



Adobe Dreamweaver 2021

6 – Cascading Style Sheets

So far, we have a functional but very bland looking website. As discussed earlier, HTML is intended to define how a website is constructed. It was never intended to define how a website looks. Over time, websites began to use formatting tags such as `` that were cumbersome and not very flexible. Even worse, page layouts were created using elements such as tables within other tables which created additional problems. As such, use of those older formatting tags is now discouraged.



To address this issue, the **Worldwide Web Consortium** added **CSS**. CSS stands for **Cascading Style Sheets** and is the common way for modern websites to be positioned and formatted. The idea is that HTML still defines the structure of the page, while CSS describes how the elements on the page actually look. The 'Cascading' part of the name refers to the way styles work. Style rules can cascade through your website to make it easy to control formatting throughout your whole site from one location. CSS allows you to control things such as:

- Where elements will be on the page - you can be very exact with positioning
- Formatting – such as font and colour

Using styles to format your website instead of using HTML formatting tags has the following advantages:

- More precise control than HTML alone can provide over layout, fonts, colours, backgrounds, and other typographical effects.
- A way to update the appearance and formatting of an unlimited number of pages by changing just one document.
- Increased compatibility across browsers and platforms.
- Less code, smaller pages, and faster downloads.

A document in a website can be formatted by CSS in several ways.

1. Linked Stylesheets

Using this method, a separate file with a .css extension will contain all of the formatting information for your website. Then each file within your website will link to that file. For example, a linked CSS file might include a style rule that specifies that level one headings will be centre aligned with large, bold navy-blue text. Every HTML file that links to that CSS file will have that rule applied to its level one headings so that the formatting for an entire website can be controlled from a single CSS file.

2. Embedded Styles

Using this technique, an HTML file will have a `<style> </style>` section within the `<head>` `</head>` section of the document's HTML code that will specify style rules that will only affect that document. If you look at the source code for your *index.html* page, you will see that an embedded style block was used to define the background colour for that page.

```
<style type="text/css">
body {
    background-color: #99ffcc;
}
</style>
```

3. Inline Styles

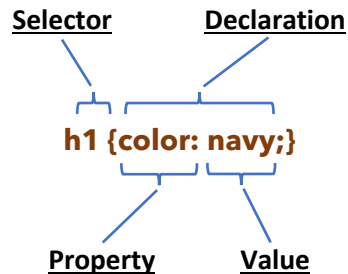
In cases where a certain style is only needed for one part of a document, then an inline style will be used. We have already used inline styles a few times on our flower pots website. Such as when we used inline styles to specify that a certain paragraph should be centre aligned. This adds some style information to a particular HTML tag which will only affect that tag. It will also overrule any embedded or linked styles. In most cases a CSS file that can provide formatting information for multiple parts of the website is preferable to using inline styles for specific parts of the site.

Note If different style rules contradict each other, then the last one specified will take preference (hence the **Cascading** part of CSS).
E.g. If a linked CSS file says that headings for the site will be blue in colour, then an embedded style section in the head section of the document says headings for that page will be blue and finally an inline style says that a particular heading will be brown – that inline style will be used for that heading since it was the last rule specified.

Style Rules

Each CSS file, embedded style block or inline style defines its styles by making use of **Style Rules**.

A Style rule consists of 2 parts. The **Selector** determines what the style rule will actually be referring to while the **Declaration** contains the style information for that selector. Look at the example below.



In this example, the **Selector** (`h1`) specifies that the style rule refers to any `h1` elements in the document(s) that are affected.

The **Declaration** `{color:navy}` specifies that any part of the document matching the style selector (in this case level 1 headings) will be formatted in navy blue. It consists of a **Property** (`color`) and a **Value** (`red`) which are separated by a colon `:`. Declarations end with a semi-colon `;` and are enclosed in curly brackets `{ }`.

Combining Selectors

When there are several selectors that will have the same declaration, they may be grouped together with each selector separated with a comma. For example, if your level 1, 2 and 3 headings will all be formatted with a navy-blue colour, then instead of having a separate style rule for each one they can be grouped into a single rule as shown below.

```
h1, h2, h3 {color: navy}
```

Combining Declarations

Declarations can also be grouped together using CSS shorthand. For example, if your headings are going to be bold, navy blue, 18 pixels high and formatted in Arial font, you can group the declarations into a single rule as shown below.

```
h1 {
  color: navy;
  font: bold 18px Arial}
```

Conditional Selectors

If a style rule contains more than one selector separated by spaces, it means that the last selector will only apply if it is within the ones before it. In the example below, paragraphs will be formatted with small font size, but only if they are within the footer section.

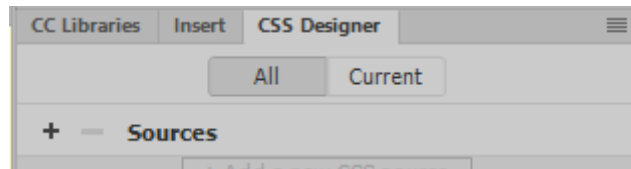
```
footer p {font-size: small;}
```

Using Dreamweaver, you can create CSS files, embedded style blocks and inline styles to take advantage of the CSS capabilities as we will do in the following exercises.

Note Dreamweaver allows you to control how CSS is written. If you want Dreamweaver to use abbreviated forms when writing CSS, you can open the Dreamweaver **Preferences** from the **Edit** menu. From the preferences you can go to the **CSS Styles** category and select the options to use shorthand.

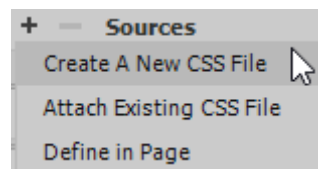
Exercise 1. Creating a Linked CSS File

1. Make sure you still have your *Francie's Fabulous Flower Pots* site open in Dreamweaver.
2. Open the *Main Pages* template.
3. Ensure the **CSS Designer** panel is open on the side of the screen. If it is not, then select **CSS Designer** from the **Window** menu.



