

Documentation & Coding Standards

HA!



HA!

DOCUMENTATION

Documentation == Evil?

Do you think that ...

- Documentation is tedious?
- An afterthought?
- Redundant?
- Messes up your self-documenting code?
- Is done only when you have to?
- ...
- All of the above?

Types of (Project) Documentation

- Written documentation
 - READMEs, tutorials, how-to/reference guides, white papers, books
 - Project website
 - Design documentation, diagrams
- Code documentation
 - API docs, comments, example code, unit tests
- Community documentation
 - Blog posts, Q&A sites, forums, talks, videos, meet-up's/support/user groups, conferences

Is It Important?



JOINS NEW PROJECT

**REALIZES THERE IS NO DOCUMENTATION OR
COMMENTS IN CODE**

memegenerator.net

Written (Project) Documentation

README files

- Very (most?) important file in codebase
- Represents first contact with new user
- Goal to “introduce” project, get them to stay
- Give pointers on what to do and how to get started
- Typical Sections:
 - Description, quick examples, quick start guide, links to more documentation, project organisation, legal notices

Tutorials / Guides

- Show the new user what can be done and patterns/best-practice way to do it.
- Highlight unique or powerful features
- Use a conversation/dialogue format (personal)
- Does not need to cover every topic,
- Does not need to go in-depth (use links)
- Interactive tutorials or videos have strong value

Reference Documentation

- Provide the user with a way to find out about specific topics (not just wander through the entire API)
- Cover all the major topics in depth
- Organise information so that it is easy to navigate and search
- Good to have key-words/index/see-also.

Project Website

- Documentation as marketing!
- Create a look-and-feel (brand) for the project
- Good for search, links and social media
- Integrated project hosting webpages (github wiki's) are an easy way.

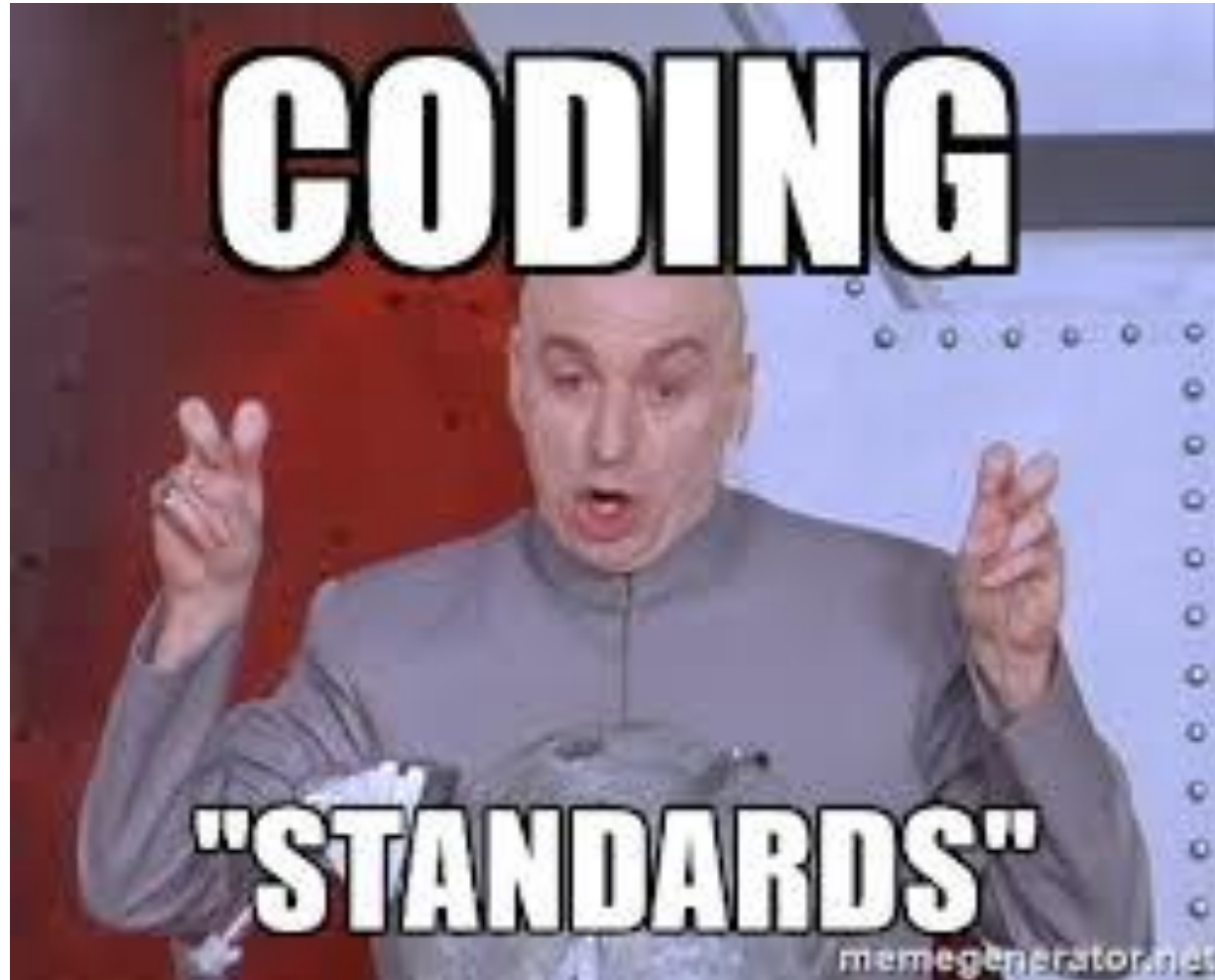
Big Ones?

