Agenda

- This week’s expected outcomes
- This week’s topics
- This week’s homework
- Upcoming deadlines
- Questions and answers
Week 4 Outcomes

• Create HTML forms
• Access form data from a script
• Construct algorithms using selection and repetition structures.
Web Forms

• Basic HTML-based web-form elements
  – Fieldsets
  – Labels
  – Text boxes
  – Text areas
  – Buttons
  – Radio buttons
  – Check boxes
  – Combo boxes
Web Forms

- Basic HTML-based web form elements
  - Fieldsets
  - Labels
  - Text boxes
  - Text areas
  - Buttons
  - Radio buttons
  - Check boxes
  - Combo boxes
Web Forms

<form action='formelements.php' method='post'>
  <fieldset>
    <legend>Personal information</legend>
    <label for='firstName'>First Name</label>
    <input type='text' id='firstName' name='firstName' />
    <label for='lastName'>Last Name</label>
    <input type='text' id='lastName' name='lastName' />
    <label for='comments'>Comments</label>
    <textarea id='comments' name='comments' rows='5' cols='40'></textarea>
  </fieldset>
</form>
Web Forms

```html
<form action='formelements.php' method='post'>
    <fieldset>
        <legend>Personal information</legend>
        <label for='firstName'>First Name</label>
        <input type='text' id='firstName' name='firstName' />
        <label for='lastName'>Last Name</label>
        <input type='text' id='lastName' name='lastName' />
        <label for='comments'>Comments</label>
        <textarea id='comments' name='comments' rows='5' cols='40'></textarea>
    </fieldset>
</form>
```

Any input, textarea, select, etc., within the form will be submitted

- **action**: script to receive data
- **method**: get or post
Web Forms

```html
<form action='formelements.php' method='post'>
  <fieldset>
    <legend>Personal information</legend>
    <label for='firstName'>First Name</label>
    <input type='text' id='firstName' name='firstName' />
    <label for='lastName'>Last Name</label>
    <input type='text' id='lastName' name='lastName' />
    <label for='comments'>Comments</label>
    <textarea id='comments' name='comments' rows='5' cols='40'></textarea>
  </fieldset>
</form>
```

fieldset groups related fields graphically (no effect on what the server receives). legend describes the fieldset.
Web Forms

```
<form action='formelements.php' method='post'>
    <fieldset>
        <legend>Personal information</legend>
        <label for='firstName'>First Name</label>
        <input type='text' id='firstName' name='firstName' />
        <label for='lastName'>Last Name</label>
        <input type='text' id='lastName' name='lastName' />
        <label for='comments'>Comments</label>
        <textarea id='comments' name='comments' rows='5' cols='40'></textarea>
    </fieldset>
</form>
```

Labels are click targets. Clicking a label activates the input control (associated by for='someid' in the label and id='someid' in the input control).
Web Forms

```html
<form action='formelements.php' method='post'>
  <fieldset>
    <legend>Personal information</legend>
    <label for='firstName'>First Name</label>
    <input type='text' id='firstName' name='firstName' />
    <label for='lastName'>Last Name</label>
    <input type='text' id='lastName' name='lastName' />
    <label for='comments'>Comments</label>
    <textarea id='comments' name='comments' rows='5' cols='40'></textarea>
  </fieldset>
</form>

PHP receives these data into the $_GET, $_POST, or $_REQUEST superglobals according to the name attribute.
Web Forms

```
<form action='formelements.php' method='post'>
  <fieldset>
    <legend>Personal information</legend>
    <label for='firstName'>First Name</label>
    <input type='text' id='firstName' name='firstName' />
    <label for='lastName'>Last Name</label>
    <input type='text' id='lastName' name='lastName' />
    <label for='comments'>Comments</label>
    <textarea id='comments' name='comments' rows='5' cols='40'></textarea>
  </fieldset>
</form>
```

type='text' produces a single-line text box for input.
Web Forms

For large, multi-line input, use a textarea. Any text between the begin/end tags is put inside the text area.

