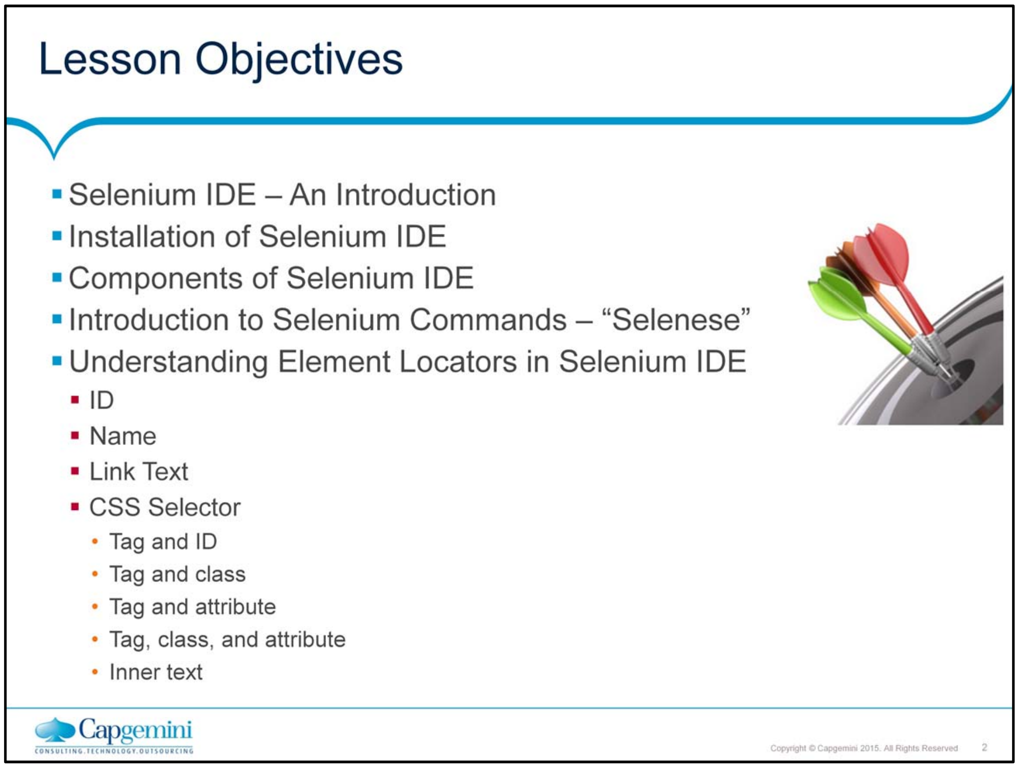
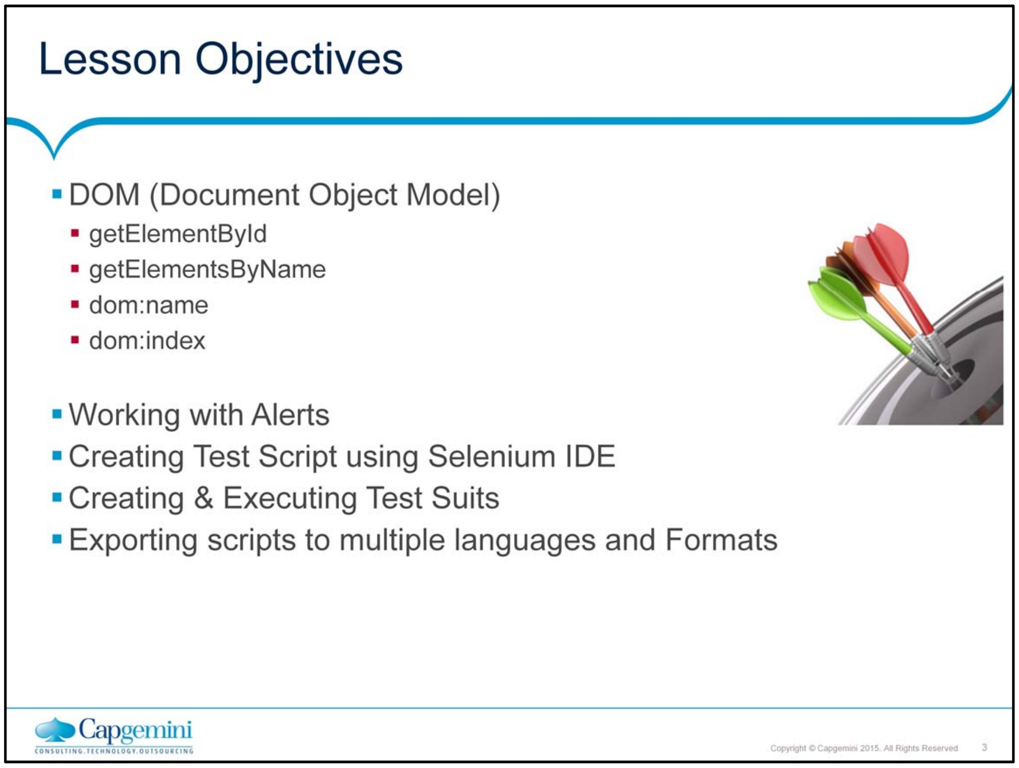
Test Automation & Advanced Selenium

Lesson 3: Working With Selenium IDE



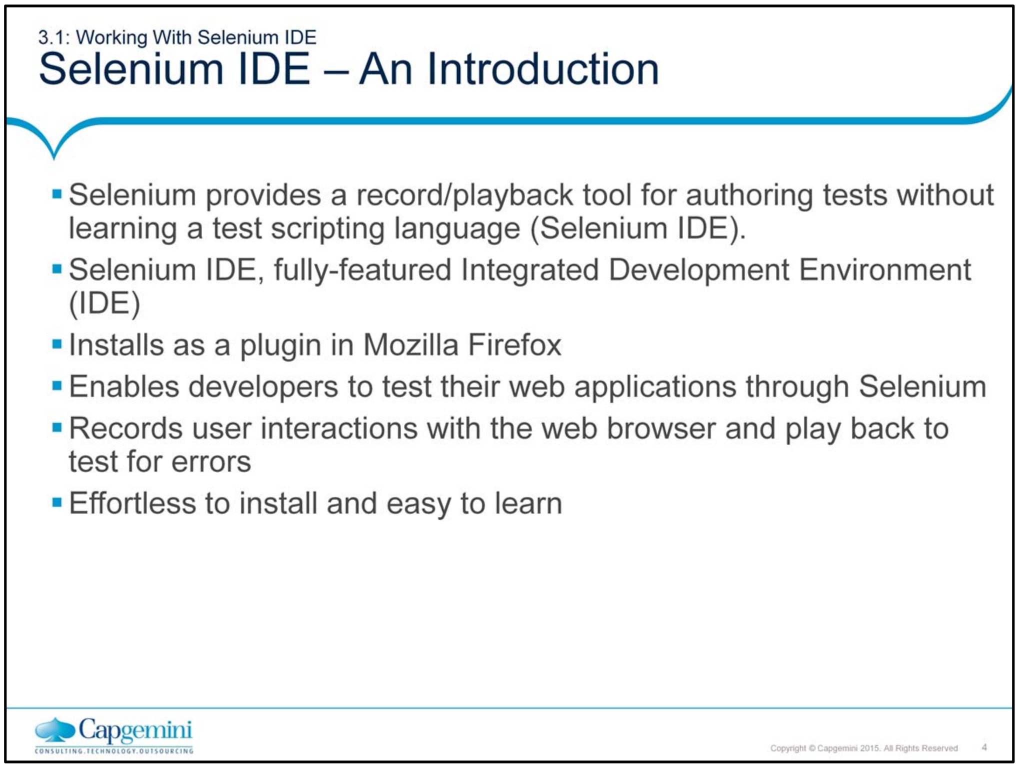
Lesson Objectives

- Selenium IDE – An Introduction
- Installation of Selenium IDE
- Components of Selenium IDE
- Introduction to Selenium Commands – "Selenese"
- Understanding Element Locators in Selenium IDE
  - ID
  - Name
  - Link Text
  - CSS Selector
    - Tag and ID
    - Tag and class
    - Tag and attribute
    - Tag, class, and attribute
    - Inner text



Lesson Objectives

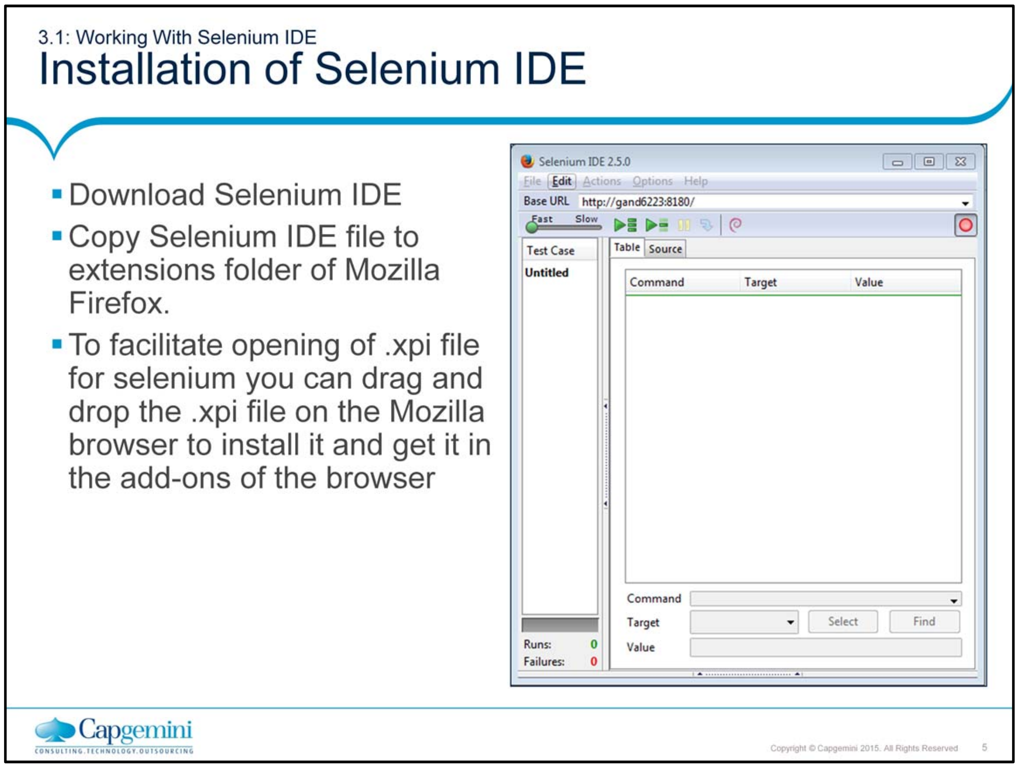
- DOM (Document Object Model)
  - getElementById
  - getElementsByName
  - dom:name
  - dom:index
- Working with Alerts
- Creating Test Script using Selenium IDE
- Creating & Executing Test Suits
- Exporting scripts to multiple languages and Formats



3.1: Working With Selenium IDE

Selenium IDE – An Introduction

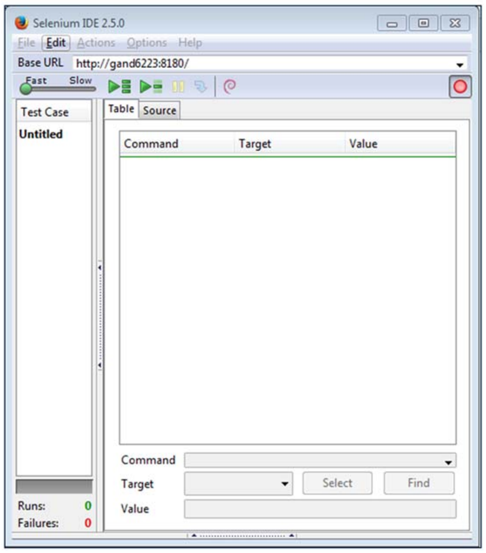
- Selenium provides a record/playback tool for authoring tests without learning a test scripting language (Selenium IDE).
- Selenium IDE, fully-featured Integrated Development Environment (IDE)
- Installs as a plugin in Mozilla Firefox
- Enables developers to test their web applications through Selenium
- Records user interactions with the web browser and play back to test for errors
- Effortless to install and easy to learn



