Capgemini

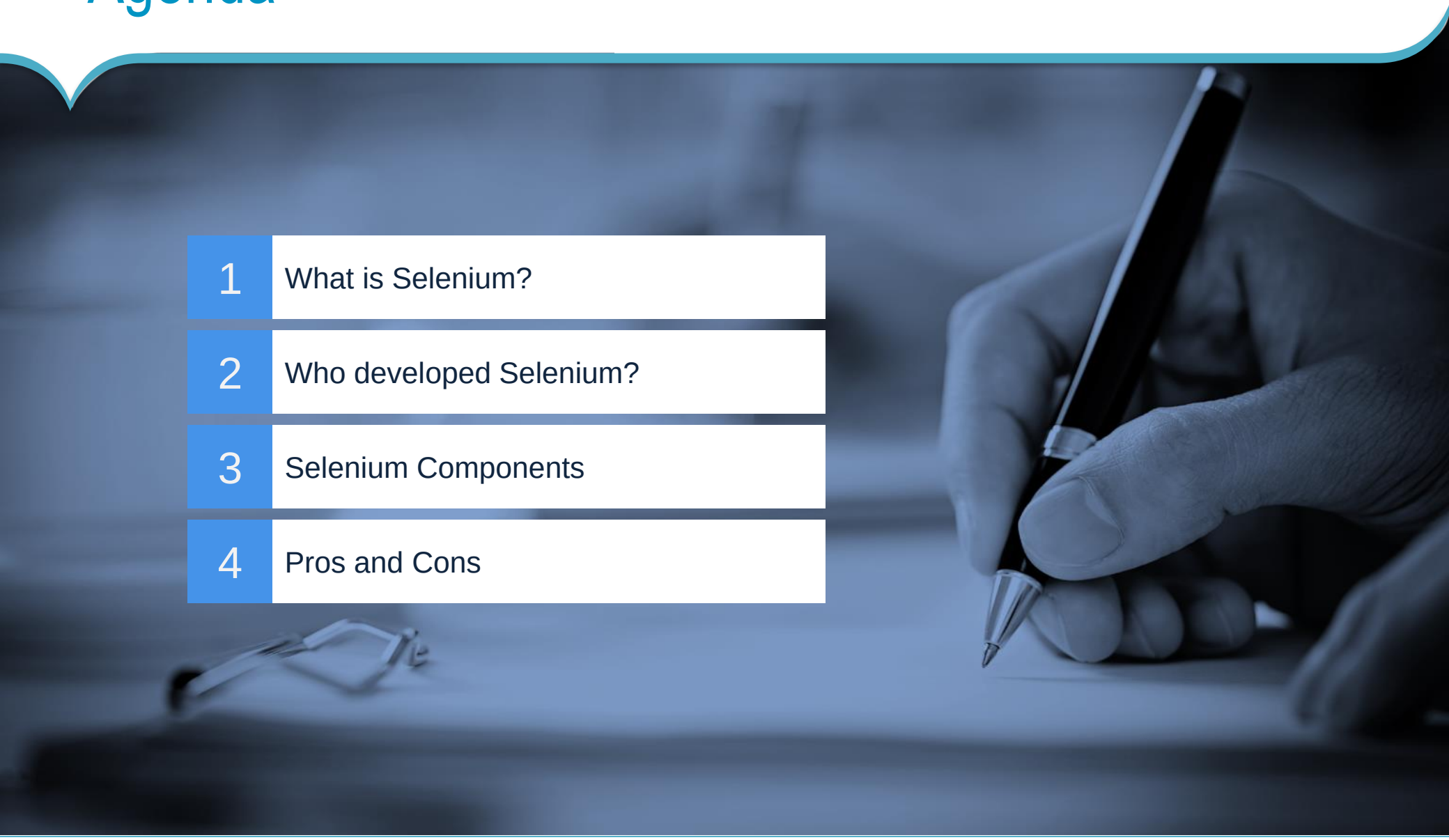
CONSULTING.TECHNOLOGY.OUTSOURCING

Selenium

2017

People matter, results count.

Agenda



1. What is Selenium?
2. Who developed Selenium?
3. Selenium Components
4. Pros and Cons

What is Selenium

What is Selenium

- Open Source
- Automation Tool
- QTP and UFT
- Focuses on automating web-based applications
- Referred as Selenium Testing

Who developed Selenium?

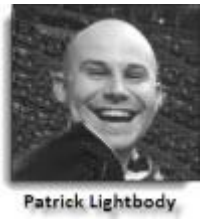
1. Primarily, Selenium was **created by Jason Huggins in 2004**. He named this program as the "**JavaScriptTestRunner.**“ JavaScriptRunner open-source which was later re-named as **Selenium Core**.

2. **Same Origin policy prohibits JavaScript code from accessing elements from a domain that is different from where it was launched**. **Birth of Selenium Remote Control (Selenium RC). Paul Hammant**, decided to create a server that will act as an HTTP proxy to "trick" the browser into believing that Selenium Core and the web application being tested come from the same domain. This system became known as the **Selenium Remote Control** or **Selenium 1**.


Paul Hammant

Who developed Selenium?

3. **Birth of Selenium Grid,** Selenium Grid was developed by **Patrick Lightbody** to address the need of minimizing test execution times as much as possible. He initially called the system "**Hosted QA**." It was capable of capturing browser screenshots during significant stages, and also of **sending out Selenium commands to different machines simultaneously.**



4. **Birth of Selenium IDE, Shinya Kasatani** of Japan created **Selenium IDE**, a Firefox extension that can automate the browser through a record-and-playback feature. He came up with this idea to further increase the speed in creating test cases. He donated Selenium IDE to the Selenium Project in **2006**.

Who developed Selenium?

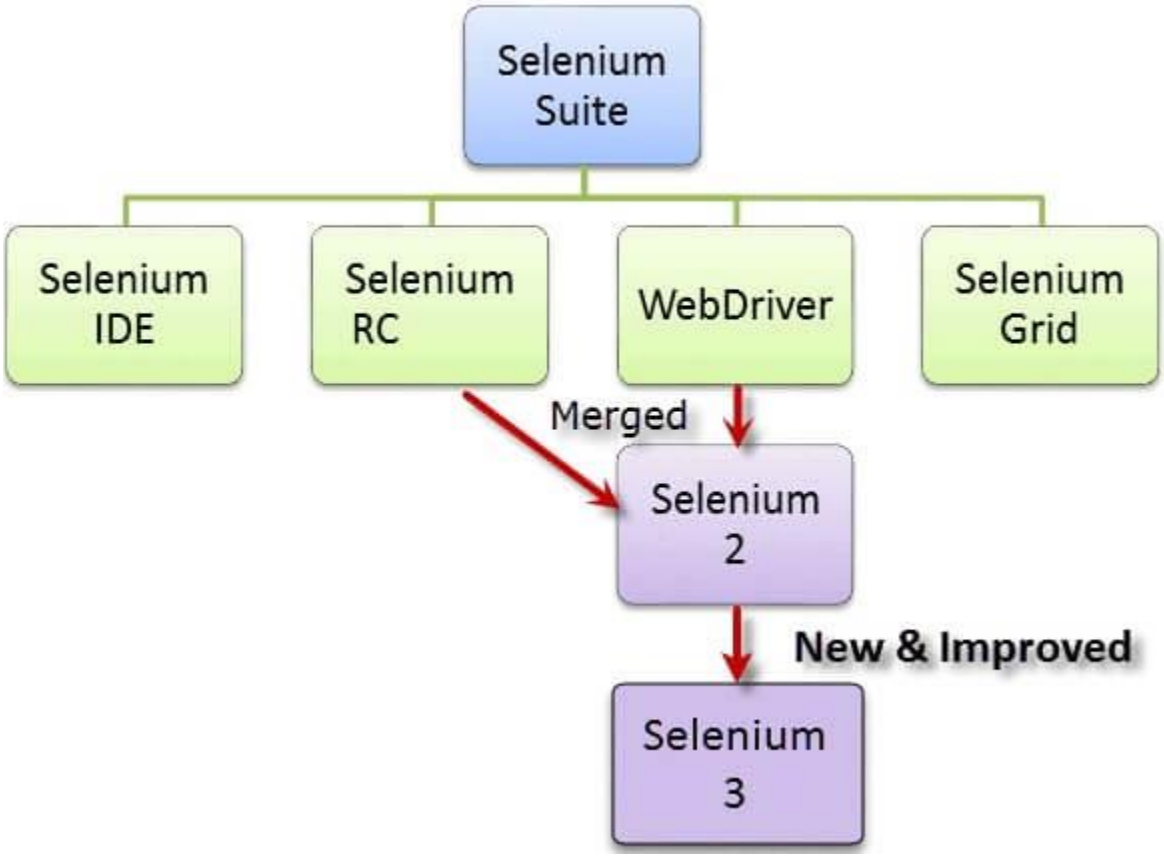
5. **Birth of WebDriver, Simon Stewart** created WebDriver circa **2006** when browsers and web applications were becoming more powerful and more restrictive with JavaScript programs like Selenium Core. **It was the first cross-platform testing framework that could control the browser from the OS level.**


Simon Stewart

6. **Birth of Selenium 2**

In **2008**, the whole Selenium Team decided to merge WebDriver and Selenium RC to form a more powerful tool called **Selenium 2**, with **WebDriver being the core**.

