中山大學
SUN YAT-SEN UNIVERSITY

Lecture 9
Server Browser Interactions

SE-805 Web 2.0 Programming (supported by Google)

http://my.ss.sysu.edu.cn/courses/web2.0/

School of Software, Sun Yat-sen University

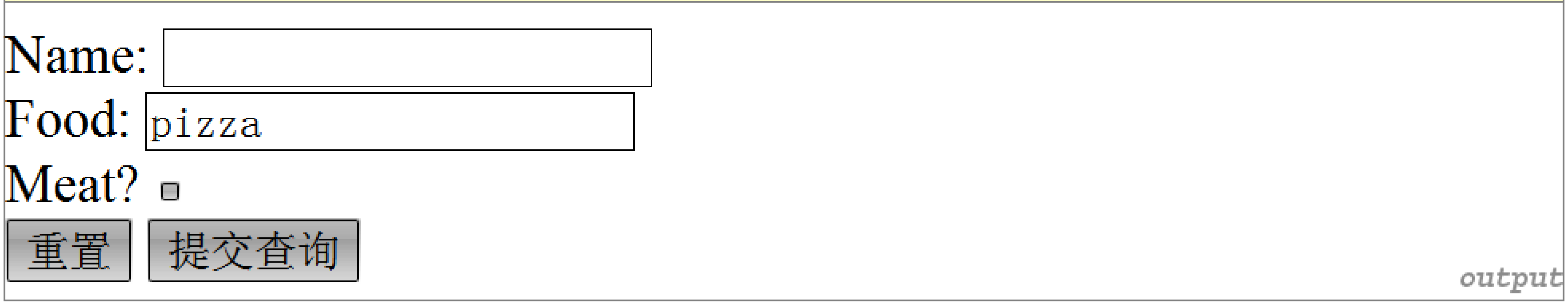
Outline

- **More HTML Forms**
- Submitting Data
- Processing Form data in PHP

Reset Buttons

```html
Name: <input type="text" name="name" /> <br />
Food: <input type="text" name="meal" value="pizza" /> <br />
<label>Meat? <input type="checkbox" name="meat" /></label> <br />
<input type="reset" />
```

HTML



Name:

Food: pizza

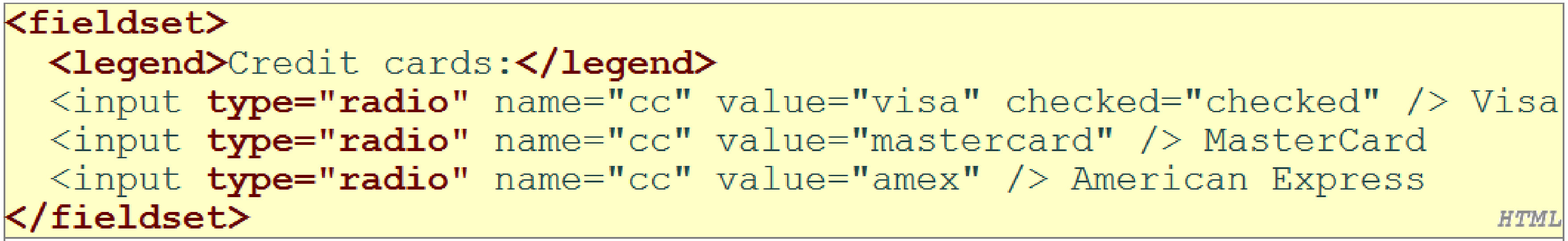
Meat? □

output

- When clicked, returns all form controls to their initial values
- Specify custom text on the button by setting its value attribute

