

Current Developments in Tooling and Editor Support in textX

Milan Šović, Daniel Elero & Igor Dejanović
University of Novi Sad

The textX logo features the word "text" in a black, lowercase, sans-serif font, followed by a large, stylized green "X". The background of the slide includes a night cityscape with a prominent church tower, a rocket launch with a searchlight, and a starburst graphic.The LangDev CON2024 logo consists of a stylized orange and red graphic element resembling a letter 'A' or a flame, followed by the text "LangDev" in a bold, white, sans-serif font, and "CON2024" in a smaller, white, sans-serif font below it.

LangDev
CON2024

Seville 17-19 October, 2024

<https://langdevcon.org>

Agenda

- textX
- Web Playground
- VS Code Extension
- Drone Example Demo
- Summary
- Q&A

JGBX



textX

Overview, Basic Concepts & textX CLI



textX

The logo for textX, featuring the word "text" in black and "X" in green.

- Meta-Language for DSL specification in Python
- Inspired by Xtext
- Project started by Igor Dejanović in 2014
- 24 contributors
- Current version – 4.0.1

GitHub: <https://github.com/textX/textX>

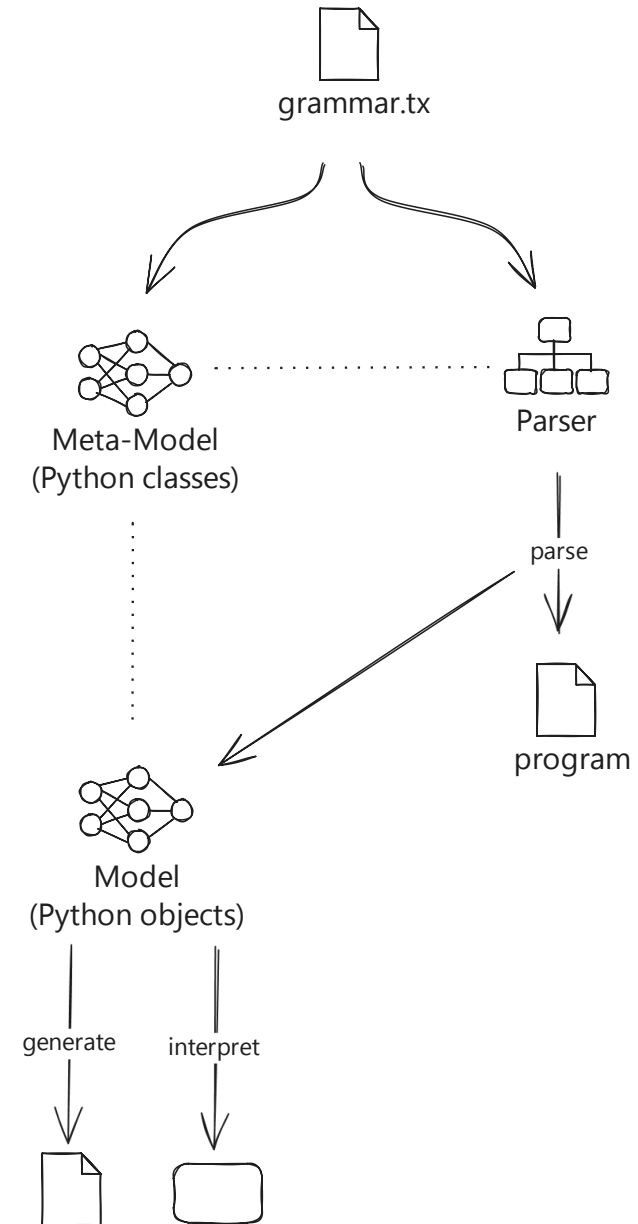
PyPi: <https://pypi.org/project/textX/>

Docs: <https://textx.github.io/textX>

textX – Overview

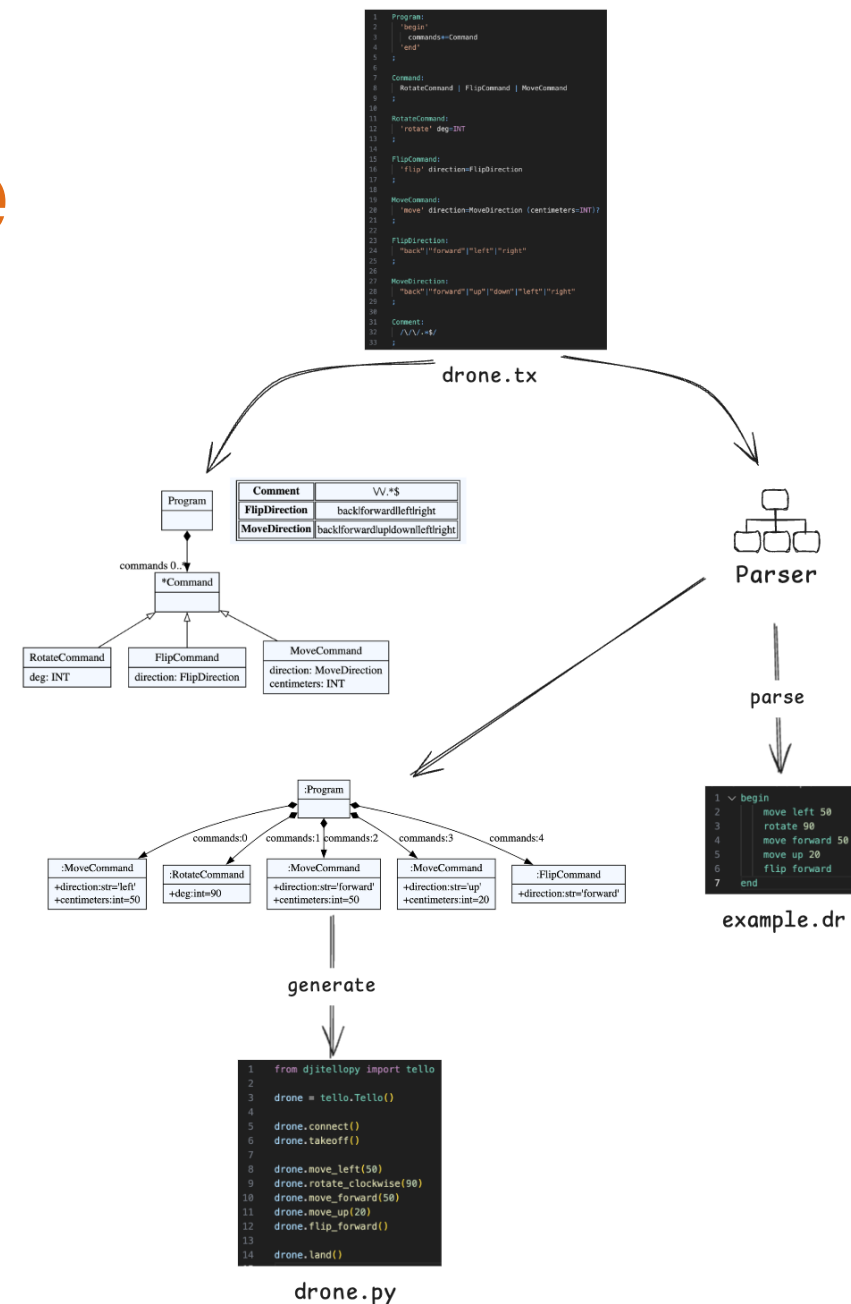
1. **Meta-model** and **Parser** are built from **Grammar file**
2. Parser builds **Model** during **Program** parsing
3. Generate code or interpret

