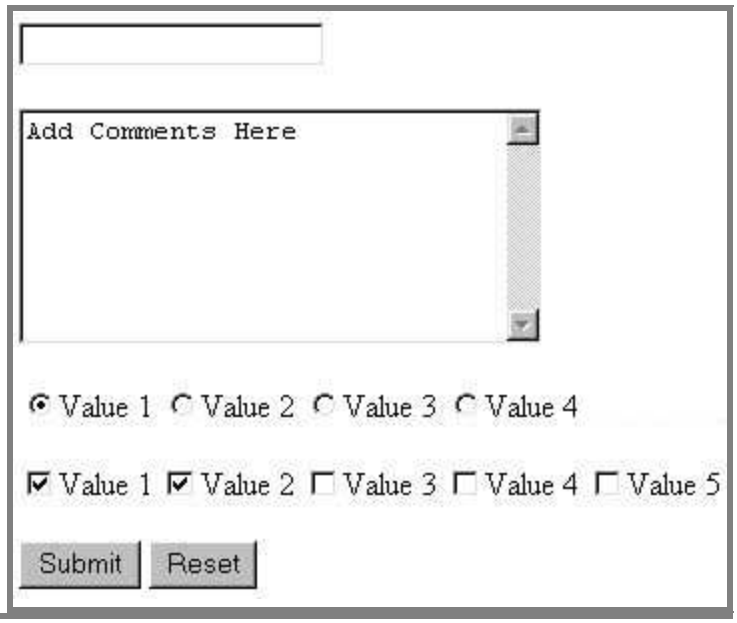
HTML User Interface Controls

CSE 190 M (Web Programming), Spring 2007
University of Washington

Reading: Sebesta Ch. 5 sections 5.1 - 5.7.2,
Ch. 2 sections 2.9 - 2.9.4

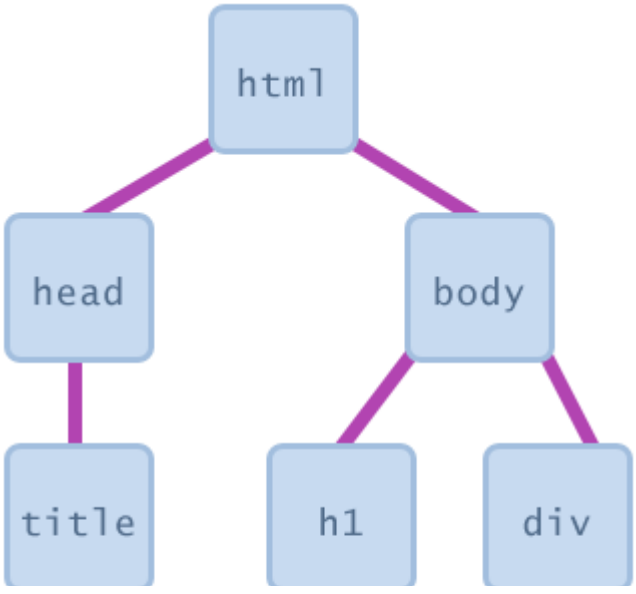
Interactive HTML user interfaces

- in this section, we'll learn how to make user interface controls (buttons, checkboxes, text fields, etc.) in HTML
- controls are often used in HTML forms (seen later)
- Javascript is integral to interactivity aspect of controls (event handlers)



Document Object Model (DOM)

- a representation of the current web page as a set of Javascript objects
- allows you to view/modify page elements in script code
- DOM tutorial



Global DOM objects

- `window` : the browser window
- `navigator` : info about the web browser you're using
- `screen` : info about the screen area occupied by the browser
- `history` : list of pages the user has visited
- `document` : current HTML page

Recall: event handlers

```
<h2 onclick="myFunction();">Click me!</h2>
```

Click me!

- HTML elements have special attributes called events
- Javascript functions can be set as event handlers
  - when you interact with the element, the function will execute
  - an example of event-driven programming
- event HTML attributes:
  - `onabort`, `onblur`, `onchange`, `onclick`, `ondblclick`, `onerror`, `onfocus`, `onkeydown`, `onkeypress`, `onkeyup`, `onload`, `onmousedown`, `onmousemove`, `onmouseout`, `onmouseover`, `onmouseup`, `onreset`, `onresize`, `onselect`, `onsubmit`, `onunload`

`document` object and `getElementById`

```
<h2 onclick="makeRed();">Sell</h2>
<p id="announce">Get it while it's hot!</p>

function makeRed() {
    var para = document.getElementById("announce");
    para.style.color = "red";
}
```

Sell

Get it while it's hot!

- `document` object's `getElementById` method returns an object representing the HTML element with the given `id` attribute (`null` if not found)
- DOM objects for all HTML elements contain the following properties:
  - `className`, `id`, `style`, `title`

DOM `style` property

```
function enlarge(id) {
    var element = document.getElementById(id);
    element.style.fontSize = "42pt";
}
```

Click me and make me big!

- `style` property represents the combined styles that apply to this element
- contains identical properties to the style properties set in CSS, except that names are changed from hyphenated to capitalized
    - examples: `backgroundColor`, `borderLeftWidth`, `fontFamily`

Buttons: `<button>`

```
<button onclick="alert('Hello!');">Click me!</button>
```

Click me!