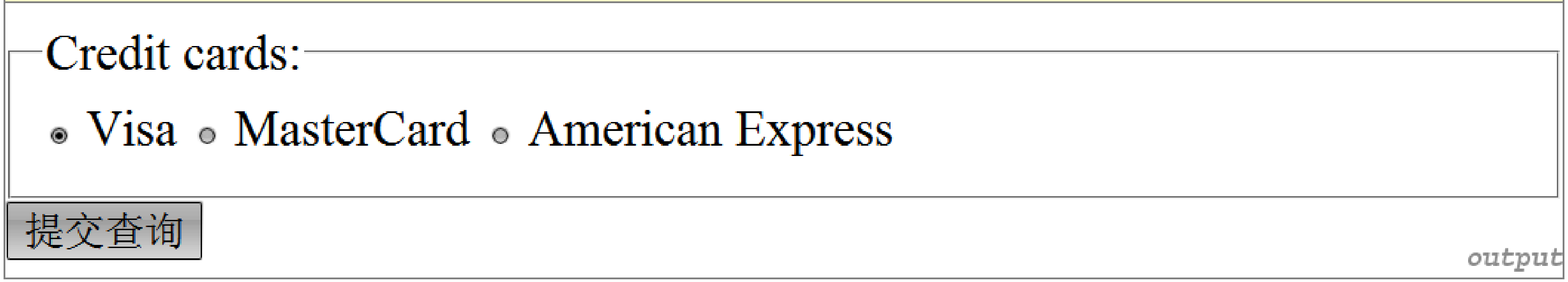
Grouping Input: <fieldset>, <legend>

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>
```

HTML



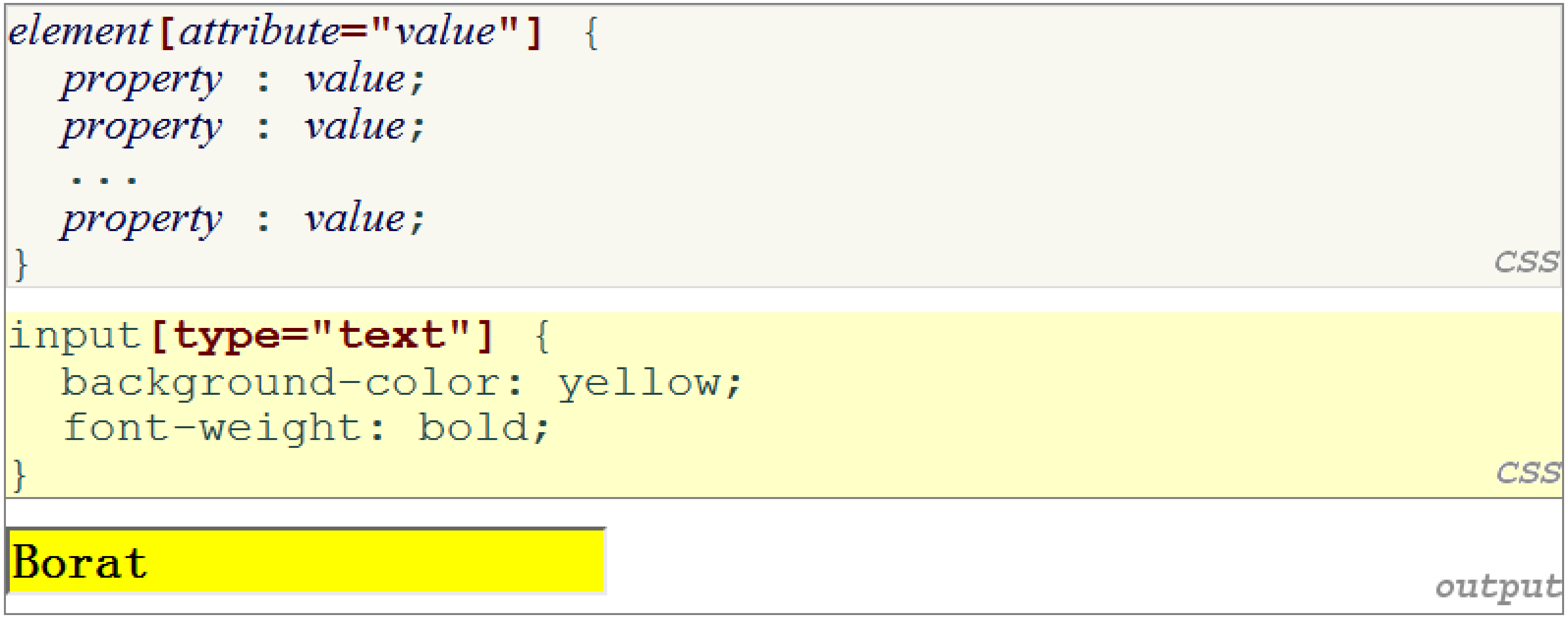
Credit cards:

● Visa ○ MasterCard ○ American Express

output

- fieldset groups related input fields, adds a border; legend supplies a caption

Styling Form Controls



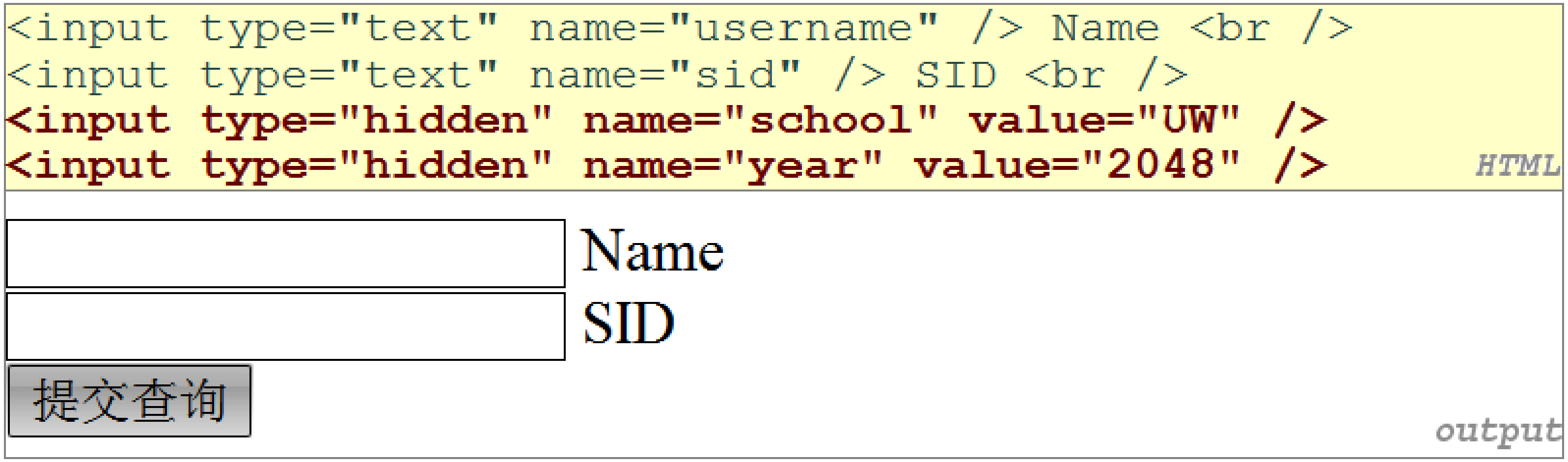
```
element[attribute="value"] {
   property : value;
   property : value;
   ...
   property : value;
}
```

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

Borat

