

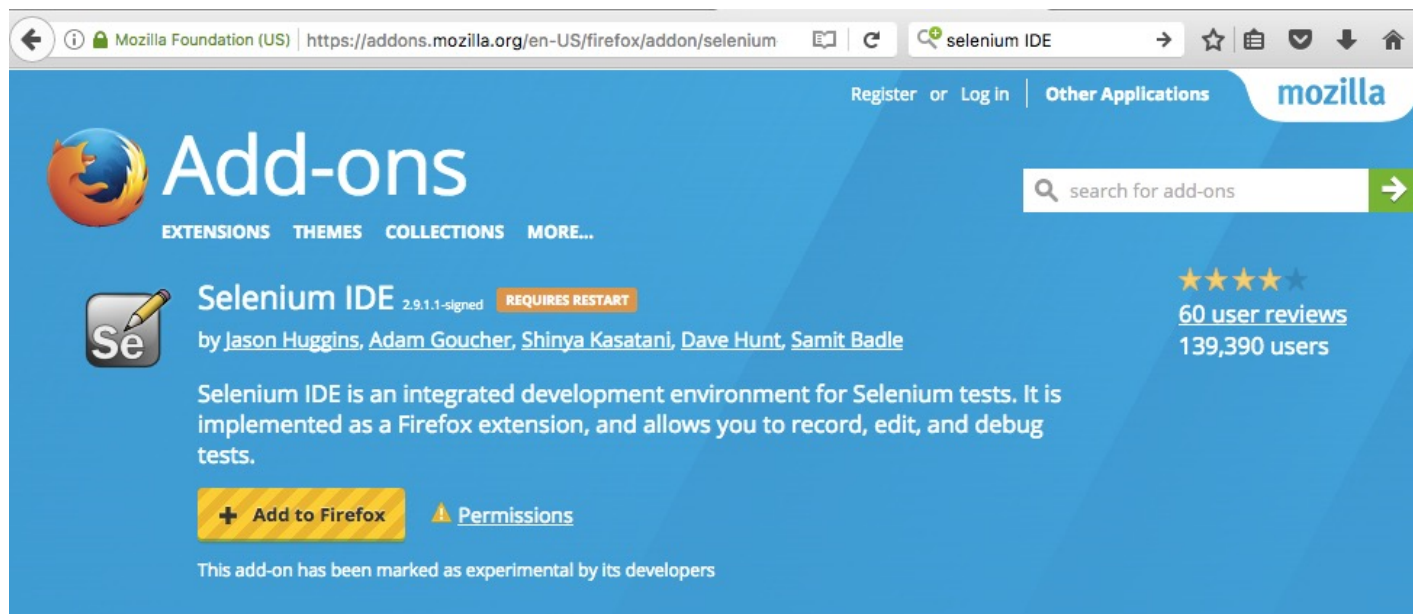


# **Selenium**

Selenium IDE

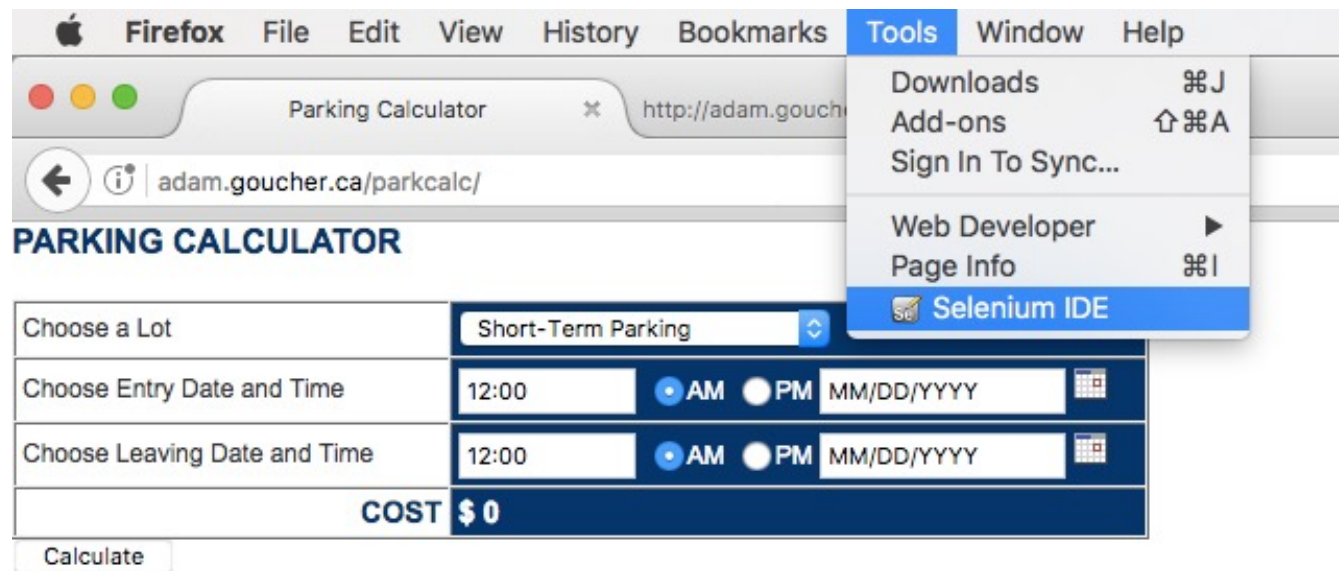
# Selenium IDE

- Firefox plugin
  - Currently compatible up to v.54
  - Work undergoing, see <https://github.com/SeleniumHQ/selenium-ide>