Model corresponds to Meta-model

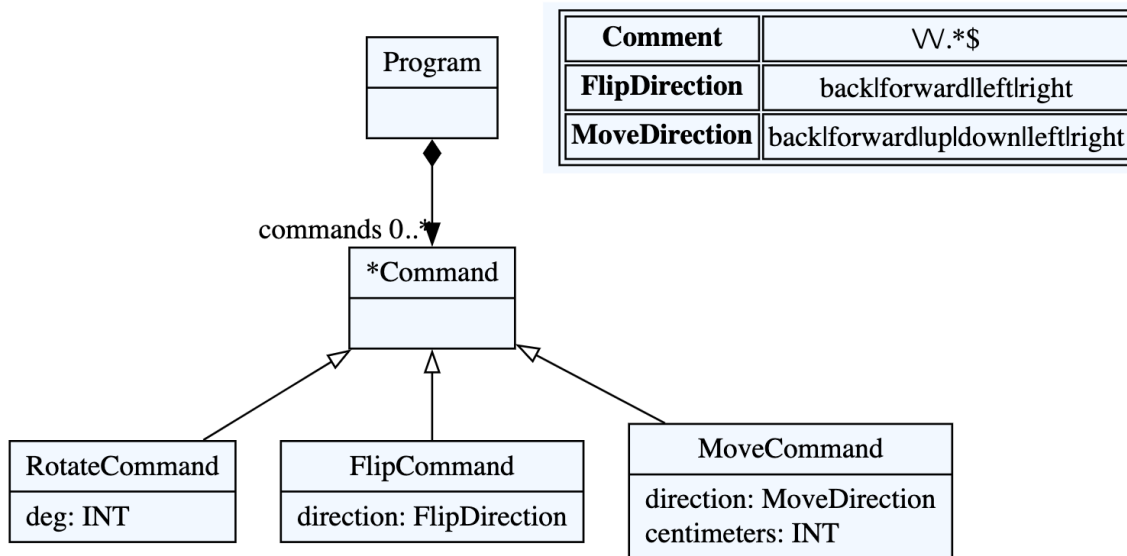


textX – Drone Example

1. Meta-model and Parser built from the `drone.tx` grammar file
2. Parser builds Model during parsing the `example.dr` file
3. Generate executable Python code in `drone.py` file