3.1: Working With Selenium IDE

Installation of Selenium IDE

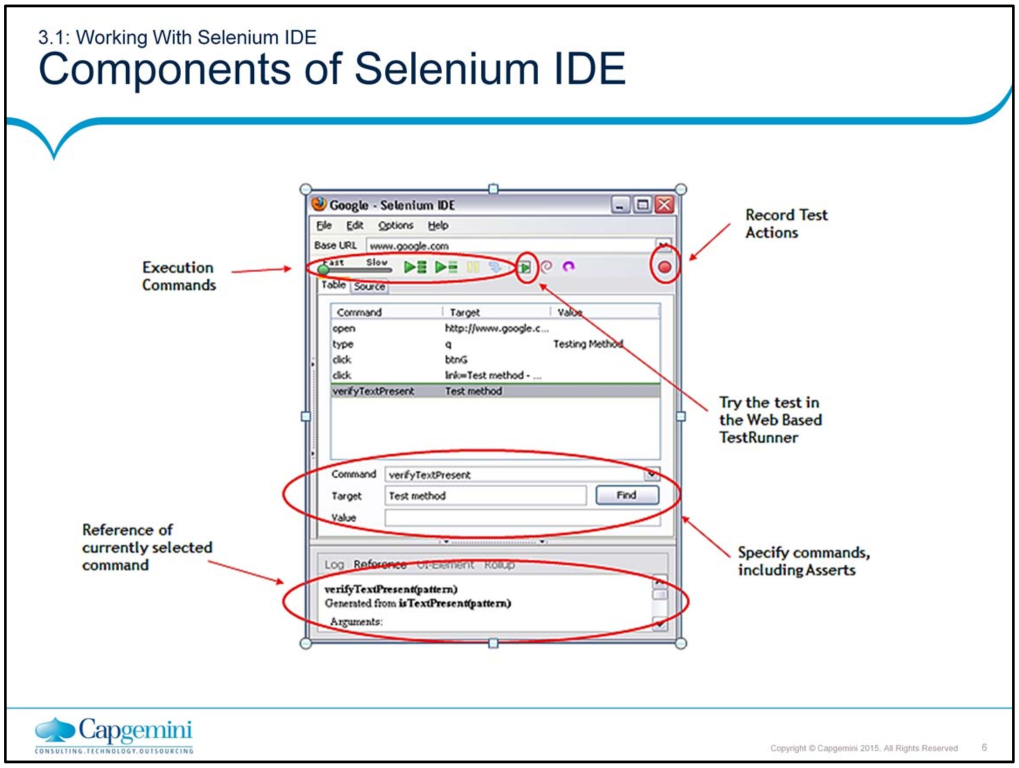
- Download Selenium IDE
- Copy Selenium IDE file to extensions folder of Mozilla Firefox.
- To facilitate opening of .xpi file for selenium you can drag and drop the .xpi file on the Mozilla browser to install it and get it in the add-ons of the browser



Base URL http://gand6223:8180/

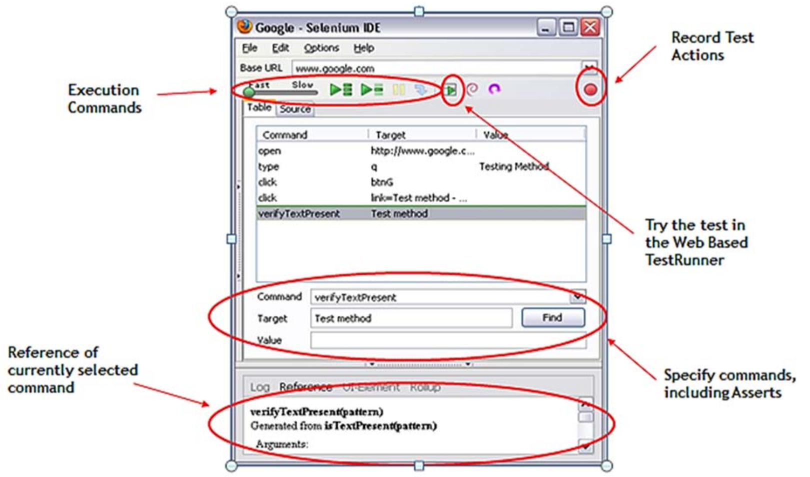
Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

5



3.1: Working With Selenium IDE

Components of Selenium IDE


Google - Selenium IDE
File Edit Options Help
Fast Slow
Table Source
Find
Log Reference UI-Element Rollup
Record Test Actions
Execution Commands
Try the test in the Web Based TestRunner
Reference of currently selected command
Specify commands, including Asserts

Base URL www.google.com

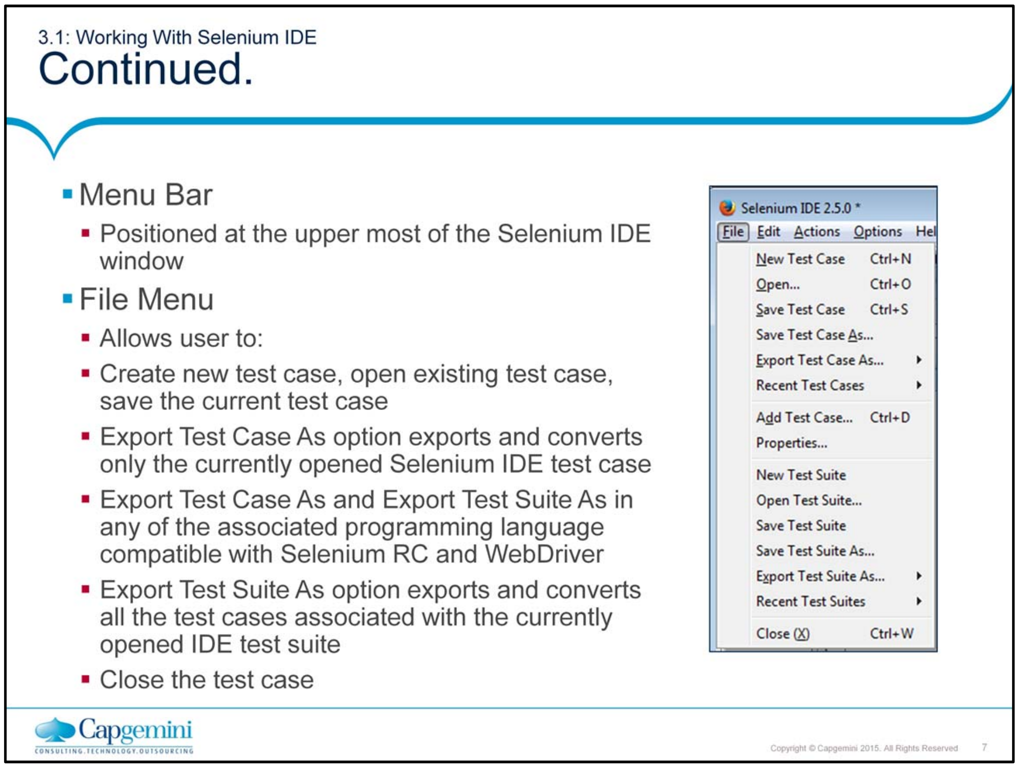
Command	Target	Value
open	http://www.google.c...	
type	q	Testing Method
click	btnG	
click	link=Test method - ...	
verifyTextPresent	Test method	

Command: verifyTextPresent
Target: Test method
Value:

verifyTextPresent(pattern)
Generated from **isTextPresent(pattern)**
Arguments:

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

6



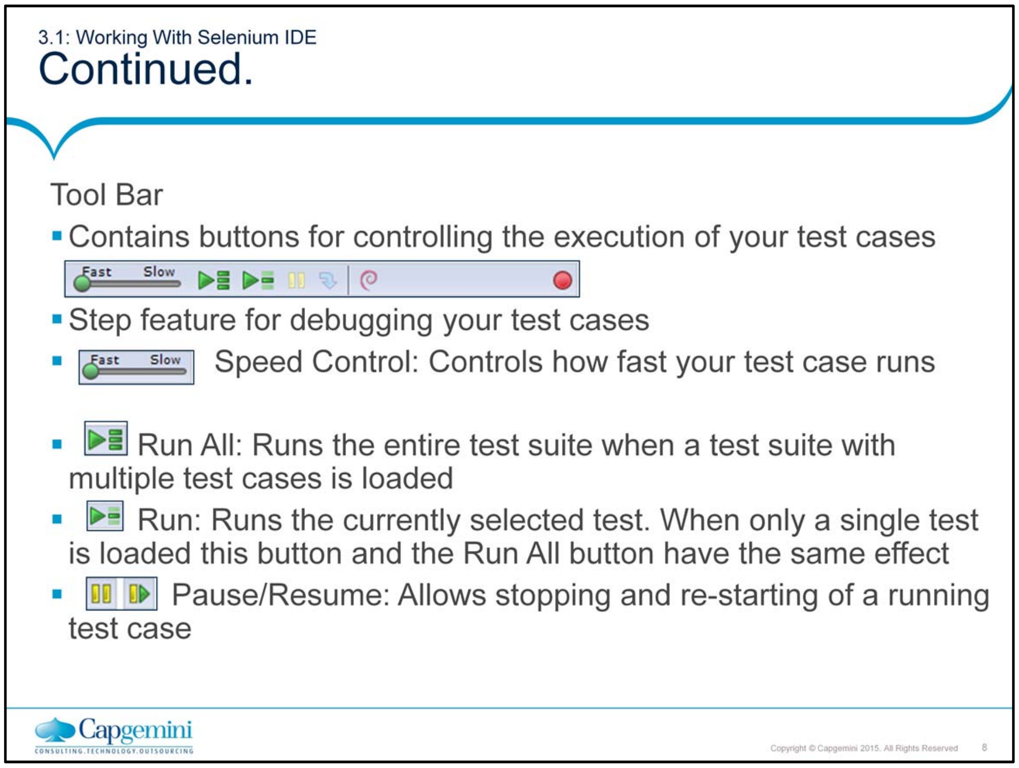
3.1: Working With Selenium IDE

Continued.

- Menu Bar
  - Positioned at the upper most of the Selenium IDE window
- File Menu
  - Allows user to:
  - Create new test case, open existing test case, save the current test case
  - Export Test Case As option exports and converts only the currently opened Selenium IDE test case
  - Export Test Case As and Export Test Suite As in any of the associated programming language compatible with Selenium RC and WebDriver
  - Export Test Suite As option exports and converts all the test cases associated with the currently opened IDE test suite
  - Close the test case

1.File Menu

It is very much analogous to the file menu belonging to any other application. Export Test Case As and Export Test Suite give the liberty to the user to prefer amid the available unit testing frameworks like jUnit, TestNG etc.
Thus an IDE test case can be exported for a chosen union of programming language, unit testing framework and tool from the selenium package.



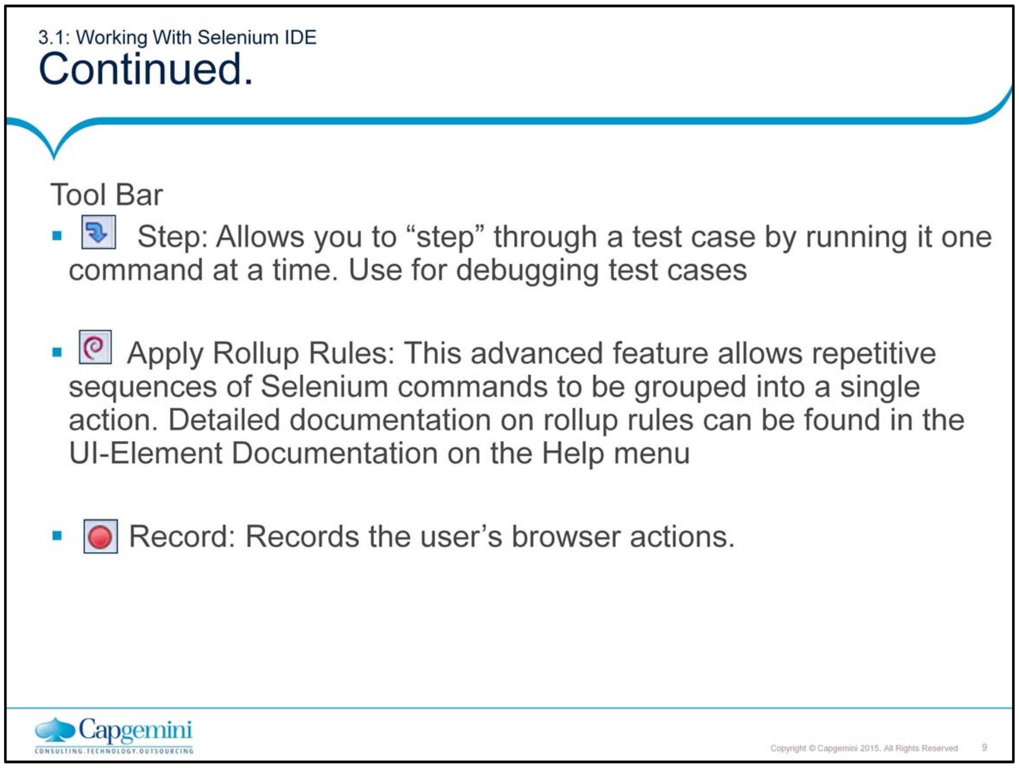
3.1: Working With Selenium IDE

Continued.

