Cupcake Bakery CSS Site

Your client operates an award-winning gourmet cupcake bakery in Southern California. Your job is to finalize the existing pages in the site and add a form that users can submit to join the client's email list.

This project incorporates the following skills:
- Working with CSS classes
- Manually editing CSS code
- Making an editable attribute in a template
- Working with the float and clear properties
- Creating an iframe element
- Creating and formatting form fields
The working files that we've seen so far are on the right track, but the pages still need some fine-tuning.

We want a different feature image to appear behind the logo on every page in the site.

The images in the different cupcake pages aren’t the same size; we’d like them to be consistent, so the pages don’t look so messy.

Also, we need to be able to update the Weekly Specials page frequently, but we don’t want to have to deal with the whole site structure. Can you set it up so we can edit one simple page and have our changes show up on the site?

Finally, we want a form that users can submit to get news of special events, coupons, and things like that. We’re offering a free cupcake in the month they were born as incentive for signing up.

The basic site design has already been created using a Dreamweaver template and CSS styles. You need to modify the existing template and CSS files where necessary to meet the client’s first two goals.

The third objective is fairly easy to accomplish using a server-side include, which will feed one HTML page into another when a browser calls the containing page. You will, however, need to apply a couple of simple workarounds when it comes to formatting the elements in the included page.

The final goal is the form. Again, Dreamweaver makes this very easy to accomplish. This doesn’t need to be a large form, and you can use built-in tools to create everything you need.

To complete this project, you will:

- Create class selectors to place background images
- Make an editable attribute in a template
- Control float and clear properties
- Search page code to apply classes
- Apply multiple classes to elements
- Insert one HTML page into another
- Create a form element and form fields
- Apply CSS to form fields
Stage 1  Working with Classes

As you complete this project, keep the following points about CSS in mind:

- **Tag selectors** define properties for a specific HTML element; all content on the page with that tag is affected. For example, properties in the a tag selector format every instance that is marked with `<a>` tags (all links on the page).

- **ID selectors** define properties for an element that has been named with a specific ID. In the page code, the ID is applied as an attribute of the relevant tag, such as `<section id="header">`.

  It is important to remember that each ID can only be applied once on a particular page, which means each element can be uniquely addressed. ID selector names begin with the # character, such as `#header`.

- **Compound selectors** (also called **descendant selectors**) define properties for a specific HTML element only within a defined path, such as `section#header a` to format all a elements (links) only within a section with the ID attribute of “header”.

- **Pseudo-classes** define properties for alternate states of an element, such as the `:hover` pseudo-class, which defines the appearance of a link when the mouse cursor hovers over that link.

