David Hatcher

COP3331

Dr. Tempestt Neal

4/24/2020

Lab 11 Notes Functional Reactive Programming

- Proactive – one object, A,t pushing a new state to another object, B, that receives the command to change states
    - Object A controls object B
    - Non-function as one object mutates the state of another object
- Reactive – one object, B, listens to events with the another object, A, and changes it's state when the even changes. (ex onEventListener("mousedown").)
    - Object B controls itself based on the states of object A
- Functional programming – the idea that no values are ever updated and always immutable
    - Giving the same inputs a function must always return the same outputs. They cannot rely on external states
    - Pure functions, functions that follow the functional programming paradigm, must not access the state of the class it is contained in.
    - Pure functions cannot consume or mutate external states
    - Uses other functions to add values to new objects
    - Map() using this allows you to make a copy of a container with different values based on the function you're providing as the mutator
    - Differs from OOP as OOP tends to use objects who's states mutate and adjust based on what is going on with the data it's being provided or the functions being called on it
- Functional Reactive Programming-
    - Combines the functional ideas and reactive ideas
    - Uses pure functions to create a full stream of all state changes into a single stream
    - Uses streams of state changes to combine all changes to be able to watch for any events that need to be reacted to
    - We use functional operations on this stream to make these changes