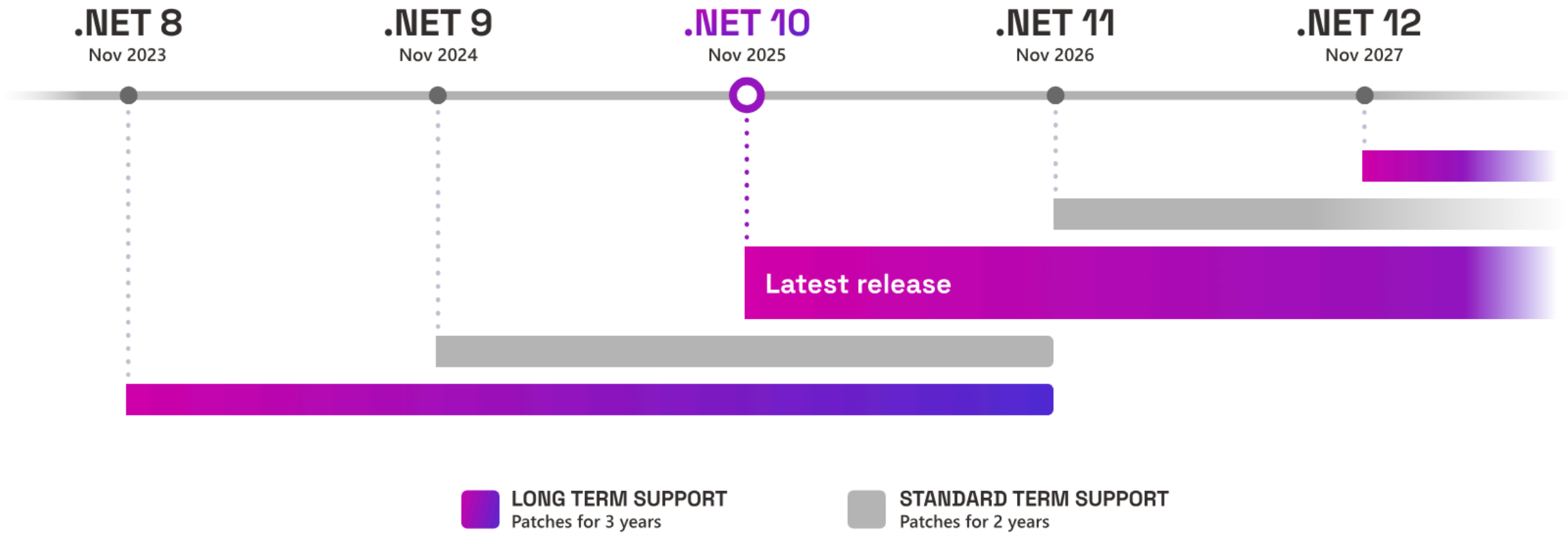
HAVIT
SOFTWARE DEVELOPMENT

Robert Haken

Novinky .NET 11 a C# 15

.NET Timeline



C# 14 in .NET 10

- ‼️ extension members
- ‼️ Null-conditional assignment using the `?.` and `?[]` operators
- ‼️ `field` access in auto-properties
- Partial instance constructors, partial events
- Unbound generic types in `nameof`
- User-defined compound assignment operators like `+=` and `-=`
- User-defined increment (`++`) and decrement (`--`) operators

C# 15 in .NET 11 - Union Types ‼️🥳🍾

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

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

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

var description = pet switch // exhaustive pattern matching
{
        Dog d => d.Name,
        Cat c => $"A {c.Adjective} cat",
        Bird b => $"A {b.Species}"
        // fallback case not needed
};
```

C# in .NET 11

DEMO

DEMO

DEMO

DEMO

C# 15 in .NET 11 - Union Types

- ad-hoc unions not implemented 😭, hopefully C# 16 in .NET 12

