open your data page (data.html) and delete the temporary text we placed in it.

b. We will start by creating the simple data table above. Click 'Insert / Table' to open the table window.

c. Enter the details as shown, with 4 rows and 3 columns.

The *Cell Padding* is the distance between the edge of the cell and anything you place in the cell. Enter '5', for 5 pixels.

The *Cell Spacing* is the distance between the different cells. Enter '0'.



Table

Table size

Rows: 4 Columns: 3

Table width: 200 pixels

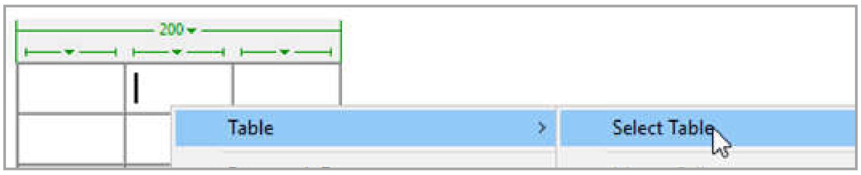
Border thickness: 1 pixels

Cell padding: 5

Cell spacing: 0

d. Click 'OK'. The table will appear on your page.

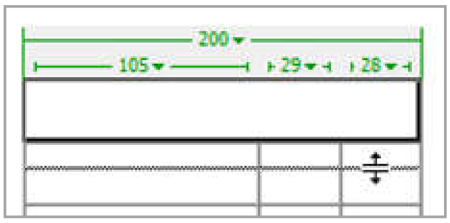
e. You can edit these settings by selecting the table (click 'Edit / Table / Select Table' or right-click in the table and use the context meu) then opening the *Property Inspector* ('Window / Properties' or 'right-click / Properties').



Merging two or more cells together

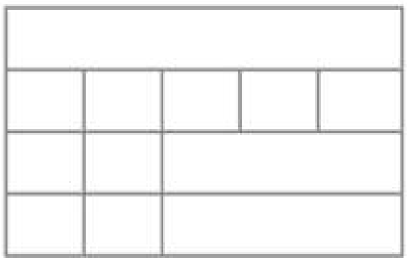
We are going to merge the top row of cells into one single cell. Select the three cells then click 'Edit / Table / Merge Cells' (or use the right-click context menu). You should now have a table something like the one planned.

You can drag the borders of the table to make the shape of the cells suit your design.



Splitting a cell into two or more columns or rows

Sometimes, you may want to split a cell to create extra rows or columns. To do this, place the cursor in the cell you wish to split and click 'Edit / Table / Split Cell'. You can then decide how many rows or columns you would like to create in that one cell and click 'OK'.



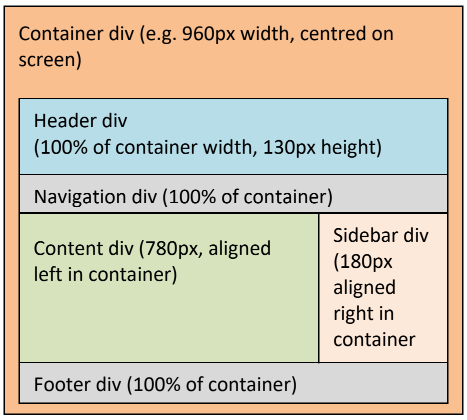
K. Creating the Header Div

So far, we have created a container for our other divs. The reason for using the container is that it will give our webpage a maximum width, whatever size screen we are using. It will also keep everything centred horizontally.

It is now time to build the other divs we need to lay out our content. Bear in mind the following points:

- If we do not set a width for a div, it will automatically take the full width of the container. This is especially useful when we start using margins or padding, both of which can mess up our calculations.
- If we do not set a height for a div, the div will generally stretch to fit the content that you are placing in it.
- All this behaviour can be controlled. For example, we could fix the height of a div and then decide whether any extra content overflows the bottom edge, is clipped or can be viewed by scrolling downwards. Divs are really quite useful once you get the hang of them.

Creating the Header Div


Container div (e.g. 960px width, centred on screen)
Header div (100% of container width, 130px height)
Navigation div (100% of container)
Content div (780px, aligned left in container)
Sidebar div (180px aligned right in container
Footer div (100% of container)

We'll use a similar method to create the header div as used before.

a. Delete the automatic text in the container div but leave the (very large) cursor in the container. Click 'Insert / Div'.

b. Insert the div 'At insertion point', give it the ID 'header' and click on the 'New CSS Rule' button then 'OK' in the next window.

c. Choose a new background colour. To make your site appear more professional, use the header colour from your other pages.

d. Set the height to 130px. Leave the width blank to make the div the full width of the container. Click 'OK / OK' to create the div.

Undoing Div Creation

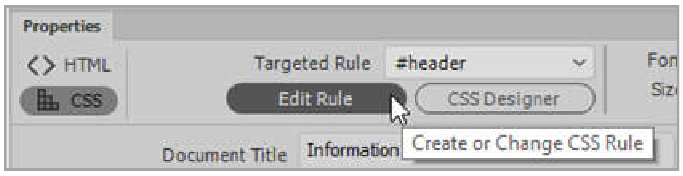
This might sound strange, but when you create a div, two extra steps are taken behind the scenes before the div appears. The extra steps involve creating more of the code that we are trying to avoid. Clicking *Undo* once will appear to remove the div, but it leaves behind some code each time. Therefore, if you want to completely undo a div you have created, you should click *Undo* three times. Clicking a fourth time should make another change that you can observe, in which case click *Redo* once.

Editing a Div

A better method to use, where possible, is to edit the div you have created. To edit a div:

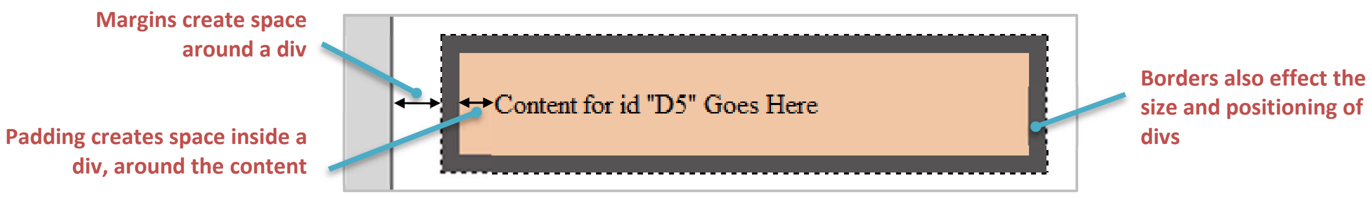
a. Place your cursor inside the div and open the *Property Inspector* ('Window / Properties').

b. Make sure the CSS tab is selected and that the 'Targeted Rule' box includes the ID of the div you would like to edit.



c. Click on the 'Edit Rule' button to open the 'CSS Rule definition' window again. You can now make your changes.

Margins and padding can help us lay out content precisely so that it looks good, but they also make things more complicated when it comes to calculating the widths and heights needed for our divs. Borders may also be added for effect, but again their dimensions need to be taken account of.


Margins create space around a div
Padding creates space inside a div, around the content
Content for id "D5" Goes Here
Borders also effect the size and positioning of divs

We will have a look at what effect these things have using some divs created in an extra page.

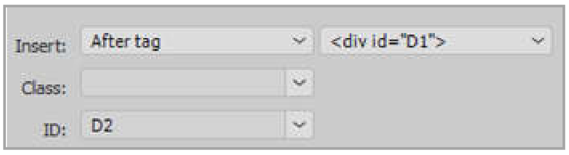
Testing the Effects of Padding, Margins and Borders

a. Create a new, blank HTML page. Save it as 'div_test.html'.

b. Add a div with an ID of 'D1'. Click on the 'New CSS Rule' button, 'OK' and then select any background colour.