```html
<form action='formelements.php' method='post'>
  <fieldset>
    <legend>Personal information</legend>
    <label for='firstName'>First Name</label>
    <input type='text' id='firstName' name='firstName' />
    <label for='lastName'>Last Name</label>
    <input type='text' id='lastName' name='lastName' />
    <label for='comments'>Comments</label>
    <textarea id='comments' name='comments' rows='5' cols='40'></textarea>
  </fieldset>
</form>
```
<label for='inquiryType'>Inquiry Type</label>

<select id='inquiryType' name='inquiryType'>
  <option value='error'>Choose an option</option>
  <option value='prodInfo'>Product information</option>
  <option value='custSvc'>Customer service</option>
  <option value='returns'>Returns</option>
  <option value='other'>Other</option>
</select>

Each option appears on its own line, value is what is submitted to the server for the name specified in the select.
Web Forms

```xml
<fieldset>
  <legend>Demographic information</legend>
  <fieldset>
    <legend>Gender</legend>
    <input type='radio' name='gender' id='male' value='M' />
    <label for='male'>Male</label>
    <input type='radio' name='gender' id='female' value='F' />
    <label for='female'>Female</label>
  </fieldset>
</fieldset>
```

For type=radio, you use different ids, but the same name. Only the selected one within the name group will be submitted.
Web Forms

<fieldset>
  <legend>Interests</legend>
  <input type='checkbox' name='interests[]' id='games' value='games' />
  <label for='games'>Games</label>
  <input type='checkbox' name='interests[]' id='reading' value='reading' />
  <label for='reading'>Reading</label>
  <input type='checkbox' name='interests[]' id='gardening' value='gardening' />
  <label for='gardening'>Gardening</label>
  <input type='checkbox' name='interests[]' id='sports' value='sports' />
</fieldset>

<input type='submit' value='Submit' />
<input type='reset' value='Clear' />
</form>
For type=checkbox, you use different ids, but the same name. The name appears as an array (using []). This creates an array on the server.
Web Forms

```html
<fieldset>
  <legend>Interests</legend>
  <input type='checkbox' name='interests[]' id='games' value='games' />
  <label for='games'>Games</label>
  <input type='checkbox' name='interests[]' id='reading' value='reading' />
  <label for='reading'>Reading</label>
  <input type='checkbox' name='interests[]' id='gardening' value='gardening' />
  <label for='gardening'>Gardening</label>
  <input type='checkbox' name='interests[]' id='sports' value='sports' />
  <label for='gardening'>Sports</label>
</fieldset>

<form>
  <input type='submit' value='Submit' />
  <input type='reset' value='Clear' />
</form>
```

An input with type=submit creates a button, as does reset.
PHP Scripts for Form Data

- Can use $_GET, $_POST, or $_REQUEST arrays to access form data.
  - All are arrays with the key equal to the name given to the form element.
  - The values are strings from the data in the form.
  - Multiple-select form elements have array values
    - Multiple select combo-boxes
    - Checkboxes
<?php
echo dumpArray($_REQUEST);
?>
<?php

function dumpArray($elements) {
    $result = "<ol>
    
    endforeach ($elements as $key => $value) {
        if (is_array($value)) {
            $result .= "<li>Key <b>$key</b> is an array containing:
            " . dumpArray($value) . "</li>
        } else {
            $value = nl2br(htmlspecialchars($value));
            $result .= "<li>Key <b>$key</b> has value <b>$value</b></li>
        }
    }
    
    return $result . "</ol>
    
?>
<?php
function dumpArray($elements) {
    $result = "<ol><li>
    foreach ($elements as $key => $value) {
        if (is_array($value)) {
            $result .= "<li>Key <b>$key</b> is an array containing:" . dumpArray($value) . "</li><li>
        } else {
            $value = nl2br(htmlspecialchars($value));
            $result .= "<li>Key <b>$key</b> has value <b>$value</b></li><li>
        }
    }
    return $result . "</ol><li>";
}
?>

Recursion: A function calling itself. Similar to a loop, but has an implicit stopping case.
<?php
function dumpArray($elements) {
    $result = "<ol>
"
    foreach ($elements as $key => $value) {
        if (is_array($value)) {
            $result .= "<li>Key <b>$key</b> is an array containing:" . dumpArray($value) . "</li>
"
        } else {
            $value = nl2br(htmlspecialchars($value));
            $result .= "<li>Key <b>$key</b> has value <b>$value</b></li>
"
        }
    }
    return $result . "</ol>"
}
?>

htmlspecialchars converts < into &lt;, > into &gt;, & into &amp;, etc.
<?php

function dumpArray($elements) {
    $result = "<ol><br/>
    foreach ($elements as $key => $value) {
        if (is_array($value)) {
            $result .= "<li>Key <b>$key</b> is an array containing:" . dumpArray($value) . "</li><br/>
        } else {
            $value = nl2br(htmlspecialchars($value));
            $result .= "<li>Key <b>$key</b> has value <b>$value</b></li><br/>
        }
    }
    return $result . "</ol><br/>
}
?>

nl2br converts newlines into <br/> so that text areas maintain their line breaks.
<?php
function dumpArray($elements) {
    $result = "<ol><li>"
    foreach ($elements as $key => $value) {
        if (is_array($value)) {
            $result .= "<li>Key <b>$key</b> is an array containing:" . dumpArray($value) . "</li>"
        } else {
            $value = nl2br(htmlspecialchars($value));
            $result .= "<li>Key <b>$key</b> has value<b>$value</b></li>";
        }
    }
    return $result . "</ol>";
}
?>
PHP Scripts for Form Data

• Caveats
  – The previous code dumped out the entire request, which is rarely what you want to do.
  – If you are looking for a specific value, you should use isset() to see if it was submitted.
function safeGet($array, $key, $default=false) {
    if (isset($array[$key])) {
        $value = $array[$key];
        if (!is_array($value)) {
            $value = htmlspecialchars(trim($array[$key]));
        }
        if ($value) {
            return $value;
        }
    }
    return $default;
}

$interests = safeGet($_REQUEST, 'interests');
$firstName = safeGet($_REQUEST, 'firstName');
Control Statements

- Control statements alter the flow of a program from straight line to conditional or repeated
  - If/else statements
  - Switch statements
  - For loops
  - Foreach loops
  - While loops
  - Do/while loops
if/else Statements

• Keywords if and else implement conditional execution

