Computer Science 140

Lab 4

Due Start of lab week of Oct 29-31

Purpose

1. Demonstrate understanding of HTML form elements
2. CSS styling practice using `div` and `span` selectors

Overview

Create the HTML necessary for an order form, then apply CSS styles to the form.

Resources

1. Read the chapter about HTML forms in the Learning Web Design book.
2. Online lecture notes on HTML form element definition
3. Review the HTML form example below. View the source of this HTML:

http://www.cs.camosun.bc.ca/~langs/comp140-13/labs/forms/form.html

Description

The owners of the Pizza Palace would like an online order form on their web site so that customers can purchase pizzas for delivery or pick-up. There are several options for customers to consider when purchasing their pizza(s) in a partial list indicated below. For this lab you may assume that a customer can place an order for only one type of pizza with optional toppings at a time.

- Type of pizza : cheese, pepperoni, or Hawaiian (must select one)
- Size: individual, medium, or large (must select one, default is individual)
- Optional toppings: tomato sauce, mushrooms, green pepper, pepperoni, pineapple
- Quantity of pizza's to order (up to 20, default selection is one pizza)
- Pickup or Delivery (must select one)

The requirement for this lab is to design and create a simple order form in HTML / CSS using the guidelines below. The HTML form will not yet be able to sum up totals; that work requires some JavaScript code which we have yet to discuss in the lectures.

Preparation

1. Review the sample HTML form at the URL above. It has two textbox elements for first name and last name, radio buttons for status (full-time and part-time), a selection element for country, a set of three checkbox buttons for music preference, and two buttons for Submit and Reset.

2. The PHP script programming is covered in a later course but for now you just need to understand that the HTML form's `submit` button causes the form's `action` attribute to run. The `action` is execute the `processform.php` script.

3. The sample `form.html` file contains CSS styling for spacing and colouring effects of the form elements. The same form page (without any CSS styling) is at

   http://www.cs.camosun.bc.ca/~langs/comp140-13/labs/forms/form_noCSS.html

Process:

1. Open the File Explorer and create the following new folders on your H: drive:

   ```
   comp140\lab04
   comp140\lab04\PizzaPalace
   comp140\lab04\PizzaPalace\images
   comp140\lab04\PizzaPalace\script
   ```

   On the H: drive copy all your previous lab 3 work (the html files and the image files) into your new lab 4 folders.

2. Open WinSCP and copy the new lab04 folder to your `deepblue` account inside the `public_html\comp140` folder.

3. Open DreamWeaver and create a new web site `PizzaPalace4`, which points to your new lab 4 on deepblue. Refer back to your lab 2 notes if you need to review how to do this step. Verify the new lab 4 web site appears as expected in the browser.

4. Copy the `index.html` file to a new file named `order.html`. Update the list of navigation links on the page to include the new text "place order" link to this new file. Remove the original `index.html` content from the `order.html` page (the Google map, etc) within its `middlebox` div. The entire `form` element must be defined inside the `middlebox` div.

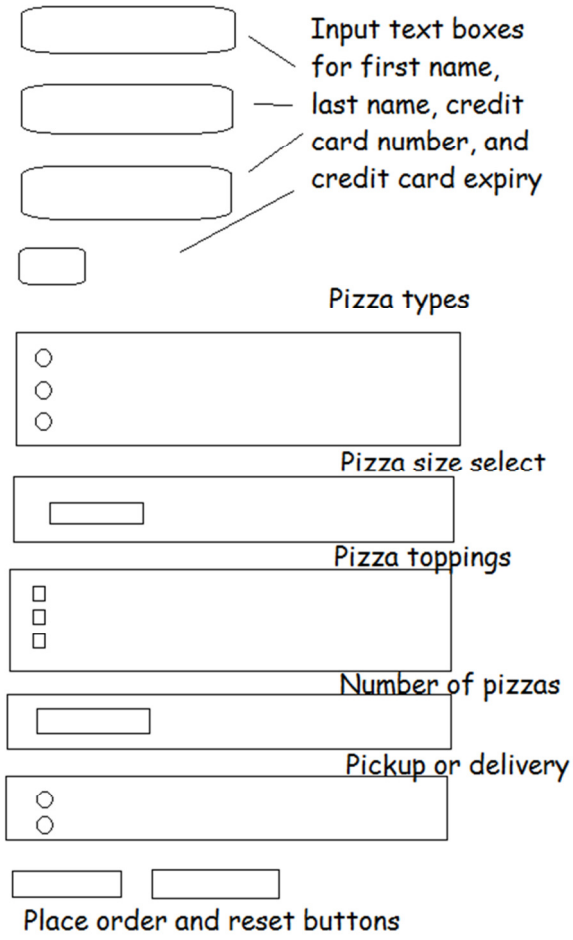
   Confirm the DOCTYPE of the HTML file is HTML5 at the top of the `order.html` file:

