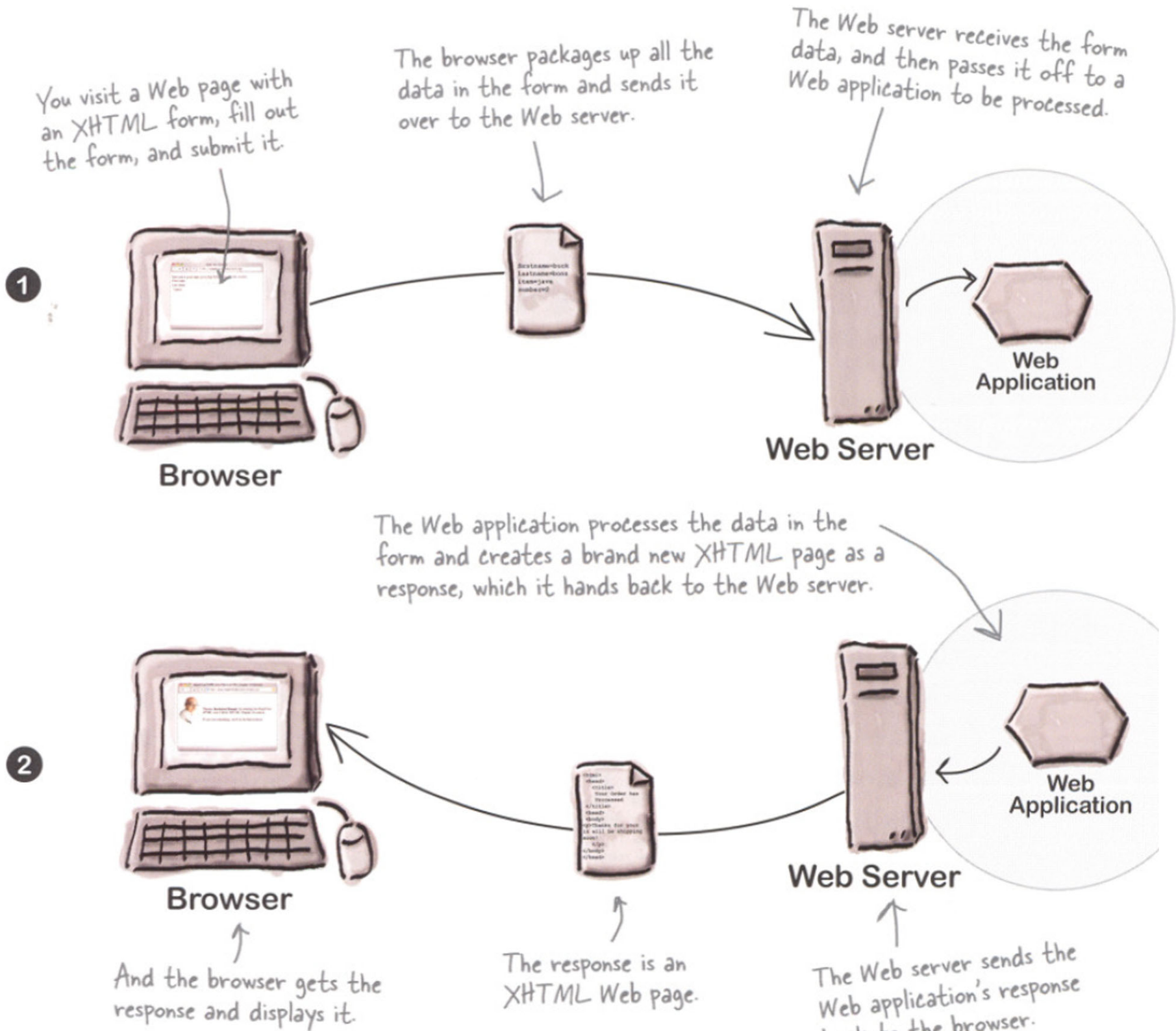
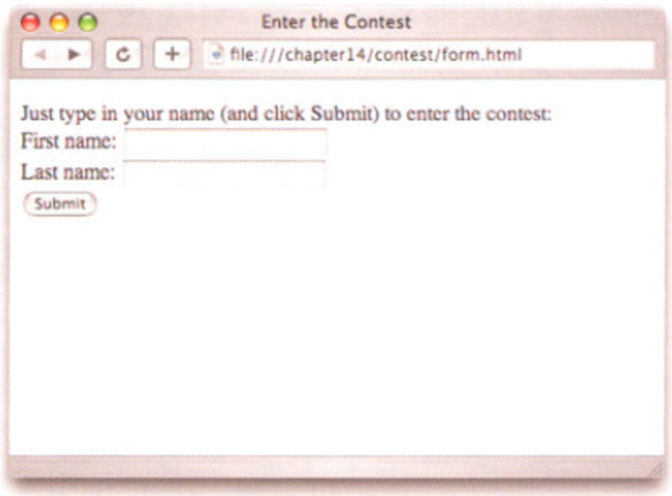
Cap 14

forms

How forms work



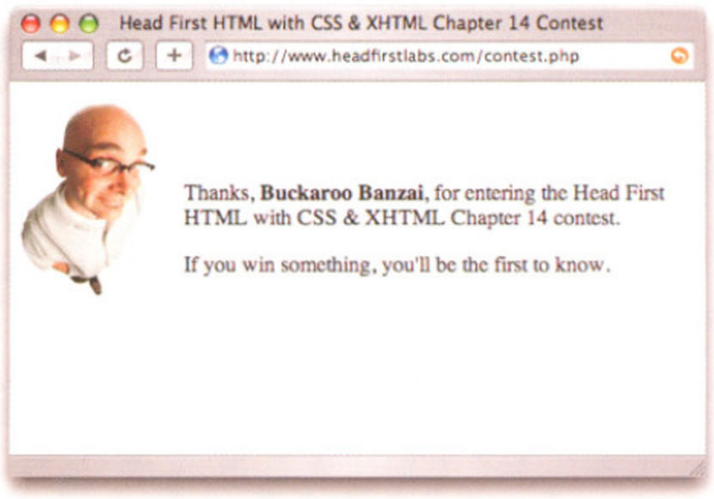
The browser loads the page



You enter data

You submit the form

The server responds



What you write in XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type"
            content="text/html; charset=ISO-8859-1" />
      <title>Enter the Contest</title>
   </head>
   <body>

      <form action="http://www.headfirstlabs.com/contest.php" method="POST">
Ⓐ        <p>Just type in your name (and click Submit) to
            enter the contest: <br />

Ⓑ        First name: <input type="text" name="firstname" value="" /> <br />
Ⓒ        Last name: <input type="text" name="lastname" value="" /> <br />
Ⓓ        <input type="submit" />

