*The Cucumber documentation is very extensive. You can view it online at:* [*https://github.com/cucumber/cucumber/wiki/A-Table-Of-Content*](https://github.com/cucumber/cucumber/wiki/A-Table-Of-Content) *A subset is presented here.*

## Scenario

A scenario is a concrete example that illustrates a business rule.  
It consists of a list of steps.

You can have as many steps as you like, but we recommend you keep the number at 3-5 per scenario. If they become longer than that they lose their expressive power as specification and documentation.

In addition to being a specification and documentation, a scenario is also a test.  
Your scenarios are an executable specification of the system.

Scenarios follow the same pattern:

* Describe an initial context (Arrange)
* Describe an event (Act)
* Describe an expected outcome (Assert)

This is done with steps.

## Steps

A step typically starts with Given, When or Then. If there are multiple Given or When steps underneath each other, you can use And or But. Cucumber does not differentiate between the keywords but choosing the right one is important for the readability of the scenario.

### Given

Given steps are used to describe the initial context of the system---the scene of the scenario. It is typically something that happened in the past.

When Cucumber executes a Given step it will configure the system to be in a well-defined state, such as creating and configuring objects or adding data to the test database.

It's ok to have several Given steps (just use And or But for number 2 and upwards to make it more readable).

### When

When steps are used to describe an event, or an action. This can be a person interacting with the system, or it can be an event triggered by another system.

It's strongly recommended you only have a single When step per scenario. If you feel compelled to add more, it's usually a sign that you should split the scenario up in multiple scenarios.

### Then

Then steps are used to describe an expected outcome, or result.

The step definition of a Then step should use an assertion to compare the actual outcome (what the system does) to the expected outcome (what the step says the system is supposed to do).

## Step Arguments

A scenario contains multiple Given/When/Then steps. An example of a step that can have multiple values in between the double-quotes can be used to provide values to its Step Definition.

When I click on “Adopt Me”

Translates into a Step Definition written in TypeScript:

When(/^I click on the puppy "(.\*?)"$/, async (name) => {

let puppiesPage = new PuppiesPage();

let puppy = await puppiesPage.getPuppyFromRow(name);

let viewDetails = await puppiesPage.viewDetails(puppy);

await viewDetails.click();

});

name will contain the value of the string that was in between the double-quotes based upon the Regular Expression in the Step Definition.

## Data Tables

A Data Table is handy for passing a list of values to a step definition:

And I fill the form in with the following values

| name | Sad Puppy |

| email | puppy@puppy.com |

| body | I am having trouble. Please help! |

This translates into a Step Definition written in TypeScript:

When(/^I fill the form in with the following values$/,

async function (table) {

// fill in form using **table**

});

When reviewing the Cucumber documentation, the table is passed in as a Cucumber object that is essentially an array of hashes. You can then iterate over the data that has been passed in and do something within your Step Definition.

## Scenario Outline

A Scenario Outline allows you to define a scenario that can be repeated for every row in the Examples table that follows the outline. A Scenario Outline looks like this:

Scenario Outline: All fields are required to complete an adoption

Given I have added a puppy to my litter

When I tap "Complete the Adoption"

And everything is filled in

And I yet leave < order\_name > blank

And I click "Place Order"

Then I see "<error>"

Examples:

| element | error |

| order\_name | Name can't be blank |

| order\_address | Address can't be blank |

Notice the words in the **<>** brackets? These correspond to the columns of data in the Examples table. For each row past the header, the scenario will repeat and plug in the values from the table in the appropriate spots within the outline.

## Tags

Tags are a way to group Scenarios. They are @-prefixed strings, and you can place as many tags as you like above Feature, Scenario, Scenario Outline or Examples keywords. Space characters are invalid in tags and may separate them.

Tags are inherited from parent elements. For example, if you place a tag above a Feature, all scenarios in that feature will get that tag.

Similarly, if you place a tag above a Scenario Outline or Examples keyword, all scenarios derived from examples rows will inherit the tags.

You can tell Cucumber to only run scenarios with certain tags, or to exclude scenarios with certain tags.

Cucumber can perform different operations before and after each scenario based on what tags are present on a scenario.

An example of a tagged scenario:

@smoketest

Scenario: Be able to view the details of a puppy

Given I am on the home page

When I click on the first puppy

Then I see "Adopt Me!"

# Appendix

[TypeScript Cheat Sheet](https://devhints.io/typescript)

[CucumberJs Cheat Sheet](https://cheatography.com/mpie/cheat-sheets/selenium-webdriver-js-for-cucumber-js/)

[Protractor Cheat Sheet](https://gist.github.com/javierarques/0c4c817d6c77b0877fda)