   ```
   <!DOCTYPE html>
   <html lang="en">
   ```

5. There is a new subfolder under `PizzaPalace4` called `script`. This folder will at some point in the future contain the PHP form handler script called `processform.php` which processes the entered form data when the form's submit button is clicked.

6. The next step is sketching out the required form components as a draft. This new order page will comprise the HTML order form containing:

a. textbox for the customer's first name
b. textbox for the customer's last name
c. textbox for the phone number (using new HTML5 element `input= "tel"`)
d. textbox for the credit card number
e. textbox for the credit card expiration date
f. radio buttons for the different pizza types (cheese, pepperoni, Hawaiian)
g. dropdown selection for the pizza sizes (individual, medium, large)
h. check boxes for the topping options.
i. number entry for the number of pizzas (1 to 20)
j. radio buttons for pickup or delivery
k. submit and reset buttons



7. The HTML for the form begins with the `form` element as follows inside the `middlebox` div.

```
1  <form name  = "menuorder"  formmethod = "post"
2        id = "formbox"
3        action = "script/processform.php">
4
```

```
5   </form>
```

This form definition goes into the content area of the `order.html` page within `middlebox`.

8. Within the form we need to define a `fieldset` to logically group the form components together.

```
1   <form name = "menuorder" formmethod = "post"
2         id = "formbox"
3         action = "script/processform.php">
4
5    <fieldset>
6       <legend>Place your order</legend>
7
8    </fieldset>
9   </form>
```

```
This is text already in your
HTML. Do not retype it.

This is new text to type
into your HTML.
```

9. Define the form elements for textboxes first name, last name, phone number, credit card, and expiry date within this `fieldset`, after the legend. If the element does not contain a `type` attribute, then `type= "text"` is used as default. Note that the value for the `name` attribute must be a single word containing no spaces, hence "`firstname`" is used, not "`first  name`".

```
1   <form name = "menuorder" formmethod = "post"
2         id = "formbox"
3         action = "script/processform.php">
4    <fieldset>
5       <legend>Place your order</legend>
6         <label>First name:</label>
7         <input type= "text" name= "firstname"
8                 required  id= "firstname"
9                placeholder= "Your first name" >
10
11        <label>Last name:</label>
12       <input type= "text" name= "lastname"
13                required  id= "lastname"
14                placeholder= "Your last name"  >
15
53   </fieldset>
54  </form>
```

The HTML5 attribute named `required` tells the browser to only submit the form if the field in question is filled out correctly. If a required field is empty or invalid, then the form will fail to submit and focus will move to the first invalid form element. Browsers Opera, Firefox, and Chrome provide the user with error messages. The

example messages "Please fill out this field" or "You have to specify a value" if left empty are shown when the data type or pattern is invalid.

The HTML5 attribute named `placeholder` allows a short text hint to be displayed inside the form element, space permitting, informing the user what data should be entered in that field.

Browsers which do not support the new HTML5 attributes will ignore them. This is known as "degrading nicely".

10. In this lab the phone form control (also required) is a special type of input – it is not `text` type but `tel` type.

11. The phone number form element should be required and made to accept only numbers in the format 999 999-9999. In the HTML5 specification form elements can be checked for input data format using the new pattern attribute. The pattern uses regular expressions[1] to describe the format of valid input. For this phone number element use the attribute `pattern='[\(]\d{3}[\)]\d{3}[\-]\d{4}'.` Define the `title` attribute for the phone number describing the expected format as (250) 999-9999. Define the `name` attribute as `phone`. Define the placeholder as "(250) 999-9999".

