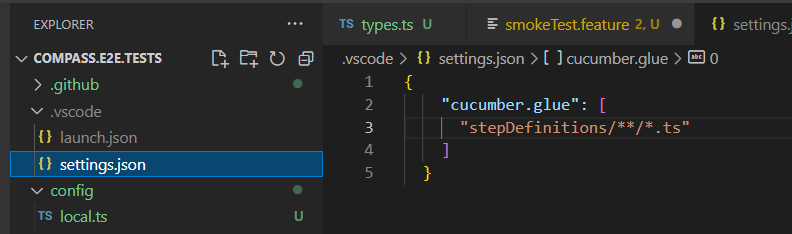
Cucumber, Playwright, Typescript

# Use VSCode

Add extension Cucumber

Add the settings.json file if it doesn’t exist in .vscode folder. Specify the step definition location for “cucumber.glue”, otherwise the extension can’t find the definition.



# Debug

Create launch.json in .vscode folder:

{

    // Use IntelliSense to learn about possible attributes.

    // Hover to view descriptions of existing attributes.

    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387

    "version": "0.2.0",

    "skipFiles": [

      "inspector\_async\_hook.js",

      "async\_hooks.js"

    ],

    "configurations": [

        {

            "type": "node",

            "request": "launch",

            "protocol": "inspector",

            "console": "integratedTerminal",

            "name": "Launch Features",

            "stopOnEntry": false,

            "program": "${workspaceRoot}/node\_modules/@cucumber/cucumber/bin/cucumber-js",

            "args": [

                "/src/features/\*\*/\*.feature",

                "--require",

                "/src/stepDefinitions/\*\*/\*.ts",

                "--require",

                "src/pages/\*.ts",

                "--require",

                "/src/support/\*.ts",

                "--require-module",

                "ts-node/register",

                "--world-parameters",

                "{\\\"env\\\":\\\"local\\\"}",

                "--format",

                "summary",

                "--tags",

                "@debug"

