

# ITEC447

# Web Projects

---

CHAPTER 6 – CSS2

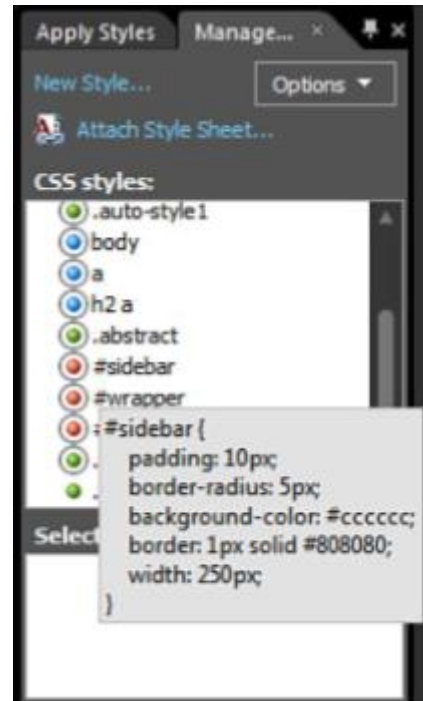
# Introducing CSS: The Code Version

---

- In the previous two hours, you created a series of styles, classes, and IDs.
- In this hour, you inspect and expand on these, so if you didn't do all the tasks in earlier hours, you should replace the *default.html* file and the *myDesk.html* file with the ones supplied in the lesson files for this hour.
- To work, CSS requires two sets of code: the styles and the tags to which the styles are attached.
- Let's take a closer look at the styles as they appear in code form.
- Expression Web 4 gives you multiple ways to view the CSS code.
- By far, the easiest way of doing so is simply to hover over the style in question in the *Manage Styles* panel.
- When you do so, a *ScreenTip* opens to display the entire style code in a pop-up window.

# Introducing CSS: The Code Version Cont.

---



# Introducing CSS: The Code Version Cont.

---

- With the *default.html* file open, hovering over the *#sidebar* style gives you the following output:

```
#sidebar {  
    padding: 10px;  
    border-radius: 5px;  
    background-color: #CCCCCC;  
    border: 1px solid #808080;  
    width: 250px;  
}
```

# Introducing CSS: The Code Version Cont.

---

- This is a typical style.
- It consists of the style name followed by a set of curly braces.
- The braces contain all the actual styling code: Each attribute followed by its respective values after a colon. A semicolon separates the attributes.
- As you can see, the CSS code Expression Web 4 generates is easy to read.
- The only reason why each attribute is on a separate line is for readability.
- If you want to, you could remove all the line breaks and write the entire style on one line, but, as you can see, it would be much harder to read:

```
#sidebar { padding: 10px; border-radius: 5px; background-color: #cccccc; border: 1px solid #808080; width: 250px;}
```

# Introducing CSS: The Code Version Cont.

---

- Now that you know what the CSS code looks like, the next logical question is, “Where is it located?” If you paid close attention when you created the styles in the previous two hours, you might already have a good idea.
- Directly under the *Selector* box in the *New and Modify Style* dialog was the Define In box, which was set to Current page.
- That means all the styles you created so far are stored in the same page as the content—more specifically, at the top of the page inside the `<head>` tag.
- The `<head>` tag contains functional but nonvisual content and information about the current page.
- To see where the styles are stored, switch to *Code* view and scroll to the top of the page.
- Directly under the `<meta>` tags is a tag that says `<style type="text/css">`.
- You can find all the styles within this tag.

# Introducing CSS: The Code Version Cont.

---

```
Site View default.html -
4 <!--
5 <meta content="en-ca" http-equiv="Content-Language">
6 <meta content="text/html; charset=utf-8" http-equiv="Content-Type">
7 <style type="text/css">
8 h2 {
9     font-family: Arial, Helvetica, sans-serif;
10    font-size: 1.2em;
11    font-weight: bold;
12    text-transform: uppercase;
13    color: #800000;
14 }
15 h1 {
16    font-family: Arial, Helvetica, sans-serif;
17    font-size: 1.4em;
18    font-weight: bold;
19    text-transform: uppercase;
20    color: #333333;
21 }
22 .newStyle1 {
23    font-family: arial, Helvetica, sans-serif;
24    color: #800000;
25    font-variant: small-caps;
26    border-bottom-color: #800000;
27    border-bottom-style: dotted;
28    border-bottom-width: 1px;
29 }
30 .auto-style1 {
31    font-family: "Courier New", Courier, monospace;
32    color: #008000;
33    border-style: solid;
34    border-width: 1px;
35    padding: 1px 4px;
36 }
```

# Introducing CSS: The Code Version Cont.

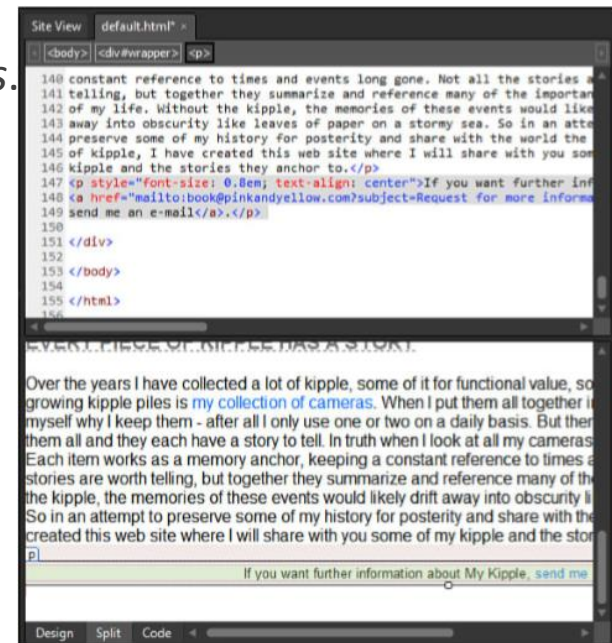
---

- To find a particular style in *Code* view, all you need to do is click the style in question in the *Manage Styles* panel, and Expression Web 4 highlights the style for you.
- While in the *Manage Styles* panel, you can even use the arrow keys to navigate between styles for quick-and-easy access to the particular style, class, or ID on which you want to work.



# Understanding Inline Styles

- With the *default.html* page open in *Split* view, scroll to the bottom of the page and place the cursor anywhere inside the last paragraph.
- Click the *New Style* button to create a new style and then use the drop-down menu to set the *selector* name to (*Inline Style*).
- In the *Font* category, set *font-size* to *0.8em* and *font-variant* to *small-caps*.
- In the *Block* category, set *text-align* to *center*.
- Click *OK* to apply the new inline style.



# Understanding Inline Styles Cont.

---

- The last paragraph of the page changes appearance after you apply the inline style.
- But what matters is what happened in *Code* view.
- Look at the tag for this particular paragraph:
- `<p style="font-size: 0.8em; font-variant: small-caps; text-align: center">`
- Rather than creating a new style and adding it to the list at the top of the page, Expression Web 4 added this style inside the tag of the affected paragraph.
- The style is in the same line as the content—hence the name inline style.
- This explains not only why the style you just created affects only this particular paragraph, but also serves as a good example of why you should always try to keep your styles separate from your content.
- Just imagine what your HTML code would look like if every tag had to contain the necessary style attributes!
- With that said, inline styles are useful if you need to apply a special style featured only once in the entire page or site.

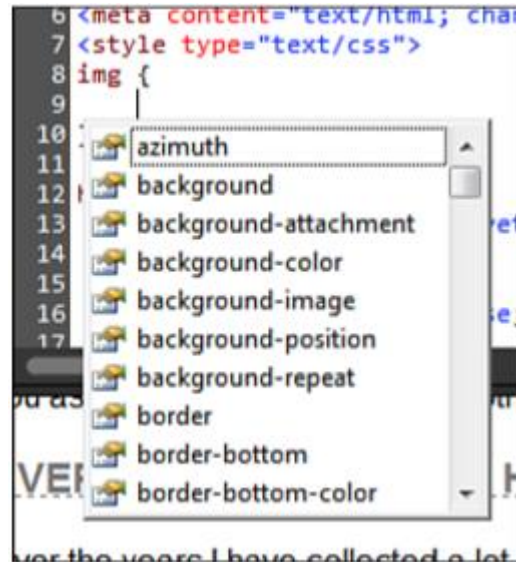
# Create a New Style in Code View Using IntelliSense

---

- In *Code* view, you can make direct changes to the CSS code or create new styles with the help of *IntelliSense*.
- In this example, you create a new *img* style from scratch to give the image the appearance of having a white background with a gray border.
- With the *myDesk.html* page open in *Code* view, find the `<style type="text/css">` tag and create a new line directly below it.

# Create a New Style in Code View Using IntelliSense Cont.

- To create the new style, type *im* and *IntelliSense* suggests *img*.
- Press the *Enter* key to accept *img*, press the *spacebar*, and type a beginning curly bracket (*{*).
- In response, *IntelliSense* automatically moves you to the next line, inserts the end curly bracket on the line below, and brings up a list of all available attributes that apply to the *img* tag.



The screenshot shows a code editor with the following content:

```
6 <meta content="text/html; char
7 <style type="text/css">
8 img {
9
10
11
12
13
14
15
16
17
```

An IntelliSense dropdown menu is open, listing the following attributes:

- azimuth
- background
- background-attachment
- background-color
- background-image
- background-position
- background-repeat
- border
- border-bottom
- border-bottom-color

# Create a New Style in Code View Using IntelliSense Cont.

---

- To create 5-pixel padding around the image, type *pa*.
- IntelliSense suggests *padding*.
- Press *Enter* to complete the word.
- *IntelliSense* now opens a *ScreenTip* to tell you what kind of information the padding attribute requires.
- Because you want 5-pixel padding on all four sides, you can type 5px and be done with it.
- If you want different values for each side, follow the *ScreenTip* and type, for example, 5px 4px 4px 10px (top, right, bottom, left).
- Complete the line by entering a semicolon and pressing *Enter* to create a new line.

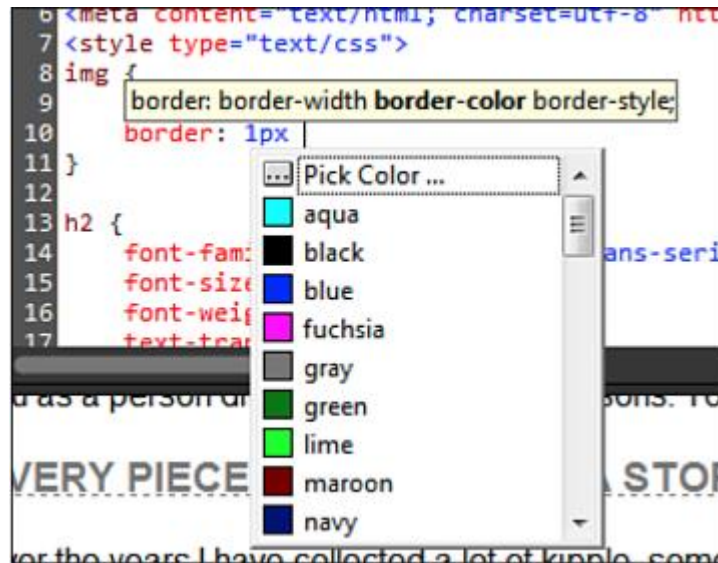
# Create a New Style in Code View Using IntelliSense Cont.

---

- To create a border, you need to set three attributes: *border-style*, *border-width*, and *border-color*.
- To help you remember this, *IntelliSense* reminds you and helps you set all three variables on one line:
- On the new line, type *border:*.
- This opens the *ScreenTip* for the border attribute.
- *IntelliSense* now asks you for the values it needs.
- First up is *border-width*.
- Type *1px* and press the *spacebar*.
- The *ScreenTip* automatically jumps to the next variable, *border-color*, and opens a drop-down menu of colors for you.
- Pick the *gray* color and press the *spacebar*.
- *IntelliSense* now asks you for the *border-style*. Select *solid* from the *drop-down* menu and finish the line with a *semicolon*.
- The two new lines should read *padding:5px;* and *border: 1px gray solid;*.

# Create a New Style in Code View Using IntelliSense Cont.

---



- To see the effects of your changes, click anywhere inside *Design* view to apply the changes.
- The image now has a 1-pixel solid gray border and 5 pixels of white padding.
- This example shows you how easy it is to write and make changes to CSS code in Code view and how *IntelliSense* works with you to simplify the code-writing process.

