Short introduction to Cucumber in JavaScript

**Introduction**

Cucumber is a software testing tool. It uses a language called Gherkin to support behavior-driven development. A feature (story) is written in plain, business-friendly language. This feature is converted to test code that will initially fail. Functionality can now be developed until all pre-written tests succeed.

**Prerequisites**

* NodeJS (with npm)
* Visual Studio Code (Recommended)

**Excercise**

* Start a new nodeJS project. (npm init)
* Add the latest Cucumber library to your project (npm install @cucumber/cucumber –save-dev)

We add the library to the devDependencies because we don’t need the testing tool at runtime.

* Create a */features* folder
* Create a *simple\_math.feature* file in this folder
* Add simple feature to this file:

Feature: Simple maths

In order to do maths

As a developer

I want to increment variables

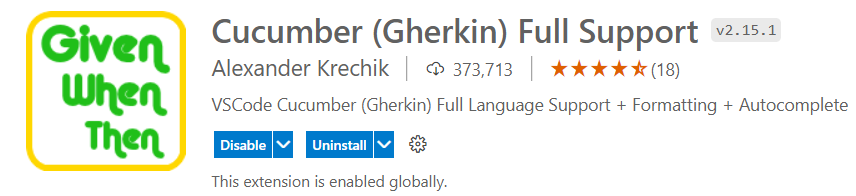
Scenario: easy maths

Given a variable set to 1

When I increment the variable by 1

Then the variable should contain 2

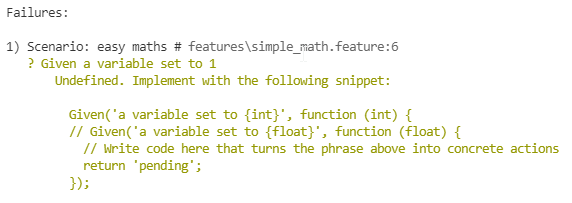
For language support in feature files, the following extension is available for VS Code



* Start the test from the root folder: ./node\_modules/.bin/cucumber-js (bash)

In the console, all failed tests will be displayed and boilerplate will be provided for each.

Example:



* Copy the boilerplate code to a new javascript file. This file needs to be inside the /features folder to be found by Cucumber.

Use import:

const { Given, When, Then } = require("@cucumber/cucumber");

* You can now implement the functionality to make the tests work.

Use the following class:

class SimpleMathsClass {

  constructor() {

    this.variable = 0;

  }

  setTo(number) {

    this.variable = number;

  }

  incrementBy(number) {

    this.variable += number;

  }

}

**Extra**

* Cucumber also supports running multiple variances of the same test. Use this simple to find out how this works:

Scenario Outline: much more complex stuff

Given a variable set to <var>

When I increment the variable by <increment>

Then the variable should contain <result>

Examples:

| var | increment | result |

| 100 | 5 | 105 |

| 99 | 1234 | 1333 |

| 12 | 5 | 17 |