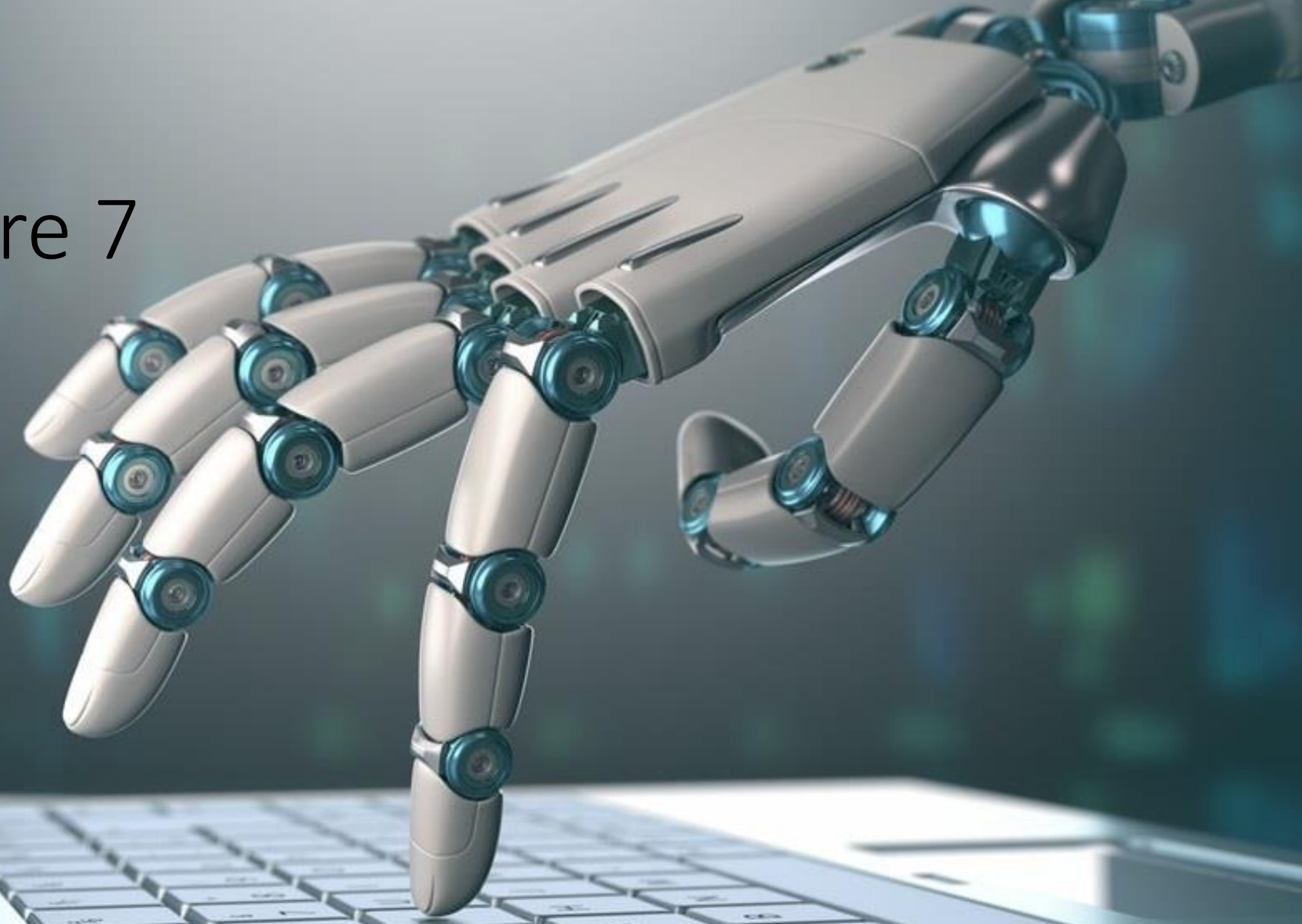
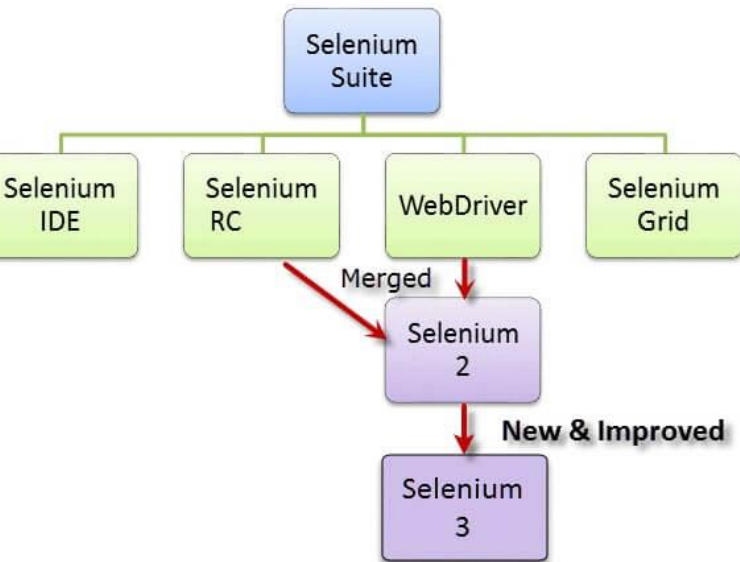


