FRESNO STATE
Discovery. Diversity. Distinction.

Web Programming (CSci 130)

Department of Computer Science
College of Science and Mathematics
California State University Fresno
H. Cecotti

Learning outcomes

- In this course,
  - "You learn how to create bottle, how to fill the bottles, not how to make the liquid (data)"
- In this class,
  - You will learn how to create **forms** with HTML5
    - Graphical User Interface to interact with the page and the website
    - This part will be used until the end of the semester

- **Remark**:
  - You need to work on the lab sessions seriously, because we **accumulate** elements over time
  - We will **reuse** a lot forms in the next classes, when the content of the form is dynamically filled/retrieved via Javascript

Remarks

- To check the different links on Canvas regularly:
  - Link to the example of the book HTML5 and CSS3.
- Next lab:
  - Some HTML5+CSS3 (animation in CSS3)
  - HTML5 forms
- HTML5+CSS3
  - Be rigorous
    - There are shortcuts …it's not because it looks like the same at the end (on the screen) that it is the same
    - → think about the structure, how documents can be displayed differently on different devices
  - Stick to the **structure**
    - No extra meaningless tags like some tape or glue to fix something broken
      - There will be side effects

Introduction

- HTML5+CSS3:
  - Presentation of the document to the users
    - Users = Customers = $$$
- Forms
  - Interaction with the user/reader/customer of the webpage
  - 2 parts
    - Fields to fill, labels, buttons
    - Processing script that takes this information and process it
      - → Happening in Javascript (next week)
- CSS3
  - For the presentation of the different controls
- Web server: to process the responses from the forms
  - To store information in a database, to send an email
  - … what we will see with PHP

Warning

- Forms
  - ➢ You allow the user to enter inputs
    - Example: inputs to a function
    - Users often try to break everything – Never trust the users
  - ➢ → Inputs should be verified (Is it a number? Is it a date? Is it a string?)
  - ➢ The verification can happen at different stages
    - **Client**
      - Solution 1:
        - Force the form to only allow a type of input (solutions through HTML5 + CSS3)
      - Solution 2:
        - Get whatever string from the form -> Warning: all inputs should not be textboxes
        - Verify the content on the client side (e.g., JS function , regular expressions)
    - **Server**
      - Solution 3:
        - Get whatever string from the client
        - Verify the content on the server side (e.g., PHP function)
        - → Don't trust the client anyways (need to deal with security issues)

Event-driven programming

- A programming paradigm
  - The flow of the program is determined by :
    - **Events** (user actions (mouse clicks, key presses))
    - Sensor outputs
    - Messages from other programs/threads
      - Client/Server communication
- The main paradigm in
  - Graphical user interfaces (GUI)
    - Desktop and Mobile
  - Web applications → JavaScript web applications
    - Based on performing certain actions in response to user input.

Event-driven programming

- In an event-driven application:
  - **A main loop** that listens for events (e.g., click on the mouse, keypress)
    1. Create a window
    2. Main loop waiting actions from the user

    → You don't have to create this loop or worry about it ☺
  - Triggers a callback function when one of those events is detected
  - → Assign functions to the events: What happens if…
    - A user clicks on a button
    - A user clicks on a link
    - A user changes the content of a textbox
    - A user closes a window
  - → Need of:
    - Proper initializations of the different elements on the screen
    - Perfect understanding of the flow of actions
      - When something is validated … what happens → OK/Cancel buttons

Event-driven programming

- Common problems
  - Do not:
    - Update the value of a variable each time it changes
    - Retrieve data at each modification
      - It can happen if you run some text verification, search words, help for words completion…
  - Do:
    - Gather the information from multiple controls and then update the related variables
    - Retrieve data once it is entered
    - Think step by step
      - 1. User fills the form
      - 2. Gather the information from the form

- Thinking ahead with what will come with JavaScript
  - Events to retrieve information once it is typed, once it is entered

Types of controls

- The same controls that you find to create any GUI (desktop, mobile,…)
  - Textboxes
  - Special password boxes
  - Radio buttons (A or B or C)
  - Checkboxes (A and/or B and/or C)
  - Drop-down menus / Combo box
  - Text area…
- Each element has a name = a **label**
  - To identify the data (where it is from on the page)

Creation of forms

- **3 Parts**
  1. The “form” element
     - URL of the script to process the form and its method
       - POST or GET
  2. The form elements
     - Checkboxes, combo box, radio buttons…
  3. **Submit** button
     - Send data to the script listening on the server
- In <form> … </form>
  - ➢Method
    - Save/delete/add data in a database: method=“post”
  - ➢Action
    - Action=“dothis.php” what will happen when the form is submitted.

Server and Client side

- What to do with the content of the form?
  - **Client side** (JavaScript)
    - Work inside the browser
      - → Using the resources of the User's computer
    - To do tasks without interacting with the server
    - To verify the integrity of the data entered in the form before being submitted to the server
  - **Server side** (PHP)
    - Runs on the computer that serves you the Web Pages
      - It is the web server
        - → Using the resources of the Server
      - Scripts must be uploaded on the server to work
      - Interpreter installed on the server to decode the script

Textbox

- One line of free form text
  - Where the users can type text
    - Names, addresses,…
  - Special attributes
    - Autofocus, required, placeholder, maxlength, pattern
  - **<input type="text"**
  - **name**="label"
    - Text to identify the input data to the server in POST/GET operations
  - **id**="label"
    - Text to identify the element to its matching label element
  - **value**="default" default text to be printed in the input field
  - Required="required" to force the field to be filled before submission
  - Placeholder="hinttext" , to give instructions to the user about what to write in the field

Textbox

- Other parameters:
  - ➢Size of the box: size="n"
    - Size of the box in numbers of characters
  - ➢Maxlength="n" : n is the maximum number of characters in the box
    - Think of how Canvas is designed for the Quiz, and it is not managed ☹
    - → Expected string → Set the maxlength
      - Example: zipcode, state, address, …
  - ➢… after all the parameters, finish the textbox with />

Password boxes

- Password box vs. text box
  - ➢Text hidden by bullets or asterisks
    - **Warning**: Hidden to you but not encrypted when sent on the server !
      - Example: When the user is typing in a public place, someone looking behind the shoulder
  - ➢Type="password"
  - ➢Same main parameters as a textbox
    - Name, id, required, size, and maxlength

- **Remark**
  - ➢You can use a textbox and code the behavior of a password in JavaScript
    - It would take more time, and probably be not as efficient

Email, telephone, and URL boxes

- **New to HTML5**
  - Similar to textboxes for the look
    - Features related to validation and inputting content
- <label for="idlabel">Email</label>
  - To connect elements between each other:
    - Label + Textbox: First Name: _____
- Attribute type of the HTML5 tag input:
  - Email box
    - <input type="email" />
  - URL box
    - <input type="url" />
  - Telephone box
    - <input type="tel" />

Regular expressions

- Regular Expressions Patterns (syntax is common to other systems)
  - [abc] Find any of the characters between the brackets
  - [0-9] Find any of the digits between the brackets
  - (x|y) Find any of the alternatives separated with |
  - \d Find a digit
  - \s Find a whitespace character
  - \b Find a match at the beginning or at the end of a word
  - \uxxxx Find the Unicode character specified by the hexadecimal number xxxx
  - n+ Matches any string that contains at least one n
  - n* Matches any string that contains zero or more occurrences of n
  - n? Matches any string that contains zero or one occurrences of n

