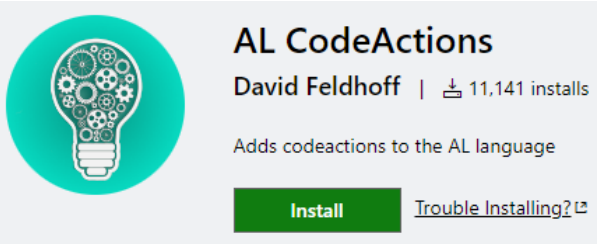
David Feldhoff

How to build VS Code Extensions

About me



- David Feldhoff
- 26 years old
- Working with NAV/BC since 2014 at GWS mbH
- Developer of the „AL CodeActions“ VS Code Extension

Agenda

- Preparations
- Create Hello World sample
- Possibilities in VSCode
  - Add new functionality
  - Interact with Built-In Commands
  - (Interact with other extensions)
- Share/Publish your extension

Preparations

- Brand new Windows 10 Virtual Machine
- Followed Guide of VS Code: https://code.visualstudio.com/api/get-started/your-first-extension
  - Installed Visual Studio Code
  - Installed Node.js
  - Installed Yeoman

Create Hello World sample

Understanding the Hello World sample

- Two important files:
  - Package.json
  - Extension.ts

- Three important things:
  - ActivationEvents
    - Specifies when your extension is activated. (package.json)
  - ContributionPoints
    - Specifies what your extension contributes to VS Code. (package.json)
  - Register your functions
    - If you add a command in the previous step, then you have to register a function to that command. (extension.ts)

Activation Events

- Package.json → activationEvents
- These are typically:
  - onCommand
  - onLanguage
  - workspaceContains
  - Full list: https://code.visualstudio.com/api/references/activation-events

Contribution Points

- Configurations
- Commands
- Menus
- Keybindings
- Snippets
- Full list: https://code.visualstudio.com/api/references/contribution-points

Register commands

- Syntax:

```
vscode.commands.registerCommand('areopa.helloWorld', () => HelloWorld.sayHi())
```

- But: Not all contributions have to be specified as contribution point
  - vscode.languages.registerHoverProvider
  - vscode.languages.registerDefinitionProvider
  - vscode.languages.registerCodeActionProvider
  - ...

Possibilities in VS Code

Add „Extract label“ CodeAction

- Possbilities: https://code.visualstudio.com/api/references/vscode-api
- New to typescript? No worries, time for a demo

Interacting with Built-in Commands

- built-in commands are listed here: https://code.visualstudio.com/api/references/commands
- Syntax:

```
let locations: vscode.Location[] | undefined =
await vscode.commands.executeCommand('vscode.executeDefinitionProvider', uri, position);
```

Recap

- Objects:
  - `vscode.TextDocument` (represents a vscode-File, contains text and Uri)
  - `vscode.Uri` (contains fsPath)
  - `vscode.Location` (contains Uri and Range)
  - `vscode.Range` (consists of a start and end position)
  - `vscode.Position` (line: number, character: number)
- Variable Declaration & execution of Built-In Commands:

```
let locations: vscode.Location[] | undefined =
await vscode.commands.executeCommand('vscode.executeDefinitionProvider', uri, position);
```

- Function returns Promise<> or Thenable<>? Use `await`

Share or publish your extension

Publish extensions

- https://code.visualstudio.com/api/working-with-extensions/publishing-extension
- Create vsix: vsce package
- Upload to marketplace: vsce publish major/minor/patch