FIT2102 Programming Paradigms 2018

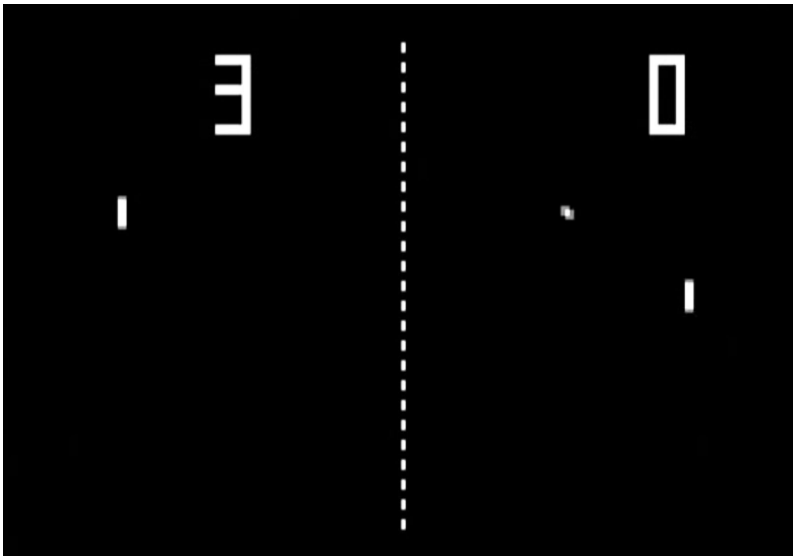
Assignment 1: Functional Reactive Programming

Due Date: Friday 7th September 2018 at 5pm

Weighting: 20% of your final mark for the unit

Submission Instructions: A zip file should be submitted via moodle by the due date. It should contain all the code for your program along with all the supporting files. It should include a tsconfig.json file such that running the tsc command from the top level directory builds all the necessary javascript files, and the "app" should be hosted by an index.html file, also at the top level. This index.html file should preserve the three links from the original but otherwise can be arranged how you like. However, it should be very clear how to access the different components of your app, the tests and the base examples as discussed below. It should include sufficient documentation that we can appreciate everything you have done. As discussed below, it should not include any external libraries (other than the test framework libraries supplied with the starter code).

Task Description: In this assignment we will use the Observable stream created for the Week 4 worksheet to create the classic pong game (you tube) in an SVG image hosted in the pong.html webpage. The youtube video is meant to give you an idea of the basic gameplay, but yours needn't look the same or work in precisely the same way. The baseline functionality required for a passing grade, and a list of alternative ideas for achieving an HD are listed below.



Minimum: all of these requirements must be reasonably executed to achieve a passing grade (detailed marking rubric below).

- Ensure all the tests pass (despite any changes you may have made to Observable) and that the basic SVG examples work
- Implement a basic one-player pong game with a paddle movable by mouse while the mouse cursor is over the svg canvas. The second paddle should follow the ball.
- Indicate the scores for each player
- The game should end and indicate the winner, first to 11 points.

If all of the above are implemented in good functional style up to a D can be achieved. To get a higher grade you have to use a little creativity and add some functionality of your own choice - suggestions below.

Ideas for getting an HD. Any one or more of the following (or something of your own devising with a similar degree of complexity) done well (on top of the basic functionality described above) will earn you an HD provided it is implemented using the functional programming ideas we have covered in lectures and tutes:

- Extend Observable with additional methods that help to make your code cleaner or more "fluent". See Rx.js for ideas.
- Add error handling to Observer/Observable (as per ES.Next proposal)
- Create unit tests to tests/main.test.js
- Incorporate gameplay from other classic arcade games: e.g. breakout, galaga, etc.
- Advanced (not recommended unless you already know how): Make a distributed multiplayer version, wrapping the comms in Observable (you'll have to make your own server for this).
- Your own ideas!

Some of the above will require a little independent research. That's what computer science is all about.

Additional Information. Similar to the tutorial exercises we will provide you with a starter project which you can download in a zip file from Moodle that contains:

- **observable.ts** you will work on an implementation of Observable in the Week 4 tutorial worksheet. After you've had a chance to do this, you will be given a basic, working version of Observable with the assignment code bundle. You can use, and possibly extend, the given Observable for this assignment.
- **basicexamples.html** this is probably where you should start. You need to get these basic interactive observable demos working source is in basicexamples.ts (below).
- **basicexamples.ts** source for basicexamples.html. In your Week 5 tutorial you will need to get all of these working.
- **checklist.html** mocha tests of basic observable functionality - ensure that these still pass despite any changes you make to observable and add more tests to the end of test/main.test.js as you see fit.

- **Index.html** this will be the first file we will open when we mark your work. Currently it simply contains links to the three other html files we are giving you. Add text to this page to explain which files you have changed.
- **pong.html** currently just an empty SVG object. Your pong game goes here! Add instruction text to the page to explain how to play your pong game. Make sure the instructions are sufficient that we can properly try out all of your features.
- **pong.ts** source code for your pong game. This is where most of your work will go.
- **svgelement.ts** contains a little helper class for working with SVG elements. This file should let you do everything with SVG necessary to complete the minimum requirements. Feel free to add additional helper code here, but be sure to document it.

Tips for getting started:

- Complete the Week 4 Tutorial Worksheet Observable exercises and begin studying Observable in the course notes.
- In week 5 Tutorial you will complete basicexamples.ts
- The final task in basicexamples.ts was to implement the function drawAndDragRectsObservable(). Once you have this working, use similar ideas to begin adding functionality to pong.ts as above.

More Tips:

- Finish all the javascript and typescript tutes up to the first part of Week 5 first. They are designed to give you the experience you need to prepare you for this assignment.
- Come to the lectures and tutes for important tips and assistance. You'll need to anyway to get the participation marks.
- Come see the Lecturer and the tutors in their consulting times. We try to get to your emails but we are flooded.
- **Start as soon as possible**. Don't leave it until it's too late.

Plagiarism:

We will be checking your code against the rest of the class and the internet using a plagiarism checker. Monash applies strict penalties to students who are found to have committed plagiarism.

Marking Rubric:

It is important to realise that (as stated above):

- If you implement the **Minimum** requirements above demonstrating application of functional programming ideas from our lectures and tutes you will achieve a pass grade.
- You can receive up to a D for perfectly implementing the Minimum requirements demonstrating a thorough understanding of how to use Observable to write clean, clear functional UI code.
- To achieve HD and up you will need to do the above plus some aspect of additional functionality as described above.
- Rubric table next page:

	Mark%	P	C	D	HD
Pong Game	30	An attempt has been made to create a pong game	Most of the minimal requirements implemented	Minimal requirements fully and elegantly implemented.	Implements ideas beyond the minimal requirements such as (but not limited to) the suggestions above
Functional Programming and general coding style	30	Attempt has been made to use Observable and apply functional programming concepts	Evidence that the student has applied concepts from the lectures and course notes	Good use of Observable to create logical function chains	Small pure functions used as much as possible Impure functions (changing the state of the diagram elements) avoided except in subscribe methods. Extensions to Observable such as (but not limited to) additional operators or error handling.
Type correctness	20	Attempted			Minimal use of <any> type Use of Generic types to make functions and classes created as reusable as possible.
Well documented code	20	Attempted			There are clear comments at the start of all files that have been modified by the student outlining their changes to the given code. The design of their functionality and the way it uses functional programming principles from the lectures and course notes should be clearly explained. The comments should make it clear what extensions have been added to the provided code (and why they have been added) - such as the Observable or Elem classes.