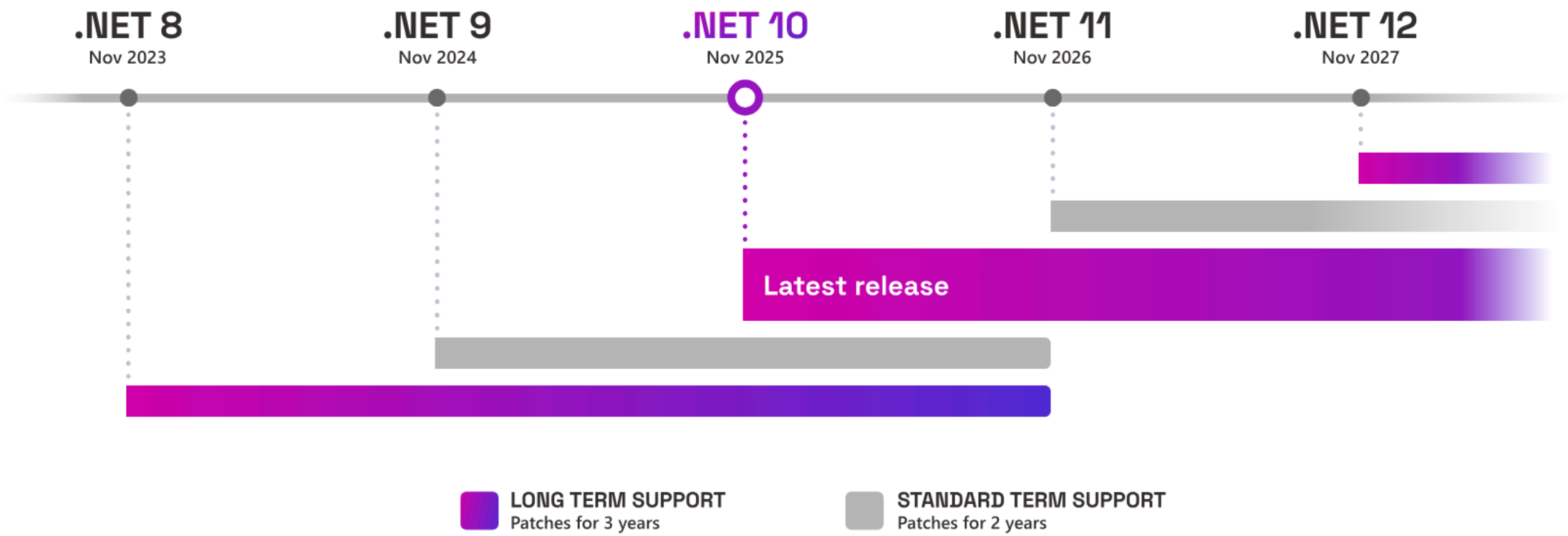
Robert Haken

Novinky .NET 10 a výhled na .NET 11

.NET Timeline

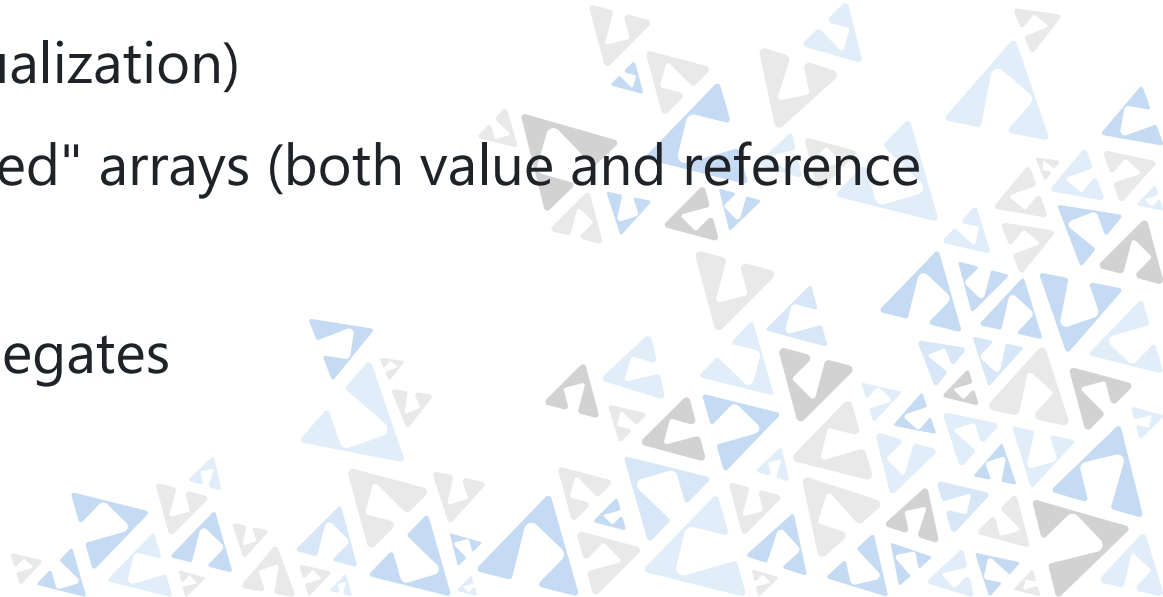


C# 14 in .NET 10 - DEMO

- ‼️ extension members
- Partial instance constructors, partial events
- ‼️ Null-conditional assignment using the `?.` and `?[]` operators
- ‼️ `field` access in auto-properties (C# 13 preview feature)
- ‼️ Unbound generic types in `nameof`
- Modifiers on simple lambda parameters - no longer require specifying the parameter type
- User-defined compound assignment operators like `+=` and `-=`
- User-defined increment (`++`) and decrement (`--`) operators

