**Computer Vocabulary S03**

|  |  |
| --- | --- |
| **resolution** n. 分辨率 | **digit** n. 数字 |
| **circular reference** phr. 循环引用 | **non-negative** adj. 非负的 |
| **semicolon** n. 分号 | **represent** v. 表示 |
| **colon** n. 冒号 | **deallocate** v. 释放（内存） |
| **parenthesis** n. 括号 | **abbreviate** v. 缩写 |
| **backslash** n. 反斜杠 | **maximum** n. 最大值 |
| **brackets** n. 方括号 | **regular expression** n. 正则表达式 |
| **comma** n. 逗号 | **pivot** n. 枢纽 |
| **dot** n. 点 | **traverse** n/v. 遍历，横越 |
| **braces** n. 大括号 | **algorithm** **complexity** phr. 算法复杂度 |
| **serial** adj. 串行的 | **manual cleanup** phr. 手动清理 |
| **Design Patterns** phr. 设计模式 | **exclamation** n. 感叹词，惊叹号 |
| **Singleton Pattern** phr. 单例模式 | **free up** phr. 释放，腾出 |
| **Observer Pattern** phr. 观察者模式 | **component** n. 组件，部件 |
| **Factory Pattern** phr. 工厂模式 | **module** n. 模块，组件 |
| **Proxy Pattern** phr. 代理模式 | **architecture** n. 架构，体系结构 |
| **Adapter Pattern** phr. 适配器模式 | **fundamental** adj. 基本的，根本的 |
| **Template Pattern** phr. 模板模式 | **double** adj. 双倍的，双浮点型 |
| **Strategy Pattern** phr. 策略模式 | **textual data** phr. 文本数据 |
| **Decorator Pattern** phr. 装饰器模式 | **throughout** prep. 遍及，贯穿 |

▲ **resolution** [ˌrɛzəˈluʃən] n. 分辨率；解决

eg. The resolution of a computer monitor is measured in pixels.

计算机显示器的分辨率是以像素为单位测量的。

▲**circular reference** [ˈsɜrkjələr ˈrɛfrəns] phr. 循环引用

eg. Circular references can cause memory leaks in a program.

循环引用可能会导致程序中的内存泄漏。

▲ **semicolon** [ˈsɛmɪˌkoʊlən] n. 分号

eg. In many programming languages, a semicolon is used to terminate a statement.

在许多编程语言中，分号用于终止语句。

▲ **colon** [ˈkoʊlən] n. 冒号

eg. In Swift, a colon is used to separate a variable's name from its type.

在Swift中，冒号用于分隔变量的名称和类型。

▲ **parenthesis** [pəˈrɛnθəsɪs] n. 括号（单数形式）

eg. In programming, parentheses are often used to group expressions or to pass arguments to functions. 在编程中，括号经常用于分组表达式或向函数传递参数。

▷ **parentheses** [pəˈrɛnθəsiz] n. 括号（复数形式）

**▲ backslash** [ˈbækˌslæʃ] n. 反斜杠

eg. In file paths on Windows, a backslash is used to separate directories.

在Windows文件路径中，反斜杠用于分隔目录。

▷ **slash** [slæʃ] n. 斜线

**▲ brackets** [ˈbrækɪts] n. 方括号 [ ]

eg. In Swift, brackets are used to declare an array or to access elements in an array.

在Swift中，方括号用于声明数组或访问数组中的元素。

**▲ comma** [ˈkɑmə] n. 逗号

eg. Commas are often used to separate items in a list or to separate parameters in a function call.

逗号经常用于分隔列表中的项目或在函数调用中分隔参数。

**▲ dot** [dɑt] n. 点

eg. In Swift, a dot is used to access a property or method of an object.

在Swift中，点用于访问对象的属性或方法。

**▲ braces** [breɪsɪz] n. 大括号 { }

eg. In Swift, braces are used to delimit blocks of code, such as function and class definitions.

在Swift中，大括号用于分隔代码块，如函数和类定义。

**▲ serial** [ˈsɪriəl] adj. 串行的

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。

▲ **design patterns** [dɪˈzaɪn ˈpætərnz] n. 设计模式

eg. Design patterns are reusable solutions to common software design problems.

设计模式是常见软件设计问题的可重用解决方案。

▲ **Singleton Pattern** [ˈsɪŋɡltən ˈpætərn] n. 单例模式

eg. The Singleton Pattern restricts the instantiation of a class to a single instance and provides a global point of access to it. 单例模式限制了一个类的实例化为单个实例，并提供全局访问点。

▲ **Observer Pattern** [əbˈzɜːrvər ˈpætərn] n. 观察者模式

eg. The Observer Pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

观察者模式定义了对象之间的一对多依赖关系，因此当一个对象更改状态时，所有依赖项都会自动收到通知并更新。

▲ **Factory Pattern** [ˈfæktri ˈpætərn] n. 工厂模式

eg. The Factory Pattern provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.

工厂模式为在超类中创建对象提供接口，但允许子类改变将被创建的对象的类型。

▲ **Proxy Pattern** [ˈprɑːksi ˈpætərn] n. 代理模式

eg. The Proxy Pattern provides a surrogate or placeholder object that acts as a substitute for another object, controlling access to it. 代理模式提供了一个替代或占位符对象，充当另一个对象的代替品，控制对其的访问。

▲ **Adapter Pattern** [əˈdæptər ˈpætərn] n. 适配器模式

eg. The Adapter Pattern allows objects with incompatible interfaces to work together by wrapping its own interface around that of an already existing class.