- **Attribute selector**: matches only elements that have a particular attribute value
- Useful for controls because many share the same element (input)

Hidden Input Parameters



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

- An invisible parameter that is still passed to the server when form is submitted
- Useful for passing on additional state that isn't modified by the user
- Especially useful when passing parameters between adjacent web requests

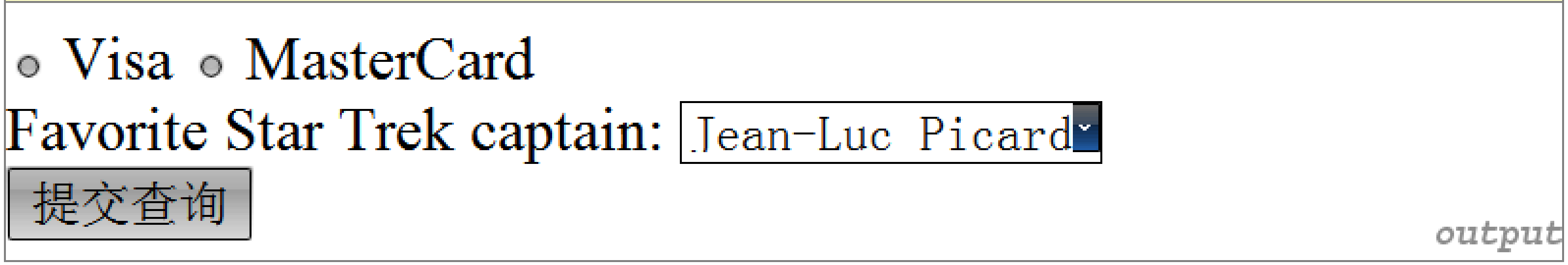
Outline

- More HTML Forms
- **Submitting Data**
- Processing Form data in PHP

Problems with Submitting Data

