# 5 - Streams Programming

## 1. Learning Outcomes

On completion of this lab you will have:

- Learn how to handle asynchronous events using RxJS streams
- Implement a stopwatch application using streams with a graphical rendering

## 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 the labs in order to obtain the credit for the associated worksheets. You may work on this worksheet during lab 4 and lab 5 with instructor assistance. You must also demonstrate your submission in order to receive credit - see below.

## 4. Submission

The deadline for submission is Wednesday Nov 23, 2016 @23:59 through Webcourses.

## 5. Demonstration

You will give a brief demonstration of your submission to the lab instructor in lab 11.

## 6. Requirements

For this lab you will need to

- 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).
- Watch the counter tutorial videos **22** and **23** (in the reactive programming theme folder on Webcourses)

## 7. Resources

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

- https://egghead.io/courses/introduction-to-reactive-programming
- https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
- https://github.com/ReactiveX/rxjs
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

## 8. Problem Sets

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

When submitting, use a zipped up version of your code with the `node_modules` folder removed or submit a link to a private (shared with me) git repo such Github or Bitbucket.

**NOTE**: Do not check your code into a public repo and there is a reasonably high likelihood of it being plagiarised.

| | | |
|---|---|---|
| 1 | Convert your calculator from worksheet 3 to use RxJS streams to handle the keyboard and mouse input events.<br><br>**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.<br><br>**Hint**: Use `Observable.interval()` to generate your timer ticks.<br><br>Your UI should implement **start**, **stop, split** and **reset** buttons for your timer.<br><br>The stopwatch should display **minutes**, **seconds** and **10ths of seconds** in digital format updating as it counts.<br><br>Splits should be shown as a list separate from the timer. The reset button should stop and reset the timer and remove any split times.<br><br>You should also provide a graphical version of your stopwatch which shows a **minutes** and **seconds** hand along the lines of the following example | |

| | | |
|---|---|---|
| | **Hint**: Use the JS Cavas API to build the clock face and animate the hands. | |