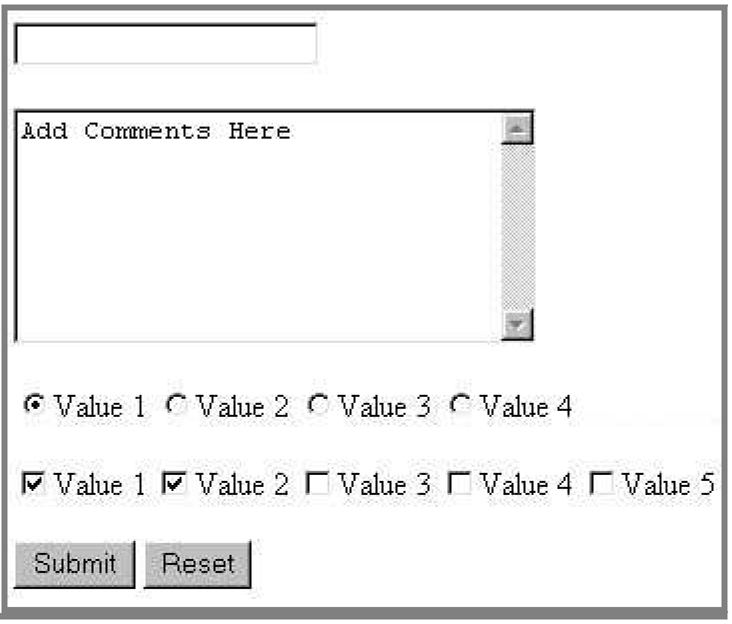
HTML User Interface Controls

CSE 190 M (Web Programming), Spring 2008
University of Washington

Reading: Chapter 3 section 3.1

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)



Buttons: `<button>`

the most common clickable UI widget (inline)

```
<button>Click me!</button>
```
HTML

Click me!

- button's text appears inside `button` tag
- A button can also contain images (`img`) and other content

Radio buttons: `<input>`

sets of mutually exclusive choices (inline)

```
<input type="radio" name="creditcards" /> Visa
<input type="radio" name="creditcards" /> MasterCard
<input type="radio" name="creditcards" checked="checked" /> American Express
```

○ Visa ○ MasterCard ● American Express

- grouped by `name` attribute (only one can be checked at a time)

Text labels: `<label>`

```
<label><input type="radio" name="creditcards" /> Visa</label>
<label><input type="radio" name="creditcards" /> MasterCard</label>
<label><input type="radio" name="creditcards" /> American Express</label>
```

○ Visa ○ MasterCard ○ American Express

- can be used with checkboxes or radio buttons
- label is clickable (better usability)
- content is more semantic
- `label` element can be targeted by CSS style rules

Checkboxes: `<input>`

an on/off toggle (inline)

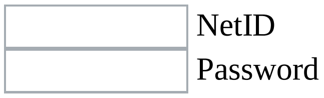
```
<label><input type="checkbox" /> Lettuce</label>
<label><input type="checkbox" checked="checked" /> Tomato</label>
<label><input type="checkbox" /> Pickles</label>
```

☐ Lettuce ☑ Tomato ☐ Pickles

- `input` element is used to create many UI controls
  - an inline, self-closing tag
- none-to-many checkboxes can be checked at same time
- use `checked="checked"` attribute in HTML to initially check the box

Text fields: `<input>`

```html
<input type="text" size="12" maxlength="8" /> NetID<br />
<input type="password" size="12" /> Password
```
HTML

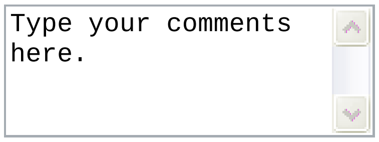

NetID
Password

- `input` attributes: `disabled`, `maxlength`, `name`, `readonly`, `size`, `type`, `value`
- `size` attribute controls onscreen width of text field
- `maxlength` limits how many characters user is able to type into field

Text boxes: `<textarea>`

a multi-line text input area (inline)

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


```
Type your comments
here.
```

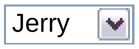
- initial text is placed inside `textarea` tag (optional)
- required `rows` and `cols` attributes specify size in characters
- optional `readonly` attribute means text cannot be modified

Drop-down list: `<select>`, `<option>`

menus of choices that collapse and expand (inline)

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

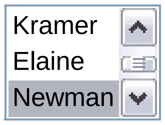
HTML



- `option` element represents each choice
- `select` optional attributes: `disabled`, `multiple`, `size`

Using `<select>` for lists

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



- optional `size` attribute controls how many items can be seen (default 1)
- optional `multiple` attribute allows selecting multiple items with shift- or ctrl-click
- `option` tags can be set to be initially `selected`

Option groups: `<optgroup>`

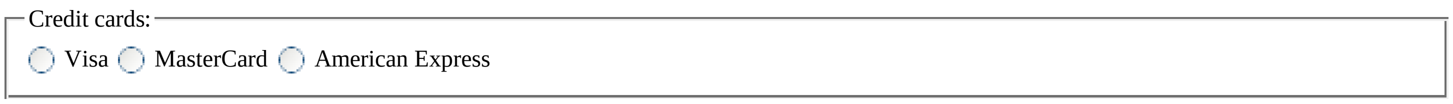
```
<select>
  <optgroup label="Major Characters">
    <option>Jerry</option>
    <option>George</option>
    <option>Kramer</option>
    <option>Elaine</option>
  </optgroup>
  <optgroup label="Minor Characters">
    <option>Newman</option>
    <option>Susan</option>
  </optgroup>
</select>
```