         </p>
      </form>

   </body>
</html>
```

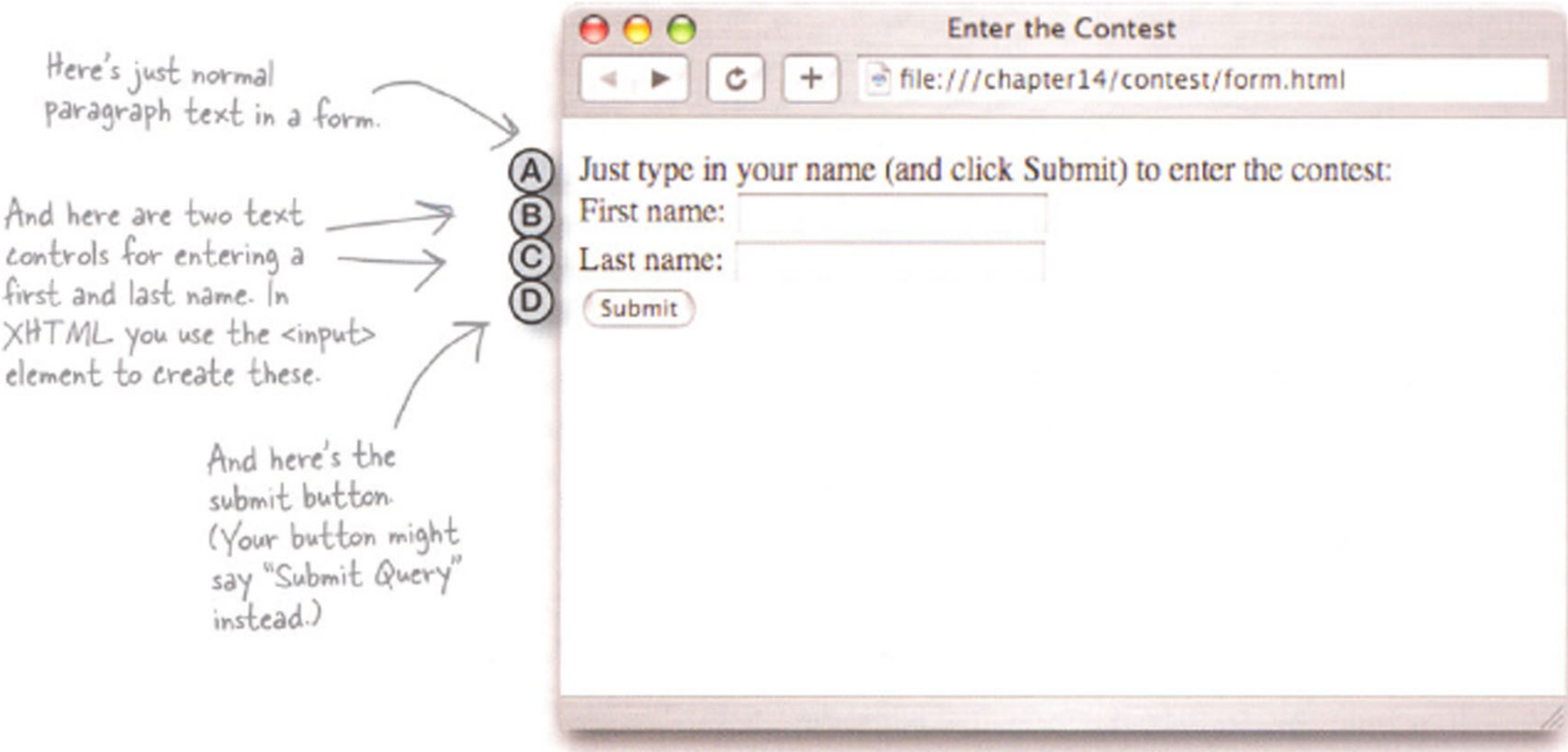
This stuff is all old hat for you now.

Here's the form.

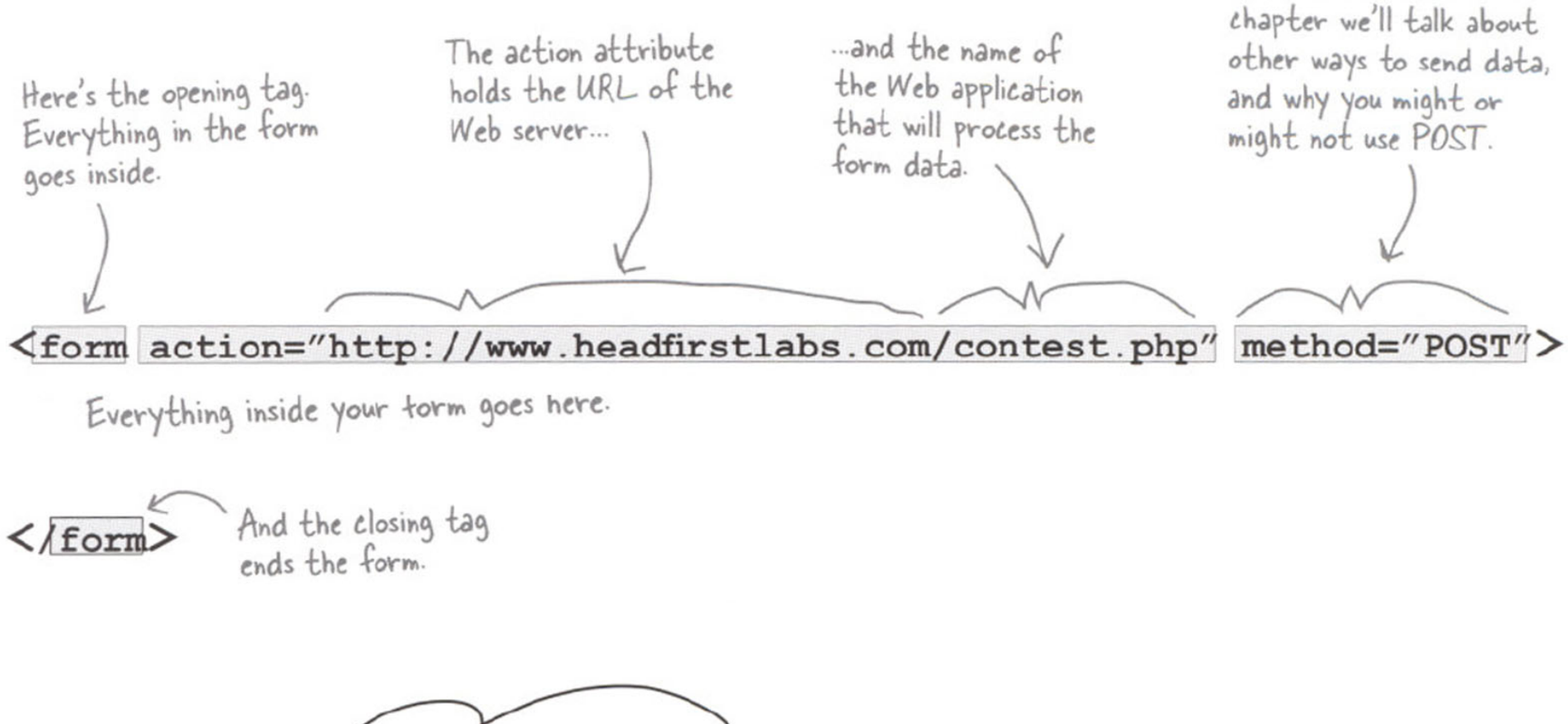
We've got the <form> element itself.

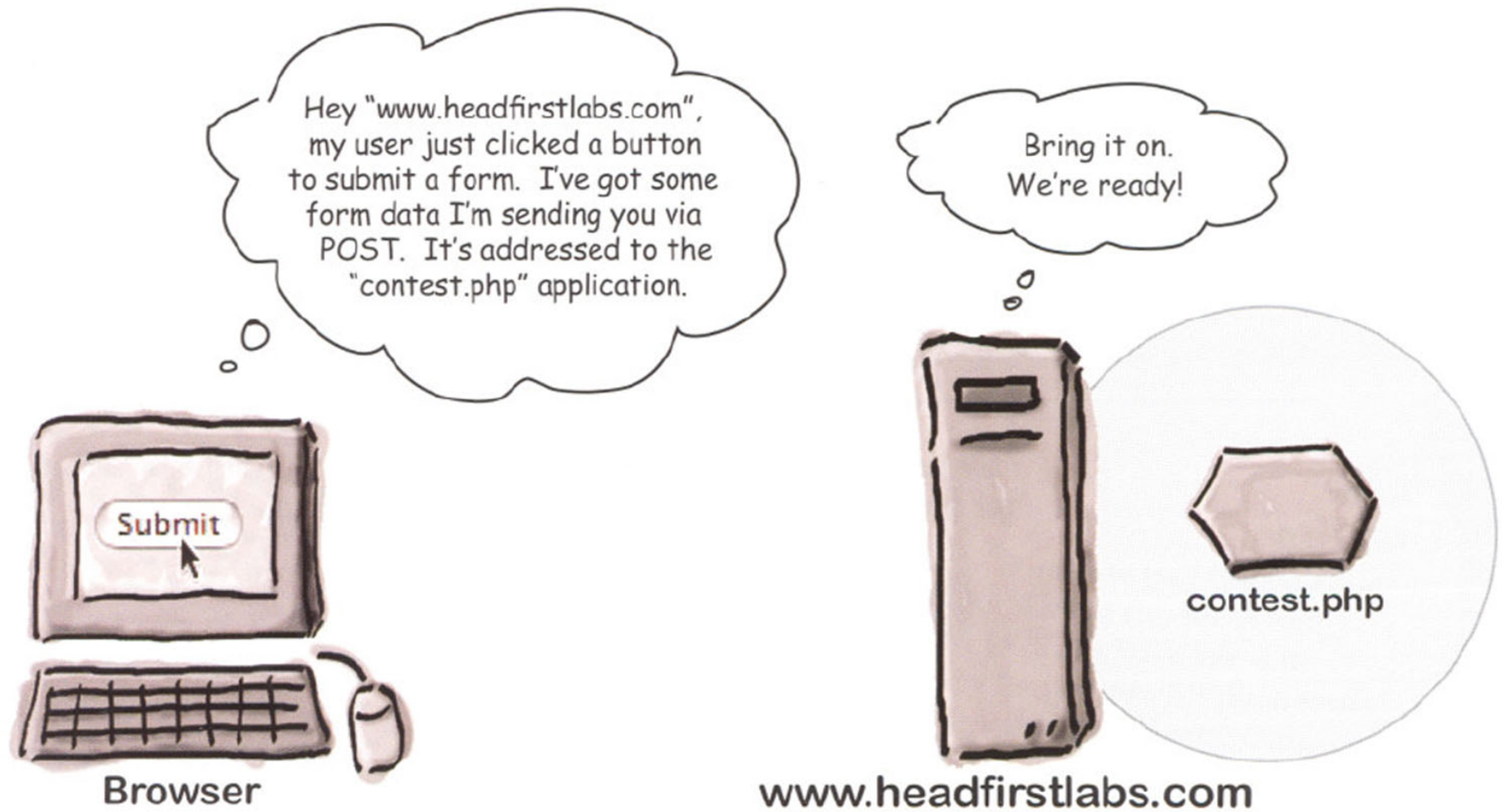
And a bunch of elements nested inside it.

What the browser create



How the form element works





What can go in a form?

text input

The text <input> element is for entering one line of text. Optional attributes let you set a maximum number of characters and the width of this control.

Name: Buckaroo Banzai

An <input> element with a type attribute of "text" creates a one-line control in the browser page.

Use the type attribute to indicate you want a "text" input.

Most form elements require a name that is used by the server script. We'll see how this works in a bit.

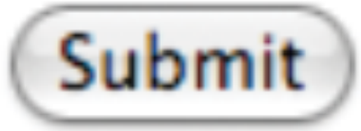
```
<input type="text" name="fullname">
```

The <input> element is a void element, so there's no content after it.

What can go in a form?

submit input

The submit <input> element creates a button that allows you to submit a form. When you click this button, the browser sends the form to the server script for processing.



The button is labeled "Submit" (or "Submit Query") by default, although you can change that (we'll show you how later).

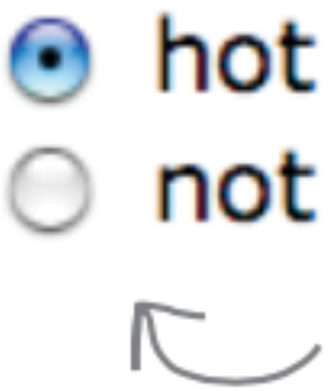
```
<input type="submit">
```

For a submit button, specify "submit" as the <input> element's type.

What can go in a form?

radio input

The radio <input> element creates a single control with several buttons, only one of which can be selected at any time. These are like old-time car radio buttons; you "push" one in, and the rest "pop out."



The radio control allows only one of a set of choices.

Use a radio <input> for each choice.

All the radio buttons associated with a given set of choices must have the same name...

...but each choice has a different value.