              ],

              "cwd": "${workspaceRoot}",

              "runtimeArgs": [

                "--nolazy"],

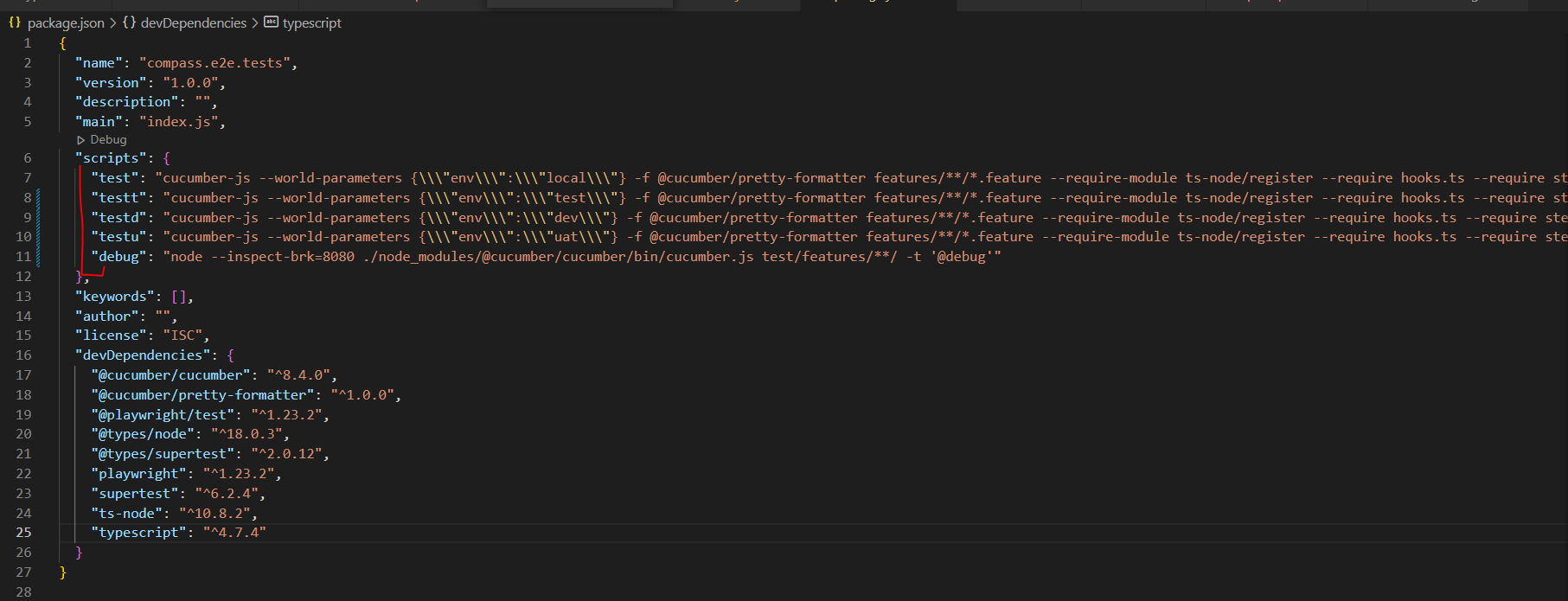
            "sourceMaps": true

