

Notes from Well House Consultants

These notes are written by Well House Consultants and distributed under their Open Training Notes License. If a copy of this license is not supplied at the end of these notes, please visit

*<http://www.wellho.net/net/whcotnl.html>
for details.*

1.1 Well House Consultants

Well House Consultants provides niche training, primarily but not exclusively in Open Source programming languages. We offer public courses at our training centre and private courses at your offices. We also make some of our training notes available under our "Open Training Notes" license, such as we're doing in this document here.

1.2 Open Training Notes License

With an "Open Training Notes License", for which we make no charge, you're allowed to print, use and distribute these notes provided that you retain the complete and unaltered license agreement with them, including our copyright statement. This means that you can learn from the notes, and have others learn from them too.

You are NOT allowed to charge (directly or indirectly) for the copying or distribution of these notes, nor are you allowed to charge for presentations making any use of them.

1.3 Courses presented by the author

If you would like us to attend a course (Java, Perl, Python, PHP, Tcl/Tk, MySQL or Linux) presented by the author of these notes, please see our public course schedule at

<http://www.wellho.net/course/index.html>

If you have a group of 4 or more trainees who require the same course at the same time, it will cost you less to have us run a private course for you. Please visit our onsite training page at

<http://www.wellho.net/course/otc.html>

which will give you details and costing information

1.4 Contact Details

Well House Consultants may be found online at

<http://www.wellho.net>

graham@wellho.net

technical contact

lisa@wellho.net

administration contact

Our full postal address is

404 The Spa

Melksham

Wiltshire

UK SN12 6QL

Phone +44 (0) 1225 708225

Fax +44 (0) 1225 707126

Arrays

Arrays are PHP variables into which you can save more than one value. You can then use a loop to process each individual value, you can refer to individual values by using a special syntax, or you can perform operations on the array as a whole.

<i>Array manipulation</i>	5
<i>Array functionality</i>	8

If you're an experienced programmer in a language such as C or Pascal, tread carefully here. Arrays are different in PHP. If you're a Perl programmer, PHP arrays are similar to hashes, and if you've written in Java, then you might compare them to hash tables. If you're a database programmer, then they're like a simple table: a number of rows, each with a unique key and an associated value.

Tcl programmers will find that arrays in PHP can work just like arrays in Tcl, for example, with named elements. But they will find many other more powerful features as well.

And if you're none of the above?

An array is a type of variable in which you can hold not one value, but many. You can create an array variable to hold a whole lot of variables, and then look each element up by its key.

Let's see an example where we have an array (or table) of data. We want to look something up in that table from our browser:

```
<head>
<title>Array lookup</title>
</head>
<body bgcolor=white>
Who lives where?<BR>
<?php
$house = $_GET[house];
$what = 16;
$home[1] = "John Smith";
$home[2] = "Daphne Jones";
$home[3] = "Ronald McDonald";
$home[10] = "The Prime Minister";
$home[11] = "The Chancellor";
$home[404] = "Lisa and Graham";
$home[Llareggub] = "Dylan Thomas";
$home[$what] = "Fred Jones";

print ("In $house lives $home[$house]");
?>
</body>
```

If we run that (using the variable "\$house" defined via the form), we'll get:

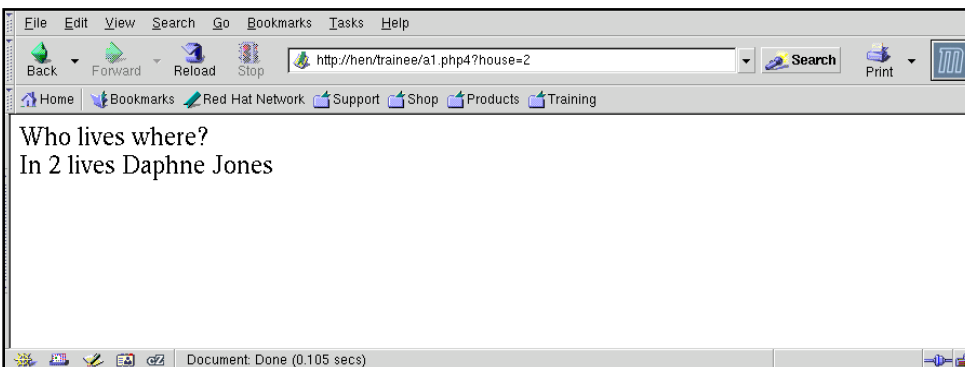
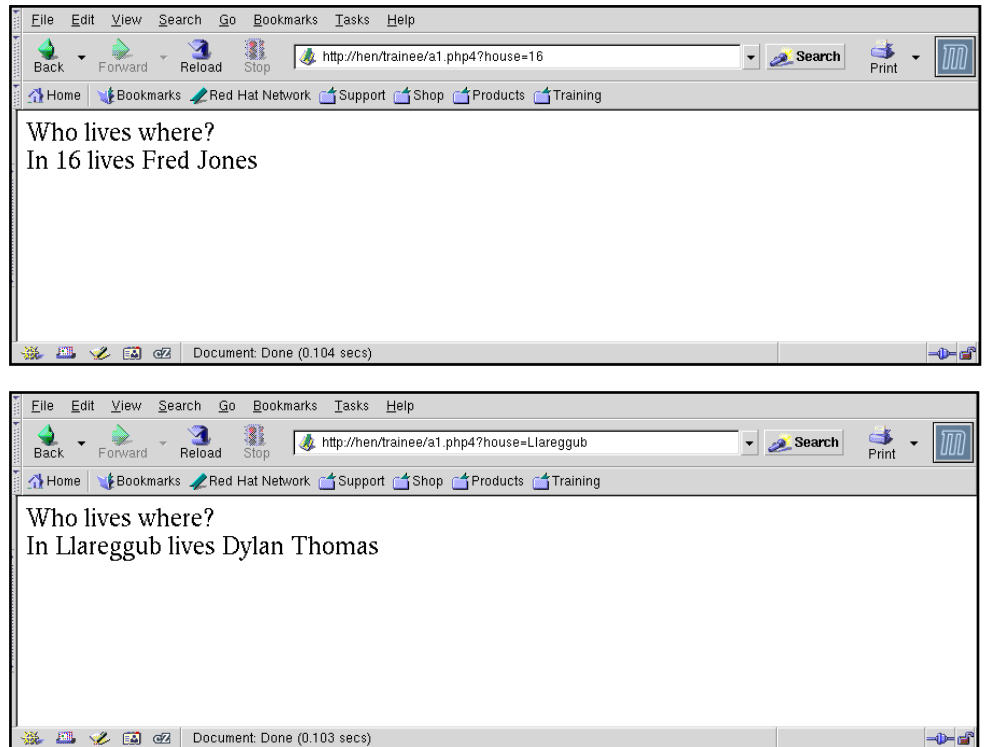


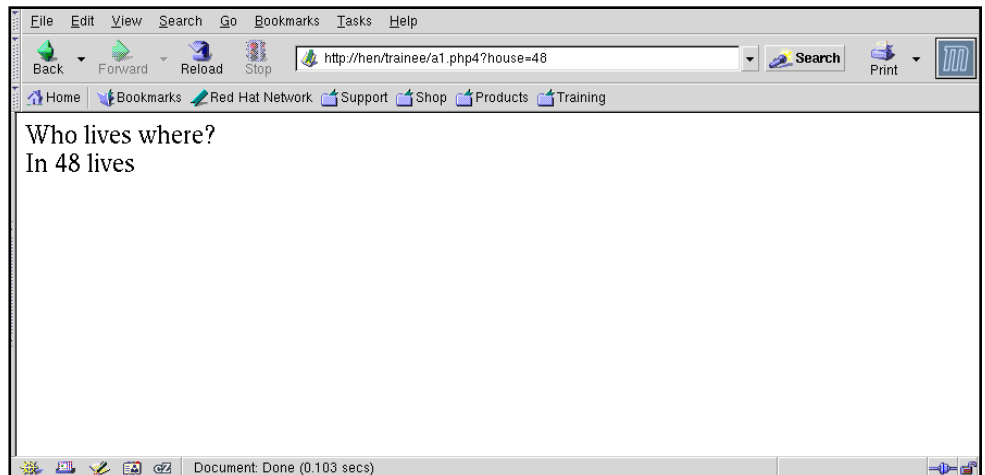
Figure 1 A successful lookup in a table

Figure 2 More successful lookups in a table



All successful lookups in the table. You'll notice that there's no distinction between numbers and names for the houses, and that gaps are allowed in the numbers.

Figure 3 Asking for a house not listed in the table



2.1 Array manipulation

Our example doesn't tell us if we ask about a house that hasn't been entered.

The solution is to test the array element¹ before you offer the answer.

Neither does our example let us get out a listing of all the known houses, or all the known inhabitants.