# Applying Classes to Tags in Code View

---

- In Hour 11, you learned to apply special styles to certain content with the use of classes.
- In one example, you used classes to change the position of the image in the *myDesk.html* page.
- This example provides a good basis for learning how Expression Web 4 applies classes to tags in *Code* view.
- If you open *myDesk.html* in *Split* view and click the image, the relevant code highlights in *Code* view.
- ``



# Applying Classes to Tags in Code View

## Cont.

```
Site View default.html myCameras.html
<body> <p> <img_alignCenter>
36 <map id="FPMap0" name="FPMap0">
37 <area alt="EOS 1 - the camera that went to war" coords="202, 5, 315, 97"
38 <area alt="An old Nikon SLR" coords="22, 141, 37, 185, 78, 169, 97, 189"
39 <area coords="103, 67, 120, 123, 146, 116, 166, 151, 194, 147, 207, 128"
40 <area coords="241, 113, 241, 154, 251, 178, 257, 178, 270, 199, 282, 19"
41 <area coords="363, 44, 355, 57, 354, 100, 350, 125, 356, 135, 367, 152"
42 <area coords="447, 102, 527, 112, 542, 127, 537, 200, 530, 229, 551, 23"
43 </map>
44 
```



Design Split Code

# Applying Classes to Tags in Code View

## Cont.

---

- Inspecting the tag code, you see a new attribute toward the end:
- `class="alignCenter"`.
- This is how Expression Web 4 applies classes to tags, whether they are selectors, spans, or divs.
- To change the class, all you need to do is edit the class name in *Code* view.
- In Hour 11, you learned how to create two more alignment classes.
- To apply one of these instead, simply change the class name to either *alignLeft* or *alignRight* and you'll immediately see the changes in *Design* view.
- Adding a class to an existing tag is just as easy: Simply type the word class before the end bracket of the beginning tag, and *IntelliSense* gives you a drop-down list of all available classes.
- To see the CSS code for the class in *Code* view, right-click the class in the *Manage Styles* panel and select *Go to Code*, or simply double-click the class to go right to it.
- No matter where you are in the program, these functions take you straight to the relevant CSS code in *Code* view.

# Using Divs and Spans to Separate Content

---

- We touched on both the `<span>` tag and the `<div>` tag earlier, and now it's time to take a closer look at these separators.
- The main difference between the two is that `span` is an inline separator, whereas `div` is a block separator.
- In other words, `span`'s `display` attribute is `inline` by default, whereas `div`'s `display` attribute is `block`.
- You saw the difference between the two when you used the `.alignCenter` class to center the thumbnail earlier:
- The `inline` value means that the content, although separated from the surrounding content, is still on the same line as the rest.
- In contrast, the `block` value creates a block or box on its own line that holds only the content inside the tag.

# Using Divs and Spans to Separate Content Cont.

---

- The *default.html* page contains two instances of the `<span>` tag that you created to highlight the word kipple and the random word you chose in the first paragraph.
- If you find the words in the *Design* portion of *Split* view and click them, you can see the corresponding `<span>` tags and how they are applied in *Code* view:
- `<span class="auto-style1"> kipple</span>`
- As you can see, the class application is no different in the span tag than in any other tag.
- However, because you were just starting to learn how to create styles, you didn't give the class a proper name, so it has the nondescript name *auto-style1*.
- It's important to give all your styles, classes, and IDs proper descriptive names so that you know what they do.

# Renaming Styles and Applying the Change to All Tags in a Page

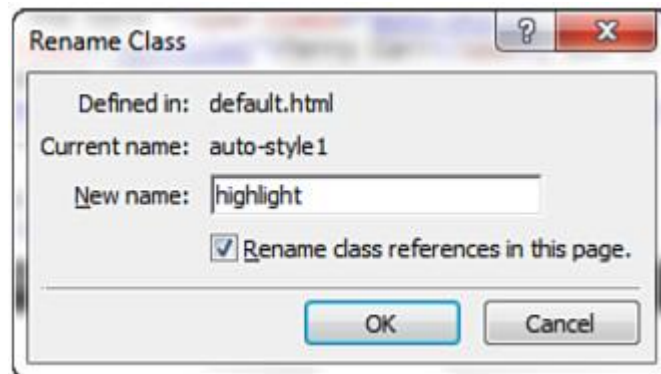
---

- When you work with large pages or sites, you often run into situations in which you need to rename a style, class, or ID. The problem is that Expression Web 4 already applied these elements to many tags within your pages, and if you change the name of the style, class, or ID, all the references have to change as well. To help simplify this process and save you from the trouble of tracking down every reference to your now changed style, class, or ID, Expression Web 4 can make all the substitutions for you.
- With the *default.html* page open in *Split* view, navigate both the *Code* and *Design* views so that you can see both span instances you created earlier by clicking one of them.
- In the *Manage Styles* panel, right-click the *.auto-style1* class and select *Rename Class "auto-style1"* from the context menu.
- This opens the *Rename Class* dialog.

# Renaming Styles and Applying the Change to All Tags in a Page Cont.

---

- In the *Rename Class* dialog, give the class the new name highlight.
- Be sure to enable the *Rename Class References in This Page* box and click *OK*.



# Renaming Styles and Applying the Change to All Tags in a Page Cont.

---

- When you click *OK*, not only does Expression Web 4 rename the class, but it also changes the references to the class in the page, as you can see in *Code* view.
- Changing all the references to a style, class, or ID to correspond with a name change extends to external style sheets, meaning that when you learn how to create an external style sheet and apply it to multiple pages, Expression Web 4 changes all references to the changed name throughout all these pages for you, even if they are not open!
- In Hour 11, you used divs to create blocks that separated and sectioned the contents of the page.
- *default.html* now has two divs: one outer box with the ID wrapper and an inner box with the ID sidebar.
- To see how Expression Web 4 applies those divs, click the sidebar in *Design* view to see all the tags applied to it.
- When you click the `<div#wrapper>` tag in the *Quick Tag Selector*, all the content affected by the tag highlights both in *Code* and *Design* view.
- To find only the beginning tag, click the *Find Matching Tag* button on the *Code View* toolbar.
- As you can see, the application of an ID is similar to that of a class: `<div id="wrapper">`.

# Renaming Styles and Applying the Change to All Tags in a Page Cont.

---

- Because divs box in larger sections of content, it can be hard to see exactly where they apply and how much content they contain.
- You already saw how to use the *Quick Tag Selector* to highlight all the content affected by a tag.
- Another way is to use the *Select Tab* button on the *Code View* toolbar.
- If you need to see where the end `</div>` tag is located, click the *Find Matching Tag* button again, and *Code* view jumps to the end tag.



# Creating Divs in Code View

---

- As you may have experienced in Hour 11, dragging and dropping divs into *Design* view can be a bit tricky.
- A much easier and more effective way of applying divs is to use *Code* or *Split* view because in *Code* view, you can see exactly what content you are wrapping and place the beginning and end tags in the precise location you want them.
- You already inserted two divs in the *default.html* page, and now you are going to insert the same divs in the *myDesk.html* page.

# Creating Divs in Code View Cont.

---

- With the *myDesk.html* page open in *Split* view, click the *Home* text button you created earlier to navigate both views to the top of the page.
- From the *Toolbox* panel, drag an instance of the `<div>` tag into *Code* view and place it directly under the `<body>` tag.
- This creates a beginning and an end div tag: `<div></div>`.
- Highlight and cut out the `</div>` end tag by pressing *Ctrl+X*.
- In *Code* view, navigate to the bottom of the page.
- There, you can see that the `</body>` tag is now red with a yellow background, indicating that the code is broken.
- Paste the `</div>` tag you just cut out into the line directly above the `</body>` tag.
- If you click an element within the page, you can see that the `<div>` tag is now present in the *Quick Tag Selector*.

# Creating Divs in Code View Cont.

---

- Add the sidebar to the page.
- Just like in the *default.html* page, the sidebar should appear alongside the content off the top, so in the markup it should appear right after the `<div>` you just created.
- Find the beginning div tag and add a new line directly underneath it.
- Drag and drop a new `<div>` tag into the new line or enter `<div>` manually.
- *IntelliSense* creates the end tag for you to keep the code from breaking.
- Again, highlight and cut out the end tag.
- Because this page doesn't have a descriptive section, the sidebar should contain only the *Home* link, so place your cursor at the end of the line containing the *Home* link and press *Enter* to create a new line.
- Paste the `</div>` end tag you cut out into this new line or enter `</div>`.
- The *myDesk.html* page now has two divs, just as the *default.html* page does.
- However, the classes and IDs you used to style the divs are still in the *default.html* file.
- To apply them to *myDesk.html* as well, you need to create an external style sheet.

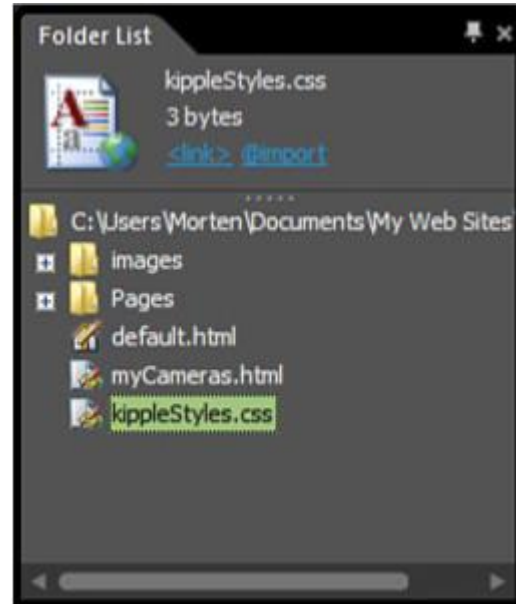
# Creating External Style Sheets

---

- An external style sheet is a dedicated file with the extension `.css` that contains only style code.
- Until now, Expression Web 4 has inserted all the style code you created into the head of your HTML pages, but doing so limits their application to that particular page.
- Now you need to move the styles from the HTML page to a new style sheet so that you can apply the same styles to multiple pages.
- To create an external style sheet, you first have to create a `.css` file.
- The easiest way to create a `.css` file is to click the down arrow next to the *New* icon on the *Common Toolbar* and select *CSS* in the context menu.
- This creates a new file named *Untitled\_1.css*.
- In most cases, the style sheet name is simply *styles.css*, but it is often a good idea to be more specific in naming to ensure that you know which site each sheet belongs to.
- After creating the new file, go to *File*, select *Save As*, and give it the name *kippleStyles.css*.
- When saved, it appears in the *Folder* panel.

# Creating External Style Sheets Cont.

---

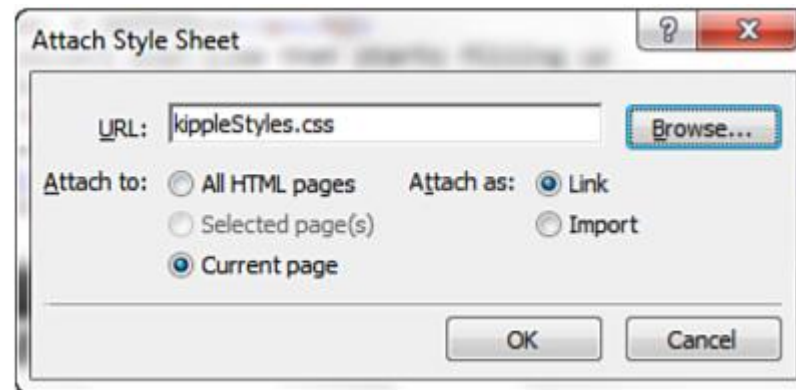


- Opening the new CSS file, you can see that it is completely empty.
- That is because unlike HTML and other markup files, a CSS file doesn't need any code other than the actual styles to function.
- Because it's a new file, there are no style definitions yet.

# Creating External Style Sheets Cont.

---

- The next step is to attach the style sheet to your pages by using the *Attach Style Sheet* button in the *Apply and Manage Styles* panel.
- With the *default.html* page open, click the *Attach Style Sheet* button to open the *Attach Style Sheet* dialog.



# Creating External Style Sheets Cont.

---

- From here, you can browse to the style sheet you want to attach and choose whether you want to attach it to all the pages in your site or just the current page. (The Selected pages option becomes available if you highlight a series of pages in the *Folders* panel before opening the *Attach Style Sheet* dialog.)
- You also have the choice of whether to attach the style sheet using the link method or the import method.
- They produce nearly the same results, but the Link option provides the most consistent results.
- Browse and select the *kippleStyles.css* file you just created.
- Select the option *Attach to All HTML Pages* and then select *Attach as Link*.
- This attaches the new style sheet to all the HTML pages within your site by inserting the following line of code in the *<head>* tag:
- `<link href="kippleStyles.css" rel="stylesheet" type="text/css" />`

# Creating External Style Sheets Cont.

- The attached style sheet now appears in the *Manage Styles* panel under the styles embedded in the current page.





