• A form is the usual way to get information from a browser to a server
• XHTML has tags to create a collection of objects that implement this information gathering
  – The objects are called *widgets* (e.g., radio buttons and checkboxes)
• When the Submit button of a form is clicked, the form’s values are sent to the server
• All of the widgets, or components of a form are defined in the content of a `<form>` tag
  – The only required attribute of `<form>` is `action`, which specifies the URL of the application that is to be called when the Submit button is clicked
  – `action = "http://www.cs.ucp.edu/cgi-bin/survey.pl"
» If the form has no action, the value of `action` is the empty string
Forms (continued)

• The method attribute of `<form>` specifies one of the two possible techniques of transferring the form data to the server, `get` and `post`
  – you might recognise these in the topic on `protocols`..

• Widgets
  – Many are created with the `<input>` tag
    » The type attribute of `<input>` specifies the kind of widget being created
      • Text
      • Text Areas
      • Checkboxes
      • Radio buttons
      • Menus
      • Reset and submit buttons
Form actions

Forms have an *action* attribute which is the script that is executed when the form is submitted. Named fields are *posted* to the script.

```html
<!DOCTYPE html>
<html>
<body>

<form action="/action_page.php">
  <fieldset>
    <legend>Personal information:</legend>
    First name:<br>
    <input type="text" name="firstname" value="Mickey">
    <br>
    Last name:<br>
    <input type="text" name="lastname" value="Mouse">
    <br>
    <input type="submit" value="Submit">
  </fieldset>
</form>

</body>
</html>
```
Forms (continued)

- **Text**
  - Creates a horizontal box for text input

- **Checkboxes** - to collect multiple choice input
  - Every checkbox requires a `value` attribute, which is the widget’s value in the form data when the checkbox is ‘checked’

- **Radio Buttons** - collections of checkboxes in which only one button can be ‘checked’ at a time
  - Every button in a radio button group MUST have the same name

- **Menus** - created with `<select>` tags
  - There are two kinds of menus, those that behave like checkboxes and those that behave like radio buttons (the default)
    - Menus that behave like checkboxes are specified by including the `multiple` attribute, which must be set to "multiple"

- **Text areas** - created with `<textarea>`
  - Usually include the `rows` and `cols` attributes to specify the size of the text area
Forms (continued)

- Reset and Submit buttons
  - Both are created with `<input>`
- `<input type = "reset"  value = "Reset Form">`
- `<input type = "submit"  value = "Submit Form">`

- Submit has two actions:
  1. Encode the data of the form
  2. Request that the server execute the server-resident program specified as the value of the `action` attribute of `<form>`
     - A Submit button is required in every form
HTML5 Forms

• New Attributes
  – placeholder
  – autocomplete (on, off)
  – required
  – autofocus

• New form controls
  – datalist

• New input types
  – search
  – Contacts (email, url, tel)
  – Slider: range
  – Spinner: number
  – Data/time: date, datetime, datetime-local, month, week

• Simplicity
• Consistency
• Validation
• Frequent design patterns
HTML5 Forms - Summary

type="text"  type="submit"  <textarea>
type="radio"  type="reset"  <fieldset>
type="check"  type="file"  <select> <option>
type="email"  
type="tel"  
type="url"  
type="date"  
type="search"  
type="range"  
type="number"  
type="color"  
type="date | datetime | datetime-local | week | month"  
<output>
The Mobile Safari changes on-screen keyboard according to different contact input types:

- Email address
- Website
- Telephone
- Email address
- Website
- Telephone
- Email address
- Website
- Telephone
One of the most popular JavaScript widgets is the calendar picker. These calendar widgets all do the same thing, but you’ll find that they’re implemented slightly differently on each site. A native calendar widget would smooth away the inconsistencies and reduce cognitive load during the date-picking process.

HTML5 introduces a raft of input types specifically for dates and times:

- **date** is for a year, month, and day.
- **datetime** is for a year, month, and day in combination with hours, minutes, and seconds and time zone information.
- **datetime-local** is the same but without the time zone information.
- **time** is for hours, minutes, and seconds.
- **month** is for a year and a month but without a day.
All of these input types will record timestamps with some subset of the standardized format YYYY-MM-DDThh:mm:ss.Z (Y is year, M is month, D is day, h is hour, m is minute, s is second, and Z is timezone). Take, for example, the date and time at which World War One ended, 11:11am on November 11th, 1918:

- date: 1918-11-11
- datetime: 1918-11-11T11:11:00+01
- datetimelocal: 1918-11-11T11:11:00
- time: 11:11:00
- month: 1918-11

There is no year input type, although there is a week input type that takes a number between 1 and 53 in combination with a year.
• Native validation without scripting.
• HTML5 browsers support basic validation on email, url and tel input types.
• HTML5 has made it even more friendly for web authoring
  – The pattern attribute that allows you to use regular expression to specify required format
  – For example:
    
    ```html
    <input id="phone" name="phone" pattern="\d{8}" type="tel">
    ```
The form output element

- Represent the results of some calculation

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

```html
<!DOCTYPE html>
<html>
<body>
<p>Write something in the text field to trigger a function.</p>
<input type="text" id="myInput" oninput="myFunction()">
<p id="demo"></p>
<script>
  function myFunction() {
    var x = document.getElementById("myInput").value;
    document.getElementById("demo").innerHTML = "You wrote: " + x;
  }
</script>
</body>
</html>
```
The `<time>` element

- Encode time and date in formats that are both Machine and Human readable
  
  `<time datetime=2011-8-12> 12 August 2011</time>`
  `<time datetime=2011-8-12> 12 <sup>th</sup> August Last Year</time>`
  `<time datetime=2012-8-12>UWA Expo 2012</time>`
  `<time datetime=2012-8-12T14:00Z>2PM on UWA Expo 2012</time>`
  `<time datetime=20:00> 8PM</time>`