Example

- For the phone
  - Pattern field
    - Pattern="\d{3}-\d{3}-\d{4}" by using regular expressions

```
<form method="post" action="page01.html">
<fieldset>
<h2 class="account">Account</h2>
<ul>


<li>
<label for="email">Email:</label>
<input type="email" id="email" name="email" class="large" />
</li>
<li>
<label for="web_site">Web:</label>
<input type="url" id="web_site" name="web_site" class="large" />
<p class="instructions">Have a homepage, Please put the address here.</p>
</li>
<li>
<label for="phone">Phone:</label>
<input type="tel" id="phone" name="phone" placeholder="xxx-xxx-xxxx" class="large" pattern="\d{3}-\d{3}-\d{4}" />
</li>

</form>
```

Example

- For the phone
  - Valid/Invalid

```
</style>
<body>
<fieldset id="mainfield">

    <legend>Contact information</legend>
    <label for="phone">Phone:</label>
    <input type="tel" id="phone" name="phone"
           placeholder="123-456-7890"
           pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}"
           required />
    <span class="validity"></span>
</fieldset>
</body>
</html>
```

```
legend {
    background-color: #111;
    color: #fff;
    padding: 3px 6px;
}

.output {
    font: 1rem 'Fira Sans', sans-serif;
}

input {
    margin: 1rem 0;
    width: 60%;
}

label {
    display: inline-block;
    font-size: .8rem;
    width: 20%;
}

input:invalid + span:after {
    content: '✖ (bad number)';
    color: #f00;
    padding-left: 5px;
}

input:valid + span:after {
    content: '✓ (valid number)';
    color: #0f0;
    padding-left: 5px;
}

#mainfield {
    width: 50vw;
    background-color: #eee
    }
```

Labeling Form Parts

- To mark up labels
  - Link between labels and textboxes
  - "for" name of the id in the associated input block
    - **an explicit relationship**
- <fieldset> tag
  - to group related elements in a form.
    - It draws a box around the related elements

```
<fieldset>
<h2 class="account">Account</h2>
<ul>
<li>
<label for="first_name">First Name:</label>
<input type="text" id="first_name" name="first_name" class="large" />
</li>
<li>
<label for="last_name">Last Name: </label>
<input type="text" id="last_name" name="last_name" class="large"/>
</li>
</ul>
</fieldset>
```

Radio buttons

- Selection of elements
  - ➢OR (A or B)
    - A single answer per set
  - ➢<input type="radio" />
  - ➢Main parameters
    - **name** : data sent to the script
    - **id** : the unique radio button id

```
<fieldset class="radios">
<ul>
<li>
<input type="radio" id="gender_male" name="gender" value="male" />
<label for="gender_male">Male</label>
</li>
<li>
<input type="radio" id="gender_female" name="gender" value="female" />
<label for="gender_female">Female</label>
</li>
</ul>
</fieldset>
```

Select boxes

- Boxes with different choices (list of choices)
  - Dropdown lists
- <select list of parameters and values> … </select>
  - List of <option list of parameters and values> …. </option>
- Main parameters
  - **name**, **id**
  - Size="n" height in lines
  - Multiple="multiple" to select more than one option with the ctrl key

Select boxes

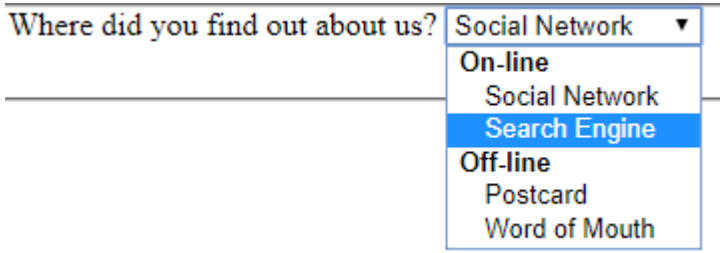
- Option
  - Selected="selected" to specify that the option is selected by default
- Examples:

```
<label for="state">State:</label>
<select id="state" name="state">
    <option value="AL">Alabama</option>
    <option value="AK">Alaska</option>
    <option value="AZ">Arizona</option>
    ...
    <option value="CA">California</option>
</select>
```

```
<select>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="vw">VW</option>
  <option value="audi" selected>Audi</option>
</select>
```

