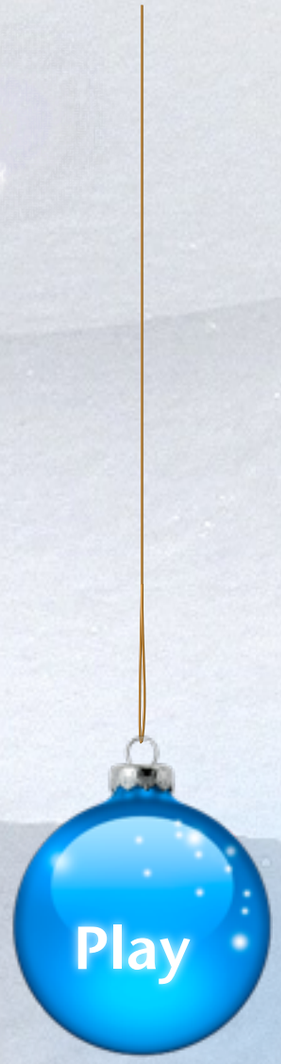
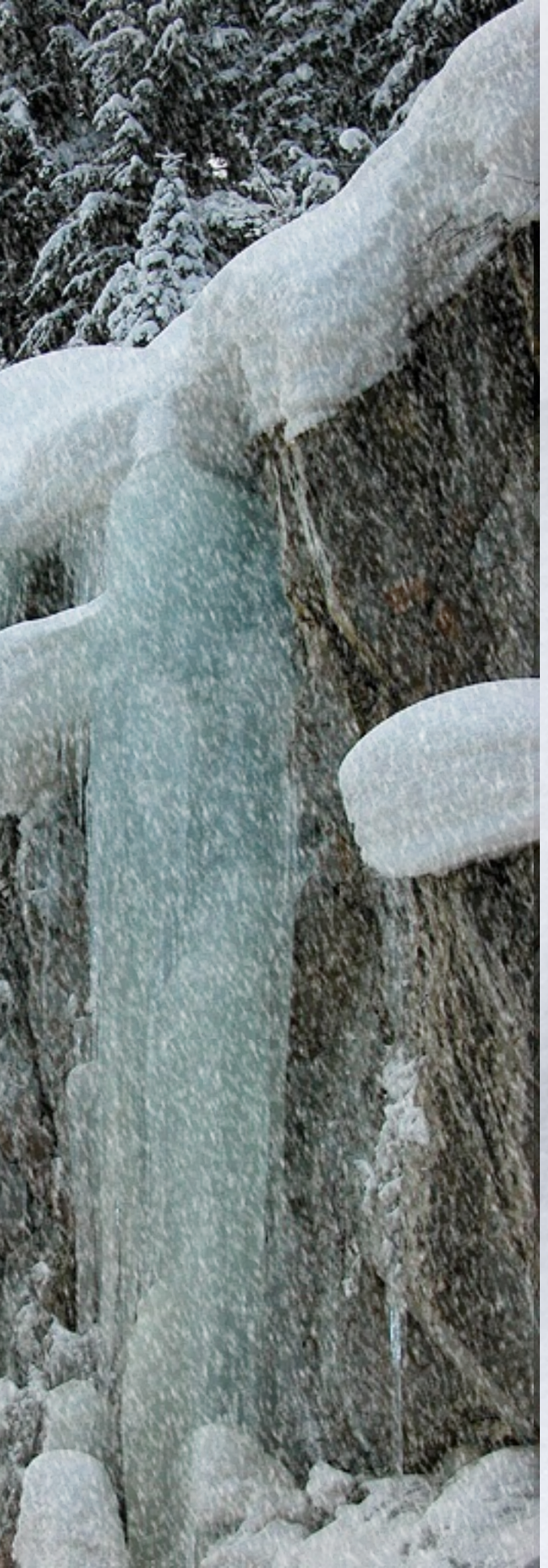
# Lecture 8
# More PHP for Server Side Programming





