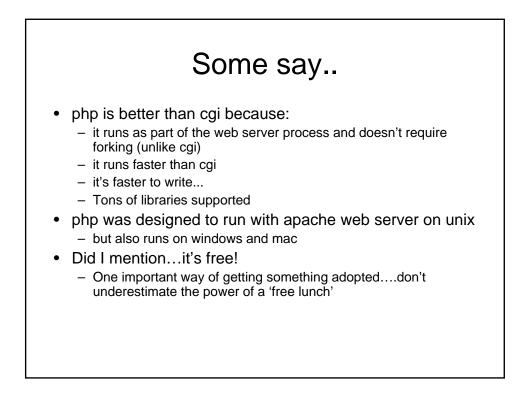


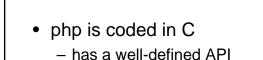
Php.net developed 1994-1995, first as collection of perl scripts and then own interpreter originally created as "Personal Home Page" tools, by Rasmus Lerdorf First, was a quick tool for embedding sql queries in a web page (v1.0)

- then structured code was added (v2.0), but with a buggy language parser
- official release (v3.0) fixed parser bugs June 1998 introduced object oriented ideas
- V4 more object, and passing variables in the system modified
- V5 new engine, many fixes etc
- Early as Jan 1999, 100,000 web pages were using php!!! Much higher now!



LAMP

- Linux
- Apache
- Mysql
- Perl/PHP/Python



- extensible
- the way it runs:
 - a php engine is installed as part of a web server
 - the engine runs the php script and produces html, which gets passed back to the browser
 - So user never sees the php code (if done right)

3 different ways !

- hello.php (plain php)
- hello2.php (php embedded in html)
- hello3.php (uses <?php start tag)



Hello2.php

<html>

```
<body bgcolor=#000000 text=#ffffff>
```

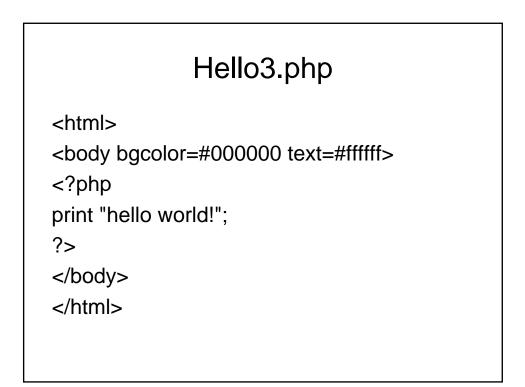
<?

print "hello world!";

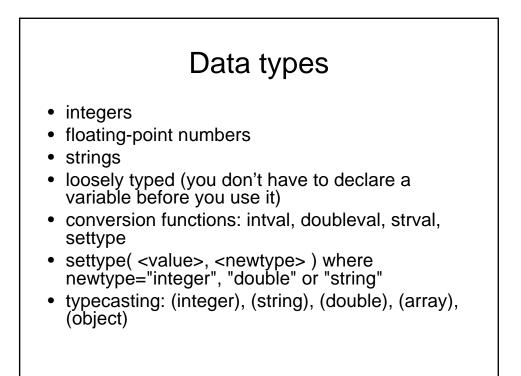
?>

</body>

</html>



basics php start and end tags: <? ... ?> also: <?php ... ?> semi-colon ends a statement (like C) string constants surrounded by quotes (") or (') you can embed multiple php blocks in a single html file variable names are preceded by dollar sign (\$) user input is through html forms the language is case-sensitive, but calls to built-in functions are not - Any ideas why ????? identifiers are made of letters, numbers and underscore (_); and cannot begin with a number expressions are similar to C



operators

- mathematical: +, -, *, /, %, ++, --
- relational: <, >, <=, >=, ==, !=
- logical: AND, &&, OR, ||, XOR, !
- bitwise: &, |, ^ (xor), ~ (ones complement), >>, <<
- assignment: =, =, -=, *=, /=,
- other:

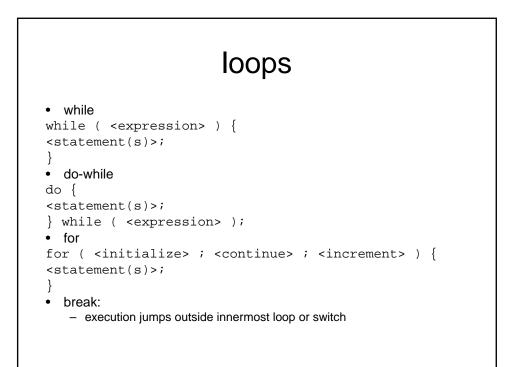
 - . concatenate
 -> references a class method or property
 - => initialize array element index

