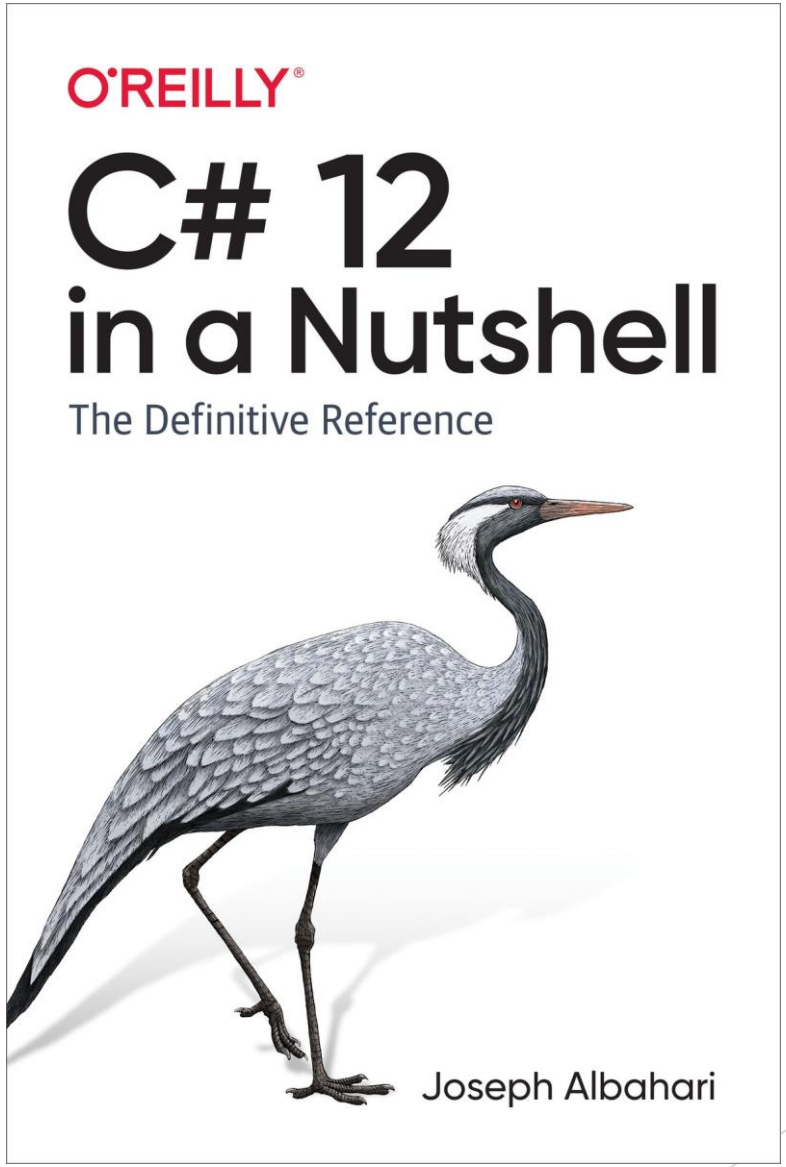
فصل اول - بخش اول

- معرفی سی شارپ
- هدف اصلی
- معمار سی شارپ
- شی گرایی
- Unified type system
- Classes and interfaces
- C# Functions
- Properties, events
- functional programming
- در حاشیه کتاب
- Type Safety
- مدیریت حافظه
- پشتیبانی از پلتفرم ها



معرفی سی شارپ

C# is a general-purpose, type-safe, object-oriented programming language.

سه ویژگی اصلی سی شارپ عبارتند از:

- همه منظوره
- Type Safe
- شی گرا

هدف اصلی

The goal of the language is programmer productivity.

هدف این زبان بهره وری برنامه نویس معرفی شده

To this end, C# balances simplicity, expressiveness, and performance.

و برای رسیدن به این هدف سه ویژگی زیر در نظر گرفته شده:

- Simplicity یا سادگی
- Expressiveness یا شفافیت (نمیدونم ترجمه درستی هست براش یا نه)
- Performance یا کارایی (که به نظرم ترجمه نشه قشنگ تره)

معمار سی شارپ

The chief architect of the language since its first version is Anders Hejlsberg (creator of Turbo Pascal and architect of Delphi).

