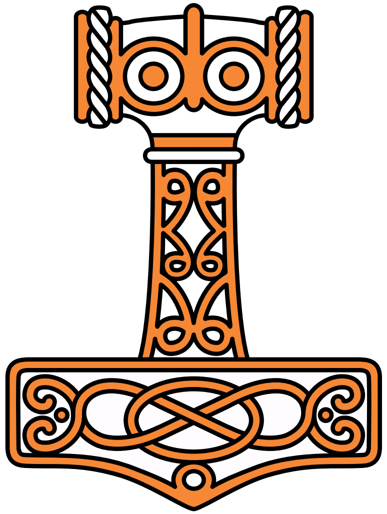
Olhão 2022

Text-Based Sources
(SA3)

Josh David
Morten Kromberg



Goals

- Give an introduction to Link
- Walk through the process of moving source from a workspace to text files
- Demonstrate how to link the source to a GitHub repository and use VS Code
- Introduce the Dado project management system and discuss benefits

Workshop Overview

09:30-10:30 Getting Started with Link

- What is Link?
- Starting a new project
- Converting an existing project

10:45-11:45 GitHub & VS Code

- Create a GitHub repo and push some code to it
- VS Code ("How Morten Develops Code") & Some Alternatives

12:00-13:00 Dado

- Managing the same application using Dado
- Benefits and Limitations of Dado vs a more open approach

Check List – Have You...

- Got Link 3.0 installed?
- Got Dado installed
  - If not – we will install it together
- Downloaded Workshop Materials?
  - If not – see next page
- Signed up for a GitHub account?
  - Have you configured your Git client to use a "PAT"?
- How many have VS Code or a similar tool with Git support installed?
- Brought your own workspace to convert to text source?
  - Never mind, we probably don't have time for that ☺

Materials

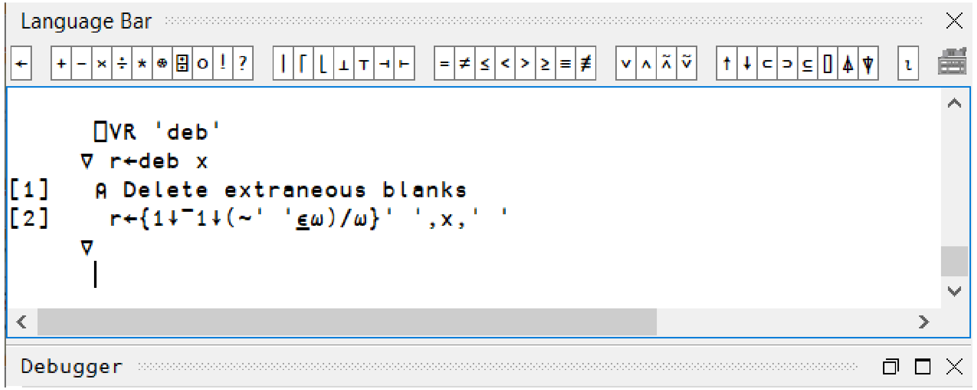
Materials used can be found in
https://github.com/dyalog-training/2022-SA3

- Unzip the latest release, or
- Copy the folder 2022-SA3 from the USB drive (workspace & presentation)

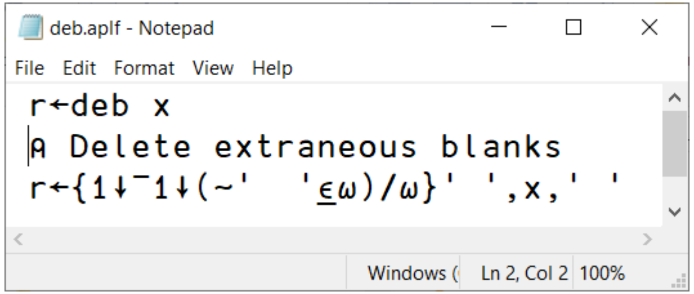
Session 1: Getting Started

- Starting a New Project
- Converting an existing workspace to text

But first - What exactly is Link?



```
      ⎕VR 'deb'
     ∇ r←deb x
[1]    ⍝ Delete extraneous blanks
[2]     r←{1↓¯1↓(~'  '⍷ω)/ω}' ',x,' '
     ∇
```



```
r←deb x
⍝ Delete extraneous blanks
r←{1↓¯1↓(~'  '⍷ω)/ω}' ',x,' '
```

- Each code item in the active workspace is **link**ed to a file
  - [Unscripted] Namespaces map to a directory structure containing these files
- If the object is edited, the file is updated
- If the file is changed, the workspace is updated

Wait – isn't that what SALT was doing?

- Link replaces SALT
  - (SALT will be available until no longer used)

- With Link…
  - The interpreter is tracking the relationships between objects and files
  - A *File System Watcher* responds to external changes (requires .NET, supported under Windows, Mac & Linux)

File System Watcher

- Appropriate for synchronising the WS with changes made in an external editor
- Not appropriate for handling "bulk" changes, such as
  - Unzipping lots of files into a watched folder
  - Doing a large checkout/revert
- Not appropriate for watching shared drives

Why is Link IMPORTANT ?

- With source code in text files we can use ***extremely*** attractive tools developed outside the APL community
  - Tools for editing, comparing, mergeing, refactoring, sharing, building, testing, computing statistics, ...
- ... in addition to all our own tools
- ... without losing any of what is good about interactive development

Visual Studio Code

git

Jenkins

Why is Link IMPORTANT ?

MuleSoft

eclipse

SUBVERSION

mercurial

Jenkins

Visual Studio Code

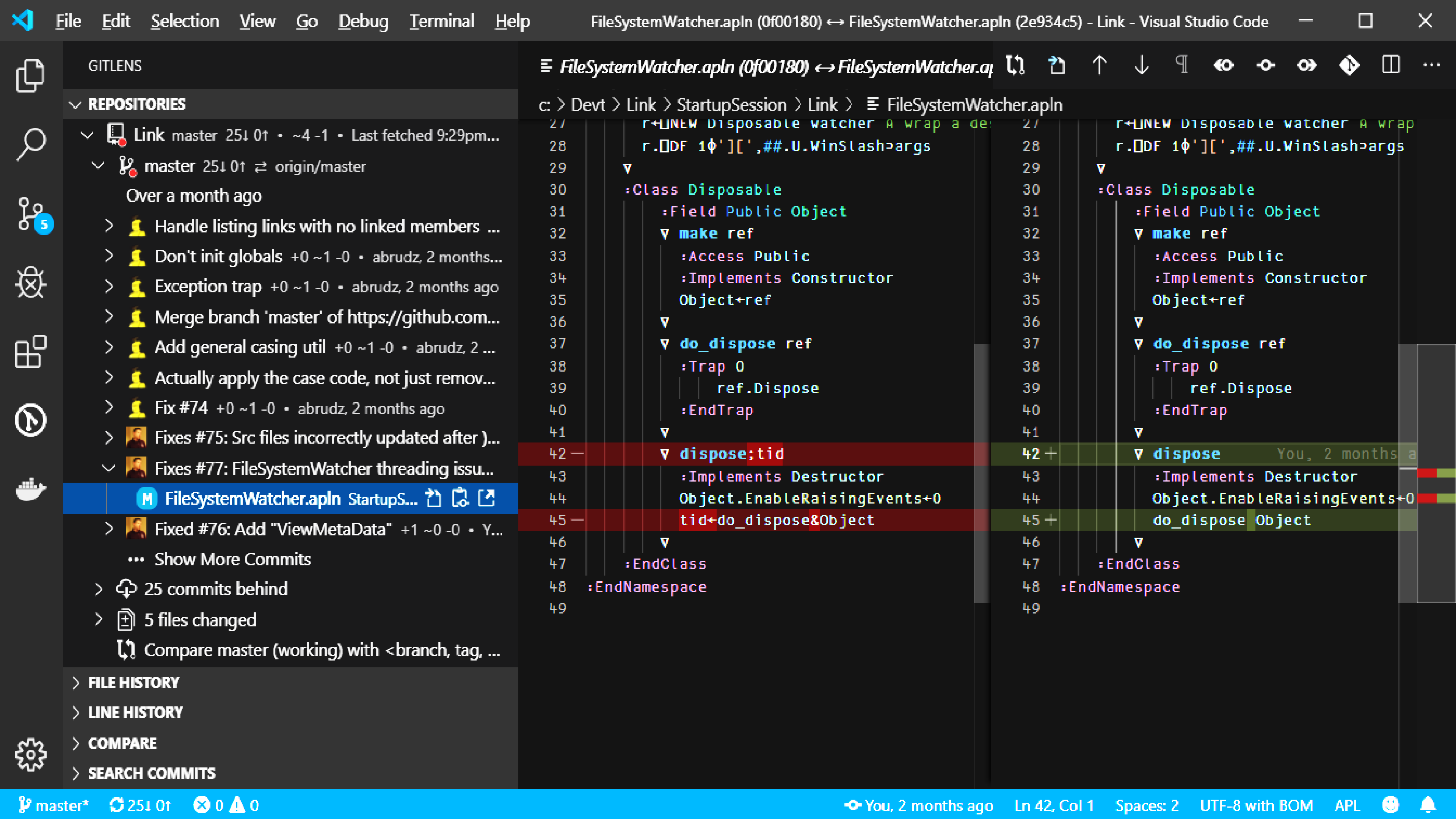
GitHub

git

circleci

Emacs

Vim


File Edit Selection View Go Debug Terminal Help FileSystemWatcher.apln (0f00180) ↔ FileSystemWatcher.apln (2e934c5) - Link - Visual Studio Code

GITLENS

REPOSITORIES

- Link master 25↓ 0↑ • ~4 -1 • Last fetched 9:29pm...
  - master 25↓ 0↑ ⇄ origin/master
    - Over a month ago
    - Handle listing links with no linked members ...
    - Don't init globals +0 ~1 -0 • abrudz, 2 months...
    - Exception trap +0 ~1 -0 • abrudz, 2 months ago
    - Merge branch 'master' of https://github.com...
    - Add general casing util +0 ~1 -0 • abrudz, 2 ...
    - Actually apply the case code, not just remov...
    - Fix #74 +0 ~1 -0 • abrudz, 2 months ago
    - Fixes #75: Src files incorrectly updated after)...
    - Fixes #77: FileSystemWatcher threading issu...
      - M FileSystemWatcher.apln StartupS...
    - Fixed #76: Add "ViewMetaData" +1 ~0 -0 • Y...
    - ••• Show More Commits
  - 25 commits behind
  - 5 files changed
  - Compare master (working) with <branch, tag, ...

FILE HISTORY

LINE HISTORY

COMPARE

SEARCH COMMITS

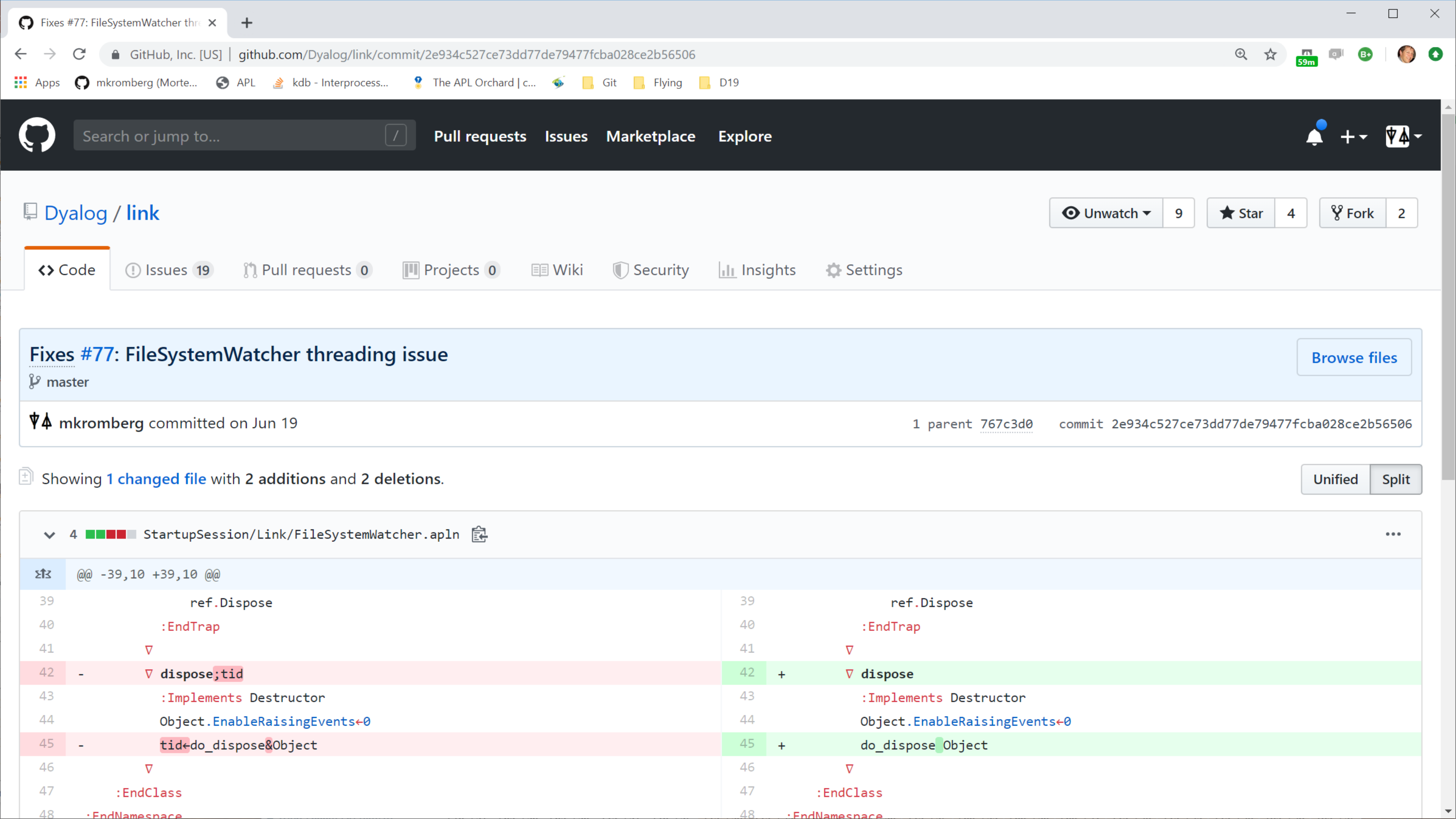
FileSystemWatcher.apln (0f00180) ↔ FileSystemWatcher.ap

c: > Devt > Link > StartupSession > Link > FileSystemWatcher.apln

```
            r←⎕NEW Disposable watcher ⍝ wrap a de
            r.⎕DF 1⌽'][',##.U.WinSlash⊃args
        ∇
        :Class Disposable
            :Field Public Object
            ∇ make ref
              :Access Public
              :Implements Constructor
              Object←ref
            ∇
            ∇ do_dispose ref
              :Trap 0
                  ref.Dispose
              :EndTrap
            ∇
-           ∇ dispose;tid
              :Implements Destructor
              Object.EnableRaisingEvents←0
-             tid←do_dispose&Object
            ∇
        :EndClass
    :EndNamespace
```

```
            r←⎕NEW Disposable watcher ⍝ wrap
            r.⎕DF 1⌽'][',##.U.WinSlash⊃args
        ∇
        :Class Disposable
            :Field Public Object
            ∇ make ref
              :Access Public
              :Implements Constructor
              Object←ref
            ∇
            ∇ do_dispose ref
              :Trap 0
                  ref.Dispose
              :EndTrap
            ∇
+           ∇ dispose        You, 2 months a
              :Implements Destructor
              Object.EnableRaisingEvents←0
+             do_dispose Object
            ∇
        :EndClass
    :EndNamespace
```

master* 25↓ 0↑ ⊗ 0 ⚠ 0 You, 2 months ago Ln 42, Col 1 Spaces: 2 UTF-8 with BOM APL



