Lecture 4: HTML Forms

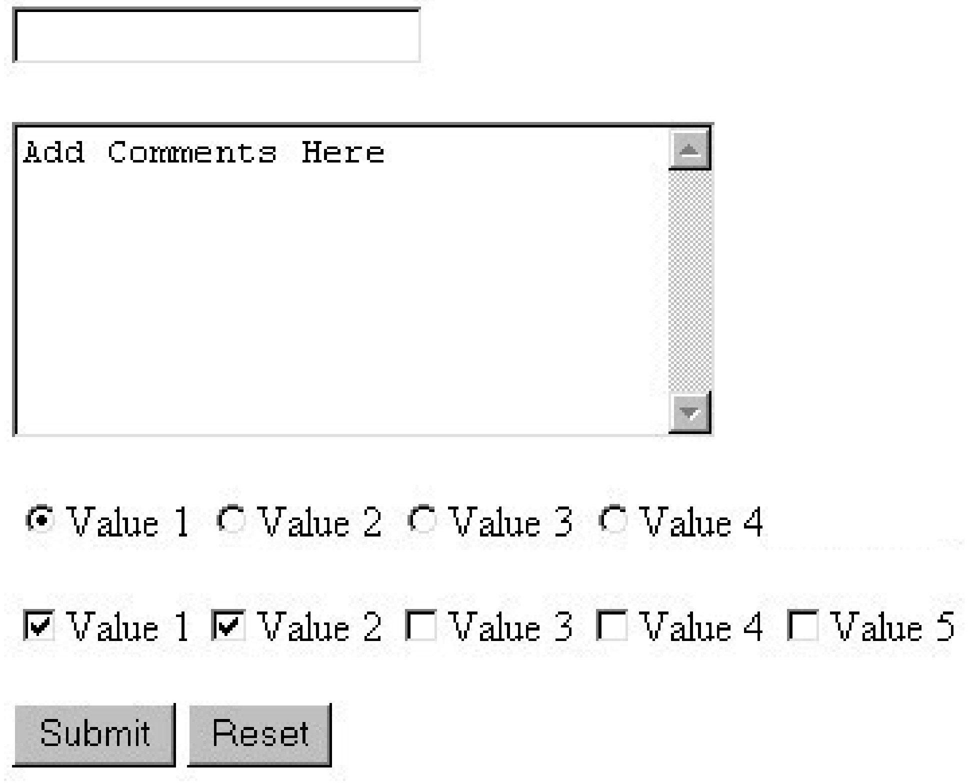
Connecting with the Source

Wholeness Statement

- HTML forms generate and process user input. Users complete HTML forms using different data widgets, and the server will process the information submitted from the forms. *Forms are a communication mechanism, connecting a model with a remote view of the model and in this case providing input to the model. Forms collect and send information to the server like our sensory systems collect and send perceptions to our awareness.*

HTML forms

- Form: a group of UI controls that accepts information from the user and sends the information to a web server
- JavaScript can be used to create interactive controls
- We can have multiple forms on a single webpage, but forms CANNOT be nested.



HTML form: `<form>`

The <**form**> tag is used to create an HTML form for user input.

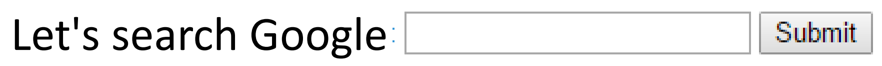
The <**form**> element can contain one or more of the following form elements:
<**input**>, <**textarea**>, <**button**>, <**select**>, <**option**>, <**optgroup**>, <**fieldset**>, <**label**>

```
<form action="" method ="" enctype="" novalidate autocomplete>
    form controls
</form>
```

- **action**: action to be performed when the form is submitted (server url).
- **method**: get, post
- **enctype**: application/x-www-form-urlencoded, multipart/form-data, text/plain
- **novalidate**: (HTML5) specifies that the form should not be validated when submitted
- **autocomplete**: (HTML5) on, off

Form Example

```
<form action="http://www.google.com/search">
   <div> Let's search Google
       <input name="q" />
       <input type="submit" />
   </div>
</form>
```


Let's search Google:

Query strings and parameters

http://www.google.com/search?q=Obama
http://mum.edu/login.jsp?username=guest&sid=1234567

Query string: a set of parameters passed from a browser to a web server. Often passed by placing name/value pairs at the end of a URL.