```
<input type="radio" name="hotornot" value="hot">
<input type="radio" name="hotornot" value="not">
```

What can go in a form?

checkbox input

A checkbox <input> element creates a checkbox control that can be either checked or unchecked. You can use multiple checkboxes together, and if you do, you can check as many or few as you like.

- [x] Salt
- [x] Pepper
- [ ] Garlic

Unlike radio buttons, a checkbox allows zero or more of a set of choices.

Like radio, you use one checkbox <input> element for each choice.

Related checkboxes also share a common name.

Each checkbox has a different value.

```
<input type="checkbox" name="spice" value="Salt">
<input type="checkbox" name="spice" value="Pepper">
<input type="checkbox" name="spice" value="Garlic">
```

What can go in a form?

textarea

The <textarea> element creates a multiline text area that you can type into. If you type more text than will fit into the text area, then a scroll bar appears on the right side.

Customer feedback:

I love my new Mini Cooper! I got the red, sporty model, and I've been zipping around town like there's no tomorrow. And, my new iPod fits perfectly in the dash drink holder. Of course, now everyone else wants one, too.

rows

cols

The <textarea> element is not an empty element, so it has both opening and closing tags.

Use the name attribute to give the element a unique name.

The cols attribute tells the browser how many characters wide to make the text area.

```
<textarea name="comments" rows="10" cols="48"></textarea>
```

The rows attribute tells the browser how many characters tall to make the text area.

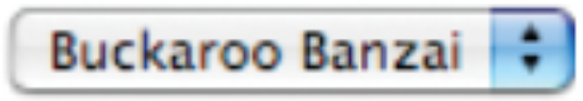
Any text that goes between the opening and closing tags becomes the initial text in the browser's text area control.

You can also specify the width and height of a textarea using CSS.

What can go in a form?

select

The <select> element creates a menu control in the web page. The menu provides a way to choose between a set of choices. The <select> element works in combination with the <option> element below to create a menu.



The select element creates a menu that looks like this (although the look will vary depending on the browser you're using).

The <select> element goes around all the menu options to group them into one menu.

Just like the other form elements, give the select element a unique name using the name attribute.

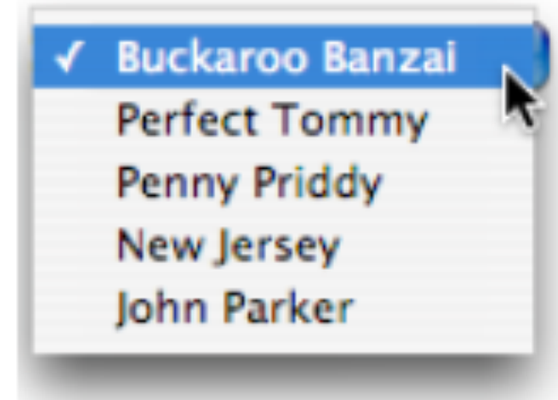
```
<select name="characters">
    <option value="Buckaroo">Buckaroo Banzai</option>
    <option value="Tommy">Perfect Tommy</option>
    <option value="Penny">Penny Priddy</option>
    <option value="Jersey">New Jersey</option>
    <option value="John">John Parker</option>
</select>
```

What can go in a form?

option

The <option> element works with the <select> element to create a menu. Use an <option> element for each menu item.

After clicking on the menu, the menu items drop down.



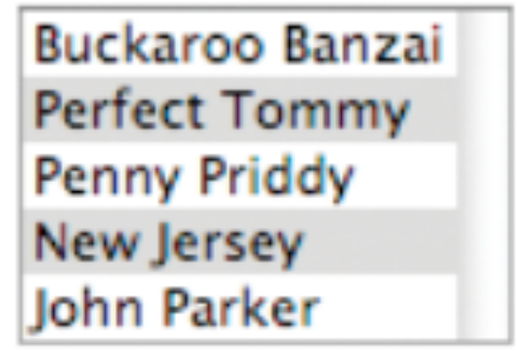

```
<select name="characters">
    <option value="Buckaroo">Buckaroo Banzai</option>
    <option value="Tommy">Perfect Tommy</option>
    <option value="Penny">Penny Priddy</option>
    <option value="Jersey">New Jersey</option>
    <option value="John">John Parker</option>
</select>
```

The content of the <option> element is used for the menu items' description. Each menu option also includes a value representing the menu item.

Multiple selection

This isn't an element, but rather a new way to use an element you already know. If you add the Boolean attribute `multiple` to your `<select>` element, you turn your single-choice menu into a multiple-choice menu. Instead of a pop-down menu, you'll get a multiple-choice menu that shows all the options on the screen (with a scroll bar if there are a lot of them); you can choose more than one by holding down the Ctrl (Windows) or Command (Mac) key as you select.

With multiple selection, you can choose more than one option at a time.




```
<select name="characters" multiple>
    <option value="Buckaroo">Buckaroo Banzai</option>
    <option value="Tommy">Perfect Tommy</option>
    <option value="Penny Priddy">Penny</option>
    <option value="New Jersey">Jersey</option>
    <option value="John Parker">John</option>
</select>
```

Just add the attribute multiple to turn a single selection menu into a multiple selection menu.

What can go in a form?

number input

The number <input> element restricts input to numbers. You can even specify a min and max number that is allowed with optional attributes.

Some browsers show arrows next to the input area you can use to increase or decrease the number.

The "number" type means you're expecting a number only, not text.

Use the max and min attributes to restrict the numbers allowed.

```
<input type="number" min="0" max="20">
```

What can go in a form?

range input

The range <input> element is similar to number except that it displays a slider instead of an input box.

Both number and range have an optional step attribute you can use to specify the number of intervals for the values.

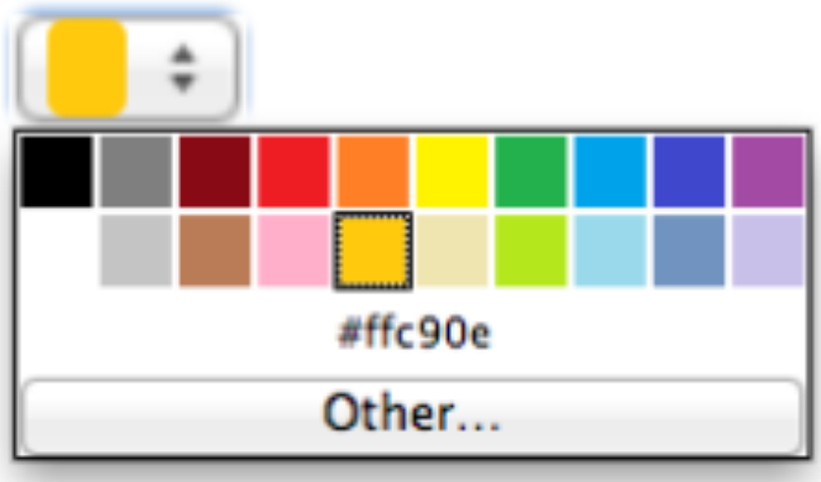
```
<input type="range" min="0" max="20" step="5">
```

What can go in a form?

color input

Use the color <input> to specify a color. When you click on the control, a color picker pops up that allows you to select a color rather than having to type in the color name or value.

If the color input is not supported by the browser, you'll just get a regular text input instead.

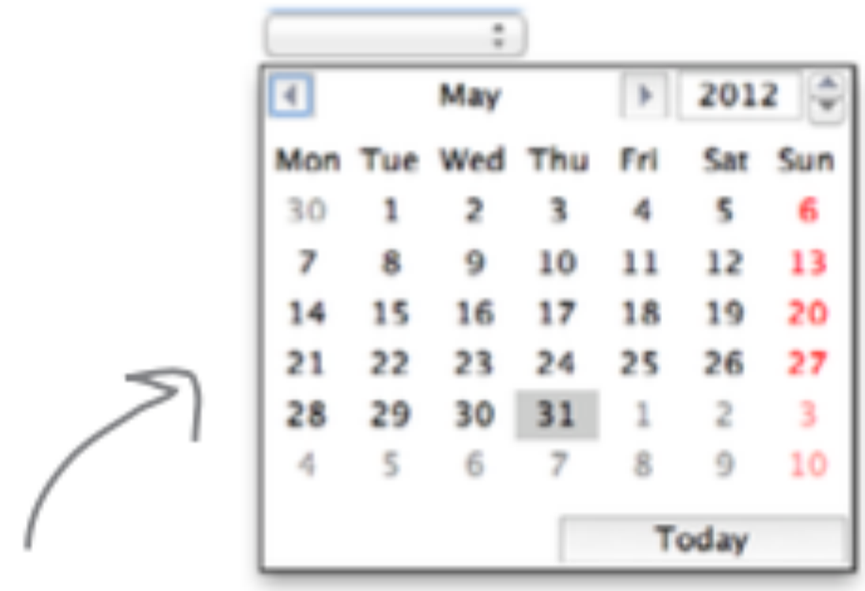



