API Collection

Debugging and Reflection

Fortify your code with runtime checks, and examine your values' runtime representation.

Topics

Printing and Dumping

`func print(Any..., separator: String, terminator: String)`

Writes the textual representations of the given items into the standard output.

`func print<Target>(Any..., separator: String, terminator: String, to: inout Target)`

Writes the textual representations of the given items into the given output stream.

`func debugPrint(Any..., separator: String, terminator: String)`

Writes the textual representations of the given items most suitable for debugging into the standard output.

`func debugPrint<Target>(Any..., separator: String, terminator: String, to: inout Target)`

Writes the textual representations of the given items most suitable for debugging into the given output stream.

`func dump<T>(T, name: String?, indent: Int, maxDepth: Int, maxItems: Int) -> T`

Dumps the given object's contents using its mirror to standard output.

```
func dump<T, TargetStream>(T, to: inout TargetStream, name: String?,
indent: Int, maxDepth: Int, maxItems: Int) -> T
```

Dumps the given object’s contents using its mirror to the specified output stream.

Testing

```
func assert(@autoclosure () -> Bool, @autoclosure () -> String, file:
StaticString, line: UInt)
```

Performs a traditional C-style assert with an optional message.

```
func assertionFailure(@autoclosure () -> String, file: StaticString,
line: UInt)
```

Indicates that an internal consistency check failed.

```
func precondition(@autoclosure () -> Bool, @autoclosure () -> String,
file: StaticString, line: UInt)
```

Checks a necessary condition for making forward progress.

```
func preconditionFailure(@autoclosure () -> String, file: StaticString,
line: UInt) -> Never
```

Indicates that a precondition was violated.

Exiting a Program

```
func fatalError(@autoclosure () -> String, file: StaticString, line:
UInt) -> Never
```

Unconditionally prints a given message and stops execution.

```
enum Never
```

A type that has no values and can’t be constructed.

Querying Runtime Values

```
struct Mirror
```

A representation of the substructure and display style of an instance of any type.

```
struct ObjectIdentifier
```

A unique identifier for a class instance or metatype.

```
func type<T, Metatype>(of: borrowing T) -> Metatype
```

Returns the dynamic type of a value.

Customizing Your Type’s Reflection

Provide a custom reflection for your types using these protocols.

`protocol CustomReflectable`

A type that explicitly supplies its own mirror.

`protocol CustomLeafReflectable`

A type that explicitly supplies its own mirror, but whose descendant classes are not represented in the mirror unless they also override `customMirror`.

`protocol CustomPlaygroundDisplayConvertible`

A type that supplies a custom description for playground logging.

`typealias PlaygroundQuickLook`

The sum of types that can be used as a Quick Look representation.

`macro DebugDescription()`

Converts description definitions to a debugger Type Summary.

See Also

Programming Tasks

Input and Output

Print values to the console, read from and write to text streams, and use command line arguments.

Macros

Generate boilerplate code and perform other compile-time operations.

Concurrency

Perform asynchronous and parallel operations.

Key-Path Expressions

Use key-path expressions to access properties dynamically.

Manual Memory Management

Allocate and manage memory manually.

Type Casting and Existential Types

Perform casts between types or represent values of any type.

C Interoperability

Use imported C types or call C variadic functions.

Operator Declarations

Work with prefix, postfix, and infix operators.