Forms in HTML

Form Elements

The principle behind web forms is that we can use a web page to collect data from a user and either send it on to a web server, or manipulate that data with JavaScript.

To send on a piece of information we need to assign it a name and/or id so that it can be referred to by scripting languages.

We can add the following types of form elements to a page:

- Text Fields
- Menus
- Check boxes
- Radio Buttons
- File Dialogues
- Buttons

While HTML5 also adds the following:

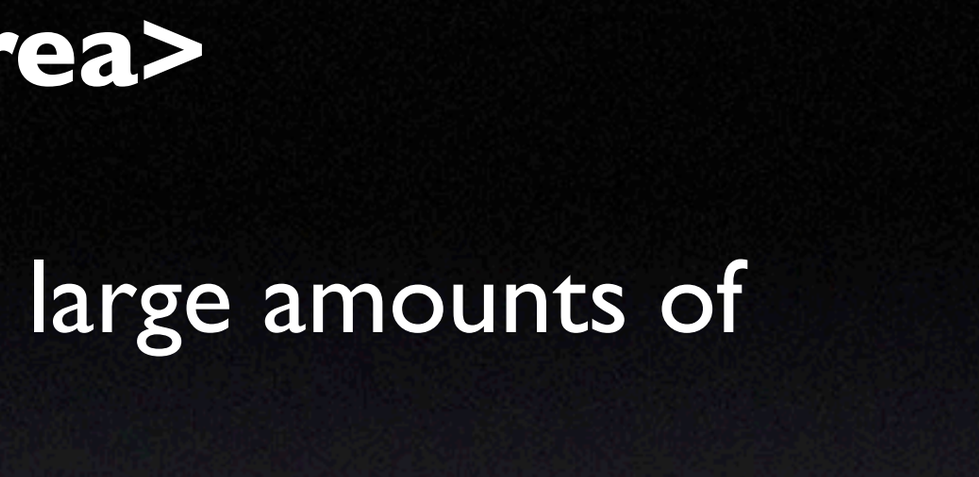
- Color Pickers
- Sliders
- Search Boxes
- Calendars/Date Pickers
- Text fields for email/url/numbers, etc.
- Progress bars

<textarea>

<textarea> ... </textarea>

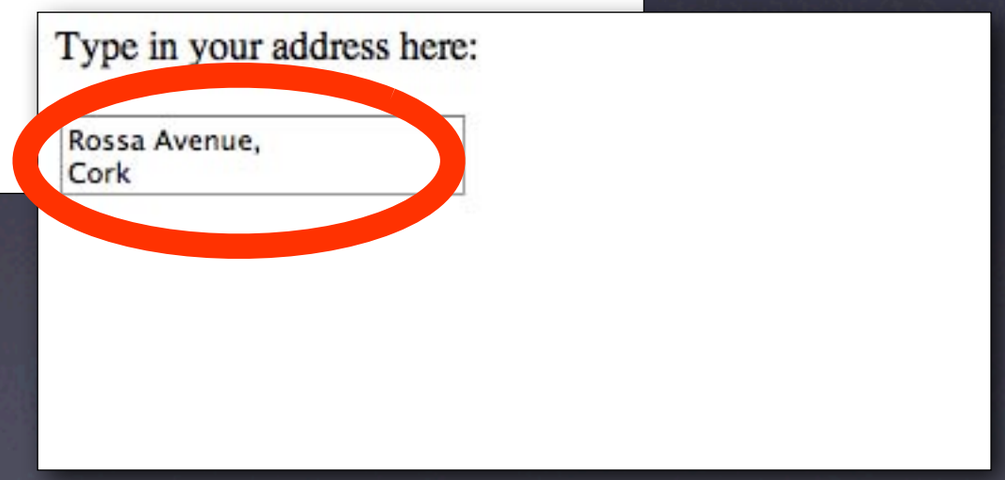
Text areas are used to input large amounts of text.

```
<p>Type in your address here: </p>

<div>
        <textarea>  </textarea>
</div>
```



You can enter multiple lines of text.

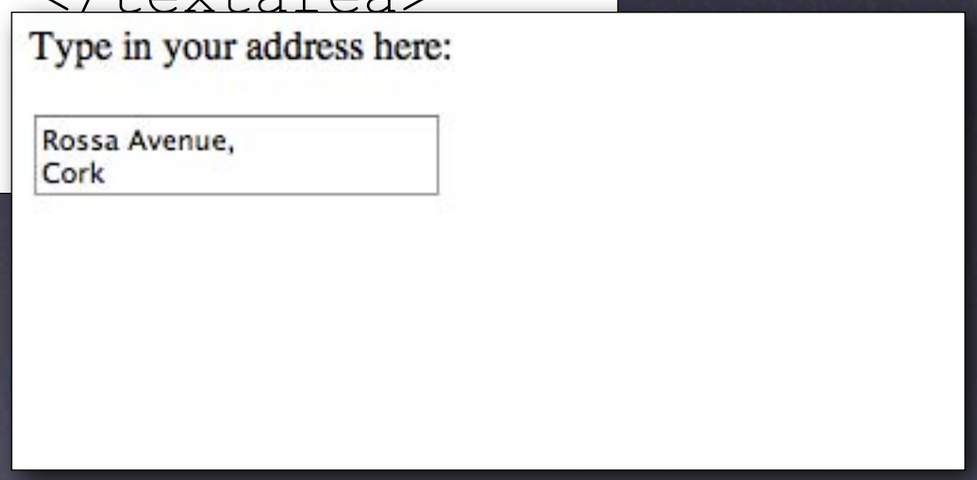
```
<p>Type in your address here: </p>

<div>
        <textarea>  </textarea>
</div>
```



If you are using the form to send the data to a script you should give the text area a name.

We give it the name *address* below.

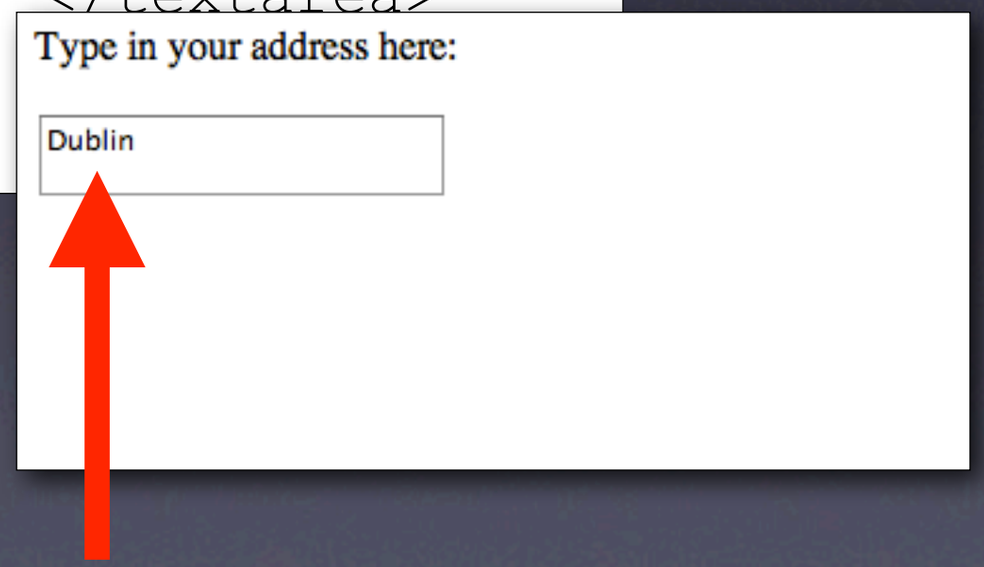
```
<p>Type in your address here: </p>

<div>
        <textarea name = "address">  </textarea>
</div>
```



In the example below the following data would be sent to the server.

address=Dublin

```
<p>Type in your address here: </p>

<div>
        <textarea name = "address">  </textarea>
</div>
```



name

Every form element supports the **name** attribute.

If you don't include the **name** attribute the information collected by that form element cannot be sent to the server.

Note: No two elements in the same form should have the same name (with one exception as we will see).

id

You can also give an element an **id**. This can be used to to help JavaScript target this particular form element.

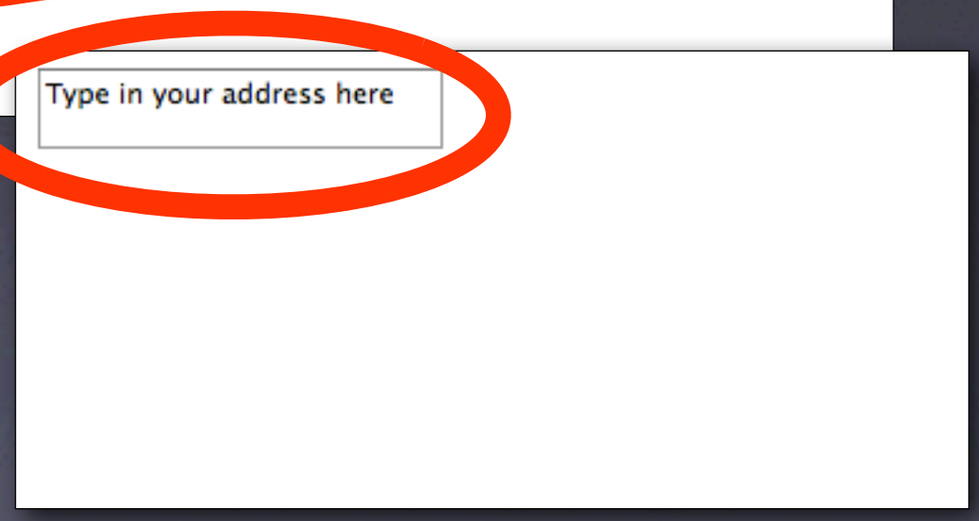
The name attribute can also be used for this purpose.

```
<div>
        <textarea id = "address1"></textarea>
</div>
```

Default Text

Text included between the opening and closing **<textarea>** tags will appear inside the text area when the user loads the page. The user can delete this text.

```
<div>
    <textarea>Type in your address here</textarea>
</div>
```

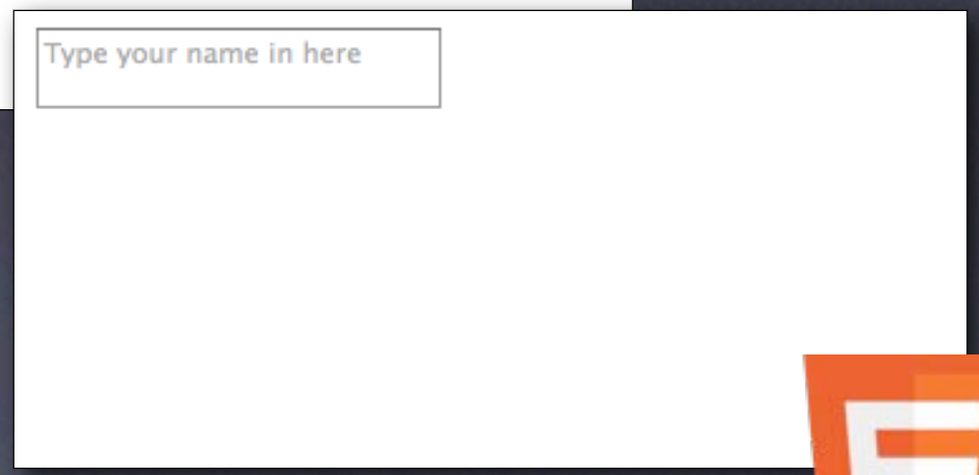


placeholder

You can also add text that will automatically disappear when the user starts typing in the text area.

```
<div>
    <textarea placeholder = "Type your name in
     here"></textarea>
</div>
```

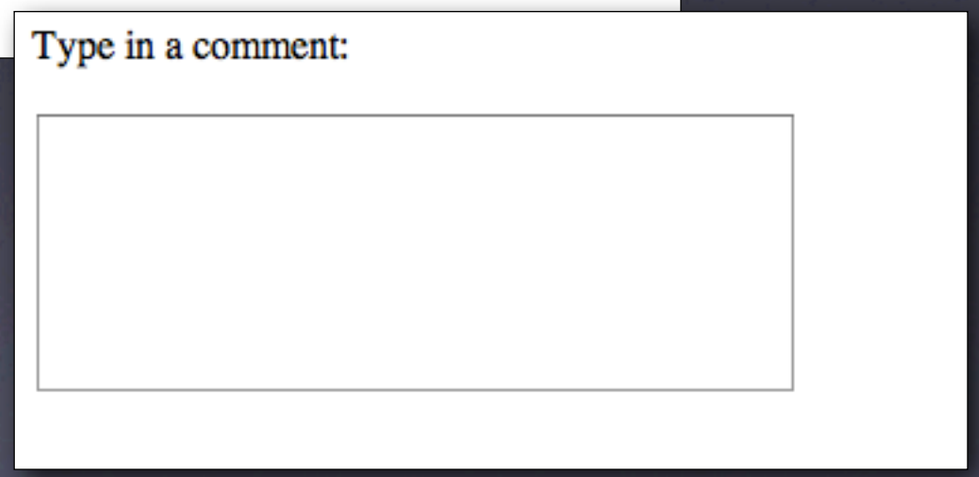
Make sure there is no space between the <textarea> tags.



cols & rows

These attributes specify the size of the text area.

```
<p>Type in a comment:</p>

<div>
        <textarea cols = "40" rows = "8"></textarea>
</div>
```



cols

This is the width of the text area.