# Moving Styles to and from the External Style Sheet

---

- After the external style sheet is attached to all the pages in your site, the styles set in the *kippleStyles.css* file affect all the pages instead of just one.
- You have already created many styles in different pages, but they are stored in the head of each page and not in the style sheet.
- The obvious way to solve this is to cut and paste the code out of the pages and into the style sheet, but this method is both cumbersome and prone to error.
- Expression Web 4 provides a better solution in the form of the *Manage Styles* panel.

# Moving Styles to and from the External Style Sheet Cont.

---

- With the *default.html* file open, click and drag the body style from the *Current Page* area down the *kippleStyles.css* area.
- When you let go, the style appears below the *kippleStyles.css* heading.
- Using the same method, move the rest of the styles, classes, and IDs from the *Current Page* area to the *kippleStyles.css* area.

# Moving Styles to and from the External Style Sheet Cont.



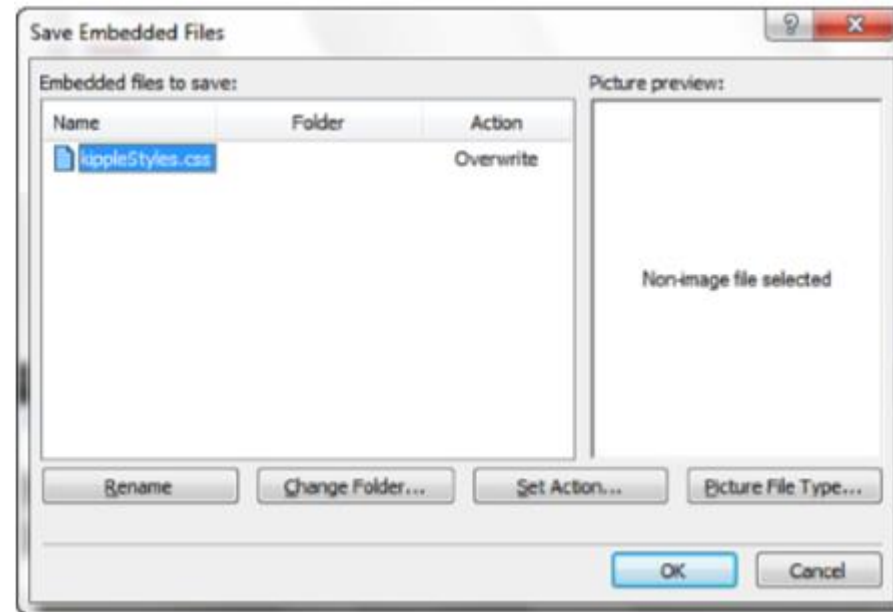
# Moving Styles to and from the External Style Sheet Cont.

---

- When you finish moving all the styles, classes, and IDs to the *kippleStyles.css* area, scroll to the top of *default.html* in *Code* view.
- Note that all the style code is gone.
- All that is left is the `<style>` tag.
- In the *Manage Styles* panel, right-click any of the styles and select *Go to Code*.
- The new *kippleStyles.css* style sheet opens, and you can see that all the code previously housed in the head of the HTML file is now in the style sheet.
- Open *myDesk.html*.
- Note that the styles you just moved from *default.html* now appear under *kippleStyles.css* in the *Manage Styles* panel for this page.
- Using the same technique, move the styles from *myDesk.html* to *kippleStyles.css*.

# Moving Styles to and from the External Style Sheet Cont.

- Press *Ctrl+S* to save the changes.
- This opens the *Save Embedded Files* dialog, which asks whether you want to save the changes to the *kippleStyles.css* file.
- Click *OK*.



# Moving Styles to and from the External Style Sheet Cont.

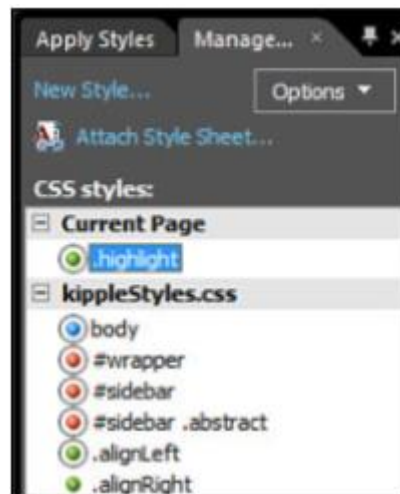
---

- The *kippleStyles.css* file now contains all the styles from both *default.html* and *myDesk.html*.
- Expression Web 4 still applies the styles to the content of those pages, and they appear the same in *Design* view and in any browser you preview them in.
- Inspecting the code in *kippleStyles.css*, you can see that the order of the styles corresponds with the list of styles in the *Manage Styles* panel.
- If you change the order of the styles in the panel, the code reorganizes in the same manner.
- This is because the order of styles in the style sheet is relevant to the cascade: The lower in the sheet the style is, the more weight it has in deciding what the content should look like.

# Moving Styles to and from the External Style Sheet Cont.

---

- In some cases a style, class, or ID applies to only one page.
- In that case, it is a good idea to keep the element in the relevant page rather than in the style sheet.
- For example, in *default.html*, you created a class called highlight that applied to two words as an inline style.
- Now, the style sheet contains this class, but because only the *default.html* page uses it, the class should be only in that file.
- To return this particular class to its original location, simply open *default.html* and drag and drop the *.highlight* class back to *Current Page*.



# Applying External Styles to a Page

---

- Styles nested in an external style sheet act in the same way as styles nested in the current document.
- Therefore, Expression Web 4 applies them in the same way.
- Earlier in this hour, you created two divs to section out the content in the *myDesk.html* page.
- Now that you have attached the external style sheet, you can apply the same IDs and classes you used to change the layout of *default.html* to change the layout of *myDesk.html*.



# Applying External Styles to a Page Cont.

---

- With the *myDesk.html* file open in *Split* view, click anywhere inside the page in *Design* view and then click the first `<div>` tag in the *Quick Tag Selector* to select the div that wraps all the content.
- This highlights all the content in both *Code* view and *Design* view.
- In the *Manage Styles* panel, right-click the `#wrapper` ID and select *Apply Style* from the context menu.
- The tag in the *Quick Tag Selector* and in *Code* view changes to `<div#wrapper>` and the wrapper ID is applied.

# Applying External Styles to a Page Cont.

The screenshot displays a web browser's developer tools interface. The top pane shows the HTML source code for a page named 'myCameras.html'. The code includes a link to an external CSS file 'kippleStyles.css' and a heading 'MY CAMERAS - A GROUP OF SNAPSHOTS LINKING ME TO MY PAST'. The bottom pane shows the rendered page with the heading and a photograph of several cameras. The right-hand pane, titled 'Apply Styles', shows a list of CSS styles from the 'kippleStyles.css' file. The 'body' style is selected, and a context menu is open over it, with the 'Apply Style' option highlighted. The 'Selected style preview' section at the bottom right shows the text 'AaBbYyGgLIJ' rendered in the selected style.

```
7 <style type="text/css">
8 </style>
9 <link href="kippleStyles.css" rel="stylesheet" type="text/css">
10 </head>
11
12 <body>
13 <div>
14 <div>
15 <p><a href="index.html" title="Go back to My Kipple">Home</a></p>
16 </div>
17 <h1>My Cameras - a group of snapshots linking me to my past</h1>
18 <p>
19 <map id="FPMap0" name="FPMap0">
20 <area alt="EOS 1 - the camera that went to war" coords="202, 5, 315,
21 <area alt="An old Nikon SLR" coords="22, 141, 37, 185, 78, 169, 97,
22 <area coords="103, 67, 120, 123, 146, 116, 166, 151, 194, 147, 207,
23 </area></map>
24 </p>
25 </div>
26 </div>
27 </body>
```

Home  
MY CAMERAS - A GROUP OF SNAPSHOTS  
LINKING ME TO MY PAST

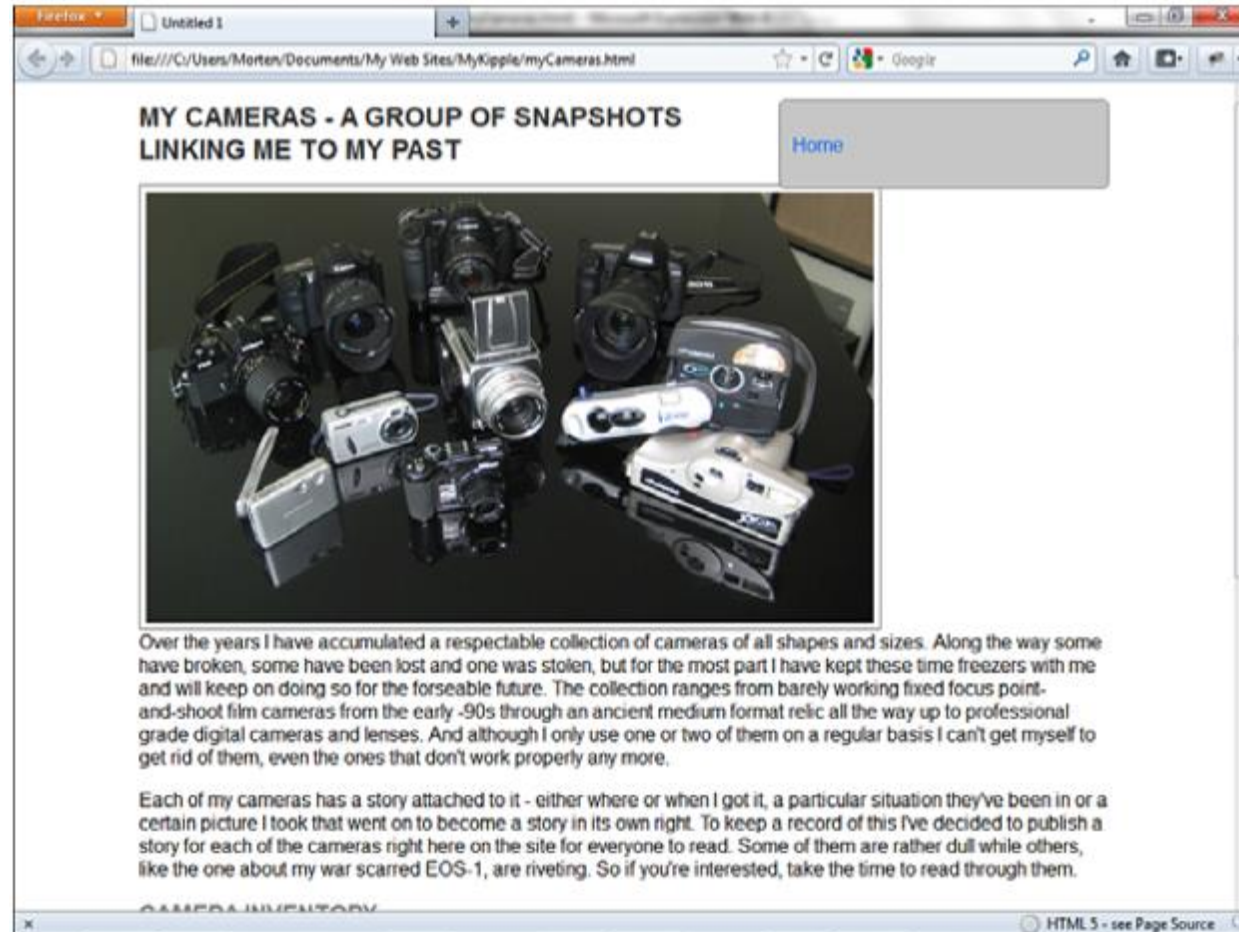
Selected style preview:  
AaBbYyGgLIJ

# Applying External Styles to a Page Cont.

---

- In *Code* view, move the cursor one line down to select the next `<div>` tag, right-click the `#sidebar` ID in the *Manage Styles* panel, and select *Apply Style*.
- The tag changes to `<div#sidebar>` and the `#sidebar` ID is applied.
- Because the position of the sidebar is no longer defined by the `#sidebar` ID, you also need to apply a class to the div.
- This action can also be done from the *Apply Styles* panel by selecting the relevant div (now `<div#sidebar>`) and clicking the appropriate alignment class (`.alignLeft` or `.alignRight`) in the *Apply Styles* panel.
- The tag changes to `<div#sidebar.alignRight>` and the alignment class is applied.
- Save the file and preview it in your browser.
- By previewing the page in a browser, you can see the styles you created for `default.html` applied to `myDesk.html`.