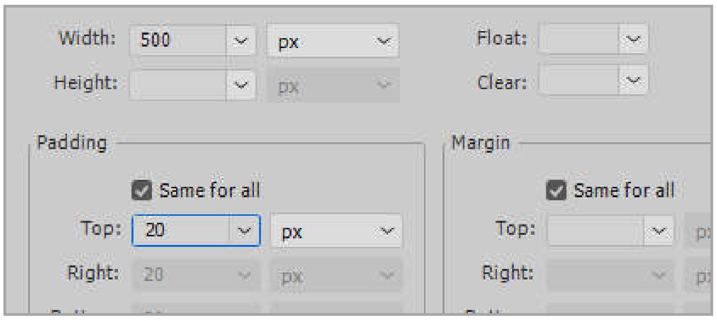
c. Set a box width of 500px. You do not need to change any other settings. Click 'OK / OK' to create the div.

d. Insert a 2nd div with an ID of 'D2'. In the 'Insert' window, select 'After tag' and '<div id="D1">' so that this div is positioned after the first, then click on the 'New CSS Rule button'. Click 'OK' in the next window.



Insert: After tag | <div id="D1">
Class:
ID: D2

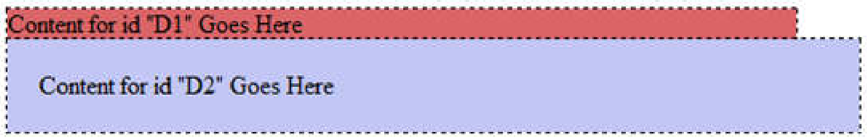
e. Give this div a different colour, again a width of 500px, but this time add a padding of 20px to all edges (see right). Click 'OK / OK'.



Width: 500 px
Height: px
Float:
Clear:
Padding
Same for all
Top: 20 px
Right: 20 px
Margin
Same for all
Top:
Right:

What is the difference between the two divs created so far?

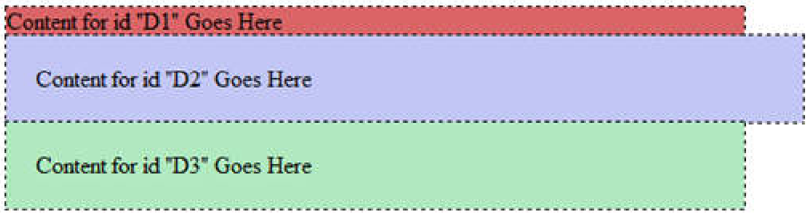
You should find that the new div is bigger than the first, even though we have given both of them a width of 500px. This is because the padding is added to the width making our second div 500+20+20=540 pixels wide.



If you wanted to include padding and keep the total width at 500px, then you would need to set a width of 500-20-20=460 pixels for this div. With the padding added on either side, the total will be back to 500px.

f. Create a 3rd div that tests this theory. Place this one after D2 and give it the ID 'D3'. Click 'New CSS rule' and select a colour, padding of 20px all around but a width of 460px.

You should end up with something like the picture on the right.



Content for id "D1" Goes Here

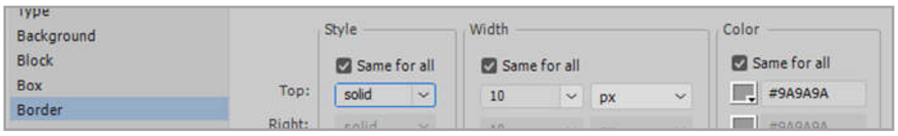
Content for id "D2" Goes Here

Content for id "D3" Goes Here

g. Add a 4th div after the last with the ID 'D4'. Again, use a different colour and a width of 500px, but this time leave the padding boxes blank and set each **margin** to 20px. You should notice that this div has been pushed out from the left-hand side by 20px and down from the div above by 20px. This is the margin.

h. Add a 5th div with an ID 'D5', after D4, with a new colour and a width of 460px. This time, set all the padding and margin boxes to 20px. We should have added the padding but taken account of this in our width setting.

i. Make the 6th div the same as the last, but select a different colour and give all sides a dark, solid border of 10px.



Style: Same for all ✓ — Top: solid; Right: solid

Width: Same for all ✓ — 10 px

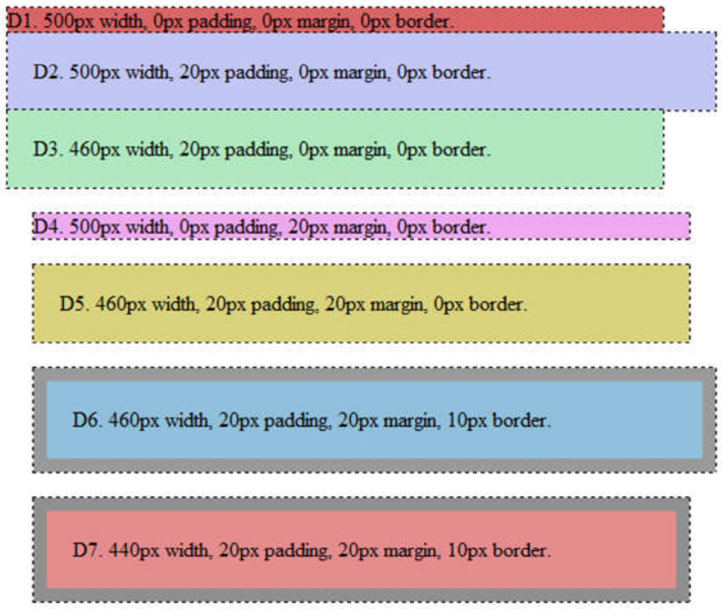
Color: Same for all ✓ — #9A9A9A

You should find that the border has a similar effect to the padding in that it also increases the size of the div. We must now subtract both the padding and the borders in order to end up with a div of 500 pixels.

j. Finally, add a 7th div which takes account of the border width. We want our final width to be 500px, so we remove from the calculation left and right padding (20px each) and left and right borders (10px each).

Width: 500 – 20 – 20 – 10 – 10 = 440 pixels

Set all padding and margin boxes to 20px and the border to 10px. The divs should appear something like the ones on the right.



D1. 500px width, 0px padding, 0px margin, 0px border.

D2. 500px width, 20px padding, 0px margin, 0px border.

D3. 460px width, 20px padding, 0px margin, 0px border.

D4. 500px width, 0px padding, 20px margin, 0px border.

D5. 460px width, 20px padding, 20px margin, 0px border.

D6. 460px width, 20px padding, 20px margin, 10px border.

D7. 440px width, 20px padding, 20px margin, 10px border.

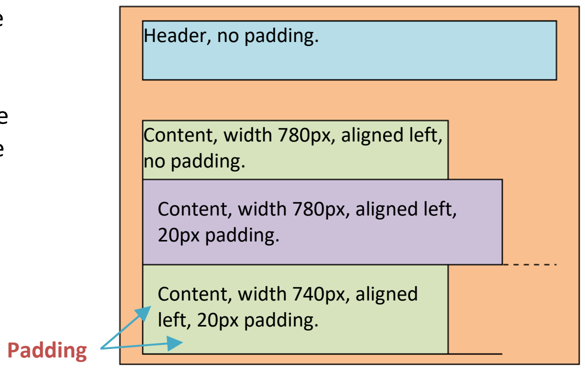
Taking Account of Padding

Our *content* div is going to be 780 pixels wide in total, aligned to the left of the container.

We are not worried about the vertical setting as we are happy for our div to be stretched downwards, but we would like to add some padding in order to take the text away from the edge of the screen and the div above (see right).

If we decide on 20px padding around all edges, our div width should be set as:

780px – 20px (left padding) – 20px (right padding) = 740px.