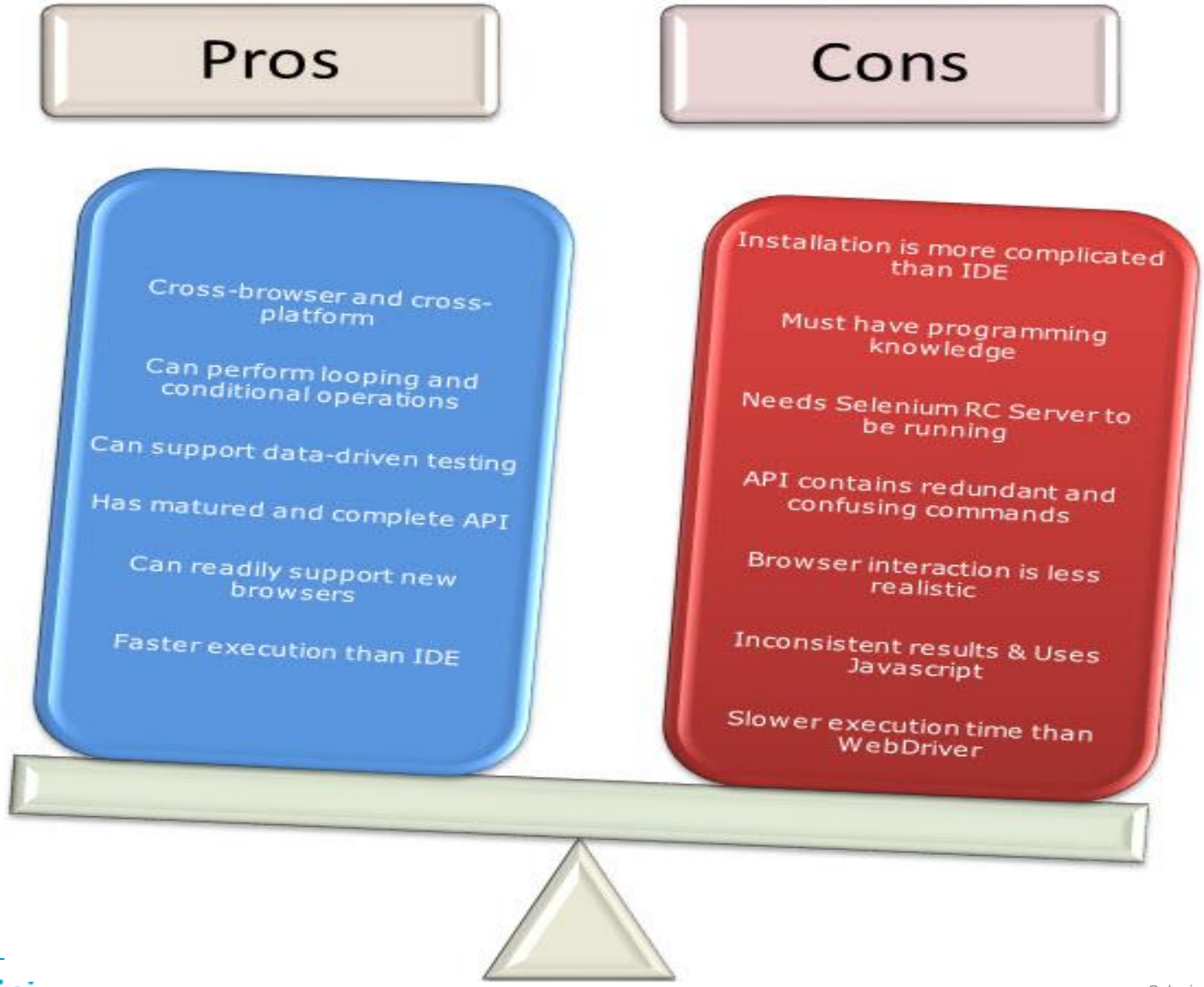
Selenium Components



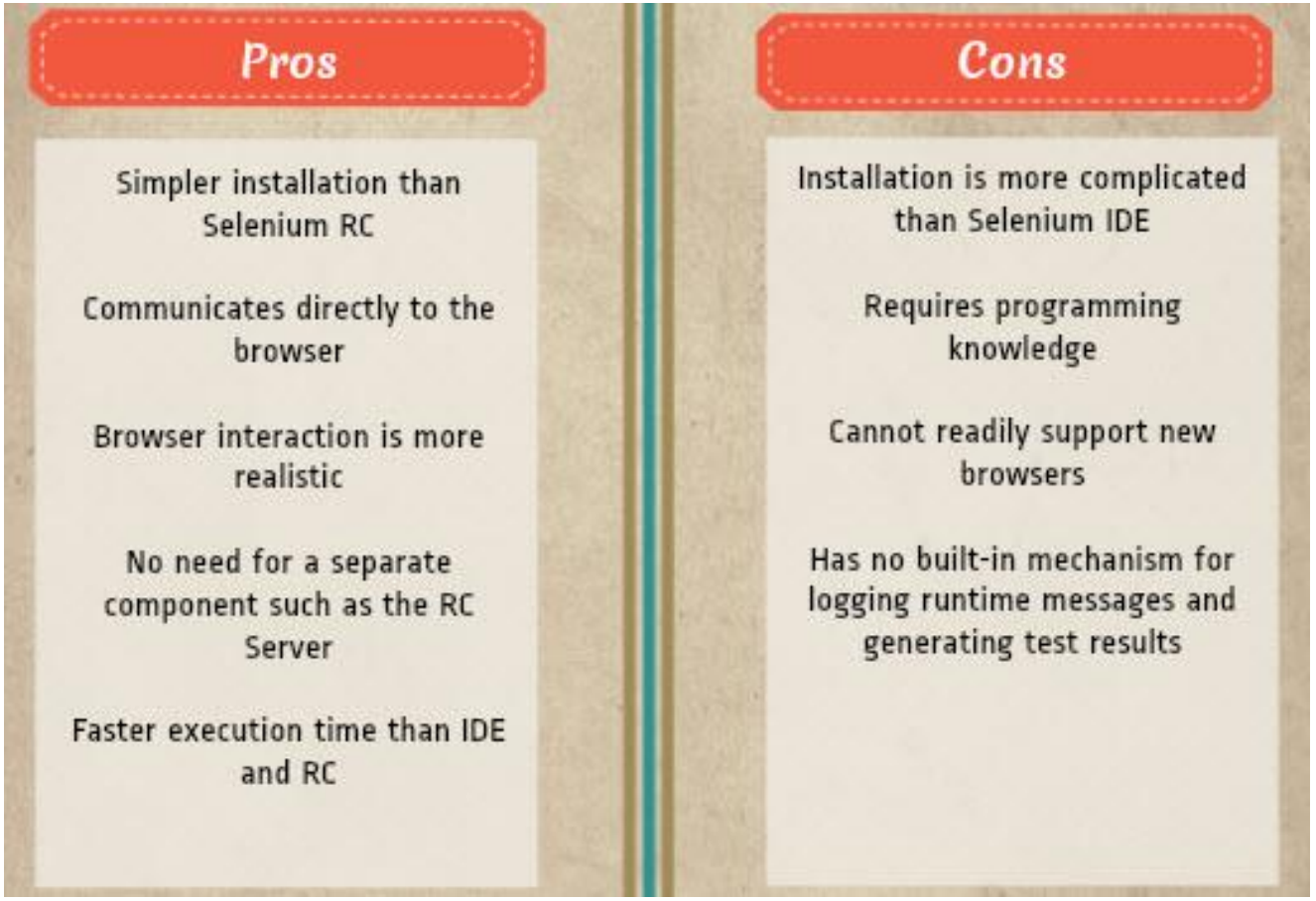
Selenium IDE

PROS	CONS
Very easy to use and install.	Available only in Firefox.
No programming experience is required, though knowledge of HTML and DOM are needed.	Designed only to create prototypes of tests.
Can export tests to formats usable in Selenium RC and WebDriver.	No support for iteration and conditional operations.
Has built-in help and test results reporting module.	Test execution is slow compared to that of Selenium RC and WebDriver.
Provides support for extensions.	

Selenium RC



Selenium WebDriver



Selenium Grid

Selenium Grid is a tool **used together with Selenium RC to run parallel tests** across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once.

Features:

1. Enables **simultaneous running of tests** in **multiple browsers and environments.**
2. **Saves time** enormously.
3. Utilizes the **hub-and-nodes** concept. The hub acts as a central source of Selenium commands to each node connected to it.

Comparision between Selenium and QTP

Selenium	QTP
Open source, **free to use**, and **free of charge.**	**Commercial**.
Highly extensible	Limited add-ons
Can run tests across **different browsers**	Can only run tests in **Firefox**, **Internet Explorer** and **Chrome**
Supports **various operating systems**	Can only be used in **Windows**
Supports **mobile devices**	QTP Supports Mobile app test automation (iOS & Android) using HP solution called - HP Mobile Center
Can execute tests **while** the **browser is minimized**	Needs to have the application under test to be visible on the desktop
Can execute tests **in parallel**.	Can only execute in parallel but using Quality Center which is again a paid product.

Selenium IDE installation

https://www.seleniumhq.org/download/

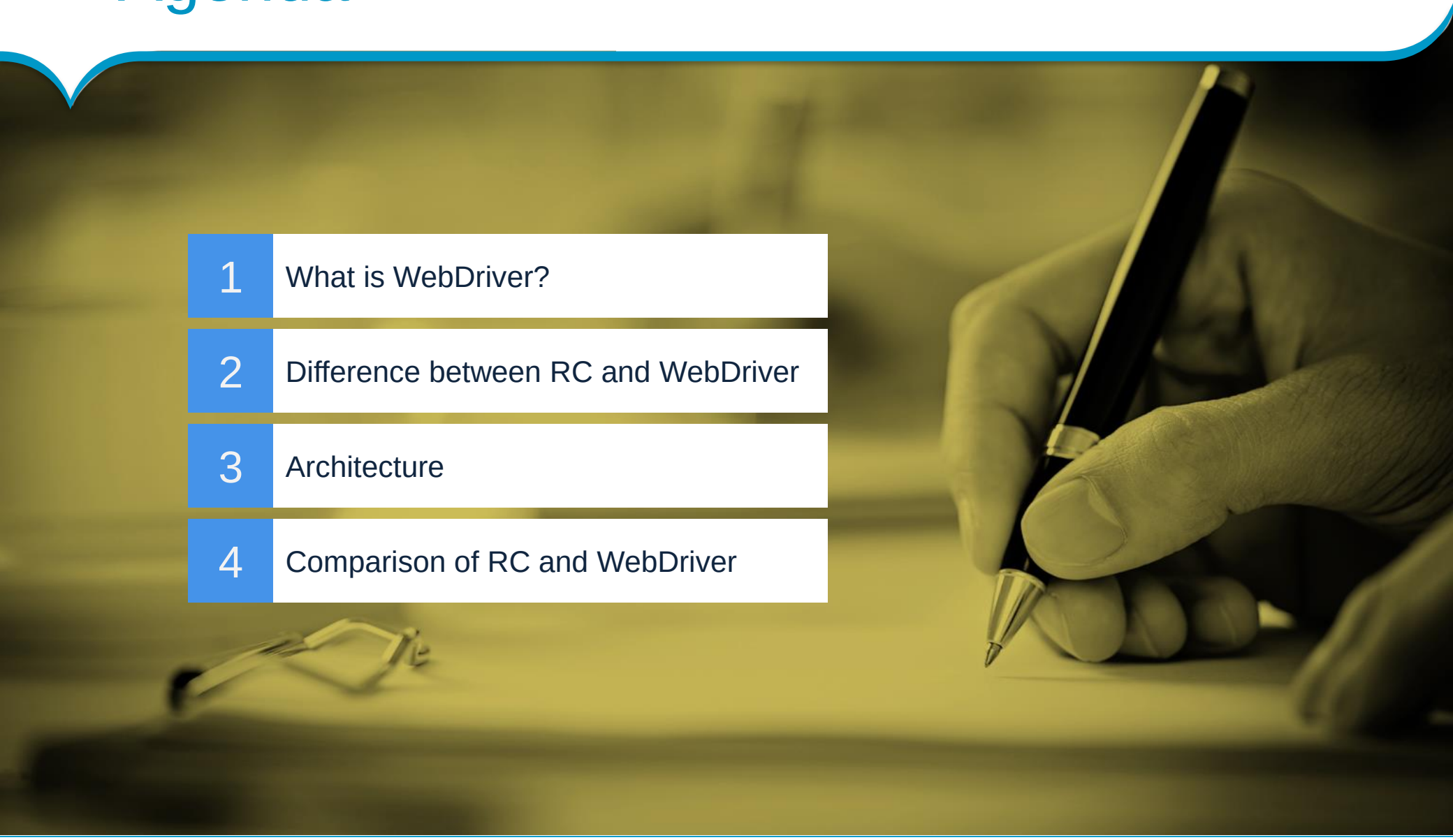
Download selenium IDE plugin for your browser

Summary

- What is Selenium?
- Who developed Selenium?
- Components
- Selenium Core
- Selenium RC
- Selenium IDE
- Selenium Web Driver
- Selenium Grid

