Programming 1 Team Solo

The basics of web programming: Lecture #4: Processing form data from PHP

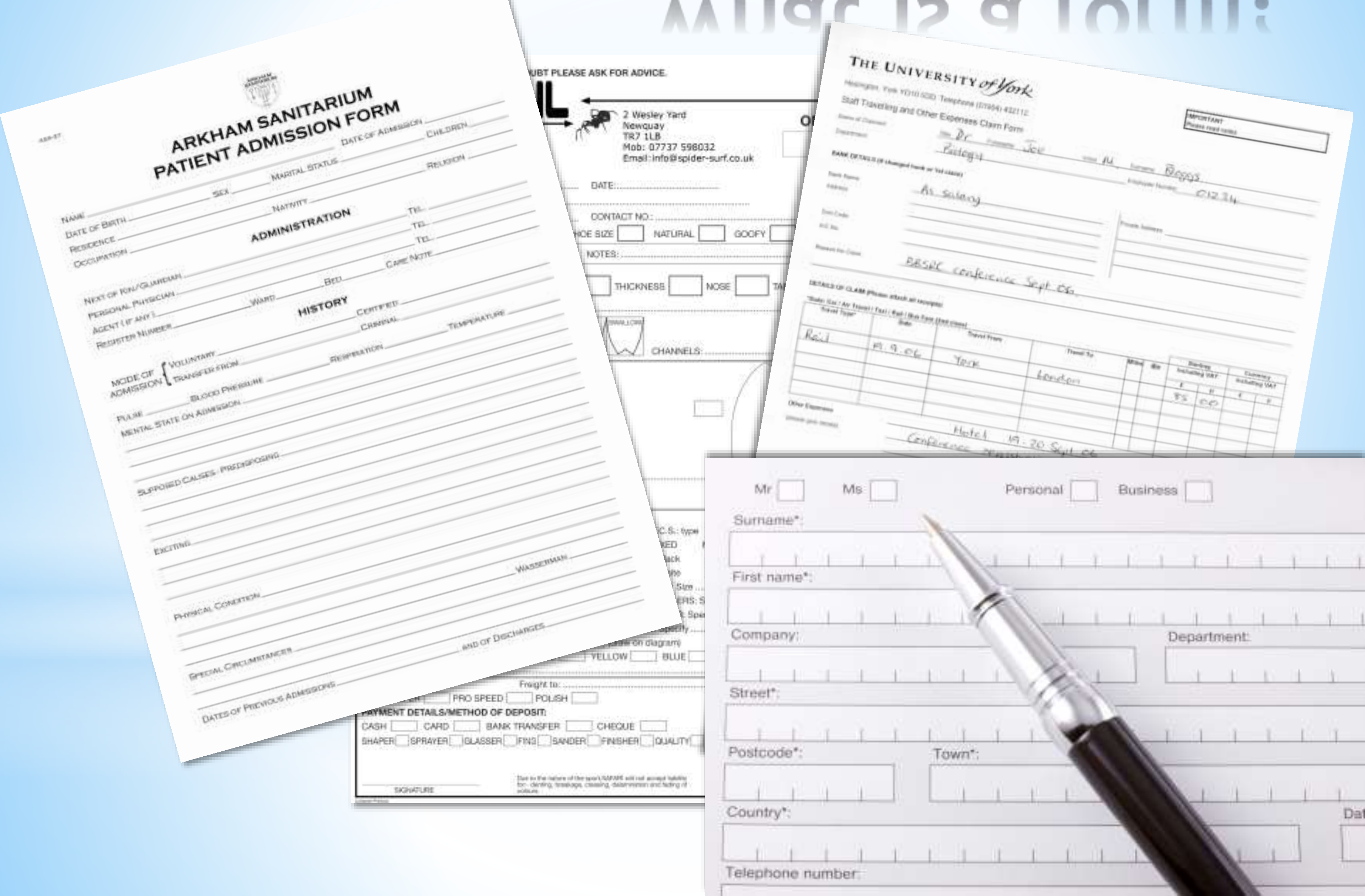
The story so far

- HTML pages describe static web content
  - They are stored by and delivered by a web server
  - A browser receives these pages from a web server and renders the HTML markup into something a human being can read
- PHP code allows for dynamic web content
  - When a browser requests a page from a server that includes some PHP, the server runs the PHP code first
  - The PHP code will do some processing
  - The PHP code will result in some HTML markup
    - This can be different each time the page is requested!
  - The HTML then gets sent to the browser
  - Net result: the end user sees a web page that changes

HTML forms

- HTML forms let us get data from the end user of a web page/site
- Form fields are encased in an enclosing <form> element
- There are different types of form fields for things like single lines of free text, larger text areas, pull down lists, checkboxes and radio buttons

What is a form?



ARKHAM SANITARIUM
PATIENT ADMISSION FORM

NAME ____ SEX ____ MARITAL STATUS ____ DATE OF ADMISSION ____ CHILDREN ____
DATE OF BIRTH ____ RELIGION ____
RESIDENCE ____ NATIVITY ____
OCCUPATION ____

ADMINISTRATION

TEL. ____
TEL. ____
TEL. ____
CARE NOTE ____

NEXT OF KIN/GUARDIAN ____
PERSONAL PHYSICIAN ____ BED ____
AGENT (IF ANY) ____ WARD ____
REGISTER NUMBER ____

HISTORY

CERTIFIED ____
CRIMINAL ____ TEMPERATURE ____
MODE OF ADMISSION { VOLUNTARY ____ TRANSFER FROM ____ RESPIRATION ____
PULSE ____ BLOOD PRESSURE ____
MENTAL STATE ON ADMISSION ____
SUPPOSED CAUSES: PREDISPOSING ____
EXCITING ____
WASSERMAN ____
PHYSICAL CONDITION ____
SPECIAL CIRCUMSTANCES ____ AND OF DISCHARGES ____
DATES OF PREVIOUS ADMISSIONS ____

UBT PLEASE ASK FOR ADVICE.

IL

2 Wesley Yard
Newquay
TR7 1LB
Mob: 07737 598032
Email: info@spider-surf.co.uk

OF

DATE: ....

CONTACT NO.: ....
HOE SIZE □ NATURAL □ GOOFY □
NOTES: ....
□ THICKNESS □ NOSE □ TA

SWALLOW

CHANNELS: ....

C.S.: type
RED
ack
ite
Size ...
ERS: S
Spe
specify ...
on diagram)
YELLOW □ BLUE □
Freight to: ....
ER □ PRO SPEED □ POLISH □
PAYMENT DETAILS/METHOD OF DEPOSIT:
CASH □ CARD □ BANK TRANSFER □ CHEQUE □
SHAPER □ SPRAYER □ GLASSER □ FINS □ SANDER □ FINISHER □ QUALITY □

SIGNATURE

THE UNIVERSITY *of York*
Heslington, York YO10 5DD Telephone (01904) 432112
Staff Travelling and Other Expenses Claim Form

IMPORTANT
Please read notes

Dr Joe M Biggs
Biology C1234
As salary
BBSRC conference Sept 06

DETAILS OF CLAIM (Please attach all receipts)

Travel Type	Date	Travel From	Travel To	Miles	Air	Sterling including VAT £	p	Currency including VAT £	p
Rail	19.9.06	York	London			85	00		

Other Expenses

Hotel 19-20 Sept 06
Conference

Mr □ Ms □ Personal □ Business □

Surname*:

First name*:

Company: Department:

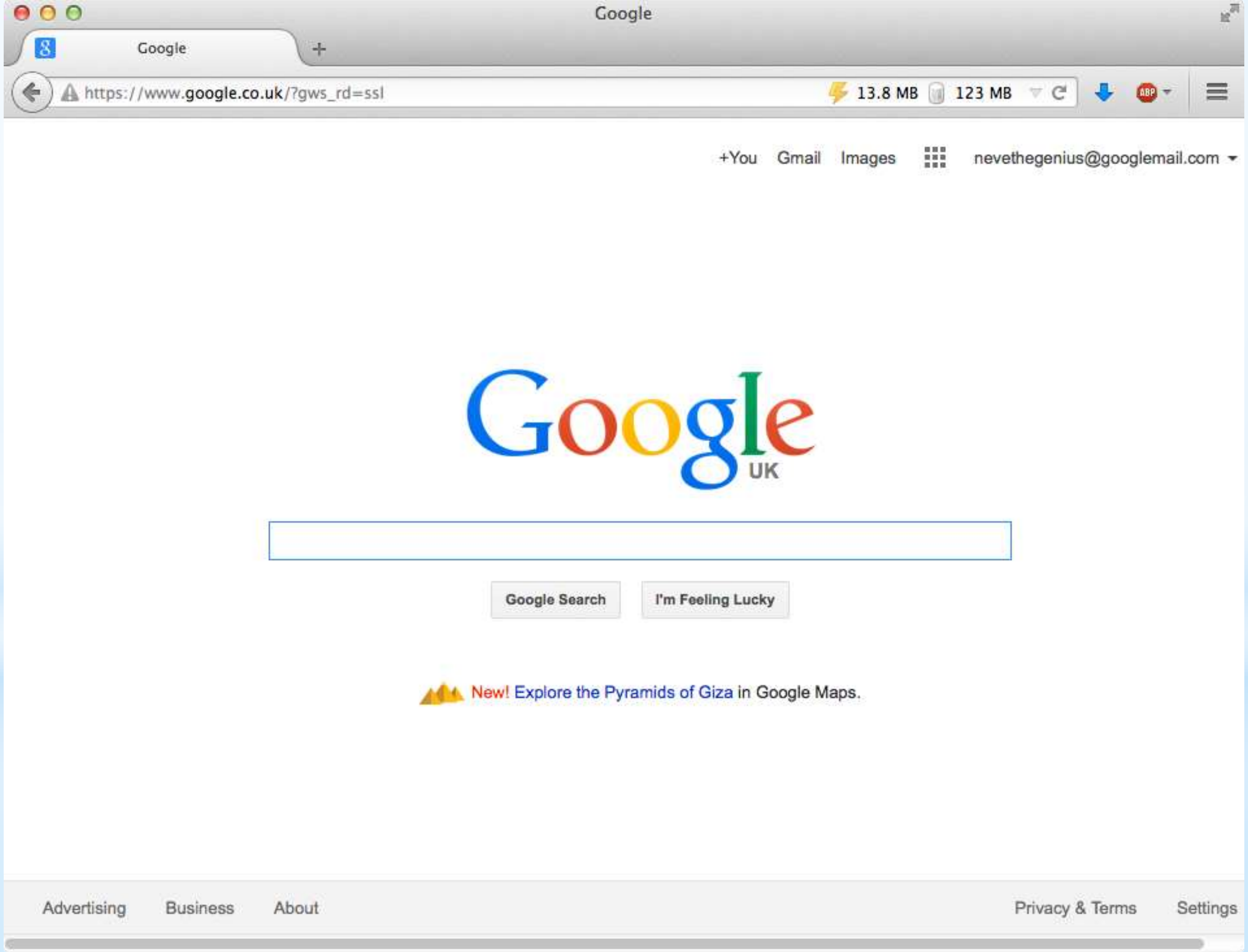
Street*:

Postcode*: Town*:

Country*: Dat

Telephone number:

The best known form on the web



Forms collect different forms of information

- A form might collect
  - Free text information
    - e.g. *Name*, *Address*, *What do you think about the price of fish?*
  - Selections from a pre-defined set of options
    - e.g. *Gender*
  - A positive or negative response to a question
    - e.g. *Have you ever done any programming before?*

- Online forms are no different!
- A variety of form controls exist to allow you to collect information from the user of a web page

How web forms work

1. A form is presented to the user within their web browser.
2. The user fills in the form fields
3. Most forms will have a "submit" button. Upon clicking this, the data that has been entered into the form is sent to a server (often the server which is hosting the web site from which the form originated, but not always)
4. A *server-side process* (usually a page in something like PHP, Perl, Python or similar) will process the data from the form. It might use other data sources to do so, e.g. databases, other files on the server hard disk, etc.
5. The server-side process will then generate a new page of HTML which represents the response to the user - i.e. any feedback or results to their input

How web forms work

1. A form is presented to the user within their web browser.

2. The u

2.5 Client-side Javascript processes the user's form field inputs and makes changes to the HTML elements of the current page to provide feedback to the user

3. Most this, the data that has been entered into the form is sent to a server (often the server which is hosting the web site from which the form originated, but not always)

4. A *server-side process* (usually a page in something like PHP, Perl, Python or similar) will process the data from the form. It might use other data sources to do so, e.g. databases, other files on the server hard disk, etc.

5. The server-side process will then generate a new page of HTML which represents the response to the user - i.e. any feedback or results to their input

The `<form>` element

- For any form that is going to be processed by a server, a **`<form>`** element must surround all of the form components
  - (think blocks within blocks)
- The **`<form>`** element will have (at least) an **`action`** attribute:

```
<form action="register.php">
…individual form components would go here…
</form>
```

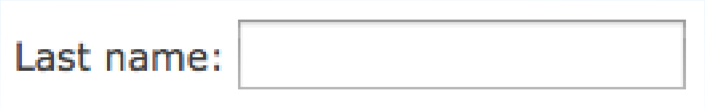
- The **`action`** attribute gives the URL for a page that will process the data that comes from the form
  - Usually, this will be something written in PHP, Python, Java, Perl or some other web programming platform…

Form controls: text input

- The **`<input>`** element creates a variety of different form components
- At its simplest, it will display a simple text field:

```
Last name:
<input type="text" name="username"/>
```

- This would display a simple, plain text field into which the user could type text:



Last name:

- (Q: Why does the input field appear next to the text?)

Form controls: text input

- <input> elements should contain (at least) a **`type`** attribute and a **`name`** attribute

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

- If the **`type`** is not given, then it assumes that the **`type`** should be **`text`** (i.e. a simple single line text field)
- The **`name`** uniquely identifies the field
- When the form is sent to a server for processing, the code that runs server-side knows which field is which based on the value of the **`name`** attributes

Form controls: text input

- There are a number of other attributes you can use on an input element
- **`size`** specifies how wide (in characters) the input element should be
  - If this is not given, 20 is usually assumed
  - You can also style and size your input elements with CSS
- **`maxlength`** specifies the maximum number of characters that can be physically typed into the input element

```
<input name="username" maxlength="6" size="4"/>
```

- Note that the **`size`** and **`maxlength`** can be different:
  - The user can happily type in more characters than the **`size`** (the text inside the input box will scroll)
  - However, if the user tries to type in more characters than the **`maxlength`**, the additional characters will be ignored
  - If the **`size`** is more than the **`maxlength`**, then there will be empty space at the right hand side of the text box

Form controls: text input

- You can also specify a default text value using the **`value`** attribute

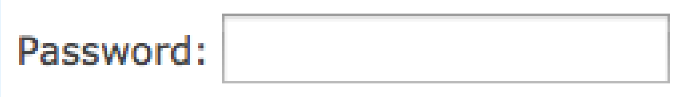
```
<input name="username" value="Paul"/>
```

- If this is given, then when the form loads the text box will already be filled in with this value…
  - …although if the user chooses they can delete the value and replace it with one of their own…

Form controls: passwords

- Passwords are one of the few controls that don't have an equivalent on a paper-based form
- They are pretty simple though:

```
Password:
<input type="password" name="pwd"/>
```

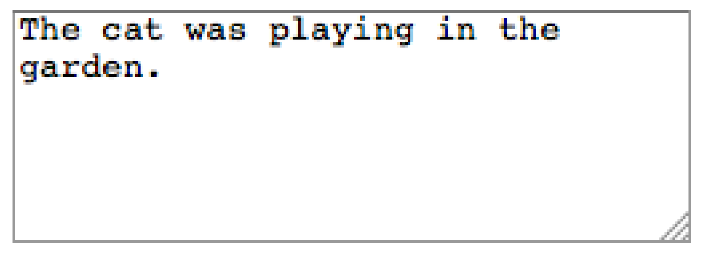


- You give a **`type`** of password
- The only difference between password fields and standard text fields is that anything the user types in is not visible on the screen

Form controls: larger text boxes

- To create a multiple line text box, you need to use the **`<textarea>`** element

```
<textarea cols="30" rows="10" name="story">
The cat was playing in the garden.
</textarea>
```

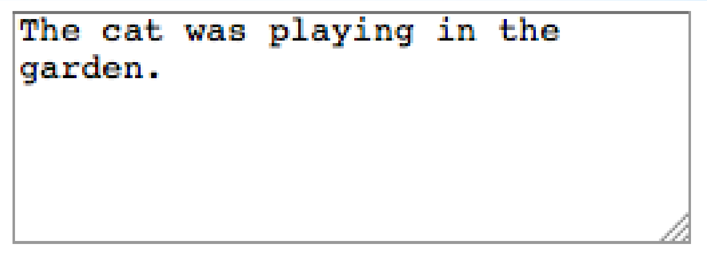


```
The cat was playing in the
garden.
```

Form controls: larger text boxes

- To create a multiple line text box, you need to use the **`<textarea>`** element

```
<textarea cols="30" rows="10" name="story">
The cat was playing in the garden.
</textarea>
```