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

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

groups of input fields with optional caption (block)

```
<fieldset>
  <legend>Credit cards:</legend>
  <label><input type="radio" name="creditcards" /> Visa</label>
  <label><input type="radio" name="creditcards" /> MasterCard</label>
  <label><input type="radio" name="creditcards" /> American Express</label>
</fieldset>
```



- `fieldset` groups related input fields; `legend` supplies an optional caption
- `fieldset` and `legend` can be targeted by CSS style rules

Common UI control errors

- "I changed the checkbox's `checked` property, the `textarea`'s inner text, the text box's `value` ... but when I refresh, the page doesn't reflect this change!"
    - By default, when you refresh a page in your browser, it leaves the previous values in all UI controls
    - it does this in case you were filling out a long form and needed to refresh it, but didn't want it to clear out all the info you'd entered
    - if you want it to clear out all UI controls' state and values, you must do a **full refresh**
        - Firefox: Shift-Ctrl-R
        - Mac: Shift-Command-R

Styling UI controls

```
element[attribute="value"] {
   property : value;
   property : value;
   ...
   property : value;
}
```
JS

```
input[type="text"] {
  background-color: yellow;
  font-style: bold;
}
```
JS

- CSS **attribute selector**: matches only XHTML elements that have a particular attribute set to a certain value
- useful for styling UI controls because many of them share the same element (`input`)

Styling Text Boxes

```html
<textarea rows="3" cols="40"></textarea>
```
HTML

```css
body { height: 100%; }
textarea {
  width: 50%;
  height: 15%;
}
```
JS

- XHTML validator requires `rows` and `cols` on a `textarea`
- if you want a `textarea` at a specific width/height in pixels or %, you must specify `rows`/`cols` in the XHTML *and* `width`/`height` in the CSS
  - the `rows`/`cols` will be ignored but must be there anyway...
  - sometimes specifying a `height` on the page's `body` helps

Making UI controls interactive

(using a bit of JavaScript)

What is JavaScript?

- a lightweight programming language (scripting)
- used to make web pages interactive
  - insert dynamic text into HTML (ex: user name)
  - react to events (ex: page load user click)
  - get information about a user's computer (ex: browser type)
  - perform calculations on user's computer (ex: form validation)
- a web standard (but not supported identically by some browsers)
- not related to Java other than by name and some syntactic similarities

Creating an interactive UI

- To make a responsive UI control:
  1. choose the control (e.g. button) and event (e.g. mouse click) of interest
  2. write a JavaScript function to run when the event occurs
  3. attach the function to the event on the control

Inserting JavaScript in HTML

- JavaScript code can be added to a web page in two ways:
  1. in the XHTML file's `body` or `head` (BAD STYLE)
  2. in an external `.js` file, linked to the XHTML file in its `head` (good style)

Linking to a JavaScript file (example)

```
<script src="filename" type="text/javascript"></script>
```

HTML

```
<script src="example.js" type="text/javascript"></script>
```

- should be placed in XHTML page's `head`
- script code is stored in a separate `.js` file

A basic JavaScript function

```
function name() {
  statement ;
  statement ;
  ...
  statement ;
}
```

JS

- the function is the fundamental unit of execution

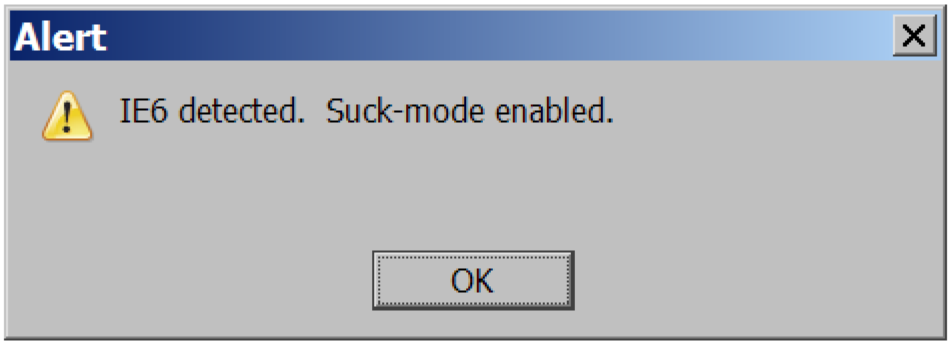
The `alert` box

```
alert("message");
```

JS

```
alert("IE6 detected.  Suck-mode enabled.");
```

JS


Alert

IE6 detected. Suck-mode enabled.

- a JS command that pops up a dialog box with a message

Function example

```
function myFunction() {
  alert("Hello!");
  alert("How are you?");
}
```

- the above could be the contents of `example.js` linked to our XHTML document

Event handlers

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

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
- `onclick` is just one of many event HTML attributes we'll see later

Another event handler

```
<select onchange="myFunction();">
  <option>Jerry</option>
  <option>George</option>
  <option>Kramer</option>
  <option>Elaine</option>
</select>
```

Jerry

- when a select box's selected item changes, an `onchange` event occurs
- other events: `onabort`, `onblur`, `onchange`, `onclick`, `ondblclick`, `onerror`, `onfocus`, `onkeydown`, `onkeypress`, `onkeyup`, `onload`, `onmousedown`, `onmousemove`, `onmouseout`, `onmouseover`, `onmouseup`, `onreset`, `onresize`, `onselect`, `onsubmit`, `onunload`