You should also understand the nested nature of CSS properties (and HTML in general). Tags in a page contain other tags, creating a nested structure that is a type of **parent-child relationship**. Nested tags (the children) inherit properties from their containing (parent) tags; tags at the same level of nesting are referred to as **siblings**. Consider the following example:

```html
<body>
  <header>
    <section id="main">
      <article id="right">
        <footer>
      
      </article>
    
  </section>
  
</header>

Any properties that you define in the body tag selector automatically apply to all content on the page, because the `<body>` tag is the parent of all the nested elements. You can override those settings in a specific element by defining different properties for a specific ID.

If you define (for example) a different font size for the `#main` ID selector, the new font size will override the font-size properties in the body tag selector for the `<section id="main">` tag. The same font size is also inherited by its child `<article id="right">`, but not by its siblings `<header>` and `<footer>`.

Review Existing Site Files

When you start any new project — especially one where some of the work is already done — you should begin by analyzing the existing files and then determining what work needs to be completed. You will then be better able to create a plan to efficiently accomplish the necessary work.

1. Download Cupcakes_0WCC17_RF.zip from the Student Files Web page.

2. Expand the ZIP archive in your WIP folder (Macintosh) or copy the archive contents into your WIP folder (Windows).
   This results in a folder named Cupcakes, which contains the files you need for this project.

3. Create a new site named Cupcakes, using the WIP>Cupcakes folder as the site root folder.

4. In the Files panel, expand the Templates folder and then open the design.dwt file in the regular Design view.

![Image of Cupcakes website design](image)

This site includes a number of pages that are based on this template. You should notice that there is only one editable area (named “Page Content”), where all page-specific text and images are placed.

A CSS file (named style.css) is attached to the template, which means it defines properties for all pages that are attached to the template.

To complete this project you need to accomplish a number of tasks:

1. Add a page-specific background image to the header area of the page.
2. Format the content on individual pages to meet aesthetic requirements.
3. Link the “Weekly Update” file to the page that appears in the site.
4. Create a form so users can sign up to receive discounts via email.

5. Continue to the next exercise.
CREATE A CLASS SELECTOR TO PLACE BACKGROUND IMAGES

A class selector defines properties for any element that is marked with that class. Classes offer a number of advantages over other kinds of selectors:

- A class can be applied multiple times on a single page, unlike an ID selector which can be applied only once on a page.
- A class can be applied to different HTML tags on the same page, which means you can apply the same class to (for example) an h2 element and an img element.
- A class only applies to specific elements where you attach it, unlike a tag selector that affects all same-tagged elements on the page. For example, an h2 tag selector affects all h2 elements on the page; a class selector can be applied to only specific h2 elements without affecting other h2 elements on the page.
- You can apply more than one class to a single element, which means you can define classes to perform very specific tasks, and then apply only and exactly what you need to a specific element.

The one primary disadvantage of classes, however, is that they must be intentionally and manually attached to every element where you want those properties to apply. As you will see in the following exercises, this can be time-consuming if you want to apply the same class to a large number of elements on multiple pages.

1. With the design.dut template open in the regular Design view, show the CSS Designer panel in All mode.

2. Select style.css in the Sources section of the panel, then click the Add Selector button in the Selectors section of the panel.

   Remember, to add a new selector in the CSS Designer panel, you must first select the source where you want to add the style. We will not continue to repeat these instructions every time you need to add a new selector in this project.

3. With the new selector name highlighted, type .bkgHome. Press Return/Enter to finalize the new selector name.

   Class selector names always begin with a period.

4. Make sure Show Set is not checked, then click the Background button at the top of the Properties section to jump to those options in the list.

   Note:
   Feel free to work in either Design view or Split view. We will tell you when you need to work in the Code pane, and when you need to change to a specific view.
5. Define the following options for the new selector:
   background-image URL: images/backgrounds/back-home.png
   background-repeat: no-repeat

The Related Files bar above the document window shows that the style.css file has been changed. However, nothing has changed in the open template file. Classes do not affect a page until they are intentionally applied to one or more elements.

6. If the new selector does not appear at the bottom of the list in the CSS Designer panel, click the new class selector and drag it to the bottom of the list.

   New selectors are automatically added below the previously selected item (yours might have been in a different location than ours). You can reorder them by simply dragging in the panel.

   Although this step isn’t strictly necessary, it is a good idea to keep your styles organized to make them easier to navigate.
7. Click to place the insertion point in the header element. In the Tag Selector, click the `<header>` tag to select the entire element.

8. In the Properties panel, open the Class menu and choose `bkgHome`.
   The top section of this menu lists all available classes. Because your CSS file includes only one class, it is the only choice in the list.

9. Click to place the insertion point back in the element.
   This effectively deselects the element and removes the visual aids, so you can better see the results of the applied class — the header element now has a background image.

10. Save the file. When prompted, update all files linked to the template.
    When the Update Pages dialog box shows the process is done, click the Close button.
11. Click style.css in the Related Files bar to show that file’s code, then choose File>Save.
   In this exercise, you changed both the page and the attached CSS file, so you need to save both files.

   Click style.css to open that file in the Code pane, then save the file.

12. Continue to the next exercise.

**Manually Edit CSS Code**

The class selector you created in the previous exercise defines a background image for the header element. Remember, though, you need to apply different images to the same element on each page. In this exercise, you will define additional classes that you will apply in the next exercise.

1. With design.html open, click the Code button in the Document toolbar. If necessary, click style.css in the Related Files bar to show that file instead of the template page source code.

2. Scroll to the bottom of the code and review the class selector that you created in the previous exercise.
   The selector code appears at the bottom of the file because you dragged it to the bottom of the list in the CSS Designer panel. The order of selector code in the CSS file matches the order you see in the CSS Designer panel.
   CSS code uses the following syntax:
   ```
   name {
     property: value;
   }
   ```

3. Click and drag from the line number for line 116 to 119.
   Clicking a line number selects the entire line of code.
   ```
   111 background: url(images/backgrounds/back-home.png);
   background-repeat: no-repeat;
   ```
   Rather than using the CSS Styles panel, you are going to simply copy and paste the required code, then make the necessary changes in each version.

4. Choose Edit>Copy (Command/Control-C) to copy the selected code.

5. Click at the beginning of Line 120 to place the insertion point, then choose Edit>Paste (Command/Control-V).
   This results in two .bkgHome selectors; you will change the selector name and background-image URL of the duplicate in the next few steps.
6. Review the Files panel.
   
The name of each HTML file gives you an idea of what each page contains. Because you will use a different class to change each page’s header background, you should use similarly indicative class names to make your work easier later.

7. In line 120 of the Code pane, change the class name to .bkgEveryday.

8. In line 121, select and delete the code that defines the image path. Leave the closing semicolon in place.

9. Type u to open the Code Hints menu.
   
   When you work directly in the Code pane, Dreamweaver provides code hints related to the current context. As soon as you type the letter u, the Code Hints menu shows the only matching available option — the url() attribute of the background-image property.

10. Press Return/Enter to accept the highlighted option in the Code Hints menu.
    
    Again, Dreamweaver provides significant help when you are working directly in the Code pane. As soon as you accept the url() attribute, the application moves the cursor inside the parentheses and presents a menu of possible values. The Browse option at the top of the menu is highlighted by default.

11. Press Return/Enter to accept the Browse option in the Code Hints menu.
    
    You can also double-click Browse in the menu to open the navigation dialog box.
12. In the resulting navigation dialog box, select `back-everyday.png` (in the site's images>backgrounds folder) and click Open/OK.

![Select File Dialog](image)

**Note:**
You can also manually type the required file name in the CSS code. If you choose to use this option, make sure the file names you type match exactly what you see in the Files panel. Capitalization matters!

13. In the Code view, move your mouse over the new image path.
This technique allows you to preview a specific image directly in the Code view.

![Code View](image)

14. Repeat this process to add four more class selectors, using the following information:

<table>
<thead>
<tr>
<th>Class name</th>
<th>Background Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>bkgWeekly</td>
<td>back-weekly.png</td>
</tr>
<tr>
<td>bkgCustom</td>
<td>back-custom.png</td>
</tr>
<tr>
<td>bkgJoin</td>
<td>back-join.png</td>
</tr>
</tbody>
</table>

**Note:**
You don’t need a class for `weekly-update.html`. Later in this project, you are going to place it into `cupcakes_weekly.html`, which will already have the required background image.

15. Save the CSS file and then continue to the next exercise.
**Make an Editable Attribute in a Template**

You need to apply different background images to the header element for each page in this site. However, the header element in the template is not an editable area, which means you can’t select it to apply the necessary class.

Rather than creating an editable region, which would leave the placed logo in that element vulnerable on individual pages, you can define an editable attribute in the template, so you can change only that attribute on each page.

1. **With design.dwt open, turn on the Split view and show the page source code in the Code pane.**

2. **Place the insertion point in the header element, then click the header tag in the Tag Selector to select the entire element.**

   In the Code pane, the code related to the active selection is also highlighted. The first line of the highlighted code shows the opening header tag, along with the applied class attributes. In the next few steps, you are going to make the class attribute editable in pages that are attached to this template.

3. **Choose Tools>Templates>Make Attribute Editable.**
4. In the resulting dialog box, make sure CLASS is selected in the Attribute menu, then check the Make Attribute Editable option.

All attributes of the selected tag are available in this menu (in this case, only CLASS is applied so it is the only one available).

![Edit Tag Attributes dialog box](image)

5. **Click OK to apply the change.**

When you make an attribute editable in a template, the previously defined value is removed from the page code. In the Design pane, the background image is no longer visible in the template.

![Design pane with edited background image](image)

6. **Save the template file, updating linked pages when asked.**

7. **Close the Update Links dialog box, then close the template file.**

8. **Using the Files panel, open the cupcakes-everyday.html file.**

![Everyday Favorites in the cupcakes-everyday.html file](image)
9. In the CSS Designer panel, select style.css in the Sources list to show all selectors that are available in that file.

In the next few steps, you will need to type these exact class names to change the attribute values for individual pages in the site. By displaying the selectors in the CSS Designer panel, you can see exactly what you need to type.


The resulting dialog box shows all editable properties of the applied template. In this case, the class attribute (of the header element, even though this information is not presented in the dialog box).

11. In the Class field at the bottom of the dialog box, type the name of the class you want to apply (bkgEveryday).

Unfortunately, Dreamweaver does not provide you with a menu of available class names; you have to type the exact class name (without the opening period).

12. Click OK to apply the change.

Even though the header element is not selectable or editable on the page, the background image defined in the bkgEveryday class now appears in the header div.

13. Save the file and close it.

14. Repeat the same process to change the class for each page in the site (except index.html and weekly-update.html).

<table>
<thead>
<tr>
<th>File name</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>cupcakes-weekly.html</td>
<td>bkgWeekly</td>
</tr>
<tr>
<td>custom-cakes.html</td>
<td>bkgCustom</td>
</tr>
<tr>
<td>join-the-club.html</td>
<td>bkgJoin</td>
</tr>
</tbody>
</table>

Note: You do not need to modify the index.html file because the default class — bkgHome — is appropriate for that page.

15. Save and close any open files, then continue to the next exercise.
**CONTROL FLOAT AND CLEAR PROPERTIES**

The Float property of CSS defines how an element attaches to its container and how other elements appear in relation to the floating element.

- If you define a left float value, the floating element attaches to the left edge of its container; other content wraps around the right edge of the floating element.

- If you define a right float value, the floating element attaches to the right edge of its container; other content wraps around the left edge of the floating element.

- If you define a none float value, the element does not float; other content does not wrap around the non-floating element.

The Clear property defines where other floating content cannot wrap around an element. In other words, if you define a right clear value to an `<img>` tag (for example), no other content can appear on the right side of that image. The clear property can have a value of left, right, both, or none (the default).

1. **Open the file cupcakes-everyday.html and turn on the Live view.**
   **Scroll through the page and review the contents.**

   Each listed cupcake includes a photo and a brief description. If you look closely, however, you will notice that the images are not uniform in size. Rather than manually editing each image, you are going to define a class that controls the size of each image where that class is applied.

   As a general rule, you should avoid scaling images without resampling in Dreamweaver because the Web server still has to transmit all of the file data for the original-size image.

   In this case, the required scaling is very slight, and would not require users to download huge amounts of unnecessary data. As a “best practice,” however, it would be better to edit the actual image files in an image-editing application such as Adobe Photoshop.
2. Using the CSS Designer panel, create a new class selector named .img150px.
Define the following properties:

- width: 150 px
- height: 150 px
- margin-left: 5 px
- margin-right: 5 px
- margin-bottom: 10 px
- float: left

3. Click the first image in the page content area to select it. In the Properties panel, open the Class menu and choose img150px.

This menu provides the same options as the contextual menu for a specific tag in the Tag Selector. You can apply any class that is available in the attached CSS file to the selected object.

As you can see, the selected image resizes slightly. Because you defined the float:left property, other content is able to wrap around the right edge of the image.

Because the descriptive text for each is short, the next cupcake name is also allowed to wrap to the right of the image. You should correct this problem before continuing.

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All of the cupcake images in this site are square, so you are not changing their aspect ratio.</td>
</tr>
</tbody>
</table>
4. In the document window, click to select the “Confetti” cupcake name.

5. Click the CSS button on the left side of the Properties panel to show those options.
   When the Properties panel appears in CSS mode, you can easily identify and edit the selector (rule) that affects the active selection.

6. Click the Edit Rule button in the left side of the panel.
   This button opens the CSS Rule Definition dialog box for the selected rule.

7. In the Box options, choose both in the Clear menu, then click OK to change the selector.
   Because you assigned the clear:both property to the h2 selector, the second cupcake name — which is formatted as an h2 element — is now forced to appear after the image where the float:left property has been applied (by the img150px class).

8. Click the HTML button on the left side of the Properties panel to return it to the regular mode.
9. Scroll through the open file and apply the .img150px class to each image in the document.

10. Choose File>Save All, then continue to the next exercise.

CSS Mode of the Properties Panel

You can easily make changes to CSS rules using the options in the CSS mode of the Properties panel:

- **Targeted Rule** shows the CSS rule applied to the current selection. You can also choose a different target rule in the attached menu, or choose New CSS Rule from the menu to create a new selector.
- **Edit Rule** opens the CSS Rule Definition dialog box for the targeted rule.
- **CSS Designer** opens the CSS Designer panel and displays properties for the active selection.
- **Font, Size, and Color** change the associated options in the targeted rule.

- The **Font Style** and **Font Weight** menus define the font-style and font-weight properties (respectively).
- **Align Left**, **Align Center**, **Align Right**, and **Justify** change the text-align property of the targeted rule.
Apply Multiple Classes to Elements

As we explained in the beginning of this project, some professionals define very specific classes to accomplish only a defined goal. This method can make it easier to understand the purpose of various classes, especially after you return to a project you created a long time ago, if you work on files created by another user, or if you hand off your site files to another designer.

Using this technique, however, often requires multiple classes applied to the same element. Fortunately, Dreamweaver makes that process relatively easy. In this exercise, you will create a new class that assigns the float:right property, then apply that class to every other image in the list of cupcakes.

1. With cupcakes-everyday.html open and the Live view active, click style.css in the Related Files bar to show that file in the Code pane.

As you know, you can use the CSS Designer panel to edit the properties associated with a specific CSS selector. Once you are comfortable with the basic concepts of CSS, you might find it easier to work directly in the Code pane to make necessary changes.

When you write code, you must follow the proper rules of syntax (the formal structure or “grammar” required in code). Dreamweaver Code Hints facilitate writing and changing code: hints make it easier to create the proper syntax and ensure that your code functions as expected.

2. In the Code pane, scroll to the end of the existing code. Place the insertion point in the first empty line and type:

