Git

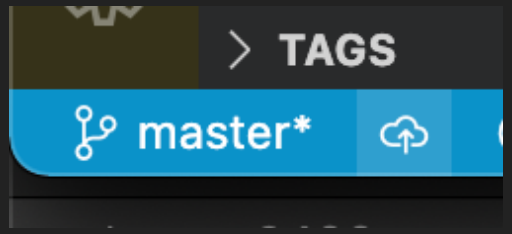
BRANCHES

JOFH@KEA.DK

branches

`git` is organized into "branches"

The main one is called "main" or "master", and it's the only one we've been using



You can see your current branch in the bottom left corner

THE IDEA

The idea is: whenever you wish to experiment, start on a new feature, section etc

you "snap a branch", which means, create a new branch

A branch is a copy of the repository at a certain point in time

Each branch has it's own commit history

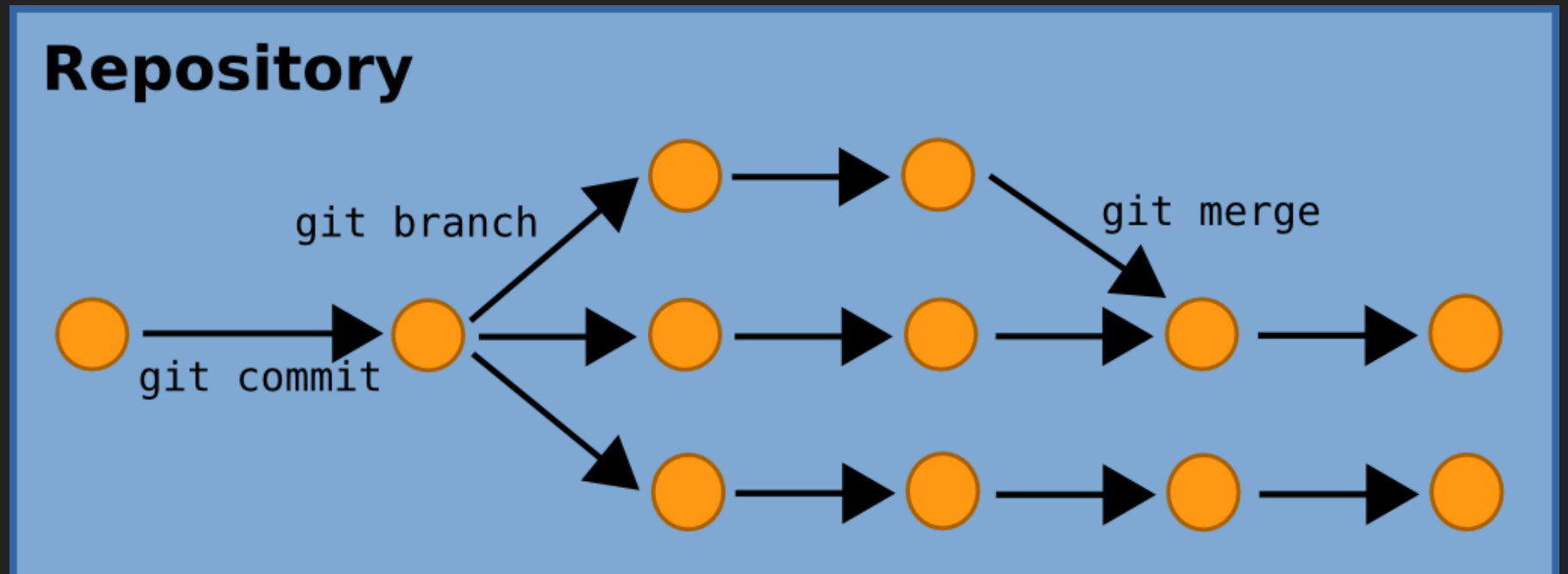
You can always switch back and forth between branches

But your files will change

Nothing will be lost, it will just be on a different branch

Working on branches also prevents unnecessary "conflicts"

Once you're done, you can "merge" the branch back in to "main"


Repository
git branch
git merge
git commit

When you create a branch you tell git what branch to base it off

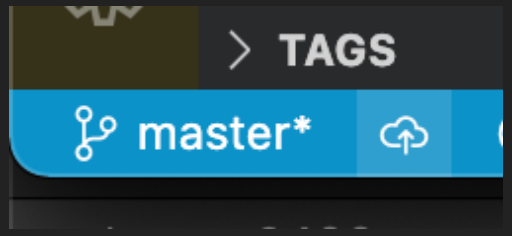
We're gonna go with `main` always, but starting from other branches is totally ok