.NET 10 Runtime - JIT (Perf)

- Physical promotion improvements - struct members placed in registers rather than on stack
- Improved loop inversion (from lexical analysis to graph-based loop recognition)
- Array interface method devirtualization (eg. `foreach (var i in myArray)` transforms to `for` -loop over the array)
- Inlining improvements (`try` - `catch` , devirtualization)
- Stack allocation of small fixed-size "unescaped" arrays (both value and reference types)
- Escape analysis for local struct fields and delegates



.NET 10 Libraries

- Cryptography enhancements (e.g. PQC = Post-Quantum Cryptography)
- `ISOWeek` API extended to support `DateOnly` (dosud jen `DateTime`) - `GetWeekOfYear()`, `GetYear()`
- ‼️ Numeric ordering for string comparison (`"v10" > "v7"`, `"02" == "2"`, ...)

```
var numericStringComparer = StringComparer.Create(CultureInfo.CurrentCulture, CompareOptions.NumericOrdering);
```

- new `TimeSpan.FromMilliseconds` overload with single parameter (LINQ expressions cannot handle optional parameters)

```
public static TimeSpan FromMilliseconds(long milliseconds, long microseconds); // Second parameter is no longer optional
public static TimeSpan FromMilliseconds(long milliseconds); // New overload
```

- `StringNormalizationExtensions` - `ReadOnlySpan<char>` APIs (dosud jen `string`)
- `Convert.FromHexString()`, `Convert.TryToHexString()` - přetížení s `ReadOnlySpan<byte>` pro UTF-8

.NET 10 Libraries

- `OrderedDictionary<TKey, TValue>` - nová přetížení `TryAdd` a `TryGetKey` s indexy (`out int index`)
- JSON: `JsonSourceGenerationOptions.ReferenceHandler` - řešení pro cycles při použití source-gen
- JSON: duplicit. properties - `bool JsonSerializerOptions.AllowDuplicateProperties`
- ‼️ JSON: `JsonSerializerOptions.Strict` preset (static property)
- JSON: podpora pro `Pipe.Reader` pro deserializaci (I/O alternativa k `Stream`), `Pipe.Writer` už podporováno
- ‼️ ZIP perf + async APIs - `ZipArchive.CreateAsync()` , `ZipFile.XyAsync()`
- `bool ProcessStartInfo.CreateNewProcessGroup()`
- new `WebSocketStream` abstraction over `WebSocket`

.NET 10 Libraries

- ‼️ LINQ: `LeftJoin()` + `RightJoin()`
- LINQ: `Shuffle()`
- `bool IpAddress.IsValid(addr)` instead of `bool IpAddress.TryParse(addr, out _)`
- `Random.Get[Hex]String()` methods

```
string GetString(ReadOnlySpan<char> choices, int length);
string GetHexString(int stringLength, bool lowercase = false);
void GetHexString(Span<char> destination, bool lowercase = false);
```

- `GCHandle<T>` - *like* `GCHandle`, *but this time it's great™*

.NET 10 SDK

- Tools: `dotnet tool exec` - downloads and runs the specified tool package
  - `dotnet tool exec dotnetsay "Hello, World!"`
- Tools: `dnx` - tool execution script (currently calls `dotnet` CLI)
  - `dnx dotnetsay "Hello, World!"`
- File-based apps `dotnet [run] singlefile.cs`
  - `#:skd`, `#:package`, `#:property`, ...
  - referencing `.csproj` project `#:project {path}`
  - publishing to executable (`dotnet publish singlefile.cs`)
  - runtime path data `System.AppContext.GetData`
  - shell-execution support (`#!/usr/bin/env dotnet` shebang for unix shells)

.NET 10 SDK

- .NET tasks usable within .NET Framework MSBuild (VS)
- NuGet: Pruning of framework-provided package references (enabled by default)
- CLI: noun-first form aliases e.g. `dotnet package add` = `dotnet add package` (verb-first form)

ASP.NET Core 10

- OpenAPI 3.1 + YAML
- `bool RedirectHttpResult.IsLocalUrl(url)`
- MinimalAPIs Input Validation support
  - `services.AddValidation()`
  - `endpoint.DisableValidation()` - enabled by default
- Treating empty string in `[FromForm]` as null for nullable value types
- WebAuthN + Passkey authentication support
- 401/403 instead of cookie redirects for unauth API requests (`[ApiController]` , ...)
- return SSE via `TypedResults.ServerSentEvents(IAsyncEnumerable<T>)`
- HybridCache - feature complete (abstraction only in .NET 9)
- memory management improvements (eviction, pooling, ...)