Tool Bar

- Contains buttons for controlling the execution of your test cases
- Step feature for debugging your test cases
- Speed Control: Controls how fast your test case runs
- Run All: Runs the entire test suite when a test suite with multiple test cases is loaded
- Run: Runs the currently selected test. When only a single test is loaded this button and the Run All button have the same effect
- Pause/Resume: Allows stopping and re-starting of a running test case

The Selenium IDE test cases can be saved into following format:
HTML format
The Selenium IDE test cases can be exported into following formats/programming languages.
java (IDE exported in Java)
rb (IDE exported in Ruby)
py (IDE exported in Python)
cs (IDE exported in C#)



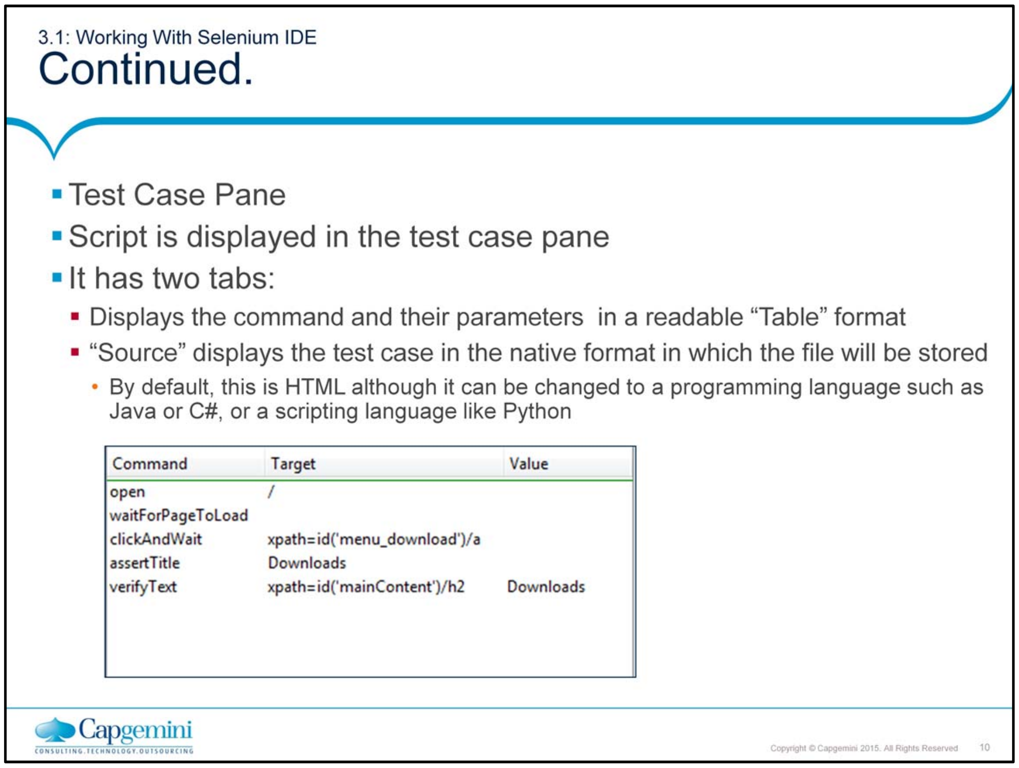
3.1: Working With Selenium IDE

Continued.

Tool Bar

- Step: Allows you to “step” through a test case by running it one command at a time. Use for debugging test cases
- Apply Rollup Rules: This advanced feature allows repetitive sequences of Selenium commands to be grouped into a single action. Detailed documentation on rollup rules can be found in the UI-Element Documentation on the Help menu
- Record: Records the user’s browser actions.

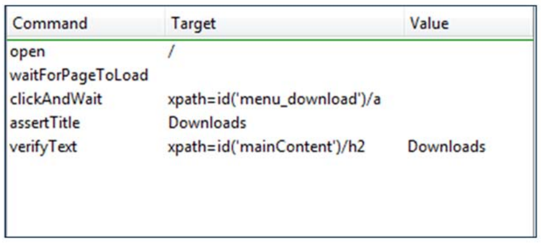
The Selenium IDE test cases can be saved into following format:
HTML format
The Selenium IDE test cases can be exported into following formats/programming languages.
java (IDE exported in Java)
rb (IDE exported in Ruby)
py (IDE exported in Python)
cs (IDE exported in C#)



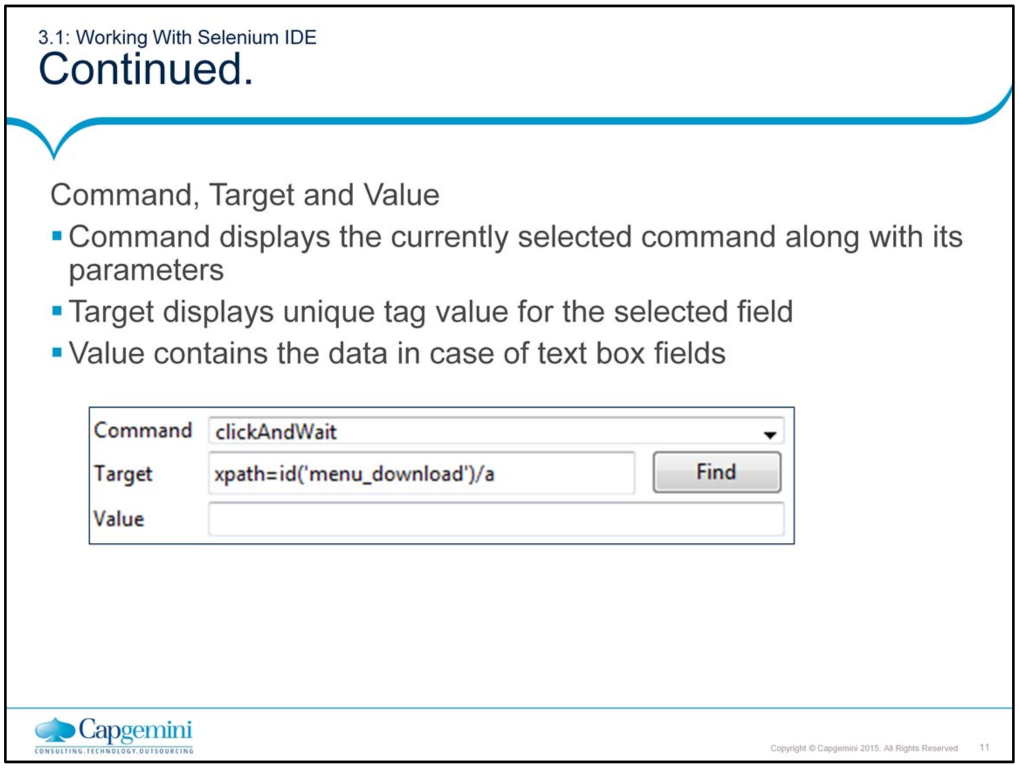
3.1: Working With Selenium IDE

Continued.

- Test Case Pane
- Script is displayed in the test case pane
- It has two tabs:
  - Displays the command and their parameters in a readable “Table” format
  - “Source” displays the test case in the native format in which the file will be stored
    - By default, this is HTML although it can be changed to a programming language such as Java or C#, or a scripting language like Python



Command	Target	Value
open	/	
waitForPageToLoad		
clickAndWait	xpath=id('menu_download')/a	
assertTitle	Downloads	
verifyText	xpath=id('mainContent')/h2	Downloads

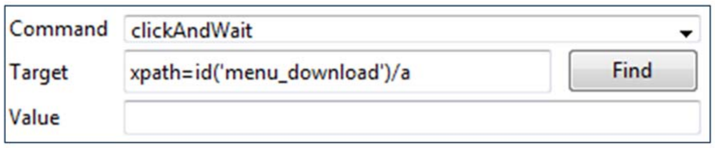


3.1: Working With Selenium IDE

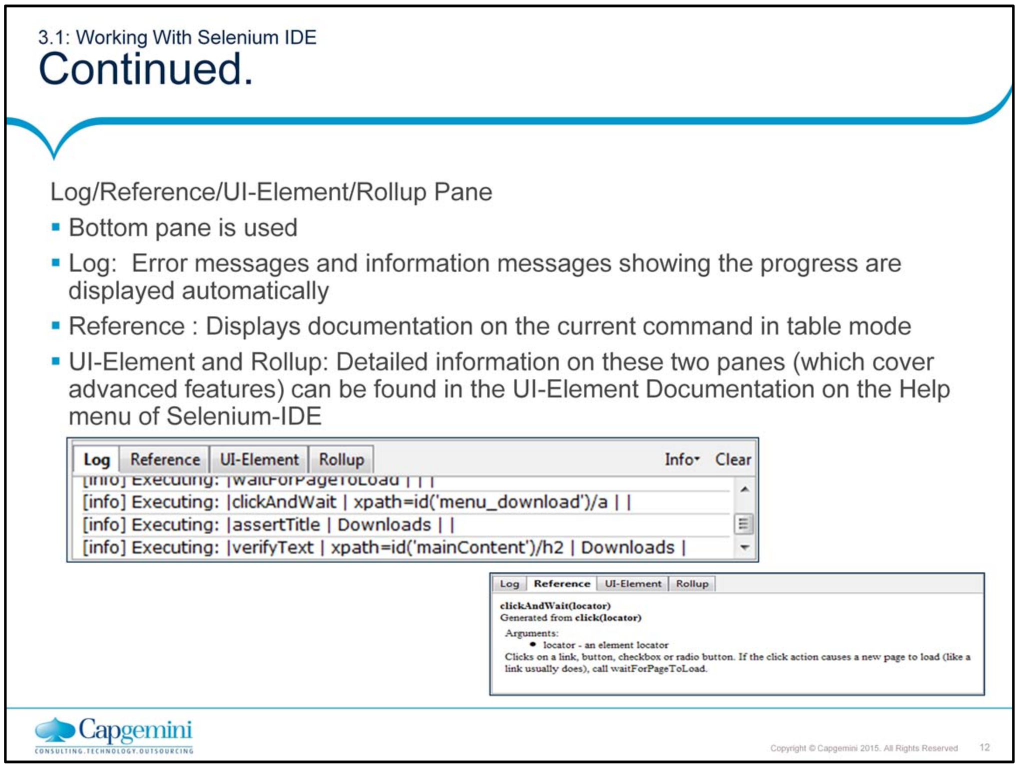
Continued.

Command, Target and Value

- Command displays the currently selected command along with its parameters
- Target displays unique tag value for the selected field
- Value contains the data in case of text box fields



Command	clickAndWait
Target	xpath=id('menu_download')/a
Value	



3.1: Working With Selenium IDE

Continued.

