

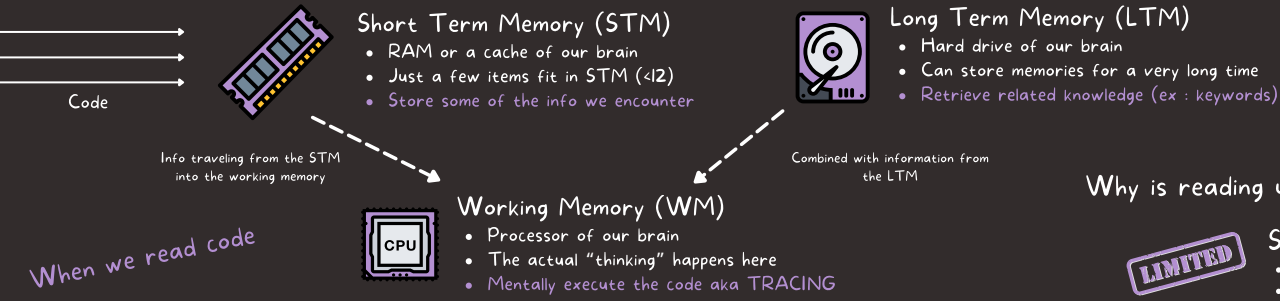
THE PROGRAMMER'S BRAIN

by Felienne Hermans

60%



of our time



When we read code

Why is reading unfamiliar code hard?

LIMITED

Short term memory

- Time : 30 seconds
- Size : 7 +/-2 things

How to read code better ?

"More concepts, data structures and syntax you know the more code you can easily chunk, and thus remember and process"

Learn programming syntax



Use Flashcards

- Front : prompt
- Back : corresponding knowledge



Remember syntax longer

- Retrieval : trying to remember something
- Elaboration : connecting new knowledge to existing memories

Read / Hide / Write code exercises

Write CHUNKABLE Code

"Experts group info in logical ways aka chunks"



Write COMMENTS
High-level comments help to chunk larger pieces of code



Use Design PATTERNS
Help to process code faster



Leave BEACONS
var names, operators (<, >), if, else, comments,...

How to not forget things ?

"We cannot remember things for a long time without extra practice"

DON'T FORGET

After 2 days, just 25% of the knowledge remains in our LTM

Spaced repetition

- Practice regularly
- Best way to prevent forgetting



Revisit your Flashcards

- Once a month
- Each repetition strengthens your memory

Read complex code easier

Reduce cognitive load

Refactoring code

Ex : replace unfamiliar language constructs



"Our ability to learn a natural language can be a predictor of your ability to learn to program."



Dependency graph

- Circle variables
- Draw lines between occurrences



State table

- Focuses on the values of variables
- 1 column / variable
- 1 line / step in the code



Roles of variables (Sajaniemi's framework)



- Fixed value : value does not change after initialization
- Stepper : variable stepping through a list of values
- Flag : has happened or is the case
- Walker : traverses a data structure (search index)
- Most-recent holder : holds the latest value encountered
- Most-wanted holder : holds the best value found so far

- Gatherer : collects data and aggregates it
- Container : holds multiple elements
- Follower : keep track of a previous value
- Organizer : transformed variable
- Temporary : used only briefly

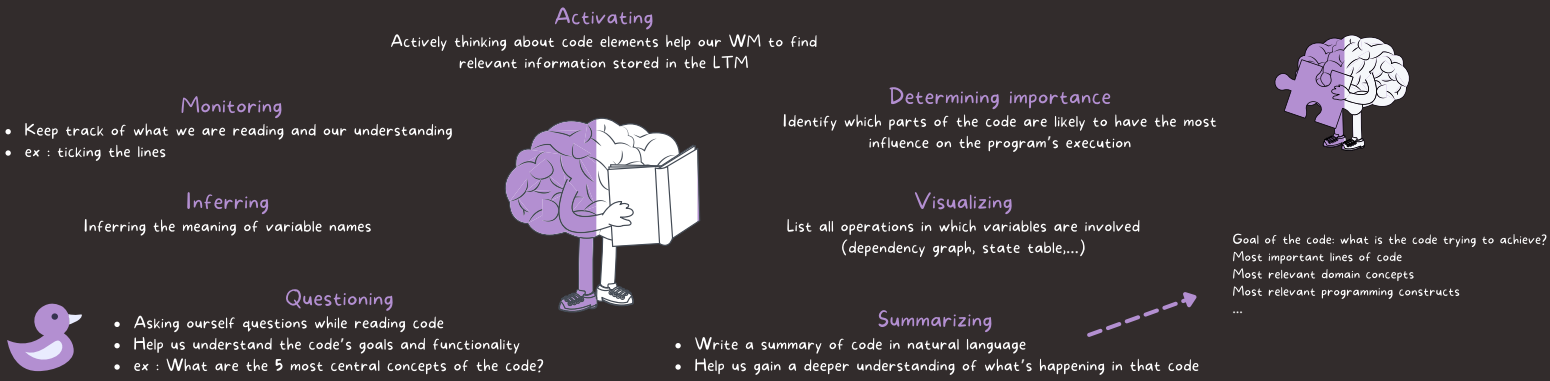
Cognitive load

- Capacity of our Working Memory
- Capacity : 2 to 6 "things"

"Understanding what types of information variables hold is key to being able to reason about and make changes to code."