Fixes #77: FileSystemWatcher thr

GitHub, Inc. [US] | github.com/Dyalog/link/commit/2e934c527ce73dd77de79477fcba028ce2b56506

Apps · mkromberg (Morte... · APL · kdb - Interprocess... · The APL Orchard | c... · Git · Flying · D19

Search or jump to... · Pull requests · Issues · Marketplace · Explore

Dyalog / link

Unwatch 9 · Star 4 · Fork 2

Code · Issues 19 · Pull requests 0 · Projects 0 · Wiki · Security · Insights · Settings

Fixes #77: FileSystemWatcher threading issue

master

Browse files

mkromberg committed on Jun 19

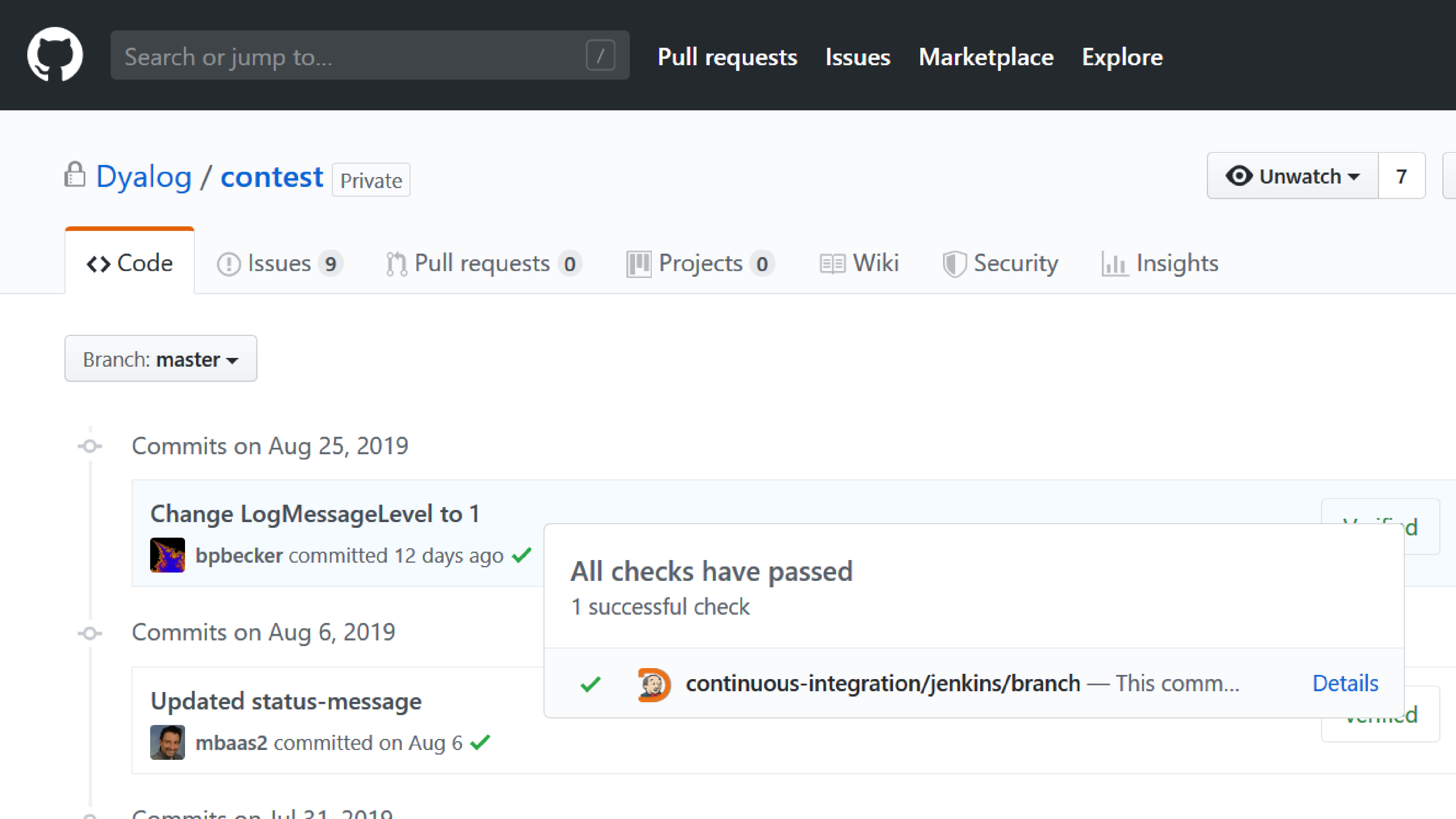
```
1 parent 767c3d0    commit 2e934c527ce73dd77de79477fcba028ce2b56506
```

Showing 1 changed file with **2 additions** and **2 deletions**.

Unified · Split

4 🟩🟩🟥🟥⬜ `StartupSession/Link/FileSystemWatcher.apln`

```
@@ -39,10 +39,10 @@
39         ref.Dispose                    39         ref.Dispose
40       :EndTrap                         40       :EndTrap
41     ∇                                  41     ∇
42 -   ∇ dispose;tid                      42 +   ∇ dispose
43       :Implements Destructor           43       :Implements Destructor
44       Object.EnableRaisingEvents←0     44       Object.EnableRaisingEvents←0
45 -     tid←do_dispose&Object            45 +     do_dispose Object
46     ∇                                  46     ∇
47   :EndClass                            47   :EndClass
```



Dyalog / contest Private

Commits on Aug 25, 2019

Change LogMessageLevel to 1

bpbecker committed 12 days ago ✓

All checks have passed

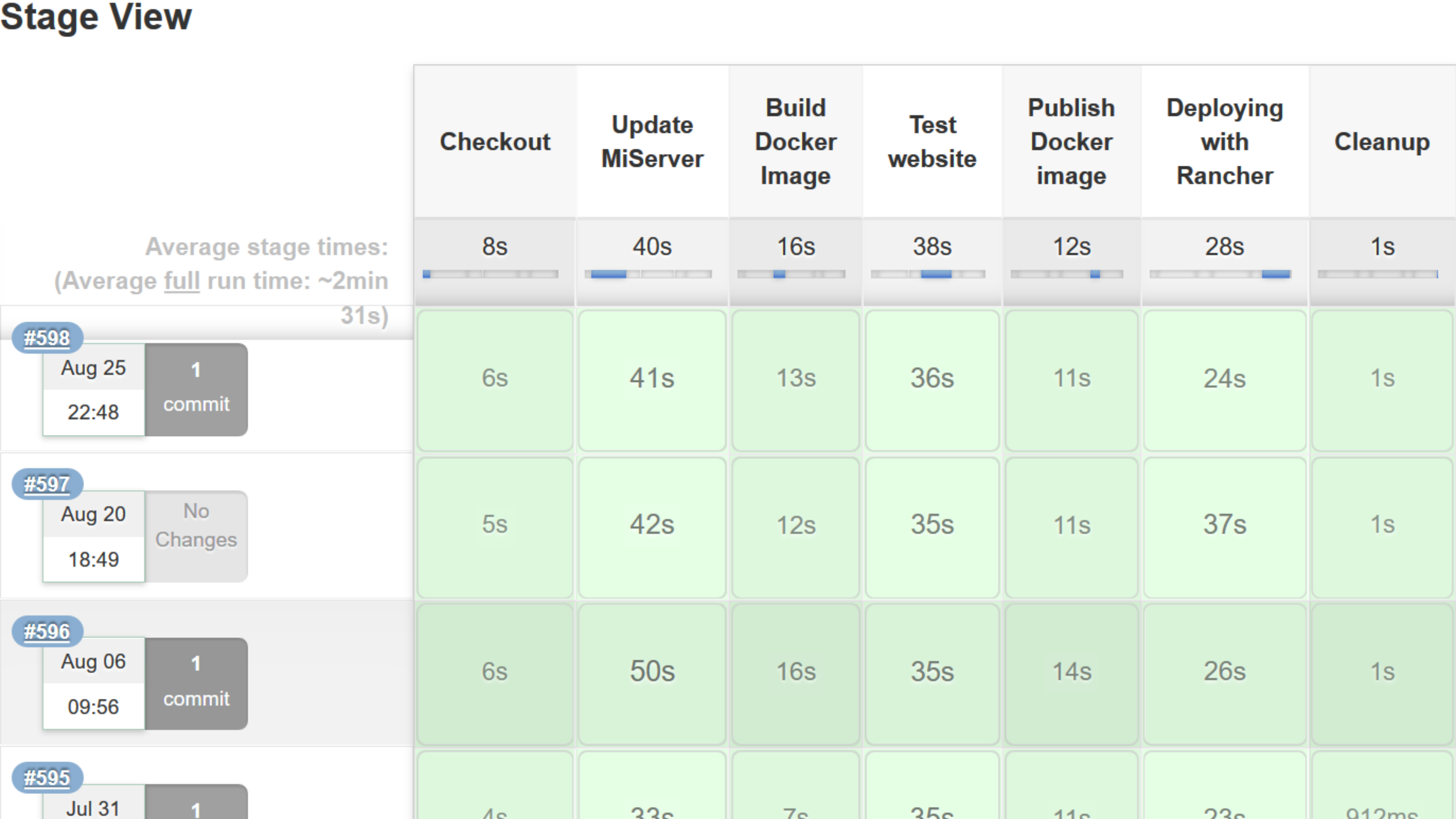
1 successful check

✓ continuous-integration/jenkins/branch — This comm... Details

Commits on Aug 6, 2019

Updated status-message

mbaas2 committed on Aug 6 ✓



Stage View

	Checkout	Update MiServer	Build Docker Image	Test website	Publish Docker image	Deploying with Rancher	Cleanup
Average stage times: (Average full run time: ~2min 31s)	8s	40s	16s	38s	12s	28s	1s
#598 Aug 25 22:48 1 commit	6s	41s	13s	36s	11s	24s	1s
#597 Aug 20 18:49 No Changes	5s	42s	12s	35s	11s	37s	1s
#596 Aug 06 09:56 1 commit	6s	50s	16s	35s	14s	26s	1s
#595 Jul 31 1	4s	33s	7s	35s	11s	23s	912ms

Other Benefits of Text Source

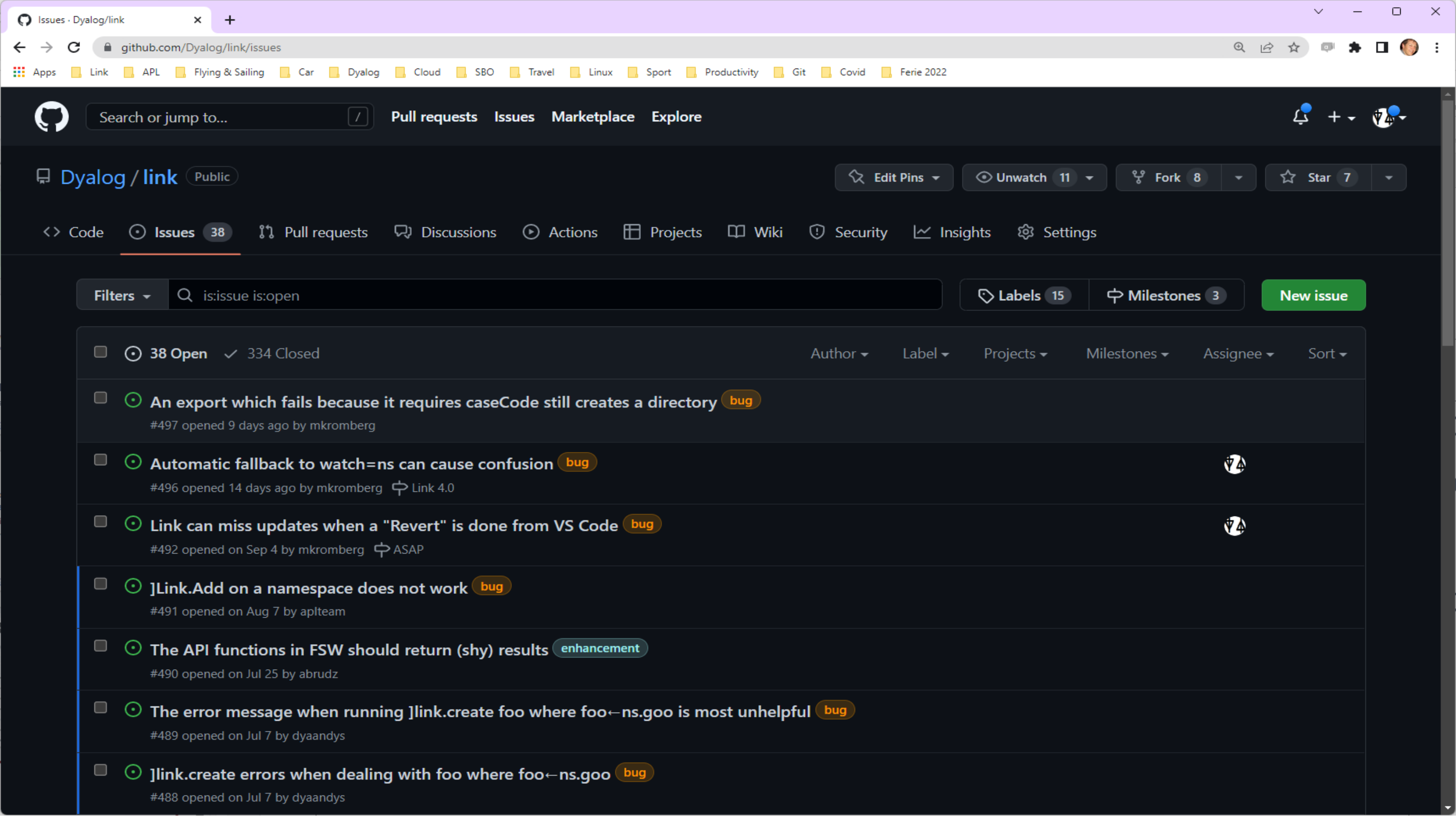
- Easily share code between APL versions
  - Text files are backwards **and** forwards compatible