12. The Credit Card form element is type `text` and has attribute `pattern='[0-9]{4,6}'` indicating that between 4 to 6 digits must be entered. No space between – and 9. Define the `name` attribute as `credit`. Set the place holder to "Credit card number".

13. The Expiry Date (MM-YY) form element is type `text` and has the pattern of exactly two digits, a hyphen, two digits. Define the `placeholder` as "`01-13`". Define the `name` attribute as `expiry`. Define the `title` attribute as "Credit card expiry as MM-YY".

14. If the user clicks the Submit button with an invalid data pattern entered on a form element, the Chrome browser will display an appropriate error message.

15. The radio buttons for the pizza type can be defined as follows and are entered on the line following the expiry date form element definition and **above** the `</fieldset>` tag on line 13 shown above.

```
22      Pizza Type
23
24      <input type= "radio"     name= "pizza"
25             value= "cheese"   id= "cheese" >
```

[1] Tutorial on regular expressions:
http://www.codeproject.com/Articles/9099/The-30-Minute-Regex-Tutorial

```
26
27        <label for="cheese"> Cheese </label>
28
29        <input type= "radio"      name= "pizza"
30               value= "pepperoni" id= "pepperoni" >
31
32        <label for="pepperoni"> Pepperoni </label>
33
34        <input type= "radio"      name= "pizza"
35               value= "hawaiian"  id= "hawaiian">
36
37        <label for="hawaiian"> Hawaiian </label>
```

The `type="radio"` attribute indicates that this is an HTML radio button element. Since all three choices of pizza types (cheese, pepperoni, Hawaiian) belong as one group, each of the radio button elements must share the same `name` attribute (such as "`pizza`"). The `value` attribute is the selected radio button information returned back to the processing script when the user clicks the form's `submit` button. The `id` attribute is bound to the associated label element's `for` attribute and can help with CSS styling.

16. The form elements for pizza size follow next. The `select` form element is used here to select one of individual, medium, or large, with individual as the default.

```
39         Pizza Size
40
41        <select name= "size" id= "size">
42           <option value= "individual" selected= "selected">
43             Individual
44           </option>
45
46           <option value= "medium">
47             Medium
48           </option>
49
50           <option value= "large">
51             Large
52           </option>
53
54        </select>
```

17. The form elements for the optional pizza toppings follow next. The square brackets after the name attribute help the PHP script work with optional lists of values.

```
56    Pizza Toppings
57
```

```
58    <input type= "checkbox" name= "topping[]"
59           value= "tomato"  id= "tomato" >
60
61    <label for= "tomato"> Tomato Sauce </label>
62
63    <input type= "checkbox"      name= "topping[]"
64          value= "greenpepper"  id= "greenpepper" >
65    <label for= "greenpepper"> Green Pepper </label>
```

… repeat for the other toppings: mushroom, pepperoni, and pineapple. For pepperoni topping, set its id attribute to "pepptopping" instead of "pepperoni" because you already defined an element id= "pepperoni" for the pizza type and id values must be unique in the HTML document to work properly.

18. The form elements for the pizza quantity follow next. Since a number of pizzas from 1 to 20 could be ordered, the best HTML form element to use is the new HTML5 `type= "number"`.

```
Pizza Quantity

<input type= "number" id= "quantity"
       name= "quantity" value= "1" min= "1" max= "20"
       required>
```

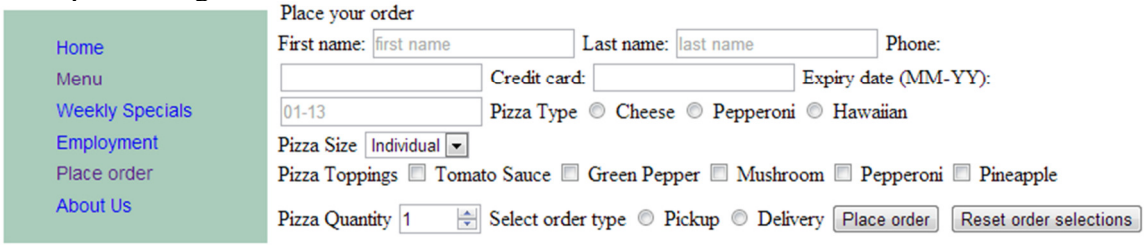
19. The pizza order can be either pickup or delivery.

```
Select order type

<input type= "radio"    name= "order"
       value= "pickup"  id= "pickup">

<label for="pickup"> Pickup </label>

<input type= "radio"      name= "order"
       value= "delivery"  id= "delivery" >

<label for="delivery"> Delivery </label>
```

