Side effect Elm:

1. …always returns the same result, given the same input.
2. …does not have any side effects.

When to embrace side effects: State change, e.g.IO

When to avoid: when/wherever possible, use currying/lambda instead

Pure function i java ex:

```
public static int sum(int a, int b) {
   return a + b;
}
```

Impure i java da returværdi kan være forskellig med samme input:

```
public static int sum(int a, int b) {
   return new Random().nextInt() + a + b;
}
```

Impure i java pga side effect:

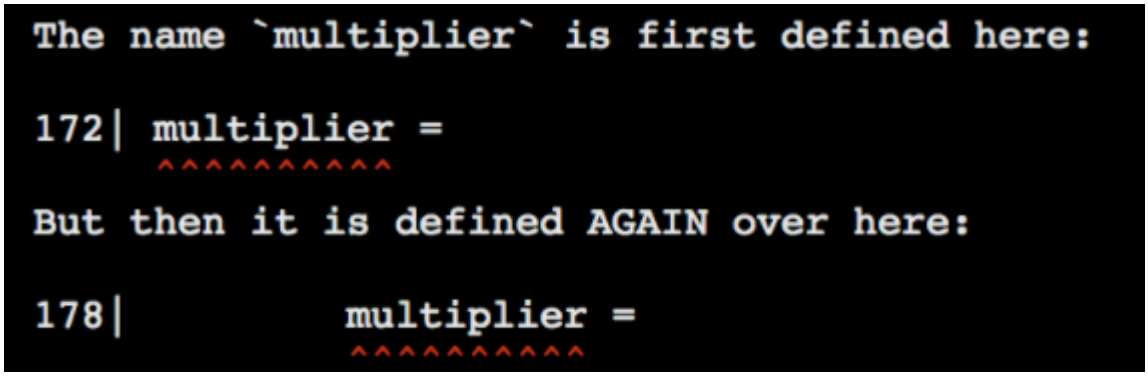
```
public static int sum(int a, int b) {
   writeSomethingToFile();
   return a + b;
}
```

Eksempel Elm med fejl besked, var multiplier forsøges ændret i funktion:

(Elm repl er en undtagelse pga "sandbox")

```
multiplier =
    6
multiplyByFive number =
    let
        multiplier =
            5
    in
    number * multiplier
```

Fejl besked:



Pure function i Java:

```
public static int sum(int a, int b) {
    return a + b;
}
```

Kode eksempel impure function JS:

Mult er en global variabel, værdien kan ændres og funktionen vil derefter returnere et andet (muteret) resultat med samme parameter (num)

```
function multiply(num, mult){
  return num * mult;
}

// Function 2

var mult = 2;

function multiply(num){
  return num * mult;
}
```

var scoreMultiplier ændres (muterer) i funktionen doubleScores, fra 2 til 3

```
var scoreMultiplier = 2;
var highestScores = [316, 320, 312, 370, 337, 318, 314];

function doubleScores(scores) {
    var newScores = [];

    for (var i = 0; i < scores.length; i++) {
        newScores[i] = scores[i] * scoreMultiplier;
    }

    scoreMultiplier = 3;

    return newScores;
}
```