Tomdoc Cheatsheet

A concise cheat sheet for Tomdoc, a documentation style emphasizing clarity through simple conventions. This guide covers syntax, tags, and best practices for generating understandable and maintainable documentation.

Tomdoc Basics

General Structure

Tomdoc uses a simple structure to document code:

1. **Public:** or **Internal:** tag
2. A description of what the code does.
3. Arguments (if any).
4. Examples (if applicable).
5. Return value.

The key is to keep it concise and focused on what the code *does*, not *how* it does it.

Tags

`Public:`	Indicates the method or class is part of the public API and intended for external use.
`Internal:`	Indicates the method or class is for internal use only and not part of the public API.
`Deprecated:`	Marks the method as deprecated, optionally including information about alternatives.

Arguments

List arguments with their descriptions and types.

```
# text - The String to be duplicated.
# count - The Integer number of times to
duplicate the text.
```

Examples and Returns

Examples

Provide clear examples of how to use the code.

```
# Examples
#
#   multiplex('Tom', 4)
#   # => 'TomTomTomTom'
```

Returns

Describe the return value, including its type and meaning.

```
# Returns the duplicated String.
```

Putting it Together

```
# Public: Duplicate some text an
arbitrary number of times.
#
# text  - The String to be duplicated.
# count - The Integer number of times to
duplicate the text.
#
# Examples
#
#   multiplex('Tom', 4)
#   # => 'TomTomTomTom'
#
# Returns the duplicated String.
def multiplex(text, count)
  text * count
end
```

Advanced Tomdoc

Options Hashes

Document options hashes, including keys, types, and descriptions.

```
# options - The Hash options used to
refine the selection (default: {})
#           :color  - The String color
to restrict by (optional).
#           :weight - The Float weight
to restrict by.
```

Yields

Describe what the code yields, including the type of yielded values.

```
# Yields the Integer index of the
iteration.
```

Raises

Document any exceptions that may be raised.

```
# Raises Errno::ENOENT if the file can't
be found.
```

Signatures and Style

Signatures (for DSLs)

Use signatures to describe the syntax of DSLs.

```
# Signature
#
#   find_by_<field>[_and_<field>...]
(args)
```

Style Considerations

- Keep descriptions concise and focused.
- Use complete sentences.
- Prefer active voice.
- Write from the perspective of someone using the code.
- Be consistent with formatting.

Example of Bad Tomdoc

```ruby
# Public
# Returns a string with the text
multiplied by count
def multiplex(text, count)
  text * count
end
```

This is bad because it doesn’t specify argument types and lacks an example