3 - Reactive Web Programming and Streams

1. Learning Outcomes

On completion of this lab you will have:

- Learn how to handle asynchronous events using ReactiveX/RxJS streams
- Implement a stopwatch application using streams with a graphical rendered UI (Canvas)

2. Organisation

Please complete the exercises individually.

3. Grading

This worksheet is worth up to 10% of your overall module grade. You must attend and sign in at a minimum of 10 labs in the semester in order to obtain full CA credit.

You may work on this worksheet during lab 5 and lab 6 with instructor assistance. You may also be requested to demonstrate your submission to the lab instructor in order to receive credit.

4. Submission

The deadline for submission is Sunday Nov 04, 2017 @23:59 through Github.

The work and submission workflow is as follows:

- Before you start working on your worksheet you must **fork** a copy of the official class repo from here (or continue with the fork you have made for a previous worksheet)
    - https://github.com/bjg/2018-ditcs-cmpu4043.git
- Then **clone** the forked repo to your development machine (or use your existing clone)
- The make a new branch in your cloned, local repo named as follows:
    - ***<student-id>***-wks-3

  where ***<student-id>*** is something like C12345678
- When you are finished developing your worksheet solution then you must push your local repo to the remote origin for that branch
- Finally, when you are submitting your solution for grading, you will generate a pull request (PR) requesting that your branch is merged with the remote origin master branch of the official class repo above

- If you are not sure about any of the described steps here, then take a look at this worked demonstration:
    - https://www.youtube.com/watch?v=FQsBmnZvBdc

- **UPDATE**: As some of you have pointed out, the instructions above only work for public Github repos. If you want to use a private repo for your working copy, then take a look at the method described here: https://help.github.com/articles/duplicating-a-repository

3. Demonstration

You will give a brief demonstration of your submission to the lab instructor in lab 7.

4. Requirements

For this lab you will need to look at the lectures and tutorials in section 2 of the lectures on Webcourses (Reactive Web Programming and Streams)

- Use your own laptop with local tools (See tutorial video **20 - Node and Webpack** (developer ecosystem theme folder on Webcourses) for an example environment you could consider using).
- Take the free online first lesses from Egghead - Introduction to Reactive Programming
- Watch the counter tutorial videos **22** and **23** (Webcources folder). Note that the API has changed a little (version 6) since those tutorials were recorded so consult the v5 to v6 migration notes if you run problems)

5. Resources

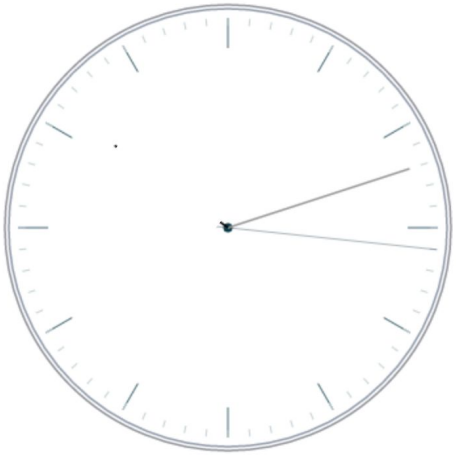
You are free to research whatever you need to solve the problems in this lab. Some recommended resources include:

- https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
- https://github.com/ReactiveX/rxjs
- https://github.com/ReactiveX/rxjs/blob/master/docs_app/content/guide/v6/migration.md
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

6. Problem Sets

Provide Javascript ES6 code for the following problems using in your own development environment

1	Convert your calculator from **worksheet 2** to use RxJS streams to handle the keyboard	30

	and mouse input events **instead of** the method you originally used (probably callbacks or promises). All other calculator functionality can stay the same. **Hint**: Merge and map your input events into a single stream and execute display updates inside the stream processing logic	
2	Build a stream-driven stopwatch application. **Hint**: Use Observable.interval() to generate your timer ticks. Your UI should implement **start**, **stop, split** and **reset** buttons for your timer. The stopwatch should also display **minutes**, **seconds** and **10ths of seconds** in digital format (beside the clock face) updating as it counts. Splits should be shown as a list separate from the timer. You should support 5 recorded split times in your UI. The reset button should stop and reset the timer and remove any split times. You should also provide a graphical version of your stopwatch which shows a **minutes** and **seconds** hand along the lines of the following example. You are free to come up with your own clock face design if you wish.  **Hint**: Use the JS Cavas API to build the clock face and animate the hands. There will a little trigonometry required to animate the hands	50

7. Lecture Review Questions

(1-2 paragraphs/bullets, etc) answers for the following. It is important that you have reviewed the module material in advance

1	Explain what is meant by the stream abstraction. What is the relationship between streams and the observer pattern? What are streams useful for modeling and when might you use them in Rich Web development?	10 Marks
2	Assume that you are building an interface to an API in your Rich Web App. Describe in detail how you could use the RxJS library to handle asynchronous network responses to API requests. In your opinion, what are the benefits to using a streams library for networking over, say, promises? And what do you think are the downsides?	10 Marks