```
<label><input type="radio" name="cc" /> Visa</label>
<label><input type="radio" name="cc" /> MasterCard</label> <br />
Favorite Star Trek captain:
<select name="startrek">
  <option>James T. Kirk</option>
  <option>Jean-Luc Picard</option>
</select> <br />
```

HTML



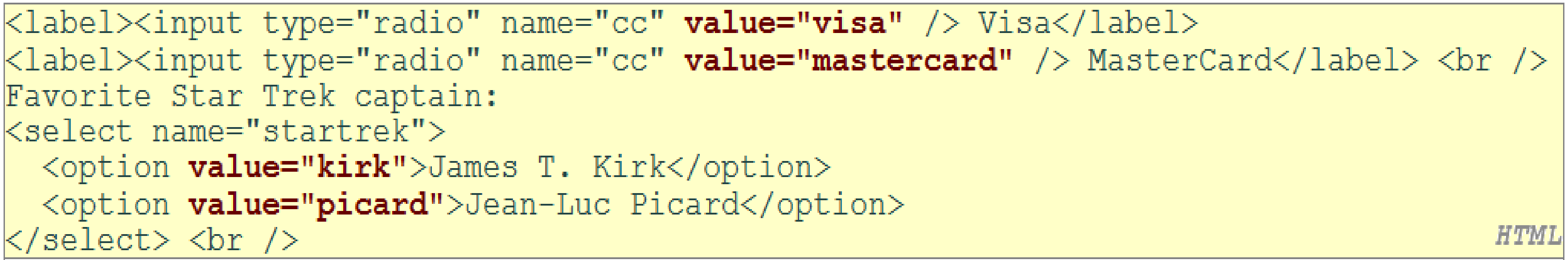
○ Visa ○ MasterCard

Favorite Star Trek captain: Jean-Luc Picard

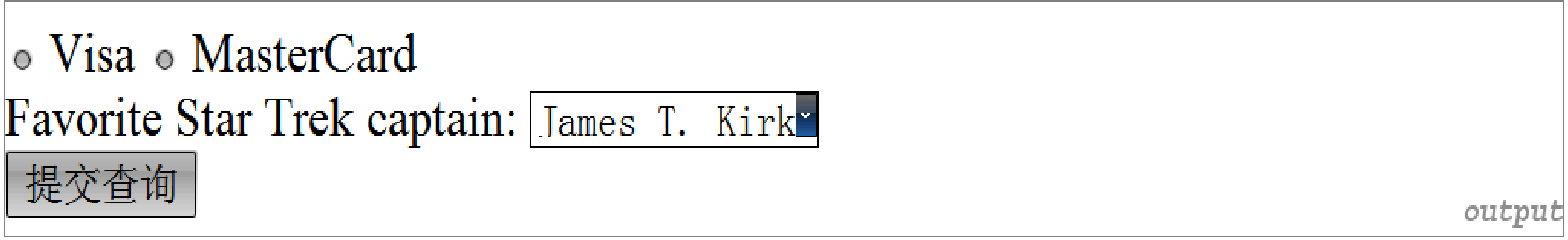
output

- This form submits to our handy params.php tester page
- The form may look correct, but when you submit it...
- ***[cc] => on***, [startrek] => Jean-Luc Picard

The value Attribute