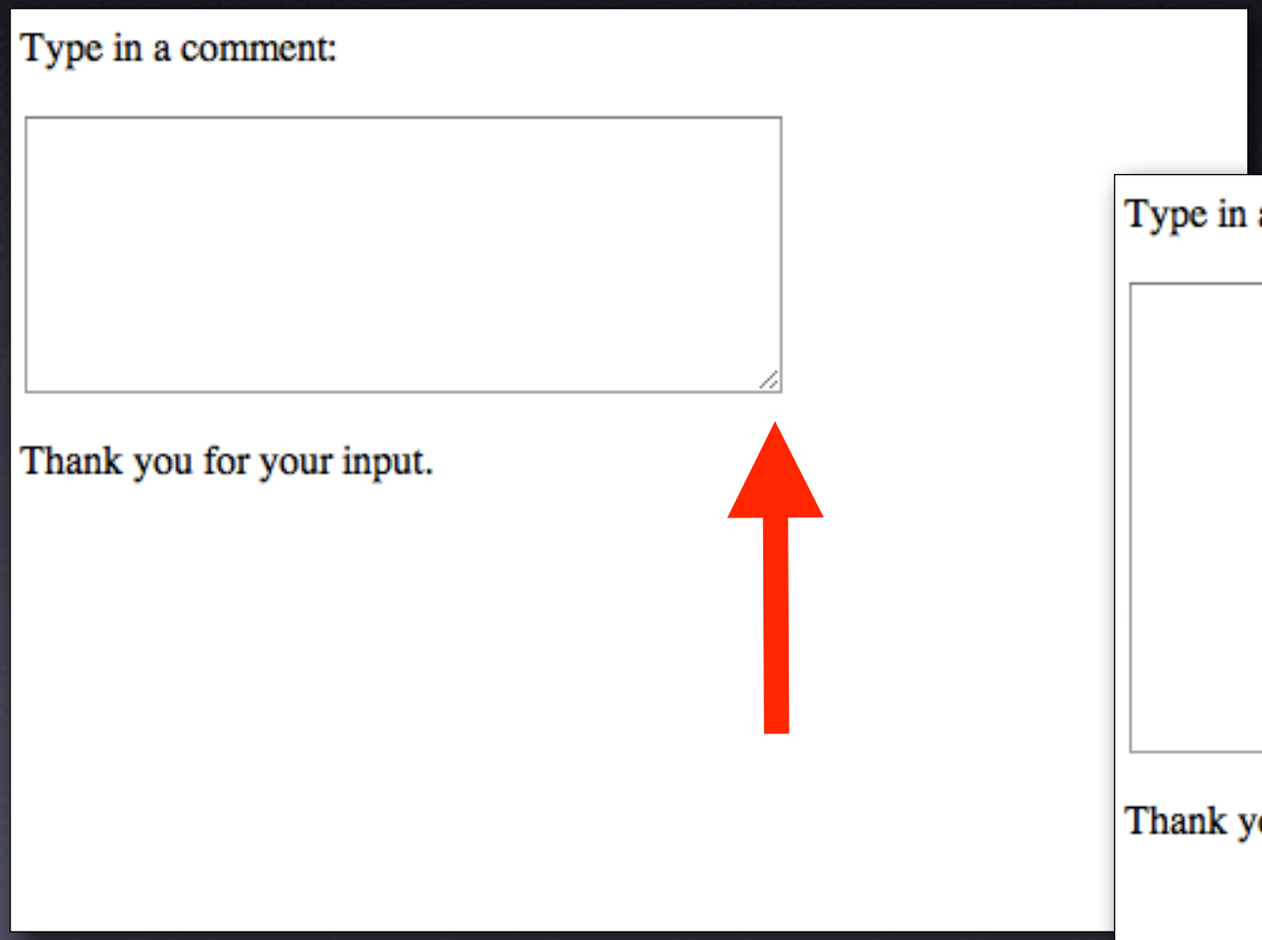
```
<p>Type in a comment:</p>

<div>
        <textarea cols = "40" rows = "8"></textarea>
</div>
```

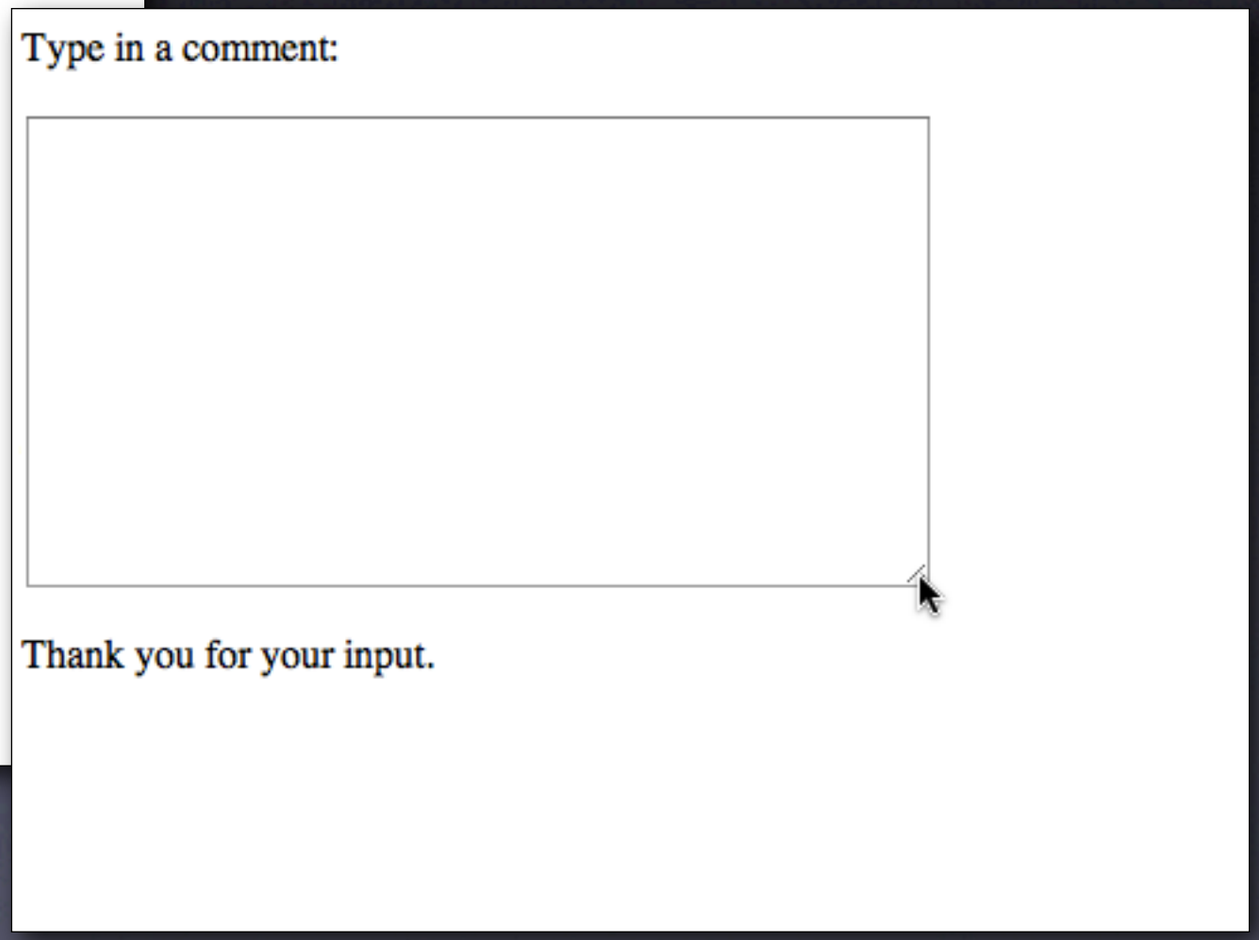
rows

This is the height of the text area.

```
<p>Type in a comment:</p>

<div>
        <textarea cols = "40" rows = "8"></textarea>
</div>
```

Modern browsers have added a grow box to the text area so the user can increase its size.





This can be turned off in CSS3.

```
textarea {
resize: none;
}
```

wrap

This attribute can specify how word wrapping within the text area is dealt with.

soft	[Default] Word wrapping displayed but no extra return characters sent to server
hard	Word wrapping is displayed and extra return characters needed are sent to server

maxlength

The maximum number of characters the text area will accept.

autofocus

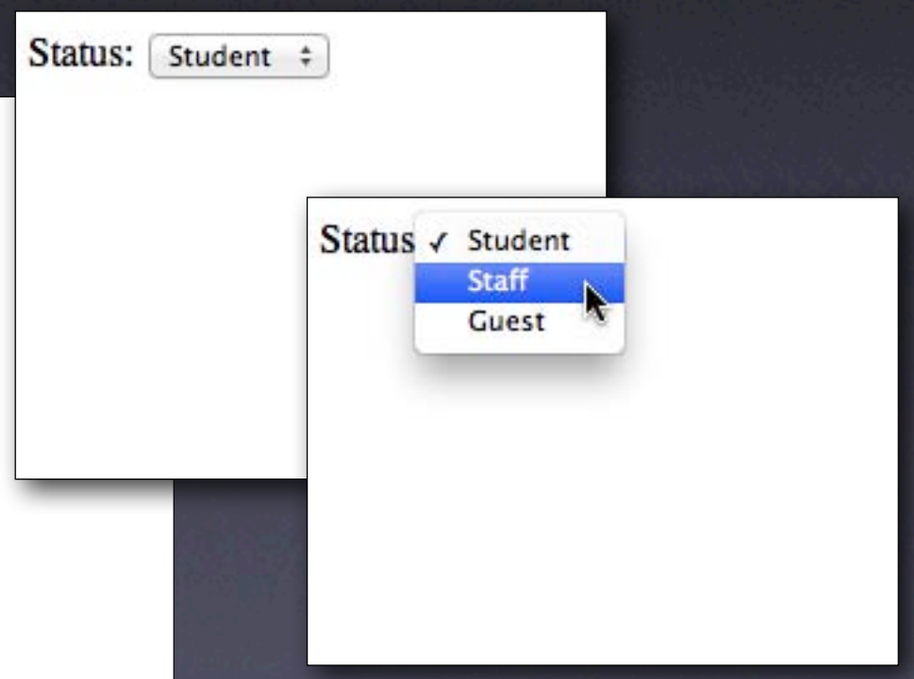
This boolean attribute, if present, will give the text area focus when the page loads. (This can only be used on most types of form element, but only one form element per form.)

<select>

<select>.... </select>
<option> ... </option>

<select> asks the user to choose one of several options. The choices are contained in the **<option>** tags.

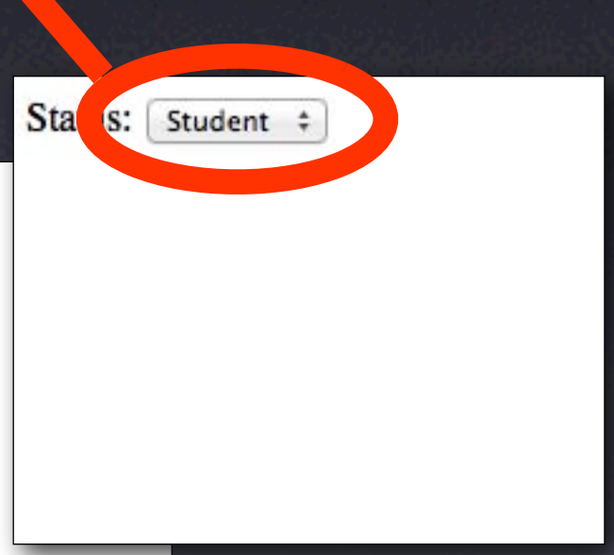
```
Status:

<select name = "type">
    <option>Student</option>
    <option>Staff</option>
    <option>Guest</option>
</select>
```



The option you chose will be sent to the server with the **name** you provided.

type=student

```
Status:

<select name = "type">
    <option>Student</option>
    <option>Staff</option>
    <option>Guest</option>
</select>
```

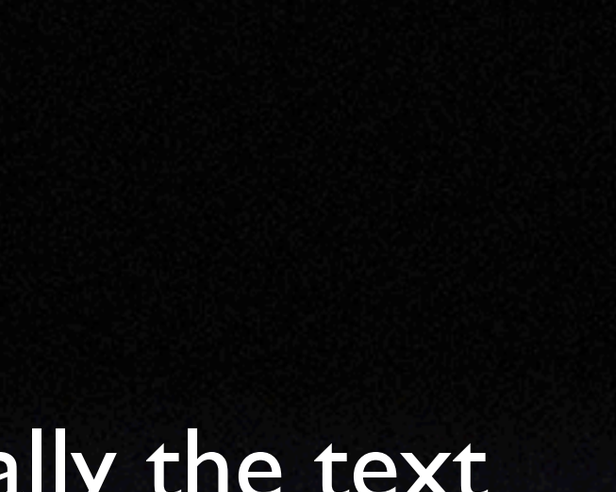


value

The value sent back is normally the text between the **<option>** container tags.

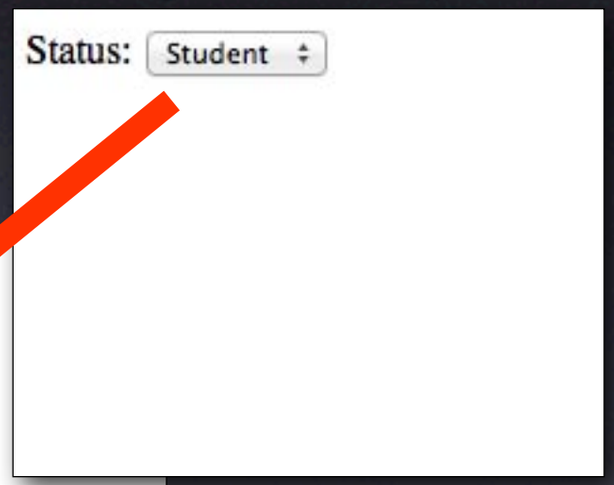
type=student

```
Status:

<select name = "type" >
    <option>Student</option>
    <option>Staff</option>
    <option>Guest</option>
</select>
```



Sometimes we don't want the onscreen option to be returned. In that case we can specify what gets returned with the **value** attribute.

type=category1




```
Status:

<select name = "type">
    <option value = "category1">Student</option>
    <option value = "category2">Staff</option>
    <option value = "category3">Guest</option>
</select>
```

It is also good practice to provide a dummy option that forces the user to select a valid option. Otherwise you wouldn't know if the user chose the default choice or just left the menu unchanged.

```
Status:
<select name = "status">
    <option>Please Select…</option>
    <option>Student</option>
    <option>Staff</option>
    <option>Guest</option>
</select>
```

selected

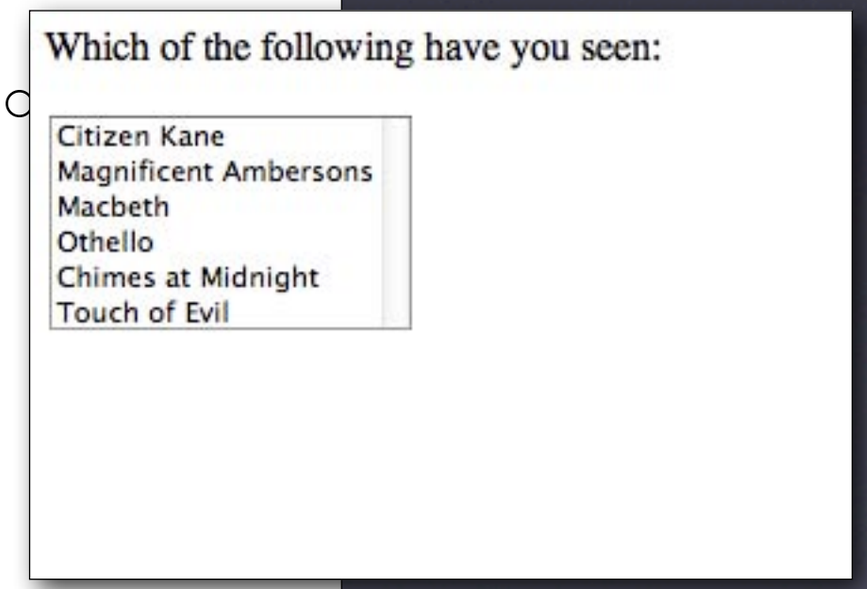
You can also add the **selected** boolean attribute to the option you wish to be shown (i.e. the default choice) when the page is loaded.

```
Status:
<select name = "status">
    <option selected >Please Select…</option>
    <option>Student</option>
    <option>Staff</option>
    <option>Guest</option>
</select>
```

