

HTML Form Structure

HTML Form

- A form structure is a practical method for collecting information from the user
- Forms on a web page are set up using a combination of HTML form tag elements and an associated script written in PHP, ASP, Perl, python, or other script language
- Scripts are commonly called CGI (Common Gateway Interface) and run on the web server

ISP Script

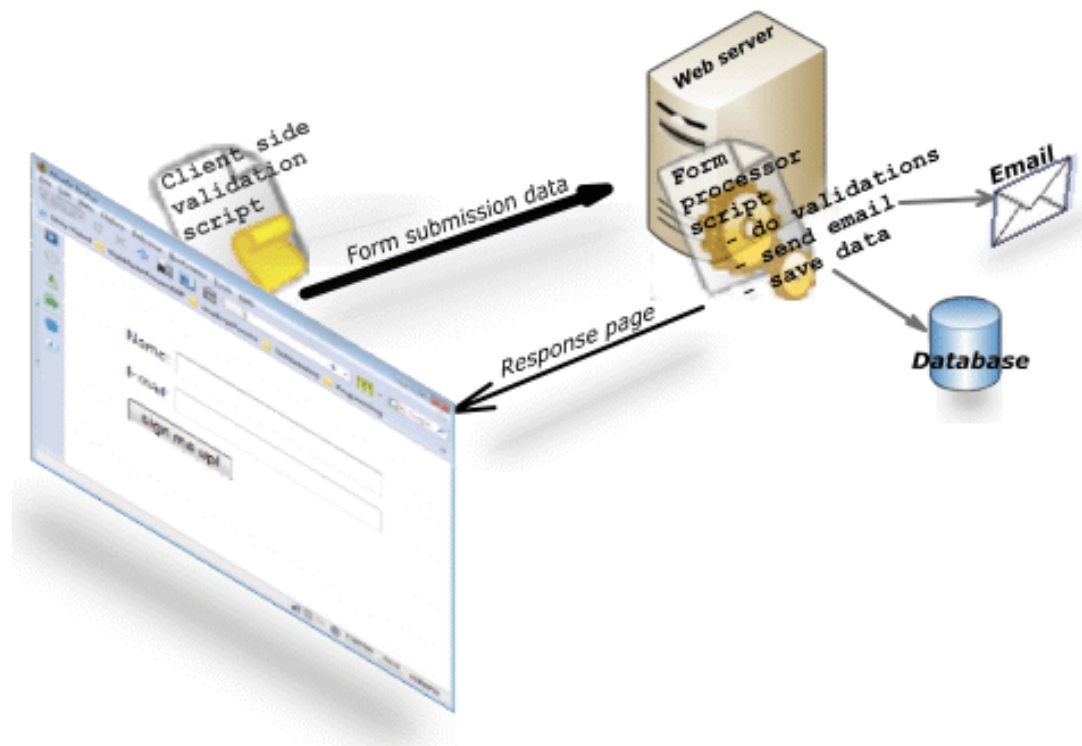
- Many Internet Service Providers supply pre-built CGI scripts to allow customers to use them in their HTML documents
- Useful for collecting email addresses, personal info, ordering products, etc
- Normally a CGI script does not have any access to the client computer

HTML v4 form elements

- HTML v4 form elements (called controls) include
 - **Input boxes** for text and numeric values
 - **Radio buttons**, also called option buttons to select a single option from a predefined list
 - **Selection lists** for longer lists of options in a drop-down listbox
 - **Check boxes** to specify an item as either being selected or not selected
 - **Text areas** for longer amounts of text
 - **Submit** button to submit the updated form data to the CGI script
 - **Reset** button to reset the form to its original state

HTML form

- Each form element in which the user can enter information is called a **field**
- Information entered into a field is called the **field value**, or simply the **value**



HTML form

- Form elements have attributes
 - **name** , used to associate the form control's current data when it is sent to the server
 - **id** , used to uniquely identify a form control in the DOM and associate a **label** (optional)
 - **value** , used to provide an initial value to a form control (optional)

```
<label for="fname">Enter name:</label>
```

```
<input type="text" name="firstname"  
      id="fname" value="Your first name">
```