Selenium IDE Lab

Agenda



1. What is WebDriver?
2. Difference between RC and WebDriver
3. Architecture
4. Comparison of RC and WebDriver

WebDriver

What is WebDriver

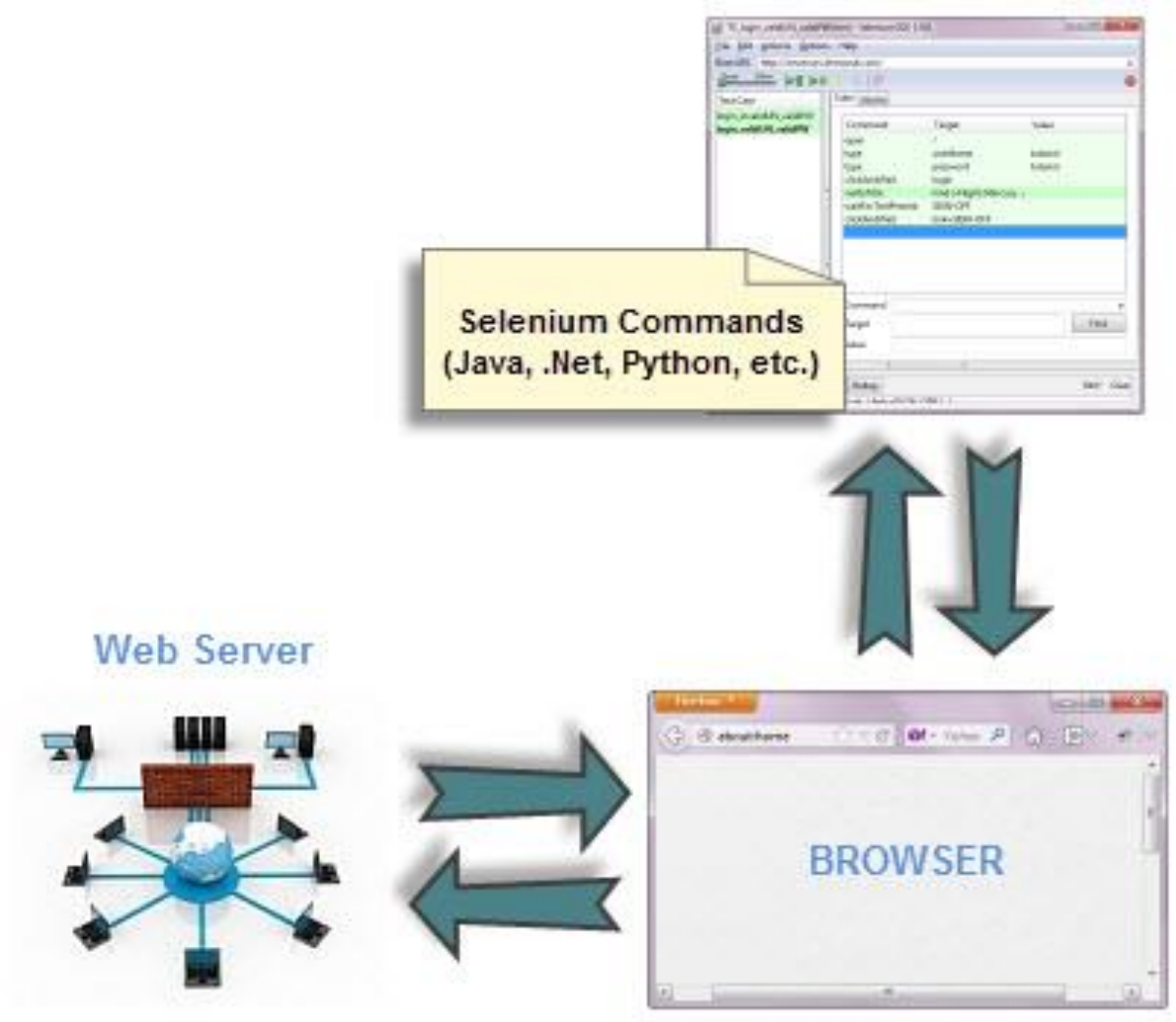
- WebDriver is a web automation framework that allows you to **execute your tests against different browsers.**
- Use a **programming language** in creating your test scripts
  - conditional operations like if-then-else or switch-case
  - perform looping like do-while.
- Languages supported:
  - Java
  - .Net
  - PHP
  - Python
  - Perl
  - Ruby

What is WebDriver

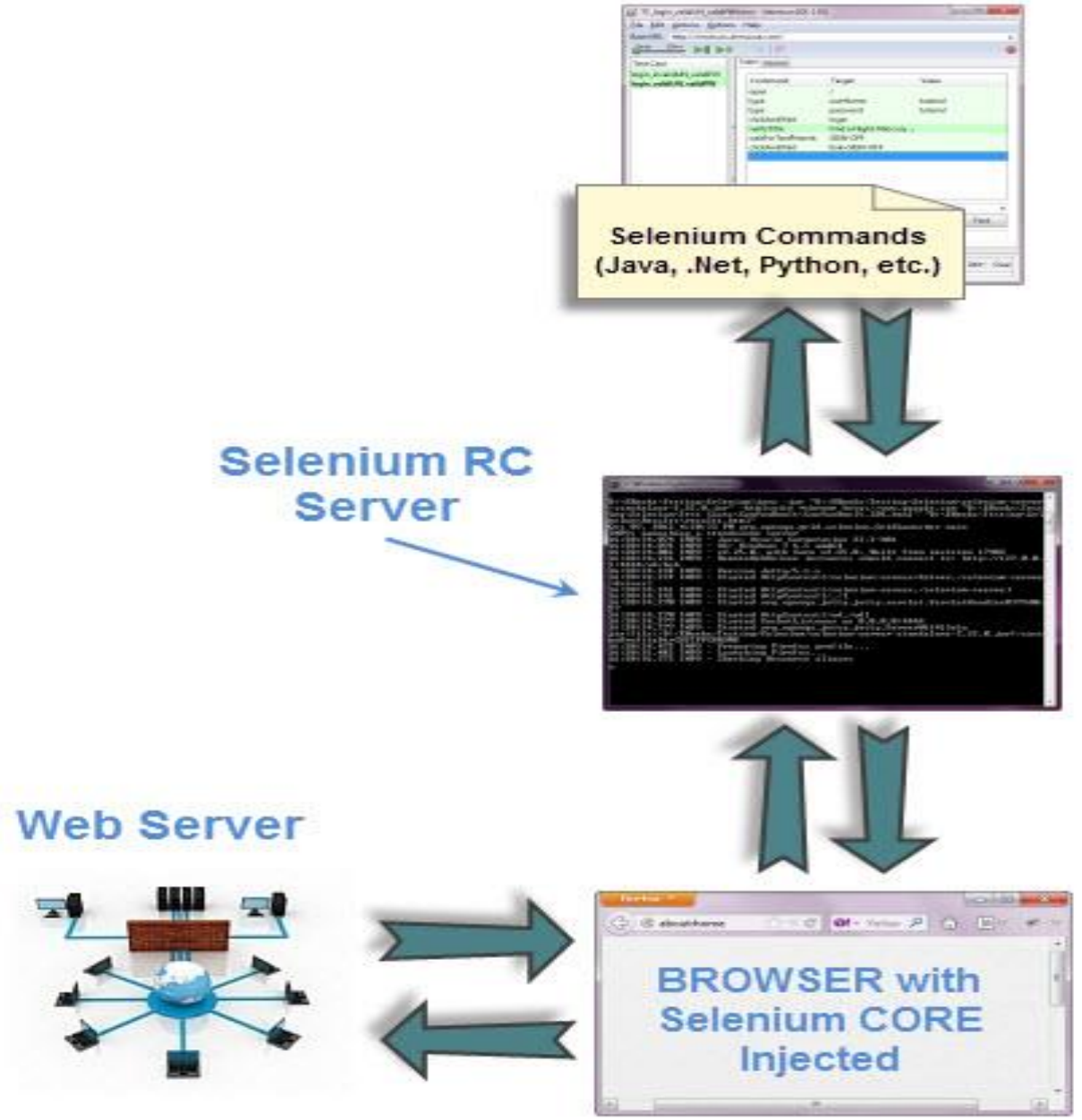
Difference between Selenium RC and WebDriver

- Both WebDriver and Selenium RC have following features:
  - They both allow you to **use a programming language** in designing your test scripts.
  - They both allow you to **run your tests against different browsers.**

Architecture - WebDriver



Architecture - RC



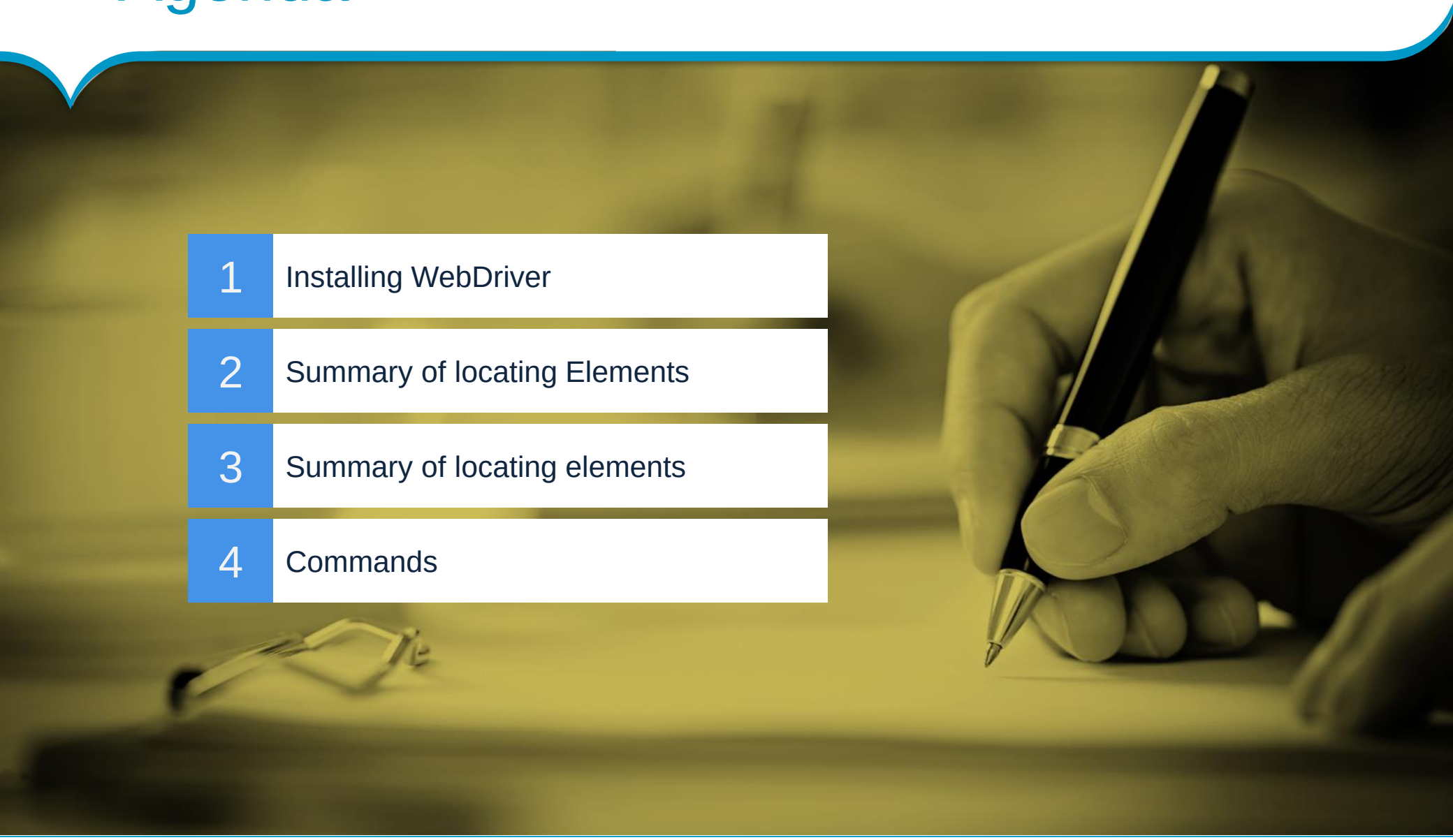
Comparison of RC and WebDriver

- Speed:
  - WebDriver is faster than Selenium RC.
  - Selenium RC is slower since it uses a Javascript program called Selenium Core.
- Real life interaction
  - WebDriver interacts with page elements in a more realistic way.
  - Selenium Core, just like other JavaScript codes, can access disabled elements.
- API
  - Selenium RC's API is more matured but contains redundancies and often confusing commands.
  - WebDriver's API is simpler than Selenium RC's. It does not contain redundant and confusing commands.
- Browser Support
  - WebDriver can support the headless HtmlUnit browser
  - Selenium RC cannot support the headless HtmlUnit browser.