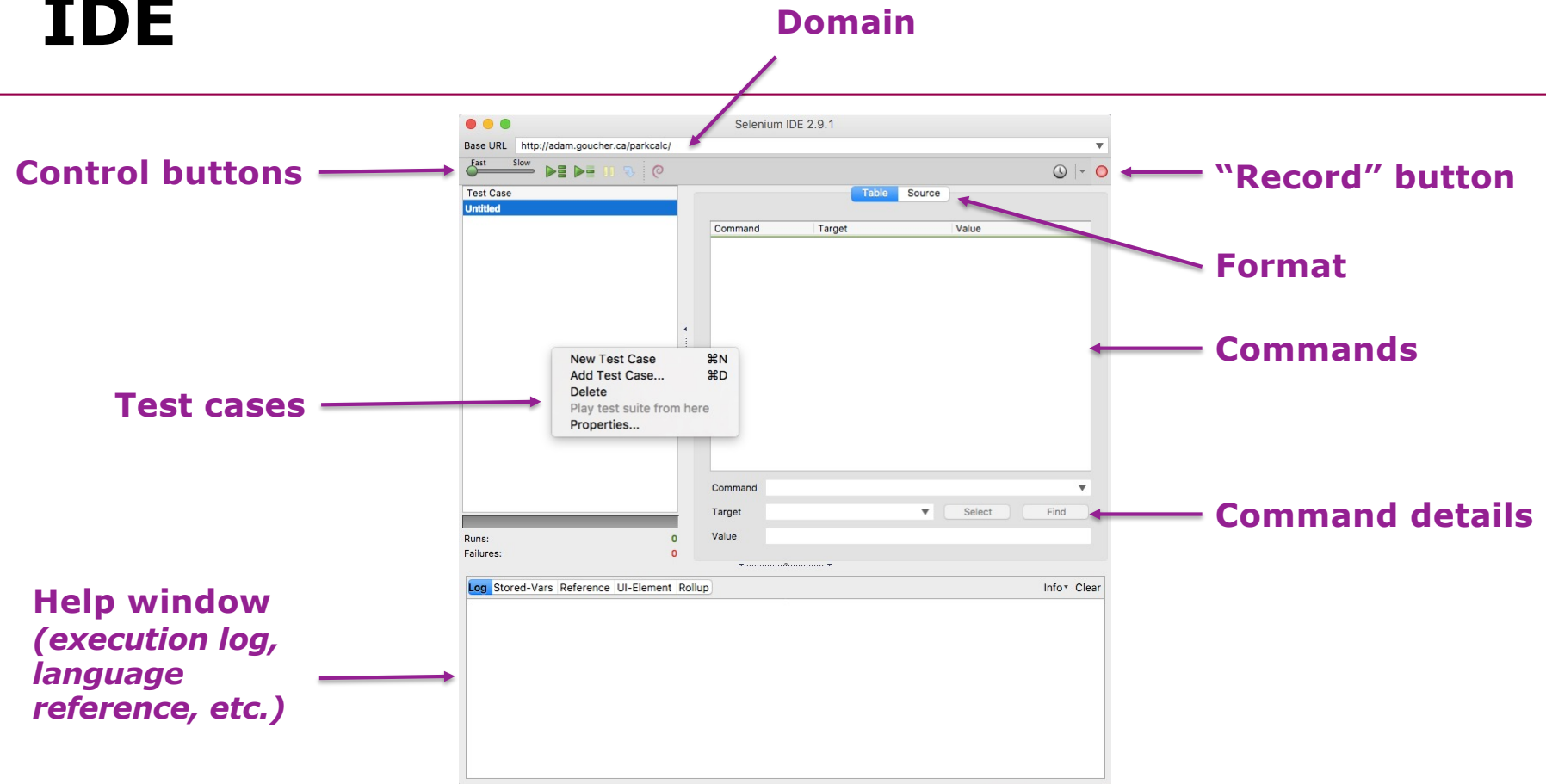


# Starting Selenium IDE

---



# IDE

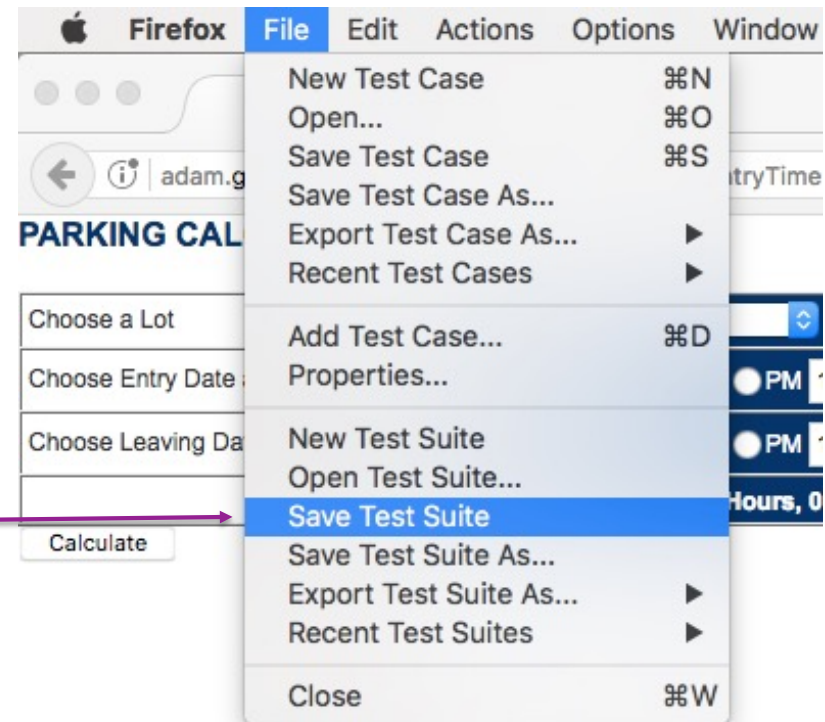


# Basic operations

## Test suites

---

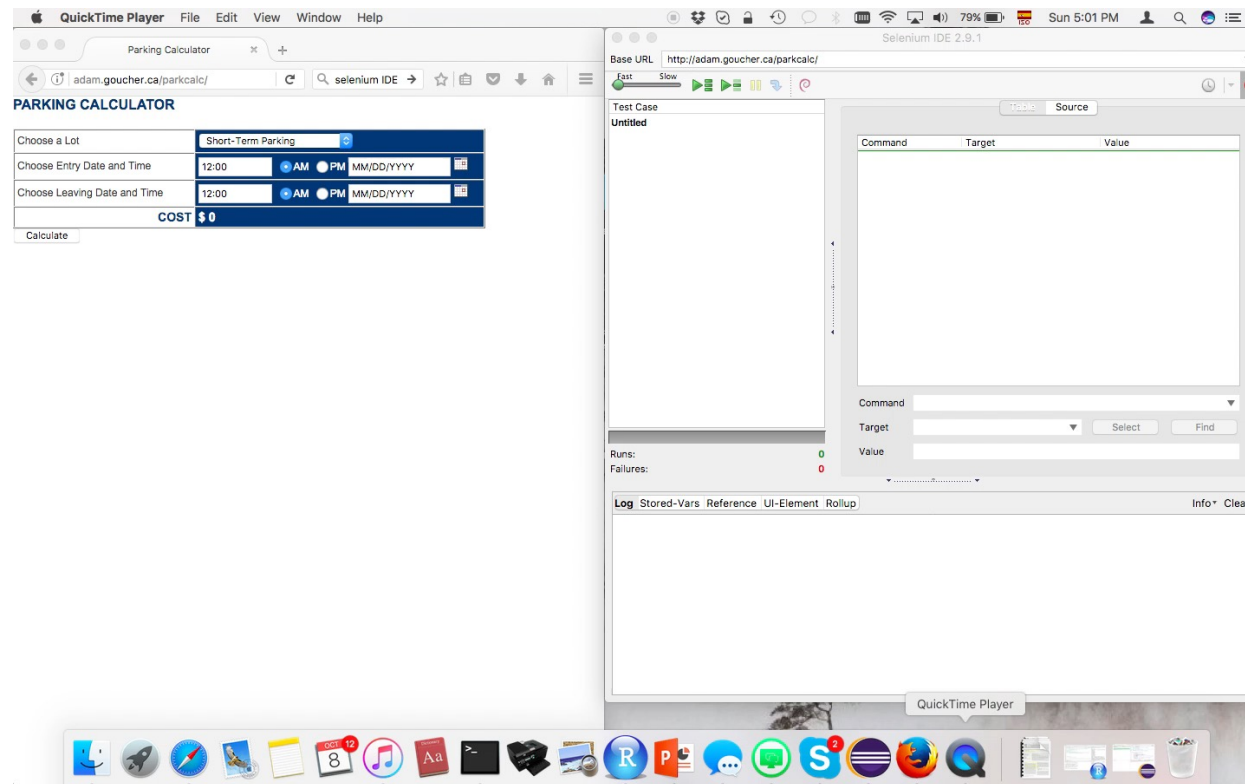
Not directly  
managed by the  
IDE



# Basic operations

## Record a test case

---



# Basic operations

## Record a test case > Specifying assertions

**Shortcut**  
*(of course, this  
value is not  
necessarily  
correct)*

The screenshot shows the Selenium IDE interface with a Firefox browser window displaying the 'PARKING CALCULATOR' form. The form has fields for 'Choose a Lot' (Short-Term Parking), 'Choose Entry Date and Time' (2:00 AM, 10/01/2017), and 'Choose Leaving Date and Time' (2:00 AM, 10/02/2017). The calculated cost is \$9.00 for 1 day. A context menu is open over the 'Calculate' button, showing various actions. The 'assertTable' command is highlighted, with its value set to 'table.3.1 \$ 9.00 (1 Days, 0 Hours, 0 Minutes)'. The right pane shows the 'Source' tab with a table of commands and their targets/values.

