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.

Dreamweaver allows you to use visual tools along with HTML editing and Cascading Style Sheets (CSS) to produce a website with a consistent, professional feel. The Dreamweaver interface can be a little daunting to begin with, but with time and practice you will be zipping around the menus, panels and code windows in order to bring your ideas together.

Building Ideas

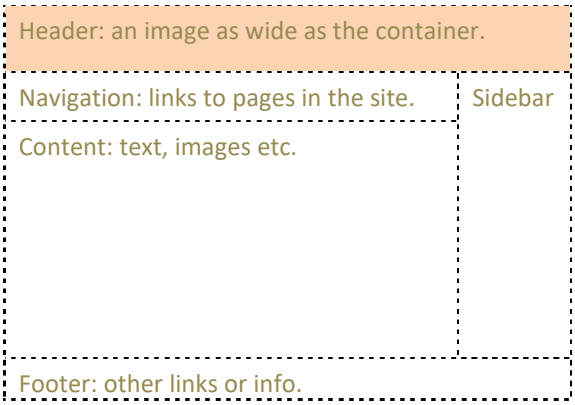
As in our basic tutorial, we are going to build a website with at least 4 pages. The difference this time is that we are going to use a professional approach to HTML and CSS from the beginning. Choose a topic of interest and focus on a small area of it rather than trying to cover too much (e.g. build a website about a specific car, club, sport, hobby or interest). Also, it may be better to choose a popular topic so that it's easy to find free images on the internet.

Write down some ideas for your website. You will need a subject, some information, images, facts and data. You could also sketch some ideas using the templates below, although you will be able to play with these ideas as your website develops.

Computer parts, prices, issues in the economy affecting the sale of technology, silicon shortage. Any issues in the market for technology would be explored.

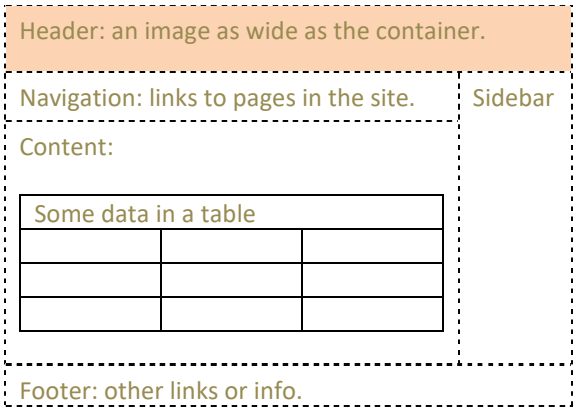
1. Homepage

Place some images, headings and information in a homepage.



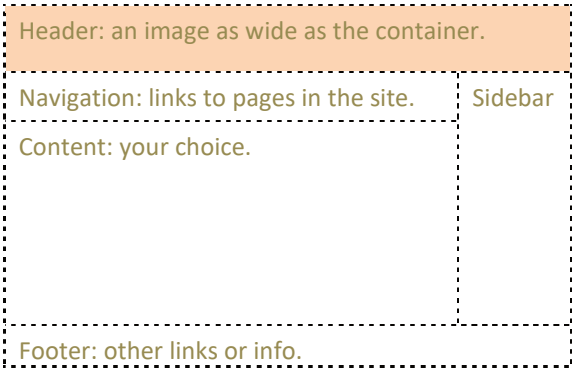
2. Data Page

We'll look at the HTML and CSS used to build and style a table.



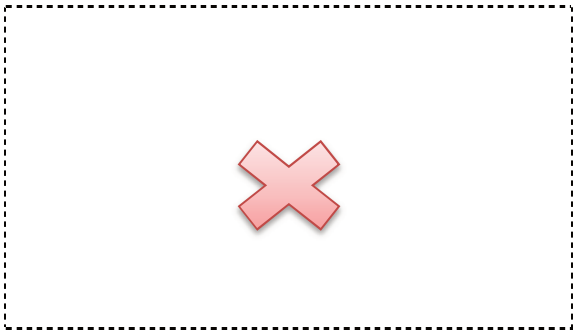
3. Your Choice

Create another page of your choice.



4. Test Pages

We'll use these pages to test ideas in.



We are going to begin by choosing a location for the files on your computer or network. We'll then create four blank pages, each of which is a separate html file. Later, we'll add hyperlinks between each of these files. Clicking on a hyperlink in your browser effectively opens a new file and closes the old one.

Task 1 - Creating Your Website

a. Before opening Dreamweaver, decide on the location in which you would like to save your files. This may be on your laptop, network or even a USB drive. Create a folder in your chosen location with a name such as "chess_website" (replacing 'chess' with whatever topic you have decided on). Use underscores rather than spaces.

b. Within this folder, create two more folders; one called 'images' and one called 'css' (for your stylesheets).

c. Open Dreamweaver and close (or ignore) any intro screens or tutorials that open.

d. Click 'Site' in this menu and select 'New Site' (this type of instruction will be written as 'Site / New Site' in future).

e. Give your website a name and click on the folder icon to the right of the *Local Site Folder* box. Locate and select the folder created previously. Ignore all the other settings on the left and click *Save*.

Note: *The main folder containing your website files is known as the 'root' folder. All files used in your website, whether webpages, images or style sheets, should remain within the root folder.*

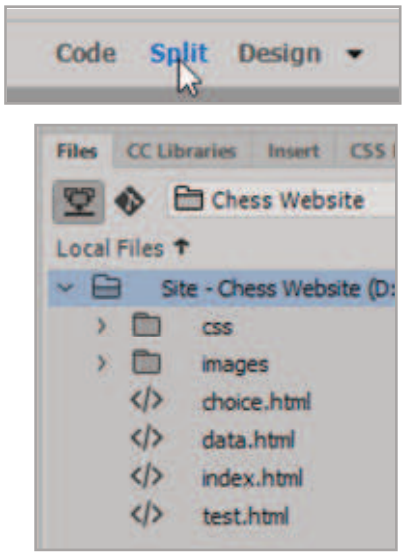
f. In future, your site could open automatically or may be available in the site manager. If you are using a different computer, however, you may need to locate the site again. To do this, click on 'Site / New Site' then the *Browse for folder* icon and locate your root folder. Click *Save*.

Task 2 - Creating the Pages

a. We'll now create our four blank pages so that we can put the hyperlinks in place first. Click on 'File / New'.

b. Make sure '**New Document**' is selected in the 1st column, '**HTML**' in the *Document Type* column and '**<none>**' in the *Framework* column. Leave 'HTML5' selected on the right and click *Create*.

c. Click the *Split* button in the top sub-menu so that the webpage code is shown in the bottom half of the screen.



d. Add the text 'Homepage' to the blank area of your page and save as '**index.html**' (*index.html* is usually the first file found when people visit your website).

e. In the same way, create a data page (add the text "Data" and save as 'data.html'), a test page and another page of your choice.

f. If it is not open already, click on the small double arrow in the top-right to expand the *Properties panel*. Click on the *Files* tab to show your 4 new files. You should also see the *css* and *images* folders created earlier. Files can be opened by double clicking on their names in this panel.

Note: *If the double arrows are not visible or the Files tab is missing, try pressing the F4 key to hide/show panels or play with the 'Workspace Layout' settings under 'Window' in the main menu.*

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	- *start using strong, or important, type (it appears bold)*
	Emphasised	- *start using emphasised type (it appears as italics)*
This is HTML		- *write the text*
	Slash emphasised	- *stop using emphasised type*
	Slash strong	- *stop using strong type*

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

This is HTML

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>
```

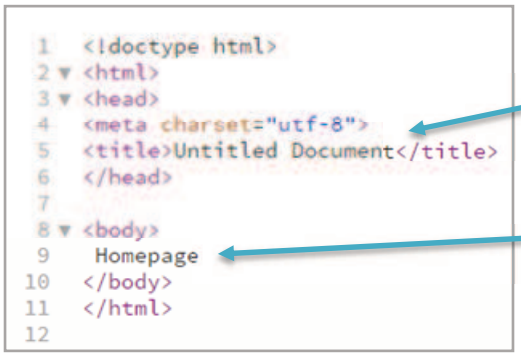

Initial HTML

HTML can be written in any web design application or text editor – it's simply text. Most design applications will put some HTML code in place when you create a new page. The initial HTML used in Dreamweaver is shown below.

Task 1 – Looking at the Initial HTML

a. Open your homepage (we gave it the filename 'index.html'). So far, we've just typed the text 'Homepage' on the page.

b. You should have *Split view* open so that you can see the HTML code in the editor.



The page title can be changed here. This is the title that appears in the browser tab when the page is open. It is also picked up by search engines such as Google.

The HTML code used to organise the layout of the web page will be placed here, between the body and slash body tags.

Note: *Our screenshots and coding are coloured as in the 'Light' theme. This can be set in 'Edit / Preferences / Interface'.*

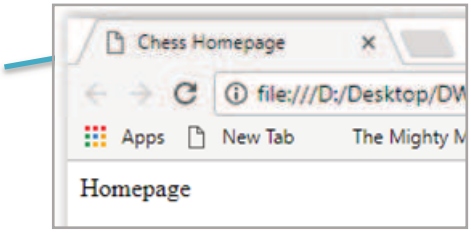
What do these tags mean?

<!doctype html>	*- Tells the web browser to expect html. This tag does not need to be closed with a slash tag.*
<html>	*- Start using HTML.*
<head>	*- Start the head section (page titles, instructions, scripts and metatags will follow).*
<meta charset="utf-8">	*- Information about the text characters used in the code. No </meta> is required.*
<title>	*- Start the page title.*
Untitled Document	*- The current title. This will be changed below.*
</title>	*- End the page title. (The title tags are actually on a single line. This makes no difference).*
</head>	*- End the head section.*
<body>	*- Start the body section. This is where your content will go.*
Homepage	*- The only content so far is the word 'Homepage' in plain text.*
</body>	*- End the body section.*
</html>	*- Stop using HTML.*

Task 2 – Editing the Page Title

We're going to have a go at editing the HTML code. First, type a title between the '<title>' and '</title>' tags (where it says 'Untitled Document'). This name might appear at the top of the screen or in the tabs when you are browsing. It will also be picked up by search engines, so choose a descriptive title.

Save the page then right-click on *index.html* in the *Files panel* on the right. 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.). The new title should be apparent in the page tabs.



Task 3 – Typing HTML (Test Page)

Open your test page and give it a new title as you did with your homepage before. Now add the following text to your page by placing HTML code **between** the *body* (<body>) and *slash body* (</body>) tags. You will need to use the , , and tags.

a. Write the word '***one***' in strong, emphasised text. You will find that Dreamweaver finishes off the job of closing tags for you.

b. Place all this code in a paragraph using the and tags, as shown on the right. You will need to cut and paste the tag Dreamweaver creates automatically.

c. In a separate paragraph, follow this text with the word 'two' using emphasised, but not strong text.

d. Finish with the word 'three' in strong, but not emphasised text.

```
 8 ▼ <body>
 9   <p><em><strong>One</strong></em></p>
10   <p><em>Two</em></p>
11   <p><strong>Three</strong></p>
12   </body>
```

Note: *The line numbers could be different in your HTML code but the order of the instructions must be the same.*

Practicing HTML with Outdated Tags

We are going to have a go at creating some more text using HTML code. **But, please keep in mind the following:**

Font tags and styles should NOT really be used any more. You will learn to style text using CSS.

Text can be formatted using the following tags:

	- sets and closes a font with particular attributes
	- sets and closes bold type
	- sets and closes italic type
<u></u>	*- sets and closes underlined type*

You can use the font tags in association with the following attributes:

<font face = "Arial">	*- use the Arial font (or whichever font you would like to use).*
<font size = "2">	*- use size 2 font (10pt) on a scale of 1 to 7 (8pt-36pt)*
<font color = "red">	*- browsers will recognise a limited number of colours as words. In general, hexadecimal numbers are used to assign colours. We will look at these later.*

You may set as many font attributes as you wish within one font tag e.g.

```
<font face = "Arial" size="2" color="red">Red Arial text</font>
```

Try out some of this code in your test page until you get the hang of opening and closing tags. See if you can reproduce the text on the right using only the paragraph, font, bold, italic and underline tags.

Arial, size 2, red, bold text

Arial black, size 4, green, italic text

Verdana, blue, bold, italic, underlined

If you completed the last activity, you should have created some Verdana, blue, bold, italic, underlined text using HTML code like the line below (you may have put the tags in a different order, but remember that they shouldn't be interlaced).

```
<p><font face="Verdana" color="blue"><b><i><u>Actual Text</u></i></b></font></p>
```

Search Engines

The problem with all the HTML tags above is that they make the code very messy. This isn't just a problem for the poor web designer who's struggling to keep on top of all these styles, but it is difficult for the search engines that are trying to work out what the page is actually about. Search engines send their robotic crawlers around looking for meaning in websites; they want to know what the content is related to so that they can send people to the right place when they search the web. However, the search engines' robots don't see a beautiful, formatted page; they have to wade through all the code looking for the actual text. Wouldn't all this be easier if you could just have the code below?