20. The bottom of the form will show the submit and reset buttons. These HTML elements should appear just above the `</fieldset>` tag.

```
<input type= "submit" name= "submit"
       value= "Place order" >

<input type= "reset" name= "reset"
       value = "Reset order selections" >
```

21. Press ctrl-S to save your work, then press the F12 function key to preview it in the browser. The result may look something like the following with all the form elements pushed together.



Part of this can be fixed by using more `fieldset` tags to organize the radio buttons and checkboxes together. The CSS styles we add in later will further fix this appearance.

22. For the Pizza Type add in the following HTML `fieldset` and `legend` elements:

```
Pizza Type

 <fieldset>
      <legend>Select type of pizza</legend>

  <input type= "radio"     name= "pizza"
         value= "cheese"   id= "cheese" >
  <label for="cheese">Cheese</label>

  <input type= "radio"      name= "pizza"
         value= "pepperoni" id= "pepperoni" >
  <label for="pepperoni">Pepperoni</label>

  <input type= "radio"      name= "pizza"
         value= "hawaiian"  id= "hawaiian">
  <label for="hawaiian">

</fieldset>  <!-- Pizza type fieldset -->
```

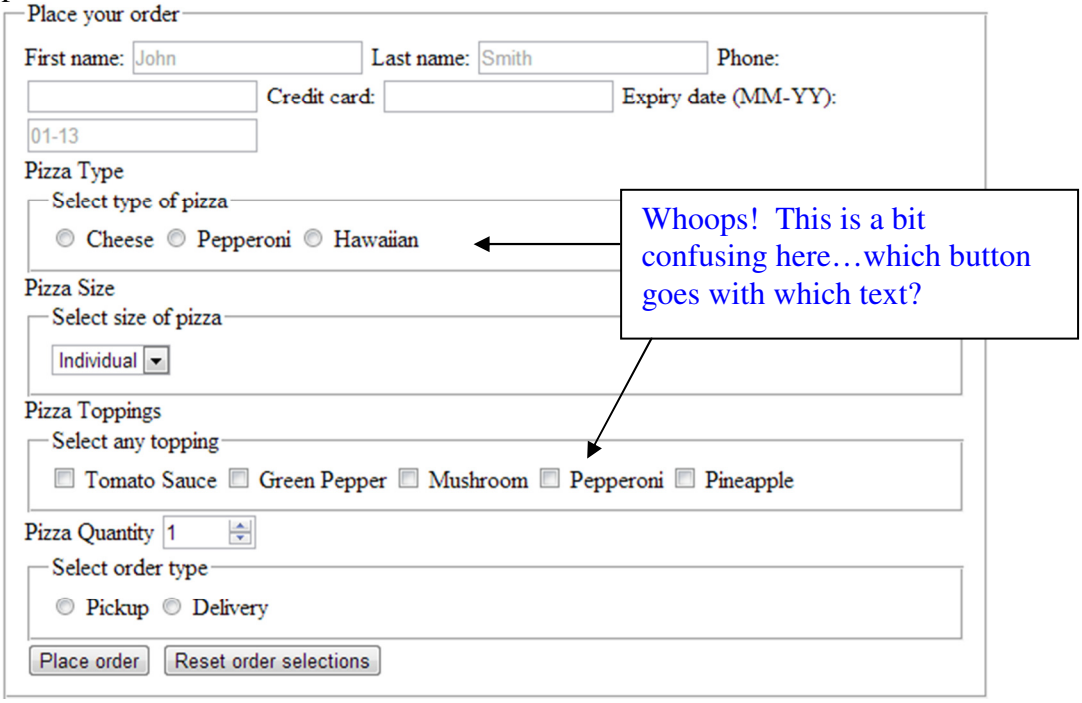
23. For the Pizza size add in the following HTML `fieldset` and `legend` elements:

```
Pizza Size

 <fieldset>
       <legend>Select size of pizza</legend>

 <select name= "size" id= "size">
    <option value= "individual" selected= "selected">
```