Lecture 7



Agenda

- **Selenium Web driver**
- Locators
- Xpath
- Css selector
- iframe
- Practice



- Selenium was **created by Jason Huggins in 2004**.
- Jason Huggins created a JavaScript program that would automatically control the browser's actions. He named this program as the "**JavaScriptTestRunner**."
- Selenium 1, refers to Selenium RC.
- Selenium RC and WebDriver are merged into a single framework to form **Selenium 2**.
- Google, who has been a long time user of Selenium, had a developer named Simon Stewart who developed WebDriver.
- This tool circumvented Selenium's JavaScript sandbox to allow it to communicate with the Browser and Operating System directly using native methods
- In 2008, Selenium and WebDriver merged technologies and intellectual intelligence to provide the best possible test automation framework
- The latest version of Selenium is 3.0.

Selenium WebDriver

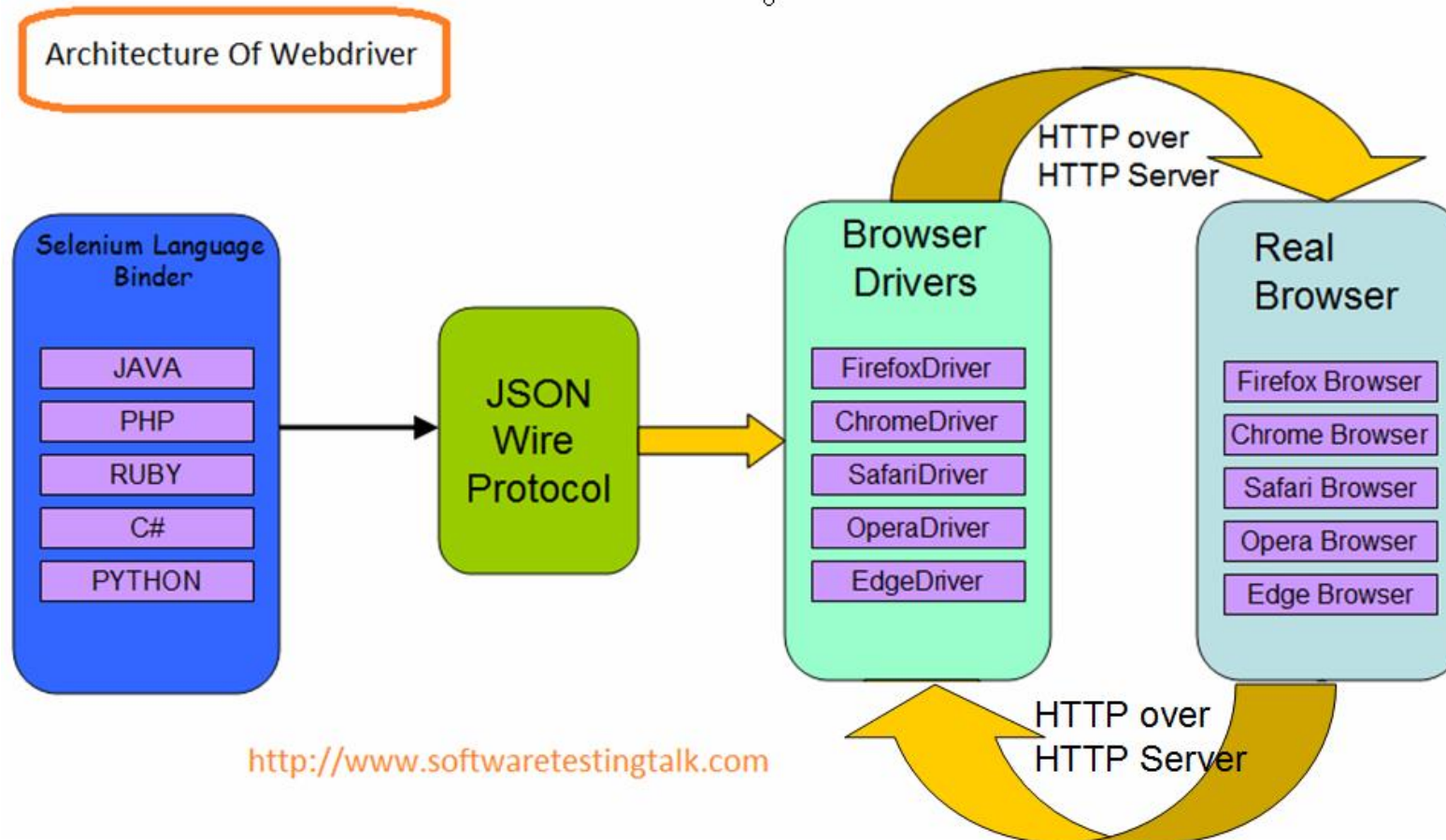
If you want to

- create robust, browser-based regression automation suites and tests
- scale and distribute scripts across many environments

Selenium-WebDriver makes direct calls to the browser using each browser's native support for automation.



Selenium WebDriver



Where to get it?

Third Party Browser Drivers **NOT DEVELOPED** by seleniumhq

| Browser | | change log | issue tracker | Implementation Status |
|---|-----------------------------|----------------------------|-------------------------------|---------------------------------------|
| Mozilla GeckoDriver | latest | change log | issue tracker | selenium wiki page |
| Google Chrome Driver | latest | change log | issue tracker | selenium wiki page |
| Opera | latest | | issue tracker | selenium wiki page |
| Microsoft Edge Driver | | | issue tracker | Implementation Status |
| GhostDriver | (PhantomJS) | | issue tracker | SeConf talk |
| HtmlUnitDriver | latest | | issue tracker | |
| SafariDriver | | | issue tracker | |
| Windows Phone | | | issue tracker | |
| Windows Phone | 4.14.028.10 | | issue tracker | Released 2013-11-23 |
| Selendroid - Selenium for Android | | | issue tracker | |
| ios-driver | | | issue tracker | |
| BlackBerry_10 | | | issue tracker | Released 2014-01-28 |
| Appium | | | issue tracker | |
| CrossWalk | | | issue tracker | Released 2014-05-05 |