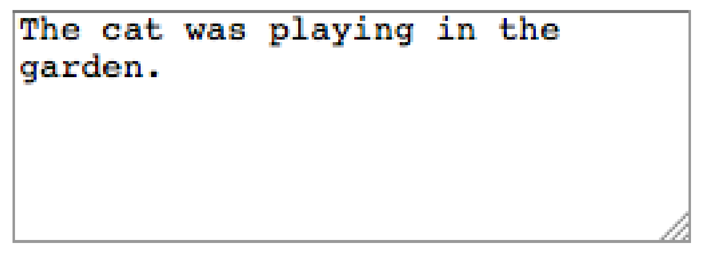
```
The cat was playing in the
garden.
```

- The **`rows`** and **`cols`** attributes specify the height (rows) and width (cols) of the textarea respectively. These are specified in characters (so this example is 30x10)

Form controls: larger text boxes

- To create a multiple line text box, you need to use the **`<textarea>`** element

```
<textarea cols="30" rows="10" name="story">
The cat was playing in the garden.
</textarea>
```



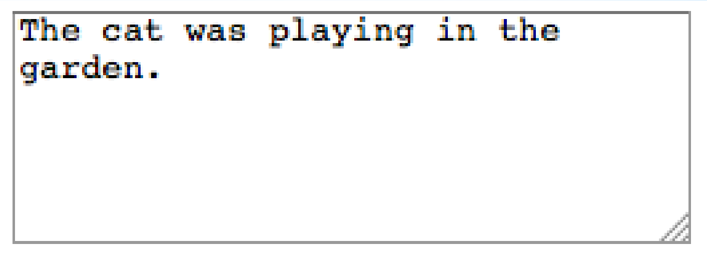
```
The cat was playing in the
garden.
```

- As usual, the name attribute specifies the unique name of this text area and distinguishes it from other form components

Form controls: larger text boxes

- To create a multiple line text box, you need to use the **`<textarea>`** element

```
<textarea cols="30" rows="10" name="story">
The cat was playing in the garden.
</textarea>
```



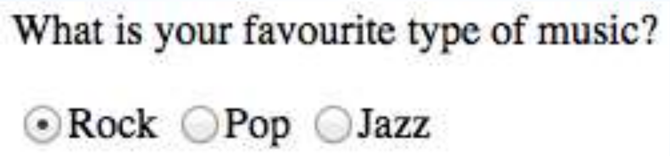
```
The cat was playing in the
garden.
```

- The text between the start and the end tags will be the default text in the text area when the form loads
- As with the **`value`** attribute on the input element, the user can delete this default text and type their own if they choose

Form controls: radio buttons

- Radio buttons allow the user to choose one options from a range of pre-defined responses:

```
<p>What is your favourite type of music?</p>
<input type="radio" name="genre" value="rock"/>Rock
<input type="radio" name="genre" value="pop"/>Pop
<input type="radio" name="genre" value="jazz"/>Jazz
```



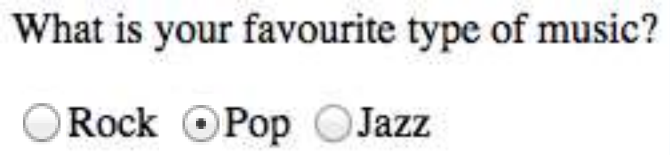
What is your favourite type of music?

◉ Rock ○ Pop ○ Jazz

Form controls: radio buttons

- Radio buttons allow the user to choose one options from a range of pre-defined responses:

```
<p>What is your favourite type of music?</p>
<input type="radio" name="genre" value="rock"/>Rock
<input type="radio" name="genre" value="pop"/>Pop
<input type="radio" name="genre" value="jazz"/>Jazz
```



What is your favourite type of music?

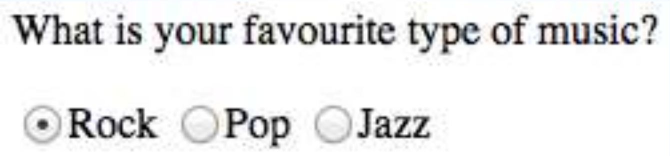
○ Rock ● Pop ○ Jazz

- Only one of the radio buttons can be selected at a time. If the user selects a new option, the previous one turns off.

Form controls: radio buttons

- Radio buttons allow the user to choose one options from a range of pre-defined responses:

```
<p>What is your favourite type of music?</p>
<input type="radio" name="genre" value="rock"/>Rock
<input type="radio" name="genre" value="pop"/>Pop
<input type="radio" name="genre" value="jazz"/>Jazz
```



What is your favourite type of music?

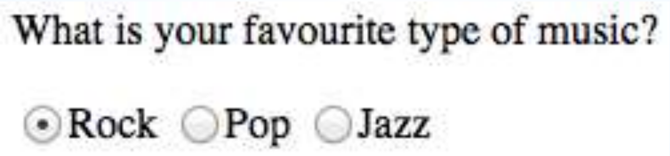
● Rock ○ Pop ○ Jazz

- All of the **`<input>`** elements that make up the group of radio buttons should have the same **`name`**. This is how the browser knows that the radio buttons go together as a group
  - …usually the radio buttons in a group will be in close proximity to each other on the page, although nothing say this has to be the case!

Form controls: radio buttons

- Radio buttons allow the user to choose one options from a range of pre-defined responses:

```
<p>What is your favourite type of music?</p>
<input type="radio" name="genre" value="rock"/>Rock
<input type="radio" name="genre" value="pop"/>Pop
<input type="radio" name="genre" value="jazz"/>Jazz
```



What is your favourite type of music?

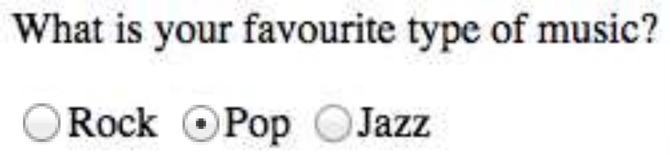
● Rock ○ Pop ○ Jazz

- The value of each radio button is what gets sent to the server if that particular option is selected when the form is submitted
  - ...this often closely matches the text (or "label") that accompanies the radio button, but doesn't have to

Form controls: radio buttons

- Radio buttons allow the user to choose one options from a range of pre-defined responses:

```
<p>What is your favourite type of music?</p>
<input type="radio" name="genre" value="rock"/>Rock
<input type="radio" name="genre" value="pop"
checked="checked"/>Pop
<input type="radio" name="genre" value="jazz"/>Jazz
```



What is your favourite type of music?

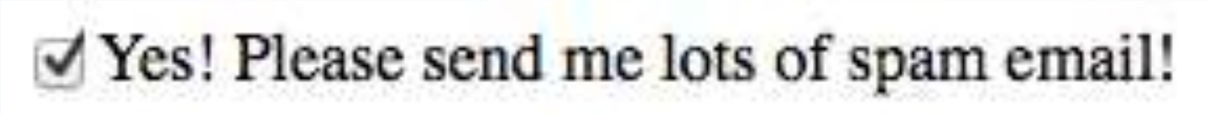
○ Rock ● Pop ○ Jazz

- You can also use the checked attribute to specify which radio button should be the default
    - ...if you leave this off, it is possible for your user to leave none of the options selected (although in some cases you might want this)

Form controls: checkboxes

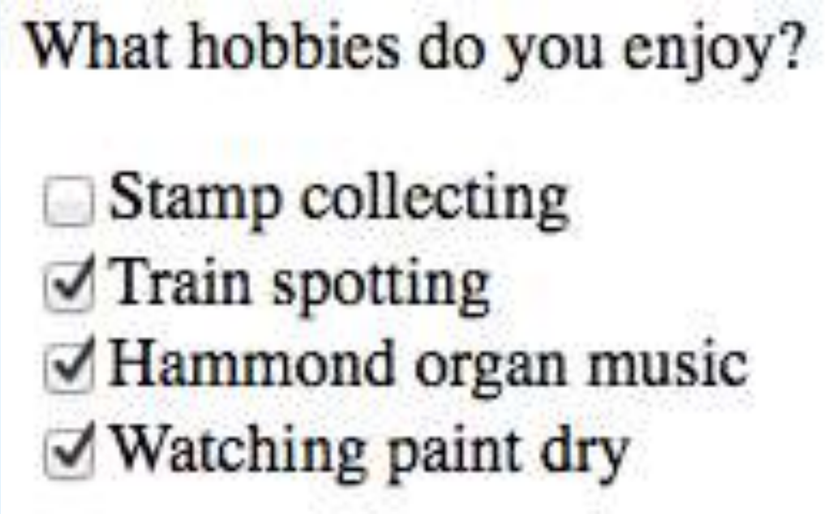
- Check boxes let the use give a positive or negative (yes or no) response to a single item
- There are two scenarios in which you might use checkboxes:

1. If you want to have a one shot yes or no item:



☑ Yes! Please send me lots of spam email!