Header, no padding.
Content, width 780px, aligned left, no padding.
Content, width 780px, aligned left, 20px padding.
Content, width 740px, aligned left, 20px padding.
Padding

Note: *There are actually other ways of creating this effect that reduce the need for these calculations but which use a more complicated layout. However, it's great practice to think about all these issues now.*

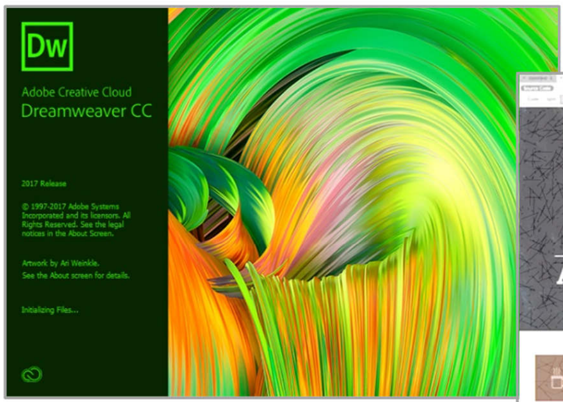
Dreamweaver CC 2017

HTML & CSS

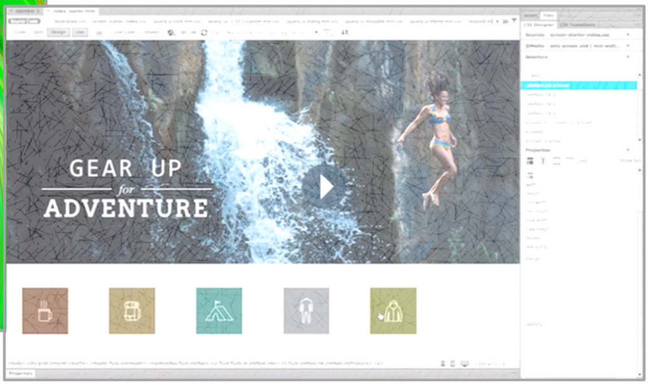
These advanced resources get stuck in to the HTML code and Cascading Style Sheets (CSS) that help designers build professional websites with a consistent feel across the pages.

If you'd prefer to avoid the coding, then perhaps have a look at our *Dreamweaver Basics* tutorial.

This version is for Dreamweaver CC 2017. There are also versions available for the earlier Dreamweaver CC 2015 and CS6.



Dw
Adobe Creative Cloud
Dreamweaver CC
2017 Release
© 1997-2017 Adobe Systems Incorporated and its licensors. All Rights Reserved. See the legal notices in the About Screen.
Artwork by Ari Weinkle.
See the About screen for details.
Initializing Files...



HTML stands for *HyperText Markup Language*. This is the language that tells your computer how to construct a webpage so that it can be viewed in your browser. The page itself is delivered as a simple text file. Your web browser has to interpret the meaning of this text so that it can display the page as it was designed to be seen. Pictures, movies and sounds are not part of the HTML, but separate files called upon by the webpage. The HTML simply shows how the browser should display these objects.

HTML makes use of tags. Most HTML tags tell the web browser when to start formatting in a certain style and then when to stop using that style. For example, to produce the strong, emphasised text '***This is HTML***' on your page, you could use the following code:

HTML Code	Tag name	Instruction to browser
<strong>	Strong	*- start using strong, or important, type (it appears bold)*
<em>	Emphasised	*- start using emphasised type (it appears as italics)*
This is HTML		*- write the text*
</em>	Slash emphasised	*- stop using emphasised type*
</strong>	Slash strong	*- stop using strong type*

The tags are the things in brackets. <strong> is the tag that instructs the web browser to use strong text. </em> is the tag that tells it to stop using emphasised text. The code for this text may all be written on one line:

```
<strong><em>This is HTML</em></strong>
```

Note: *There are actually tags called bold (<b>) and italic (<i>) which were once commonly used in HTML. These have a similar appearance to strong and emphasised. The difference is that with the strong and emphasised tags, the browser knows that the actual words between the tags are important. These days, this can make a difference in many situations, for example when the text is being read by the screen readers used by the visually impaired or on small mobile screens. Bold and italic effects can still be used, but these should be put into a style sheet (as shown later).*

Some points about HTML

1. Dreamweaver and other web design applications will allow you to build much of your site in WYSIWYG view (what you see is what you get – or *Design view*). However, to build professional, efficient websites based on a specific design, you need to get stuck into the HTML.
2. HTML can be written in *UPPER* or *lower*-case text (or a mixture). Generally, lower case is preferred.
3. Web browsers will ignore tags that they do not understand.
4. HTML is not a programming language. Mistakes will not cause your computer to crash – they will just cause the page to be displayed incorrectly in a browser.
5. Users can set their own preferences regarding text sizes, link colours and whether or not to display pictures. They also have different browsers, platforms and screen sizes. The task of a web designer is usually to design pages that are acceptable to as many people as possible. For example, you should not spend time trying to write text that exactly fits across your screen – it is highly unlikely to appear as you planned on different devices.
6. Tags should not be interlaced. i.e. for the ***strong, emphasised type*** above, we used the code:

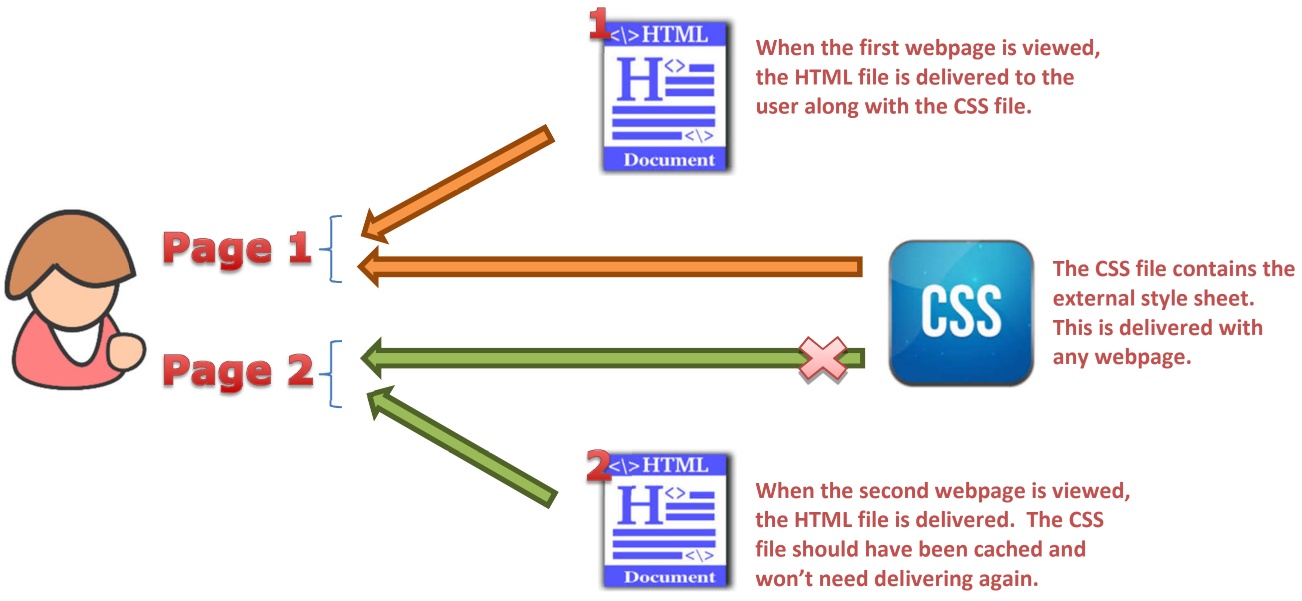
```
<strong>
<em>
This is HTML
</em>
</strong>
```

rather than...

```
<strong>
<em>
This is HTML
</strong>
</em>
```

Internal style sheets are great, but what if we want to use all those styles again in another page? We'd have to create the class selectors again and have all the same code in the head section. This is a waste of time and results in duplicate code being passed to the user each time a page loads.