Let's add the following:

- An error message if a number or name is specified, but unknown
- A complete listing if we've given no input

¹ to see if it exists, or if it contains a value

```

<head>
<title>Array lookup, with check etc</title>
</head>
<body bgcolor=white>
Who lives where?<BR>
<?php
$house = $_GET[house];
$home[1] = "John Smith";
$home[2] = "Daphne Jones";
$home[3] = "Ronald McDonald";
$home[10] = "The Prime Minister";
$home[11] = "The Chancellor";
$home[12] = "";
$home[404] = "Lisa and Graham";
$home["Llareggub"] = "Dylan Thomas";

if ($house) {
    if ($home[$house]) {
        print ("In $house lives $home[$house]");
    } else {
        print ("In $house lives we know not who");
    }
} else {
    print ("So you want a street listing then?<BR>");
    reset($home);
    while ($residence = each($home)) {
        $addy = $residence["key"];
        $person = $residence["value"];
        print ("$person lives in $addy<BR>");
    }
}
?>
</body>

```

You'll notice that we check to see whether we've got anything (non-empty) in "\$house" to see if our user completed the form, and then whether the particular element is empty to see if there's anything to report. If a variable doesn't exist, it will report back a false value, but we need to be careful if a house is empty ...

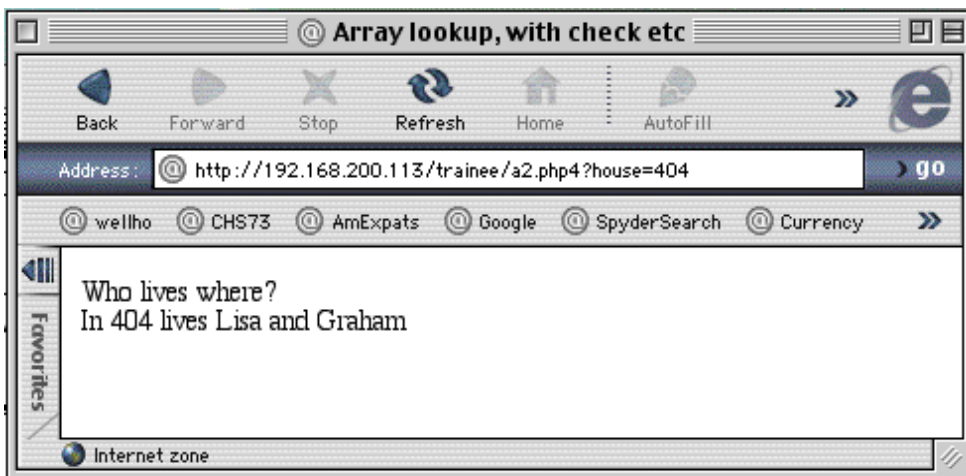
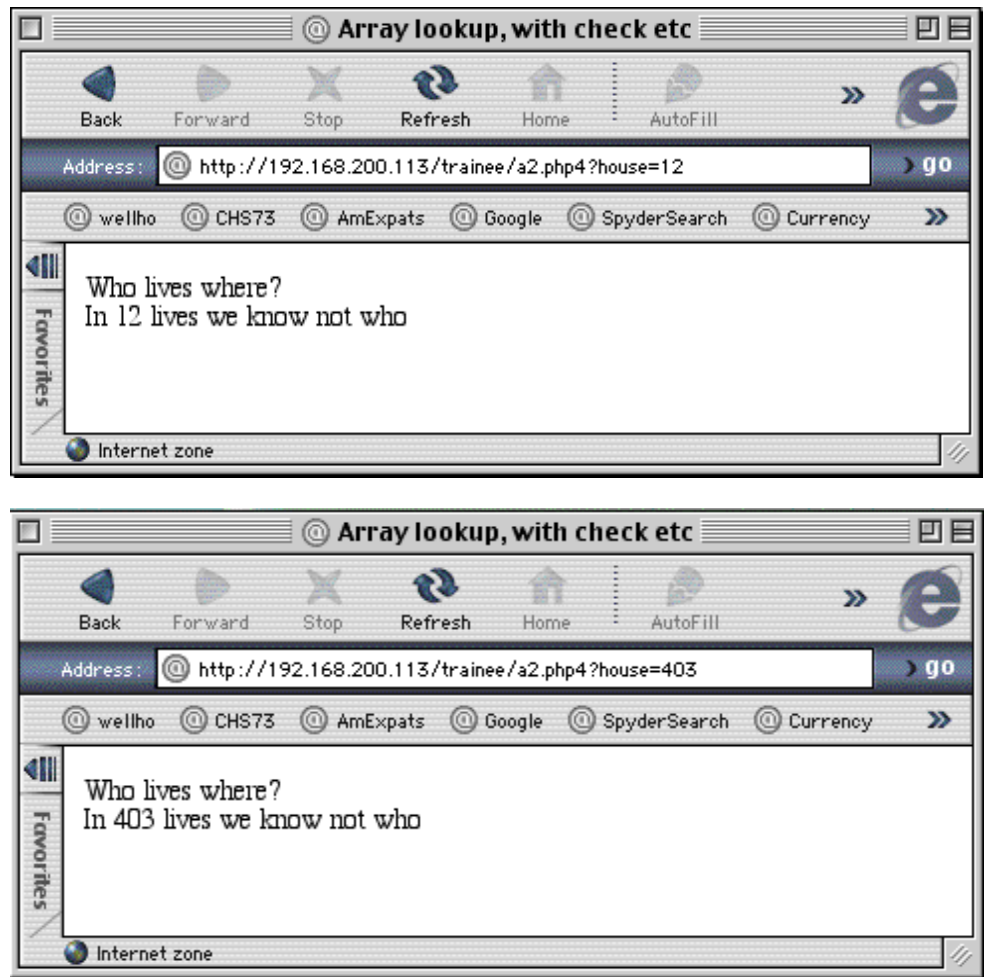