```css
.imgfloatRight {
```

3. Press Return/Enter.

When you press Return/Enter, Dreamweaver automatically adds the required closing bracket to the code.

4. Type `float`.

As soon as you begin typing the Code Hints menu presents options that match the characters you type.
5. With the `float` property highlighted in the menu, press Return/Enter to add that property to the selector code.

Code Hints are invaluable for creating the proper code syntax. If you don’t know the exact name of a property, you can use the Code Hints menu to find exactly what you need.

After you add a specific property to the selector code, Dreamweaver automatically adds the required colon and shows the default value options in a secondary Code Hints menu.

![Code Hints menu example](image)

6. Choose right in the resulting Code Hints menu, then type a semicolon at the end of the line.

Make sure you type a semicolon at the end of the line.

![Code Hints menu example](image)

7. With the Live view active in the Design pane, click to select the image under the “Confetti” heading.

8. Click the Add Class/ID button in the Element Display.

The Element Display appears at the top of the selected element.

![Element Display example](image)

Click the image in the Live view to reveal the Element Display.

9. In the resulting field, type `.` and then click `.imgFloatRight` in the menu to apply that class.

As you type in the field, a Code Hint menu presents a list of available options that match the characters you type. You can use the menu to accept the required class, or simply type the entire class name.

![Code Hints menu example](image)

The menu presents options that match the characters you type.
10. Click Source Code in the Related Files bar to show the page source in the Code pane.

In the code for the selected `<img>` tag, you can see the two classes are listed inside the quotes for the class attribute.

If you remember, the `img150px` defined a `float: left` property. Because the `imgFloatRight` class comes second — it is later in the nesting order — the `float: right` property overrides the previous `float` value.

11. Using the same technique, apply the `imgFloatRight` class to every other image on the `cupcakes-everyday.html` page.

12. Choose File > Save All, close the `cupcakes-everyday.html` file, and then continue to the next exercise.

**Apply Classes to Unlinked Files**

By now you should understand that the properties and attributes of a CSS style are stored in a separate file. HTML files can be linked to the external CSS file so that the selectors in the CSS file apply to matching elements in the HTML file.

This project incorporates a separate file that lists your client’s weekly specials. This file will be placed in a special container on another HTML file, so that your client can easily edit the specials without having to interact with the overall site files. In this exercise, you will apply classes for the images in the client’s HTML file.

1. Open the file `weekly-update.html` and make sure the Live view is active.

As you can see at the top of the document window, this file has no Related Files bar. It is not yet linked to the CSS file that contains the classes you defined in previous exercises.

2. In the Design pane, click to select the image under the “Adam’s Apple” heading to reveal the Element Display.

The file has no Related Files bar.
3. Click the Add Class/ID (+) button in the Element Display. In the resulting field, type .img150px and press Return/Enter.

Notice that no Code Hints menu appears with available options. Remember, this file is not attached to the style.css file, so the img150px class does not technically exist for this HTML file (the class is “not defined” in the file). In other words, there are no available options for the hint list to present.

When you Press Return/Enter, Dreamweaver asks you to identify a source for the class you just defined. The class does exist in the site’s CSS file and you don’t want to rewrite it with a new one, so you are simply adding the class name to the img tag.

4. Press the ESC key to close the pop-up window without defining a source.

Because the class does not technically exist in or to this HTML file, the image size does not change in the Design view. This problem will be corrected in the next exercise.

5. Repeat Steps 2–4 for the other images on this page.

6. Repeat Steps 2–4 to apply the .imgFloatRight class to the second and fourth images on the page.

Again, the class does not technically exist in or to this HTML file, so the image position does not change in the Design view.

7. Save the HTML file and close it, then continue to the next exercise.
**Insert One HTML Page into Another**

As you have seen, the weekly-update.html file does not include any of the same formatting as the other pages in the site. This file is created every week by your client, then emailed to you along with the required image files. (This scenario is not uncommon in professional Web design, whether the included file comes directly from your client or from some other source.)

Rather than copying and pasting the content from one file to another, you can use an iframe element to load one HTML file inside another. When a browser calls the page, the Web server reads the link and delivers (“includes”) the linked file as part of the parent page.

1. **Open cupcakes-weekly.html and make sure the Live view is active.**

2. **Click to select the paragraph element after the “Flavors of the Week” heading.**

3. **Click the iframe button in the HTML Insert panel, then click the After button in the Position Assistant.**

![Example of iframe usage](image)

4. **Show the page source in the Code pane.**

The iframe element appears after (below) the previously selected paragraph element in both the Code and Design panes. At this point it has no content because you have not yet defined the iframe's source.
5. Place the insertion point before the closing “>” of the opening tag. Press the spacebar once to open the Code Hint menu.

6. Type `src`, and press Return/Enter to accept `src` in the Code Hint menu. Accepting the menu option automatically places the required `""` characters in the code for you and opens the Code Hint menu with the Browse option already highlighted.

7. When the second Code Hints menu appears, press the down-arrow key until `weekly-update.html` is highlighted, then press Return/Enter.

After defining the `src` attribute for the `iframe` element, you can see the defined file’s content inside the `iframe` when the Live view is active.

The `iframe` element is commonly used to display entirely external Web sites (such as a displaying a Twitter or Facebook feed), styles that apply to the parent page are not inherited by the child page in the `iframe` element.
8. Using the CSS Designer panel, define a new tag selector named iframe using the following settings:

- width: 560px
- height: 500px
- border: 0px

An iframe is still an HTML element, which means you can format its basic appearance using CSS selectors.


The source page in this iframe is not linked to styles.html, because you don’t want some of the attributes applied to the embedded page (such as the background image of the overall body element). Instead, you are going to link the update page to a second CSS file that contains only the styles you need for those elements.

10. In the CSS Designer panel, click the Add CSS Source button and choose Attach Existing CSS File.

11. Click the Browse button in the resulting dialog box. Navigate to styles-weekly.css (in the site root folder) and click OK to return to the Attach Existing CSS File dialog box.
12. Choose the Link option, then click OK to attach the CSS file to the weekly-update.html file.