Summary

- RC and WebDriver
- Comparison
- Architecture
- More understanding

Agenda



1. Installing WebDriver
2. Summary of locating Elements
3. Summary of locating elements
4. Commands

Installing WebDriver

Pre-Requisites

- Install JDK
- Install Eclipse /Spring Tool Suite (STS)
- Selenium Java Client

- Location
  - **Java Development Kit (JDK).**http://www.oracle.com/technetwork/java/javase/downloads/index.html
  - **Eclipse IDE** - http://www.eclipse.org/downloads/
  - **Java Client Driver** - http://seleniumhq.org/download/

Summary of Locating Elements

Variation	Description	Sample
By.**className**	finds elements based on the value of the "class" attribute	findElement(By.className("someClassName"))
By.**cssSelector**	finds elements based on the driver's underlying CSS Selector engine	findElement(By.cssSelector("input #email"))
By.**id**	locates elements by the value of their "id" attribute	findElement(By.id("someId"))
By.**linkText**	finds a link element by the exact text it displays	findElement(By.linkText("REGISTRATION"))
By.**name**	locates elements by the value of the "name" attribute	findElement(By.name("someName"))
By.**partialLinkText**	locates elements that contain the given link text	findElement(By.partialLinkText("REG"))
By.**tagName**	locates elements by their tag name	findElement(By.tagName("div"))
By.**xpath**	locates elements via XPath	findElement(By.xpath("//html/body/div/table/tbody/tr/td[2]/table/ tbody/tr[4]/td/table/tbody/tr/td[2]/table/tbo

Get Commands

get() *Sample usage:*	•It automatically opens a new browser window and fetches the page that you specify inside its parentheses. •It is the counterpart of Selenium IDE's "open" command. •The parameter must be a **String** object.
getTitle() *Sample usage:*	•Needs no parameters •Fetches the title of the current page •Leading and trailing white spaces are trimmed •Returns a null string if the page has no title
getPageSource()*Sample usage:*	•Needs no parameters •Returns the **source code of the page** as a String value
getCurrentUrl()*Sample usage:*	•Needs no parameters •Fetches the string representing the **current URL** that the browser is looking at
getText() *Sample usage:*	•Fetches the **inner text** of the element that you specify

Navigate Commands

navigate().to()*Sample usage:*	•It automatically **opens a new browser window and fetches the page** that you specify inside its parentheses. •**It does exactly the same thing as the get() method.**
navigate().refresh()*Sample usage:*	•Needs no parameters. •It **refreshes** the current page.
navigate().back()*Sample usage:*	•Needs no parameters •Takes you **back by one page** on the browser's history.
navigate().forward()*Sample usage:*	•Needs no parameters •Takes you **forward by one page** on the browser's history.

Closing and Quitting Browser Window Commands

close() *Sample usage:*	•Needs no parameters **•It closes only the browser window that WebDriver is currently controlling**.
quit() *Sample usage:*	•Needs no parameters **•It closes all windows that WebDriver has opened.**

Other Commands

- **Switching Between Frames**

```
WebDriver driver = new FirefoxDriver();
                driver.get("http://demo.guru99.com/selenium/deprecated.html");
                driver.switchTo().frame("classFrame");
                driver.findElement(By.linkText("Deprecated")).click();
                driver.close();
```

- **Switching Between Pop-up Windows**

```
driver.get("http://jsbin.com/usidix/1");
        driver.findElement(By.cssSelector("input[value=\"Go!\"]")).click();
        alertMessage = driver.switchTo().alert().getText();
        driver.switchTo().alert().accept();
```

Wait Command

- Implicit wait - used to set the default waiting time throughout the program
- Explicit wait - used to set the waiting time for a particular instance only
- **Implicit Wait**
  - It is simpler to code than Explicit Waits.
  - It is usually declared in the instantiation part of the code.
  - You will only need one additional package to import.
- **Explicit Wait**
  - **Explicit waits are done using the WebDriverWait and ExpectedCondition classes**. For the following example, we shall wait up to 10 seconds for an element whose id is "username" to become visible before proceeding to the next command.

Wait Command

```
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

this means that you are setting 10 seconds as your default wait time. You can change "10" and "SECONDS" to any number and time unit you want.

WebDriver instance that will use the explicit wait

```
WebDriver driver = new FirefoxDriver();
WebDriverWait myWaitVar = new WebDriverWait(driver, 10);
```

number of seconds to wait

Other Command

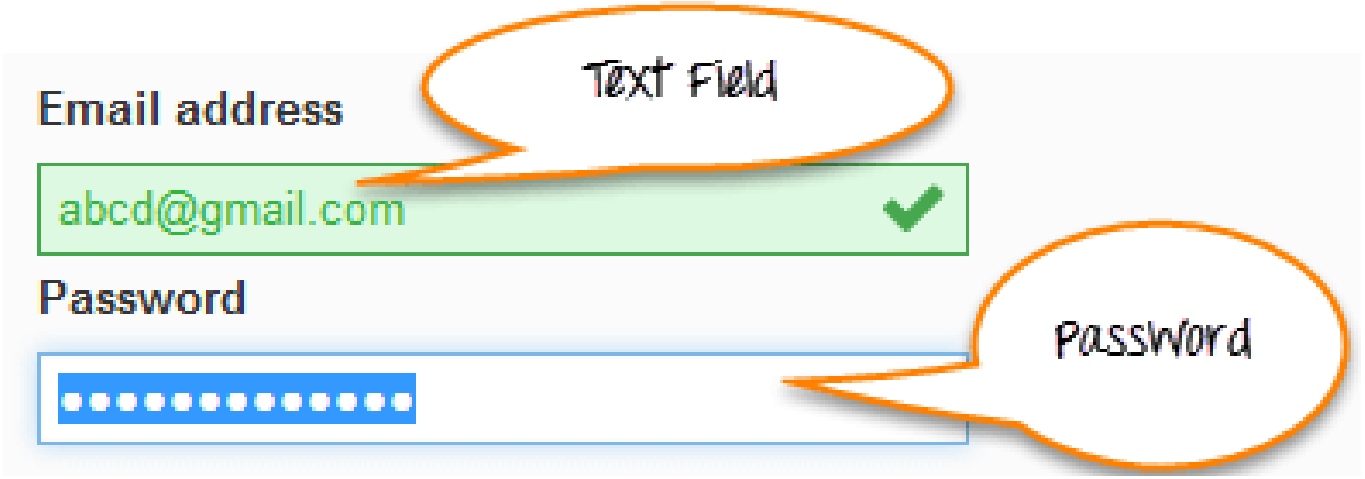
```
WebDriverWait myWaitVar = new WebDriverWait(driver, 3);
try {
    myWaitVar.until(ExpectedConditions.visibilityOfElementLocated(By
            .id("username")));
    driver.findElement(By.id("username")).sendKeys("tutorial");
} catch (TimeoutException toe) {
    System.out.println(toe.toString());
}
```

- Conditions
  - isEnabled()
  - isDisplayed()
  - isSelected()
- Using ExpectedConditions
  - alertIsPresent()
  - elementToBeClickable()
  - frameToBeAvailableAndSwitchToIt()
- Catching Exceptions

Accessing Form Elements

Accessing Form Elements

```
import org.openqa.selenium.WebElement;
```



Accessing Form Elements

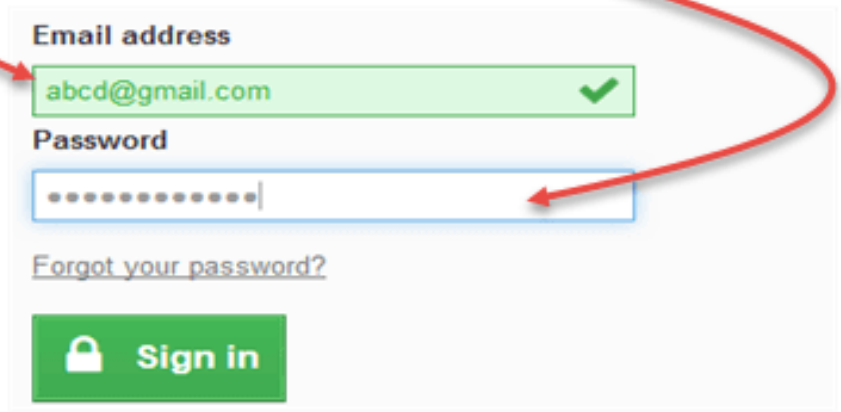
```
// Get the WebElement corresponding to the Email Address(TextField)
WebElement email = driver.findElement(By.id("email"));

// Retrieve the Webelement corresponding to the Password Field
WebElement password = driver.findElement(By.name("passwd"));
```

1) email text field is located by ID

2) Password field is located by name

Accessing Form Elements

```
// Get the WebElement corresponding to the Email Address(TextField)
WebElement email = driver.findElement(By.id("email"));  1

// Retrieve the WebElement corresponding to the Password Field
WebElement password = driver.findElement(By.name("passwd"));  2

email.sendKeys("abcd@gmail.com");  3

password.sendKeys("abcdefghlkjl");  4
```



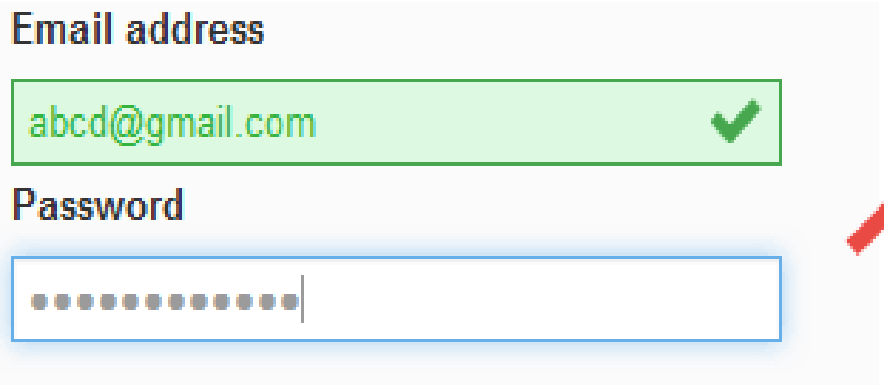
1) Find the "Email Address" Text Field using id locator

2) Find the "Password" Field using name locator

3) Enter text into the "Email Address"

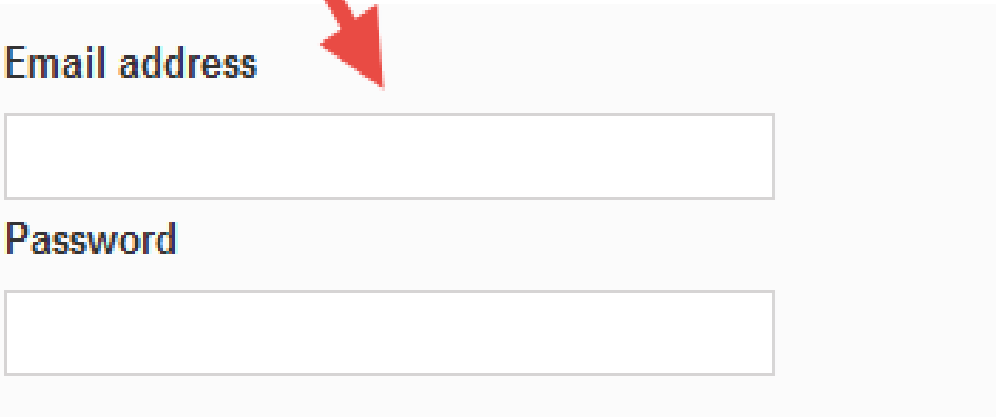
4) Enter password into the "Password" using sendKeys()

Accessing Form Elements - Deleting Values in Input Boxes