- button's text appears inside `button` tag
- JS `onclick` event handler specifies button's behavior

The DOM `innerHTML` property

```
<button id="b1" onclick="myFunction('I did it!');">Click me!</button>
<p id="target">This text will be replaced.</p>

function myFunction(text) {
    var p = document.getElementById("target");
    p.innerHTML = text;
}
```

Click me!

This text will be replaced.

- `innerHTML` refers to the HTML text inside of an element:
  `<p>this is the innerHTML of the p tag</p>`
- event handler can modify the `innerHTML` of another element

Another <button> example

```
<button id="b1" onclick="addText();">Click me!</button>

function addText() {
    var button = document.getElementById("b1");
    button.innerHTML += " narf";
}
```

Click me!

- also acceptable in this case:
  `<button onclick="this.innerHTML += ' narf';">Click me!</button>`

Text boxes: <textarea> (DOM)

```
<textarea rows="4" cols="20">
Type your comments here.
</textarea>
```

Type your comments here.

- initial text placed inside `textarea` tag (optional)
- optional `readonly` attribute means text cannot be modified
- DOM properties: `disabled`, `readOnly`, `value`
  - NOTE: get/set area's text using `value`, NOT `innerHTML`

Practice problem: Shuffle

- Write the HTML and Javascript code to shuffle the lines of text within a text area whenever a Shuffle button is clicked.
- shuffling algorithms

Drop-down list: <select> (DOM), <option> (DOM)

```
<select>
<option>Jerry</option>
<option>George</option>
<option>Kramer</option>
<option>Elaine</option>
</select>
```

- `option` element represents each choice
- `select` optional attributes: `disabled`, `multiple`, `size`
- attach `onchange` handler to `select` to cause behavior on each selection
    - `<select onchange="alert('You chose ' + this.value);">`

Using <select> for lists

```
<select multiple="multiple" size="3">
<option value="Jerry">Jerry</option>
<option value="George">George</option>
<option value="Kramer">Kramer</option>
<option value="Elaine">Elaine</option>
<option value="Newman">Newman</option>
<option value="Susan">Susan</option>
</select>
```

- DOM properties: `disabled`, `length`, `multiple`, `name`, `selectedIndex`, `size`, `value` (selected item text)
- DOM methods: `add(option, index)`, `remove(index)`

Option groups: <optgroup>

```
<select>
<optgroup label="Major Characters">
<option value="Jerry">Jerry</option>
<option value="George">George</option>
<option value="Kramer">Kramer</option>
<option value="Elaine">Elaine</option>
</optgroup>
<optgroup label="Minor Characters">
<option value="Newman">Newman</option>
<option value="Susan">Susan</option>
</optgroup>
</select>
```

Jerry

- what should we do if we don't like the bold italic?

Input fields: <input>

```
<input type="text" /><br />
<input type="password" size="12" />
```

- creates many different types of input controls, depending on its `type` attribute
- always empty; contains attributes only
- attributes: `accept`, `alt`, `disabled`, `maxlength`, `name`, `readonly`, `size`, `src`, `type`, `value`
- DOM properties for `type="text"` and `type="password"`: `disabled`, `maxLength`, `readOnly`, `size`, `value` (text in field)

Radio buttons (DOM)

```
<input type="radio" name="creditcards" id="visa" />
    <label for="visa">Visa</label>
<input type="radio" name="creditcards" id="mastercard" />
    <label for="mastercard">MasterCard</label>
<input type="radio" name="creditcards" id="amex" />
    <label for="amex">American Express</label>
```

Visa MasterCard American Express

- grouped by (required) `name` attribute
- button's text is a `label` element with `for` attribute set to button's `id`
- DOM properties: `checked`, `defaultChecked`, `disabled`

Checkboxes (DOM)

```
<input type="checkbox" name="toppings" value="lettuce" id="lettuce" />
    <label for="lettuce">Lettuce</label>
<input type="checkbox" name="toppings" value="tomato" id="tomato" />
    <label for="tomato">Tomato</label>
<input type="checkbox" name="toppings" value="pickles" id="pickles" />
    <label for="pickles">Pickles</label>
```

☐ Lettuce ☐ Tomato ☐ Pickles

- `name` attribute is required
- use `checked="checked"` attribute in HTML to initially check the box
- DOM properties: `checked` (Boolean), `defaultChecked`, `disabled`

Grouping input: `<fieldset>`, `<legend>`

```
<fieldset>
<legend>Credit cards:</legend>
<input type="radio" name="creditcards" id="visa" />
    <label for="visa">Visa</label><br />
<input type="radio" name="creditcards" id="mastercard" />
    <label for="mastercard">MasterCard</label><br />
<input type="radio" name="creditcards" id="amex" />
    <label for="amex">American Express</label><br />
</fieldset>
```

Credit cards:
- ◯ Visa
- ◯ MasterCard
- ◯ American Express

- groups related input fields; `legend` supplies an optional caption

Practice problem: Colored text

- Write the HTML and Javascript code to present a text area and three on/off options for red, green, and blue.
- When the user checks each box, it will add or remove that color from the text area's text.