适配器模式通过在已有类的接口周围包装自己的接口，允许具有不兼容接口的对象一起工作。

▲ **Template Pattern** [ˈtɛmplət ˈpætərn] n. 模板模式

eg. The Template Pattern defines the skeleton of an algorithm in a superclass, but allows subclasses to override specific steps of the algorithm without changing its structure.

模板模式在超类中定义算法的框架，但允许子类覆盖算法的特定步骤，而不改变其结构。

▲ **Strategy Pattern** [ˈstrætədʒi ˈpætərn] n. 策略模式

eg. The Strategy Pattern defines a family of interchangeable algorithms and encapsulates each one so that they can be swapped out as needed at runtime.

策略模式定义了一组可交换的算法，并封装了每个算法，以便在运行时根据需要进行交换。

▲ **Decorator Pattern** [ˈdekəreɪtər ˈpætərn] n. 装饰器模式

eg. The Decorator Pattern attaches additional responsibilities to an object dynamically by wrapping it in a class that provides the needed functionality, without altering the original object.

装饰器模式通过将对象包装在提供所需功能的类中，动态地向对象附加额外的责任，而不会更改原始对象。

▲ **digit** [ˈdɪdʒɪt] n. 数字

eg. The number 357 is composed of three digits: 3, 5, and 7.

数字357由三个数字组成：3、5和7。

▲ **non-negative** [nɑn ˈnɛɡətɪv] adj. 非负的

eg. In mathematics, a non-negative number is a number that is greater than or equal to zero.

在数学中，非负数是大于或等于零的数。

▲ **represent** [ˌrɛprɪˈzɛnt] v. 表示

eg. In computer programming, we can use variables to represent values or objects.

在计算机编程中，我们可以使用变量来表示值或对象。

▲ **deallocate** [dɪˈæləkeɪt] v. 释放（内存），解除分配

eg. After using a dynamic memory allocation method, we must deallocate the memory when it is no longer needed. 使用动态内存分配方法后，我们必须在不再需要时记得释放内存。

▲ **abbreviate** [əˈbriːvɪeɪt] v. 缩写

eg. Sometimes we abbreviate words or phrases to make them shorter and more convenient to use, such as using "w/" to mean "with". 有时我们缩写单词或短语，以使它们更短、更方便使用，比如用“w/”表示“with”。

▲ **maximum** ['mæksɪməm] n&adj. 最大值；最大的

eg. The maximum value in this array is 10.

这个数组中的最大值是10。

▲ **regular expression** [ˈrɛɡjələr ɪkˈsprɛʃən] n. 正则表达式

eg. Regular expressions are a powerful tool for pattern matching and text processing.

正则表达式是一种强大的模式匹配和文本处理工具。

▲ **pivot** ['pɪvət] n&v. 枢轴；以…为中心转动

eg. We can use a pivot table to summarize and analyze large amounts of data.

我们可以使用数据透视表来汇总和分析大量的数据。

▲ **traverse** [ˈtrævɜːrs] n&v. 遍历；横越

eg. The algorithm traverses the graph in breadth-first order.

该算法以广度优先顺序遍历图形。

▲ **complexity** [kəmˈplɛksəti] n. 算法复杂度

eg. The time complexity of this algorithm is O(n^2).

该算法的时间复杂度为O(n^2)。

▲ **manual cleanup** ['mænjuəl 'kliːnʌp] phr. 手动清理

eg. We need to perform a manual cleanup of the database to remove duplicate records.

我们需要对数据库进行手动清理以删除重复记录。

▲ **exclamation** [ˌekskləˈmeɪʃn] n. 感叹词；惊叹号

eg. The exclamation "Oh no!" expressed her surprise and disappointment.

惊叹词“哦，不！”表达了她的惊讶和失望。

▲ **free up** [fri ʌp] v. 释放；腾出

eg. We need to free up some disk space by deleting unnecessary files.

我们需要通过删除不必要的文件来释放一些磁盘空间。

▲ **component** [kəmˈpoʊnənt] n. 组件；部件

eg. The software system is composed of many different components.

该软件系统由许多不同的组件组成。

▲ **module** ['mɑdʒuːl] n. 模块；组件

eg. The program is organized into several modules that can be reused in other projects.

该程序被组织成几个模块，可以在其他项目中重用。

▲ **architecture** ['ɑrkɪtɛktʃər] n. 架构；体系结构

eg. The system architecture defines the high-level structure of the software and its components.

系统架构定义了软件及其组件的高级结构。

▲ **fundamental** [ˌfʌndəˈmɛntl] adj. 基本的，根本的

eg. Understanding fundamental programming concepts is essential for any software developer.

了解基本的编程概念对于任何软件开发者来说都是必要的。

▲ **double** [ˈdʌbl] adj. 双倍的，双浮点的

eg. We need to double the amount of memory allocated to the application.

我们需要将分配给应用程序的内存数量加倍。

eg. These classes allow the various geometric shapes to be constructed with coordinates of either double or float precision. 这些类允许各种几何图形建立在双精度或浮点精度的坐标系上。

▲**textual data** [ˈtɛkstʃuəl ˈdeɪtə] n. 文本数据

eg. Our natural language processing algorithm works on textual data, such as emails or social media posts. 我们的自然语言处理算法适用于文本数据，如电子邮件或社交媒体帖子。

▲ **throughout** [θruːˈaʊt] prep. 遍及，贯穿

eg. We use version control throughout the development process to keep track of changes to the codebase. 我们在整个开发过程中使用版本控制来跟踪代码库的更改。