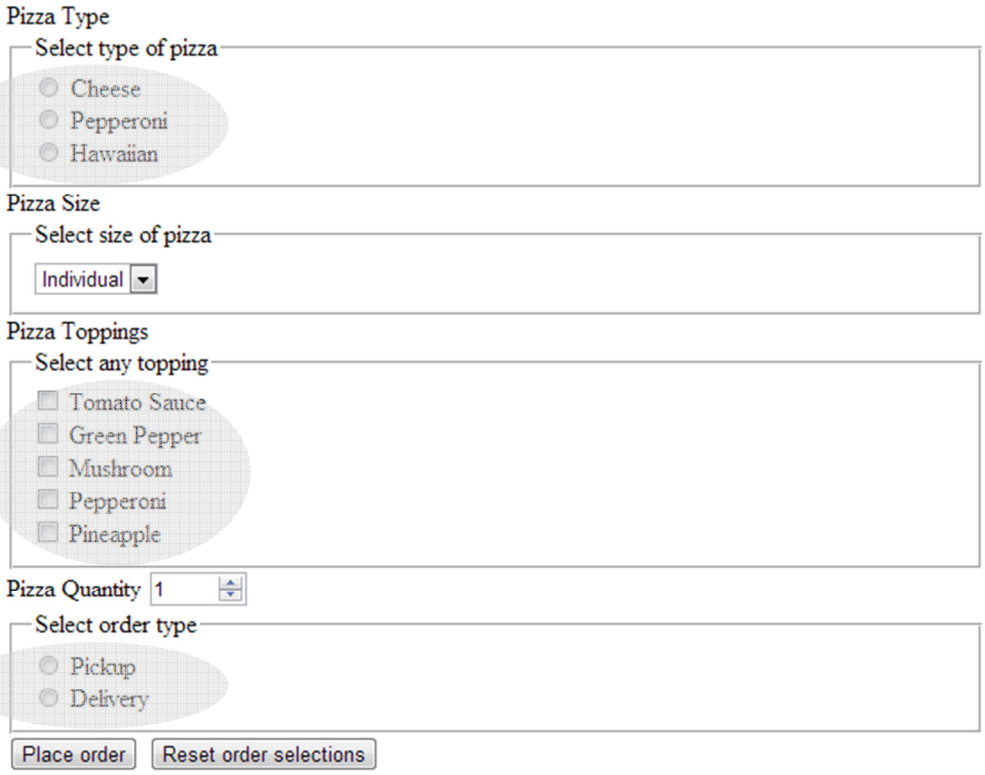
```
      Individual
    </option>

    <option value= "medium">
      Medium
    </option>

    <option value= "large">
      Large
    </option>

  </select>
</fieldset>
```

24. Add in the required `fieldset` and `legend` HTML elements for the pizza toppings and the order type (pickup or delivery).

25. The preview of the form will resemble this outline:



26. Since it appears confusing to show a button between two possible text options, we will place each radio button and checkbox option separately on a line. The easiest way to accomplish this in HTML is to add a line break tag `<br>` between the option elements. Repeat for the pizza type, pizza toppings and order type. Adding the line break HTML element should result in this display:



27. The browser preview will show the elements on separate lines as in the figure above. If the preference is to show the text before the radio button or checkbox, then reverse the order of the text and the form element so that the text is before the element. Usually the better format is to show the checkbox or radio button in front of the text.

28. The next step is to apply CSS style. Each form component such as first name, last name, phone number, etc will appear on a separate line on the screen; there will be a CSS class called `row` defined to do this. Within each row there will be a style for the form component label (class "`form_label`") and a style for the form component element (class "`form_element`").

Create a new file called `form.css` (in the same folder containing the `order.html` file) and add the following CSS to it:

```
/* CSS style for the entire form */

#formbox { width: 620px;
           background-color: #cc9;
           border: 1px #333;
           padding: 5px;
           margin: 0px auto;
         }

/* CSS style for each row in the form. */
```

```
div.row {
            clear: both;
            padding-top: 10px;
         }


/* CSS style for the left side of the form. */

div.row span.form_label {
                     float: left;
                     width: 100px;
                     text-align: right;
                   }

/* CSS style for the right side of the form. */

div.row span.form_element {
                      float: right;
                      width: 460px;
                      text-align: left;
                   }
```

29. Add the appropriate `<link>` element definition inside the `<head>` element for `order.html` so that the new `form.css` file is used. This new link element can be defined on a line after the link element for `default.css` is defined.

