**Computer Vocabulary S02**

|  |  |
| --- | --- |
| **global** adj. 全局的 | **concurrency** n. 并发 |
| **deduce** v. 推断，演绎 | **infer** v. 推断 |
| **synthesize** v. 合成，综合 | **blueprint** n. 蓝图 |
| **nest** v. 嵌套，筑巢 | **scalability** n. 可扩展性 |
| **refer** v. 参考，提及 | **security** n. 安全性 |
| **compound** adj. 复合的，混合的 | **recursion** n. 递归 |
| **omit** v. 省略，遗漏 | **iteration** n. 迭代 |
| **arbitrary** adj. 任意的，武断的 | **heap** n. 堆 |
| **iterate** v. 迭代，重复 | **stack** n. 栈 |
| **exit** n. 退出，离开 | **interpreter** n. 解释器 |
| **pattern** n. 模式，设计 | **expression** n. 表达式 |
| **unwrap** v. 解包，拆包 | **declaration** n. 声明 |
| **implicit** adj. 隐含的，暗示的 | **implementation** n. 实现 |
| **boolean** adj. 布尔的，逻辑的 | **asynchronous** adj. 异步的 |
| **indices** n. 索引，下标 | **atomicity** n. 原子性 |
| **explicit** adj. 明确的 | **interface** n. 接口 |
| **automatical** adj. 自动的 | **selector** n. 选择器 |
| **lightweight** adj. 轻量级的 | **type conformance** phr. 类型一致性 |
| **concise** adj. 简洁的 | **codable** adj. 可编解码的 |
| **opaque** adj. 不透明的 | **dispatch** v. 调度 |

**▲ global** ['ɡloʊbl]adj. 全局的

eg. The use of global variables should be avoided in programming as they can lead to unexpected behavior. 编程中应避免使用全局变量，因为它们可能导致意外的行为。

▲**deduce** [dɪˈduːs] v. 推断，演绎

eg. Based on the input data, the program can deduce the correct output.

根据输入数据，程序可以推断出正确的输出。

▲ **synthesize** ['sɪnθəsaɪz] v. 合成，综合

eg. The compiler will synthesize a default implementation of the equality operator if one is not provided in Swift. 如果在Swift中没有提供等号操作符的实现，编译器将综合出一个默认实现。

▲ **nest** [nest] v. 嵌套；筑巢

eg. In object-oriented programming, classes can be nested within other classes to organize code and data. 在面向对象编程中，类可以嵌套在其他类中，以组织代码和数据。

**▲ refer** [rɪˈfɜr] v. 参考，提及

eg. The Java API documentation is a valuable resource for developers to refer to when writing code. Java API文档是开发人员编写代码时的有价值的参考资源。

**▲** **compound** ['kɑmpaʊnd] adj. 复合的，混合的

eg. In programming, a compound data type is one that consists of multiple data elements.

在编程中，复合数据类型是由多个数据元素组成的数据类型。

**▲** **omit** [oʊˈmɪt] v. 省略，遗漏

eg. When declaring a function in Swift, you can omit the return keyword if the function returns a single expression. 在Swift中声明函数时，如果函数返回单个表达式，则可以省略return关键字。

**▲** **arbitrary** ['ɑrbɪtrɛri] adj. 任意的，武断的

eg. The order in which elements are returned by a hash table is arbitrary and may change over time. 哈希表返回元素的顺序是任意的，并且可能随时间变化。

**▲** **iterate** ['ɪtəreɪt] v. 迭代，重复

eg. In programming, we often use loops to iterate over a collection of data and perform a series of operations. 在编程中，我们经常使用循环迭代一个数据集合，并执行一系列操作。

**▲ exit** ['ɛksɪt] n. 退出，离开

eg. In Unix-based systems, the exit status of a program is an integer value returned to the operating system when the program exits. 在基于Unix的系统中，程序的退出状态是程序退出时返回给操作系统的整数值。

**▲ pattern** ['pætərn] n. 模式，设计

eg. In Swift, you can use the pattern matching operator to match a value against a set of patterns.

在Swift中，可以使用模式匹配操作符将一个值与一组模式进行匹配。

**▲ unwrap** [ʌnˈræp] v. 解包，拆包

eg. In Swift, an optional value must be unwrapped before it can be used in most cases.

在Swift中，大多数情况下必须解包可选值才能使用它。

**▲ implicit** [ɪmˈplɪsɪt] adj. 隐含的，暗示的

eg. In Swift, you can use the implicit self keyword to refer to the current instance of a class or struct. 在Swift中，可以使用隐式的self关键字引用一个类或结构体的当前实例。

**▲ boolean** [ˈbuːliən] adj. 布尔的，逻辑的

eg. A boolean value in programming represents a logical truth value, either true or false.

编程中的布尔值表示逻辑真值，即true或false。

**▲ indices** ['ɪndɪsɪz] n. 索引，下标

eg. You can use the indices property of a collection to access the indices of its elements.

你可以使用集合的indices属性来访问其元素的索引。

**▲ explicit** [ɪkˈsplɪsɪt] adj. 明确的

eg. In Swift, it's recommended to use explicit optionals instead of implicitly unwrapped optionals.

在Swift中，建议使用明确的可选项而不是隐式解包的可选项。

**▲ automatical** [ˌɔːtəˈmætɪkl] adj. 自动的

eg. Automatic reference counting (ARC) is a memory management feature in Swift that automatically releases memory when it's no longer needed. Swift中的自动引用计数（ARC）是一种内存管理功能，当不再需要时自动释放内存。

**▲ lightweight** [ˈlaɪtˌweɪt] adj. 轻量级的

eg. Swift is a lightweight language that's easy to read and write.