- <https://www.seleniumhq.org/download/>

Build.gradle

```
dependencies {  
    // compile group: 'org.sikuli', name: 'sikuli-api', version: '1.0.+'  
    compile group: 'org.seleniumhq.selenium', name: 'selenium-java', version: '3.+'  
    compile group: 'commons-io', name: 'commons-io', version: '1.3.+'  
    compile group: 'junit', name: 'junit', version: '4.+'  
    compile group: 'org.slf4j', name: 'slf4j-api', version: '1.7.+'  
    compile group: 'net.sf.opencsv', name: 'opencsv', version: '2.+'  
    compileOnly "org.projectlombok:lombok:1.16.16"  
    // testCompile 'org.hamcrest:hamcrest-library:1.3'  
    compile("org.assertj:assertj-core:3.11.1")  
}
```

How to launch webdriver

```
String driverPath = "C:/Users/Tarkon/IdeaProjects/FinalProjectDemo/src/main/resources/";  
System.setProperty("webdriver.chrome.driver", driverPath + "chromedriver.exe");  
driver = new ChromeDriver();
```


Webdriver top commands

- #1) `get()`
- #2) `getCurrentUrl()`
- #3) `findElement(By, by)` and `click()`
- #4) `isEnabled()`
- #5) `findElement(By, by)` with `sendKeys()`
- #6) `findElement(By, by)` with `getText()`
- #7) `Submit()`
- #8) `findElements(By, by)`

Never Forget to CLOSE or QUIT driver

#6) close() and quit() methods

There are two types of commands in WebDriver to close the web browser instance.

a) close(): WebDriver's close() method closes the web browser window that the user is currently working on or we can also say the window that is being currently accessed by the WebDriver. The command neither requires any parameter nor does it return any value.

b) quit(): Unlike close() method, quit() method closes down all the windows that the program has opened. Same as close() method, the command neither requires any parameter nor does it return any value.

Simple action that can be done with WebElements

- Click();
- sendKeys();
- Submit();
- SelectBy();

sendKeys();

```
// 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("abcdefghijkl"); 4
```

Email address

abcd@gmail.com ✓

Password

.....|

[Forgot your password?](#)



Sign in

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()

click();

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

```
login.click();
```

Email address

abcd@gmail.com



Password

.....

