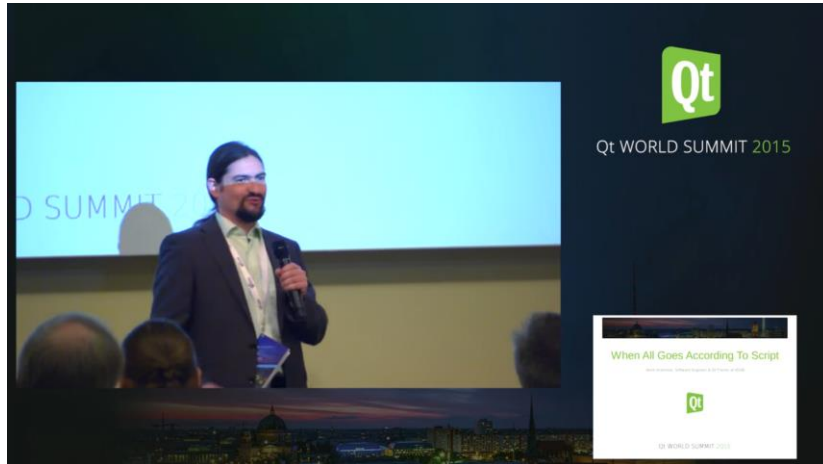


# Script automation

## Recording user actions

Nicolas Arnaud-Cormos  
[nicolas.arnaud-cormos@kdab.com](mailto:nicolas.arnaud-cormos@kdab.com)

# Scripting Qt application



## When all goes according to script

QtWS 2015

Kevin Krammer, KDAB



## Practical application scripting with QML

QtWS 2019

Kevin Krammer, KDAB

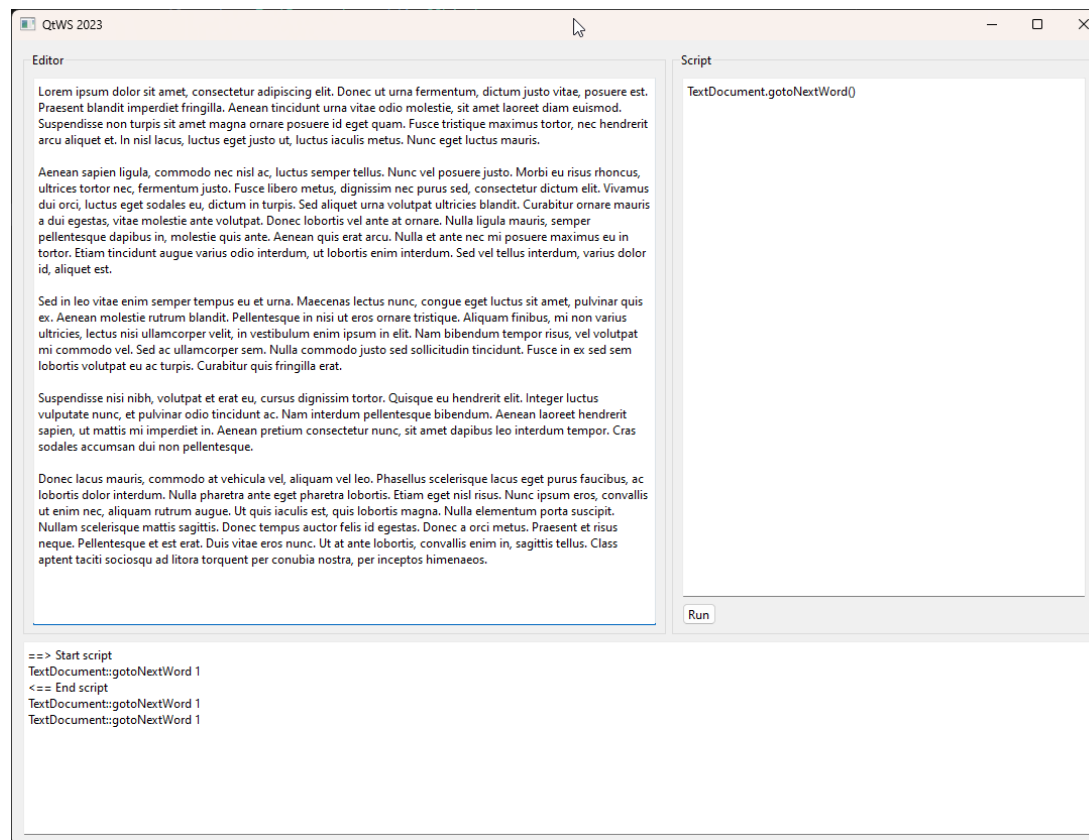
You got your application scriptable!

