Forms

- Collection of specific elements know as *controls*
  - Allow the user to enter information
  - Submit the data to a web server
- Controls are usually assigned names via an attribute

```html
<controlTag name="controlName" ...

<controlTag name="controlName" ...> ... </controlTag>
```

- Should name each control whose data will be submitted
- A name may not be entirely unique
  - May be applied to a series of elements representing different values for the same property
Forms

- Information is sent to the server as a *query string*
  - Series of name/value pairs
  - Automatically generated upon submission

```html
<form action="appURL" method="submitMethod">
  ...
</form>
```

- **appURL** is the URL of an application on a web server
  - The program that will manage the form data server-side

- **form** is a block element
  - All controls are inline

- **submitMethod** must either be *get* or *post*

---

get and post

- **get**
  - Default if no other method is specified
  - Submits query string appended to **appURL**
    - `appURL?name1=value1&...&nameN=valueN`
  - Can also pass information to servers without forms
  - May hit a limit on the length of the URL
    - Some servers may truncate the query string if it is too long

- **post**
  - Submits query string separately from **appURL**
    - The user doesn't see it
  - No limits on length
  - Generally seen as more secure
Input Controls

- Represents a wide range of control elements
  - Boxes, buttons, etc.
- The variety of the element depends on `inputType`
  - The same can be said for its attributes and properties
    - HTML and DOM
- Behaviour also depends on `inputType`
  - `tel` engages the keypad for input on phones
  - `email` automatically does some address validation

---

Text Box

- Creates a single-line text input area
  - The user can type in a small amount of data
- HTML attributes
  - `value`: initial text placed in the box
  - `size`: number of visible characters before scrolling
  - `maxlength`: number of characters that may be entered
- DOM Properties
  - `value`: the content of the text box
  - `disabled`: whether the user may enter text or not
Checkboxes

- Creates a checkbox with `description` as a label
  - The user may check or uncheck the box
- Value in query string is "on" if checked
  - *e.g.*, `...&name=on&...`
  - No value submitted if unchecked
- HTML attributes
  - `checked`: box is initially checked
- DOM Properties
  - `checked`: whether the box is checked or not

---

Radio Buttons

- Creates a radio button with `description` as a label
- Use the same `groupName` for a group of buttons
  - Only one may be selected at any time
  - Query string will contain single value for the group
    - *e.g.*, `...&groupName=buttonValue&...`
- HTML attributes
  - `value`: value submitted in query string (default is "on")
- DOM Properties
  - `checked`: whether the button is selected or not
Labels

- Makes a control's description "clickable"
  - Click to select a checkbox, radio button, ...
- Why use labels?
  - Adds functionality
  - Can apply tag-specific style with CSS
  - Helps screen readers better describe forms

Text Area

- Creates a multi-line text input area
  - The user can type in a large amount of data
- Size is specified
  - `numRows` lines high and `numCols` characters wide
  - Vertical scrollbar appears if more text is entered
- DOM Properties
  - `value`: the content of the text area
  - `disabled`: whether the user may enter text or not
  - `cols/rows`: the width/height of the text area (as String)
Menus and Lists

- Creates a menu with the given entries
  - Default is a drop-down menu

- Lists are created by specifying the size attribute
  - Sets number of entries shown at once
    - *e.g.*, `<select size="5" ...> ...
  - Permit multiple selection via attribute multiple
    - *e.g.*, `<select size="5" name="x[]" multiple>`

- Options are specified like list items
  - Use selected attribute to specify an initial selection
    - *e.g.*, `<option selected ...> ...
  - value is sent in the query string
submit and reset

```
<input type="submit" value="buttonText"/>
<input type="reset" value="buttonText"/>
```

- Submit button tells the form to process the data
  - Creates the query string
    - Taken from the names and values of the elements
  - Sends it to the server
    - Done via get or post
    - Uses provided `appURL` in form

- Reset button clears the contents of the form
  - Restores all controls to default values

Some Useful Things

- Styling
  - Tables were often used to organize forms
  - A better approach is to use CSS
    - More flexible
    - Gives finer control

```
<input type="hidden" name="..." value="..."/>
```

- A control that is never displayed
- Useful for sending additional data to the server

- HTML5 input attribute required
  - Form will not submit with an empty value in that field
### Some Useful Things

```html
<fieldset>
  <legend>caption</legend>
  controls
</fieldset>
```

- Groups controls with a similar purpose or theme
  - Draws a box around them
  - Labels the box with caption
- Can be styled using CSS

- `<input type="button" value="buttonText" ...>`
  - Creates a button labelled `buttonText`
  - Use to perform some action
    - Can link to a JavaScript function via the `click` event

---

### Form Events

<table>
<thead>
<tr>
<th>Name</th>
<th>The User Must ...</th>
<th>Applicable Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>focus / blur</td>
<td>Apply/lose focus (e.g., click in/out of a text field)</td>
<td>Many HTML elements</td>
</tr>
<tr>
<td>change</td>
<td>apply focus to a control, alter the value and remove focus</td>
<td>Most <code>&lt;input&gt;</code>s, <code>&lt;textarea&gt;</code> and <code>&lt;select&gt;</code></td>
</tr>
<tr>
<td>reset</td>
<td>reset the form</td>
<td><code>&lt;form&gt;</code></td>
</tr>
<tr>
<td>submit</td>
<td>submit the form</td>
<td><code>&lt;form&gt;</code></td>
</tr>
</tbody>
</table>

- Can also apply mouse and keyboard events to many controls
Three Phases of Validation

- As the user types or selects data
  - Manage events such as `onkeydown`, `onclick`, etc.
  - Test current value against desired format

- As the user finishes altering existing data
  - Manage the change event
  - Test current value against desired format

- As the user submits the form
  - Manage the `submit` event
  - Test all values against desired formats

The first two phases allow you to warn the user
- Alert him or her to errors before attempting to submit
  - Try to be obvious without hindering functionality
- Strategies
  - Dynamically alter a status message in the document
  - Dynamically alter the control's style

The last phase prevents submission of invalid data
- Return `true` or `false` in a handler for `submit`
- Can use `onsubmit="return handler()"` in the HTML

HTML5 also includes the concept of *constraint validation*
- Not currently well-supported in Safari