# Outline

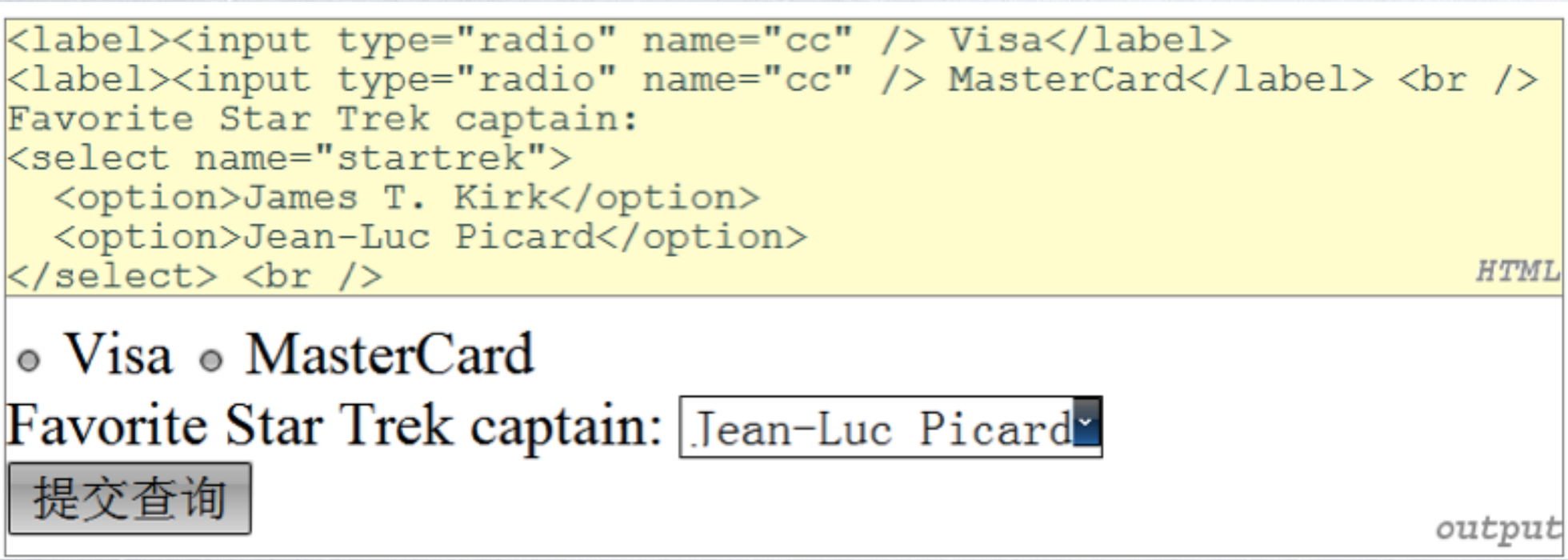
- Submitting Data
- Processing Form data in PHP
- Form Validations
- Regular Expression
- Object-Oriented PHP

# Problems with submitting data

- this form submits to our handy params.php tester page
- the form may look correct, but when you submit it...

```
<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




output

[cc] => on, [startrek] => Jean-Luc Picard

# The value attribute

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

```
<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 />
```

HTML

Visa MasterCard

Favorite Star Trek captain: James T. Kirk

提交查询

output

# URL-encoding

- certain characters are not allowed in URL query parameters:
  - examples: " ", "/", "=", "&"
- when passing a parameter, it is URL-encoded(reference table)
  - "Marty's cool!?" → "Marty%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 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

