Ownership, References & Borrowing, Slices Handout

The borrowing and ownership mechanism can be simplified down to:

- Passing a variable by value will move ownership, dropping the original variable from memory. Note: applies to not copy types.
- Passing a variable by reference will keep the original variable.
- Passing a variable by mutable reference will keep the original variable, but allow you to modify the variable.
- You may only borrow a variable mutably once at a time, and you may not immutably borrow while mutably borrowing.
- You may have as many immutable borrows as you want, so long as you aren't modifying that value.
- You may mutably borrow a field in a struct, and then mutably borrow a different field in the same struct simultaneously, so long as you aren't also mutably borrowing the overall struct.
- You may mutably borrow multiple slices from the same array simultaneously so long as there is no overlap. Note: safe code cannot convince the compiler that two slices don't overlap. You‘ll have to use unsafe code.
- Safe memory practices means that instead of mutably borrowing the same variable in multiple places, you queue the changes to make in a separate location and apply them serially one after another.

Quellen:

https://www.reddit.com/r/rust/comments/5kq2s9/borrowing_and_ownership_cheat_sheet/ Zuletzt Abgerufen am 15. April 2019