REACTIVE AND FUNCTIONAL PROGRAMMING

Produced by Chase Packer, Anna McKinney, Katelyn Beasley, and Ryan Russell

Functional_and_Reactive_Programming_Presentation.pptx

Repository

REACTIVE PROGRAMMING

What is Reactive Programming?

Programming to handle asynchronous event streams

- Consists of functions that monitor for certain conditions or changes and functions that respond to those changes

Applications

- GUI programming
- Google maps location tracking

PRINCIPALS OF REACTIVE PROGRAMMING

1) Responsive
2) Resilient
3) Elastic
4) Message Driven

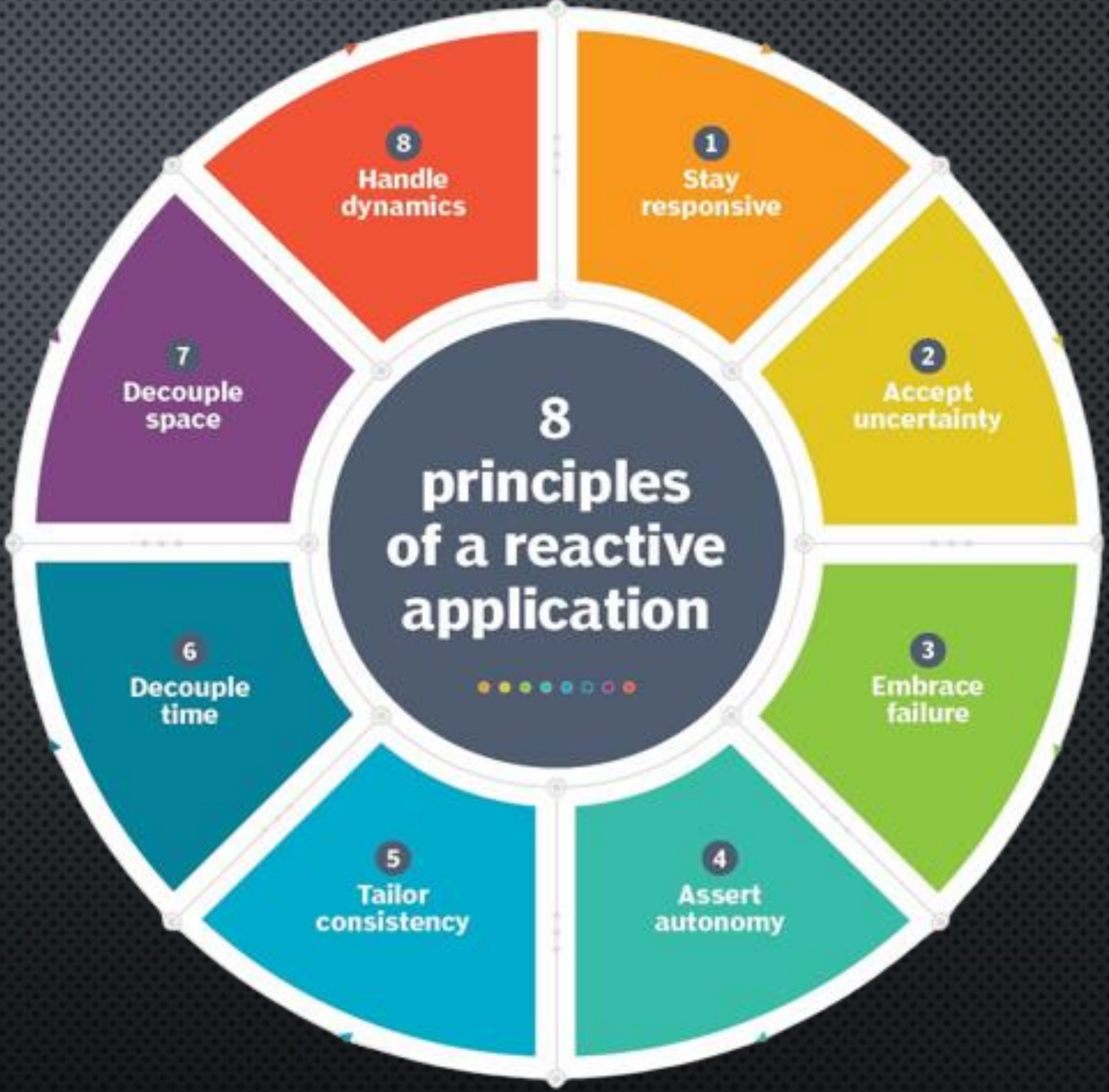


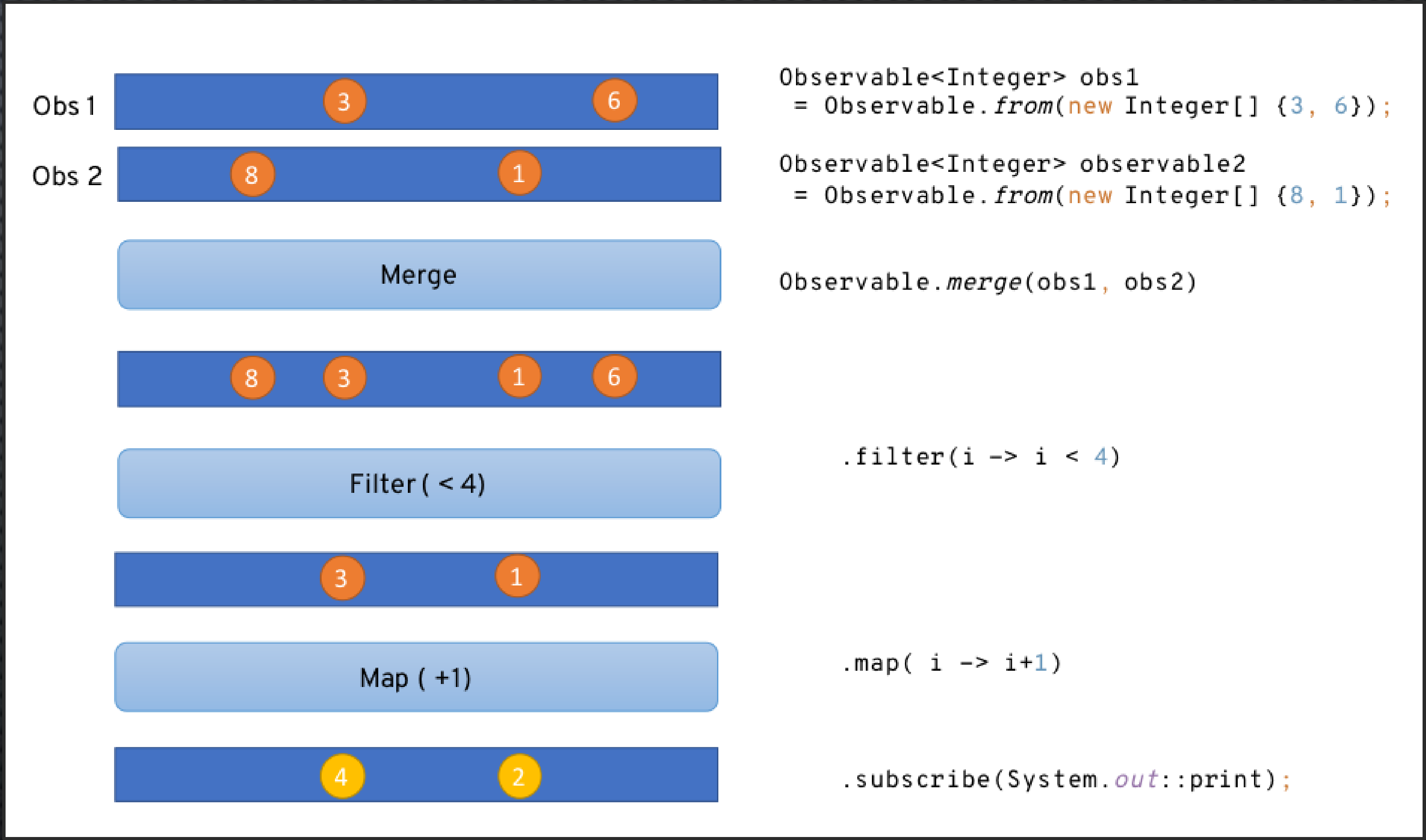
Figure 1: Nolle, T. (2021, March 24). *What is reactive programming? what you need to know*. App Architecture. https://www.techtarget.com/searchapparchitecture/definition/reactive-programming

EXAMPLES

Using ReactiveX / RxJava

Core Concepts:

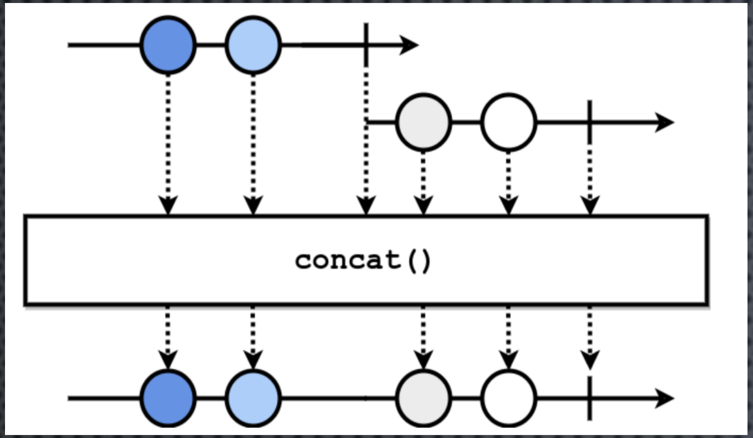
- Observable
- Observer
- Subscribe

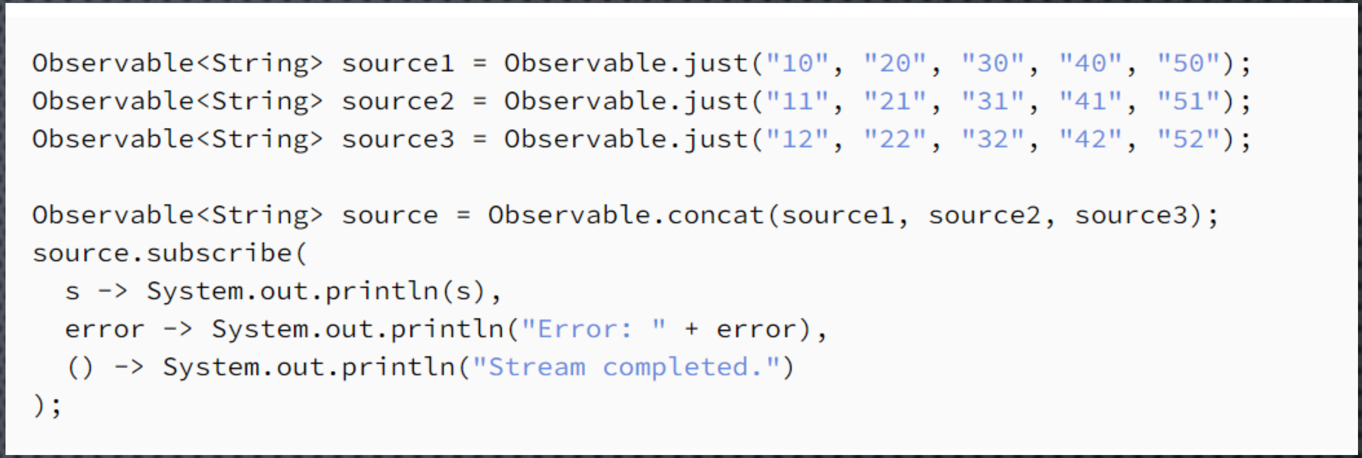


```
Observable<Integer> obs1
 = Observable.from(new Integer[] {3, 6});

Observable<Integer> observable2
 = Observable.from(new Integer[] {8, 1});

Observable.merge(obs1, obs2)

    .filter(i -> i < 4)

    .map( i -> i+1)

    .subscribe(System.out::print);
```

Figure 2: Escoffier, C. (2023, July 31). *5 things to know about reactive programming*. Red Hat Developer. https://developers.redhat.com/blog/2017/06/30/5-things-to-know-about-reactive-programming

