CSY2028
Web Programming
Topic 6

Tom Butler
thomas.butler@northampton.ac.uk

Topic 6

- User input using HTML forms

User Input

- Last week we covered user input with $_GET
- http://v.je/file.php?num=123
- Changing the URL allows you to feed values to the script

```php
<?php

var_dump($_GET);

?>
```

```
array (size=1)
  'num' => string '123' (length=3)
```

User input

- You can link to different pages using this method:

```php
<?php
echo '<a href="page.php?choice=1">One</a>';
echo '<a href="page.php?choice=2">Two</a>';
echo '<a href="page.php?choice=2">Three</a>';

if (isset($_GET['choice'])) {
        echo 'You chose ' . $_GET['choice'];
}
else {
        echo 'You did not make a choice';
}
```

User input

- This is useful for links where the user has to make a choice between one of several predetermined outcomes
- It's not useful when the user needs to type something in:
  - A name
  - Login details
  - Etc
- Getting the user to amend the address bar is confusing!

HTML Forms

- HTML provides a method of allowing user input:
  - Forms
- HTML forms have the following interactive elements:
  - Text (single line)
  - Text area (multi line)
  - Checkbox
  - Drop down (select)
  - Radio button
  - File upload
  - Buttons

HTML Forms

- All form elements must be placed inside a <form> tag
- The form tag has two required attributes:
  - *Method* - This is either GET or POST
  - *Action* – URL to which the form is *submitted,* this can be the same script you are using to display the form

HTML Forms

- Each element in a form has a *name* attribute
- This name can be referenced in PHP after the form has been *submitted*
- Each element should have a unique name

Basic HTML form

- A form needs at least one *input element* (e.g. a text box)
- Each form needs a *submit button*

```
<form action="file.php" method="GET">

        <input type="text" name="myinput" />

        <input type="submit" value="Submit" name="submit" />

</form>
```

Basic HTML form

Action attribute

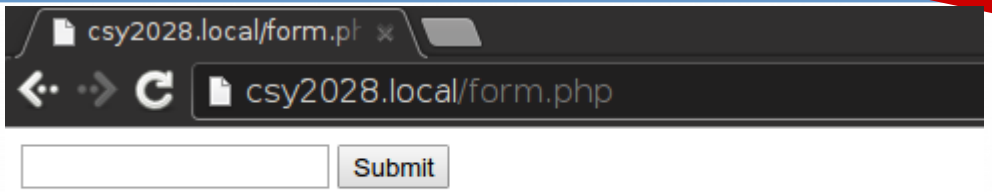
Text box named “myinput”

Form element

Method attribute

```
<form action="form.php" method="GET">

        <input type="text" name="myinput" />

        <input type="submit" value="Submit" name="submit" />

</form>
```

The value attribute is used as the button text

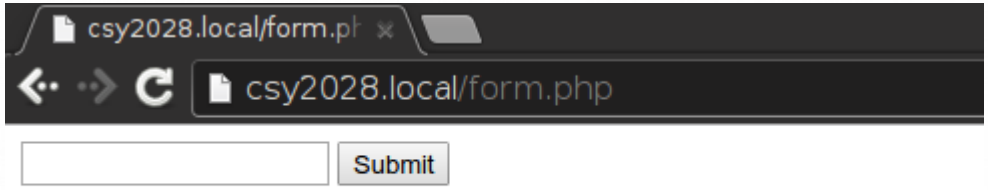


Submit button element

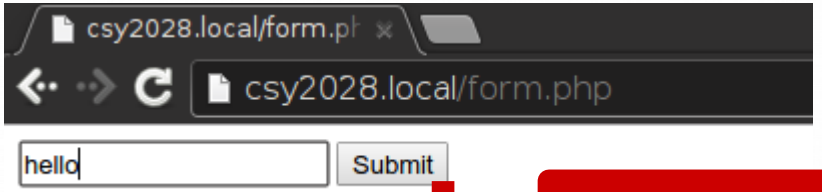
method="GET"

- When you set the form method to GET (it is case sensitive!) and the form is submitted, the url *query string* is automatically
- Last week we set it manually e.g. typing

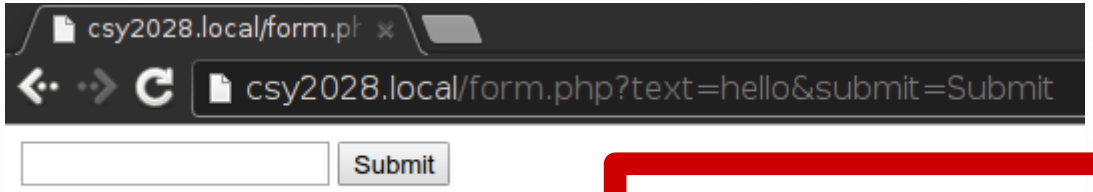
  http://v.je/file.php?key=value
- When a GET form is submitted, it *serializes* all the form elements and automatically generates the URL parameters and navigates to the page:



Enter a value into the box

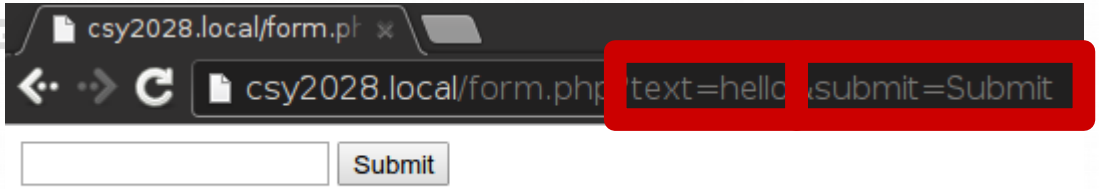


Press the submit button



The contents of the form have been cleared and the URL has been updated

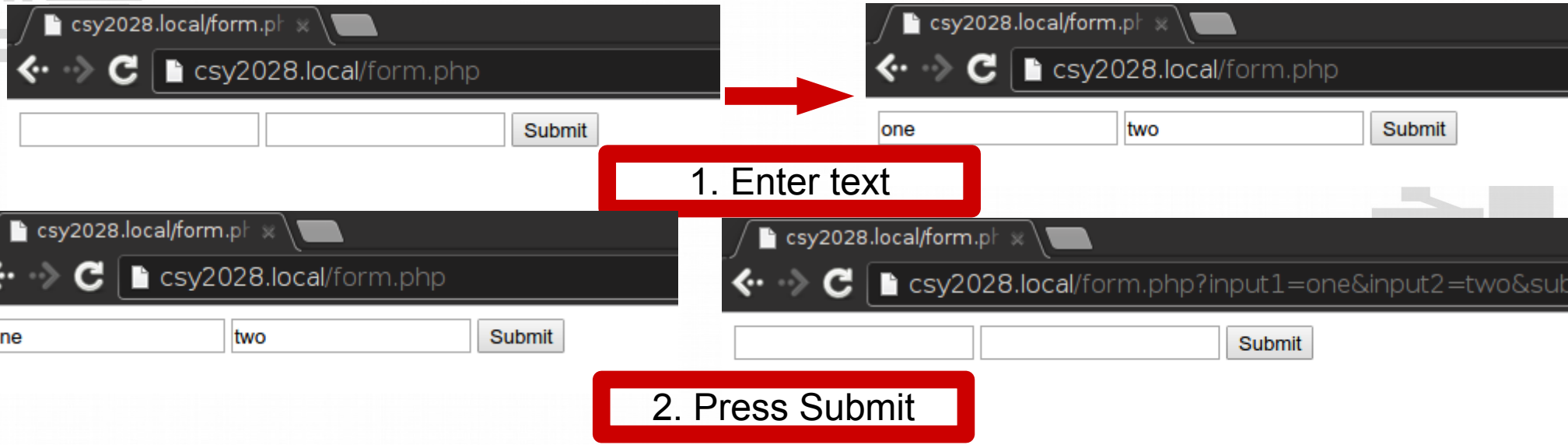
```
<form action="form.php" method="GET">

        <input type="text" name="text" />

        <input type="submit" value="Submit" name="submit" />

</form>
```



Form elements

- Forms can have as many elements as you like:

```
<form action="form.php" method="GET">

        <input type="text" name="input1" />
        <input type="text" name="input2" />

        <input type="submit" value="Submit" name="submit" />

</form>
```

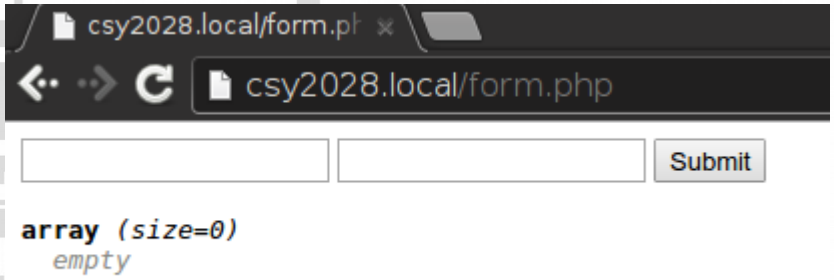


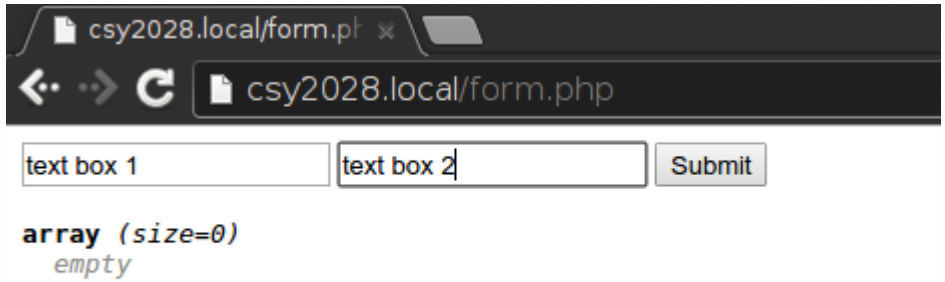
Using form data

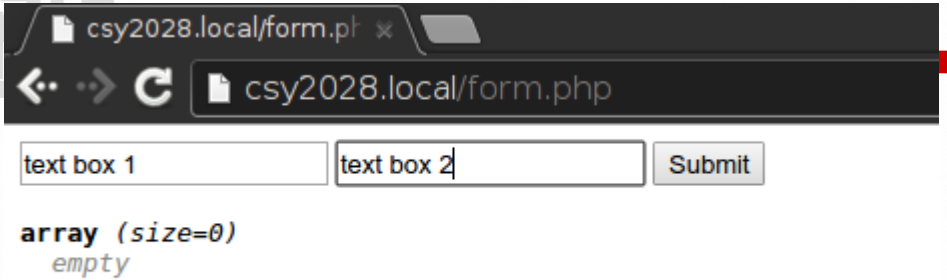
- Once you've submitted the form, you can use its contents via $_GET (see last week's notes)

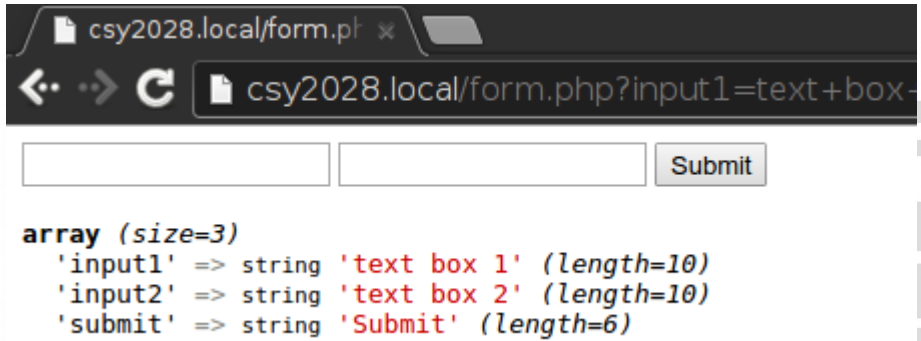
```
<form action="form.php" method="GET">

        <input type="text" name="input1" />
        <input type="text" name="input2" />

        <input type="submit" value="Submit" name="submit" />

</form>

<?php
var_dump($_GET);
?>
```

Using form data







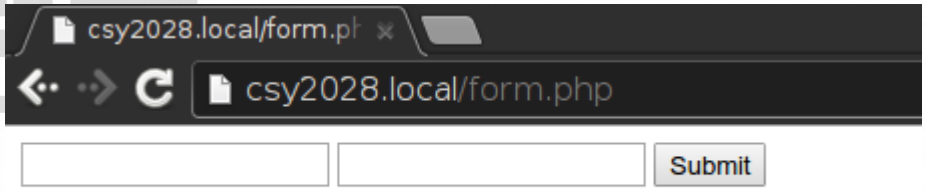


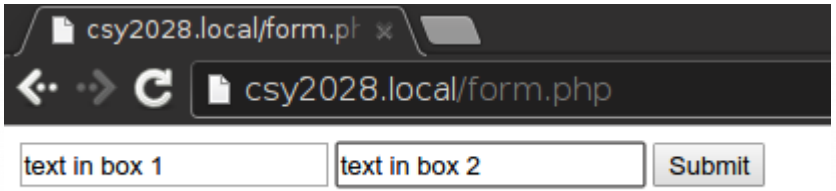
Using form data

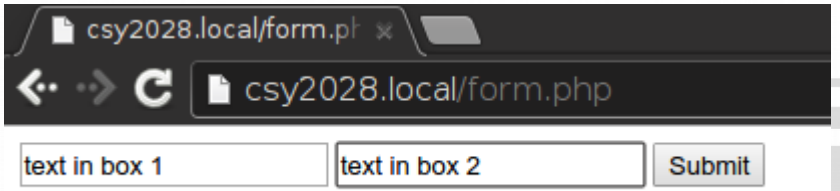
- You can then use the $_GET array in the same way as last week:

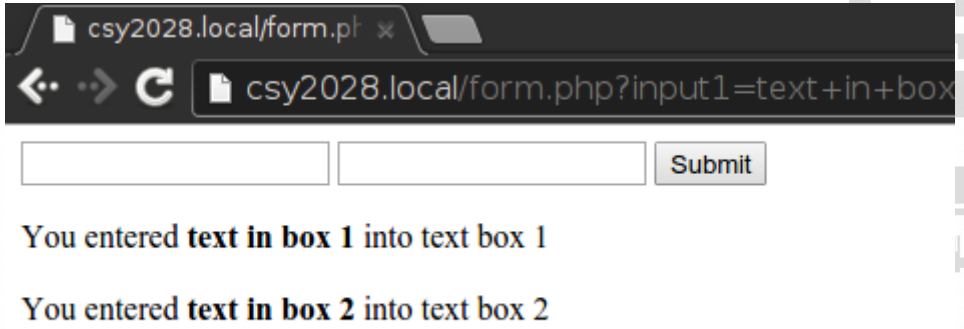
```
<form action="form.php" method="GET">

        <input type="text" name="input1" />
        <input type="text" name="input2" />

        <input type="submit" value="Submit" name="submit" />

</form>

<?php

if (isset($_GET['input1'])) {
        echo '<p>You entered <strong>' . $_GET['input1'] . '</strong> into text box 1</p>';
}

if (isset($_GET['input2'])) {
        echo '<p>You entered <strong>' . $_GET['input2'] . '</strong> into text box 2</p>';
}

?>
```

```php
<form action="form.php" method="GET">
..
</form>

<?php

if (isset($_GET['input1'])) {
        echo '<p>You entered <strong>' . $_GET['input1'] . '</strong> into text box 1</p>';
}

if (isset($_GET['input2'])) {
        echo '<p>You entered <strong>' . $_GET['input2'] . '</strong> into text box 2</p>';
}
```









Exercise 1

- 1) Create a HTML form that asks the user for their name. When the form is submitted it should say "Hello [name]" where name is the name they entered in the text box
- 2) Create a HTML form with two boxes for entering numbers. When the form is submitted show the result of multiplying the numbers together along with the calculation itself e.g. if the user entered 5 and 6 show "5 x 6 = 30"
- 3) **Optional** Amend (2) to include a third box for "Operation". Let the user type in "+", "-", "/" or "*" to select the mathematical operation to perform.

```php
<?php

if (isset($_GET['name'])) {
        echo 'Hello ' . $_GET['name'];
}
else {
?>
<form action="name.php" method="GET">
        What is your name?
        <input type="text" name="name" />

        <input type="submit" value="Submit" name="submit" />

</form>

<?php
}
?>
```

```php
<form action="calculator.php" method="GET">
        <input type="text" name="num1" />
        <input type="text" name="operation" />
        <input type="text" name="num2" />

        <input type="submit" value="submit" type="submit" />
</form>

<?php
if (isset($_GET['submit'])) {
        echo $_GET['num1'] . ' ' . $_GET['operation'] . ' ' . $_GET['num2'] . ' = ';

        if ($_GET['operation'] == '*') {
                echo $_GET['num1'] * $_GET['num2'];
        }
        else if ($_GET['operation'] == '/') {
                echo $_GET['num1'] / $_GET['num2'];
        }
        else if ($_GET['operation'] == '+') {
                echo $_GET['num1'] + $_GET['num2'];
        }
        else if ($_GET['operation'] == '-') {
                echo $_GET['num1'] - $_GET['num2'];
        }
}


?>
```

POST forms

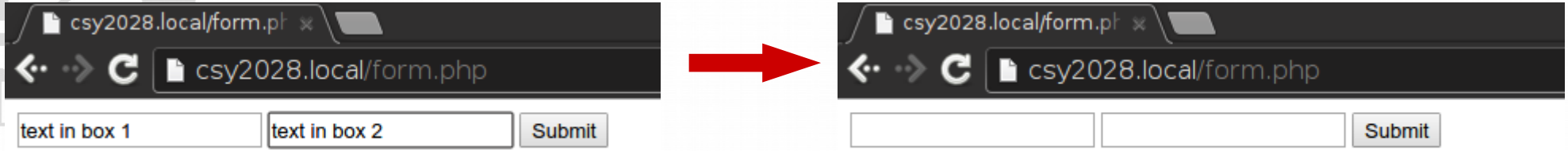
- Post forms work in an almost identical way to GET forms
- You can change the *method* attribute to POST (uppercase!) to make the form a *POST* form:

```
<form action="form.php" method="POST">

        <input type="text" name="input1" />
        <input type="text" name="input2" />

        <input type="submit" value="Submit" name="submit" />

</form>
```

```
<form action="form.php" method="POST">

        <input type="text" name="input1" />
        <input type="text" name="input2" />

        <input type="submit" value="Submit" name="submit" />

</form>
```

POST forms

- When a form uses POST the URL is not changed:

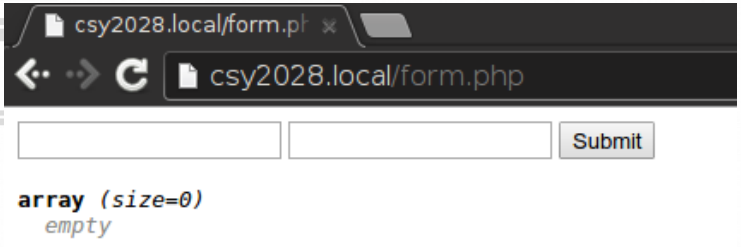


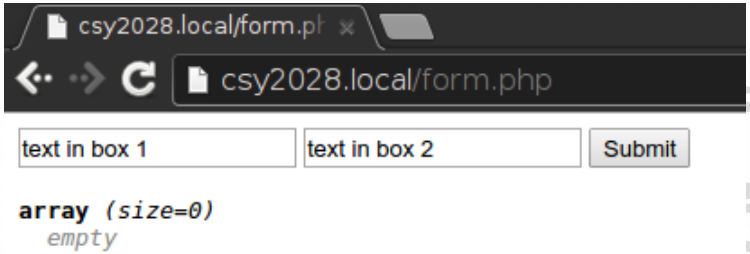
- You can also not use the $_GET variable to access submitted values