```
<input type="color">
```

What can go in a form?

date input

Use the date <input> element to specify a date, with a date picker control. The control creates a valid date format string to send to the server script.




```
<input type="date">
```

Like with color, if the date input isn't supported by the browser yet, you'll get a regular text input instead.

What can go in a form?

email input

The email <input> element is just a text input, but on some mobile browsers, you'll get a custom keyboard for email when you start typing.

```
<input type="email">
```

Email: Buckaroo Banzai

tel input

The tel <input> element is also just a text input, but like email, causes a custom keyboard to pop up on mobile devices.

```
<input type="tel">
```

Phone: 555-1212

url input

Like email and tel, the url <input> type is just a text input, but causes a custom keyboard to pop up on mobile devices.

```
<input type="url">
```

URL: http://banzai.com

These three <input> types are all variations of the text <input> type. On desktop browsers you won't notice a difference. But on mobile browsers, you might get a custom keyboard that makes it easier to get to the characters you need, like / and @ and numbers.

Even with these specialized types, it's up to you to make sure you know what values the server script is expecting and use the right <input> type.

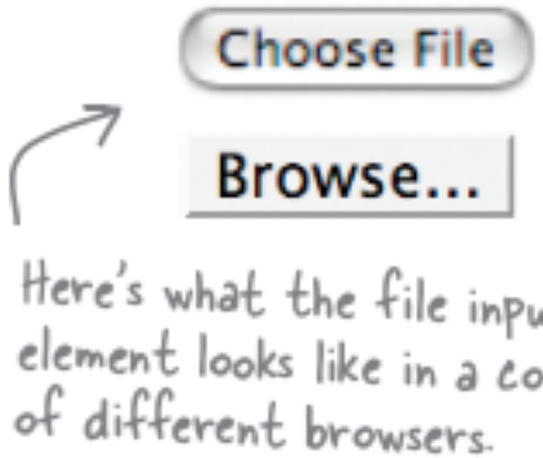
Watch it!

Not all browsers fully support these input types yet.

The input types on these two pages are new in HTML5, and while you can use them in all web pages now, some may not display as you see them here.

File input

Here's a whole new input element we haven't talked about. If you need to send an entire file to a server script, you'll once again use the `<input>` element, but this time set its type to "file". When you do that, the `<input>` element creates a control that allows you to select a file and—when the form is submitted—the contents of the file are sent with the rest of your form data to the server. Remember, your server script will need to be expecting a file upload, and also note that you must use the POST method to use this element.



Here's what the file input element looks like in a couple of different browsers.

```
<input type="file" name="doc">
```

To create a file input element, just set the type of the <input> element to "file".

Passwords

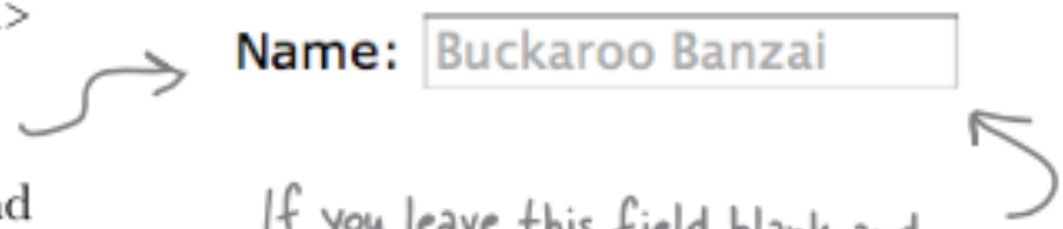
The password `<input>` element works just like the text `<input>` element, except that the text you type is masked. This is useful for forms that require you to type in a password, a secret code, or other sensitive information that you may not want other people to see as you type. Keep in mind, however, that the form data is *not* sent from the browser to the server script in a secure way, unless you make it secure. For more on security, contact your hosting company.

```
<input type="password" name="secret">
```

The password <input> element works exactly like the text <input> element, except the text you type is masked.

Placeholder

You can use the `placeholder` attribute with most of the `<input>` types in a form to give the person who's filling out the form a hint about the kind of content you expect him to enter into the control. For instance, if you have a text field that expects a first and last name, you can provide a sample first and last name using the `placeholder` attribute. The value in the attribute is shown in the control, but is fainter than normal content that you add to a control, and as soon as you click into the text field, the placeholder text will disappear so it doesn't get in the way of what you're typing.



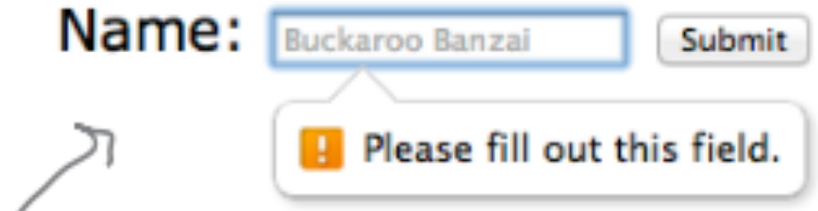

```
<input type="text" placeholder="Buckaroo Banzai">
```

The placeholder attribute allows you to provide a hint about the kind of content you're expecting in this part of the form.

Required

This is an attribute you can use with any form control; it indicates that a field is required, so you shouldn't submit the form without specifying a value for the controls that have this attribute set. In browsers that support this attribute, if you try to submit the form without specifying a value for a `required` field, you'll get an error message and the form will not be submitted to the server.

Notice that this attribute is another *Boolean* attribute, like we saw in the `<video>` element. That just means that the value of the attribute is simply "there" or "not there." That is, if the attribute's there, then it's set, and if the attribute's not there, then it's not set. So in this example, `required` is there, so that means the attribute is set and the field is required to submit the form.



This is a screenshot from Chrome. As of this writing, not all browsers support required, but you can put it there anyway. You'll be able to submit the form, but then of course, the server script will complain that you haven't filled in the field.

required is a Boolean attribute, so if it's in the form control, that means the field must have a value for the form to submit correctly.