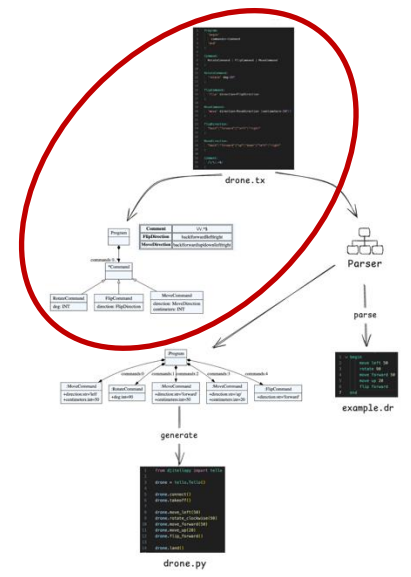
textX – Grammar and Metamodel



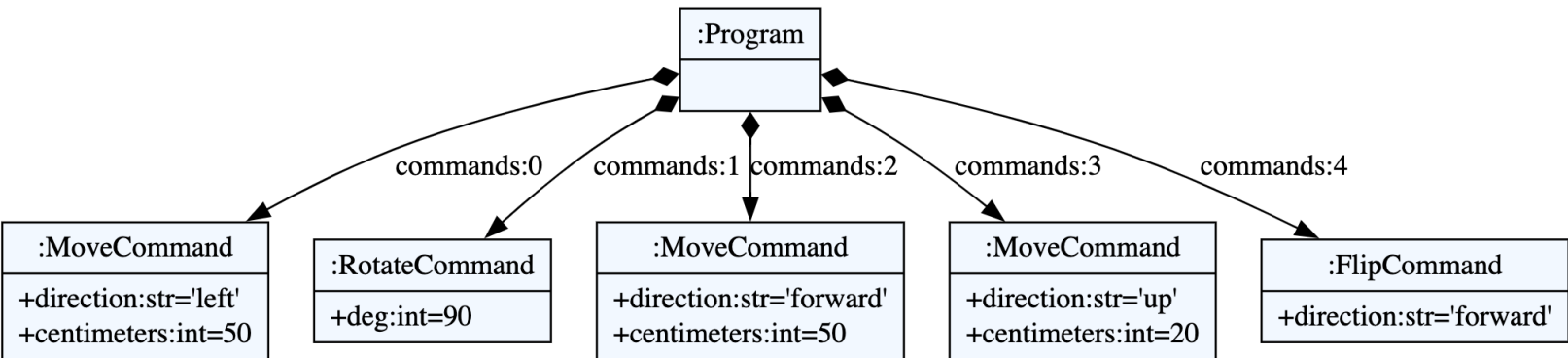
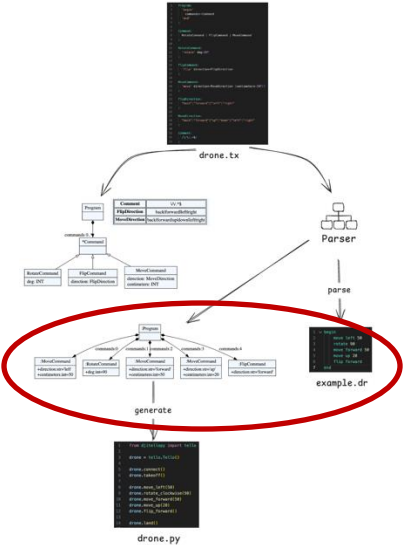
```

1 Program:
2   'begin'
3   commands*=Command
4   'end'
5   ;
6
7 Command:
8   RotateCommand | FlipCommand | MoveCommand
9   ;
10
11 RotateCommand:
12   'rotate' deg=INT
13   ;
14
15 FlipCommand:
16   'flip' direction=FlipDirection
17   ;
18
19 MoveCommand:
20   'move' direction=MoveDirection (centimeters=INT)?
21   ;
22
23 FlipDirection:
24   "back"|"forward"|"left"|"right"
25   ;
26
27 MoveDirection:
28   "back"|"forward"|"up"|"down"|"left"|"right"
29   ;
30
31 Comment:
32   /\\.*$/
33   ;
  
```

drone.tx



textX – Program and Model

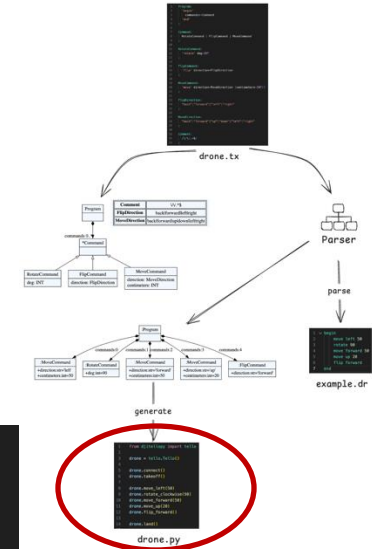


```

1  ✓ begin
2      move left 50
3      rotate 90
4      move forward 50
5      move up 20
6      flip forward
7  end
  
```

example.dr

textX – Code Generation



```
1 from djitellopy import tello
2
3 drone = tello.Tello()
4
5 drone.connect()
6 drone.takeoff()
7
8 {% for command in commands %}
9     {% if command.__class__.__name__ == 'MoveCommand' %}
10    drone.move_{{command.direction}}({{command.centimeters}})
11    {% endif %}
12    {% if command.__class__.__name__ == 'RotateCommand' %}
13    drone.rotate_{{'clockwise' if command.deg > 0 else 'counter_clockwise'}}({{command.deg}})
14    {% endif %}
15    {% if command.__class__.__name__ == 'FlipCommand' %}
16    drone.flip_{{command.direction}}()
17    {% endif %}
18 {% endfor %}
19
20 drone.land()
```

drone.py.jinja

```
1 from djitellopy import tello
2
3 drone = tello.Tello()
4
5 drone.connect()
6 drone.takeoff()
7
8 drone.move_left(50)
9 drone.rotate_clockwise(90)
10 drone.move_forward(50)
11 drone.move_up(20)
12 drone.flip_forward()
13
14 drone.land()
```

drone.py

textX – Validation from python script

```
from textx import metamodel_from_file

# validate metamodel
metamodel = metamodel_from_file('drone.tx')

# validate model based on metamodel
model = metamodel.model_from_file('example.dr')
```

textX – Validation from CLI

- Metamodel validation

```
textx check drone.tx
```

- Model validation

- language not registered

```
textx check example.dr --grammar drone.tx
```

- language registered

```
textx check example.dr
```

```
textx check example.dr --language drone
```



textX – Other CLI commands

- List registered languages

```
textx list-languages
```

- List registered generators

```
textx list-generators
```

- Generate code

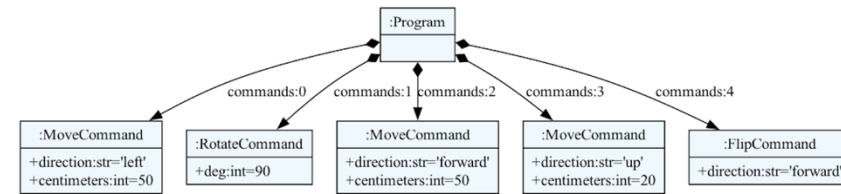
```
textx generate --language Drone --target python --overwrite
```