[Forgot your password?](#)



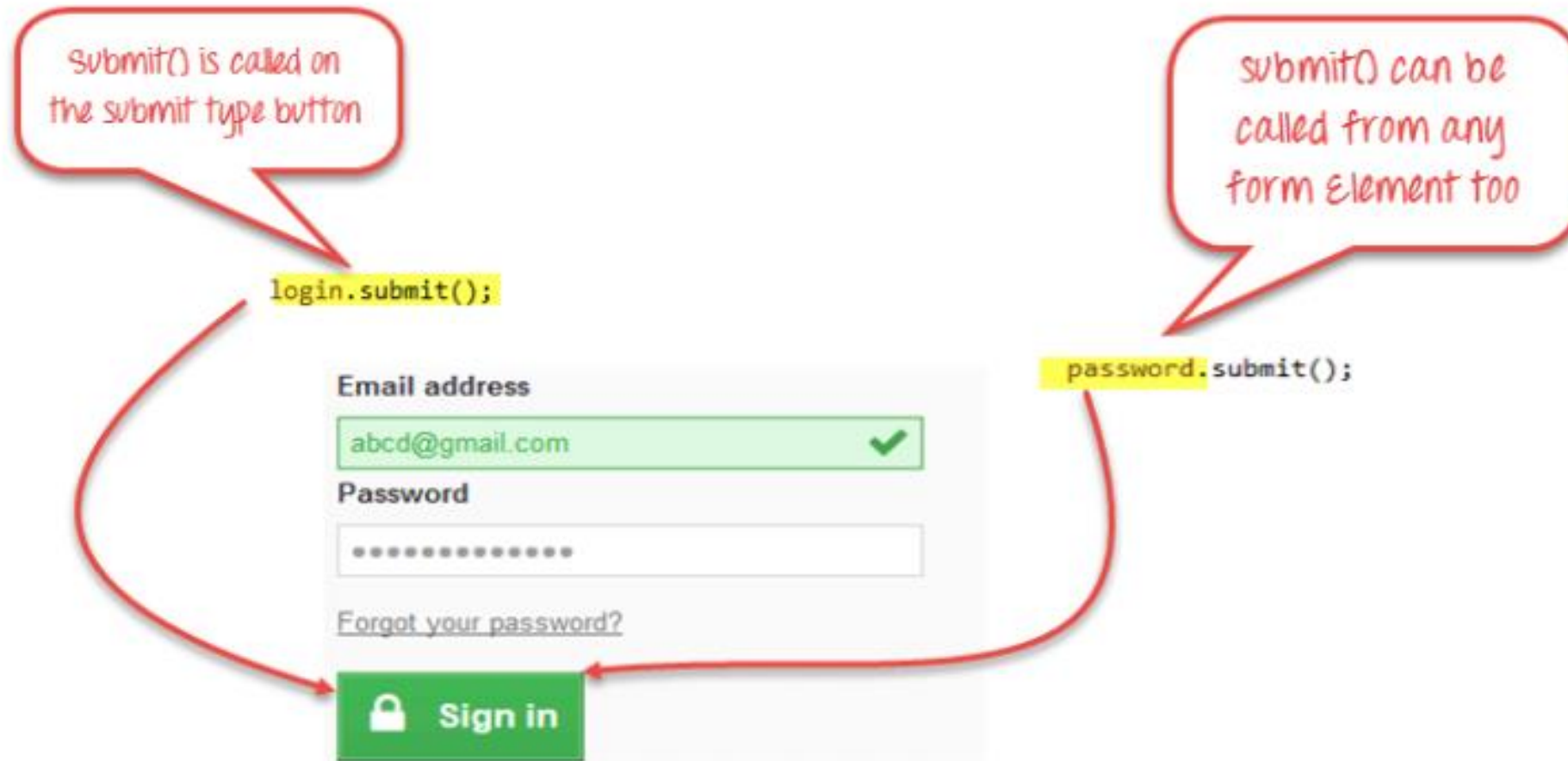
Sign in

clicks the
sign in button

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

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

submit();

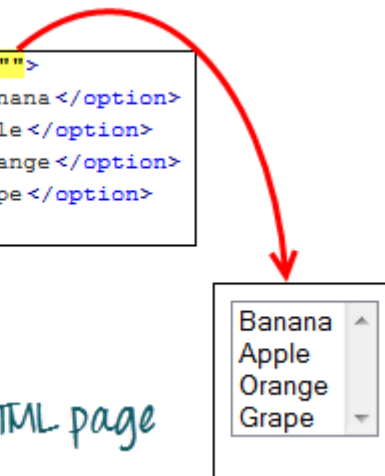


selectByN();

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



Banana
Apple
Orange
Grape

```
//Selecting Items in a Multiple SELECT elements  
driver.get("http://jsbin.com/osebed/2");  
Select fruits = new Select(driver.findElement(By.id("fruits")));  
fruits.selectByVisibleText("Banana");  
fruits.selectByIndex(1);
```


selectByN();