2. If you want to have a list of items from which the user can select one or more options



What hobbies do you enjoy?

□ Stamp collecting
☑ Train spotting
☑ Hammond organ music
☑ Watching paint dry

Form controls: checkboxes

- If you want to have a one shot yes or no item:

```
<input type="checkbox" name="spam"
value="true"/> Yes! Please send me lots of spam
email!
```

- Use an **`input`** element of with a **`type`** of checkbox
- The server gets sent a field called whatever the **`name`** attribute is
- The value of this field will be whatever is in the **`value`** attribute
- As with radio buttons, you can use the **`checked`** attribute to make the box default to selected (i.e. ticked)

Form controls: checkboxes

- If you want to have a list of options which can have multiple things selected:

```
<p>What hobbies do you enjoy?</p>
<input type="checkbox" name="hobbies[]" value="stamps"/>
Stamp collecting<br/>
<input type="checkbox" name="hobbies[]" value="trains"/>
Train spotting<br/>
<input type="checkbox" name="hobbies[]" value="rimmerism"/>
Hammond organ music<br/>
<input type="checkbox" name="hobbies[]" value="paintdry"/>
Watching paint dry
```

- Note the `name` attributes!
  - If you want the different options to go to the server within the same field, you must use the same `name` for all of them
  - If you use the same name on checkboxes and you expect to be able to read them server-side in PHP, you must append [] to the name

Form controls: drop down lists

- A drop down list lets the user select one of a set of pre-defined options
- It does not use the input element – instead, a surrounding select element contains a number of option elements:

```
<p>Who is usually credited with the invention
of the World Wide Web?</p>
<select name="web">
  <option value="tim">Tim Berners-Lee</option>
  <option value="bill">Bill Gates</option>
  <option value="geoff">Geoff Hurst in the 1966
                          World Cup</option>
</select>
```

Form controls: drop down lists

- A drop down list lets the user select one of a set of pre-defined options
- It does not use the input element – instead, a surrounding select element contains a number of option elements:

```
<p>Who is usually credited with the invention
of the World Wide Web?</p>
<select name="web">
  <option value="tim">Tim Berners-Lee</option>
  <option value="bill">Bill Gates</option>
  <option value="geoff">Geoff Hurst in the 1966
                        World Cup</option>
</select>
```

The surrounding select element indicates that there is a drop down list

Form controls: drop down lists

- A drop down list lets the user select one of a set of pre-defined options
- It does not use the input element – instead, a surrounding select element contains a number of option elements:

```
<p>Who is usually credited with the invention
of the World Wide Web?</p>
<select name="web">
  <option value="tim">Tim Berners-Lee</option>
  <option value="bill">Bill Gates</option>
  <option value="geoff">Geoff Hurst in the 1966
                      World Cup</option>
</select>
```

Each option element represents a different option that the user can select

Form controls: drop down lists

- A drop down list lets the user select one of a set of pre-defined options
- It does not use the input element – instead, a surrounding select element contains a number of option elements:

```
<p>Who is usually credited with the invention
of the World Wide Web?</p>
<select name="web">
  <option value="tim" >Tim Berners-Lee</option>
  <option value="bill">Bill Gates</option>
  <option value="geoff">Geoff Hurst in the 1966
                         World Cup</option>
</select>
```

The **`value`** of an option is the text that will be sent to the server if the user chooses that option within their browser

Form controls: drop down lists

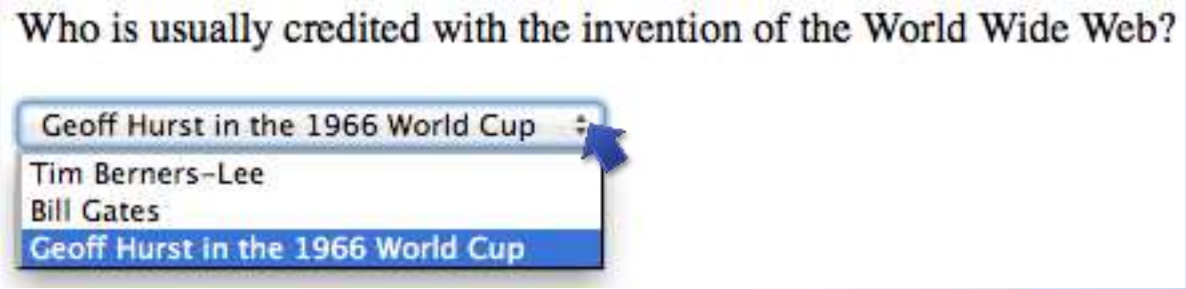
- A drop down list lets the user select one of a set of pre-defined options
- It does not use the input element – instead, a surrounding select element contains a number of option elements:

```
<p>Who is usually credited with the invention
of the World Wide Web?</p>
<select name="web" >
  <option value="tim">Tim Berners-Lee</option>
  <option value="bill">Bill Gates</option>
  <option value="geoff">Geoff Hurst in the 1966
                        World Cup</option>
</select>
```

The **name** of the select element contains the field name under which the results will be sent to the server.

Form controls: drop down lists

```
<p>Who is usually credited with the invention
of the World Wide Web?</p>
<select name="web">
  <option value="tim">Tim Berners-Lee</option>
  <option value="bill">Bill Gates</option>
  <option value="geoff" selected="selected">
        Geoff Hurst in the 1966 World Cup
  </option>
</select>
```



Who is usually credited with the invention of the World Wide Web?

Geoff Hurst in the 1966 World Cup
Tim Berners-Lee
Bill Gates
Geoff Hurst in the 1966 World Cup

Form controls: submit button

- A submit button allows the user to send the form to the server
- All the fields will be sent to the server, along with their values
- Any server-side process will be then able to read these field/value pairs
- The submit button is easy – a very basic **`input`** element with a type of **`submit`**

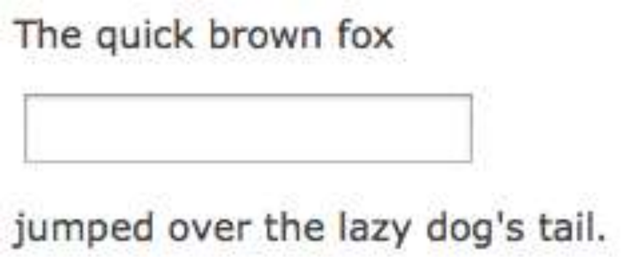
```
<input type="submit"/>
```

- You don't need to have a value or a name
- If you don't have a value, then the browser will fill in something like "Submit Query" as the text on the butto
  - ..so if you want the button to say something sensible, put it in your **`value`**

An aside: Inline v block elements

- The surrounding form element is an example of a *block* element
- This means that any surrounding content will always be separated from the form itself

```
The quick brown fox jumped
<form action="submit.php">
  <input type="text" name="username"/>
</form>
over the lazy dog's tail
```



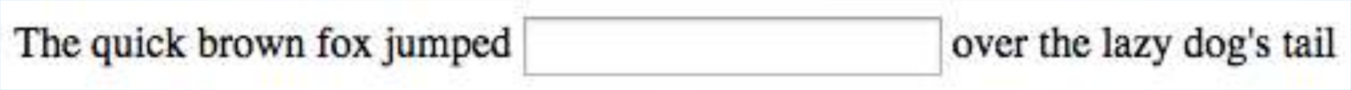
The quick brown fox

jumped over the lazy dog's tail.

An aside: Inline v block elements

- All of the elements within forms (e.g. input, select) are inline elements
- This means that any surrounding content will flow around the element unless you use paragraphs, line breaks etc:

```
<form action="submit.php">
  The quick brown fox jumped
    <input type="text" name="username"/>
  over the lazy dog's tail
</form>
```

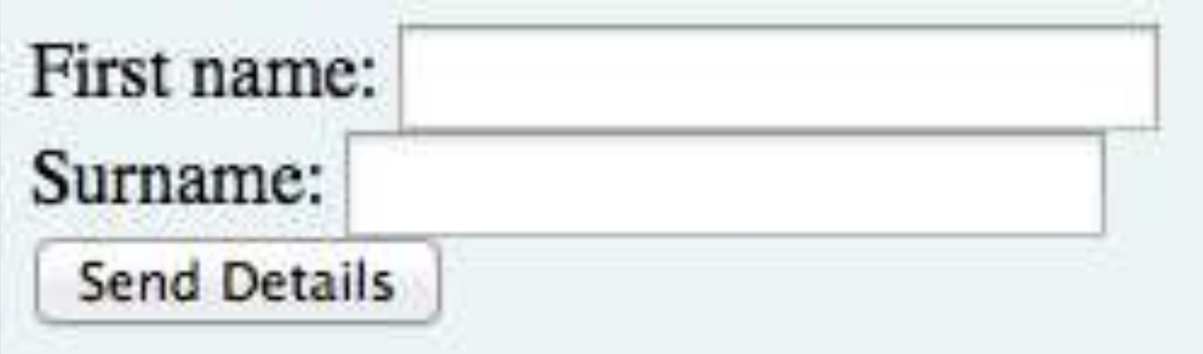


The quick brown fox jumped over the lazy dog's tail

The mechanics of form processing

- Consider the following form:

```
<form action="processdetails.php">
   First name: <input name="firstname"/><br/>
   Surname: <input name="surname"/><br/>
   <input type="submit" value="Send Details"/>
</form>
```