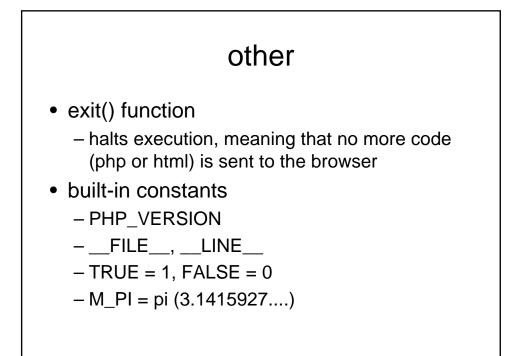
Conditionals

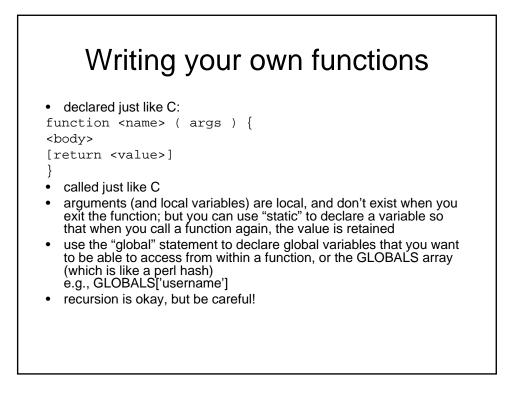
```
• if/elseif/else:
if ( <expression1> ) {
<statement(s)>
}
elseif ( <expression2> ) {
<statement(s)>
}
else {
<statement(s)>
}
```

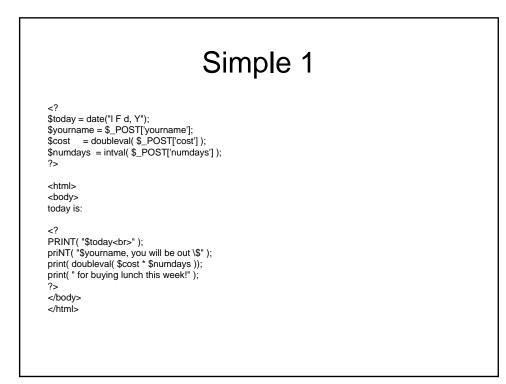
Conditional II

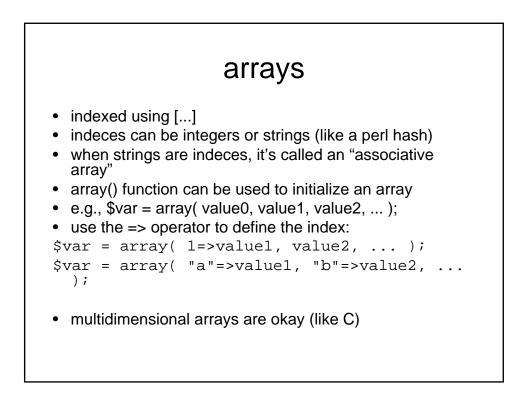
```
• tertiary operator:
  <conditional-expression> ?
  <true-expression> : <false-expression>;
• switch:
  switch( <root-expression> ) {
   case <case-expression>:
    <statement(s)>;
   break;
   default:
    <statement(s)>;
   break;
}
```