# Applying External Styles to a Page Cont.



# Images as Backgrounds: A Crash Course

---

- In Hours 6 and 7, you learned how to insert images into the content of your page.
- But as you saw, these images were content elements.
- What you want now are design elements, and that requires a somewhat different approach.
- As you have already learned, when you insert an image into a web page, you are actually inserting a replaced item, a link that is replaced with an external file.
- This same technique can be used to replace items such as CSS backgrounds, meaning that rather than giving your box a flat-color (or no-color) background, you can use the image of your choice as the background.
- Furthermore, you can control the way in which this image displays to achieve different effects.
- As with the content images, any image used as a background must be RGB and in one of the three main formats: GIF, JPEG, or PNG.

# Using an Image as a Background with CSS

---

- In the lesson files for this hour (Hour 13) is a series of image files.
- Before going any further, you need to create a new folder called *Graphics* and import these files into it using the technique you learned in Hour 6.
- In this first lesson, you apply an image as a background to see how you can easily change the look of an entire page, even with a small image file.

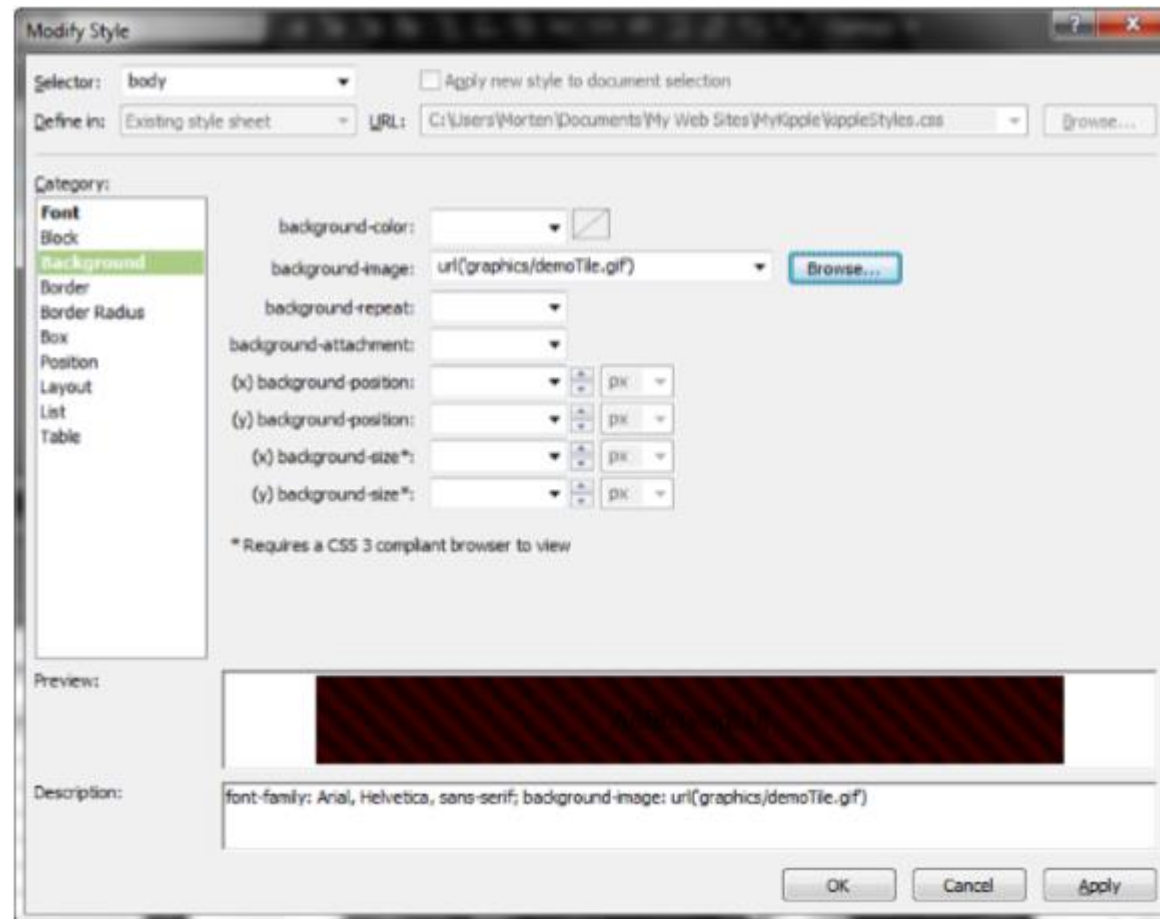
# Using an Image as a Background with CSS

## Cont.

---

- With the *default.html* page open in *Design* view, open the *Modify Style* dialog for the body style you created previously.
- In the *Background* category, use the *Browse* button next to the *background-image* attribute to navigate to the *Graphics* folder you just created.
- Select the file called *tile.gif* and click *OK*.
- In the *Preview* box, you see a red-and-black texture appear behind the text.
- Click *OK*.

# Using an Image as a Background with CSS Cont.





# Using an Image as a Background with CSS

## Cont.

---

- In *Design* view, you now see that the entire background of the page has been overtaken by a red-and-black texture.
- However, if you open the *tile.gif* file you just inserted as a background in your regular file explorer outside of Expression Web 4, you see that it is, in fact, just a 20-pixel-by-20-pixel image.
- So, how is it covering the entire background?
- By default, if you set an image as a background for any style, the image automatically repeats or tiles both horizontally and vertically from the uppermost-left corner down to the end of the page.
- This gives the designer the ability to use a tiny image to cover a large area.
- Otherwise, the background image would need to be as wide as the display on the largest monitor imaginable and as tall as the longest page you could come up with, and that would be a very large image indeed.

# The Background Attributes

---

- To see how this tiling works in real life, you can turn it off.
- To do so, go back to the *Background* category of the body style and set the *background-repeat* attribute to no-repeat.
- Now, only one instance of the background appears in the upper-left corner.
- The *background-repeat* attribute can be set to one of four values that give you a variety of options in terms of how to display your background image.
- You've already seen what *no-repeat* does; *repeat* is the default setting by which the image is tiled both horizontally and vertically; *repeat-x* tiles the image only along the *x-axis*, meaning there will be one line of tiles along the top going from left to right; and *repeat-y* tiles the image only along the *y-axis*, meaning there will be one line of tiles along the left side going from top to bottom.

# The Background Attributes Cont.

Background  
image



# The Background Attributes Cont.

---

- You can further specify the location of the background image using the *(x) background-position* and *(y) background-position* attributes, in which you can give the background image an absolute position with a numeric value or you can set it to left, center, or right for the x value and top, center, or bottom for the y value.
- For example, by setting both the *(x) background-position* and *(y) background-position* attributes to center, the image will be located in the middle of the area relative to the total width and height of the area within the tags.
- The Background category also lets you set the *background-attachment* attribute for the image.
- The default value of this attribute is scroll, meaning that the background is affected if you scroll up, down, left, or right in the page, just like the rest of the content.
- If you set this attribute to fixed, the background image is fixed in the same location relative to the browser window, so the image stays even if you scroll through the content.
- X and Y *background-size* attributes let you resize the background image to a specific size or a percentage. This is a CSS3 feature and only works in some browsers.

# The Background Attributes Cont.



Background  
image

# The Background Attributes Cont.

---

- The *background-color* attribute comes into play whenever you use a *background-repeat* value other than default or repeat and when the image does not display for whatever reason.
- All the area not covered by the background image has the color defined by the *background-color* attribute.

# Applying a Background Image to an ID

---

- To fully understand how background images in CSS work, you need to see how they relate to the content and to each other.
- In this example, you apply two background images: one to the body style and one to a new ID you are going to create to contain the individual page content.

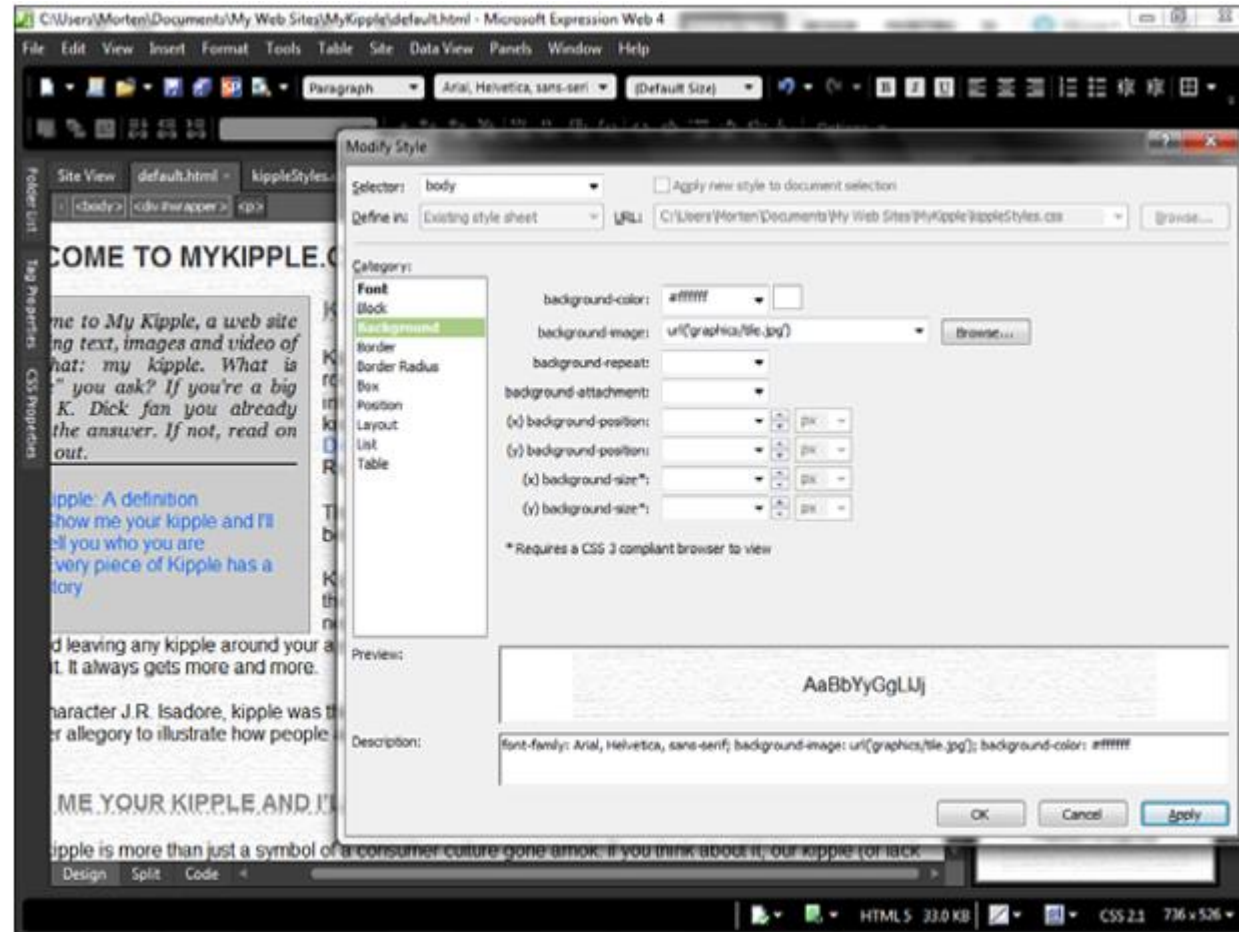
# Applying a Background Image to an ID Cont.

---

- Open the *Modify Style* dialog for the body style and change the background image to the file *backgroundTile.jpg*.
- Remove all the other styling to ensure that the image tiles throughout the entire page.
- The new image is a gray-and-white texture on which the page will be built.
- At the same time, set the *background-color* to *#FFFFFF* (white) to ensure that the page looks great even if images are disabled.



# Applying a Background Image to an ID Cont.



# Applying a Background Image to an ID Cont.

---

- To separate the header area of the page (where the title and menu will go) from the rest of the content, you need to wrap the main content in a new ID.
- In *Split* view, find the line wrapped in `<h1>` tags and create a new line directly below.
- Type `<div id="mainContent">` and *IntelliSense* creates the end `</div>` tag for you.
- Select and cut out the end `</div>` tag, scroll to the bottom of the page, and paste it in just before the end `</body>` tag.
- Now, the content of the page is wrapped separately and can be styled separately.

# Applying a Background Image to an ID Cont.



```
14 padding: 1px 4px;
15 }
16 </style>
17 <link href="kippleStyles.css" rel="stylesheet" type="text/css">
18 </head>
19 <body>
20 <div id="wrapper">
21 <h1>Welcome to MyKipple.com</h1>
22 <div id="mainContent"></div>
23 <div id="sidebar" class="alignLeft">
24 <p class="abstract">Welcome to My Kipple, a web site featuring text,
25 that: my kipple. What is "kipple" you ask? If you're a big Philip K.
26 you already know the answer. If not, read on to find out.</p>
27 <ul>
28 <li><a href="#Kipple: A definiton" title="Jump to Kipple: a defi
29 Kipple: A definition</a></li>
30
```

er

## COME TO MYKIPPLE.COM

me to My Kipple, a web site  
ing text, images and video of  
hat: my kipple. What is  
:" you ask? If you're a big  
K. Dick fan you already  
the answer. If not, read on  
out.

ipple: A definition  
show me your kipple and I'll  
ll you who you are.