Command	Target	Value
open	/parkcalc/	
select	id=Lot	label=Economy Parking
type	id=EntryTime	9:00
click	document.Calculator.EntryT...	
type	id=EntryDate	10/01/2017
type	id=ExitTime	9:00
click	document.Calculator.ExitTI...	
type	id=ExitDate	10/02/2017
clickAndWait	name=Submit	

# Basic operations

## Run a test case

Green/Red bar

Execution log

The screenshot displays the Selenium IDE interface with a Firefox browser window showing a 'PARKING CALCULATOR' form. The form includes fields for 'Choose a Lot' (Short-Term Parking), 'Choose Entry Date and Time' (2:00 AM, 10/01/2017), and 'Choose Leaving Date and Time' (2:00 AM, 10/02/2017). The calculated cost is \$9.00 for 1 day, 0 hours, and 0 minutes. The Selenium IDE window shows a test case named 'untitled' with a table of commands:

Command	Target	Value
open	/parkcalc/	
select	id=Lot	label=Economy Parking
type	id=EntryTime	9:00
click	document.Calculator.EntryT...	
type	id=EntryDate	10/01/2017
type	id=ExitTime	9:00
click	document.Calculator.ExitT...	
type	id=ExitDate	10/02/2017
clickAndWait	name=Submit	

Below the table, the 'Command' dropdown is set to 'open', the 'Target' is '/parkcalc/', and the 'Value' field is empty. The 'Runs' section shows 0 runs and 0 failures. The 'Execution log' at the bottom shows the 'open(url)' command being executed, with a note: 'Opens an URL in the test frame. This accepts both relative and absolute URLs. The "open" command waits for the page to load before proceeding, i.e. the "AndWait" suffix is implicit. Note: The URL must be on the same domain as the runner HTML due to security restrictions in the browser (Same Origin Policy). If you need to open a URL on another domain, use the Selenium Server to start a new browser session on that domain.'

Commands  
run

One  
assertion I  
have just  
added



# Basic operations

Run a test case > Many types of assertions

These commands are editable

The screenshot displays the Selenium IDE interface. On the left, a web browser shows the 'PARKING CALCULATOR' page with form elements highlighted. On the right, the Selenium IDE interface shows a test case named 'untitled \*'. The 'Table' tab is selected, displaying a list of commands and their targets. The 'Log' tab at the bottom shows the execution log, indicating that the test case passed.

Command	Target	Value
open	/parkcalc/	
select	id=Lot	label=Economy Parking
type	id=EntryTime	9:00
click	document.Calculator.EntryT...	
type	id=EntryDate	10/01/2017
click	document.Calculator.ExitT...	
type	id=ExitDate	10/02/2017
clickAndWait	name=Submit	
selectWindow	null	
assertTable	css=table.3.1	\$ 9.00 (1 Days, 0 Hours...

Log: Stored-Vars Reference UI-Element Rollup Info+ Clear

```
[info] Executing: [type | id=EntryTime | 9:00 |  
[info] Executing: [click | document.Calculator.EntryTimeAMPM[1] | |  
[info] Executing: [type | id=EntryDate | 10/01/2017 | |  
[info] Executing: [type | id=ExitTime | 9:00 | |  
[info] Executing: [click | document.Calculator.ExitTimeAMPM[1] | |  
[info] Executing: [type | id=ExitDate | 10/02/2017 | |  
[info] Executing: [clickAndWait | name=Submit | | |  
[info] Executing: [selectWindow | null | | |  
[info] Executing: [assertTable | css=table.3.1 | $ 9.00 (1 Days, 0 Hours, 0 Minutes) | |  
[info] Test case passed  
[info] Test suite completed: 1 played, all passed!
```

The DOM elements are highlighted

The error is described here

UI elements can be automatically selected and highlighted in the web page

# Basic operations

## Run a test case > Some considerations

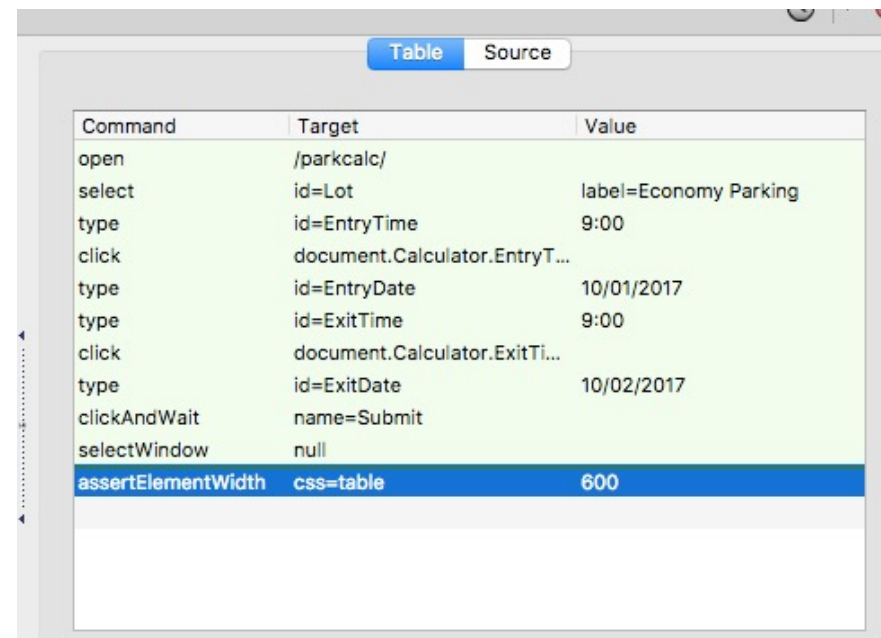
- Selenium returns what the browser contains
  - E.g., not 100%, but 545

```
<head>
  <title>Parking Calculator</title>
  <link href="style.css" rel="stylesheet"
  <script language="JavaScript" type="text
</head>
<body bgcolor="#FFFFFF" leftmargin="0" topma
  <p align="left" class="PageTitle">PARKIN
  <form name="Calculator" method="get" act
    <table width="545" border="1" cellpa
      <tr>
        <td width="201" class="BodyC
        <td width="326" bgcolor="#00
          <option value="STP" sele
          <option value="EP" >Ecor
          <option value="LTS" >Lor
          <option value="LTG" >Lor
          <option value="VP" >Vale
```

# Basic operations

## Test case design > commands

- Open a browser window
  - open
- Selecting options from a drop-down listbox
  - *select*
- Entering values
  - *Type*
- Clicking checkboxes or radio buttons
  - *click*
- Clicking a button or link
  - *click* or *clickAndWait*



The screenshot shows a window with two tabs: 'Table' and 'Source'. The 'Table' tab is active, displaying a table with three columns: 'Command', 'Target', and 'Value'. The table contains several rows of test commands, with the last row highlighted in blue.