```
(Dog or Cat or Bird) pet = new Dog("Tobby");
```

- under the hood

```
[Union]
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;
}
```

C# 15 in .NET 11

- ‼️ `closed` class modifier - prevents deriving in other assemblies, allows exhaustive `switch`
- collection expr arguments `List<string> l = [with(capacity: 32), a, b, c];`
- extension indexers
- ‼️ labeled `break` and `continue`
- memory safety improvements (safe `Pointer` operations no longer require `unsafe` context)

C# 16 in .NET 12 🤞

- Dictionary expressions

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

- closed `enum`

.NET 11 - Runtime

- runtime-native async (instead of compiler-generated state machines)
  - replacing compiler-generated async state machines with runtime-managed suspension and resumption
  - cleaner live stack traces
  - perf
- JIT perf improvement
  - Bounds check elimination
  - Constant-folding `string.Equals()` and `ReadOnlySpan<T>.SequenceEqual()`
- .NET WebAssembly on CoreCLR interpreter (replacing Mono) - preview?
- support for more than 1024 CPUs

.NET 11 - Libraries - Process

- new `Process` APIs and helpers
  - `Process.Run(..)` + `Async`
  - ‼ `Process.RunAndCaptureText(..)` + `Async`
  - `Process.StartAndForget()`
  - `Process.ReadAllLines(..)` + `Async`
  - ...and many more

```
ProcessTextOutput result = await Process.RunAndCaptureTextAsync("git", ["status", "--porcelain"]);

Console.WriteLine(result.StandardOutput);
Console.WriteLine($"Exit code: {result.ExitStatus.ExitCode}");
```

.NET 11 - Libraries - In-memory stream adapters

- `new ReadOnlyMemoryStream(ReadOnlyMemory<byte> ...)`
- `new WritableMemoryStream(Memory<byte> ...)`
- `new ReadOnlySequenceStream(ReadOnlySequence<byte> ...)` e.g. buffers from `System.IO.Pipelines`
- !! `new StringStream(string s, Encoding e)`, `new StringStream(ReadOnlyMemory<char> s, Encoding e)`

```
// Pass a string to an API that takes a Stream, with no intermediate byte[] needed.
using Stream configStream = new StringStream(yamlText, Encoding.UTF8);
var settings = ParseConfiguration(configStream);

// Expose an existing buffer as a read-only stream, for example as an HTTP request body.
ReadOnlyMemory<byte> payload = GetPayload();
using Stream bodyStream = new ReadOnlyMemoryStream(payload);
await httpClient.PostAsync(uri, new StreamContent(bodyStream));
```

.NET 11 - String and character enhancements

- !! `Rune` support in `String` methods - `.Contains(Rune)`, `Starts+EndsWith`, `[Last]IndexOf`, `Replace`, `Split`, `Trim[Start|End]`, ...

```
string text = "A😀";
Console.WriteLine(text.Length); // 3
foreach (Rune rune in text.EnumerateRunes())
{
    Console.WriteLine(rune); // A  + 😀
}
```

- `Utf8.IndexOfInvalidSubsequence(bytes)`, `Utf16.IsValid(chars)`
- `RegexOptions.AnyNewLine` flag - all line endings recognized, not just `\n`
- `Char.Equals(Char, StringComparison)` - new `StringComparison` parameter
- !! `Base64` static helpers class - new `EncodeToChars`, `EncodeToString` and `DecodeFromChars` methods (`EncodeToUtf8` and `DecodeFromUtf8` existed)

.NET 11 - Libraries

- `System.Text.Json` improvements
  - new `JsonNamingPolicy.PascalCase` policy (camelCase, snake_case, and kebab-case already exist)
  - per-member naming policy with new `[JsonNamingPolicy(JsonNamingPolicy.Xy)]` attribute
  - F# discriminated unions support, C# union type serialization
- Compression & archives improvements
  - ZIP password support + CRC32 checksum validation
  - Zstandard compression (new, `ZstandardStream`, `ZstandardEncoder`)
  - TAR and GNU formats support
- OpenTelemetry metrics for `new MemoryCache(new MemoryCacheOptions() { TrackStatistics = true })`

.NET 11 - Libraries - Collections

- !! `Join`, `LeftJoin`, `RightJoin` - new tupple-returning overloads (when result-selector parameter omitted)
  - and `GroupJoin` - returns `IGrouping` when result-selector parameter omitted
  - new `FullJoin` method (both with result-selector and tuple-returning)

```
var leftJoined = products.LeftJoin(categories, p => p.CategoryId, c => c.Id);
foreach (var (product, category) in leftJoined)
    ...
```

- !! new `EqualityComparer<T>.Create()` factory method (with key selector)

```
var byName = EqualityComparer<(string Name, int Age)>.Create(p => p.Name);
var people = new HashSet<(string Name, int Age)>(byName); // will not allow duplicate names
```

.NET 11 - Libraries - Numerics

- new IEEE 754-2019 types `Decimal32`, `Decimal64` and `Decimal128`
- `.TryParsePartial(...)` for parsing delimited formats (CSV), reports consumed input
- `Complex<T>` new generic type
- floating-point hex formatting and parsing