Form tag

- The **<form>** element includes properties that control how the form is processed, information on which CGI script is used, how the data is transferred to the script
- `<form name = "myForm"`
 `action = "script name"`
 `method = "method name" >`
 ... form elements
 `</form>`

Form tag

- A form structure may not be defined inside an existing form structure
- Multiple forms may be defined in a page but each form should have a unique **name** attribute value
- Form names should not contain spaces
- Omitting the form tag's action attribute will not break the form but will mean nothing happens when user clicks on the Submit
- Form controls must be defined inside the <form> element

TextArea

- A field for entering multiple lines of information
- By default, the shape is a blank field of four lines and 40 characters wide
- **Name** attribute required
- To control how the text is wrapped in a text area, use the **wrap** attribute (soft is the default)
 - wrap="off" turns off text wrapping
 - wrap="soft" or "virtual" turns text wrapping on but does not send text wrapping data to web server
 - wrap="hard" or "physical" turns text wrapping on and also sends text wrapping data to web server

TextArea Form Element

- `<form name="myForm" ... >`
 `<textarea name="comments"`
 `rows="4"`
 `columns="40">`
 `</textarea>`
 `</form>`
- Defines a text area in the form with four lines and 40 characters wide

HTML Textarea example

A simple form that uses textarea

1. Story:

2. Comments:

3. Notes:

Some default text appears in here.

4. Confirm:

You cannot edit
this text.

[Page sample](#)

[Page sample CSS enhanced](#)

Select Form Element

- Has two parts: `<select>` tag and `<option>` tag
- Shows a list of options in either a pop-up menu or a scrolling list
- Number of options to show use `size="n"`
- Multiple selections use `multiple="multiple"`
- Preselect options with `selected="selected"` in the option tag

Select

- ```
<select name="payment">
 <option selected="selected"
 value="credit">Credit Card
</option>
 <option value="debit">Bank Debit
</option>
 <option value="cheque">Cheque
</option>
</select>
```

# Select optgroup

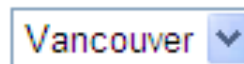
- Items can be grouped into a separate submenu

```
<optgroup label= "Province">
 <option value="BC">BC </option >
 <option value="AB">Alberta </option >
</optgroup>
```

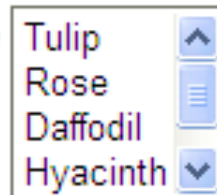
# HTML Select example

A simple form that uses select

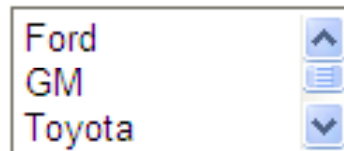
1. Favourite city:



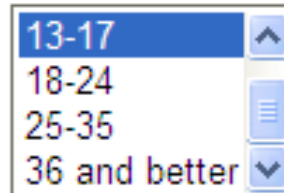
2. Garden flowers: (pick multiple)



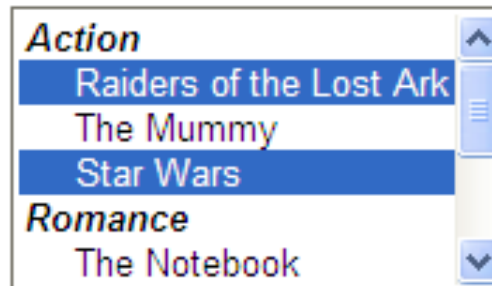
3. Cars Owned:



4. Age Group (set a default):



5. Favourite Movies:



[Page sample](#)

[Page sample CSS enhanced](#)

# Input Form Element

- Unlike `textarea` and `select` elements the `input` element has a single tag
- Used for text box, check boxes, radio buttons, passwords, and many other form element types
- Attributes:
  - `name`, the name of the data field
  - `size`, the length of the field in characters, default is 20
  - `maxlength`, max number of input characters to accept
  - `value`, usage depends on the type of input element



# Input Form Element

- **value** attribute, if the input element type is text or password, this is the default text to display; for checkbox or radio button, this is the value returned to the server; for submit and reset buttons this is the text to show inside the button
- **checked** attribute, sets a radio button or check box to "on" – usage: `checked="checked"`

# Input Element – text type

- Type attribute
  - **Text**, default value for type attribute, the **text** type is used in cases a type is specified unknown by the browser (e.g. HTML 5 has new input types like **tel** and **email**), this form element accepts text from the user

```
<input name="firstname">
```

```
<input type="text"
 name="phone"
 size="15"
 maxlength="12">
```

```
<input name="country"
 value="Canada">
```

# HTML Input text example

1. First name

2. Last name

3. Address

Enter your address here

4. Reason

This is readonly text.

