

PHP (cont.)

Functions

- Definition
 - `function name(arguments) {
 ...
}`
 - syntax
 - `name` as for a variable, but without \$
 - `arguments` as for variables
 - to return a value use
 - `return expression;`
- Call
 - `name(arguments)`
 - the number of arguments must match

Function Examples

```

• function copyright() {
    echo "<p>Copyright ICS 2010</p>";
}

• function toFahrenheit($celsius) {
    return $celsius * 9 / 5 + 32;
}

• function loanPayment($amount,$interest,$years) {
    $power = pow($interest / 12 + 1, $years * 12)
    return $amount * $interest * $power / ($power - 1);
}

```

Returning Multiple Values

- Multiple values can be returned as an array
 - `return array(values-list);`
 - e.g.,
 - `function today() {
 return array('We', '4', '1', '2010', 'fools day');
 }`
- To assign the returned values to variables, use `list()`
 - e.g.,
 - `list ($weekday, $month, $day, $year, $event) = today();`

Default Argument

- A value can be assigned to an argument
 - e.g.,
 - » `function mortgage($amount,$interest,$years=30) {
 ...
}`
- Such argument can be omitted in the function call
 - e.g.,
 - » `$monthlyPayment = mortgage(200000,0.05);`

Scope

- If a variable is defined outside of functions
 - its scope is the page where it is defined
 - it is not accessible within a function
 - unless it is declared there as `global`, e.g.
 - `function f () {
 global $var;
}`
 - avoid global variables
- If a variable is defined within a function
 - its scope is that function
 - it is not accessible outside of that function
 - an argument is such a variable, too

External files

- typically used for
 - often used code
 - CSS
 - JavaScript scripts
 - PHP functions
 - PHP with HTML snippets common to multiple pages
 - PHP classes
- `include_once(file-path)`
 - absolute or relative path
 - also `include()`, `require_once()`, `require()`
 - file needs `<?php ... ?>`
 - inserts file's text, i.e., scope rules persist

Form Handling Revisited

- One PHP file can define a form and handle it
 - use a branch to test whether the form was displayed
 - e.g., `$isset($_REQUEST['respond'])`
 - `respond` is the `name` attribute of a hidden form element:


```
- <input type="hidden" name="respond" value="set" />
```
- i.e.


```
- <?php
  if ($isset($_REQUEST['respond'])) {
    // handle form
  } else {
    // echo page with <form> and <input type="hidden">
  }
?>
```

Bugs

- Errors:
 - syntax
 - easy to correct
 - exceptions/run-time
 - harder to interpret and to find the cause
 - the cause may happen earlier
 - logical
 - hardest to find
 - may not be detected, only results are wrong

Debugging

- To make PHP display error messages
 - set `display_errors` :
 - `ini_set('display_errors', 1);`
 - see textbook Table 7.1 on p. 208 for error levels
- Rules:
 - Don't change code without clear reason & goal
 - Try to figure out the error just by reading your code
 - Explain your code to someone else
 - (even if they don't know PHP)
 - Take a break

Debugging Tips

- Make sure you
 - edit the right file
 - have saved changes
 - run PHP pages via URL
- Try
 - different browser
 - different server

Debugging Tips (cont.)

- Validate HTML
 - validator.w3.org
- Visualize
 - e.g. show table borders
- Print values of key variables
 - use `echo`, `print_r()`, `var_dump()`
- Comment out possibly offending code
- Use debugging plugins
 - Firebug in Firefox

Handling Errors

- You should handle errors gracefully
 - use `die(message)` to abort script execution
 - it can be used in a condition, e.g.,
 - `include('code.php')`
OR `die("Can't find 'code.php'");`

Error Handler

- To make a custom error handler
 - define
 - `function myHandler($number,$message,$file,$line,$vars){`
...
}
 - then install it
 - `set_error_handler('myHandler');`
 - see textbook pp.212