The perfect solution will involve putting all your styling information into an external style sheet. This sheet is in a CSS file, named something like 'style.css'. The CSS file can then be delivered to the user with each webpage that is viewed. The beauty of this solution is that you only have to edit the CSS file in order to change the appearance of every webpage. Some large websites may have thousands of pages, each of which can be redesigned simply by editing the CSS file.


1
HTML
Document
When the first webpage is viewed, the HTML file is delivered to the user along with the CSS file.
Page 1
Page 2
CSS
The CSS file contains the external style sheet. This is delivered with any webpage.
2
HTML
Document
When the second webpage is viewed, the HTML file is delivered. The CSS file should have been cached and won't need delivering again.

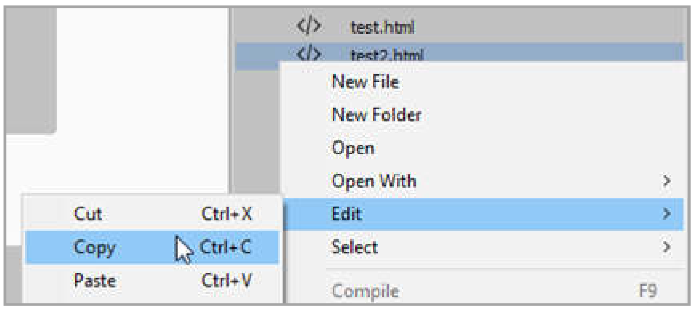
Caching

Another benefit of this method is that the CSS file doesn't actually have to be delivered through the internet each time a page loads. The user's computer will store a copy in a local folder called a cache. When another page loads using the same CSS file, the user's computer will check to see if the file on the website is still the same as the cached copy. If it is the same, the computer will just use the local copy. If it is different, the new version will be downloaded.

Task 1 – Creating a CSS File

a. Start by making a copy of your 2nd test page, partly as a record of your work and partly as a backup in case things go wrong. You may copy and paste the file in the folder outside Dreamweaver, or use the *Files* panel on the right.

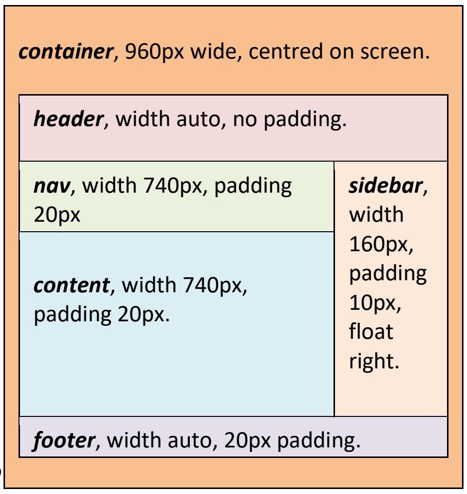
   If using the *Files* panel, right click on the file name and use 'Edit / Copy' and 'Edit / Paste' to duplicate the file.
   Name the file 'test3.html'.



H. Creating the Container Div

We will now create our divs. You may have learned a method of creating divs in our basic tutorial that used the menus and windows to avoid complex CSS processes. However, if you have understood everything so far, then creating divs using CSS is easy. What's more, with the div styles being placed in an external CSS file, we can use them again quickly in our other pages.

Task 1 – Creating your Container Div


container, 960px wide, centred on screen.
header, width auto, no padding.
nav, width 740px, padding 20px
sidebar, width 160px, padding 10px, float right.
content, width 740px, padding 20px.
footer, width auto, 20px padding.

We are going to create the 6 divs shown on the right in order to control the layout of the pages. For now, we'll colour each one so that we can see what it looks like. We can remove the colour later.

The first div is the container.

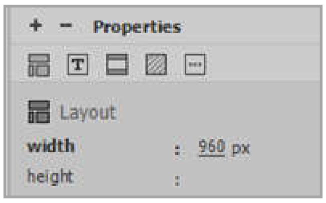
a. Open your *index.html* page (the homepage) and delete any text that you entered previously.

b. We will place all the div styles in a CSS file so that we can use them on each page. Using the CSS Designer, create a new source called '**style.css**'. Place this in your *css* folder.

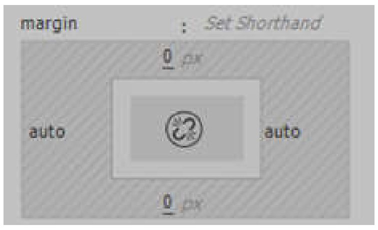
c. Select the 'style.css' file in the *Sources* section of the CSS Designer, then click the *Add Selector* button. Name this selector '**#container**'.

Why the #?
This time, our selector starts with a hash (#) rather than a dot. This is because o
An ID is an element that will appear only once on a page. A class is a style that can be used repeatedly. There is a reason for the difference, but for now, just remember that an ID is unique on a page and starts with a hash; a class can be used multiple times and starts with a dot.

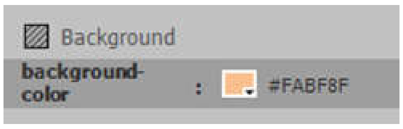
d. Select the new *container* selector and set the properties as shown below. The width should be 960px, the left and right margins set to 'auto' (resulting in the div being aligned centre) and a temporary background colour chosen from further down the list. Check that it is the background colour that you are setting and not the text or border colour.



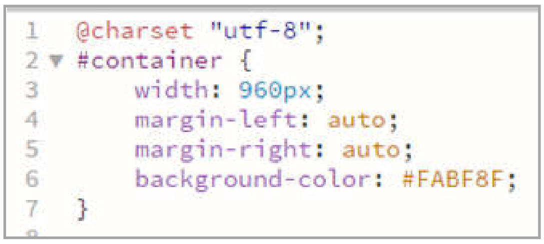
width : 960 px
height :



margin : Set Shorthand
0 px
auto
auto
0 px



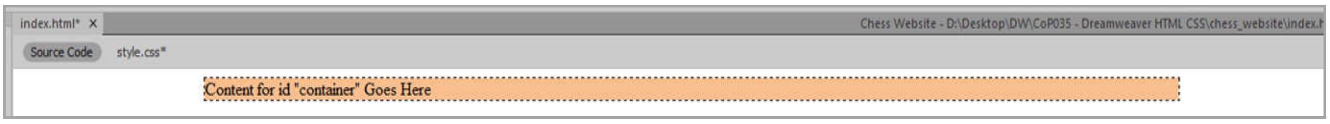
background-color : #FABF8F



```
@charset "utf-8";
#container {
    width: 960px;
    margin-left: auto;
    margin-right: auto;
    background-color: #FABF8F;
}
```

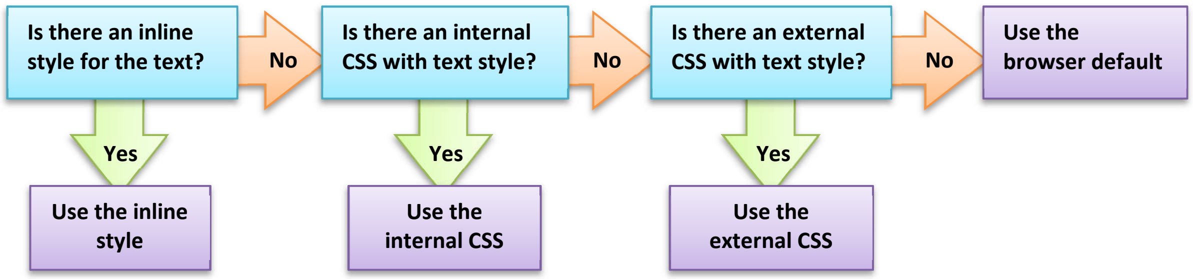
Have a quick look at the CSS file. It should now look like the one on the right (with the properties in any order and probably a different colour in hexadecimal).

e. With your selector now ready, place your cursor back in the empty Design window and click 'Insert / Div'. Select 'container' from the *ID* box and click *OK*. The div should appear in the top middle of the screen. Remember it will get taller as we place other content inside.