When we switch branches, we also call it "checkout"

LET'S SEE BASIC

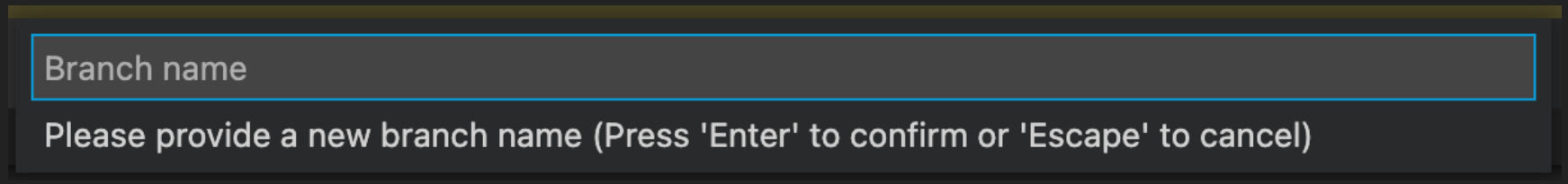
BRANCHING IN ACTION

1. This is where you manipulate branches

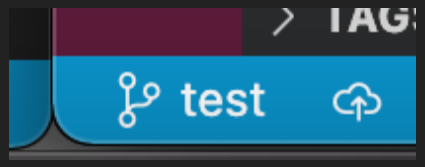


(or Shift + Cmd + P and then "create new branch")

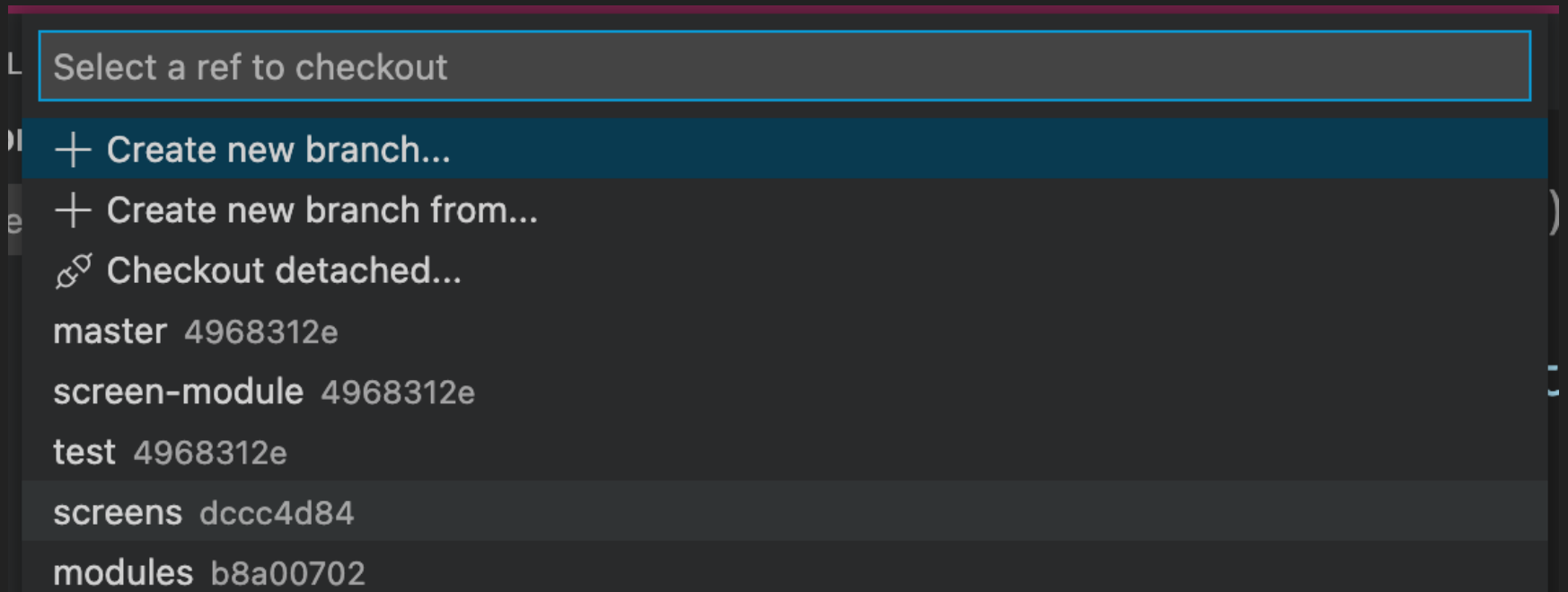
3. Give it a unique name, without spaces