### KIPPLE: A DEFINITON

Kipple is all the useless stuff we collect over  
rooms, houses and eventually our lives. The  
introduced by science fiction author **TERRY**  
known with the release of Philip K. Dick's  
[Dream of Electric Sheep?](#) – the book that  
Runner.

The best explanation for what "kipple" is ca  
between two of the characters in the book:

Design Split Code

# Applying a Background Image to an ID Cont.

---

- Create a new style with the ID *#mainContent* to match the `<div>` you just created.
- Make sure it is defined in the *kippleStyles.css* style sheet.
- In the *Background* category, use the *Browse* button to set *background-image* to *paperTile.png*.
- Set the *background-repeat* attribute to *repeat-y* so that the image tiles only vertically.
- When you click *Apply*, you see that the background image has an edge to it and that the text is currently overflowing it.
- To realign the text so it fits on the inside of the graphic, go to *Box* and set *padding-right* and *padding-left* to *15px*.
- Click *OK* to apply the changes; then save and preview the page in your browser.

# Applying a Background Image to an ID Cont.



# Applying a Background Image to an ID Cont.

---

- As you can see, the backgrounds are relative to the tag to which they are applied; the body background covers the entire page regardless of how wide or tall the window is, whereas the *#mainContent* background is constrained inside the div.

# Background Images and the Box Model

---

- To see how the background images relate to the box model, open the *Modify Styles* dialog for the *h2* style, set *background-image* to *underline.png* and *background-repeat* to *no-repeat*.
- Click *OK* to apply the style change.
- By looking at the headings in *Design* view, you see that they now have a gray line behind them toward the top.
- However, there are two problems: First, the image is called “*underline*,” so it is obviously meant to be an underline, and second, the image is missing from the top *h2* title.
- If you click the top title and look at the tag selector box, it is obvious why you can’t see the underline graphic behind the text: Even though the sidebar box pushes the text to the side, the *h2* box still stretches to the far-left side of the page and the background with it.
- To solve this problem, you need to change the display attribute for the *h2* style.
- By default, all headings have the display attribute set to *block*, meaning they appear on their own line and span the width of the box in which they are placed (in this case, the *#mainContent* div).
- To make the box shift to the side to accommodate the sidebar, open the *Modify Style* dialog for the *h2* style, go to *Layout*, and set *display* to *inline-block*.

# Background Images and the Box Model Cont.





# Background Images and the Box Model

## Cont.

---

- The first problem is less cryptic and can be solved with some common-sense thinking: To move the line to the bottom of the box, simply set the *(y) background-position* to *bottom* and the *(x) background-position* to *left*.
- However, even so, the line still appears behind the text, not under it.
- This is where the box model comes into play.

# Background Images and the Box Model Cont.

---

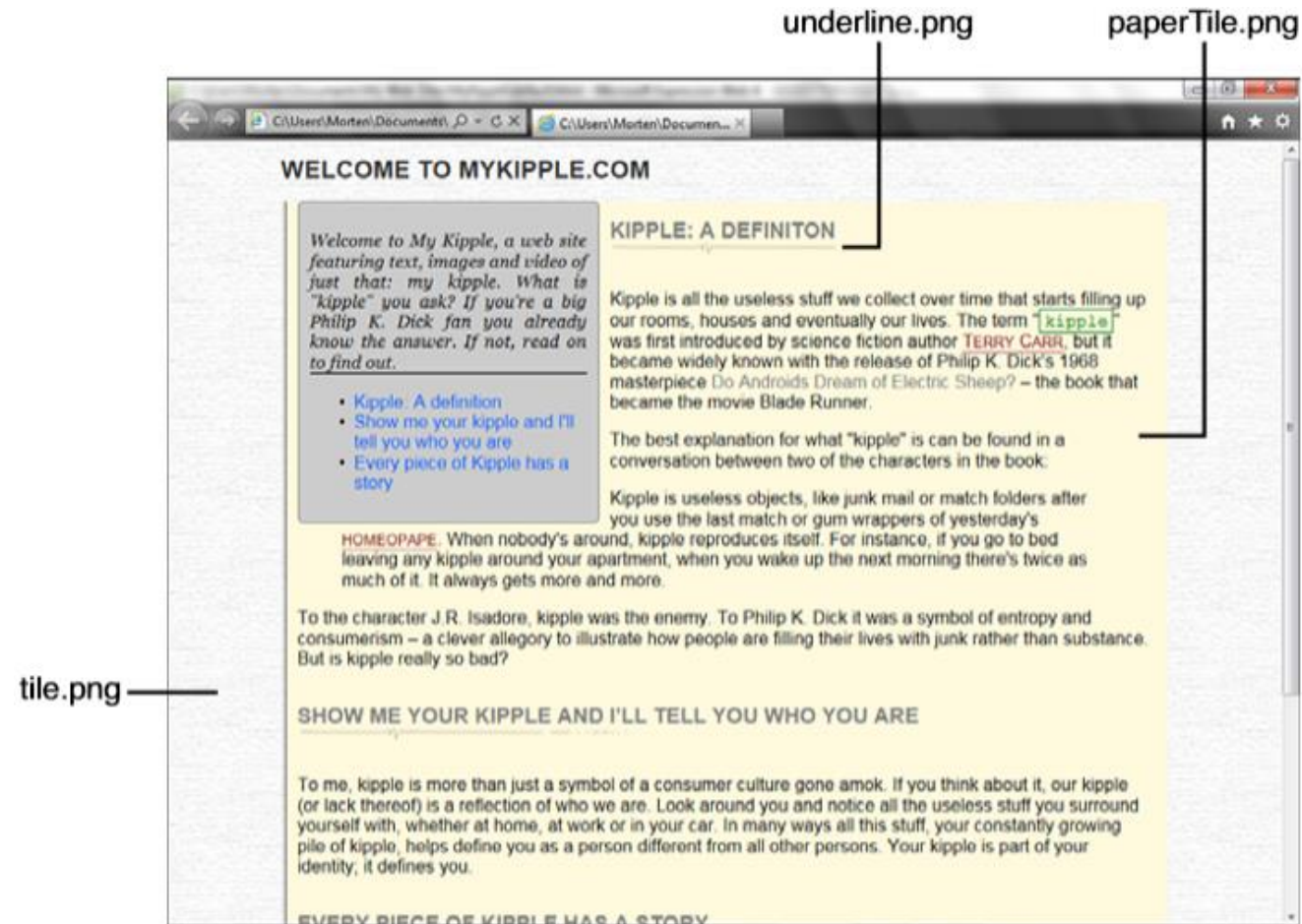
- In this case, you want the background image to appear below the text content of the *h2* headings.
- You set the (*y*) *background-position* attribute to *bottom*, which means the image hugs the bottom of the box.
- Therefore, if you increase the bottom padding, the image shifts down relative to the text.
- To do so, open the *Modify Style* dialog for the *h2* style and go to the *Box* category.
- The next step is a bit of trial and error: Uncheck the *Same for All* box for *Padding* and then set *bottom-padding* to *5px*.
- Click *Apply* and move the *Modify Style* dialog to the side to see whether this produced the result you were looking for.
- In this case, the image didn't move far enough down, so use the up and down buttons to increase the *bottom-padding* value to *10px* and click *Apply* again.
- If you are satisfied with what you see, click *OK* to finalize the change.

# Stacking Order Means You Can Pile Your Images

---

- Because background images are style elements contained within the box model, they act just like any other feature in the box.
- The most obvious advantage of this is that it gives you the ability to stack multiple images on top of each other, just like a pile of photos, by setting them as backgrounds for different elements.
- This method also preserves any transparencies (if the browser supports them), meaning you can place one transparent image on top of another one and thus see one through the other.
- In fact, using this method, there is no limit to how deep your stack of images can be, except the capabilities of the computer that displays them.
- As with all other style elements, the cascade decides the stacking order of images used as backgrounds: The tag closest to the content has top position, followed by the second closest, and so on.
- This way, the background of the most relevant object ends up on top.
- While you were following the lessons earlier in this hour, what you were actually doing was stacking one image on top of another.
- When you test the current page in your browser and look at the *h2* headings, you see the *underline.png* image stacked on top of the *paperTile.png* image, which in turn is stacked on top of the *tile.jpg* image.
- And as you can see, the transparencies of the two topmost images stay intact throughout the page.

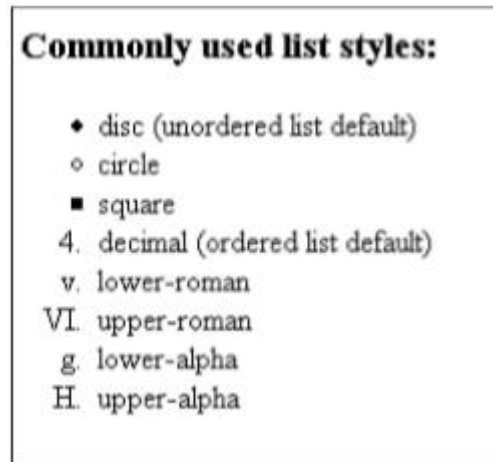
# Stacking Order Means You Can Pile Your Images Cont.



# Using Images as List Bullets

---

- By default, a bullet precedes all unordered list items.
- However, this is but one of many different options.
- The figure below shows how the eight main list styles appear in a browser when no additional styling has been applied.
- But for all their convenience and functionality, these items are basic and exchanging them for images is a quick-and-easy way to bring your design to that next level.

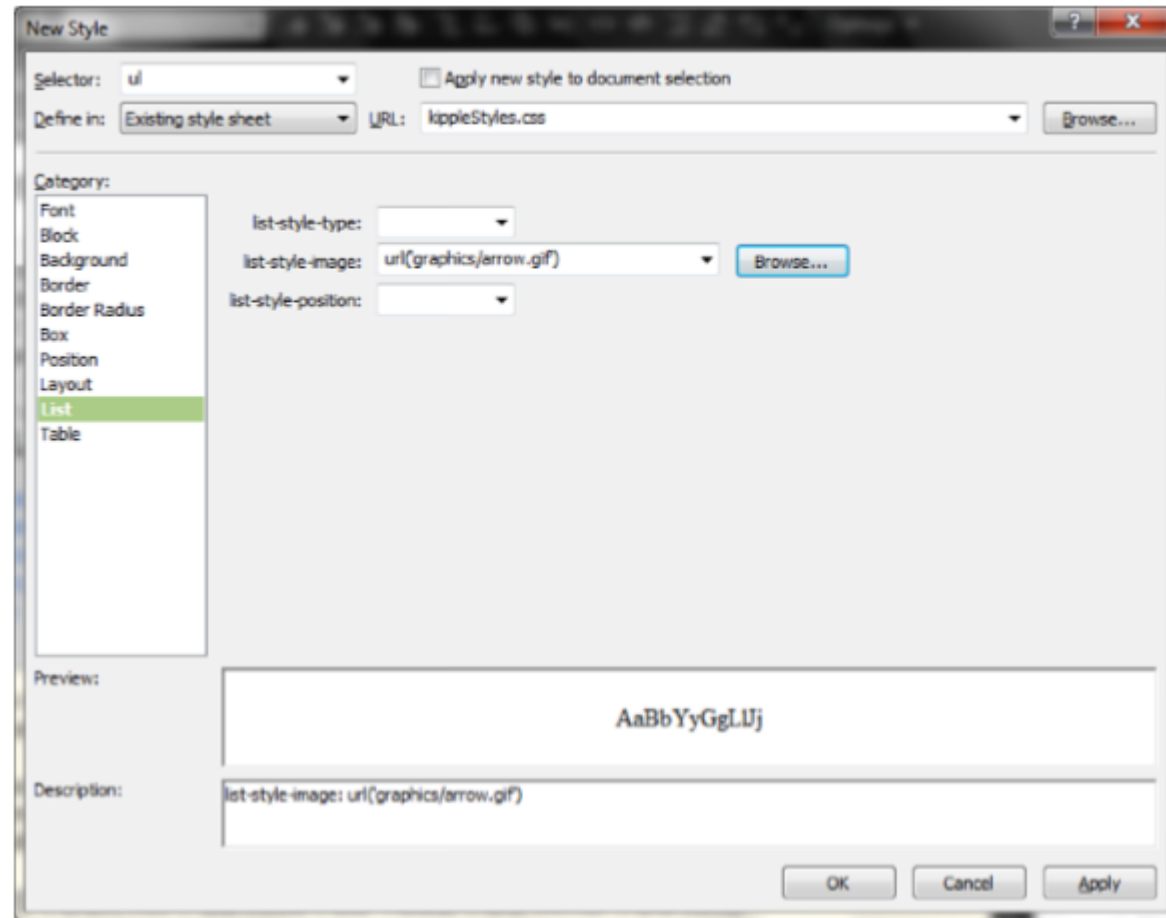


# Using Images as List Bullets Cont.

---

- Switching out the bullet for an image is similar to changing the bullet style.
- Because list items are a special type of content in HTML, there is a separate section for them in the *New and Modify Style* dialogs.
- With the *default.html* page open in *Design* view, create a new style with the selector name *ul*.
- This style affects all list items in unordered lists.
- Make sure *Define In* is set to *Existing Style Sheet* and *kippleStyles.css* is the *URL*.
- In the *List* category, use the *Browse* button to select the *listArrow.gif* file.
- Click *OK* to create the new style.

