1. Web Data Management

152-155

Unit 5 - Control Structures & Arrays

| Notes | Activity |
| --- | --- |
| 1. Quick Links & Text References    * [Decision Control Structures](#decision) Pages 70 – 73  232 – 247    * [Arrays](#arrays) Pages 312 – 337    * [For Loops](#for) Pages 252 – 253   316 – 317   322 – 323    * [While Loops](#while) Pages 74 – 75   248 – 251    * [Validating with PHP](#validate) Pages    * [PHP Validation Tips](#valtips) Pages | Copy unit5start to htdocs |
| 1. Decision Control Structures    * Relational Operators  == != <> < <= > >= === !==    * Logical Operators ! && || (this is the order of precedence)    * Conditional operator (inline IF) (condition) ? doTrue : doFalse | Discussion only |
| * + IF statements   **if (condition) statement;**  **if (condition) {**  **block of statements**  **} //end if**  **if (condition) {**  **block**  **} else {**  **block**  **} //end if**  **if (condition) {**  **block**  **} else if (condition) {**  **block**  **} else {**  **block**  **}//end if** |  |
| * + IF with HTML embedded  1. <?php if (condition) : ?> Note the ending colon   <p>The if condition was true</p>  <?php else: ?> Note the ending colon  <p>The if condition was false</p>  <?php endif; ?>  //Option 2 (HTML in echo)  <?php  if(condition) {  echo '<p>The if condition was true</p>';  } else {  echo '<p>The if condition was false</p>';  }//end if  ?> | |
| * + NOTE: PHP Dates can be compared using relational operators: if($date1 <= $date2) is legal in PHP (but not in JavaScript)   + Note: colons in PHP with HTML can be replaced with curly brackets | |
| * + Switch Statements   switch ($switchVariable) {  case value:  block of statements  break;  case value2:  block of statements  break;  default:  block of statements  break; //optional  }//end switch |  |
| * + Switch with embedded HTML   <?php switch($color)  {  case 'green':?>    <h1 style='color:green'>Color is green</h1>    <?php break;  case 'yellow':?>    <h1 style='color:yellow'>Color is yellow</h1>    <?php break;  case 'red':?>    <h1 style='color:red'>Color is red</h1>    <?php break;  }?> | |
| 1. Arrays    * Before we discuss loops, let’s take a look at how arrays are implemented in PHP (since loops are usually used to process arrays)    * Indexed arrays   $arrayName = array(); $arrayName = []; //Starting with PHP5.4  $arrayName[0] = 'Volker';   * + PHP allows you to dynamically add and remove elements from an array     - Add element to end of array $arrayName[] = 'Fred';       * Note: no index specified     - Remove element from array unset($arrayName[0]);       * Sets element 0 to NULL   unset($arrayName);   * + - * Sets the entire array to NULL (no elements)     - To remove NULL elements from an array (reindex) $arrayName = array\_values($arrayName); |  |
| * + Associative Arrays     - Remember, the super global arrays are all associative arrays     - Remember from JavaScript, these arrays don’t use indexes, but *keys* instead   $arrayName = array();  $arrayName['age'] = 35;  $arrayName = array(  'WI' => 'Wisconsin',  'MI' => 'Michigan',  'IL' => 'Illinois');  $arrayName = [ //Starting with PHP5.4  'WI' => 'Wisconsin',  'MI' => 'Michigan',  'IL' => 'Illinois'];  unset($arrayName['WI']);   * + Array Functions     - $numElements = count($arrayName);     - isset($arrayName[$key]); //Does this key/index have a value?     - $total = array\_sum($arrayName);     - $isThere = in\_array($findMe, $arrayName);     - $isKey = array\_key\_exists($findKey, $arrayName);     - $key = array\_search($value, $arrayName);     - sort($arrayName); //For indexed arrays     - rsort($arrayName); //Descending     - asort($arrayName); //Sorts values, keeps keys connected     - arsort($arrayName); //Descending     - ksort($arrayName); //Sort keys, keep values attached     - krsort($arrayName); //Descending     - shuffle($arrayName); //Randomly arranges key/value pairs     - $key = array\_rand($arrayName); //Get random key     - $array = explode($separator, $string); //Make array from string       * Note there is also str\_split     - $string = implode($separator, $array); //Make string from array   See unit 1 notes for examples of these   * + - print\_r($arrayName); //Prints all elements of array     - extract($arrayName); //Converts key/values to variables | |
| * + Multi-Dimension Arrays     - Like in JavaScript, multi-dimension arrays are simply arrays of arrays | |
|  | Create function createWordList to build word list array (assoc)  Explode $sentence into an array ($words) return $words  Call in case controlResults  print\_r in results to test  remove commas and periods |
| 1. For Loops    * for ($index=0; $index<count($arrayName); $index++) {  block of statements }//end for    * foreach ($arrayName as $value) {   //returns each value from array in $value }//end for   * + foreach ($arrayName as $key => $value) {   //returns each key and value from array as $key and $value  }//end for   * + For loop with HTML embedded in it   <?php for ($i=1; $i<=10; $i++) : ?> Note the ending colon  <h1>Current index is <?php echo $i ?></h1>  <?php endfor; ?>  <?php endforeach; ?> //For foreach loops  //Option 2  <?php for ($i=1; $i<=10; $i++) {  echo '<h1>Current index is ' . $i . '</h1>'; Note HTML part of echo  ?> | |
| 1. While Loops    * while (condition) {  block of statements }//end while    * do {  block of statements } while(condition);    * <?php while(condition) : ?>  HTML HERE   <?php endwhile; ?> | Count occurrences of each word.  Use ForEach loop to add each word in $words to $wordlist, bumping up the counter.  Return wordlist  Note: if word isn’t on the list, adds it and bumps up counter (started at 1)  Use Word with 2 tables representing arrays and walk through code [go here](#example)  print\_r in body  Make process case insensitive (strtoupper)  ksort list  sort list (note key values gone)  asort list  arsort list |
|  | Add if statement to body. If wordlist contains Volker (use key\_exists) display a message otherwise display a different message  Add table to body to display each word and the number of times it appeared. |
| 1. Validating Data with PHP    * Most PHP validation can be avoided if the user has JavaScript turned and/or is using an HTML5 compliant browser.      + Required fields can be designated in JavaScript or HTML5      + Numeric data can be insured using input type number or range      + Min and Max numeric values can be designated in JavaScript or HTML5      + Unfortunately, only Chrome and Opera validate input type date properly. |  |
| 1. How to Transfer Business Requirements to GUI    * Define PHP constants    * Insert into GUI    * How insert into JavaScript? |  |
| 1. Form-Level Validation    * Some form-level validation must still be done by PHP      + Most common: duplicate record check    * If a form-level error occurs, the edit form must be redisplayed with all the original data and an error marker should appear in the appropriate location that includes text explaining the problem (*duplicate record*) |  |
| * + Before submitting the data, the controller calls the form-level validation function     - If there are no errors, the data is saved and the next view in the story is displayed     - If there **are** errors, the errors array is passed to the edit view.   + Each field in the edit view must check to see if an error exists for that field.     - If the error exists, display the error marker and set the title to the error message in the error array. | Using unit5b (end result of Unit 4)  Define the validateWeightData function.  Comment out link to weights.js in edit view  Update the controller  Update the form to display $details fields and to show error markers when appropriate |