```HTML
<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>
```

output

Name:

Food:

Meat?

提交查询

# GET or POST?

- 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

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

# Uploading files

- add a file upload to your form as an input tag with typeof file
- must also set the enctype attribute of the form
- what's going on exactly when you click the submit button?

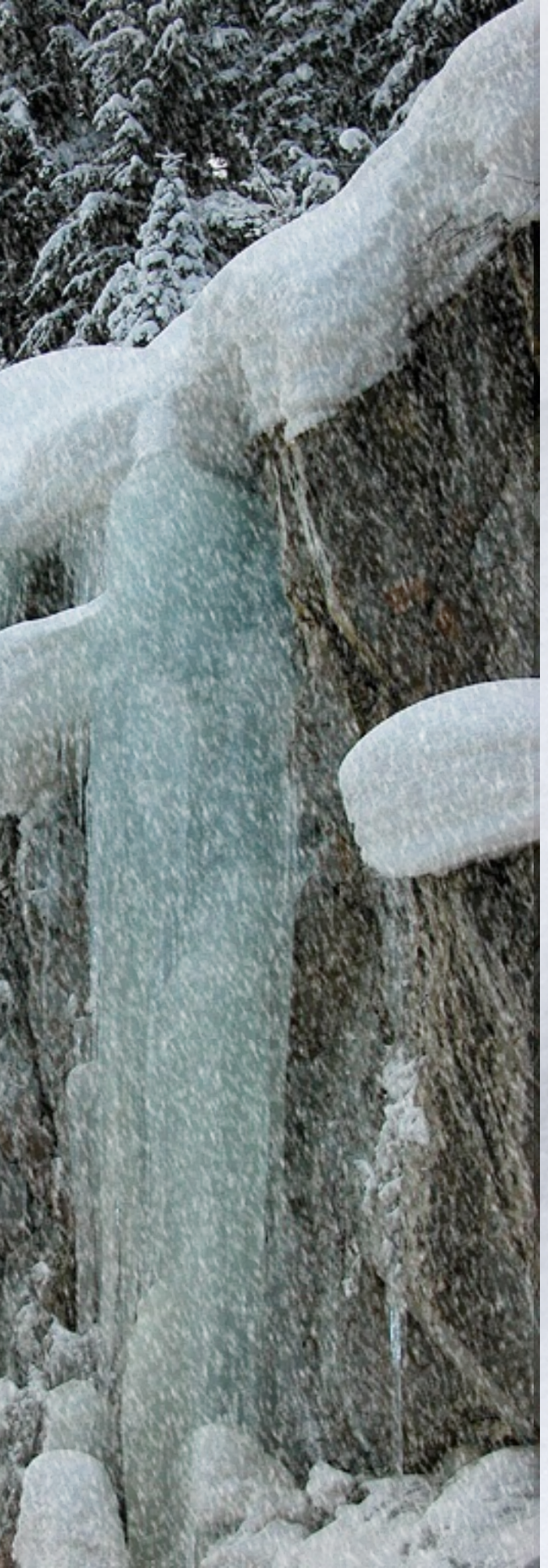
```
<form action="http://webster.cs.washington.edu/params.php"
      method="post" enctype="multipart/form-data">
  Upload an image as your avatar:
  <input type="file" name="avatar" />
  <input type="submit" />
</form>
```

HTML

Upload an image as your avatar: 浏览…

提交查询

output



# Outline

- Submitting Data
- Processing Form data in PHP