```
<p style="crazy">Actual Text</p>
```

That's much tidier. All we have to do now is tell the visitor that our style called "crazy" should use the font Verdana and be blue, bold, italic and underlined. This information can be stored well out of the way in a style sheet. The task of finding meaning will then be much more straightforward for the search engine. This is one advantage of style sheets.

Separating Styles

Have a look at the example HTML code below. There are three paragraphs all using blue, underlined Verdana text.

```
<p><font face="Verdana" color="blue"><u>Some Text</u></font></p>

<p><font face="Verdana" color="blue"><u>Some More Text</u></font></p>

<p><font face="Verdana" color="blue"><u>Even More Text</u></font></p>
```

If we want to change the colour of all three paragraphs to red, how many places do we have to change the HTML code? What about in this example?

```
<Separate style code which states that the 'class' crazy = blue, Verdana, underlined>

<p class="crazy">Some Text</p>

<p class="crazy">Some More Text</p>

<p class="crazy">Even More Text</p>
```

Another advantage of separating styles is therefore that we cut down the work required when changing the design of our website. If we use our 'crazy' style in a number of places, changing the style will affect all the places it is used. What about this last one?

```
<Separate style code which states that the 'class' crazy = blue, Verdana, underlined>

<Use crazy style>

<p>Some Text</p>

<p>Some More Text</p>

<p>Even More Text</p>

<Stop using crazy style>
```

Now we're really starting to separate the formatting from the actual text. This is what Cascading Style Sheets are all about.

Inline Styles

Having said all this, we are going to start with an easy method of styling text. We'll move onto separate style sheets once we have got our heads around how styles work. Like HTML styles, inline styles are placed within the HTML tags. Try carefully typing the code below into your test page.

```
<p style="font-weight: bold; color: blue;">This is some bold blue text</p>

<p style="color: red; font-style: bold; margin-left: 30px;">This is red italic text with a margin</p>
```

Can you work out what all this code is doing? You must type the colons and semicolons accurately for it to work correctly.

Writing CSS

CSS uses *selectors*, *properties* and *values* to create each *declaration*. We have used the code font-weight: bold above, for example. In this declaration, 'font' is known the *selector*, 'weight' is the *property* and 'bold' is the *value*. There are lots of other properties and values that you can use to create font declarations, as shown below.

Selector	Property	Value	Declaration Example	Meaning of Example
font	style	normal, italic, oblique	font-style:italic	Use italic text
	weight	normal, bold, 900 etc.	font-weight:bold	Use bold text
	size	e.g. 24px, 200% etc.	font-size:16px	Use text 16 pixels high
	family	Times, Courier, Serif etc.	font-family:Arial	Use Arial text

To create our inline styles, declarations such as font-weight: bold are placed within the style attribute e.g. style="font-weight:bold". This attribute can be part of a paragraph tag e.g. <p style="font-weight:bold"> or perhaps a heading <h1 style="font-weight:bold">.

Note: *<h1> is generally the largest heading used in HTML. The headings <h2>, <h3> …. <h6> have decreasing sizes.*

Example Inline CSS - Font Properties

1. You basically start with an HTML tag, for example:

   ```
   <p>Some Text</p>
   ```

2. You then insert the style attribute.

   ```
   <p style="">Some Text</p>
   ```

3. Add a declaration made up of a property / value pair. The property should be separated from the value using a colon ':'.

   ```
   <p style="font-style: italic">Some Text</p>
   ```

4. Add a second declaration. Each declaration should be separated by a semicolon.

   ```
   <p style="font-style: italic; font-weight: bold";>Some Text</p>
   ```

5. And add a font size declaration for good luck.

   ```
   <p style="font-style: italic; font-weight: bold; font-size: 24px;">Some Text</p>
   ```

6. And finally, a font family declaration (the declarations can actually be placed in any order).

   ```
   <p style="font-style: italic; font-weight: bold; font-size: 24px; font-family: Arial;">Text</p>
   ```


Task 1 – Testing the Code

One by one, test each line of code above in your test page (you only need one line of code; simply edit it each time). Click on the upper *Design* section of the screen to see what effect the code has.

a. Which addition makes no difference? Why do you think this is?

Notice that the browser will simply ignore anything that it doesn't understand or seems incomplete.

b. Try and create the following text using CSS styles.

i. A paragraph of bold, size 20px text.

ii. A heading that uses the <h1> tag and italic text.

Task 2 – Shorthand

You can often save time by typing styles as shorthand. However, for this to work, you must type the properties in the correct order (although you can miss out any values that you don't need to use). With the four properties considered so far, the order has to be:

font: *font-style* *font-weight* *font-size* *font-family*

```
<p style="font: italic bold 24px Arial">Some Text</p>
```

Try the above code and then use shorthand to create another line of bold, size 20px text. Miss out any values that aren't used.

Using the Menus

Dreamweaver offers both wysiwyg editing (what you see is what you get) and code editing. Although it is necessary for a professional web designer to learn how to code, remember that it is often faster to use the tools available in the visual interface. We'll have a quick look at creating HTML and inline styles using the Dreamweaver menus.

Task 1 – Adding Spaces and Text

a. Using the *Design view*, place your cursor after the last text in your test page. Press the *Enter* key twice to create a paragraph space (perhaps not the best method, but hey!) then type the lines below:

Format with HTML tags

Format with Inline Styles

b. Remove any styles from the code that have carried over. The code should appear something like that on the right.

```
<p> </p>
<p>Format with HTML tags</p>
<p>Format with Inline Styles</p>

</body>
</html>
```

Note: *** *** *stands for 'non-breaking space'. This is just the code for a space like the ones you use between words. Without this, the paragraph would be empty and may be ignored by some browsers. We want a paragraph space on our page so we must put something in it. A similar situation occurs with empty cells in a table.*

Task 2 – Formatting HTML using the Menu

There are several formatting changes you can make by simply selecting some text and applying HTML formatting.

a. Select your first line of text ("Format with HTML tags") and click 'Edit / Text / Bold' (or 'right-click / Style / Bold').

b. Repeat this, selecting the *Italic* and *Underline* options.

c. The code should now appear similar to that on the right.

```
<p> </p>
<p><strong><em><u>Format with HTML tags</u></em></strong></p>
```

d. Have a play with the other HTML styles to see what is possible. Keep in mind that HTML styles should be avoided unless you have a good reason to choose them over CSS.

Task 3 – CSS Styles using the Properties Window

a. Select the second line of text and click 'Window / Properties' to open the *Property Inspector*.

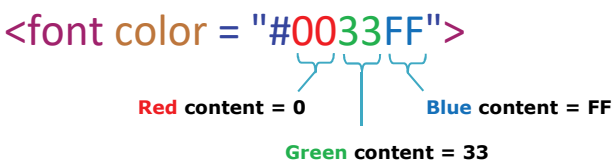
b. Click on the *CSS* tab on the left (the *HTML* tab changes the HTML tags as used above).

c. Make sure that the *Targeted Rule* is '<New Inline Style>' (change it if necessary) then select a style and size for your text.

d. Select a colour from the selector and click outside to close it again. Notice that this will produce a 6-digit hexadecimal number (#A200D7 in our case). We have explained these below.

e. Look at the code. Dreamweaver has produced some inline styles like the ones we typed.

A Short Introduction to Colours in Hexadecimal



To access the full range of colours, you need to use their 6-digit hexadecimal code. The code is actually 3 pairs of digits. The first 2 digits give the red content of the colour, the second 2 digits give the green content and the last 2 digits give the blue content. All colours are therefore a mixture of red, green and blue.

Hexadecimal has a base of 16, meaning that there are 16 different single digit numbers. These start at 0 as usual, but need to go past the ones used in base 10. In this system, letters are used for the remaining characters i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. The highest possible content for a single colour is FF (equivalent to 256 in decimal). It is therefore possible to have 256 x 256 x 256 (about 16 million) different colours.

The codes for a few colours are shown below:

Black = #000000 **White = #FFFFFF** **Red = #FF0000** **Green = #00FF00** **Blue = #0000FF**

Orange = #FFAA00 **Navy blue = #000080** **Yellow = #FFFF00** **Purple = #800080**

So far, we've learned how to write inline styles. These are styles written within the HTML tag itself. However, if you've understood everything so far, you may be asking the following question:

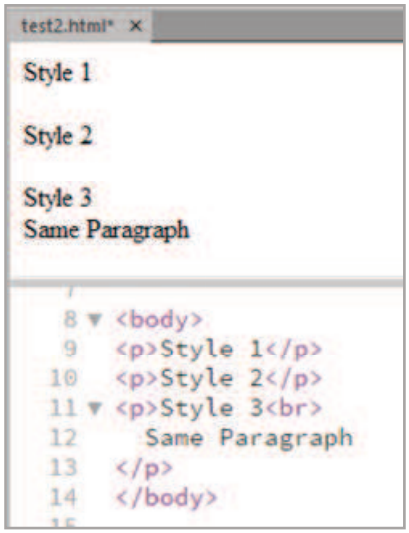
What's the point in avoiding HTML font tags if we're just going to fill our code with styles instead?

The answer is that there's no point. We didn't want to confuse search engines with font tags placed everywhere and equally we shouldn't start using inline styles frequently either. In a professional website, inline styles will rarely (if ever) be used.

It's time to start separating the styles from the HTML. We are going to begin by placing our styles in style sheets at the top of the page, inside the head section of the code. Search engines know that they can ignore much of this section so they are better able to find meaning in the rest of the text.

Task 1 – Creating an Internal Style Sheet

Your test page is probably getting a bit messy. Keep it as a record of your work so far, but create and save a new page named 'test2.html'.

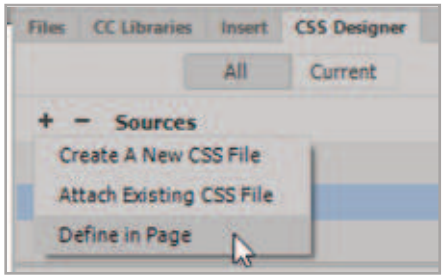


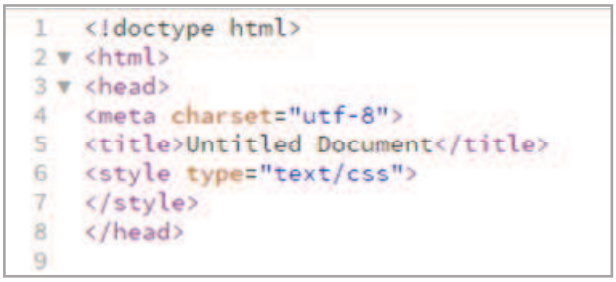
a. Using *Design* view, type the lines "Style 1", "Style 2" and "Style 3", pressing the *Enter* key to create new paragraphs after each line.

Note: *When you want to start a new line immediately below the previous one, hold down the 'Shift' key and press 'Enter'. This is called a Line Break. Try this now and then type the text "Same paragraph".*

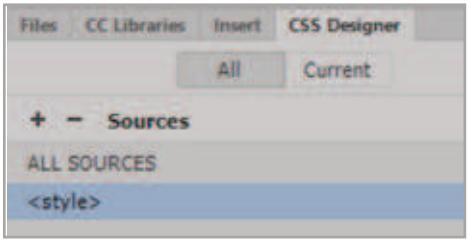
*Look at the code on the right. Notice that the line break has the code '
'. Any styles set on the paragraph will affect both lines of text.*

b. We are going to create a new style sheet in the head section. Open the *CSS Designer* tab in the panel on the right, then click the '**+**' symbol alongside *Sources* and select 'Define in Page'. Notice that some *style tags* are added to the head.

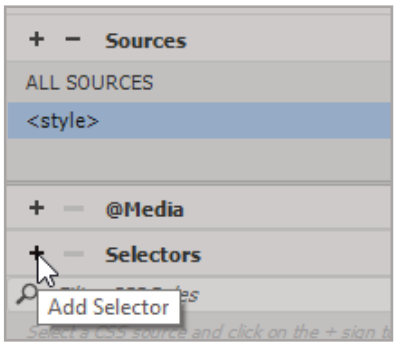




c. Next, we are going to define three different text styles for our three paragraphs. In the *CSS Designer*, make sure that '<style>' is selected in the *Sources* section. This is the empty style sheet we have just created.



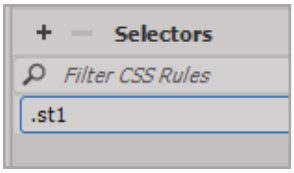
d. The second section of the *CSS Designer* is not needed for this task, so, if necessary, click on '@Media' to collapse it and make things a little tidier.



e. The styles used will be placed in sections of code called *class selectors*. Once you have created a *class selector*, you can apply it to any text in your webpage. Click on the '**+**' button in the *Selectors* section to add a selector (see right).

f. Type '**.st1**' in the box as a name for our first *class selector* and press the *Enter* key. Look at the code and notice that an empty selector has been created in the style sheet. It has the form:

```
.st1 {
}
```



Note: *In CSS, there are two types of selector. One is an 'ID selector' which is used to apply styles to a single object with the given ID. The other is a 'class selector' which can be applied to any number of objects. ID selectors have names preceded by a hash (#). Class selectors have names preceded by a dot (.).*