معمار این زبان آقای Anders Hejlsberg هستند که خالق پاسکال و دلفی هم هستند ایشون.

The C# language is platform neutral and works with a range of platform-specific runtimes.

اینجا رو درست اگه متوجه شده باشم داره میگه سی شارپ کاری به پلتفرم نداره و رو تعداد زیادی از پلتفرم ها قابل اجراست. (اگه اشتباه متوجه شدم لطفا اصلاحم کنید)

شی گرایی

C# is a rich implementation of the object-orientation paradigm, which includes encapsulation, inheritance, and polymorphism.

سی شارپ یک پیاده سازی قدرتمند از الگوی شی گرایی شامل کپسوله سازی، وراثت و چندریختی است.

Encapsulation means creating a boundary around an object to separate its external (public) behavior from its internal (private) implementation details.

کپسوله سازی یعنی ایجاد یک حصار دور یک آبجکت تا رفتارهای عمومی اونو از مشخصات اختصاصی اون جدا کنیم.

Following are the distinctive features of C# from an object-oriented perspective

تو اسلایدهای بعد یه سری ویژگی های خاص سی شارپ معرفی میشه.

Unified type system

The fundamental building block in C# is an encapsulated unit of data and functions called a type. C# has a unified type system in which all types ultimately share a common base type.

پایه و اساس ساختار سی شارپ یک واحد کپسوله شده از داده ها و توابع هستش که به همش میگیم (نوع). در سی شارپ یک نوع واحدی داریم که همه انواع دیگه در نهایت از این نوع مشتق میشن. (امیدوارم درست فهمیده باشم).

This means that all types, whether they represent business objects or are primitive types such as numbers, share the same basic functionality.

یعنی اینکه همه نوع های موجود در سی شارپ یک سری رفتار مشابه دارند مثل تابع ToString

For example, an instance of any type can be converted to a string by calling its ToString method.

Classes and interfaces

In a traditional object-oriented paradigm, the only kind of type is a class.

در الگوی اولیه شی گرایی فقط مفهوم class وجود داشت.

In C#, there are several other kinds of types, one of which is an interface.

سی شارپ یه سری انواع دیگه هم داره از جمله interface.

An interface is like a class that cannot hold data. This means that it can define only behavior (and not state), which allows for multiple inheritance as well as a separation between specification and implementation.

اینترفیس شبیه کلاس هست ولی داده ای رو نگه نمیداره، یعنی:

- فقط یه تعریف از کارهایی که قراره انجام بده رو داره.
- بهمون اجازه میده که ارث بری چندگانه داشته باشیم.

C# Functions

In the pure object-oriented paradigm, all functions are methods.

در الگوی شی گرایی ناب(یعنی میخواد بگه سی شارپ خیلی هم ناب نیست یعنی؟ شایدم منظورش الگوی اولیه بوده!) همه عملیات(function) ها متد هستند.

In C#, methods are only one kind of function member, which also includes properties and events (there are others, too).

ولی در سی شارپ متدها یکی از انواع functionهای موجود هستند. انواع function در سی شارپ عبارتند از:

- Methods
- Properties
- Events
- و غیره

Properties, events

Properties are function members that encapsulate a piece of an object's state such as a button's color or a label's text.

پراپرتی ها یک نوع function هستند که وضعیت یک بخشی از یک آبجکت رو کنترل میکنند. مثل رنگ یک دکمه یا متن یک لیبل.

Events are function members that simplify acting on object state changes.

ایونت ها function هایی هستند که وقتی وضعیت یک آبجکت تغییر میکنه، کار میکنند. مثل تغییر متن درون تکست باکس یا تغییر موقعیت مکانی ماوس و ...

functional programming

Although C# is primarily an object-oriented language, it also borrows from the functional programming paradigm, specifically:

Functions can be treated as values

Using delegates, C# allows functions to be passed as values to and from other functions.

C# supports patterns for purity

Core to functional programming is avoiding the use of variables whose values change, in favor of declarative patterns. C# has key features to help with those patterns, including the ability to write unnamed functions on the fly that "capture" variables (lambda expressions), and the ability to perform list or reactive programming via query expressions. C# also provides records, which make it easy to write immutable (read-only) types.

functional programming

سی شارپ با وجود شی گرا بودن از قافله برنامه نویسی functional هم عقب نمونده و از این الگو هم تبعیت میکنه از جمله:

Function به عنوان مقدار

با استفاده از delegate ها در سی شارپ میتونیم فانکشن ها رو به عنوان مقدار ورودی به فانکشن های دیگه پاس بدیم.