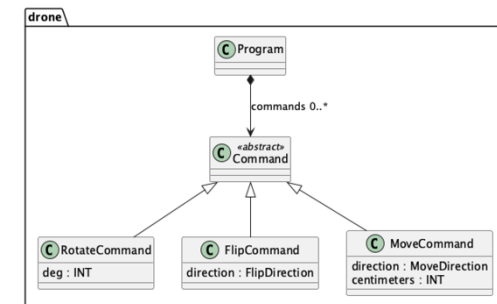
textX – Visualization

- Visualize model and metamodel

```
textx generate --grammar drone.tx --target dot example.dr  
dot -Tpng -O example.dot
```



```
textx generate drone.tx --target plantuml  
plantuml drone.pu
```




Match rules:	
Name	Rule details
FlipDirection	back forward left right
MoveDirection	back forward up down left right
Comment	\\/\\/.*\$

Web Playground

Motivation, Considerations and Solution

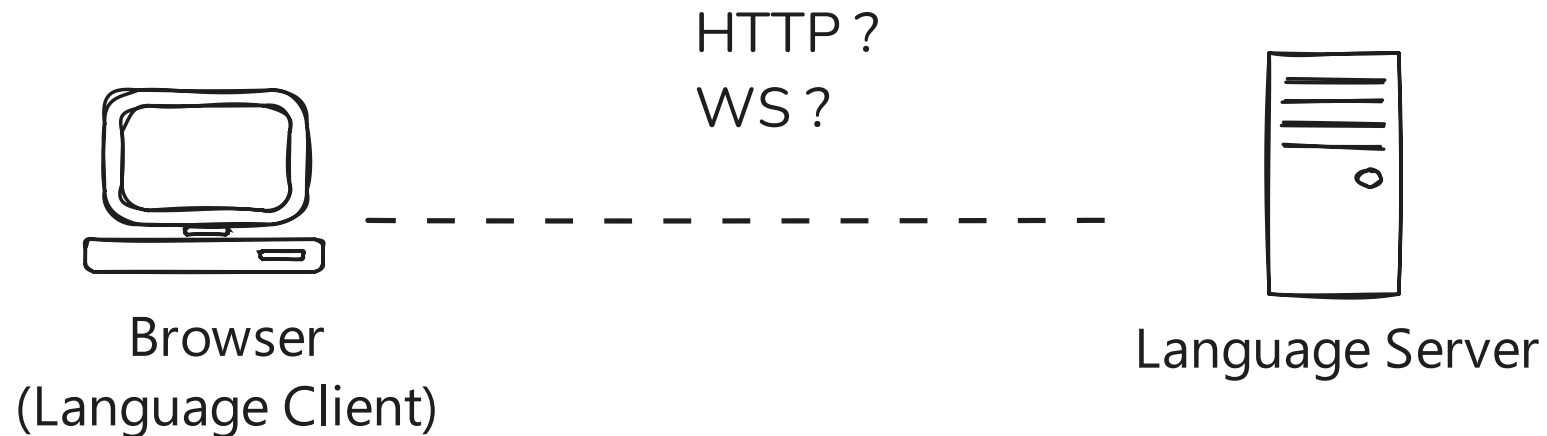


Web Playground – Motivation

- Trying textX includes:
 - Prerequisites - python and pip
 - Creating virtual env (optional)
 - Installing textX
 - Running python script or textX CLI commands
- Web Playground 

