FIT2102 Programming Paradigms 2017

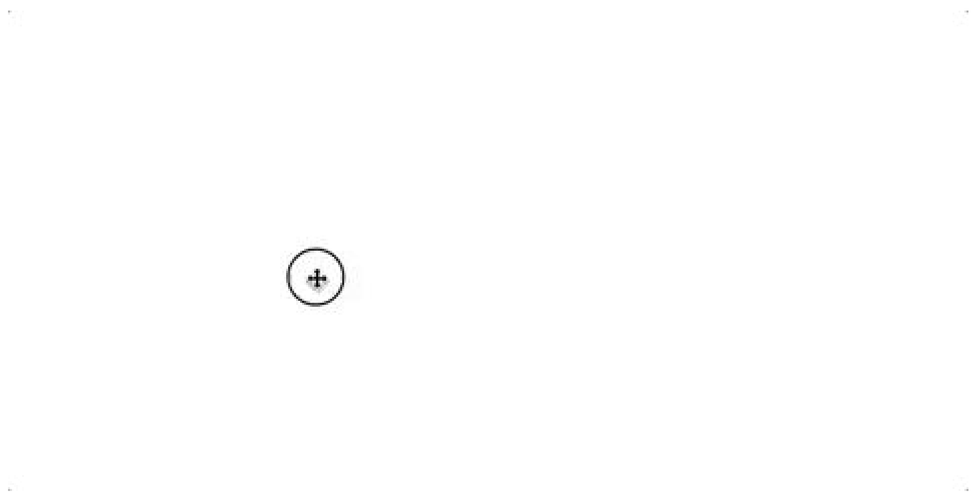
Assignment 1: Functional Reactive Programming

Due Date: Friday 8th September 2017

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 a basic SVG diagram editor. The following animation is meant to inspire you, but yours needn't look the same or achieve the same functionality. The baseline functionality required and a list of alternative ideas for achieving an HD are listed below.



Minimum - all of these must be reasonably executed to achieve a passing grade (detailed marking rubric to follow).

- Make the all the tests pass and the basic SVG examples work
- Implement a basic diagramming editor with draggable shapes and lines connecting them and that stay attached to the shapes as they are dragged
- A palette of different shapes (e.g. rects/ellipses/other polygon) that can be added to the canvas through (e.g.) drag and drop
- UI allowing connectors to be added to link pairs of shapes

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 with a similar degree of complexity) done well will earn you an HD provided it is implemented using the functional ideas we have covered in lectures and tutes:

- UI to make your shapes resizable and/or rotatable
- Implement a simple physics model
- Implement editable labels on nodes with a blinking cursor
- Generate diagram nodes and links from wikipedia API
- Run an animated simulation on your digital circuit diagram using the simulator from worksheet 2

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:

- **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. Before you submit you should add text here that gives a high-level overview of everything you have done.
- **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).
- moch**checklist.html** a tests of basic observable functionality - you will need for all of these to pass. Feel free to add more to the end as you see fit.
- **diagrameditor.html** currently just an empty SVG object. Your app goes here!
- **observable.ts** in which you will need to create your observable (begin by transferring the Observable you created for the Week 4 tutorial worksheet.
    - You will need to implement Observable.flatMap to make the tests pass
- **basicexamples.ts** source for basicexamples.html. You will need to get all of these working by correctly implementing Observable in observable.ts. There is one additional incomplete function here, drawAndDragRectsObservable that you will need to get working as per the description. This should be a step toward your diagram editor, which you will implement in:
- **diagrameditor.ts** source code for you diagram editor
- **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:

- Start transferring your tute 4 Observable code into observable.ts
- Add flatMap to Observable to get the test passing
- Follow the instructions in the comments at the end of basicexamples.ts
- The final task in basicexamples.ts was to implement the function drawAndDragRectsObservable(). Once you have this working, copy this code to diagrameditor.ts and begin adding functionality as above.

More Tips:

- Finish all the javascript and typescript tutes up to Week 4 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 Tim and Andrew 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.

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.

	Mark%	P	C	D	HD
Observable fully implemented	15	Implementation of observable sufficient for use in the assignment	Observable tests pass, All Basic Examples Working	Observable well documented, indicating good understanding	Not absolutely necessary but if additional methods have been added to Observable this will be highly regarded (see Rx.js for ideas).
Diagram Editor Functionality	30	An attempt has been made to create a diagram editor with the ability to create multiple shapes	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	30	Attempt has been made to use	Evidence that the student	Good use of Observable to	Small pure functions used as much as possible

and general coding style		Observable and apply functional programming concepts	has applied concepts from the lectures and course notes	create logical function chains	Impure functions (changing the state of the diagram elements) avoided excepted in subscribe methods.
Type correctness	10	Attempted			Minimal use of <any> type
Well documented code	15	Attempted			It is clear from the inline comments and comments in declarations of functions and methods that the student understands and has thought about the design and functionality