Command	Target	Value
open	/parkcalc/	
select	id=Lot	label=Economy Parking
type	id=EntryTime	9:00
click	document.Calculator.EntryT...	
type	id=EntryDate	10/01/2017
type	id=ExitTime	9:00
click	document.Calculator.ExitTi...	
type	id=ExitDate	10/02/2017
clickAndWait	name=Submit	
selectWindow	null	
assertElementWidth	css=table	600

# Basic operations

## Test case design > locators

- There are different *locators*

- *id/name*
- *CSS*
- *link*
- *xpath*
- *dom*

```
<html>
<head>
<title>Parking Calculator</title>
<link href="style.css" rel="stylesheet" type="text/css">
<script language="JavaScript" type="text/JavaScript" src="datetimepicker.js">
</head>
<body bgcolor="#FFFFFF" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<p align="left" class="PageTitle">PARKING CALCULATOR<br>
<form name="Calculator" method="get" action="index.php?action=calculate">
  <table width="545" border="1" cellpadding="3" cellspacing="0" bordercolor="#000000">
    <tr>
      <td width="201" class="BodyCopy">Choose a Lot </td>
      <td width="326" bgcolor="#00306C" class="BodyCopy"><select name="Lot" id="Lot">
        <option value="STP" selected>Short-Term Parking</option>
        <option value="EP" >Economy Parking</option>
        <option value="LTS" >Long-Term Surface Parking</option>
        <option value="LTG" >Long-Term Garage Parking</option>
        <option value="VP" >Valet Parking</option>
      </select></td>
    </tr>
    <tr>
      <td width="201" class="BodyCopy">Choose Entry Date and Time
      <td width="326" bgcolor="#00306C" class="BodyCopy">
        <font color="#FFFFFF">
          <input name="EntryTime" type="text" id="EntryTime" value="9:00" />
          <input name="EntryTimeAMPM" type="radio" value="AM" />
          <input name="EntryTimeAMPM" type="radio" value="PM" />
          <input name="EntryDate" type="text" id="EntryDate" value="10/01/2010" />
          <a href="javascript:NewCal('EntryDate','mmddyyyy','f')>
        </font>
      </td>
    </tr>
  </table>
  <input type="submit" value="Submit" />
</form>
</body>
</html>
```

Command	Target	Value
open	/parkcalc/	
select	id=Lot	label=Lot
type	id=EntryTime	9:00
type	id=EntryDate	10/01/2010
type	id=ExitTime	9:00
type	id=ExitDate	10/02/2010
clickAndWait	name=Submit	
assertElementWidth	css=table	600

# Basic operations

## Test case design > locators

- There are different *locators*

- id*
- css*
- link*
- xpath*
- dom*

```
1
2
3 <html>
4   <head>
5     <title>Parking Calculator</title>
6     <link href="style.css" rel="stylesheet" type="text/css">
7     <script language="JavaScript" type="text/javascript">
8   </head>
9   <body bgcolor="#FFFFFF" leftmargin="0" topmargin="0">
10    <p align="left" class="PageTitle">
11      <form name="Calculator" method="post">
12        <table width="545" border="1">
13          <tr>
14            <td width="201" class="Text">
15              <td width="326" bgcolor="#FFFFFF">
16                <option value="0">
17                <option value="1">
```

Command	assertElementWidth
Target	xpath=/html/body/form/table
Value	600

Runs: 1  
Failures: 1

Log | Stored-Vars | Reference | UI-Element | Rollup

[info] Executing: |open | /parkcalc/ | |

[info] Executing: |select | id=Lot | label=Economy Parking |

[info] Executing: |type | id=EntryTime | 9:00 |

[info] Executing: |type | id=EntryDate | 10/01/2017 |

[info] Executing: |type | id=ExitTime | 9:00 |

[info] Executing: |type | id=ExitDate | 10/02/2017 |

[info] Executing: |clickAndWait | name=Submit | |

[info] Executing: |assertElementWidth | xpath=/html/body/form/table | 600 |

**[error] Actual value '545' did not match '600'**

[info] Test case failed

# Basic operations

## Test case design > locators

- There are different *locators*

- id*
- css*
- link*
- xpath*
- dom*

```
1
2
3 <html>
4   <head>
5     <title>Parking Calculator</title>
6     <link href="style.css" rel="stylesheet" type="text/css">
7     <script language="JavaScript" type="text/javascript">
8   </head>
9   <body bgcolor="#FFFFFF" leftmargin="0" topmargin="0">
10    <p align="left" class="PageTitle">
11      <form name="Calculator" method="post">
12        <table width="545" border="1">
13          <tr>
14            <td width="201" class="Text">
15              <td width="326" bgcolor="#FFFFFF">
16                <option value="0">0
17                <option value="1">1
```

document  
body  
children[1]  
children[0]

Command	Target	Value
assertElementWidth	dom=document.body.children[1].children[0]	600

uns: 1  
failures: 1

Log Stored-Vars Reference UI-Element Rollup

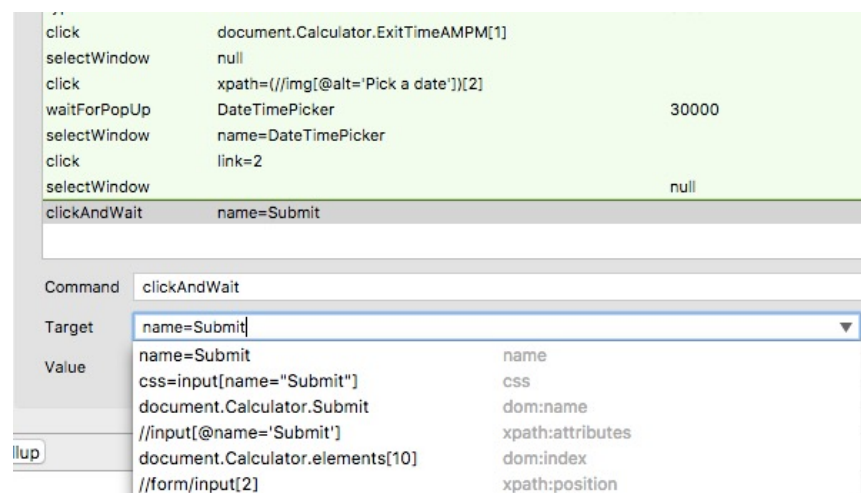
[info] Executing: |open | /parkcalc/ | |  
[info] Executing: |select | id=Lot | label=Economy Parking |  
[info] Executing: |type | id=EntryTime | 9:00 |  
[info] Executing: |type | id=EntryDate | 10/01/2017 |  
[info] Executing: |type | id=ExitTime | 9:00 |  
[info] Executing: |type | id=ExitDate | 10/02/2017 |  
[info] Executing: |clickAndWait | name=Submit | |  
[info] Executing: |assertElementWidth | dom=document.body.children[1].children[0] | 600 |  
[error] Actual value '545' did not match '600'  
[info] Test case failed



# Basic operations

## Test case design > locators

- Selenium IDE provides different ways to *locate* an element
  - *Important, as WebDriver do not implement all access locators, e.g., DOM*