To make use of a linked CSS file we either need to create a new CSS file or use an existing one. Since we don't yet have a CSS file, we will create a new one.

4. Click the + button next to **Sources** in the **CSS Designer** panel.



If we were going to create style rules that were only going to be used in this page, then we would use the **Define in Page** option to create a style block within the page's **head** section. When we specified a background colour for the index.html page earlier, Dreamweaver created a style block in that page as you can see from the code on that page.

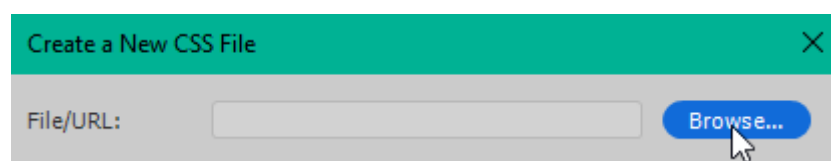
```

2 ▼ <html>
3 ▼ <head>
4 <meta charset="utf-8">
5 <title>Francie's Fabulous Flower Pot Website</title>
6 ▼ <style type="text/css">
7 ▼ body {
8     background-color: #99ffcc;
9 }
10 </style>
11 </head>

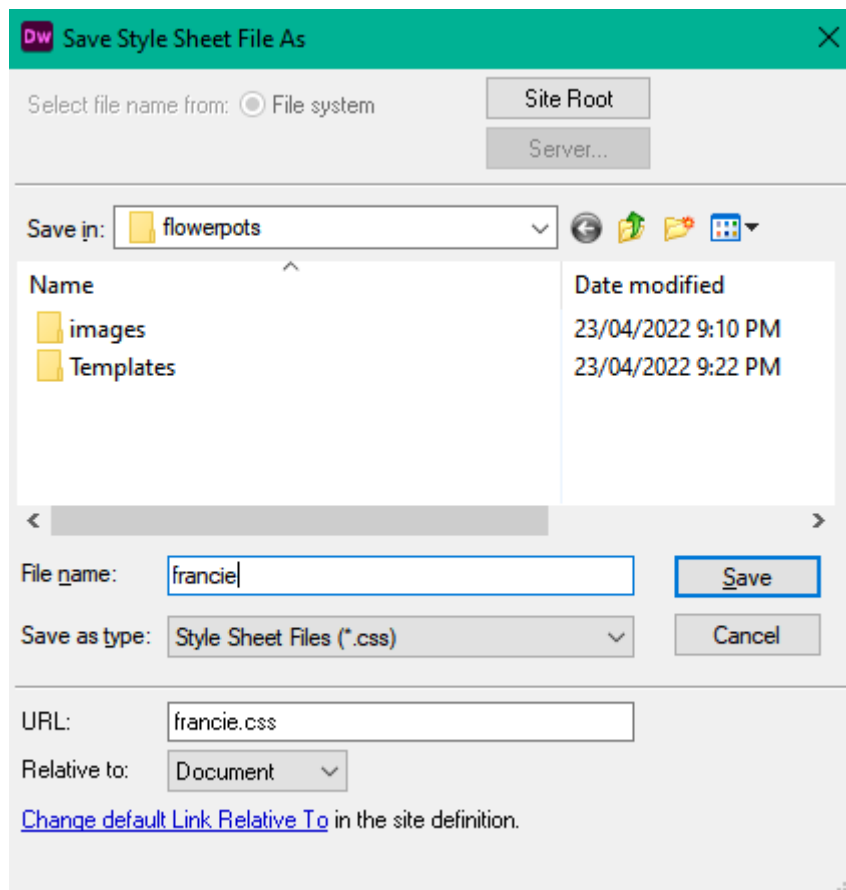
```

In this instance though we want to create a CSS file that will be used by all of the pages in our site.

5. Select **Create A New CSS File**.
6. Click the **Browse** button to specify where the new file will be saved.

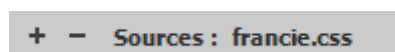
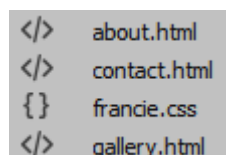


7. Make sure the save location is the main folder of the Francie's website and enter *francie* for the name of the css file.



8. Click the **Save** button.
9. Confirm the information is correct and then click **OK**.

The new file will now appear in your **Files** panel, and it will also appear as a **Source** in your **CSS Designer** panel.

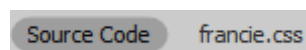


You will also notice that at the top of the editing area there are 2 buttons. Clicking the **Source Code** button will show you the HTML for the current page if you are in code view (or if you are in design view then it will switch to split view with the HTML for the current page displaying).

The following line of HTML has been added to head section of the template.

```
<link href="../francie.css" rel="stylesheet" type="text/css">
```

If you click the **francie.css** button then that will show you the code for the linked CSS file which is currently blank.



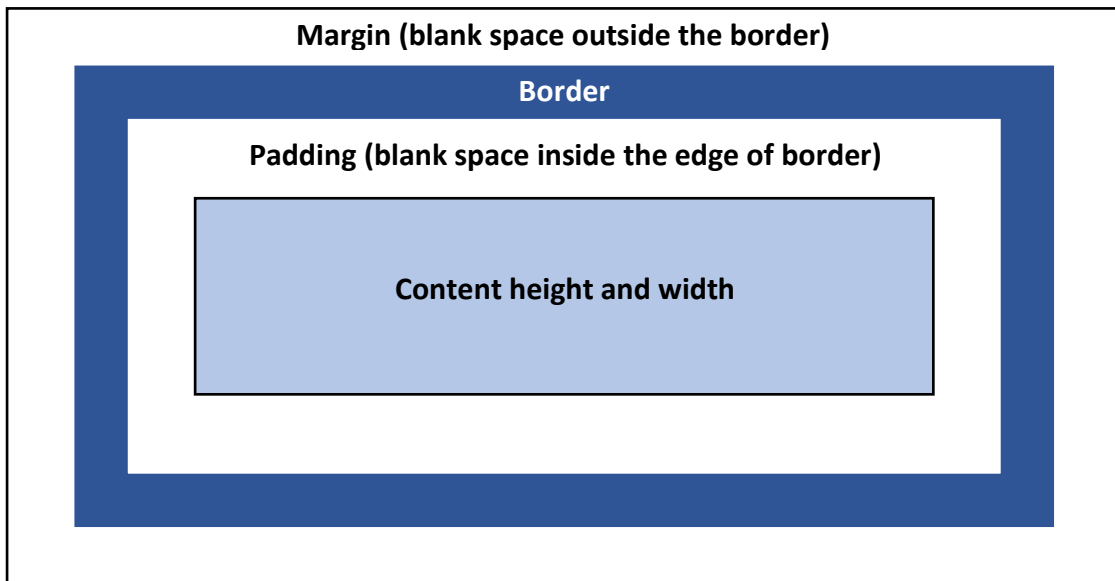
10. Save the template. Since other pages are based on this template, you will be prompted to update those pages so that they will also link to the CSS file.