```
<input type="text" placeholder="Buckaroo Banzai" required>
```

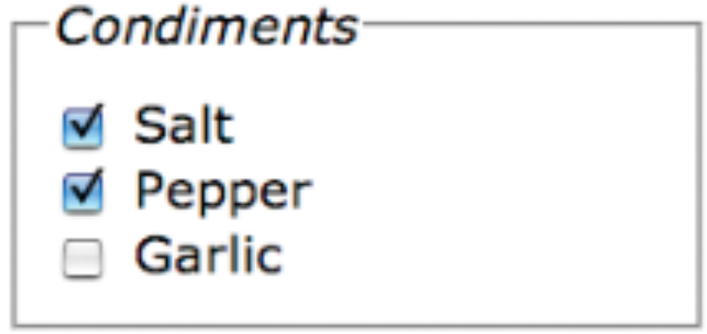
Fieldsets and legends

When your forms start getting large, it can be helpful to visually group elements together. While you might use `<div>`s and CSS to do this, HTML also provides a `<fieldset>` element that can be used to group together common elements. `<fieldset>` makes use of a second element, called `<legend>`. Here's how they work together:

The <fieldset> element surrounds a set of input elements.

The <legend> provides a label for the group.

```
<fieldset>
   <legend>Condiments</legend>
      <input type="checkbox" name="spice" value="salt">
         Salt  <br>
      <input type="checkbox" name="spice" value="pepper">
         Pepper <br>
      <input type="checkbox" name="spice" value="garlic">
         Garlic
</fieldset>
```



Here's how the fieldset and legend look in one browser. You'll find that browsers display them differently.

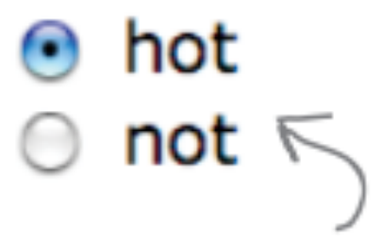
Label

To use a <label> element, first add an id attribute to your form element.

```
<input type="radio" name="hotornot" value="hot" id="hot">
<label for="hot">hot</label>

<input type="radio" name="hotornot" value="not" id="not">
<label for="not">not</label>
```

Then add a <label> and set its "for" attribute to the corresponding id.

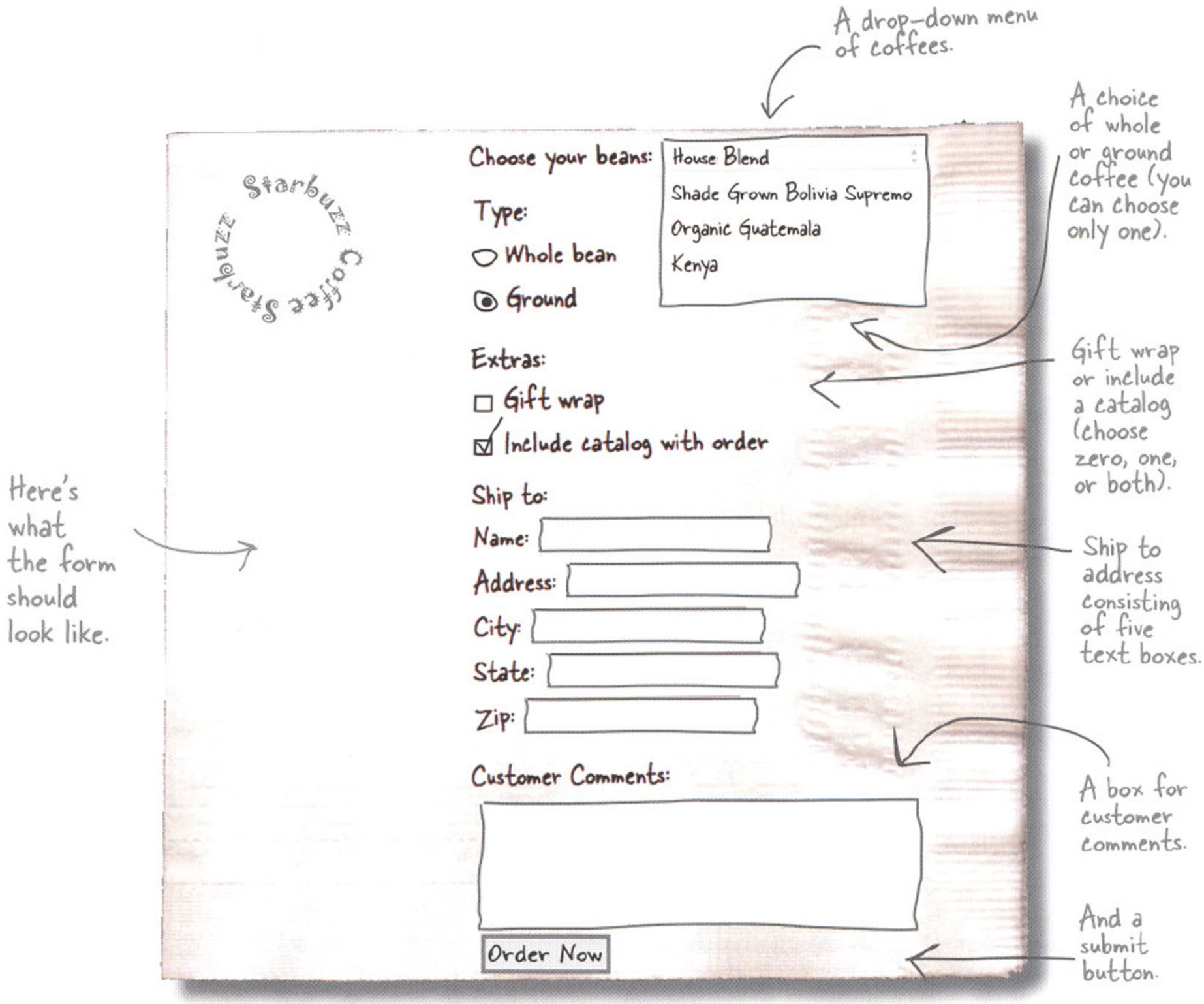


Now the text next to these radio buttons is a label.

Autofocus

```
<form>
  <input name="q" autofocus>
  <input type="submit" value="Search">
</form>
```

Esercizio


A drop-down menu of coffees.
A choice of whole or ground coffee (you can choose only one).
Gift wrap or include a catalog (choose zero, one, or both).
Ship to address consisting of five text boxes.
A box for customer comments.
And a submit button.
Here's what the form should look like.
Starbuzz Coffee
Choose your beans:
House Blend
Shade Grown Bolivia Supremo
Organic Guatemala
Kenya
Type:
Whole bean
Ground
Extras:
Gift wrap
Include catalog with order
Ship to:
Name:
Address:
City:
State:
Zip:
Customer Comments:
Order Now

form.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en" >
   <head >
      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
      <title>The Starbuzz Bean Machine</title>
   </head>
   <body>

      <h1>The Starbuzz Bean Machine</h1>
      <h2>Fill out the form below and click submit to order</h2>

   </body>
</html>
```

The form is going to go here.

