Watir Cheat Sheet

Browser Commands

```
# start new driver session
b = Watir::Browser.new :firefox
b = Watir::Browser.new :chrome
b = Watir::Browser.new :ie

# goto url
b.goto "http://www.centricconsulting.com"

# refresh
b.refresh

# close
b.quit
```

TextBox Interactions

```
# enter value
b.text_field(:id => "text").set "watir-webdriver"

# get value
b.text_field(:id => "text").value

# clear
b.text_field(:id => "text").clear
```

Button Interactions

```
# is enabled?
b.button(:id => "btn").enabled?

# button's text
b.button(:id => "btn").text

# click
b.button(:id => "btn").click
```

Checkbox Interactions

```
# check
b.checkbox(:id => "btn").set
b.checkbox(:id => "btn").set(true)

# uncheck
b.checkbox(:id => "btn").clear
b.checkbox(:id => "btn").set(false)

# is checked?
b.checkbox(:id => "btn").set?
```

Radio Interactions

```
# select value
b.radio(:id => "radio").set

# is var selected?
b.radio(:id => "radio").set?
```

DIV Interactions

```
# get text
b.div(:class => "body").text

# get text of 2nd div when it appears
b.divs[1].when_present.text
```

Table Interactions

```
# row 1, col 1
b.table(:id => "table")[0][0].text

# row 1, col 2 (alternate)
b.table(:id => "table").tr{0}.cell{1}.text

# row 2 - entire text
puts b.table(:id => "table")[1].text

# click row #4
puts b.table(:id => "table")[3].click

# get column count
b.table(:id => "table").row.cells.length

# row count
b.table(:id => "table").row_count
b.table(:id => "table").rows.length
```

Waits

```
# [wait_until_present]
b.button(:id => "btn").wait_until_present

# [when_present]
b.button(:id => "btn").when_present.click
b.button(:id => "btn").when_present(10).click

# [wait_while_present]
b.button(:value => "submit").click
b.button(:value =>
"submit").wait_while_present
```

Watir Cheat Sheet

Browser Commands

```
# start new driver session
b = Watir::Browser.new :firefox
b = Watir::Browser.new :chrome
b = Watir::Browser.new :ie

# goto url
b.goto "http://www.letstest.com"

# refresh
b.refresh

# close
b.quit
```

Listbox Interactions

```
# select from list text
b.select_list(:id => "list").select "var"

# select using value
b.select_list(:id => "list").select_value "var2"

# value is selected?
b.select_list(:id => "list").selected?("var2")

# get value
puts b.select_list(:id => "list").value

# get all items
b.select_list(:id =>
"list").options.each do |i|
  puts "#{i.text}"
end
```

Image Interactions

```
is image loaded?
b.image(:src => "img.gif").loaded?

# height
b.image(:src => "img.gif").height

# width
b.image(:src => "img.gif").width

# click
b.image(:src => "img.gif").click

# click 1st image
b.images[0].click
```

General Tips

```
# [exists?]
b.text_field(:id => "text").exists?

# [enabled?]
b.select_list(:id => "list").enabled?

# [present?]
b.element(:id => "e").present?

# [tag_name]
b.element(:id => "e").tag_name

# [screenshot]
b.screenshot.save "c:\\page.png"

# [to_subtype] # returns button
b.element(:id => "btn").to_subtype

# [index] click 2nd image on page
b.image(:index => 1).click

# [loops]
# get names of all text-fields
b.text_fields.each do |i|
  puts i.name
end

# get name of first text-field
puts b.text_fields[0].name

# get name of second text-field
puts b.text_fields[1].name
```

Gherkin Cheat Sheet

Keywords Commands

```
@tag @tag
Feature: <Title of feature>
description
Background:
  Given
  And
@tag @tag
Scenario <Outline>:
  Given  Puts system in known state
  And
  But
  When  Key actions to be performed
  And
  But
  Then Observable business value output
  And
  But
Example:
|var1 |var2|
```

Tables

```
# select from list text
Given the following users exist:
    |name    |age  |
    |Sarah   | 42  |
    |Gentry  |28   |
    |Amrit   |34   |

    #A DataTable will be passed into the
underlying
  #step_def containing the rows and cols
of data



Scenario Outline: Addition
  Given there are currently <start>items
  When I add <num> more items
  Then I will see <result> total items
  Examples:
  |start  |num  |result|
  |0      |5    |5     |
  |7      |12   |19    |
```

General Tips

Feature:
Some terse yet descriptive text of what is des ired starts the feature and gives it a title.

Scenario:
Some determinable business situation starts the scenario, and contains a description of the scenario.

Scenario: A different situation starts the next scenario, and so on.

Background:
Backgrounds allows you to add some context to all scenarios in a single feature. A Back- ground is like an untitled scenario, contain- ing a number of steps. The difference is when it is run: the background is run before each of your scenarios, but after your BeforeSce- nario hooks

Cucumber Outputs
cucumber features --format html --out reports.html

Run by Tag
cucumber --tags @billing

Run without Tag
cucumber --tags ~@billing

Tags can be applied at both the feature and scenario lev- el. Tags follow logic with AND/OR keywords. Multiple tags can be run by comma sepa- ration.

```
Feature: Multiple site support

  Background:
    Given a global administrator named "Greg"
    And a blog named "Greg's anti-tax rants"
    And a customer named "Wilson"
    And a blog named "Expensive Therapy" owned by "Wilson"

  Scenario: Wilson posts to his own blog
    Given I am logged in as Wilson
    When I try to post to "Expensive Therapy"
    Then I should see "Your article was published."

  Scenario: Greg posts to a client's blog
    Given I am logged in as Greg
    When I try to post to "Expensive Therapy"
    Then I should see "Your article was published."
```