```
email.clear();

password.clear();
```

after clear()

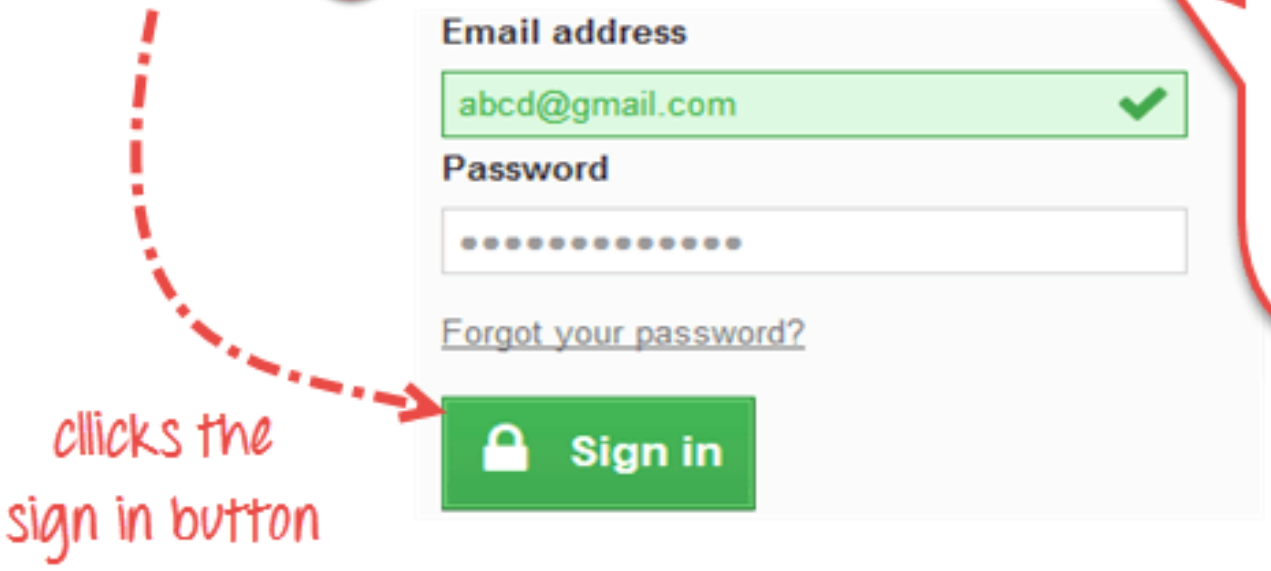


Accessing Form Elements - Buttons

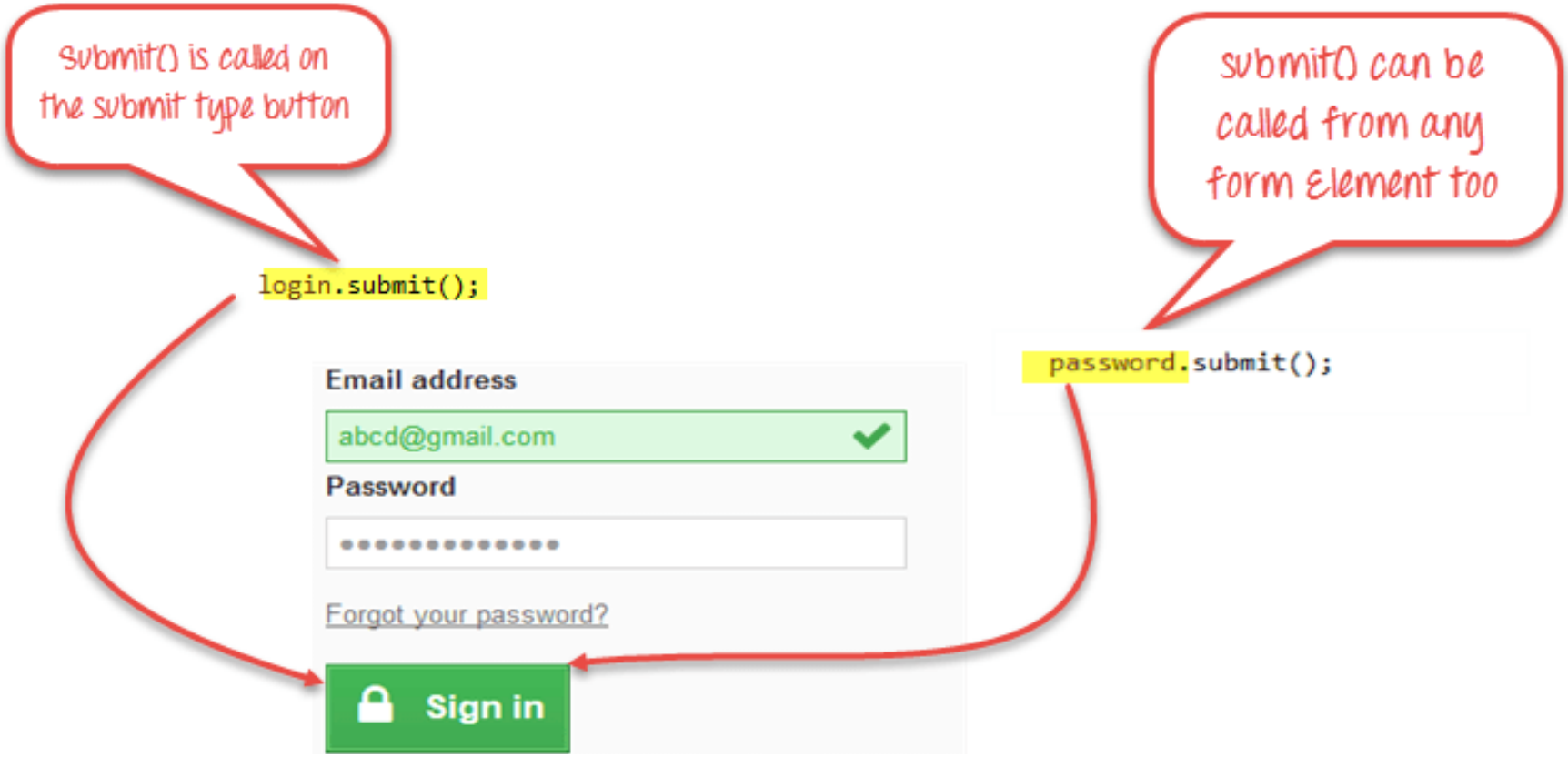
```
WebElement login = driver.findElement(By.id("SubmitLogin"));
login.click();
```

1) Find the "Sign-in" button located by ID

2) click() on the button element clicks the button

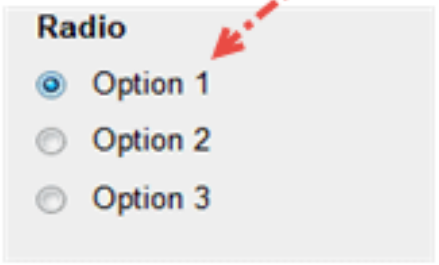


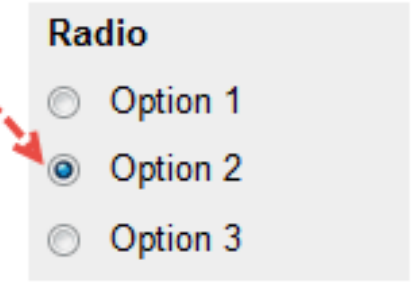
Accessing Form Elements – Submit Buttons



Accessing Form Elements – Radio Buttons

```
WebElement radio1 = driver.findElement(By.id("vfb-7-1"));
WebElement radio2 = driver.findElement(By.id("vfb-7-2"));

//Radio Button1 is selected
radio1.click();

//Radio Button1 is de-selected and Radio button2 is selected
radio2.click();
```

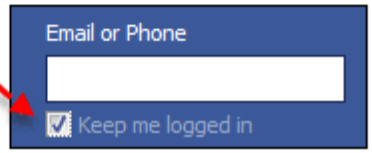




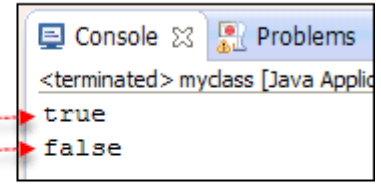
click() is the method to toggle on the radio buttons

Accessing Form Elements - Checkbox

```
public static void main(String[] args) {
    WebDriver driver = new FirefoxDriver();
    String baseURL = "http://www.facebook.com";

    driver.get(baseURL);
    WebElement chkFBPersist = driver.findElement(By.id("persist_box"));
    for(int i=0; i<2; i++){
        chkFBPersist.click();
        System.out.println(chkFBPersist.isSelected());
    }
    driver.quit();
}
```



First click - checkbox was toggled on



Second click - checkbox was toggled off

Accessing Form Elements - Checkbox

```
@Test
public void tryCheckbox(){

    WebElement option1 = driver.findElement(By.id("vfb-6-0"));  1

    //This will Toggle On the Check box
    option1.click();  2

    //Check whether the Check box is toggled on
    if(option1.isSelected()){  3
        System.out.println("Checkbox is Toggled On");

    }else{
        System.out.println("Checkbox is Toggled Off");
    }

    // This should Toggle Off the Check box
    option1.click();  4

    // Lets see whether its Toggled Off
    if(!option1.isSelected()){
        System.out.println("Checkbox is now Toggled Off !!");
    }


}
```

1. Locate the checkbox element by its ID

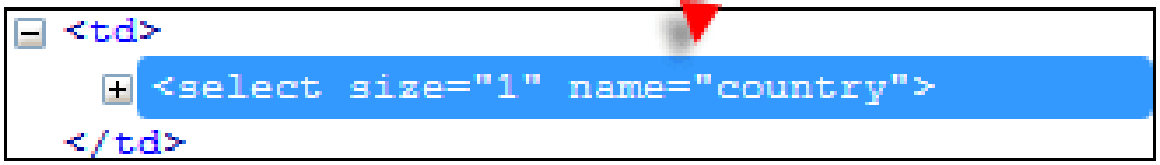
2. click() toggles on the checkbox

3. isSelected() gives the Toggle status of the checkbox

4. click() again on the checkbox turns the Toggle off

Accessing Form Elements – Drop Downbox

Mercury Tours Registration Page



Firebug

```
import org.openqa.selenium.support.ui.Select;
```

```
Select drpCountry = new Select(driver.findElement(By.name("country")));
```

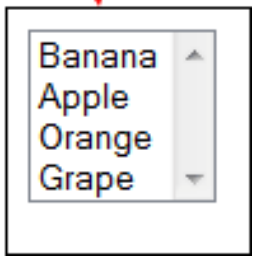
Accessing Form Elements - Drop Downbox

```
drpCountry.selectByVisibleText("ANTARCTICA");
```

page source

```
<select id="fruits" multiple="">
    <option value="banana">Banana</option>
    <option value="apple">Apple</option>
    <option value="orange">Orange</option>
    <option value="grape">Grape</option>
</select>
```

HTML page



Accessing Form Elements - Drop Downbox