# W3C tutorials

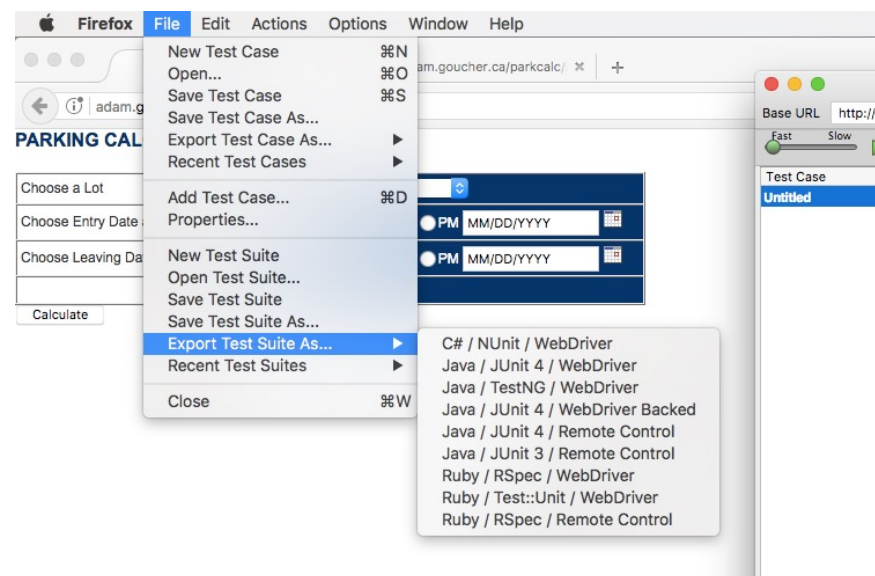
---

- Document object model (DOM)
  - [https://www.w3schools.com/xml/dom\\_intro.asp](https://www.w3schools.com/xml/dom_intro.asp)
- XML Path Language (Xpath)
  - [https://www.w3schools.com/xml/xpath\\_intro.asp](https://www.w3schools.com/xml/xpath_intro.asp)



# Open/save/export capabilities

- Test cases can be:
  - Opened/saved in native format (HMTL)
  - Exported to different WebDriver dialects
- Test suites can be defined to automatically execute sets of test cases



# Test case design

## Waiting times

*...andWait  
command tells  
Selenium  
to wait for the  
page to load  
after the  
action has  
been done*

The screenshot shows the Selenium IDE interface with a test case named 'untitled \*'. The test case list contains the following commands:

Command	Target	Value
open	/parkcalc/	
select	id=Lot	label=Economy Parking
type	id=EntryTime	9:00
click	document.Calculator.EntryT...	
type	id=EntryDate	10/01/2017
type	id=ExitTime	9:00
click	document.Calculator.ExitTI...	
type	id=ExitDate	10/02/2017
clickAndWait	name=Submit	

Below the test case list, there is a section for editing commands. The 'clickAndWait' command is selected, and its arguments are shown:

clickAndWait(locator)  
Generated from click(locator)  
Arguments:  
• locator - an element locator  
Clicks on a link, button, checkbox or radio button. If the click action causes a new page to load (like a link usually does), call waitForPageToLoad.

**New windows  
can be open**

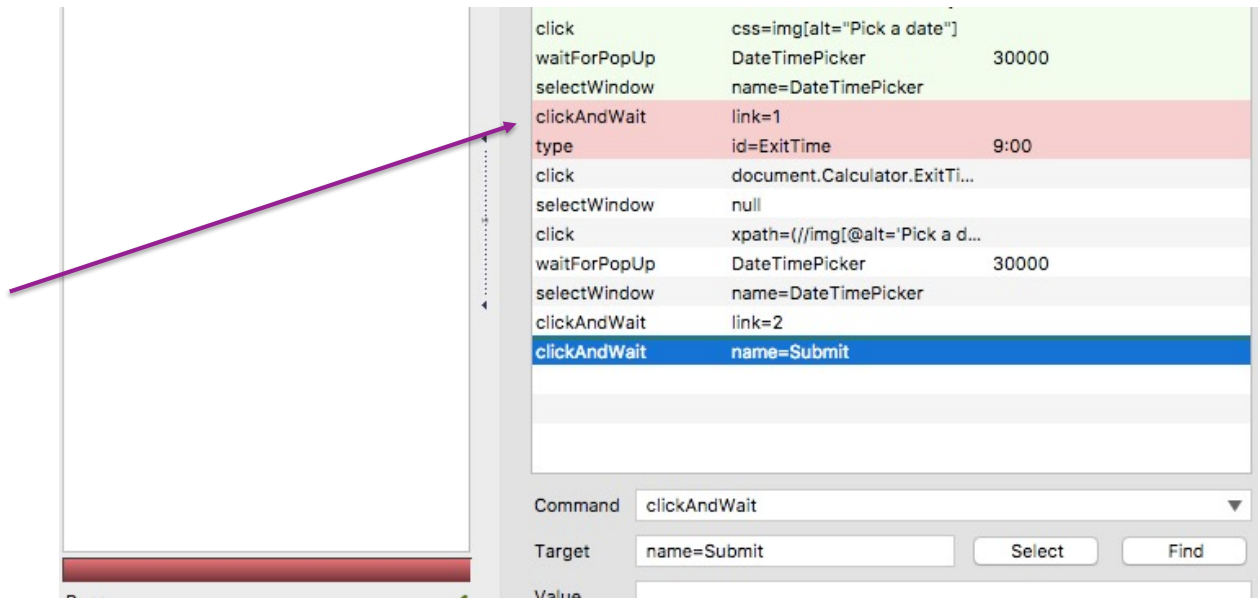
**Commands  
can be edited**

# Test case design

## Waiting times

---

*...andWait*  
fails because  
the page is  
not reloading



click	css=img[alt="Pick a date"]	
waitForPopUp	DateTimePicker	30000
selectWindow	name=DateTimePicker	
clickAndWait	link=1	
type	id=ExitTime	9:00
click	document.Calculator.ExitTi...	
selectWindow	null	
click	xpath=(((img[@alt='Pick a d...	
waitForPopUp	DateTimePicker	30000
selectWindow	name=DateTimePicker	
clickAndWait	link=2	
clickAndWait	name=Submit	

Command: clickAndWait

Target: name=Submit

Value:

# Test case design

## Windows

We are not longer working in the original (*null*) browser window

Remove the *...andWait* avoids a problem but manifest another

click	document.Calculator.EntryT...
click	css=img[alt="Pick a date"]
waitForPopUp	DateTimePicker 30000
selectWindow	name=DateTimePicker
click	link=1
type	id=ExitTime 9:00
click	document.Calculator.ExitTi...
selectWindow	null
click	xpath=(//img[@alt="Pick a d...
waitForPopUp	DateTimePicker 30000
selectWindow	name=DateTimePicker
click	link=2
clickAndWait	name=Submit

Command: click

Target: link=2

Value:

# Test case design

## Windows

We need to select the appropriate window

The screenshot shows a test case design tool interface. On the left, a 'Test Case' pane shows an 'untitled \*' test case. On the right, a 'Table' pane displays a list of commands and their targets. Below the table, there are input fields for 'Command', 'Target', and 'Value', along with 'Select' and 'Find' buttons. At the bottom left, a status bar shows 'Runs: 1' and 'Failures: 0'.

Command	Target	Value
open	/parkcalc/	
select	id=Lot	label=Economy Parking
type	id=EntryTime	9:00
click	document.Calculator.EntryT...	
click	css=img[alt="Pick a date"]	
waitForPopUp	DateTimePicker	30000
selectWindow	name=DateTimePicker	
click	link=1	
selectWindow		null
type	id=ExitTime	9:00
click	document.Calculator.ExitTI...	
selectWindow	null	
click	xpath=//img[@alt='Pick a d...	
waitForPopUp	DateTimePicker	30000
selectWindow	name=DateTimePicker	
click	link=2	
selectWindow		null
clickAndWait	name=Submit	

Command:   
Target:     
Value:

Runs: 1  
Failures: 0

# Test case design

## Alerts

- Checks the text
- Closes the alert automatically

The screenshot shows a test case design tool interface. On the left, a 'Test Case' pane shows an 'Untitled \*' case. The main area displays a table of commands with columns 'Command', 'Target', and 'Value'. The 'assertText' command is highlighted in blue. Below the table, a detailed view of the 'assertText' command is shown, including a 'Command' dropdown, a 'Target' input field with 'document.body.children[0]', and a 'Value' input field with 'Click\*'. A 'Runs:' counter at the bottom left shows '1'.