Drawbacks of Text Source

- This space intentionally left blank

Link in 2022 – Version 3.0

- Link 3.0 was shipped with 18.2
  - Also works with 18.0, but not 17.1
- Rapidly growing user base
- Pretty good documentation
- Link is developed & supported on GitHub



38 Open ✓ 334 Closed

An export which fails because it requires caseCode still creates a directory bug
#497 opened 9 days ago by mkromberg

Automatic fallback to watch=ns can cause confusion bug
#496 opened 14 days ago by mkromberg Link 4.0

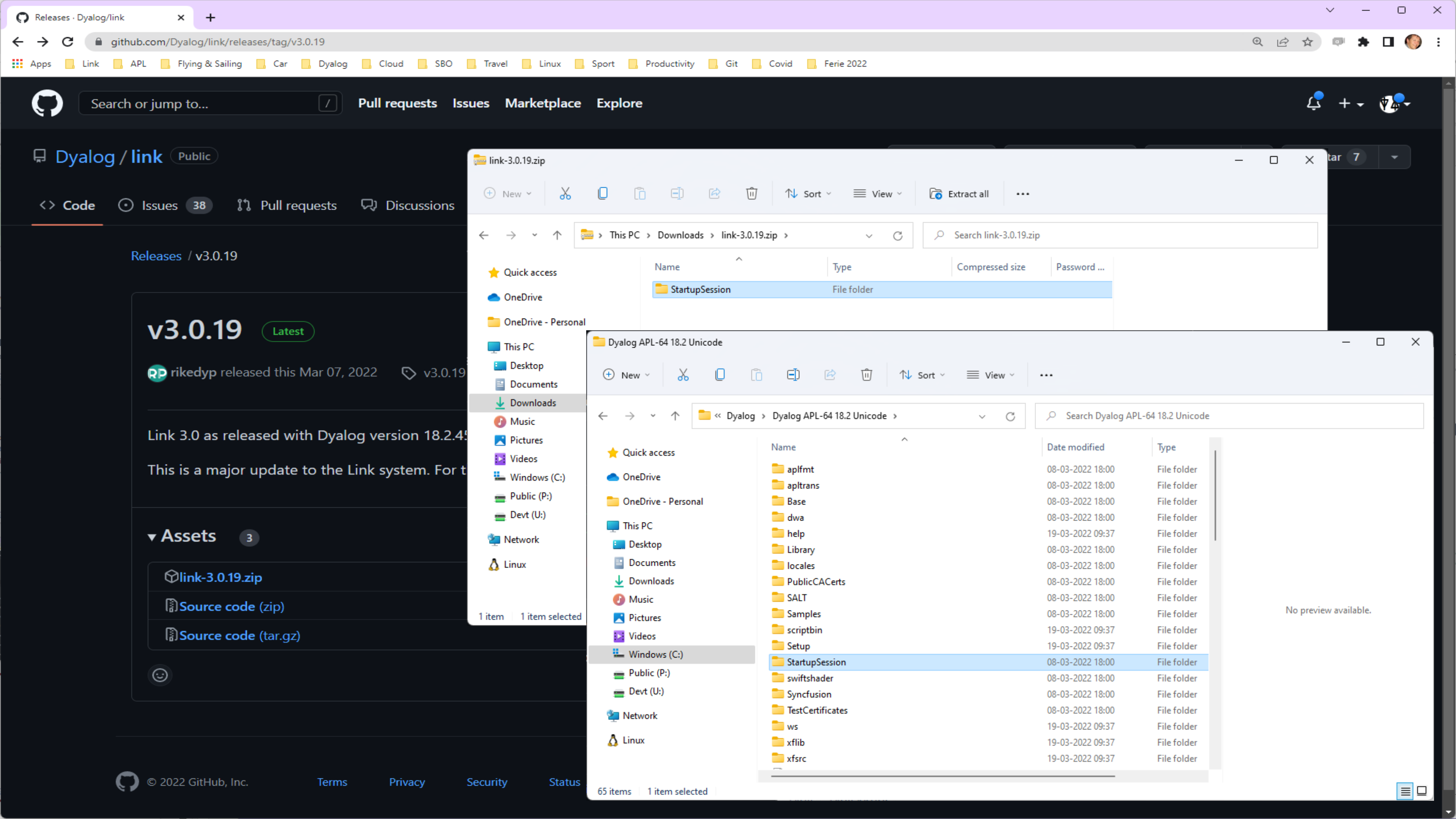
Link can miss updates when a "Revert" is done from VS Code bug
#492 opened on Sep 4 by mkromberg ASAP

]Link.Add on a namespace does not work bug
#491 opened on Aug 7 by aplteam

The API functions in FSW should return (shy) results enhancement
#490 opened on Jul 25 by abrudz

The error message when running]link.create foo where foo←ns.goo is most unhelpful bug
#489 opened on Jul 7 by dyaandys

]link.create errors when dealing with foo where foo←ns.goo bug
#488 opened on Jul 7 by dyaandys


Releases · Dyalog/link
github.com/Dyalog/link/releases/tag/v3.0.19
Apps Link APL Flying & Sailing Car Dyalog Cloud SBO Travel Linux Sport Productivity Git Covid Ferie 2022
Search or jump to...
Pull requests Issues Marketplace Explore
Dyalog / link Public
Code Issues 38 Pull requests Discussions

Releases / v3.0.19

v3.0.19 Latest

rikedyp released this Mar 07, 2022 v3.0.19

Link 3.0 as released with Dyalog version 18.2.4

This is a major update to the Link system. For t

Assets 3

- link-3.0.19.zip
- Source code (zip)
- Source code (tar.gz)

© 2022 GitHub, Inc. Terms Privacy Security Status

link-3.0.19.zip

New Sort View Extract all
This PC > Downloads > link-3.0.19.zip
Search link-3.0.19.zip
Quick access
OneDrive
OneDrive - Personal
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Public (P:)
Devt (U:)
Network
Linux

Name	Type	Compressed size	Password ...
StartupSession	File folder		

1 item 1 item selected

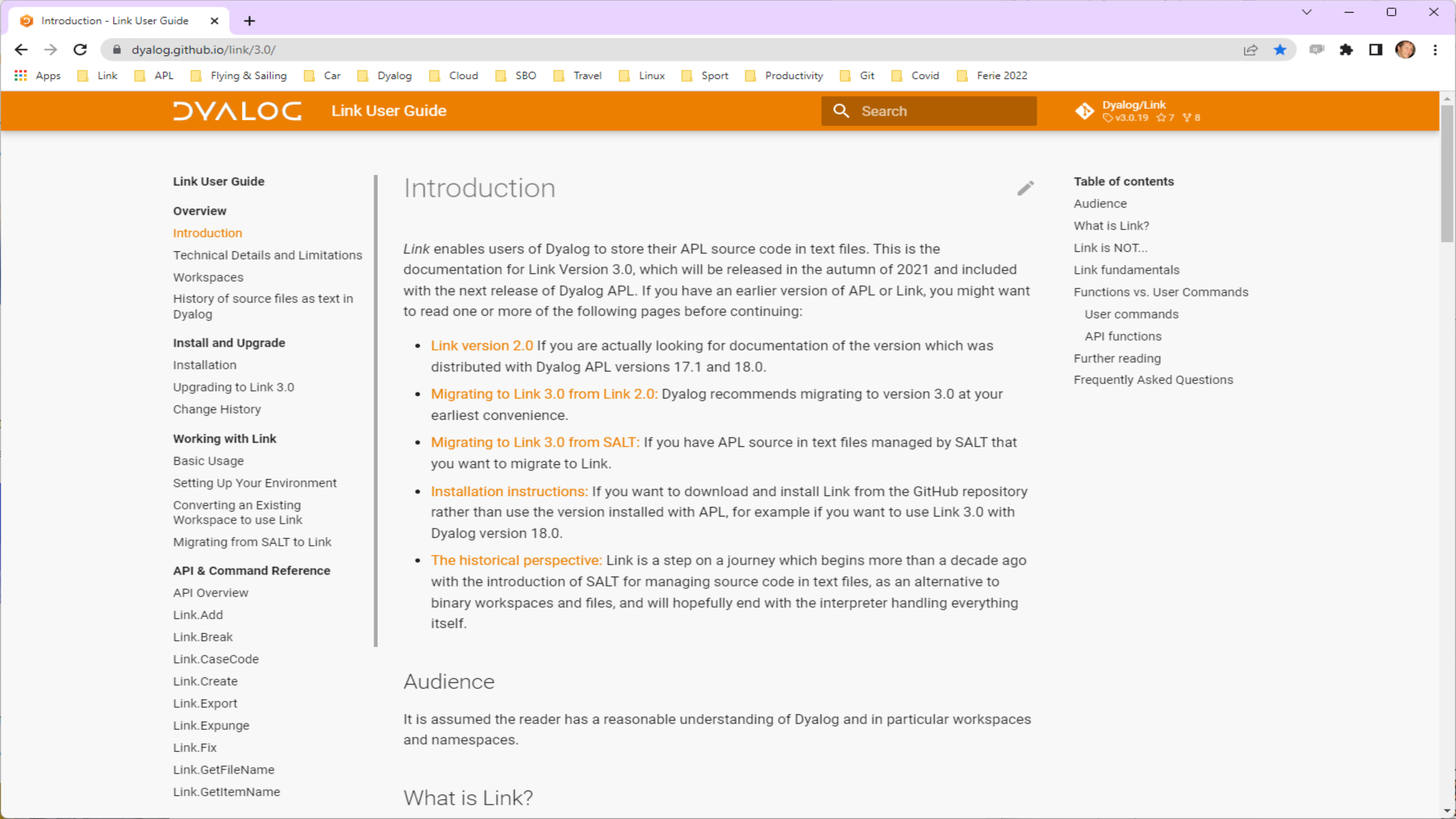
Dyalog APL-64 18.2 Unicode

New Sort View
« Dyalog > Dyalog APL-64 18.2 Unicode >
Search Dyalog APL-64 18.2 Unicode
Quick access
OneDrive
OneDrive - Personal
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Public (P:)
Devt (U:)
Network
Linux

Name	Date modified	Type
aplfmt	08-03-2022 18:00	File folder
apltrans	08-03-2022 18:00	File folder
Base	08-03-2022 18:00	File folder
dwa	08-03-2022 18:00	File folder
help	19-03-2022 09:37	File folder
Library	08-03-2022 18:00	File folder
locales	08-03-2022 18:00	File folder
PublicCACerts	08-03-2022 18:00	File folder
SALT	08-03-2022 18:00	File folder
Samples	08-03-2022 18:00	File folder
scriptbin	19-03-2022 09:37	File folder
Setup	19-03-2022 09:37	File folder
StartupSession	08-03-2022 18:00	File folder
swiftshader	08-03-2022 18:00	File folder
Syncfusion	08-03-2022 18:00	File folder
TestCertificates	08-03-2022 18:00	File folder
ws	19-03-2022 09:37	File folder
xflib	19-03-2022 09:37	File folder
xfsrc	19-03-2022 09:37	File folder

No preview available.

65 items 1 item selected



Introduction

Link enables users of Dyalog to store their APL source code in text files. This is the documentation for Link Version 3.0, which will be released in the autumn of 2021 and included with the next release of Dyalog APL. If you have an earlier version of APL or Link, you might want to read one or more of the following pages before continuing:

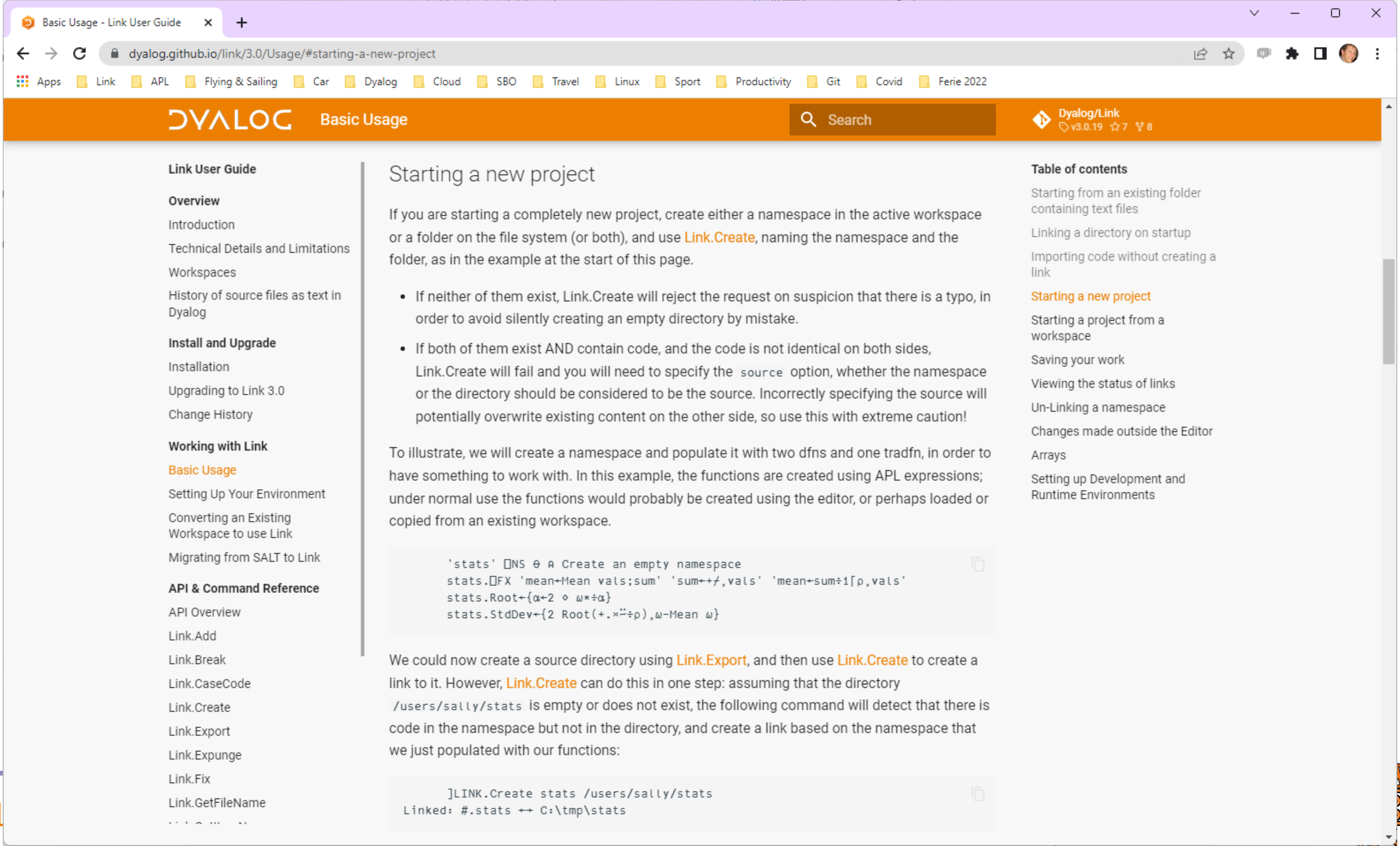
- Link version 2.0 If you are actually looking for documentation of the version which was distributed with Dyalog APL versions 17.1 and 18.0.
- Migrating to Link 3.0 from Link 2.0: Dyalog recommends migrating to version 3.0 at your earliest convenience.
- Migrating to Link 3.0 from SALT: If you have APL source in text files managed by SALT that you want to migrate to Link.
- Installation instructions: If you want to download and install Link from the GitHub repository rather than use the version installed with APL, for example if you want to use Link 3.0 with Dyalog version 18.0.
- The historical perspective: Link is a step on a journey which begins more than a decade ago with the introduction of SALT for managing source code in text files, as an alternative to binary workspaces and files, and will hopefully end with the interpreter handling everything itself.