index.html* ×
Chess Website - D:\Desktop\DW\CoP035 - Dreamweaver HTML CSS\chess_website\index.h
Source Code
style.css*

Content for id "container" Goes Here

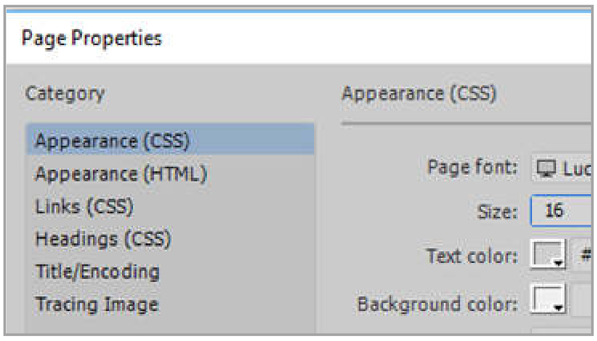
In our basic tutorial, we looked at how to use the *Page Properties* window to set out formatting and layout for the whole page. These settings are actually stored in an internal style sheet. We'll now look at how to set these properties for the whole website using our external CSS file. The diagram below shows how the styling decision is made by a browser when displaying some text on a web page.


Is there an inline style for the text?
No
Is there an internal CSS with text style?
No
Is there an external CSS with text style?
No
Use the browser default
Yes
Use the inline style
Yes
Use the internal CSS
Yes
Use the external CSS

Note: *The actual process takes place in reverse. The browser will at first adopt the most general rule and then replace it with any more specific rules it finds (e.g. an inline style will replace an internal style sheet). In this way, styles will cascade down to the most specific available, replacing the general rules on their way. This is why they are called <u>Cascading</u> Style Sheets.*

Task 1 - Setting Page Properties

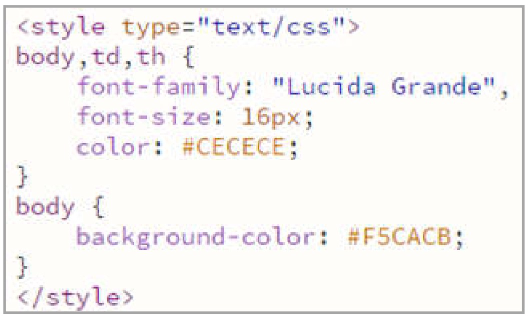
a. Still in your homepage, click on 'File / Page Properties'.
b. In the *Appearance (CSS)* tab, set any text properties you would like to apply to the whole page. Remember that you can override these settings on individual bits of text later, so go with what you would like to use most of the time.
c. Click *Apply* to see the effect of your changes and *OK* when you have made your decision.



Task 2 - Cutting and Pasting the Internal CSS

Have a look at your code. You should notice that setting the Page Properties has created an internal style sheet at the top of your webpage code (in the head section).

Any declaration block with the selector 'body' will apply the styles to the **body** section. Any declaration block using the selector 'body,td,th' will apply your font selections to anything in the **body** section, or in **t**able **d**ata / **t**able **h**eader cells.



```
<style type="text/css">
body,td,th {
    font-family: "Lucida Grande",
    font-size: 16px;
    color: #CECECE;
}
body {
    background-color: #F5CACB;
}
</style>
```

Cut these declaration blocks out and paste them into your CSS file (at the top, under the '@charset "utf-8";' line). They will now be applied to every page that uses your external style sheet. You can delete the leftover '<style type="text/css">' and '</style> tags from your webpage.

Note: *You could paste them further down your style sheet, but it's normal to put the major declaration blocks first.*

We will now add some images to your homepage. We'll have a go at creating an image of a specific size using Photoshop (or another image editor). Images can actually be cropped in Dreamweaver but the results are often unpredictable. Incorporating Photoshop is a good skill to learn.

It is very common to find pictures on the internet to use on websites. Please remember that every image is owned by someone and you can't just take another person's property without their permission (especially if you are going to publish your website on the internet). Thankfully, there are a large number of files on the web that you have been given permission to use. Images from Wikimedia commons, Flikr and many other sources can often be legally used. An easy way to find these images is to use Google.

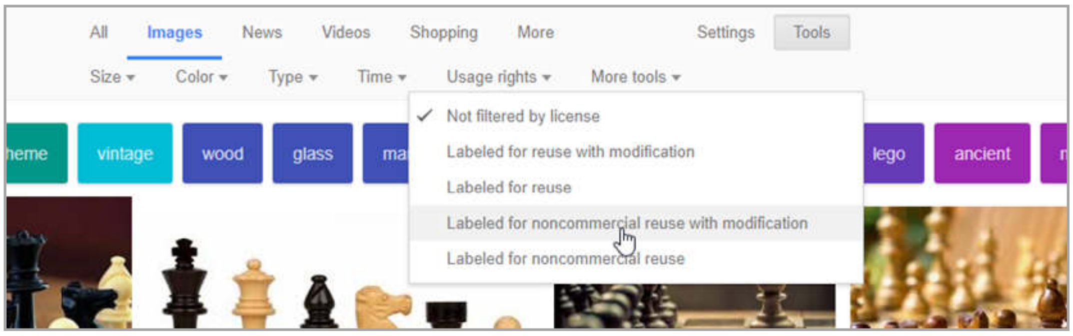
Note: *The images do not become part of the webpage file; they remain a separate file which is 'collected' by the page as it is viewed. You will need to save a copy of the images to your computer before adding them to your pages.*

Task 1 - Finding Legal Images

a. Perform a Google images search as usual. We are looking for an image, or part of an image, that will make a good banner across the top of the page. The part of the image you will use should be short and wide.

A possible target section of image.

b. Click on 'Tools' in the top menu and then on 'Usage rights' in the submenu that appears. Select 'Labeled for non-commercial reuse with modification' so that you are free to change the appearance of the images you find.


All
Images
News
Videos
Shopping
More
Settings
Tools
Size
Color
Type
Time
Usage rights
More tools
✓ Not filtered by license
Labeled for reuse with modification
Labeled for reuse
Labeled for noncommercial reuse with modification
Labeled for noncommercial reuse
heme
vintage
wood
glass
ma
lego
ancient

c. Mouse-over some of the results and find one that is at least 960px wide, or considerably more if you are not going to use the full width (Google gives you the dimensions). Remember that we are looking for a short, wide section of image.

Why should you find images of the right size?

Images that are smaller than the space you want to fill will need to be stretched. This lowers the quality of the display and might result in the image appearing blurred or fuzzy.

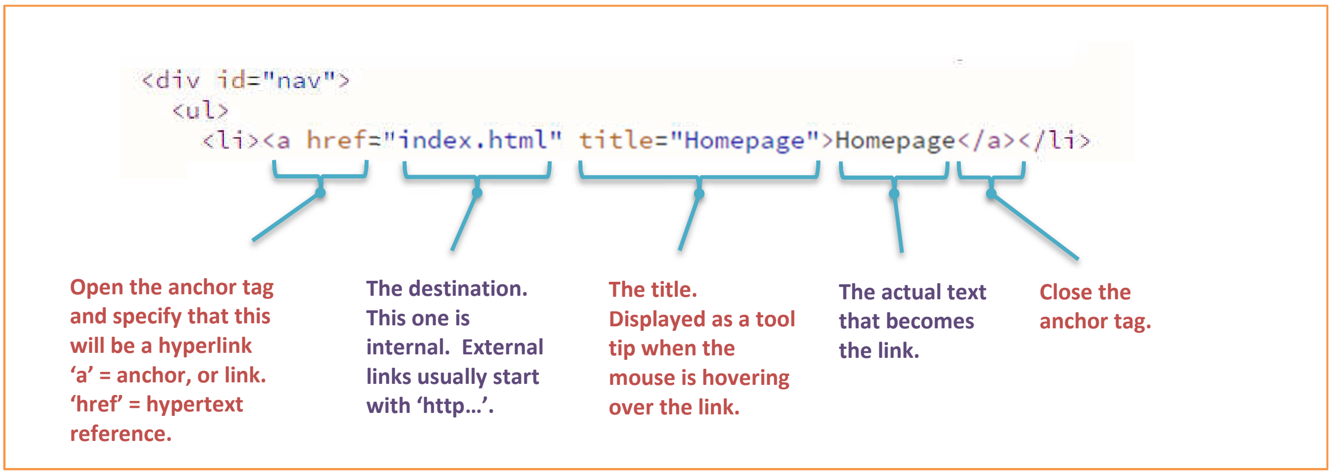
Images that are larger than the space you want to fill will need to be shrunk in size. This means that you are transferring more data than necessary over the web. Although internet speeds are increasing, you should still try and keep file sizes to a minimum.