# Using Images as List Bullets Cont.



# Using Images as List Bullets Cont.

---

- The new style you made replaces the bullets in front of the list items with small arrows.
- As with the backgrounds, any image file can be used as a list bullet. (Although, for obvious reasons, I advise you to use small ones.)
- However, unlike the backgrounds, the bullet image is not directly related to the box but rather to the first line of text in each item.
- In practical terms, this means that if you add left padding, the distance between the bullet image and the text increases, but if you add bottom padding, the bullet stays on the line.
- This situation can produce some strange results if you use a large font size because the bullet image stays in its original position in relation to the first line of text rather than grow with the font size.
- The figure presented in the following slide demonstrates how both regular bullets and bullet images react to different styling of the *li* tag.
- Note that standard HTML bullets size according to the font size.



# Using Images as List Bullets Cont.

---

- The *List* category in the *New and Modify Style* dialogs also includes the *list-style-position* attribute.
- It can be set to either inside or outside. (outside is default.)
- This attribute describes the position of the bullet in relation to the box.
- If it is set to outside, the bullet or bullet image appears outside of the box.
- If it is set to inside, the bullet or bullet image appears inside the box.
- This value does not change the actual distance between the bullet and the text, however.
- As you can see, the distance remains the same and is changed with the *padding-left* attribute.
- The difference between the two becomes apparent when the list item has more than one line of text: If *list-style-position* is set to *inside*, the bullet is treated as if it is part of the text, and the second line of text lines up with the edge of the box underneath the bullet.

# Using Images as List Bullets Cont.

Regular bullets in relation to list items	Image bullets in relation to list items
<ul style="list-style-type: none"><li>• list-style-position: outside (default)</li></ul>	<ul style="list-style-type: none"><li>➤ list-style-position: outside (default)</li></ul>
<ul style="list-style-type: none"><li>• list-style-position: inside</li></ul>	<ul style="list-style-type: none"><li>➤ list-style-position: inside</li></ul>
<ul style="list-style-type: none"><li>• list-style-position: outside second line of text</li></ul>	<ul style="list-style-type: none"><li>➤ list-style-position: outside second line of text</li></ul>
<ul style="list-style-type: none"><li>• list-style-position: outside second line of text</li></ul>	<ul style="list-style-type: none"><li>➤ list-style-position: outside second line of text</li></ul>
<ul style="list-style-type: none"><li>• list-style-position: outside; padding-left: 10px</li></ul>	<ul style="list-style-type: none"><li>➤ list-style-position: outside; padding-left: 10px</li></ul>
<ul style="list-style-type: none"><li>• list-style-position="inside"; padding-left: 10px</li></ul>	<ul style="list-style-type: none"><li>➤ list-style-position="inside"; padding-left: 10px</li></ul>
<ul style="list-style-type: none"><li>• list-style-position: outside; padding-bottom: 10px</li></ul>	<ul style="list-style-type: none"><li>➤ list-style-position: outside; padding-bottom: 10px</li></ul>
<ul style="list-style-type: none"><li>• small font size</li></ul>	<ul style="list-style-type: none"><li>➤ small font size</li></ul>
<ul style="list-style-type: none"><li>● large font size</li></ul>	<ul style="list-style-type: none"><li>➤ large font size</li></ul>

# Importing Adobe Photoshop Files

---

- The term web design is a somewhat confusing one, especially when talking about the tools used.
- You can question whether the actual design of a page or site takes place in web-authoring software, such as Expression Web 4, or in other applications, with Expression Web 4 and similar programs merely used to build the framework to display the design.
- Regardless of where you stand in this debate, you cannot dismiss one fact: Pure design applications, such as Expression Design and Adobe Photoshop, play an integral part in creating visually stunning websites.
- Expression Web 4 has a dedicated function for importing Adobe Photoshop (*.psd*) files.
- And because *.psd* files usually contain several layers, you are given the ability to select which layers to include in the import so that you can produce several different images out of one file.

# Importing Adobe Photoshop Files Cont.

- With the *default.html* page open in *Design* view, click *File* on the menu bar and select *Adobe Photoshop (.psd)* under *Import*.
- This opens a browser dialog where you select the *.psd* file you want to import.
- Select the file named *myKipple sticker art.psd* from the lesson files and click *Open*.
- This opens the *Import Adobe Photoshop (.psd) File* dialog.



# Importing Adobe Photoshop Files Cont.

---

- By default, the *Check All Layers to Import* option is checked, meaning the output will be the image as it appears with all the layers activated.
- By unchecking individual layers, you can select which layers to include in the final export.
- Although you can't see it, the total area covered by the exported image changes with the content you choose to export, so if you uncheck the *Background* layer, the image size shrinks to contain only the remaining content.
- When you are satisfied with your layer selection, you can choose whether to export the image as a GIF, JPEG, or PNG from the *Encoding* drop-down menu.
- If you choose *JPEG*, you have the further option of deciding how much compression to apply (lower quality means higher compression).
- Sliding the bar back and forth gives you a real-time preview of what the different compression settings do to the image.
- Choose the lowest possible quality that doesn't produce artifacts that ruin the image.
- For this particular image, you can go as low as 33% without losing much in terms of quality.

# Importing Adobe Photoshop Files Cont.

---



# Importing Adobe Photoshop Files Cont.

---

- With the *Background* layer unchecked, set the *encoding* to *PNG* and click *OK*.
- This opens a standard *Save As* dialog.
- Give the compressed image file the name *MyKippleSticker.png* and place it in the *Images* folder.
- The image is added to your file tree in the *Folder List* task pane.
- In the *default.html* file, delete the heading and then drag and drop the new image into the *h1* box instead.
- Give it the alternative text *MyKipple.com sticker*.
- Save the file and preview the page in your browser.

# Importing Adobe Photoshop Files Cont.





# Importing Adobe Photoshop Files Cont.

---

- After placing the image on your page, you might find that it doesn't quite work with the rest of the elements, or maybe you need to omit more layers or even change another element within the image.
- Because you created the image using the Photoshop import option, Expression Web 4 retains a link to the original *.psd* file, so if you want to make changes to the image, you can do so directly from *Design* view.
- When you right-click an image imported from a *.psd* file, a new item called *Adobe Photoshop (.psd)* appears in the drop-down menu.
- From this submenu, you can choose *Edit Source* to open the original *.psd* file in whatever software is set as your default *.psd* editor (probably Photoshop), or you can choose *Update from Source* to reopen the *Import Adobe Photoshop (.psd) File* dialog.
- From here, you can perform all the same functions as when you first imported the image, and those changes will be applied to the imported file.
- If you have made changes to the *.psd* file outside of Expression Web 4 and want to update the image in the page to reflect these changes, click *Import* again.

# Starting with Pen and Paper

---

- This might come as a bit of a surprise, but it is often a good idea to start designing a website by sketching it out on a piece of paper.
- Not only is a sketch faster and easier to change than any other design method, it also gives you a blueprint of sorts to go by when you start building the framework to display the content of your site.
- The benefit of starting with a sketch is that you can see almost right away whether the overall layout works and, if it does, what sections you need to define to make it work.
- As you learned previously in this book, creating layouts using CSS means creating boxes within boxes within boxes, and you need to know and understand the relationship between these boxes before you build them.

# Starting with Pen and Paper Cont.

---



# Starting with Pen and Paper Cont.

---

- The layout has several different elements that need to be treated separately: the corkboard background that hovers in the center behind the content, the header (containing the page title and main menu), the pageContent (containing all the text and images of the respective page including the sidebar), and the footer.
- All the page content is centered and should, therefore, be placed in a separate container.
- From this information, you can draw a set of boxes to indicate how to separate the content.

# Starting with Pen and Paper Cont.



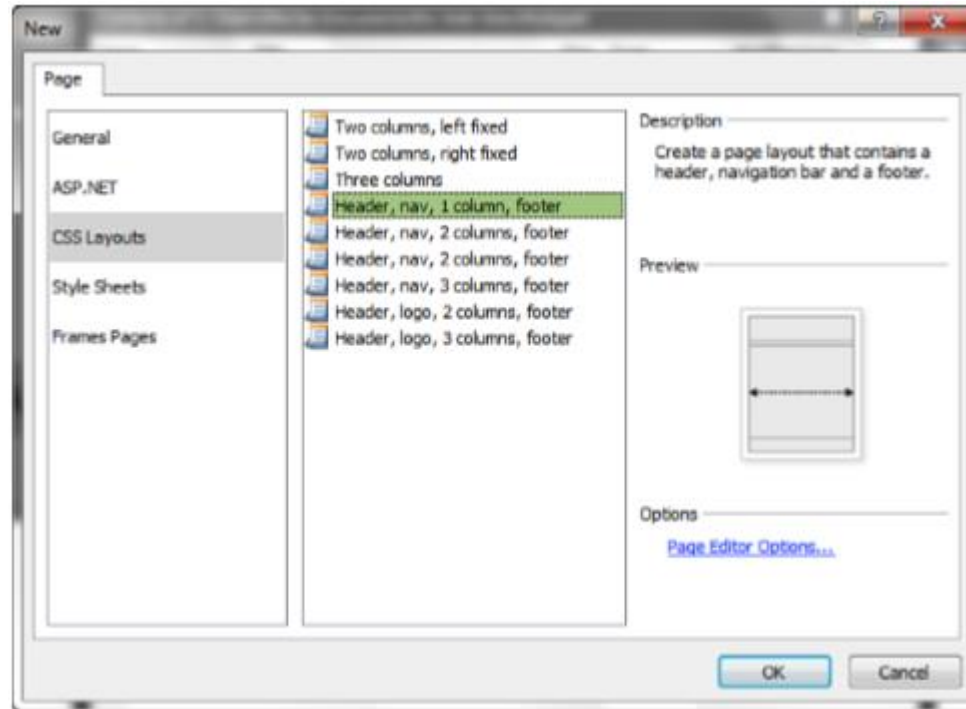
# Building the Framework from Boxed Parts

---

- Many designers prefer to build the framework by hand, but it can be nice to get some help if you are new at design.
- Expression Web 4 has a series of ready-to-use, prepackaged CSS layouts that give you a bit of a head start.
- Click *New, Page* from the *File* option on the menu bar.
- This opens the *New* dialog with the *Page* tab selected.
- Click *CSS Layouts* in the first list to open the prepackaged CSS layouts in Expression Web 4.
- By clicking each of the options in the second list, you get a short description of the layout along with a preview.

# Building the Framework from Boxed Parts Cont.

---



# Building the Framework from Boxed Parts Cont.

---

- Select the layout closest to the framework you drew in your sketch.
- In this case, it is the *Header, Nav, 1 Column, Footer* design.
- Select this option and click *OK*.
- After you click *OK*, Expression Web 4 opens two new files: *Untitled\_1.css* and *Untitled\_1.html*.
- Both are empty except for the layout styles.
- This gives you the ability to work with the layout boxes without content and to match the overlay drawings you previously created.
- Because you already have a series of styles defined for your pages, what you do is create a separate style sheet that contains only the layout portions of the pages (so that you end up with two style sheets: one for content styles, one for layout styles).
- That way, you can make quick changes to the layout without touching all the other styles.



# Employing CSS Reset

---

- To apply the CSS reset, simply copy the code in its entirety from the web page and paste it into the top of the *Untitled\_1.css* file you just created.
- Save the CSS file as *layout.css*.
- Because the *Untitled\_1.html* file already links to the CSS file, the link updates automatically.
- Save the *Untitled\_1.html* file as *layoutTest.html*.