```
<title>Test2</title>
<style type="text/css">
.st1 {
}
</style>
```

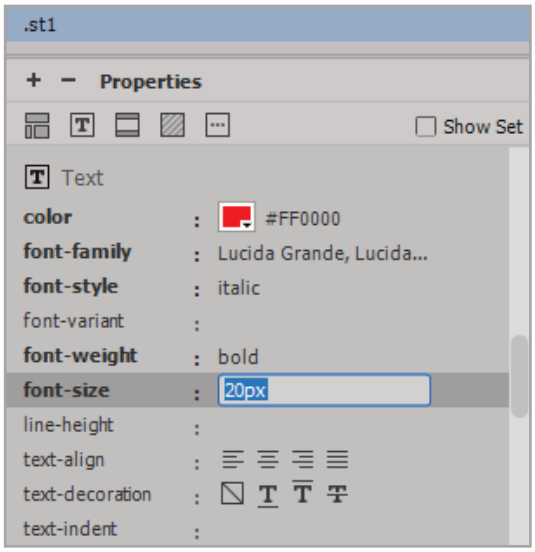
g. Back in the CSS Designer, click on the selector '.st1' (it should be highlighted).

h. The *Properties* box below this should be empty because we haven't set any styles for this selector yet. To see a list of all the styles that we can choose from, uncheck the 'Show Set' box.

i. Scroll down the list of styles until you find the *Text* section (or click on the *Text* button along the top of the *Properties* box). You can adjust the height of each panel to make things easier to see.

j. Select a colour for the style. Once you have clicked on a colour, click outside the colours box to close it.

Note: *Don't stress if the Properties box seems to turn blank; you have most likely selected something else by accident. Simply click on the '<style>' source again, then the '.st1' selector and the 'Text' button.*



k. Choose a font family, style, weight and size from the other options. Have a look at the style sheet. It should now look something like the one below but probably with different values.

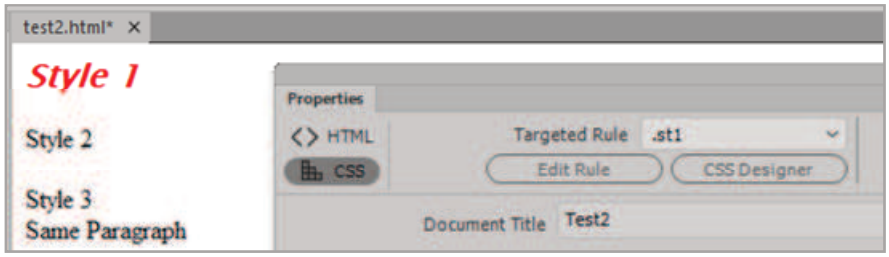
```
<style type="text/css">
.st1 {
    color: #FF0000;
    font-family: "Lucida Grande", "Lucida Sans Unicode", "Lucida S
    font-style: italic;
    font-weight: bold;
    font-size: 20px;
}
</style>
```


Task 2 – Using the Class Selector

Now our first style is in place, we can use it anywhere in our page.

In the *Design* window, select the text "Style 1" then open the *Property Inspector* ('Window/Properties').

The new 'st1' selector should be in the *Targeted Rule* list. Select it to apply the style.



Look back at the code window. You should now see where our class attribute has been added.

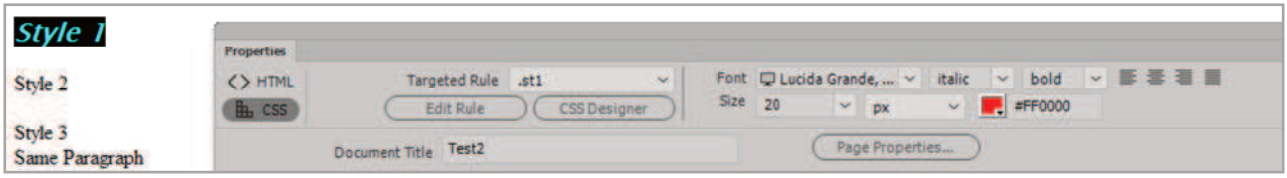
```
<p class="st1">Style 1</p>
```

```
<p class="st1">Style 1</p>
<p>Style 2</p>
<p>Style 3<br>
  Same Paragraph
</p>
```

Task 3 – Editing the Class Selector

If you change your mind about the styles chosen for a selector, you may edit it in any of the following ways:

- Edit the code in the page head section.
- Change the settings in the CSS Designer properties panel on the right (where you set up the selector).
- Change the settings in the *Property Inspector* whenever the selector is showing as the *Targeted Rule*.

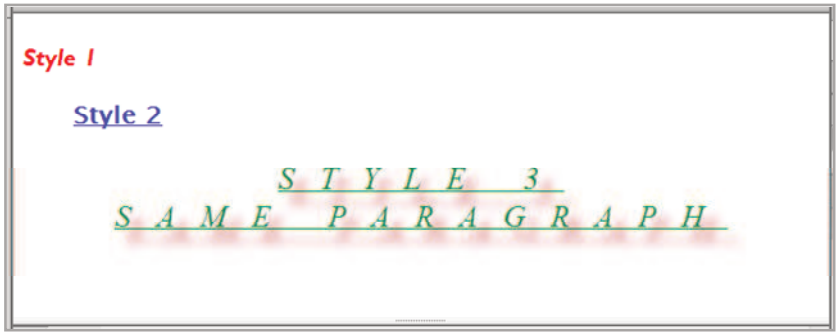


Task 4 – More Styles

Create two more class selectors for your other two paragraphs of text. Name these '.st2' and '.st3'.

In the second of these classes, try out some of the other styles, including the *text-indent*.

For the third paragraph, see if you can reproduce the text shown on the right. It uses green small caps, is aligned centre and has a blurred, red shadow displaced 10px both horizontally and vertically.

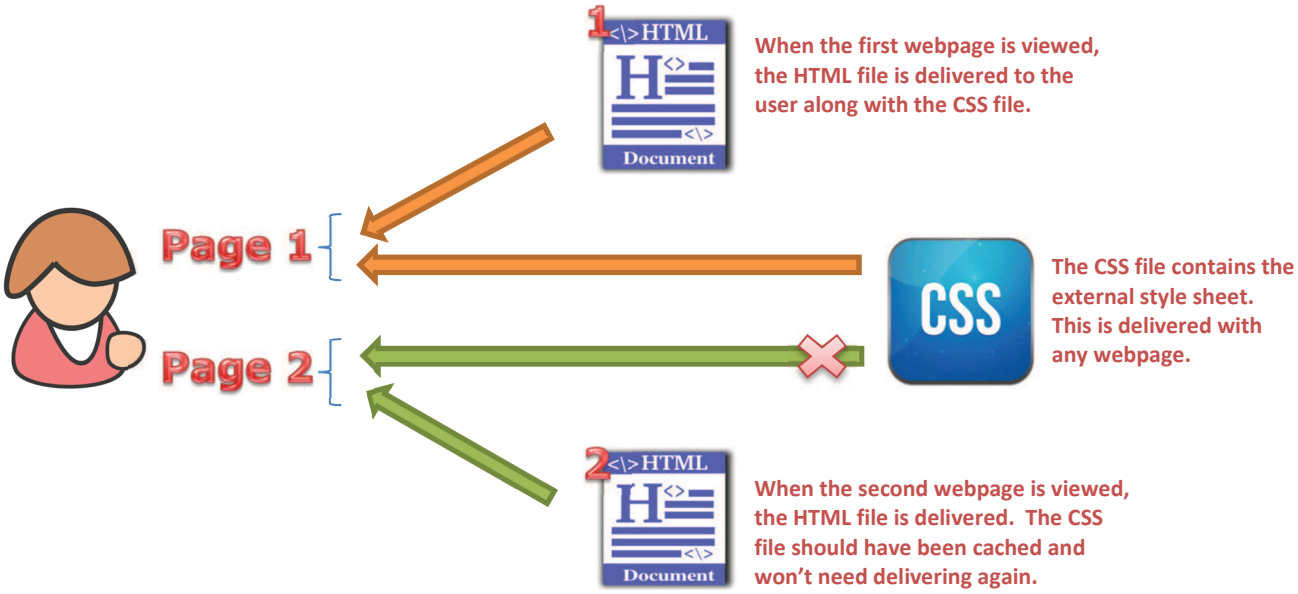


Note: *You will need to look at 'Live' view or use your browser to see some of these effects.*

Look at your style sheet and find out how all these *properties* and *values* were encoded.

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.



Caching

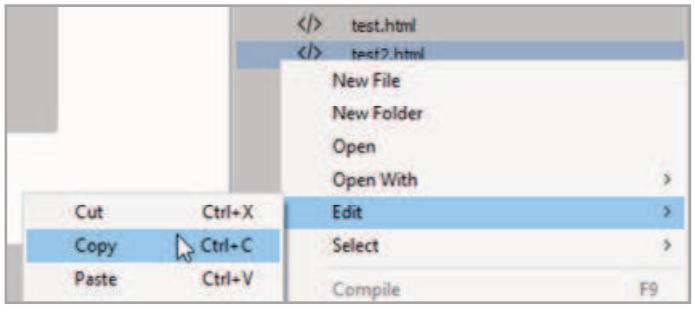
Another benefit of this method is that the CSS file doesn't actually have to be delivered through the internet each time a page loads. The user's computer will store a copy in a local folder called a cache. When another page loads using the same CSS file, the user's computer will check to see if the 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.

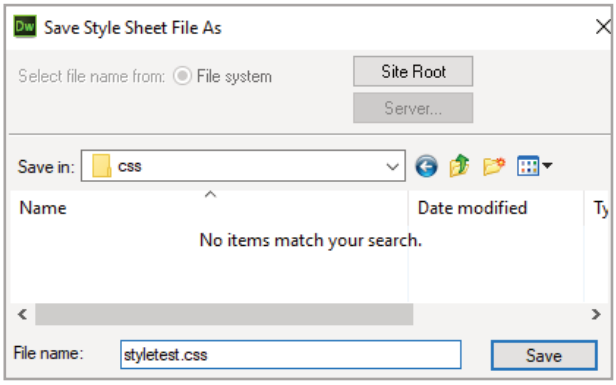
If using the *Files* panel, right click on the file name and use 'Edit / Copy' and 'Edit / Paste' to duplicate the file.

Name the file 'test3.html'.



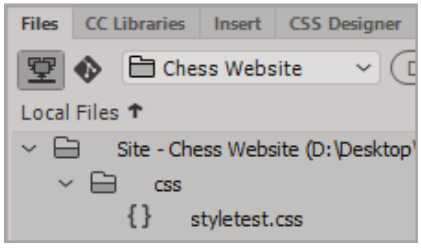
b. We are going to create a style sheet in a separate file and the move the class selectors from the test page into it.

Open your new test page then click on the '**+**' symbol in the *CSS Designer* tab. Select 'Create A New CSS File'.



c. Click the *Browse* button in the window that opens then locate and enter the 'css' folder created earlier (or create one now if you missed that step).

d. Enter the filename '**styletest.css**' and click *Save,* then *OK*. Use the *Files* panel to check that the new file has been created (see right).

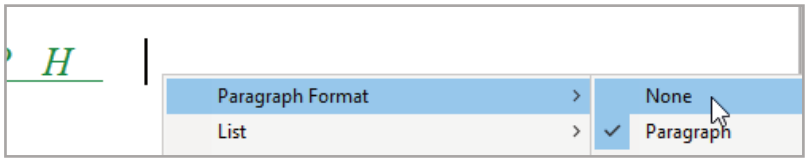


e. Look at your webpage code. You should see that a line has been added at the base of the head section linking to your new CSS file. This is telling the browser that it also needs to download the file mentioned.

```
</style>
<link href="css/styletest.css" rel="stylesheet" type="text/css">
</head>
```

Task 2 – Adding a Class Selector to the CSS File

a. Place your cursor at the end of the last text in the Design window, press *Enter* then select 'right-click / Paragraph Format / None' to cancel the styling used in the previous paragraph.

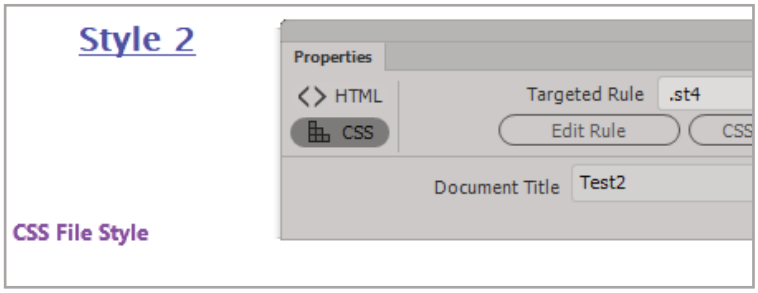


b. Start a new paragraph and type the text "CSS File Style".

c. Open the CSS Designer in the panel on the right and select your *styletest.css* file in the *Sources* section.

d. Create a new selector named '**.st4**' and set some style properties for it. Notice that your new selector doesn't appear in the page code. It is being added to the *styletest.css* file.

e. Select the text "CSS File Style" in the Design window, open the *Property Inspector* ('Window / Properties') and select 'st4' in the *Targeted Rule* box to apply the style.

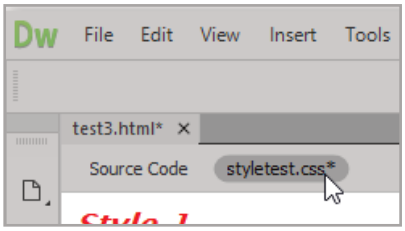


f. Save the test page.

Task 3 – Editing the CSS File

There are two methods of opening and directly editing your CSS file. One is to use the *Files* panel to open it fully; the easier method is to click on 'styletest.css' tab in the top-left of the screen. This means you can still view your design whilst editing your CSS file.

Open your stylesheet and save it. (This step is sometimes necessary when viewing the page in your browser before Dreamweaver has saved the file itself).



At the moment, your CSS file should consist of nothing much more that your class selector, 'st4'. We will now cut all the other styles from our internal style sheet and paste them into our CSS file.