Basically, if you know what size space you have, try and use an image of the same size.

So far, you have used the menu options to create hyperlinks which navigate between your pages. We will now look at the HTML behind some of the other types of hyperlink. These include:

- **Internal hyperlinks** *- link to a page on the same website*
- **External hyperlinks** *- link to a page on a different website*
- **Image hyperlinks** *- an image made into either of the hyperlinks above*
- **File hyperlinks** *- link to a file which is downloaded*

Have a look at the HTML code for your list of links. The first link should look something like the code below:



```
<div id="nav">
   <ul>
      <li><a href="index.html" title="Homepage">Homepage</a></li>
```

Open the anchor tag and specify that this will be a hyperlink 'a' = anchor, or link. 'href' = hypertext reference.
The destination. This one is internal. External links usually start with 'http…'.
The title. Displayed as a tool tip when the mouse is hovering over the link.
The actual text that becomes the link.
Close the anchor tag.

Internal (or Relative) Hyperlinks

Internal hyperlinks point to another page in the same website. All we need as a destination is the path to another file (this is why they are also known as *relative* hyperlinks). This example link points to a file called 'members.html'.

`<a href="members.html"></a>` *- links to a page in the same folder*

If you have a lot of pages, you may organise these into folders. For example, you may have created a page for each of the members in a club. These pages may be placed in a folder called 'personal'. In this case, the link from a webpage in your root folder to one of your member pages will look like this:

`<a href="personal/paul.html"></a>` *- links to the page 'paul.html' in a folder named 'personal'*

If you want to link from your page 'paul.html' back to your 'members.html' page, you will need to use the code '../'. This code effectively means 'move up one folder'.

`<a href="../members.html"></a>` *- links from the page "paul.html" back to the members page.*

Note: *In reality, a website designer wouldn't create lots of static member pages. They would store the information in a database and generate each page dynamically. This is a lot more complicated and won't be covered here.*

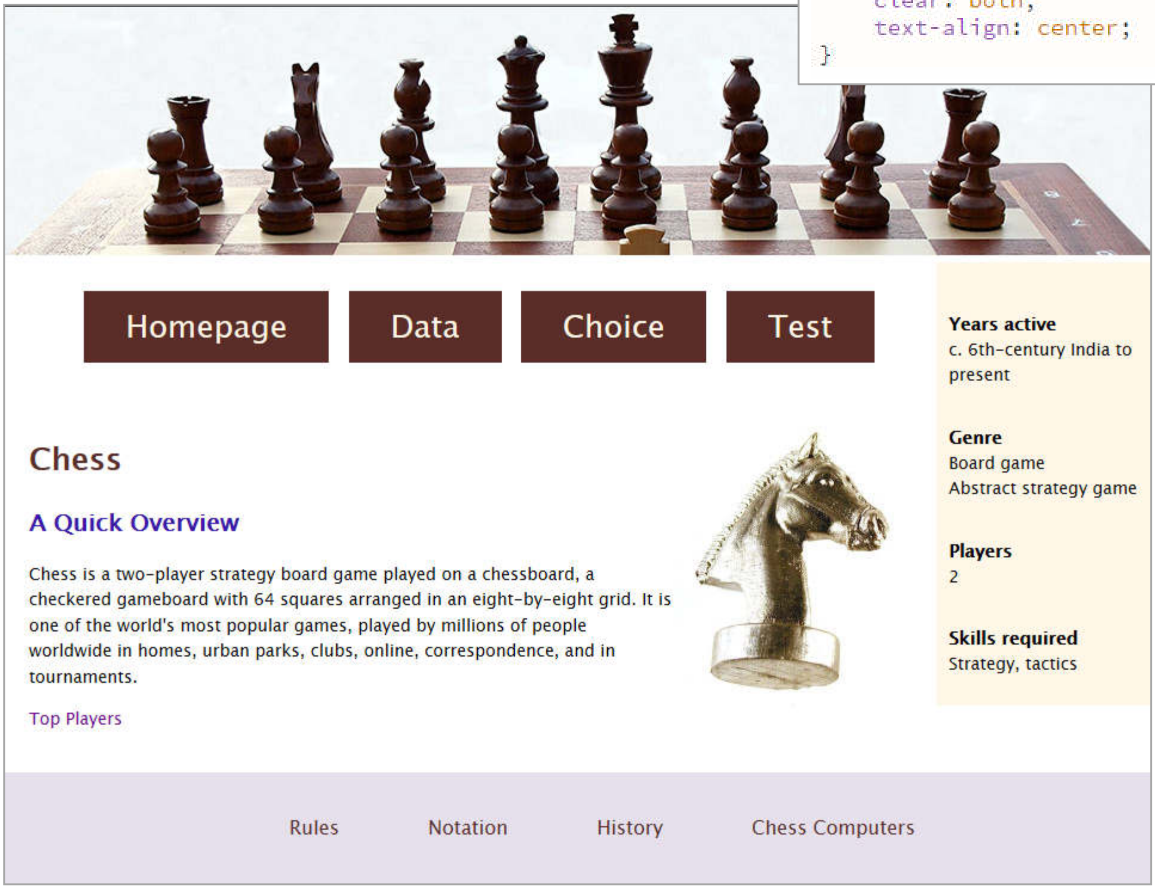
You should by now have enough skills to improve the quality of your homepage. You may do any of the following:

- Change or remove the coloured backgrounds from your divs. Perhaps change the container div to white (#FFF) so that your text is on a white background where it can be read easily. You could use the colour dropper to pick colours from your images.
- Improve the colours used for your navigation buttons, headings and text so that they fit with your header image colours.
- Add a margin to the left and right of your navigation buttons to separate them a little more.
- Add further images and text to the content and sidebar divs.
  Note: *We have used the heading style 'h3' in our sidebar, with a negative value used for the bottom margin. This has the effect of pulling up the paragraph below. This can be a useful trick.*
- Add links to other websites, perhaps in the footer. We have used the declaration 'text-align: center;' to align this list of links in the centre of the div.
- Make any other changes that improve your page.

```
#container {
    width: 960px;
    margin: auto;
    background-color: #FFF;
}
#header {
}
#nav {
    width: 740px;
    padding: 20px;
}
```

```
h3 {
    font-size: 15px;
    margin: 30px 0px -15px 0px;
}
```

```
#footer {
    padding: 20px;
    background-color: #E6E0EC;
    clear: both;
    text-align: center;
}
```



Chess

A Quick Overview

Chess is a two-player strategy board game played on a chessboard, a checkered gameboard with 64 squares arranged in an eight-by-eight grid. It is one of the world's most popular games, played by millions of people worldwide in homes, urban parks, clubs, online, correspondence, and in tournaments.