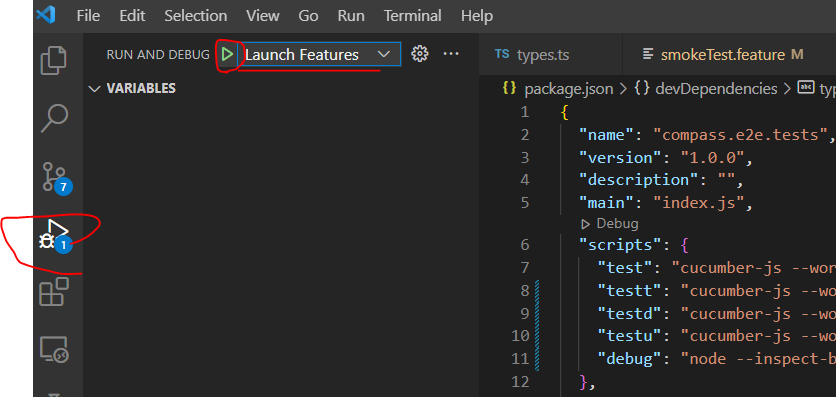
            }

          ]

        }

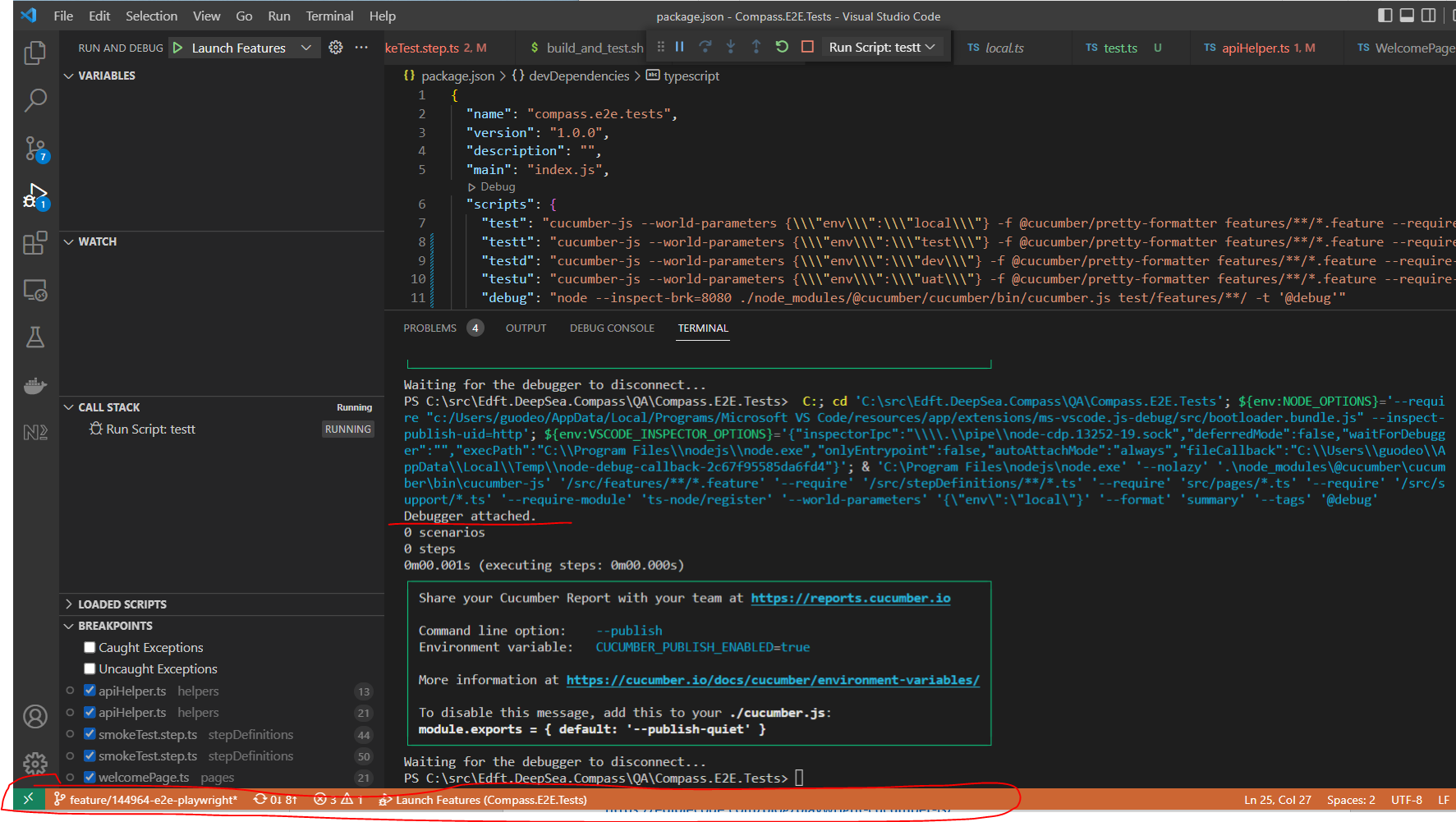
Specify the scripts in package.json file