Log/Reference/UI-Element/Rollup Pane

- Bottom pane is used
- Log: Error messages and information messages showing the progress are displayed automatically
- Reference : Displays documentation on the current command in table mode
- UI-Element and Rollup: Detailed information on these two panes (which cover advanced features) can be found in the UI-Element Documentation on the Help menu of Selenium-IDE

```
[info] Executing: |waitForPageToLoad | | |
[info] Executing: |clickAndWait | xpath=id('menu_download')/a | |
[info] Executing: |assertTitle | Downloads | |
[info] Executing: |verifyText | xpath=id('mainContent')/h2 | Downloads |
```

clickAndWait(locator)
Generated from **click(locator)**
Arguments:
- locator - an element locator

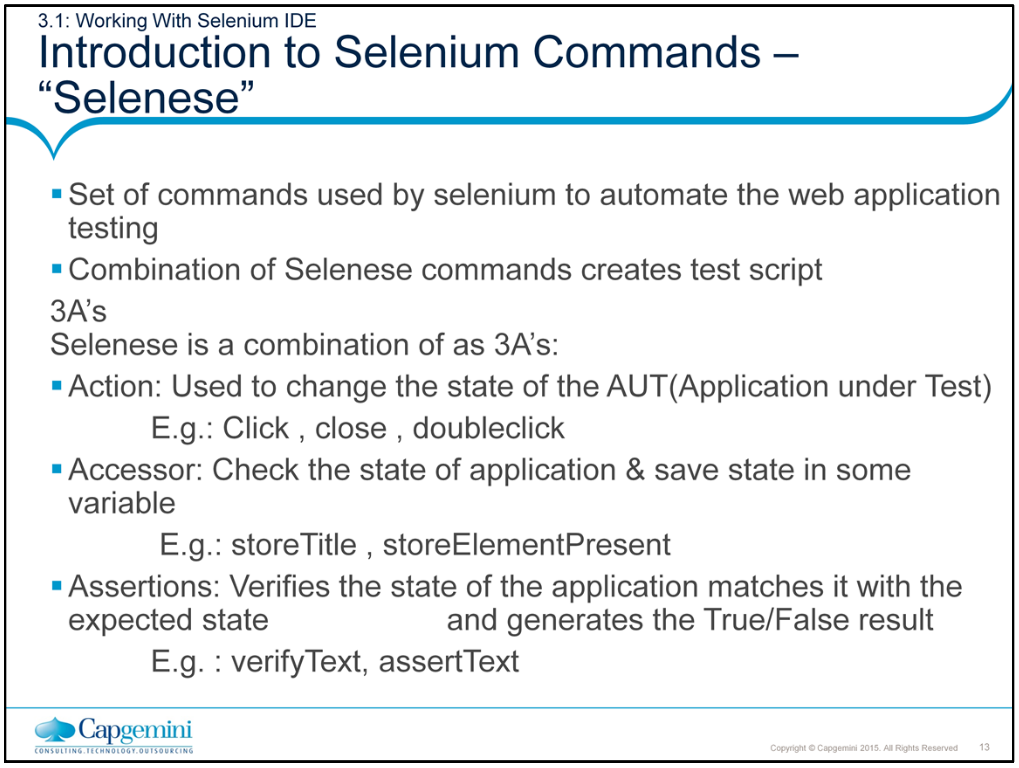
Clicks on a link, button, checkbox or radio button. If the click action causes a new page to load (like a link usually does), call waitForPageToLoad.

Log/Reference/UI-Element/Rollup Pane
The bottom pane is used for four different functions–Log, Reference, UI-Element, and Rollup–depending on which tab is selected.

Log
When you run your test case, error messages and information messages showing the progress are displayed in this pane automatically, even if you do not first select the Log tab. These messages are often useful for test case debugging. Notice the Clear button for clearing the Log. Also notice the Info button is a drop-down allowing selection of different levels of information to log.

Reference
The Reference tab is the default selection whenever you are entering or modifying Selenese commands and parameters in Table mode. In Table mode, the Reference pane will display documentation on the current command. When entering or modifying commands, whether from Table or Source mode, it is critically important to ensure that the parameters specified in the Target and Value fields match those specified in the parameter list in the Reference pane. The number of parameters provided must match the number specified, the order of parameters provided must match the order specified, and the type of parameters provided must match the type specified. If there is a mismatch in any of these three areas, the command will not run correctly.

While the Reference tab is invaluable as a quick reference, it is still often necessary to consult the Selenium Reference document.



3.1: Working With Selenium IDE

Introduction to Selenium Commands – "Selenese"

- Set of commands used by selenium to automate the web application testing
- Combination of Selenese commands creates test script

3A's
Selenese is a combination of as 3A's:

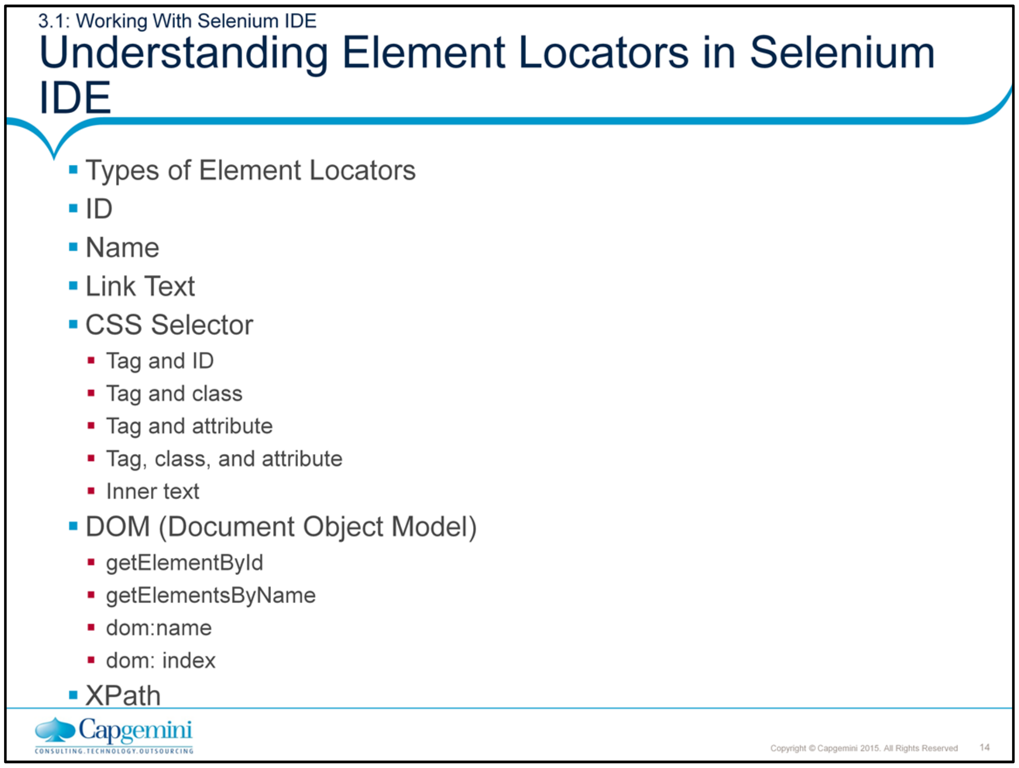
- Action: Used to change the state of the AUT(Application under Test)
  E.g.: Click , close , doubleclick
- Accessor: Check the state of application & save state in some variable
  E.g.: storeTitle , storeElementPresent
- Assertions: Verifies the state of the application matches it with the expected state and generates the True/False result
  E.g. : verifyText, assertText

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Action:
Used to change the state of the AUT(Application under Test)like click on some link, type a value in edit box, select some options on the page, select a value from drop down etc.
When action is performed on AUT the test will fail if the action is not achieved.

Accessor:
This command check the state of the application and save the application state in some variable. It can be used for automatic generation of assertions.

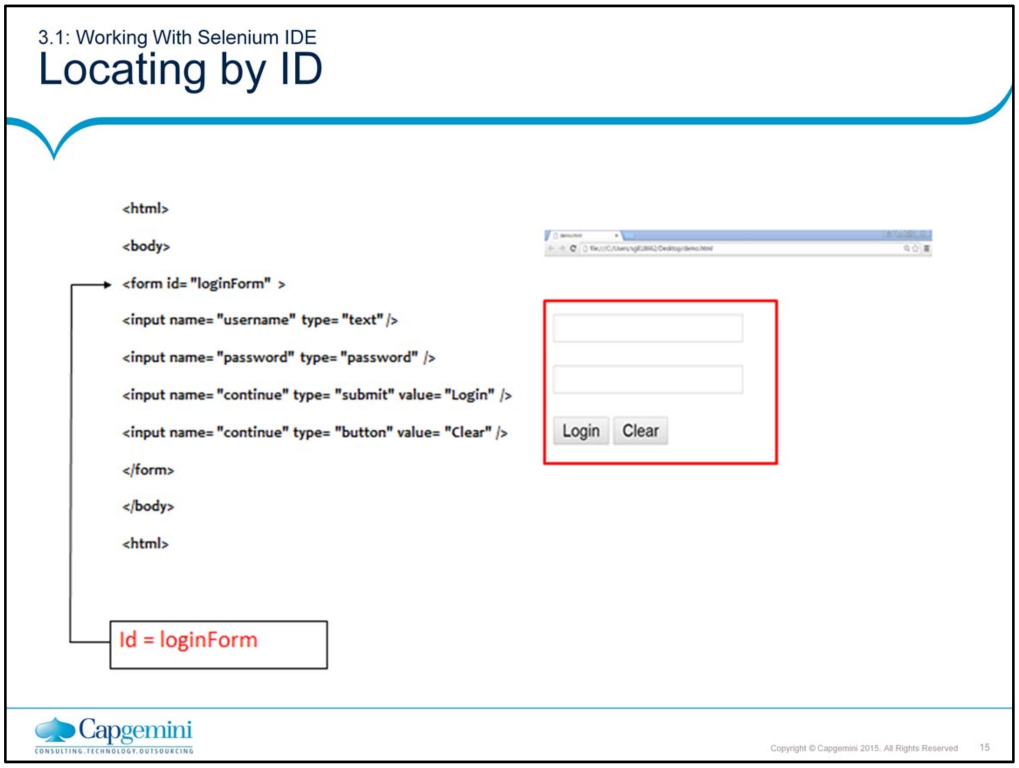
Assertions:
Are very similar to checkpoint in UFT/QTP. Assertion verifies the state of the application matches it with the expected state and generates the True/False result.



3.1: Working With Selenium IDE

Understanding Element Locators in Selenium IDE

- Types of Element Locators
- ID
- Name
- Link Text
- CSS Selector
  - Tag and ID
  - Tag and class
  - Tag and attribute
  - Tag, class, and attribute
  - Inner text
- DOM (Document Object Model)
  - getElementById
  - getElementsByName
  - dom:name
  - dom: index
- XPath



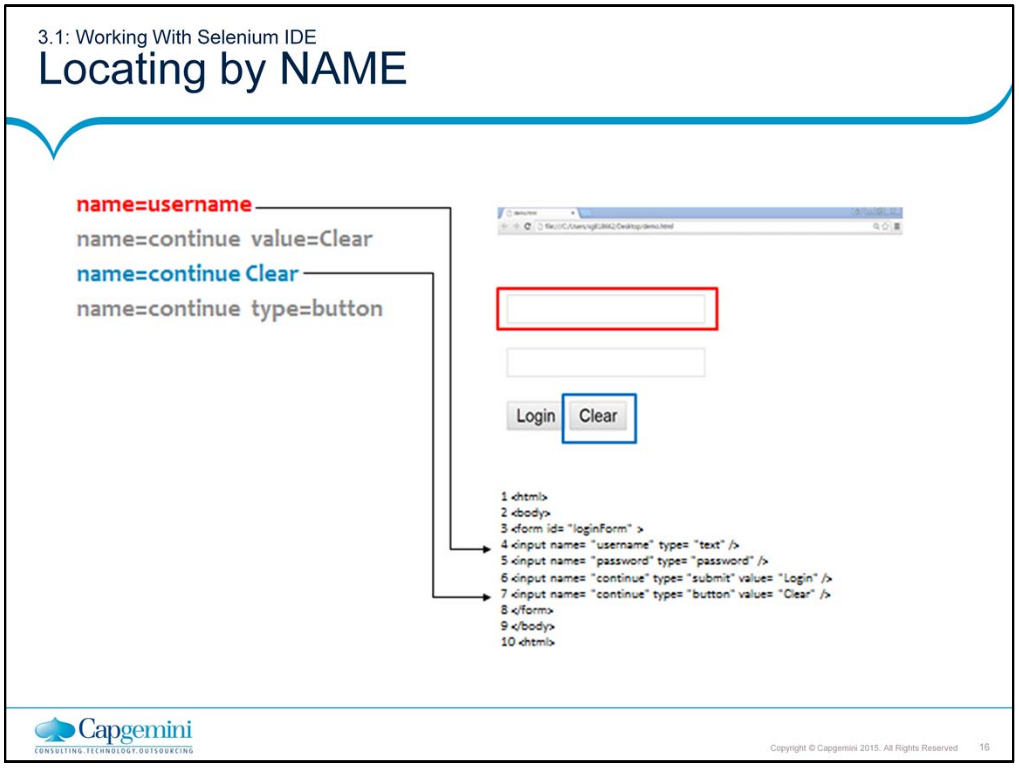
3.1: Working With Selenium IDE

Locating by ID

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text" />
<input name= "password" type= "password" />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