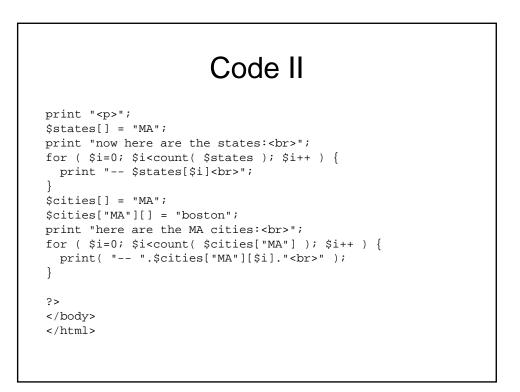


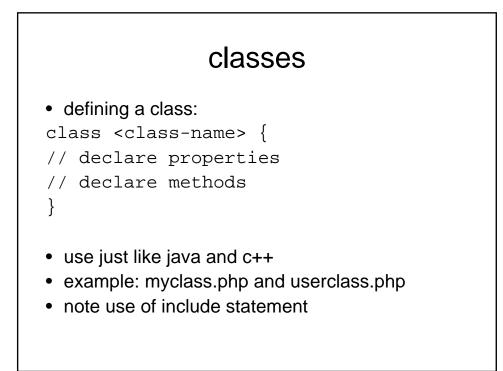






```
code
<html>
<body bgcolor=#ffffff>
<?
$states = array( "CA","NY" );
print "here are the states:<br>";
for ( $i=0; $i<count( $states ); $i++ ) {</pre>
print "-- $states[$i]<br>";
}
print "";
$cities = array( "CA"=>array( "san francisco","los angeles" ),
                "NY"=>array( "new york", "albany", "buffalo" ));
print "here are the CA cities:<br>";
for ( $i=0; $i<count( $cities["CA"] ); $i++ ) {</pre>
 print( "-- ".$cities["CA"][$i]."<br>" );
}
print "here are the NY cities:<br>";
for ( $i=0; $i<count( $cities["NY"] ); $i++ ) {</pre>
print( "-- ".$cities["NY"][$i]."<br>br>" );
}
```



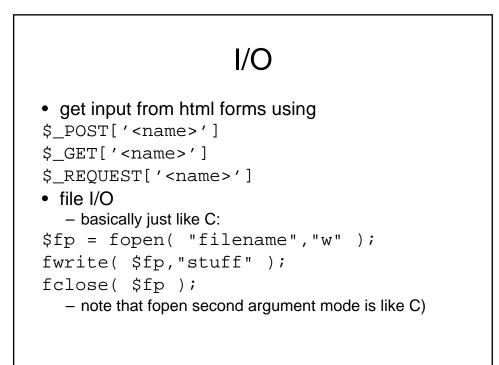


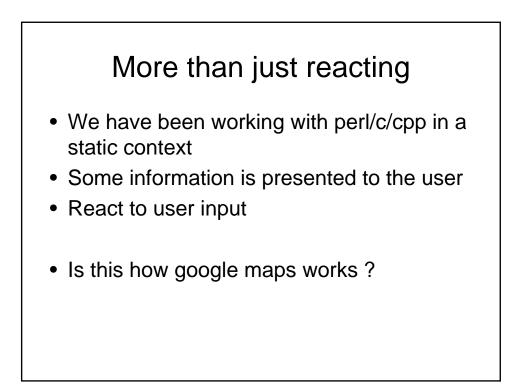


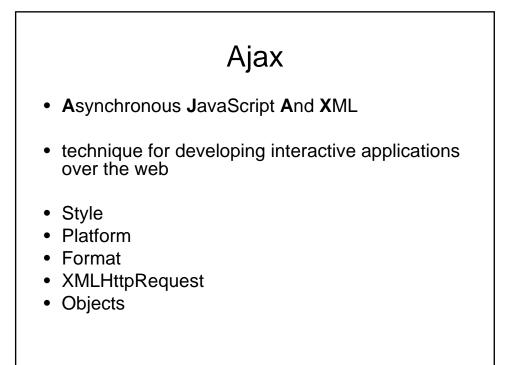
myclass.php

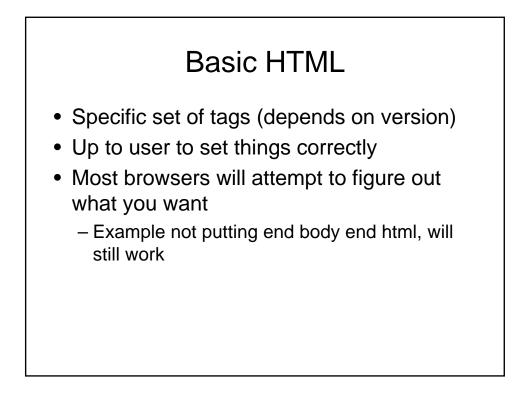
```
<html>
<body>
<?
include "userclass.php";
$currentuser = new user;
$currentuser->init( "yaddi","cat" );
print( "name = ".$currentuser->name."<br>" );
print( "last login = ".$currentuser->getLastLogin() );
?>
</body>
</html>
```

```
<?php
class Car {
 public $miles; // variable that can be accessed outside the class
private $mpg; // variable that can only be accessed within the class
protected $mph; // variable that can only be accessed from within the class, and
 // from any inherited child classes
 public function __construct($param) { // constructor is called when object "Car" is
 created
$this->miles = $param;
 public function start() {
 // starts the car...
 }
а
 public function stop() {
  // stops the car...
 }
 public function getMpg() {
 return $this->mpg;
 }
}
$car = new Car($param);
echocar->miles; // echos the value of the property "miles" of the class "Car"
?>
```



