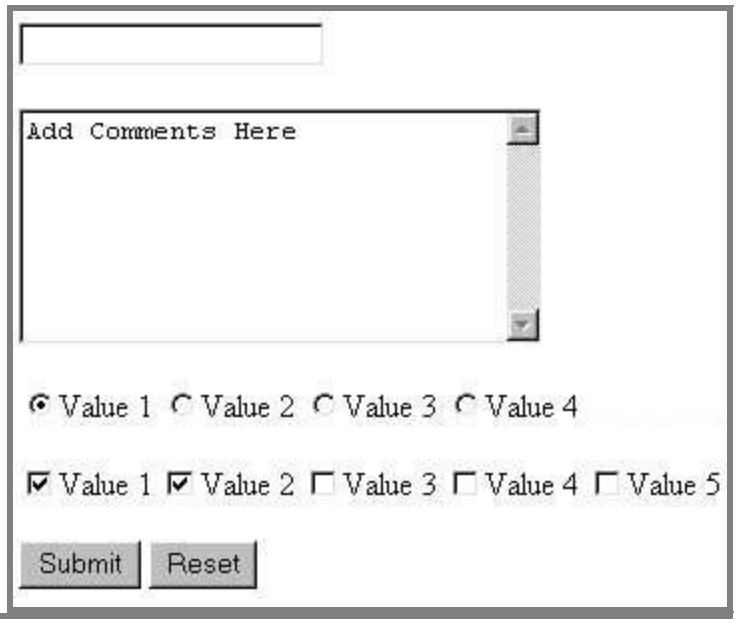
HTML Forms

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

Reading: Sebesta Ch. 2 section 2.9
References: JavascriptKit, w3schools

HTML forms

- an HTML form is a group of UI controls that accepts information from the user and sends the information to a web server
- forms use HTML UI controls (buttons, checkboxes, text fields, etc.)
- the information is sent to the server as a query string



HTML form: `<form>`

```
<form action="web app URL" method="get or post">
  <fieldset>
    form controls
  </fieldset>
</form>
```

- required `action` attribute gives the URL of the server web application that will process this form's data
- `method` attribute specifies whether the server should use an HTTP GET or POST command (explained later)

Form example

```
<form action="http://www.foo.com/app.php" method="get">
 <fieldset>
  <label>Name: <input type="text" name="name" /></label>
  <label>Meal: <input type="text" name="meal" /></label>
  <label>Meat?
    <input type="checkbox" name="meat" />
  </label>
  <input type="submit" />
 <fieldset>
</form>
```

- always wrap the form's controls in one or more `fieldset`s

Recall: the HTML UI controls

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

Type your comments here.

```
<select name="seinfeld">
 <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

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

```
<label><input type="radio" name="cc" />Visa</label>
<label><input type="radio" name="cc" />MasterCard</label>
```

Visa MasterCard

```
<label>
<input type="checkbox" name="lettuce" />
 Lettuce</label>
<label>
<input type="checkbox" name="tomato" />
 Tomato</label>
```

Lettuce Tomato

Submit

Importance of `name` attribute

```
<form action="http://foo.com/app.php" method="get">
 <fieldset>
  <label>Name: <input type="text" name="name" /></label>
  <label>Meal: <input type="text" name="meal" /></label>
  <label>Meat?
    <input type="checkbox" name="meat" />
  </label>
  <input type="submit" />
 <fieldset>
</form>
```

- each control's `name` specifies the query string parameter to pass
- if user types "Sue" as name, "pizza" as meal, and checks Meat? box, then clicks Submit button, the browser will go to this URL:

  `http://foo.com/app.php?name=Sue&meal=pizza&meat=on`

`get` vs. `post`

```
<form action="http://www.foo.com/app.php" method="post">
  ...
</form>
```

- a `get` request passes the parameters to the server as a query string
- a `post` request embeds the parameters in HTTP request, not in the URL
- advantages of `post` :
  - `get` is limited to browser's URL length, around 100-200 characters
  - information is more private (not shown in browser address bar)
  - Example of a bad use of `get` (click "Buy it!")
- disadvantages of `post` :
  - can't be bookmarked
  - browser can't easily go back (the famous POSTDATA error)

`submit` and `reset` buttons

```
<form action="http://www.foo.com/app.php" method="get">
 <fieldset>
  ...
  <input type="submit" />
  <input type="reset" />
 <fieldset>
</form>
```

- an `input` element with a `submit` attribute is displayed as a button that, when clicked, will send the parameters to the server and show the response
- an `input` element with a `reset` attribute is displayed as a button that, when clicked, will change the controls back to their original state

`submit`, `reset` example

Name: Marty
Meal: pizza
Meat?
Submit Reset

- specify custom text on the buttons by setting their `value` attribute

