C# 6-News

- See template solution in csharp/templates folder as a basis
- Story: Theming of an application, theme is specified in app.config, problem with casing
- Hint: Alternative impementation: String.Compare("AA", "aa", StringComparison.OrdinalIgnoreCase);

Null-Conditional Operator

```
var themeConfig = ConfigurationManager.AppSettings["Theme"];
var theme = themes.FirstOrDefault(p => p.Name.ToLower() == themeConfig.ToLower());

Console.ForegroundColor = theme.ForegroundColor;
Console.BackgroundColor = theme.BackgroundColor;
```

- Problem: NullReferenceException (no Key in app.config)
- still NullReferenceException (one Theme without a name)

```
var theme = themes.FirstOrDefault(p => p.Name?.ToLower() == themeConfig?.ToLower());
```

- In combination with Coalesce operator (uncomment app.config):

```
var themeConfig = ConfigurationManager.AppSettings["Theme"]?.ToLower() ?? "light";
```

Chaining of expressions

- the first NULL occuring leads to NULL
- value types are transformed to Nullable

```
themes = null;
var theme = themes?.FirstOrDefault(p => p.Name?.ToLower() == themeConfig)?.BackgroundColor;
```

? with Indexoperator

```
var name = themes?[0].Name;
// equivalent to    (themes != null) ? themes[0] : null;
```

Invoking delegates

- Change Theme.Name-Property to propfull

- implement INotifyPropertyChanged

```
PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("Name"));
```

nameof-Operator

- show different parameters you can pass to nameof
  - Main, Program, Theme, Variable

```
PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Name)));
```

Dictionary Initializers

```
var settings = new Dictionary<string, string>()
            {
                ["Theme"] = themeConfig,
                ["User"] = "Roman"
            };
```

```
var json = new JObject
{
    ["x"] = 12,
    ["y"] = 27,
    ["CreationDate"] = DateTime.UtcNow,
    ["User"] = "Roman"
};

Console.WriteLine(json);
```

Auto-Property Initializers

- what was necessary to implement an immutable property?
- Theme: change Color-Properties to...

```
public ConsoleColor BackgroundColor { get; }

public void Foo()
{
    this.BackgroundColor = ConsoleColor.Red;    // compiler error, write access is only allc
}
```

```
public ConsoleColor BackgroundColor { get; } = ConsoleColor.Black;
```

Expression Bodied Functions and Properties

```
// Properties:
public string FullName => string.Format("Theme: {0}, Background: {1}", this.name, this.Backg

// Methods:
public override string ToString() => this.FullName;
```

Static Using Statements

```
// Program.cs:
using static System.Console;
```

- change all WriteLine to WriteLine(..)

Exception-Handling

```
// Add validation to Theme constructor (winter theme throws exception now)
if (backgroundColor == foregroundColor)
{
    throw new TypeInitializationException(nameof(Theme), new ArgumentException("Background c
}

// try/catch for InitializeThemes
catch (Exception e)
{
    LogException(e); // Strg + ., generate...
}

// Assumption: LogException-Methode is async -> async/await wasn't available in catch/finall
private static Task LogException(Exception e) => Task.Delay(1000);

catch (Exception e)
{
    await LogException(e); // Strg + ., generate...
//  -----    -> themes-Collection gets null now, code might not work anymore.
}
```

Exception Filters

```
catch (TypeInitializationException e) when (e.InnerException is ArgumentException)
{
    await LogException(e);
}
```

String Interpolation

- delete Winter-Theme
- correct build errors

```
public string FullName => $"Theme: {this.Name}, Background: {this.BackgroundColor}";
public string FullName => $"Theme: {this.Name,20}, Background: {this.BackgroundColor}";
```

C# 7

- What's new in C# 7
- Language feature status
- Slide deck: C# 7 features