[Page sample](#)

[Page sample CSS enhanced](#)

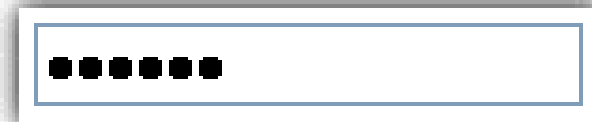
# Input Element – password type

- `password`, a type of text field but the user's typed characters are shown onscreen as bullets to prevent echoing of characters – this is **not** encrypting the data sent to the server

```
<input type="password"
 name="password"
 size="30"
 maxlength="30" >
```

[Sample page](#)

[Sample page CSS enhanced](#)



# Input Element – checkbox type

- **checkbox**, a switch control is "on" when its checked attribute is set (e.g. **checked**="checked" )
- checkbox controls within the form may share the same name value
  - Zero, one, or many selectable options
  - Can make them preselected with checked="checked"
  - To make PHP scripts work easier place **[]** at the end of the **name** attribute
  - If **value** attribute is **not** provided, then either 0 or 1 is used

# Checkbox example

```
<input type="checkbox" name="os[]" value="unix"> Unix
```

```
<input type="checkbox" name="os[]" value="Win" > Windows XP
```

```
<input type="checkbox" name="os[]" value="Mac" checked="checked">Mac OS
```

☐ Unix ☐ Windows XP ☒ Mac OS

[Sample page](#)

[Sample page CSS enhanced](#)

# Input Element – radio button type

- Allows only one option selected
- All the radio elements must have same name attribute
- Can supply optional **checked="checked"** to preselect as the default

**<input** type="radio" name="user" value="home">  
Home

**<input** type="radio" name="user" value="business"  
**checked="checked"** > Business

[Sample page](#)

[Sample page CSS enhanced](#)

# Input Element – hidden type

- Creates an undisplayed field in the form
- Used to send calculated or some default value information to the server when form data is submitted

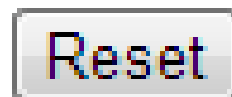
```
<input type="hidden" name="IM" value =
"smmh@hostmail.com">
```



# Input Element – reset type

- Displays a push button with the present function of clearing all the data in the form back to its original state
- Name and value attributes are optional unless there is more than one reset button
- Value attribute can be used to show text in the reset button

`<input type="reset">`



# Input Element – submit type

- Displays a push button with the preset function of sending the entered data in the form to the server for processing
- Value attribute defines what text to show in the button
- Name and value attributes optional unless there is more than one submit button

```
<input type="submit" value= "Click here to
download" name = "download">
```

A rectangular button with a light gray background and a thin black border. The text "Click here to download" is centered on the button in a blue, sans-serif font.

# Input Element – image type

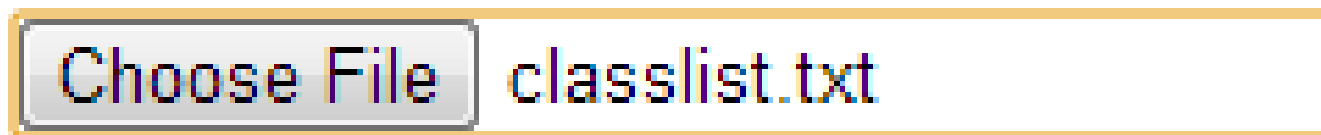
- Can be used to act like a submit button
- Rarely used anymore, think imagemaps
- The data name.x coordinate and value.y coordinate are passed to the script

```
<input type="image" src="CanadaMap.png"
name="CANADA" value="Province" >
```

# Input Element – file type

- Used to upload a file to the web server
- Useful when submitting a document created by another program such as a spreadsheet

`<input type="file" name = "reports.xls" />`



# Form Action Attribute

- The <form> tag uses an **action** attribute
    - Specifies the name of the script to run when the form's submit button is clicked
    - If no action attribute is provided, or the script does not exist, nothing will happen when submit is clicked
    - CGI scripts are usually organized into their own folder sometimes called script or cgi-bin
- ```
<form name="myform"  
  action="script/runform.php" >
```