    Preview the `order.html` page in a browser to confirm it is using the new CSS styles defined in `form.css`.

30. Add in the form element `<div class= "row"> ... </div>` for each form component starting with the pizza type as follows:

```
<div class= "row">

 Pizza Type
  <fieldset>
      <legend>Select type of pizza </legend>
     ...
  </fieldset>  <!-- Pizza type fieldset -->
</div>  <!-- end div row -->

<div class= "row">

 Pizza Size
```

```
    <fieldset>
        <legend>Select size of pizza </legend>
        ...
    </fieldset>  <!-- Pizza size fieldset -->
</div>  <!-- end div row -->

<div class= "row">

    Pizza Toppings
    <fieldset>
        <legend>Select any topping </legend>
        ...
    </fieldset>  <!-- Pizza type fieldset -->
</div>  <!-- end div row -->

<div class= "row">

    Pizza Quantity
    <input ...

</div>  <!-- end div row -->

<div class= "row">

    Select order type
    <input ... (the two radio buttons for pickup / delivery)

</div>  <!-- end div row -->
<div class= "row">

    <input ... (the two buttons for submit and reset)

</div>  <!-- end div row -->
```

Do not place any `div` elements between the individual radio buttons or checkboxes. When the `div` rows are all defined, the form's preview in the browser should resemble this figure.

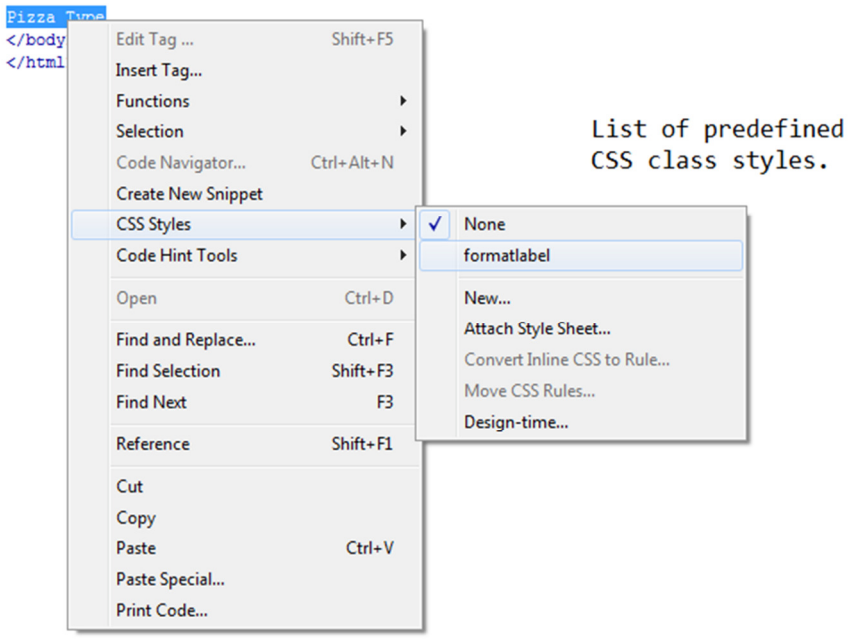
31. The next step is to place `span` element definitions inside each of the `div class="row"` elements so that the labels and form elements are positioned correctly. Form labels line up on the left side of the form; form controls on the right. Examine the below example carefully as it shows where the `<span>` elements go.