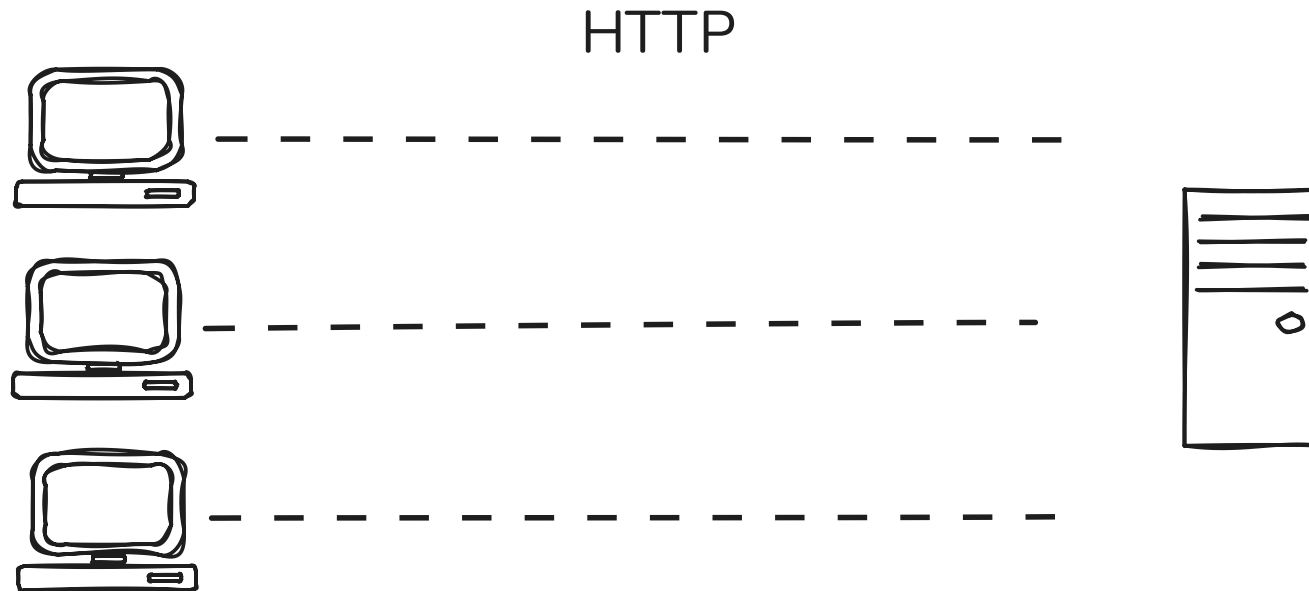
Web Playground - Considerations

- Language Client and Server architecture and communication



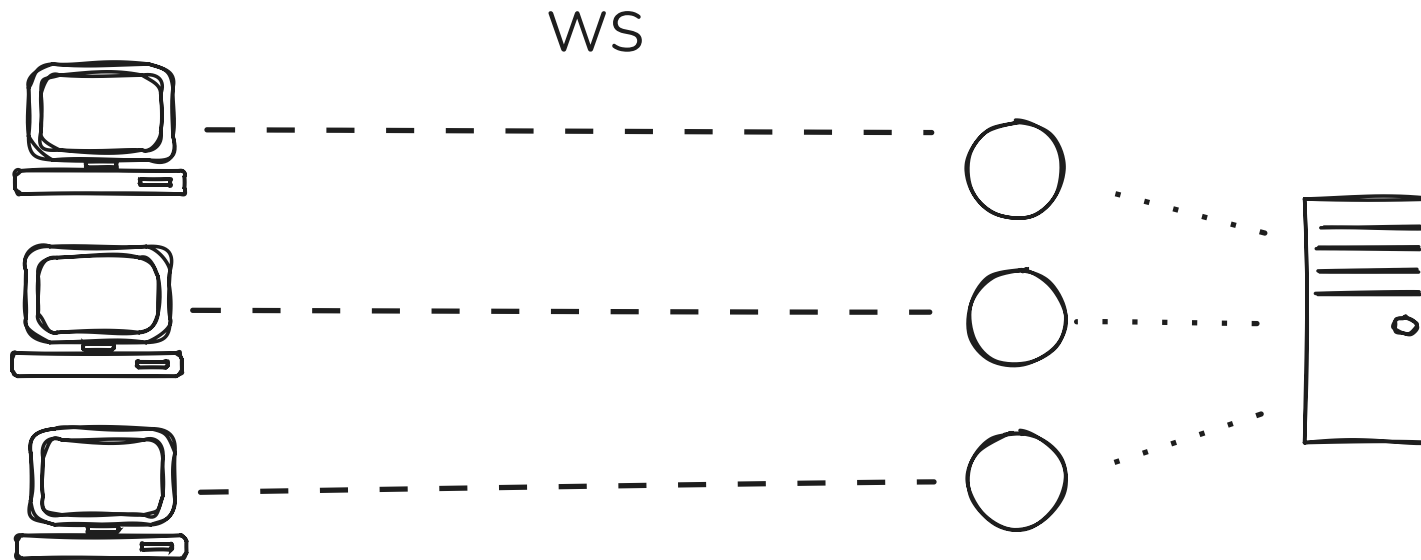
Web Playground - Considerations

- HTTP connection, manage sessions and documents state



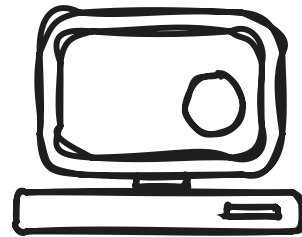
Web Playground - Considerations

- Separate process per client with WebSocket connection



Web Playground - Considerations

- Language server and client on the same machine



Download LS and run it?
Run LS in the browser?

Web Playground

- Created in 2024 by Milan Šović
- Language Server based on [Pygls](#)
- Runs in a Web Worker, using [Pyodide](#)
- Monaco editor
- Language Client based on [monaco-languageclient](#) and [vscode-languageclient](#)

Github: <https://github.com/textX/textx-playground>

Playground: <https://textx.github.io/textx-playground>



Web Playground

The screenshot displays the textX web playground interface, which is split into two main panels: "Language Grammar" on the left and "Model (Program)" on the right. The top navigation bar includes the textX logo, a menu for "Examples", a link to "Documentation", and icons for a dark theme and GitHub. The "Language Grammar" panel shows a tree view of the grammar rules, including Program, Command, RotateCommand, FlipCommand, MoveCommand, FlipDirection, MoveDirection, and Comment. The "Model (Program)" panel shows a simple program with the following code:

```
1 begin
2   move left 50
3   rotate 90
4   move forward 50
5   move up 20
6   flip forward
7 end
```

At the bottom of each panel, there are status indicators: "Grammar is valid" on the left and "Model is valid" on the right. A share icon is visible in the bottom right corner of the Model panel.

Web Playground

textX Examples Documentation

Language Grammar

```
1 Program:
2   'begin'
3   commands*=Command
4   'end'
5 ;
6
7 Command:
8   RotateCommand | FlipCommand | MoveCommand
9 ;
10
11 RotateCommand:
12   'rotate' deg=INT
13 ;
14
15 FlipCommand:
16   'flip' direction=FlipDirection
17 ;
18
19 MoveCommand:
20   'move' direction=MoveDirection (centimeters=INT)?
21 ;
22
23 FlipDirection:
24   "back"|"forward"|"left"|"right"
25 ;
26
27 MoveDirection:
28   "back"|"forward"|"up"|"down"|"left"|"right"
29 ;
30
31 Comment:
32   /\n\n.*$/
33 ;
34
```

Model (Program)

```
1 begin
2   move left 50
3   rotate 90
4   move forward 50
5   move up 20
6   flip forward
7 end
```

Visualize

Share

Status Bar

Grammar is valid Model is valid

Web Playground – Features

- Grammar and Program definition with instant validation
- Basic syntax highlighting
- Grammar and Program visualization
- Light and dark theme
- Examples
- Share

[Share link to Drone example](#)



VS Code Extension



VS Code Extension

- Created by Daniel Elero in 2018
- Contains:
 - **textX Language Server** (based on [Pygls](#))
 - **VS Code Extension**