The CSS Box Model

HTML elements are formatted and positioned with CSS using the Box Model. Each element can be modified using the following measures.

- **Height** and **Width** of the actual content box.
- The **Border** (if there is one) surrounding the box including the thickness of the border lines.
- **Padding**: The blank space between the border and the content
- **Margins**: Blank space outside the border, or space between the element and surrounding elements.

The diagram below illustrates how this works.



In most modern web browsers, you can right-click on part of a page and select an option to **Inspect** that element. This provides a lot of information about the selected element including the HTML code for that element and related CSS properties. Some browsers will also show the box model for the selected element. The example below shows the information Mozilla Firefox provides when we inspect the heading image on the *index.html* page of the Francie’s site. Note the box model on the right side of the inspect details.

The screenshot shows the browser's developer tools. The HTML code for the heading image is highlighted:

```

```

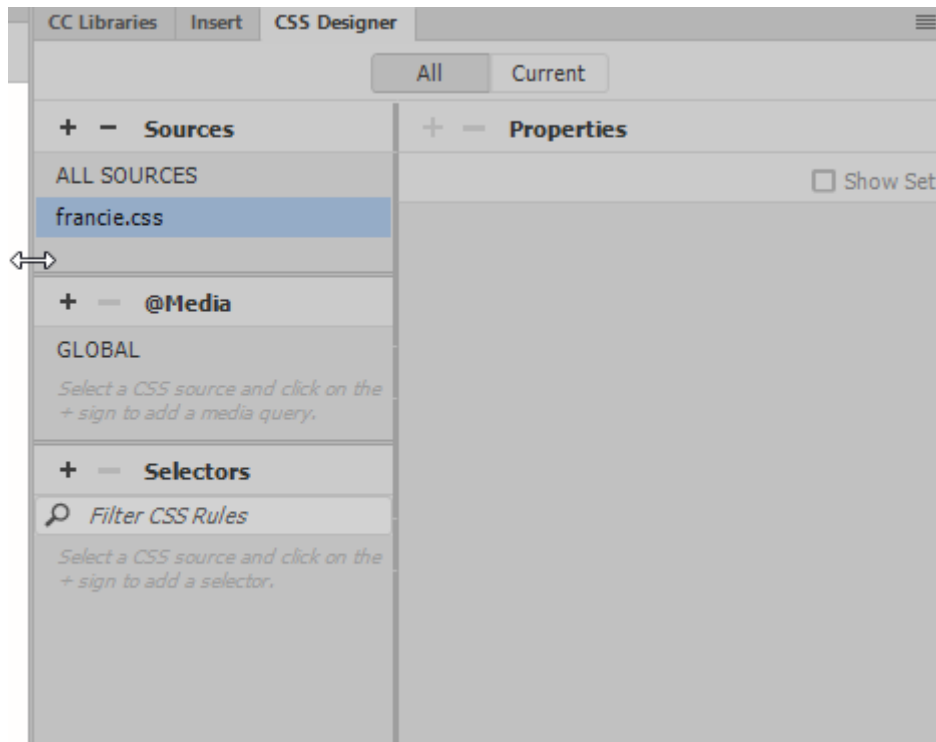
The CSS box model diagram on the right shows the following dimensions:

- Content: 1000 x 148
- Padding: 0
- Border: 0
- Margin: 0

Exercise 2. Adding Page Layout with CSS Rules

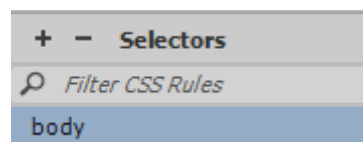
The **CSS Designer** panel can be a handy tool but it's more useful when it has more space.

1. If you have a large enough screen (or more than one monitor) then resize the panel so that it is wide enough to show the CSS properties to the right of the selectors instead of beneath them.



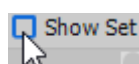
We'll start by adding a CSS rule that will provide some layout information for the **Body** of all the pages that link to the CSS file.

2. In the **CSS Designer** panel click the **+** icon next to **Selectors** to specify the selector for the new rule.
3. If you have part of your document selected then the selector will already be filled in based on the part of the document you have selected.
4. Change the **Selector** to **body**. You might need to press **[Enter]** when you have finished typing the name of the selector. This means that whatever CSS properties we choose will apply to everything in the body section of the document.

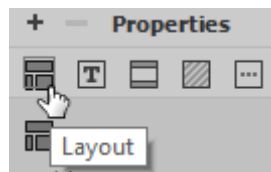


While the body selector is still selected, the properties that will apply to that selector will appear in the **Properties** section of the **CSS Designer** panel.

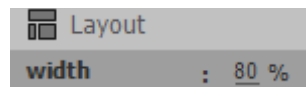
5. Untick the **Show Set** option. Otherwise the Properties section will only show properties that are already in use – which might not be very helpful when we haven't yet used any.



6. The **Properties** section of the **CSS Designer** panel has several categories that can be changed from icons along the top. Make sure the first one (**Layout**) is selected.