```
public static void main(String[] args) {

    WebDriver driver = new FirefoxDriver();

    driver.get("http://jsbin.com/osebed/2");
    Select fruits = new Select(driver.findElement(By.id("fruits")));
    fruits.selectByVisibleText("Banana");
    fruits.selectByIndex(1);
}
```

Accessing Form Elements - Drop Downbox

Method	Description
selectByVisibleText() and**deselectByVisibleText()** *Example:* `drpCountry.selectByVisibleText("ANTARCTICA");`	•Selects/deselects the option that displays the text matching the parameter. •**Parameter**: The exactly displayed text of a particular option

selectByValue() and**deselectByValue()** *Example:*
`drpCountry.selectByValue("234");`
	•Selects/deselects the option whose "value" attribute matches the specified parameter.
	•**Parameter**: value of the "value" attribute
	•Remember that not all drop-down options have the same text and "value", like in the example below.
	`<option value="10">ANGUILLA </option>`
	`<option value="234">ANTARCTICA </option>`
	`<option value="1">ANTIGUA AND BARBUDA </opti`

selectByIndex() and**deselectByIndex()** *Example:* `drpCountry.selectByIndex(0);`	•Selects/deselects the option at the given index. •**Parameter**: the index of the option to be selected.

isMultiple() *Example:*
`if (drpCountry.isMultiple()) {`
`  //do something here`
`}`
	•Returns TRUE if the drop-down element allows multiple selections at a time; FALSE if otherwise.
	•**No parameters needed**

deselectAll() *Example:* `drpCountry.deselectAll();`	•Clears all selected entries. This is only valid when the drop-down element supports multiple selections. •**No parameters needed**

Accessing Links and WebTables

Accessing Links

```
<html>
    <head>
        <title>Sample</title>
    </head>
    <body>
        <a href="http://www.google.com">click here</a>
        <br>
        <a href="http://www.fb.com">click here</a>
    </body>
</html>
```

Accessing Links

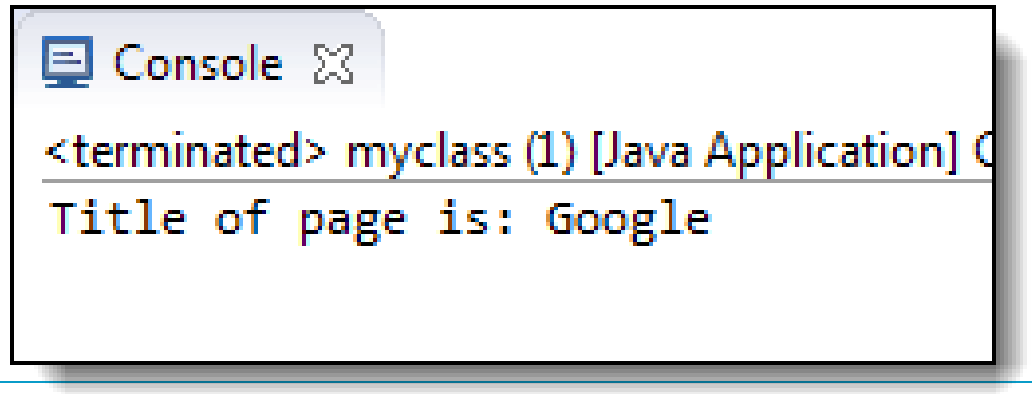
```
public static void main(String[] args) {
    String baseUrl = "file:///D:/newhtml.html";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    driver.findElement(By.linkText("click here")).click();
    System.out.println("Title of page is: " + driver.getTitle());
    driver.quit();
}
```

Accessing Links – How it works

```
driver.findElement(By.linkText("Create a new account")).click();
```

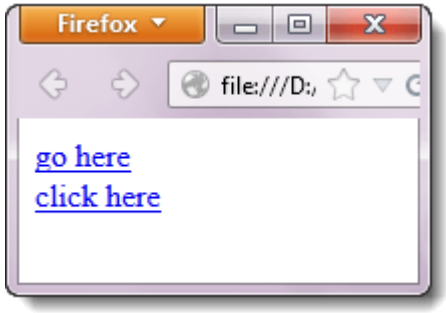
findElement() is used to find Links in the page

click() is the method to access links



Accessing Links – Partial Match

```
<html>
    <head>
        <title>Partial Match</title>
    </head>
    <body>
        <a href="http://www.google.com">go here</a>
        <br>
        <a href="http://www.fb.com">click here</a>
    </body>
</html>
```

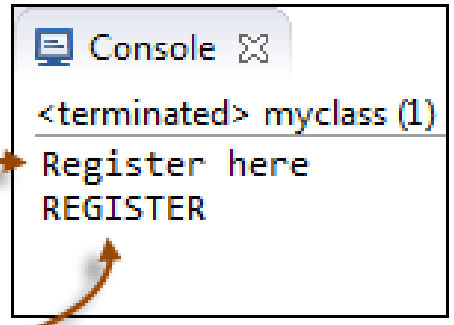


Accessing Links

```
public static void main(String[] args) {
    String baseUrl = "file:///D:/partial_match.html";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    driver.findElement(By.partialLinkText("here")).click();
    System.out.println("Title of page is: " + driver.getTitle());
    driver.quit();
}
```

Accessing Links

```
public static void main(String[] args) {
    String baseUrl = "http://newtours.demoaut.com/";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);

    String theLinkText = driver.findElement(By
            .partialLinkText("egis"))
            .getText();
    System.out.println(theLinkText);
    theLinkText = driver.findElement(By
            .partialLinkText("EGIS"))
            .getText();
    System.out.println(theLinkText);

    driver.quit();
}
```



Accessing Links – All links

```
List<WebElement> allLinks = driver.findElements(By.tagName("a"));

for(WebElement link:allLinks){
    // filter the links with the required text
    if(link.getText().equals("Create a new account")){
        System.out.println(" Link : "+ link.getText());
```

Returns all the links in the web page

Loops through all the links in the page.

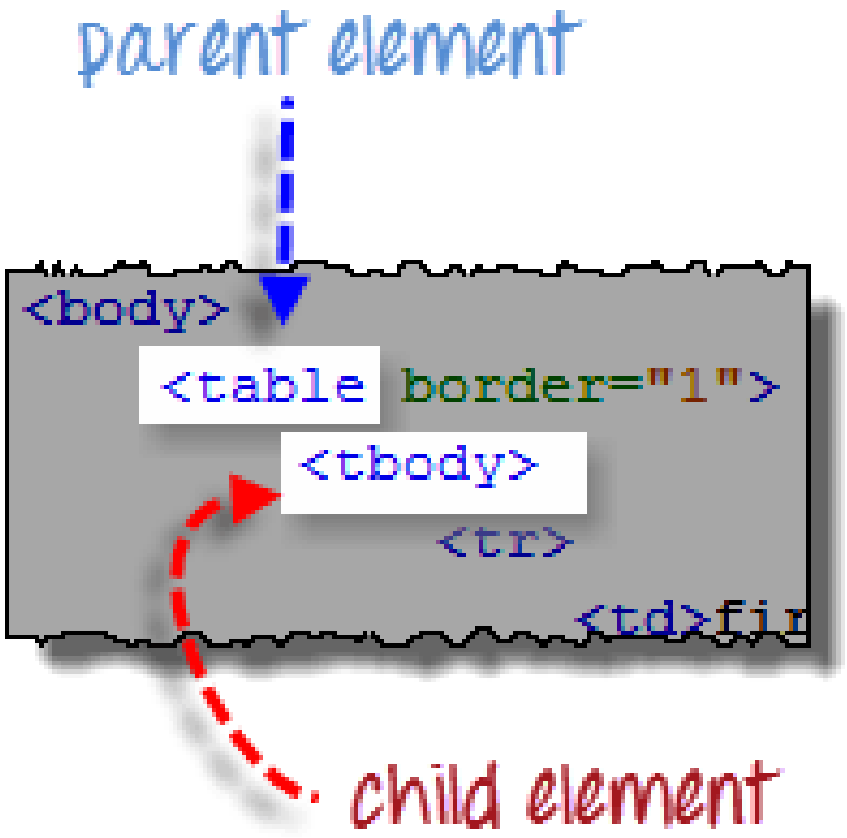
Accessing Links – Image links

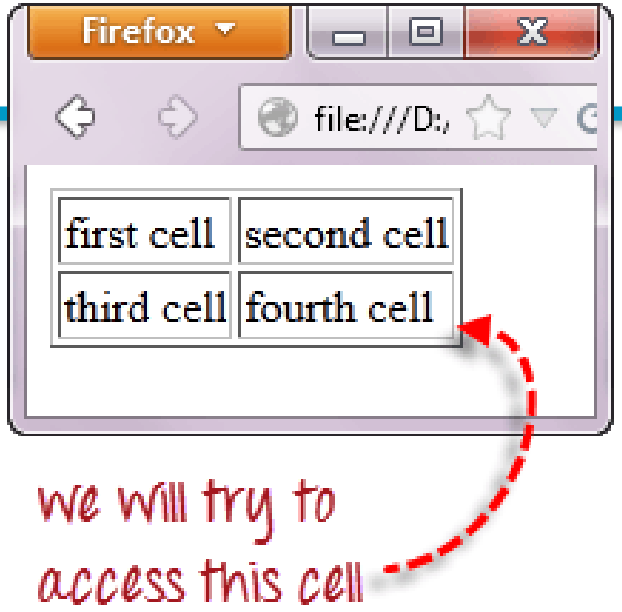
Reading a Table

- using the "By.xpath()" method

```
<html>
    <head>
        <title>Sample</title>
    </head>
    <body>
        <table border="1">
            <tbody>
                <tr>
                    <td>first cell</td>
                    <td>second cell</td>
                </tr>
                <tr>
                    <td>third cell</td>
                    <td>fourth cell</td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```

Reading a Table – Xpath syntax





Reading a Table – Xpath syntax

```
1  //table/tbody/tr[2]
```

The [2] predicate denotes that we are accessing the 2nd <tr> of the parent <tbody>

```
//table/tbody/tr
```

this will automatically access the first <tr> because no predicate was used

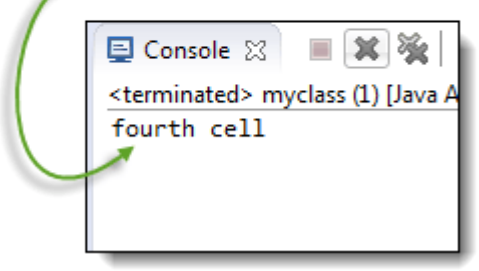
```
//table/tbody/tr[1]
```

this will access the first <tr> because the predicate **[1]** explicity says it

Reading a Table – Xpath syntax

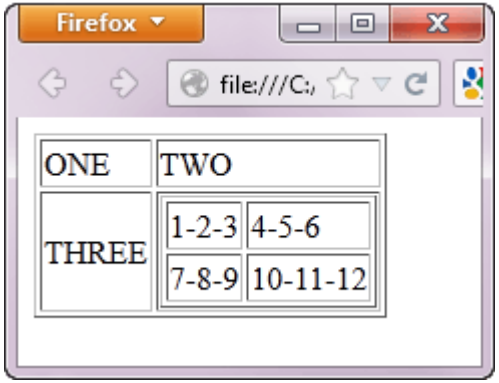
```
    public static void main(String[] args) {
        String baseUrl = "file:///C:/newhtml.html";
        WebDriver driver = new FirefoxDriver();

        driver.get(baseUrl);
        String innerText = driver.findElement(
                By.xpath("//table/tbody/tr[2]/td[2]")).getText();
        System.out.println(innerText);
        driver.quit();
    }
}
```

the inner text was successfully retrieved using xpath

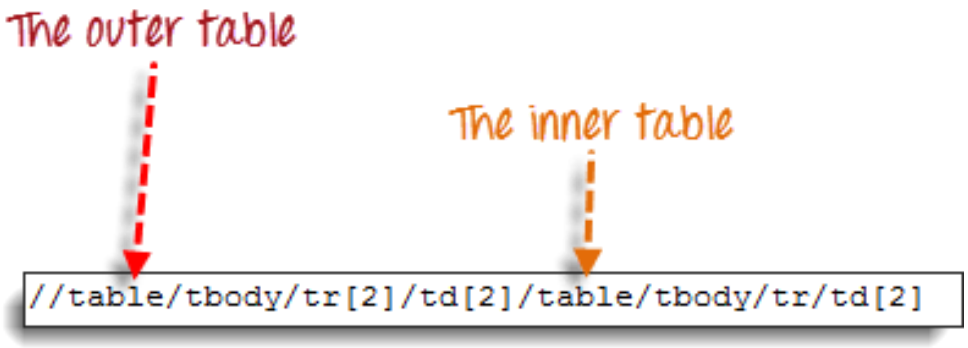


Reading a Nested Table – Xpath syntax

```
<html>
    <head>
        <title>Sample</title>
    </head>
    <body>
        <!--outer table-->
        <table border="1">
            <tbody>
                <tr>
                    <td>ONE</td>
                    <td>TWO</td>
                </tr>
                <tr>
                    <td>THREE</td>
                    <td>
                        <!--inner table-->
                        <table border="1">
                            <tbody>
                                <tr>
                                    <td>1-2-3</td>
                                    <td>4-5-6</td>
                                </tr>
                                <tr>
                                    <td>7-8-9</td>
                                    <td>10-11-12</td>
                                </tr>
                            </tbody>
                        </table>
                    </td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```

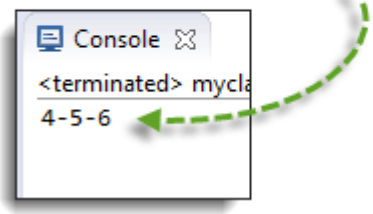


Reading a Nested Table – Xpath syntax