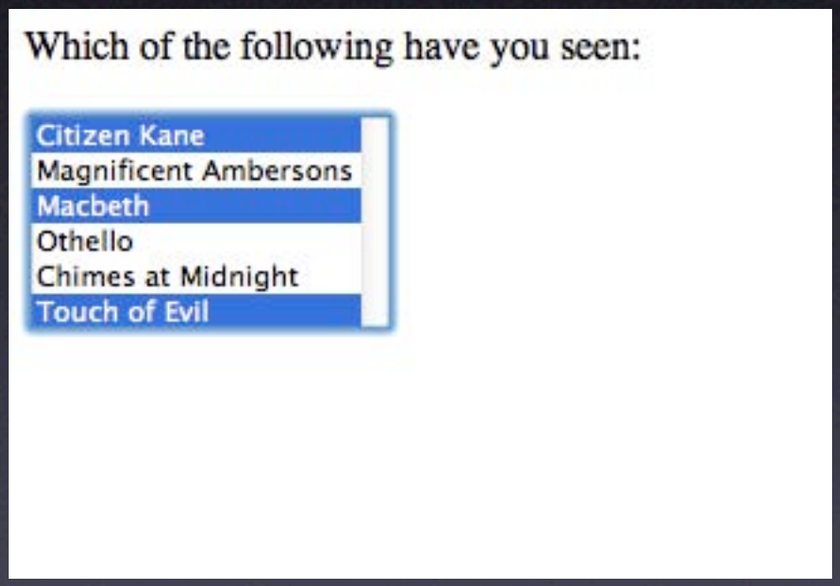
multiple

The **multiple** boolean attribute (in the <select> tag) will display several of the options at once.

```
<p>Which of the following have you seen:</p>
<div>
    <select multiple name = "films">
        <option>Citizen Kane</option>
        <option>Magnificent Ambersons</optio
        <option>Macbeth</option>
        <option>Othello</option>
        <option>Chimes at Midnight</option>
        <option>Touch of Evil</option>
    </select>
</div>
```



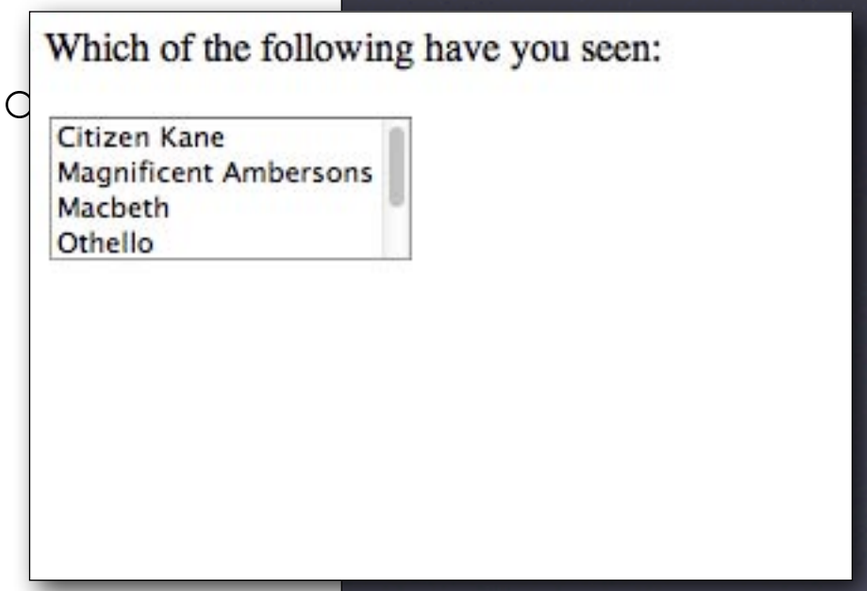
The user can now select multiple items from the menu.



size

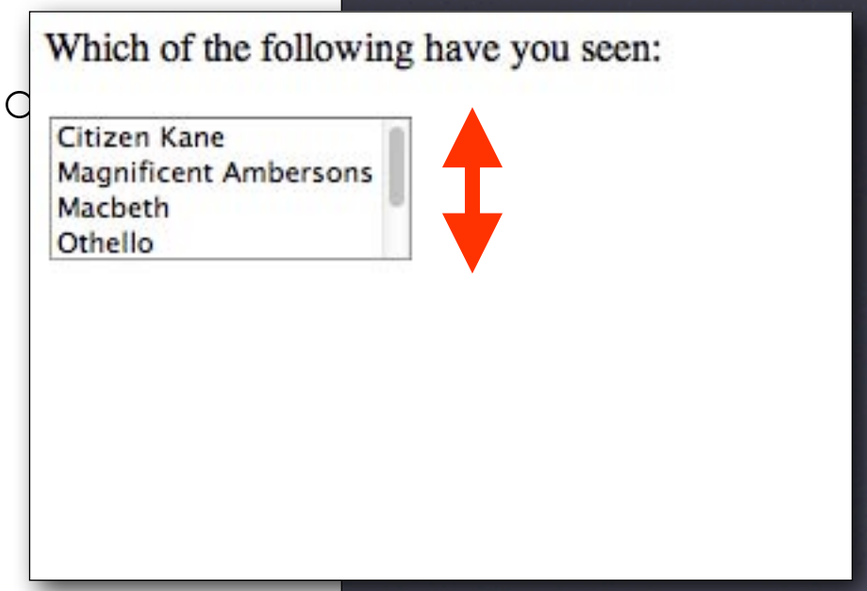
This attribute (for **<select>**) sets how many options can be shown at one time (if you are using **multiple**).

```
<p>Which of the following have you seen:</p>
<div>
    <select multiple size = "4" name = "films">
        <option>Citizen Kane</option>
        <option>Magnificent Ambersons</optio
        <option>Macbeth</option>
        <option>Othello</option>
        <option>Chimes at Midnight</option>
        <option>Touch of Evil</option>
    </select>
</div>
```



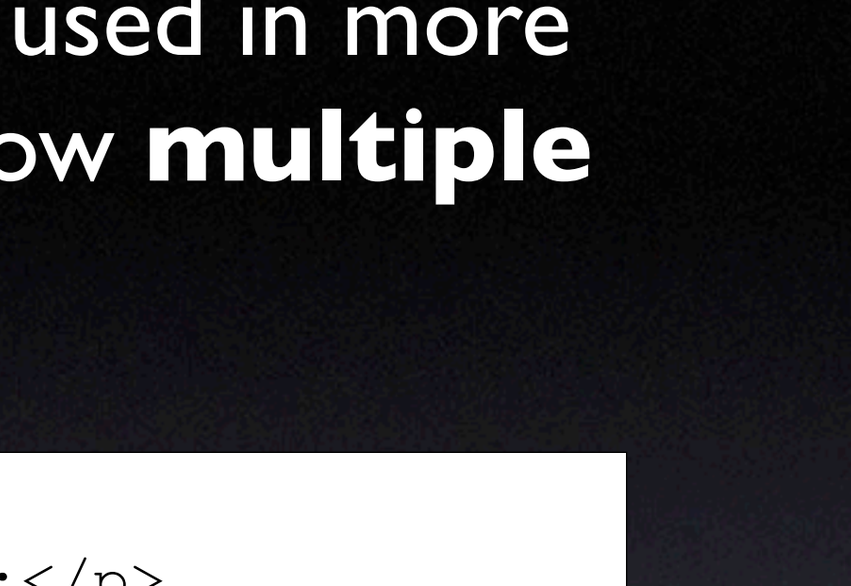
The user can scroll through the list to see the rest.

```
<p>Which of the following have you seen:</p>
<div>
    <select multiple size = "4" name = "films">
        <option>Citizen Kane</option>
        <option>Magnificent Ambersons</optio
        <option>Macbeth</option>
        <option>Othello</option>
        <option>Chimes at Midnight</option>
        <option>Touch of Evil</option>
    </select>
</div>
```



The **selected** attribute can be used in more than one <option> tag if you allow **multiple selection**.

```
<p>Which of the following have you seen:</p>
<div>
    <select multiple size = "6" name = "films">
        <option selected>Citizen Kane</option>
        <option>Magnificent Ambersons</option>
        <option>Macbeth</option>
        <option>Othello</option>
        <option selected>Chimes at Midnigh
        <option>Touch of Evil</option>
    </select>
</div>
```



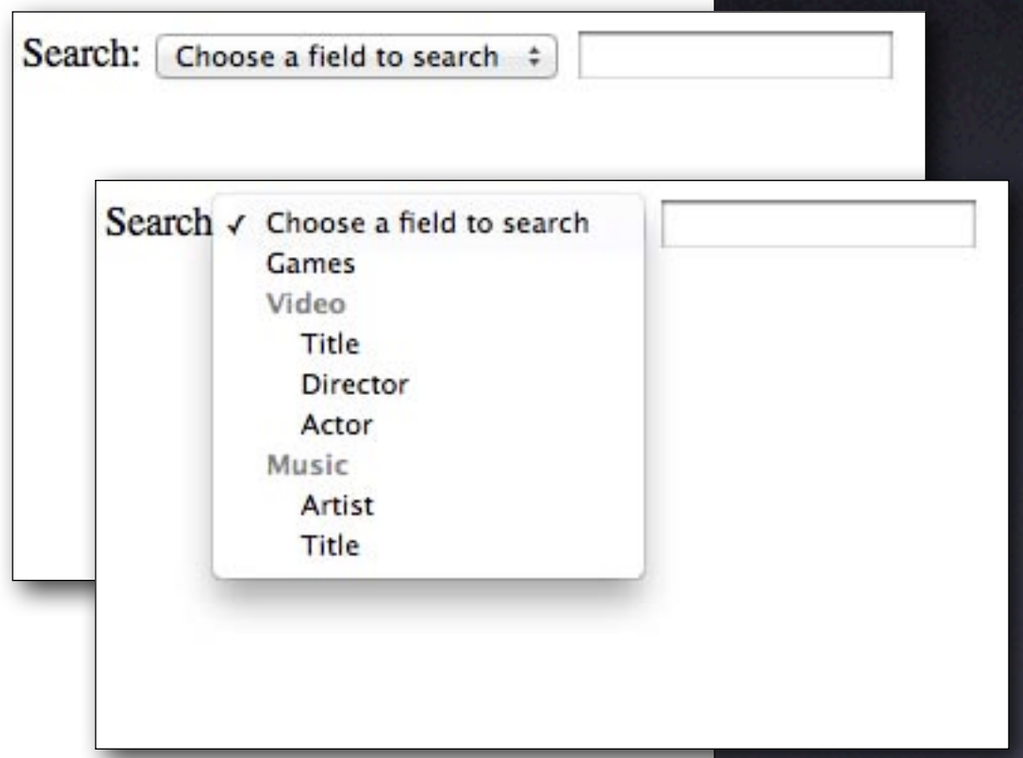
<optgroup>...</optgroup>

We can also organise the options into groups.

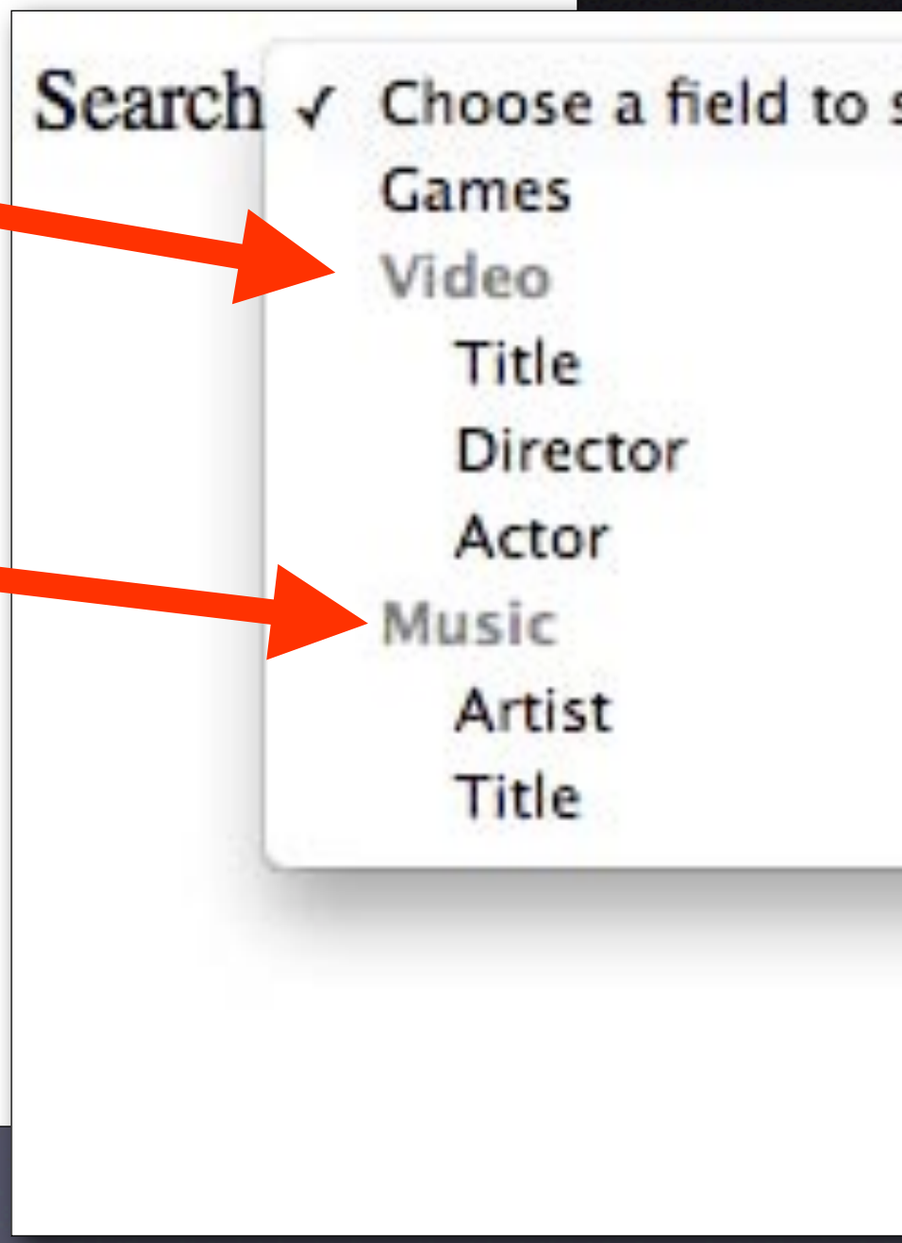
To do this we simply place **<optgroup>** tags around the **<option>** tags we want in a group.

To give the group a name we use the **<optgroup> label** attribute.