Github: <https://github.com/textX/textX-LS>

PyPi: <https://pypi.org/project/textx-ls-server/>

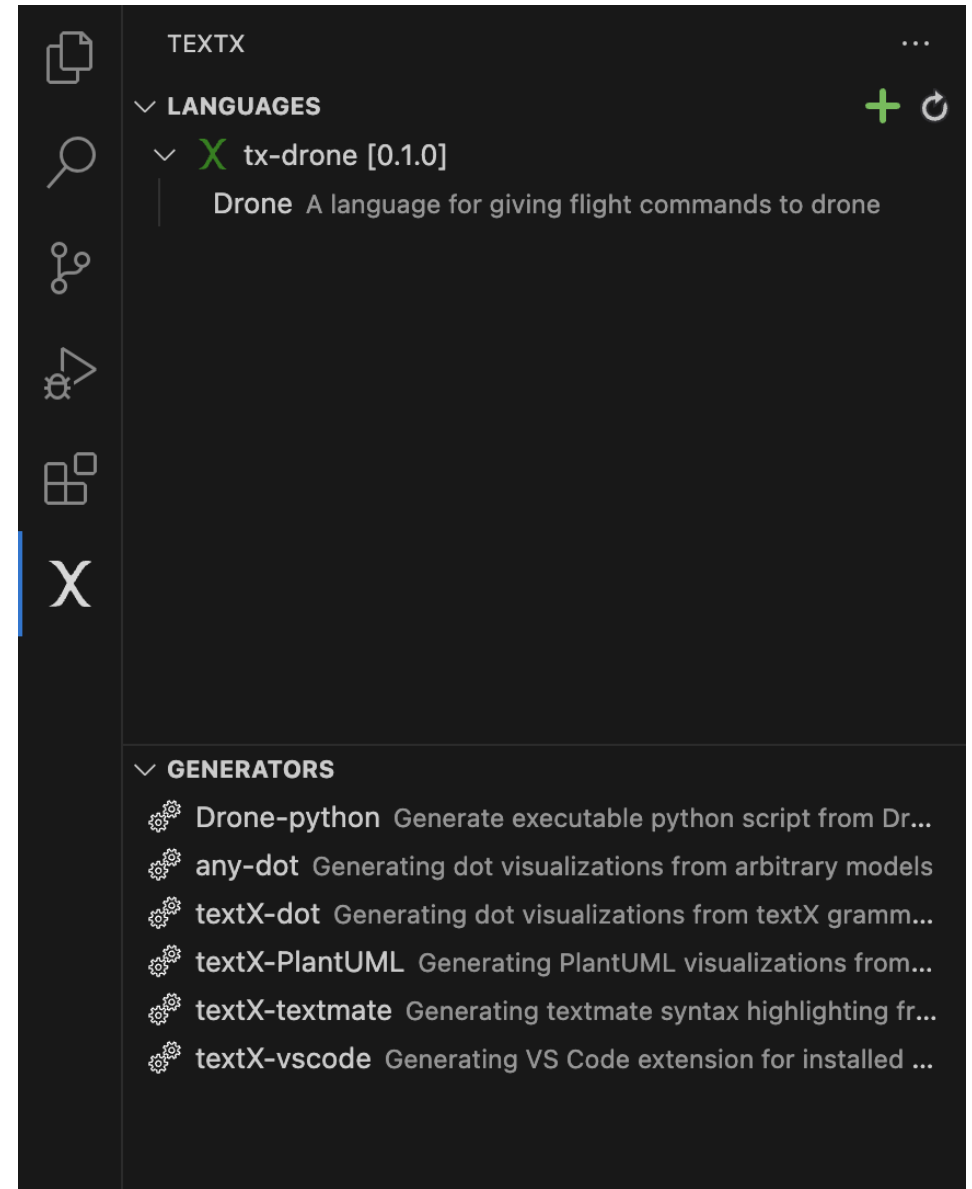
VS Code Marketplace:

<https://marketplace.visualstudio.com/items?itemName=textX.textX>



VS Code Extension - Sidebar

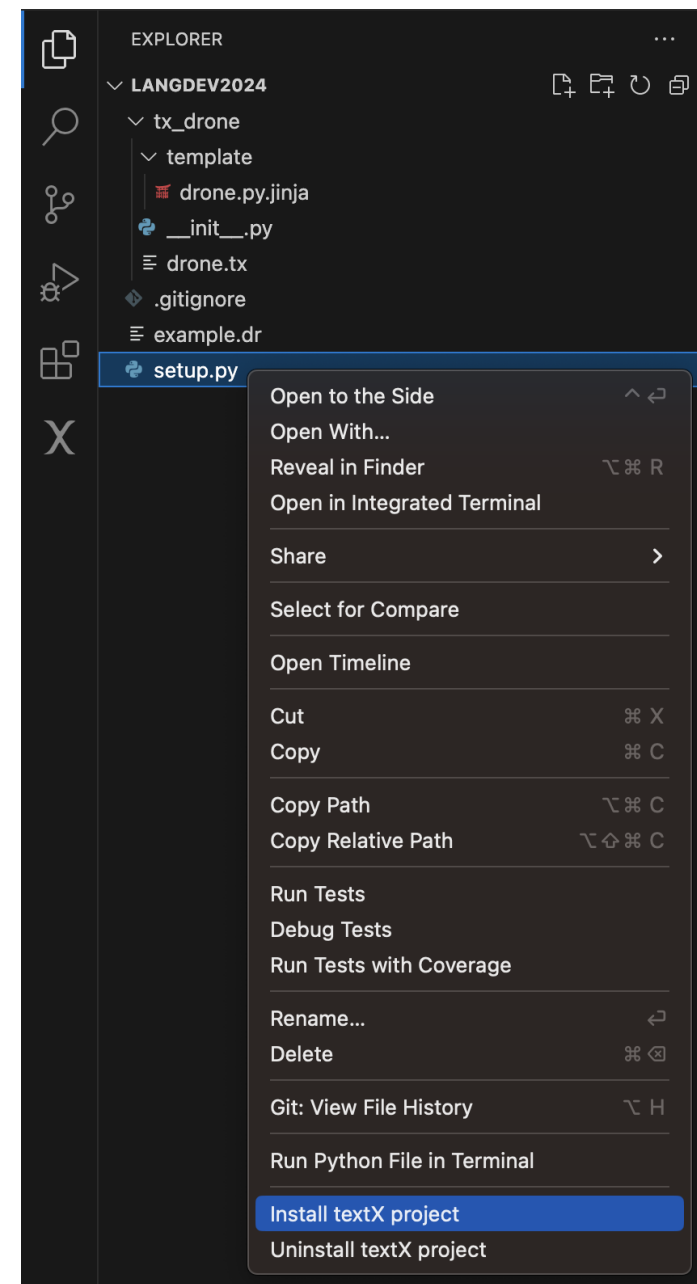
- Project installation from wheel file
- Show and Refresh projects with registered languages
- Show registered generators