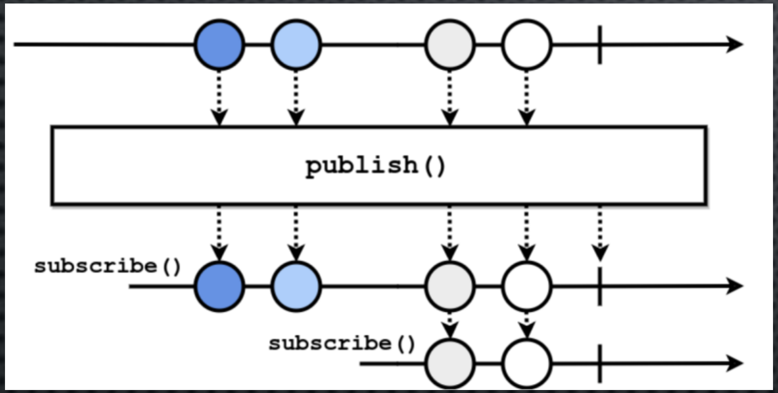
Transformation

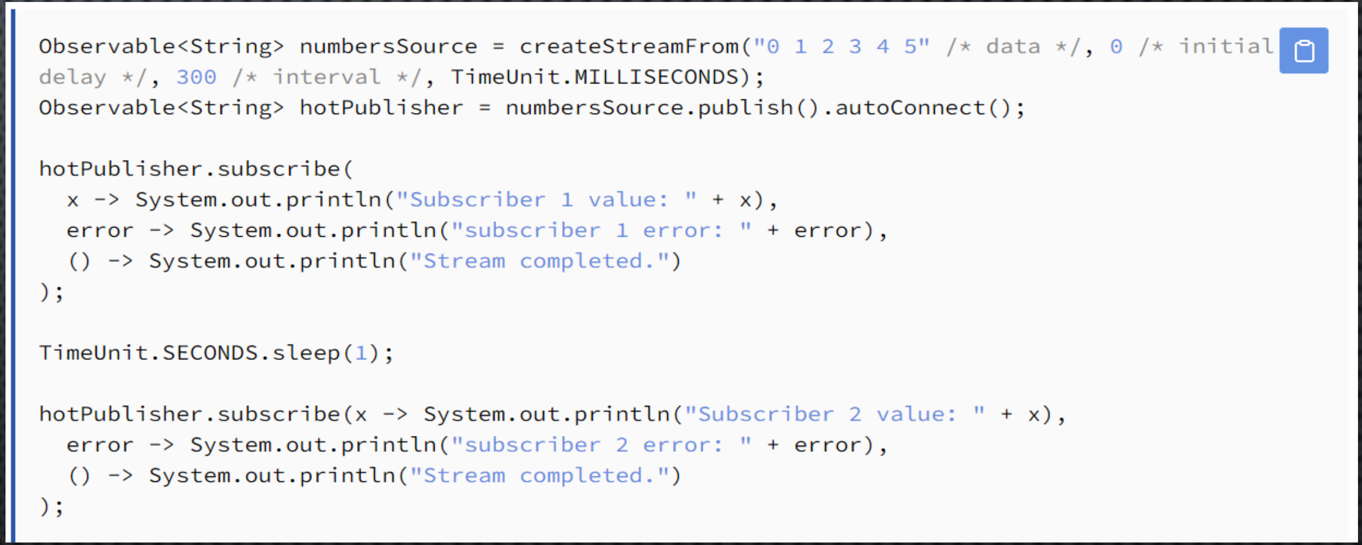




```
Observable<String> source1 = Observable.just("10", "20", "30", "40", "50");
Observable<String> source2 = Observable.just("11", "21", "31", "41", "51");
Observable<String> source3 = Observable.just("12", "22", "32", "42", "52");

Observable<String> source = Observable.concat(source1, source2, source3);
source.subscribe(
  s -> System.out.println(s),
  error -> System.out.println("Error: " + error),
  () -> System.out.println("Stream completed.")
);
```

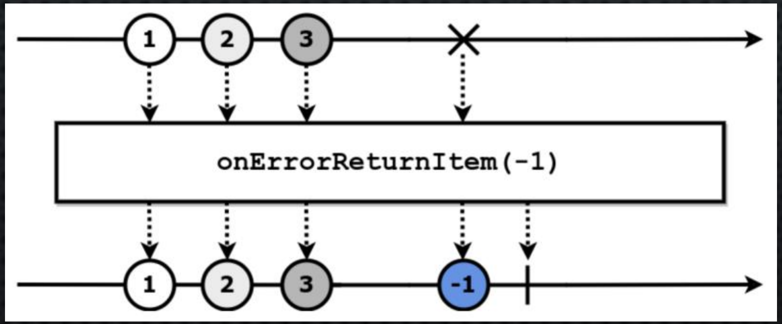
Multicasting

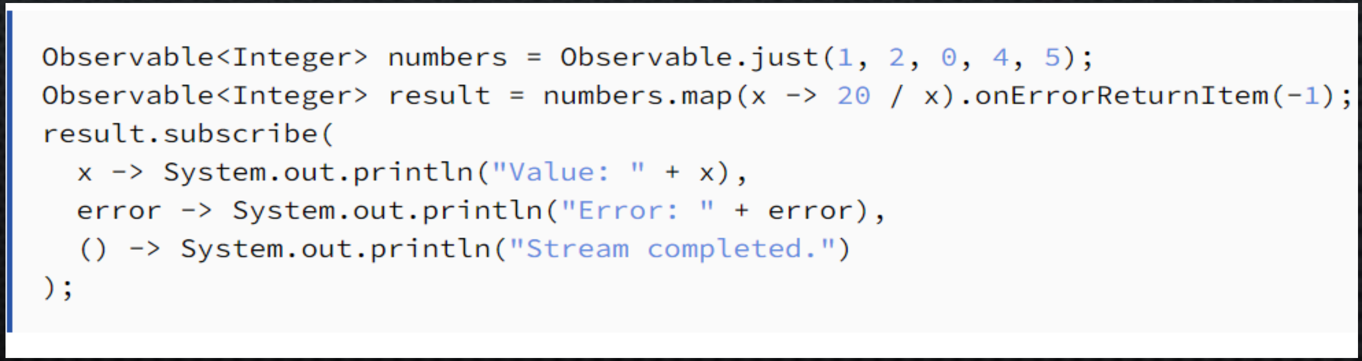




```
Observable<String> numbersSource = createStreamFrom("0 1 2 3 4 5" /* data */, 0 /* initial
delay */, 300 /* interval */, TimeUnit.MILLISECONDS);
Observable<String> hotPublisher = numbersSource.publish().autoConnect();

hotPublisher.subscribe(
  x -> System.out.println("Subscriber 1 value: " + x),
  error -> System.out.println("subscriber 1 error: " + error),
  () -> System.out.println("Stream completed.")
);

TimeUnit.SECONDS.sleep(1);

hotPublisher.subscribe(x -> System.out.println("Subscriber 2 value: " + x),
  error -> System.out.println("subscriber 2 error: " + error),
  () -> System.out.println("Stream completed.")
);
```