Audience

It is assumed the reader has a reasonable understanding of Dyalog and in particular workspaces and namespaces.

What is Link?

Starting a New Project


Basic Usage - Link User Guide
dyalog.github.io/link/3.0/Usage/#starting-a-new-project
Apps Link APL Flying & Sailing Car Dyalog Cloud SBO Travel Linux Sport Productivity Git Covid Ferie 2022
DYALOG Basic Usage
Search
Dyalog/Link v3.0.19 7 8

Starting a new project

If you are starting a completely new project, create either a namespace in the active workspace or a folder on the file system (or both), and use Link.Create, naming the namespace and the folder, as in the example at the start of this page.

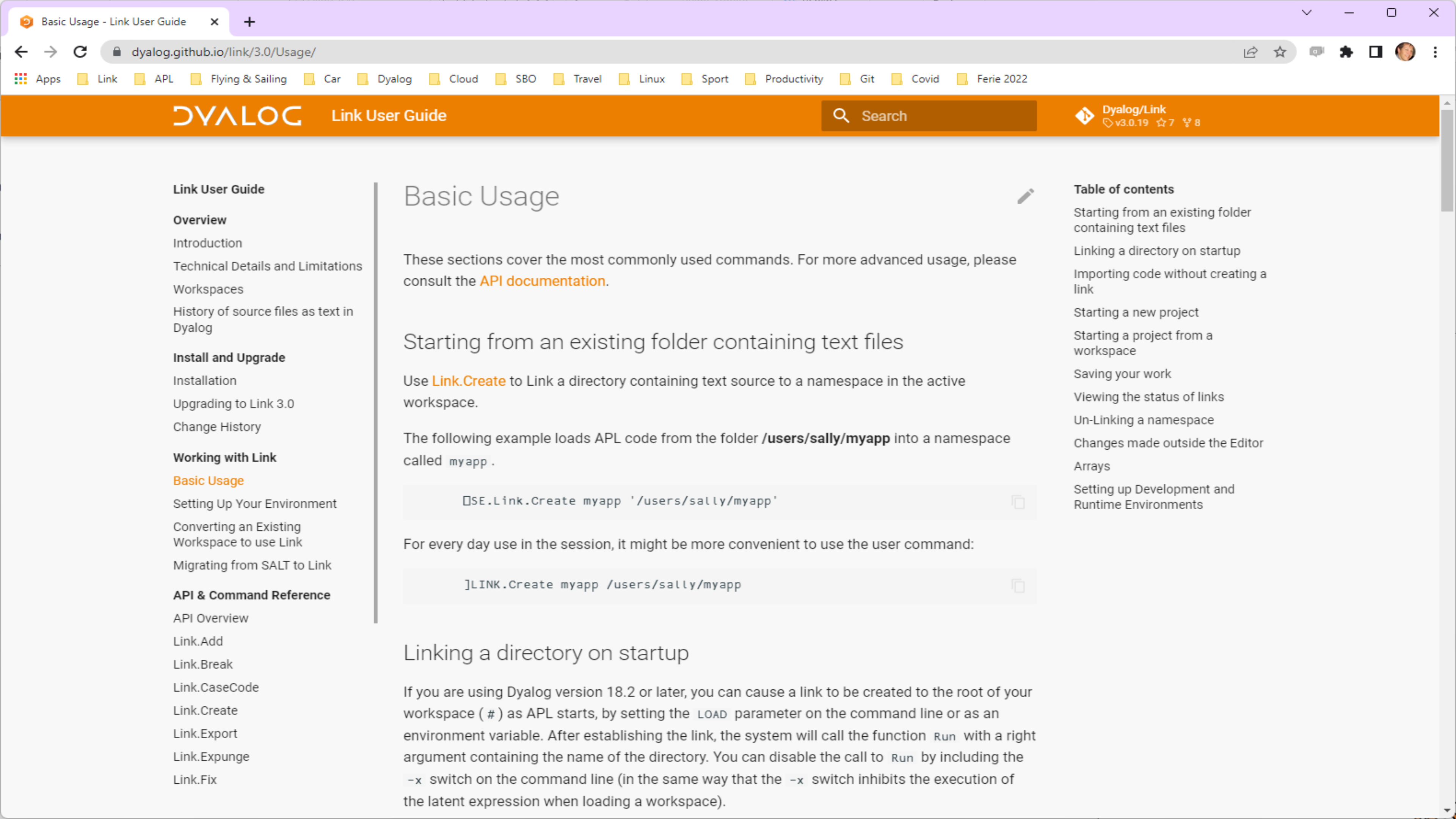
- If neither of them exist, Link.Create will reject the request on suspicion that there is a typo, in order to avoid silently creating an empty directory by mistake.
- If both of them exist AND contain code, and the code is not identical on both sides, Link.Create will fail and you will need to specify the `source` option, whether the namespace or the directory should be considered to be the source. Incorrectly specifying the source will potentially overwrite existing content on the other side, so use this with extreme caution!

To illustrate, we will create a namespace and populate it with two dfns and one tradfn, in order to have something to work with. In this example, the functions are created using APL expressions; under normal use the functions would probably be created using the editor, or perhaps loaded or copied from an existing workspace.

```
      'stats' ⎕NS ⍬ ⍝ Create an empty namespace
      stats.⎕FX 'mean←Mean vals;sum' 'sum←+⌿,vals' 'mean←sum÷1⌈⍴,vals'
      stats.Root←{⍺←2 ⋄ ⍵*÷⍺}
      stats.StdDev←{2 Root(+.×⍨÷⍴),⍵-Mean ⍵}
```

We could now create a source directory using Link.Export, and then use Link.Create to create a link to it. However, Link.Create can do this in one step: assuming that the directory `/users/sally/stats` is empty or does not exist, the following command will detect that there is code in the namespace but not in the directory, and create a link based on the namespace that we just populated with our functions:

```
      ]LINK.Create stats /users/sally/stats
Linked: #.stats ←→ C:\tmp\stats
```



Basic Usage

These sections cover the most commonly used commands. For more advanced usage, please consult the API documentation.

Starting from an existing folder containing text files

Use Link.Create to Link a directory containing text source to a namespace in the active workspace.

The following example loads APL code from the folder **/users/sally/myapp** into a namespace called `myapp`.

```
      ⎕SE.Link.Create myapp '/users/sally/myapp'
```

For every day use in the session, it might be more convenient to use the user command:

```
      ]LINK.Create myapp /users/sally/myapp
```

Linking a directory on startup

If you are using Dyalog version 18.2 or later, you can cause a link to be created to the root of your workspace (`#`) as APL starts, by setting the `LOAD` parameter on the command line or as an environment variable. After establishing the link, the system will call the function `Run` with a right argument containing the name of the directory. You can disable the call to `Run` by including the `-x` switch on the command line (in the same way that the `-x` switch inhibits the execution of the latent expression when loading a workspace).

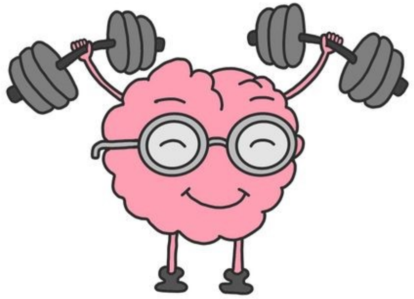
Starting a New Project

- To start a new project

```
      )ns myns
      ]link.create myns /my/dir
```

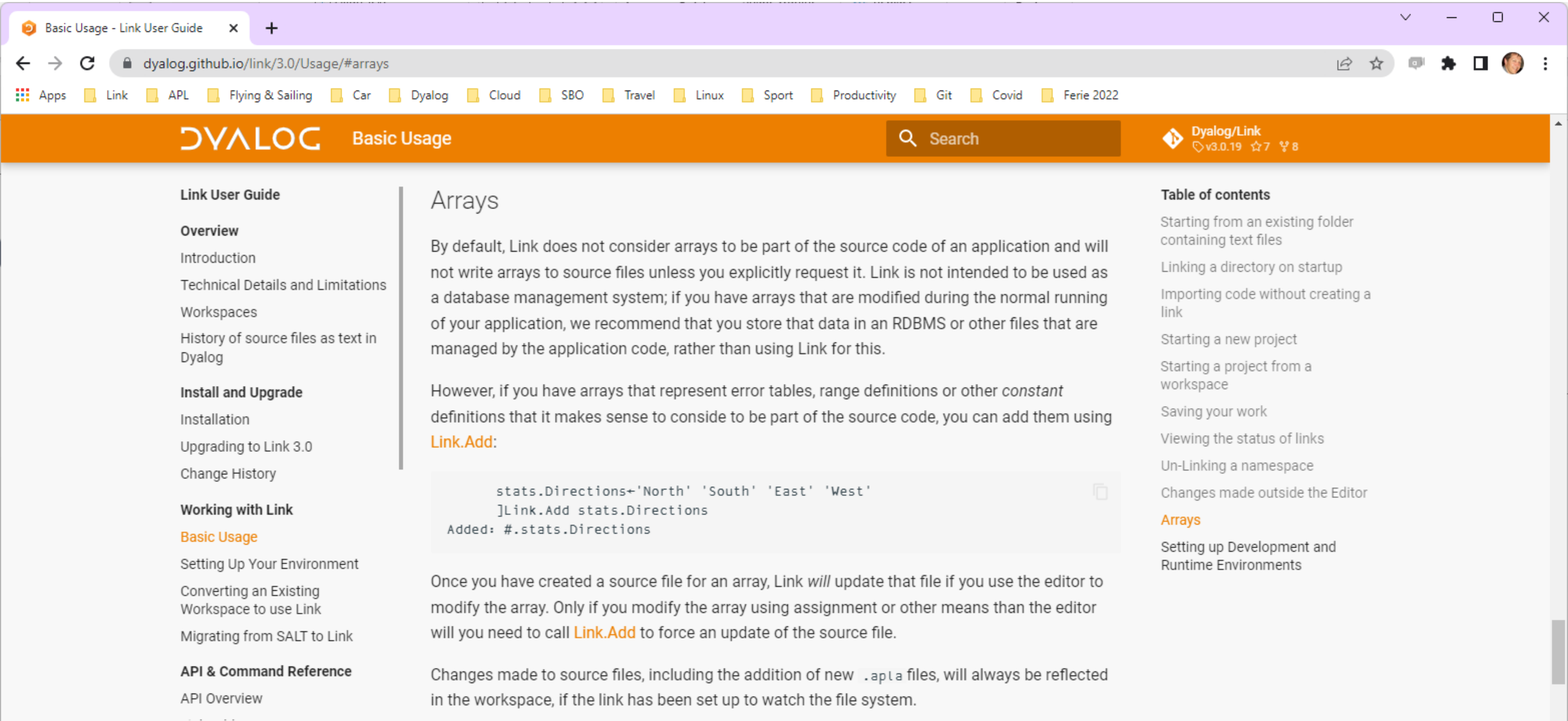
- At least one of `myns` or `/my/dir` must exist
- Only one of `myns` or `/my/dir` may be populated

Exercise 1



- Create an empty namespace
- Create a link to a directory name which does not already exist
- `)ED` a function in the namespace
- Verify that a file is created in the directory
- Edit the file using notepad or another external editor
- Verify that the function is updated in the WS
-)CLEAR, and re-create the link
- Verify that your code is loaded

Variables


Basic Usage - Link User Guide
dyalog.github.io/link/3.0/Usage/#arrays
Apps
Link
APL
Flying & Sailing
Car
Dyalog
Cloud
SBO
Travel
Linux
Sport
Productivity
Git
Covid
Ferie 2022
DYALOG
Basic Usage
Search
Dyalog/Link
v3.0.19 7 8
Link User Guide
Overview
Introduction
Technical Details and Limitations
Workspaces
History of source files as text in Dyalog
Install and Upgrade
Installation
Upgrading to Link 3.0
Change History
Working with Link
Basic Usage
Setting Up Your Environment
Converting an Existing Workspace to use Link
Migrating from SALT to Link
API & Command Reference
API Overview
Table of contents
Starting from an existing folder containing text files
Linking a directory on startup
Importing code without creating a link
Starting a new project
Starting a project from a workspace
Saving your work
Viewing the status of links
Un-Linking a namespace
Changes made outside the Editor
Arrays
Setting up Development and Runtime Environments

Arrays

By default, Link does not consider arrays to be part of the source code of an application and will not write arrays to source files unless you explicitly request it. Link is not intended to be used as a database management system; if you have arrays that are modified during the normal running of your application, we recommend that you store that data in an RDBMS or other files that are managed by the application code, rather than using Link for this.

