**Week 1 Day 2**

**Test-Driven Development**

* **Test-Driven Development** is a method of adapting code that involves three stages:
  1. Write a failing test (and make sure the test fails for the right reasons) for the most basic functionality possible
  2. Write the most basic code possible to achieve that functionality
  3. Polish up the code to make it maintainable and fit the required functionality
* This methodology ensures quality by:
  1. focusing on requirements before writing the code.
  2. assisting in keeping the code clear simple and testable by breaking it down into small achievable steps
  3. providing documentation about how the system works for anyone coming onto the team later on
  4. building a suite of repeatable regression tests
  5. acting as an enabler for rapid change
* con of TDD – need strict requirements to do it
* A **unit test** is a simple automated test that fulfils requirements
* A **test suite** is a set of **test cases** that intends to show that a software programme fulfils a set of behaviours
* **Refactoring** is the process of restructuring existing computer code—changing the factoring—without changing its external behaviour.
* **Test coverage** a technique designed to check whether test cases are actually covering the application code. It checks what percentage of the source code is executed when a test suite is run.
* We fail a test on purpose to ensure we write minimum code possible – makes code more versatile
* Always put a variable that you’re using in a test if an object is being referred to
* Always keep tests in a separate file

**Deno Testing Library**

* *Deno.test()* registers a test case. It is a function that takes two inputs, a message to yourself and a function. You need to return a value in your function. Syntax:

*Deno.test("url test", () => {*

*const url = new URL("./foo.js", "https://deno.land/");*

*assertEquals(url.href, "https://deno.land/foo.js");*

*});*

* *assertEquals()* checks if a test is equal to the expected value (returns Boolean). Assert in general tests a piece of code. First argument is value you are checking, second argument is expected answer.
  + AssertionError: tells you that assert has failed
* Red/green means what you are getting (failure) vs what you should be getting
* *deno test* runs a test in the command line

**Other Notes**

* *import* takes properties from an object at a web address. You are basically importing function from a web address
* *export* allows you to access a function in different files
* *../directory* refers to two folders outside of current directory
* *.toFixed* sets number of decimal places
* *…* is spread operator – takes a copy of a variable – avoids pointing to original variable
* *\t* is a tab in a string
* **Destructuring** is a very useful way to manipulate objects
* Use str for string, arr for array, dt for date etc.