پشتیبانی از الگوهایی برای purity

هسته اصلی برنامه نویسی functional برمبنای پرهیز از متغیرهایی است که مقدار آنها تغییر میکند تا بتونیم الگوی declarative رو رعایت کنیم. سی شارپ امکانات کلیدی برای رعایت این الگوها داره از جمله تعریف فانکشن بدون اسم یا on the fly که بتونه مقادیر مورد نیاز رو بگیره(lambda expressions) و یا query expressions.
سی شارپ همینطور یه نوعی به اسم record رو معرفی کرده تا بتونیم نوع های immutable یا فقط خواندنی رو ایجاد کنیم.

در حاشیه کتاب

خداییش تو این اسلاید قبلی یه مقدار دچار سرگیجه شدم و برای رفعش یه سرچی کردم نتیجشو باهاتون به اشتراک میذارم، خوشحال میشم هرجاشو اشتباه رفتم اصلاحم کنید.

Pure Programming

در برنامه نویسی فانکشنال یک مفهوم مهم وجود داره به عنوان purity که بهش referential transparency هم میگند. اینقدر این مفهوم مهمه که میشه گفت همه هدف این الگوی برنامه نویسی همین purity هستش.

(این موضوع رو به عنوان یه بدهی برای خودم در نظر میگیرم که حتما بخونم در موردش و یک پست مجزا براش بنویسم)

Declarative Programming

این مفهوم در مقابل مفهوم imperative programming قرار میگیره و یکی دیگه از مفاهیم مهم در برنامه نویسی فانکشنال هستش.

(این هم بدهی بعدی که حتما در موردش مطالعه میکنم)

در حاشیه کتاب

lambda expressions

عبارات لامبدا همون توابع بی نام یا on the fly هستند که خیلی جاها بهمون کمک میکنه تا با کد کمتر به هدفمون برسیم.

query expressions

این عبارات هم مشابه عبارات لامبدا برای پیدا کردن یک مقدار از یک لیست بهمون کمک میکنه، مثل این:

```
IEnumerable<int> highScoresQuery =
    from score in scores
    where score > 80
    orderby score descending
    select score;
```

Type Safety

C# is primarily a type-safe language, meaning that instances of types can interact only through protocols they define, thereby ensuring each type's internal consistency. For instance, C# prevents you from interacting with a string type as though it were an integer type.

سی شارپ یک زبان type-safe هستش، یعنی نوع های داده ای فقط از طریق یک سری پروتکل تعریف شده میتونند باهم تعامل داشته باشند. (در کل یعنی نوع ها در سی شارپ ثبات دارند و به راحتی تغییر نمیکنند). مثلا در سی شارپ شما اگه بخواهید با یک نوع داده string شبیه به داده عددی رفتار کنید، جلوتون گرفته میشه(به خطا میخورید).

Type Safety
C# supports static typing

More specifically, C# supports static typing, meaning that the language enforces type safety at compile time. This is in addition to type safety being enforced at runtime.

سی شارپ نوع داده رو همون موقع کامپایل چک میکنه و اگه اشتباه تعریف شده باشه جلوتونو میگیره.

Static typing eliminates a large class of errors before a program is even run.

این ویژگی باعث میشه جلوی خیلی از خطاها قبل از اجرای برنامه گرفته بشه.

It shifts the burden away from runtime unit tests onto the compiler to verify that all the types in a program fit together correctly.

به عبارتی مسوولیت رو از گردن runtime برمیداره و میده به کامپایلر تا صحت استفاده از نوع ها رو چک کنه.

Type Safety
C# supports static typing

This makes large programs much easier to manage, more predictable, and more robust.

این باعث میشه مدیریت برنامه های بزرگ راحت تر و قابل پیش بینی تر بشه و برنامه مستحکم تر بشه.

Furthermore, static typing allows tools such as IntelliSense in Visual Studio to help you write a program because it knows for a given variable what type it is, and hence what methods you can call on that variable.