VS Code Extension

- Project installation from python package configuration file
- It will be installed in extension's virtual environment

```
textx-ls-core      0.2.0
textx-ls-server   0.2.0
tx-drone           0.1.0
```



VS Code Extension

- Grammar and Program instant validation (error reporting)
- Syntax highlighting

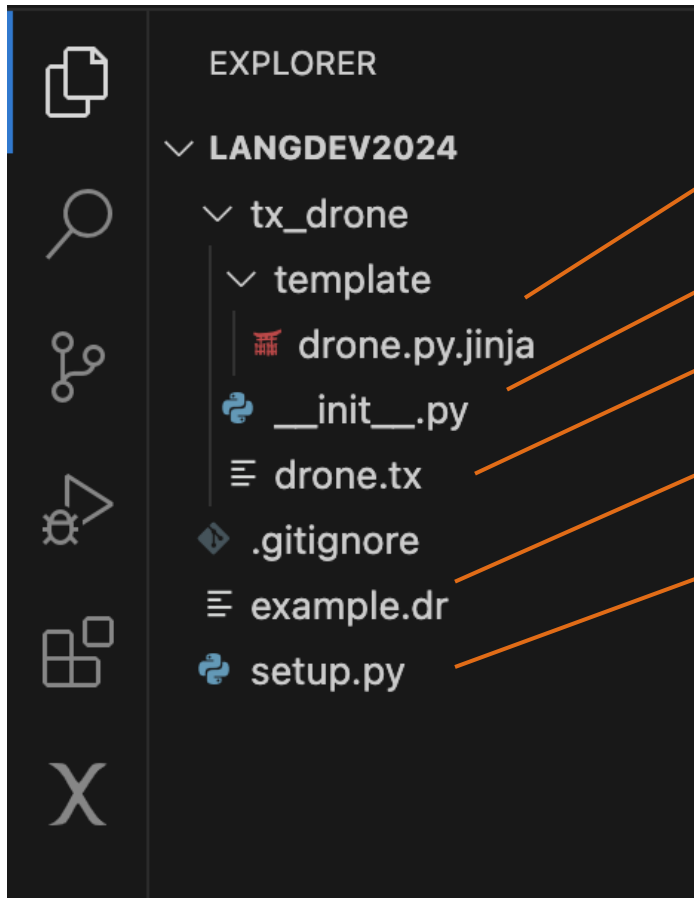
```
tx_drone > ☰ drone.tx
1 Program:
2   'begin'
3   commands*=Command
4   'end'
5 ;
6
7 Command:
8   RotateCommand | FlipCommand | MoveCommand
9 ;
10
11 RotateCommand:
12   'rotate' deg=INT
13   Expected '=' or '*=' or '+=' or '?=' or '*' or '?' or '+' or '#' or '-' or
14   View Problem (⌘.) No quick fixes available
15
16 FlipCommand:
17   'flip' direction=FlipDirection
18 ;
```

```
☰ example.dr
1 begin
2   Expected 'rotate' or 'flip' or 'move' or 'end'
3   View Problem (⌘.) No quick fixes available
4
5   mov
6   fli abc move
7 end
```

Drone Example Demo



Drone Example



- Template
- Language and generator registration
- Grammar
- Program
- Project config file

GitHub: <https://github.com/textX/langdev2024>

Drone Example

- Language and generator registration

```
1 from os.path import dirname, join
2
3 from textx import generator, language, metamodel_from_file
4 from textxjinja import textx_jinja_generator
5
6
7 @language("Drone", ".dr")
8 def drone():
9     "A language for giving flight commands to drone"
10    return metamodel_from_file(join(dirname(__file__), "drone.tx"))
11
12
13 @generator('Drone', 'python')
14 def drone_generator(metamodel, model, output_path, overwrite, debug):
15     "Generate executable python script from Drone model"
16
17     current_dir = dirname(__file__)
18     templates_path = join(current_dir, 'template')
19     default_output_path = join(current_dir, '..', 'dist')
20
21     context = {
22         'commands': model.commands
23     }
24
25     textx_jinja_generator(templates_path, output_path or default_output_path, context, overwrite)
```

The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows a project structure with a file named `setup.py` selected. A context menu is open over this file, listing various actions. The action `Install textX project` is highlighted in blue. On the right, the editor shows the content of `setup.py`. The `entry_points` dictionary is highlighted with a red rounded rectangle, showing the registration of the `drone` language and `drone_generator` generator.

```
1 from setuptools import find_packages, setup
2
3 PACKAGE_NAME = "tx-drone"
4 VERSION="0.1.0"
5 AUTHOR="textX"
6
7 setup(
8     name=PACKAGE_NAME,
9     version=VERSION,
10    packages=find_packages(),
11    include_package_data=True,
12    package_data={"": ["*.tx"]},
13    install_requires=["textx_ls_core", "textX-jinja"],
14    entry_points={
15        "textx_languages": [
16            "drone = tx_drone:drone"
17        ],
18        "textx_generators": [
19            "drone_generator = tx_drone:drone_generator"
20        ]
21    },
22 )
```