```
public static void main(String[] args) {
    String baseUrl = "file:///C:/newhtml.html";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    String innerText = driver.findElement(By
            .xpath("//table/tbody/tr[2]/td[2]/table/tbody/tr/td[2]"))
            .getText();
    System.out.println(innerText);
    driver.quit();
}
```

The inner text was successfully retrieved



Reading a Nested Table – Using Attributes as Predicates

```
<body>
  <div>
    <table height="100%" cellspacing="0" cellpadding="0" border="0">
      <tbody>
        <tr>
          <td valign="top" bgcolor="#003366">
          <td valign="top">
            <table cellspacing="0" cellpadding="0" border="0">
              <tbody>
                <tr>
                <tr>
                <tr>
                <tr>
                  <td>
                    <table cellspacing="0" cellpadding="0" border="0">
                      <tbody>
                        <tr>
                          <td width="14"> </td>
                          <td>
                            <table width="492" cellspacing="0" cellpadding="0" borde
                              <tbody>
                                <tr> </tr>
                                <tr>
                                  <td width="273" valign="top">
                                    <p>
                                    <table width="100%" cellspacing="0" cellpa
                                      <tbody>
                                        <tr>
                                        <tr>
                                        <tr valign="top">
                                          <td height="101">
                                            <table width="270" cellspac
                                              <tbody>
                                                <tr bgcolor="#CCCCCC
                                                  <td width="80%">
                                                    <font size="2
                                                    Vegas</font>
                                                  </td>
                                                  <td width="20%">
```

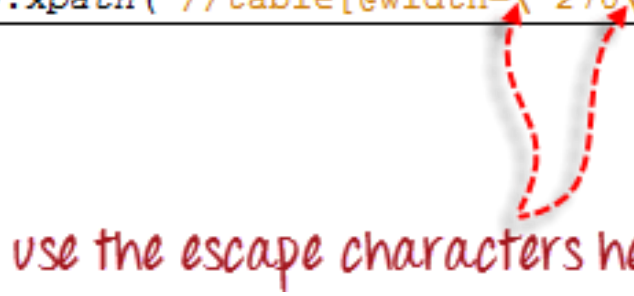
Specials	
Atlanta to Las Vegas	$398
Boston to San Francisco	$513
Los Angeles to Chicago	$168
New York to Chicago	$198
Phoenix to San Francisco	$213

this is the <table> that holds the New York to Chicago cell. Notice that it is burried deep into the HTML code, and the number to use as Predicate is difficult to determine.

Reading a Nested Table – Xpath syntax

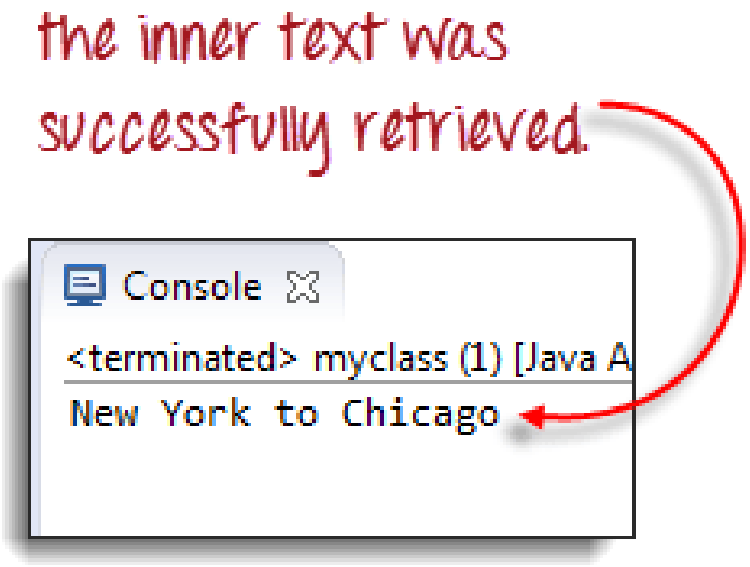
```
//table[@width="270"]/tbody/tr[4]/td
```

```
By.xpath("//table[@width=\"270\"]/tbody/tr[4]/td"))
```

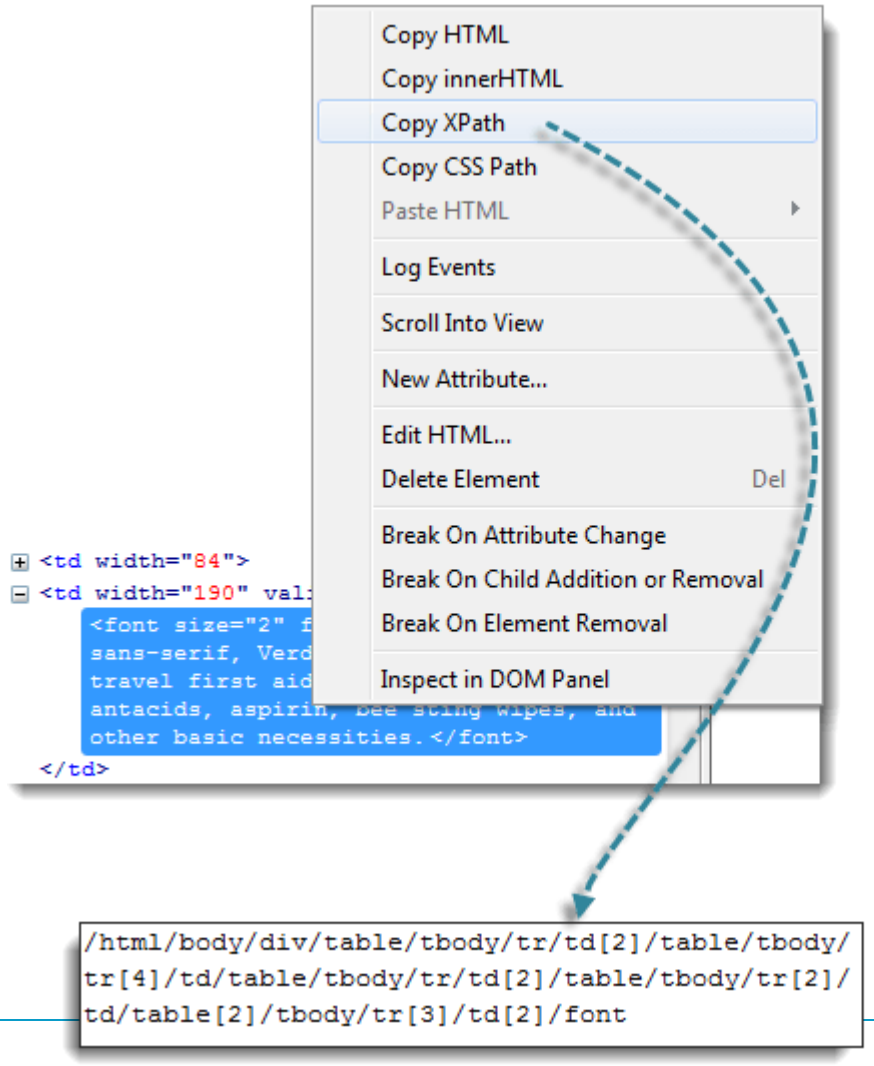



```
public static void main(String[] args) {
    String baseUrl = "http://newtours.demoaut.com/";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    String innerText = driver.findElement(By
            .xpath("//table[@width=\"270\"]/tbody/tr[4]/td"))
            .getText();
    System.out.println(innerText);
    driver.quit();
}
```

Reading a Nested Table – Xpath syntax



Reading a Nested Table – Xpath syntax – Shortcut in Firebug / Chrome



Reading a Nested Table – Xpath syntax – Shortcut in Firebug / Chrome

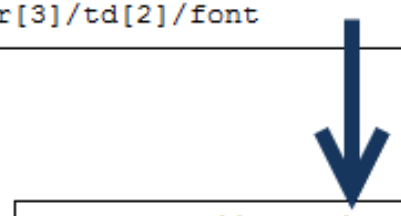
delete these

this is the first "parent" table element

```
/html/body/div/table/tbody/tr/td[2]/table/tbody/tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[2]/td/table[2]/tbody/tr[3]/td[2]/font
```

The remaining portion of the code, trimmed and prefixed with "//"

```
//table/tbody/tr/td[2]/table/tbody/tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[2]/td/table[2]/tbody/tr[3]/td[2]/font
```

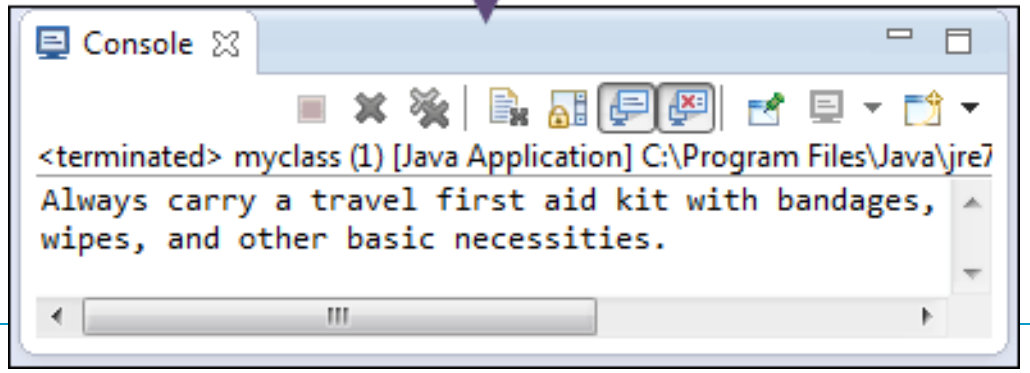


```
By.xpath("//table/tbody/tr/td[2]"
+ "/table/tbody/tr[4]/td"
+ "/table/tbody/tr/td[2]"
+ "/table/tbody/tr[2]/td"
+ "/table[2]/tbody/tr[3]/td[2]/font"))
```

When pasted onto the By.xpath() method

Reading a Nested Table – Xpath syntax – Shortcut in Firebug / Chrome