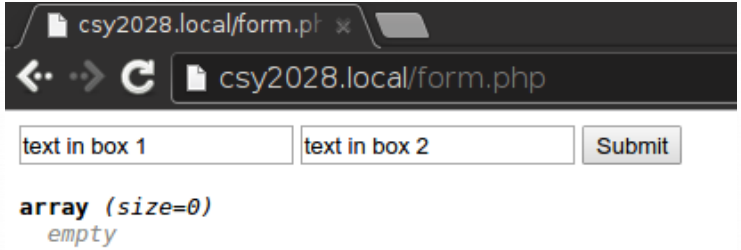
POST forms

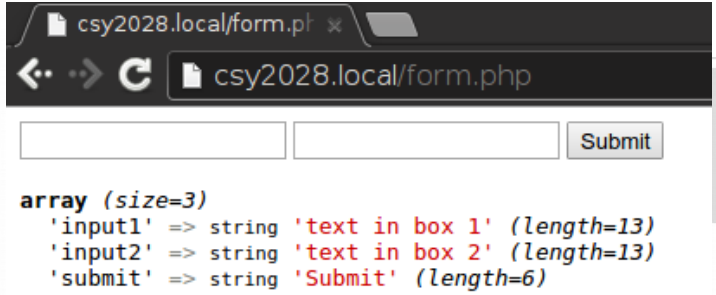
- Instead, you must use the $_POST variable:

```
<form action="form.php" method="POST">
        <input type="text" name="input1" />
        <input type="text" name="input2" />

        <input type="submit" value="Submit" name="submit" />
</form>
<?php
var_dump($_POST);
?>
```









POST forms

- There a several practical differences between GET and POST forms, although mostly they can be used for any purpose:
  - POST forms do not amend the URL
  - You cannot send values to POST forms via the URL
  - With GET forms, you can submit the form and then hyperlink to the results by copying the URL
  - With POST you must fill the form manually each time

GET vs POST

- Generally POST forms should be used:
  - When something on the server needs to be *changed* (e.g. inserting/updating information, logging in)
- Generally GET forms should be used:
  - When a set of *results* should be generated (e.g. search results, switching page)
- If unsure, use POST.

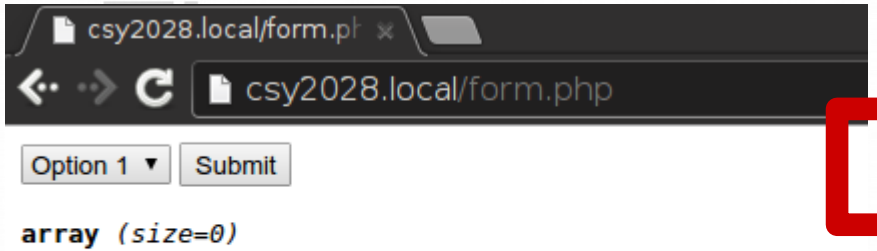
GET vs POST

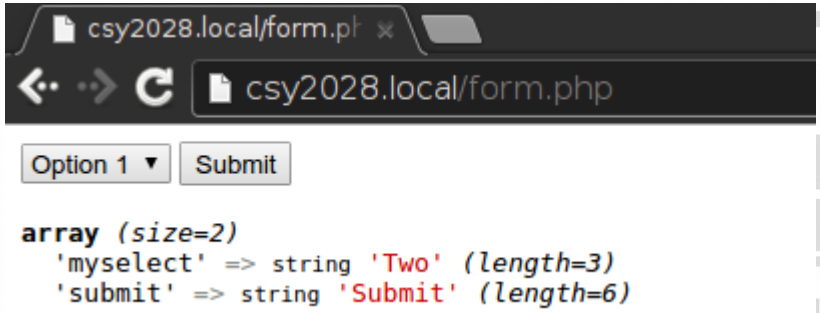
- With GET forms, going to the URL will trigger the form submission
- This means that when a user clicks “back” in their browser it will be treated as a new submission
- If you're *inserting data* or doing something that will update information on the server and the user *reloads* the page or goes *back* to it, the submission will be duplicated!
- With POST, the browser issues a *warning* before resubmission.

HTML Forms – Select Boxes

- The HTML <select> tag enforces selection of a specific value
- The user cannot enter *free text*. They are forced to choose between one of the supplied *options*
- You must specify the available options inside the HTML
- Each option has a *value* which is sent back to the server and a *label* which is displayed to the user
  - The user never sees the *value*
  - The *label* is never sent to the server

```php
<form action="form.php" method="POST">

        <select name="myselect">
                <option value="One">Option 1</option>
                <option value="Two">Option 2</option>
                <option value="Three">Option 3</option>
        </select>
        <input type="submit" value="Submit" name="submit" />

</form>

<?php
var_dump($_POST);
?>
```





Checkboxes

- A checkbox is an input element that allows the user to *tick* a box. This represents a *boolean* on a HTML form, the user can choose to tick the box or not
- When the form is submitted, if the checkbox is *ticked* the $_POST or $_GET array will be filled with the *value attribute* from the form
- If the checkbox is not ticked, the $_GET or $_POST array will not contain the supplied value!
- You must use *isset* to check whether the checkbox was ticked!

```php
<form action="form.php" method="POST">

        <input type="checkbox" name="mycheckbox" value="ticked" />
        <input type="submit" value="Submit" name="submit" />

</form>

<?php
var_dump($_POST);
?>
```

Checkboxes

- You must use isset to determine whether a checkbox has been ticked or not:

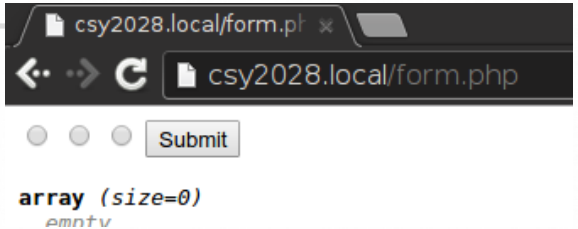
```
<form action="form.php" method="POST">

        <input type="checkbox" name="mycheckbox" value="ticked" />
        <input type="submit" value="Submit" name="submit" />

</form>

<?php
if (isset($_POST['mycheckbox'])) {
        echo 'The checkbox was ticked';
}
else {
        echo 'The checkbox was not ticked';
}
?>
```

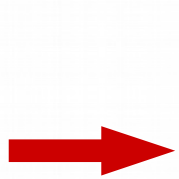
Radio Buttons

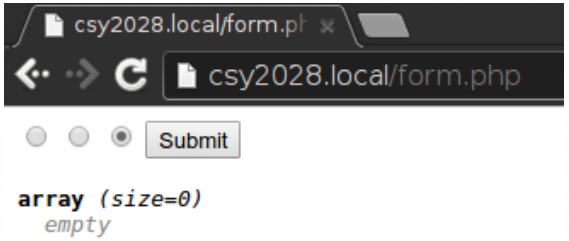
- Radio buttons are similar to checkboxes
- You can *tick* and *untick* them
- However, only one radio button can be ticked at the same time
- You must create multiple radio buttons with the same *name.*
- Only one radio button with that *name* can be ticked at any time