Id = loginForm

Login
Clear



3.1: Working With Selenium IDE

Locating by NAME

name=username

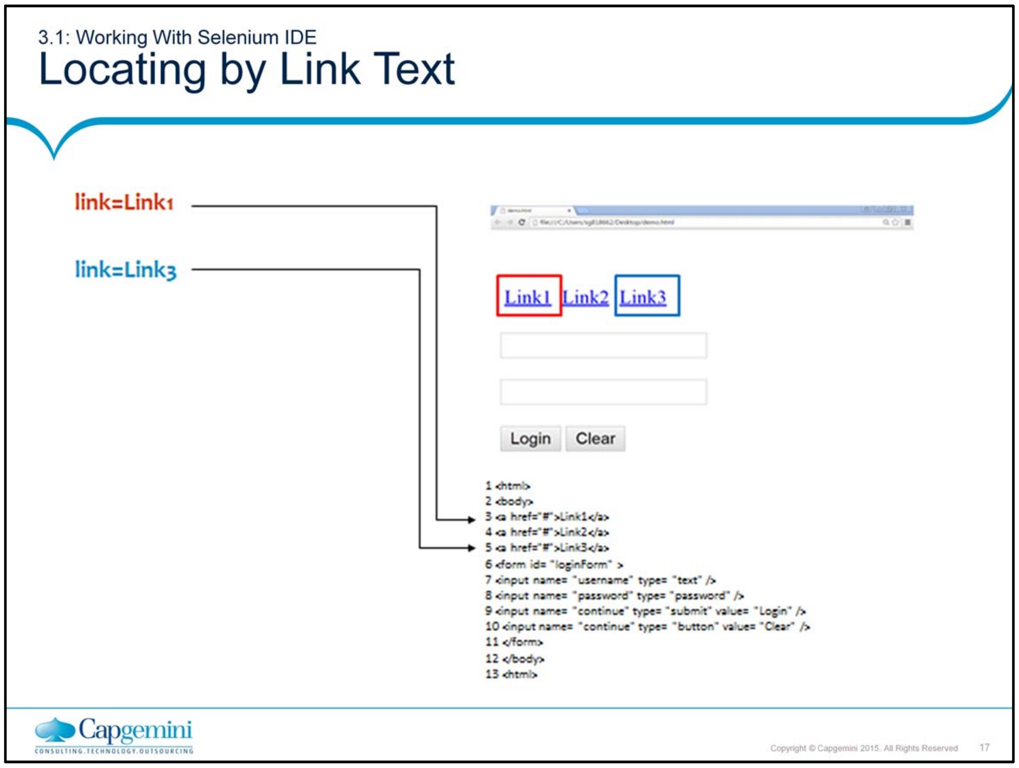
name=continue value=Clear

name=continue Clear

name=continue type=button

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text" />
<input name= "password" type= "password" />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

16



3.1: Working With Selenium IDE

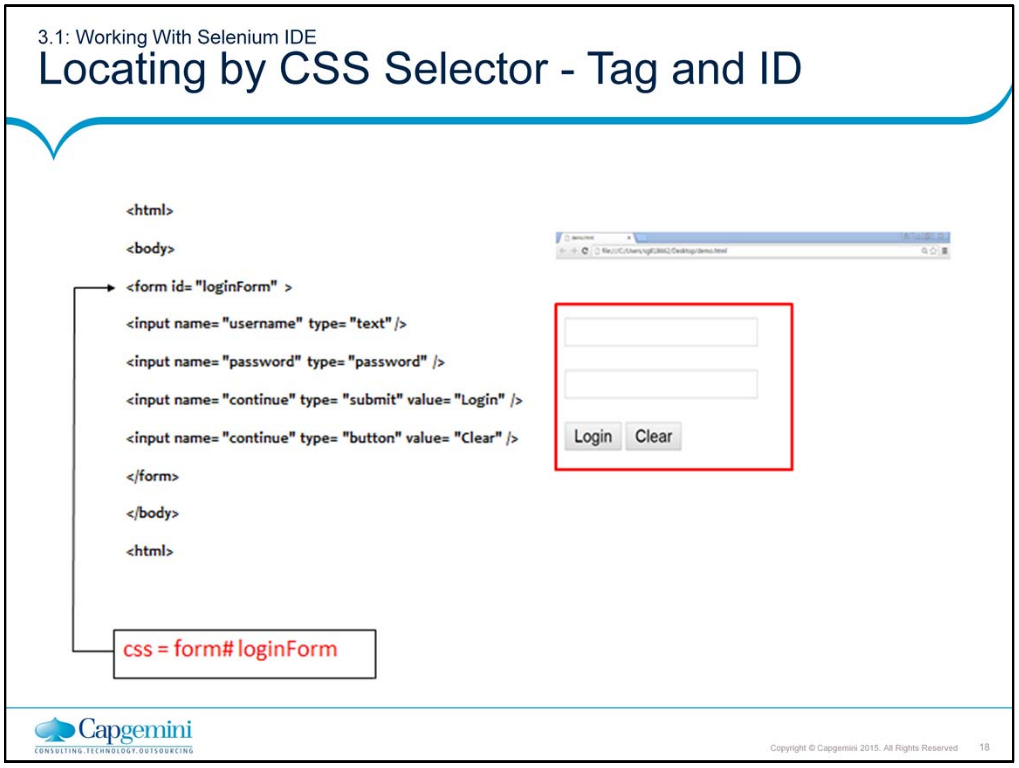
Locating by Link Text

link=Link1
link=Link3
Link1
Link2
Link3
Login
Clear

```
<html>
<body>
<a href="#">Link1</a>
<a href="#">Link2</a>
<a href="#">Link3</a>
<form id= "loginForm" >
<input name= "username" type= "text" />
<input name= "password" type= "password" />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

17

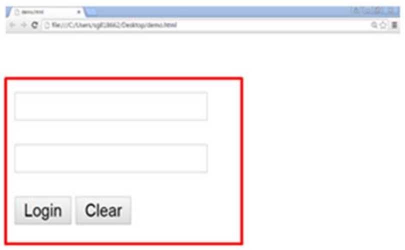


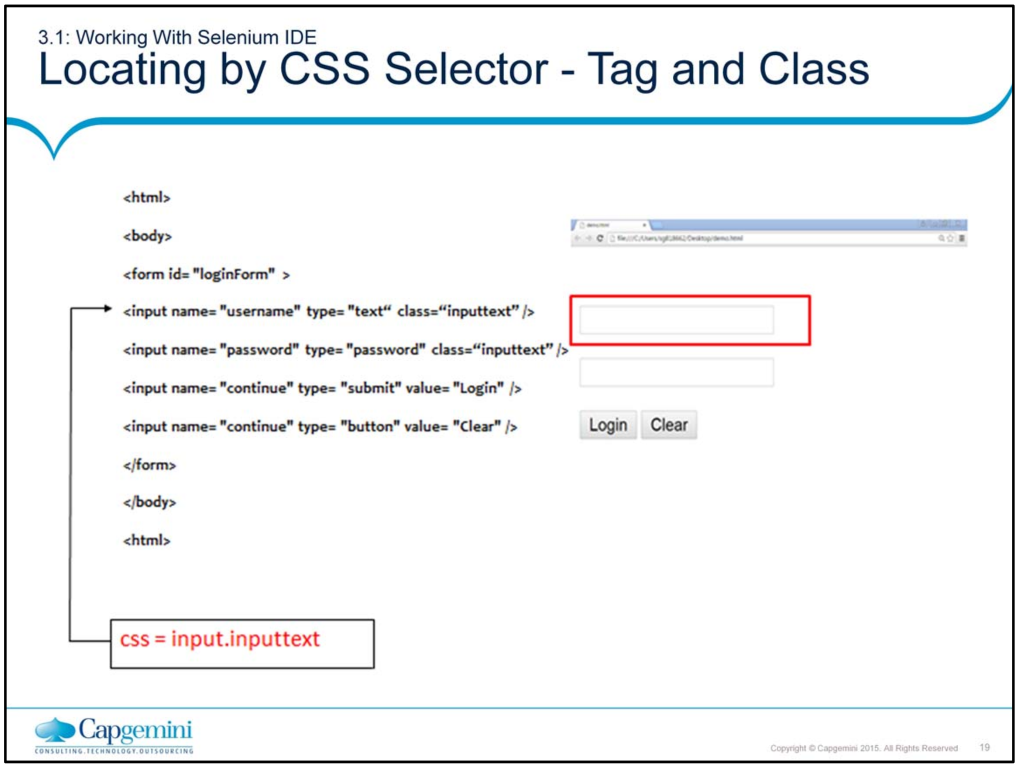
3.1: Working With Selenium IDE

Locating by CSS Selector - Tag and ID

```
<html>
<body>
<form id= "loginForm"  >
<input name= "username"  type= "text" />
<input name= "password"  type= "password"  />
<input name= "continue" type= "submit" value= "Login"  />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

css = form# loginForm



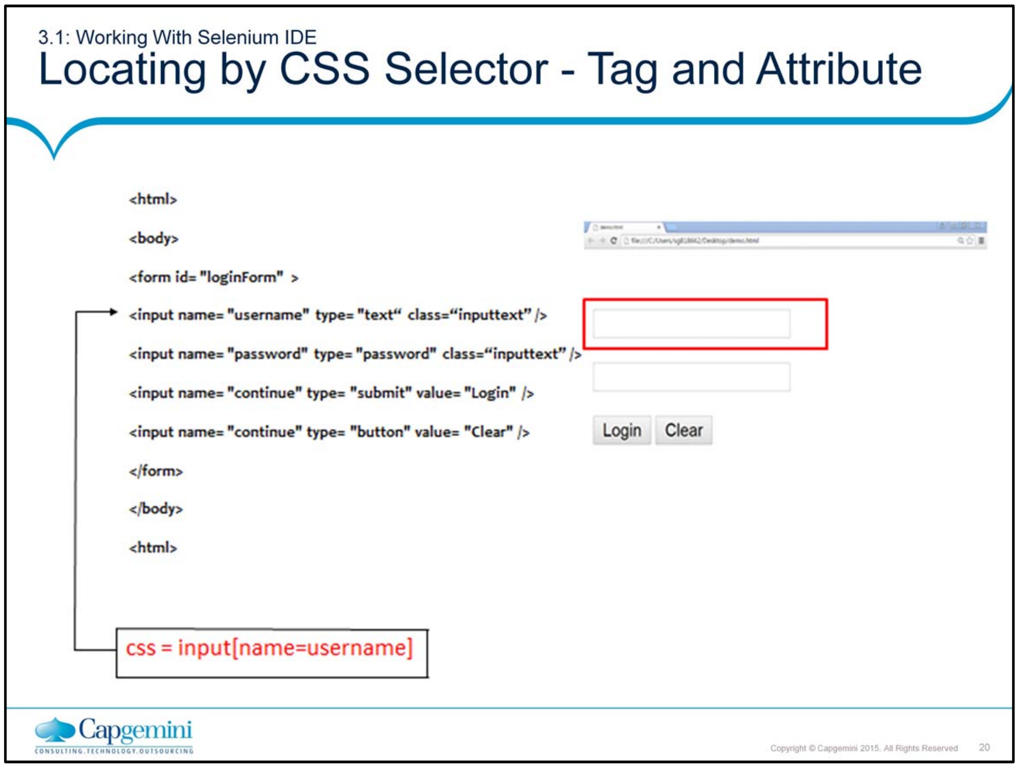


3.1: Working With Selenium IDE

Locating by CSS Selector - Tag and Class

```
<html>
<body>
<form id= "loginForm"  >
<input name= "username"  type= "text“  class=“inputtext” />
<input name= "password"  type= "password"  class=“inputtext” />
<input name= "continue"  type= "submit"  value= "Login"  />
<input name= "continue"  type= "button"  value= "Clear" />
</form>
</body>
<html>
```

