

# Title slide

Date, Author

Tell what the presentation is about



A screenshot of a GitHub code editor window. The file being viewed is `ResultMethodRewriter.cs`. The code is a C# class with several methods for rewriting syntax nodes. A hand is pointing at the code editor, specifically at the line where a parameter list is being modified. The GitHub interface includes a sidebar with navigation links like 'OUTLINE' and 'TIMELINE', and a status bar at the bottom.

```
var modifiedNode = node.RemoveNodes(membersToRemove, SyntaxRemoveOptions.KeepNoTrivia)
    ?.WithIdentifier(Identifier.Create("Result"));

return base.VisitClassDeclaration(modifiedNode)!.NormalizeWhitespace();
}

public override SyntaxNode VisitParameterList(ParameterListSyntax node)
{
    _resultType = node.Parameters[0].Type;

    var newNode = ParameterList(node.Parameters.RemoveAt(0));
    _parameters = newNode;

    return node.ReplaceNode(node, newNode);
}

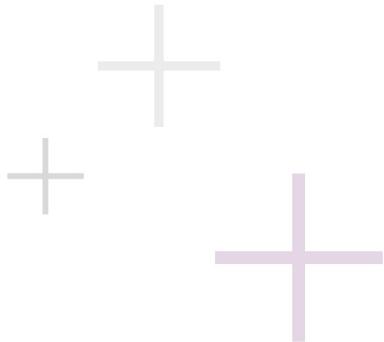
public override SyntaxNode? VisitMethodDeclaration(MethodDeclarationSyntax node)
{
    VisitParameterList(node.ParameterList);

    if (node.Body is not null)
    {
        // remove documentation comments
        var commentNode = node.Body.GetLeadingTrivia().RemoveComments();
        if (commentNode != default)
        {
            var index = node.GetLeadingTrivia().IndexOf(commentNode) - 1;
            node = node.WithLeadingTrivia(node.GetLeadingTrivia().RemoveAt(index).Remove());
        }
    }

    // new Passed() or new Passed< TValue >()
    var objectCreationExpression = ObjectCreationExpression(ParseTypeName(_resultType) is GenericTypeSyntax
        ? "Passed< TVal
          , ArgumentList(),
          null).NormalizeWhitespace();

    SimpleNameSyntax methodNameSyntax = node.TypeParameterList is null
        ? IdentifierName(node.Identifier)
        : GenericName(node.Identifier);
}
```





**01** | Introduction

**03** | Why Zühlke?

**05** | Our vision

**07** | Summary

**02** | About us

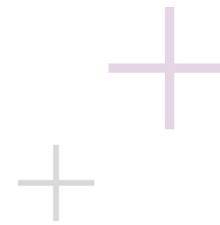
**04** | Your goals

**06** | Further challenges

**08** | Q&A



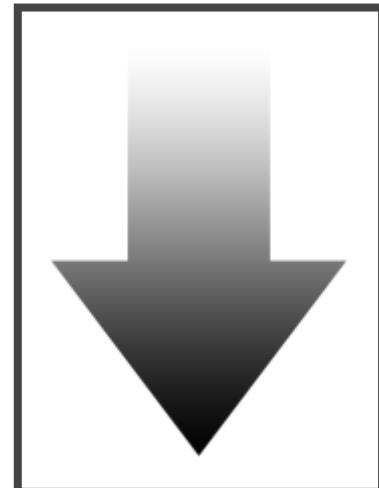
## Subtitle slide



# Vertical Slides

Slides can be nested inside of each other.

Use the *Space* key to navigate through all slides.

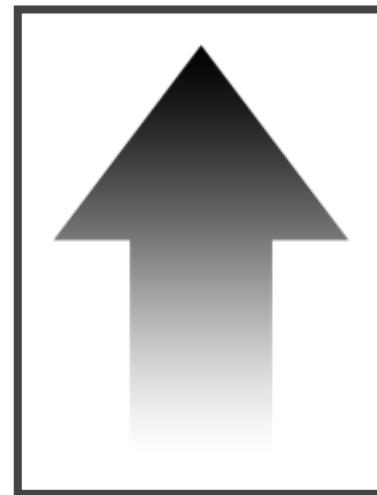


# **Basement Level 1**

Nested slides are useful for adding additional detail underneath a high level horizontal slide.

# Basement Level 2

That's it, time to go back up.



# Pretty Code

```
1 import React, { useState } from 'react';
2
3 function Example() {
4   const [count, setCount] = useState(0);
5
6   return (
7     ...
8   );
9 }
```

Code syntax highlighting courtesy of [highlight.js](#).

# With animations

```
1 import React, { useState } from "react";
2
3 function Example() {
4   const [count, setCount] = useState(0);
5
6   return (
7     <div>
8       <p>You clicked {count} times</p>
9       <button onClick={() => setCount(count + 1)}>Click me</button>
10    </div>
11  );
12}
```

# Fragments

Hit the next arrow...

... to step through ...

... a fragmented slide.

# Intergalactic Interconnections

You can link between slides internally, like this.

# **Point of View**

Press **ESC** to enter the slide overview.

# Speaker View

There's a speaker view. It includes a timer, preview of the upcoming slide as well as your speaker notes.

Press the S key to try it out.

# Export to PDF

Presentations can be exported to PDF, here's an example:

