**Computer Vocabulary S01**

|  |  |
| --- | --- |
| **syntax** n. 语法，句法 | **variable** n. 变量 |
| **thread** n. 线程 | **constant** n. 常量 |
| **compile** v. 编译 | **object** n. 对象 |
| **optimization** n. 优化 | **property** n. 属性 |
| **delegate** n. 代理 | **array** n. 数组 |
| **instance** n. 实例 | **dictionary** n. 字典 |
| **initialization** n. 初始化，初始化方法 | **loop** n. 循环 |
| **protocol** n. 协议，接口 | **conditional** adj. 条件的 |
| **framework** n. 框架 | **type inference** phr.类型推断 |
| **algorithm** n. 算法，计算程序 | **casting** n. 强制类型转换 |
| **parameter** n. 参数 | **typecasting** n. 类型转换 |
| **debugging** n. 调试 | **struct** n. 结构体 |
| **refactoring** n. 重构，代码重整 | **function** n. 函数 |
| **override** n. 覆盖，重写 | **tuple** n. 元组 |
| **inheritance** n. 继承 | **pointer** n. 指针 |
| **encapsulation** n. 封装，包封 | **operator** n. 运算符 |
| **polymorphism** n. 多态 | **optional** adj. 可选的，可为 nil 的 |
| **enumeration** n. 枚举 | **closure** n. 闭包 |
| **subscript** n. 下标，下标脚本 | **guard** n. 守卫，守卫语句 |
| **extension** n. 扩展，延伸 | **generic** adj. 泛型的 |

▲ **syntax** [ˈsɪntæks] n. 语法

eg. Understanding the syntax of a programming language is crucial to writing correct code.

理解编程语言的语法对于编写正确的代码至关重要。

**▲ thread** [θrɛd] n. 线程

eg. A thread is a sequence of instructions that can be executed independently of the main program. 线程是一组可以独立执行于主程序之外的指令序列。

▲**compile** [kəmˈpaɪl] v. 编译

eg. The source code needs to be compiled into machine code before it can be executed.

源代码需要编译成机器代码才能被执行。

▲ **optimization** [ˌɑːptɪməˈzeɪʃn] n. 优化

eg. Optimization is the process of making a program or system more efficient, faster, or using fewer resources. 优化是使程序或系统更加高效、更快或使用更少资源的过程。

▲**delegate** [ˈdelɪɡət] n. 代理

eg. A delegate is an object that acts on behalf of, or in coordination with, another object.

代理是一个代表另一个对象行动或协调的对象。

▲ **instance** ['ɪnstəns] n. 实例

eg. An instance is a unique copy of a class or structure that you can create and work with. 实例是类或结构体的唯一副本，您可以创建并与之交互。

▲ **initialization** [ɪˌnɪʃəlaɪˈzeɪʃn] n. 初始化，初始化方法

eg. Initialization is the process of preparing an instance of a class, structure, or enumeration for use. 初始化是为使用类、结构体或枚举的实例准备过程。

▲ **protocol** ['proʊtəkɔl] n. 协议

eg. A protocol defines a set of rules that networked devices use to communicate with each other.

协议定义了网络设备用于彼此通信的一组规则。

▲ **framework** [ˈfreɪmwɔrk] n. 框架

eg. A framework is a collection of libraries and tools that help developers build software more efficiently. 框架是一组库和工具的集合，可以帮助开发人员更高效地构建软件。

▲ **algorithm** [ˈælɡərɪðəm] n. 算法

eg. An algorithm is a set of instructions that solves a specific problem or performs a specific task.

算法是一组解决特定问题或执行特定任务的指令。

▲ **parameter** [pəˈræmɪtər] n. 参数

eg. A parameter is a value that can be passed to a function or method to customize its behavior.

参数是可以传递给函数或方法以定制其行为的值。

▲ **debugging** [diːˈbʌɡɪŋ] n. 调试

eg. Debugging is the process of identifying and fixing errors, or bugs, in a program.

调试是识别和修复程序中错误或漏洞的过程。

▲ **refactoring** [ˌriːˈfæktərɪŋ] n. 重构

eg. Refactoring is the process of improving code by restructuring it without changing its functionality. 重构是通过重组代码来改进代码，而不改变其功能的过程。

▷ refractor [rɪˈfræktər] v. 重构

▲ **override** [oʊvərˈraɪd] v. 覆盖，重写

eg. To override a method, you provide a new implementation that replaces the existing one.

要重写方法，您需要提供一个新的实现来替换现有的实现。

▲**inheritance** [ɪnˈherɪtəns] n. 继承

eg. Inheritance is a mechanism in object-oriented programming that allows a class to inherit properties and methods from a superclass. 继承是面向对象编程中的一种机制，允许一个类从一个超类继承属性和方法。

▲**encapsulation** [ɪnˌkæpsjuˈleɪʃ(ə)n] n. 封装，包装

eg. Encapsulation is a programming concept that hides implementation details of a class from other parts of the program. 封装是一种编程概念，它将一个类的实现细节隐藏起来，不让程序的其他部分看到。

▷ encapsulate [ɪnˈkæpsjuleɪt] v. 概括，压缩，简述

▲**polymorphism** [ˌpɑliˈmɔrfɪzəm] n. 多态

eg. Polymorphism allows objects of different types to be treated as if they were the same type.