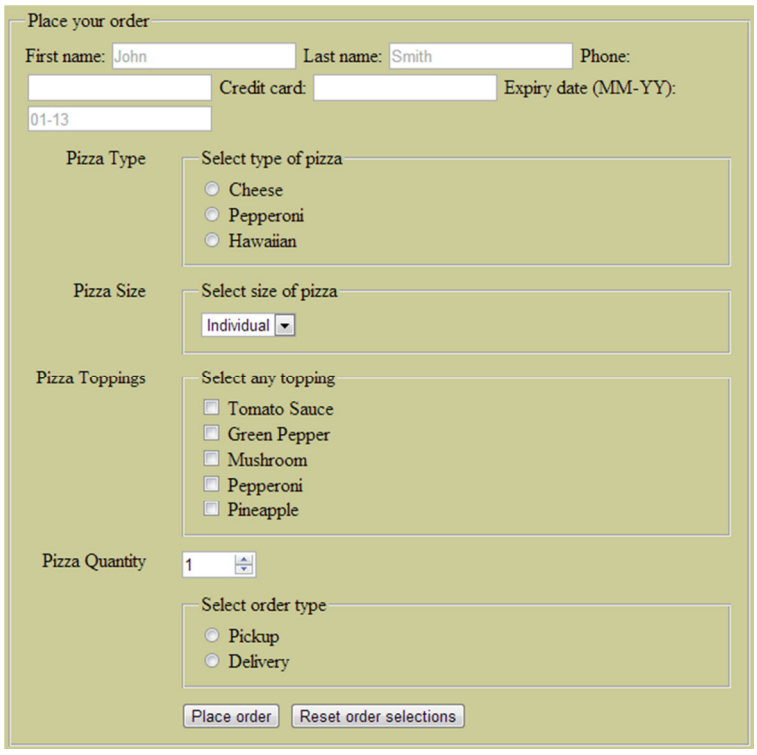
```
<div class= "row">
   <span class= "form_label">
      Pizza Type:
   </span>
   <span class= "form_element">
      <fieldset>
       <legend>Select type of pizza </legend> ...etc
             ...
      </fieldset>
   </span>
</div> <!-- end div row -->
```

… repeat the span definitions for the rest of the form

Do not place the `<span>` definitions inside the `<fieldset>` elements or the fieldset border will extend out a bit too far.

In Dreamweaver you can highlight the label text, right-click, and select the CSS style form_label from the context menu to apply the proper span class definition.





32. The input `type = text` and `type = tel` form elements can be styled with a minimum height and width, a margin, border radius, and an outline CSS properties. Define these <u>embedded</u> CSS styles within the order.html document.

```
input[type=text], input[type=tel] {
  font-family: Helvetica, Arial, sans-serif;
  border:1px solid #ccc;
  font-size:20px;
  width:200px;
  min-height:30px;
  display:block;
  margin-bottom:15px;
  margin-top:5px;
  outline: none;

  -webkit-border-radius:5px;
  -moz-border-radius:5px;
  -o-border-radius:5px;
  -ms-border-radius:5px;
  border-radius:5px;
}

#expiry  {
   width:100px;  }
```

33. The type = `submit` and type= `reset` form elements can be styled with some padding white space and no background:

```
input[type=submit], input[type=reset] {
  background:none;
  padding:10px;
}
```

34. New CSS3 properties include support for `required` and `invalid` form value entries:

```
:invalid {
  border-color: #e88;
  -webkit-box-shadow: 0 0 5px rgba(255, 0, 0, .8);
  -moz-box-shadow: 0 0 5px rgba(255, 0, 0, .8);
  -o-box-shadow: 0 0 5px rgba(255, 0, 0, .8);
  -ms-box-shadow: 0 0 5px rgba(255, 0, 0, .8);
  box-shadow:0 0 5px rgba(255, 0, 0, .8);
}

:required {
  border-color: #88a;
  -webkit-box-shadow: 0 0 5px rgba(0, 0, 255, .5);
  -moz-box-shadow: 0 0 5px rgba(0, 0, 255, .5);
  -o-box-shadow: 0 0 5px rgba(0, 0, 255, .5);
  -ms-box-shadow: 0 0 5px rgba(0, 0, 255, .5);
```

```
    box-shadow: 0 0 5px rgba(0, 0, 255, .5);
  }
```

35. Download the images from the lab web page (http://hal.cs.camosun.bc.ca/~langs/comp140-13/labs/index.html) into the `images` folder. There should be three image files: `valid.png, invalid.png`, and `red_asterisk.png`.