Submenus

- Tag:
  - optgroup
- Example:




```
<label for="referral">Where did you find out about us?</label>
<select id="referral" name="referral">
    <optgroup label="On-line">
        <option value="social_network">Social Network</option>
        <option value="search_engine">Search Engine</option>
    </optgroup>
    <optgroup label="Off-line">
        <option value="postcard">Postcard </option>
        <option value="word_of_mouth">Word of Mouth</option>
    </optgroup>
</select>
```

Checkboxes

- Selection of items independently
  - ➢You can select or not each checkbox
- <input type=“checkbox”
- Main parameters
  - ➢Name, value
  - ➢Checked=“checked” : to check the box

```
<ul class="checkboxes">
<li>
<input type="checkbox" id="email_ok_msg_from_users" name="email_signup" value="user_emails" />
<label for="email_ok_msg_from_users">You can email me with messages from other users.</label>
</li>
<li>
<input type="checkbox" id="email_ok_occasional_updates" name="email_signup" value="occasional_updates" />
<label for="email_ok_occasional_updates">You can email me about our other products.</label>
</li>
</ul>
```

Text area

- For comments, blogs, answers, on a webpage
- When you have to write **more than one line**
  - Comments, questions…
    - Body of an email
  - <textarea …> …. </textarea>
  - Main parameters
    - Name, maxlength="n" (max number of characters)
    - Rows="n", Cols="n", specify the size of the text area in number of characters
  - If you write more than what the box can contain
    - Scrollbars come automatically

```
<label for="bodyemail">Body of email:</label>
<textarea id="bodyemail" name="body_email" rows="12" cols="50" class="large"></textarea>
```

Upload files

- To allow users of the webpage to upload files…
  - <form method="post" enctype="multipart/form-data"
    - Action="file.html" → URL of the script to process the incoming files
  - Main parameters
    - Name, id, size="n" (width of the field)

```
<form method="post" action="showform.php" enctype="multipart/form-data">
<label for="picture">Picture:</label>
<input type="file" id="picture" name="picture" />
<p class="instructions">Maximum size of 800k. (JPG, GIF, PNG).</p>
</form>
```

Submit button

- <input type=“submit”
- Main parameters
  - Value=“submit message”
- Submit button with an image
  - <button type=“submit”> …. Code for the image … </button>
  - Alt=“alternate text” (if no image)

```
<input type="submit" class="create_profile" value="Create My Personal Account">
```

Disable/Enable elements

- Need of JavaScript
  - ➢To be presented in next classes…
  - ➢Through the **DOM** (Document Object Model)
    - Document Model to access particular elements
      - Object with properties

- In this class,
  - ➢We have seen how to create forms by typing the code in HTML5
  - ➢We will see how to arrive to the **same** results via the DOM, so it can be done automatically

Conclusion

- Forms: to enter data/to contain/ to visualize data
  - ➢ Description of the different elements: HTML5
  - ➢ Presentation of the form elements: CSS3
  - ➢ Intermediate between the user and the website
    - To **exchange** information
    - Different levels of data processing
      - Client / Server
- **Remarks**
  - ➢ You have to decide "in the team" of who is responsible of the format of data that transfers between the database and the web page
    - Verify the **integrity** of the strings, numbers, …. Through:
      - HTML5 (you constraint the format of the input)
      - Javascript (get a string, check if it is what you need) → it should not be something that can be interpreted
      - PHP (get a string, check if it is what you need) → it should not be something that can be interpreted

Questions?

- Reading
  - See HTML + CSS code (available on Canvas)
    - class_html5_css3_form_a.html
    - class_html5_css3_form_b.html
    - class_html5_css3_form_c.html

- Next session:
  - Introduction to JavaScript