- Form Validations
- Regular Expression
- Object-Oriented PHP

# Methods of Processing Form data in PHP

- php.ini, register_globals=On(不建议,会有诸多不安全可能性)
  - variables_order，EGPCS（Environment，GET，POST，Cookie，Server）
- $HTTP_GET_VARS、$HTTP_POST_VARS (不建议)
- $_POST、$_GET
- import_request_variables()
  - import_request_variables("gp", “rvar_")
  - import_request_variables("gp", "")

# "Superglobal" arrays

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

# 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

- 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"

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

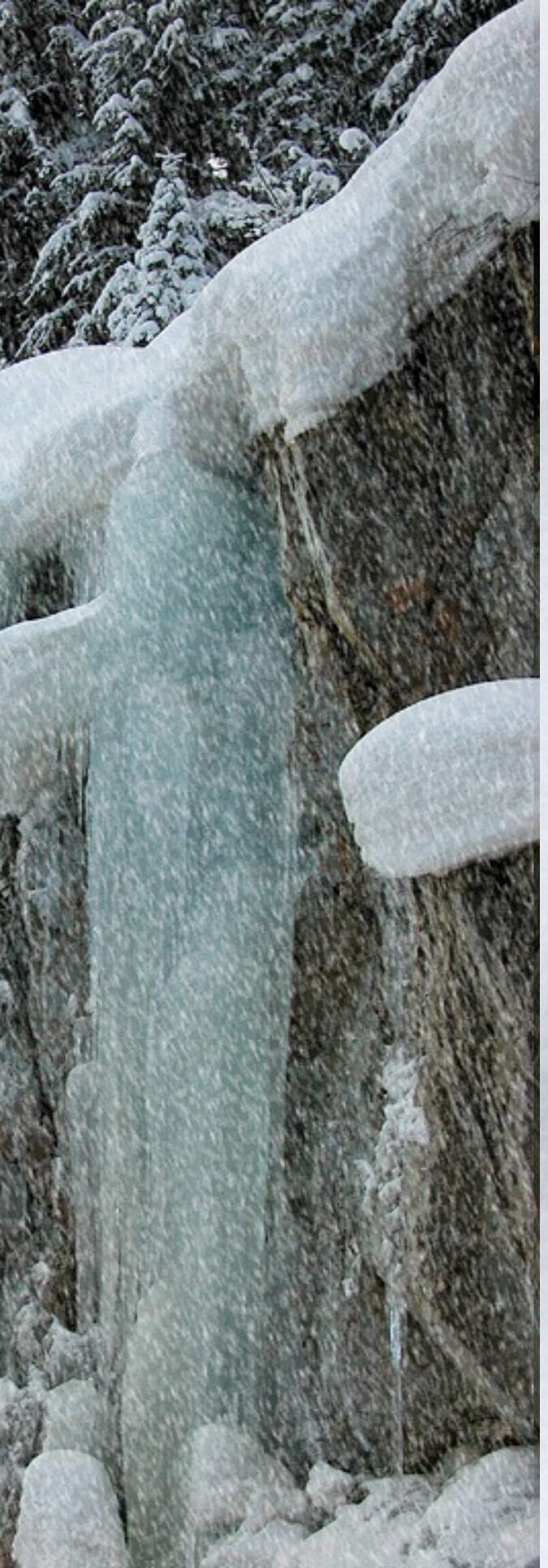
浏览… 提交查询

output

# Processing uploaded file, example

- 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