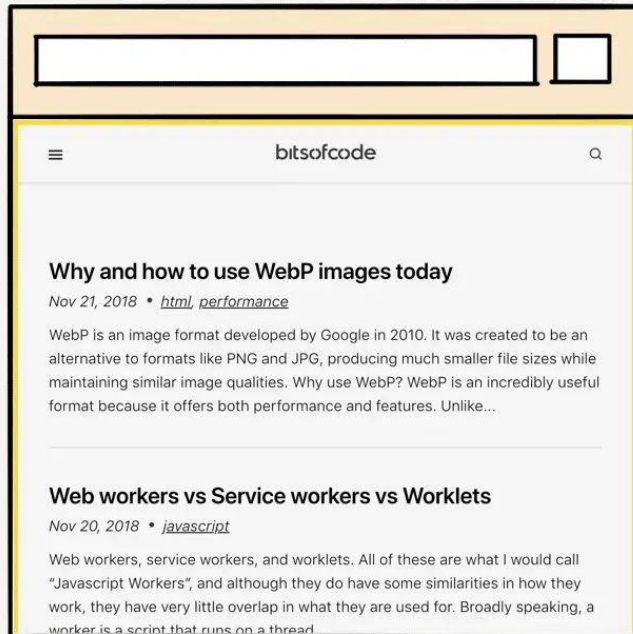
Summary

| Element | Command | Description |
|---------------|---|---|
| Drop-Down Box | <i>selectByVisibleText()/</i> <i>deselectByVisibleText()</i> | selects/deselects an option by its displayed text |
| | <i>selectByValue()/</i> <i>deselectByValue()</i> | selects/deselects an option by the value of its "value" attribute |
| | <i>selectByIndex()/</i> <i>deselectByIndex()</i> | selects/deselects an option by its index |
| | <i>isMultiple()</i> | returns TRUE if the drop-down element allows multiple selection at a time; FALSE if otherwise |
| | <i>deselectAll()</i> | deselects all previously selected options |

Headless browser

- Headless testing is a way of running browser UI tests without the *head*, which in this case means that there's no browser UI, no GUI of any sorts. This is useful since when running tests, especially in a CI environment, there is nobody “watching” the visuals, so there is no need to have the extra overhead of the browser GUI.

Headless browser



headful

```
<html lang="en">
  <head>
    <title> bitsofcode </
  </head>
  <body>
    <header>
      <h1> bitsofcode </h
    </header>
    <main>
```

headless

Headless browser advantages

- Able to run far more instances simultaneously than non-headless drivers.
- Can make use of large amounts of factory-generated or manually created test variables in Data-Driven Testing
- Run-time can be reduced by up to 50% for most tasks.
- Can be executed without taking up the screen context of a computer
- Can be used for scrapping
- Less chances of failure due to 'human intervention.'
- Any desired screenshots are still stored just like in regular automation testing.

Headless browser disadvantages

- Hard to debug inconsistent failures on locating elements due to page loading too fast
- Unintended interactions (losing the benefit of automated UI testing vs integration or unit testing) due to speed/headless state of browser
- When you run headless tests, you are not really mimicking the real user experience.
- Cosmetic bugs can't be identified while doing headless browser testing, such as the location of a button, or the color of a web element, etc.