7. Next to the **width** property enter **80%**.



You will notice that the content of the page (i.e. everything within the body section) will only take up 80% of the available width. On larger screens we might want to set an upper limit for the size.

8. Enter **1000px** for **max-width**.

Now on larger screens the width will be no larger than 1000px (1000 pixels wide).

9. Check the code for the CSS file and you will see the following.

```
@charset "utf-8";
body {
    width: 80%;
    max-width: 1000px;
}
```

Our CSS file so far has a single CSS rule. The selector is for the **body** element. The selector has one declaration that sets the value of the **width** property and another declaration that sets the value of the **max-width** property.

Measurement units

Dimensions in CSS can be measured using a wide range of units including the following:

Absolute measurements

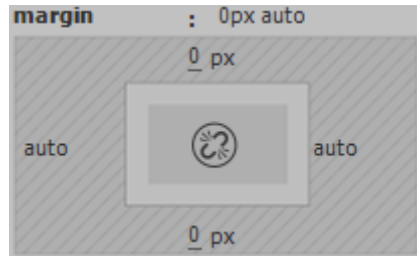
cm	Centimetres
mm	Millimetres
in	Inches
pt	Points – commonly used with text. 1 inch = 72 points
pc	Picas – commonly used with text. 1 pica = 12 points
px	Pixels (the dots that make up the image on a screen)

Relative (variable) measurements

%	Percentage of available space
em	Relative to the font-size of the element. E.g. 3em for text size could be used to make text 3 times larger than normal.
rem	Relative to the size of the root element. E.g. Font size could be set for a whole document using an absolute amount such as px. The different elements in the page could then have their size set in relation to that using rem.
vw	Viewport width. Size relative to the width of the screen.
vh	Viewport height. Size relative to the height of the screen.

We'll add 2 more declarations to our body selector.

10. With the **body** selector still selected in the **CSS Designer**, enter **100%** for the **height** property.
11. For the **margin** property enter **auto** for both the **left** and right **margins** with **0px** for both the **top** and **bottom** margins.



This will make sure there is no blank space above and below the body (some browsers put a bit of blank space at the top by default). The auto on the sides means that any available space will be equally shared between the left and right, resulting in our body being centre positioned right between the left and right sides.

Normally this would be written in your CSS as something like:

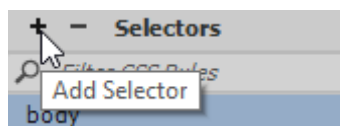
```
margin-top: 0px;
margin-left: auto;
margin-bottom: 0px;
margin-right: auto;
```

If you changed your Dreamweaver preferences to use CSS shorthand then it will write the same CSS much more efficiently in a single line which sets top and bottom set to 0 with the sides to auto.

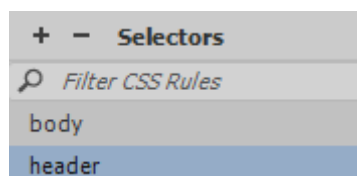
```
margin: 0px auto;
```

Now we'll create selectors for some of the other elements on the page.

1. Add a new selector.



2. Enter **header** for the name of the selector so that this will apply to the header section of our pages.



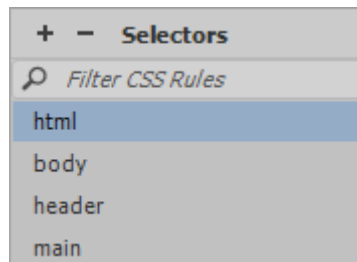
3. Enter **75px** for the **height** property.
4. Create a selector for the main element with properties as shown below.

```
Main          padding: 5px;
              min-height: 100%;
```

Exercise 3. Background Properties

In an earlier section we specified the background for a single page using an embedded style block. Now we will use our linked stylesheet to specify backgrounds for multiple pages as well as for specific elements on each page.

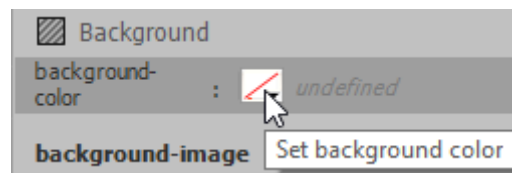
1. Create a new selector for *html*.
2. Drag it up the list so that it is above the other selectors. This won't make any difference to how our page looks but placing our selectors in a logical order can make editing easier.



3. Select the **Background** category in properties.

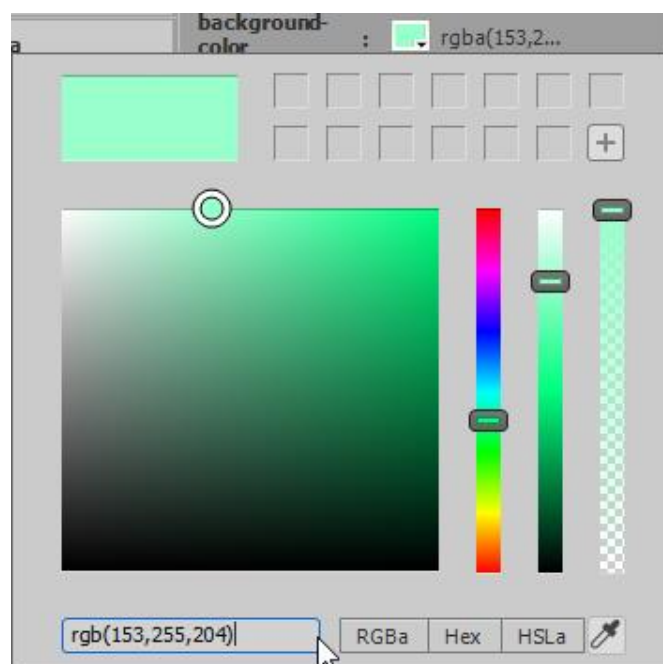


4. With the *html* selector still selected, click the colour box next to **background-color**.



You can use the colour picker to select a colour or if you already know the colour to use, you can type its values.


