Uncontained Side Effects

John Clow

What are Side Effects

A side effect if when code modifies some state outside its scope or has an observable interaction with its calling functions or the outside world besides returning a value

What is NOT a side effect?

- Creating a new value
- Assigning values to a variable in the local scope
- Modifying a value that is only in the local scope
- Returning a value

Danger of Side Effects example:

<Code in Terminal>

Purity

- The function always evaluates the same result value given the same argument value(s)
- Evaluation of the result does not cause any semantically observable side effect
- This includes Statelessness
    - The function is not dependent on any state in the program

Benefits of Purity

- Easy to test
  - No need to set up external variables to test function
  - No worries about hidden variables causing untested edge cases
- Easy to reason about
  - Don’t have to understand how entire program state will affect the function
  - Easy to write proofs about
- Repeatability
  - You can cache the output of a function
  - You can evaluate the function at another time - Lazy Evaluation
    - Varun will talk about this

Ocaml and Purity

- In OCaml and most functional languages, the goal is to have mostly pure code
- Purity makes it easy for Programming language researchers to reason about code
- This extends to their data structures. You are always encouraged to use immutable data structures, for that allows for aliasing without harming purity.
    - If you want to return different data structures, you have to build new ones.

What are side effects again?

Side Effects include:

- Printing to the terminal
- Reading from the terminal
- Writing a file
- Displaying anything on a screen
- Communicating on a network

Without side effects, your code can do nothing of value

Some side effects can be avoided

- Global State
- Singleton object patterns
- State in shared objects

Rust

- Rust inherits the desire for purity from functional languages
  - But it wants to be a systems language
- Copy on modification is extremely expensive
  - Therefore they need a new solution to prevent side effects
  - Borrow Checker

Managing Side Effects in web environments

- React
  - One of the hottest interactive web UI libraries
  - Designed based on a (mostly) stateless hierarchy of containers to simplify webpages
- Microservices
  - Small serverside functions with limited global state
  - Lots are written in Scala, a functional-lite language

Questions?

Purity Example

Pure:

```
function square(x):
    return x * x
end
```

Impure:

```
a = 3
function mult_by_a(x):
    return a * x
end
```

Escape Hatches for Side Effects