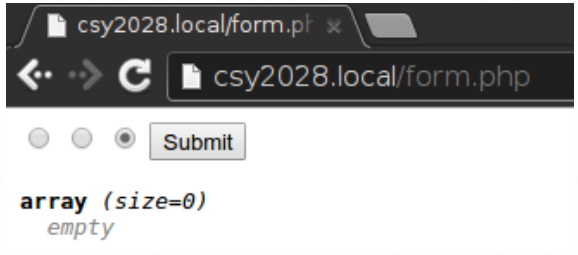
Radio Buttons

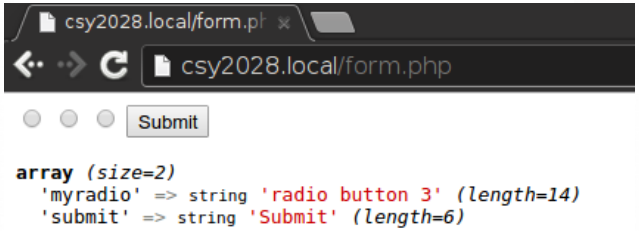
```
<form action="form.php" method="POST">

        <input type="radio" name="myradio" value="radio button 1" />
        <input type="radio" name="myradio" value="radio button 2" />
        <input type="radio" name="myradio" value="radio button 3" />

        <input type="submit" value="Submit" name="submit" />

</form>

<?php
var_dump($_POST);
?>
```











Password input

- Password boxes allow entering of *free text*
- However, the value is never displayed on screen, it's replaced with asterisks:

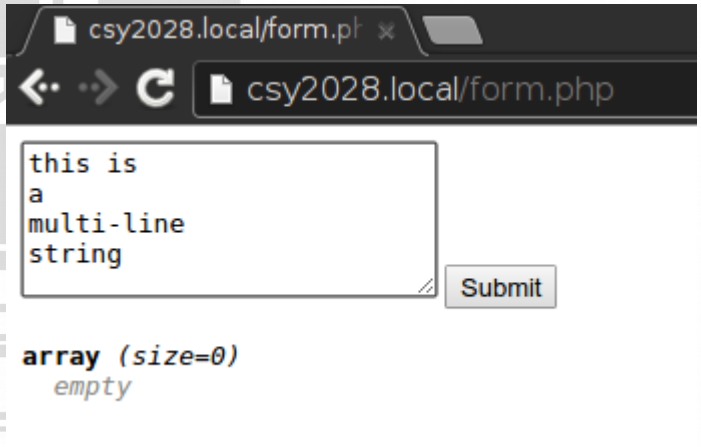
```
<form action="form.php" method="POST">
        <input type="password" name="mypassword" />
        <input type="submit" value="Submit" name="submit" />

</form>

<?php
var_dump($_POST);
?>
```

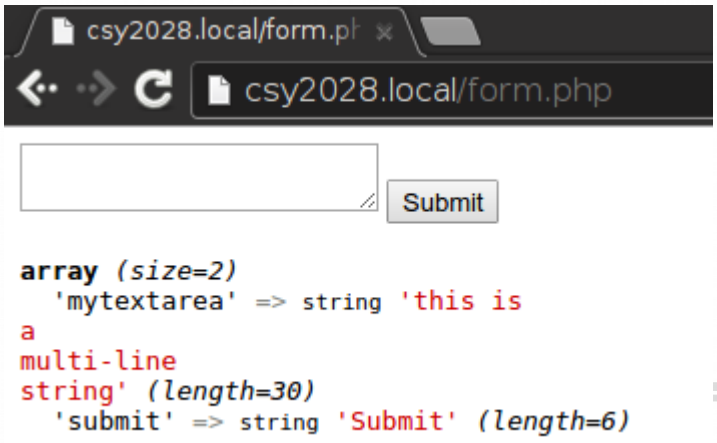
Textarea

- Textarea inputs allow *multi-line* user input
- Textarea, like <select> has an opening and closing tag (input doesn't!)

```
<form action="form.php" method="POST">
        <textarea name="mytextarea" />
            </textarea>
        <input type="submit" value="Submit" name="submit" />

</form>

<?php
var_dump($_POST);
?>
```

Textarea





- <textarea> can contain *line breaks* wnereas <ınput type="text" /> cannot

GET and POST

- Any input type (checkboxes, textboxes, textareas, etc) can be used with either GET or POST
- GET has a character limit of 4096 characters in some browsers so long forms should always be handled with POST

Submitting Forms

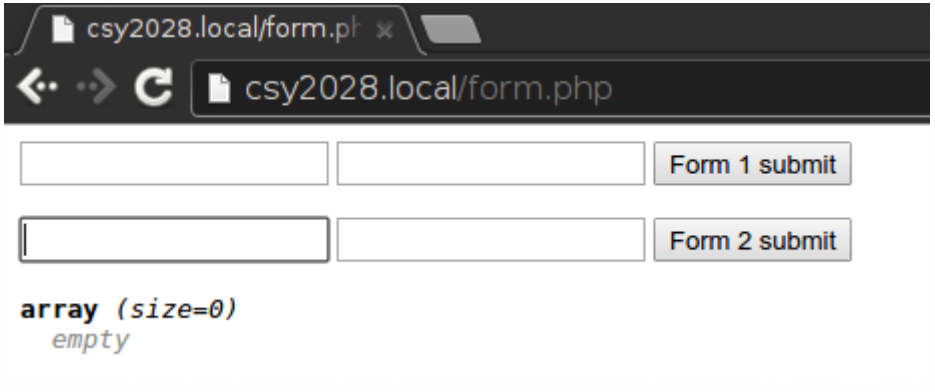
- Forms must be submitted by a user action
- That user action can be:
  - Clicking the *Submit* button
  - Pressing *Enter* while a field has *focus* (is currently selected)

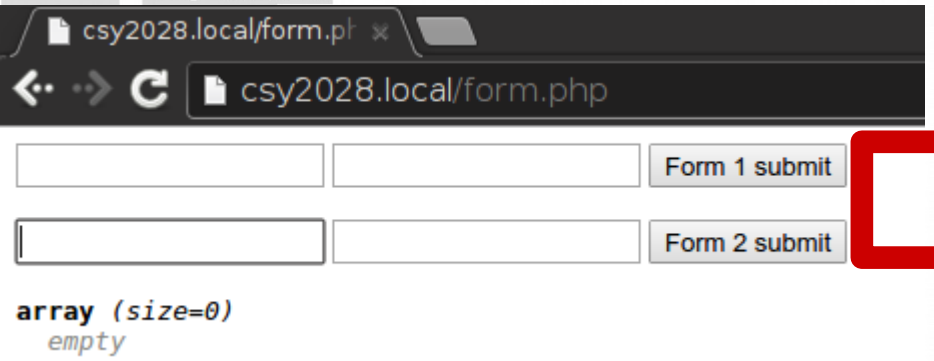
Forms

- A HTML form element may have as many elements as you like
- There can be more than one form on a page
- However, only one can be submitted