# Updating the ID Names

---

- The next step is to change the ID names to match your drawing.
- You can do so directly in the CSS file or using the *Modify Style* option from the *Manage Styles* task pane.
- Change the name of *#masthead* to *#header*, *#top-nav* to *#mainMenu*, and *#page-content* to *#pageContent*.
- Leave *#footer* as it is.
- In *layoutTest.html*, go to *Split* view and make the same changes to the div IDs.

# Updating the ID Names Cont.

---

- The mainMenu ID should reside within the header ID.
- Go to *Code* view and move the `</div>` end tag for the header ID below the one for the mainMenu ID—now the header div wraps the mainMenu div.
- The drawing also calls for two new IDs: *#centeredBG* and *#wrapper*.
- To wrap all the existing IDs, create a new `<div>` on the line before the first div and give it the ID wrapper.
- Then, create another box to contain the *#wrapper* ID and give it the *#centeredBG* ID.

# Updating the ID Names Cont.

---

- With all the changes made, the page's code inside the `<body>` tags should look like this (comments added to make it easier to read):

```
<body>
  <div id="centeredBG">
    <div id="wrapper">
      <div id="header">
        <div id="mainMenu">
          </div> <!-- #end mainMenu -->
        </div> <!-- #end header -->
      <div id="pageContent">
        </div> <!-- #end pageContent -->
      <div id="footer">
        </div> <!-- end #footer -->
      </div> <!-- end #wrapper -->
    </div> <!-- end #centeredBG -->
  </body>
```

- Now that you have inserted calls to the `#centeredBG` and `#wrapper` IDs, you need to add these styles in the `layout.css` file.
- You can do this manually in *Code* view with the help of *IntelliSense* or by using the *New Style* button in the *Manage Styles* pane.
- For now, just create the styles without any style code.

# Styling the Layout Boxes

---

- With the layout boxes created, it is time to style them to make the page match the sketch.
- This requires the use of all the techniques you learned in earlier hours and some new ones.
- The goal here is to remove all the layout styling from the *kippleStyles.css* file and store it in the new *layout.css* file.
- You can choose to make the following style modifications using the *Modify Style* dialog, directly in the *layout.css* file using *IntelliSense*, or both.

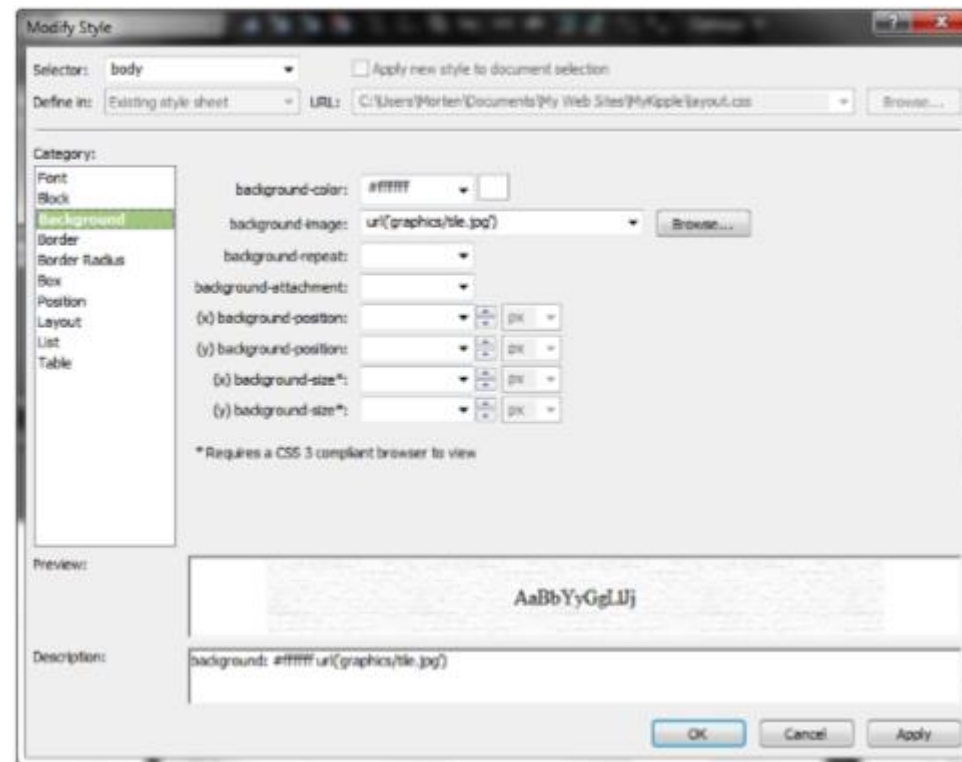
# Styling the Layout Boxes Cont.

---

- You already did this step in Hour 13, but there is no harm in repeating it:
- The layout drawing calls for a tiled graphic background that covers the entire page.
- In the new page *layoutTest.html*, create a new style with the selector *body*, set *background-image* to the *tile.jpg* file found in the *Graphics* folder, and set *background-color* to *#FFFFFF*.
- Note that you want to create a new *body* style even though the CSS reset already has one.
- Always leave the CSS reset code alone.

# Styling the Layout Boxes Cont.

---



# Styling the Layout Boxes Cont.

---

- The content of the page hovers in the middle of the page with a specific width.
- Center the content by setting the *#wrapper* ID *margin-left* and *margin-right* attributes to *auto*.
- Set the *width* attribute to *810px*.
- The design calls for a background image that spills outside of the edges of the wrapper.
- This is a popular effect that is heavily used in blog design.
- The premise is that you have a background graphic that spills beyond the outer-left and outer-right edges of the content yet moves with the content when the window is resized.
- This is achieved using the CSS attribute *min-width*, which is not available in the *New* and *Modify Style* dialogs.



# Styling the Layout Boxes Cont.

---



# Styling the Layout Boxes Cont.

---

- To achieve the hovering centered background image effect, first import the *headerBack.png* file from the lesson files for this hour and place it in the *Graphics* folder.
- In *Code* view of the *layout.css* file, find the *#centeredBG* style and set the *background-image* attribute to the *headerBack.png* file with the help of *IntelliSense*.
- Set *background-repeat* to *no-repeat* and *background-position* to *center* and *top*.
- Now the *headerBack* image hovers in the center of the screen behind the content, even when the window is resized.
- Unfortunately, it insists on hovering in the center also when the window is smaller than the width of the image, which will ruin the effect.
- To prevent this, set *min-width* to the exact width of the image, which in this case is *1000px*.
- This tells the browser that the *#centeredBG* box should not be reduced to a width less than 1,000 pixels.

```
61
62 #centeredBG {
63     background-image: url('graphics/corkboard.png');
64     background-repeat: no-repeat;
65     background-position: center top;
66     min-width: 1000px;
67 }
```

# Styling the Layout Boxes Cont.

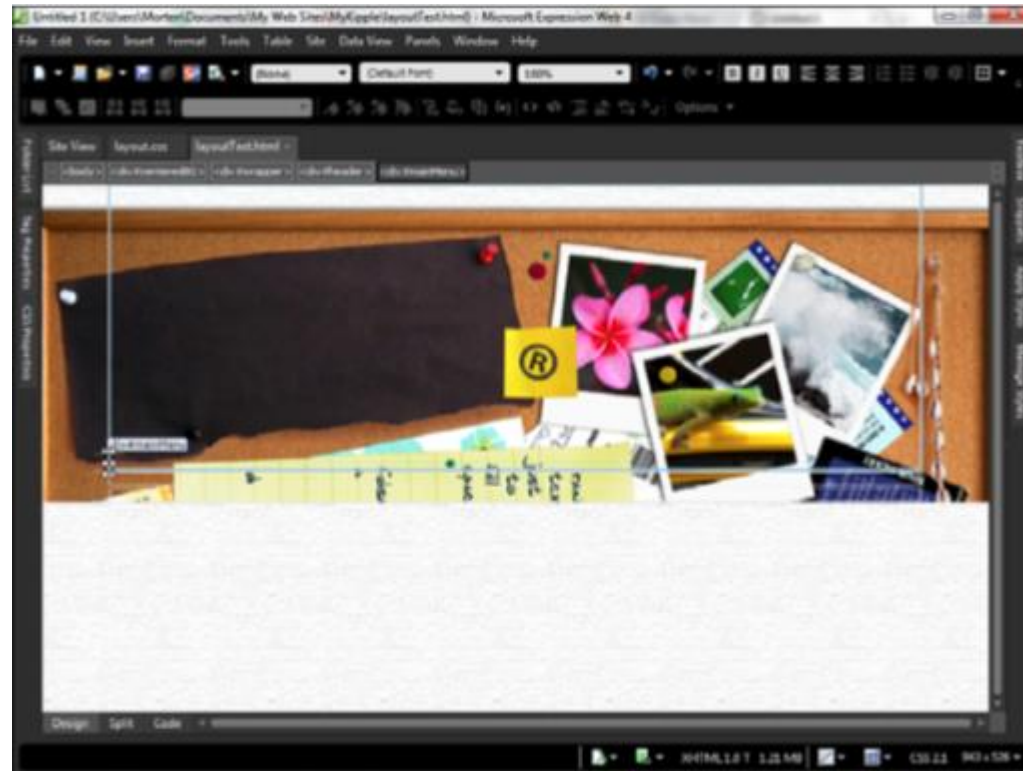
---

- The header of the page has to be set to a fixed height to ensure that the relationship between the header and the main content remains the same regardless of the size of the main content.
- To do this, set the *#header* style height to *285px*.
- On the drawing, the *#mainMenu* ID is along the bottom and aligned to the left of the *#header* box.
- To make this happen, you need to make some changes to the position attributes of both *#header* and *#mainMenu*.
- First, change the position of the *#header* ID to relative.
- Then, set the position attribute for the *#mainMenu* ID to absolute and set the bottom and left attributes (under the *Position* category in the *Modify Style* dialog) to *0px*.

# Styling the Layout Boxes Cont.

---

- You now have the basic framework for the page as it appears in the drawing.
- And all this was done using only CSS, which means the HTML markup has not changed.



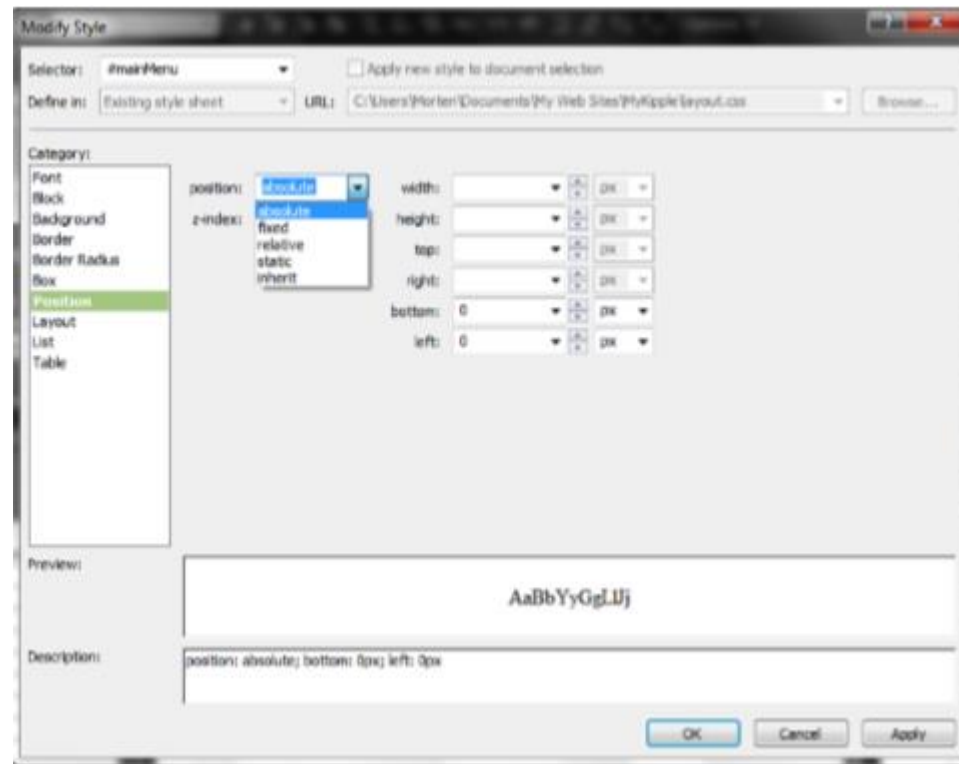
# Understanding Positioning

---

- In the last part of the preceding example, you used the *position* attribute to place a div in the lower-left corner of another div.
- This is a nice segue into the confusing and often misunderstood issue of positioning in CSS.
- If you open the *Modify Style* dialog for any of the current styles, classes, or IDs, you see that the position attribute (found under the Position category) has five options: *absolute*, *fixed*, *relative*, *static*, and *inherit*.
- The physical position of any object in a page depends on what this attribute is set to in the current style and whatever style that wraps it.