Above, parameter **username** has value “guest”, and **sid** has value “1234567”

GET vs. POST requests

GET : asks a server for a page or data

- if the request has parameters, they are sent in the URL as a query string (request header)
- URLs are limited in length (~ 1024 characters)
- URLs cannot contain special characters without encoding
- private data in a URL can be seen or modified by users

POST : submits data to a web server for processing

- parameters are embedded in the HTTP request body, not the URL
    - More secure than GET for this reason only, but it still is plain text.

Request Command

There are 3 ways to send a request from a browser tab to the server:

- Type url (GET)
- Form (GET, POST)
- Using JavaScript: XHR object (GET, POST, PUT, DELETE, ...)

Request methods:

- GET: only has header (parameter are sent in the header, NO body)
- POST: has header and body ([parameters are sent in the body)
- PUT
- DELETE
- OPTIONS
- TRACE
- HEAD

Main Point

- An HTML form allows the user to send data (input parameters) to the server. Forms are created with the <form> tag, and can be submitted with either an HTTP GET or POST method.
- *HTTP is a protocol for contacting the server and thereby gaining access to all the resources on the server. Similarly, the TM technique is a protocol for contacting the Self and thereby gaining access to all the resources of pure consciousness.*

`<input />`

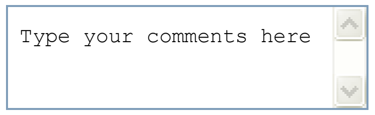
`input` element is used to create inline input element and MUST be self-closed

- **`name`** attribute specifies name of query parameter to pass to server
- **`type`** attribute can be button, checkbox, color, date, datetime, datetime-local, email, file, hidden, image, month, number, password, radio, range, reset, search, submit, tel, text, time, url, week
- **`value`** attribute specifies control's initial text
- **`required`** `(HTML5)` if the input field must be filled out before submitting the form
- **`placeholder`** `(HTML5)` a short hint that describes the expected value of an <input> element
- **`pattern`** `(HTML5)` a regular expression that an <input> element's value is checked against
- **`size`** the width, in characters, of an <input> element
- **`maxlength`** the maximum number of characters allowed in an <input> element
- **`readonly`**

<textarea>

- The <**textarea**> tag defines a multi-line text input control.
- A textarea can hold an unlimited number of characters, and the text renders in a fixed-width font (usually Courier).
- The size of a textarea can be specified by the **cols** and **rows** attributes, or even better; through CSS' **height** and **width** properties.

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



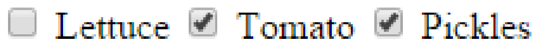
```
Type your comments here
```

Checkboxes

yes/no choices that can be checked and unchecked

- none, 1, or many checkboxes can be checked at same time
- Use the **`checked`** attribute in HTML to initially check the box

```
<input type="checkbox" name="lettuce" /> Lettuce
<input type="checkbox" name="tomato" checked /> Tomato
<input type="checkbox" name="pickles" checked /> Pickles
```

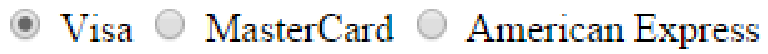


Radio buttons

Sets of mutually exclusive choices (**inline**)

- Grouped by **`name`** attribute (only one can be checked at a time)
- Must specify a **`value`** for each one or else it will be sent as value **`on`**