```php
$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";
}
```



# Outline

- Submitting Data
- Processing Form data in PHP
- Form Validations
- Regular Expression
- Object-Oriented PHP

# What is form validation?

- validation: ensuring that form's values are correct
- 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 6-digit integer)
  - ensuring that values fit together (user types email twice, and the two must match)

# Client vs. serve-side validation

- Validation can be performed:
  - client-side (before the form is submitted)
    - can lead to a better user experience, but not secure (why not?)
  - server-side (in PHP code, after the form is submitted)
    - needed for truly secure validation, but slower
  - both
    - best mix of convenience and security, but requires most effort to program

# An example to be validated

- Let's validate this form's data on the server...

```
<form action="http://foo.com/foo.php" method="get">
  <div>
    City:  <input name="city" /> <br />
    State: <input name="state" size="2" maxlength="2" /> <br />
    ZIP:   <input name="zip" size="5" maxlength="5" /> <br />
    <input type="submit" />
  </div>
</form>
```

HTML

City:

State:

ZIP:

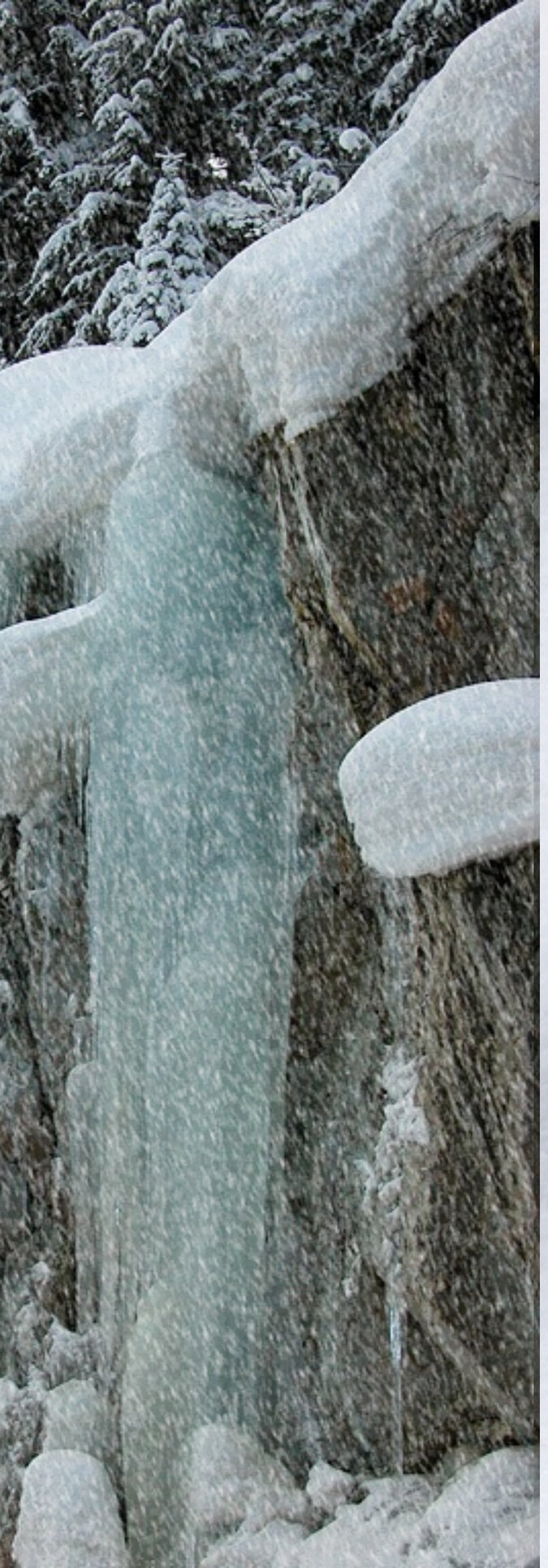
提交查询

output

# Basic server-side validation code

- basic idea: examine parameter values, and if they are bad, show an error message and abort
- validation code can take a lot of time/lines to write
  - How do you test for integers vs. real numbers vs. strings?
  - How do you test for a valid credit card number?
  - How do you test that a person's name has a middle initial?
  - (How do you test whether a given string matches a particular complex format?)

```php
$city  = $_REQUEST["city"];
$state = $_REQUEST["state"];
$zip   = $_REQUEST["zip"];
if (!$city || strlen($state) != 2 || strlen($zip) != 5) {
  ?>
  <h2>Error, invalid city/state submitted.</h2>
  <?php
}
```



# Outline

- Submitting Data
- Processing Form data in PHP
- Form Validations
- Regular Expression
- Object-Oriented PHP

# What is a regular expression?

- regular expression ("regex"): a description of a pattern of text
  - can test whether a string matches the expression's pattern
  - can use a regex to search/replace characters in a string
- regular expressions are extremely powerful but tough to read(the above regular expression matches email addresses)
- regular expressions occur in many places:
  - Java: Scanner, String's split method
  - supported by PHP, JavaScript, and other languages
  - many text editors (Notepad++, TextPad) allow regexes in search/replace

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

# Basic regular expressions

- in PHP, regexes are strings that begin and end with /
- the simplest regexes simply match a particular substring
- the above regular expression matches any string containing "abc":
  - YES: "abc", "abcdef", "defabc", ".=.abc.=.", ...
  - NO: "fedcba", "ab c", "PHP", ...

```
"/abc/"
```

# Wildcards: .

- A dot . matches any character except a \n line break
  - "/.oo.y/" matches "Doocy", "goofy", "LooNy", ...
- A trailing i at the end of a regex (after the closing /) signifies a case-insensitive match
  - "/mart/i" matches "Marty Stepp", "smart fellow", "WALMART", ...

# 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"
- ^ matches the beginning of a line; $ the end
  - "/^<!--$/" matches a line that consists entirely of "<!--"
- \ 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
  - "/abc*/" matches "ab", "abc", "abcc", "abccc", ...
  - "/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", "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??!", ...
- What regular expression matches DNA (strings of A, C, G, or T)?

# 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 ( \r, \t, \n, etc.); \S any non-whitespace

- What regular expression matches dollar amounts of at least $1000.00 ?

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

# Regular expressions in PHP

- regex syntax: strings that begin and end with /, such as "/[AEIOU]+/"

| function | description |
|---|---|
| preg_match(regex, string) | returns TRUE if string matches regex |
| preg_replace(regex, replacement, string) | returns a new string with all substrings that match regex replaced by replacement |
| preg_split(regex, string) | returns an array of strings from given string broken apart using the given regex as the delimiter (similar to explode but more powerful) |

# Regular expression example

- notice how \ must be escaped to \\