Top Players

Years active
c. 6th-century India to present

Genre
Board game
Abstract strategy game

Players
2

Skills required
Strategy, tactics

Rules Notation History Chess Computers

R. Summary of HTML and CSS

HTML

Header Tags	
<html> </html>	*- use HTML code*
<head> </head>	*- begin and end header section*
<body> </body>	*- begin and end body section*
<title> </title>	*- begin and end title*
<meta …>	*- include meta-tags*

Text Tags	
<strong> </strong >	*- strong text*
<em> </em>	*- emphasised text*
<h1> </h1>	*- use largest heading*
<h6> </h6>	*- use smallest heading*
<font face… size… color…><b><i><u> should all be set in CSS	

Layout	
<br>	*- insert single line break*
<p> </p>	*- create a paragraph*
<blockquote> </b..>	*- create a block quote*
<div> </div>	*- create a div*

Lists	
<ol> </ol>	*- create a numbered list*
<ul> </ul>	*- create a bulleted list*
<li> </li>	*- create a list item*
<dl>, <dt> & <dd> also used for definition lists	

Hyperlinks	
<a href="file.htm">	*- create a relative hyperlink*
<a href="http:// ">	*- create an absolute hyperlink*
<a href="#DivName">	*- link to target on page*
<a href="mailto:. ">	*- email hyperlink (now dated)*

Images	
<img src="file.png">	*- include an image*
<img width="50%">	*- set image width*
<img height="100">	*- set image height*

Table Layout	
<table> </table>	*- create a table*
<tr> </tr>	*- start and end a table row*
<td> </td>	*- start and end a table data cell*
<td colspan="2">	*- set columns to span*
<td rowspan ="2">	*- set rows to span*
<table width… height…>	*- fix size of table*
&nbsp;	*- space placed in empty cells*
<td width… height…> set in CSS	

Selectors	
<div style="…	*- apply inline styles to a div*
<div class="…	*- apply a class from a CSS to a div*
<div id="…	*- name a div and use the styles set*
Or eg. <p style="…, <td class="… etc	

There are lots more HTML tags, but try and place your styles in the CSS wherever possible.

CSS

Size and Position of Elements	
width	*- fixed width in pixels or as a percentage*
height	*- fixed height in pixels or as a percentage*
max-height	*- max height if not fixed (or min/width)*
position	*- fixed on the page or after the last element*
left	*- a set distance from the left of its container…*
right, top, bottom	*… or from the right, top or bottom*
float	*- eg. float right and let other elements flow*
margin-left	*- the left margin, or right, top, bottom*
margin	*- margin on all sides, top, right, bottom, left*
padding-left	*- the left padding, or right, top, bottom*
padding	*- padding on all sides, top, right, bottom, left*
clear	*- stop elements left, right or both overlapping*
visibility	*- set to 'hidden' for invisible elements*
z-index	*- stack order of overlapping elements*
overflow	*- allow contents to overflow*
display	*- display option eg. for lists.*

Text	
text-align	*- alignment of text, left, right or center*
text-indent	*- indentation of the first line*
line-height	*- line height in pixels or percentage of font size*
text-transform	*- uppercase, lowercase or capitilize (first letter)*
font-family	*- font family specific / generic eg. "arial", serif;*
font-size	*- size of the text in pixels*
font-style	*- normal, italic or oblique*
font-weight	*- normal, bold or e.g. 700 (where 400 is normal)*
font	*- set font properties in one declaration N.B. order.*
text-decoration	*- underline, overline, line-through*
text-shadow	*- position h, position v, colour e.g. 2px 2px #aaa*

Backgrounds	
background-color	*- background colour of an element*
background-image	*- location of background image*
background-position	*- position of a background image eg. left top*
background-repeat	*- how a background image repeats*
background-size	*- background image size in pixels or percentage*
background	*- set all the background properties N.B. order*
box-shadow	*- attach a drop-shadow to the box*

Borders	
border-width	*- width of all borders e.g. 3px*
border-style	*- style of all borders e.g. solid, dotted*
border-color	*- set the colour of all four borders*
border	*- set all widths, styles and colours*
border-top-width	*- width e.g. 3px (or left, right, bottom)*
border-top-style	*- style e.g. solid, dotted (or left, right, bottom)*
border-top-color	*- colour of the top border (or left, right, bottom)*
border-top	*- set width, style and colour in one declaration*
border-image	*- use an image to create a border*

Tables	
border-collapse	*- collapse, to create a single border*
border-spacing	*- space between cells eg. 10px*

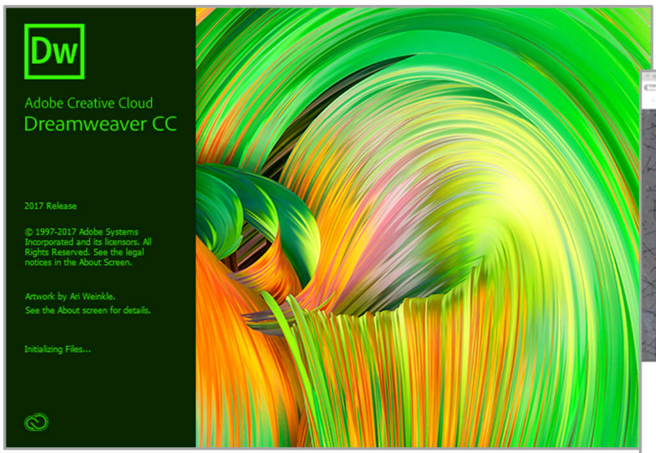
Look online for more CSS properties

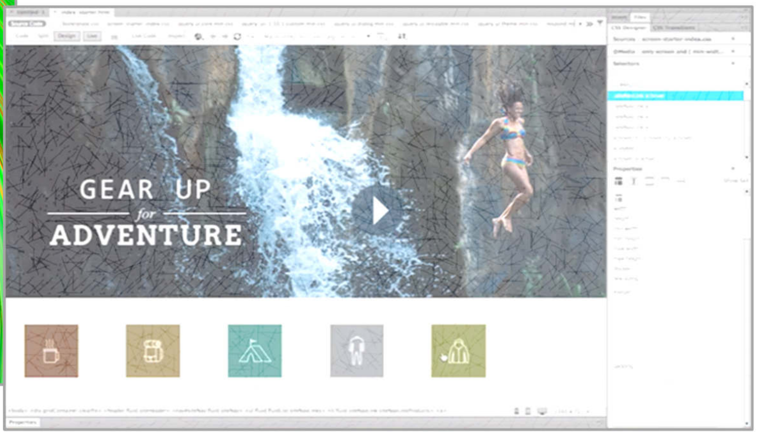
Dreamweaver CC 2017

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.


Dw
Adobe Creative Cloud
Dreamweaver CC
2017 Release
© 1997-2017 Adobe Systems Incorporated and its licensors. All Rights Reserved. See the legal notices in the About Screen.
Artwork by Ari Weinkle.
See the About screen for details.
Initializing Files...



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.

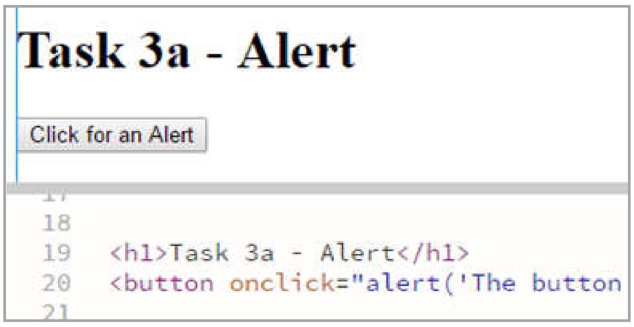
Note: *Many programmers use the developer tools in Chrome to debug their JavaScript. If you're feeling brave you could look at these, but they are also difficult to use for a beginner.*

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.