Error Handling





```
Observable<Integer> numbers = Observable.just(1, 2, 0, 4, 5);
Observable<Integer> result = numbers.map(x -> 20 / x).onErrorReturnItem(-1);
result.subscribe(
  x -> System.out.println("Value: " + x),
  error -> System.out.println("Error: " + error),
  () -> System.out.println("Stream completed.")
);
```

REACTIVE PROGRAMMING TOOLS

ReactiveX

Reactive libraries

Different versions for a wide variety of languages

Project Reactor

Spring Webflux

Reactive-stack web framework

Vert.X

Reactive toolkit

JVM

Akka

Reactive toolkit

JVM

DEMO TIME

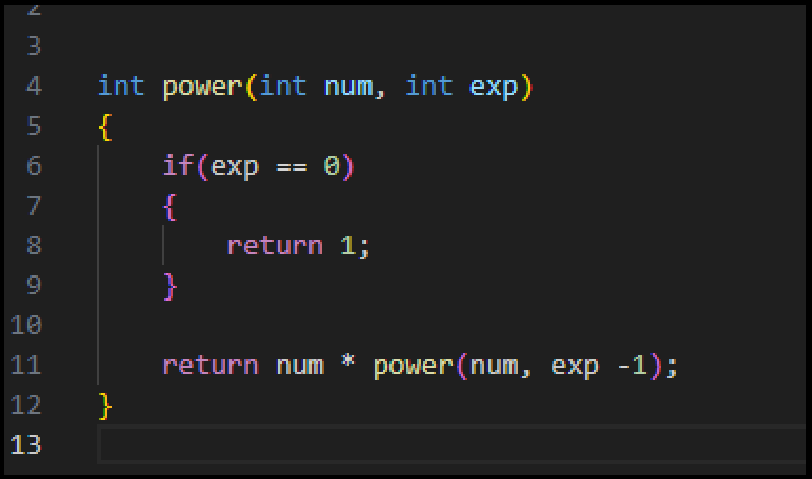
FUNCTIONAL PROGRAMMING

What is Functional Programming

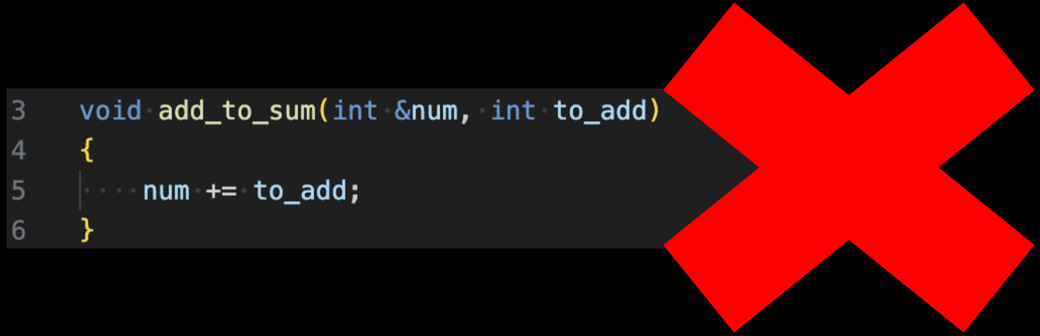
- Programming Language Paradigm based on functions and lambda calculus
  - Code works by evaluating series of "mathematical" Functions
- Has 5 Main Principles:
  - Pure Functions
  - Immutable Variables (No side-effects)
  - Referential Transparency
  - Recursion (No Loops)
  - First-Class and higher-order functions
- Useful for Mathematics, Concurrency, and Parallelism

PURE FUNCTIONS

- Given the same input, function will produce the same output
- No Side-Effects



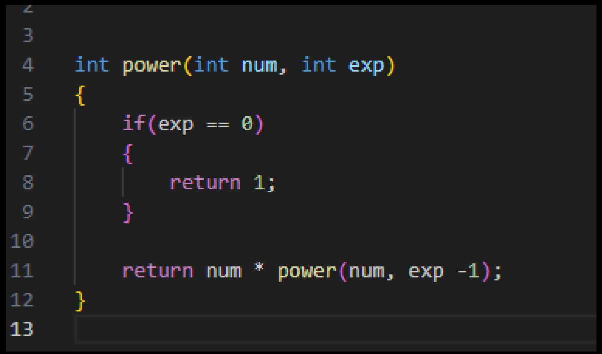
```
int power(int num, int exp)
{
    if(exp == 0)
    {
        return 1;
    }

    return num * power(num, exp -1);
}
```