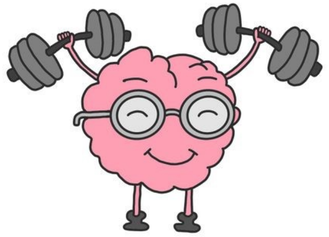
However, if you have arrays that represent error tables, range definitions or other *constant* definitions that it makes sense to conside to be part of the source code, you can add them using Link.Add:

```
      stats.Directions←'North' 'South' 'East' 'West'
      ]Link.Add stats.Directions
Added: #.stats.Directions
```

Once you have created a source file for an array, Link *will* update that file if you use the editor to modify the array. Only if you modify the array using assignment or other means than the editor will you need to call Link.Add to force an update of the source file.

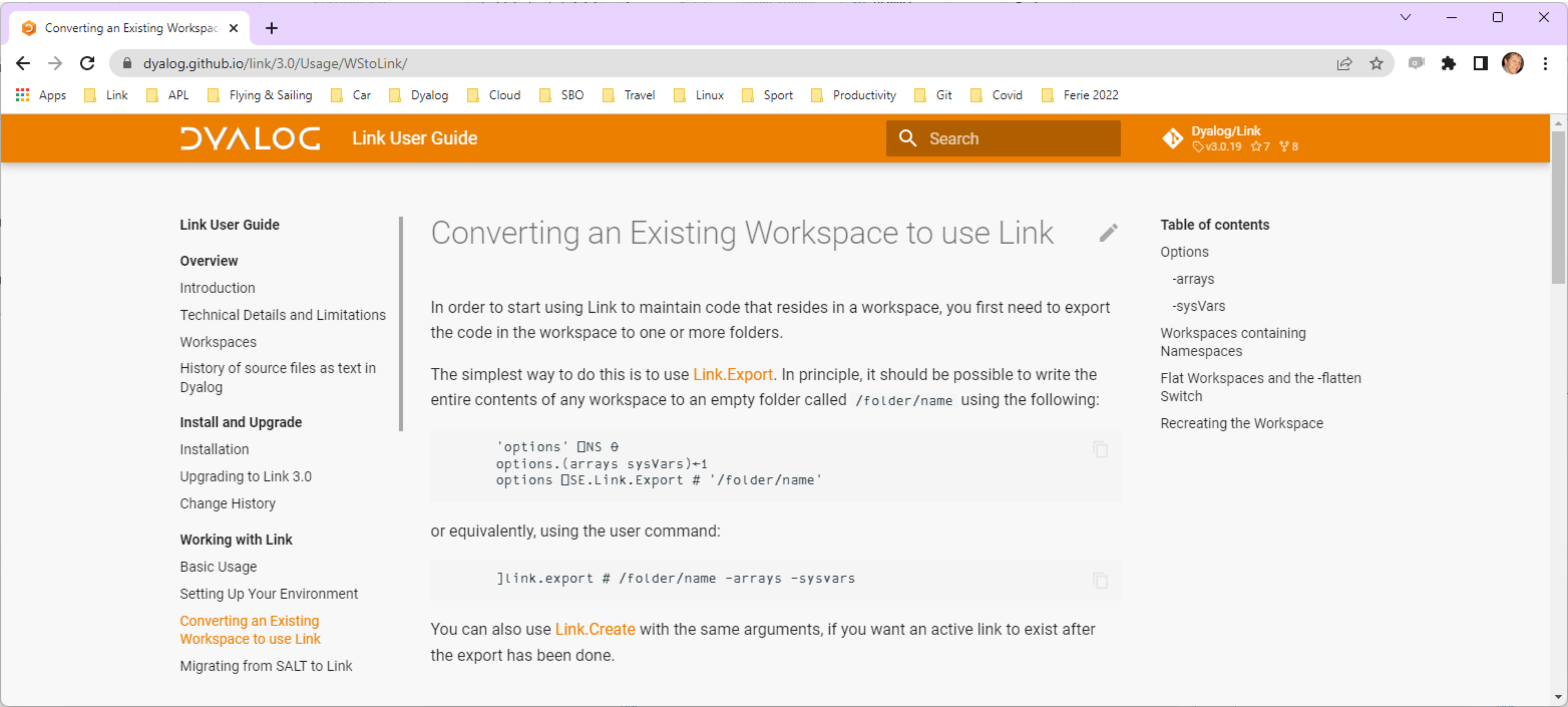
Changes made to source files, including the addition of new `.apla` files, will always be reflected in the workspace, if the link has been set up to watch the file system.

Exercise 2



- Create a variable containing a constant that your application needs
- Cause it to be written to a file using `]Link.Add`
- Inspect the file, edit it, and verify that the new value appears in the workspace
- Can you write system variables to file?

Converting an Existing Workspace



Converting an Existing Workspace to use Link

In order to start using Link to maintain code that resides in a workspace, you first need to export the code in the workspace to one or more folders.

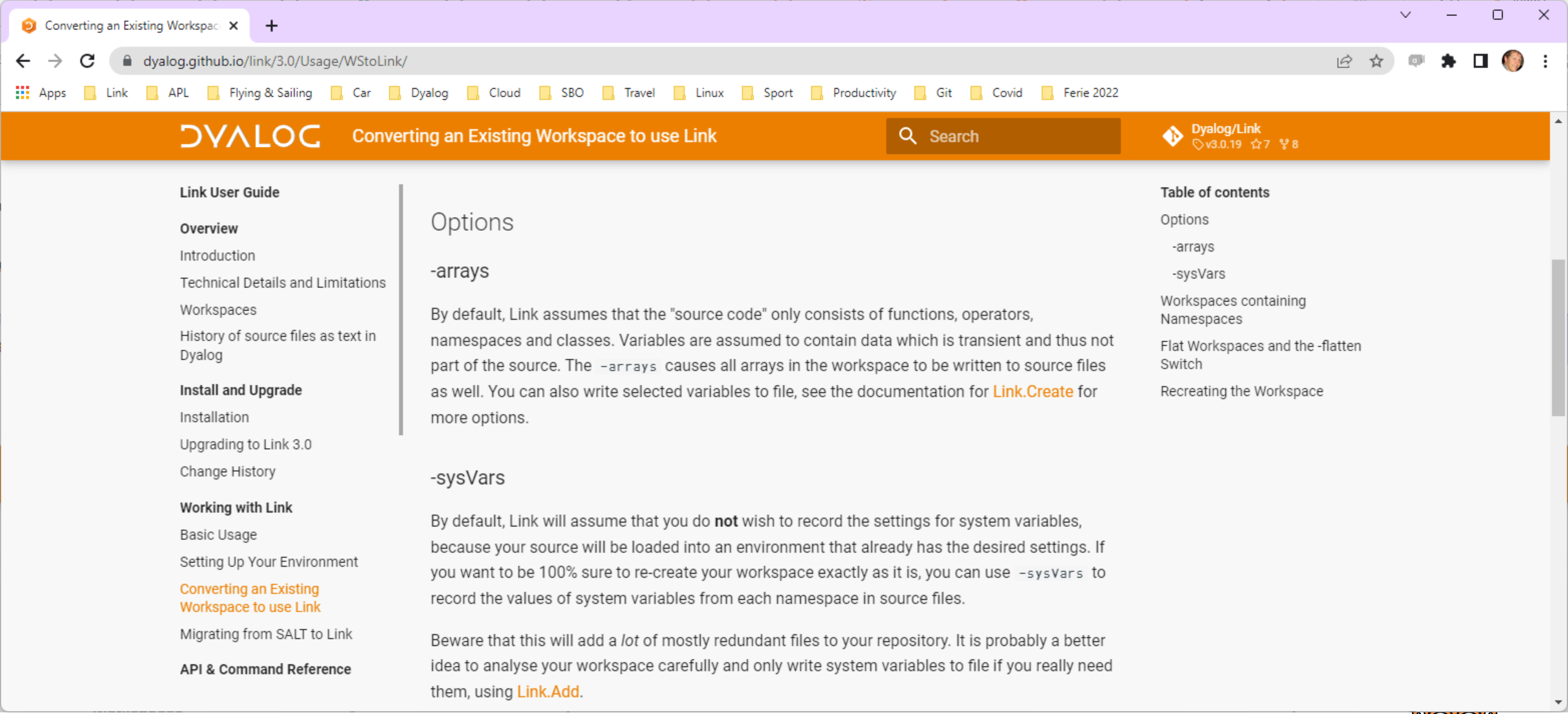
The simplest way to do this is to use Link.Export. In principle, it should be possible to write the entire contents of any workspace to an empty folder called `/folder/name` using the following:

```
      'options' ⎕NS ⍬
      options.(arrays sysVars)←1
      options ⎕SE.Link.Export # '/folder/name'
```

or equivalently, using the user command:

```
      ]link.export # /folder/name -arrays -sysvars
```

You can also use Link.Create with the same arguments, if you want an active link to exist after the export has been done.



Converting an Existing Workspace to use Link

dyalog.github.io/link/3.0/Usage/WStoLink/

Options

-arrays

By default, Link assumes that the "source code" only consists of functions, operators, namespaces and classes. Variables are assumed to contain data which is transient and thus not part of the source. The `-arrays` causes all arrays in the workspace to be written to source files as well. You can also write selected variables to file, see the documentation for Link.Create for more options.

-sysVars

By default, Link will assume that you do **not** wish to record the settings for system variables, because your source will be loaded into an environment that already has the desired settings. If you want to be 100% sure to re-create your workspace exactly as it is, you can use `-sysVars` to record the values of system variables from each namespace in source files.

Beware that this will add a *lot* of mostly redundant files to your repository. It is probably a better idea to analyse your workspace carefully and only write system variables to file if you really need them, using Link.Add.

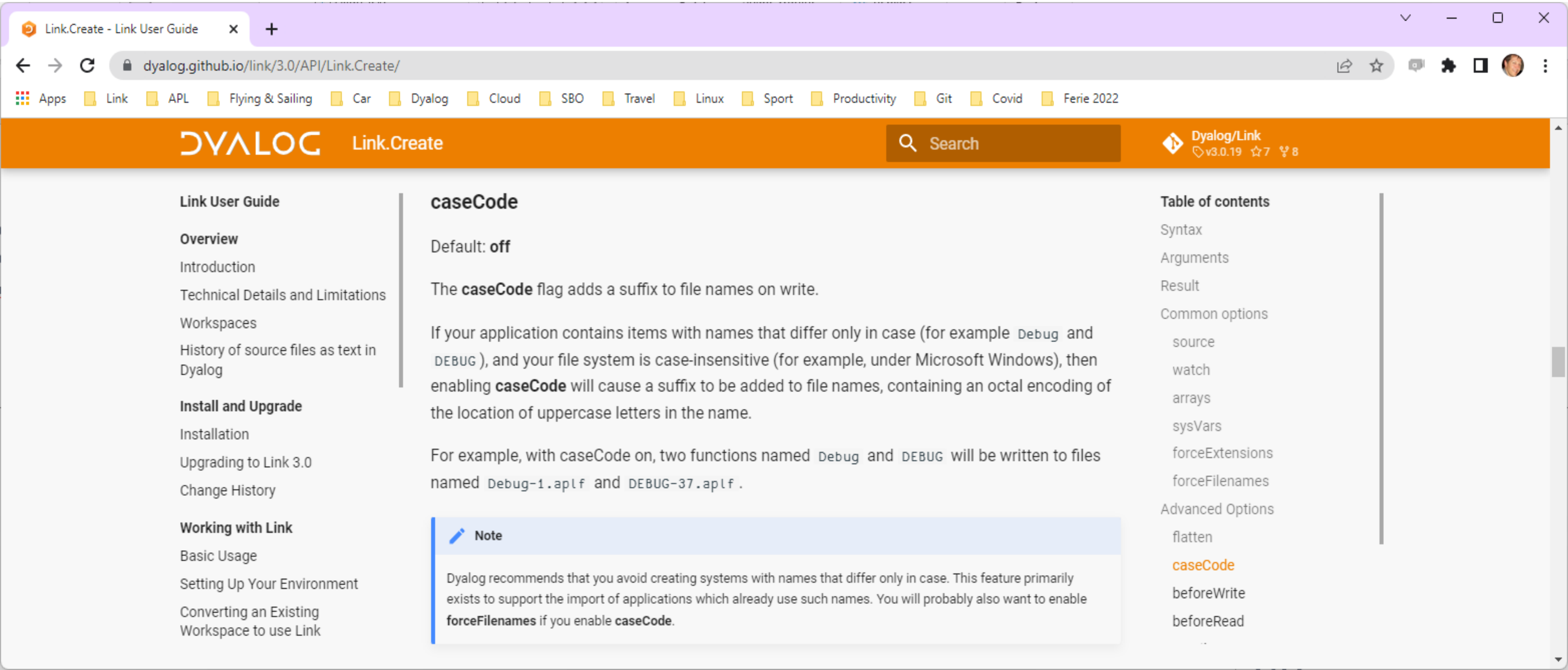
Namespace ← → Folder

- Each function, operator or array is linked to a file
- Each namespace links to a directory
- If an exported namespace contains sub-namespaces
  - Each one becomes a sub-directory
- If an imported directory contains subdirectories
  - Each one becomes a namespace