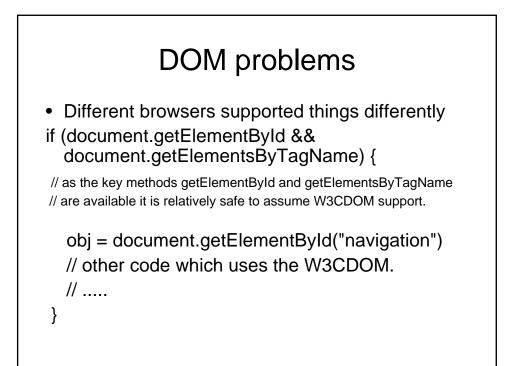






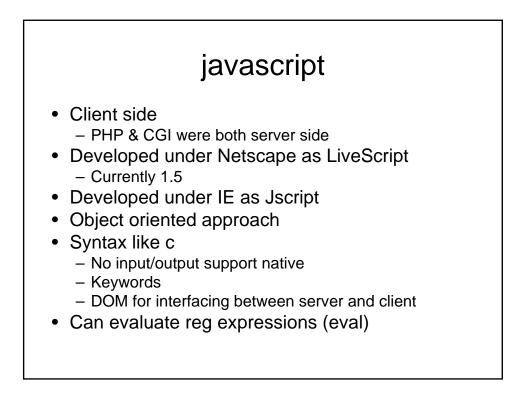
Advanced HTML

- CSS
 - Cascading style sheets
 - Define format of the WebPages
 - Central location of style
 - With a few clicks can completely change thousands of WebPages.
- DOM
 - Document object model
 - Formal data structure to represent web based documents/information
- Client side scripting



Examples

- http://www.dynamicdrive.com/dynamicinde x12/pong/pong.htm
- http://www.dynamicdrive.com/dynamicinde x4/butterfly.htm

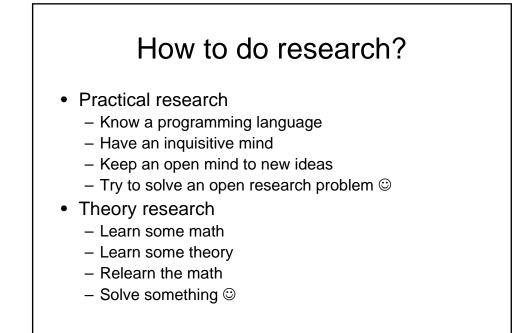


Javascript

- Heavy use of defined functions
 - Example: MouseOver
- Need to adopt to specific browser if doing anything fancy
- Adobe
 - Support javascript in pdf
- MAC
 - Dashboard widgets

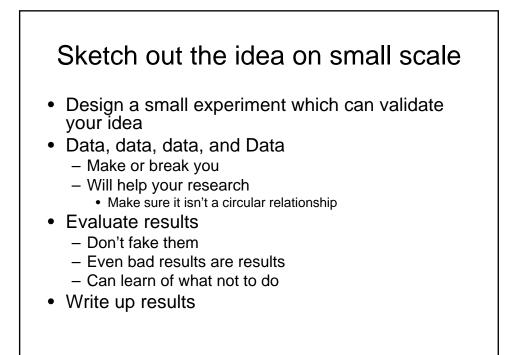
Programming

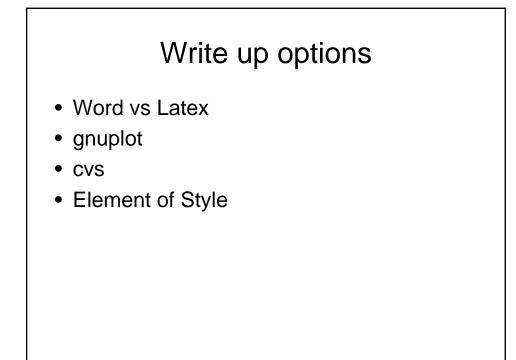
- You need to learn on your own
- Many good books/websites
- Most of the time .js file if not in html
- Powerful example:
 - Thunderbird/firefox
- Get good debugger



Where to start?