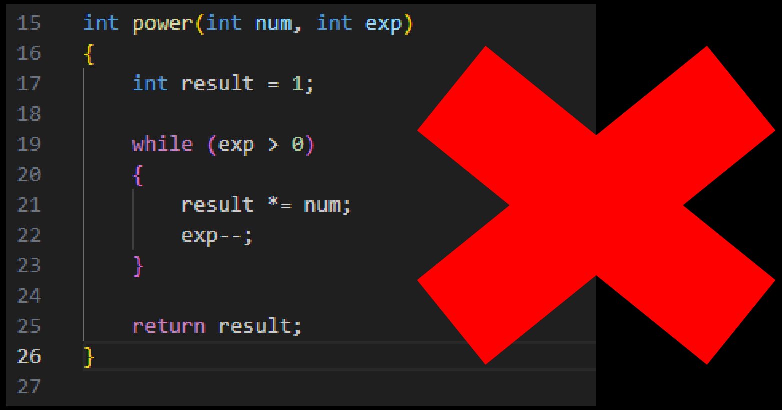
```
void add_to_sum(int &num, int to_add)
{
    num += to_add;
}
```

RECURSION

- Functional Programming does not use loops
- Instead, you use recursion



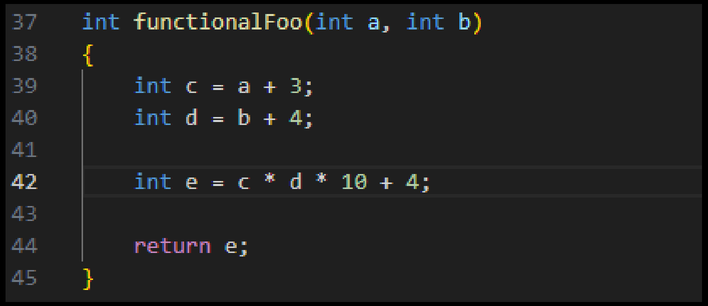
```
int power(int num, int exp)
{
    if(exp == 0)
    {
        return 1;
    }

    return num * power(num, exp -1);
}
```



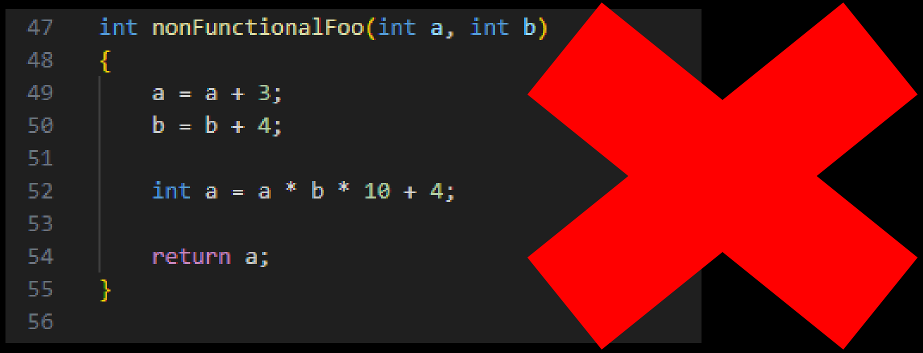
```
int power(int num, int exp)
{
    int result = 1;

    while (exp > 0)
    {
        result *= num;
        exp--;
    }

    return result;
}
```

Referential Transparency (Immutable Variables)

- Variables, once they are defined, cannot be changed
- If you want to store a new value, a new variable must be created
- Maintains state of program and does not allow for side-effects
- Can lead to inefficiency when dealing with large data structures