```php
# replace vowels with stars
$str = "the quick    brown         fox";

$str = preg_replace("/[aeiou]/", "*", $str);
                          # "th* q**ck    br*wn         f*x"

# break apart into words
$words = preg_split("/[ ]+/", $str);
                          # ("th*", "q**ck", "br*wn", "f*x")

# capitalize words that had 2+ consecutive vowels
for ($i = 0; $i < count($words); $i++) {
  if (preg_match("/\\*{2,}/", $words[$i])) {
    $words[$i] = strtoupper($words[$i]);
  }
}                         # ("th*", "Q**CK", "br*wn", "f*x")
```

Perl或者PHP 的正则中，遇到元字符需要转义时，直接把元字符放在\Q\E之间即可。

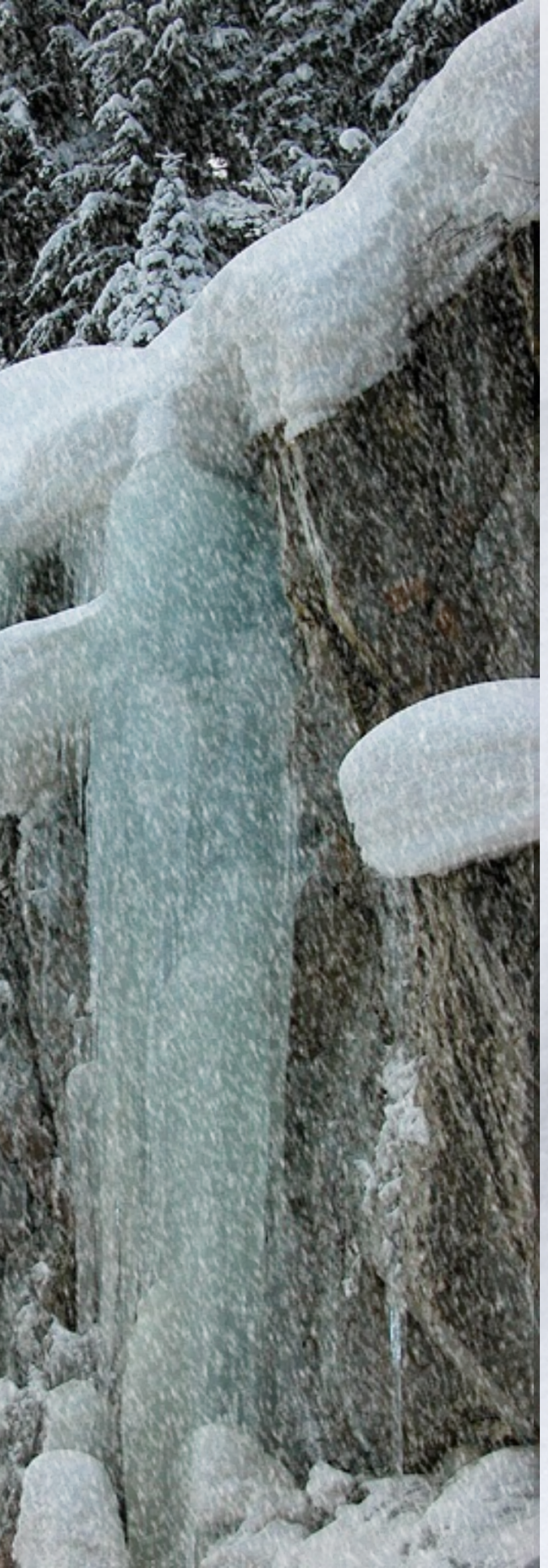
# PHP form validation tips

- using preg_match and well-chosen regexes allows you to quickly validate parameters
- interestingly, we often DON'T want to give a very descriptive error message here (why?)