Swift是一种易于阅读和编写的轻量级语言。

**▲ concise** [kənˈsaɪs] adj. 简洁的

eg. Swift's syntax is concise, making code easier to read and understand.

Swift的语法简洁，使代码更易于阅读和理解。

**▲ opaque** [oʊˈpeɪk] adj. 不透明的

eg. In Swift, opaque types allow you to hide implementation details and expose only what's necessary. 在Swift中，不透明类型允许您隐藏实现细节，仅公开必要的部分。

**▲** **concurrency** [kənˈkɜːrənsi] n. 并发

eg. Swift provides built-in support for concurrency with features like async/await and actors.

Swift提供了内置的并发支持，例如async/await和actors。

▷ **concurrent** [kənˈkɜːrənt] adj. 并发的，同时的

**▲ infer** [ɪnˈfɜːr] v. 推断

eg. In Swift, the compiler can infer the type of a variable based on its value.

在Swift中，编译器可以根据变量的值推断其类型。

**▲ blueprint** [ˈbluːprɪnt] n. 蓝图

eg. In Xcode, a blueprint is a reusable component that defines an interface for a type of object.

在Xcode中，蓝图是一种可重复使用的组件，为一种类型的对象定义接口。

**▲ scalability** [ˌskeɪləˈbɪləti] n. 可扩展性

eg. We need to design our system with scalability in mind to accommodate future growth.

我们需要考虑可扩展性，以适应未来的增长。

▷ **scalable** [ˈskeɪləbl] adj. 可扩展的

▲ **security** [sɪˈkjʊrəti] n. 安全性

eg. Security is important for our application to protect our users' data.

我们需要确保我们的应用程序是安全的，以保护我们用户的数据。

▲ **recursion** [rɪˈkɜːrʒn] n. 递归

eg. Recursion is a powerful tool in programming, but it can also lead to performance issues if not used carefully. 递归是编程中一个强大的工具，但如果不小心使用，它也可能导致性能问题。

▲ **iteration** [ɪtəˈreɪʃən] n. 迭代

eg. We can use a loop to perform iterations and process a collection of data.

我们可以使用循环来进行迭代，处理一组数据。

▲ **heap** [hiːp] n. 堆

eg. Heap is a data structure that allows efficient access to the highest or lowest value in a collection. 堆是一种数据结构，可以高效地访问集合中最高或最低的值。

▲ **stack** [stæk] n. 栈

eg. Stack is a data structure that allows last-in, first-out (LIFO) access to its elements.

栈是一种数据结构，可以按照后进先出（LIFO）的顺序访问其元素。

▲ **interpreter** [ɪnˈtɜːrprətər] n. 解释器

eg. An interpreter translates high-level code into machine code on the fly, allowing for more dynamic programming. 解释器可以实时将高级代码转换成机器码，从而实现更加动态的编程。

▲ **expression** [ɪkˈspreʃən] n. 表达式

eg. In programming, an expression is a combination of values, variables, operators, and function calls that can be evaluated to produce a result. 在编程中，表达式是由值、变量、运算符和函数调用组成的，可以求值得到结果。

▲ **declaration** [ˌdekləˈreɪʃn] n. 声明

eg. In programming, a declaration introduces a name and its type, allowing us to use that name later in the code. 在编程中，声明引入一个名称及其类型，使我们能够在代码中后续使用该名称。

▲ **implementation** [ˌɪmpləmɛnˈteɪʃən] n. 实现

eg. Implementation is the process of turning a design or specification into a working software system. 实现是将设计或规范转化为工作软件系统的过程。

▲ **asynchronous** [eɪˈsɪŋkrənəs] adj. 异步的

eg. Asynchronous programming can help improve the responsiveness and performance of your application. 异步编程可以帮助提高应用程序的响应性和性能。

▷ **synchronous** [ˈsɪŋkrənəs] adj. 同步的，共时的

▲ **atomicity** [ˌætəˈmɪsəti] n. 原子性

eg. In database systems, atomicity is a property that ensures that database transactions are treated as a single, indivisible unit of work. 在数据库系统中，原子性是一种属性，它确保数据库事务被视为单个不可分割的工作单元。

▷ **atomic** [əˈtɑːmɪk] adj. 原子的

▲ **interface** [ˈɪntərfeɪs] n. 接口

eg. An interface defines a set of methods or properties that a class or struct must implement in order to provide a specific capability. 接口定义了一组方法或属性，类或结构必须实现这些方法或属性以提供特定的功能。

▲ **selector** [sɪˈlɛktər] n. 选择器

eg. In Objective-C, a selector is a way of referring to a method by name.

在Objective-C中，选择器是一种按名称引用方法的方式。

▲ **type conformance** 类型一致性

eg. In Swift, type conformance refers to a type's ability to conform to a protocol.

在Swift中，类型一致性指的是类型符合协议的能力。

▲ **codable** [ˈkoʊdəbl] adj. 可编码的，可解码的

eg. The Codable protocol in Swift provides a convenient way to convert between data in your app and external representations, such as JSON or Property List. Swift中的Codable协议提供了一种方便的方法，用于在应用程序中将数据与外部表示形式（如JSON或Property List）之间进行转换。

**▲ dispatch** [dɪˈspætʃ] v. 调度

eg. We can use Grand Central Dispatch (GCD) to schedule tasks in a concurrent or serial queue.

我们可以使用Grand Central Dispatch（GCD）在并发或串行队列中调度任务。