Headless browser list

- Firefox -> geckodriver
- Chrome
- PhantomJS
- ZombieJS
- HtmlUnit
- Puppeteer

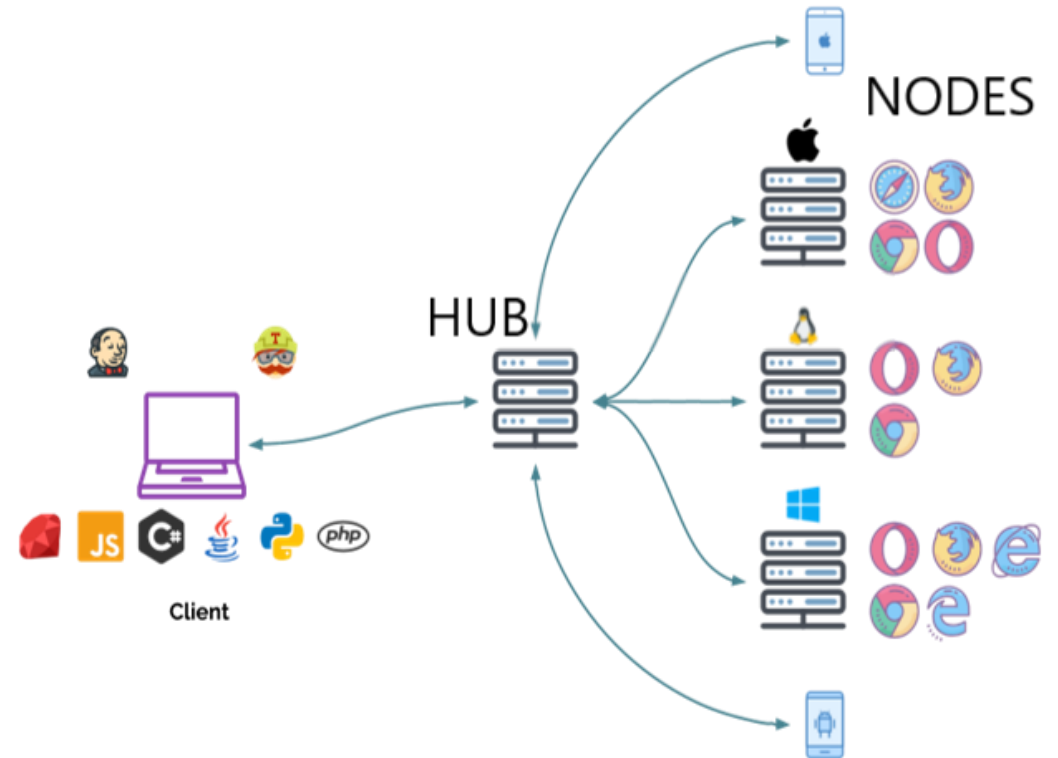
Using headless browser

```
String driverPath = "C:/Users/Tarkon/IdeaProjects/FinalProjectDemo/src/main/resources/";  
ChromeOptions options = new ChromeOptions();  
options.addArguments("headless");  
System.setProperty("webdriver.chrome.driver", driverPath + "chromedriver.exe");  
driver = new ChromeDriver(options);
```


Selenium Grid

- Selenium Grid is a server that allows tests to use web browser instances running on remote machines. With Selenium Grid, one server acts as the hub.
- A grid consists of a single **hub**, and one or more **nodes**.

https://www.seleniumhq.org/docs/07_selenium_grid.jsp



Selenium Grid

```
import org.openqa.selenium.*;
import org.openqa.selenium.remote.DesiredCapabilities;
import java.net.MalformedURLException;