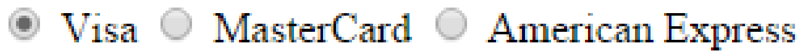
```
<input type="radio" name="cc" value="visa" checked /> Visa
<input type="radio" name="cc" value="mastercard" /> MasterCard
<input type="radio" name="cc" value="amex" /> American Express
```


● Visa ○ MasterCard ○ American Express

Text labels: `<label>`

- Associates nearby text with control, so you can **click text to activate control**
- Can be used with **checkboxes** or **radio** buttons
- **`label`** element can be targeted by CSS style rules

```html
<label><input type="radio" name="cc" value="visa" checked="checked" /> Visa</label>
<label><input type="radio" name="cc" value="mastercard" /> MasterCard</label>
<label><input type="radio" name="cc" value="amex" /> American Express</label>
```



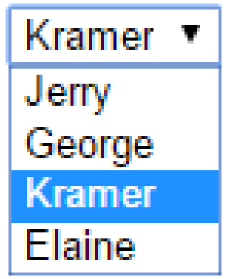
● Visa ○ MasterCard ○ American Express

Drop-down list `<select>` and `<option>`

Menus of choices that collapse and expand (inline)

- **`option`** element represents each choice
- **`select`** optional attributes: **`disabled`**, **`multiple`**, **`size`**
- optional **`selected`** attribute sets which one is initially chosen

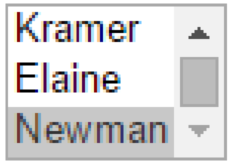
```
<select name="favoritecharacter">
    <option>Jerry</option>
    <option>George</option>
    <option selected>Kramer</option>
    <option>Elaine</option>
</select>
```



Multiple Selection

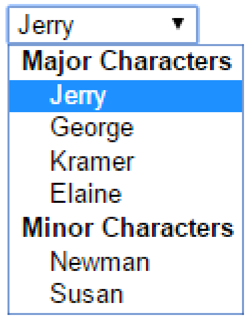
- optional **`multiple`** attribute allows selecting multiple items with shift- or ctrl-click
- **`option`** tags can be set to be initially **`selected`**

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



Option groups: <optgroup>

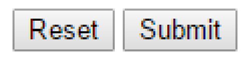
```
<select name="favoritecharacter">
    <optgroup label="Major Characters">
        <option>Jerry</option>
        <option>George</option>
        <option>Kramer</option>
        <option>Elaine</option>
    </optgroup>
    <optgroup label="Minor Characters">
        <option>Newman</option>
       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



Reset and Submit buttons

- When we click **`reset`** button, it returns all form controls to their initial values
- When we click **`submit`** buttons, it sends all data with the specified **`method`** (Get/Post) to the **`action`** page in the form
- Specify custom text on the button by setting its **`value`** attribute

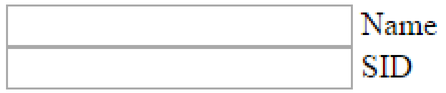
```html
<input type="reset" />
<input type="submit" />
```



Hidden input parameters

An invisible parameter that is still passed to the server when form is submitted, it’s useful for passing on additional state that isn't modified by the user

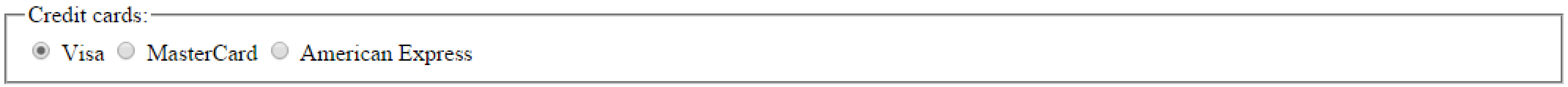
```
<input type="text" name="username" /> Name <br />
<input type="text" name="sid" /> SID <br />
<input type="hidden" name="school" value="MUM" />
<input type="hidden" name="year" value="2048" />
```



Grouping `<fieldset>`, `<legend>`

Groups of input fields with optional caption (block)

