# Semantic Versioning and the PHP Ecosystem: A Reality Check

Daniel Scherzer

# Daniel Scherzer

- Full-stack developer and open source contributor
- https://scherzer.dev
- https://github.com/DanielEScherzer
- PHP core contributor
  - ext/reflection maintainer
  - 8.5 rookie release manager
  - 8.6 veteran release manager

# Overview

- SemVer
- Case analysis
- Public API
- Questions

# SemVer Spec Item 1

> Software using Semantic Versioning MUST declare a public API. This API could be declared in the code itself or exist strictly in documentation. However it is done, it SHOULD be precise and comprehensive.

And yet, how often do packages do this?

# Hyrum's Law

With a sufficient number of users of an API, it does not matter what you promise in the contract: all observable behaviors of your system will be depended on by somebody.

## XKCD 1172 ----->

# Symfony

https://symfony.com/doc/current/contributing/code/bc.html

> However, backward compatibility comes in many different flavors. In fact, almost every change that we make to the framework can potentially break an application. For example, if we add a new method to a class, this will break an application which extended this class and added the same method, but with a different method signature.

> Also, not every BC break has the same impact on application code. While some BC breaks require you to make significant changes to your classes or your architecture, others are fixed by changing the name of a method.

# Private member removal

```
diff --git a/src/lib/src/HasPrivateStuff.php b/src/lib/src/HasPrivateStuff.php
index b140eb2..98c4c85 100644
--- a/src/lib/src/HasPrivateStuff.php
+++ b/src/lib/src/HasPrivateStuff.php
@@ -5,13 +5,4 @@ namespace DanielEScherzer\PHPTek2026DemoLib;
 class HasPrivateStuff {

         // Remove each of these private elements
-
-       private $instanceProp;
-       private static $staticProp;
-
-       private const MY_CONST = true;
-
-       private function instanceMethod() {}
-
-       private static function staticMethod() {}
 }
```

# Parameter type widening (1)

```
diff --git a/src/lib/src/ExceptionsAndErrors.php b/src/lib/src/ExceptionsAndErrors.php
index dfdca18..9032597 100644
--- a/src/lib/src/ExceptionsAndErrors.php
+++ b/src/lib/src/ExceptionsAndErrors.php
@@ -23,7 +23,7 @@ class ExceptionsAndErrors {

        public static function withWrongType(
                // convert to bool|string
-               bool $param
+               bool|string $param
        ) {}

        public static function withAnyType(
```

# Parameter type widening (2)

```
diff --git a/src/lib/src/ExceptionsAndErrors.php b/src/lib/src/ExceptionsAndErrors.php
index 9032597..cd42ff8 100644
--- a/src/lib/src/ExceptionsAndErrors.php
+++ b/src/lib/src/ExceptionsAndErrors.php
@@ -28,7 +28,7 @@ class ExceptionsAndErrors {

         public static function withAnyType(
                 // remove the type
-                bool $param
+                $param
         ) {}

 }
```

# Exception subclassing

```
diff --git a/src/lib/src/ExceptionsAndErrors.php
b/src/lib/src/ExceptionsAndErrors.php
index 9166c6d..8886d4f 100644
--- a/src/lib/src/ExceptionsAndErrors.php
+++ b/src/lib/src/ExceptionsAndErrors.php
@@ -11,8 +11,8 @@ class ExceptionsAndErrors {

         public static function throwSpecificType() {
                 // Uncomment this earlier exception
-                // throw new LibraryException( "Subclass" );
-                throw new \RuntimeException( "Base class" );
+                throw new LibraryException( "Subclass" );
+                // throw new \RuntimeException( "Base class"
);
         }

         public static function throwGeneralType() {
```

```
diff --git a/src/lib/src/LibraryException.php
b/src/lib/src/LibraryException.php
new file mode 100644
index 0000000..b90a925
--- /dev/null
+++ b/src/lib/src/LibraryException.php
@@ -0,0 +1,4 @@
+<?php
+
+namespace DanielEScherzer\PHPTek2026DemoLib;
+class LibraryException extends \RuntimeException {}
```

# Exception reclassification

```
diff --git a/src/lib/src/ExceptionsAndErrors.php b/src/lib/src/ExceptionsAndErrors.php
index 8886d4f..dfdca18 100644
--- a/src/lib/src/ExceptionsAndErrors.php
+++ b/src/lib/src/ExceptionsAndErrors.php
@@ -17,8 +17,8 @@ class ExceptionsAndErrors {

        public static function throwGeneralType() {
                // Uncomment this earlier exception
-               // throw new \LogicException( "LogicException" );
-               throw new \RuntimeException( "RuntimeException" );
+               throw new \LogicException( "LogicException" );
+               // throw new \RuntimeException( "RuntimeException" );
        }

        public static function withWrongType(
```

# Exception messages

```
diff --git a/src/lib/src/ExceptionsAndErrors.php b/src/lib/src/ExceptionsAndErrors.php
index df979bd..9166c6d 100644
--- a/src/lib/src/ExceptionsAndErrors.php
+++ b/src/lib/src/ExceptionsAndErrors.php
@@ -6,7 +6,7 @@ class ExceptionsAndErrors {

        public static function throwMessage() {
                // Fix this typo
-               throw new \Exception( "has a tyop" );
+               throw new \Exception( "has a typo" );
        }

        public static function throwSpecificType() {
```