|  |  |
| --- | --- |
| 1. Validation Tips    * If PHP discovers errors, transfer the $\_REQUEST array to an array called $details (or anything else)      + In Unit 6 we’ll be transferring database field values to the form, but they won’t be in a $\_REQUEST array. We’ll used $details as the common name.      + Extract the details array at the beginning of the edit view.      + Before defining each form field, check to see if the field value (e.g. $itemName) is set. If not, set it to the empty string (to handle null values later).   <?php $itemName = (isset($itemName))?$itemName:"" ?> | Update the form to display $details fields and to show error markers when appropriate  Test each field separately |
| 1. Using the Date Picker    * in form view      + require(‘calendar/classes/tc\_calendar.php’); |  |
| 1. <label>Date:</label> 2. <?php 3. //NOTE: DON'T USE $(frm).reset() IN WINDOW.ONLOAD if you're using this date picker 4. //NOTE: calendar folder must be in the same folder as index.php 5. $myCalendar = new tc\_calendar("dateCompleted", true); //Required //Must be first 6. $myCalendar->setPath("calendar/"); //Required 7. $myCalendar->setIcon("calendar/images/iconCalendar.gif"); //Required 9. //See calendar website for more options 10. if(isset($book['dateCompleted'])) 11. $myCalendar->setDateYMD($book['dateCompleted']); 12. else if ($action=="bookEdit" && $book['dateCompleted']=="") { 13. $today = date\_create(); 14. $myCalendar->setDateYMD($today->format('Y-m-d')); 15. }//end if 17. $myCalendar->writeScript(); //Required //Must be last 18. ?> | |
|  |  |