Document Public API

What is an API

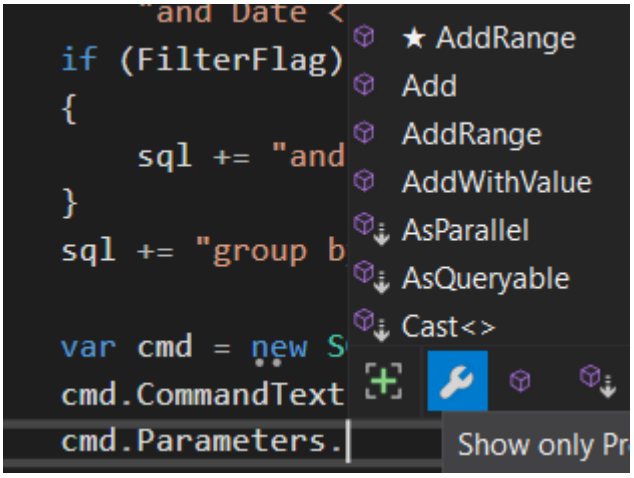
An application programming interface is a connection between computers or between computer programs. **It is a type of software interface, offering a service to other pieces of software.** A document or standard that describes how to build or use such a connection or interface is called an API specification. Wikipedia

The *Home Calendar* program will eventually be turned into an API, so that it can be used as a library when we create a GUI calendar program.

Why Do We Need to Document Them?

> Read the code, dude.

- Reading code can be tedious.
- You might not care how the code does what it does. It could be a layer that is an abstraction for your program.
- More often than not, you don't have the code.
- Properly written documentation, will show up as hints in the Visual Studio IDE.
    - For example, the list of methods that are available to an object



- Properly written documentation forms the basis for the help documentation.
    - Example, the help files for C# String class

IF THERE IS A DISCREPENCY BETWEEN THE COMMENT/DOCUMENTATION AND THE CODE, THEN YOU IMMEDIATELY KNOW THAT SOMETHING IS WRONG!

Goals

- Identify public API methods, fields
- Identify the components of a public function/method to document
- Document functions/methods using appropriate XML tags
- Generate web pages with API documentation
- Declare the contract of every public method

    - Why? - What information about a public function do I need to know to use it?

Using XML to Document the Components

How to document with XML comments

Student Activity

Read the above documentation, and answer the following questions

- Are the XML documentation comments compiled?
- What does an XML documentation line start with?
- How do you specify additional information about a parameter?
- What does DocFx generate?

Generating Web Pages

DocFX generates a static website which can then be deployed anywhere.

It can run on Linux, macOS, and Windows.

Student Activity

- Install package in Visual Studio using Nuget: `docfx.console`
- Build the solution to generate the web pages.

More questions

- Where do you find the generated web pages?
- What happens if you add XML documentation comments to a public function in an internal class?

Style Guide

The following is excerpts from the google style guide.

Documentation basics

The API reference **must** provide a description for each of the following:

- Every class, interface, struct, and any other similar member of the API (such as union types in C++).
- Every constant, field, enum, typedef, etc.
- Every method, with a description for each parameter, the return value, and any exceptions thrown.

Classes, interfaces, structs

In the first sentence of a class description, briefly state the intended purpose or function of the class or interface with information that can't be deduced from the class name and signature. In additional documentation, elaborate on how to use the API, including how to invoke or instantiate it, what some of the key features are, and any best practices or pitfalls.

Many doc tools automatically extract the first sentence of each class description for use in a list of all classes, so make the first sentence unique and descriptive, yet short. Additionally:

- Do not repeat the class name in the first sentence.
- Do not say "this class will/does ...."
- Do not use a period before the actual end of the sentence, because some doc generators naively terminate the "short description" at the first period.
    - For example, some generators terminate the sentence if they see "*e.g.*", so use "*for example*" instead.

Methods

In the first sentence for a method description, briefly state **what** action the method performs. In subsequent sentences, explain **why and how to use** the method, state any prerequisites that must be met before calling it, give details about **exceptions** that may occur, and **specify any related APIs**.

Use **present tense** for all descriptions—for example:

- *Adds a new bird to the ornithology list.*
- *Returns a bird.*

Description

- If a method performs an operation and returns some data, start the description with a verb describing the operation—for example:
    - *Adds a new bird to the ornithology list and returns the ID of the new entry.*
- If it's a "getter" method and it returns a boolean, start with
    - *Checks whether ....*
- If it's a "getter" method and it returns something other than a boolean, start with
    - *Gets the ....*
- If it has no return value, start with a verb like one of the following:
    - Turning on an ability or setting:
        - *Sets the ....*
    - Updating a property:
        - *Updates the ....*
    - Deleting something:
        - *Deletes the ....*
- If it's a convenience method that constructs the class object, start with
    - *Creates a ....*

Parameters

For parameter descriptions, follow these guidelines:

- Capitalize the first word, and end the sentence or phrase with a period.

Non Boolean

- Begin descriptions of non-boolean parameters with "*The*" or "*A*" if possible:
    - *The ID of the bird you want to get.*
    - *A description of the bird.*

Boolean

- For boolean parameters for requesting an action, start sentences with
    - *If true …* and *If false ….*
        - Example: *If true, turn traffic lines on. If false, turn them off.*
- For boolean parameters for setting the state of something (not making a request), use the format
    - *True if …; false otherwise.*
        - Example: *True if the zoom is set; false otherwise.*
- In this context, don't put the words "true" and "false" in code font or quotation marks.

Return Values

Be as brief as possible in the return value's description; put any detailed information in the class description.

Non Boolean

If the return value is anything other than a boolean, start with

- *The ….*
    - Example: *The bird specified by the given ID.*

Boolean

If the return value is a boolean, use the format

- *True if …; false otherwise.*
    - Example: *True if the bird is in the sanctuary; false otherwise.*

Exceptions

In languages where the reference generator automatically inserts the word `Throws`, begin your description with

- *If …*
    - Example: *If no key is assigned.*

Otherwise, begin with

- *Thrown when …*

- Example: *Thrown when no key is assigned.*

Style Choice - Table of Contents

If you want to see the table of contents in a sidebar, modify the `docfx.json` file (found in your solution explorer). Look for the `dest:` keyword, near the end of the file

change from:

```
    "dest": "_site",
    "template": [
      "default"
    ]
```

to:

```
    "dest": "_site",
    "template": [
      "statictoc"
    ]
```