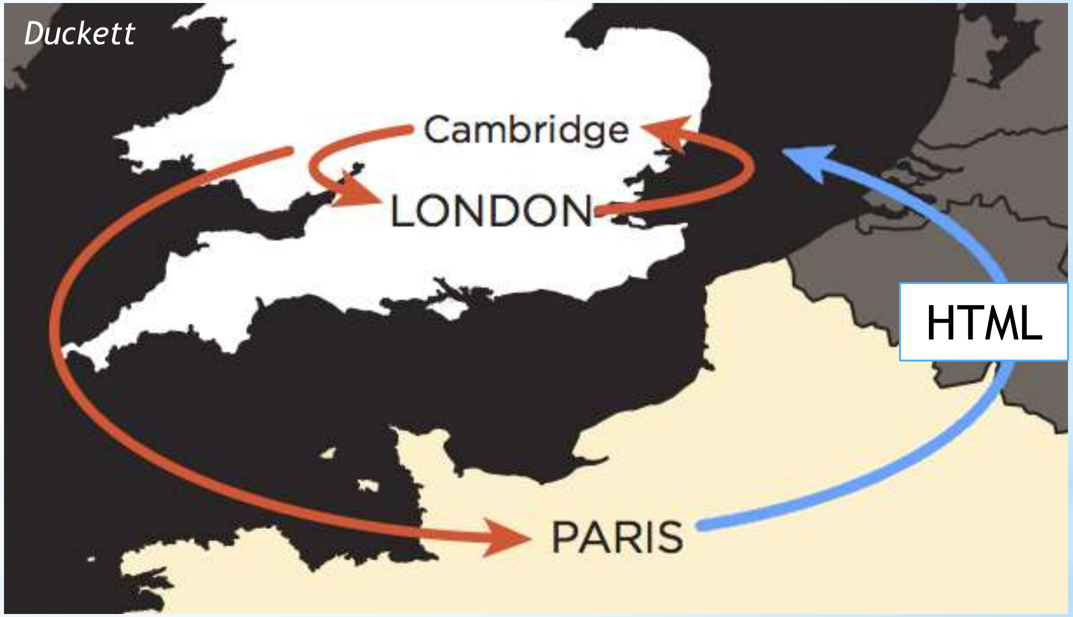


First name:

Surname:

The mechanics of form processing

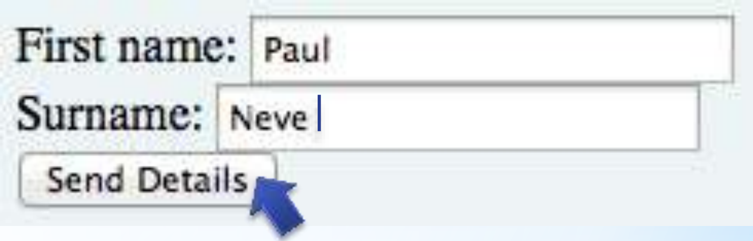
- Stage 1
  - The end user navigates their browser to a page with a form on it
  - The browser requests the page from the server
  - The server sends back a page of HTML (including the form markup)
  - The browser renders the page (including the form)



The mechanics of form processing

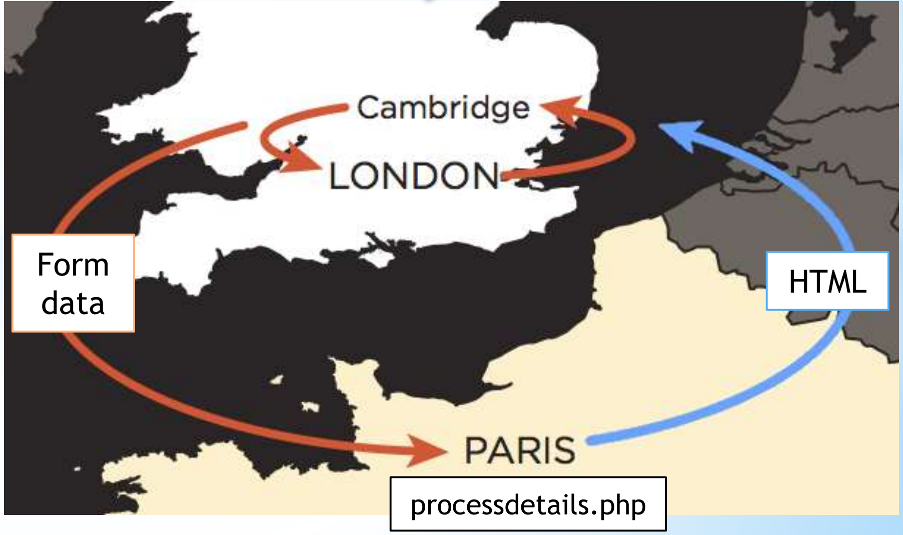
- Stage 2
  - The end user fills in the form within their browser page
  - The user clicks on the submit button on the form
  - The browser sends a request to the server for the URL that was in the **action** attribute of the form element
  - As part of the request, the browser also sends the form data
  - On the server, a server side process will examine the form data and generate a response in HTML
  - This HTML is sent back to the browser and then rendered like any other page

```
<form action="processdetails.php">
   First name: <input name="firstname"/><br/>
   Surname: <input name="surname"/><br/>
   <input type="submit" value="Send Details"/>
</form>
```



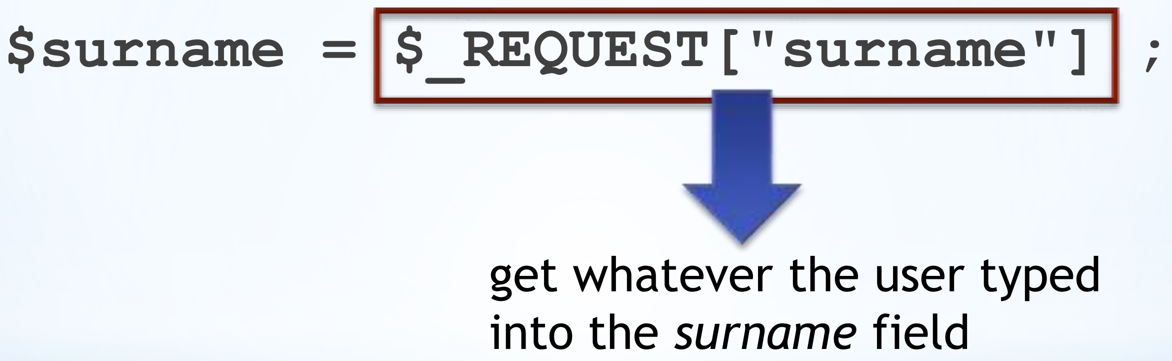
First name: Paul

Surname: Neve



Handling forms from PHP

- PHP has a special variable called $_REQUEST which is used for handling form input
- You can use this variable to read a specific form field thus:



```
$surname = $_REQUEST["surname"] ;
```

get whatever the user typed
into the *surname* field

Handling forms from PHP

- PHP has a special variable called $_REQUEST which is used for handling form input
- You can use this variable to read a specific form field thus:

```
$surname  = $_REQUEST["surname"];
```

put it into the new variable called *surname*

Handling forms from PHP

- PHP has a special variable called $_REQUEST which is used for handling form input
- You can use this variable to read a specific form field thus:

```
$banana = $_REQUEST["surname"];
```

put it into the new variable called *banana*

(you don't have to match any variable names in your PHP with your form field names - although often it makes good sense to do so!)