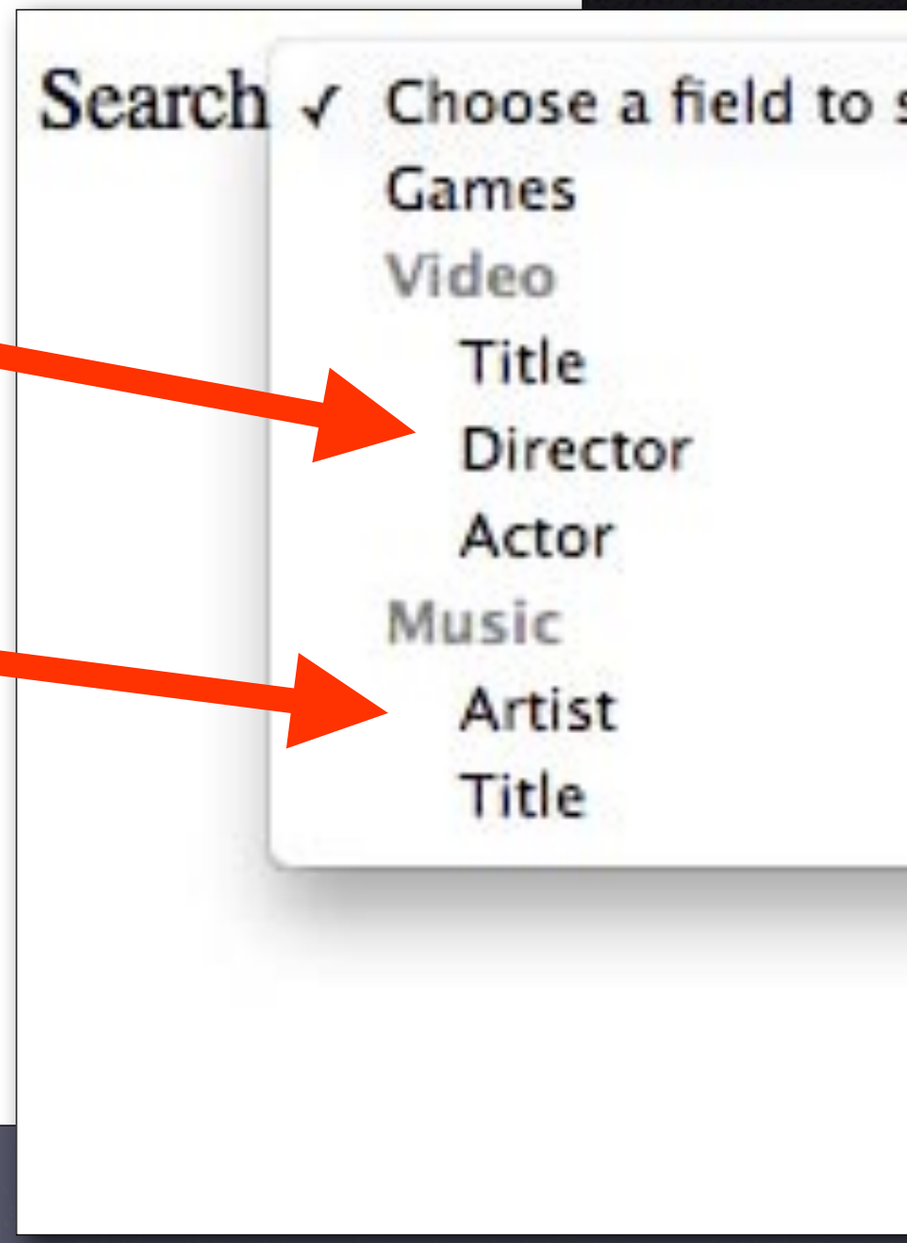
```
<p>Search:

<select name = "category">

    <option selected>Choose a field to search</option>

    <option>Games</option>

    <optgroup label = "Video">
        <option>Title</option>
        <option>Director</option>
        <option>Actor</option>
    </optgroup>

    <optgroup label = "Music">
        <option>Artist</option>
        <option>Title</option>
    </optgroup>

</select>

<input type = "text">
```



```
<p>Search:

<select name = "category">

    <option selected>Choose a field to search</option>

    <option>Games</option>

    <optgroup label = "Video">
        <option>Title</option>
        <option>Director</option>
        <option>Actor</option>
    </optgroup>

    <optgroup label = "Music">
        <option>Artist</option>
        <option>Title</option>
    </optgroup>

</select>

<input type = "text">
```



```
<p>Search:

<select name = "category">

    <option selected>Choose a field to search</option>

    <option>Games</option>

    <optgroup label = "Video">
        <option>Title</option>
        <option>Director</option>
        <option>Actor</option>
    </optgroup>

    <optgroup label = "Music">
        <option>Artist</option>
        <option>Title</option>
    </optgroup>

</select>

<input type = "text">
```



```
<p>Search:</p>
<div>
    <select multiple name = "films">

        <option>Games</option>
        <optgroup label = "Video">
            <option>Title</option>
            <option>Director</option>
            <option>Actor</option>
        </optgroup>
        <optgroup label = "Music">
            <option>Artist</option>
            <option>Title</option>
        </optgroup>

    </select>
</div>
<p><input type = "text"></p>
```

You can use <optgtoup> in multiple selection menus as well.

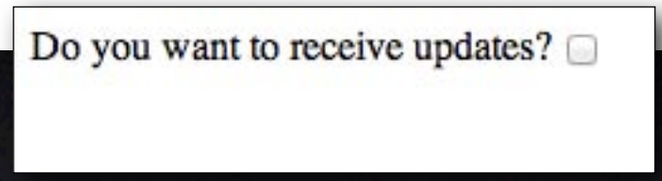
<input type = "checkbox">

<input type = "checkbox">

Checkboxes can be turned on or off. If a checkbox is turned *on* data is sent to the server. If it is *off* no data is sent.

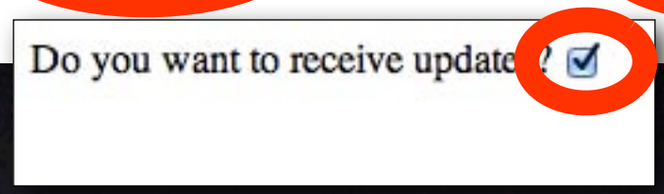
The data sent is specified in the **value** attribute (although many browsers send the value "on" if the **value** attribute is not present).

```
Do you want to receive updates?
<input type = "checkbox" name = "update" value = "yes">
```



If the checkbox is not selected nothing is sent to the server.

```
Do you want to receive updates?
<input type = "checkbox" name = "update" value = "yes">
```



If the checkbox *is* selected the name/value pair **update=yes** is sent to the server.

checked

This attribute (which needs no value) is added to the **<input type = "checkbox">** tag so that the checkbox will be on (i.e. will be checked) when the page loads (or is reset).

Checkboxes are off by default otherwise.

```
Do you want to receive updates?
<input type = "checkbox" name = "update"
                         value = "yes" checked >
```

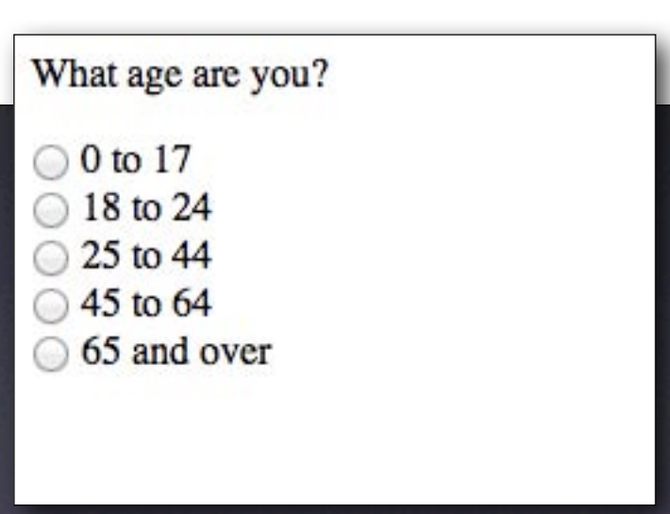
<input type = "radio">

<input type = "radio">

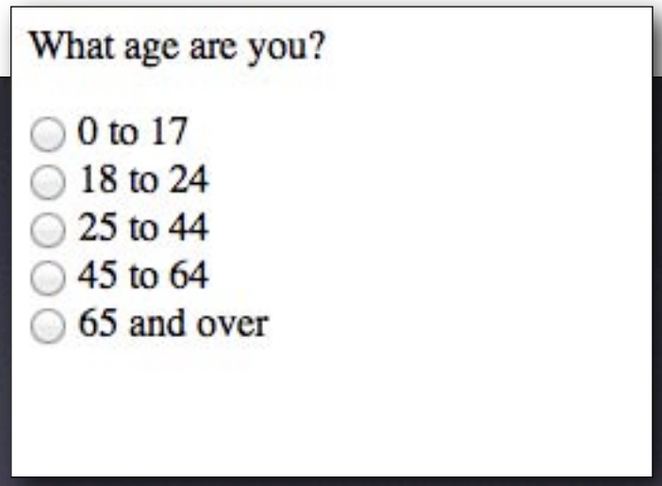
This tag creates a radio button. If several radio buttons have the same **name** the browser will only allow the user select one of them.

We therefore give each radio button a different **value** to distinguish them.

```
<p>What age are you?</p>

<div>
<input type = "radio" name = "age" value = "under18"> 0 to 17<br>
<input type = "radio" name = "age" value = "18to24"> 18 to 24<br>
<input type = "radio" name = "age" value = "25to44"> 25 to 44<br>
<input type = "radio" name = "age" value = "45to64"> 45 to 64<br>
<input type = "radio" name = "age" value = "over65"> 65 and over
</div>
```

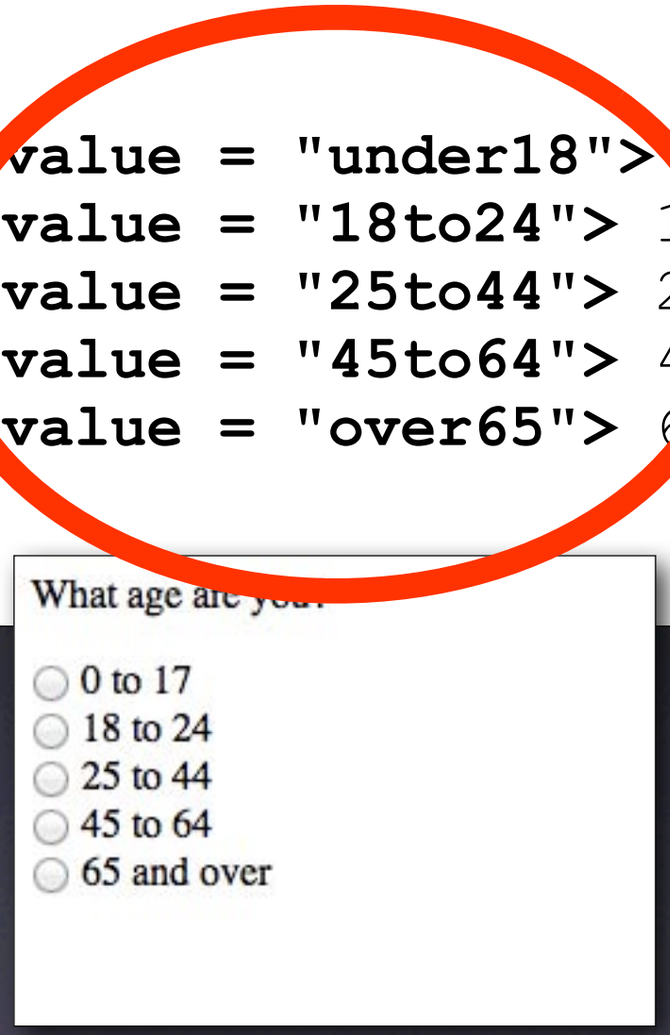


```
<p>What age are you?</p>

<div>
<input type = "radio" name = "age" value = "under18"> 0 to 17<br>
<input type = "radio" name = "age" value = "18to24"> 18 to 24<br>
<input type = "radio" name = "age" value = "25to44"> 25 to 44<br>
<input type = "radio" name = "age" value = "45to64"> 45 to 64<br>
<input type = "radio" name = "age" value = "over65"> 65 and over
</div>
```



The names must be the same for the browser to know you can only select one in this group.

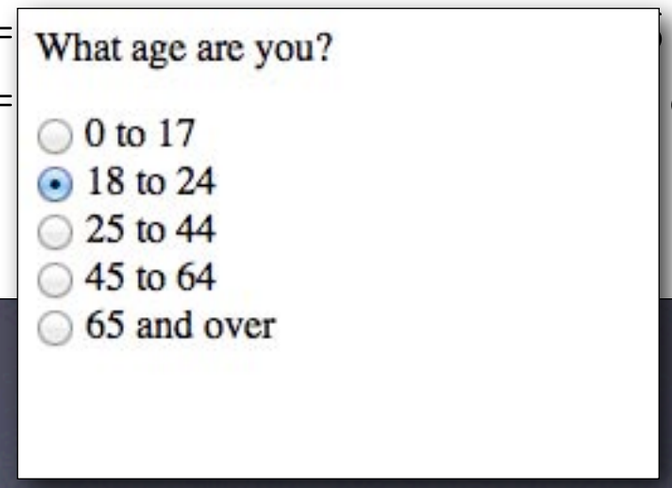
```
<p>What age are you?</p>

<div>
<input type = "radio" name = "age" value = "under18"> 0 to 17<br>
<input type = "radio" name = "age" value = "18to24"> 18 to 24<br>
<input type = "radio" name = "age" value = "25to44"> 25 to 44<br>
<input type = "radio" name = "age" value = "45to64"> 45 to 64<br>
<input type = "radio" name = "age" value = "over65"> 65 and over
</div>
```



Each radio button should have its own unique value that will be returned to the server.

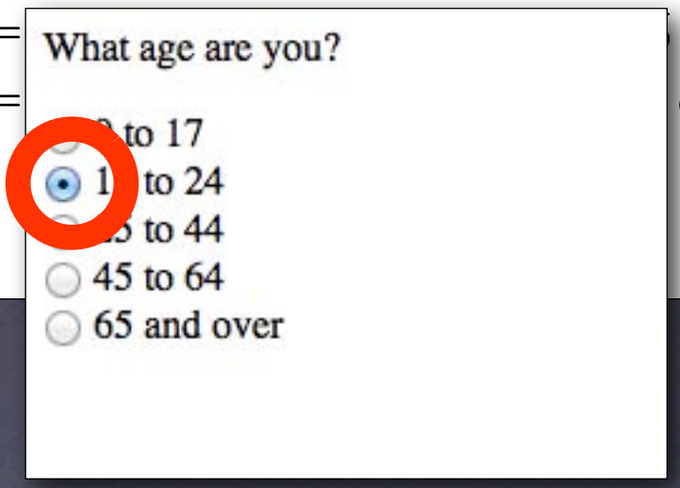
If you use the **checked** attribute for *one* of the radio buttons, it will automatically be selected when the page is first displayed.

```
<p>What age are you?</p>

<div>
<input type = "radio" name = "age" value = "under18"> 0 to 17<br>
<input type = "radio" name = "age" value = "18to24" checked> 18
to 24<br>
<input type = "radio" name = "age" value = "25to44"> 25 to 44<br>
<input type = "radio" name = "age" value =                  4<br>
<input type = "radio" name = "age" value =                  over
</div>
```



Only one radio button (with the same name) can have the checked attribute.

```
<p>What age are you?</p>

<div>
<input type = "radio" name = "age" value = "under18"> 0 to 17<br>
<input type = "radio" name = "age" value = "18to24" checked> 18
to 24<br>
<input type = "radio" name = "age" value = "25to44"> 25 to 44<br>
<input type = "radio" name = "age" value =                  4<br>
<input type = "radio" name = "age" value =                  over
</div>
```



<input>

<input>

This tag supports the **name** and **id** attributes in the same way as the previous form tags.

In many cases we can use the **value** attribute to specify the initial default value of a form element.

The **type** attribute, however, changes the way in which the element functions. Each possible value is compatible with different sets of attributes.

<input type = "text">

<input type = "text">

This is like a one line version of **<textarea>** (suitable for a single line of text).

```
Email: <input type = "text" name = "email1">
```

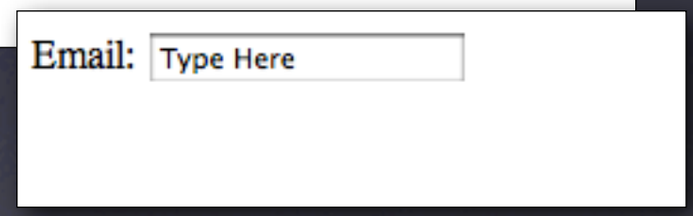
Email:

The following attributescan be used with this vartiant of **<input>**:

size	The length of the text-entry box (in characters)
maxlength	The maximum number of characters you are allowed enter

value Sets the default text

```
Email:
<input type = "text" name = "email1" value = "Type Here">
```



placeholder

Similar to its use in <textarea>

autocomplete

on - browser autocompletes what you type based on previous uses

off - browser doesn't autocompelte

list

You can specify a series of possible values for this form element.