In Debug view, Run and Debug: launch features shows as an option – came from the launch.json file. 

Run it by clicking on the Play button.

The debug status bar shows as orange. Debugger attached.



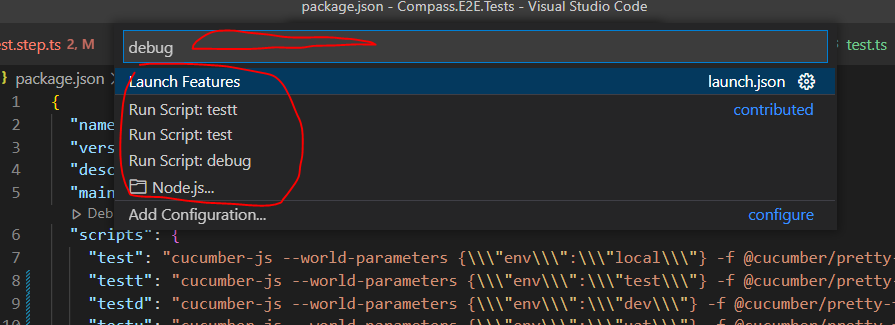
Now that the debugger is attached, can run the tests and start debugging in a few ways:

## 1. type in terminal: npm run testt (or any script exists in package.json, e.g.: npm run testd)

Note: this doesn’t seem to work half of the time… it just runs test without debugging.