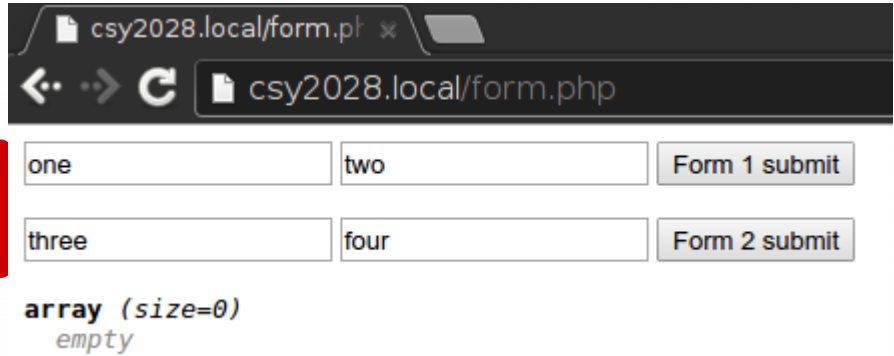
Multiple forms on one page

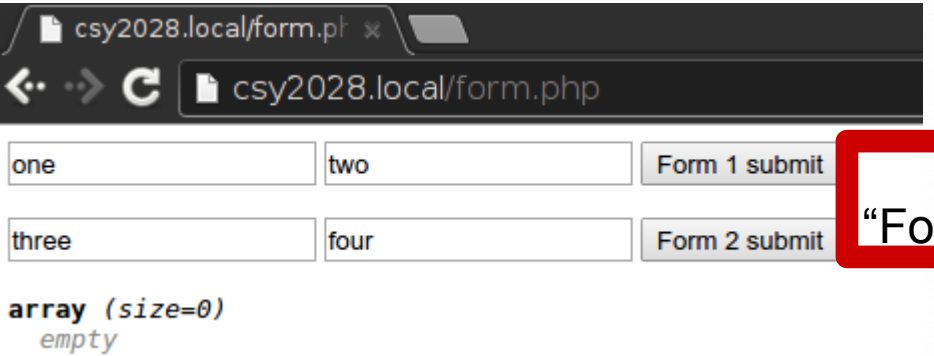
```
<form action="form.php" method="POST">
        <input type="text" name="form1text1" />
        <input type="text" name="form1text2" />
        <input type="submit" name="submit" value="Form 1 submit" />

</form>

<form action="form.php" method="POST">
        <input type="text" name="form2text1" />
        <input type="text" name="form2text2" />
        <input type="submit" name="submit" value="Form 2 submit" />

</form>

<?php
var_dump($_POST);
?>
```



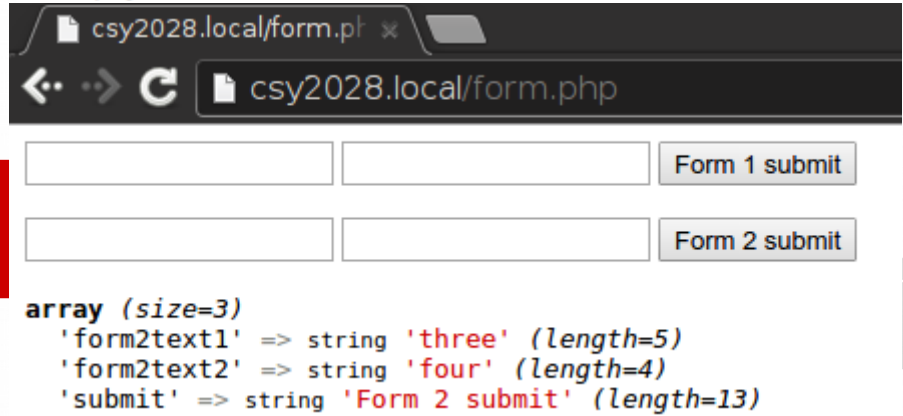


Enter text
In boxes





Press
“Form 2 Submit”



Note: Only the information from the submitted form is sent to the server!

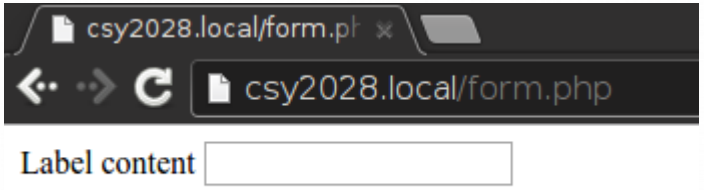
Labels

- Any form field can have a *label*
- Labels use the <label> element and can be linked to a form element using the *for* attribute
- To do this, the element must be given an *ID* attribute which is then referenced in the label's *for* attribute

Labels

- Once a label is assigned to an element, you can click on it and it will act like clicking on the referenced element

```
<label for="myinput">Label content</label> <input type="text" id="myinput" />
```

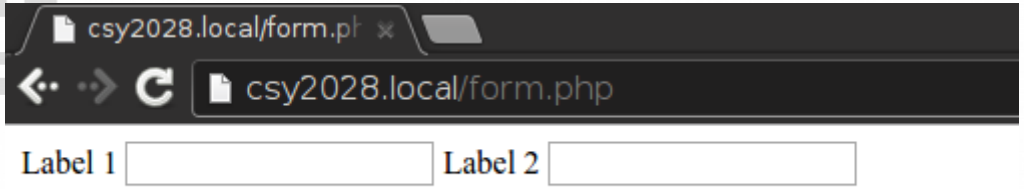


Labels

- Labels can be styled using CSS
- Some useful CSS is defining float/clear on inputs and labels to put them on rows
- Without CSS, all inputs and labels will appear on the same line

Labels

```
<form action="form.php" method="POST">

                <label for="myinput1">Label 1</label> <input type="text" id="myinput1" />
                <label for="myinput2">Label 2</label> <input type="text" id="myinput2" />


</form>
```

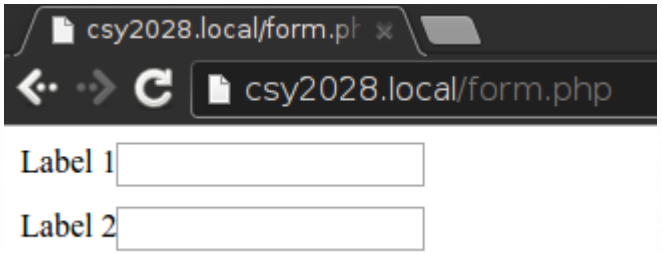


Labels

- By using float: left and clear: left you can position the elements on the same line (You don't need to use tables or divs to achieve this!)

```
<form action="form.php" method="POST">
                <label for="myinput1">Label 1</label> <input type="text" id="myinput1" />
                <label for="myinput2">Label 2</label> <input type="text" id="myinput2" />
</form>
```

```
label {float: left; clear: left; margin-bottom: 10px;}
input {float: left; margin-bottom: 10px;}
```