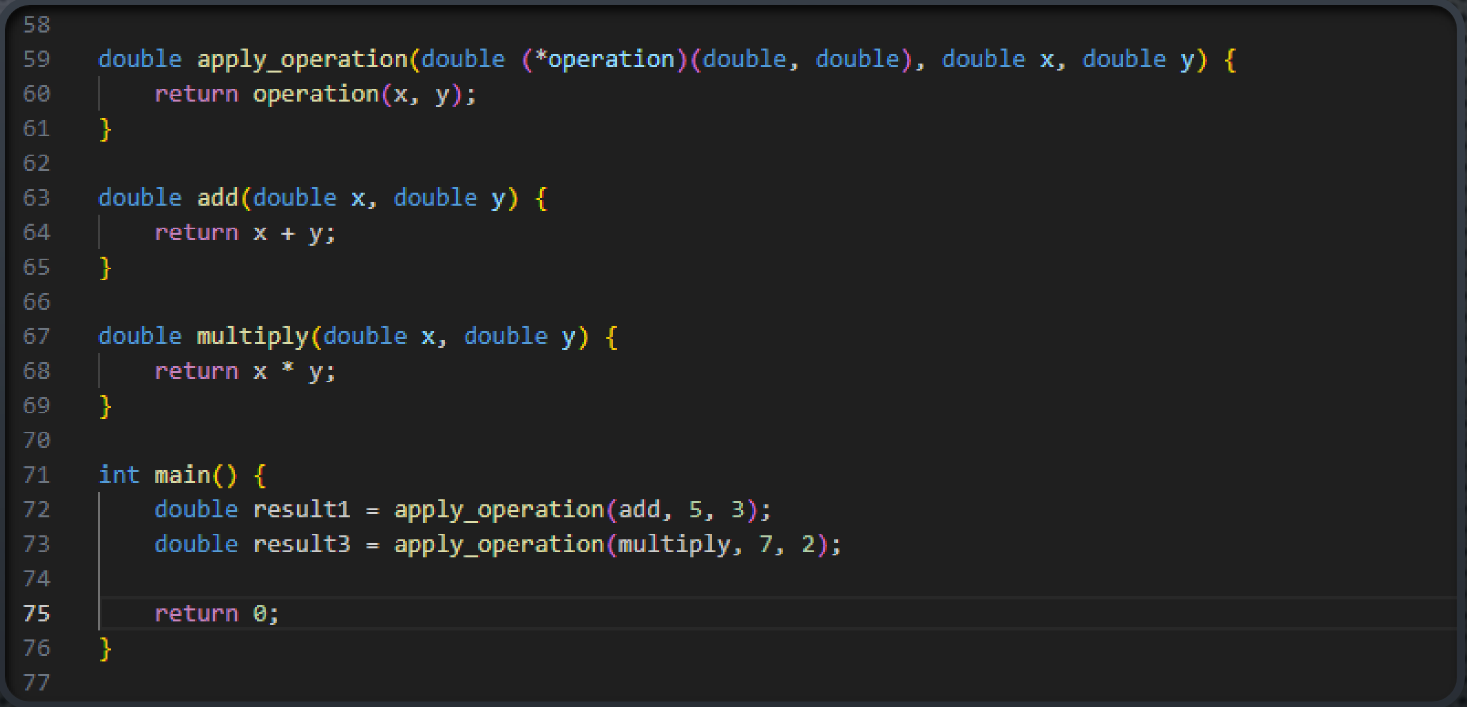
```
int functionalFoo(int a, int b)
{
    int c = a + 3;
    int d = b + 4;

    int e = c * d * 10 + 4;

    return e;
}
```



```
int nonFunctionalFoo(int a, int b)
{
    a = a + 3;
    b = b + 4;

    int a = a * b * 10 + 4;

    return a;
}

```

First-Class and Higher-Order Functions

- Functions can act as parameters to other functions
- Functions can also be the return value from another function
- **Higher-Order Functions** are functions that accept a function as an argument or return a new function



```
double apply_operation(double (*operation)(double, double), double x, double y) {
    return operation(x, y);
}

double add(double x, double y) {
    return x + y;
}

double multiply(double x, double y) {
    return x * y;
}

int main() {
    double result1 = apply_operation(add, 5, 3);
    double result3 = apply_operation(multiply, 7, 2);

    return 0;
}
```

FUNCTIONAL PROGRAMMING LANGUAGES

- Haskell
- F#
- Scala
- Lisp
- Ocaml
- Technically, you can write in a "Functional" style in most programming languages

DEMO TIME

FUNCTIONAL REACTIVE PROGRAMMING

FUNCTIONAL REACTIVE PROGRAMING (FRP)

- Programming designed to handle events using code based on mathematical functions
- Useful in situations where there is a lot of data coming in in real time
- Applications:
  - Algorithmic trading systems in finance
  - Video game Development

HOW DOES FUNCTIONAL REACTIVE PROGRAMMING COMPARE TO OBJECT ORIENTED PROGRAMMING?

- Object Oriented Programming
  - State-Driven – code changes state of program
  - Based on imperative programming commands
  - Smaller-scale applications
- Functional Reactive Programming
  - Reacts to events
  - Based on functional programming (mathematical) concepts
  - Handles large amounts of data in real time

SOURCES

Boner, J., Farley, D., Kuhn, R., & Thompson, M. (n.d.). The Reactive Manifesto. https://www.reactivemanifesto.org/

Escoffier, C. (2023, July 31). *5 things to know about reactive programming*. Red Hat Developer. https://developers.redhat.com/blog/2017/06/30/5-things-to-know-about-reactive-programming

Giraldo, J. E., & Giraldo, J. P. (2023, March 9). *Tools for reactive programming in Java and .NET*. Globant Blog. https://stayrelevant.globant.com/en/technology/quality-engineering/reactive-programming-tools-backend-languages/

Kulkarni, B. (2021, June 15). Reactive Programming Vs Functional Programming. Medium. https://bhagyashree9214.medium.com/reactive-programming-vs-functional-programming-4d4d17786abc