## 2. click on the status bar

then type in here (or select an existing option): npm run test



## 3. Or select the option and run from here

use method 2 or 3 to debug

# Cucumber

<https://ediblecode.com/blog/playwright-cucumber-js/>

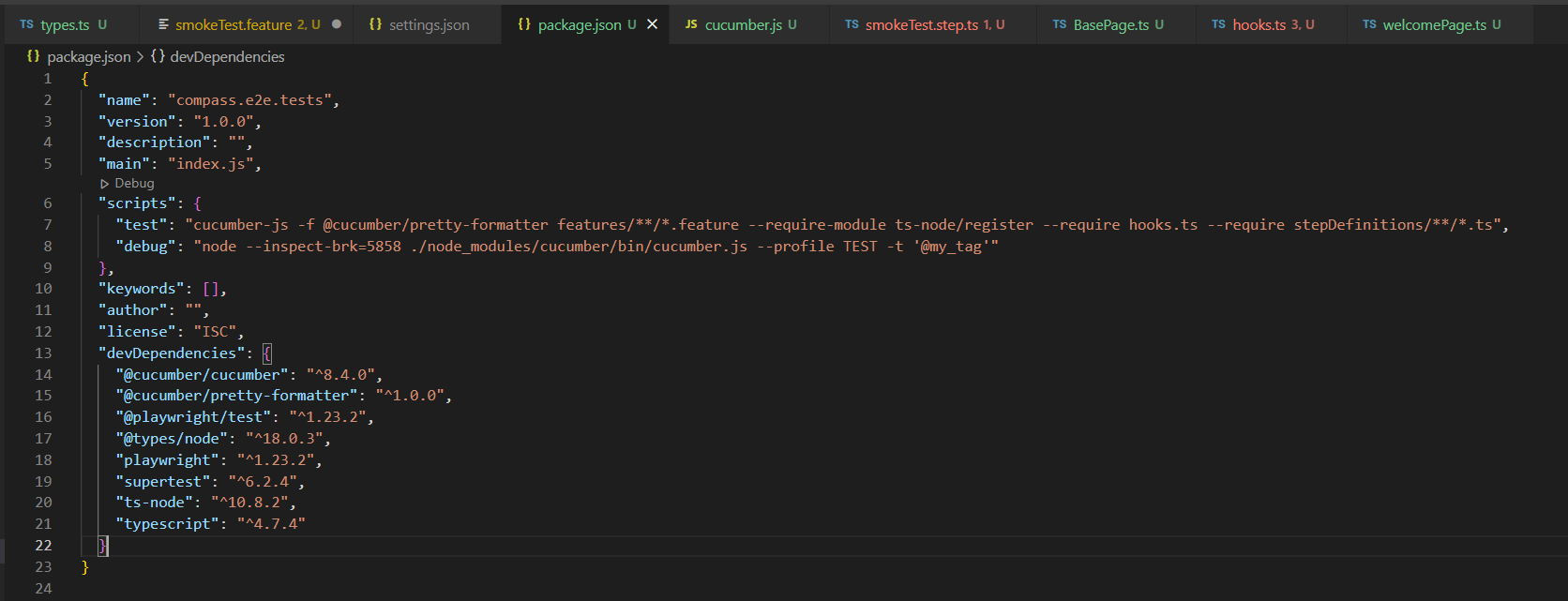
<https://dev.to/jankaritech/behavior-driven-development-bdd-using-playwright-n1o>

## Package.json

Add to “script” – so that ‘npm run test’ picks it up and runs this command

"test": "cucumber-js -f @cucumber/pretty-formatter features/\*\*/\*.feature --require-module ts-node/register --require hooks.ts --require stepDefinitions/\*\*/\*.ts",

* -f pretty-formatter is to produce result in gherkin style
* features/\*\*/\*.feature - specifies feature files location
* --require hooks.ts – this is the hooks file we created to set up browser, etc
* --require stepDefinitions/\*\*/\*.ts – specifies the step files location
* Note: the “debug” script command is not working…



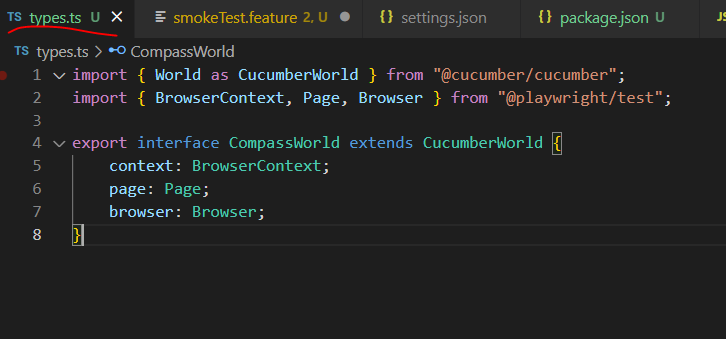
## Use hooks to set up

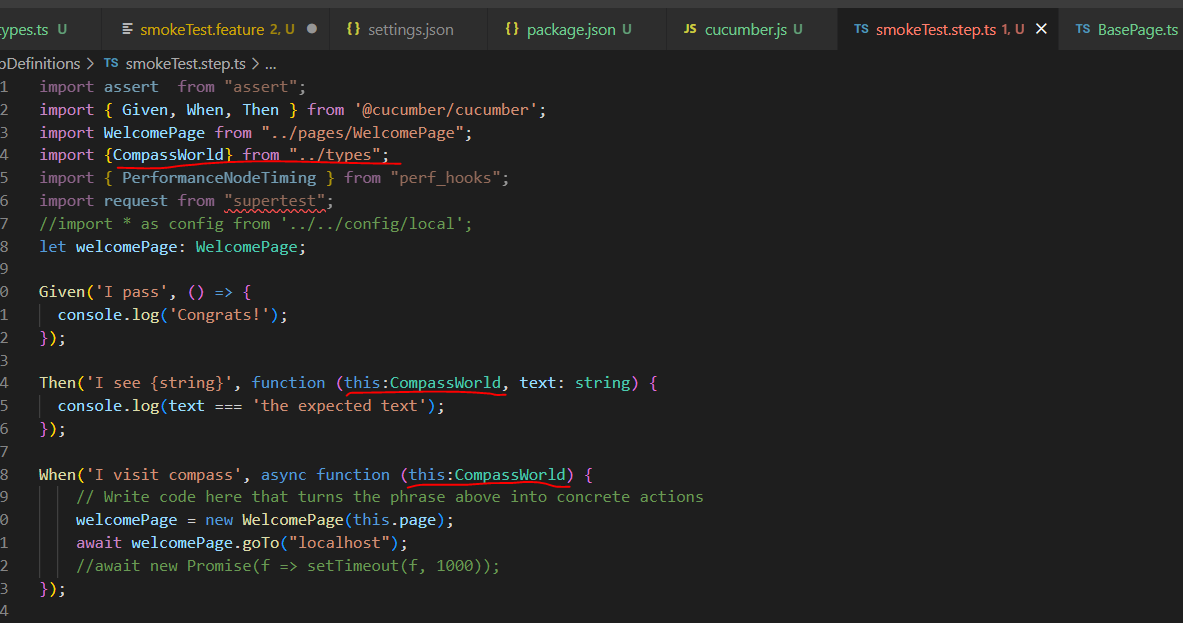
* BeforeAll set global browser – per test run
* Before scenario to create a page object and add to the World object