# Understanding Positioning Cont.

---



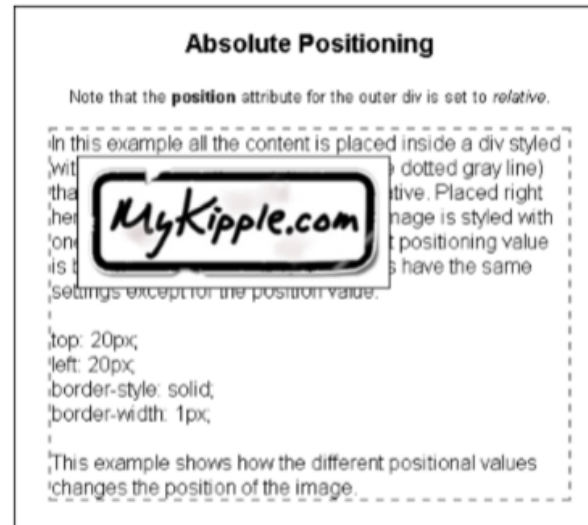
# Understanding Positioning Cont.

---

- ***position: absolute;***
- The easiest way to explain an element with an absolute position is to think of it as an image (or any other object) pasted onto a page and ignore the rest of the content.
- The physical placement of an element with an absolute position is decided by setting the pixel value of its distance from the nearest positioned container with the *top*, *right*, *bottom*, and *left* attributes.
- In other words, an object with absolute position that has a top value of 20px and a left value of 30px appears exactly 20 pixels from the top and 30 pixels to the left of the edge of the page or the closest container box that has a *position* value other than *static*.
- Setting an object's *position* attribute to absolute removes it from the flow of the page.
- That means unless you pay close attention, you might accidentally place objects with absolute positions directly on top of other content.

# Understanding Positioning Cont.

- In the *layoutTest.html* page, the *#menu* div has an absolute position of 0 pixels from the bottom and 0 pixels from the right side of the *#header* div because the *#header* position is set to relative.
- If you change the position attribute of *#header* to static, the *#menu* div is positioned absolutely in relation to the nearest parent with a position other than *static* (in this case, the body), which means it aligns itself with the edge of the page and ends up in the lower-right portion of the window.
- If you set the position attribute of a style, class, or ID to absolute without setting values for *top*, *right*, *bottom*, and *left*, the object appears in the default upper-left corner position.






# Understanding Positioning Cont.

---

- *position: fixed;*
- Fixed positioning works similarly to absolute positioning except that whereas the physical position of an object with an absolute position can relate to other positioned objects, the physical position of a fixed object is always based solely on the outer edges of the page as a whole.



### Fixed Positioning

Note that the **position** attribute for the outer div is set to relative.

In this example all the content is placed inside a div styled with an ID called #wrap (marked by the dotted gray line) that has its position attribute set to relative. Placed right here inside the text is an image. The image is styled with one of four classes depending on what positioning value is being demonstrated. All four classes have the same settings except for the position value:

```
top: 20px;
left: 20px;
border-style: solid;
border-width: 1px;
```

This example shows how the different positional values changes the position of the image.

# Understanding Positioning Cont.

---

- *position: relative;*
- The easiest way to explain relative positioning is to imagine that you have cut an image out of a printed page and repositioned it somewhere else on the page.
- Because you cut out the image from the page, there is a hole where it was, and the image covers content wherever you glue it.
- Placement of an object with a relative position is in relation to its original location in the flow of the page.
- As an example, that means an image with its position attribute set to *relative* and its *bottom* attribute set to 20px appears 20 pixels above its normal location where it was originally inserted, and the space it would have occupied remains empty.


# Understanding Positioning Cont.

---

**Relative Positioning**

Note that the **position** attribute for the outer div is set to *relative*.

In this example all the content is placed inside a div styled with an ID called #wrap (marked by the dotted gray line) that has its position attribute set to relative. Placed right



here side the text is  
an image our classes  
depending on what positioning value is being  
demonstrated. All four classes have the same settings  
except for the position value:

```
top: 20px;  
left: 20px;  
border-style: solid;  
border-width: 1px;
```

This example shows how the different positional values  
changes the position of the image.

# Understanding Positioning Cont.


---

- ***position: static;***
- Static is the default setting for any style.
- Setting an object's position attribute to *static* places the object in the flow as normal.
- The object is unaffected by the *top*, *right*, *bottom*, and *left* attributes.

**Static Positioning**

Note that the **position** attribute for the outer div is set to *relative*.

In this example all the content is placed inside a div styled with an ID called #wrap (marked by the dotted gray line) that has its position attribute set to *relative*. Placed right



here inside the text is an image. The image is styled with one of four classes depending on what positioning value is being demonstrated. All four classes have the same settings except for the position value:

```
top: 20px;
left: 20px;
border-style: solid;
border-width: 1px;
```

This example shows how the different positional values changes the position of the image.

# Understanding Positioning Cont.

---

- ***position: inherit;***
- If you look closely, you'll see that the value *inherit* appears in almost every drop-down menu when you create CSS.
- It literally means that the current element "*inherits*" this style from whatever elements are directly above it in the cascade.

# Applying the Framework to Existing Pages

---

- Now that you have created the framework for the *myKipple* site in a different style sheet from the one you were working on before, you need to alter the existing markup files and style sheet.
- Because *default.html* is the home page, it is a good place to start.
- Before you go any further, remove the big sticker graphic in the header and replace it with the text *Welcome to MyKipple.com*.
- The first step is to remove some of the styles you created in Hour 13 to avoid the new styles clashing with the old ones.
- You can delete these styles directly from the *Manage Styles* panel by highlighting them and pressing the *Delete* button on your keyboard or by right-clicking and selecting *Delete* from the context menu.
- *Delete* the *body* style and the *#wrapper* style.

# Applying the Framework to Existing Pages Cont.

---

- With the old styles removed, it is time to apply the new ones.
- To do this, you need to attach the new *layout.css* style sheet to the page.
- As you previously learned, you do this by clicking the *Attach Style Sheet* button in the *Apply* and *Manage Styles* task panes.
- Select the *layout.css* file with the *Browse* button and be sure to enable the *Attach to All HTML Pages* option.

# Applying the Framework to Existing Pages Cont.

---

- With the new style sheet attached, you see a dramatic change in how the page appears in *Design* view.
- That is because the new style sheet contains the CSS reset that removes all default styling from the content.
- Furthermore, when you attach a second style sheet, you insert it below the first one in the HTML code, and this means it gets more weight or importance in the style cascade.
- The CSS reset affects all default selectors, but in the *kippleStyles.css* file, you have already styled several of these, so you want your old styles to have more weight.
- To give *kippleStyles.css* more weight than *layout.css*, simply change the order of the two styles as they appear in the HTML code so that *layout.css* is first and *kippleStyles.css* is second.



# Applying the Framework to Existing Pages Cont.

- After the change, the two lines of code directly before the </head> end tag should read as follows:

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

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



# Applying the Framework to Existing Pages Cont.

---

- If you preview the page in your browser at this point, you notice three things:
- First, the header graphic you spent so much time aligning earlier is nowhere to be seen.
- Second, all the text is bunched together; that is, there is no breathing room between the paragraphs, headings, and blockquotes, except for the *h2* headings.
- This might look like a big problem, but it is exactly what you want: If the different selector boxes are stacked directly on top of one another (meaning there is no breathing room between the paragraphs, headings, and so on), the browser is not making any assumptions about your styles if you do not provide a style.
- In other words, your page looks the same in all browsers.
- Third, the text has reverted to the *Times New Roman* font.
- This is because you deleted the old body styles that contained the font-family definition for the entire page. This last one is the easiest one to fix, so let's do it before you get into the meat of things: Find the second body style in the *layout.css* style sheet (the one that has the background tile image defined), open the *Modify Style* dialog, and set *font-family* to *Arial, Helvetica, sans-serif*.
- Click *OK* to apply the change, and the font switches back to *Arial*.

# Applying the New Framework to the Page

---

- Now that you have attached the new style sheet, you can apply the new framework to the page.
- Most of it is already there because you added it in Hour 13, but some elements are still missing:
- With *default.html* open in *Split* view, find the `<body>` tag in the *Code* area, and create a new line directly below it before the `#wrapper` div.
- On the new line, insert a new div with the ID `#centeredBG`.
- When you close the tag, *IntelliSense* automatically inserts an end `</div>` tag.
- Cut it out as before, scroll to the bottom of the code page where the `#mainContent` and `#wrapper` divs are being closed, create a new line directly over the `</body>` end tag, and paste in the `</div>` end tag, giving it the comment `<!-- END #centeredBG -->`.
- Find the `<h1>` tag that contains the heading you previously inserted.
- Create a new line directly above it, and insert a new div with the ID `#header`.
- Place the closing `</div>` tag on a new line directly after the `<h1>` tag.

# Applying the New Framework to the Page Cont.

- Save and preview the page in your browser, and you see that the page now has the *headerBack* background off the top and the header has plenty of space before the page content begins.



# Applying the New Framework to the Page Cont.

---

- Looking back on the drawing, you can see that one element still has not been added to the page: the footer.
- Adding this element is done in the exact same way as before:
- Scroll to the bottom of the page and find the paragraph that starts with “*If you want further information...*”.
- Place your cursor anywhere on the line and look at it in *Code* view.
- You see that the `<p>` tag is littered with style code.
- Remove all the style code, leaving only the clean `<p>` tag.
- Create a new line directly over it and insert a new div with the ID `#footer`.
- Place the closing `</div>` tag directly after the closing `<p>` tag before the closing tag for the `#mainContent` ID.
- Find the `#footer` style in the `layout.css` file and set `font-variant` to `small-caps` and `font-size` to `0.8em`.
- To align the text to the center, go to *Block* and set `text-align` to `center`.
- Click *OK* to save the new style, and the footer text should look exactly as it did before; however, this time it is styled from the `layout.css` style sheet rather than an inline style.

# Adding a Header Image and a Menu

---

- In the sketch of the page layout, the header features a large *MyKipple.com* sticker and a menu.
- These are important elements of any website— the header image (or site name) provides an intuitive link back to the home page and the main menu.
- In effect, the header functions as a primary navigational tool for the visitor.

# Adding a Header Image and a Menu Cont.

---

- Remove the `<h1>` tag along with the heading inside it.
- Because the header contains the sticker, there is no need to have the text there as well.
- The Images folder contains the image file *MyKippleSticker.png* that you created earlier.
- Click and drag the image into the header in *Design* view, and give it the alternative text *Welcome to MyKipple.com*.
- When inserted, the image has both padding and a 1-pixel gray border.
- This is because it is being styled by the *img* style you created in a previous hour.
- To ensure that the *img* style applies only to images within the *#mainContent* area, use the *Manage Styles* task pane to change the *Selector Name* of the *img* style to *#mainContent img*.
- When you change the style name, the *kippleSticker.png* file changes position to hug the upper-left corner of the *#header* box.

# Adding a Header Image and a Menu Cont.

---





# Adding a Header Image and a Menu Cont.

---

- To line up the sticker image with the background, you have to create a new style.
- Click the *New Style* button in the *Manage Styles* task pane, and set the *Selector Name* to *#header img*.
- This style applies only to images within the *#header* ID.
- Change the *Define In* field to *Existing Style Sheet*, and select *kippleStyles.css* from the drop-down menu.
- Under the *Box* category, set *padding-top* to *90px*.
- Click *OK* to apply the style.
- The top of the header image now lines up with the background image.

# Adding a Header Image and a Menu Cont.

---



# Adding a Header Image and a Menu Cont.

---

- Right now, the sticker is too big.
- To change it, right-click the picture and select *Picture Properties* from the pop-up menu or double-click the picture to open the same dialog.
- In the *Picture Properties* dialog, go to the *Appearance* tab and change the *Height* to *130px*, making sure that the *Keep aspect ratio* box is checked.
- Click *OK* to apply the change, and use the *Picture Actions* button that appears at the bottom of the picture to *Resample* the picture to match size.
- To make the image link back to the home page, right-click it again, select *Hyperlink* from the pop-up menu, and set the hyperlink to the *default.html* page and set the *ScreenTip* to *Home*.
- With the new image heading inserted and formatted, save and preview the page.
- Expression Web 4 asks you if you want to save the attached files because you changed both the style sheets and the kipple sticker file.

# Adding a Header Image and a Menu Cont.

---