Drone Example

- Generated code execution

```
begin
  move left 50
  rotate 90
  move forward 50
  move up 20
end
```

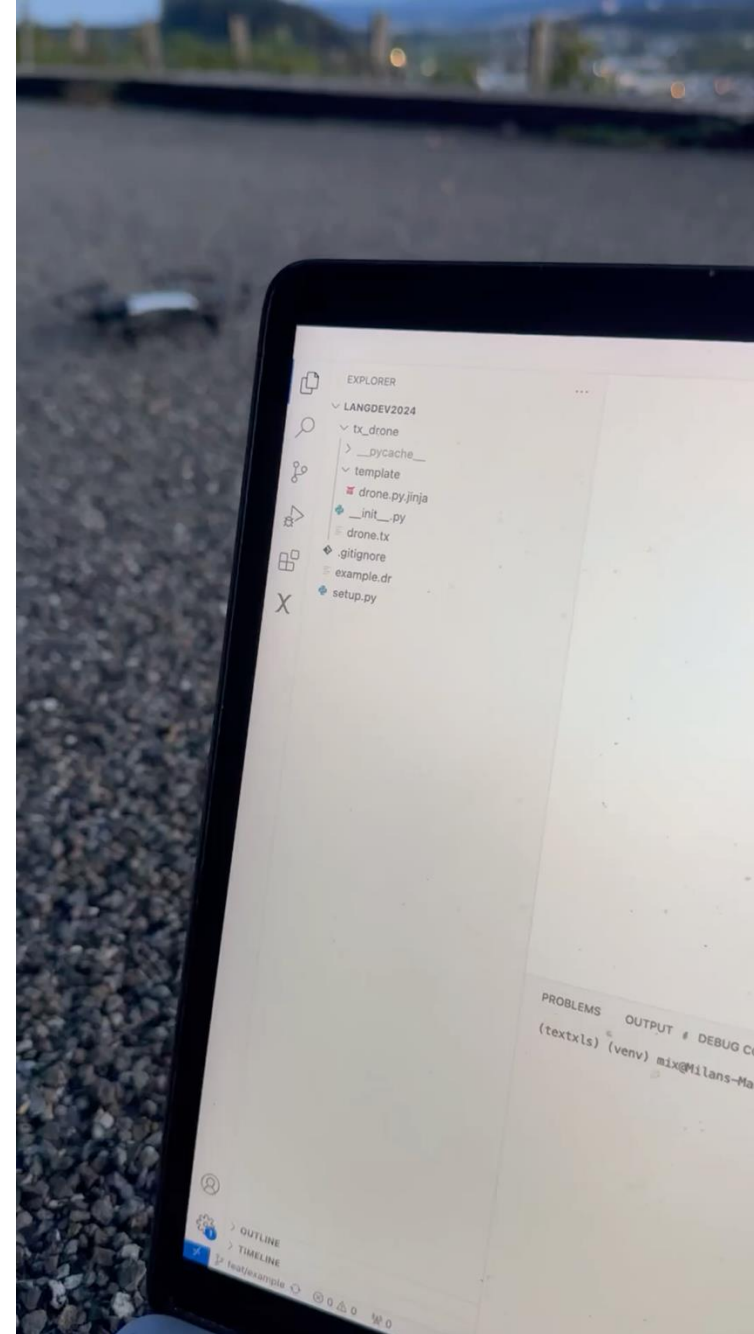
```
from djitellopy import tello

drone = tello.Tello()

drone.connect()
drone.takeoff()

drone.move_left(50)
drone.rotate_clockwise(90)
drone.move_forward(50)
drone.move_up(20)

drone.land()
```



Summary

Latest Updates and Next Steps



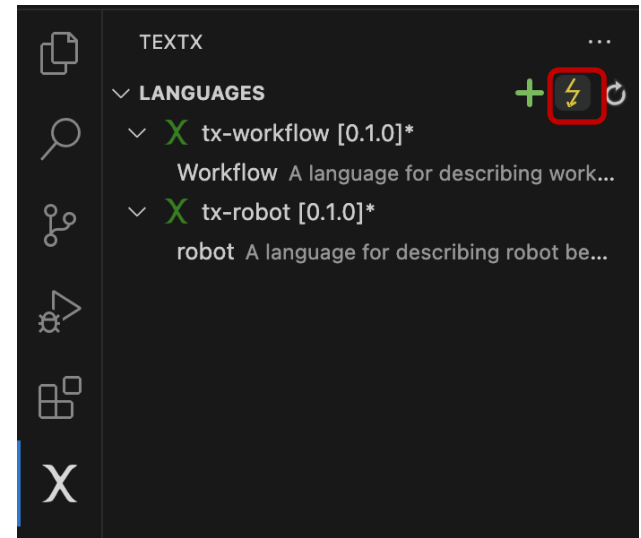
Summary – Latest Updates

- Created textX Web Playground
- New textX LS version:
 - upgraded python, textX and other dependencies versions
 - packages **textx-ls-core 0.2.0** and **textx-ls-server 0.2.0**
- New VS Code Extension version – **textX v0.2.0**



Summary – Next Steps

- VS Code Extension improvements:
 - project scaffolding
 - run code generators from tree view
 - run textX LS in VS Code inside Pyodide (no Python dependency)
- Playground improvements:
 - replace custom Pygls implementation with textX LS



JGBX



Q&A

textX Organization | <https://github.com/textX>

Presentation and Demos | <https://github.com/textX/langdev2024>

Appendix – textX Related Repos

- [textX](#) – language
- [textX-LS](#) (upgraded) – language server and VC Code extension
- [textx-playground](#) (new) – web playground
- [textX-jinja](#) – template-based code generation from textX models
- [textX-dev](#) – project scaffolding
- [textX-lang-questionnaire](#) – questionnaire DSL, used by textX-dev

textX