# Dependency Injection with Word object

<https://github.com/cucumber/cucumber-js/blob/main/docs/support_files/world.md>

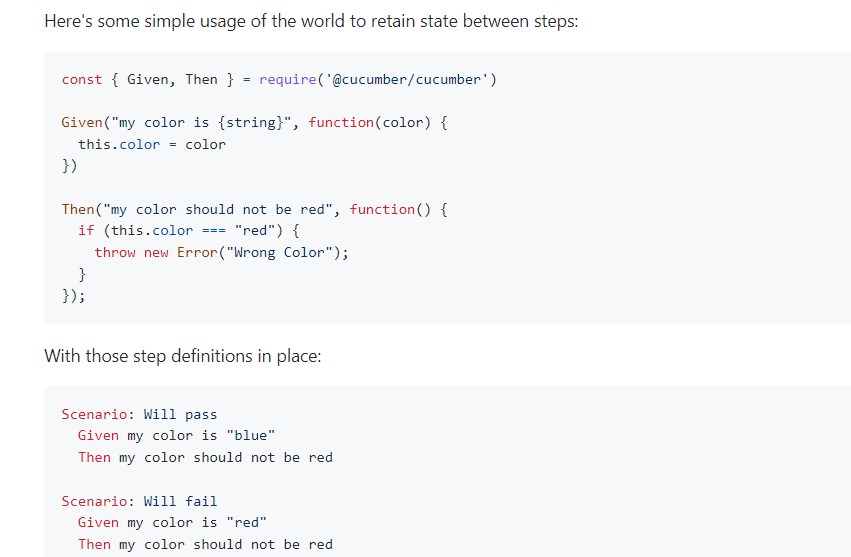
In each step definition, ‘this’ refers to the World object.

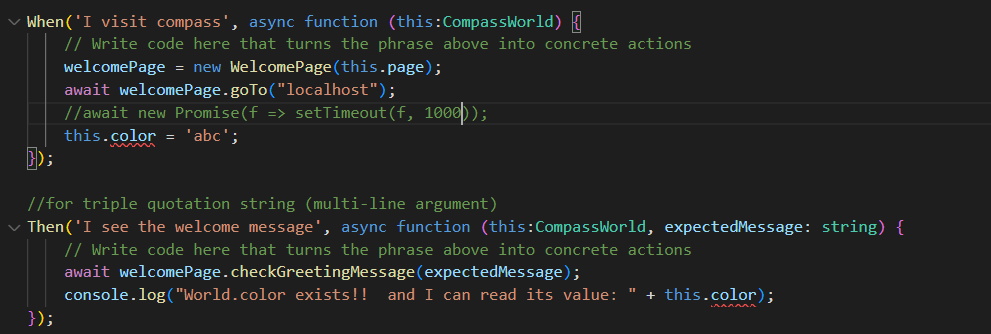
To use Playwright components, extend World class in types.ts file 

In step definitions, import the extended World object, pass it into each step method 

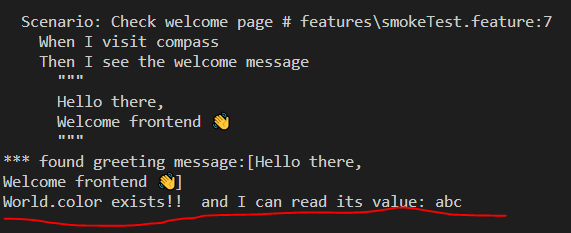
The World object has the scope of the whole test scenario.