5. Enter **rgb(153,255,204)** for the colour value.



Tip [Refer to the end of this document](#) for an explanation of how colours can be specified in CSS.

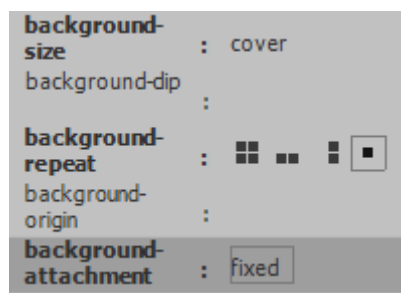
You might not see the background colour in **Design** view, but you will see it in **Live** view and in your **Browser Preview**.

We will also set an image to be used as the background. The image will cover the background colour, but the colour will still be used in case the image does not load (or while the browser with a slower connection is waiting for the image to download).

6. Click the **Browse** icon  next to the **background-image** property.
7. Select the *background.jpg* file from the *images* sub folder.

We'll use some additional properties to set how the background image will display.

8. Change the background properties as shown below.



The CSS for the html selector should look like the following.

```
html {
  background-color: rgba(153,255,204,1.00);
  background-image: url(images/background.jpg);
  background-repeat: no-repeat;
  background-size: cover;
  background-attachment: fixed;
}
```

Preview your page in your browser. We have a problem, but it's easily fixed. The text is very hard to read on that background. We will fix that problem by setting a different background for parts of the page that will have text on them.

9. Change the **background-color** property for other selectors as below. Note that they use an RGBA colour with the 4th value making the colour slightly transparent so you can partially see the background image through the colour.

header	rgba(255,255,255,0.80)
main	rgba(255,255,255,0.90)
footer (you will need to add a new selector)	rgba(153,255,204,1.00)

Exercise 4. Font Styling

Before setting font sizes for different parts of our pages, we will set a baseline font size for the html element. Then we can use `rem` measurements to set sizes for specific sections of the page in relation to that base size. E.g. we can set the baseline font to 10 pixels in height. Then if we set the font size of a certain section as 2rem, that will mean double the baseline size or 20 pixels.

1. Click the **html** selector in the **CSS Designer** panel.
2. Click on the **Text** category icon.



3. Enter **10px** for the **font-size** property.

This is a very small size for text, but we are simply using this as a baseline for the document. Each section will set its own size relative to this. Setting font sizes in this way makes it easier to create designs that scale well for different screen sizes. Also, we can adjust the all of the font sizes by changing this one value.

```
font-size : 10 px
```

Type faces

Like most documents, you can use different type faces in web pages. The problem is that not all computers have the same type faces available and a type face will only work if it is available on the computer that is viewing your web page.

To get around that problem, more than one type face can be specified. That way if your preferred font isn't displayed, you can designate a more common font to be used as an alternative.

4. Click next to the **font-family** property. Dreamweaver includes several pre-defined choices. Each choice has a few type faces with the last options being used if the first are not available.
5. Choose "Gill Sans", "Gill Sans MT", "Myriad Pro", "DejaVu Sans Condensed" from the list. Note that type faces with more than one word in the name must have quotation marks around them. Since we have set this for our html selector, the whole page will use this font selection unless individual page elements over-ride that choice.

```
font-family : Gill Sans, Gill San...
```

6. Click the **header** selector and set the following properties in the **Text** category.

```
font-family: "Lucida Grande", "Lucida Sans Unicode", "Lucida Sans  
font-size: 4.5rem;  
text-align: center;  
font-weight: bold;  
color: rgba(61,106,83,1.00);
```

7. Click the **main** selector and set the **font-size** property to **1.7rem**.
8. Click on the footer selector and change text properties as follows.

```
font-size: 1.4rem;  
text-align: center;
```

Exercise 5. Pseudo Classes

Pseudo classes allow you to add specific states to certain elements. There are pseudo classes available for elements like hyperlinks, form input controls and table elements. Pseudo classes are shown by placing adding a colon to the selector followed by the required pseudo class.

Hyperlink Pseudo Classes

Pseudo classes are commonly used for hyperlinks so that different formatting can be shown depending on whether a link to another site has been visited or not, whether the link is active, and whether the mouse is hovering over the link. Let's try these for the links in our site.

First, we'll create a general **"a"** selector for anchors (hyperlinks) in our site.

1. Create a new **"a"** selector.

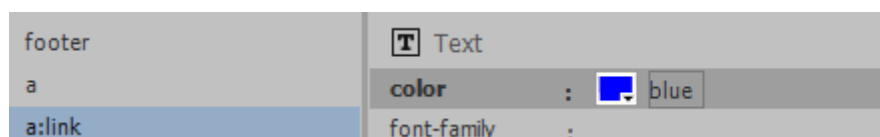


2. Under the Text category, set the text-decoration property to none. Hyperlinks are underlined by default but changing this property will mean that they will not be underlined.

There are four pseudo classes for hyperlinks:

a:link	Specifies properties for links in their normal state
a:visited	Specifies properties for links that point to locations that the user has already visited
a:active	Specifies properties for links that are active. E.g. when you have clicked on a link and are waiting for the next page to load.
a:hover	Specifies properties for links when the mouse pointer hovers over them.

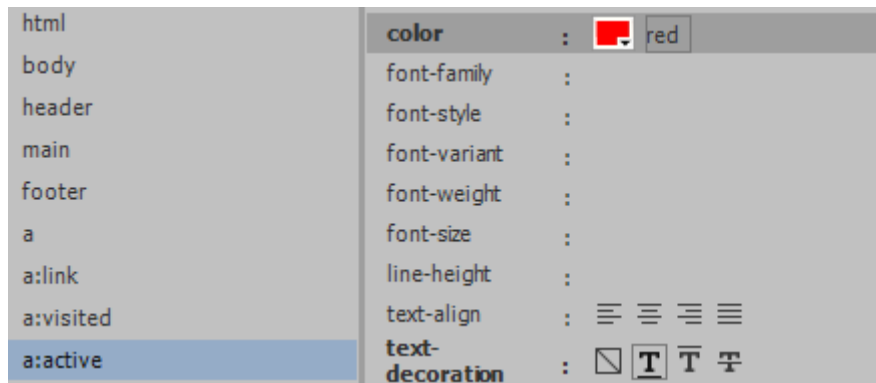
3. Add a new selector for normal links (**a:link**). Links are normally blue by default so there is no need to specify that but we will anyway to demonstrate this pseudo selector.
4. Under the text category set the **color** property to **blue**. Of course you are welcome to pick a different colour if you prefer.



5. Add a new selector for visited links (**a:visited**). Visited links are purple by default. Again we don't need to specify that but we will to demonstrate the use of this pseudo class.



6. Add a new selector for `a:active`. This time change the **color** property to **red** and set the **text-decoration** property to **underline**.



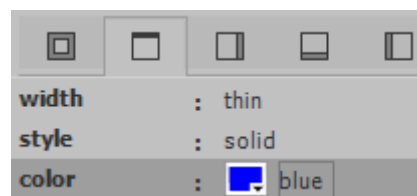
7. Lastly create a new selector for `a:hover`.
8. Change the **color** property in the **Text** category to **red**.
9. Select the **Border** category.



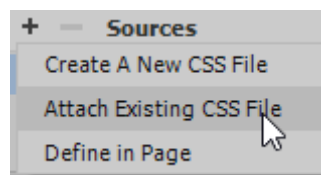
10. Select the **top border** properties.



11. Set the properties for the **top border** to **thin**, **solid** and **blue**.



12. Set the same properties for the bottom border. Now when the mouse moves over a link, the colour will change to red while a blue border will appear above and below the link.
13. Open the `index.html` page.
14. Add an existing CSS file and choose the **francie.css** file that we have been working on.



Now that all of our pages are linked to the same CSS file, they will all use the CSS rules we are creating in that file. Changing CSS rules in that one file will effect all of those pages.

15. Preview any one of the pages in your browser and test the links. You might need to select **Save All** from the **File** menu first to ensure all HTML and CSS files are saved before you preview.

Position based Pseudo Classes

1. Open the *gallery.html* page.

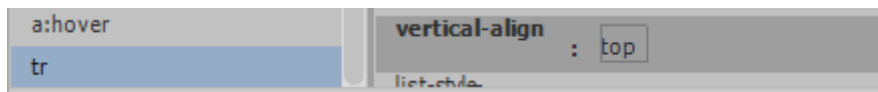
On this page (and on the ordering page) we have tables. We can use a pseudo class to apply different formatting to even number or odd numbered rows in tables. Note that if a table had an ID attribute, then we could use a pseudo class referring to that table alone but in this case we will create a selector that applies to any table.

First, we'll add a selector that will ensure the contents of all our table rows are vertically aligned to the top rather than the default middle.

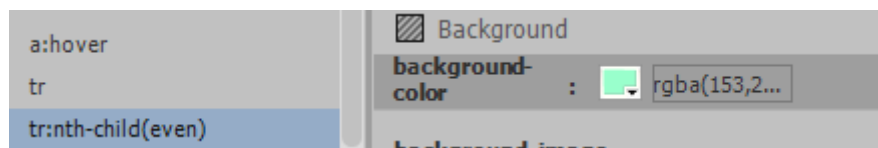
2. Add a **tr** selector.



3. From the **Text** category set the **vertical-align** property to **top**.






4. Create a new selector for the table row even numbered pseudo class (**tr:nth-child(even)**).
5. From the **Background** category set the **background-color** property to **rgba(153,255,204,0.30)**.



6. Preview the **gallery** and **ordering** pages in your browser.

A selection of our products can be seen below. Remember that we have many more types of pots available. For a full listing of our pots, or to request a custom made pot, see our [Contact Us](#) page.

Name	Description	
Red Earth	This ceramic pot is hand crafted from the finest materials so that each one is unique. The pots are coated to increase durability.	
Night Sky	This glazed ceramic pot is available in lighter and darker shades. These are especially well suited to formal gardens.	
Country Planter	The more traditional look of this pot with its glaze design and decorative pattern will make it ideal for an informal outdoor setting.	

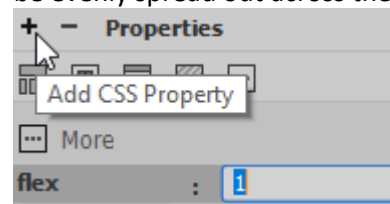
Exercise 6. Styling our Navigation

We’ve been neglecting our navigation in the navigation and footer section. Or more precisely, we’ve saved them to do after some of the other important tasks that we’ve now completed. At the moment, the text in navigation is too small, it is hard to read over the background and doesn’t really look like a navigation bar. We’ve got quite a few changes to make, but the result will be an easy to use and easy to see navigation bar that can be set to scale easily with different screen sizes.

1. **Create** the following selectors and properties. Brief explanations are provided for each. You can preview after each selector has been added to see the changes.

All of these properties are in the **Layout** or **Text** categories in the **CSS Designer** (though no matter what category you pick, you can scroll up or down to see properties in the other categories). If you prefer, you can type these selectors directly in to code view rather than using the CSS Designer panel. As you become more familiar with CSS you may find this a better way to work.