```html
<fieldset>
     <legend>Credit cards:</legend>
     <input type="radio" name="cc" value="visa" checked="checked" /> Visa
     <input type="radio" name="cc" value="mastercard" /> MasterCard
     <input type="radio" name="cc" value="amex" /> American Express
</fieldset>
```


Credit cards:
● Visa ○ MasterCard ○ American Express

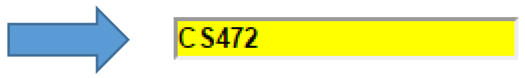
Styling form controls

Because most input element are created using input tag, we target each group of elements using this CSS selector:

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

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

input[type="button"] {
        width: 120px;
        display: block;
}
```



Main Point

- HTML provides many different types of input widgets, including text fields, text areas, check boxes, radio buttons, and dropdown lists, this is also an area HTML 5 is expanding to make form filling more efficient and effortless.

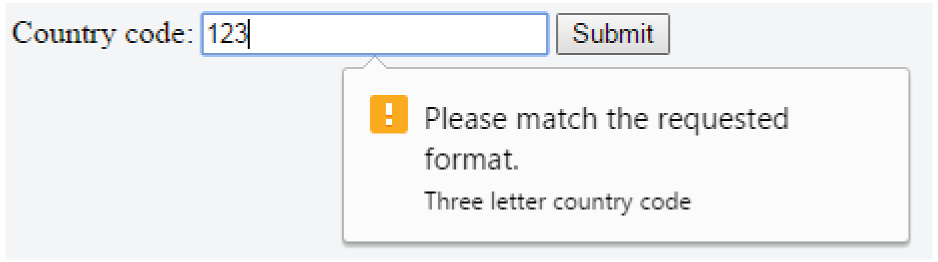
- *Nature supports the growth of things that are efficient. Do less and accomplish more.*

Pattern and regex

The **pattern** attribute specifies a regular expression that the <**input**> element's value is checked against. The **pattern** uses the ECMAScript (i.e. JavaScript) flavor of regex.

Note: The **pattern** attribute works with the following input types: **text**, **date**, **search**, **url**, **tel**, **email**, and **password**.

```
<form action="demo_form.jsp">
      Country code:
      <input type="text" name="country_code" pattern="[A-Za-z]{3}"
                                 title="Three letter country code">
      <input type="submit">
</form>
```


Country code: 123 Submit

Please match the requested format.
Three letter country code

Regular expressions

```
^[a-zA-Z_\-]+@(([a-zA-Z_\-])+\.)+[a-zA-Z]{2,4}$
```

- Regular expression ("regex"): a description of a pattern of text
- Can test whether a string matches the expression's pattern
- Regular expressions are extremely powerful but tough to read
  - (the above regular expression matches email addresses)
- Regular expressions are used in all languages:
  - Java, PHP ,JavaScript, HTML, C#, and other languages
- Many IDEs allow regexes in search/replace

Basic regular expressions

The simplest regexes simply matches any string that contains that text.

```
abc
```

The above regular expression matches any string containing "abc":

- YES: "abc", "abcdef", "defabc", ".=.abc.=.", ...
- NO: " ABC" , " fedcba", "ab c", "PHP", ...

- Regular expressions are case-sensitive by default.

Anchors: ^ and $

`^` represents the beginning of the string or line;

`$` represents the end

- `Jess` matches all strings that contain Jess;
- `^Jess` matches all strings that start with Jess;
- `Jess$` matches all strings that end with Jess;
- `^Jess$` matches the exact string "Jess" only
- `^Mart.*Stepp$` matches "MartStepp", "Marty Stepp", "Martin D Stepp", ...
  but NOT "Marty Stepp stinks" or "I H8 Martin Stepp"

Wildcards .