```
<label><input type="radio" name="cc" value="visa" /> Visa</label>
<label><input type="radio" name="cc" value="mastercard" /> MasterCard</label> <br />
Favorite Star Trek captain:
<select name="startrek">
  <option value="kirk">James T. Kirk</option>
  <option value="picard">Jean-Luc Picard</option>
</select> <br />
```



○ Visa ○ MasterCard

Favorite Star Trek captain: James T. Kirk

- Value attribute sets what will be submitted if a control is selected
- [cc] => visa, [startrek] => kirk

URL-encoding

- Certain characters are not allowed in URL query parameters:
  - Examples: " ", "/", "=", "&"
- When passing a parameter, it is **URL-encoded** (reference table)
  - “Eric’s cool!?" → “Eric*%27s*+cool*%3F%21*"
- You don't usually need to worry about this:
  - The browser automatically encodes parameters before sending them
  - The PHP $_REQUEST array automatically decodes them
  - ... but occasionally the encoded version does pop up (e.g. in Firebug)

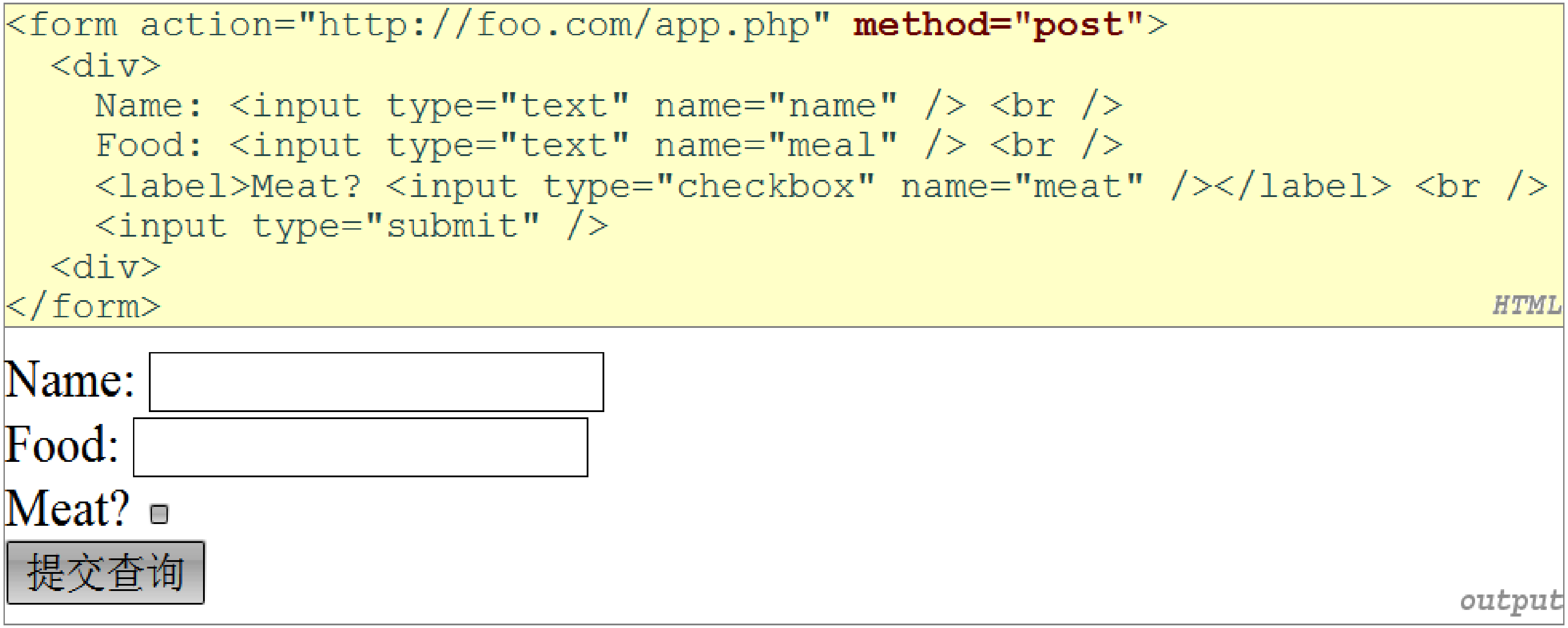
Submitting Data to a Web Server

- Though browsers mostly retrieve data, sometimes you want to submit data to a server
  - Hotmail: Send a message
  - Flickr: Upload a photo
  - Google Calendar: Create an appointment
- The data is sent in HTTP requests to the server
  - With **HTML forms**
  - With **predefined URLs**
  - With **Ajax** (seen later)
- The data is placed into the request as parameters

HTTP 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
- **POST** : submits data to a web server and retrieves the server's response if the request has parameters, they are embedded in the request's HTTP packet, not the URL
  - For submitting data, a POST request is more appropriate than a GET **GET** requests **embed their parameters** in their URLs
  - URLs are **limited in length** (~ 1024 characters)
  - URLs cannot contain **special characters** without encoding
  - Private data in a URL can be seen

Form POST Example



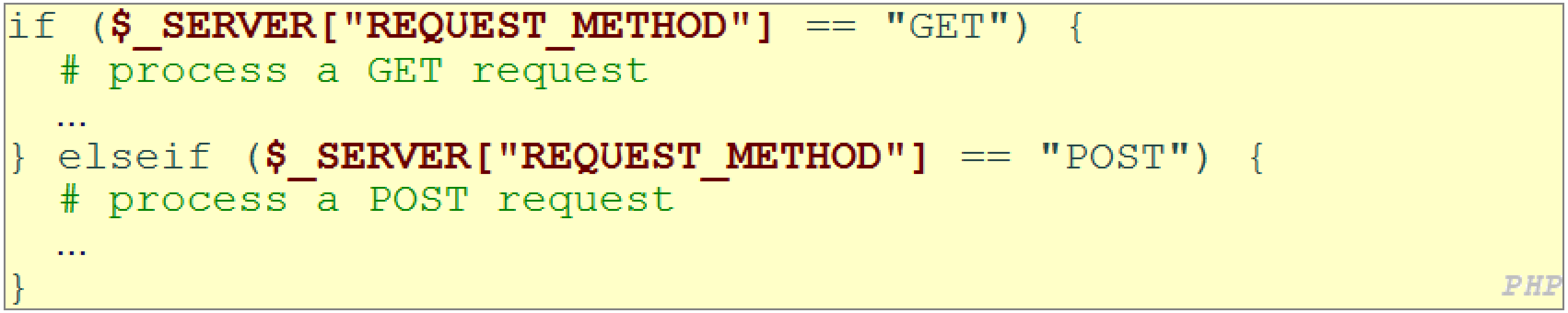
```
<form action="http://foo.com/app.php" method="post">
  <div>
    Name: <input type="text" name="name" /> <br />
    Food: <input type="text" name="meal" /> <br />
    <label>Meat? <input type="checkbox" name="meat" /></label> <br />
    <input type="submit" />
  <div>
</form>
```

Name:

Food:

Meat? □

GET or POST?