## Sharing values between steps through World

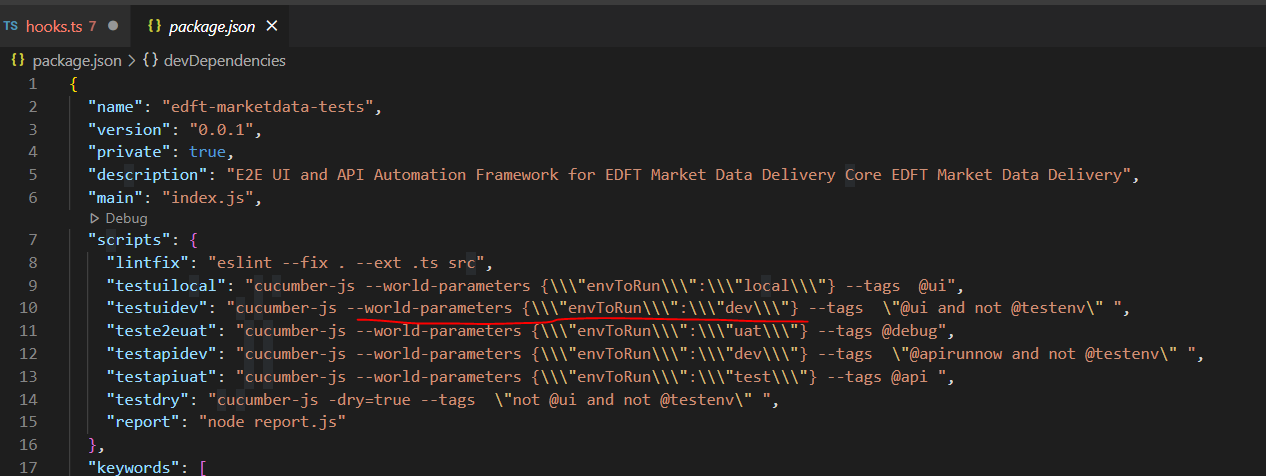


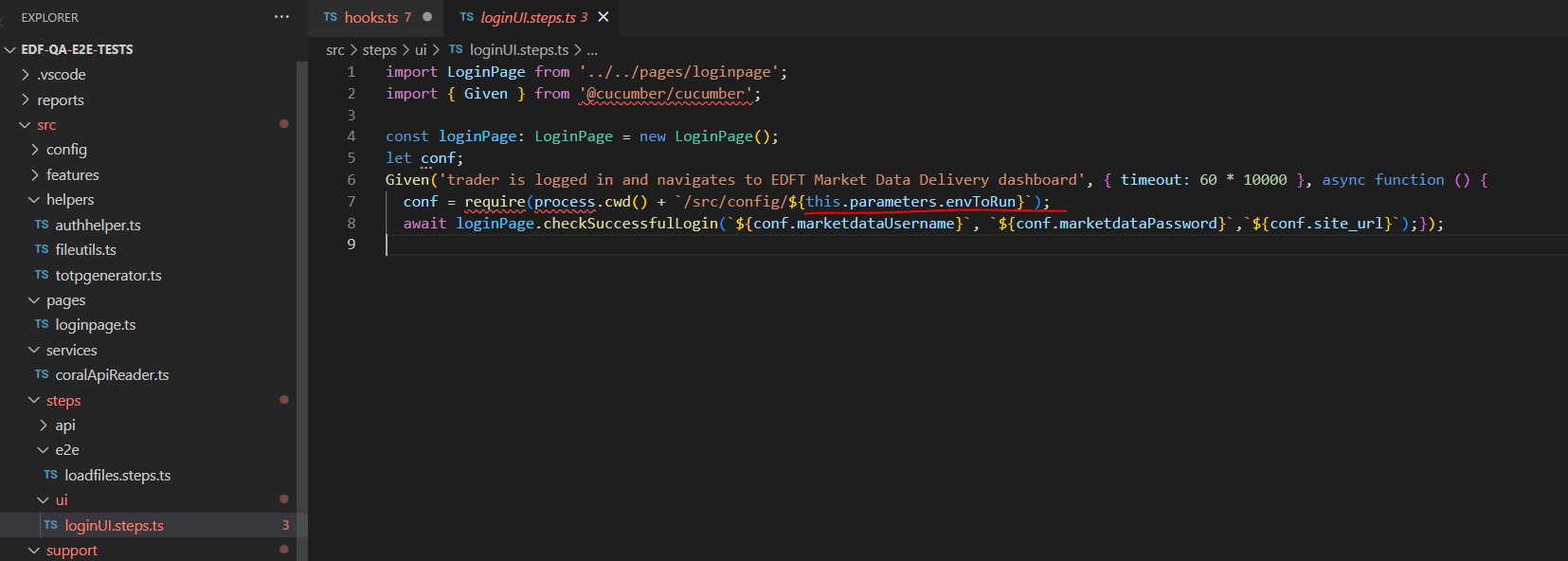


No need to declare color as a property of World – just do this.color to create and set the property.

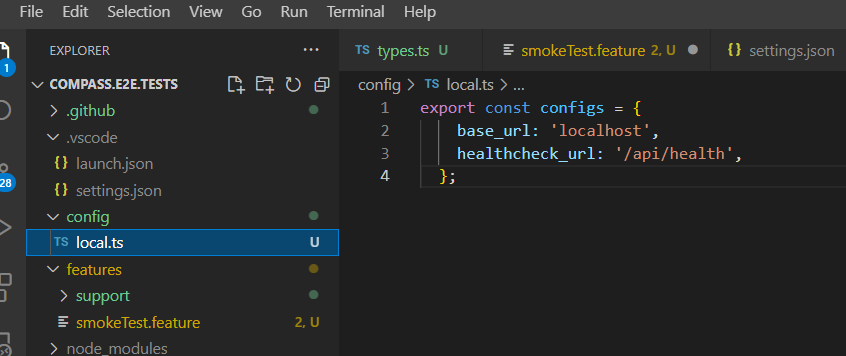
Somehow VSCode complains this.color, but it runs fine 

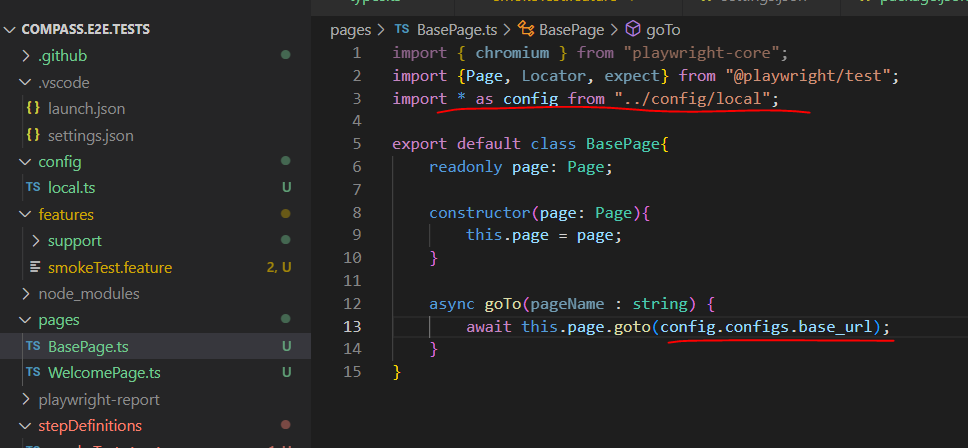
## Pass environment variables into World parameter

In package.json, scripts, the CLI command, specify the environment variable and value as –world-parameters: 