```
<input type="submit" value="Order Meal" />
```

Practice problem: Online banking

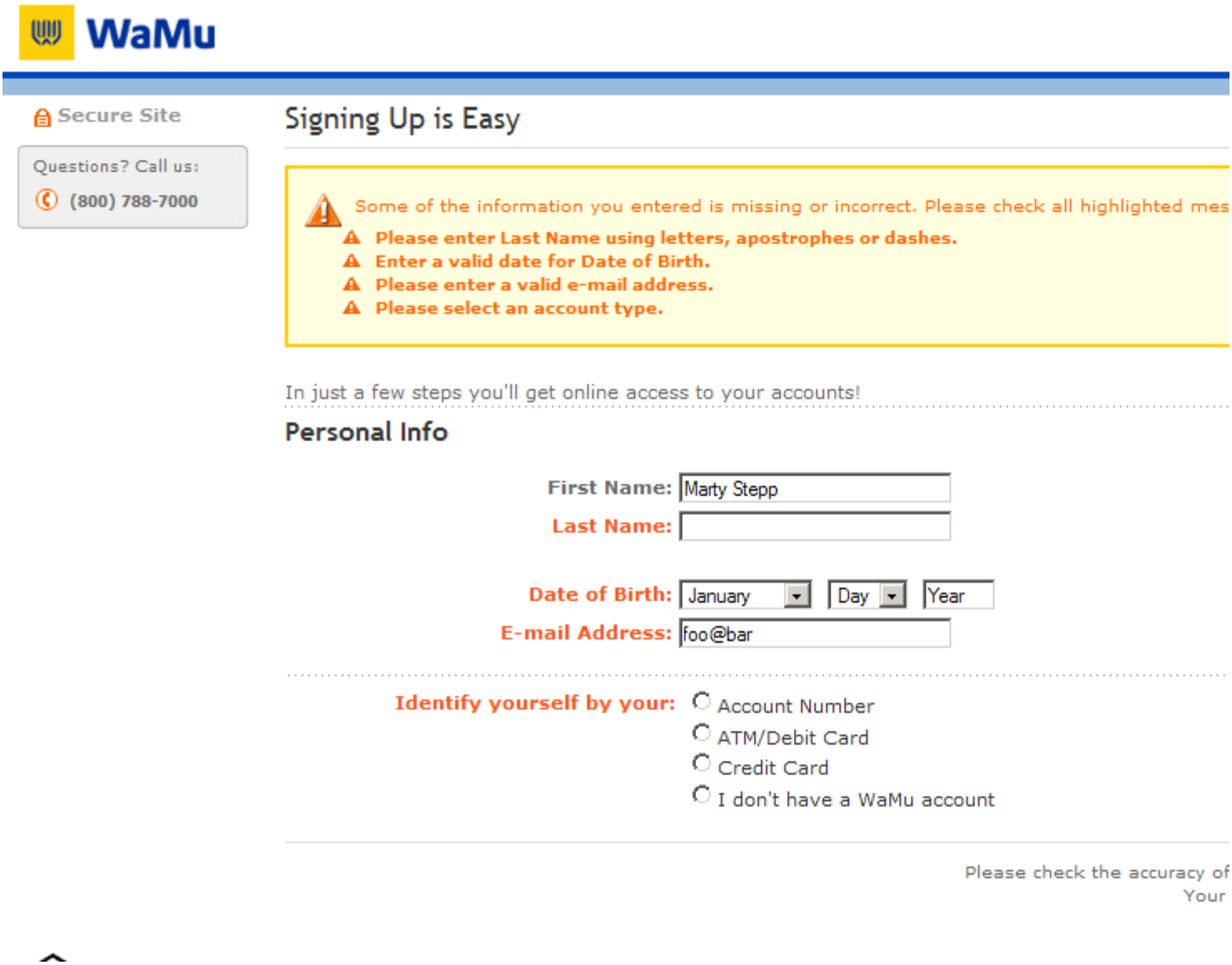
Create a web page with a form that lets the user sign up for an online banking account such as those offered at Washington Mutual or Orange Savings.

- Once the form has been filled out, send the user to http://faculty.washington.edu/stepp/params.php. (Try using both a GET and a POST request.)

Form validation

- validation: ensuring that form's values are correct
- can be performed on client (JS), on server (PHP), or both
- some types of validation:
  - preventing blank values (email address)
  - ensuring the type of values
    - integer, real number, currency, phone number, Social Security number, postal address, email address, date, credit card number, ...
  - ensuring the format and range of values (ZIP code must be a 5-digit integer)
  - ensuring that values fit together (user types email twice, and the two must match)

Form validation example



Cancelling form submission

```
    // in window.onload handler:
    var submit = document.getElementById("submitbutton");
    submit.onclick = submitClick;
    ...

function submitClick(event) {
    if (document.getElementById("name").value == "") {
        event.preventDefault();  // cancel submit
    }
}
```

- if the form's values are not valid, we can stop the form from submitting
- call `preventDefault` to tell the button to stop its default behavior
- IE6 uses `return false;` instead of `preventDefault`

----- Regular expressions -----

Form validation
with regular expressions

Reading: Sebesta Ch.4 section 4.12

What is a regular expression?