```
5  <title>Test3</title>
6  <style type="text/css">
7  .st1 {
8      color: #FF0000;
9      font-family: "Lucida Grande", "Lucida
```

```
31     letter-spacing: 20px;
32 }
33
34 </style>
```

a. Back in your 3rd test page (click the *Source Code* tab if you used the second method above) select all the code between, but **not** including the lines below.

<style type="text/css">

..select everything…

</style>

b. Click 'Edit / Cut'.

c. Open your CSS file and place the cursor in a new line at the end of the code. Click 'Edit / Paste'. You may cut and paste further to reorder the selectors if you like.

d. Return to your test page and delete the empty style tags. Save your work

Notice that, as in the picture on the right, the styles have all been removed from the code but the text on the page is still formatted as it was before. We have created and used an external style sheet.

Doesn't our webpage code look much tidier?

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Test3</title>
6  <link href="css/styletest.css" rel="styleshe
7  </head>
8
9  <body>
10 <p class="st1">Style 1</p>
11 <p class="st2">Style 2</p>
12 <p class="st3">Style 3<br>
13   Same Paragraph
14 </p>
15 <p class="st4">CSS File Style</p>
16 </body>
17
```

Task 4 – Using the External Style Sheet in Other Pages

Now that our CSS file is in place, we can use it in other webpages.

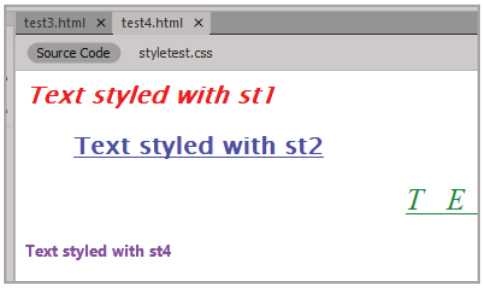
a. First, initiate a new page by clicking 'File / New'. In the window that opens, click on the 'Attach style sheet' icon in the bottom-right (as shown).

b. Browse for and select your *styletest.css* file, then create your page.

c. When the page is first created, the link in your code to the CSS file will be long. Save the page as 'test4.html' to change the link to a shortened version like the ones used before.

What is actually happening here?
Before saving, the new page doesn't know where it will be located. It therefore finds the styletest.css file using a full (absolute) link. Once you've saved your page to the website folder, it can just use the short link which is relative to the page ('css/styletest.css').

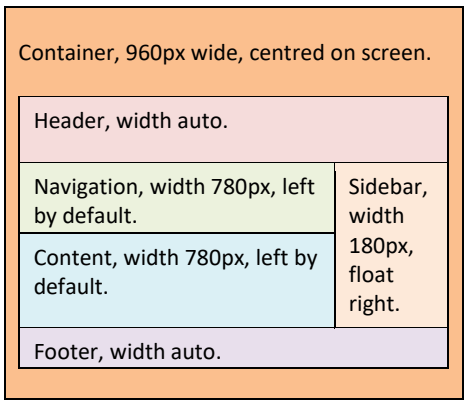
d. Test each of your styles with some text. Remember that the styles are selected in the *Property Inspector*.



Divs are used to lay out the content on a webpage. A *div* is a container which can be fixed in the desired place on a page. A div can have a certain height and width, a background colour and can be positioned using the alignment tools and margins.

The measurement for the size of a div can be fixed as either a percentage of whatever is containing it (e.g. the screen size, another div etc.) or as a set number of pixels. A div set to 100% of the screen width will appear very differently on a large and small screen.

The diagram on the right shows a page layout formed using six different divs. Layouts like this are very common.



Container Div

This has been used to control all the other divs. Without the container, the other divs would shift around when a larger screen is used (those aligned left would shoot off to the left etc). Our container is fixed at 960 pixels wide, centred on the screen. Its height will automatically stretch to fit the contents.

Header Div

The header div will contain an image that is the full 960px wide. To ensure that the actual div is also 960px, we will set the width to 'auto' (by default, this means 100% of any container it is enclosed in). The height of this div will stretch to fit the image.

Navigation Div

The 'nav' div will contain a set of links to the other pages in our website. This div will be fixed with a total width of 780px, but some of this will be padding (see next section). It will automatically align to the left of the container if not told otherwise.

Content Div

The content div will contain most of the page specific content. This may be information, further images, tables of data etc. This div will also be fixed with a total width of 780px (including padding) and will align to the left of the container.

Sidebar Div

The sidebar will contain some other information or links. This div will be fixed with a total width of 180px including padding. It will float to the right of the container and be of a height that fits its contents. The *float* property will allow other elements to flow around this div, so it will actually be placed in the code before the nav div. The nav div will then flow directly on from the header.

Footer Div

The footer may contain standard information that appears on every page, such as contact information or links to general info pages ('About Us' etc).

Margins, Padding and Borders

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 into account (see our Dreamweaver Basics resources for a full exploration of this).

A ***margin*** simply shifts the div in one direction. You do not need to adjust the size unless the div will overlap another.

Padding adds extra size to the div. If you add 10px of padding to the left and right of a div but want to keep the overall width the same, then you need to reduce the div size by 10 + 10 = 20px.

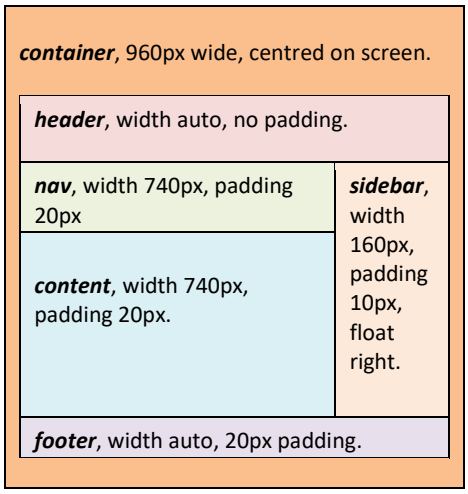
Borders are like padding in that you must take account of their size.

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

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.

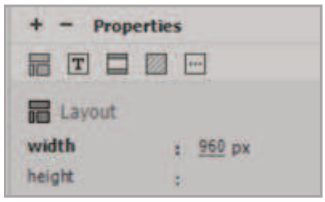
a. Open your *index.html* page (the homepage) and delete any text that you entered previously.

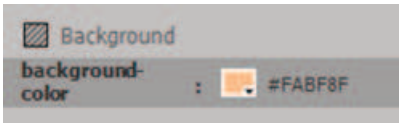
b. We will place all the div styles in a CSS file so that we can use them on each page. Using the CSS Designer, create a new source called '**style.css**'. Place this in your *css* folder.

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 our container is an **ID** rather than a **class**. 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.

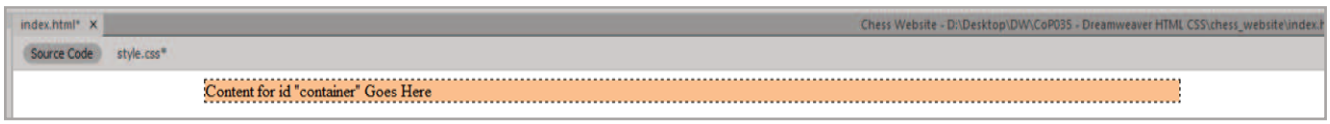




Have a quick look at the CSS file. It should now look like the one on the right (with the properties in any order and probably a different colour in hexadecimal).

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

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.



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 of 960 pixels, whatever size screen we are using. It will also keep the page centred horizontally on a large screen.

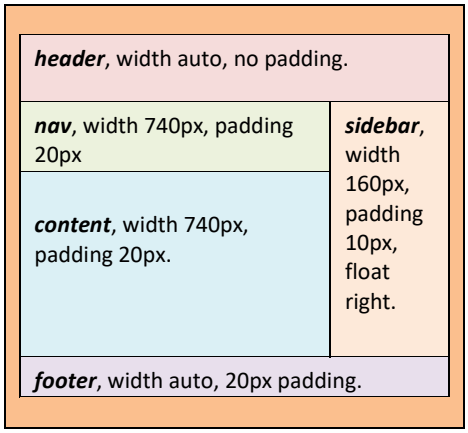
It is now time to build the other divs we need to set out our content. Bear in mind the following points:

- If we do not set a width for a div, it will automatically have 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 these behaviours can actually 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.

Selector	Width	Padding	Total width	Float	Insert
#header	auto		960px		At Insertion Point
#nav	740px	20px all	780px		After Tag - header
#content	740px	20px all	780px		After Tag - nav
#sidebar	160px	10px all	180px	Right	After Tag - header
#footer	auto	20px all	960px		After Tag – content

Task 1 – Creating the Other Divs

We'll use a similar method to create the rest of our divs. Refer to the diagram on the right and the table above for further clarification.



a. Create your '#header' selector. No properties have to be set except a temporary background colour – it will automatically fill the width of the container.

b. Delete the text '*Content for id "container" Goes Here*' and leave the cursor in the (now very short) container. Click 'Insert / Div' and select the ID *header*. Leave the *Insert* option as 'At Insertion Point' and click *OK*.

Note: *The 'container' ID is no longer in the list because we have already used it once.*

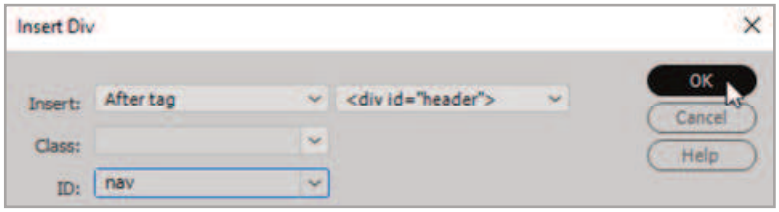
c. Look at the HTML code. Notice how the *header* div is inside the *container* div.

```
<div id="container">
  <div id="header">Content for  id "header" Goes Here</div>
</div>
```

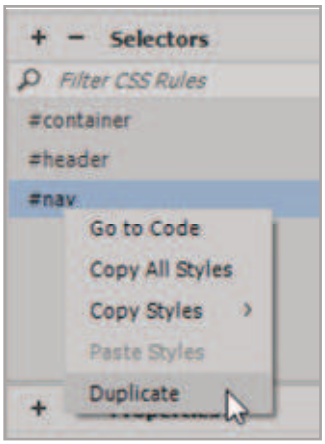

Task 1 – Creating the Other Divs (cont)

d. Now create your '#nav' selector. Use a width of 740px and add 20px padding on all sides making a total width of 780px. Choose a different background colour.

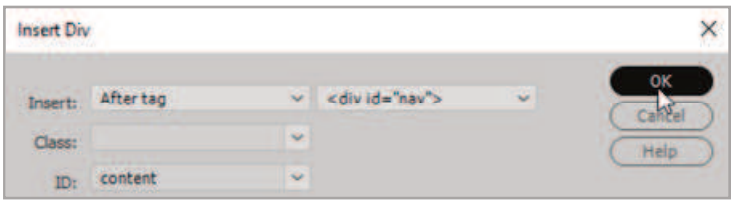
e. Place your cursor back in the header div on the Design page but leave the text in place. Click 'Insert / Div' and use the settings shown on the right. This will insert the *nav* div after (or below) the *header* div. You should notice some of the container div showing through on the right.



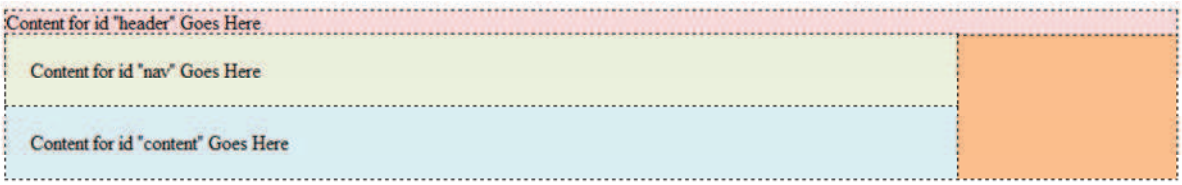
f. The *content* div is very similar to the *nav* div. The quickest way to reproduce the style is to right-click on the '#nav' selector and choose 'Duplicate' from the context menu. Give the new selector the name '#content' and choose a different background colour.



g. Insert the *content* div after the *nav* div as shown.



h. Your homepage should now look similar to the one below, although your colours will no doubt vary. Notice that the only colour visible from your container div is in the lower-right corner. This is where our sidebar will go.