Blazor in .NET 10

- new `<InputHidden />` component
- State persistence for Blazor Server (was `PersistentComponentState` service)
  - `[PersistentState] public int MyProperty { get; set; }`
  - `builder.Services.RegisterPersistentService<MyService>(..)`
  - `IPersistentComponentStateSerializer` customization (default JSON)
- Circuit state persistence (uses `MemoryCache` or `HybridCache`)
- 💕 Blazor WebAssembly performance profiling & memory diagnostics 🥷
- ‼️ `NavigateTo()` no longer scrolls to the top for same-page navigations

Blazor in .NET 10

- ‼️ Support for HTTP Not Found responses
  - `<Router NotFoundPage="typeof(MyPage)" />`
  - `NavigationManager.NotFound(args)` method
  - `NavigationManager.OnNotFound` event
- ‼️ Preload Blazor WebAssembly resources - `<ResourcePreloader />`
- ‼️ Blazor script as static web asset (compression, fingerprinting)
  - `<script src="@Assets["_framework/blazor.web.js"]"></script>`
- Fingerprinting support for standalone Blazor WebAssembly apps
- `OwningComponentBase` now implements `IAsyncDisposable`

Blazor in .NET 10 - Security

- Blazor Web App security samples (OIDC, Entra ID, Windows Authentication) incl. API project
- Entra auth scaffolding
- Auth cookie expiration in Blazor Server
- Web Authentication API (passkey) support for ASP.NET Core Identity

Blazor in .NET 10

- ‼️ `<DataAnnotationsValidator />` - nested objects, collections
  - `services.AddValidation();` - per assembly (source-generation)
  - `[ValidatableType]` to top-level model type
- new `<QuickGrid RowClass="GetRowCssClass" />` parameter (selector from `TItem`)
- ‼️ `ReconnectModal` - reconnection UI component in BWA project template (CSP compliant)
- `NavLinkMatch.All` now ignores query string and fragment (configurable)
- `HttpClient` response streaming enabled by default (WASM, configurable)
- new `IJSRuntime.InvokeConstructorAsync()` and `Get/SetValueAsync<TValue>()`

Blazor in .NET 10

- `<WasmApplicationEnvironmentName>` property in `.csproj` (instead of `launchSettings.json`)
- `blazor.boot.json` inlined into `dotnet.js`
- `<BlazorDisableThrowNavigationException>` for Static-SSR (where `NavigationException` is thrown)
- Blazor WASM respects `DefaultThreadCurrentUICulture`
- Blazor metrics, Distributed tracing for Blazor Server

NOT in Blazor 10

- Multithreading
- Full trimming
- NativeAOT for server
- PWA tooling
- Nested routing
- Avoid completely replacing the DOM after prerendering
- Ability to run multiple Blazor apps in the same document
- State transitions for animations
- Hashed routing

.NET 11

C# 15 in .NET 11 🤞 - Type Unions 🥳🍾

```
public record class Dog(string Name);
public record struct Cat(string Adjective);
public record class Bird(string Species);

public union Pet (Dog, Cat, Bird, Shark);  // Nominal Type Union

Pet pet = new Dog("Rover");  // implicit conversion

var description = pet switch // exhaustive pattern matching
{
        Dog(var name) => name,
        Cat(var adjective) => $"A {adjective} cat",
        Bird bird => $"A {bird.Species}"
        Shark => $"Sharkie"
        // No warning about missing cases (fallback case not needed)
};

// ad-hoc unions - similar to tuple types (implemented as Union<T1, T2, T3, ...>)
(Dog or Cat or Bird) pet = new Dog("Alík");
global using Pet = (Dog or Cat or Bird); // global naming
```

C# 15 in .NET 11 🤞 - Unions

under the hood

```
public partial record struct Pet : IUnion
{
        public object? Value { get; }

        public Pet(Dog value) => Value = value;
        public Pet(Cat value) => Value = value;
        public Pet(Bird value) => Value = value;
}

public interface IUnion
{
    object? Value { get; }
}
```

C# 15 in .NET 11

- Dictionary expressions

```
Dictionary<string, int> currentStudents = ["joe" : 1, "jane" : 2, "john" : 3];
Dictionary<string, int> newStudents = [.. currentStudents, "mds" : 5];
```

- Optional and named arguments support in Expression trees
- extension indexers
- collection expr arguments `List<string> l = [args(capacity: 32); a, b, c];`
- target-typed interference enhancements
- `yield` inside `try` / `catch`
- allow `using _ = GetDisposable()` with discard

Blazor in .NET 11

- new "Media Component Suite" (from `Stream` or `byte[]`)
  - `<Image ImageSource="..." />`
  - `<Video />`
  - `<FileDownload />`
- new `<EnvironmentBoundary />` component
- `[SupplyParameterFromQuery]` fixes
- `NavigationManager.GetUriWithHash()`
- `InputFile` - correct `Dispose` pattern implementation

Links

- https://github.com/hakenr/CSharp14Demo
- What's new in .NET 10 | Microsoft Learn
- What's new in C# 14 | Microsoft Learn