# Cucumber

## Doing it right

tom-clements.com | github.com/seenmyfate
@Seenmyfate

# Good Cucumber

starts a focused conversation

helps us share a common language

helps us find edge cases

improves our communication

# Bad Cucumber

is a waste of time and energy

# Positive side effects

living, executable documentation

tracks intention

prevents regressions

# Negative side effects

more code to write

more code to maintain

complexity!

# The real benefit

clarity

but not without effort

*Do or do not. There is no try*

To get benefit from cucumber, we must be disciplined

# Features != User Stories

User Stories are a planning tool

Delete them once they're implemented

Features are a communication tool

They describe how the system works today

Every cucumber example on the internet, ever

```
Feature:
  In order to get access to the site
  As a visitor
  I want to sign in
```

# Features != User Stories

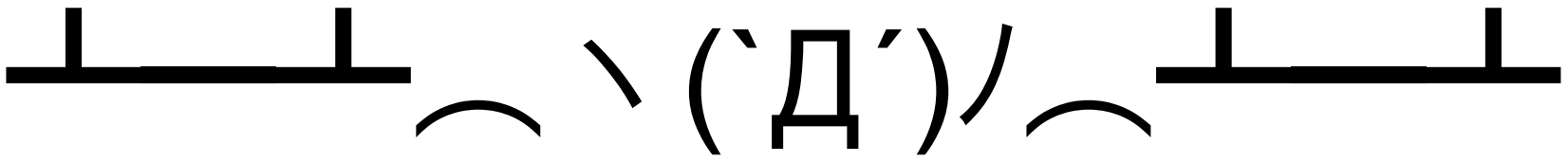
# Features describe how the system works today

```
Feature:
  The content of our site is protected.
  You must sign in before you can see it
```

Scenarios != Steps

# Every cucumber example on the internet, ever

```
Scenario:
  Given I visit the signin page
  And I fill in "username" with "tom"
  And I fill in "password" with "test"
  And I press "submit"
  Then I can see "Welcome"
  And I can see "Signed in"
  And I am on the dashboard page
```



This is unmaintainable nonsense

*"If your scenario starts with 'When the user enters 'Smurf' into 'Search' text box…' then that's far too low-level. However, even "When the user adds 'Smurf' to his basket, then goes to the checkout, then pays for the goods" is also too low-level. You're looking for something like, 'When the user buys a Smurf.'"*

Liz Keogh

# Scenarios

are acceptance criteria

written at a high level

written using consistent language

are independent and deterministic

# Steps

Write and organise your steps like you write and organise your code

- modular
- reusable
- DRY
- do one thing
- delegate to other steps

# An example

```
Scenario:
  Given I visit the signin page
  And I fill in "username" with "tom"
  And I fill in "password" with "test"
  And I press "submit"
  Then I can see "Welcome"
  And I can see "Signed in"
  And I am on the dashboard page
```

# An example

```
Scenario:
  Given I sign in
  ..
```

# Delegate steps

```
Given /^I sign in$/ do
  steps %{
    * I sign up
    * I complete the sign in form
  }
end
```

# Delegate some more

```
Given /^I sign up$/ do
  steps %{
    * I complete the sign up form
    * I confirm my email address
  }
end
```

## And some more

```
Given /^I complete the sign up form$/ do
  steps %{
    * submit the sign up form with valid information
  }
end
```

## And some more

```
Given /^submit the sign up form with valid information$/ do
  steps %{
    * I fill in "username" with "#{valid_user.name}"
    * I fill in "password" with "#{valid_user.password}"
    * I press "#{I18n.t(:submit)}"
  }
end
```

We should only ever change scenarios when the behaviour changes, not the implementation

# Happy Cuking

tom-clements.com | github.com/seenmyfate
@Seenmyfate