Handling forms from PHP

- A practical example:

index.html

```
<form action="processdetails.php">
   First name: <input name="firstname"/><br/>
   Surname: <input name="surname"/><br/>
   <input type="submit" value="Send Details"/>
</form>
```

NB: We are not showing complete pages here! We assume the usual structural requirements of an HTML page are present!

processdetails.php

```
<?php
  $surname = $_REQUEST["surname"];
  if ($surname == "Neve")
  {
     echo "<p>Welcome, oh mighty one!</p>";
  }
  else
  {
     echo "<p>Please leave, oh lowly one!</p>";
  }
?>
```

Handling forms from PHP

- A practical example:

index.html

```
<form action="processdetails.php">
   First name: <input name="firstname"/><br/>
   Surname: <input name="surname"/><br/>
   <input type="submit" value="Send Details"/>
</form>
```

The user starts by navigating the browser to this page.

processdetails.php

```
<?php
  $surname = $_REQUEST["surname"];
  if ($surname == "Neve")
  {
     echo "<p>Welcome, oh mighty one!</p>";
  }
  else
  {
     echo "<p>Please leave, oh lowly one!</p>";
  }
?>
```

Handling forms from PHP

- A practical example:

index.html

```
<form action="processdetails.php">
   First name: <input name="firstname"/> <br/>
   Surname: <input name="surname"/> <br/>
   <input type="submit" value="Send Details"/>
</form>
```

In their browser window, they fill in the form fields represented by these two elements.

processdetails.php

```
<?php
  $surname = $_REQUEST["surname"];
  if ($surname == "Neve")
  {
     echo "<p>Welcome, oh mighty one!</p>";
  }
  else
  {
     echo "<p>Please leave, oh lowly one!</p>";
  }
?>
```

Handling forms from PHP

- A practical example:

index.html

```
<form action="processdetails.php">
   First name: <input name="firstname"/><br/>
   Surname: <input name="surname"/><br/>
   <input type="submit" value="Send Details"/>
</form>
```

When they have completed the form, they click this button.

processdetails.php

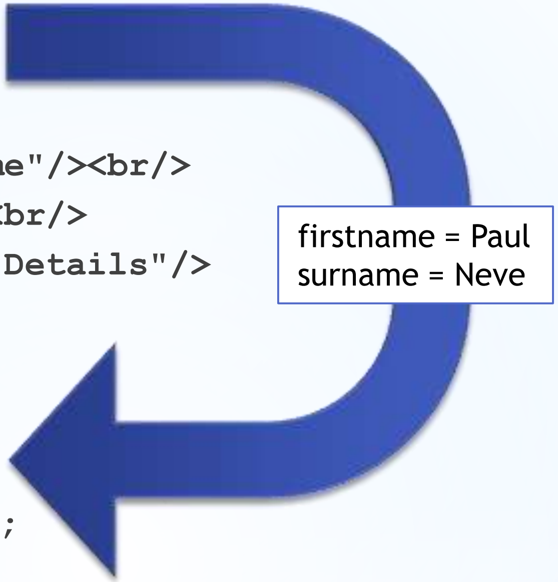
```
<?php
  $surname = $_REQUEST["surname"];
  if ($surname == "Neve")
  {
     echo "<p>Welcome, oh mighty one!</p>";
  }
  else
  {
     echo "<p>Please leave, oh lowly one!</p>";
  }
?>
```

Handling forms from PHP

- A practical example:

index.html

```
<form action="processdetails.php" >
   First name: <input name="firstname"/><br/>
   Surname: <input name="surname"/><br/>
   <input type="submit" value="Send Details"/>
</form>
```



firstname = Paul
surname = Neve

processdetails.php

```
<?php
  $surname = $_REQUEST["surname"];
  if ($surname == "Neve")
  {
     echo "<p>Welcome, oh mighty one!</p>";
  }
  else
  {
     echo "<p>Please leave, oh lowly one!</p>";
  }
?>
```

The act of clicking the button makes the browser send a request to the server for the processdetails.php page. As part of the request all the values of the form fields are included.

Handling forms from PHP

- A practical example:

index.html

```
<form action="processdetails.php" >
   First name: <input name="firstname"/><br/>
   Surname: <input name="surname"/><br/>
   <input type="submit" value="Send Details"/>
</form>
```

processdetails.php

```
<?php
  $surname = $_REQUEST["surname"];
  if ($surname == "Neve")
  {
     echo "<p>Welcome, oh mighty one!</p>";
  }
  else
  {
     echo "<p>Please leave, oh lowly one!</p>";
  }
?>
```

processdetails.php contains PHP code which is run server side. Within the code is a reference to the special **`$_REQUEST`** variable. Otherwise, it's just PHP code using the standard constructs you've all seen before…

One crucial concept

- Unlike programs you've written previously in languages like Banana, where you could mix input and processing, PHP has to handle user input in a batch

Banana

```
display "What is your name?"
get name
display "Hello, "+name
display "What is your age?"
get age
if (age >= 18)
  display "Go and have a beer!"
else
  display "Go and have a coke!"
endif
```

HTML and PHP

```
<form action="beercheck.php">
  <p>What is your name?</p>
  <input name="name"/>
  <p>What is your age?</p>
  <input name="age"/><br/>
  <input type="submit"/>
</form>

<?php
  $name = $_REQUEST["name"];
  $age = $_REQUEST["age"];
  echo "<p>Hello, ".$name." </p>";
  if ($age >= 18)
  {
     echo "<p>Go and have a beer!</p>";
  }
  else
  {
     echo "<p>Go and have a coke!</p>";
  }
?>
```

One crucial concept

- Unlike programs you've written previously in languages like Banana, where you could mix input and processing, PHP has to handle user input in a batch

Banana

```
display "What is your name?"
get name
display "Hello, "+name
display "What is your age?"
get age
if (age >= 18)
  display "Go and have a beer!"
else
  display "Go and have a coke!"
endif
```

HTML and PHP

```
<form action="beercheck.php">
  <p>What is your name?</p>
  <input name="name"/>
  <p>What is your age?</p>
  <input name="age"/><br/>
  <input type="submit"/>
</form>
```

```
<?php
  $name = $_REQUEST["name"];
  $age = $_REQUEST["age"];
  echo "<p>Hello, ".$name." </p>";
  if ($age >= 18)
  {
     echo "<p>Go and have a beer!</p>";
  }
  else
  {
     echo "<p>Go and have a coke!</p>";
  }
?>
```

Another crucial concept

- Often, your "program" will be split across two files – one which is responsible for prompting your user for input, and the other which is responsible for processing the input and providing a result:

index.html

```
<form action="beercheck.php">
  <p>What is your name?</p>
  <input name="name"/>
  <p>What is your age?</p>
  <input name="age"/><br/>
  <input type="submit"/>
</form>
```

beercheck.php

```
<?php
  $name = $_REQUEST["name"];
  $age = $_REQUEST["age"];
  echo "<p>Hello, ".$name." </p>";
  if ($age >= 18)
  {
     echo "<p>Have a beer!</p>";
  }
  else
  {
     echo "<p>Have a coke!</p>";
  }
?>
```

Some more practical examples

- Counting application (again, these are excerpts from structurally complete files)

index.html

```
<form action="count.php">
  <p>Please type a number to
      count to:</p>
  <input name="number"/>
  <input type="submit"/>
</form>
```

count.php

```
<ul>
  <?php
    $max = $_REQUEST["number"];
    for ($count = 0; $count < $max; $count++)
    {
      echo "<li>".$count."</li>";
    }
  ?>
</ul>
```

Some more practical examples

- Counting application (again, these are excerpts from structurally complete files)

index.html

```
<form action="count.php">
  <p>Please type a number to
      count to:</p>
  <input name="number"/>
  <input type="submit"/>
</form>
```

The user clicks this…

count.php

```
<ul>
  <?php
    $max = $_REQUEST["number"];
    for ($count = 0; $count < max; $count++)
    {
      echo "<li>".$count."</li>";
    }
  ?>
</ul>
```

Some more practical examples

- Counting application (again, these are excerpts from structurally complete files)

To send this..

index.html

```
<form action="count.php">
   <p>Please type a number to
        count to:</p>
   <input name="number"/>
   <input type="submit"/>
</form>
```

count.php

```
<ul>
  <?php
    $max = $_REQUEST["number"];
    for ($count = 0; $count < max; $count++)
    {
       echo "<li>".$count."</li>";
    }
  ?>
</ul>
```

...to this.

Building a form dynamically

- Your PHP code can output any HTML you like…
- …so nothing stops you writing PHP that builds a form and the field elements within that form

- Consider this specification:
  - A web application is needed that will generate three random numbers between 1 and 10. It will then ask the user to choose which number is their favourite. It should then display if their chosen number is more than five, equal to five or less than five.

Building a form dynamically

- Consider this specification:
  - A web application is needed that will generate three random numbers between 1 and 10. It will then ask the user to choose which number is their favourite. It should then display if their chosen number is more than five, equal to five or less than five.

- How many individual HTML/PHP pages will there be? Which will be static (pure HTML) and which will be dynamic (includes PHP)?

Building a form dynamically

- Consider this specification:
  - A web application is needed that will generate three random numbers between 1 and 10. It will then ask the user to choose which number is their favourite. It should then display if their chosen number is more than five, equal to five or less than five.