And now...

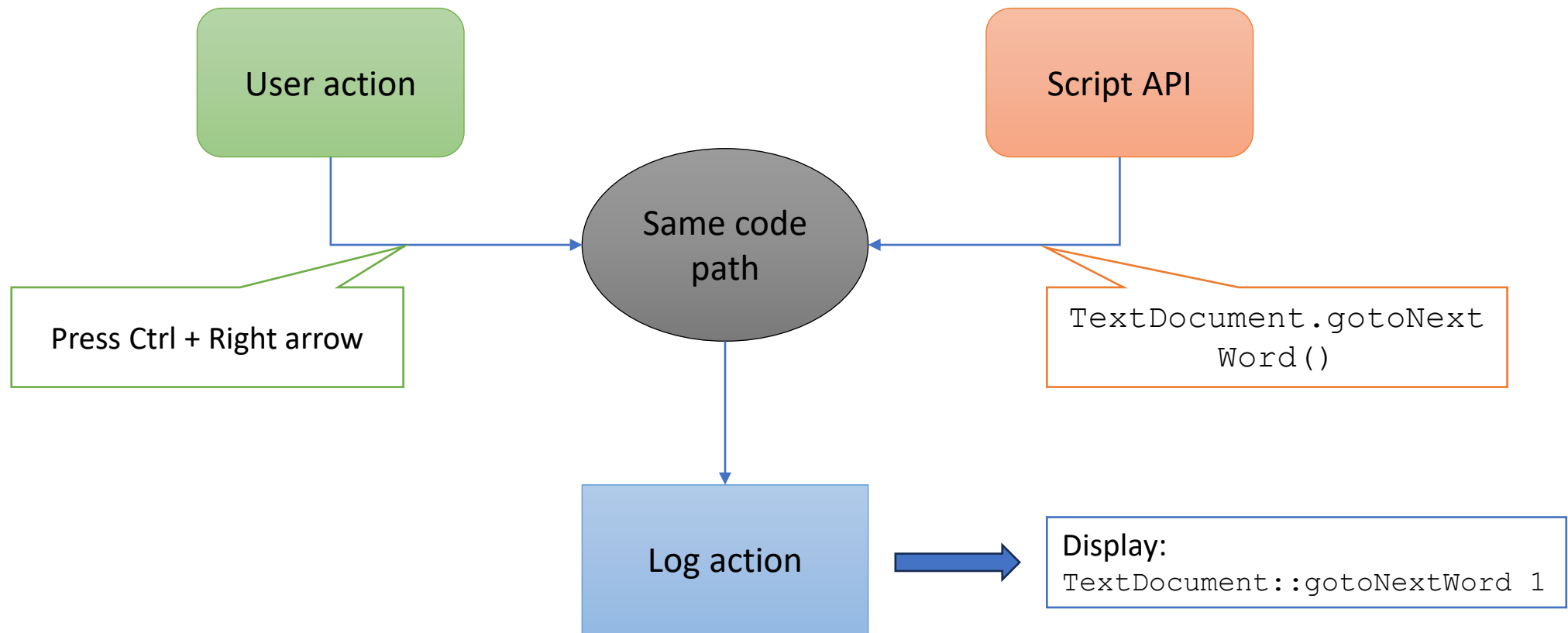
# How to create a script from user actions ?