- API generated documentation
- Project level documentation
- Code examples / how-to documentation
- User documentation

UP TO DATE DOCUMENTATION

**NOW THERE IS SOMETHING I
HAVEN'T SEEN IN A LONG TIME.**

Coding Conventions & Standards



Conventions & Standards

- Good coding conventions:
 - Create consistent looking code so a reader can focus on function, not layout
 - Familiarity allows a reader to make correct assumptions based on layout
 - Helps in changing and maintaining code
 - Reduces errors in logic

What are they?

Rules/Standards
(that must be followed)

vs

Conventions/Guidelines
(that should be followed)

Why?

“Productivity by convention”

- Better for maintenance
- Clear code == Less bugs
- Good for teamwork
- Flexible teams
- Lower costs

... and because you care about quality!

Where do they come from?

- Set by the team, or a company
 - Often written by senior or lead developers
 - Based on experience
 - Based on personal preferences
- Set by language and tool developers (vendors)
 - Works well with the languages
 - Works well with the tools
 - Because they like it that way ...

Naming Conventions

Based on Microsoft C# design
guidelines

Naming Conventions

- Capitalization:
 - Used to identify each word in a name
 - PascalCasing (most) and camelCasing (params)
- Don't use `_` anywhere (or hyphens etc)
- Choose readable (long) over brevity
- Avoid conflict with identifiers / common
- Avoid language (type) specific names

Naming Conventions

- Use nouns or noun phrases for classes and structs (in PascalCasing)
- Use adjective phrases for interfaces, with “I” as a prefix
- Methods should be verbs or verb phrases, and always PascalCasing
- Use plural phrase for collections
- Events should be a verb or verb phrase

Naming Conventions

Pascal Casing

Class, Struct	AppDomain
Interface	IBusinessService
Enumeration type	ErrorLevel
Enumeratiion values	FatalError
Event	Click
Protected field	MainPanel
Const field	MaximumItems
Read-only static field	RedValue
Method	ToString
Namespace	System.Drawing
Property	BackColor
Type Parameter	TEntity

Camel Casing

Private field	listItem
Variable	listOfValues
Const variable	maximumItems
Parameter	typeName

Layout Conventions

- Empty lines
 - Between members
 - After the closing parenthesis
 - Between multi-line statements
 - Between unrelated code blocks
 - Around the #region keyword
 - Between using statements of different root names

Member Order

1. Private fields and constants
2. Public constants
3. Public read-only static fields
4. Factory Methods
5. Constructors and the Finalizer
6. Events
7. Public Properties
8. Other methods/private properties in calling order

Layout Conventions

- Maximum line length is 130 characters.
- Indent 4 spaces, don't use Tabs
- Keep one white-space between keywords like `if` and the expression, but don't add white-spaces after `(` and before `)`.
- Add a white-space around operators, like `+`, `-`, `==`, etc.
- Always add parentheses after keywords `if`, `else`, `do`, `while`, `for` and `foreach`
- Always put opening and closing parentheses on a new line.
- Don't indent object initializers and initialize each property on a new line.
- Don't indent lambda statements
- Put the entire LINQ statement on one line, or start each keyword at the same indentation.
- Add braces around comparison conditions, but don't add braces around a singular condition.

Code Comments

Comments

“Code tells you how. Comments tell you why.”

- Can be a very important information source, but use them wisely!
- There is no point writing trivial comments
- Use them in moderation
- Always focus on the “why”