- How many individual HTML/PHP pages will there be? Which will be static (pure HTML) and which will be dynamic (includes PHP)?
  - Break things down into processing and input
  - Whenever there is input, you can consider this the "end of a page"

Building a form dynamically

- Consider this specification:
  - A web application is needed that will generate three random numbers between 1 and 10. It will then ask the user to choose which number is their favourite. It should then display if their chosen number is more than five, equal to five or less than five.

- Our first page will be the part of the specification in green
- Our second page will be the part in blue

- What kind of pages will they be? Static HTML only, or will they need PHP code?

Building a form dynamically

- Page 1
  - A web application is needed that will **generate three random numbers between 1 and 10**. It will then ask the user to choose which number is their favourite.
- Page 2
  - It should then display **if their chosen number is more than five, equal to five or less than five**.

- Both pages have dynamic elements (highlighed in **orange**)
- So both pages will need to include PHP code

Building a form dynamically

- Page 1

1. Generate our random numbers
   - PHP code:

```
$num1 = rand(1,10);
$num2 = rand(1,10);
$num3 = rand(1,10);
```

2. Present the numbers in a form that allows the user to choose one of them
   - The way in which a user of a web app supplies input to the app is through a form
   - So, we will need to generate a form that includes the numbers and lets the user select one

Building a form dynamically

- Page 1

2. (continued) Present the numbers in a form that allows the user to choose one of them
   - …we will need to generate a form that includes the numbers and lets the user select one
   - We could use radio buttons OR a select box

Building a form dynamically

- We need to write PHP `echo` statements that will generate HTML to create radio buttons
- For each radio button, we will need HTML like

```
<input type="radio" name="favenum" value="1">
Number 1<br/>
```

- If we were `echo`'ing this HTML, we might do something like

```
echo '<input type="radio" name="favenum" value="1">';
echo 'Number 1<br/>';
```

Building a form dynamically

- We need to write PHP **echo** statements that will generate HTML to create radio buttons
- For each radio button, we will need HTML like

```
<input type="radio" name="favenum" value="1">
Number 1<br/>
```

- If we were **echo**'ing this HTML, we might do something like

why the single quotes rather than double quotes?

```
echo ' <input type="radio" name="favenum" value="1"> ';
echo 'Number 1<br/>';
```

Building a form dynamically

- We need to write PHP **`echo`** statements that will generate HTML to create radio buttons
- For each radio button, we will need HTML like

```
<input type="radio" name="favenum" value="1">
Number 1<br/>
```

- The grey text stays the same each time – only the numbers need to change
- Our PHP will need to use string constants and concatenate our random number variables onto the text that stays the same

Building a form dynamically

- We need to write PHP **`echo`** statements that will generate HTML to create radio buttons
- For each radio button, we will need HTML like

```
<input type="radio" name="favenum" value="1">
Number 1<br/>
```

- The grey text stays the same each time – only the numbers need to change
- Our PHP will need to use string constants and concatenate our random number variables onto the text that stays the same

```
echo '<input type="radio" name="favenum" value="'.$num1.'"/>';
echo 'Number '.$num1."<br/>";
```

Building a form dynamically

- We need to write PHP **echo** statements that will generate HTML to create radio buttons
- For each radio button, we will need HTML like

```
<input type="radio" name="favenum" value="1">
Number 1<br/>
```

- The grey text stays the same each time – only the numbers need to change
- Our PHP will need to use string constants and concatenate our random number variables onto the text that stays the same

```
echo '<input type="radio" name="favenum" value="'. $num1.' "/>';
echo 'Number'. $num1."<br/>";
```

- Parts highlighed with blue are constants
- Those in red are variables
- Note the use of the full stop symbol . for concatenation!

Building a form dynamically

- Page 1

2. (continued) Present the numbers in a form that allows the user to choose one of them
   - …we will need to generate a form that includes the numbers and lets the user select one
   - We could use **radio buttons** OR a select box

```
<form action="choose.php">
  <?php
    echo '<input type="radio" name="favenum" value="'.$num1.'"/>';
    echo 'Number '.$num1."<br/>";
    echo '<input type="radio" name="favenum" value="'.$num2.'"/>';
    echo 'Number '.$num2."<br/>";
    echo '<input type="radio" name="favenum" value="'.$num3.'"/>';
    echo 'Number '.$num3."<br/>";
  ?>
  <input type="submit"/>
</form>
```

Building a form dynamically

- Page 1

2. (continued) Present the numbers in a form that allows the user to choose one of them
   - …we will need to generate a form that includes the numbers and lets the user select one
   - We could use radio buttons OR **a select box**

```
<form action="choose.php">
  <select name="favenum">
  <?php
    echo '<option value="'.$num1.'">Number '.$num1.'</option>';
    echo '<option value="'.$num2.'">Number '.$num2.'</option>';
    echo '<option value="'.$num3.'">Number '.$num3.'</option>';
  ?>
  </select>
  <input type="submit"/>
</form>
```

Building a form dynamically

- Complete page (index.php)

```
<?php
  $num1 = rand(1,10);   $num2 = rand(1,10);  $num3 = rand(1,10);
?>
<html>
<head>
   <title>Random number app</title>
</head>
<body>
<p>Choose your favourite random number:</p>
<form action="choose.php">
   <?php
      echo '<input type="radio" name="favenum" value="'.$num1.'"/>';
      echo 'Number '.$num1."<br/>";
      echo '<input type="radio" name="favenum" value="'.$num2.'"/>';
      echo 'Number '.$num2."<br/>";
      echo '<input type="radio" name="favenum" value="'.$num3.'"/>';
      echo 'Number '.$num3."<br/>";
   ?>
   <input type="submit"/>
</form>
</body>
</html>
```

Building a form dynamically

- What the server might send to the browser, and what the browser would "see":

```
<html>
<head>
   <title>Random number app</title>
</head>
<body>
 <p>Choose your favourite random number:</p>
 <form action="choose.php">
  <input type="radio" name="favenum" value="1">
  Number 1<br/>
  <input type="radio" name="favenum" value="7">
  Number 7<br/>
  <input type="radio" name="favenum" value="4">
  Number 4<br/>
  <input type="submit"/>
 </form>
</body>
</html>
```

Building a form dynamically

- This is now just an HTML form like any other
- The browser doesn't care that the form fields were built dynamically in PHP – in fact it doesn't even know
- The user can then interact with the form, supply data, make choices, in the normal way
- When they click the **submit** button, the form data gets sent to the URL in the `action` attribute of the form element just as normal:

```
<form action="choose.php">
```

Building a form dynamically

- Consider this specification (reminder)
  - A web application is needed that will generate three random numbers between 1 and 10. It will then ask the user to choose which number is their favourite. It should then display if their chosen number is more than five, equal to five or less than five.
- The second highlighted part is now what we need to handle in choice.php

1. Get the data that was submitted in the previous form in the input element named **`favenum`**
   - e.g. `$num = $_REQUEST["favenum"];`

2. Process the number and display the result
   -
```
if ($num < 5)
{
     echo "<p>Less than 5</p>";
}
else if ($num == 5)
{
     echo "<p>Equal to 5</p>";
}
else
{
  echo "<p>More than 5</p>";
}
```

Building a form dynamically

- Complete page (choose.php)

```
<html>
  <head>
     <title>Random number checker</title>
  </head>
  <body>
  <p>Your random number is
  <?php
     $num = $_REQUEST["favenum"];
     if ($num < 5)
     {
       echo "Less than 5";
     }
     else if ($num == 5)
     {
       echo "Equal to 5";
     }
     else
     {
       echo "More than 5";
     }
  ?>
  </p>
  </body>
</html>
```

Double quotes, single quotes and "escaping"

- This was a clunky line of code, wasn't it…?

```
echo '<input type="radio" name="favenum" value="'.$num1.'"/>';
```

- We used single quotes around the string literals in our echo statement so we could have double quotes within the string we were building
- We could have flipped things round and had single quotes in our HTML, and used double quotes to delimit the string literals, i.e.

```
echo "<input type='radio' name='favenum' value='".$num1."'/>";
```

- Alternatively, we could have *escaped* our quotes

Double quotes, single quotes and "escaping"

- Certain symbols are called *escape characters*
- These symbols are those that are not easily represented within the rules that govern strings in a given programming language
  - Maybe they break delimiting rules – e.g. quotes
  - Maybe they are non-printable symbols which have other meaning in a given language – e.g. a carriage return
- Escape characters are represented by an escape sequence when they need to appear in a string literal

Double quotes, single quotes and "escaping"

- Some useful escape sequences:

Sequence	Meaning
\"	Double quote
\t	Horizontal tab
\\	Backslash
\$	Dollar sign

Double quotes, single quotes and "escaping"