Figure 4 Results when an error message is added

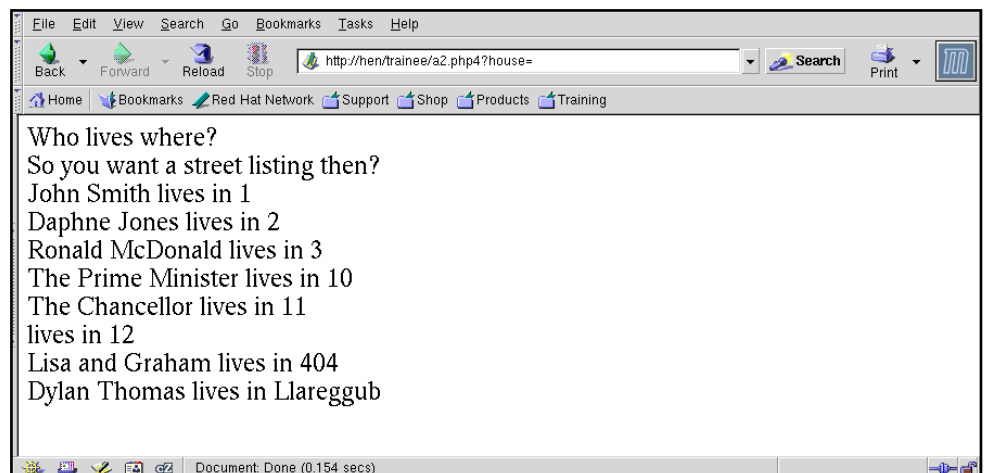
Figure 5 More results with an error message added



We have a different problem when we want a listing of everyone on the street because we don't necessarily know what all the houses are numbered or called, and it's not practical for us to write any form of loop to guess.

There are a number of ways of doing this; we've chosen to use the `each` function that's a standard part of PHP to look through each of the key and value pairs in turn, and return us a new array called "\$residence" for each address. We can handle this array easily, since we know that it has members¹ called `key` and `value`.

Figure 6 Returning the value pairs



¹ elements, indexes, keys - call them whatever you like!

It's worthwhile commenting on the **reset** statement here.

There's nothing in our code that tells the `each` function explicitly which particular pair we're interested in each time we call it, so it simply returns the next pair. `reset` is used to ensure that it starts at the beginning rather than in the middle, as it might in more complex examples where `each` might have already been used on the same array. It's not, strictly, needed in this example, but we want to teach you good practice from the start.

2.2 Array functionality

Arrays are variables. You can refer to the whole of an array by using its name,¹ or you can refer to individual elements by referring to the elements by key in square braces, as also shown in the previous example. You don't need to declare an array before you use it, just save something into a new array and it's automatically created. Save something into a new element of an already existing array and that extra element will be created.

PHP provides functions to make your use of arrays easy and flexible, so much so that they are used in place of a number of different structures in other languages.

Creating Arrays

The `array` command lets me create an array with a number of elements all in one statement:

```
$days = array ( "Monday", "Tuesday", "Wednesday", "Thursday" );
```

creates an array with four elements. Be careful, though, as the indexes start at 0 if you use this form, so a reference later to `$days[3]` would mean "Thursday".