If the user types in some characters they will be shown any of those values that match. If the user selects one of those options it is entered into the text field.

<datalist>

To specify the options used by **list** you must create a data list and give it an id.

```
<datalist id = "cities">

    <option>Carlow</option>
    <option>Cavan</option>
    <option>Cork</option>
    <option>Dublin</option>
    <option>Galway</option>
    <option>Limerick</option>

</datalist>
```

The **<option>** tag is used to specify the possible values.

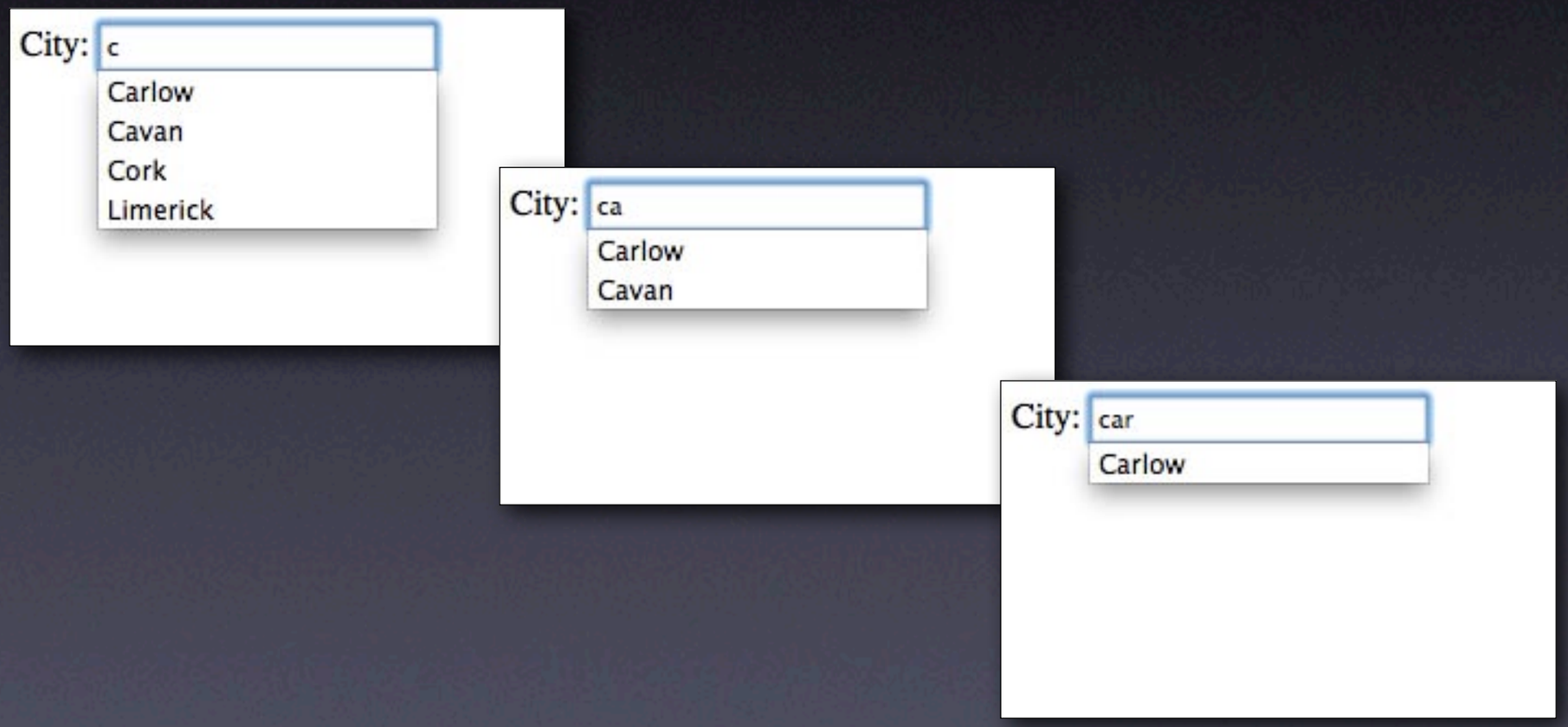
```
<datalist id = "cities">

    <option>Carlow</option>
    <option>Cavan</option>
    <option>Cork</option>
    <option>Dublin</option>
    <option>Galway</option>
    <option>Limerick</option>

</datalist>
```

You then specify the **id** of the **<datalist>** in the **list** attribute of the **<input>** tag.

```
<p>City:
<input type = "text" name = "email"  list = "cities">
</p>

<datalist id = "cities">

    <option>Carlow</option>
    <option>Cavan</option>
    <option>Cork</option>
    <option>Dublin</option>
    <option>Galway</option>
    <option>Limerick</option>

</datalist>
```

```
<p>City:
<input type = "text" name = "email" list = "cities">
</p>

<datalist id = "cities">

    <option>Carlow</option>
    <option>Cavan</option>
    <option>Cork</option>
    <option>Dublin</option>
    <option>Galway</option>
    <option>Limerick</option>

</datalist>
```

Now as you type in the text field you will be shown the appropriate options. Clicking on an option places it in the text field.

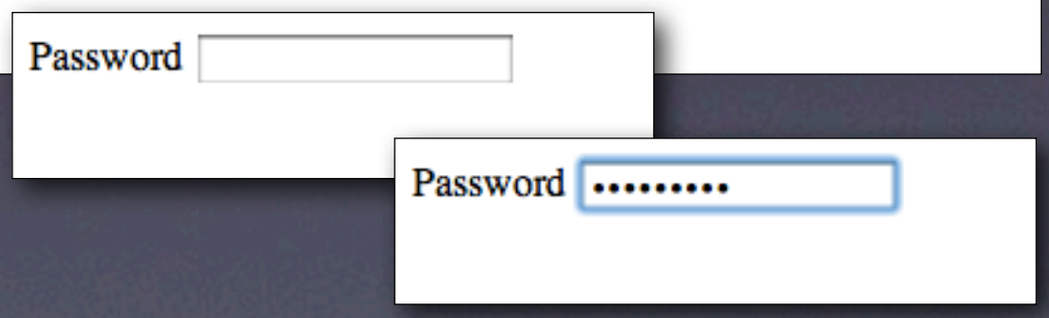


<input type = "password">

<input type = "password">

Identical to **<type = "text">** except the browser will display a bullet (•) for each character you type (there are no extra security precautions involved).

```
<div>
    Password
    <input type = "password" name = "pass">
</div>
```



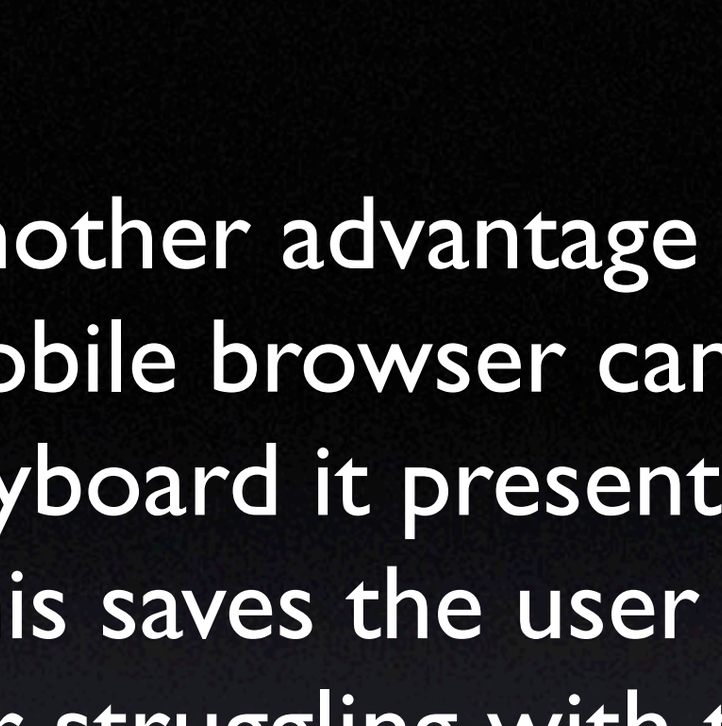
```
<input type = "tel">
<input type = "number">
<input type = "url">
<input type = "search">
<input type = "email">
```

These elements can be used instead of regular text fields to accept different types of data. In many cases they may not appear any different visually but the browser will know what kind of data you are expecting.

This adds another layer of semantics to your page (always a good thing) and can help the browser validate the data.

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

Another advantage of these tags is that a mobile browser can change the type of keyboard it presents to user to input the data. This saves the user switching keyboard layouts (or struggling with the incorrect one)




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

type="email"

type="url"

type="number"

type="search" changes the look of text field in most browsers.

```
<input type = "search" name = "searchterms">
```

These elements also support the same attributes as **type="text"**.

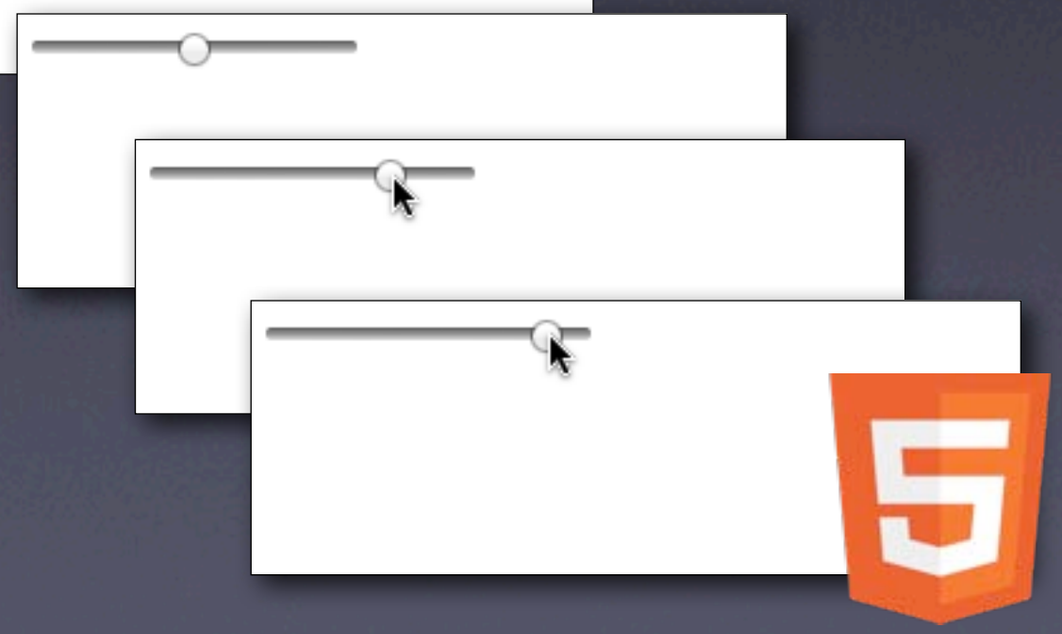
<input type = "range">

HTML5 provides many new form elements. Support for these will vary from browser to browser.

If a browser doesn't support one of these elements it will display a text field instead so you will still be able to collect the data (**type = "text"** is the default setting for the **<input>** element).

If you want the user to choose a discrete number value between two set values you can use type=range to allow them do this graphically.

```
<input type = "range" name = "rating">
```

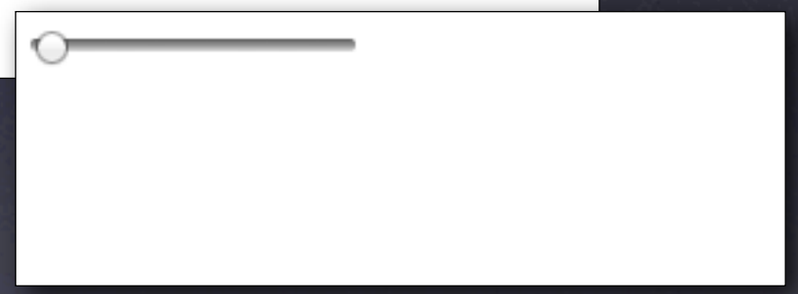


By default the slider returns a value between 0 and 100 and initially appears selecting the half way value (50 in this case).

min	The lowest value returned
max	The highest value returned
value	The initial selected value

The following example returns a number between 1 and 10, and selects 1 by default.

```
<input type = "range" name = "rating"
          min = "1" max = "10" value = "1">
```



step

[defaults to 1] The increments to the value made by the slider.

E.g **step = "5"** means the slider returns 0, 5, 10, 15, 20, etc.

If your browser doesn't support **type="range"** it will default to **type="text"** and collect the data that way.

```
Please rate this product (out of 10):
<input type = "range" name = "rating"
          min = "1" max = "10" value = "1">
```

Please rate this product (out of 10):

supports
type = "*range*"

Please rate this product (out of 10): 1

doesn't support
type = "*range*"

<input type="color">

If you want to the user to enter a hex colour value you can use this form element. If supported it displays a color picker.

```
Please choose a colour:
<input type = "color" name = "myColour">
```

You can also chose a default initial selection:

```
Please choose a colour:
<input type = "color" name = "myColour" value = "#ff0000">
```

If the element is not supported it may not be sufficient to offer a text field as an alternative (since it would assume the user knows the hex value of the colour they want).

In cases like this you can write javaScript that detects whether the element is supported. If not it can use a JavaScript solution instead.

```
<input type = "date">
<input type = "datetime">
<input type = "datetime-local">
<input type = "month">
<input type = "time">
<input type = "week">
```

These elements are for selecting a time value.

The method for picking the time may vary from browser to browser.

Time is usually returned in a standard format.