Form Method Attribute

- **Method** attribute controls how your browser sends data to the web server
 - **Get** method is default. This sends the form data as a complete text appended to the URL. All the form data are shown in the URL separated by ampersands
`http://www.myscript.com/....cgi?name=John+Smith
&email=jsmith@mail.com`
 - **Post** method is preferable. This sends the form data in a separate data stream and is more flexible as some web servers will truncate the amount of data they receive from the GET method

Form Enctype Attribute

- Specifies the format of the data when it is transferred from the web page to the script
- Default value is either 'multipart/form-data' (for type="file") or 'application/x-www-form-urlencoded' (all other types)
- text/plain for HTML 5
- Tells the server what type of data it should be expecting from the browser

Organizing Form Elements

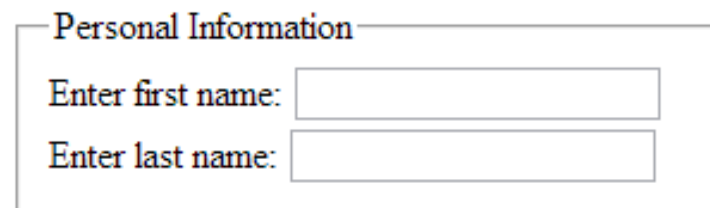
- You can group different form components into a related group called a *fieldset*
- Makes it easier for visitors to the page to understand the form and fill it out properly
- The legend tag is descriptive text

`<fieldset>`

`<legend>Personal Information</legend>`

... Rest of form elements

`</fieldset>`



Personal Information

Enter first name:

Enter last name:

Form Label

- Label tag enables CSS formatting to the form elements

```
<label for="fname">First Name: </label>
```

```
<input type="text" name="firstname"  
      size="15" id="fname">
```

Tab Order

- **Tabindex** attribute changes where pressing the tab key while in the form moves the focus
- Lower numbered tabindex elements get the focus first
- Tabindex values can range from 0 to 32767

```
<input type="text" name="firstname"  
  tabindex="3">
```

```
<input type="text" name="lastname"  
  tabindex="6">
```

Disabling Form Elements

- Any form element can be made initially unavailable to the user with `disabled="disabled"` in the form tag
- The form element will appear grayed out
- Disabled controls do not receive focus
- Disabled controls are skipped in tabbing navigation
- Can use Javascript code to re-enable or dynamically disable form elements

Accesskey Attribute

- The **accesskey** attribute works with the tags label, a, and caption for defining a case-insensitive "hotkey" for making an element have focus (declared obsolete for HTML 5)

```
<label accesskey="n">Name<input type="text" name="user"> </label>
```

- makes alt-n give focus to this form field

Readonly Attribute

- For textarea, text and password form elements
- Prevents user input into those fields having the **readonly** attribute defined
- Readonly elements can receive focus but cannot be modified by the user
- Readonly elements are included in tabbing navigation

```
<input type="text" name="dept"  
value="T2EH-QW" readonly="readonly" />
```

Title Attribute

- Allows for tool-tips pop-up
- Provides meaningful explanation or a short description of an element

```
<input type="text" name="dept"  
title="Department Category" />
```

Button tag

- User defined image buttons
- Define an **onclick** attribute and an image

```
<button onclick="script/mybutton.php">  
    
  Click this blue button.  
</button>
```

HTML5

- New form elements and attributes in HTML5
- Not supported by all browsers – features degrade nicely (e.g. they become text boxes)

`<input type= "tel">` for phone numbers

`<input type= "number">` numbers only

`<input type= "search">` search box (like text)

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

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

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

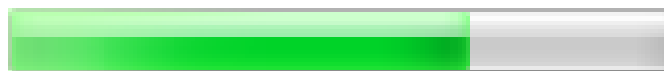
`<input type= "color">` for colour selection

HTML5

- New form elements
 - `<progress>` shows progression along a set of tasks
 - `<meter>` a scalar measurement within a range
 - `<datalist>` for making options within a dropdown
 - `<keygen>` a control for key-pair generation
 - `<output>` shows the output of a calculation