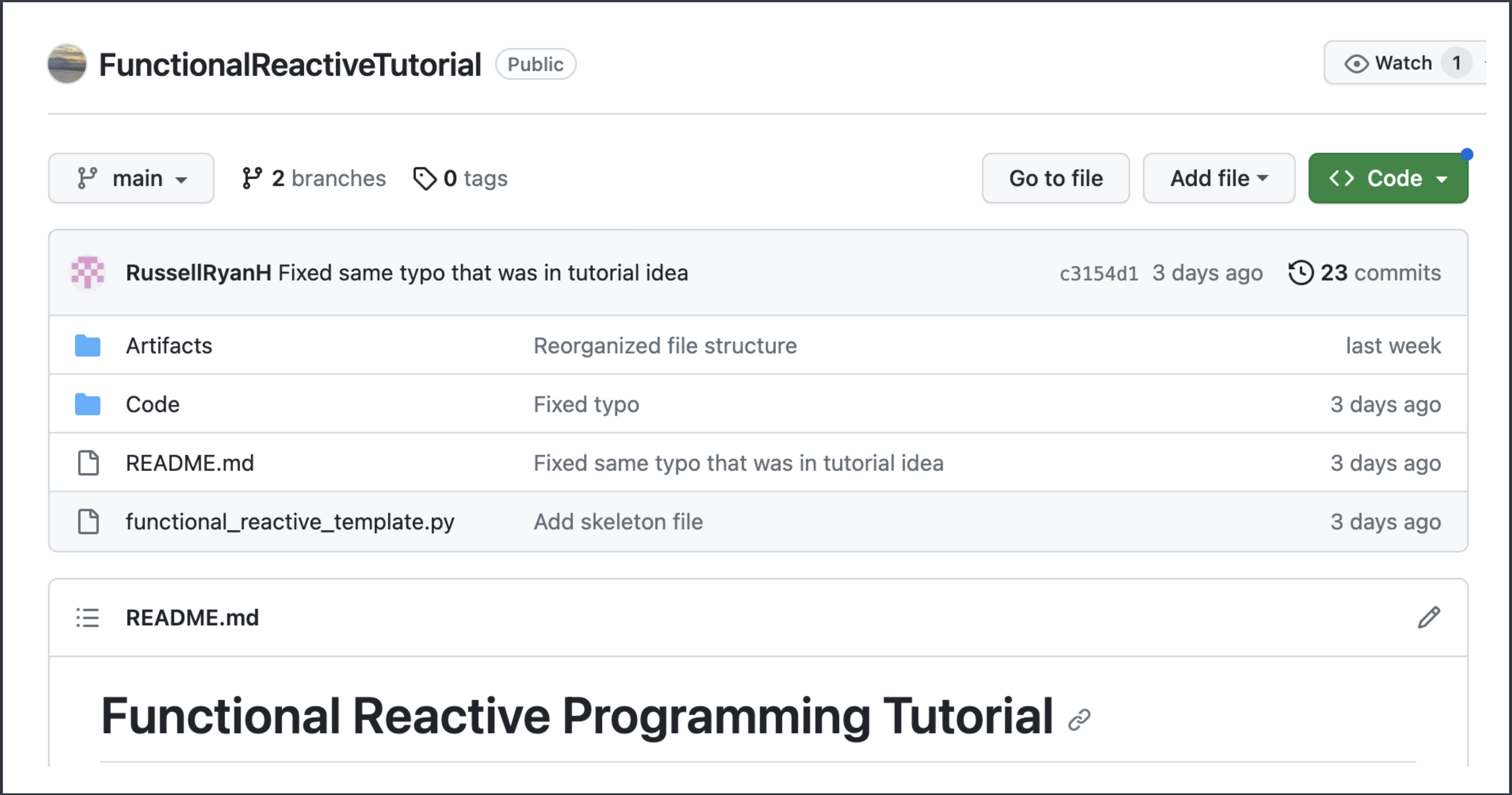
Nolle, T. (2021, March 24). *What is reactive programming? what you need to know*. App Architecture. https://www.techtarget.com/searchapparchitecture/definition/reactive-programming

Vishalxviii. (2022, June 28). *Functional Programming Paradigm*. GeeksforGeeks. https://www.geeksforgeeks.org/functional-programming-paradigm/

What is functional reactive programming | Saturn Cloud Blog. (2023, June 13). https://saturncloud.io/blog/what-is-functional-reactive-programming/

What is reactive programming? What you need to know. (n.d.). App Architecture. https://www.techtarget.com/searchapparchitecture/definition/reactive-programming

TUTORIAL


FunctionalReactiveTutorial Public

RussellRyanH Fixed same typo that was in tutorial idea c3154d1 3 days ago 23 commits

Artifacts	Reorganized file structure	last week
Code	Fixed typo	3 days ago
README.md	Fixed same typo that was in tutorial idea	3 days ago
functional_reactive_template.py	Add skeleton file	3 days ago

README.md

Functional Reactive Programming Tutorial