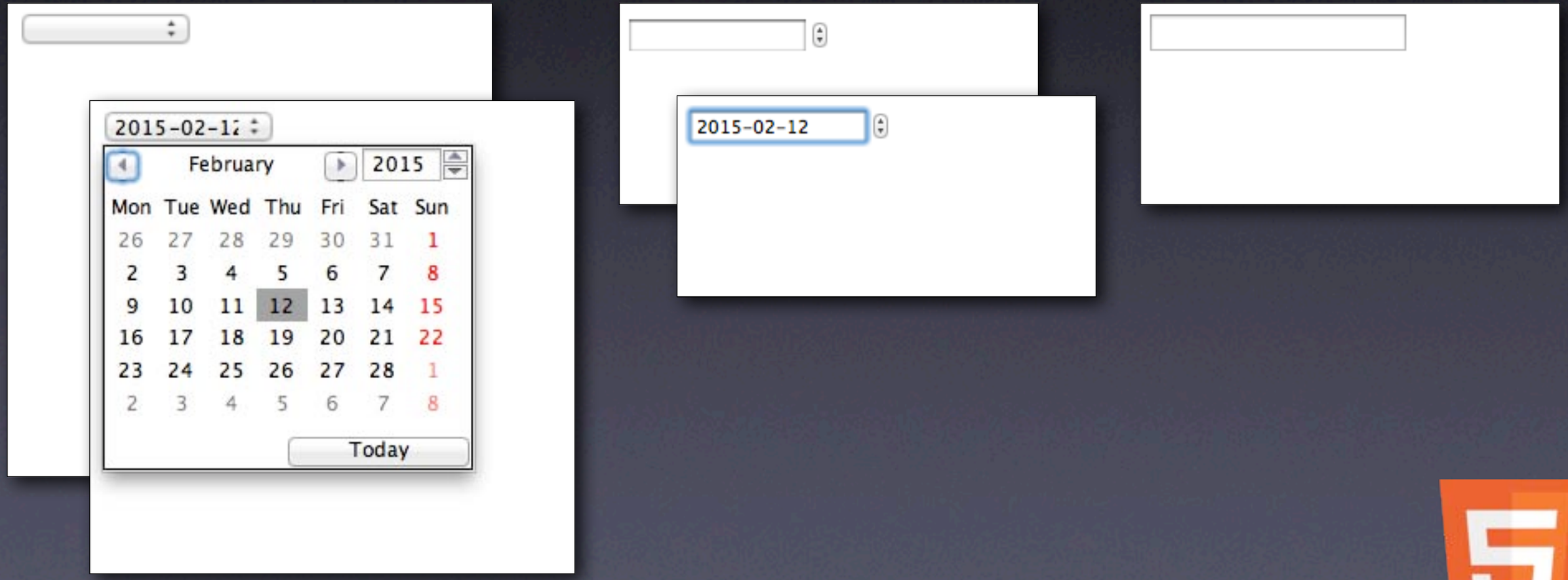
for

20:21 on February 12th 2015

type=...	value
datetime	2015-02-12T20:21Z
datetime-local	2015-02-12T20:21
week	2015-W7
month	2015-2
date	2015-02-12
time	20:21

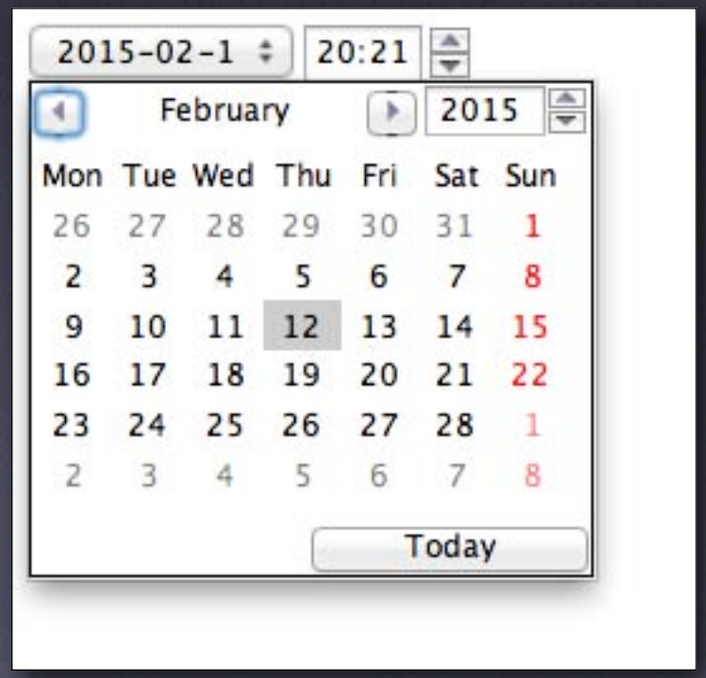
The support for this element can vary from browser to browser.

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



datetime includes the date and time and the timezone, whereas **datetime-local** doesn't include the timezone information.

```
<input type = "datetime-local" name = "when">
```



Browsers for smaller form factors can also select a more appropriate input method for the type of data your are expecting.

type="datetime-local"

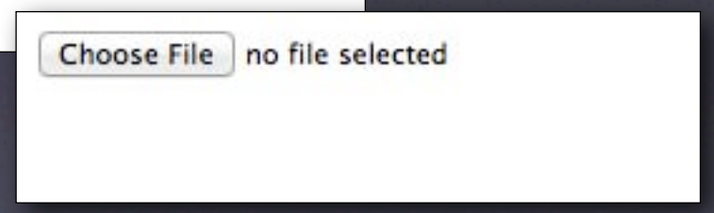
type="date"

<input type = "file">

<input type = "file">

This tag allows the user to submit a filename and the entire file to the server. The browser will implement it by letting the user browse their file system to select the file in question.

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



You can restrict the types of the files that the user can submit by specifying the permissible MIME types with the **accept** attribute.

Other Elements

Some form elements are not involved in collecting data from users.

Some of these may be non-interactive and some are for providing feedback to the user.

They are particularly useful when combined with JavaScript.

<input type = "hidden">

<input type = "hidden">

This tag does not interact with the user but sends a specified name/value pair to the server.

```
<input type = "hidden" name = "page"
                              value = "homepage">
```

<progress>

<progress>

This element allows you communicate the current status of some action via a progress bar (if supported).

You should provide an alternative feedback method for browsers that don't support it.

value

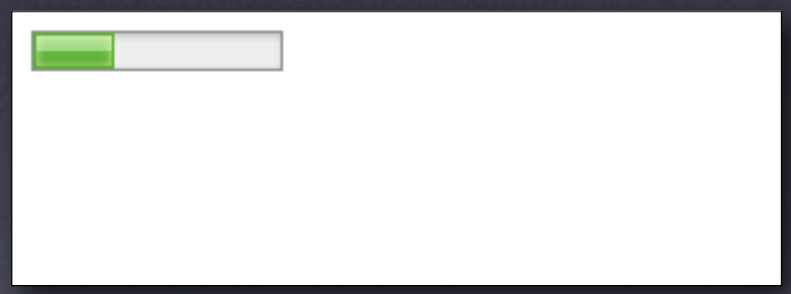
A number indicating the amount of progress

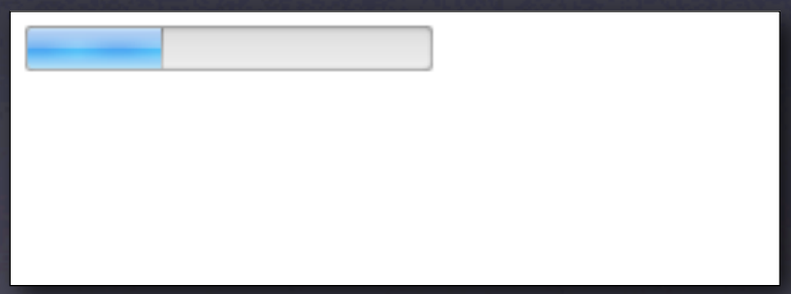
max

A number representing the total amount of work required to complete the task. If **value** = **max** the task is complete. The progress bar will indicate the proportion of **max** that **value** represents.

The implementation of the progress bar may vary from browser to browser.

```
<progress max = "100" value = "33" id = "p1"> 33% </progress>
```





If the browser doesn't support the element then the code between the tags will be displayed.

```
<progress max = "100" value = "33" id = "p1"> 33% </progress>
```

33%

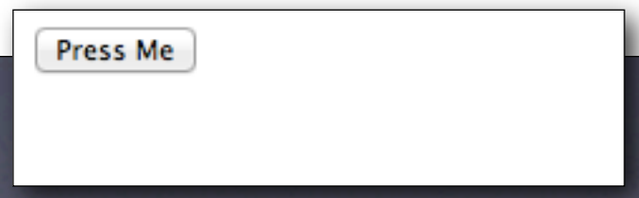
<input type = "button">

<input type = "button">

This tag renders a button that the user can click on. These can be used to generate JavaScriptnevents. Otherwise buttons perform no action of their own.

The **value** attribute sets the label for the button and is required.

```
<input type = "button" value = "Press Me" id = "bt1">
```



<input type = "reset">
<input type = "submit">

These are special types of buttons in that they actually have a function supported by HTML.

reset sets all the elements in the same form as the button to their default values.

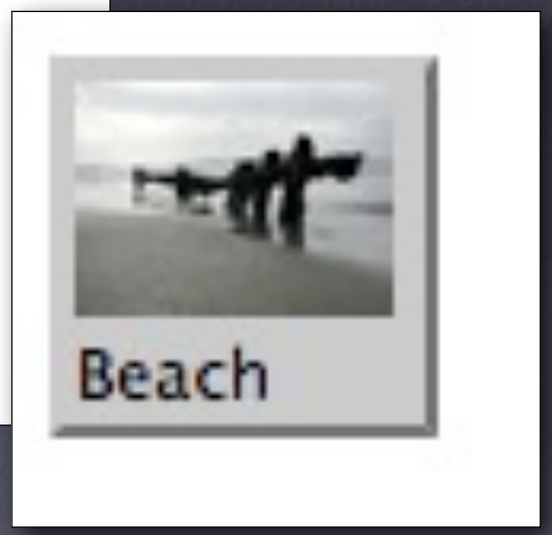
submit sends all the data in the same form as the button to a specifies script.

<button>

<button>...</button>

This tag is a replacement for **<input type = "button">** and **<input type = "image">**. The advantage being that any HTML and/or images between the tags appear on the button.

```
<button name = "button1" type = "button">

<img src = "beach.jpg" width = "50" alt = "beach">
<br>Beach

</button>
```



type

This attribute specifies how the button should act.

button Same as <input type = "button">

submit Same as <input type = "submit">

reset Same as <input type = "reset">

<input type = "image">

<input type = "image" >

This tag is similar to **<input type="submit">** (i.e. it causes the form data to be sent to a server) except that we can use a specified image (supplied by the required **src** attribute) instead of the regular HMTL button.

```
<input type = "image" src = "button.png">
```

Common Form Element Attributes

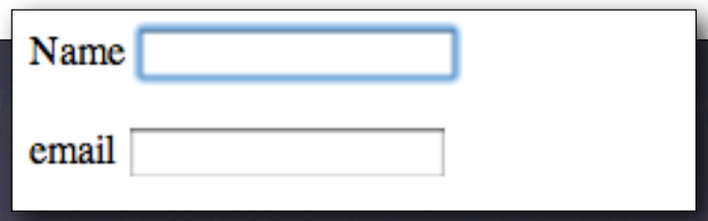
autofocus

Whenever you select a form element you are said to be giving it *focus*. *Blur* is the term used whenever a form element is deselected (i.e. you select another form element instead or another part of the document.). This is also known as *losing focus*.

There are several attributes common to all form elements that effect the way we manage focus and blur.

In this example the *name* text field is automatically selected and ready to accept input when the page is loaded.

```
<p>Name <input type = "text" name = "name" autofocus></p>

<p>email <input type = "email" name = "name"></p>
```



tabindex & accesskey

Every form element supports these attributes. (every element as of HTML5)

They serve the same purpose as their <a> counterparts.

I.e. you can specify the other in which the tab key navigates the elements (or the key that automatically selects an element).

disabled

In some cases you may want to ensure that some form elements never get focus, i.e. can never be used.

To disable a form element you use the **disabled** boolean attribute. This will not only disable the form element but will also grey out the element on the page.

This can also be set dynamically with JavaScript.

E.g.

```
<input type = "button" value = "Don't Press Me" disabled>
```

In this example the element on the right in each pair is disabled, as is the ***Two*** option in the menu.

readonly

A weaker version of **disabled** that stops you changing the value of an element. This may not preclude you interacting with that element (unlike **disabled**).

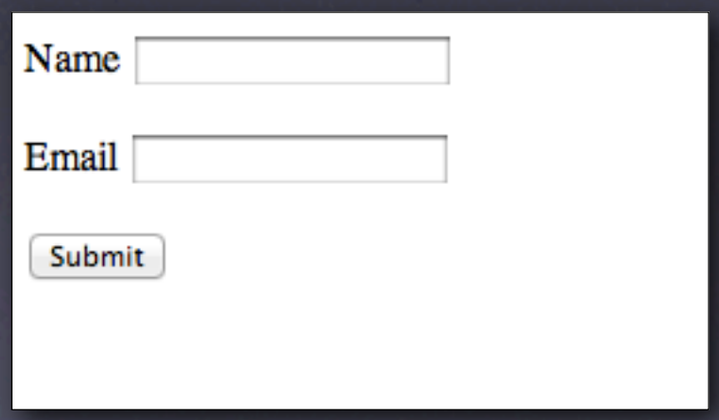
required

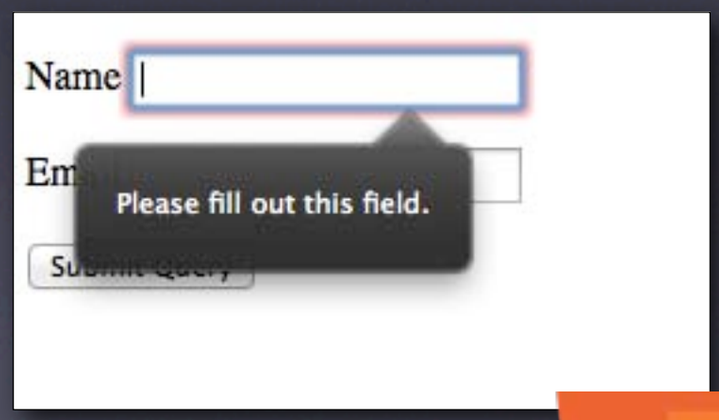
You can indicate to the browser that some elements must be filled in in order for the form to tbe submitted to the server.

If the element is not filled in the form will not send its data and will highlight the offending element(s).

```
<form action = "script.php">

<p>Name <input type = "text" name = "name"  required></p>

<p>Email <input type = "email" name = "email"></p>

<input type = "submit">

</form>
```

After pressing *Submit*





CSS styles are applied to offending form elements. You can style these your self using the following CSS3 pseudo classes.

```
:valid

:invalid
```

There are attributes of **<form>** and other elements that can override the validation process.

Organising Forms

<label>

<label>...</label>

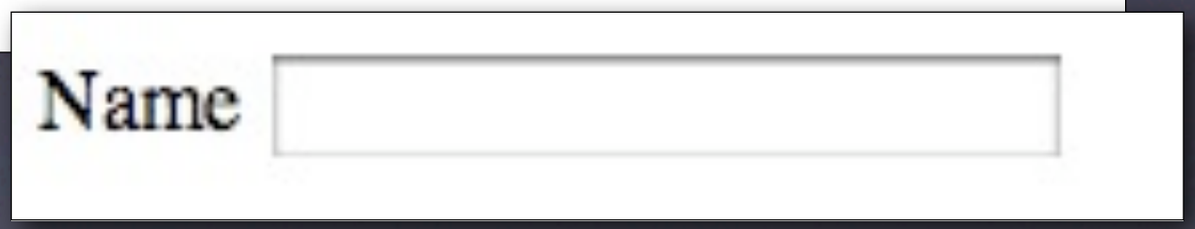
This tag associates a label with a form element.

It has no visual effect and is intended for semantic and accessibility reasons.

In the form below a person viewing the page can assume the "Name" text is the label of the text field.

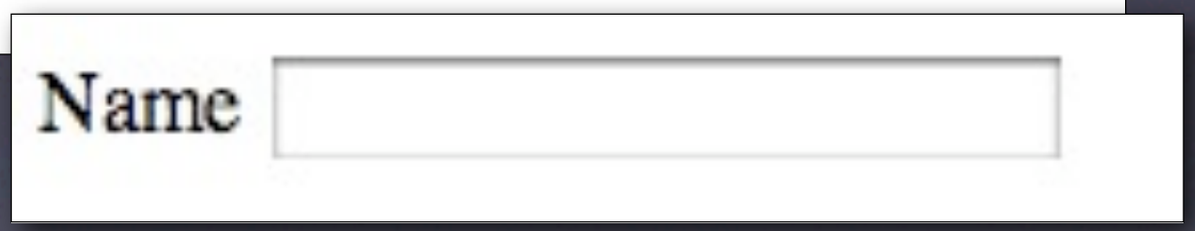
However, examining the HTML gives no indication that that is the case. We need a way to specify that the "Name" text is the label for the form element.

```
<p>
Name
<input type = "text" name= "name" >
</p>
```



First we can indicate that the text is a label by using the **<label>** tag.

```
<p>
<label>Name</label>
<input type = "text" name= "name" >
</p>
```



Next we must specify which form element we are labelling. To do this we give the form element a unique **id**. The **for** attribute of the **<label>** tag should reference this.

```
<p>
<label for = "name1">Name</label>
<input type = "text" name= "name" id = "name1">
</p>
```

Name

<fieldset> & <legend>

<fieldset>...</fieldset>

This tag is used to better describe the layout of a form. It is also intended for improving accessibility on alternative browsing devices.