- Previously, you could only mark up precise dates, which could be a problem (e.g. historians)
  
  - 13 November 1905 could be expressed in HTML but not November 1905
    `<time datetime="1905-11-13">`

- Now, "fuzzy dates" are possible:

  `<time datetime="1905"> means the year 1905
  `<time datetime="1905-11"> means November 1905
  `<time datetime="11-13"> means 13 November (any year)
  `<time datetime="1905-W21"> means week 21 of 1905`
The `<time>` element - Durations

- In HTML5 `<time>`, you can represent durations, with the prefix "P" for "period".
  - The `datetime` attribute value: "D" for days, "H" for hours, "M" for minutes and "S" for seconds.
- You can separate them with spaces (but you don't have to).
  - `<time datetime="P4D">` is a duration of 4 days, same as `<time datetime="P 4 D">`
- Using a "T" after the "P" marker allows you to be more precise:
  - `<time datetime="PT23H 9M 2.345S">` is a duration of 23 hours, 9 minutes and 2.345 seconds.
- The `pubdate` attribute is a boolean to indicate when a page is published

```html
<section>
  <article>
    <header>
      <h1>Seminar: What is ARIA?</h1>
      <p><time datetime="2012-08-12T11:00">12 August 2012 11:00am</time></p>
    </header>
    <p>This seminar is about accessibility.</p>
    <footer>
      Published at: <time datetime="2012-08-08T20:00" pubdate>8 August 2012 8:00pm</time>
    </footer>
  </article>
</section>

http://introducinghtml5.com/errata/ch02.html
The audio Element

• Prior to HTML5, a plug-in was required to play sound while a document was being displayed
• Audio encoding algorithms are called audio codecs – e.g., MP3, Vorbis
• Coded audio data is stored in containers—e.g., Ogg, MP3, and Wav (file name extension indicates the container, not the audio code)
  – Vorbis code is stored in Ogg containers
  – MP3 code is stored in MP3 container
  – Wav code is stored in Wav containers
The audio Element (continued)

• General syntax:
  <audio attributes>
    <source src = "filename_1" >
    ...
    <source src = "filename_n" >
  Your browser does not support the audio element
</audio>

• The controls attribute, which is set to controls", creates a start/stop button, a clock, a progress slider, total time of the file, and a volume slider

```html
<!DOCTYPE html>
<!– audio.html Test the audio element -->
<html lang = "en">
  <head>
    <title> Test audio element </title>
    <meta charset = "utf-8" />
  </head>
  <body>
    This is a test of the audio element
    <audio controls = "controls">
      <source src = "nineoneone.ogg" />
      <source src = "nineoneone.wav" />
      <source src = "nineoneone.mp3" />
    Your browser does not support the audio element
    </audio>
  </body>
</html>
```
The video Element

• Prior to HTML5, there was no standard way to play video clips while a document was being displayed
  
  <video width = "600" height = "500" autoplay = "autoplay"
         controls = "controls" preload = "preload">
    <source src = ""/>
    <source src = ""/>
  ...
  Your browser does not support the video element
  </video>

• The width and height attributes set the screen size

• The autoplay attribute, set to "autoplay", specifies that the video should play as soon as it is ready

• The preload attribute, set to "preload", specifies that the video should be loaded as soon as the document is loaded
<!DOCTYPE html>
<!-- testvideo.html test the video element -->
<html lang = "en">
<head>
    <meta charset = "UTF-8" />
    <title> test video element </title>
</head>
<body>
    This is a test of the video element.....
    <video width = "600" height = "500" autoplay = "autoplay"
        controls = "controls" preload = "preload">
        <source src = "NorskTippingKebab.mp4" />
        <source src = "NorskTippingKebab.ogv" />
        <source src = "NorskTippingKebab.webm" />
        Your browser does not support the video element
    </video>
</body>
</html>
The video Element (continued)

• Video codecs:
  – H.264 (MPEG-4 AVC) – can be stored in an MPEG-4 container
  – Theora – can be stored in any container
  – VP8—can be stored in WebM container

• Video codecs are stored in containers
  – Common containers MPEG-4 (.mp4), Flash Video (.flv), Ogg (.ogg), WebM (.webm) and Audio Video Interleave (.avi).
  – Convert video format free online: http://video.online-convert.com/
The HTML <canvas> element is used to draw graphics on the fly, via scripting (usually javascript)

The Canvas element is only a container for graphics and you use a script to draw the graphics.

Canvas has several methods for drawing paths boxes, circles, text and adding images.

- [http://www.blobsallad.se/](http://www.blobsallad.se/)
The coolest part about HTML5 is definitely its offline capabilities. Programs like Thunderbird and Outlook (and now GMail to an extent) let you browse through your old data while staying offline. With HTML5, you’ll have this same functionality, but in the browser. This is the first serious step towards bridging the gap between the desktop and the Web, and opens all sorts of doors for the future of Web apps.

The W3C has taken the best parts from the various Web technologies and rolled them into, what is being dubbed the most powerful markup language to date.

Some other features of the HTML5 APIs

- **Drag & Drop**
  The drag and drop API defines an event-based drag and drop system. However, it never defines what "drag and drop" is. This API requires JavaScript to fully work as normal think drag and drop functionality.

- **Geolocation**
  Geolocation is a very cool API available within HTML5. Its object can be used to programmatically determine location information through a device’s user agent (hint: mobile devices).
Useful Cheat Sheet

- HTML5 Quick Reference
- HTML5 and HTML4 Comparison at a Glance
  http://websitesetup.org/HTML5-cheat-sheet.pdf
  http://www.cheat-sheets.org/saved-copy/HTML5_Canvas_Cheat_Sheet.pdf