Selector	Properties (shown as they will appear in CSS code)	Explanation
nav	font-size: 1.7rem; line-height: 3.5rem; font-weight: bold;	Sets the font size and line height in relation to the 10px size we set for the page. Adds bold formatting to all text in the nav section.
nav ul	list-style-type: none; padding: 0; display: flex;	A conditional selector for any unordered lists (ul) elements that are inside the nav section. Items in the list will not show bullet points and there will be no padding around elements. The list will use a flexible layout for list elements so that they can sit next to each other rather than beneath each other.
nav ul li	display: inline; text-align: center; flex: 1;	A selector for list items that are within unordered lists within the nav section. List items will display as inline instead of block elements. If the flex property is not shown in the CSS Designer , you can click the + button at the top to add it. It allows the list items to be evenly spread out across the space.
nav a	color:blue; display: block; background-color: rgba(153,255,204,1.00); text-decoration: none;	A conditional selector for links inside the nav section. For links in the nav section, these properties will over-ride properties set for links in the rest of the document. The display block property will mean that the whole block will work as a link and not just the text in the middle of the block.



nav a:hover	background-color: rgba(255,255,255,0.90); color: rgba(61,106,83,1.00); border: 0; text-decoration: none;	Selector for hover links within the navigation. This includes 0 borders to over-ride the top and bottom borders set for hover links in the rest of the document (preview it without the border 0 to see the difference).
nav a:visited	color:blue;	Sets the visited link colour to blue for the navigation to over-ride the purple visited link colour set for the rest of the document.
nav a:active	background-color: white;	Sets a white background for navigation links when they are clicked.

Preview your pages and you will see that there is a margin gap around the navigation as well as some of the other elements. We can remove this by setting the margin to 0 for everything in the entire page. For elements that do need a margin, they can have individual selectors to over-ride the global selector. The asterisk symbol is used for global selectors.

2. Add a new ***** selector before all of your other selectors. This selector can set properties for all elements. Place it before your other selectors.
3. Set the **margin** property for this selector to **0**.

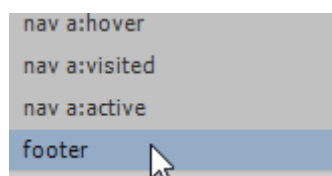


Exercise 7. Styling the Footer

We will now add some new selectors (and add to some existing selectors) to style our footer. Much of the properties will be similar to ones we set for the navigation section.

We will add to the footer selector that we have already created, and we will also add a conditional selector for list items inside the footer.

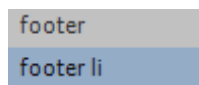
1. For the sake of being organised, drag the footer selection so that it is beneath the other selectors.



2. Add the following properties to the footer selector.

<code>clear: both;</code>	Currently some pages have an image that is aligned to the right of the page with text continuing next to it. The clear property ensures that this block element starts below any floating elements – forcing it to start on its own line and clearing text wrapping.
<code>position: fixed;</code> <code>bottom:0;</code>	These properties fix the footer element at the bottom of the page.
<code>width: 80%;</code> <code>max-width: 1000px;</code>	Sets the width and max width for the footer.

3. Create a new selector for list items in the footer area.



4. Add the following properties.

<code>list-style-type: none;</code>	List items will not display bullet points
<code>text-align: center;</code>	Centre aligns list items
<code>padding: 0 5px;</code>	No padding above and below but padding to the side of each list item
<code>display: inline;</code>	Displays list items next to each other instead of block elements
<code>line-height: 2rem;</code>	Makes the text size twice as big as the size set in the <code>html</code> selector

Since we've positioned the footer section at the bottom of the page, we'll add a property to the `html` selector to make it take up 100% of the page height to remove any gap above the footer.

5. Add the following property to the `html` selector.

```
height:100%;
```

IDs and Classes

HTML elements can be given names by using an **ID** attribute in the tag. You can then create CSS rules that refer to this ID.

Classes are custom made selectors that can be added to any element in your site.

Exercise 8. Adding ID and Class Selectors

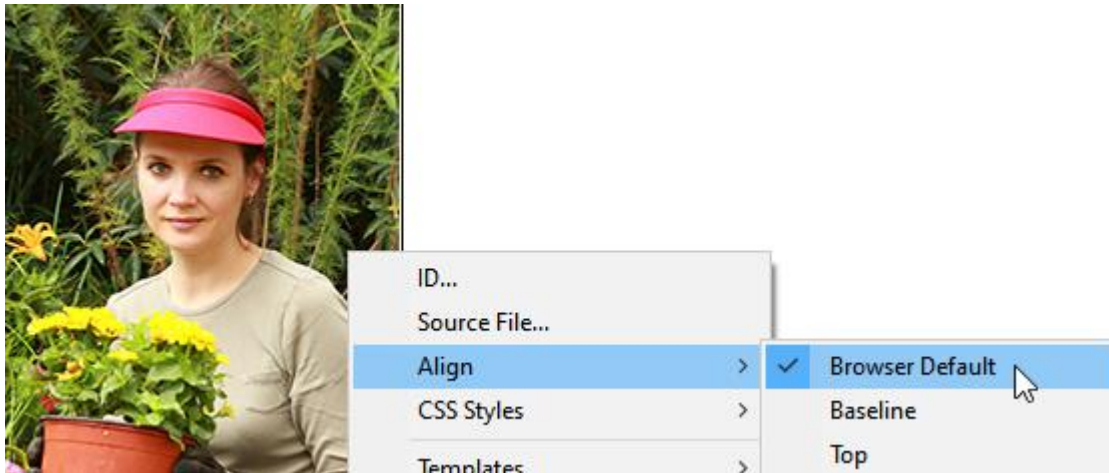
1. Open the `about.html` page.
2. Select the `about_francie.jpg` image in the main section of the page.

First, we'll clear some of the properties in the tag since we will use a CSS rule to provide the information instead.

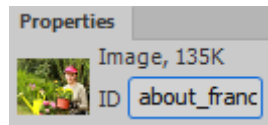
3. Clear the **Width** and **Height** attributes in the Properties panel.



4. **Right-click** on the image, select **Align** and then **Browser Default**.

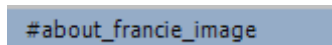


5. With the image still selected, name the image by setting the **ID** property to *about_francie_image*.



Now we can create a new style rule that uses the ID as its selector. Style rules that use an ID as the selector begin with the # symbol.

6. Create a new **#about_franacie_image** selector.



7. Set the **float** property to **right**.



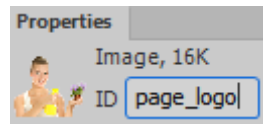
8. Set the **width** property to **400px** and the **max-width** property to **50%**.