This tag is placed around logical groupings of form elements to better describe them.

Some browsers will represent the fieldset graphically.

The following is a basic form.

```
<form action = "get" action = "script.php">

<p>Name <input type = "text" name= "name">
</p>

<p>Address<br>
<textarea name = "address" cols = "20"
rows = "5"></textarea>

<p>Credit Card Nr.
<input type = "text" name= "card"></p>

<p>Expiry Date
<input type = "text" name= "date"></p>

<p>Comments<br>
<textarea name = "comments" cols = "20"
rows = "5"></textarea>

</form>
```

Name

Address

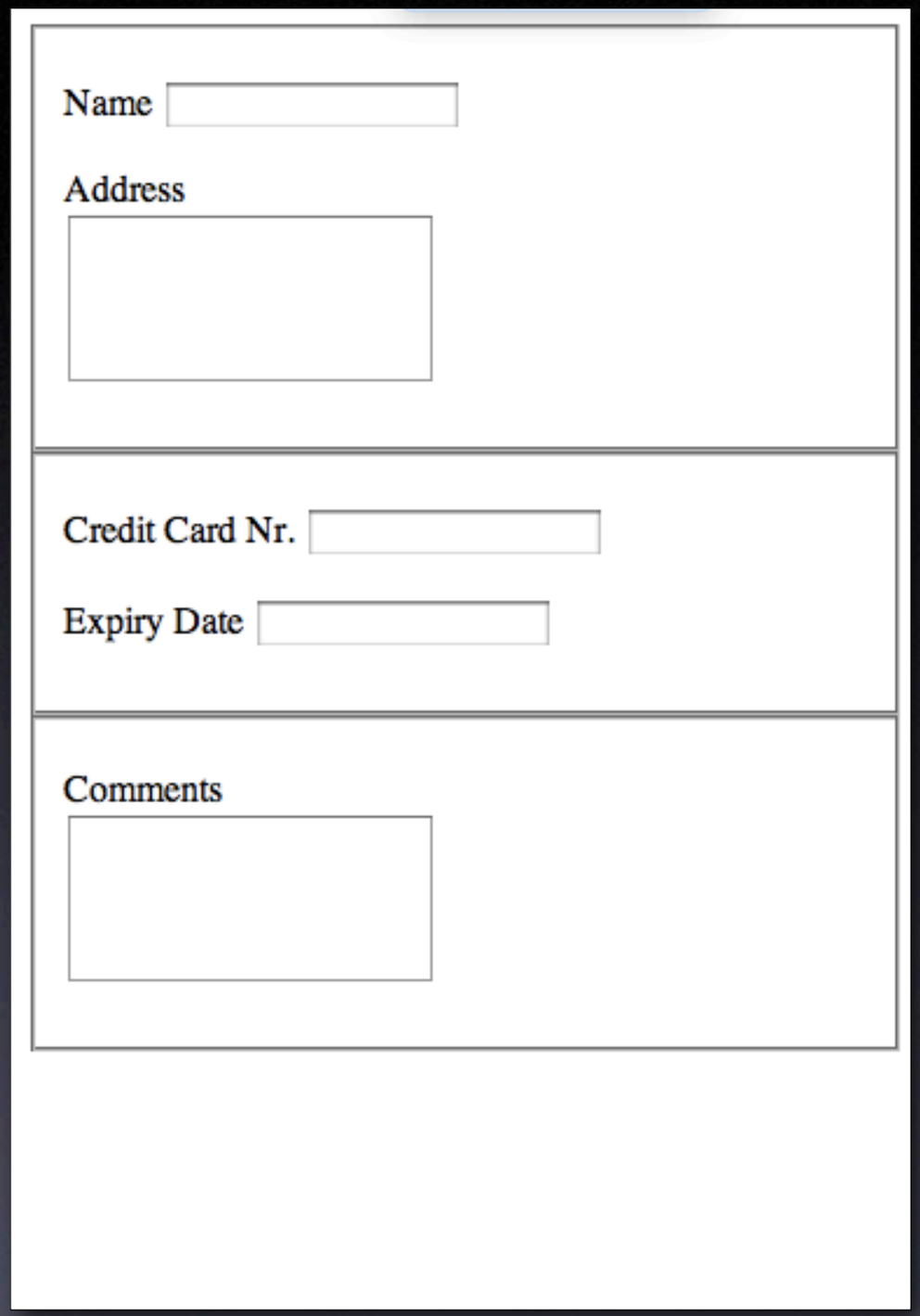
Credit Card Nr.

Expiry Date

Comments

Adding **<fieldset>** tags around groups of elements makes the form easier to understand.

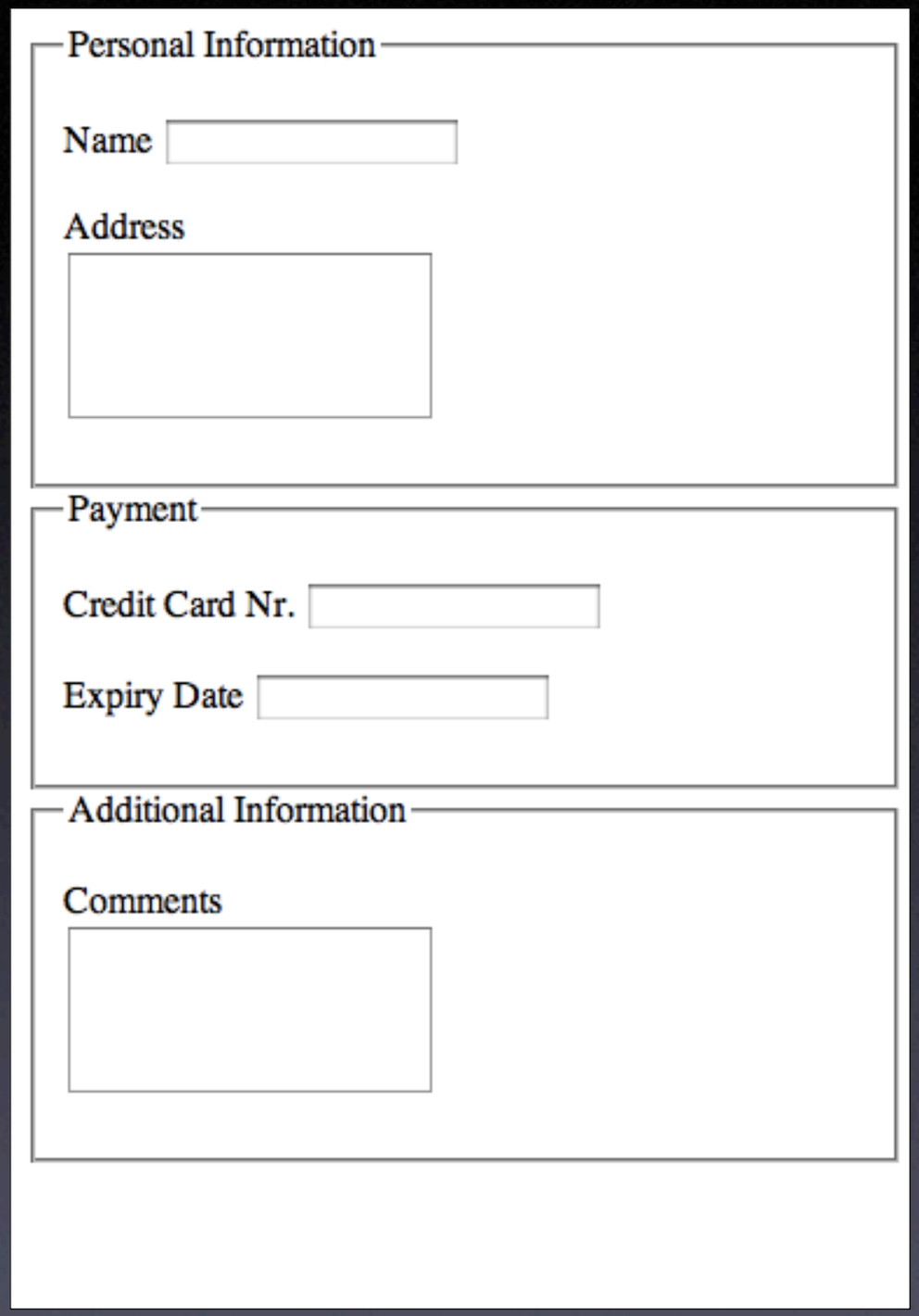
```
<form action = "get" action = "script.php">

<fieldset>
<p>Name <input type = "text" name= "name">
</p>

<p>Address<br>
<textarea name = "address" cols = "20"
rows = "5"></textarea>
</fieldset>

<fieldset>
<p>Credit Card Nr.
<input type = "text" name= "card"></p>

<p>Expiry Date
<input type = "text" name= "date"></p>
</fieldset>

<fieldset>
<p>Comments<br>
<textarea name = "comments" cols = "20"
rows = "5"></textarea>
</fieldset>

</form>
```



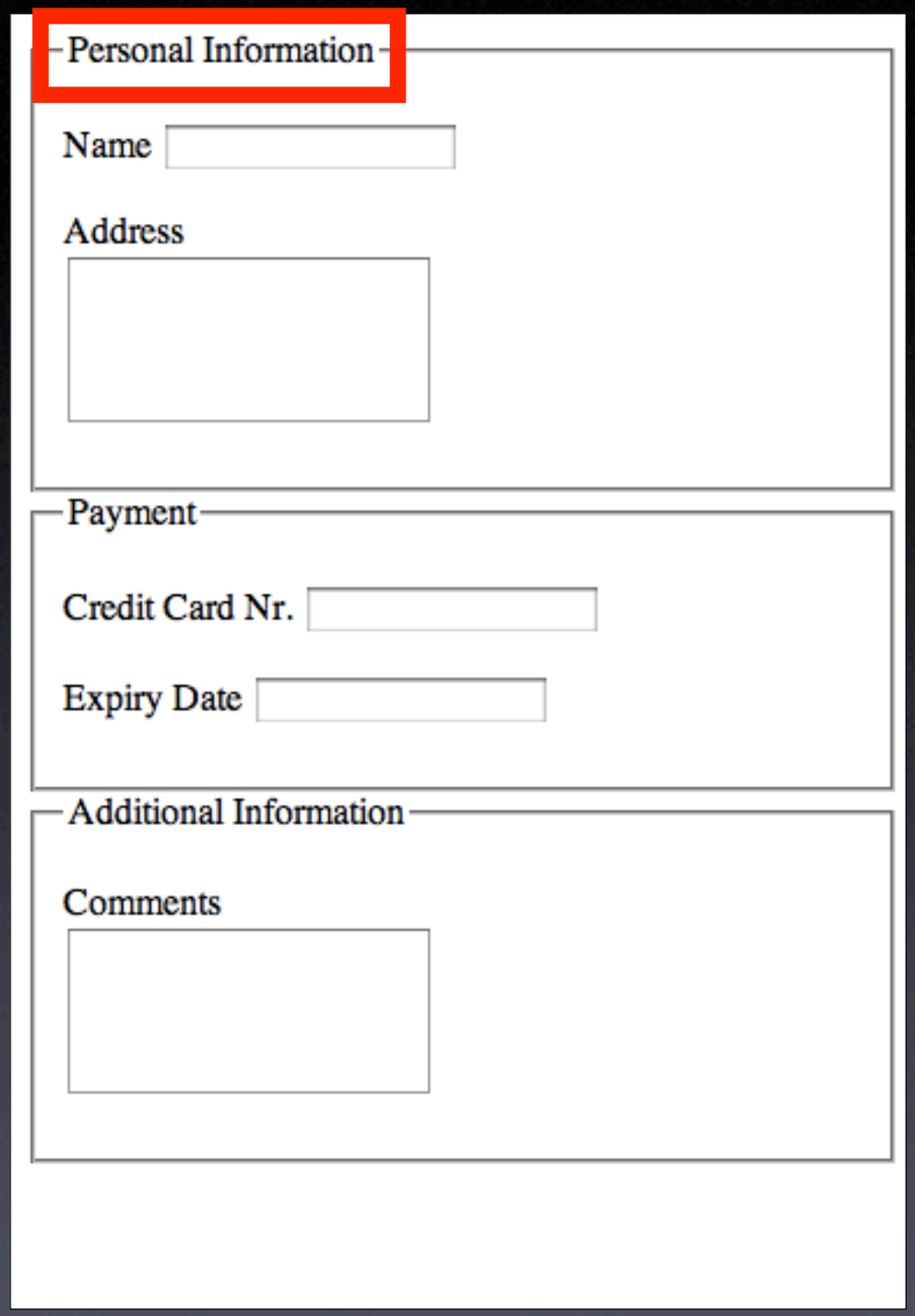
<legend>...</legend>

This tag adds a caption to a fieldset and must appear ***inside*** the **<fieldset>** it applies to.

```
<form action = "get" action = "script.php">

<fieldset>
<legend>Personal Information</legend>

<p>Name <input type = "text" name= "name">
</p>

<p>Address<br>
<textarea name = "address" cols = "20"
rows = "5"></textarea>
</fieldset>

<fieldset>
<legend>Payment</legend>
<p>Credit Card Nr.
<input type = "text" name= "card"></p>
<p>Expiry Date
<input type = "text" name= "date"></p>
</fieldset>

<fieldset>
<legend>Additional Information</legend>

<p>Comments<br>
<textarea name = "comments" cols = "20"
rows = "5"></textarea>
</fieldset>

</form>
```



```
<form action = "get" action = "script.php">

<fieldset>
<legend>Personal Information</legend>

<p>Name <input type = "text" name= "name">
</p>

<p>Address<br>
<textarea name = "address" cols = "20"
rows = "5"></textarea>
</fieldset>

<fieldset>
<legend>Payment</legend>
<p>Credit Card Nr.
<input type = "text" name= "card"></p>
<p>Expiry Date
<input type = "text" name= "date"></p>
</fieldset>

<fieldset>
<legend>Additional Information</legend>

<p>Comments<br>
<textarea name = "comments" cols = "20"
rows = "5"></textarea>
</fieldset>

</form>
```



We can further style these elements with CSS.