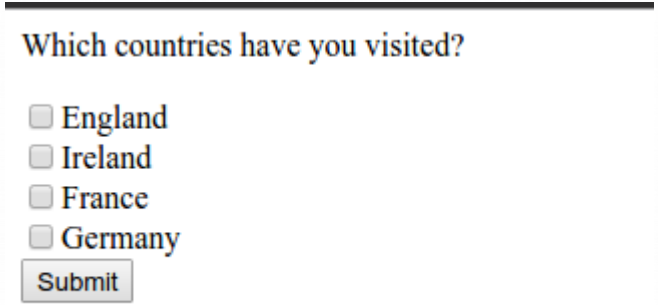


Arrays of form elements

- Sometimes it's useful to create an *array* of form elements.
- This is particularity useful for *checkboxes* where you may want to enable an unknown number of elements to be selected

Arrays of form elements

- Consider this form:



- With separate entities you will need an if statement to check for each of the available countries:

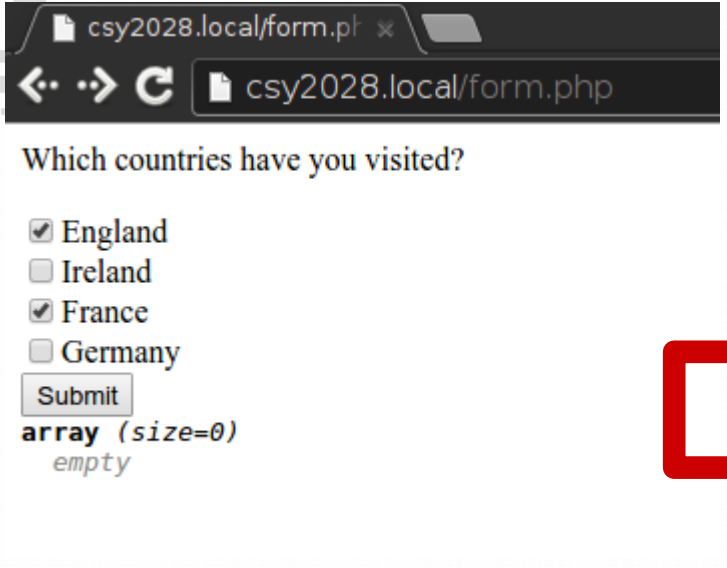
Arrays of form elements

```
<form action="form.php" method="POST">
        <p>Which countries have you visited?</p>
        <input type="checkbox" name="england" value="england" /> <label>England</label>
        <input type="checkbox" name="ireland" value="ireland" /> <label>Ireland</label>
        <input type="checkbox" name="france" value="france"  /> <label>France</label>
        <input type="checkbox" name="germany" value="germany"  /> <label>Germany</label>
        <input type="submit" name="submit" value="Submit" />
</form>


<?php


if (isset($_POST['england'])) {
        echo '<p>You have visited england</p>';
}

if (isset($_POST['ireland'])) {
        echo '<p>You have visited ireland</p>';
}

if (isset($_POST['france'])) {
        echo '<p>You have visited france</p>';
}

if (isset($_POST['germany'])) {
        echo '<p>You have visited germany</p>';
}
?>
```

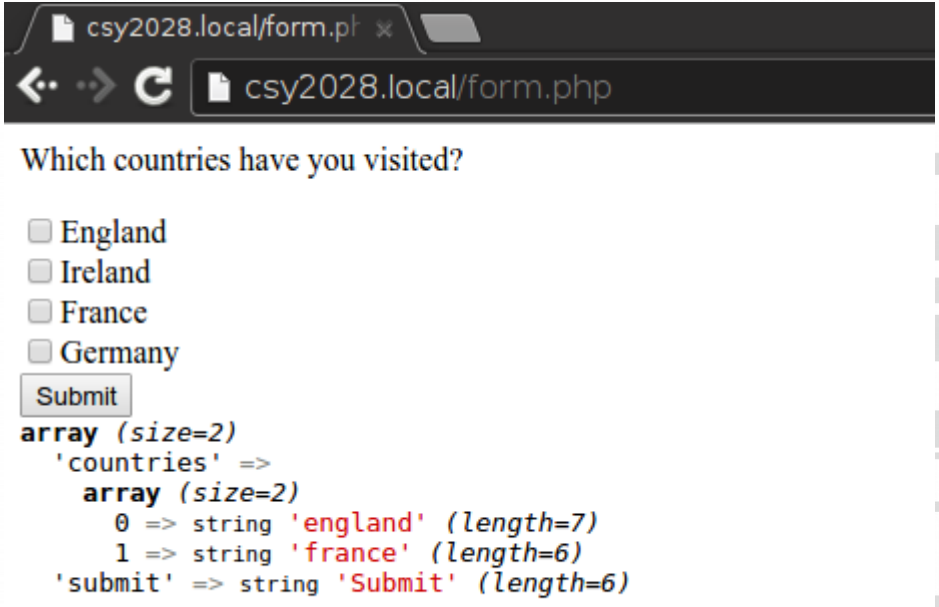
Arrays of form elements

- This needs a lot of code. Instead you can specify form elements as an *array*
- This is done by using square brackets after the name e.g.
- name="countries[]"
- All of the form elements should have the exact same *name*

```php
<form action="form.php" method="POST">

        <p>Which countries have you visited?</p>
        <input type="checkbox" name="countries[]" value="england" /> <label>England</label>
        <input type="checkbox" name="countries[]" value="ireland" /> <label>Ireland</label>
        <input type="checkbox" name="countries[]" value="france"  /> <label>France</label>
        <input type="checkbox" name="countries[]" value="germany"  /> <label>Germany</label>
        <input type="submit" name="submit" value="Submit" />

</form>
<?php

var_dump($_POST);
?>
```



Submit



Arrays of elements

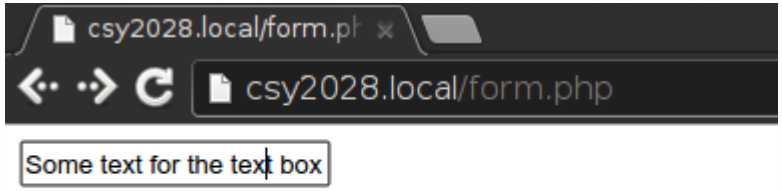
- When a set of elements are in an array you can loop through them using *foreach.*
- Only the boxes that were ticked will be set:

```
<form action="form.php" method="POST">

        <p>Which countries have you visited?</p>
        <input type="checkbox" name="countries[]" value="england" /> <label>England</label>
        <input type="checkbox" name="countries[]" value="ireland" /> <label>Ireland</label>
        <input type="checkbox" name="countries[]" value="france"  /> <label>France</label>
        <input type="checkbox" name="countries[]" value="germany"  /> <label>Germany</label>
        <input type="submit" name="submit" value="Submit" />

</form>

<?php
foreach ($_POST['countries'] as $id => $country) {
        echo '<p>You have visited '   . $country . '</p>';
}
?>
```

Setting element values

- You can pre-populate forms using the *value* attribute.
- This will place some text in the field when the page loads, without the user having to type it in
- The user can overwrite it or leave it

```
<input type="text" name="myinput" value="Some text for the text box" />
```



Setting textarea values

- For textareas you must specify the content between the opening and closing tags:

```
<textarea name="mytextarea" id="myinput" >
Some text for the text box
</textarea>
```

- Note: Textareas do not use the value attribute!