css = input.inputtext

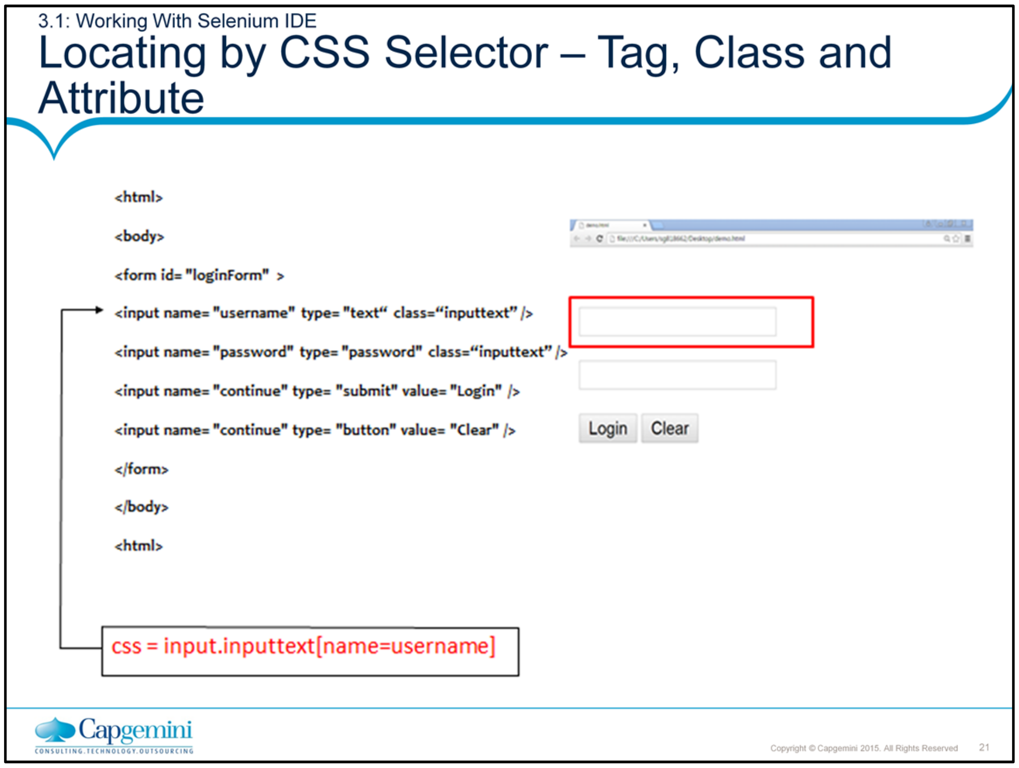


3.1: Working With Selenium IDE

Locating by CSS Selector - Tag and Attribute

```
<html>
<body>
<form id= "loginForm"  >
<input name= "username" type= "text" class="inputtext" />
<input name= "password" type= "password" class="inputtext" />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

css = input[name=username]

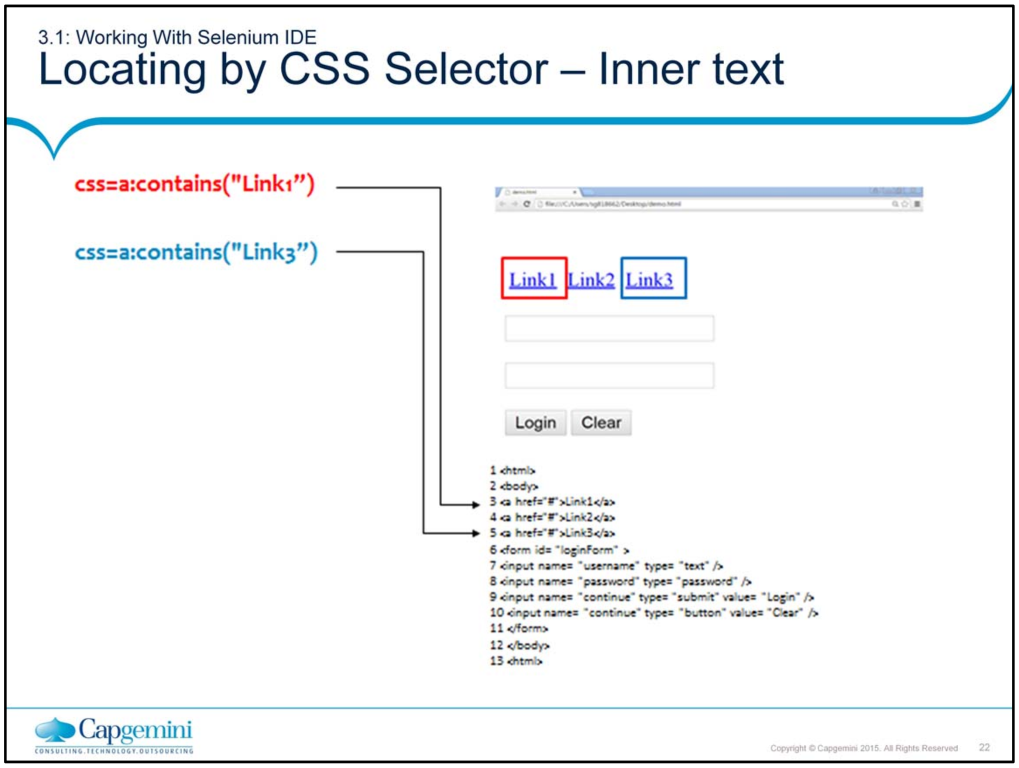


3.1: Working With Selenium IDE

Locating by CSS Selector – Tag, Class and Attribute

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

css = input.inputtext[name=username]

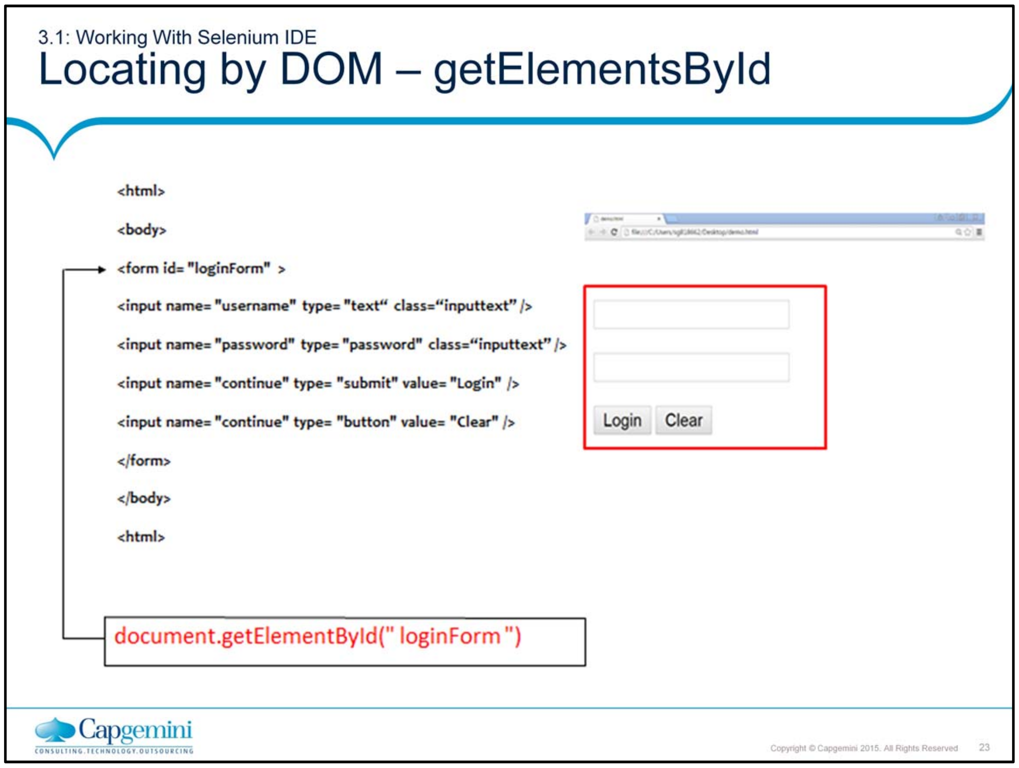


3.1: Working With Selenium IDE

Locating by CSS Selector – Inner text

```
css=a:contains("Link1")
css=a:contains("Link3")
```

```
<html>
<body>
<a href="#">Link1</a>
<a href="#">Link2</a>
<a href="#">Link3</a>
<form id= "loginForm" >
<input name= "username" type= "text" />
<input name= "password" type= "password" />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

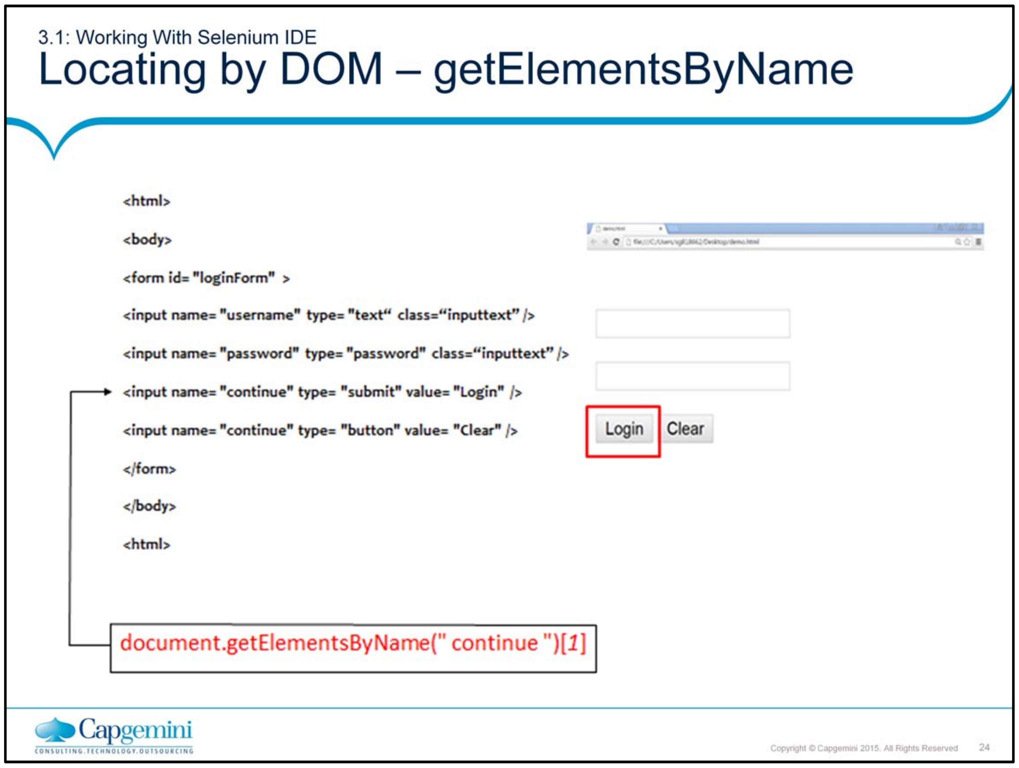


3.1: Working With Selenium IDE

Locating by DOM – getElementsById

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

```
document.getElementById(" loginForm ")
```



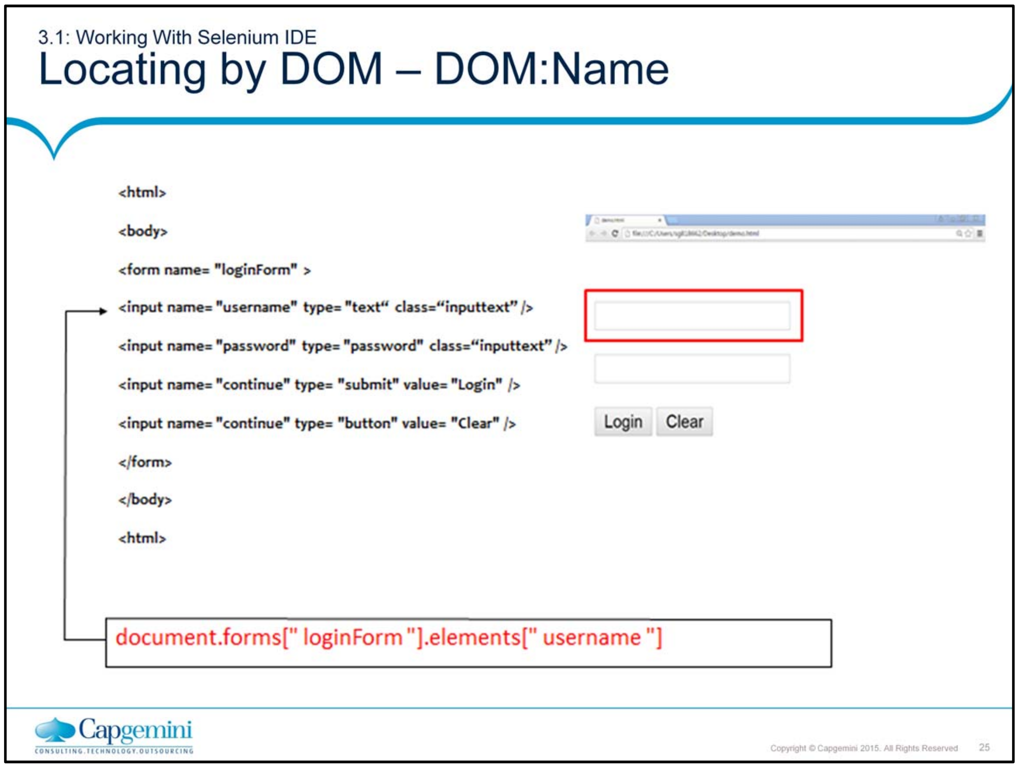
3.1: Working With Selenium IDE

Locating by DOM – getElementsByName

```
<html>
<body>
<form id= "loginForm"  >
<input name= "username"  type= "text“  class=“inputtext” />
<input name= "password"  type= "password"  class=“inputtext” />
<input name= "continue"  type= "submit"  value= "Login"  />
<input name= "continue"  type= "button"  value= "Clear" />
</form>
</body>
<html>
```

document.getElementsByName(" continue ")[1]