```php
$state = $_REQUEST["state"];
if (!preg_match("/[A-Z]{2}/", $state)) {
?>

  <h2>Error, invalid state submitted.</h2>

<?php
}
```



# Outline

- Submitting Data
- Processing Form data in PHP
- Form Validations
- Regular Expression
- Object-Oriented PHP

# Why OOP?

- PHP is a primarily procedural language
- small programs are easily written without adding any classes or objects
- larger programs, however, become cluttered with so many disorganized functions
- grouping related data and behavior into objects helps manage size and complexity

# Constructing and using objects

```php
# construct an object
$name = new ClassName(parameters);

# access an object's field (if the field is public)
$name->fieldName

# call an object's method
$name->methodName(parameters);
```

```php
$zip = new ZipArchive();
$zip->open("moviefiles.zip");
$zip->extractTo("images/");
$zip->close();
```

- the above code unzips a file
- test whether a class is installed with class_exists

# Object example: Fetch file from web

- PHP's HttpRequest object can fetch a document from the web

```php
# create an HTTP request to fetch student.php
$req = new HttpRequest("student.php", HttpRequest::METH_GET);
$params = array("first_name" => $fname, "last_name" => $lname);
$req->addPostFields($params);

# send request and examine result
$req->send();
$http_result_code = $req->getResponseCode();    # 200 means OK
print "$http_result_code\n";
print $req->getResponseBody();
```

# Class declaration syntax

- inside a constructor or method, refer to the current object as $this

```php
class ClassName {
  # fields - data inside each object
  public $name;     # public field
  private $name;    # private field

  # constructor - initializes each object's state
  public function __construct(parameters) {
    statement(s);
  }

  # method - behavior of each object
  public function name(parameters) {
    statements;
  }
}
```

# Class example

```php
<?php
class Point {
  public $x;
  public $y;

  # equivalent of a Java constructor
  public function __construct($x, $y) {
    $this->x = $x;
    $this->y = $y;
  }

  public function distance($p) {
    $dx = $this->x - $p->x;
    $dy = $this->y - $p->y;
    return sqrt($dx * $dx + $dy * $dy);
  }

  # equivalent of Java's toString method
  public function __toString() {
    return "(" . $this->x . ", " . $this->y . ")";
  }
}
?>
```

PHP

# Class usage example

- $p1 and $p2 are references to Point objects

```php
<?php
# this code could go into a file named use_point.php
include("Point.php");

$p1 = new Point(0, 0);
$p2 = new Point(4, 3);
print "Distance between $p1 and $p2 is " . $p1->distance($p2) . "\n\n";

var_dump($p2);   # var_dump prints detailed state of an object
?>
```

PHP

```
Distance between (0, 0) and (4, 3) is 5

object(Point)[2]
  public 'x' => int 4
  public 'y' => int 3
```

PHP

# Basic inheritance

- the given class will inherit all data and behavior from ClassName

```
class ClassName extends ClassName {
  ...
}
```
PHP

```
class Point3D extends Point {
  public $z;

  public function __construct($x, $y, $z) {
    parent::__construct($x, $y);
    $this->z = $z;
  }

  ...
}
```
PHP

# Static methods, fields, and constants

- static fields/methods are shared throughout a class rather than replicated in every object

```
static $name = value;      # declaring a static field
const $name = value;       # declaring a static constant
```
PHP

```
# declaring a static method
public static function name(parameters) {
   statements;
}
```
PHP

```
ClassName::methodName(parameters);
self::methodName(parameters);
```

# Abstract classes and interfaces

- interfaces are supertypes that specify method headers without implementations
  - cannot be instantiated; cannot contain function bodies or fields
  - enables polymorphism between subtypes without sharing implementation code
- abstract classes are like interfaces, but you can specify fields, constructors, methods
  - also cannot be instantiated; enables polymorphism with sharing of implementation code

# Reading materials

- PHP Regular Expression tutorials:
- http://www.phpro.org/tutorials/Introduction-to-PHP-Regex.html
- http://www.webcheatsheet.com/php/regular_expressions.php
- PHP Regular Expression examples: http://www.roscripts.com/PHP_regular_expressions_examples-136.html
- 精通正则表达式:第3版。(美)Jeffrey E.F.Friedl

# Thanks!!!