HTML5 - progress

- Show a progress bar to indicate progression along a sequence of tasks
- Use the **orient** property to specify horizontal (default) or vertical display
- Use JavaScript to dynamically update
- `<progress value="70" max="100">
70 %</progress>`



HTML5 - meter

- The meter element is similar to progress but meter shows the status of a value within a range (ideally between a min and max)

- `<meter min="200" max="500" value="350"></meter>`



value is between min and max 😊

- `<meter low="69" high="80" max="100" value="84"></meter>`



value is > high ☹

- `<meter low="69" high="80" min="50" max="100" value="60"></meter>`



value is < low ☹

HTML5 - datalist

- Use datalist to define a predefined set of valid entries to a text box (not supported by Safari)

```
<datalist id="soup" >
```

```
  <option value="chicken">
```

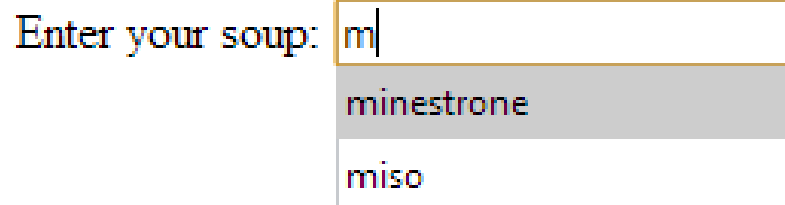
```
  <option value="minestrone">
```

```
  <option value="tomato">
```

```
  <option value="miso">
```

```
</datalist>
```

Enter your soup: <input name="sel" list="soup" >



A screenshot of a web form. On the left, the text "Enter your soup:" is displayed. To its right is a text input field. The input field contains the letter "m". A dropdown menu is open below the input field, showing two options: "minestrone" and "miso". The "minestrone" option is highlighted with a grey background.

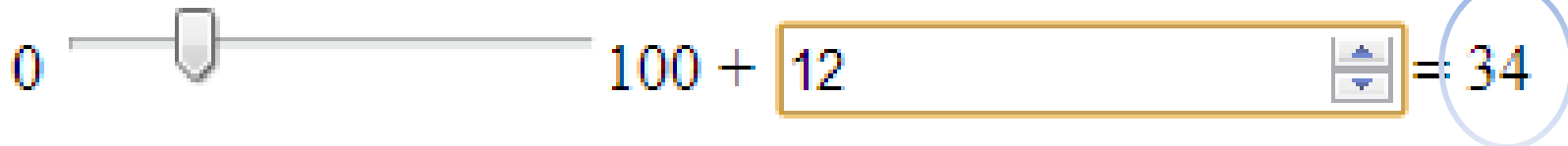
HTML5 - keygen

- The **keygen** form element creates a public key as part of HTML form to be used for web-based certificate management systems (encrypt the form information)
- Microsoft does not support this element
<http://support.microsoft.com/kb/190282>
- PHP script can access the local private key stored within the browser certificate area

HTML5 - output

- The **output** element represents the result of a calculation

- `<form oninput=`
 `"result.value=parseInt(a.value)` some DHTML here ☺
 `+parseInt(b.value)">`
 `0<input type="range" name="b" value="50">100`
 `+ <input type="number" name="a" value="10">`
 `= <output name="result"></output>`
 `</form>`



HTML5

- Attributes for the `<input>` element
 - `placeholder` – shows a light coloured hint of input
 - `autofocus` – focuses on element when page is loaded
 - `required` – if present, the element cannot be left blank by user when submit is clicked
 - `pattern` – the input data must match the provided regular expression
 - `autocomplete` – default is on, set it to off for elements you don't want autofilled (credit card)

HTML5

- The pattern attribute specifies a regular expression:
 - `\d` indicates a single number 0 to 9 same as `[0-9]`
 - `\d{3}` indicates exactly three digits
 - `\d{4,7}` indicates min 4 max 7 digits are needed
- Use `title` attribute to let users know input
- Special characters like hyphen, (, and) are enclosed by square brackets preceded by \
 - `[\-]` or `\[` or `\]`

HTML5

- `<input type= "text" placeholder= "first name" required>`
- `<input type= "number" value="1" min="1" max= "10">`
- `<input type= "tel" pattern="\d{7}" >`



HTML5

- `<input type="date">` works in Chrome, Opera but not yet in Firefox or IE



A screenshot of a date picker interface. At the top, a text box displays '10/20/2012' with a dropdown arrow. Below it is a calendar for October 2012. The calendar has navigation buttons for previous/next month and previous/next year. The days of the week are listed in the header. The date '20' is highlighted in blue. At the bottom are 'Today' and 'Clear' buttons.