```
if ($_SERVER["REQUEST_METHOD"] == "GET") {
  # process a GET request
  ...
} elseif ($_SERVER["REQUEST_METHOD"] == "POST") {
  # process a POST request
  ...
}
```

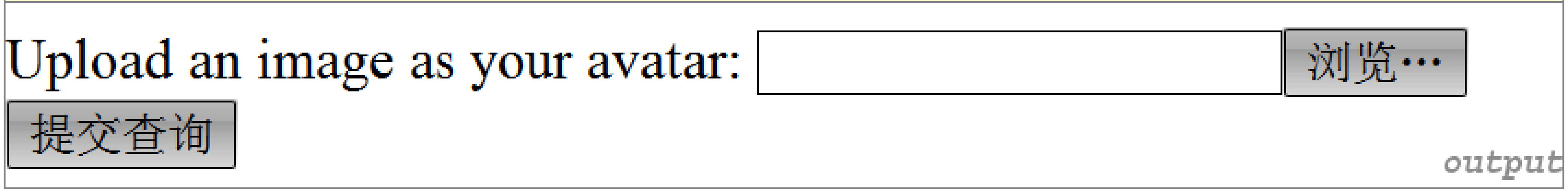
PHP

- Some PHP pages process both GET and POST requests
- To find out which kind of request we are currently processing, look at the —
  - Global **$_SERVER** array's "**REQUEST_METHOD**" element

Uploading Files

```
<form action="http://webster.cs.washington.edu/params.php"
      method="post" enctype="multipart/form-data">
  Upload an image as your avatar:
  <input type="file" name="avatar" />
  <input type="submit" />
</form>
```

HTML



Upload an image as your avatar:

output

- Add a file upload to your form as an input tag with **type** of **file**
- Must also set the **enctype** attribute of the form
- What’s going on exactly when you click the submit button?

Outline

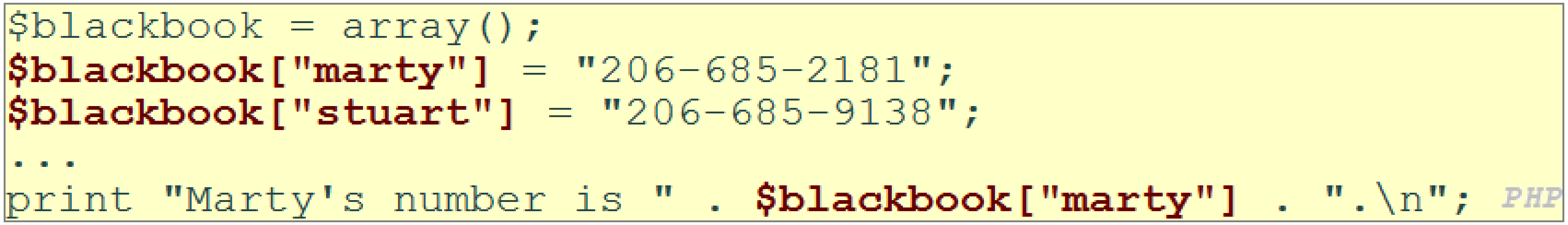
- More HTML Forms
- Submitting Data
- **Processing Form data in PHP**

"Superglobal" Arrays

Array	Description
$_GET, $_POST	parameters passed to GET and POST requests
$_REQUEST	parameters passed to any type of request
$_SERVER, $_ENV	information about the web server
$_FILES	files uploaded with the web request
$_SESSION, $_COOKIE	"cookies" used to identify the user (seen later)

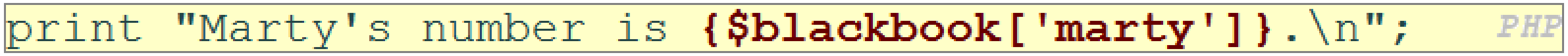
- PHP **superglobal** arrays (global variables) contain information about the current request, server, etc.
- These are special kinds of arrays called **associative arrays**.

Associative Arrays