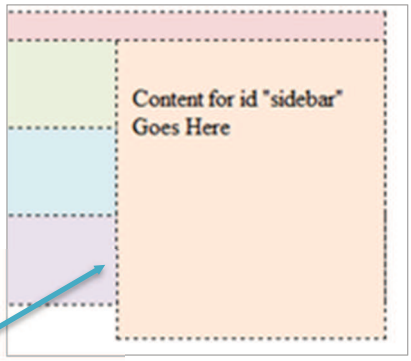
i. Create your '#sidebar' selector. Use a width of 160px and add 10px padding on all sides, making a full width of 180px. Float your div to the right as shown and select a different background colour.

j. Insert the sidebar div **after the header div** by selecting 'After tag' and ' . This will place the div below the header.

k. Finally, add the '#footer' selector and footer. Leave the width blank so that it fills the width of the container but add a padding of 20px on all edges and select one more background colour. Place the *footer* after the *content* div.

Task 2 – Stopping the Overlap

Place your cursor in the new sidebar div and press the *Enter* key several times. You should find that the div extends down as expected, but then overlaps the footer as shown on the right. We can tell the footer not to allow this, but to shift down as required.



Return to your footer selector in the CSS Designer and find the *clear* property. Select the value 'Both'. This basically means that any divs above the footer (which stretch downwards as content is added) will now force the footer further down the page.

The sidebar initially overlaps the footer. By setting the clear property to 'both', the footer will be pushed down the page.

Task 3 - CSS Shorthand

CSS shortcuts save time and result in less code. This means faster transfers and less data to move. An example of a css shortcut we used earlier allowed us to write a font style, weight, size and family all in one declaration, as below:

```
<p style="font: italic bold 24px Arial">Some Text</p>
```

If you look at your style sheet, you should see the following code in your 'nav' selector:

```
padding-right: 20px;
padding-left: 20px;
padding-top: 20px;
padding-bottom: 20px;
```

We can use shorthand to condense these properties into the following line (think about the hands of a clock, starting at twelve):

```
padding: top right bottom left;
```

So, our padding declaration could be shortened to:

```
padding: 20px 20px 20px 20px;
```

Or, as all these values are the same, we could actually go one step further and use the following:

```
padding: 20px;
```

If in another situation the padding top and bottom should be 50px, but the padding left and right 25px, we could write:

```
padding: 50px 25px;
```

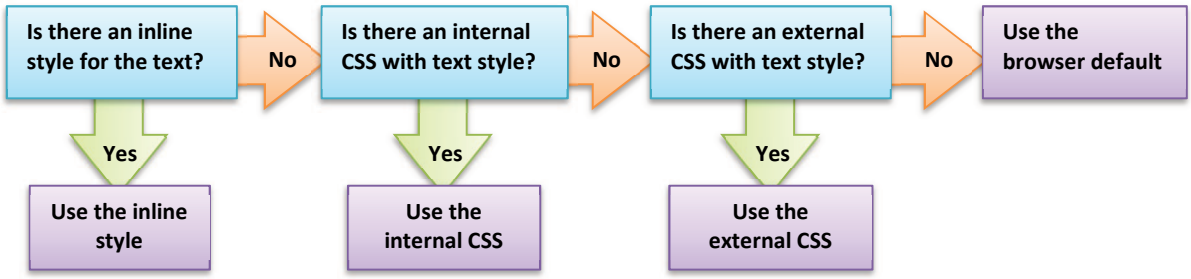
There are lots of these shortcuts. You may find more to use as you develop your webpages, but you should also be able to look up the meaning when you see shorthand used elsewhere. Look at your CSS file and change some of the longhand to shorthand. Note that if you set 'auto' for the top and bottom margins, then the div will align to the top.

Task 4 – Explaining CSS

Your CSS code might now look something like that below. Explain the effect each line of code indicated has.

```
#container {
  width: 960px;
  margin: auto;
  background-color: #FABF8F;
}
#header {
    background-color: #F4D9DA;
}
#nav {
    width: 740px;
    padding: 20px;
    background-color: #EAF1DD;
}
#content {
    width: 740px;
    padding: 20px;
    background-color: #D8EDF2;
}
#sidebar {
    width: 160px;
    padding: 10px;
    float: right;
    background-color: #FFEADA;
}
#footer {
    padding: 20px;
    background-color: #E6E0EC;
    clear: both;
}
```


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.



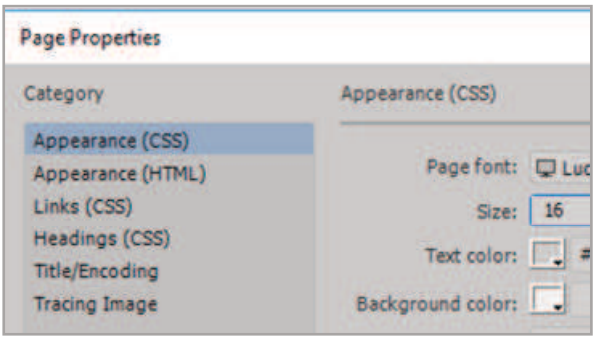
Note: *The actual process takes place in reverse. The browser will at first adopt the most general rule and then replace it with any more specific rules it finds (e.g. an inline style will replace an internal style sheet). In this way, styles will cascade down to the most specific available, replacing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

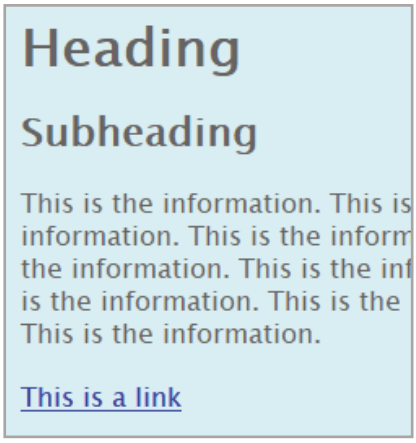
Cut these declaration blocks out and paste them into your CSS file (at the top, under the '@charset "utf-8";' line). They will now be applied to every page that uses your external style sheet. You can delete the leftover '<style type="text/css">' and '</style> tags from your webpage.

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

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

Task 3 - Headings and Link Text

You can do a similar thing to set the default heading and hyperlink styles for your website.



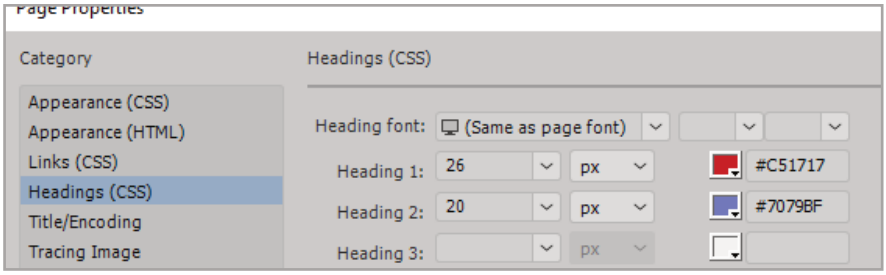
a. Start by typing some text for a heading, a subheading (if you want to use one) and some normal text, all in the 'Content' div. You may simply add some temporary text if you don't know what you'd like to add yet.

b. Select the heading text with your mouse and click 'Edit / Paragraph Format / Heading 1'. This should now pick up the default 1st heading size. Do the same for the subheading but select 'Heading 2' from the list.

c. Add some more text to make into a link, then select it and click 'Insert / Hyperlink'. Just place a '#' symbol in the *Link* box and click *OK* (this will create the link, but send the user nowhere).

Click 'File / Page Properties' and select the *Headings (CSS)* tab.

d. Select a size and colour for the Headings 1 and 2. You may leave the font boxes empty if you are happy for the browser to use the body declarations made earlier.



e. You can also set any default styles you want to use for the links in the 'Links (CSS)' tab. We will actually be formatting the main set of links in the 'nav' div separately, so the styles set in the *Page Properties* will generally be utilised by normal blocks of text.

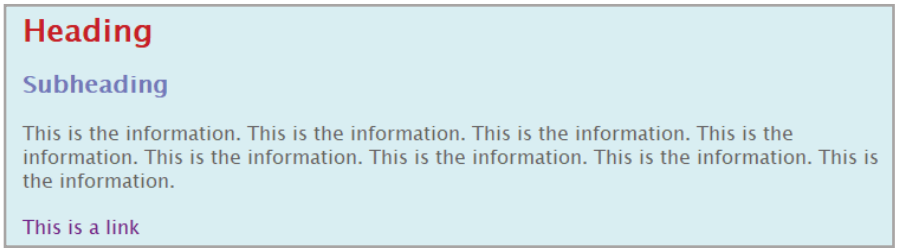
f. Have a look back at your webpage code. You should again see a set of styles declared in an internal CSS in the *head* section.

h1	*Heading 1*
h2	*Heading 2*
a:link	*The initial hyperlink style*
a:hover	*The hyperlink style when the mouse is hovering over it*
a:visited	*The hyperlink style once the page has been visited*
a:active	*The hyperlink style during the click*
text-decoration	*Underline, over line or a line through the text.*

Note: *The declaration blocks can be in any order in the style.css file. It is best to organise them so that you can easily locate any that you need later.*

```
h1 {
    font-size: 26px;
    color: #C51717;
}
h2 {
    font-size: 20px;
    color: #7079BF;
}
a:link {
    color: #710B92;
    text-decoration: none;
}
a:visited {
    color: #2030B1;
    text-decoration: none;
}
a:hover {
    color: #5A2D28;
    text-decoration: underline;
}
a:active {
    text-decoration: none;
}
```

g. As before, cut all these styles from your webpage and paste them into your CSS file so that they apply to all pages.



K. Adding a List of Links

We are going to create and style a list of hyperlinks, linking to the other pages in our website. Vertical lists are defined in the HTML but can then be set as horizontal lists in the CSS. In our case, the list will be horizontally laid out across the *nav* div.

There are a number of HTML tags that help with the layout of lists.

  creates and closes a numbered list (ordered list)

  creates and closes a bulleted list (unordered list)

  used before and after each item in the list

There is also another type of list called a *Definition* (or *Description*) list which is not covered here.

Task 1 - Creating a List in HTML

It's often easier to place your list items in paragraphs first and then format them as a list afterwards.

a. Place your cursor in the *nav* div in the *Design* section of your screen and delete the default text. Create a list of words you can use as links, pressing the *Enter* key between each one. Check the HTML is like that shown on the right.

```
<div id="nav">
  <p>Homepage</p>
  <p>Data</p>
  <p>Choice</p>
  <p>Test</p>
</div>
```

b. Select the entire list of words then click 'Edit / List / Unordered list'. You should now get a list formatted in HTML with a <ul> tag opening the unordered list and a tag closing it.

- Homepage
- Data
- Choice
- Test

```
<ul>
  <li>Homepage</li>
  <li>Data</li>
  <li>Choice</li>
  <li>Test</li>
</ul>
```

Task 2 - Adding Hyperlinks

Hyperlinks are the things that tie the web together. Clicking on a hyperlink usually takes you to a new place on the internet. This location may be in the same website (possibly a different place on the same page) or could be a completely different location.

We will start by creating hyperlinks from the homepage to each page in the website. Our first link will be from the homepage to the homepage. This might seem strange, but it is normal to have the same set of links on each page. This way, the user isn't confused by things jumping around when they switch pages.

a. Select the word 'Homepage' and click 'Insert / Hyperlink'. Click on the small *Browse* icon and select the 'index.html' file. Add the title 'Homepage'. The title will become the text in the screen tip that appears when you place your mouse over the link. It may also be read out loud by screen readers designed for the vision impaired.

b. Repeat this process, creating links that point to your other pages.

Text	Link to	Title
Data	data.html	Data Page
*Choice**	*choice.html**	*Choice Page**
Test	test.html	Test Page

**Use your own file and link*

If you make a mistake, you can undo your change or select the link then click 'Edit / Link / Change Link' or 'Remove Link'.

Testing Your Hyperlinks

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

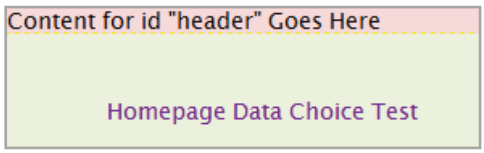
a. Save the page then right-click on *index.html* in the *Files* panel on the right. Select 'Open in Browser' and choose one of the browsers on your system (Internet Explorer, Safari, Chrome etc.).

b. Clicking on the homepage link should go nowhere, but clicking on the *Data* link should take you to your data page. Use the *Back* button in your browser to return to your homepage and check your other links. Return to Dreamweaver when you have finished checking.

Task 3 - Styling Your Hyperlinks

Now that the links are in place, we will add some CSS styles to make them look prettier.

a. Open your 'style.css' file and add the following selector to the end:




```
#nav li {
    display: inline;
}
```

This should have the effect of making the list of links horizontal.

Note: *The 'li' code tells the browser to add the style to list elements; the '#nav' will cause the style to only be applied to lists in the 'nav' div. We could leave out the '#nav' but then the style would be applied to all lists on the page.*

b. Add some further code to this declaration block, under the line of code *display: inline;*.