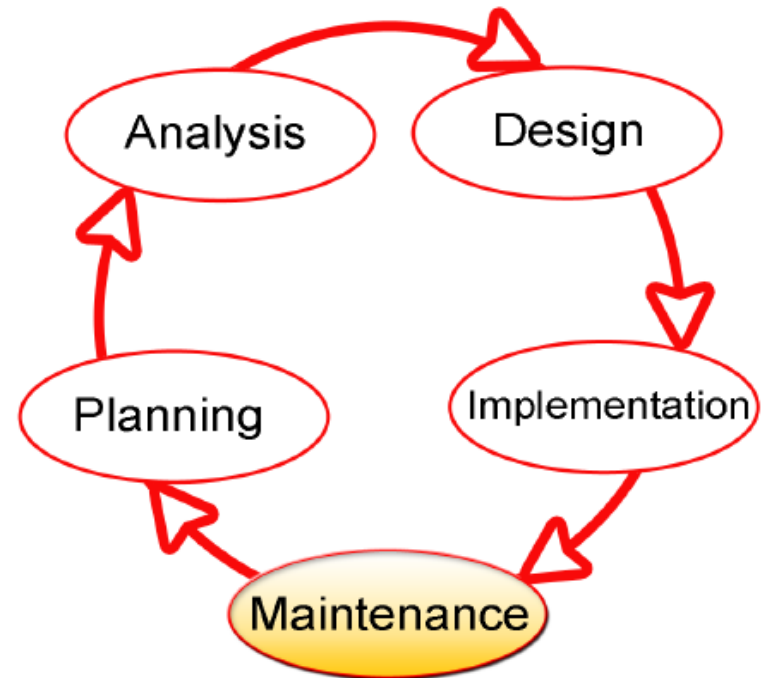
Documentation is a BIG topic!

It's worth doing it well.

Coding Standards

Why

- Better for maintenance
- Less bugs
- Good Teamwork
- Flexible Teams
- Lower Cost
- ... because you care about quality!



Example: Indentation

- Be consistent to be clear (convention)
 - Easier to read and understand
 - Easier to maintain
- Used to clearly show scope or blocks
 - Supports the programming ideas
- Rules for functions, objects, blocks
- Does not affect what the code means in most languages

Example: Comments

... Yes! That topic again!

- Applies to all programming languages
- Typically ignored by the compiler/interpreter
- Should clearly explain the “why” not the “what” (which is the code)
- Should NOT have a comment on every line
- If it gets messy, it’s probably bad code design

Example: Whitespace

- Very important for clarity
- Can be easily overlooked
- Often the lack of whitespace can hide the logic or flow control problem (i.e. you don't have the blocks you think you have.)

Example: Naming Convention

- The “names” you choose are part of your documentation.
- Take care, get it right, change if you need to!
- There are some very common rules about what characters can be used in names, but that doesn't mean your program should use every possible variation!

Why Have Coding Standards

- Tasks are small divisions of work that need to be done.
- Big tasks should be broken down into small tasks
- The outcome of the task should be clearly defined
- Productive, flexible and visible

Documentation Generators



doxygen



perldoc 2.0
International Perl Documentation



Documentation Generator

- A programming tool to generate software documentation, for either programmers (API) or end users (end-user guides), by extracting details and special comments from source code files.
- The output format is typically HTML

Application Programming Interface (API)

- A set of clearly defined methods for communication between software components
- API documentation is intended for other developers so that they can use the services described
- Good quality documentation is essential

API Documentation

- API: Application Programming Interface are written to be used (by programmers)
- Typically each module/file, class, function (method) and variable are documented
- Provides a fine-grained format for describing details of all input and output
- Provides the opportunity for the author to explain why the code exists

Documentation Comments

Recommended C#
xml-doc Tags

C# xml-doc Tags

<c> <code>

<example> <exception>

<include> <list>

<para> <param> <paramref>

<permission> <remarks> <returns>

<see> <seealso> <summary>

<typeparam> <typeparamref>

<value>

C# Example

```
/// <summary>
/// This class performs an important function.
/// </summary>
public class MyClass{}

/// <summary>
/// Summary
/// </summary>
/// <param name="param1">Some Parameter.</param>
/// <returns>What this method returns.</returns>
```

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");

        Console.ReadKey();
    }

    ///The left side operand for an arithmetic operation
    static int Operand1 { get; set; }

    ///The right side operand for an arithmetic operation
    static int Operand2 { get; set; }

    /// This method adds two integers
    static int Add(int operand1, int operand2)
    {
        return operand1 + operand2;
    }

    ///This method subtracts two integers
    static int Subtract(int operand1, int operand2)
    {
        return operand1 - operand2;
    }
}
```

```
///
```

Javadoc Example

```
/**
 * <h1>Hello, World!</h1>
 * The HelloWorld program implements an application that
 * simply displays "Hello World!" to the standard output.
 * <p>
 * Giving proper comments in your program makes it more
 * user friendly and it is assumed as a high quality code.
 *
 *
 * @author   Zara Ali
 * @version  1.0
 * @since    2014-03-31
 */
public class HelloWorld {

    public static void main(String[] args) {
        // Prints Hello, World! on standard output.
        System.out.println("Hello World!");
    }
}
```

Community Documentation

People and Tools Create ...

- Project software (bugs, issues, tasks ...)
- Blog posts,
- Q&A sites (stack overflow ...)
- Forums,
- (Tech) Talks,
- Meet-up/support groups,
- Conferences

People and Tools Create ...

- Over time community documentation can be the most valuable support for new users
 - Have you ever searched using google with an error message string and found someone who had the same problem, and the solution?!
 - Add a diversity to the vocabulary that the original authors would not think of