An alternative form of the `array` command lets you specify each key as you go:

```
$days = array( 1 => "Monday", 2 => "Tuesday", "Match" => "Saturday" );
```

will create an array with three elements, keyed 1, 2 and Match.

There's also a function called `range` that lets you set up an array of numbers. I could write:

```
$days = range (1,31);
```

to create an array of 31 elements, numbered 0 to 30, with the values 1 to 31.

Manipulating arrays

Array elements can be deleted using the **unset** command; this is not the same as replacing the contents of an array element with an empty string.

You can check whether an array element exists by using the **in_array** function, and get a count of the number of elements by using the **count** function. **is_array** returns you true if the argument you give it is an array, false otherwise.

The **list** command² can be used to extract a series of elements from the start of an array into a series of individual variables.

As an alternative to `each`,³ iterations through an array can also be performed using the functions **key**, **value**, **current**, **next**, **end** and **prev**. Yet another alternative is **array_walk**, which runs a function that you define to it on every element in an array. The flexibility is breathtaking, and there's much more for us to look at later.

Perhaps at this point, you'll be starting to get used to the saying "there's a function to do that" which you'll hear said so many times as you learn PHP. Since this is a modern language, written after the dramatic memory and speed improvements made

¹ "Shome" in our previous example

² an unusual one this, since it's written on the left of an assignment

³ we saw that earlier in this module

in computers in recent years, the authors of PHP have been able to add a huge raft of functionality without any appreciable hit on performance.

We suggest that newcomers to the language learn a subset of the functions that they're comfortable with, and then look up extra functions when they find themselves thinking "there should be a function to do that" ... as there probably is. Don't try to learn them all at once; you wouldn't try to learn every word in a foreign language as your first step to learning it, now would you?

This extra flexibility can bring a headache for maintenance engineers, especially those of them who only maintain PHP for a small proportion of their time. If you're going to be writing code, consider who will be looking after it later. It's said that only 20 percent of commercial code is maintained by its author throughout its life, and that much the heaviest expense of code is not the original writing but the upgrading and maintenance during its life. Consider:

- Comment your code well
- Provide documentation too
- Inset your blocks so that they can be followed
- Structure your code into functions
- Use a sensible subset of the functions that PHP provides (not as many as you can!)

```
<head>
<title>Array lookup, with check etc</title>
</head>
<body bgcolor=white>
Who lives where?<BR>
<?php
function prwho($person,$addy) {
    $person or $person = "no-one";
    print ("$person lives in $addy<BR>");
}

$home = array( 1 => "John Smith",
              2 => "Daphne Jones",
              3 => "Ronald McDonald",
              4 => "",
              "221B" => "Sherlock Holmes",
              84 => "Pizza Hut" );

if ($house = $_GET[house]) {
    if ($home[$house]) {
        print ("In $house lives $home[$house]");
    } else {
        if (is_string($home[$house])) {
            print ("$house is empty");
        } else {
            print ("Don't know about
$house");
        }
    }
} else {
    print ("So you want a street listing then?<BR>");
    array_walk($home,"prwho");
}
?>
</body>
```

If you're already an experienced programmer¹, you might like to note that you can have arrays of arrays. Use two (or more) sets of square brackets after a variable name, and it all works for you. Newcomers to programming are advised to wait a while before they jump too deeply into this one!