A dot `.` matches exactly **one-character** except a `\n` (line break)

`.oo.y` matches "Doocy", "goofy", "LooNy", ...

Special characters: `|`, `()`, `\`

`|` means OR

`abc|def|g` matches "abc", "def", or "g"

There's no AND symbol. Why not?

`()` are for grouping

`(Homer|Marge) Simpson`

matches "Homer Simpson" or "Marge Simpson"

`\` starts an escape sequence

many characters must be escaped to match them literally: / \ $. [ ] ( ) ^ * + ?

`<br \/>` matches lines containing <br /> tags

Quantifiers: `*`, `+`, `?`

`*` means 0 or more occurrences

- `a*` matches "", "a", "aa", "aaa", ...
- `.*` matches "", "abc", "abcd", "aabbcc", ...
- `abc*` matches "ab", "abc", "abcc", "abccc", ...
- `a(bc)*` matches "a", "abc", "abcbc", "abcbcbc", ...
- `a.*a` matches "aa", "aba", "a8qa", "a!?xyz__9a", ...

`+` means 1 or more occurrences

- `a(bc)+` matches "abc", "abcbc", "abcbcbc", ...
- `Goo+gle` matches "Google", "Gooogle", "Goooogle", ...

`?` means 0 or 1 occurrences

- `a(bc)?` matches "a" or "abc"

More quantifiers: `{min,max}`

`{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}` means exactly 3

Character sets: [ ]

`[]` group characters into a character set, will match any **single character** from the set

- `[bcd]art` matches strings containing "bart", "cart", and "dart"
- equivalent to `(b|c|d)art` but shorter

inside [], many of the modifier keys act as normal characters

- `what[!*?]*` matches "what", "what!", "what?**!", "what??!", ...

Character ranges: `[start-end]`

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

an initial `^` inside a character set negates it

- `[^abcd]` matches any character other than a, b, c, or d

inside a character set, - must be escaped to be matched

- `[+\-]?[0-9]+` matches an optional + or -, followed by at least one digit

What regular expression matches letter grades such as A, B+, or D- ?

`[ABCDF][+\-]?`

Escape sequences

Special escape sequence character sets:

`\d` matches any digit (same as [0-9])

`\D` any non-digit ([^0-9])

`\w` matches any word character (same as [a-zA-Z_0-9])

`\W` any non-word char

`\s` matches any whitespace character (, \t, \n, etc.)

`\S` any non-whitespace

What regular expression matches dollar amounts of at least $100.00 ?

```
\$[1-9]\d{2,}\.\d{2}
```

Advanced: Lookaround

- Look ahead positive `(?=)`
  - Find expression A where expression B follows: `A(?=B)`
- Look ahead negative `(?!)`
  - Find expression A where expression B does not follow: `A(?!B)`
- Look behind positive `(?<=)`
  - Find expression A where expression B precede: `(?<=B)A`
- Look behind negative `(?<!)`
  - Find expression A where expression B does not precede: `(?<!B)A`

Examples

An <input> element with type="password" that must contain 8 or more characters that are of at least one number, and one uppercase and lowercase letter:

```
<form action="demo_form.jsp">
     Password:
     <input type="password" name="pw"
            pattern="((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,})"
            title="Must contain at least one number and one uppercase and lowercase
                                                 letter, and at least 8 or more characters">
     <input type="submit">
</form>
```

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

HTML Forms: Connecting with the Source

1. Forms let us submit data to the web server, which can then generate a custom response based on server side information.
2. GET requests are intended to only retrieve information and should be idempotent. POST requests are intended to submit data and not request a direct response.

3.Transcendental consciousness is the experience of the source of thought.
4.Impulses within the Transcendental field: inputs and perceptions are appreciated in their full value at the quiet levels of the mind.
5.Wholeness moving within itself: In Unity Consciousness one appreciates the full value of all inputs and perceptions. Everything is appreciated as some expression of Totality.