```
/^[\w\.%\-]+@[\w.\-]+\.[a-zA-Z]{2,4}$/
```

- regular expression: a description of a pattern of text characters
- a given string can be tested against a regular expression:
  - to see whether the string matches the expression's pattern
  - to replace characters in the string
- regular expressions are extremely powerful but tough to read
  (the above regular expression matches email addresses)
- regular expressions are used all over the place:
  - Java: `Scanner`'s `useDelimiter` method, `Pattern` class, `String`'s `split` method (CSE 143 sentence generator)
  - Other languages: Javascript, PHP, ...
  - Search/replace function in many text editors (TextPad)

Regular expression examples

```
/abc/
```

- regular expressions generally begin and end with `/`
- the simplest regular expressions simply match a particular string
- the above regular expression matches the following strings:
  - `"abcdef"`, `"defabc"`, `".=.abc.=."`, ...
- the above expression doesn't match any of these:
  - `"fedcba"`, `"ab c"`, `"JavaScript"`, ...
- `.` matches any character except newline
  - `/.oo.y/` matches "Doocy", "goofy", "PooPy", ...

Special characters in regular expressions

- `|` means *or*
  - `/abc|def|g/` matches "abc", "def", or "g"
- `()` are for grouping
  - `/(Homer|Marge) S/` matches "Homer S" or "Marge S"
  - What does `/Homer|Marge|Bart|Lisa|Maggie S/` match?
- `^` means beginning of line; `$` means end
  - `/^<html>$/` matches a line that consists entirely of `"<html>"`

Quantifiers

- `*` means 0 or more occurrences
  - `/abc*/` matches "ab", "abc", "abcc", ...
  - `/a(bc)*/` matches "a", "abc", "abcbc", "abcbcbc", ...
  - `/a.*a/` matches "aa", "aba", "a8qa", "a!?_a", ...
- `+` means 1 or more occurrences
  - `/a(bc)+/` matches "abc", "abcbc", "abcabcabc", ...
  - `/Goo+gle/` matches "Google", "Gooogle", "Goooogle", ...
- `?` means 0 or 1 occurrences
  - `/a(bc)?/` matches "a" or "abc"

More quantifiers

- `{min,max}` means between min and max occurrences (inclusive)
  - `/a(bc){2, 4}/` matches "abcbc", "abcbcbc", or "abcbcbcbc"
- min or max may be omitted to specify any number
  - `{2,}` means 2 or more
  - `{,6}` means up to 6
  - `{3}` exactly 3
- What regular expression matches dollar amounts of at least $100.00 ?

Character sets

- `[]` group characters into a character set; the expression will match any single character from the set
  - `/[bcd]art/` matches "bart", "cart", and "dart"
- inside `[]`, many of the modifier keys act as normal characters
  - `/what[!*?]*/` matches "what", "what!", "what?**!", "what??!", ...
- What regular expression matches letter grades such as A, B+, or D- ?
- What regular expression matches DNA (strings where every letter must be one of A, C, G, or T)?

Character ranges

- inside a character set, specify a range of characters with `-`