```php
$blackbook = array();
$blackbook["marty"] = "206-685-2181";
$blackbook["stuart"] = "206-685-9138";
...
print "Marty's number is " . $blackbook["marty"] . ".\n";
```

- **Associative array** (a.k.a. **map**, **dictionary**, **hash table**) : uses **non-integer indexes** associates a particular index “key” with a value
  - i.e. key "marty" maps to value "206-685-2181"
- Syntax for embedding an associative array element in interpreted string:



```php
print "Marty's number is {$blackbook['marty']}.\n";
```

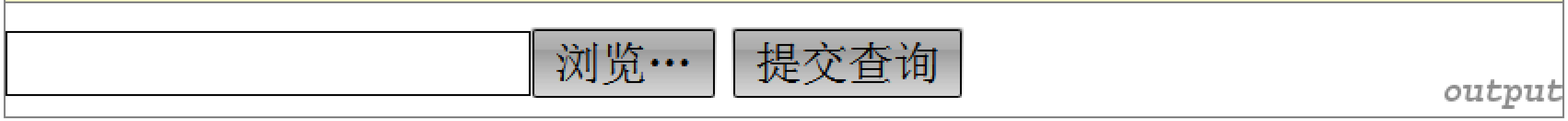
Processing an Uploaded File in PHP

- Uploaded files are placed into global array **$_FILES**, **NOT $_REQUEST**
- Each element of **$_FILES** is itself an associative array, containing:
  - **name** : the local filename that the user uploaded
  - **type** : the MIME type of data that was uploaded, such as image/jpeg
  - **size** : file's size in bytes
  - **tmp_name** : a filename where PHP has temporarily saved the uploaded file
    - To permanently store the file, move it from this location into some other file

Uploading Details

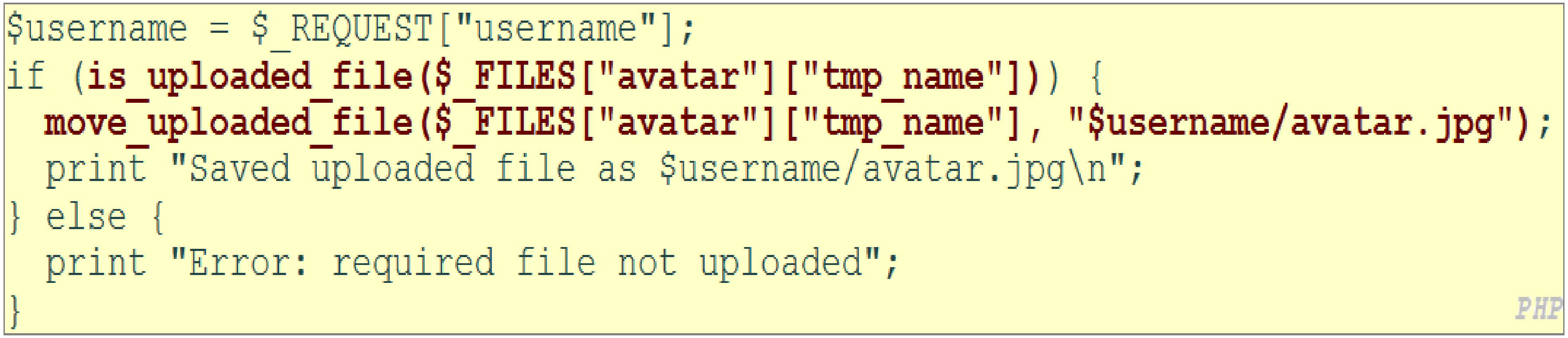
```html
<input type="file" name="avatar" />
```

HTML



- Example: if you upload borat.jpg as a parameter named avatar,
  - $_FILES["avatar"]["name"] will be "borat.jpg"
  - $_FILES["avatar"]["type"] will be "image/jpeg"
  - $_FILES["avatar"]["tmp_name"] will be something like "/var/tmp/phpZtR4TI"

Processing Uploaded File, Example