Exercise 2.5

- Create a subdirectory in your source folder
- Verify that a corresponding namespace is created in the workspace
-)ED a function in the namespace
- Rename the subdirectory
- Verify the effect in the workspace

caseCode



Link.Create - Link User Guide

dyalog.github.io/link/3.0/API/Link.Create/

caseCode

Default: **off**

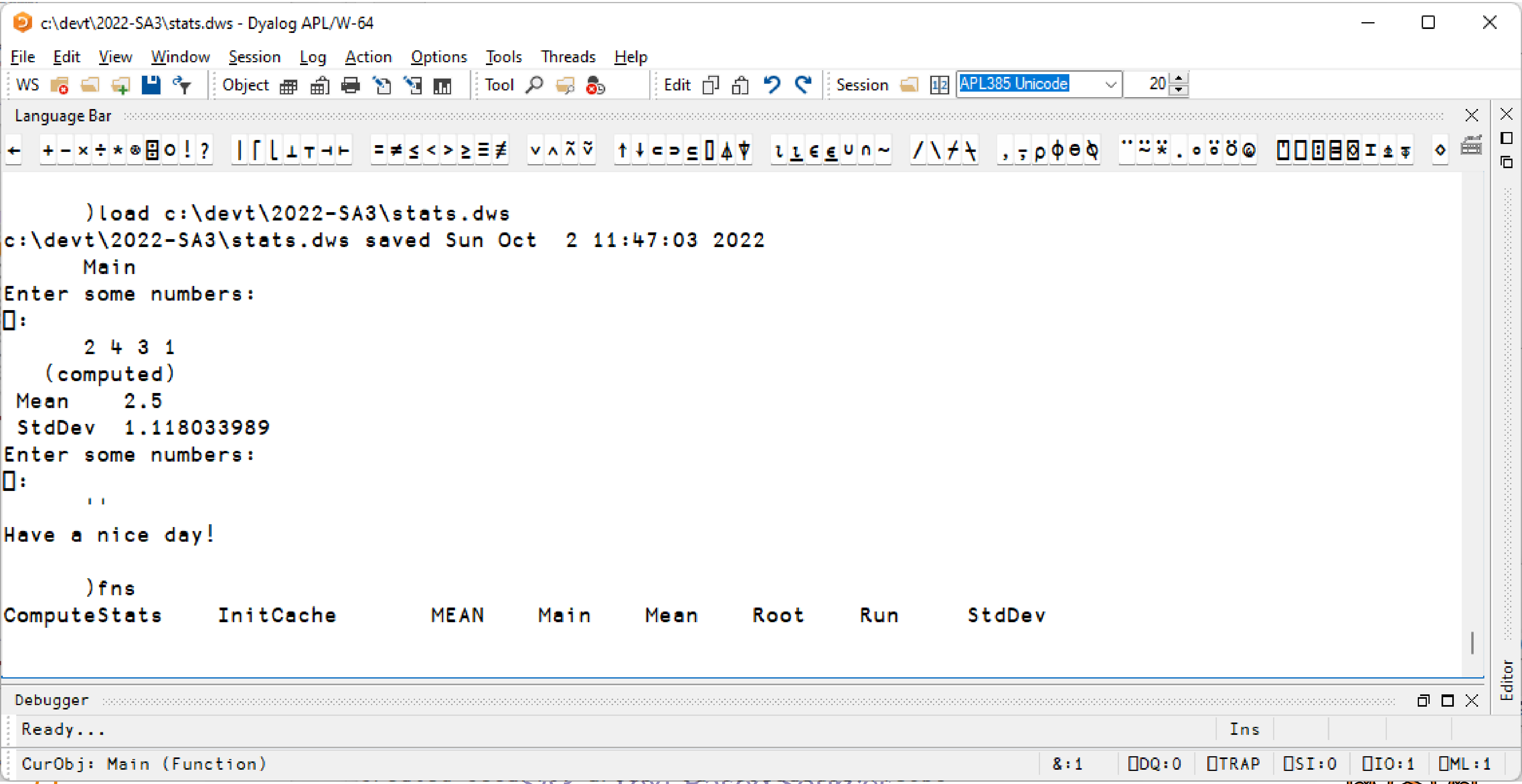
The **caseCode** flag adds a suffix to file names on write.

If your application contains items with names that differ only in case (for example `Debug` and `DEBUG`), and your file system is case-insensitive (for example, under Microsoft Windows), then enabling **caseCode** will cause a suffix to be added to file names, containing an octal encoding of the location of uppercase letters in the name.

For example, with caseCode on, two functions named `Debug` and `DEBUG` will be written to files named `Debug-1.aplf` and `DEBUG-37.aplf`.

Note

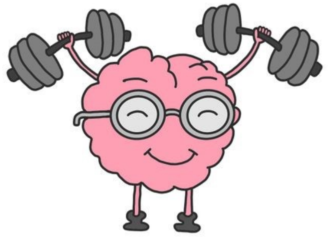
Dyalog recommends that you avoid creating systems with names that differ only in case. This feature primarily exists to support the import of applications which already use such names. You will probably also want to enable **forceFilenames** if you enable **caseCode**.


c:\devt\2022-SA3\stats.dws - Dyalog APL/W-64
File Edit View Window Session Log Action Options Tools Threads Help
WS Object Tool Edit Session APL385 Unicode 20
Language Bar
Debugger
Editor

```
      )load c:\devt\2022-SA3\stats.dws
c:\devt\2022-SA3\stats.dws saved Sun Oct  2 11:47:03 2022
      Main
Enter some numbers:
⎕:
      2 4 3 1
   (computed)
 Mean     2.5
 StdDev   1.118033989
Enter some numbers:
⎕:
      ''
Have a nice day!

      )fns
ComputeStats    InitCache       MEAN    Main    Mean    Root    Run     StdDev
```

```
Ready...                                                         Ins
CurObj: Main (Function)                    &:1   ⎕DQ:0  ⎕TRAP  ⎕SI:0  ⎕IO:1  ⎕ML:1
```

Exercise 3



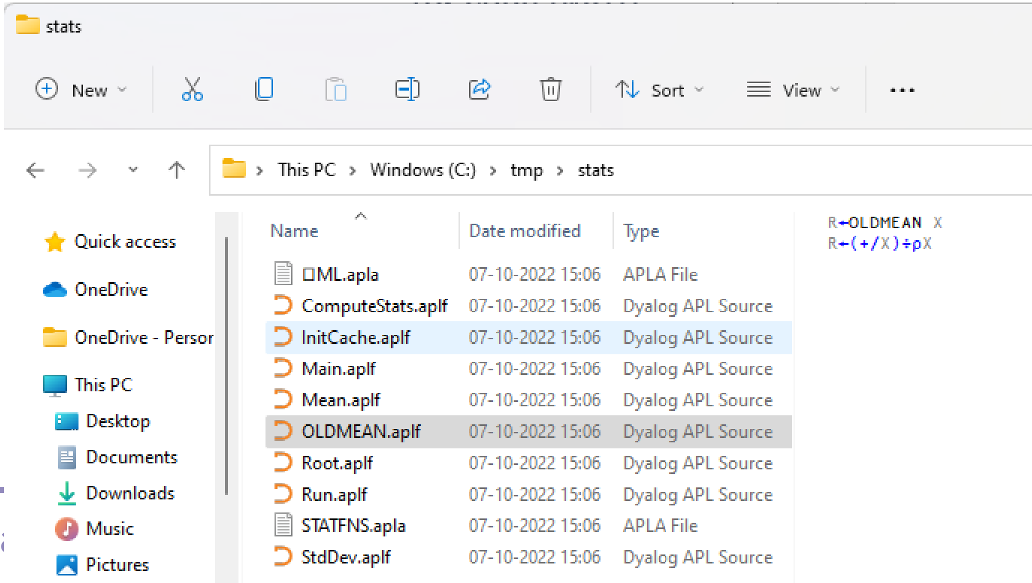
- Export the workspace `stats.dws` to a directory
- Note that
  - It contains two variables
  - Has a non-default ⎕ML
  - Has two names which differ only in case

Exercise 3 - Discussion

- Use -caseCode or rename?
- Which variables should be considered "source"
- ⎕ML=3:
  - Use -sysvars,]link.add ⎕ML, or refactor

Exercise 3 – Morten's Solution

```
      )load c:\devt\2022-SA3\stats
c:\devt\2022-SA3\stats.dws saved Tue Oct 4 22:59:08 2022
      )fns
ComputeStats InitCache MEAN Main Mean Root Run StdDev
      )ed MEAN ⍝ rename to OLDMEAN
      )erase MEAN
      )vars
RESULTS STATFNS
      ]link.export # c:\tmp\stats
Exported: # → c:\tmp\stats
      ]link.export ⎕ML c:\tmp\stats
Exported: #.⎕ML → c:/tmp/stats/⎕ML.apla
      ]link.export STATFNS c:\tmp\stats
Exported: #.STATFNS → c:/tmp/stats/STATFNS.apla
```



Name	Date modified	Type
⎕ML.apla	07-10-2022 15:06	APLA File
ComputeStats.aplf	07-10-2022 15:06	Dyalog APL Source
InitCache.aplf	07-10-2022 15:06	Dyalog APL Source
Main.aplf	07-10-2022 15:06	Dyalog APL Source
Mean.aplf	07-10-2022 15:06	Dyalog APL Source
OLDMEAN.aplf	07-10-2022 15:06	Dyalog APL Source
Root.aplf	07-10-2022 15:06	Dyalog APL Source
Run.aplf	07-10-2022 15:06	Dyalog APL Source
STATFNS.apla	07-10-2022 15:06	APLA File
StdDev.aplf	07-10-2022 15:06	Dyalog APL Source

```
R←OLDMEAN X
R←(+/X)÷⍴X
```

Non-Representable Objects

- Some objects that CAN be saved in a Dyalog workspace have no meaningful textual representation
  - GUI & COM objects
- It was already a questionable practice to save such "binary" objects, they cannot be transferred between 32/64 or classic/unicode.
- You need to write code which creates these objects at run or build time

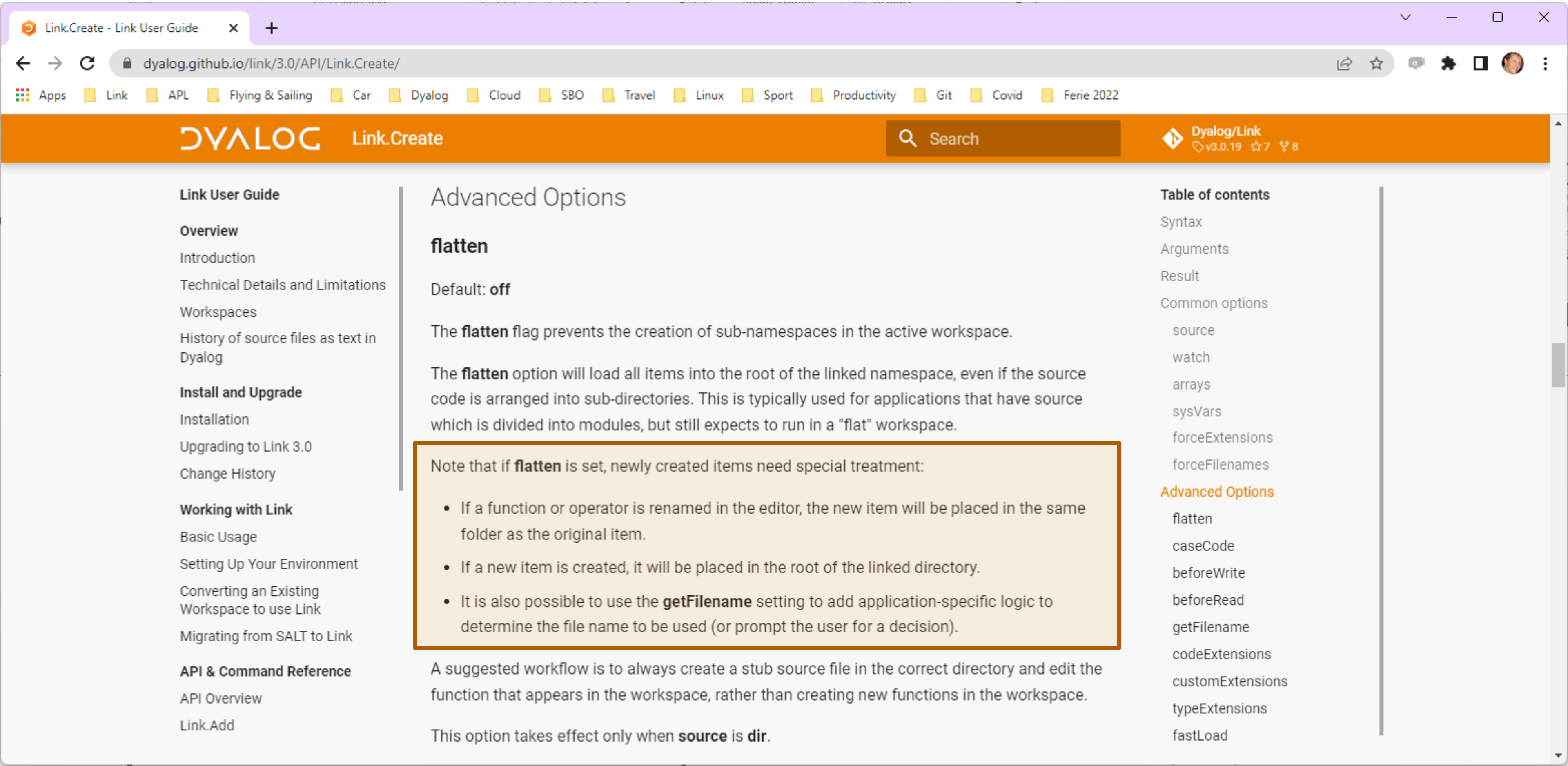
Create an OLE Server…

Example of explict creation of an otherwise un-saveable object:

```
     ∇ R←MakeOLEServer;ns;spec
[1]    ⍝ Recreate the OLE Server before WS is built
[2]     ns←#.StatsServer
[3]     ns.⎕WC'OleServer'('ClassID' '{395E64DF-6B44-4515-B409-6A0A2E1ACD9B}')
                         ('RunMode' 'SingleUse')
[4]     spec←⊂'This function returns the mean' 'VT_VARIANT'
[5]     spec,←⊂'InputNumbers' 'VT_VARIANT'
[6]     ns.SetFnInfo'Mean'spec
[7]     R←0
     ∇
```

The `-flatten` switch

- Even if your "legacy" workspace is "flat"...
- It may contain modules that can benefit from being organised into separate directories
- The `-flatten` allows you to load code organised into directories, into a flat workspace
- The link between individual functions and files is maintained


Link.Create - Link User Guide
dyalog.github.io/link/3.0/API/Link.Create/
Apps Link APL Flying & Sailing Car Dyalog Cloud SBO Travel Linux Sport Productivity Git Covid Ferie 2022
DYALOG Link.Create
Search
Dyalog/Link v3.0.19 7 8

Advanced Options

flatten

Default: **off**

The **flatten** flag prevents the creation of sub-namespaces in the active workspace.

The **flatten** option will load all items into the root of the linked namespace, even if the source code is arranged into sub-directories. This is typically used for applications that have source which is divided into modules, but still expects to run in a "flat" workspace.

Note that if **flatten** is set, newly created items need special treatment:

- If a function or operator is renamed in the editor, the new item will be placed in the same folder as the original item.
- If a new item is created, it will be placed in the root of the linked directory.
- It is also possible to use the **getFilename** setting to add application-specific logic to determine the file name to be used (or prompt the user for a decision).

A suggested workflow is to always create a stub source file in the correct directory and edit the function that appears in the workspace, rather than creating new functions in the workspace.

This option takes effect only when **source** is **dir**.