Setting select values

- To set the value of a <select> when the page loads element you must add selected=”selected” to the relevant option

```
<form action="form.php" method="POST">

        <select name="myselect">
                <option selected=”selected” value="One">Option 1</option>
                <option value="Two">Option 2</option>
                <option value="Three">Option 3</option>
        </select>
        <input type="submit" value="Submit" name="submit" />

</form>

<?php
var_dump($_POST);
?>
```

Exercise 2

- 1) Amend the rock-paper-scissors game from last week to use a <select> element and a POST form instead of links
- 2) Create a form that asks for the following information:
  - Firstname, surname, email address, favourite colour (use a select with suitable options), address (use a textarea to allow multiple lines). Do you accept the terms and conditions? Checkbox
  - When the form is submitted, display each of the submitted values in the format:
  - **First name**: [name]
  - **Surname**: [name]
  - **Do not use var_dump for this, print out each value with a description individually**

Exercise 2

- 3) Add validation to the form. If the "Do you agree to the terms and conditions?" checkbox is not ticked display an error message that says "You must accept the terms and conditions" and display the form again with all the information already inside it so the user does not need to retype everything
- 4) Add more validation to the form. Ensure there is a value inside each field. If there is not, display an error message that says which field(s) is missing a value and allow the user to correct their mistake
- 5) **Optional** Validate the user has entered a valid email address e.g. "something" is not valid but "something@something.com" is. You can use PHP's filter_var function to do this: http://php.net/manual/en/function.filter-var.php

Exercise 2

- 6) Create a lottery simulation by allowing the user to select any six numbers from 1-25
  - Hint: You should not need to write out the HTML code for 25 checkboxes!
- 7) Add validation to ensure they have selected exactly 6 numbers
- 8) When the form is submitted randomly generate the 6 winning numbers and ensure the same number cannot be selected twice!. Display the winning numbers on the page
- 9) Match the user's selected numbers to the winning numbers. Display many of the winning numbers they selected

```php
<form action="rockpaperscissors.php" method="POST">
        <select name="choice">
                <option value="rock">Rock</option>
                <option value="paper">Paper</option>
                <option value="scissors">Scissors</option>
        </select>

        <input type="submit" value="submit" />
</form>
<?php

if (isset($_GET['choice'])) {
    $computerChoices = ['rock', 'paper', 'scissors'];
    $randChoice = rand(0, 2);

    //random computer choice
    $computer = $computerChoices[$randChoice];
    $player = $_POST['choice'];

    if ($computer == $player) {
            echo 'Draw! Both players chose ' .  $player;
    }
    else {
            echo 'You chose ' . $player . ' and the computer chose ' . $computer . '. ';

            if ($computer == 'rock' && $player == 'scissors') {
                    echo 'Computer wins!';
            }
            else if ($computer == 'rock' && $player == 'paper') {
                    echo 'You win!';
            }
            else if ($computer == 'paper' && $player == 'rock') {
                    echo 'Computer wins!';
            }
            else if ($computer == 'paper' && $player == 'scissors') {
                    echo 'You win!';
            }
            else if ($computer == 'scissors' && $player == 'paper') {
                    echo 'Computer wins!';
            }
            else if ($computer == 'scissors' && $player == 'rock') {
                    echo 'You win!';
            }
    }
}
?>
```

```php
<?php
$valid = false;

if (isset($_POST['submit'])) {
    $valid = true;

    $nonempty = ['firstname', 'surname', 'color', 'email', 'address'];

    foreach ($nonempty as $field) {
        if ($_POST[$field] == '') {
            $valid = false;
            echo '<p>Please enter your ' . $field . '</p>';
        }
    }

    if (!isset($_POST['terms'])) {
        $valid = false;
        echo '<p>You must accept the terms and conditions</p>';
    }

    if (!filter_var($_POST['email'], FILTER_EMAIL)) {
        $valid = false;
        echo '<p>You must enter a valid email address</p>';
    }
}

if ($valid == true) {
    echo '<p>First name: ' . $_POST['firstname'] . '</p>';
    echo '<p>Surname: ' . $_POST['surname'] . '</p>';
    echo '<p>Email: ' . $_POST['email'] . '</p>';
    echo '<p>Favourite colour: ' . $_POST['colour'] . '</p>';
    echo '<p>Address: ' . $_POST['address'] . '</p>';
}
```

```php
else {
?>
<form action="signup.php" method="POST">
    <label for="firstname">First name</label>
    <input type="text" name="firstname" value="
    <?php if (isset($_POST['firstname'])) echo $_POST['firstname']; ?> />"

    <label for="surname">Surname</label>
    <input type="text" name="surname" value="
    <?php if (isset($_POST['surname'])) echo $_POST['surname']; ?>"
    />

    <label for="email">Email address</label>
    <input type="text" name="email" value="
    <?php if (isset($_POST['email'])) echo $_POST['email']; ?>"
    />

    <label for="colour">Favourite Colour?</label>
    <select name="colour">
        <option
        <?php
        if (isset($_POST['colour']) && $_POST['colour'] == 'Red') echo 'selected="selected"';
        ?> value="Red">Red</option>
        <option
        <?php
        if (isset($_POST['colour']) && $_POST['colour'] == 'Yellow') echo 'selected="selected"';
        ?> value="Yellow">Yellow</option>
        <option
        <?php
        if (isset($_POST['colour']) && $_POST['colour'] == 'Green') echo 'selected="selected"';
        ?> value="Green">Green</option>
        <option
        <?php
        if (isset($_POST['colour']) && $_POST['colour'] == 'Blue') echo 'selected="selected"';
        ?> value="Blue">Blue</option>
    </select>

    <label for="address">Adress</label>
    <textarea name="address">
    <?php if (isset($_POST['address'])) echo $_POST['address']; ?>
    </textarea>

    <label for="terms">Do you accept the terms and conditions?</label>
    <input type="checkbox" value="yes" />

        <input type="submit" value="submit" />
</form>
<?php
}
?>
```