```
$username = $_REQUEST["username"];
if (is_uploaded_file($_FILES["avatar"]["tmp_name"])) {
  move_uploaded_file($_FILES["avatar"]["tmp_name"], "$username/avatar.jpg");
  print "Saved uploaded file as $username/avatar.jpg\n";
} else {
  print "Error: required file not uploaded";
}
```

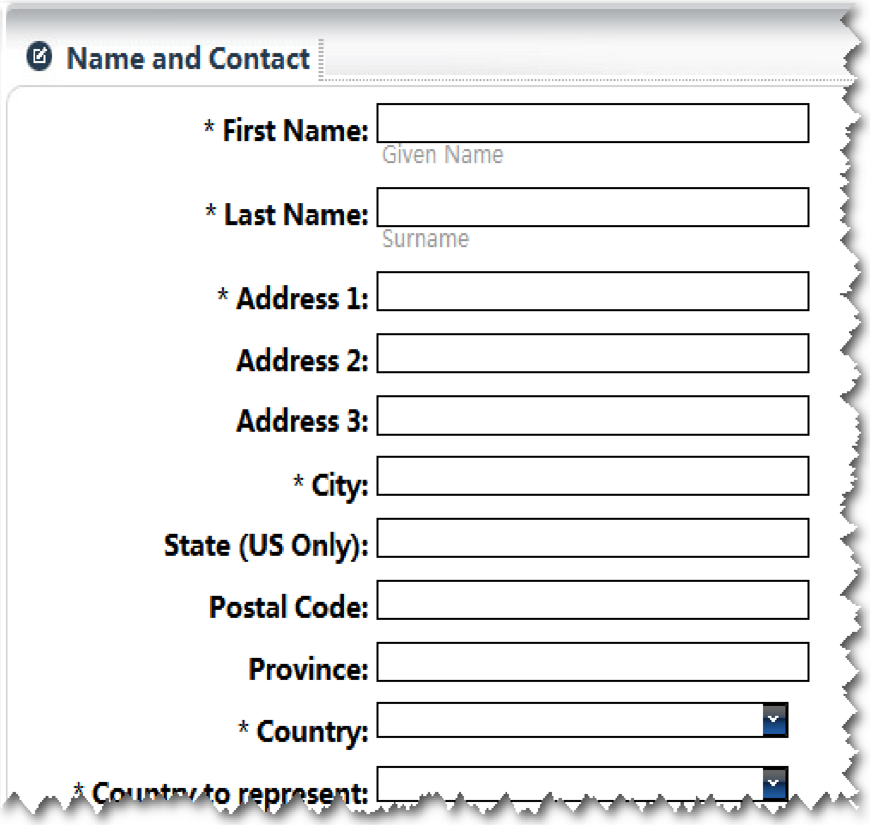
- Functions for dealing with uploaded files:
  - **is_uploaded_file**(*filename*)
    returns TRUE if the given filename was uploaded by the user
  - **move_uploaded_file**(*from*, *to*)
    moves from a temporary file location to a more permanent file
- Proper idiom: check is_uploaded_file, then do move_uploaded_file

Summary

- More HTML Forms
  - reset, hidden
  - fieldset, legend
- Submitting Data
  - Value (radio, option)
  - URL-encoding
  - GET vs. POST
  - Uploading files
- Processing Form data in PHP
  - Superglobal arrays
  - Process uploaded files

Exercises

- Write a php page to mimic the registration page of Topcoder
  - Get initial html from https://www.topcoder.com/reg/
  - Convert it to a PHP page, which **only** shows data you submitted at the top of this page when your first name is “Eric”, otherwise it shows the message “Strangers Forbidden!” instead



Name and Contact

* First Name: Given Name

* Last Name: Surname

* Address 1:

Address 2:

Address 3:

* City:

State (US Only):

Postal Code:

Province:

* Country:

* Country to represent:

Further Readings

- W3C XHTML Controls: http://www.w3.org/TR/html4/interact/forms.html
- W3Schools HTML forms tutorial http://www.w3schools.com/html/html_forms.asp
- PHP home page: http://www.php.net/
- W3Schools PHP tutorial: http://www.w3schools.com/PHP/
- Practical PHP Programming: http://hudzilla.org/phpwiki/
- PHP Cookbook: http://commons.oreilly.com/wiki/index.php/PHP_Cookbook
- CSS fun with forms: http://www.picment.com/articles/css/funwithforms/

Thank you!