In STM you can also establish invariants, which check properties of your transactions.

E.g. Check, if two variables always contain the same value or if the amount of an account is positiv.

always :: STM Bool -> STM ()

Argument is a transaction (usually only reading TVars) and checking a property of the system state. Return True if the state is okay, False otherwise.

Check if the amount of an account is positiv:

```
main = do
  acc <- new_account 0
  atomically $ always $ do
     bal <- readTVar acc
     return bal >= 0
```

Assume: Invariantes are always intrduced with the direct combination of atomically and always
always :: STM Bool -> IO ()

When should we check an invariant?
- when establishing a new invariant, if invariant does not hold, throw an exception
- before committing another transaction, in the validation, if invalid, then rollback

Store every invariant within a global state, such that they can be checked before every(?) commit.

An invariant must only be checked, if the WS of a transaction contains TVars read in the last execution of that invariant. So more than one invariante maybe checked.

The check has to consider the WS of the transaction, but without realizing them in the TVars. write_tvar actions of the invariant should not be commited(?).

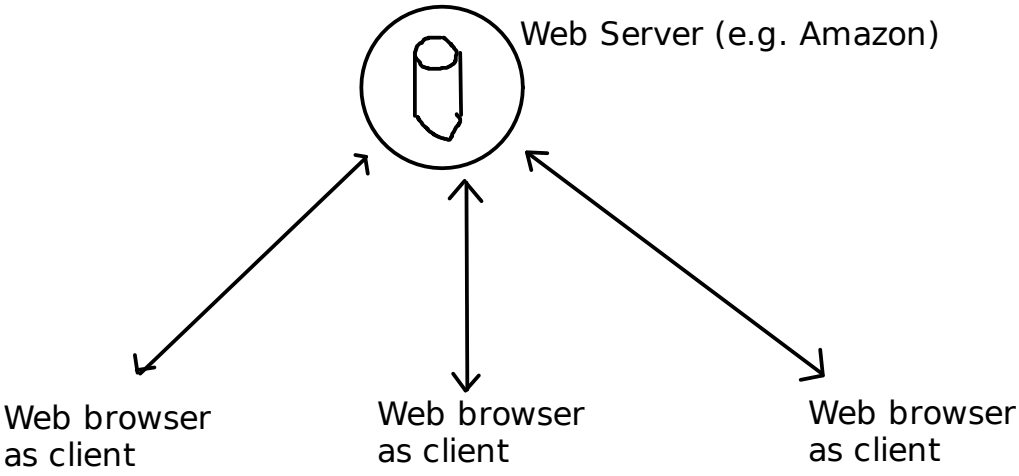
if one invariant yields true -> commit the transaction and update the rs of the invariant for the next invariant initiation
otherwise -> rollback transaction (or break it by an exception?)

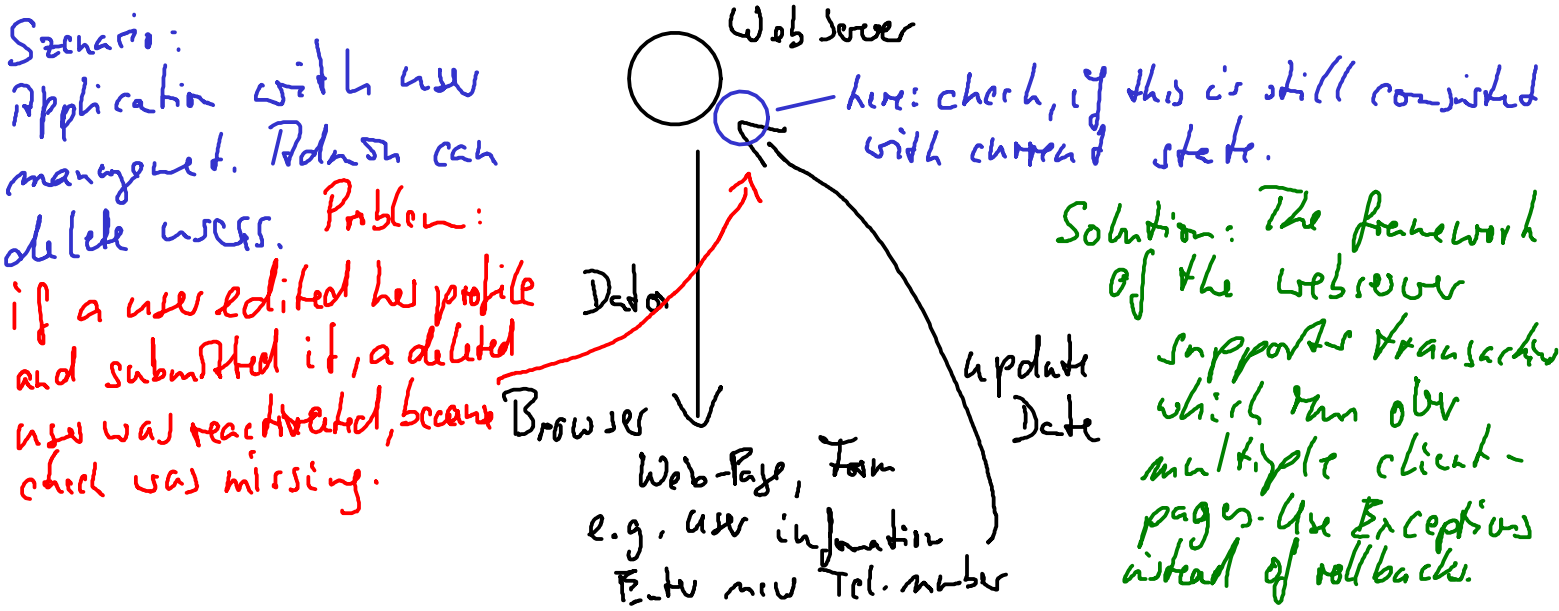
Open problems in STM:

- garbage collection, in our implementation no TVar becomes garbage
- often lists should be replaced by balanced search trees
- we lock all TVars with respect to a global order. Some experients show that the strategie of putting the locks back, if you don't get the next lock can be much more efficient in independent concurrency. Or a global lock might also be more efficient.
- Haskels type system avoids side effect within transaction, this is not guaranteed in our Erlang implementation.

Transactions also occure in distributed settings:

- distributed data base (we will discuss distributed commit protocolls next lecture)
- Web applications assume transactions as well.





Shopping basket is usually a state of a transaction. But, thinks inside the basket are not yet yours! There read/writes on resources) like TVars within the transaction, which however are not critical. E.g. decrementing the number of items, when the number is very large. However, out approach would show that one transaction has to rollback instead of simply sequentiallizing the two decrement transactions. This idea of improvement can also be used to improve STM.