```
// Format as hexadecimal IEEE-754: preserves all bits exactly
string hex = Math.PI.ToString("X"); // 0X1.921FB54442D18P+1
double parsed = double.Parse(hex, NumberStyles.HexFloat);
```

- !! `Random.NextInteger<T>()` and `.NextBinaryFloat<T>()` new generic methods

```
byte b = Random.Shared.NextInteger<byte>(maxValue: 10);
```

.NET 11 - Libraries

- ‼️ configuration binding opt-out with `[ConfigurationIgnore]` attribute

```
public sealed class AppOptions
{
    public string Endpoint { get; set; } = "";

    [ConfigurationIgnore]
    public string ComputedKey => Endpoint + ":default";
}
```

- options builder validation with `IValidateOptions<TOptions>` (pre: only a `Func`), DI support

```
services.AddSingleton<IValidateOptions<MyOptions>, MyOptionsValidator>();
services.AddOptions<MyOptions>()
    .Bind(configuration.GetSection("MyOptions"))
    .Validate<MyOptionsValidator>();
```

.NET 11 - Libraries - Asynchronous validations ‼

- use `Validator.ValidateObjectAsync`, `TryValidateObjectAsync`, `ValidatePropertyAsync` or `ValidateValueAsync`
- derive from `AsyncValidationAttribute` and override `IsValidAsync` method

```
public sealed class ValidVatNumberAttribute : AsyncValidationAttribute
{
    protected override ValidationResult? IsValid(object? value, ValidationContext context)
            => throw ...

    protected override async Task<ValidationResult?> IsValidAsync(
        object? value, ValidationContext context, CancellationToken cancellationToken)
    {
        var registry = context.GetRequiredService<IVatRegistry>();
        return await registry.IsRegisteredAsync((string?)value, cancellationToken)
            ? ValidationResult.Success
            : new ValidationResult("That VAT number isn't registered.");
    }
}
```

.NET 11 - Libraries - Asynchronous validations ‼️

- or implement `IAsyncValidatableObject` on the model

```
public class User : IAsyncValidatableObject
{
        public string Email { get; set; }

        public async IAsyncEnumerable<ValidationResult> ValidateAsync(
                ValidationContext validationContext,
                [EnumeratorCancellation] CancellationToken cancellationToken = default)
        {
                var userService = validationContext.GetService<IUserService>();
                if (await userService.IsEmailExistsAsync(Email, cancellationToken))
                {
                        yield return new ValidationResult(
                                "Email is already registered.", new[] { nameof(Email) });
                }
        }

        public IEnumerable<ValidationResult> Validate(ValidationContext validationContext)
                => throw new InvalidOperationException("Synchronous validation isn't supported.");
}
```

.NET 11 - Libraries - Networking

- HTTP request compression - `GZipCompressedContent` and other `XyCompressedContent` wrappers for request bodies (sets `Content-Encoding` and streams compressed content as the request serializes)
- `Dns` adds typed record-resolution APIs
  - `Dns.ResolveCName[Async](String)`
  - `Dns.ResolveMx[Async](String)`
  - `Srv`, `Txt`, `Ptr`, `Ns`, …
- new `MediaTypeNames.Video` class with constants for common video media types
  - `MediaTypeNames.Video.Mp4` — `"video/mp4"`
  - `MediaTypeNames.Video.Mpeg` — `"video/mpeg"`
  - `MediaTypeNames.Video.Ogg`, `QuickTime`, `WebM`

.NET 11 - SDK

- `#:include` for single-file apps (splitting across multiple files 🤔), supports DLLs
- ‼️ `dotnet run -e` to pass environment variables from command line

```
dotnet run -e ASPNETCORE_ENVIRONMENT=Development -e LOG_LEVEL=Debug
```

- `dotnet reference add` fallbacks to current directory

```
cd ClassLib2
dotnet reference add ../ClassLib1/ClassLib1.csproj   # now works without --project
dotnet reference remove ../ClassLib1/ClassLib1.csproj
```

- TypeScript outputs (Razor Class Libraries) properly integrate with Static Web Assets

ASP.NET Core Blazor in .NET 11

- new `<DisplayName />` component (support for `[Display]` and `[DisplayName]` attributes)