# The setup

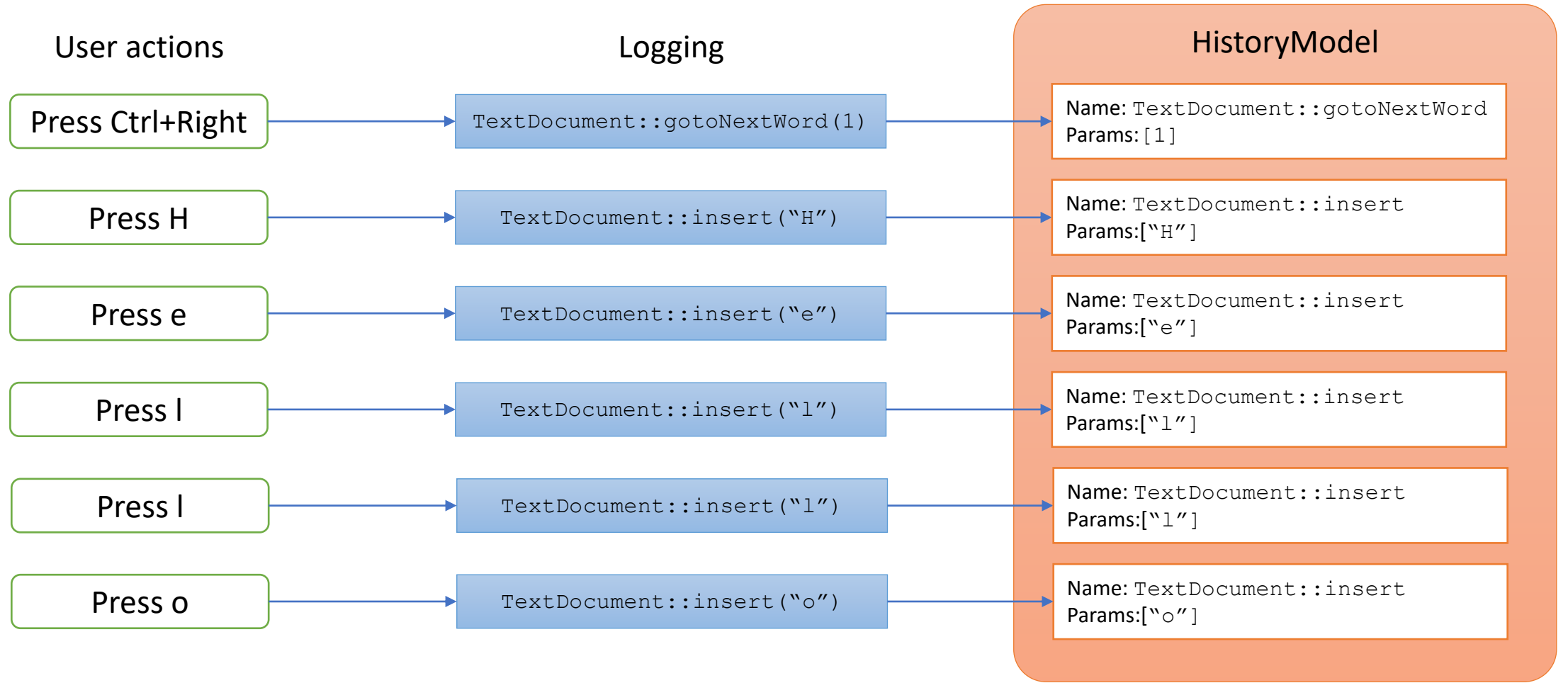
- Simple text editor with scripting ability.
  - Code available here: <https://github.com/KDABLabs/qtws23-script-automation>



# Code preparation



# Use and abuse logging system



# Create script from history

## HistoryModel

Name: TextDocument::gotoNextWord  
Params: [1]

Name: TextDocument::insert  
Params: ["H"]

Name: TextDocument::insert  
Params: ["e"]

Name: TextDocument::insert  
Params: ["l"]

Name: TextDocument::insert  
Params: ["l"]

Name: TextDocument::insert  
Params: ["o"]