Command	Target	Value
open	/jsref/tryit.asp?filename=tryj...	
selectFrame	iframeResult	
click	css=button	
assertAlert	Hello! I am an alert box!	
selectWindow		null
selectFrame	iframeResult	
assertText	document.body.children[0]	Click*

Command: assertText  
Target: document.body.children[0]  
Value: Click\*

Frame selection

Wildcards (\*) are allowed

# Test case design

## Alerts

Missing the *assertAlert* causes the test case to fail

The screenshot shows the Selenium IDE interface. On the left, the 'Test Case' pane shows an 'Untitled \*' test case. Below it, the 'Runs' pane shows 1 run and 1 failure. The main pane displays a table of commands:

Command	Target	Value
open	/jsref/tryit.asp?filename=tryj...	
selectFrame	iframeResult	
click	css=button	
selectWindow		null
selectFrame	iframeResult	
assertText	document.body.children[0]	Click*

Below the table, the 'Command' dropdown is set to 'selectWindow', and the 'Value' field contains 'null'. The 'Log' pane at the bottom shows the following messages:

```
[info] Executing: |open | /jsref/tryit.asp?filename=tryjsref_alert | |
[info] Executing: |selectFrame | iframeResult | |
[info] Executing: |click | css=button | |
[info] Executing: |selectWindow | | null |
[error] There was an unexpected Alert! [Hello! I am an alert box!]
[info] Test case failed
[info] Test suite completed: 1 played, 1 failed
```


A purple arrow points from the text 'Missing the *assertAlert* causes the test case to fail' to the 'Log' pane, specifically to the error message.

# Test case design

## Confirmations

---

**Automatic  
response to the  
confirm box**



Command	Target	Value
open	/jsref/tryit.asp?filename=tryj...	
selectFrame	iframeResult	
chooseOkOnNextConfirmation		
click	css=button	
assertConfirmation	Press a button!	




# Test case design

## Prompts

---

**Automatic  
response to the  
confirm box**



Command	Target	Value
open	/jsref/tryit.asp?filename=tryj...	
selectFrame	iframeResult	
answerOnNextPrompt	Oscar	
click	css=button	
assertPrompt	Please enter your name	

# Test case design

## Variables

---

- Store a value into a variable:

Command	Target	Value
store	<a href="mailto:paul@mysite.org">paul@mysite.org</a>	userName

- Check the value against something:

Command	Target	Value
assertText	//div/p	\${userName}

- Use the value:

Command	Target	Value
type	id=login	\${userName}

# Test case design

## Variables

---

- Store some element into a variable:

Command	Target	Value
storeText	id=username	userName

- Evaluate an expression using *javascript*

Command	Target	Value
storeEval	storedVars['userName'].toLowerCase()	name

# Test case design

## Variables > example

I cannot see how to use variables to switch on/off radio buttons unless we activate the flow control extension

The screenshot displays the Selenium IDE interface with a test case named 'useVariables'. The test case is loaded with the following commands:

Command	Target	Value
open	/parkcalc/	
store	10:00	initTime
store	11:00	endTime
store	10/01/2017	initDate
store	10/03/2017	endDate
click	document.Calculator.EntryTimeAMPM(1)	
click	document.Calculator.ExitTimeAMPM(0)	
type	id=EntryTime	\$(initTime)
type	id=EntryDate	\$(initDate)
type	id=ExitTime	\$(endTime)
type	id=ExitDate	\$(endDate)
clickAndWait	name=Submit	
assertValue	id=EntryTime	10:00

The 'Stored Vars' panel at the bottom shows the following variables:

- 1 ☐ initTime = "10:00" [edit](#)
- 2 ☐ endTime = "11:00" [edit](#)
- 3 ☐ initDate = "10/01/2017" [edit](#)
- 4 ☐ endDate = "10/03/2017" [edit](#)

The page refresh may change the injected values

The "Stored Variables" plug in enables the examination of values

# Test case design

## Variables > store commands

- An equivalent store command exists for each **assert command**
  - The value of the corresponding element is stored in a variable, e.g.:

The screenshot displays the Selenium IDE interface. On the left, the 'Test Case' pane shows a single command: 'useVariables \*'. The main workspace contains a table of commands:

Command	Target	Value
open	/jsref/tryit.asp?filename=tryj...	
selectFrame	iframeResult	
click	css=button	
storeAlert	alertText	

Below the table, the 'Command' dropdown is set to 'storeAlert', the 'Target' is 'alertText', and the 'Value' field is empty. At the bottom, the 'Stored Vars' tab is active, showing a log entry: '1 alertText = "Hello! I am an alert box!" edit'. A yellow warning box above the log states: 'Selenium IDE keeps the old stored variables and they will also be shown. Click the Refresh button to refresh.'

# Other issues

## Test case design > more about waiting times

---

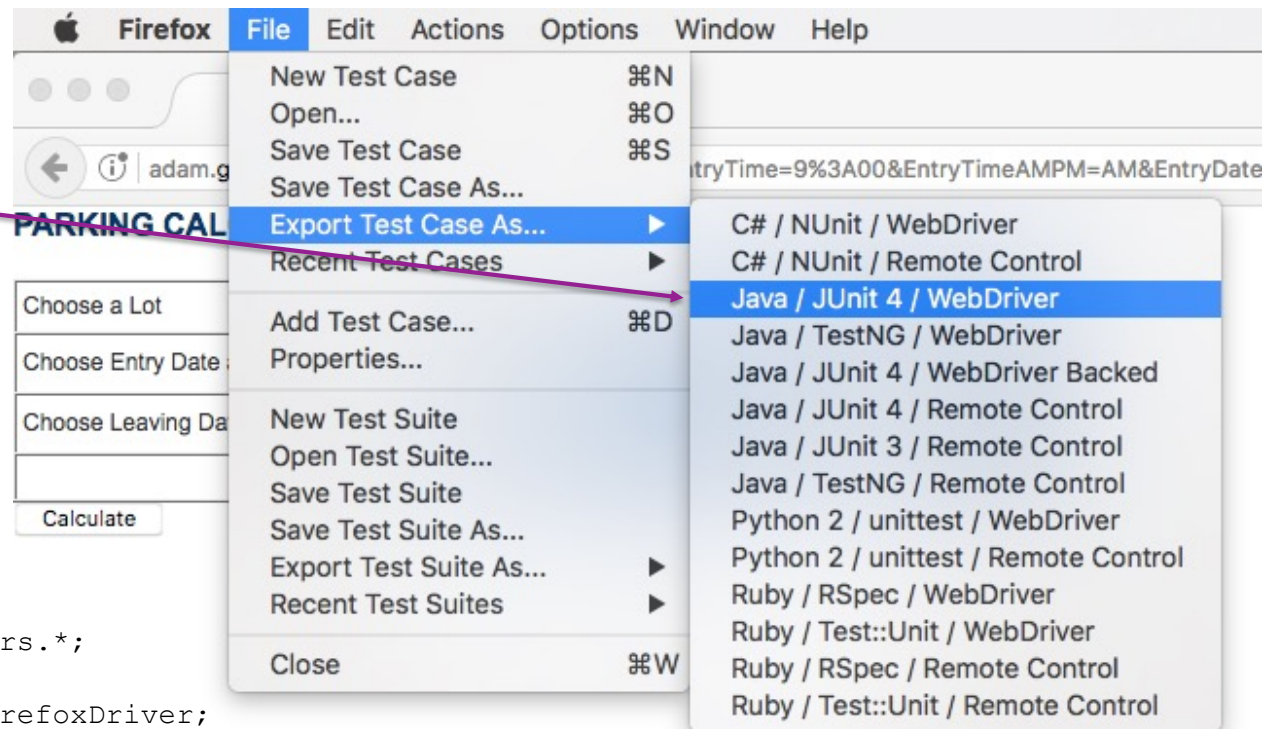
- In AJAX driven web applications, data is retrieved from server without refreshing the page. Using **...andWait** commands will not work as the page is not actually refreshed
- The best approach would be to **wait for the needed element** in a dynamic period and then continue the execution as soon as the element is found
- This is done using **waitFor** commands, as *waitForElementPresent* or *waitForVisible*, which wait dynamically, checking for the desired condition every second and continuing to the next command in the script as soon as the condition is met