```
public static void main(String[] args) {
    String baseUrl = "http://newtours.demoaut.com/";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    String innerText = driver.findElement(
            By.xpath("//table/tbody/tr/td[2]"
            + "/table/tbody/tr[4]/td"
            + "/table/tbody/tr/td[2]"
            + "/table/tbody/tr[2]/td"
            + "/table[2]/tbody/tr[3]/td[2]/font"))
            .getText();
    System.out.println(innerText);
    driver.quit();
}
```



Further Reading

- Keyboard & Mouse Event using Action Class in Selenium Webdriver
- How to Upload & Download a File using Selenium Webdriver
- How to Install TestNG in Eclipse for Selenium WebDriver
- How TestNG makes Selenium tests easier

Agenda

1. What is POM?
2. Advantages of POM
3. How to implement POM?
4. Page Factory

Page Object Model

What is POM?

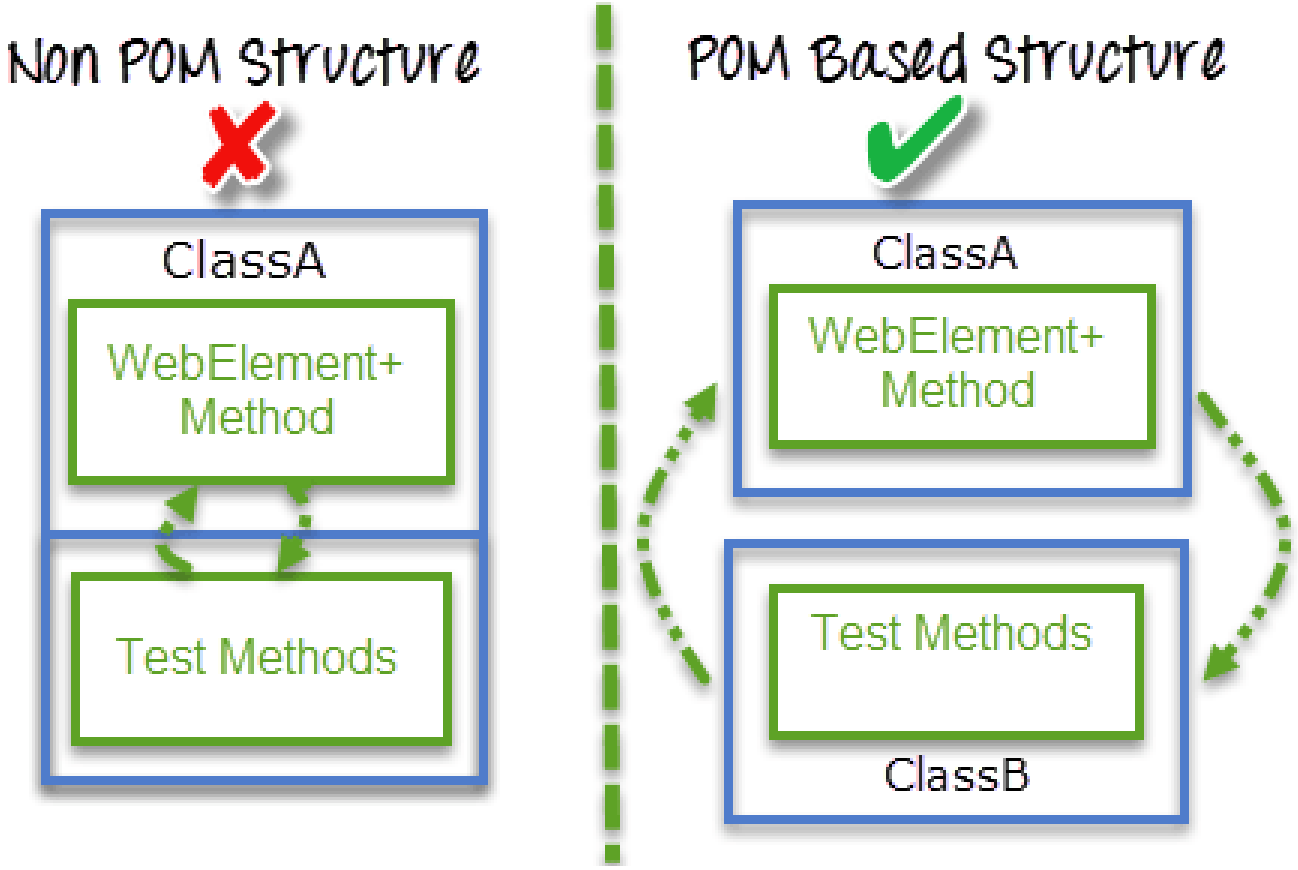
- Page Object Model (POM) & Page Factory in Selenium
- UI Automation in Selenium WebDriver is NOT a tough task
- Find elements and perform operations
- Create separate class file which would find web elements
- Fill them or verify them
- Class can be reused in all the scripts using that element.
- In future, if there is a change in the web element, we need to make the change in just 1 class file

What is POM?

```
/**
 * This test case will login in http://demo.guru99.com/V4/
 * Verify login page title as guru99 bank
 * Login to application
 * Verify the home page using Dashboard message
 */
@Test(priority=0)
public void test_Home_Page_Appear_Correct(){
    WebDriver driver = new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("http://demo.guru99.com/V4/");
    //Find user name and fill user name
    driver.findElement(By.name("uid")).sendKeys("demo");
    //find password and fill it
    driver.findElement(By.name("password")).sendKeys("password");
    //click login button
    driver.findElement(By.name("btnLogin")).click();
    String homeText = driver.findElement(By.xpath("//table//tr[@class='heading3']")).getText();
    //verify login success
    Assert.assertTrue(homeText.toLowerCase().contains("guru99 bank"));
}
}
```

1. Find user name and fill it
2. Find password and fill it
3. Find Login button and click it
4. Find home page text and get it
5. Verify home page has text 'Guru99 Bank'

What is POM?



What is POM?

- **Page Object Model** is a design pattern to create **Object Repository** for web UI elements.
- Under this model, for each web page in the application, there should be corresponding page class.
- This Page class will find the WebElements of that web page and also contains Page methods which perform operations on those WebElements.
- Name of these methods should be given as per the task they are performing, i.e., if a loader is waiting for the payment gateway to appear, POM method name can be waitForPaymentScreenDisplay().
-

Advantages of POM

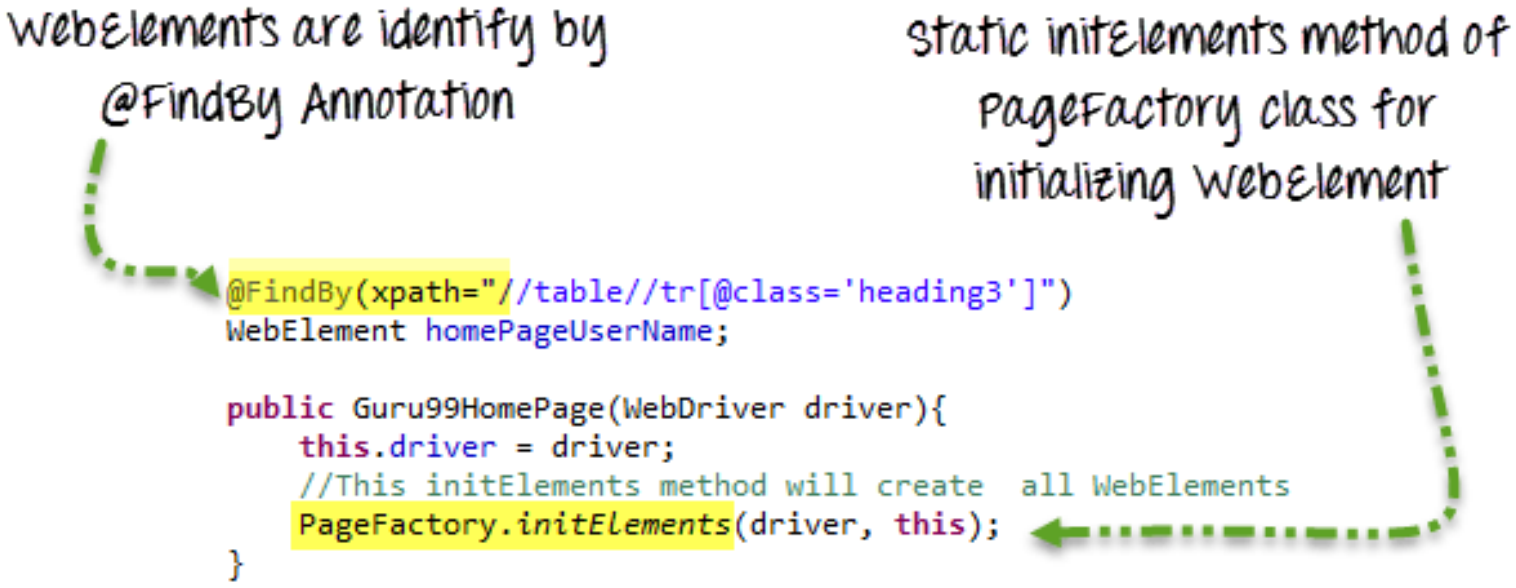
- Page Object Patten says operations and flows in the UI should be separated from verification. This concept makes our code cleaner and easy to understand.
- **Object repository is independent of test cases**, so we can use the same object repository for a different purpose with different tools. For example, we can integrate POM with TestNG/JUnit for functional Testing and at the same time with JBehave/Cucumber for acceptance testing.
- Code becomes less and optimized because of the reusable page methods in the POM classes.
- **Methods** get **more realistic names** which can be easily mapped with the operation happening in UI. i.e. if after clicking on the button we land on the home page, the method name will be like 'gotoHomePage()'.

How to implement?

- Create supporting PageClass.
- Add all fields with appropriate setter and getter
- Implement test cases
- Test it

Page Factory

- Page Factory is an inbuilt Page Object Model concept for Selenium WebDriver but it is very optimized.
- **@FindBy** can accept **tagName, partialLinkText, name, linkText, id, css, className, xpath** as attributes.



Page Factory - AjaxElementLocatorFactory

- One of the key advantages of using **Page Factory pattern** is AjaxElementLocatorFactory Class.
- Eorking on **lazy loading** concept.
- **Timeout for a WebElement** will be assigned to the Object page class with the help of AjaxElementLocatorFactory.
- Test Case execution will throw '**NoSuchElementException**' exception.

Page Factory - AjaxElementLocatorFactory

after 100 sec if element is not visible to perform an operation , timeout exception will appear

```
AjaxElementLocatorFactory factory = new AjaxElementLocatorFactory(driver, 100);
PageFactory.initElements(factory, this);
```

This is a lazy loading,wait will start only if we perform operation on control

THANK YOU

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

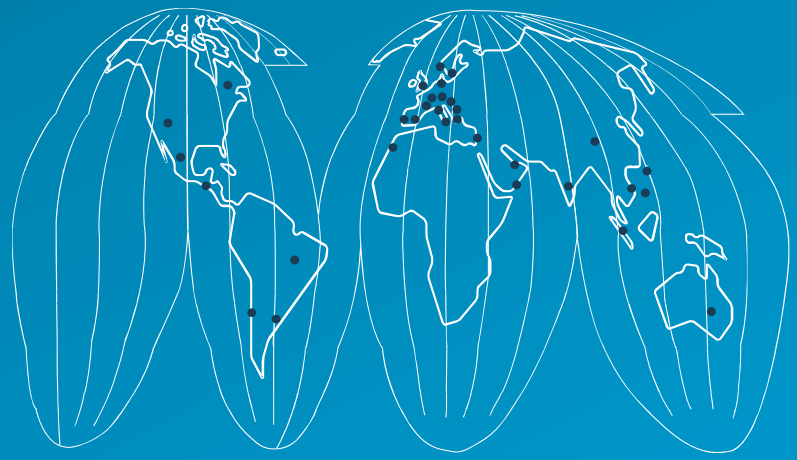
People matter, results count.

About Capgemini

With more than 145,000 people in 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2014 global revenues of EUR 10.5 billion.

Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Rightshore® is a trademark belonging to Capgemini



www.capgemini.com