همچنین این ویژگی باعث میشه تا ابزارهایی به وجود بیاند که تو نوشتن کد بهمون کمک کنند مثل IntelliSense در ویژوال استودیو(مثلا یه متد مشخصه چه نوع متغیری میخواد پس ویژوال استودیو میتونه درست پیشنهاد بده).

Such tools can also identify everywhere in your program that a variable, type, or method is used, allowing for reliable refactoring.

همچنین ریفکتور هم راحت میشه(مثلا با تغییر نوع یک متغیر هرجا ازش استفاده کردیم خطا میده و باید اصلاح کنیم).

strongly typed

C# is also called a strongly typed language because its type rules are strictly enforced (whether statically or at runtime). For instance, you cannot call a function that's designed to accept an integer with a floating-point number, unless you first explicitly convert the floating-point number to an integer. This helps prevent mistakes.

سی شارپ رو به عنوان یک زبان strongly type هم میشناسند، چرا که قوانین نوع ها در آن اجرا شده است. چه به صورت ثابت(موقع کامپایل) و چه در موقع اجرا. برای مثال شما نمیتونید یک function که برای دریافت پارامتری از جنس عدد صحیح طارحی شده رو با پارامتر عدد اعشاری فراخوانی کنید. مگر اینکه اول عدد اعشاری رو به عدد صحیح تبدیل کنید. این جلوی بسیاری از اشتباهات رو میگیره.

Dynamic Types

C# also allows parts of your code to be dynamically typed via the dynamic keyword. However, C# remains a predominantly statically typed language.

البته که سی شارپ این اجازه رو به شما میده که بخشی از کدتون رو با نوع داینامیک یا غیر استاتیک باشه. این امکان با کلمه کلیدی dynamic امکان پذیر است. هرچند سی شارپ رو همچنان به عنوان یک زبان statically typed میشناسیم.

مدیریت حافظه
garbage collector

C# relies on the runtime to perform automatic memory management.

سی شارپ برای مدیریت خودکار حافظه به زمان اجرا یا runtime وابسته است.

The Common Language Runtime has a garbage collector that executes as part of your program, reclaiming memory for objects that are no longer referenced.

CLR یا زبان مشترک زمان اجرا(چه ترجمه بدی) یک زباله جمع کن (garbage collector) داره که به عنوان بخشی از برنامه شما اجرا میشه تا حافظه رو از آبجکتهایی که دیگه رفرنس ندارند پس بگیره.

This frees programmers from explicitly deallocating the memory for an object, eliminating the problem of incorrect pointers encountered in languages such as C++.

این ویژگی برنامه نویس ها رو از آزادسازی دستی حافظه نجات میده و مشکل پوینترهای اشتباه رو که در زبانهایی مثل c++ رخ میداد از بین میبره.

مدیریت حافظه
unsafe keyword

```
// compile with: -unsafe
class UnsafeTest
{
    // Unsafe method: takes pointer to int.
    unsafe static void SquarePtrParam(int* p)
    {
        *p *= *p;
    }

    unsafe static void Main()
    {
        int i = 5;
        // Unsafe method: uses address-of operator (&).
        SquarePtrParam(&i);
        Console.WriteLine(i);
    }
}
// Output: 25
```

C# does not eliminate pointers: it merely makes them unnecessary for most programming tasks.

سی شارپ پوینترها رو حذف نمیکنه، در اصل کاری میکنه که دیگه نشه از اونها توی بیشتر جاهای کد استفاده کرد.

For performance-critical hotspots and interoperability, pointers and explicit memory allocation is permitted in blocks that are marked unsafe.

میگه اگه لازم داشتید میتونید با کلمه کلیدی unsafe به اون جاهایی از حافظه که در حالت معمول نمیشه دسترسی داشته باشید، اینو ببینید:

پشتیبانی از پلتفرم ها

C# has runtimes that support the following platforms:

Windows 7+ Desktop (for rich-client, web, server, and command-line applications)

macOS (for web and command-line applications—and rich-client applications via Mac Catalyst)

Linux (for web and command-line applications)

Android and iOS (for mobile applications)

Windows 10 devices (Xbox, Surface Hub, and HoloLens) via UWP

There is also a technology called Blazor that can compile C# to web assembly that runs in a browser.

خلاصه اینکه رو اکثر پلتفرم های موجود قابل اجراست و مشکلی نداره باهاشون