# Other issues

## Export to *webdriver*

Generates the  
corresponding  
code

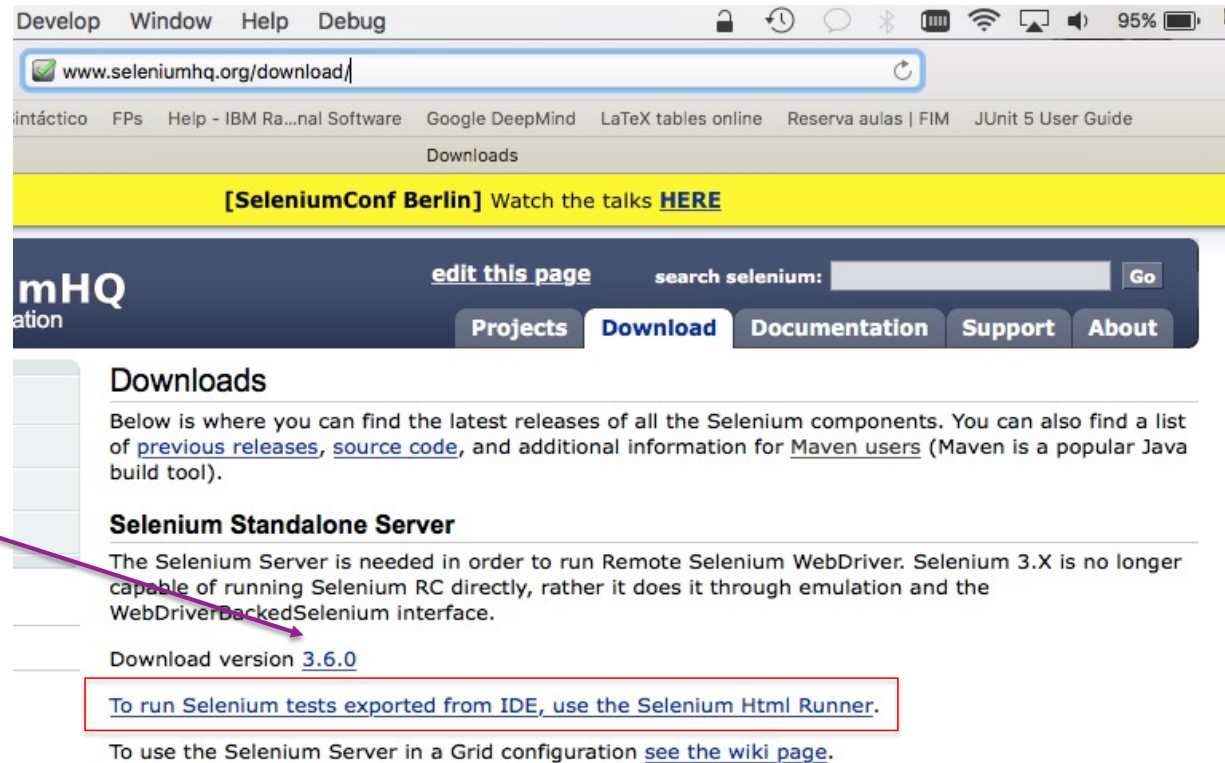
```
package com.example.tests;
import java.util.regex.Pattern;
import java.util.concurrent.TimeUnit;
import org.junit.*;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.*;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;
...
```



# Other issues

## Export to *webdriver*

The standalone server is necessary to run tests locally. Another option is to set up Maven bindings



Develop Window Help Debug

www.seleniumhq.org/download/

intático FPs Help - IBM Rational Software Google DeepMind LaTeX tables online Reserva aulas | FIM JUnit 5 User Guide

Downloads

[SeleniumConf Berlin] Watch the talks [HERE](#)

mHQ  
ation

[edit this page](#) search selenium:  [Go](#)

[Projects](#) [Download](#) [Documentation](#) [Support](#) [About](#)

### Downloads

Below is where you can find the latest releases of all the Selenium components. You can also find a list of [previous releases](#), [source code](#), and additional information for [Maven users](#) (Maven is a popular Java build tool).

### Selenium Standalone Server

The Selenium Server is needed in order to run Remote Selenium WebDriver. Selenium 3.X is no longer capable of running Selenium RC directly, rather it does it through emulation and the [WebDriverBackedSelenium](#) interface.

Download version [3.6.0](#)

[To run Selenium tests exported from IDE, use the Selenium Html Runner.](#)

To use the Selenium Server in a Grid configuration [see the wiki page](#).



# Other issues

## Export to *webdriver*

Package is defined by Selenium IDE

FirefoxDriver() is used

Implementations are not completely aligned


Default timeout

```
1 package com.example.tests;
2
3 import java.util.regex.Pattern;
4
11
12 public class ParkingTest {
13     private WebDriver driver;
14     private String baseUrl;
15     private boolean acceptNextAlert = true;
16     private StringBuffer verificationErrors = new StringBuffer();
17
18     @Before
19     public void setUp() throws Exception {
20         driver = new FirefoxDriver();
21         baseUrl = "http://adam.goucher.ca/";
22         driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
23     }
24
25     @Test
26     public void testParking() throws Exception {
27         driver.get(baseUrl + "/parkcalc/");
28         new Select(driver.findElement(By.id("Lot"))).selectByVisibleText("Economy Parking");
29         driver.findElement(By.id("EntryTime")).clear();
30         driver.findElement(By.id("EntryTime")).sendKeys("9:00");
31         driver.findElement(By.name("EntryTimeAMPM")).click();
32         driver.findElement(By.id("EntryDate")).clear();
33         driver.findElement(By.id("EntryDate")).sendKeys("10/01/2017");
34         driver.findElement(By.id("ExitTime")).clear();
35         driver.findElement(By.id("ExitTime")).sendKeys("11:00");
36         // ERROR: Caught exception [Error: Dom locators are not implemented yet!]
37         driver.findElement(By.id("ExitDate")).clear();
38         driver.findElement(By.id("ExitDate")).sendKeys("10/01/2017");
39         driver.findElement(By.name("Submit")).click();
40         assertEquals("$ 9.00", driver.findElement(By.cssSelector("b")).getText());
41     }
42
43     @After
44     public void tearDown() throws Exception {
45         driver.quit();
46     }
47 }
```