# Do not expect errors

- Private vs missing properties, constants, and methods
- Parameter types - error messages vs acceptance
- Exceptions as covariant, messages as meaning

# Documentation changes

```
diff --git a/src/lib/src/Reflectable.php b/src/lib/src/Reflectable.php
index ff9571d..7e291e1 100644
--- a/src/lib/src/Reflectable.php
+++ b/src/lib/src/Reflectable.php
@@ -3,7 +3,7 @@
 namespace DanielEScherzer\PHPTek2026DemoLib;

 /**
- * This has a tyop
+ * This has a typo
 *
 * Fix the typo in ^^^
 */
```

# Property reorganization

```
diff --git a/src/lib/src/Reflectable.php b/src/lib/src/Reflectable.php
index 7e291e1..ffcc74d 100644
--- a/src/lib/src/Reflectable.php
+++ b/src/lib/src/Reflectable.php
@@ -10,8 +10,8 @@ namespace DanielEScherzer\PHPTek2026DemoLib;
 class Reflectable {

        // Flip the order of these property declarations
-       public mixed $a;
        public mixed $b;
+       public mixed $a;

        public function __construct(mixed $a, mixed $b) {
                $this->a = $a;
```

# Do not use reflection

- Presence of properties/methods
- Not limited to ext/reflection:
  - get_class_vars()
  - get_object_vars()
  - get_mangled_object_vars()
  - get_class_methods()
  - method_exists()
  - property_exists()
  - constant()
  - class_uses()
  - Array casting?

# Refactoring (1)

```
diff --git a/src/lib/src/Helper.php b/src/lib/src/Helper.php
new file mode 100644
index 0000000..9e4abea
--- /dev/null
+++ b/src/lib/src/Helper.php
@@ -0,0 +1,8 @@
+<?php
+
+namespace DanielEScherzer\PHPTek2026DemoLib;
+class Helper {
+	public static function add(int $a, int $b): int {
+		return $a + $b;
+	}
+}
```

```
diff --git a/src/lib/src/WithHelper.php b/src/lib/src/WithHelper.php
index 36bdd24..7e133c4 100644
--- a/src/lib/src/WithHelper.php
+++ b/src/lib/src/WithHelper.php
@@ -5,8 +5,8 @@ namespace
DanielEScherzer\PHPTek2026DemoLib;
 class WithHelper {

 	public static function add(int $a, int $b): int {
-		// return Helper::add($a, $b);
-		return $a + $b;
+		return Helper::add($a, $b);
+		// return $a + $b;
 	}

 }
```

# Namespace ownership

- PHP and extensions “own”:
  - Global namespace
  - Namespaces with only one part (e.g. ext/uri, ext/random)
    - And any sub-namespaces (e.g. \Uri\Rfc3986*, \Uri\WhatWG*)
- Libraries “own”:
  - The namespace under their names
    - And any sub-namespaces

# Refactoring (2)

```
diff --git a/src/lib/src/InvokesCallback.php b/src/lib/src/InvokesCallback.php
index 4b8aca4..4849f49 100644
--- a/src/lib/src/InvokesCallback.php
+++ b/src/lib/src/InvokesCallback.php
@@ -6,8 +6,12 @@ class InvokesCallback {

        public function __construct(private \Closure $cb) {
                // Switch to using doInvoke()
-               $cb();
-               // $this->doInvoke();
+               // $cb();
+               $this->doInvoke();
+       }
+
+       private function doInvoke() {
+               ($this->cb)();
        }

 }
```

# Not a straw-man

```php
$old = Writer::getExtensionManager();
$manager = new class( $old ) extends ExtensionManager {
	public function has( $entryName ) {
		if ( $entryName === 'Slash\Renderer\Entry' ) {
			// Needs to return true when checking if the extension
			// exists, otherwise Laminas throws an exception
			$bt = debug_backtrace( DEBUG_BACKTRACE_IGNORE_ARGS, 2 );
			if (
				$bt[1]['class'] === Writer::class
				&& $bt[1]['function'] === 'hasExtension'
			) {
				return true;
			}
			return false;
		}
		return parent::has( $entryName );
	}
};
Writer::setExtensionManager( $manager );
```

# Typing

```
diff --git a/src/lib/src/HasUntypedConstant.php b/src/lib/src/HasUntypedConstant.php
index 7366515..17207b2 100644
--- a/src/lib/src/HasUntypedConstant.php
+++ b/src/lib/src/HasUntypedConstant.php
@@ -5,6 +5,6 @@ namespace DanielEScherzer\PHPTek2026DemoLib;
 class HasUntypedConstant {

         // Add a type here
-        public const /* bool */ MY_CONST = true;
+        public const bool MY_CONST = true;

 }
```

# Deprecations and escalations

- `#[\Deprecated]`
- E_DEPRECATED, E_USER_DEPRECATED
- set_error_handler()

# Updated Spec

The public API of a PHP library, if not otherwise documented:

- Does **not** include the presence or type of any specific engine-created errors (trying to use something private vs missing, wrong types, etc.)
- Does **not** include any information only accessible via reflection or debugging
- Does **not** include the specific class of any exceptions thrown, just the type (i.e. use `instanceof` rather than `get_class()`)
- Does **not** include the string contents of error messages
    - But **does** include the semantic meaning of those messages
- Assumes ownership of the entire applicable PHP namespace(s)

# Thank You

Questions?

# Feedback on Joind.in