Note:
To simplify this exercise, we provided the second external CSS file for you. In a professional environment, you would probably have to determine the appropriate styles and create the file yourself.

13. Save and close weekly-update.html.

14. Turn the Live view off and back on, then review the iframe content.
At times your changes do not reflect unless you first toggle the Live view off and on.

15. Save and close all files, then continue to the next stage of the project.
Stage 2 Creating Online Forms

Online forms are used to collect user information, and then transfer that information to Web servers. Surveys, electronic commerce, guest books, polls, and membership applications all make use of online form technologies. While well-designed forms are easy to use, poorly designed forms prove troublesome for both users (who complete the forms) and Web server managers (who access user data). Fortunately, Dreamweaver makes it easy to create robust yet understandable forms — simplifying and streamlining the interaction between users and Web servers.

Web-based forms are composed of a series of **form objects** (also referred to as **form fields**). Different types of form objects have different purposes, and they gather different types of information. Basically, all form objects allow users to enter data; each object type facilitates a distinct format of data input. You can use Dreamweaver to create a number of different form objects, including text fields, radio buttons, check boxes, menus, and buttons.

It is important to realize that processing a submitted form requires some type of script. These contain instructions to identify the user's data and then perform tasks based on that data. You can write your own scripts, and many hosting providers offer sample form-processing scripts (as well as instructions on how to implement those scripts when you build a form).

### Create a Form Element

A form element is a distinct element of a Web page; it is the container for all form objects. As a container, the form element ensures that different form objects are related to one another, which makes it possible to combine all of the form information as a single submission. A form's Submit button identifies all related form objects and collects the information they contain in a single string.

1. **Open join-the-club.html from the Cupcakes site root folder. Turn off the Live view to show the regular Design view.**

   This page was created from the defined site template file. You will create the form in the editable region of this page.

2. **Open the Insert panel and show the Form options.**

3. **Click the Form button and drag to the empty paragraph below the paragraph element in the editable region.**

   ![Drag the Form button into the empty paragraph element.](image)

4. **Click to place the insertion point in the new form element.**

   When the Live view is turned off, a red outline indicates the boundary of the newly inserted form. (If you don't see this outline, choose View>Visual Aids>Invisible Elements.)

---

**Note:**

PHP and CGI scripts are commonly used to process forms. You should consult your hosting provider for specific instructions for serving a form on your Web site.
5. In the Properties panel, type register in the Form ID field and type mailto:info@sassycupcakes.atc in the Action field. Leave the remaining fields at their default values.

The form ID creates a unique name for the form you are creating. This will be useful later when you apply CSS to format the various form elements.

The mailto protocol in the Action field is one method of receiving data without using a script. When a user submits the form, the user’s default email client opens and creates a message with the form data. This is not a particularly reliable method for receiving form data, but it suits the purpose if you are gathering generic information; however, it is not suitable for gathering sensitive information such as credit card numbers.

6. Save the file and continue to the next exercise.

The Form Properties Panel in Depth

The Form Properties panel allows you to control the options related to a specific form area.

- **Form ID** is a unique name that identifies the form for scripting or CSS formatting.
- **Action** specifies the page or script that will process the form data, essentially determining what should be done with the form content.
- **Class** allows you to apply a class selector (from a CSS style sheet) to the form.
- **Method** determines how the browser and Web server present the form data to the application that processes the form (the action page):
  - **Default** uses the browser’s default settings to send the form data to the server. According to W3C specifications, the default method for forms is GET.
  - **GET** attaches form data to the URL of the action page that processes the form data. This method limits the amount and format of data that can be passed to the action page.
  - **POST** sends the form data as a standard input to the action page. This method does not impose any limits on the passed data.
- **No Validate** adds the HTML5 novalidate attribute to the form tag, so the form is not automatically validated when the form is submitted.
- **Auto Complete** sets the HTML5 autocomplete attribute to “on” when checked; this allows the form to autofill based on a user’s settings.
- **Enctype** (short for “encoding type”) specifies the format in which the data will be sent to the server so the server software can interpret the input correctly. The default is application/x-www-form-urlencoded. The text/plain enctype is used for email replies. If a file is being uploaded with the form, multipart/form-data must be used.
- **Target** defines the window or frame in which the server displays the action page’s response (data) to the form.
- **Accept CharSet** defines character encodings that are to be used for the form submission. Common values include UTF-8 (Unicode) and ISO-8859-1 (for the Latin alphabet).

Note:
You use the mailto protocol in this exercise because we cannot be certain that everyone has access to the same script on a specific type of server.
CREATE FORM TEXT FIELDS

Text fields, defined with the `<input>` tag, are the fundamental building blocks of almost all online forms. HTML5 adds a number of field types and attributes to the `<input>` tag, which makes it easy to define special fields such as email addresses, phone numbers, and passwords.

Dreamweaver incorporates these options directly in the Insert Form panel. Keep in mind that not all browsers support all HTML5 input types and other options. If a user's browser does not support certain options, the special fields will behave as regular text fields.

1. With `join-the-club.html` open, display the Split view and show the page source in the Code pane.

2. Click the Text button in the Form Insert panel and drag into the existing form element.

3. Review the Code pane.

When you insert a form field, Dreamweaver automatically creates a field and a field label.

In the Code pane, the `<input>` tags define the text field and `<label>` tags define the text field's label. The field name that you defined is used as both the `name` and the `id` attribute of the `<input>` tag. In the code for the `<label>` tag, the name of the text field is assigned to the `for` attribute — basically defining which text field this is the label for.

4. In the Properties panel, change the Name field for the selected field to `firstname`. Press Return/Enter to finalize the new name.

In the Code pane, you can see that this changes both the name and ID attributes of the `<input>` tag, as well as the `for` attribute of the `<label>` tag.