# Other issues

## Export to *webdriver*

Also includes  
some utility  
code (can or  
cannot be  
used)

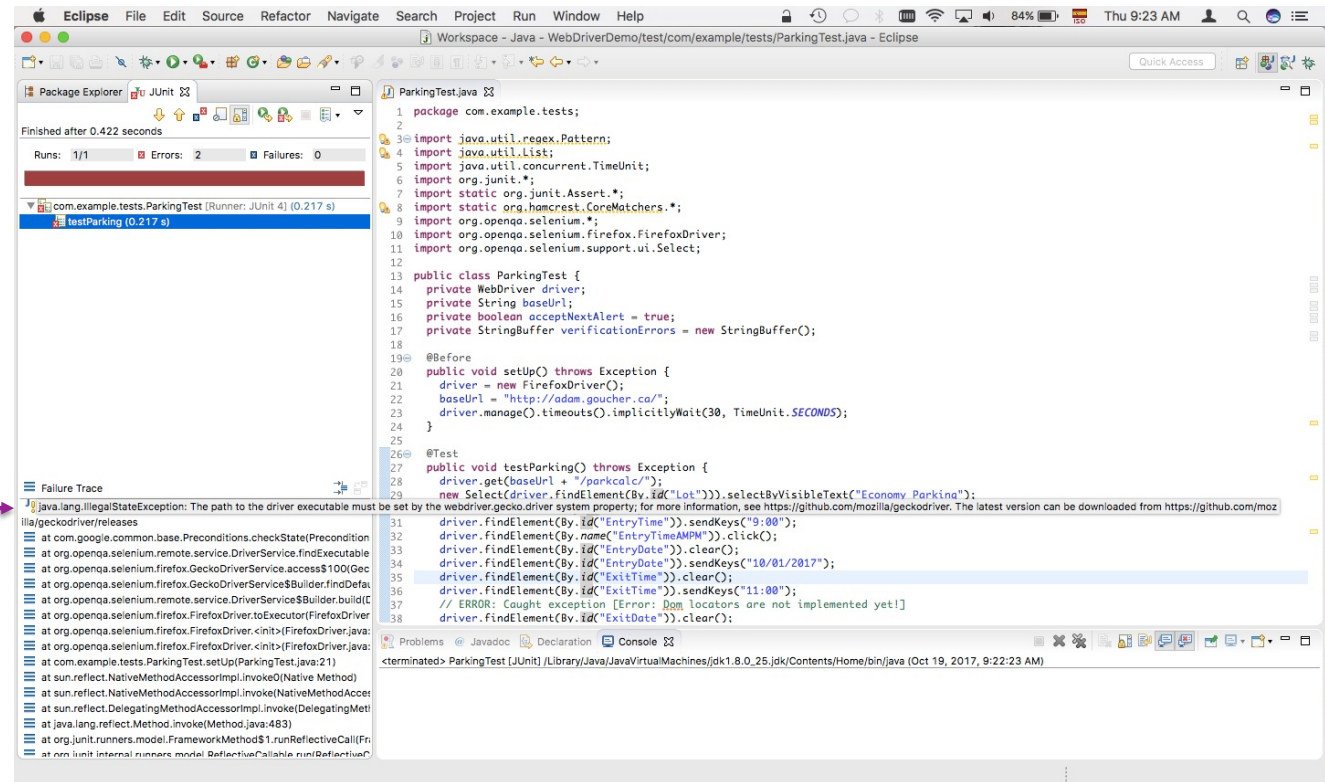


```
41 }
42
43 @After
44 public void tearDown() throws Exception {
45     driver.quit();
46     String verificationErrorString = verificationErrors.toString();
47     if (!"".equals(verificationErrorString)) {
48         fail(verificationErrorString);
49     }
50 }
51
52 private boolean isElementPresent(By by) {
53     try {
54         driver.findElement(by);
55         return true;
56     } catch (NoSuchElementException e) {
57         return false;
58     }
59 }
60
61 private boolean isAlertPresent() {
62     try {
63         driver.switchTo().alert();
64         return true;
65     } catch (NoAlertPresentException e) {
66         return false;
67     }
68 }
69
70 private String closeAlertAndGetItsText() {
71     try {
72         Alert alert = driver.switchTo().alert();
73         String alertText = alert.getText();
74         if (acceptNextAlert) {
75             alert.accept();
76         } else {
77             alert.dismiss();
78         }
79     }
```

# Other issues

## Run tests in *webdriver*

Each browser  
may need  
specific support,  
with the  
exception of  
HtmlUnitDriver();



```
1 package com.example.tests;
2
3 import java.util.regex.Pattern;
4 import java.util.List;
5 import java.util.concurrent.TimeUnit;
6 import org.junit.*;
7 import static org.junit.Assert.*;
8 import static org.hamcrest.CoreMatchers.*;
9 import org.openqa.selenium.*;
10 import org.openqa.selenium.firefox.FirefoxDriver;
11 import org.openqa.selenium.support.ui.Select;
12
13 public class ParkingTest {
14     private WebDriver driver;
15     private String baseUrl;
16     private boolean acceptNextAlert = true;
17     private StringBuffer verificationErrors = new StringBuffer();
18
19     @Before
20     public void setUp() throws Exception {
21         driver = new FirefoxDriver();
22         baseUrl = "http://adam.goucher.ca/";
23         driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
24     }
25
26     @Test
27     public void testParking() throws Exception {
28         driver.get(baseUrl + "/parkcalc/");
29         new Select(driver.findElement(By.id("Lot"))).selectByVisibleText("Economy Parking");
30         driver.findElement(By.id("EntryTime")).sendKeys("9:00");
31         driver.findElement(By.name("EntryTimeAMPN")).click();
32         driver.findElement(By.id("EntryDate")).clear();
33         driver.findElement(By.id("EntryDate")).sendKeys("10/01/2017");
34         driver.findElement(By.id("ExitTime")).clear();
35         driver.findElement(By.id("ExitTime")).sendKeys("11:00");
36         // ERROR: Caught exception [Error: Dom locators are not implemented yet!]
37         driver.findElement(By.id("ExitDate")).clear();
38     }
39 }
```

Failure Trace

```
java.lang.IllegalStateException: The path to the driver executable must be set by the webdriver.gecko.driver system property; for more information, see https://github.com/mozilla/geckodriver. The latest version can be downloaded from https://github.com/mozilla/geckodriver/releases
at com.google.common.base.Preconditions.checkNotNull(Precondition
at org.openqa.selenium.remote.service.DriverService.findExecutable
at org.openqa.selenium.firefox.GeckoDriverService.access$100(Gec
at org.openqa.selenium.firefox.GeckoDriverService$Builder.findDefa
at org.openqa.selenium.remote.service.DriverService$Builder.bu
at org.openqa.selenium.firefox.FirefoxDriver.toExecutor(FirefoxDri
at org.openqa.selenium.firefox.FirefoxDriver.<init>(FirefoxDriver.java
at com.example.tests.ParkingTest.setUp(ParkingTest.java:21)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAcces
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMet
at java.lang.reflect.Method.invoke(Method.java:483)
at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(Fri
at org.junit.internal.runners.model.ReflectiveCallable.run(Reflective
```

Console

```
<terminated> ParkingTest [JUnit] /Library/Java/JavaVirtualMachines/jdk1.8.0_25-jdk/Contents/Home/bin/java (Oct 19, 2017, 9:22:23 AM)
```