import java.net.URL;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.testng.Assert;
import org.testng.annotations.*;

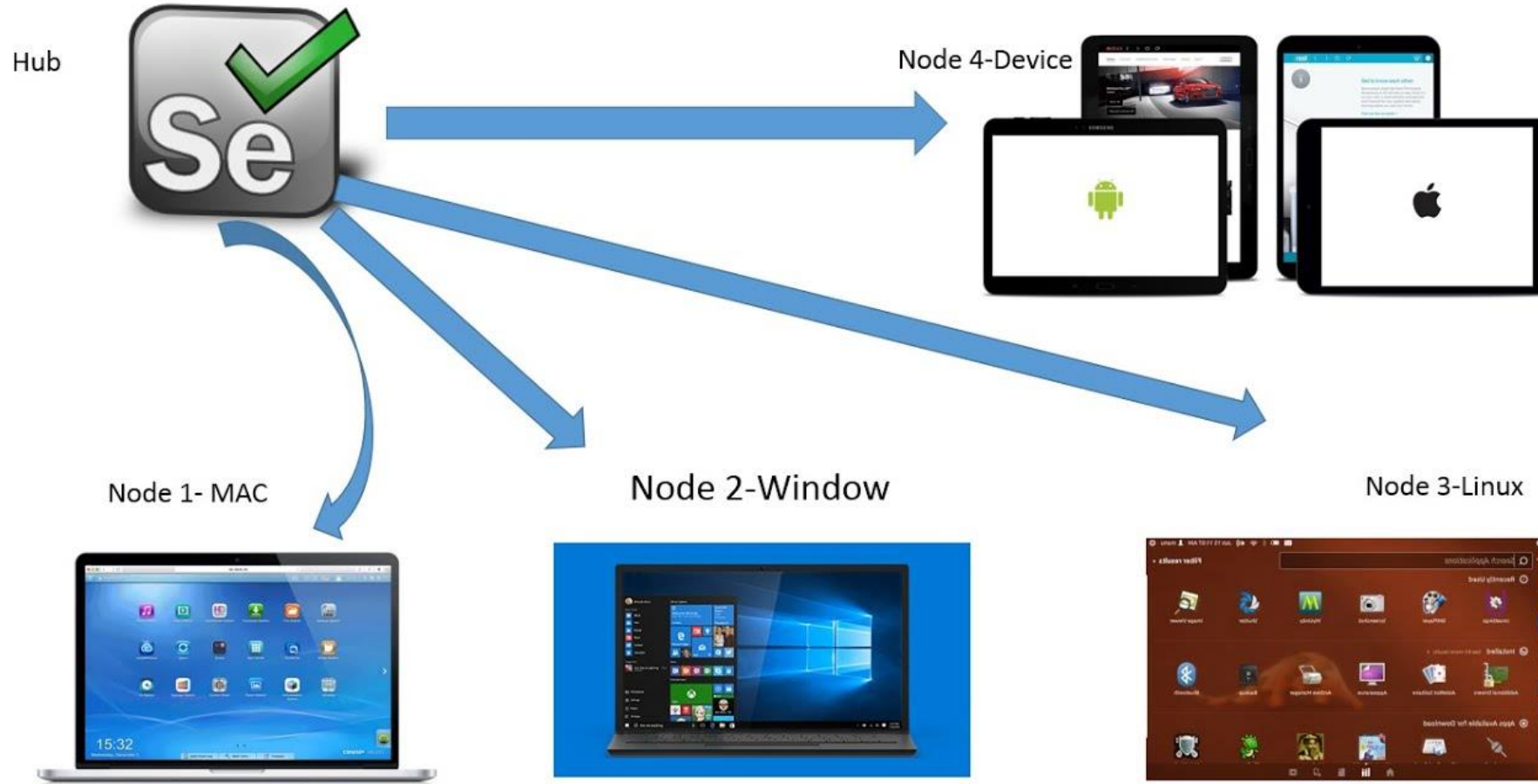
public class Grid_2 {
    WebDriver driver;
    String baseUrl, nodeURL;

    @BeforeTest
    public void setUp() throws MalformedURLException {
        baseUrl = "http://newtours.demoaut.com/";
        nodeURL = "http://192.168.1.4:5566/wd/hub";
        DesiredCapabilities capability = DesiredCapabilities.firefox();
        capability.setBrowserName("firefox");
        capability.setPlatform(Platform.XP);
        driver = new RemoteWebDriver(new URL(nodeURL), capability);
    }

    @AfterTest
    public void afterTest() {
        driver.quit();
    }

    @Test
    public void simpleTest() {
        driver.get(baseUrl);
        Assert.assertEquals("Welcome: Mercury Tours", driver.getTitle());
    }
}
```