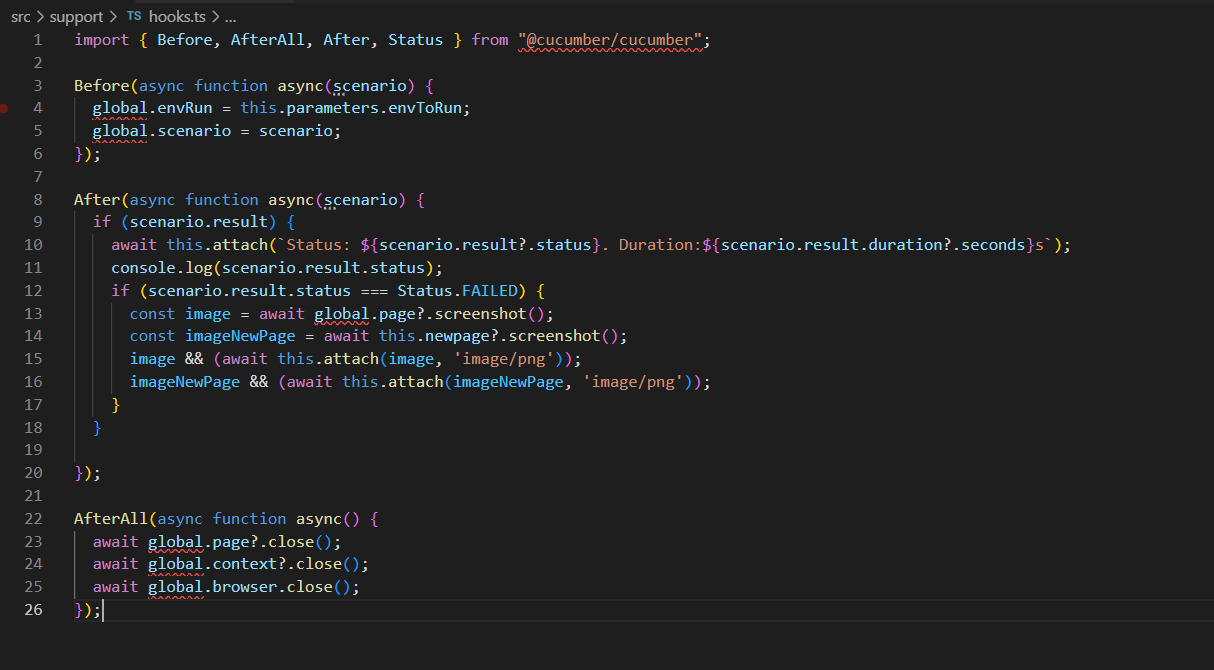
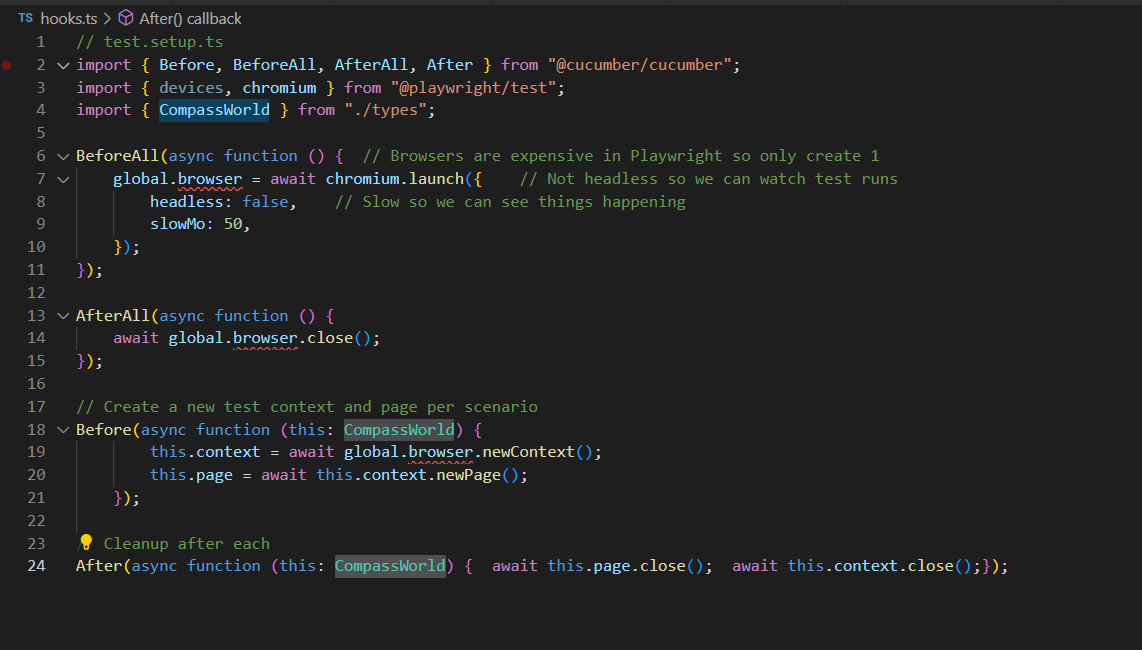
In the step definitions, access the environment variable through World object (this)

# Config

Use config files to store configuration data

Import the config file and reference it directly 

A better way would be to set up the config object at test setup (hooks) so that later steps we can just reference to that config object instead of reading from the config file each time. However, this may require creating a config class definition… just read each time for now.

Use World object (‘this’ in step def) for context injection per scenarioIssue with ‘global’

# Playwright

<https://thats-it-code.com/playwright/playwright__find-locate-search-elements-tags/>