```java
if (<condition>) {
    <statements>
}
```
if/else Statements

• Keywords if and else implement conditional execution

```java
if (<condition>) {
    <statements>
}
```

condition should be a boolean expression (after type juggling).
if/else Statements

• Keywords if and else implement conditional execution

```
if (<condition>) {
    <statements>
} else {
    <statements>
}
```
switch Statements

• Switch statements
  – A shortcut to compare many values and conditionally execute code based strictly on equality.
  • **Good** for a limited number of enumerable options.
  • **Bad** for testing ranges of values, deciding between two mutually exclusive options.
switch Statements

• Switch statements – suitability

<table>
<thead>
<tr>
<th>Example</th>
<th>If/else or switch?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determining point values of letter grades.</td>
<td></td>
</tr>
<tr>
<td>Determining letter grades from a percentage.</td>
<td></td>
</tr>
<tr>
<td>Determining insurance rates based on age.</td>
<td></td>
</tr>
<tr>
<td>Determine the name of a month based on a month number.</td>
<td></td>
</tr>
<tr>
<td>Determine form letter salutation based on gender.</td>
<td></td>
</tr>
</tbody>
</table>
switch Statements

• Switch statements

```java
switch (<expression>) {
    case <value1>:
        <statements1>;
        break;
    case <value2>:
        <statements2>;
        break;
    //...
    case <valueN>:
        <statementsN>;
        break;
    default:
        <statements>;
        break;
}
```

What if “break” is missing?
switch Statements

• Switch statements

```
switch (<expression>) {
    case <value1>:
        <statements1>;
        break;
    //...
    case <valueN>:
        <statementsN>;
        break;
    default:
        <statements>;
        break;
}
```

Execution “falls through” to the next case!
Repetition

• Four parts to every loop
  – Initialization
  – Continuation condition
  – Body
  – Update
Repetition

• Pre-test loops
  – Condition is evaluated before the body of the loop is executed.
  – Key idea: body may not ever execute.
Repetition

• Post-test loops
  – Condition is evaluated after the body of the loop is executed.
  – Key idea: body always executes at least once
while loops

• While loops:
  – Pre-test loop syntax

```java
while (condition) {
    body_statements;
}
```

All that is really required. But, which of the four parts are missing?
while loops

• While loops:
  – Pre-test loop syntax

```plaintext
initialization;
while (condition) {
  body_statements;
  update_statement;
}
```
for loops

- For loops:
  
  Pre-test loop syntax

  ```
  for (initialization; condition; update) {
    body_statements;
  }
  ```
for loops

• For loops:
  – Pre-test loop syntax

```plaintext
for (initialization; condition; update) {
    body_statements;
}
```

Equivalent to:

```plaintext
initialization;
while (condition) {
    body_statements;
    update;
}
```
for vs. while loops

• When to use for vs. while
  – Equivalent at runtime
  – while loops are a little more flexible (i.e. the update step can be conditional or in the middle of the body)
  – for loops are generally used for counting (i.e. the bounds are known)
do...while loops

- Post-test loop syntax

```java
do {
    body_statements;
} while (condition);
```

```java
(initialization;
    do {
        body_statements;
        update;
    } while (condition);
```
do...while loops

- Post-test loop syntax

```
do {
    body_statements;
} while (condition);
```

**Required elements.**

```
initialization;
do {
    body_statements;
    update;
} while (condition);
```

**All 4 elements.**
do...while loops

- Post-test loops
  - Body always guaranteed to execute at least once.
  - But, we could still copy-and-paste the body above a pre-test loop and achieve the same results.
**foreach/as loops**

- Specifically designed to solve the problem of iterating through *every* element of an array.

```php
foreach ($array_var as $value) {
    do_something;
}
```

```php
foreach ($array_var as $key => $value) {
    do_something;
}
```
**foreach/as loops**

- Specifically designed to solve the problem of iterating through *every* element of an array.

```php
foreach ($array_var as $value) {
    do_something;
}
```

Assigns values from the array into $value each trip through the loop.

```php
foreach ($array_var as $key => $value) {
    do_something;
}
```
foreach/as loops

- Specifically designed to solve the problem of iterating through every element of an array.

Assigns the index into $key and the value into $value each trip through the loop.

```php
foreach ($array_var as $key => $value) {
    do_something;
}
```
foreach/as loops

Example:

```php
$alphabet = array(
    'a' => 'apple',
    'b' => 'banana',
    'c' => 'carrot'
);

foreach ($alphabet as $value) {
    print "$value<br />
}

foreach ($alphabet as $key => $value) {
    print "$key is for $value<br />
}
```

Prints:
apple
banana
carrot
foreach/as loops

• Example:

```php
$alphabet = array(
    'a' => 'apple',
    'b' => 'banana',
    'c' => 'carrot'
);

foreach ($alphabet as $value) {
    print "$value<br />
};

foreach ($alphabet as $key => $value) {
    print "$key is for $value<br />
};
```

Prints:
- a is for apple
- b is for banana
- c is for carrot
foreach/as loops

• Example:

```php
$alphabet = array(
    'a' => 'apple',
    'b' => 'banana',
    'c' => 'carrot'
);

foreach ($alphabet as $value) {
    print "$value<br />
}

foreach ($alphabet as $key => $value) {
    print "$key is for $value<br />
}
```

If $alphabet is not an associative array, the keys will be numeric, numbered [0, n-1], where n is the length of the array as determined by sizeof() or count().
foreach/as loops

• Ex: roll 2 dice many times. Find probabilities.

```php
function probability($count=10000) {
    $rolls = array();
    for ($i = 2; $i <= 12; ++$i) {
        $rolls[$i] = 0;
    }
    for ($i = 0; $i < $count; ++$i) {
        $die1 = mt_rand(1, 6);
        $die2 = mt_rand(1, 6);
        $rolls[$die1 + $die2]++;
    }
    foreach ($rolls as $die => $times) {
        print "$die was rolled $times times, for a probability of ". $times/$count * 100 . "%."
    }
}
```
foreach/as loops

- Ex: roll 2 dice many times. Find probabilities.

```php
function probability($count) {
    $rolls = array();
    for ($i = 2; $i <= 12; ++$i) {
        $rolls[$i] = 0;
    }
    for ($i = 0; $i < $count; ++$i) {
        $die1 = mt_rand(1, 6);
        $die2 = mt_rand(1, 6);
        $rolls[$die1 + $die2]++;}
    foreach ($rolls as $die => $times) {
        print "$die was rolled $times times, for a probability of ".
        print $times/$count * 100 . "%."
    }
}
```

Real probability of rolling a 7 is 6 in 36, or 16.67%. 
Upcoming Deadlines

• Readings for next week
  – Chapters 9 and 10 in *PHP and MySQL*

• Assignments
  – Homework 3 due end of week 4
  – Lab 2 due end of week 7

• Next week:
  – Strings, numbers, and dates
General Q & A

• Questions?
• Comments?
• Concerns?