"Many similarities between reading code and reading natural language"

Text comprehension strategies applied to code



Write better code

Search....

Avoid



Abbreviation

Check Hofmeister research



Snake Case -> use camel Case
camelCase leads to higher accuracy

Clear names help our LTM

LTM searches for related informations

Avoid Arnaoudova's linguistic anti-patterns

Methods that do more than they say

Methods that do the opposite than they say

Identifiers whose name says that they contain more than what the entity contains

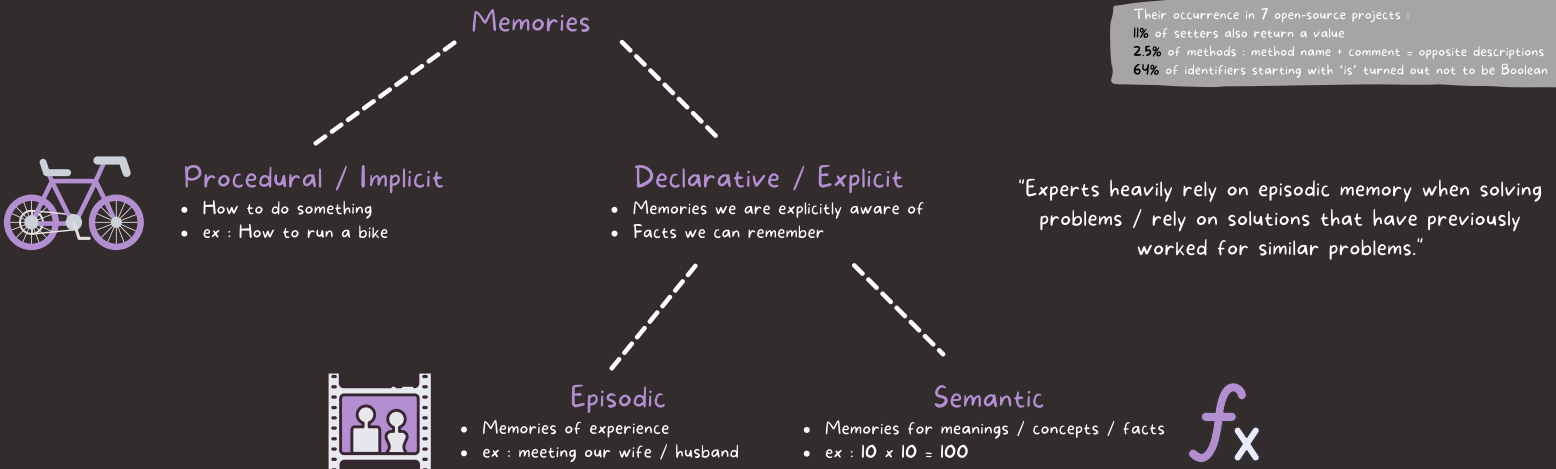


Methods that say more than they do

Identifiers whose name says that they contain less than what the entity contains

Identifiers whose name says the opposite than the entity contains

LTM can store different types of memory



Getting better at solving complex problems



Deliberate practice : requires focused attention and is conducted with the specific goal of improving performance.

Worked examples : something like a recipe which describes in detail the steps that are needed to solve the problem.

20% of developers time on interrupts

Better handle interruptions

Prepare for it



Store mental model

- Apart from the code
- Comments : excellent location to leave it
- Warm-up period in comprehension activities



Help your "Prospective memory"

- Put TODO comments in the part of the code
- Remind you to complete / improve part of the code



15' to start editing code after an interruption



Label subgoals

- Write down small steps of a problem
- Use mind maps for example

Prospective memory : memory of remembering to do something in the future. (related to planning / problem solving)

On-boarding process

Typical

dev throws information



Senior dev



Newcomer

High cognitive load

Explain only relevant informations

Separate

Domain learning

Exploring code

Support the LTM of the newcomer



Exploration

- Browse the codebase
- Get a general sense of the codebase



Transcription

- Give the newcomer a clear plan
- Implement it



Searching

ex : find a class that implements a certain interface

Limit tasks to ONE programming activity



Comprehension

Understand aspects of the code
ex : summarize a specific method in natural language



Incrementation

- Add a feature to an existing class
- Creation of the plan for the feature.

Start with it : read code together