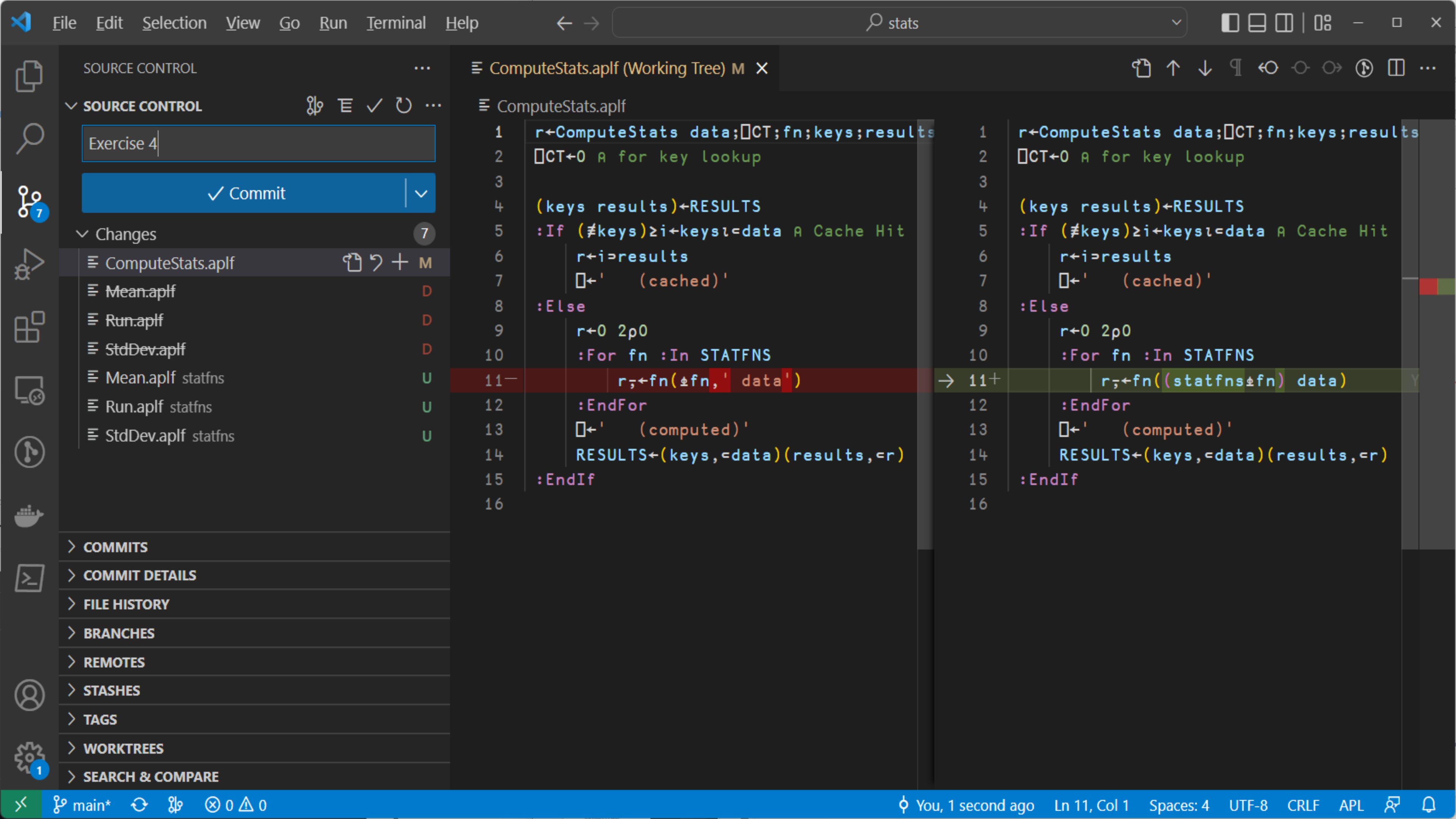
SA3 – Text Based Sources

Exercise 4

- Move the source files for the statistical functions (Mean, StdDev, Root) to a sub-directory called "statfns"
- Get the application to run again

Exercise 4 - Discussion

- `-flatten` or refactor?

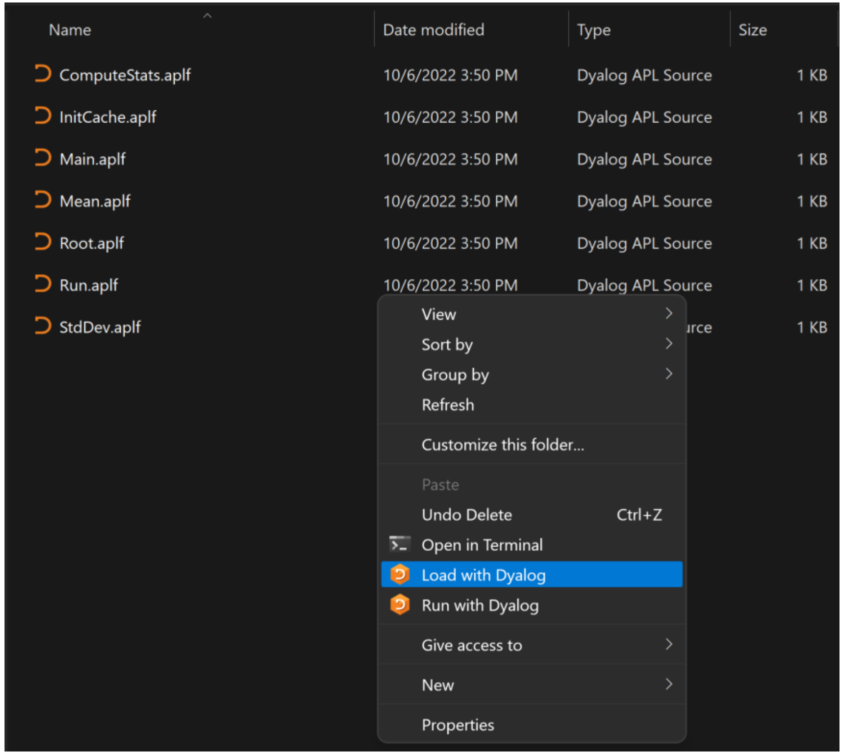


```
r←ComputeStats data;⎕CT;fn;keys;results
⎕CT←0 ⍝ for key lookup

(keys results)←RESULTS
:If (≢keys)≥i←keys⍳⊂data ⍝ Cache Hit
    r←i⊃results
    ⎕←'   (cached)'
:Else
    r←0 2⍴0
    :For fn :In STATFNS
−       r⍪←fn(⍎fn,' data')
    :EndFor
    ⎕←'   (computed)'
    RESULTS←(keys,⊂data)(results,⊂r)
:EndIf
```

```
r←ComputeStats data;⎕CT;fn;keys;results
⎕CT←0 ⍝ for key lookup

(keys results)←RESULTS
:If (≢keys)≥i←keys⍳⊂data ⍝ Cache Hit
    r←i⊃results
    ⎕←'   (cached)'
:Else
    r←0 2⍴0
    :For fn :In STATFNS
+       r⍪←fn((statfns⍎fn) data)
    :EndFor
    ⎕←'   (computed)'
    RESULTS←(keys,⊂data)(results,⊂r)
:EndIf
```

Launch from Source

- Right click in the file explorer
  - "Load with Dyalog" will do a Link.Create on a selected folder, or import a selected file
  - "Run with Dyalog" will look for a function called Run and invoke it if it exists after the link has been created.
-]FileAssociations can be used to select APL version
  - 18.2 Unicode required



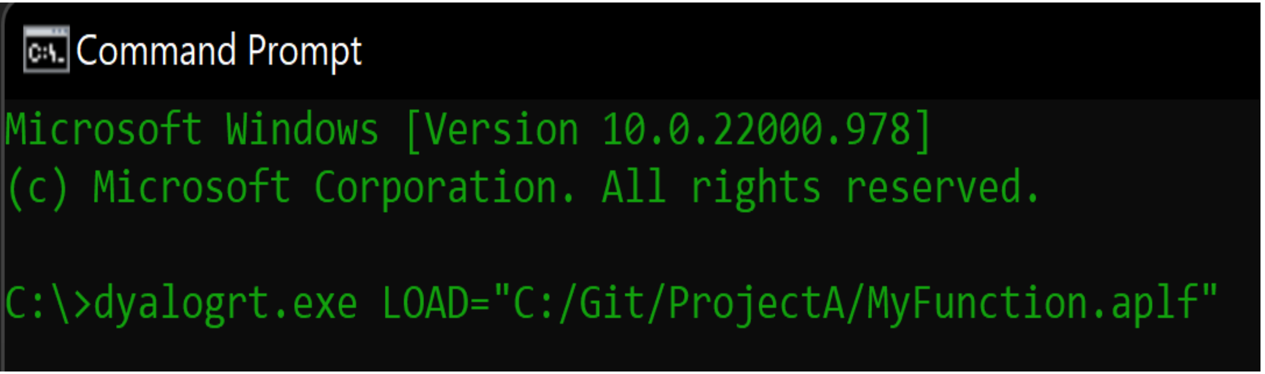
Name	Date modified	Type	Size
ComputeStats.aplf	10/6/2022 3:50 PM	Dyalog APL Source	1 KB
InitCache.aplf	10/6/2022 3:50 PM	Dyalog APL Source	1 KB
Main.aplf	10/6/2022 3:50 PM	Dyalog APL Source	1 KB
Mean.aplf	10/6/2022 3:50 PM	Dyalog APL Source	1 KB
Root.aplf	10/6/2022 3:50 PM	Dyalog APL Source	1 KB
Run.aplf	10/6/2022 3:50 PM	Dyalog APL Source	1 KB
StdDev.aplf		urce	1 KB

LOAD

- Point to a file, or a directory
- Can be specified on the command line, or in a .dcfg file
- Add –x to disable startup (just setting LOAD is actually equivalent to "Run with Dyalog")

LOAD

- Point to a file, or a directory
- Can be specified on the command line, or in a .dcfg file
- Add –x to disable startup (just setting LOAD is actually equivalent to "Run with Dyalog")



```
Microsoft Windows [Version 10.0.22000.978]
(c) Microsoft Corporation. All rights reserved.

C:\>dyalogrt.exe LOAD="C:/Git/ProjectA/MyFunction.aplf"
```

Example .dcfg file:

```
{
  Settings: {
        AutoPW: 1,
        MaxWS: "512M",
        DadoProjectsFolder: "C:/Git",
        PropertyExposeRoot: 1,
        LOAD: "C:/Git/stats"
        }
}
```

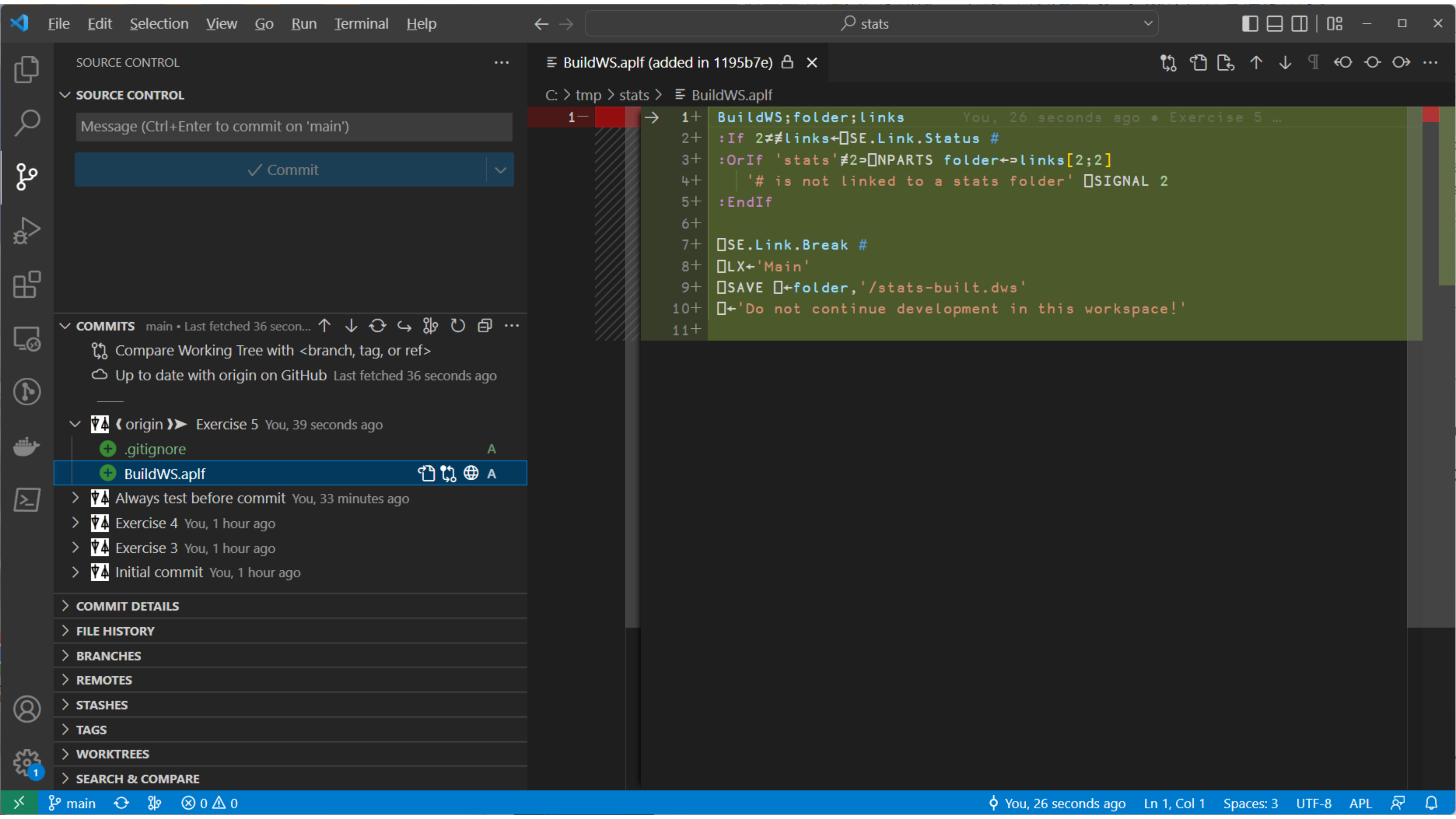
Boot or Build?