All we've got so far is a heading identifying the page, along with instructions.

For now, we're going to build these forms without all the style we've been using on the Starbuzz site. That way we can concentrate on the form XHTML. We'll add the style in later.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en" >
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
      <title>The Starbuzz Bean Machine</title>
   </head>
   <body>

      <h1>The Starbuzz Bean Machine</h1>
      <h2>Fill out the form below and click submit to order</h2>

      <form action="http://www.starbuzzcoffee.com/processorder.php" method="POST">

      </form>
   </body>
</html>
```

Here's the form element

The action attribute contains the URL of the Web application.

And remember we're using the "POST" method to deliver the form data to the server. More on this later.

Go ahead and add the form closing tag too.

We use the <input> element for a few different controls. The type attribute determines what kind of control it is.

```
<input type="text" name="name" />
<input type="text" name="address" />
<input type="text" name="city" />
<input type="text" name="state" />
<input type="text" name="zip" />
```

We've got one text input for each input area in the form: Name, Address, City, State, and Zip.

Here the type is "text" because this is going to be a text input control.

The name attribute acts as an identifier for the data the user types in. Notice how each one is set to a different value. Let's see how this works...

Each input control in your form has a name attribute

When you type the elements for a form into your XHTML file, you give them unique names. You saw this with the text inputs:

```
<input type="text" name="name" />
<input type="text" name="address" />
<input type="text" name="city" />
<input type="text" name="state" />
<input type="text" name="zip" />
```

Notice here we've got an element whose name is "name" (which is perfectly fine).

Each <input> element gets its own name.

When you submit a form, the browser packages up all the data using the unique names:

Say you type your name, address, city, state, and zip into the form and click submit. The browser takes each of these pieces of data and labels them with your unique name attribute values. The browser then sends the names and values to the server. Like this:

What you enter into the form.

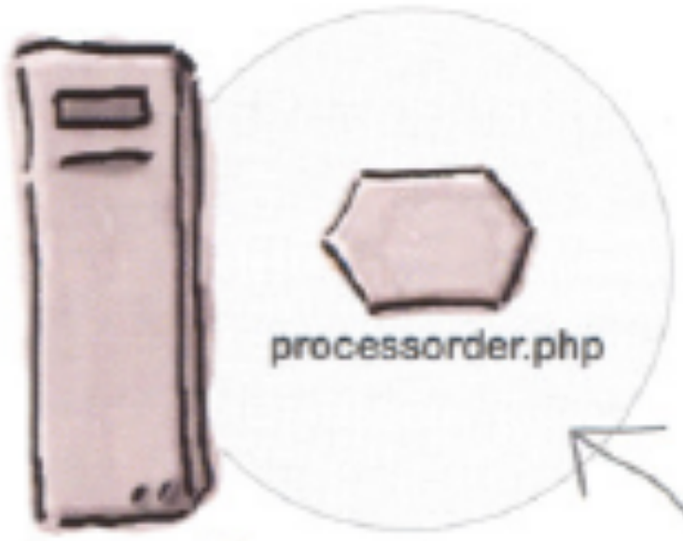
Name: Buckaroo Banzai
Address: Banzai Institute
City: Los Angeles
State: CA
Zip: 90050

The unique names for each form element.

```
name = Buckaroo Banzai
address = Banzai Institute
city = Los Angeles
state = CA
zip = 90050
```

Each unique name gets a value from the data you type into the form.

What the browser packages up for the server.



The Web application needs the form data to be labelled so it can tell what is what.

We're going to start by putting everything inside a <p> element.

Nest elements directly inside a form.

Here's just the form snippet from "form.html". Hey, we've got to save a few trees here!

```
<form action="http://starbuzzcoffee.com/processorder.php" method="POST">
  <p>Ship to: <br>
      Name: <input type="text" name="name"> <br>
      Address: <input type="text" name="address"> <br>
      City: <input type="text" name="city"> <br>
      State: <input type="text" name="state"> <br>
      Zip: <input type="text" name="zip"> <br>
      Phone: <input type="tel" name="phone"> <br>
  </p>
  <p>
      <input type="submit" value="Order Now">
  </p>
</form>
```

Here are all the <input> elements: one for each input in the "Ship to" section of the form.

We've added a label for each input so the user knows what goes in the text input.

And you should also know that <input> is an inline element, so if you want some linebreaks between the <input> elements, you have to add
s. That's also why you need to nest them all inside a paragraph.

Finally, don't forget that users need a submit button to submit the form. So add a submit button by inserting an <input> at the bottom with a type of "submit". Also add a value of "Order Now", which will change the text of the button from "Submit" to "Order Now".

Adding the <select> element

```
<form action="http://starbuzzcoffee.com/processorder.php" method="post">
  <p>
    Choose your beans:
    <select name="beans">
      <option value="House Blend">House Blend</option>
      <option value="Bolivia">Shade Grown Bolivia Supremo</option>
      <option value="Guatemala">Organic Guatemala</option>
      <option value="Kenya">Kenya</option>
    </select>
  </p>
```

Here's our brand-new <select> element. It gets a unique name too.

Inside, we put each <option> element, one per choice of coffee.

```
  <p>
    Ship to: <br>
    Name: <input type="text" name="name" value=""><br>
    Address: <input type="text" name="address" value=""><br>
    City: <input type="text" name="city" value=""><br>
    State: <input type="text" name="state" value=""><br>
    Zip: <input type="text" name="zip" value=""><br>
    Phone: <input type="tel" name="phone" value=""><br>
  </p>
  <p>
    <input type="submit" value="Order Now">
  </p>
</form>
```

adding the <radio> element

```
<p>Type: <br>

   <input type="radio" name="beantype" value="whole"> Whole bean <br>
   <input type="radio" name="beantype" value="ground"> Ground

</p>
```

We're using the <input> element for this, with its type set to "radio".

Here's the unique name. All radio buttons in the same group share the same name.

And here's the value that will be sent to the server script. Only one of these will be sent (the one that is selected when the form is submitted).

Notice that we often label radio buttons on the righthand side of the element.