  - `/[a-z]/` matches any lowercase letter
  - `/[a-zA-Z0-9]/` matches any lower- or uppercase letter or digit
- inside a character set, `^` means negation
  - `/[^abcd]/` matches any character other than a, b, c, or d
- What regular expression would match UW Student IDs?
- What regular expression would match consonants, assuming that the string consists only of lowercase letters?

Escape sequences

- `\` starts an escape sequence
- special escape sequence character sets:
  - `\d` matches any digit (same as `[0-9]`)
  - `\D` matches any non-digit (same as `[^0-9]`)
  - `\w` matches any "word character" (same as `[a-zA-Z_0-9]`)
  - `\W` matches any non-word character (same as `[^a-zA-Z_0-9]`)
  - `\s` matches any whitespace (, `\t`, `\n`, etc.)
  - `\S` matches any non-whitespace
- What regular expression would match HTML tags?

More about escape sequences

- to match these chars, you must escape them with a `\`:
  `\ | ( ) [ { ^ $ * + ? .`
  - `/\\|\/\?/` matches `"\|/?"`
- inside a character set, `-` must be escaped to be matched
  - `/[0-9+\-]/` matches any single digit, a `+`, or a `-`
- What regular expression would match numbers in scientific notation?

Regular expressions in Javascript

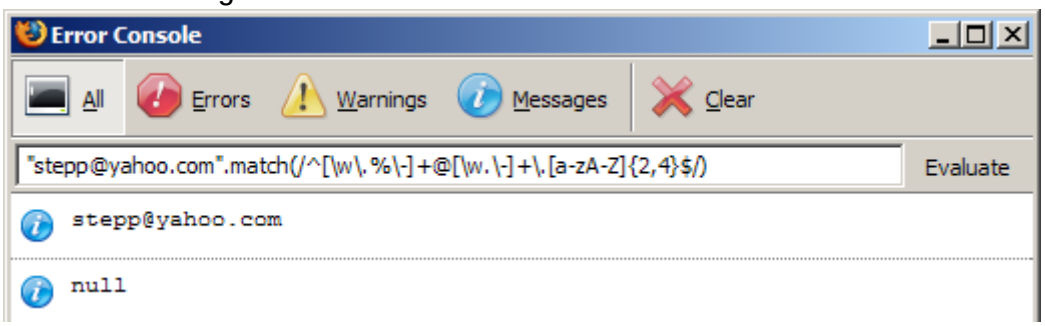
- string.match(regexp)
  - if string fits the pattern, returns matching text; else returns null
  - can be used in Boolean context
    if (myString.match(/[a-z]+/)) { ... }
- an i can be placed after the regexp for a case-insensitive match
  - myString.match(/Marty/i) will match "marty", "MaRtY", ...
- storing a regular expression as a variable
  - var regexp = /(Bart|Lisa|Maggie) Simpson/;
    if (myString.match(regexp)) { ... }

Replacing text with regular expressions

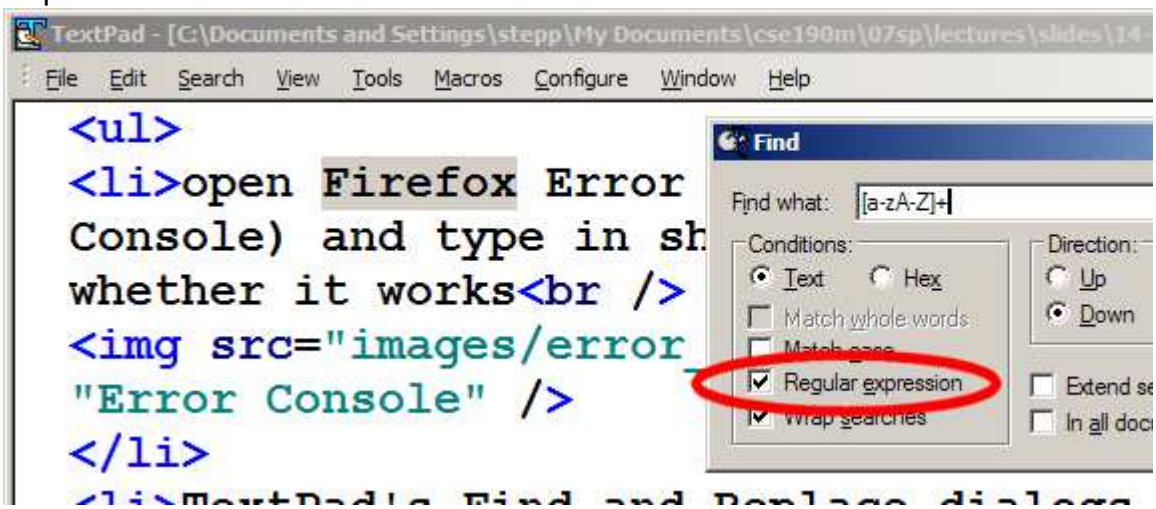
- string.replace(regexp, "text")
  - replaces the first occurrence of given pattern with the given text
  - var str = "Marty Stepp";
    str.replace(/[a-z]/, "x") returns "Mxrty Stepp"
  - returns the modified string as its result; must be stored
    str = str.replace(/[a-z]/, "x")
- a g can be placed after the regexp for a global match (replace all occurrences)
  - str.replace(/[a-z]/g, "x") returns "Mxxxx Sxxxx"
    - using a regexp as a filter
      - str = str.replace(/[^A-Z]+/g, "") turns str into "MS"

Debugging/testing regular expressions

- open Firefox Error Console (Tools, Error Console) and type in short regex code to see whether it works



- TextPad's Find and Replace dialogs allow regular expressions



Practice problem: Form validation

Use regular expressions to validate the online banking form you wrote previously.

- Start out by showing an alert if the data is bad, and not submitting the form.
- Enhance the code by removing the alert and instead highlighting invalid data in pink.