Now the image will appear at 400 pixels width except on smaller screens when it will resize to 50% of the main section’s width.

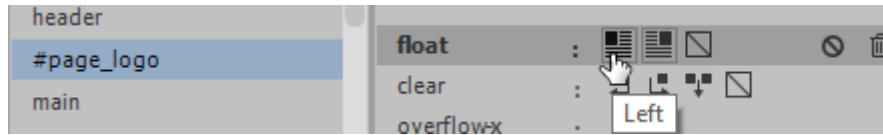
9. Open the **Main Pages** template file.
10. Select the image in the header area.



11. Enter **page_logo** for the **ID** in the **Properties** panel.



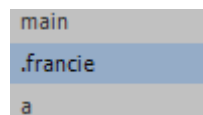
12. Create a new **#page_logo** selector and set the **float** property to **left**.



13. Save the template, updating the pages based on the template.

Now we'll create a new class selector that will be used to format the business name whenever needed. Class selectors begin with a full stop / period (.).

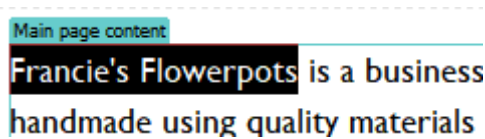
14. Create a new **.francie** selector.



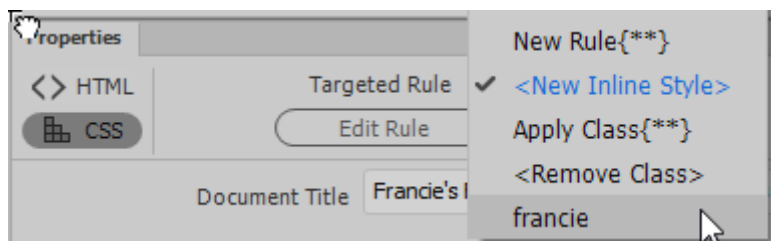
15. Set the text **color** property to **#660000** and set the **background-color** property to **#99ffcc**.

16. Open the *about.html* page.

17. In the **Main** section, select the text *Francie's Flowerpots*.



18. To make use of our new class selector, click on the Targeted Rule property in the properties panel and then select the name of our new francie class.



19. Open other pages and apply the same class where you see the name of the business.

Extra Things to Do

Index page

Make use of our main and footer selectors on the index.html page.

Change the main text from a paragraph to using the main selector. Change the row of links at the bottom from a paragraph to a footer selector. Both changes can be easily made in the HTML code.

Change

```
<p>Welcome to <em>Francie's Fabulous Flower Pot Website</em>
needs, you're sure to find a flower pot ideal for your need
<p style="text-align: center; font-size: small;"><a href="ab
href="ordering.html">Ordering</a> - <a href="contact.html">C
```

To (remember to change the closing </p> tags also)

```
<main>Welcome to <em>Francie's Fabulous Flower Pot
needs, you're sure to find a flower pot ideal for
<footer><a href="about.html">About Us</a> - <a href="
href="contact.html">Contact Us</a> - <a href="lin
```

CSS Comments

Like HTML, CSS can have comments, though the format is different. CSS comments begin with `/*` and end with `*/`.

Comments can be used to separate your CSS rules into different sections for easier editing. Since Dreamweaver (and most other editors) show comments in a different colour it makes them easy to spot in your code.

Add comments in your CSS file to create headings for similar rules. Such as the following.

```
/* general layout */
/* type */
/* header layout */
/* nav layout */
/* main layout */
/* footer layout */
```

Now you can re-arrange your CSS code so that rules that match a certain category are placed beneath the relevant comment. This makes your CSS code much easier to read and makes it a lot easier to find certain rules when you edit the CSS file.

```
/* type */
a {
    text-decoration: none;
}
a:link {
    color: blue;
}
```

CSS Colours

CSS can be used to set colours for many page elements including backgrounds, borders and text colours. Colours in CSS can be defined in several ways.

Hexadecimal codes

Hexadecimal is often used by programmers to represent large numbers. Instead of using 10 digits like we're used to with decimal numbers (0 to 9), it uses 16 digits (0 to 9 and then A to F).

E.g. the decimal number 1,563,987 can be represented as the hexadecimal value 17DD53.

Originally in HTML, hexadecimal values were the primary means of specifying colours.

RGB (and RGBA) values

Colours in computer screens are made up of varying combinations of **Red**, **Green** and **Blue** light. Each of these colours can be represented as an amount from 0 to 255. In CSS a 4th value can also be added for **Alpha** which means transparency with a value from 0.0 (completely transparent) to 1.0 (not at all transparent or completely opaque).

E.g. An RGB value of **255, 0, 120** means a colour consisting of lots of red, no green and a moderate amount of blue. An RGBA value of **255, 0, 120, 0.8** is the same except partially transparent.


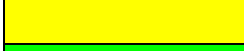



HSL values

Similar to RGB except that this method of specifying colours uses different values for **Hue**, **Saturation** and **Lightness**. HSL can also include a 4th value for transparency.

Named colours

Some of the more common colours can be specified simply by typing their names.

Common Colours

Numeric	Name	Hex	R,G,B	H,S,L
	black	#000000	0,0,0	0,0,0
	white	#FFFFFF	255,255,255	0,0,100
	silver	#C0C0C0	192,192,192	0,0,75
	gray	#808080	128,128,128	0,0,50
	olive	#808000	128,128,0	60,100,50
	yellow	#FFFF00	255,255,0	60,100,100
	lime	#00FF00	0,255,0	120,100,100
	green	#008000	0,128,0	120,100,50
	teal	#008080	0,128,128	180,100,50
	aqua	#00FFFF	0,255,255	180,100,100
	blue	#0000FF	0,0,255	240,100,100
	navy	#000080	0,0,128	240,100,50
	purple	#800080	128,0,128	300,100,50
	fuchsia	#FF00FF	255,0,255	300,100,100
	maroon	#800000	128,0,0	0,100,50
	red	#FF0000	255,0,0	0,100,100