API Collection

Initialization with Literals

Allow values of your type to be expressed using different kinds of literals.

Topics

Collection Literals

`protocol ExpressibleByArrayLiteral`

A type that can be initialized using an array literal.

`protocol ExpressibleByDictionaryLiteral`

A type that can be initialized using a dictionary literal.

Value Literals

`protocol ExpressibleByIntegerLiteral`

A type that can be initialized with an integer literal.

`protocol ExpressibleByFloatLiteral`

A type that can be initialized with a floating-point literal.

`protocol ExpressibleByBooleanLiteral`

A type that can be initialized with the Boolean literals `true` and `false`.

`protocol ExpressibleByNilLiteral`

A type that can be initialized using the nil literal, `nil`.

`struct StaticBigInt`

An immutable arbitrary-precision signed integer.

String Literals

`protocol ExpressibleByStringLiteral`

A type that can be initialized with a string literal.

`protocol ExpressibleByExtendedGraphemeClusterLiteral`

A type that can be initialized with a string literal containing a single extended grapheme cluster.

`protocol ExpressibleByUnicodeScalarLiteral`

A type that can be initialized with a string literal containing a single Unicode scalar value.

`protocol ExpressibleByStringInterpolation`

A type that can be initialized by string interpolation with a string literal that includes expressions.

`protocol StringInterpolationProtocol`

Represents the contents of a string literal with interpolations while it's being built up.

`struct DefaultStringInterpolation`

Represents a string literal with interpolations while it's being built up.

Default Types for Literals

Default Literal Types

Type aliases representing the concrete type that a literal takes when no other type information is provided.

See Also

Tools for Your Types

Basic Behaviors

Use your custom types in operations that depend on testing for equality or order and as members of sets and dictionaries.

Encoding, Decoding, and Serialization

Serialize and deserialize instances of your types with implicit or customized encoding.