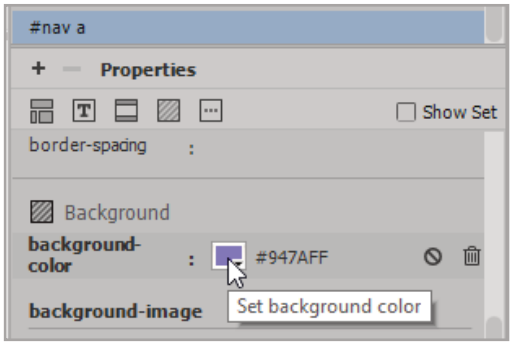
```
font-size: 26px;
list-style-type: none;
```

You may make any other font choices here or use the defaults from your CSS.
This tells the browser not to use bullets. Most don't anyway.

c. We will now add another declaration block, this time targeting the anchor tags (i.e. the hyperlinks) in the *nav* div.

```
#nav a {
    background-color: #947AFF;
}
```

d. To select your own background colour, open the CSS Designer in the panel on the right, select your 'style.css' file and the '#nav a' block, then use the *Properties* section to open the selector for the background colour.



e. White text often looks good on a coloured background. The full hex value for the colour white is #FFFFFF, or we can use the shorthand #FFF. We also need some padding around the text. Add the following lines to your *#nav a* declaration block.

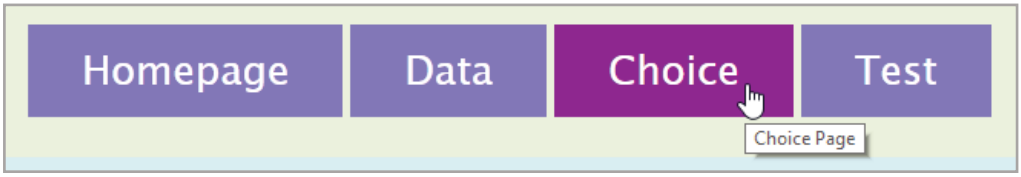
```
color: #FFF;
padding: 10px 35px;
text-decoration: none;
```

Note: *You may have already set this when choosing the page properties.*

f. Finally, add another declaration block with some rollover effects that will be seen when the mouse hovers over the links. Choose your own values. You will need to look at your page in *Live view* or your browser to see the effects. Save the page and your *style.css* file first if using your browser.

```
#nav a:hover {
    background-color: #92179A;
}
```

Note: *You may also set a different font colour or text decoration.*



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.

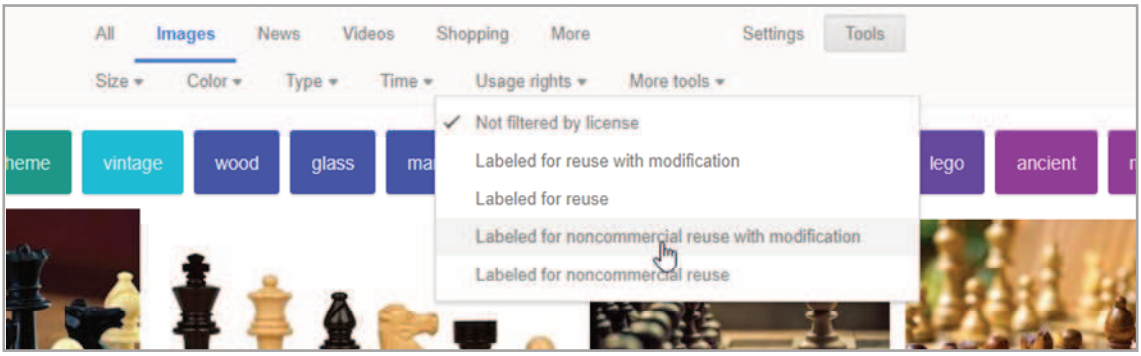
Note: *The images do not become part of the webpage file; they remain a separate file which is 'collected' by the page as it is viewed. You will need to save a copy of the images to your computer before adding them to your pages.*

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.



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.

d. The thumbnails in Google are low quality copies. They are only useful for small thumbnail images in your webpage (expanding an image will expose the poor quality). If you want the full-sized version, click on an image and wait for the larger copy to appear on Google's search page.

e. If you decide to use an image in your website, save it to your images folder (on a PC, right-click on the image and select 'Save picture as…' or 'Save image as…'). Make sure the file names you use do not contain spaces. The web doesn't like spaces, so it's best to avoid them. The use of underscores is advised if you'd prefer to separate words.

Task 2 - Cropping and Resizing an Image in Photoshop (or elsewhere)

a. We are going to crop our image so that it is the exact width of our 'header' div (i.e. 960px). On any screen 960px or wider, the image will be displayed actual size. This is our desired outcome as the image will appear at its highest quality. On smaller screens, the image will be shrunk to fit. We want to avoid enlargement as the quality of the image will be reduced.

b. We have chosen Photoshop to edit our image. Other image editors generally work in the same way and can be used instead. Avoid the very basic graphics editors such as Windows Paint as these applications resize images in a way that results in a significant loss of quality (it is better to use the Dreamweaver crop tool).

> ***Cropping in Dreamweaver***
>
> *If you don't have an image editor on your system, or you just want to save a bit of time, you may use the crop tool built into Dreamweaver. Insert the full image, right-click on it and select 'Properties'. Click on the Crop tool, select a section of your image (preferably 960px wide) then click on the Crop tool again. The results aren't always what you might expect.*

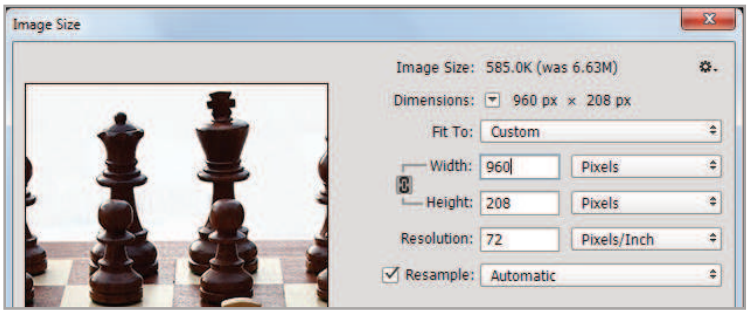
c. Open your image in Photoshop (presuming this is your chosen method). Click 'Window / Info' to display the info panel. Click the icon in the top-right of this panel and select *Panel Options*. Change the *Ruler Units* to pixels.

d. Select the *Crop* tool in the bar on the left. Draw a rectangle with your mouse around the part of the image that you would like to use. It should be at least 960px wide (any more is fine). The height isn't important but we only want a banner image so it should preferably be less than one third of the width. The pixel dimensions are shown in the Info panel (3261 x 728 in the example below).

e. Apply the crop by clicking the tick/check icon in the top centre ('Commit current crop operation').

f. Click 'Image / Image Size' in the main menu and in the *Image Size* window, make sure that there is a link between the width and height measurements. This forces the image to change both dimensions proportionally rather than being stretched or squashed. If the link isn't present then click the link icon.

g. Set the *Width* to 960 pixels. Providing the *aspect ratio* link is in place, the height should change automatically. In our example, the final image will be 960 x 208 pixels. Your width should be the same but the height will probably be different.



h. Set the *Resample* to 'Automatic' and click *OK*.

i. Click 'File / Export As' and select 'JPG' in the *Format* box.

j. Use the '+' icon to set the magnification to 500%, then reduce the quality setting as much as you can without significantly affecting the image quality. The aim is to get the smallest file size possible without losing detail (the file size is shown in the top-left).

k. Click Export All and save the exported image to your images folder. Remember the name you give to this exported image. You may close Photoshop without saving the changes to your original image.

Task 3 - Adding an Image to your Header

a. Select and delete the text in the *Header* div, then click 'Insert / Image'.

b. Find the edited image that you saved to your computer, select it and click *OK*.

Note: *If you insert an image from a location outside your website folder, a window will open asking if you'd like to make a copy to your root folder. Click 'Yes', find the images folder inside your root folder and open it. Click 'Save'. The image will be saved to your website and placed on your page.*

c. Save and view your page, either in your browser or using *Live view*.

Image HTML and CSS

Look at your webpage code and find the HTML used to insert the image. Ours is shown below.

```
<body>
<div id="container">
  <div id="header"><img src="images/chess.jpg" width="960" height="208" alt=""/></div>
```

Notice that the width and height settings have been fixed to the size of the image, the width being 960px as intended. Although it isn't essential to include the width and height here, adding these measurements to the HTML means that the browser knows instantly how big the image is and can leave a space for it whilst the page is loading. Without these measurements, the webpage might appear to jump around until all the images have been downloaded.

Alt Text and Image Titles

Notice that Dreamweaver has put in place some empty 'alt' tags. Open the *Property Inspector* for the image. You will find an *Alt* box and a *Title* box. Although behaviour can vary, most of the time the *Title* will be displayed as a screen tip (as shown on the right). *Alt text*, on the other hand, will be read aloud by the screen readers used by those with impaired vision. It will also be shown when images are disabled by the user and can be picked up by search engines. We suggest that you add titles and alt text to all your images.


```
width="960" height="208" alt="Chess homepage (Alt)"  title="Chess homepage (Title)"
```

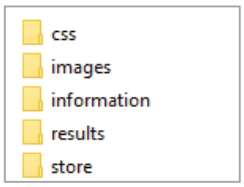
Note: *We have only added the words in brackets to demonstrate which tag is displayed as a screen tip in the image above.*

Image HTML and CSS (cont)

Images can be inserted to the HTML using the following tags:

```
<img src="images/photo1.jpg">
```
- links from a page in the root folder to an image in a folder called 'images'

```
<img src="../images/photo1.jpg">
```
- links from a page in a folder to an image in a folder called 'images'



Note: *The second of these can be used if you have so many pages that you place them in separate folders, as on the right.*

As we've seen, you can then add width and height attributes, along with some *alt text* and a title.

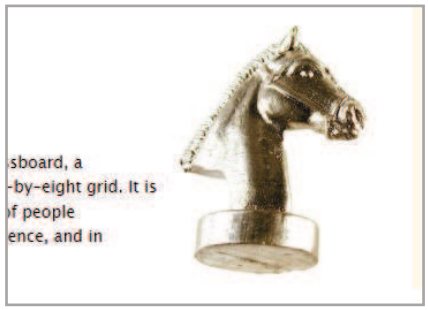
```
<img src="photo1.jpg" width="100" height="60" alt="My Photo" title="My Photo">
```

Other styles should be placed in the CSS. In the example below, we have wrapped the image in a div with the class 'img'. We can then target this class in our CSS with properties such as margin, padding, border and float.

```
.img {
  margin: 5px;
  border: 1px solid #0000FF;
  float: right;
}

<div class="img">
    <img src="photo1.jpg" width="100" height="60" alt="My Photo" title="My Photo">
</div>
```

Task 1 - Inserting and Positioning an Image



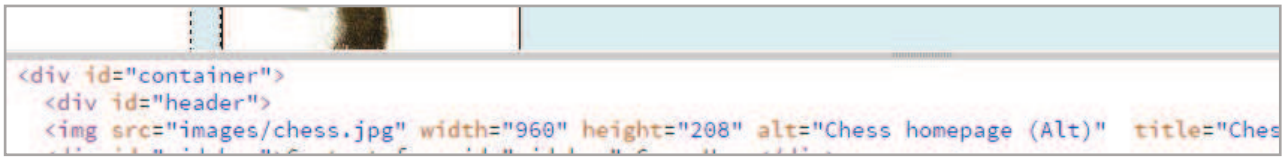
a. Find another image on the internet that you would like to use. It can be placed pretty much anywhere on your page if you can work out how.

b. Save the image to your images folder. Don't worry about the size for now; we will fix up resizing issues later.

c. Place the cursor at the start of the *content* div and press the *Enter* key to start a new paragraph. Back at the top of the div, right-click and select 'Paragraph Format / None'.

d. Insert your image and with it still selected, click 'Insert / Div'. Wrap the new div around the image with an ID of your choice. Click *OK*.

e. You may tidy up your HTML code if necessary. Add alt text and a title.



f. Add a declaration block to your CSS. In our case, we want the image to appear to the right of the text.

```
#knight {
    float:right;
}
```

g. Resize your image using the handles in *Design view*. Once you have the size you are after, have a look back at the HTML and see how big your final image will be.