- It is fine (even encouraged!) to dynamically load text source during development
- It is NOT recommended to dynamically load source from large numbers of text files in production environments
- Break links and)SAVE to build a workspace before production use

Exercise 5

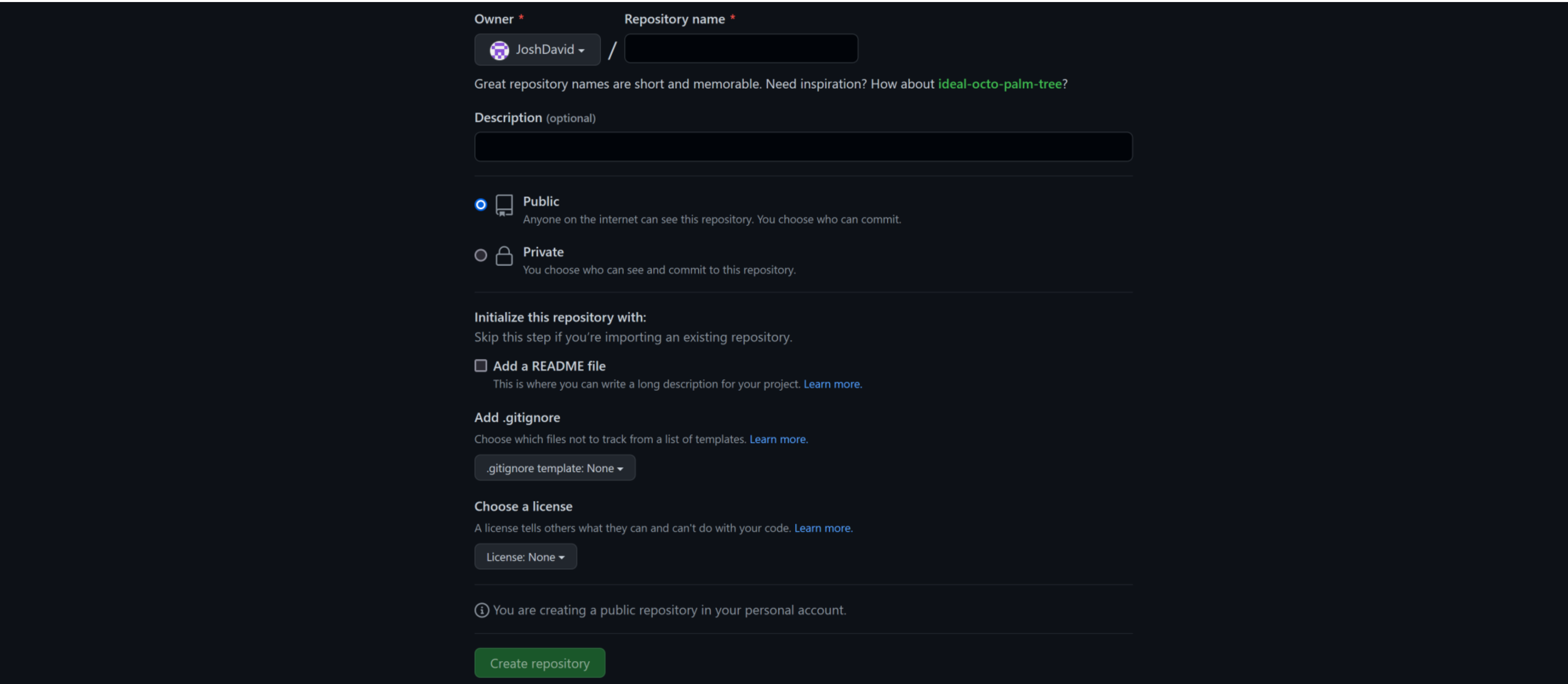
- Write a "Build" function

Saved Workspaces with Links


File Edit Selection View Go Run Terminal Help
stats
SOURCE CONTROL
SOURCE CONTROL
Message (Ctrl+Enter to commit on 'main')
✓ Commit
COMMITS main • Last fetched 36 secon...
Compare Working Tree with <branch, tag, or ref>
Up to date with origin on GitHub Last fetched 36 seconds ago
❪ origin ❫➤ Exercise 5 You, 39 seconds ago
.gitignore A
BuildWS.aplf A
Always test before commit You, 33 minutes ago
Exercise 4 You, 1 hour ago
Exercise 3 You, 1 hour ago
Initial commit You, 1 hour ago
COMMIT DETAILS
FILE HISTORY
BRANCHES
REMOTES
STASHES
TAGS
WORKTREES
SEARCH & COMPARE
BuildWS.aplf (added in 1195b7e)
C: > tmp > stats > BuildWS.aplf
You, 26 seconds ago • Exercise 5 …
main
⊗ 0 ⚠ 0
You, 26 seconds ago Ln 1, Col 1 Spaces: 3 UTF-8 APL

```
BuildWS;folder;links
:If 2≠≢links←⎕SE.Link.Status #
:OrIf 'stats'≢2⊃⎕NPARTS folder←⊃links[2;2]
    '# is not linked to a stats folder' ⎕SIGNAL 2
:EndIf

⎕SE.Link.Break #
⎕LX←'Main'
⎕SAVE ⎕←folder,'/stats-built.dws'
⎕←'Do not continue development in this workspace!'

```

Creating a GitHub Repository



Owner * JoshDavid / Repository name *

Great repository names are short and memorable. Need inspiration? How about ideal-octo-palm-tree?

Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

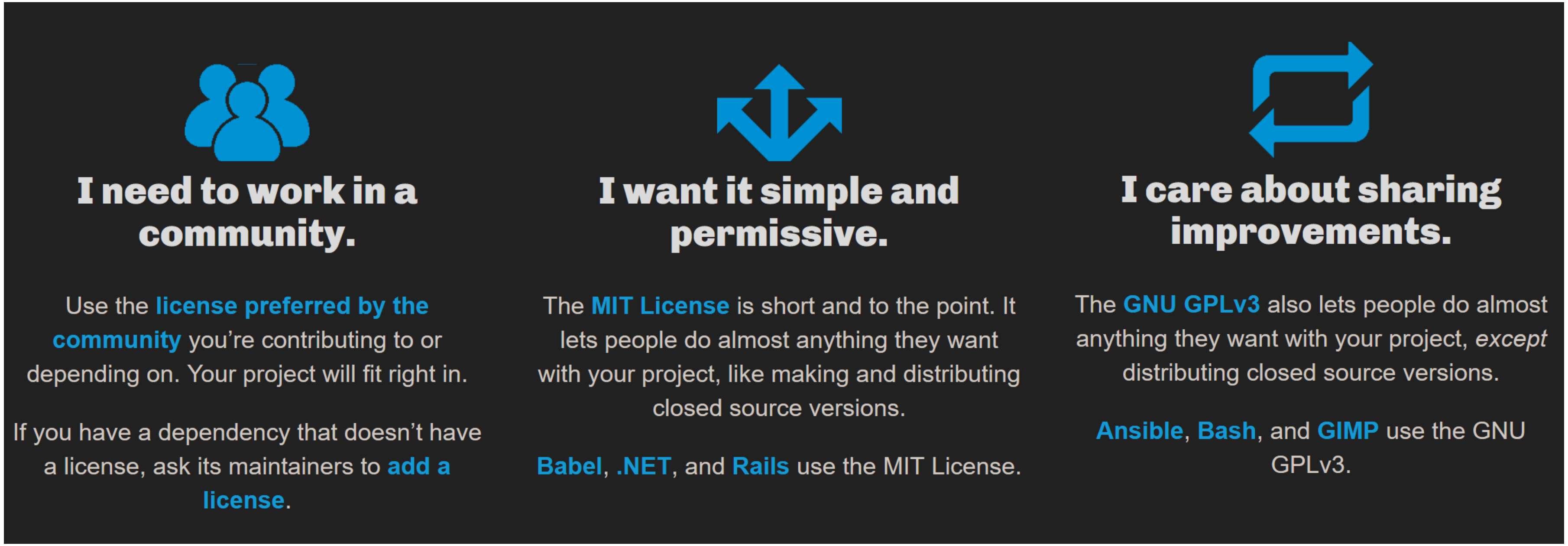
.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.

License: None

You are creating a public repository in your personal account.

Picking a License



I need to work in a community.

Use the license preferred by the community you're contributing to or depending on. Your project will fit right in.

If you have a dependency that doesn't have a license, ask its maintainers to add a license.

I want it simple and permissive.

The MIT License is short and to the point. It lets people do almost anything they want with your project, like making and distributing closed source versions.

Babel, .NET, and Rails use the MIT License.

I care about sharing improvements.

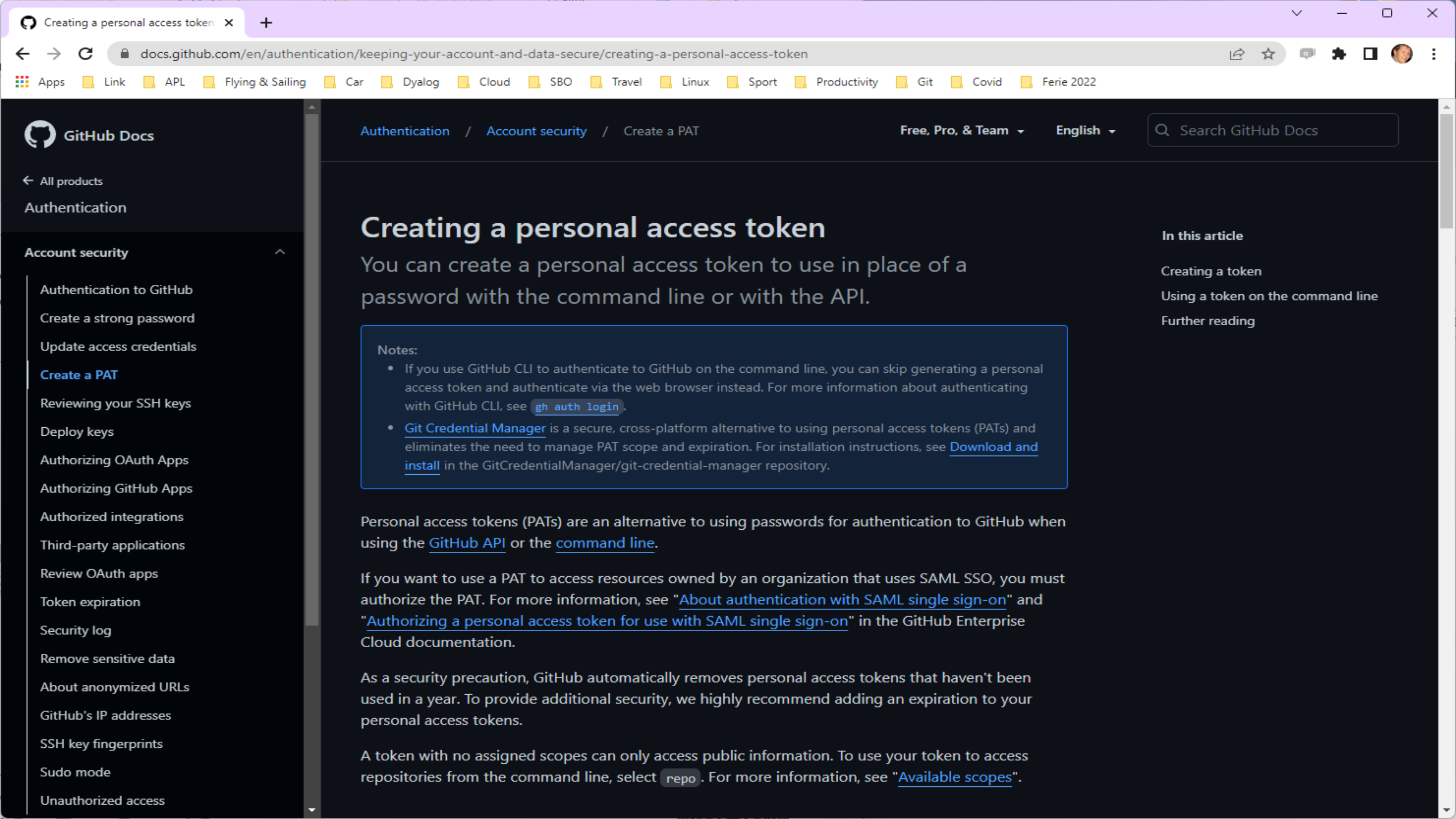
The GNU GPLv3 also lets people do almost anything they want with your project, *except* distributing closed source versions.

Ansible, Bash, and GIMP use the GNU GPLv3.

Cloning your repo

- You can set your Git client up to communicate via HTTPS or SSH
- HTTPS is easier to setup, and with Personal Access Tokens (PAT) and 2FA (Two-factor Authentication) it probably satisfies your security needs
- Once set up, try cloning your new "repo", or:

```
git clone https://github.com/dialog-training/2022-SA3 /some/folder
```



Creating a personal access token ×

docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

Apps | Link | APL | Flying & Sailing | Car | Dyalog | Cloud | SBO | Travel | Linux | Sport | Productivity | Git | Covid | Ferie 2022

GitHub Docs

← All products

Authentication

Account security

- Authentication to GitHub
- Create a strong password
- Update access credentials
- Create a PAT
- Reviewing your SSH keys
- Deploy keys
- Authorizing OAuth Apps
- Authorizing GitHub Apps
- Authorized integrations
- Third-party applications
- Review OAuth apps
- Token expiration
- Security log
- Remove sensitive data
- About anonymized URLs
- GitHub's IP addresses
- SSH key fingerprints
- Sudo mode
- Unauthorized access

Authentication / Account security / Create a PAT

Free, Pro, & Team ▾ English ▾ Search GitHub Docs

Creating a personal access token

You can create a personal access token to use in place of a password with the command line or with the API.

Notes:

- If you use GitHub CLI to authenticate to GitHub on the command line, you can skip generating a personal access token and authenticate via the web browser instead. For more information about authenticating with GitHub CLI, see `gh auth login`.
- Git Credential Manager is a secure, cross-platform alternative to using personal access tokens (PATs) and eliminates the need to manage PAT scope and expiration. For installation instructions, see Download and install in the GitCredentialManager/git-credential-manager repository.

Personal access tokens (PATs) are an alternative to using passwords for authentication to GitHub when using the GitHub API or the command line.

If you want to use a PAT to access resources owned by an organization that uses SAML SSO, you must authorize the PAT. For more information, see "About authentication with SAML single sign-on" and "Authorizing a personal access token for use with SAML single sign-on" in the GitHub Enterprise Cloud documentation.

As a security precaution, GitHub automatically removes personal access tokens that haven't been used in a year. To provide additional security, we highly recommend adding an expiration to your personal access tokens.

A token with no assigned scopes can only access public information. To use your token to access repositories from the command line, select `repo`. For more information, see "Available scopes".

In this article

- Creating a token
- Using a token on the command line
- Further reading

How Morten Works – Live Demo

- VS Code & Git Lens

How Josh Works – Live Demo

- Dado