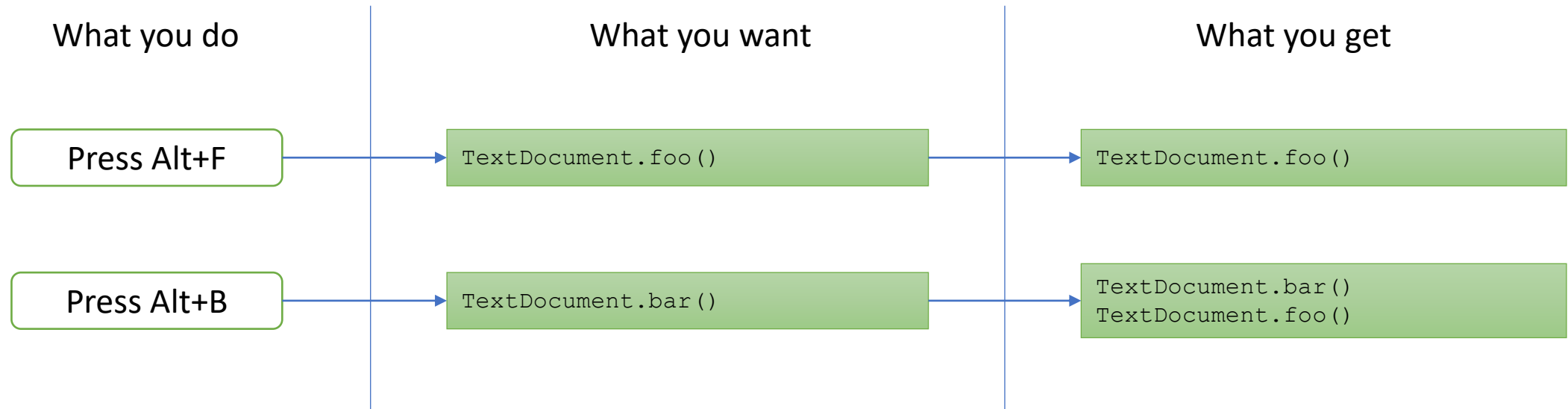


```
TextDocument.gotoNextWord(1)
TextDocument.insert("H")
TextDocument.insert("e")
TextDocument.insert("l")
TextDocument.insert("l")
TextDocument.insert("o")
```

# Don't log everything – API called by an API

## Example

- one API `TextDocument::foo (Alt+F)`
- one API `TextDocument::bar (Alt+B)` calling `TextDocument::foo`

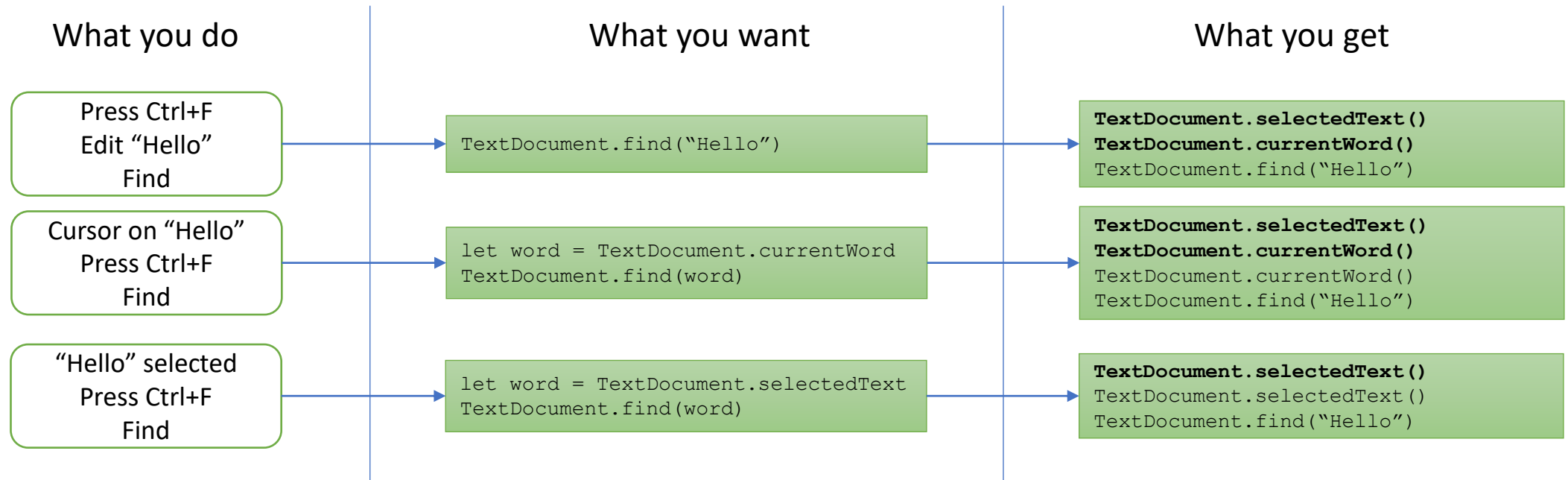




# Don't log everything – API used internally

## Example: Find feature

- Call `TextDocument::selectedText` and if empty `TextDocument::currentWord` to fill the line edit
- On find, if the user has not changed the text from the line edit, re-call the previous API