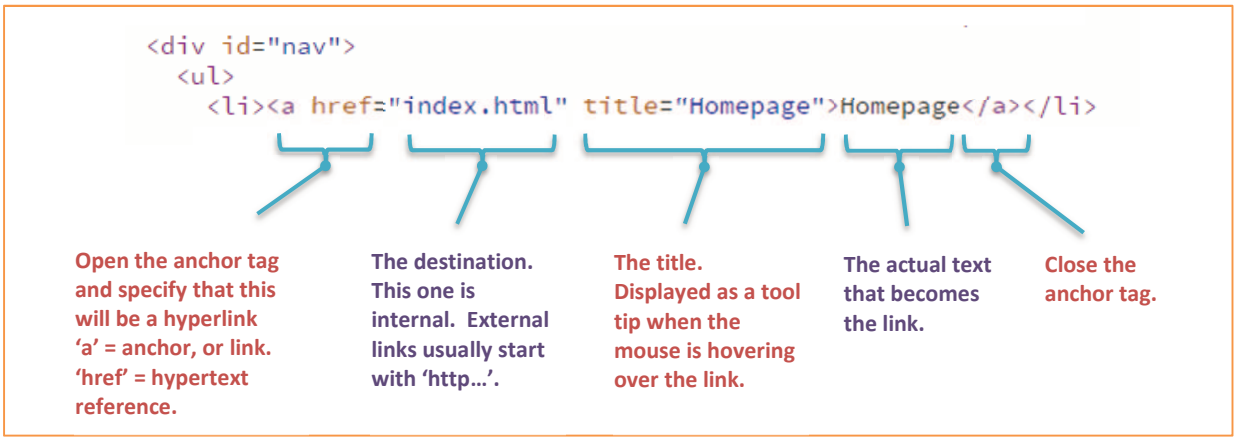
h. Use Photoshop (or whichever application you are using) to resize the image to the values set in your HTML. Replace your original image with the resized copy. Check everything in your browser.

Note: *It is possible to fix an image to an exact position on a page. This is only advised if other elements will not be shifted around by the image.*

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:



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:

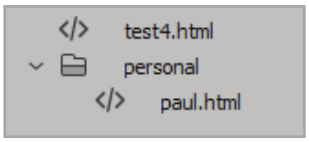
```
<a href="personal/paul.html"></a>
```
- links to the page 'paul.html' in a folder named 'personal'

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.*

Task 1 - Creating Internal Hyperlinks



a. Right-click on any page in the *Files* panel and select 'New Folder'. Name this folder "personal". Right-click on your new folder and create a 'New File'. Name this "paul.html".

b. Create a new test page named 'Test5' and type the word "Paul Page" in a new paragraph. Using code view, edit the HTML so that this text becomes a link to the new page. You will need an anchor tag and the destination 'personal/paul.html'.

c. Similarly, in your new file, type the words "Back to Test Page" in the design area and edit this link so that it takes you back to the test page. Test the links in a browser.

External (or Absolute) Hyperlinks

External hyperlinks take the web browser to other locations on the World Wide Web. These are likely to be other websites related to your chosen topic. Each webpage has a unique address, a bit like your home. This address is called the URL or *Uniform Resource Locator*. You will have seen URLs before; they look something like this:

http://www.google.com

When you are setting up hyperlinks to different sites on the web, you need to copy the URL and paste it into Dreamweaver.

Task 2 - Creating an External Hyperlink

Use any of the tools in Dreamweaver to create some external hyperlinks. The more you can work in HTML code, the better you will be prepared for the more advanced skills. To prevent errors, your hyperlinks should be copied and pasted from your browser. Some browsers will not display the 'http://' part of the URL, but this should appear when you paste the link. If it doesn't, then try adding 'http://' before the 'www' in the address. If this fails, try 'https://'.

Links to a Div within a Page

You may create hyperlinks to a particular div on a page. This may be on the same page or a different location altogether.

`<a href="#footer"></a>` - *links to the footer div on the same page*

`<a href="index.html#footer"></a>` - *links to the footer div on the page 'index.html'*

You may not notice these links doing anything unless the page is longer than the screen that it is being viewed on.

Image Hyperlinks

Image hyperlinks are like normal hyperlinks except that the visitor clicks on an image to move to the new location. Image hyperlinks are really only useful if the graphics tell the user where the hyperlink will take them, but it is worth practising the skill anyway. We will follow the common practice of having the header banner image point to the website homepage.

Task 3 - Creating an Image Hyperlink

Using HTML like that below, with the anchor tags wrapped around the image, turn your banner into a homepage link. In the case of the homepage this won't actually go anywhere, but we'll copy it to our other pages later.

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.

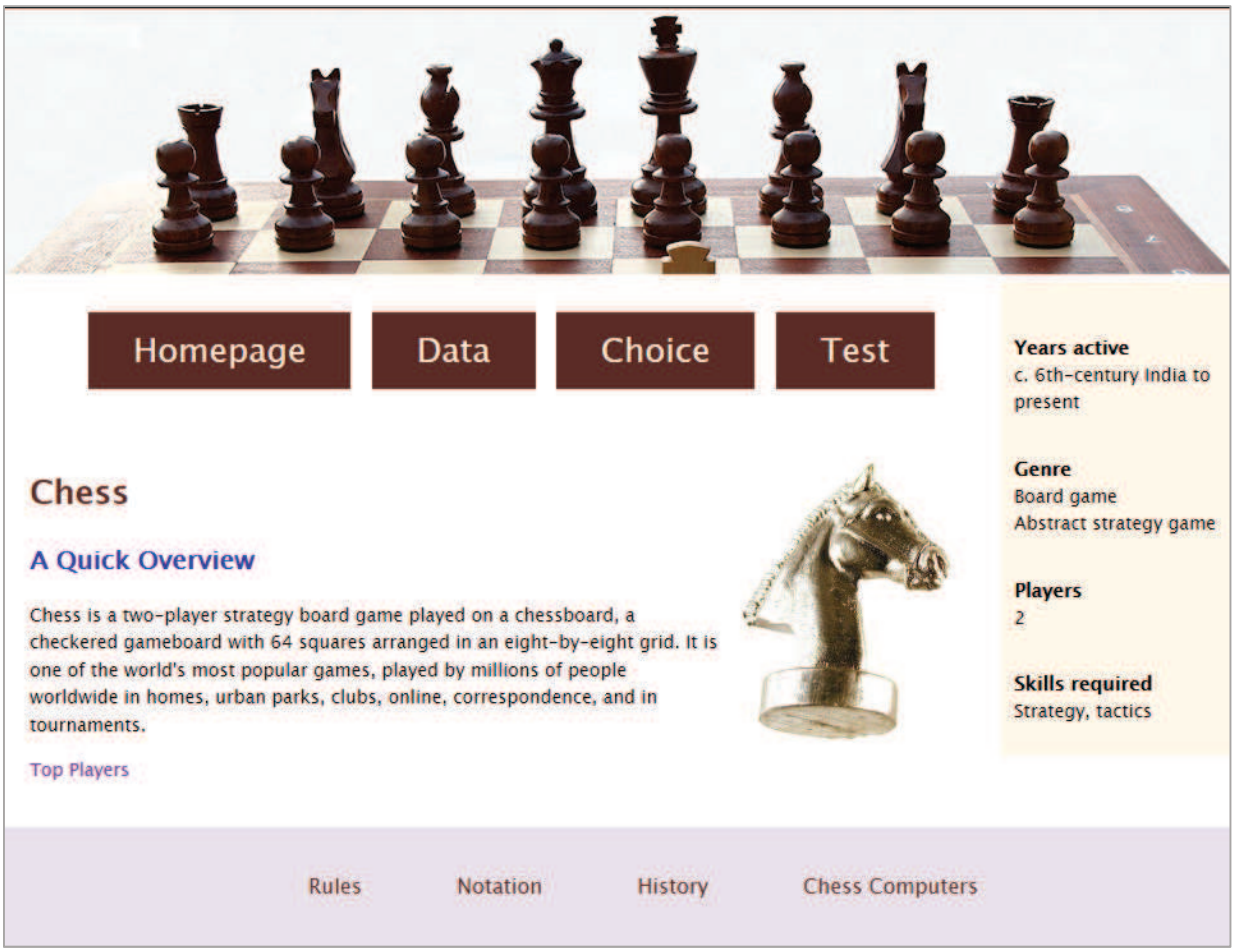
  Note: *We have used the heading style 'h3' in our sidebar, with a negative value used for the bottom margin. This has the effect of pulling up the paragraph below. This can be a useful trick.*

- 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.

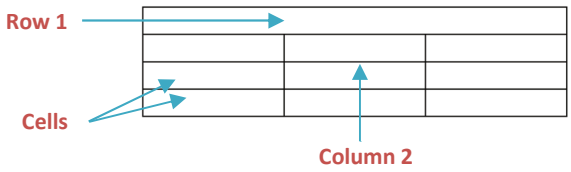
```
#container {
    width: 960px;
    margin: auto;
    background-color: #FFF;
}
#header {
}
#nav {
    width: 740px;
    padding: 20px;
}
```

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

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



We're now going to look at the HTML and CSS used to create and style tables of data. 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.



Tables are produced using the following pairs of HTML tags:

<table> </table>	*- creates and closes a table*
<tr> </tr>	*- starts and ends a row in a table*
<td> </td>	*- creates and closes a cell in a table*

Tables also use the following attributes in their layout:

<td colspan="2">	*- creates a cell which spans 2 columns (or any other number)*
<td rowspan="2">	*- creates a cell which spans 2 rows in the table*

A note about spaces

Some browsers will not recognise cells unless they contain content. It is therefore common to put a single space in cells that are to remain empty. The browser generally ignores spaces in the HTML, so you have to use the code for a space i.e. <td> </td>

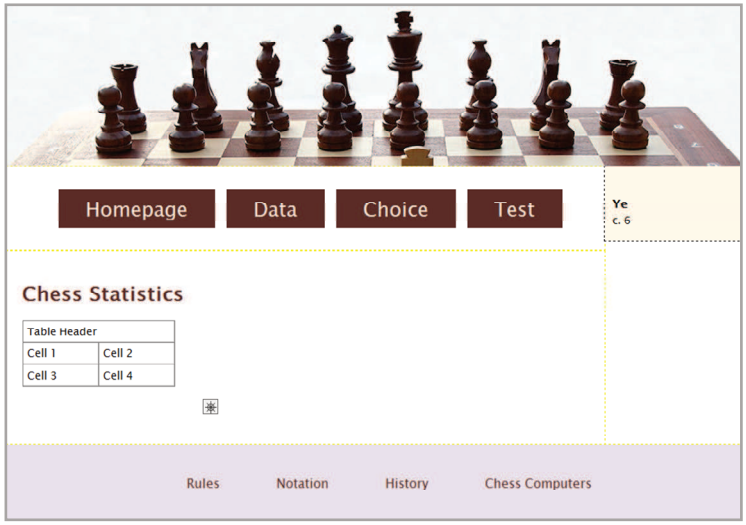
Task 1 - Table HTML

a. We are going to start by copying and pasting all the code from our homepage into our data page.

b. Once you have copied the code, delete all the text and images that you won't need in your data page, leaving the header, nav and footer divs as they were. These elements will be common to all pages.

c. You may leave the sidebar div in place if you think you may use it again, or delete it completely from the HTML.

Note: *A professional web designer wouldn't necessarily replicate code in this way. They can* store *bits of code separately which are then called upon whenever needed.*



d. We are not going to waste time typing all the HTML for a table when Dreamweaver has tools for the job. With your cursor in the *content* div, click 'Insert / Table' and select 3 rows, 2 columns, a table width of 200px, a border of thickness 1px, cell padding of 5 and cell spacing of 0.

e. Still in Design view, select the 2 cells in the top row and click 'Edit / Table / Merge Cells'. Type the text into the table cells as shown above, then have a look at the HTML that has been created.

```
<table width="200" border="1" cellspacing="0" cellpadding="5">          - set table properties
 <tbody>     -     -     -     -     -     -     identifies the body of the table (not necessarily required)
  <tr>       -     -     -     -     -     -     start the first table row
   <td colspan="2">Table Header</td>  -     -     the only cell in this row spans 2 columns
  </tr>      -     -     -     -     -     -     close the first table row
  <tr>       -     -     -     -     -     -     start the second table row
   <td>Cell 1</td>  -     -     -     -     -     second row, first data cell
   <td>Cell 2</td>  -     -     -     -     -     second row, second data cell
  </tr>            -     -     -     -     -     close the second table row
  <tr>
   <td>Cell 3</td>
   <td>Cell 4</td>
  </tr>
 </tbody>    -     -     -     -     -     -     close the table body (not necessarily required)
</table>     -     -     -     -     -     -     close the table
```

Learning about Table Dimensions in HTML

Although table styles will generally be set in the CSS, it's a little easier to get the hang of things using the HTML methods first.

Tables can be sized in either percentage terms or by the number of pixels.

`<table width="75%">` - *creates a table which will be 75% of the container width.*

`<table width="400">` - *creates a table which will be exactly 400 pixels wide.*

You can fix the height of tables using the *height* attribute.

`<table height="400">` - *creates a table which will be exactly 400 pixels high.*

The cells within tables can also be fixed in size, either as a percentage of the table width or as a set number of pixels.

`<td width="50%">` - *creates a cell which is 50% the width of the table.*

`<td width="100">` - *creates a cell with a width of exactly 100 pixels.*

`<td width="40%" height="20">` - *creates a cell with a width of 40% of the table and height of 20 pixels.*

Your browser might not always produce the tables that you have planned. Reasons for this include:

- Inconsistent measurements (one faulty requirement can throw the whole table out);
- The addition of text or pictures that do not fit into the cells;
- Creating a table that is not big enough for the cell sizes you have requested;
- Viewing on a screen that is a different size to the one you designed the page on.