4. It has now switched to that branch



5. And you can switch back and forth between branches



GENERAL THOUGHTS

PART 2

DAILY FLOW

1. Setup the project (as shown earlier)
2. Find a "feature" you wanna work on
3. Create a new `branch` for that feature
4. `checkout` the feature `branch`
5. 1. code
   2. add
   3. commit
   4. push
   5. go to 5.1
6. `merge` the feature `branch` into `main`

Whiteboard

MERGING

SO MANY WAYS

But two main places we can do it

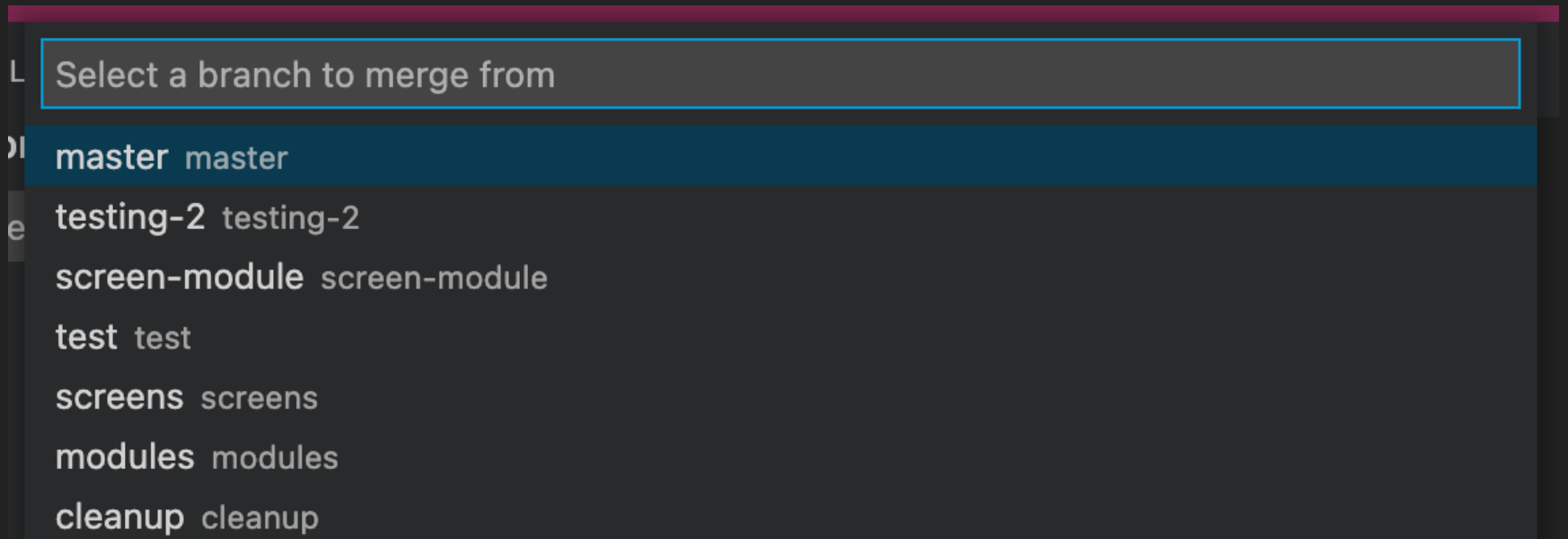
1. Merging our own local branches
2. ~~Merging two branches through github~~

If we've done as we're supposed to do, there should be no problems

Let's try it

1. Checkout the branch you want to merge in to (usually `main`)
2. press `Cmd + Shift + P` to open the Command Palette and find "Git: Merge Branch..."

   Select it
3. The select the branch that should go in to main



3. We might get asked a question here, just pick "merge"

4. Unless we messed up, the `main` branch is now identical to the branch we merged in

EXERCISE

Fronter: 1. Branches

"RULES"

1. Implement one feature per branch
2. Do "one thing" per commit
3. Never work directly on main
4. Make you commit messages meaningfull
   ("changed stuff" is bad, "add the modal" is good)
5. Have a standard for commit messages
   (usually, imagine it says "this commit will" in front of the message)

DICTIONARY

Add

Mark the file to be part of a commit (in brackets, that's checking off the file)

Branch

Create a new timeline for your project. Can be discarded / ignored if not needed

Checkout (of a branch)

Feature

Not a git word, more of a concept. An isolated part of the app that deserves it's own branch(es)

git

The version control system

github

A website, where we can store/backup/share our

main

The main branch in our repository

Merge

Merge two branches together into one

Push

Put your commits (and code) to the remote

Remote

LET'S TRY

Depending on what I just did

I just want to make sure you've seen it :-)