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

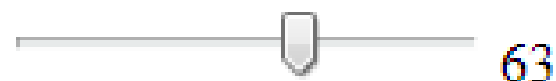
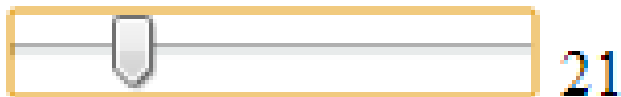
HTML5

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

What skill level?



- `<form oninput= "amount.value=rangeInput.value">
 <input type="range" id="rangeInput"
 name="rangeInput" min="0" max="100">
 <output name="amount" for="rangeInput">0
 </output>
</form>`



HTML5

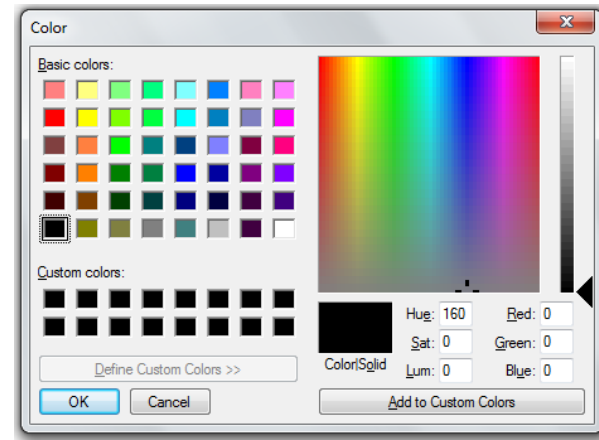
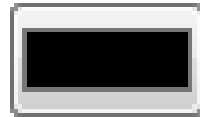
- Other date and time variations ...
implemented in the Opera browser only
- `<input type= "datetime">`
- `<input type= "month">`
- `<input type= "week">`
- `<input type= "time">`
- `<input type= "datetime-local">`

HTML5

- `<input type="email">`
 - If attribute `multiple` is specified, two or more email addresses can be entered separated by commas
 - User must enter an address having an `@` symbol

HTML5

- `<input type="color">`



- `<input type="range" min="-100" max="100" value="0" step="10" name="power" list="powers">`
`<datalist id="powers">`
 `<option value="0">` `<option value="-30"`
 `<option value="30">` `<option value="+50">`
`</datalist>`

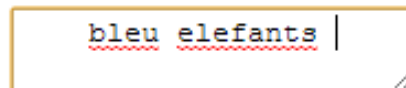


Spell Checking

- Text fields can be spell checked using HTML 5 `spellcheck` attribute
- Based on the `lang` attribute in `<html>`
`<html lang="en">` means use English spelling

`<textarea spellcheck="true"> </textarea>`

`<input type="text" spellcheck="true">`



CSS3

- pseudoclasses :invalid and :required can be styled in CSS

```
:invalid {  
    box-shadow: 0 0 5px rgba(255, 0, 0, .8);  
    border-color: red;  
}  
:required {  
    box-shadow: 0 0 5px rgba(0, 0, 255, .5);  
    border-color: blue;  
}
```


Form design guidelines

- Form field descriptions on the left, alignment
- Label input boxes with clear instructions
- Group related form items into a fieldset
- Control user's entries with radio buttons, checkboxes and selection lists whenever possible
- Let users know the correct format to enter date fields (yyyy/mm/dd)
- Use selection lists for many possible options
- Use radio buttons for five or fewer options

Form Anomalies

- Can a form prevent users hitting submit twice?
 - Can't -- unless you use cookies or use a unique ID in the form
- Some form elements can be easily styled with CSS (form, fieldset, label, input, output) – other form elements not so easily (date, select, option, progress, meter, legend)

HTML Forms

- Mozilla Developer guide HTML forms

<https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Forms>