- So, if we wanted to delimit our previous echo statement with double quotes, and still have double quotes within our HTML that was being generated, we could do:

```
echo "<input type=\"radio\" name=\"favenum\" value=\"".$num1."\"/>";
```

Double quotes, single quotes and "escaping"

- So, if we wanted to delimit our previous echo statement with double quotes, and still have double quotes within our HTML that was being generated, we could do:

```
echo "<input type=\"radio\" name=\"favenum\" value=\"".$num1."\" />";
```

Double quoted strings and variables

- One of the advantages of using double quotes to delimit our strings is that you don't need to use concatenation to append or insert variable values
- Unlike other programming languages, any variable names that are placed within a double quoted string literal will be expanded – the value of the variable will appear in the output
- So, there is no need to do

```
echo "<input type=\"radio\" name=\"favenum\" value=\"".$num1."\"/>";
```

- We could just do

```
echo "<input type=\"radio\" name=\"favenum\" value=\"$num1\"/>";
```

- Or even (if we were willing to accept single quotes in our HTML output)

```
echo "<input type='radio' name='favenum' value='$num1'/>";
```

Double quotes strings and variables

- This is a PHP-specific thing!
- Don't get in the habit of relying on this – you can't do it in other languages
- Consider
  - PHP:
    - `echo "Hello there, $name";`
  - Banana:
    - `display "Hello there, name"`
- PHP variables are easily distinguished from other things because of their leading dollar sign
- In most other languages, variables easily "blend in" to other text, so this sort of feature isn't possible

Each page is an island…

- One important concept to grasp is that each individual page has no "sight" of any previous pages

page1.php

```
<?php
   $myNum = rand(1,10);
   echo "<p>My number was $mynum</p>"
?>
<a href="page2.php">Next page</a>
```

page2.php

```
<?php
   $echo "<p>The number was $mynum</p>";
?>
```

Each page is an island…

- One important concept to grasp is that each individual page has no "sight" of any previous pages

page1.php

```
<?php
   $myNum = rand(1,10);
   echo "<p>My number was $mynum</p>"
?>
<a href="page2.php">Next page</a>
```

page2.php

```
<?php
   $echo "<p>The number was $mynum</p>";
?>
```

WOULD NOT WORK!

HTTP is *stateless*

- HTTP – the protocol which transfers web pages between client and server – is *stateless* – no state is maintained between requests
- In English: each individual request for a page exists in its own bubble
- Short version: there is no way (within the HTTP protocol itself, at least) to store data and then recall it later down the line

Simulating state

- There are ways with which we can simulate state
- Cookies – small files that are stored on your computer and the contents of which get sent with every request for a page
- Hidden input fields:

page1.php

```
<?php
   $myNum = rand(1,10);
   echo "<p>My number was $mynum</p>"
?>
<form action="page2.php">
<?php echo "<input type='hidden' name="banana" value='$mynum'/>" ?>
<input type="submit" value="Next page"/>
</form>
```

page2.php

```
<?php
   $mynum = $_REQUEST["banana"];
   $echo "<p>The number was $mynum</p>";
?>
```

The PHP session API

- PHP's session API will simulate state for you
- Behind the scenes, it uses whatever technique (usually cookies) is needed to transmit data between pages
- Ultimately, all you need to know is that if you put something in a PHP *session variable*, it will be available across all the pages of your website for the duration of the user's session

PHP session variables

- At the VERY START of your page, you should have the command **`session_start();`**
  - (this is PHP code, so make sure it is enclosed in **`<?php ?>`** tags)
- Once you have started a session on a page, you can put data into it using the $_SESSION variable, e.g.

  -
    ```
    $_SESSION["name"] = "Paul";
    ```
- You can extract data from a session variable simply by referring to the variable, e.g.

  -
    ```
    $username = $_SESSION["name"];
    echo "<p>Your name is $username</p>";
    ```
- This can be done on the same page or on a subsequent page

PHP session variables

- So the previous page where we used a hidden input field to store state might be re-written with a session variable thus:

page1.php

```
<?php
  session_start();
?>
 …structural HTML here…
<?php
   $myNum = rand(1,10);
   $_SESSION["banana"] = $mynum;
   echo "<p>My number was $mynum</p>"
?>
<a href="page2.php">Next page</a>
```

page2.php

```
<?php
  session_start();
?>
 …structural HTML here…
<?php
   $mynum = $_SESSION["banana"];
   $echo "<p>The number was $mynum</p>";
?>
```

PHP session variables

- So the previous page where we used a hidden input field to store state might be re-written with a session variable thus:

page1.php

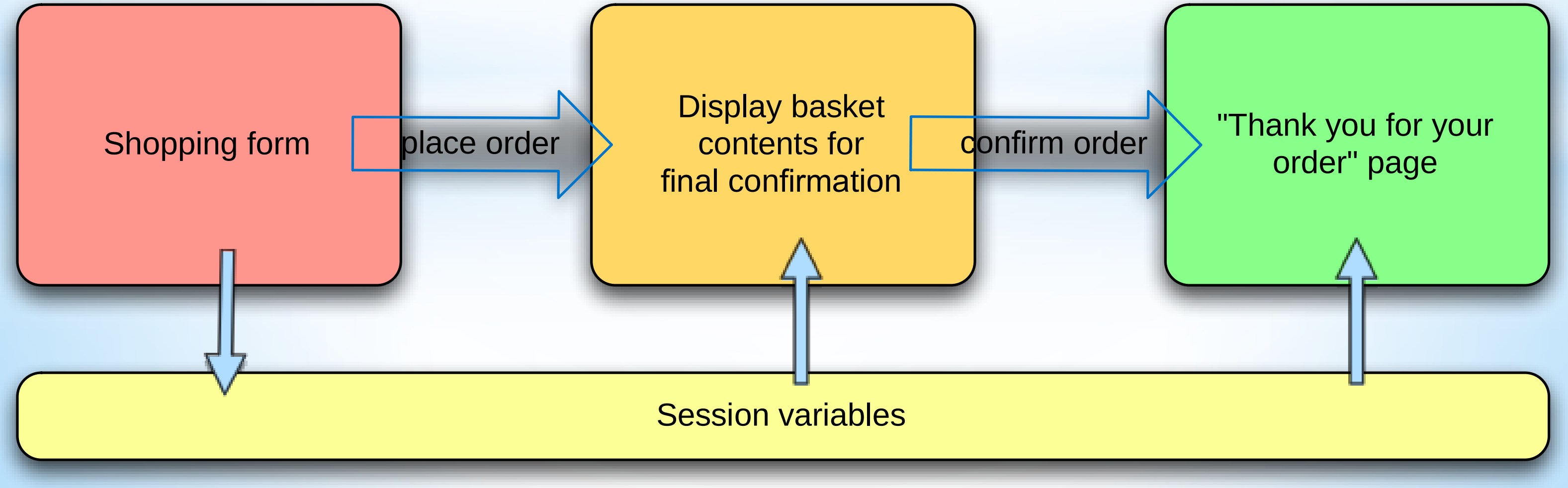
```
<?php
  session_start();
?>
 …structural HTML here…
<?php
   $myNum = rand(1,10);
   $_SESSION["banana"] = $mynum;
   echo "<p>My number was $mynum</p>"
?>
<a href="page2.php">Next page</a>
```

page2.php

```
<?php
  session_start();
?>
 …structural HTML here…
<?php
   $mynum = $_SESSION["banana"];
   $echo "<p>The number was $mynum</p>";
?>
```

Sessions: a common pattern

- On an e-commerce site, it is common to display the contents of someone's basket prior to going ahead and placing an order
  - The user will have placed items into their basket using a form
  - An intermediate screen displays all their items and asks them to confirm
  - A final screen displays a thank you and confirmation message
- All this could be done using session variables:


Shopping form
place order
Display basket contents for final confirmation
confirm order
"Thank you for your order" page
Session variables

Summary

- Web forms are filled in by the user within their web browser
- Each field in a form usually has a name and a value
- When the user submits the form, the form data is sent to the server to be processed by a server-side process
- Alternatively, or additionally (or both!) the form data can be processed by client-side Javascript
- The top level "block" of a form is the form element
- Inside the form element can be a variety of different input elements
- A submit button, when pressed, will send the form data to the server

Summary

- Any and all user input in PHP we need to get through an HTML form
- This means that we need to process user input in batches – we can't mix processing and input
- This means your PHP "program" – or more accurately, your website – may be split across two or more files
- We can read form fields using the $_REQUEST variable
- Nothing stops us generating a form dynamically – in fact for things like pull down lists and radio buttons we might have to do that

Summary

- HTTP is stateless – every individual request is its own thing and has no "sight" of any other
- There are ways around this – in PHP we have the session API
- Use the $_SESSION variable to put items into the session
- These variables persist across pages for the duration of the user's browser session on your site