Copyright © Capgemini 2015. All Rights Reserved 24

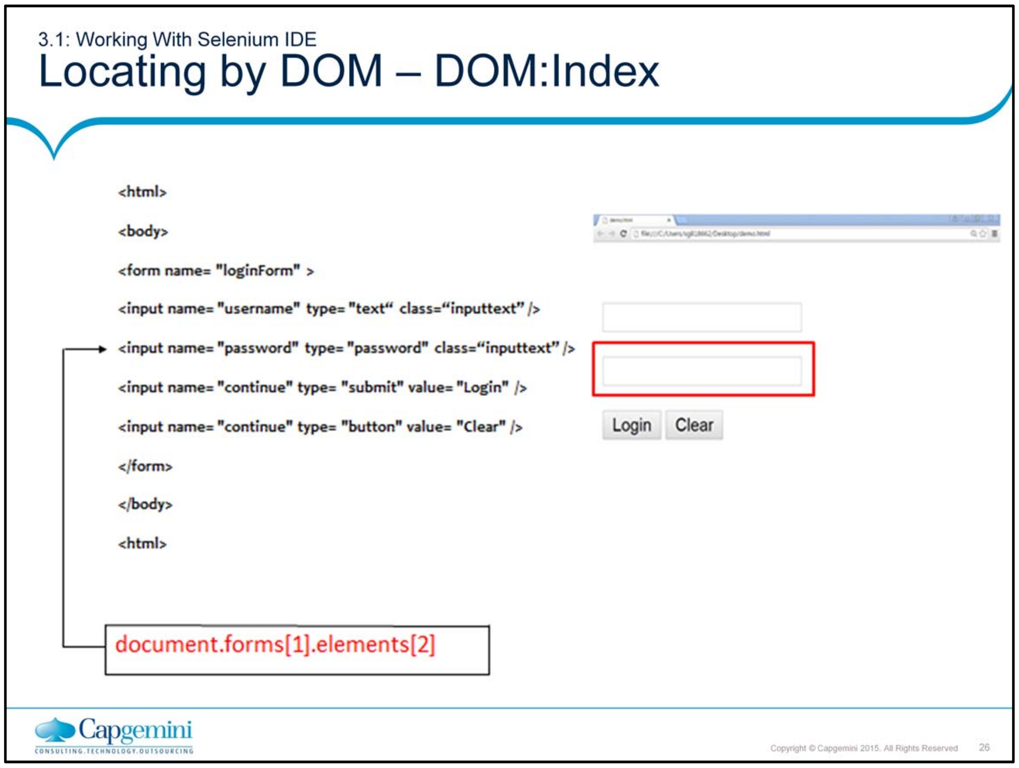


3.1: Working With Selenium IDE

Locating by DOM – DOM:Name

```
<html>
<body>
<form name= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

```
document.forms[" loginForm "].elements[" username "]
```

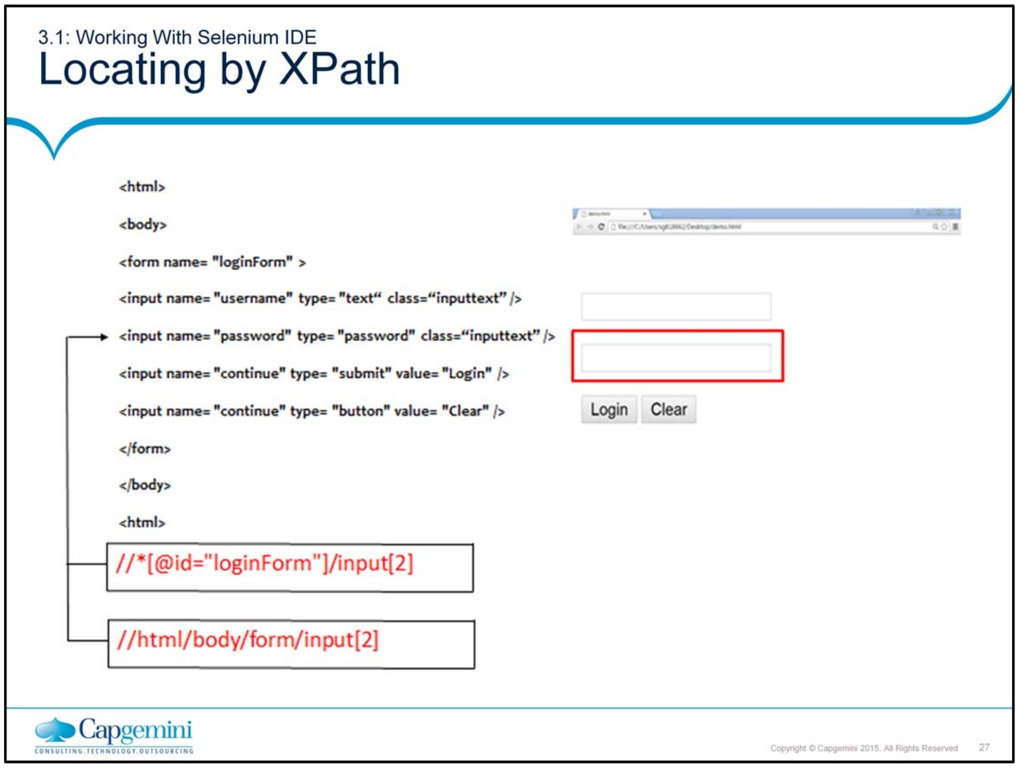


3.1: Working With Selenium IDE

Locating by DOM – DOM:Index

```
<html>
<body>
<form name= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

document.forms[1].elements[2]



3.1: Working With Selenium IDE

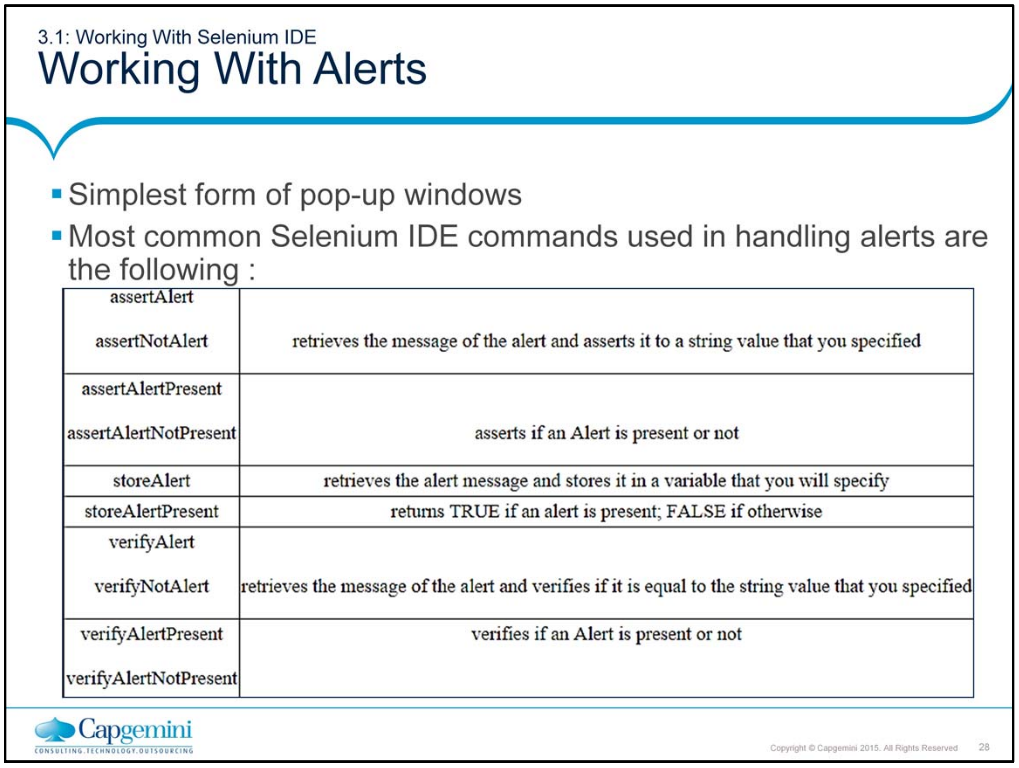
Locating by XPath

```
<html>
<body>
<form name= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