```html
The <label> tag defines the field label. The <input> tag defines the text field.
```

The new name becomes the `<input>` tag's `name` and `id` attribute, as well as the `for` attribute of the `<label>` tag.

---

Note:
A text area is similar to a standard text field, but intended for larger amounts of text, such as multiple sentences or paragraphs.
5. With the field still selected in the Design pane, check the Required option in the Properties panel.

Because you checked the Required box in the Properties panel, the HTML5 `required` attribute is added to the `<input>` tag. This attribute is a boolean attribute; it is either true or false. It does not need an actual value in quotation marks; its presence in the tag indicates that the attribute is “true” — in other words, this field is required.

6. In either pane of the document window, change the field label text to **First Name**.

7. In the Design pane, place the insertion point after the existing text field to deselect the existing field and label.

8. Press Return/Enter to create a new paragraph.

When you press Return/Enter, the line with the first text field (and its label) is automatically tagged as a paragraph; the `<label>` and `<input>` tags are now surrounded by opening and closing `<p>` `</p>` tags.
9. Insert another text field at the insertion point, using `lastname` as the field name and `Last Name` as the label text. Make the field required.

   ![Image of text field with 'Last Name' as label]

   **Note:**
   You have to select the actual text field in the document window to change properties in the Properties panel.

10. Create a new paragraph and insert another text field using `company` as the field name. Do not make this field required. As the label text, type:

    ```
    Company Name (Required if you want to enter the monthly drawing for free office cupcakes)
    ```

   ![Image of text field with 'Company Name' as label]

   **Note:**
   Later in this project you will use CSS to format the various fields and labels in this form.

11. Create a new paragraph after the company text field, then click the Email button in the Form Insert panel.

    An `<input>` tag with the `email` type attribute is automatically validated when the form is submitted on HTML5 browsers. Keep in mind that email field validation only looks for the correct pattern of characters (e.g., xxx@xxx.xxx). This validation does not check to see if the user's entry is an actual, working email address.

    ![Image of email field with additional options]

    The new field has "email" as the type, name, and ID attributes of the `<input>` tag and "Email:" as the label text.

**Note:**
On mobile-device browsers that recognize the email type attribute, bringing this type of field into focus causes the on-screen keyboard to show the "@" and " .com" options.
12. Use the Properties panel to make the field required, then delete the colon from the label text.

13. Create a new paragraph after the email field, then click the Password button in the Form Insert panel. Make the field required, then delete the colon from the label text.

When an `<input>` tag uses the `password` type attribute, the user-entered text appears as asterisks or dots.

14. Save the file and continue to the next exercise.

CREATE A MENU FIELD

Menus and lists display a set of options from which users can select one or more responses. These two types of form fields have the same basic underlying structure, but with different appearances and purposes.

A basic menu shows a single option; when a user clicks the menu, the menu opens (drops down) and more options appear. With a standard menu field, users can choose only a single response from the available options. The menu closes when the user chooses a response, displaying only the selected option.

1. With `join-the-club.html` open, create a new empty paragraph after the company field in the existing form.

2. Drag the Select button from the Form Insert panel to the new empty paragraph.

   A menu or list object is created using the `<select>` tag.

3. In either panel of the document window, change the field label to `When were you born?`.
4. Click the select field in the Design pane to select it, then use the Properties panel to change the field name to **birthday**.

![Select field](image)

Menus and lists are created with the `<select>` tag.

5. With the field selected in the Design view, click the List Values button in the Properties panel to define the selections that will appear in the menu.

![List Values](image)

6. Macintosh users: Click the first field in the Item Label column to place the insertion point.

Windows users do not need to click in the field because the insertion point is already in place when you open the dialog box.

7. Type **January** as the first item label. Press Tab to move to the Value column and type **Jan**.

![List Values](image)

8. Press Tab again to move to the second line of the Item Label field.

Each line in the dialog box represents a new list/menu option. When the insertion point is within the last item value, pressing Tab adds a new list item. Alternatively, you can simply click the “+” button above the Item Label column to add a new list item (or click the “−” button to remove the currently selected list item).

9. Type **February** as the item label, press Tab, and type **feb** as the item value.
10. Repeat Steps 8–9 to add ten more list items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>March</td>
<td>mar</td>
</tr>
<tr>
<td>April</td>
<td>apr</td>
</tr>
<tr>
<td>May</td>
<td>may</td>
</tr>
<tr>
<td>June</td>
<td>jun</td>
</tr>
<tr>
<td>July</td>
<td>jul</td>
</tr>
<tr>
<td>August</td>
<td>aug</td>
</tr>
<tr>
<td>September</td>
<td>sep</td>
</tr>
<tr>
<td>October</td>
<td>oct</td>
</tr>
<tr>
<td>November</td>
<td>nov</td>
</tr>
<tr>
<td>December</td>
<td>dec</td>
</tr>
</tbody>
</table>

11. After typing the December value, press Tab to add a final list item.

12. Type -Select- in the Item Label field and leave the Value field blank.

13. Click the Up Arrow button (above the Value column) until the -Select- option appears at the top of the list.

14. Click OK to add the list values to the menu field in the form.

15. In the Properties panel, choose -Select- in the Selected list.

   The Code pane shows that each item in the defined list has been created as an option element within the opening and closing <select> tags. The first option (-Select-) element includes the selected boolean attribute, which means that option is automatically selected in the menu when the page opens in a browser.

16. Save the file and continue to the next exercise.
CREATE A CHECKBOX

Individual checkboxes, which have an on or off state, are commonly used to indicate agreement. A checkbox group is simply a group of checkboxes with the same name; when the user submits a form, the value of each selected box is sent as the value for that field name.

1. With join-the-club.html open, create a new empty paragraph before the label of the first text field (First Name) but within the form object.
   You can place the insertion point before the label, then press Return/Enter to add the new paragraph within the form object.

2. Drag the Checkbox button from the Form Insert panel into the new empty paragraph.

3. In either pane of the document window, change the field label to Yes! I want to join the Sassy Club!

4. Select the field to make it active. In the Properties panel, change the field name to agree and make the field required.
   In the Properties panel, you can use the Checked option to determine whether a checkbox object is already selected when the form first displays. By default, the checkbox is not selected.
   The Value field defines the value that will be sent to the server when the form is submitted. For example, you can define a checked value of “A” for a form object named “Checkbox1”; if the user checks that box, information will be sent to the server as Checkbox1=ON. If you do not define a checked value, information will be sent to the server as Checkbox1=ON when the box is checked.

5. Save the file and continue to the next exercise.
**ADD SUBMIT AND RESET BUTTONS**

Buttons perform an assigned task when clicked. The **Submit** button is crucial to any form, ensuring that the data is not sent to the server until the user chooses. The **Reset** button clears all entries in form objects and restores the form to its original (empty) state.

1. With `join-the-club.html` open, create a new empty paragraph after the last text field in the form.

2. Drag the Submit Button button from the Form Insert panel into the new empty paragraph.
   
   An input element with the `submit` type attribute submits the form data to the server.

3. In the Properties panel, type **Sign me up!** in the Value field.
   
   In the Properties panel, the Name field shows the ID of the button object; this is just a name used for CSS purposes; it does not control the button's behavior. The Value field defines the text that appears on the button.

4. Drag the Reset Button button from the Form Insert panel to the immediate right of the existing button.
   
   An input element with the `reset` type attribute clears the entered data.

5. Save the file and continue to the next exercise.
**HTML Form Fields in Depth**

Most of the options in the Form Insert panel create input elements (using opening and closing `<input>` tags); the button text in the Form Insert panel determines the type attribute of the resulting input element.

We describe the most common input element — the `textfield` — first. However, for ease of use, we then list the other types of elements in alphabetical order rather than the order they appear in the Form Insert panel.

**Text.** This creates an input element with the "textfield" type attribute; users can enter alphanumeric content in the field. In the Properties panel, you can define a number of attributes for a textfield input element:

- **Auto Complete** defines a field that will be automatically filled in with information that is stored somewhere on the user's computer.
  This attribute requires a value of on or off. Because you can control this attribute for an entire form and for individual fields in a form, you can prevent certain fields in an autocomplete form from being included:
  ```html
  <input type="email" name="email" autocomplete="off">
  ```

- **Auto Focus** defines a field that is automatically in focus when the form loads.
  This attribute is a boolean value, which means it is either true or false. If true, the autofocus attribute is added to the `<input>` tag:
  ```html
  <input type="email" name="email" autofocus>
  ```

- **Disabled** defines a field that cannot be modified by the form user. You can use disabled fields to include required, developer-defined information with the form submission. This attribute is a boolean value; if true, the disabled attribute is added to the `<input>` tag:
  ```html
  <input type="email" name="email" disabled>
  ```

- **Form** can be used to define which form an element is related to. (HTML5 allows you to place form fields outside the boundaries of the actual form element.)

- **List** can be used to call an existing list, which you define somewhere on the page using the `datalist` element; each item in the list is created as an option element inside the opening and closing `<datalist>` tags:
  ```html
  <datalist id="list1">
    <option value="Item 1">
    <option value="Item 1">
  </datalist>
  ```
  (The `datalist` element does not need to be defined within the form element to be applied to a specific form field.)

Keep in mind that many of these form-object types were added in HTML5, which means they are not yet uniformly supported. If a browser does not support a specific type, the field will simply appear as a regular text field. (HTML5-specific options are noted in blue.)


**Max Length** is the maximum number of characters that can be entered in the text field.

**Pattern** defines a regular expression that is the pattern of characters that must be used in a field.

Regular Expressions are based on JavaScript syntax, which is beyond the scope of this book. The site [html5pattem.com](http://html5pattem.com) includes a library of common patterns that have already been created by experienced users; you can copy the necessary pattern code from that site and paste it into Dreamweaver.

**Placeholder** defines text that appears inside a field when the form first loads. You can use this to provide hints or other advice (such as the expected content).

**Read Only** defines a field that cannot be modified by the user. This is a boolean value; if true, the readonly attribute is added to the `<input>` tag:

```html
<input type="email" name="email" readonly>
```

**Required** defines a field that must be filled in for the form to be submitted. This is a boolean value; if true, the required attribute is added to the `<input>` tag:

```html
<input type="email" name="email" required>
```

**Size** is the number of characters that are visible in the field. (You can also use CSS to define the width property of a field, using any unit of measurement that is supported by CSS.)

**Tab Index** is in order in which a field is selected if a user presses the tab key to move through fields in a form.

**Title** defines text that appears in a tooltip when the user's mouse moves over a field.

If you define both value and placeholder attributes for the same field, the value appears in the field when the form opens:

![Placeholder Text: This is the value](image)

When a user deletes the existing value, the placeholder attribute content appears in the form field:

![Placeholder Text: This is the value](image)

**Value** defines an initial value that appears in the field when the form loads. This attribute is actually transmitted when the form is submitted.
**HTML Form Fields in Depth (continued)**

**Button.** This creates a basic button element. You can change the button’s name/ID and value in the Properties panel and use scripting to cause the button to do something.

**Checkbox.** This creates an input element with the “checkbox” type attribute. A checkbox has only two possible values — selected or not selected.

**Checkbox Group.** A checkbox group is simply a group of checkboxes with the same name attribute. When you click the Checkbox Group button in the Form Insert panel, a dialog box opens where you define the items in the group. (To add more than the default two options, you have to use the + button above the Label field.)

**Color.** This creates an input element with the “color” type attribute. When a user clicks the field, a browser-specific color picker allows the user to select a color.

**Date.** This creates an input element with the “date” type attribute. In addition to the attributes that can be defined for a textfield input element, you can also use the Properties panel to define the following attributes:

- **Min** and **Max** values are the minimum possible date and the maximum possible date, respectively.
- **Step** defines the legal interval for the field.

Users can select a specific date from the pop-up selector. If you have defined a step attribute for the field, legal values are white; in this example, the step value of 3 means only every third day is a legal value.

**DateTime.** This creates an input element with the “datetime” type attribute. Users can select a date and time based on the Coordinated Universal Time or UTC clock (also known as Greenwich Mean Time or Zulu Time).

In the Properties panel, you can define the same properties as for the date input element. You can also define the time zone for the value, min, and max attributes — for example, UTC+06:00.

**DateTime Local.** This creates an input element with the “datetime-local” type attribute. This field allows users to select a date and time (with no time zone).

**Email.** This creates an input element with the “email” type attribute. Users should enter an email address in this field.

In addition to the same attributes that you can define for a textfield input element, you can also check the **Multiple** option in the Properties panel to allow users to enter multiple values in the field (separated by a comma). This attribute is a boolean value; if true, the multiple attribute is added to the `<input>` tag:

```html
<input type="email" name="email" multiple>
```

On supporting mobile browsers, bringing an email-type field into focus adds the `@` and `.` options to the on-screen keyboard.

**Fieldset.** This button creates a fieldset element by wrapping field objects with opening and closing `<fieldset>` tags. Fieldsets can be used to combine multiple form fields into a group; you can define a legend (using `<legend>` tags) for the grouped form objects.

**File.** This creates an input element with the “file” type attribute. A file field is simply a text field with an attached Browse button, which allows users to select a file and upload it to your server. (If you include a file field in your form, the selected file uploads to the server using the POST method; you cannot use the GET method.)

**Hidden.** This creates an input element with the “hidden” type attribute. The field does not appear in the browser window; you can use these in scripts to pass information that is transparent to the form user.
**HTML Form Fields in Depth (continued)**

**Image Button.** This creates an input element with the "image" type attribute. The selected image functions as a submit button for the defined form.

**Label.** This button creates opening and closing <label> tags at the location of the insertion point. If something is selected when you click this button, the label tags wrap around the existing selection.

**Month.** This creates an input element with the "month" type attribute. Users can choose a specific month from the pop-up selector.

**Number.** This creates an input element with the "number" type attribute. Users can click the field's arrows to increase or decrease the number in the field. In the Properties panel, you can define all the same attributes as for a textfield input element. You can also define Min, Max, and Step values, which have the same function as for a date input element.

**Password.** This creates an input element with the "password" type attribute. Content entered in this field appears as asterisks or bullets (dots).

**Radio Button.** This creates an input element with the "radio" type attribute. A radio button has a yes-or-no value — it is either selected or not selected.

**Radio Button Group.** A radio group is simply a group of radio buttons with the same name attribute. When you click the Radio Button Group button in the Form Insert panel, a dialog box opens where you define the items in the group. (Unlike a checkbox group, which allows multiple selections, users can select only one option in a radio group.)

**Range.** This creates an input element with the "range" type attribute. Users can drag a slider to define specific value. Properties for this kind of field are the same as for the textfield and number types of field.

**Search.** This creates an input element with the "search" type attribute. This field behaves like a textfield type element.

**Tel.** This creates an input element with the "tel" type attribute. This type if field is intended to be used to gather users’ phone numbers.

**Textarea.** This creates a textarea element, which is simply a multi-line text field. In the Properties panel, you can define:
- **Rows and Cols**, or the number of rows and columns that are visible in the area
- **Max Length**, or the maximum number of characters that can be entered in the area.
- **Wrap**, or how text should be displayed inside the field.

**Time.** This creates an input element with the "time" type attribute. Users can enter the time in a specially formatted field.

**Url.** This creates an input element with the "url" type attribute. The value of the url field is automatically validated when the form is submitted. On supporting mobile browsers, the bringing a url-type field into focus adds the ".com" option to the on-screen keyboard.

**Week.** This creates an input element with the "week" type attribute. Users can select a specific week from the pop-up selector.
**Apply CSS to the Form**

To effectively format forms using CSS, you should remember that several HTML tags are used for most objects in the form:

- The `<form>` tag encloses the entire form.
- The `<input>` tag identifies each form object that allows user input. Different types of form objects are identified with the `type` attribute, such as:
  ```html
  <input name="last" type="text" id="last" />
  <input type="reset" name="reset" id="reset" value="Reset" />
  ```
- The `<select>` tag creates drop-down menus and selection lists.
- The `<label>` tag creates the text that identifies form fields (including the text of checkboxes).

   
   It is a good idea to use the Live view to preview the accuracy of the CSS selectors and properties you define.

2. *In the Design pane, click to select the label for the checkbox field.*

3. *Click the Current button at the top of the CSS Designer panel to show only selectors related to the current insertion point.*

   The editable region where the form object is placed is inside the section `#right_column` element. Each line in this form is also tagged as a paragraph element (using the `<p>` tag). So, the `#right_column p` selector controls the default appearance of form labels.
4. Click the `<p>` tag in the Tag Selector to select the entire paragraph, then choose Heading 2 in the Format menu of the Properties panel.

After applying the Heading 2, the first line is now tagged with the `<h2>` tag instead of the `<p>` tag; the applied formatting is now defined by the `#right_column h2` selector.

CSS rules apply to content within a form object just as they do to content in a regular page area. The only difference is that you now have additional tags that can be defined to format specific form objects.

5. In the CSS Designer panel, turn off the Current mode. Click the Add Selector button in the Selectors section of the panel, then change the name of the new selector to `input`. Define the following properties for the new selector:

   - width: 100%
   - margin-bottom: 10px
   - float: left
   - clear: both

   This selector is a tag selector, defining properties for all `<input>` tags. The input objects are now attached to the left edge of the containing area. The `clear` property should force the labels onto their own lines; the `clear` property only applies to other floating objects, but the labels have not yet been specifically defined as floating.

When the Live view is active and a selector is selected in the CSS Designer panel, all items affected by the active selector are highlighted in the document window. This makes it easier to identify elements that would be affected by changing properties in the active selector. (You might have to click the selector name in the panel after defining the properties to show the affected elements in the Design view.)
6. Create another new tag selector named **label** and define the following properties:

- **width**: 100%
- **float**: left
- **clear**: both

![Image of the previous code](image1.png)

These input and label tag selectors did most of the formatting work for you. You should notice a few issues that need to be addressed:

- The settings in the **input** tag selector also affected the actual checkbox object and both buttons.
- The settings in the **label** tag selector also affected the label of the checkbox.

To override these selector properties in only certain input and label tags, you will create a class selector.

7. Create a new class selector named **.noClear** in the **style.css** file. Define the following properties:

- **width**: auto
- **clear**: none

The auto value allows the object to occupy only what space is necessary (rather than the defined percentages in the **input** and **label** tag selectors).

8. In the Design pane of the document window, click to select the checkbox object at the top of the form. Use the Element Display to add the **.noClear** class to the selected object.

Selecting the object in the Live view adds a checkmark to the box; this has no effect on the page code, simply provides a reflection of what users will see in a browser window.

There is not much apparent difference from applying the class, however, because you also need to apply the class to the checkbox label.

![Image of the modified code](image2.png)
9. **Click to select the checkbox label.** Using the Element Display, apply the .noClear class to the label.

   After applying the class to the `<label>` tag, the clearnone property allows the name label to move into the space to the right of the checkbox label. You need to create another selector to solve this problem.

![Checkbox Label](image1.png)

10. **Using the Element Display, apply the noClear class to the label of the select element.**

   If, at any time, you don’t see the Element Display when you select an item in the Live view, open the Live View Options menu in the Document toolbar and toggle off the Hide Live View Displays option.

![Select Element](image2.png)

11. **Apply the noClear class to both buttons at the bottom of the form.**

   Because the noClear class allows objects’ width to automatically size, the two buttons no longer fill the width of the containing area; they again appear on a single line.

![Buttons](image3.png)

12. **Choose View > Live View Options > Hide Live View Displays.**

   Toggling the Hide Live View Displays item hides the element highlighting that occurs when you click a selector in the CSS Designer panel, as well as the Element Displays that appear when you select an element in the Live view.

![Hide Live View Displays](image4.png)
13. Review the final form in the Live view.

14. Save the open HTML file and the style.css file, then close the HTML file.

15. Export a site definition named **cuppcaes.ste** into your WIP>Cupcakes folder, and then remove the site from Dreamweaver.
1. A(n) ____________ can be used to change the appearance of multiple elements on a single page, including elements of different types, once it is applied to those elements.

2. You can make a(n) ____________ editable in a template to protect the container’s contents in pages where that template is applied.

3. The _______________ property can be used to prevent other floating objects from appearing on the same line as an element.

4. The _______________ tag identifies a text field in a form.

5. The _______________ type of text field displays entered text as asterisks or dots.

6. A(n) ____________ can be used to place one HTML file into another.

7. A(n) ____________ attribute has only two possible values: true or false.

8. The _______________ tag creates the text that identifies form fields (including the text of checkboxes).

9. The _______________ tag creates a selection list.

10. The _______________ determines the text that appears on a Submit button.

1. Briefly explain the concept of a class in CSS.

2. Briefly explain the float CSS property

3. Briefly explain the clear CSS property
Use what you learned in this project to complete the following freeform exercise. Carefully read the art director and client comments, then create your own design to meet the needs of the project. Use the space below to sketch ideas; when finished, write a brief explanation of your reasoning behind your final design.

You have been hired by the Green California Initiative, a non-profit organization dedicated to increasing the use of alternative energy sources in private homes. The group wants to create a simple Web page to promote the benefits of home solar power and allow people to request more information.

- Download the Solar_DWCC17_PB.zip archive from the Student Files Web page to access provided client files.
- Find or create artwork to support the client’s message.
- Design a Web page to present the client-supplied content in a clear, attractive manner.
- Create a form so users can request more information about installing solar power on their homes.

Our group does not sell anything or promote any specific company. Instead, we try to encourage people to consider the benefits of reducing energy consumption from the main power grid.

The new page will feature home solar power, which is a perfect source for Southern California homes with our abundant sunshine. The page needs to include our logo, a brief text explanation, and a short form with fields to:

- Gather contact information (name, mailing address, phone, and email address).
- Determine home size, household income, and average monthly electric bill.
- Determine how users learned about Green California Initiative.
- Determine how much interest each user has in installing alternative energy (use a 0–5 scale)

Project justification:

[Blank lines for justification]

259
In this project, you learned more about CSS selectors — specifically, using class selectors to control specific elements in a page. Because of the versatility of using classes, some professional Web designers use only classes to define an entire site layout. You also learned several different techniques for editing CSS selectors, both in the CSS Designer panel and in the Code pane. Once you are familiar with all of the options, you can better determine which method suits your working style.

This project also expanded on your knowledge of Dreamweaver templates. You learned how to make only a specific attribute editable, allowing different options on individual pages while protecting placed content.

Finally, this project introduced the concept of HTML forms, including using CSS to control the appearance of various elements in a form.