# Use return values in script

What you do

Cursor on "Hello"  
Press Ctrl+F  
Find

What you want

```
let word = TextDocument.currentWord
TextDocument.find(word)
```

What you get

```
TextDocument.currentWord()
TextDocument.find("Hello")
```

To fix the issue, we need to do different steps

1. Store the return value from the API
2. Tag return values and parameters to know which ones are compatible
3. Check previous return value and parameter value during script creation
  - If they are equal, use a variable to pass it

Logging

TextDocument::currentWord()

TextDocument::find("Hello")

HistoryModel

Name: TextDocument::currentWord  
Params: []  
Return: (**text**, "Hello")

Name: TextDocument::find  
Params: [ (**text**, "Hello") ]  
Return: ()

```
let text = TextDocument.currentWord()
TextDocument.find(text)
```

# Final touch: handling properties

What you do

Cursor on "Hello"  
Press Ctrl+F  
Find

What you want

```
let word = TextDocument.currentWord
TextDocument.find(word)
```

What you get

```
let word = TextDocument.currentWord()
TextDocument.find(word)
```

To fix the issue, we need to do different steps

1. Store all properties for all QML objects
2. During script creation, check if the API is a property or not

Code

```
TextDocument::TextDocument (...)
{
    LOG_REGISTER(TextDocument);
```

property  
List

HistoryModel

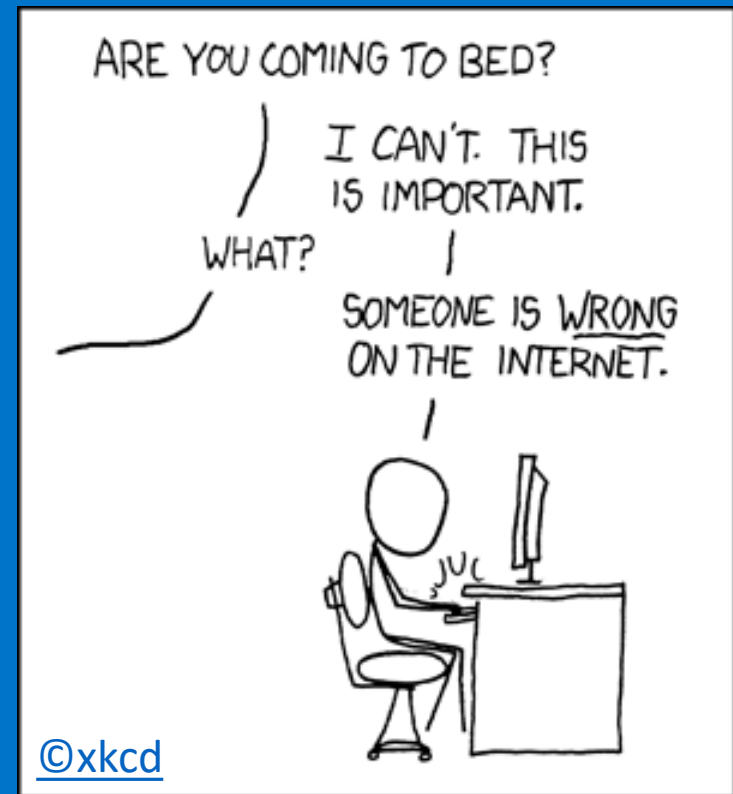
```
if (apiCall is in
    propertyList) {
    if (no params)
        // read property
    else
        // write property
} else {
    // handle function
}
```

```
let foo = TextDocument.apiCall
```

```
TextDocument.apiCall = foo
```

```
let foo = TextDocument.apiCall()
// or
TextDocument.apiCall(text)
```

# When you think you are done...



# There are always more!

- Merge calls that could be merged

```
TextDocument.insert("H")  
TextDocument.insert("e")  
TextDocument.insert("l")  
TextDocument.insert("l")  
TextDocument.insert("o")
```



```
TextDocument.insert("Hello")
```

As an exercise for the developer

- Handle complex parameters
- Handle QML singletons and non-singletons
- Simplify the LOG macro (using `std::source_location`)
- Store properties at compilation time
- ...



Follow us

# Thank you!



Get the code