```
//*[@id="loginForm"]/input[2]
```

```
//html/body/form/input[2]
```

Login
Clear



3.1: Working With Selenium IDE

Working With Alerts

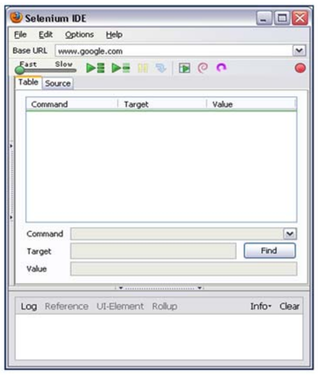
- Simplest form of pop-up windows
- Most common Selenium IDE commands used in handling alerts are the following :

assertAlert assertNotAlert	retrieves the message of the alert and asserts it to a string value that you specified
assertAlertPresent assertAlertNotPresent	asserts if an Alert is present or not
storeAlert	retrieves the alert message and stores it in a variable that you will specify
storeAlertPresent	returns TRUE if an alert is present; FALSE if otherwise
verifyAlert verifyNotAlert	retrieves the message of the alert and verifies if it is equal to the string value that you specified
verifyAlertPresent verifyAlertNotPresent	verifies if an Alert is present or not

3.1: Working With Selenium IDE

Creating Test Scripts using Selenium IDE

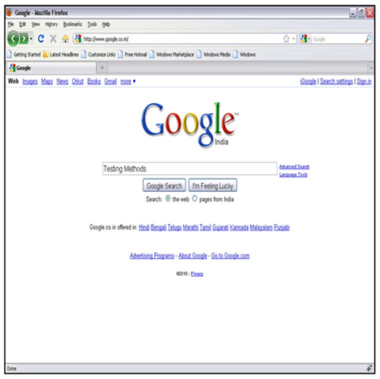
- Go to the Web Page for which you want to carry out the test
- Hit the record button on IDE



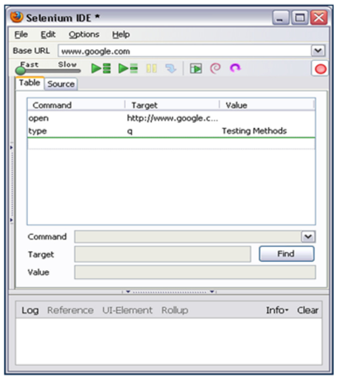
3.1: Working With Selenium IDE

Creating Test Scripts using Selenium IDE(Contd.)

- Enter the text on Web Page and submit



- IDE should be updated, stop the recorder and add the assertions

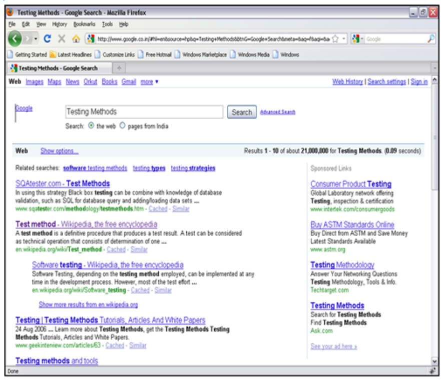


Base URL: www.google.com

Command	Target	Value
open	http://www.google.c...	
type	q	Testing Methods

3.1: Working With Selenium IDE

Creating Test Scripts using Selenium IDE(Contd.)


Testing Methods - Google Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.google.co.in/#hl=en&source=hp&q=Testing+Methods&btnG=Google+Search&meta=&aq=f&aqi=&a
Getting Started Latest Headlines Customize Links Free Hotmail Windows Marketplace Windows Media Windows
Testing Methods - Google Search
Web Images Maps News Orkut Books Gmail more
Web History | Search settings | Sign in
Search Advanced Search
Done

Testing Methods

Search: the web pages from India

Web Show options... Results **1** - **10** of about **21,000,000** for **Testing Methods**. (**0.09** seconds)

Related searches: software testing methods testing types testing strategies

SQAtester.com - Test Methods
In using this strategy Black box **testing** can be combine with knowledge of database validation, such as SQL for database query and adding/loading data sets ...
www.sqatester.com/methodology/testmethods.htm - Cached - Similar

Test method - Wikipedia, the free encyclopedia
A **test method** is a definitive procedure that produces a test result. A test can be considered as technical operation that consists of determination of one ...
en.wikipedia.org/wiki/Test_method - Cached - Similar

Software testing - Wikipedia, the free encyclopedia
Software Testing, depending on the **testing method** employed, can be implemented at any time in the development process. However, most of the test effort ...
en.wikipedia.org/wiki/Software_testing - Cached - Similar

Show more results from en.wikipedia.org

Testing | Testing Methods Tutorials, Articles And White Papers
24 Aug 2006 ... Learn more about **Testing Methods**, get the **Testing Methods Testing Methods** Tutorials, Articles and White Papers.
www.geekinterview.com/articles/63 - Cached - Similar

Testing methods and tools

Sponsored Links

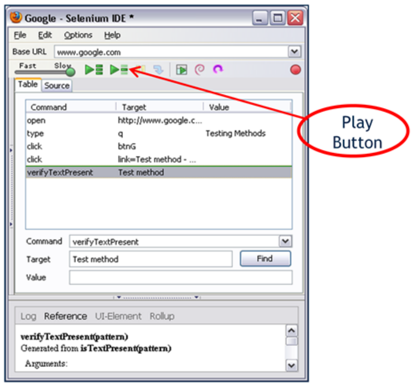
Consumer Product Testing
Global Laboratory network offering **Testing**, inspection & certification
www.intertek.com/consumergoods

Buy ASTM Standards Online
Buy Direct from ASTM and Save Money
Latest Standards Available
www.astm.org

Testing Methodology
Answer Your Networking Questions
Testing Methodology, Tools & Info.
Techtarget.com

Testing Methods
Search for **Testing Methods**
Find **Testing Methods**
Ask.com

See your ad here »



Base URL www.google.com

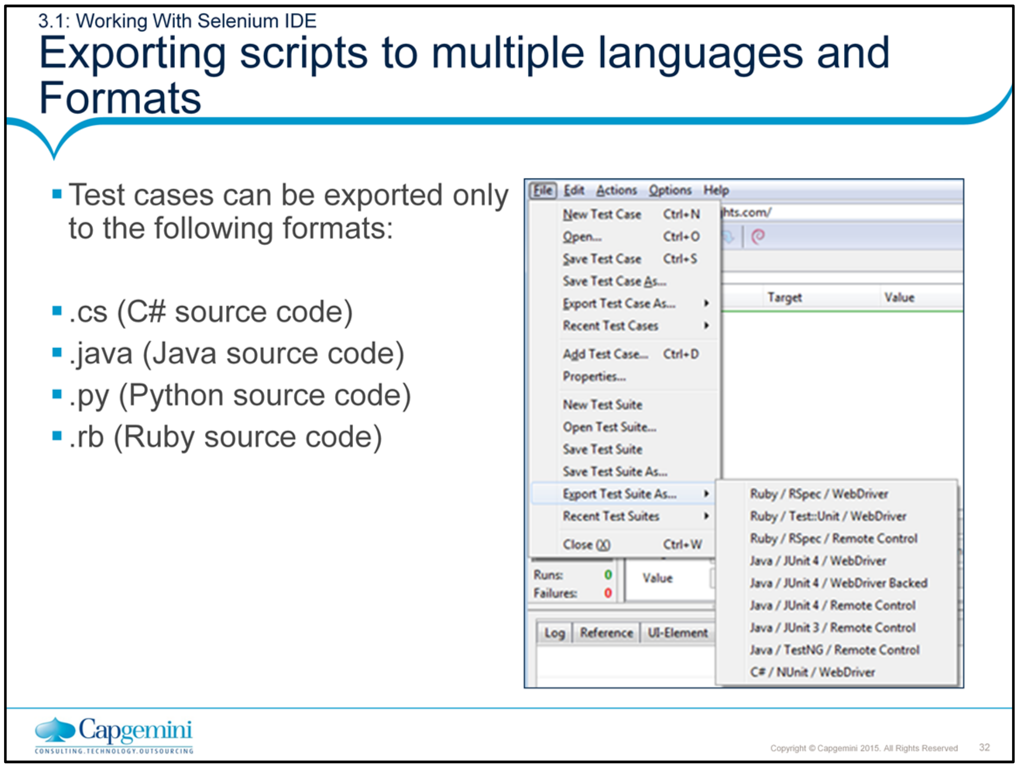
Command	Target	Value
open	http://www.google.c...	
type	q	Testing Methods
click	btnG	
click	link=Test method - ...	
verifyTextPresent	Test method	

Command verifyTextPresent
Target Test method
Value

verifyTextPresent(pattern)
Generated from **isTextPresent(pattern)**
Arguments:

Hit the play button to play the recorded scripts

Capgemini Internal

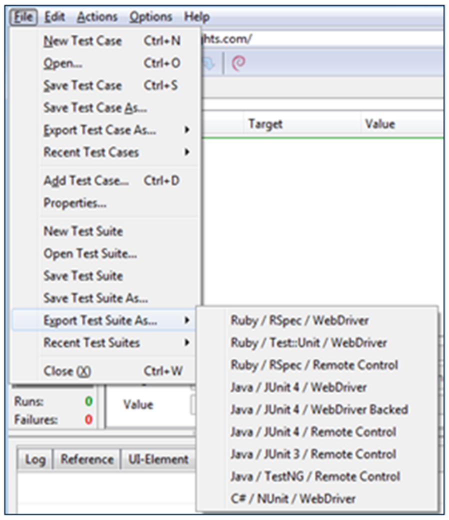


3.1: Working With Selenium IDE

Exporting scripts to multiple languages and Formats

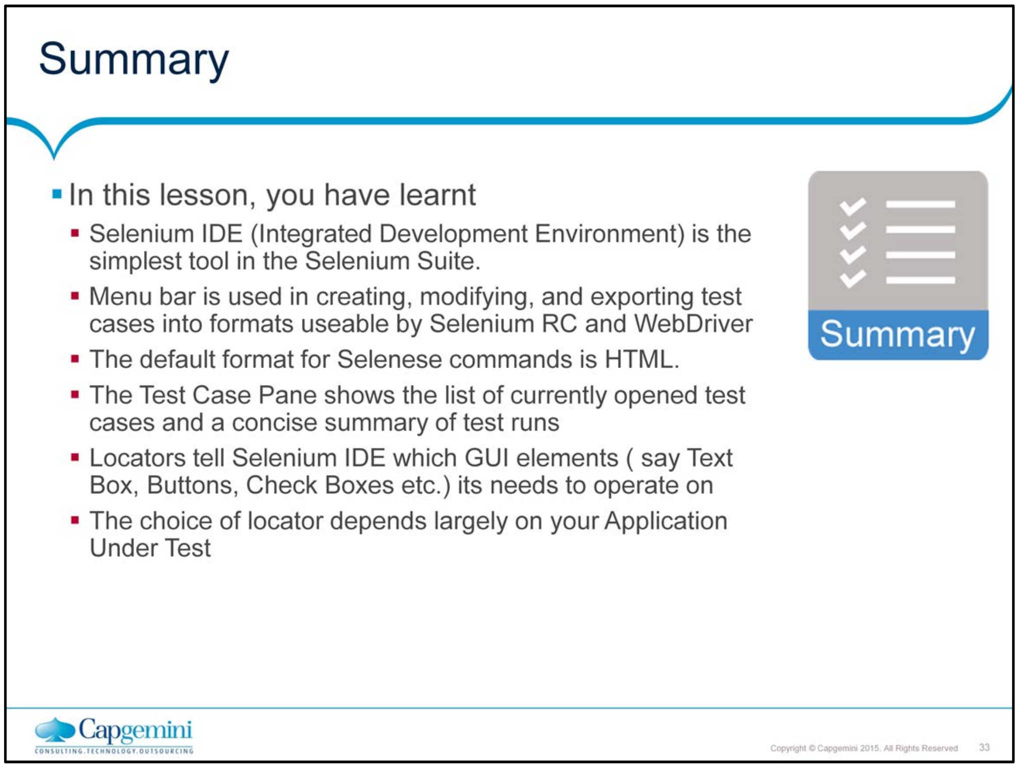
- Test cases can be exported only to the following formats:

- .cs (C# source code)
- .java (Java source code)
- .py (Python source code)
- .rb (Ruby source code)



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

32



Summary

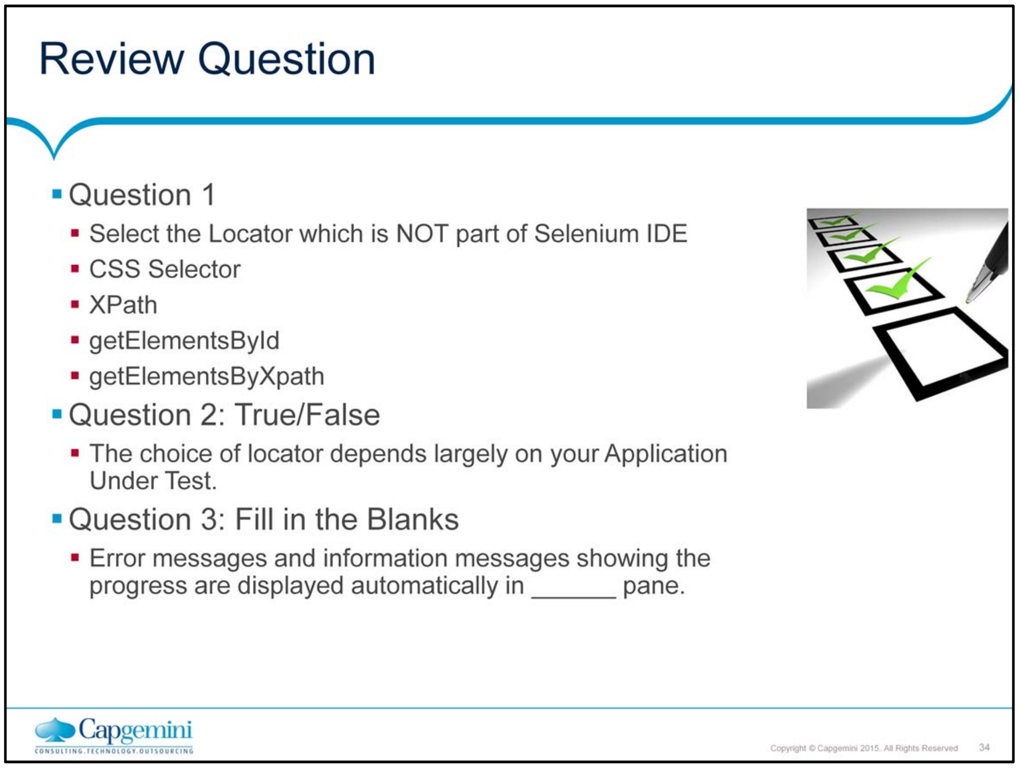
- In this lesson, you have learnt
  - Selenium IDE (Integrated Development Environment) is the simplest tool in the Selenium Suite.
  - Menu bar is used in creating, modifying, and exporting test cases into formats useable by Selenium RC and WebDriver
  - The default format for Selenese commands is HTML.
  - The Test Case Pane shows the list of currently opened test cases and a concise summary of test runs
  - Locators tell Selenium IDE which GUI elements (say Text Box, Buttons, Check Boxes etc.) its needs to operate on
  - The choice of locator depends largely on your Application Under Test

Summary

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

33

Add the notes here.



Review Question

- Question 1
  - Select the Locator which is NOT part of Selenium IDE
  - CSS Selector
  - XPath
  - getElementsById
  - getElementsByXpath
- Question 2: True/False
  - The choice of locator depends largely on your Application Under Test.
- Question 3: Fill in the Blanks
  - Error messages and information messages showing the progress are displayed automatically in ______ pane.