Task 2 - Table HTML

Study the following HTML code and draw a scale diagram of the table that would be created. Draw on your diagram the dimensions of each cell in pixels. Remember that *rowspan* results in a single cell spanning multiple rows (so no cell needs to be created below).

```
<table border="1" width="400">
 <tr>
  <td width="20%" height="50"> </td>
  <td width="80%" height="50" colspan="2"> </td>
 </tr>
 <tr>
  <td width="20%" height="20"> </td>
  <td width="40%" rowspan="2"> </td>
  <td width="40%" height="20"> </td>
 </tr>
 <tr>
  <td width="20%" height="20"> </td>
  <td width="40%" height="20"> </td>
 </tr>
</table>
```

Extension: Practicing Table HTML

In days gone by, tables were frequently used to fix the layout of webpages. These tables sometimes became quite complicated, with tables nested inside tables. However, things have moved on. Tables should now only be used to hold data, so the design of the tables used these days is fairly simple with just a set number of rows and columns. However, there may still be occasions when a more complicated design is needed, especially when grouping data.

Complete these extension exercises to get a real grasp of table design.

a. Sketch what you think this table may look like, then create it on your last test page.

```
<table border="1">
 <tr>
  <td rowspan="2"> </td>
  <td> </td>
 </tr>
 <tr>
  <td> </td>
 </tr>
</table>
```

b. Write code to produce the tables on the right.

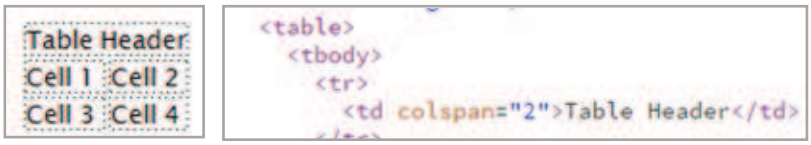
i. 

ii. 

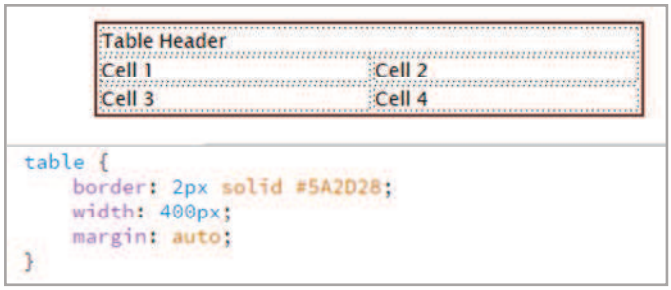
It is possible to format tables with colours, borders, margins, padding and alignment using HTML, but for the same reasons discussed earlier, it's generally considered better practice to put most of this formatting into the CSS. We will apply styles to all tables in our website, although you could target them individually if you need to with a specific ID.

Task 1 - Setting Table Styles

a. Switch back to the table created in your data page and find the HTML code. Remove all the formatting from the <table> tag so that it is like the one shown on the right. All our styling will now come from our CSS.

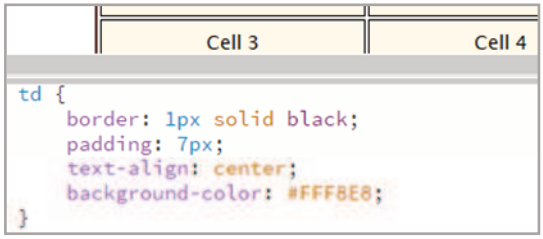


b. In your CSS file, create the definition block shown on the right (choose your own colour). Notice that the border is applied to the outside of the table only. The table width is 400px and the margin of 'auto' effectively aligns it horizontally in the centre of the *content* div.

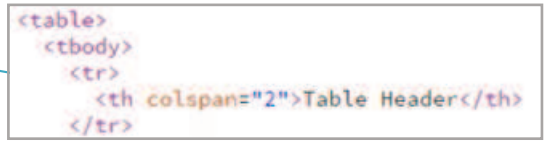


Note: *You can switch to your CSS Designer whenever you like. This is especially useful if you want to use the colour dropper to pick up colours from your images and other elements.*

c. We will now add some further styling to the table cells. The definition block on the right adds a 1px solid black border to each individual cell, along with some padding, a background colour and the text alignment 'center'.



d. Let's create a different appearance for the top row of cells. We will first edit the HTML so that the top row uses <th> and </th> tags in place of the <td> and </td> tags. The formatting should all disappear from the top row of the table.



e. You may decide that most of the formatting you have used for the general table cells (<td>) should also apply to the heading cells. We can therefore replace the line in our CSS:

td { with… td, th {

This tells the browser to apply the styles to both the <td> and <th> tags. You may then add a further definition block for the <th> tags only, something like the one shown on the right.

f. Finally, you may not like the double borders. To lose these, add the following line to your table definitions. The effect of this may only be seen when using Live view or your browser.

border-collapse: collapse;

Use all these HTML and CSS methods to add some data to your data page. It should be relevant to the subject of your website.

Table Header	
Cell 1	Cell 2
Cell 3	Cell 4

We will finish our website by setting up a page with a background image. Sometimes, background images may be used which have a simple pattern or graphic, but on other occasions, a background image may include buttons and other elements essential to the website.

We're going to start by introducing a background that is for visual effect only. If you keep your container and other divs coloured, then you can add a background to the page that will only be seen either side of our content. In this case, you can select whatever images you like. On the other hand, if you are placing a background image behind text in one of your divs, then you should be careful that this doesn't distract the reader – it can be very difficult to read text on anything other than a white background. Faint images or graphics with gradients of one colour often work well in this situation.

One other issue to consider when selecting a background image is the size of the file. Large pictures with thousands of colours tend to have large file sizes, which then have to be downloaded by the user before they appear on the screen. It is common to have a small background picture which is repeated across or down the screen, although this is becoming less of an issue as download speeds increase.

A. Page background behind a coloured container div.

B. Faint background placed behind some text

C. Thin slither repeated across the screen. This tiny image file is only downloaded once.

Task 1 - Page Background Images

a. Copy the HTML from one of your other pages into your third page (ours is still called 'choice.html'). We want to keep the consistent feel of the website. Delete any unwanted content.

b. We'll start by having a look at Example A above left i.e. an image that will stretch out behind our *container* div. Search the web for a suitable background image. Try adding the word "background" to your search terms.

c. Save potential files to your *images* folder. You can delete any unused images after you have finished.

Note: *Try selecting smaller images than you think you may need at this point. Enlarging pictures can result in fuzziness, but this is sometimes effective for backgrounds. If necessary, enlarge your image so that it is in the order of 1000px across before use.*

d. Use 'File / Page Properties' to insert a background image. This will create a style section near the top of your webpage HTML. We can move the declaration into your CSS file later if you want to use the background image on all pages.

Note: *Leave the link to the external CSS file in place as well or you will lose all your other styles.*

```
<title>Chess Homepage</title>
<link href="css/style.css" rel="stylesheet" type="text/css">
<style type="text/css">
body {
    background-image: url(images/back1.jpg);
}
</style>
</head>
```


Task 2 – Background Image Adjustments

a. Save the page and view it in your browser. Here are some of the things you may notice:

- If the image is larger than the screen then only part of it may be visible. We can adjust this in the css.
- If the image is thinner than the screen then it should be repeated across the page. We can change this behaviour as well.
- Similarly, if the image is shorter than the screen then it will be repeated down the page.
- If the background shows through in an undesirable way, you may need to add a background colour to your divs. This is especially the case in sections where text will be added.
- The background may just not work. The colours may not match or it could look out of place. In this case, you can either recolour your image in a graphics application, look for another image to test or redesign your website around the background. Finding another image may be the easiest solution.

b. As suggested, your background image will repeat both horizontally and vertically by default if it has space in the browser window to do so. This can sometimes result in a positive effect, but at other times you may only want it to repeat in one direction.

If it isn't already, change the background image to something reasonably small so that you can test the following pieces of code.

Adding the line below to the *body* declaration block in the head section will cause the image to repeat in a horizontal direction only.

```
background-repeat: repeat-x;
```

```
body {
    background-image: url("images/back1.jpg");
    background-repeat: repeat-x;
}
```

And this line will cause the image to repeat in a vertical direction only:

```
background-repeat: repeat-y;
```

c. You may stop a background image repeating completely using this line of code (in place of either of the above):

```
background-repeat: no-repeat;
```

You may then position the single image on the screen using additional code such as this:

```
background-position: right top;
```

d. You could try stretching your image to fill the background with this code:

```
background-size: 100%;
```

Test this setting, viewing the results in your browser. Try reducing the size of the browser window; you should find that the background image changes size. This may work for you, depending on your intentions.

e. You can stop your image scrolling with a long page using:

```
background-attachment: fixed;
```

f. Finally, for the difficult question of how large to make your background image. This is a huge topic because you must take into consideration screen and file sizes. As we have suggested before, a large image will cover all circumstances without the need to enlarge it in the browser, but you don't want to end up with a large file size or low-quality image. A size such as 1920 x 1080 pixels may work if you can keep the file size down. Have a play and make a decision as to what size to use. In the end, it doesn't have to be perfect – we are just learning about these things.

g. Delete any unused images when you have finished, or move them to a location outside your root folder.

Task 3 – Using the Background Image on All Pages

If you'd like to use the background on all pages, then move the CSS declarations into the *body* declaration block of your external CSS file (or copy this too if there isn't one in place). Remove the stylesheet from the head of your webpage.

You should find that the background image disappears. This is because we have broken the link to it. The *style.css* file is in the *css* folder and the background image is in the *images* folder. To link from the css file, we need to say 'go up one folder in the folder tree, then go into the images folder. This is achieved with the code *../images*.

Edit your image location so that the background reappears.

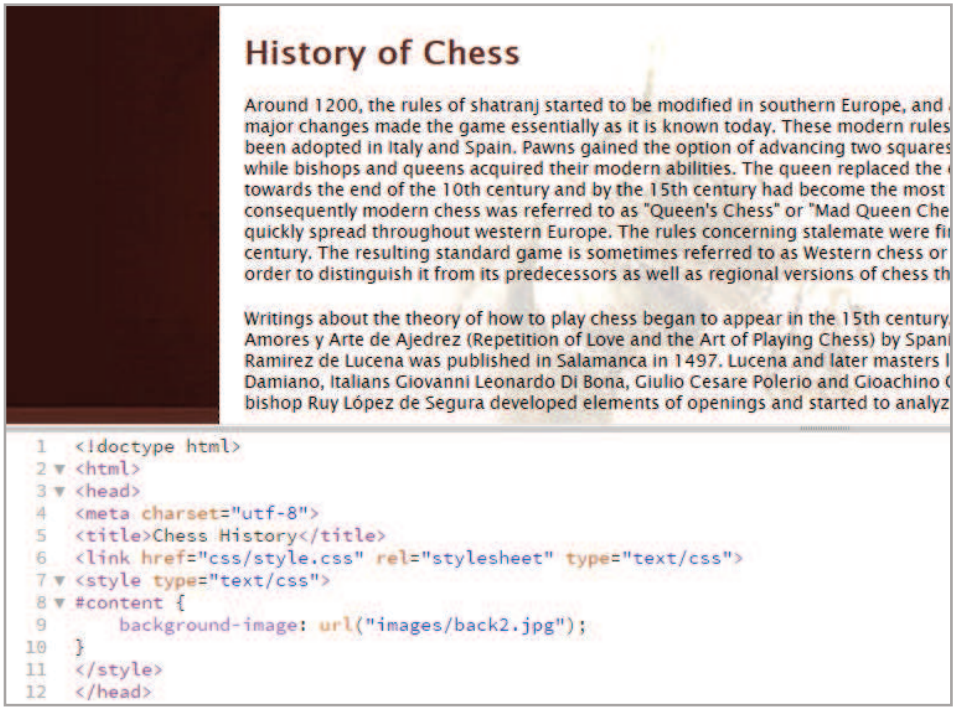
```
body {
    background-image: url("../images/back1.jpg");
    background-attachment: fixed;
}
```

Note: *If you have used the background size of 100% then you may find that it jumps slightly when navigating between long pages which have a vertical scroll bar, and short ones that don't need a scroll bar. We have demonstrated this by adding a lot of text to our 'choice' page.*

Task 4 - Div Background Images

Previously, we added the background image to our body declaration so that it appeared across the whole page. We can also add an image to a particular div. However, if this image is going to be behind any text, you should be very careful not to make the text difficult to read.

In our example, we have found a picture of a knight and used a graphics application to make it very faint (adjusting brightness controls or colour balances can achieve this). We have then placed the declaration in the *content* div in the actual webpage, rather than in the external CSS file. The image will be displayed only in this one page.



Task 5 - Finishing Your Website

Using all these tools and tricks, along with any that you discover yourself, make further changes to your website. There are lots of help guides on the internet that, although complicated for a beginner, are very useful once you get the hang of the basics. Good luck :)

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 text*
	- *emphasised text*
	- *use largest heading*
	- *use smallest heading*
<font face… size… color…><b><i><u> should all be set in CSS	

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

Lists	
	- *create a numbered list*
	- *create a bulleted list*
	- *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*
	- *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