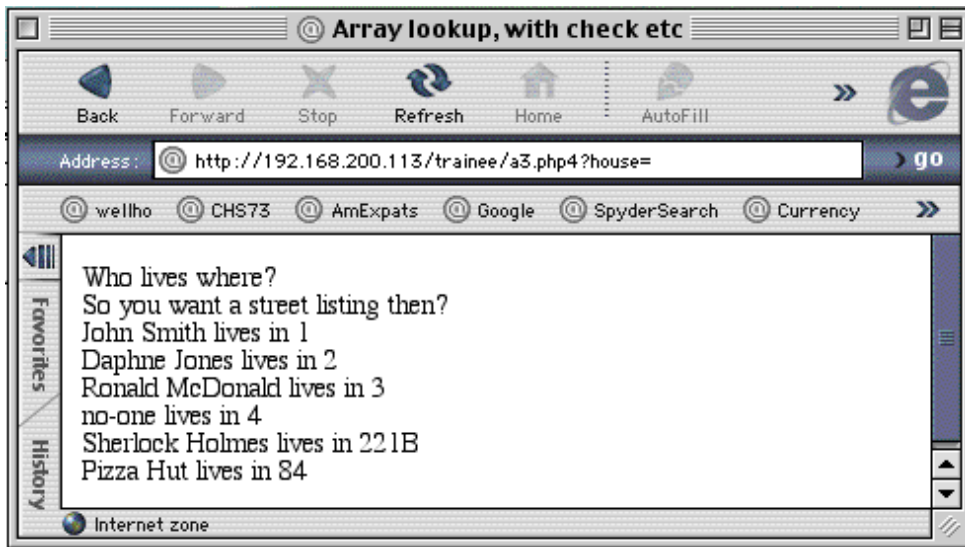


Figure 7 An array lookup with check

Coming back to our recent comments about the flexibility of PHP, did you notice the following in that last example?

```
if ($house = $_GET[house]) {
```

which assigns the value from the form into the house field and also checks it at the same time, and:

```
$person or $person = "no-one";
```

which checks the value of `$person`, and if it doesn't contain anything, loads it with the word "no-one"?

Whilst both of these are excellent examples of common idioms used by PHP programmers, you might want to consider avoiding their use if you're to be writing code that has to be maintained by folks whose major job isn't in handling PHP code.

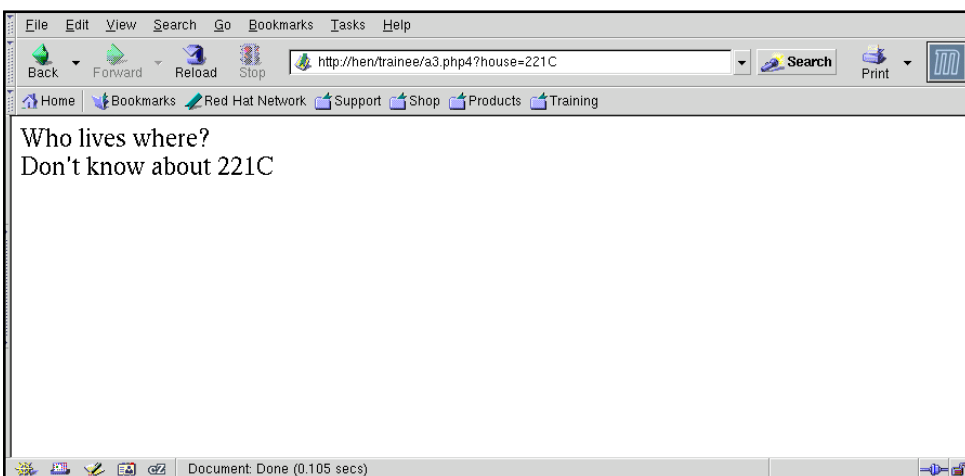


Figure 8 Filling in the house number

¹ or if you want to blow your mind now

License

*These notes are distributed under the **Well House Consultants Open Training Notes License**. Basically, if you distribute it and use it for free, we'll let you have it for free. If you charge for its distribution of use, we'll charge.*

3.1 Open Training Notes License

Training notes distributed under the **Well House Consultants Open Training Notes License** (WHCOTNL) may be reproduced for any purpose PROVIDE THAT:

- This License statement is retained, unaltered (save for additions to the change log) and complete.
- No charge is made for the distribution, nor for the use or application thereof. This means that you can use them to run training sessions or as support material for those sessions, but you cannot then make a charge for those training sessions.
- Alterations to the content of the document are clearly marked as being such, and a log of amendments is added below this notice.
- These notes are provided "as is" with no warranty of fitness for purpose. Whilst every attempt has been made to ensure their accuracy, no liability can be accepted for any errors of the consequences thereof.

Copyright is retained by Well House Consultants Ltd, of 404, The Spa, Melksham, Wiltshire, UK, SN12 6QL - phone number +44 (1) 1225 708225. Email contact - Graham Ellis (graham@wellho.net).

Please send any amendments and corrections to these notes to the Copyright holder - under the spirit of the Open Distribution license, we will incorporate suitable changes into future releases for the use of the community.

If you are charged for this material, or for presentation of a course (Other than by Well House Consultants) using this material, please let us know. It is a violation of the license under which this notes are distributed for such a charge to be made, except by the Copyright Holder.

If you would like Well House Consultants to use this material to present a training course for your organisation, or if you wish to attend a public course is one is available, please contact us or see our web site - <http://www.wellho.net> - for further details.

Change log
Original Version, Well House Consultants, 2004

Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____

License Ends.