```
<form action="http://starbuzzcoffee.com/processorder.php" method="post">
    <p>
      Choose your beans:
      <select name="beans">
        <option value="House Blend">House Blend</option>
        <option value="Bolivia">Shade Grown Bolivia Supremo</option>
        <option value="Guatemala">Organic Guatemala</option>
        <option value="Kenya">Kenya</option>
      </select>
    </p>
    <p>
      Type:<br>
        <input type="radio" name="beantype" value="whole">Whole bean<br>
        <input type="radio" name="beantype" value="ground" checked>Ground
    </p>

    <p>
      Number of bags: <input type="number" name="bags" min="1" max="10">
    </p>
    <p>
      Must arrive by date: <input type="date" name="date">
    </p>

    <p>
      Ship to: <br>
      Name: <input type="text" name="name" value=""><br>
      Address: <input type="text" name="address" value=""><br>
      City: <input type="text" name="city" value=""><br>
      State: <input type="text" name="state" value=""><br>
      Zip: <input type="text" name="zip" value=""><br>
      Phone: <input type="tel" name="phone" value=""><br>
    </p>
    <p>
      <input type="submit" value="Order Now">
    </p>
</form>
```

We've added the new code here. Remember that browsers may display these differently, depending on which browser you're using. Try more than one browser!

```
<p>
  Type:<br>
    <input type="radio" name="beantype" value="whole">Whole bean<br>
    <input type="radio" name="beantype" value="ground" checked>Ground
</p>
<p>Number of bags: <input type="number" name="bags" min="1" max="10"></p>
<p>Must arrive by date: <input type="date" name="date"></p>
```

Here we've added a checkbox for each option. Notice that these share the same name, "extras[]"...

...but have different values.

```
<p>
  Extras:<br>
  <input type="checkbox" name="extras[]" value="giftwrap">Gift wrap<br>
  <input type="checkbox" name="extras[]" value="catalog" checked>Include catalog
     with order
</p>
```

We're using the checked attribute to specify that the catalog option should be checked by default. You can add a checked attribute to more than one checkbox.

As with the radio buttons, we've put these labels to the right of the checkboxes.

```
<p>
  Ship to: <br>
  Name: <input type="text" name="name" value=""><br>
  Address: <input type="text" name="address" value=""><br>
  City: <input type="text" name="city" value=""><br>
  State: <input type="text" name="state" value=""><br>
  Zip: <input type="text" name="zip" value=""><br>
  Phone: <input type="tel" name="phone" value=""><br>
</p>
```

```
<p>Customer Comments:<br>
    <textarea name="comments"></textarea>
</p>
```

Here's the text area.

```
<p>
  <input type="submit" value="Order Now">
</p>
```

The Starbuzz Bean Machin

file:///Users/carlo/Desktop/lezioni%20SciPol/all/esercizi/chapter14/solution/starbuzz/form.html

The Starbuzz Bean Machine

Fill out the form below and click "order now" to order

Choose your beans:

House Blend

Type:

- ○ Whole bean
- ◉ Ground

Number of bags:

Must arrive by date:

gg/mm/aaaa

Extras:

- ☐ Gift wrap
- ☑ Include catalog with order

Ship to

Name:

Buckaroo Banzai

Address:

Banzai Institute

City:

risultato

Getting the form elements into HTML structure for table display layout

```
<div class="tableRow">
    ...
</div>
```

Getting the form elements into HTML structure for table display layout

```
<form action="http://starbuzzcoffee.com/processorder.php" method="post">
    <div class="tableRow">
        <p>
          Choose your beans:
        </p>
        <p>
            <select name="beans">
              <option value="House Blend">House Blend</option>
              <option value="Bolivia">Shade Grown Bolivia Supremo</option>
              <option value="Guatemala">Organic Guatemala</option>
              <option value="Kenya">Kenya</option>
            </select>
        </p>
    </div>
    <div class="tableRow">
        <p> Type: </p>
        <p>
            <input type="radio" name="beantype" value="whole"> Whole bean<br>
            <input type="radio" name="beantype" value="ground" checked> Ground
        </p>
    </div>
    <div class="tableRow">
        <p> Number of bags: </p>
        <p> <input type="number" name="bags" min="1" max="10"> </p>
    </div>
    <div class="tableRow label">
        <p> Must arrive by date: </p>
        <p> <input type="date" name="date"> </p>
    </div>
    <div class="tableRow">
        <p> Extras: </p>
        <p>
            <input type="checkbox" name="extras[]" value="giftwrap"> Gift wrap<br>
            <input type="checkbox" name="extras[]" value="catalog" checked>
            Include catalog with order
        </p>
    </div>
```

We're using a with the class "tableRow" for each row in the table.

And the content for each cell is nested inside a <p> element.

For the bean selection menu, the "beantype" radio buttons, and the "extras" checkboxes, we put all the form elements for each menu in one data cell.

Code continues on the next page.

continua…

```
<div class="tableRow">
    <p class="heading"> Ship to </p>
    <p></p>
</div>
<div class="tableRow">
    <p> Name: </p>
    <p> <input type="text" name="name" value=""> </p>
</div>
<div class="tableRow">
    <p> Address: </p>
    <p> <input type="text" name="address" value=""> </p>
</div>
<div class="tableRow">
    <p> City: </p>
    <p> <input type="text" name="city" value=""> </p>
</div>
<div class="tableRow">
    <p> State: </p>
    <p> <input type="text" name="state" value=""> </p>
</div>
<div class="tableRow">
    <p> Zip: </p>
    <p> <input type="text" name="zip" value=""> </p>
</div>
<div class="tableRow">
    <p> Phone: </p>
    <p> <input type="tel" name="phone" value=""> </p>
</div>
<div class="tableRow">
    <p> Customer Comments: </p>
    <p>
        <textarea name="comments" rows="10" cols="48"></textarea>
    </p>
</div>
<div class="tableRow">
    <p></p>
    <p> <input type="submit" value="Order Now"> </p>
</div>
</form>
```

Notice that we've also got an empty cell in the right column, so we can just put an empty <p> element here.

All the rows are straightforward: a "tableRow" for the row, and each cell in a <p>.

And for the last row, we've got an empty cell in the left column, so again, we can use an empty <p> element for that.

styling the form with css

```
body {
    background: #efe5d0 url(images/background.gif) top left;
    margin: 20px;
}

form {
    display: table;
    padding: 10px;
    border: thin dotted #7e7e7e;
    background-color: #e1ceb8;
}

form textarea {
    width: 500px;
    height: 200px;
}

div.tableRow {
    display: table-row;
}

div.tableRow p {
    display: table-cell;
    vertical-align: top;
    padding: 3px;
}
div.tableRow p:first-child {
    text-align: right;
}
p.heading {
    font-weight: bold;
}

```

The Starbuzz Bean Machin ×

file:///Users/carlo/Desktop/lezioni%20SciPol/all/esercizi/chapter14/solution/starbuzz/form.html

The Starbuzz Bean Machine

Fill out the form below and click "order now" to order

Choose your beans: House Blend

Type: Whole bean / Ground

Number of bags:

Must arrive by date: gg/mm/aaaa

Extras: Gift wrap / Include catalog with order

Ship to

Name: Buckaroo Banzai

Address: Banzai Institute

City: Los Angeles

State: CA

Zip: 90050

Phone: 310-555-1212

Customer Comments:

Order Now

risultato