多态性允许将不同类型的对象视为同一类型。

▲ **enumeration** [ˌenjʊməˈreɪʃən] n. 枚举

eg. An enumeration defines a common type for a group of related values and enables you to work with those values in a type-safe way within your code. 枚举定义了一组相关值的常见类型，并使您可以在代码中以类型安全的方式使用这些值。

▲ **subscript** [ˈsʌbskrɪpt] n. 下标，下标脚本

eg. A subscript allows you to access the value of a collection, list, or sequence using a shortcut syntax. 下标允许您使用快捷语法访问集合、列表或序列的值。

▲ **extension** [ɪkˈstenʃn] n. 扩展，延伸

eg. An extension adds new functionality to an existing class, structure, enumeration, or protocol type. 扩展可以向现有的类、结构体、枚举或协议类型添加新功能。

▲ **variable** ['vɛriəbl] n. 变量

eg. The value of a variable can be changed during the execution of a program.

变量的值可以在程序执行过程中更改。

▲ **constant** ['kɑnstənt] n. 常量

eg. Constants are used to represent values that cannot be changed. 常量表示不能更改的值。

▲ **object** ['ɑbdʒɛkt] n. 对象

eg. An object is an instance of a class that contains data and behavior.

对象是包含数据和行为的类的实例。

▲ **property** ['prɑpərti] n. 属性

eg. A property is a value or characteristic of an object or a class.

属性是对象或类的一个值或特征。

▲ **array** [ə're] n. 数组

eg. An array is a collection of elements, each identified by an index or a key.

数组是一个由元素组成的集合，每个元素都有一个索引或键来标识。

▲ **dictionary** ['dɪkʃənɛri] n. 字典

eg. A dictionary is a collection of key-value pairs that provides fast access to its elements.

字典是一组键值对的集合，提供快速访问其元素的能力。

▲ **loop** [lu:p] n. 循环

eg. A loop is a control structure that allows a piece of code to be executed repeatedly.

循环是一种控制结构，允许重复执行一段代码。

▲ **conditional** [kən'dɪʃənəl] adj. 条件的

eg. A conditional statement allows different actions to be taken based on a certain condition.

条件语句允许根据某种条件采取不同的行动。

▲ **type inference** [taɪp 'ɪnfərəns] n. 类型推断

eg. Type inference is a feature in Swift that allows the compiler to deduce the type of a variable.

类型推断是Swift中的一项功能，允许编译器推断变量的类型。

▲ **casting** [kæstɪŋ] n. 强制类型转换

eg. We need to do a casting to convert an AnyObject to a specific type.

我们需要进行强制类型转换，将AnyObject转换为特定类型。

▲ **typecasting** ['taɪpkæstɪŋ] n. 类型转换

eg. Typecasting is the process of converting an object from one type to another.

类型转换是将一个对象从一种类型转换为另一种类型的过程。

▲ **struct** [strʌkt] n. 结构体

eg. A struct is a value type that can contain properties, methods, and subscripts.

结构体是一个值类型，可以包含属性、方法和下标。

▲ **function** ['fʌŋkʃən] n. 函数

eg. We need to refactor this function to make it more efficient.

我们需要重构这个函数，使其更加高效。

▲ **tuple** ['tʌpl] n. 元组

eg. A tuple is a collection of values of different types.

元组是不同类型的值的集合。

▲ **pointer** ['pɔɪntər] n. 指针

eg. A pointer is a variable that stores the memory address of another variable.

指针是一种变量，它存储另一个变量的内存地址。

▲ **operator** ['ɑːpəreɪtər] n. 运算符

eg. Swift provides a variety of operators for performing different operations.

Swift提供了各种运算符，用于执行不同的操作

**▲ optional** [ˈɑːpʃənl] adj. 可选的，可为 nil 的

eg. In Swift, an optional is a type that can either have a value or be nil.

在 Swift 中，optional 是一种类型，它可以有值也可以是 nil。

▲ **closure** [ˈkloʊʒər] n. 闭包

eg. A closure is a self-contained block of functionality that can be passed around and used in your code. 闭包是一个自包含的功能块，可以在代码中传递和使用。

▲ **guard** [ɡɑːrd] n. 守卫，守卫语句

eg. The guard statement is used to require that a condition must be true in order for the code after the guard statement to be executed. 守卫语句用于要求一个条件必须为真，以便执行guard语句之后的代码。

▲**generic** [dʒəˈnerɪk] adj. 泛型的，通用的

eg. Generic code allows you to write flexible and reusable functions and types that can work with different data types. 泛型代码可以让您编写灵活且可重用的函数和类型，可以处理不同的数据类型。