Coding Standards & Documentation

Magnus White
mpw@ecs.soton.ac.uk

Coding Standards

What do we mean by coding standards?

- Indentation
- Comments
- Variables
- Functions
- White space
- File organisation
- Naming conventions
- Programming practices
- Programming rules of thumb
- Architectural best practices

Coding Standards

There are no universal coding standards.

Specific languages may some guidelines.

Syntax is not a coding standard. It is a language rule.

What Standard to Follow?

It doesn't matter which!

Be consistent and logical!

Naming Conventions

There are a number of different naming conventions used.

Often a different one is used to describe different types of statements.

eg. variable name and global variables names

Flat Names

Flat names are simply the words joined together with no uppercase letters

$vartwenty

$filmname

Used for a variety of reasons, usually down to personal preference.

Camel Case

Camel case capatalises the first letter of each word. Usually keep the first word in lowercase.

$varTwentyOne

$filmName

Extremely popular and easy to read.

Underscore Case

Underscore case repalces spaces with an underscore. The words can be upper or lowercase. Usually not combined with camel case.

$var_twenty_one

$film_name

Very popular and easy to read.

Underscore First

Underscore first adds a underscore as the first character.

$_globalVar

$_global_var

Usually used for global, class scope or inherited variables and functions.

Uppercase

The word is all in uppercase. The words may be connected with underscores or other characters

CONSTANT

Commonly used for constant names but little else.

Hyphens

Many languages do not support hyphens as name separators.

eg. This is NOT valid in PHP

```
$var-name
```

This is because the hyphen (-) is a logical operator.

Using a syntactical highlighter will make this very obvious as you write your code.

Personal Prefernce

It is up to you to decide what convention you wish to use.

You may decide this based on ease of readability or completely irrational reasons.

For example, I almost only use camel case.

Indentation

Without indentation this code is very hard to read.

```
function bookContent($book){
$topicsArray = [];

foreach($book->getChapters() as $chapers){
foreach($chapers->getTopic() $topic){
if(!in_array($topic, $topicsArray)){
$topicsArray[] = $topic;
}
}
}
return $topicsArray;
}
```

Indentation

With indentation the code is much easier to read and understand each level.

```
function bookContent($book){
        $topicsArray = [];

        foreach($book->getChapters() as $chapers){
              foreach($chapers->getTopic() $topic){
                    if(!in_array($topic, $topicsArray)){
                          $topicsArray[] = $topic;
                    }
              }
        }
        return $topicsArray;
}
```

Indentation Types

Indentation as a number of possible types. These depend on personal preference or might be enforced where you work.

- Tabs (most common and easy to move code)
- Spaces
  - 2 spaces (common version of using spaces)
  - any number of spaces

White Space

White space is just the blank characters. This includes indention (such as tabs) as well as empty lines.

It is important to break up your code. This instantly and visually shows the separation of code sections.

White Space

White space is just the blank characters. This includes indention (such as tabs) as well as empty lines.

It is important to break up your code. This instantly and visually shows the separation of code sections.

White space costs nothing!

Brackets

Bracket placement is another stylistic choice. These may depend on personal preference or might be enforced where you work.

```
function magic() {
    return $magicVarFromNoWhere;
}

function magic()
{
    return $magicVarFromNoWhere;
}
```

```
if($apple == "red"){
    eatApple();
}
else{
    giveAwayApple();
}

if($apple == "red"){
    eatApple();
} else{
    giveAwayApple();
}
```

```
function magic(){ return $magicVarFromNoWhere; }
function magic() return $magicVarFromNoWhere;
```

Your Style

It doesn't matter what style you use as long as, anyone who reads your code, can understand it.

You may be required to write using a certain style when working.

Remember, be consistent and logical!

Documentation

It is just as important as well written and efficient code!

Documentation

Documentation allows others to understand what your code does.

Your notes may have decision rationale or other information which cannot me extrapolated from your code.

Example

What is going on here?

```php
function processItems($w = 0, $ipb = null, $u){
   if($w == 0) return "no items where processed.";
   else{
         $bt = $w * $ipb;
         return "$w$u of items using $bt boxes.";
   }
}

echo processItems(14, 0.5,'T');
```

Example Answer

```php
// $w is the weight of the items
// $ipb is the items per box i.e. how many items fit in the
   standard size shipping box
// $u is the unit of weight
function processItems($w = 0, $ipb = null, $u){
   // if the weight is 0 then return the error message
   if($w == 0) return "no items where processed.";
   else{
            // otherwise work out how many boxes are needed
            // $bt is the boxes total required
            $bt = $w * $ipb;

            // return the string with the weight and units and
            reporting the number of boxes required.
            // e.g. 14T of items using 7 boxes.
            return "$w$u of items using $bt boxes.";
   }
}

echo processItems(14, 0.5,'T');
```

Comments

In PHP they are started using `//comment`

This comments out the line from that point.

You can also block comment code using

```
/* block of code */
```

Comments

Use comments anywhere you think extra information is needed.

They can be used to describe a block of code or act as titles informing the reader what code is coming next.

Formal PHP doc

PHPdoc is a formal standard for documentating PHP code.

It is very easy to understand and is machine readable.

http://manual.phpdoc.org/HTMLSmartyConverter/HandS/phpDocumentor/tutorial_tags.pkg.html

Formal PHP doc

It consists of a number of tags which describe attributes of the code. Possible a function, class or variable.

Most common tags:

- @param
- @return
- @todo
- @author

Formal PHP doc

Break down it's components.

```
/**
*  Function to find the total M class planets in a galaxy.
*  It iterates through planet in each solar system to check for M
   class planets.
*
*  @param Galaxy Object $galaxy
*  @return int
*/
```

Example Function with PHPdoc

```php
/**
*  Function to find the total M class planets in a galaxy.
*  It iterates through planet in each solar system to check for
   M class planets.
*
*  @param Galaxy Object $galaxy
*  @return int
*/
function mClassPlanetsInGalaxy($galaxy){
    $totalMClassPlanets = 0;

    foreach($galaxy->getSolarSystem() as $solarSystem){
          foreach($solarSystem->getPlanets() as $planet){
              if($planet->getClass() == 'M'){
                    $totalMClassPlanets++;
              }
          }
    }
    return $totalMClassPlanets;
}
```

PHPdoc

Please read up on the PHPdoc as it is a requirement for future labs and the coursework.

Coding Standards & Documentation

Magnus White
mpw@ecs.soton.ac.uk