36. CSS3 provides a rich style variety for online form creation. Enter this set of CSS styles. These pseudo-classes are used when the first five form elements have focus and have valid (or invalid) entry values.

```
input:required:valid {
   background: #fff url(images/valid.png) no-repeat 98% center;
   box-shadow: 0 0 5px #5cd053;
   border-color: #28921f;
}

input:required {
  background: #fff url(images/red_asterisk.png)
              no-repeat 98% center;
 }

input:focus {
  background: #fff;
  border: 1px solid #555;
  box-shadow: 0 0 3px #aaa;
  padding-right: 20px;
}

input:focus:invalid {
  background: #fff url(images/invalid.png) no-repeat 98% center;
  box-shadow: 0 0 5px #d45252;
  border-color: #b03535;
}
```

37. Enter this CSS at the end of the declaration for `input[type=text], input[type=tel]`

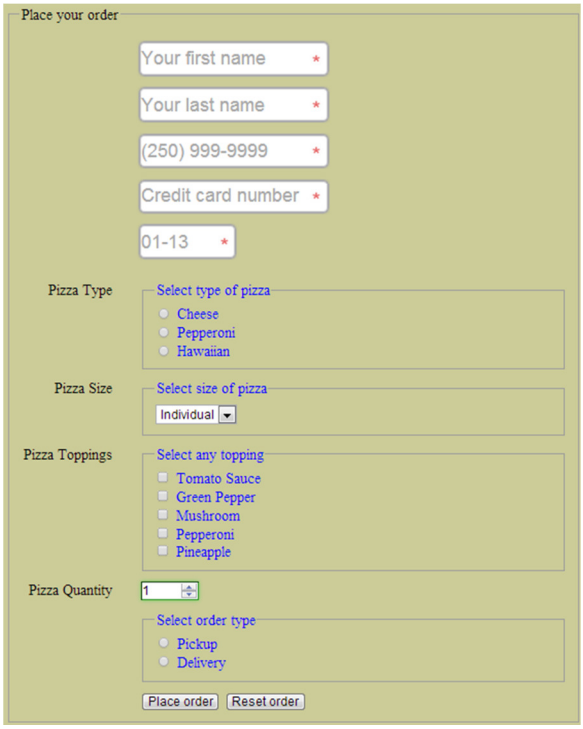
```
        -moz-transition: padding .25s;
        -webkit-transition: padding .25s;
        -o-transition: padding .25s;
        transition: padding .25s;
```

    This makes the focus form element grow wider.

38. The first five form elements (name, phone, credit card) would look better on the form if they are contained within the span `form_element` rather than appear flush on the left. Add in the following lines after the `<legend> Place your order </legend>`

```
4   <fieldset>
5      <legend>Place your order</legend>
6         <div class= "row">
7           <span class="form_element">
8
9              <input type= "text" name= "firstname"
10              required  id= "firstname"
11              placeholder= "Your first name" >
12
        … (the next four form elements up to credit card)
23          </span>
24        </div>
```

39. The preview in the browser will resemble something like the figure below.



40. [optional] The `submit` and `reset` buttons can be styled using CSS3. Replace the existing input type= "submit" form definition of the `submit` form element with:

```
<button class="submit" type="submit">Place order</button>
```

and include the CSS button style definitions which are found on the lab web page. The Reset button can also be styled in a similar way.

41. [optional] The `processform.php` script can be downloaded from the lab page web site and placed in a folder named script. If your form elements have all their name attributes defined correctly, the submit button will show all the entered form data back to you. In addition within the script folder you can create a blank text file called `myOrders.txt` and assign it write permission to "others" to save your order data there (see instructor how to set that up).

Hand In :

1. [20 marks] When you have completed the lab work (CSS HTML form styled with CSS) send the instructor an email message (langs@camosun.bc.ca) with the subject: Comp 140 Lab 4. The body of the message should contain the URL to your work above, e.g. the URL

   `http://deepblue.cs.camosun.bc.ca/~cst0xx/comp140/lab04/PizzaPalace`

2. [5 marks] Questions to be answered in a text file email attachment:
   a. What is the benefit of validating elements within a form (for example, checking that required form elements must be filled before submitting)?

   b. In general what are some ways a form can indicate which element currently has focus?

   c. Which HTML form element would you recommend be used for prompting the following information:
      i. home address (e.g. 101 North Street)
      ii. postal code (e.g. V7A-3G1)
      iii. favourite ice cream flavours (can select up to 6 out of 12 different kinds)