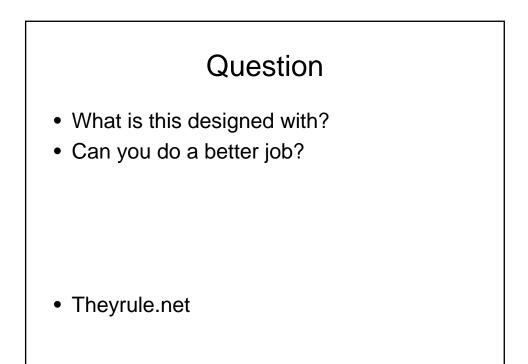
- 1. Need an idea
- See if anyone's done it or tried it in your way
 - 1. Citeseer (citeseer.ist.psu.edu)
 - 2. Google
 - 3. Appropriate Faculty/Researcher
 - 4. Google groups





In the real world

- 1. Keep it simple
 - 1. Don't re-invent the wheel
 - 2. Design first
 - 3. Even with fancy blinking lights, a bad idea is still a bad idea (but with bad taste)
- 2. Incremental testing
 - 1. Recognize when the bug is your fault
 - 2. See if others have faced it too
- 3. Make sure version 1 works on most popular browsers



Bottom line

- We've covered a lot this semester
 - Some of it was fun
 - Some of it was hard work (ok most)
 - Some of it was frustrating.
- BUT
 - You have lots of tools
 - Have an idea of where to start when dealing with programming projects

Important lessons for learning new languages

- CS is not meant to be a trade school
- Language isn't important...things change
- · Ideas and design are more important
- Lessons:
 - Choose correct environment
 - Choose correct tools
 - Make sure to test out ideas...might be someone else's fault (program think)
 - Enjoy what you are doing

Important

• To get the most out of a language find comfortable programming environment

- Emacs color files
- Eclipse
- Others , see
 - www.freebyte.com/programming/cpp/

Review time

- Focus on C
- Focus on CPP
- Shell programming stuff
- Idea of PHP
- Perl
- Review the labs

Word list

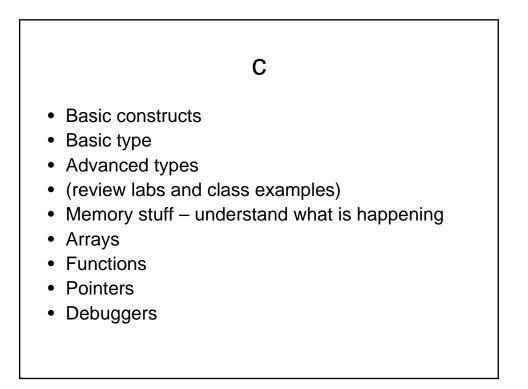
- Compiling
- Linking
- Reference parameter
- Variable scope
- Stdio.h
- Stdlib.h
- cout
- cast
- Inline
- Linked list

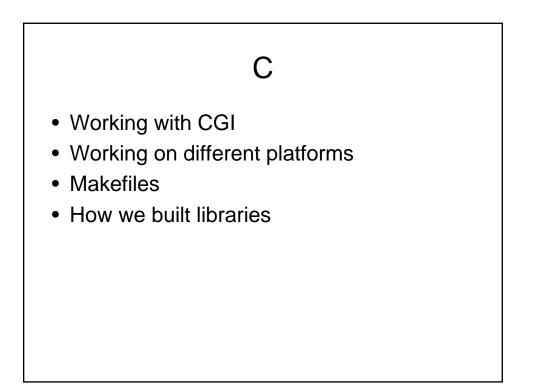
- Preprocessor
- Typedef
- Struct
- Pointer
- Void pointer
- . Vs ->
- Function pointer
- Reference
- const
- malloc

Word list II

- Huffman
- getopt
- constructor
- destructor
- iostream
- overloading
- extern
- private
- Public
- GDB

- Cgi
- GET/POST
- overload
- overriding
- Template
- This
- Friend class
- New/delete
- virtual





C++

- Basic language
- Difference to c
- Classes
- Permissions
- new/free memory allocations
- Inheritance and polymorphism
- Keywords
- Working with files....



- You've done most of the work for the course, the exam is just to make sure you remember the important concepts
- Will be posted online
- Couple Definitions
- 2 code checking question
- Shell code question
- C++ class manip question
- Small CGI question

Thinking question

- Say you are writing code which uses a random number generator....
- What is important to know about it ?
- How can your code be affected ?
- If you crash, how to reconstruct events, since based on random numbers ??