```
<label>
    <DisplayName For="@(() => Model!.ProductionDate)" />
    <InputDate @bind-Value="Model!.ProductionDate" />
</label>
```

- ‼️ asynchronous validation support
  - `EditForm` submit validation awaits async validations
  - `EditContext.RegisterAsyncFieldValidator()`
  - `EditContext.IsValidationPending(field)` + `EditContext.IsValidationFaulted(field)`

ASP.NET Core Blazor in .NET 11 - Static SSR

- `TempData` data persistence between HTTP requests during static server-side rendering (static SSR)

```
[CascadingParameter] public ITempData? TempData { get; set; }
[SupplyParameterFromTempData] public string? Message { get; set; }
```

- Session data persistence (cookie-based) for static SSR
  - `builder.Services.AddSession()`
  - `app.UseSession()`
  - `[SupplyParameterFromSession]` attribute (with optional `Name` property)

```
[SupplyParameterFromSession] public string Message { get; set; }
[SupplyParameterFromSession(Name = "flash_message")] public string FlashMessage { get; set; }
```

- new `<CacheView />` component for static-SSR (with `ExpiresAfter` and `VaryByXy` parameters)

ASP.NET Core Blazor in .NET 11

- new `<BasePath />` component, replaces `<base href="/">`
- relative navigation support

```
Navigation.NavigateTo("/configuration", new NavigationOptions { RelativeToCurrentUri = true });
<NavLink href="configuration" RelativeToCurrentUri="true">Configuration</NavLink>
```

- `NavigationManager.GetUriWithFragment()` - new extension method
- new `<EnvironmentView />` component (with `Include` and `Exclude` parameters)

ASP.NET Core Blazor in .NET 11

- ‼️ Virtualization enhancements - `<Virtualize />`
  - no longer requires same height items
  - `InitialItemIndex` parameter and `ScrollToIndexAsync(...)` method
  - `OverscanCount` default changes to 15
  - new `AnchorMode` parameter - `None`. `Start` (default), `End` (behavior when items being added)
  - CSP compliance
- `QuickGrid` enhancements
  - URL-based navigation (page, sort, order) support (enables static SSR support)
  - `OnRowClick` event

ASP.NET Core Blazor in .NET 11

- `IHostedService` support in Blazor WebAssembly for running background services in the browser
- WebAssembly component metrics and tracing
- configure Blazor client behavior from server `.WithBrowserOptions()` + `<ConfigureBrowser />` component

```
app.MapRazorComponents<App>()
    .AddInteractiveServerRenderMode()
    .WithBrowserOptions(options =>
    {
        options.LogLevel = LogLevel.Warning;
        options.Server.ReconnectionMaxRetries = 10;
        options.Server.ReconnectionRetryInterval = TimeSpan.FromSeconds(1.5);
        options.Ssr.PreserveDom = true;
        options.WebAssembly.EnvironmentName = "Staging";
        options.WebAssembly.EnvironmentVariables["OTEL_EXPORTER_OTLP_ENDPOINT"] =
            "https://localhost:4318";
    });
```

ASP.NET Core Blazor in .NET 11

- ‼️ new "Media Component Suite" (from `Stream` or `byte[]`)
  - `<Image ImageSource="..." />`
  - `<Video />`
  - `<FileDownload />`
  - separate package `Microsoft.AspNetCore.Components.Media`
- ‼️ `Microsoft.AspNetCore.Components.AI` component library (WIP 🪠)
  - https://github.com/dotnet/aspnetcore/issues/66178

ASP.NET Core 11 - Minimal APIs

- Minimal APIs
  - C# union types support
  - asynchronous validation support
- short-circuit endpoints with a `[ShortCircuit]` attribute (also applicable to controller/action)
- OpenAPI 3.2 - incl. SSE, QUERY verb, binary content description etc.

EF Core 11

- Complex types improvements
- `FullJoin()` support
- `MaxBy` and `MinBy` support
- `EF.Functions.JsonPathExists()`, `EF.Functions.JsonContains()`, JSON indexes
- `VectorSearch()` extension on `DbSet`, `WithApproximate()`, …
- full-text catalogue support
  - annotations `.HasFullTextCatalog()`, `.HasFullTextIndex()`, `.UseCatalog()`, …
  - queries - `.FreeTextTable()`, `.ContainsTable().`
- migrations - latest migration ID included in model snapshot (implies conflicts when merging)

Links

- https://github.com/hakenr/CSharp15Demo
- What's new in .NET 11 | Microsoft Learn
- HAVIT - YouTube
- HAVIT Knowledge Base