<form>

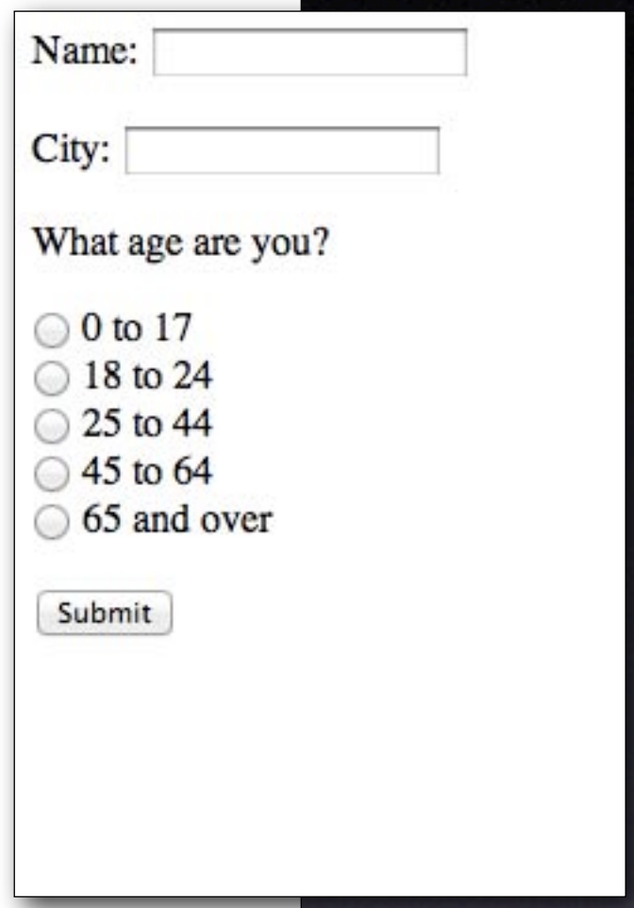
<form>...</form>

A form is a *collection of individual form elements* on a web page whose data can be collected and sent to a script on the server.

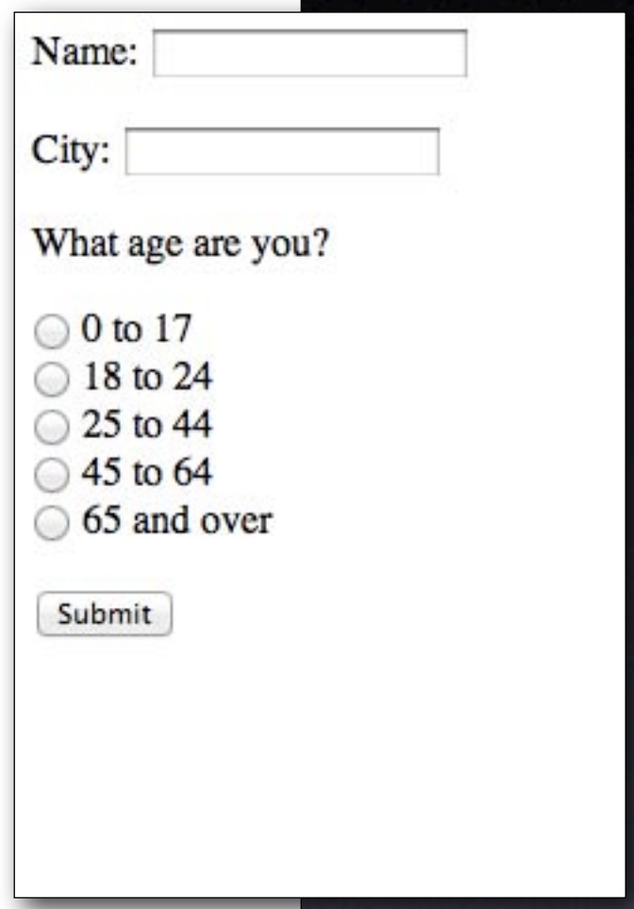
It basically identifies a series of form elements as belonging together.

We can identify the elements that make up a single form by placing them between **<form> ... </form>** tags.

```
<form action = "script.php" method = "get">

<p>Name: <input type = "text" name = "name"></p>
<p>City: <input type = "text" name = "city"></p>
<p>What age are you?</p>
<p><input type = "radio" name = "age"
  value = "under18"> 0 to 17<br>
  <input type = "radio" name = "age"
  value = "18to24"> 18 to 24<br>
  <input type = "radio" name = "age"
  value = "25to44"> 25 to 44<br>
  <input type = "radio" name = "age"
  value = "45to64"> 45 to 64<br>
  <input type = "radio" name = "age"
  value = "over65"> 65 and over
</p>
<div><input type = "submit"></div>
</form>
```

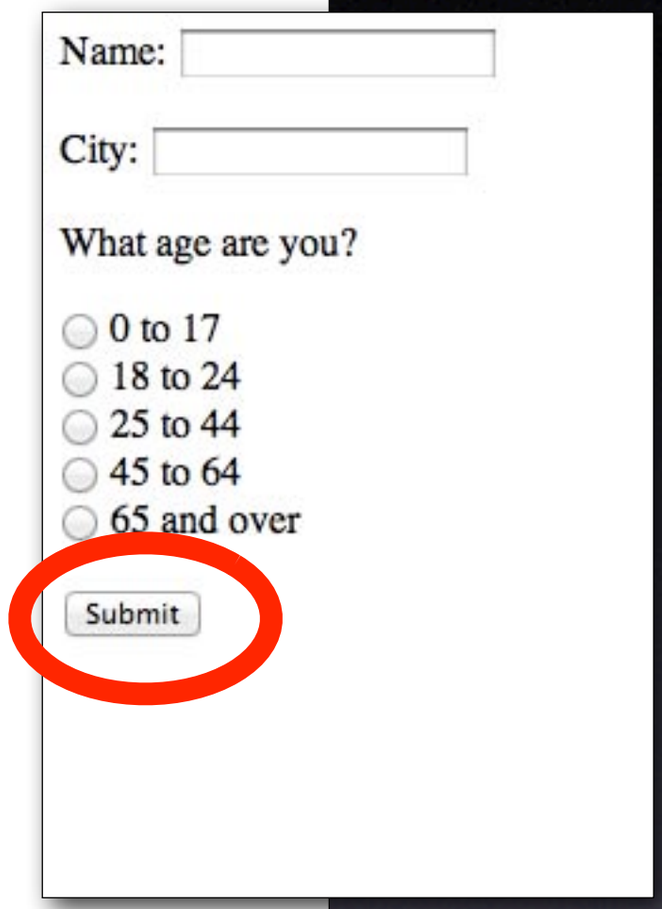


```
<form action = "script.php" method = "get">

<p>Name: <input type = "text" name = "name"></p>
<p>City: <input type = "text" name = "city"></p>
<p>What age are you?</p>
<p><input type = "radio" name = "age"
  value = "under18"> 0 to 17<br>
  <input type = "radio" name = "age"
  value = "18to24"> 18 to 24<br>
  <input type = "radio" name = "age"
  value = "25to44"> 25 to 44<br>
  <input type = "radio" name = "age"
  value = "45to64"> 45 to 64<br>
  <input type = "radio" name = "age"
  value = "over65"> 65 and over
</p>
<div><input type = "submit"></div>
</form>
```



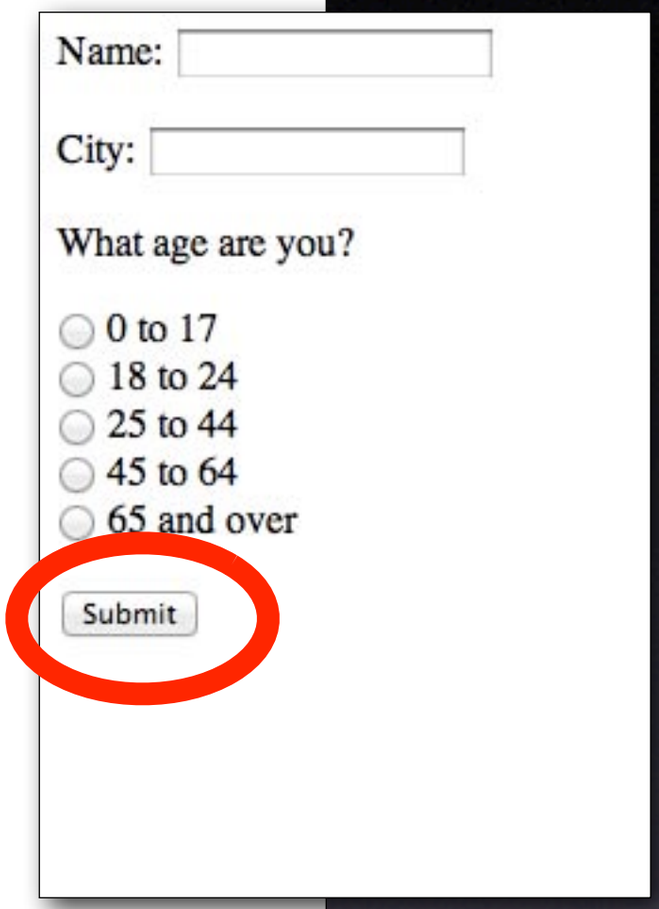
The form tag encloses a series of form elements.

```
<form action = "script.php" method = "get">

<p>Name: <input type = "text" name = "name"></p>
<p>City: <input type = "text" name = "city"></p>
<p>What age are you?</p>
<p><input type = "radio" name = "age"
  value = "under18"> 0 to 17<br>
  <input type = "radio" name = "age"
  value = "18to24"> 18 to 24<br>
  <input type = "radio" name = "age"
  value = "25to44"> 25 to 44<br>
  <input type = "radio" name = "age"
  value = "45to64"> 45 to 64<br>
  <input type = "radio" name = "age"
  value = "over65"> 65 and over
</p>
<div><input type = "submit"></div>
</form>
```



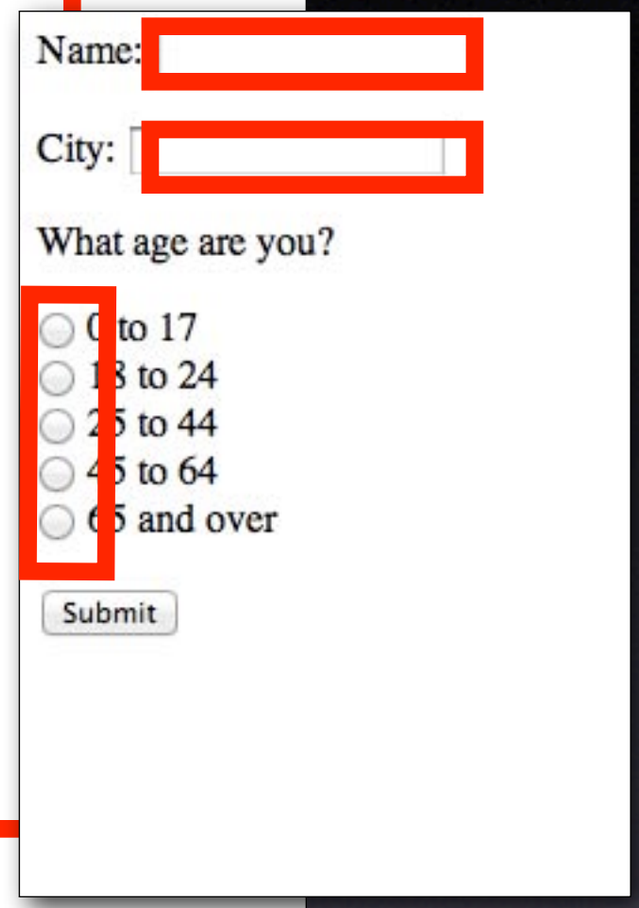
After you fill in the form you can click on the **Submit** button.

```
<form action = "script.php" method = "get">

<p>Name: <input type = "text" name = "name"></p>
<p>City: <input type = "text" name = "city"></p>
<p>What age are you?</p>
<p><input type = "radio" name = "age"
  value = "under18"> 0 to 17<br>
  <input type = "radio" name = "age"
  value = "18to24"> 18 to 24<br>
  <input type = "radio" name = "age"
  value = "25to44"> 25 to 44<br>
  <input type = "radio" name = "age"
  value = "45to64"> 45 to 64<br>
  <input type = "radio" name = "age"
  value = "over65"> 65 and over
</p>
<div><input type = "submit"></div>
</form>
```



... and locates the enclosing <form> tags.

```
<form action = "script.php" method = "get">

<p>Name: <input type = "text" name = "name"></p>
<p>City: <input type = "text" name = "city"></p>
<p>What age are you?</p>
<p><input type = "radio" name = "age"
  value = "under18"> 0 to 17<br>
  <input type = "radio" name = "age"
  value = "18to24"> 18 to 24<br>
  <input type = "radio" name = "age"
  value = "25to44"> 25 to 44<br>
  <input type = "radio" name = "age"
  value = "45to64"> 45 to 64<br>
  <input type = "radio" name = "age"
  value = "over65"> 65 and over
</p>
<div><input type = "submit"></div>
</form>
```

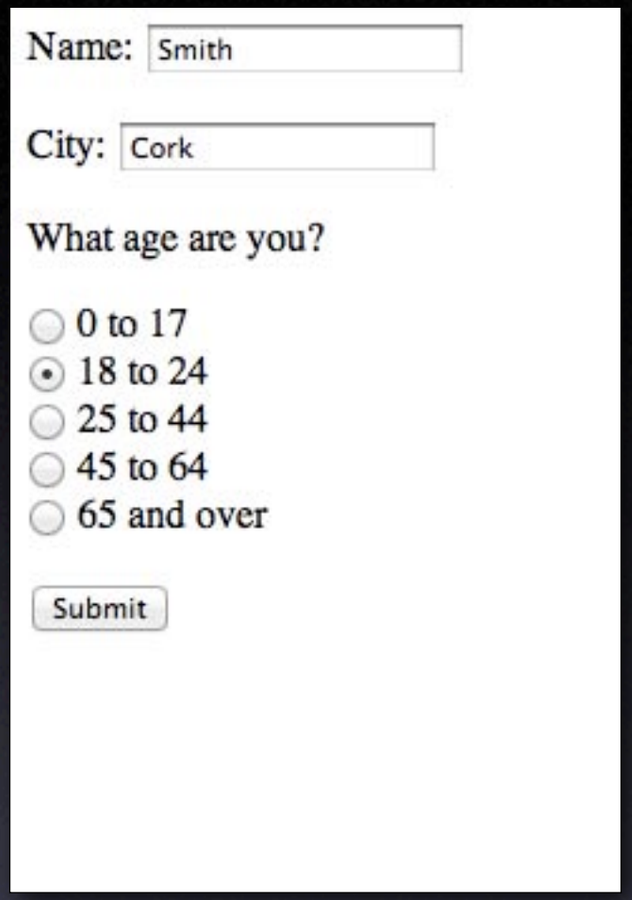


Data in the enclosed elements is then collected.

name=Smith

city=Cork

age=18to24



The data collected by the form is a series of name/value pairs.

Now the browser must send the data to a script on the server.

We specify this script with the **action** attribute in the <form> tag.

```
<form action = "script.php" method = "get">
```

We can also specify how the information is sent to the script. In this example we use the GET method (the default).

```
<form action = "script.php" method = "get">
```

If our form was located in this directory...

http://server.com/dir/

Then the browser invokes this script:

http://server.com/dir/script.php?name=Smith&city=Cork&age=18to24

Notes

- A reset button will reset all the form elements in the same form as itself
- Form tags can contain any other HTML tags as well as the form elements
- Pressing return in a text field can submit a form on some browsers.

form = ...

In HTML5 we don't need to place the **<form>** tag around the other input/select/etc. tags. I.e. the form elements don't have to be descendants of the **<form>** element.

Instead we can place the **<form>** tag anywhere on the page. It need not contain any HTML, but it must have an **id** attribute to identify it.

```
<form action = "script.php" id = "form1"></form>
```

Every form element that we wish to associate with that form should indicate the form it belongs to with its **form** attribute.

```
<div>
Name: <input type = "text" name = "name" form = "form1">
</div>

<div>
Password: <input type = "pass" name = "pass" form = "form1">
</div>
```