Task 3a - Alert

Click for an Alert

```
<h1>Task 3a - Alert</h1>
<button onclick="alert('The button
```

E. Strings

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"***

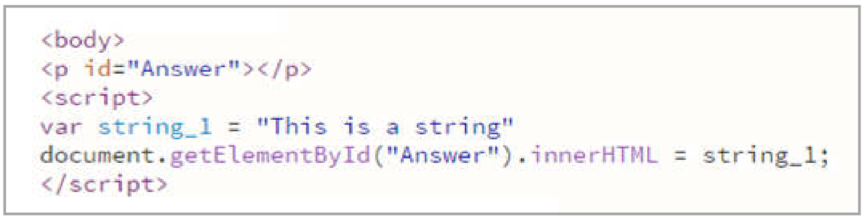
Strings are used in association with variables e.g.

	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.*

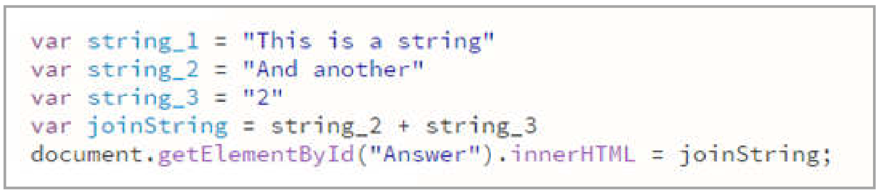


```
<body>
<p id="Answer"></p>
<script>
var string_1 = "This is a string"
document.getElementById("Answer").innerHTML = string_1;
</script>
```

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.



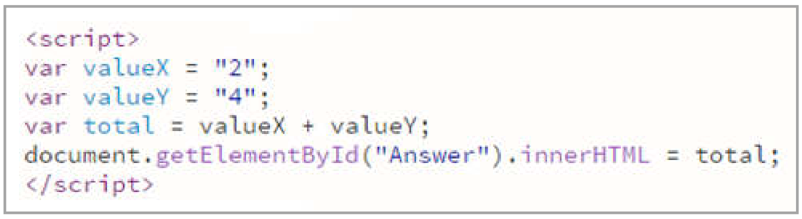
```
var string_1 = "This is a string"
var string_2 = "And another"
var string_3 = "2"
var joinString = string_2 + string_3
document.getElementById("Answer").innerHTML = joinString;
```

____

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.



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

____

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

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()'.

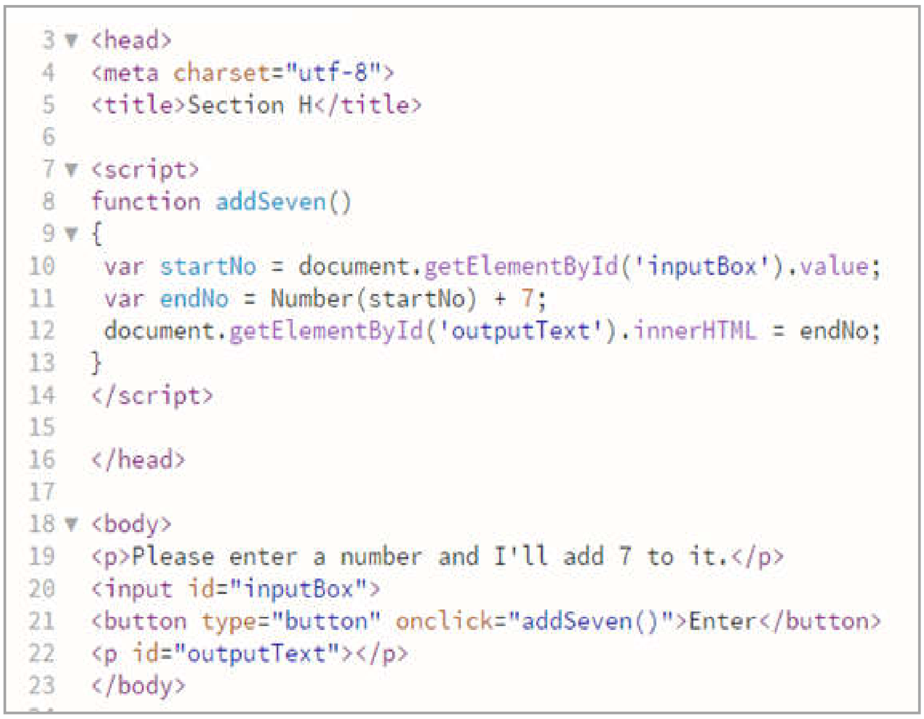
Open the head section

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.

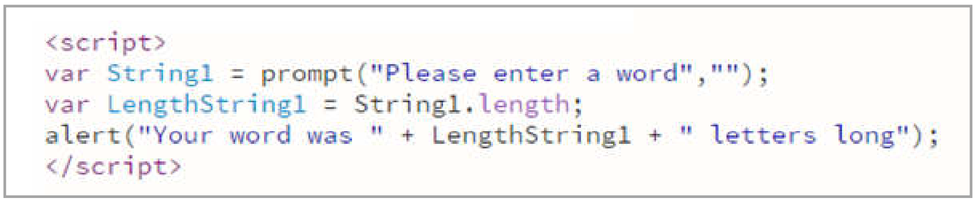
As far as the validation of form data is concerned, you may want to refuse information for any of the following reasons:

- 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.

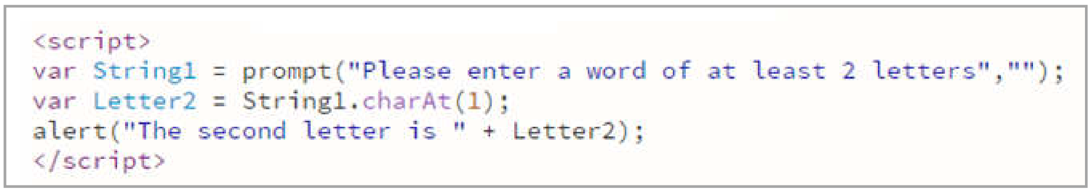


```
<script>
var String1 = prompt("Please enter a word","");
var LengthString1 = String1.length;
alert("Your word was " + LengthString1 + " letters long");
</script>
```

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.

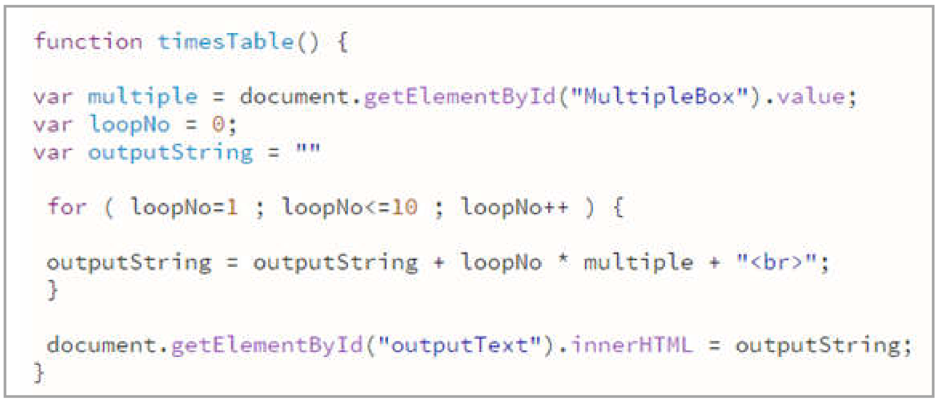
Statement 2
This is the condition for running the loop. We will continue looping while the variable 'loopNo' is less than or equal to 10.

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 <br> adds a line break between each result.



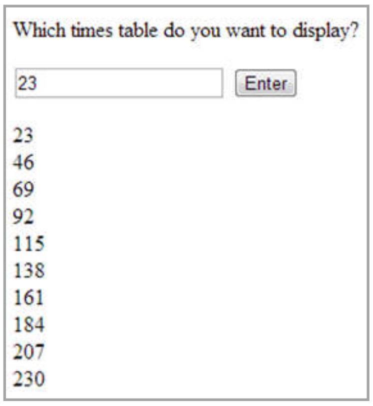
```
function timesTable() {

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;
}
```



Which times table do you want to display?

```
23
46
69
92
115
138
161
184
207
230
```