Selenium Grid

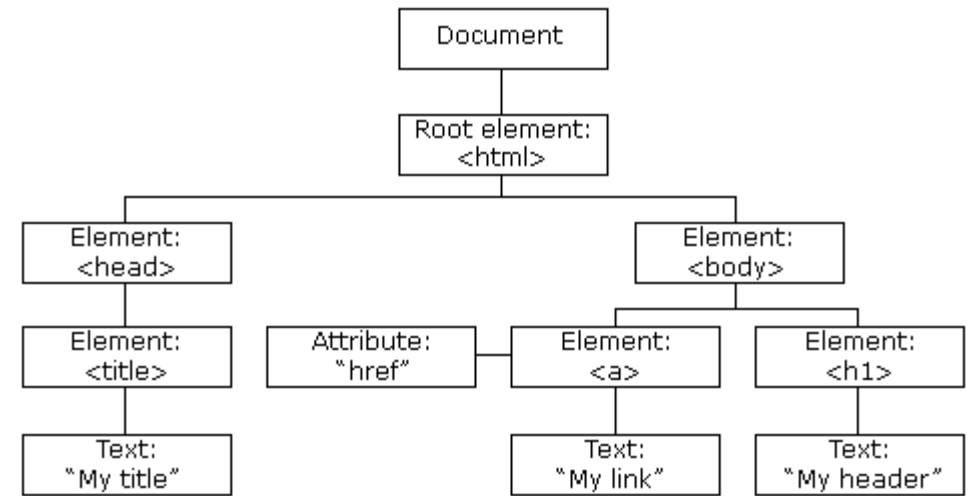


Agenda

- Selenium Web driver
- **Locators**
- Xpath
- Css selector
- Iframe
- Practice

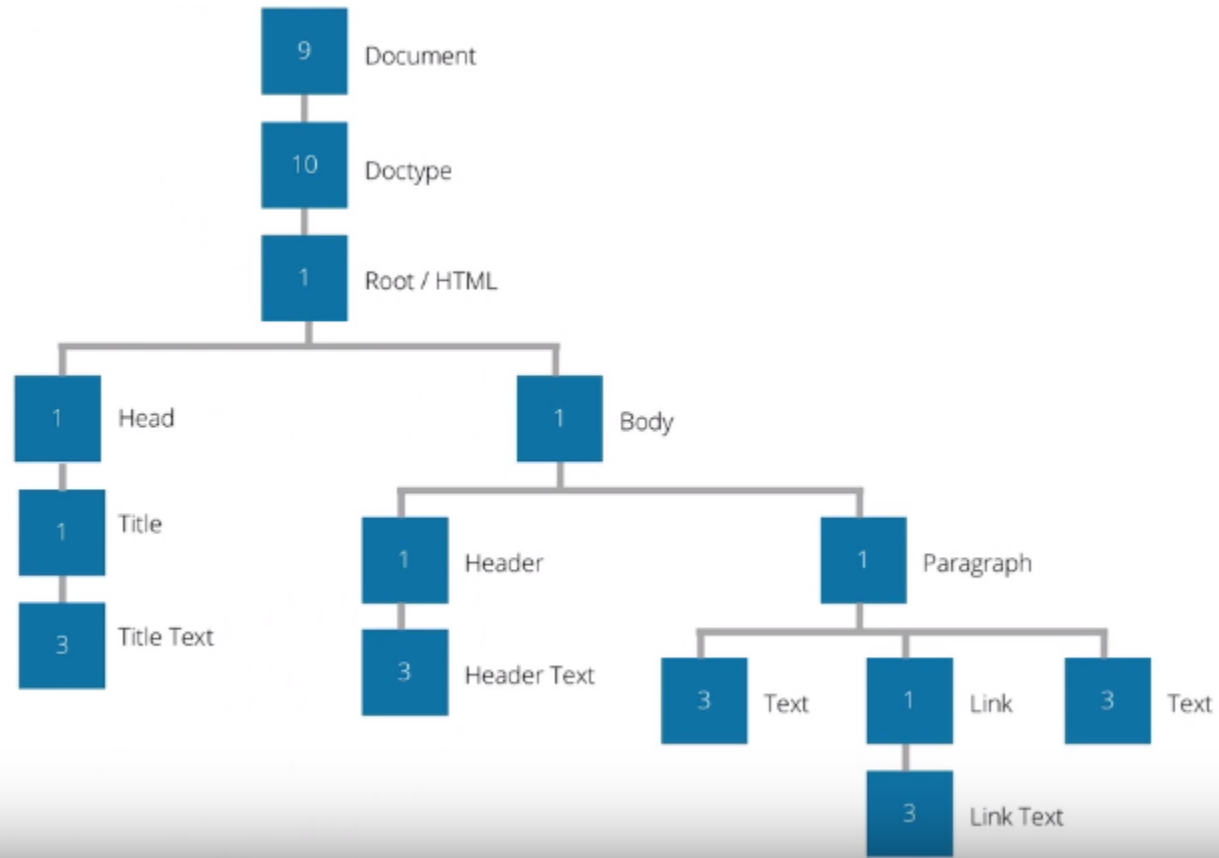
The HTML DOM (Document Object Model)

- When a web page is loaded, the browser creates a Document Object Model of the page
- The HTML DOM model is constructed as a tree of Objects
- In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements
- With the document object model, JavaScript gets all the power it needs to create dynamic HTML
- Document Object Model or DOM is an essential component of web development and **automation**



HTML DOM example

```
<!DOCTYPE html>
<html>
<head>
  <title>The DOM</title>
</head>
<body>
  <h1>Title</h1>
  <p>Lorem <a title="Learn more"
    href="#">to the</a> ipsum</p>
</body>
</html>
```



HTML (Hyper Text Markup Language)

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Page Title</title>
```

```
</head>
```

```
<body>
```

```
<h1>This is a Heading</h1>
```

```
<p>This is a paragraph.</p>
```

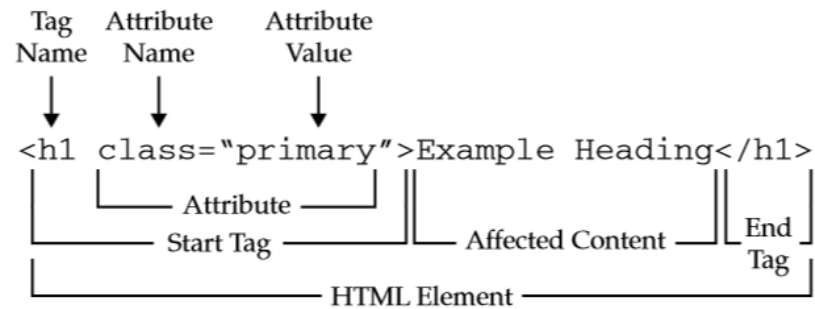
```
</body>
```

```
</html>
```

- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page
- All HTML documents must start with a document type declaration: `<!DOCTYPE html>`
- Begins with `<html>` and ends with `</html>`
- The visible part of the HTML document is between `<body>` and `</body>`

HTML syntax overview

A graphical overview of the HTML markup syntax shown so far is presented here:

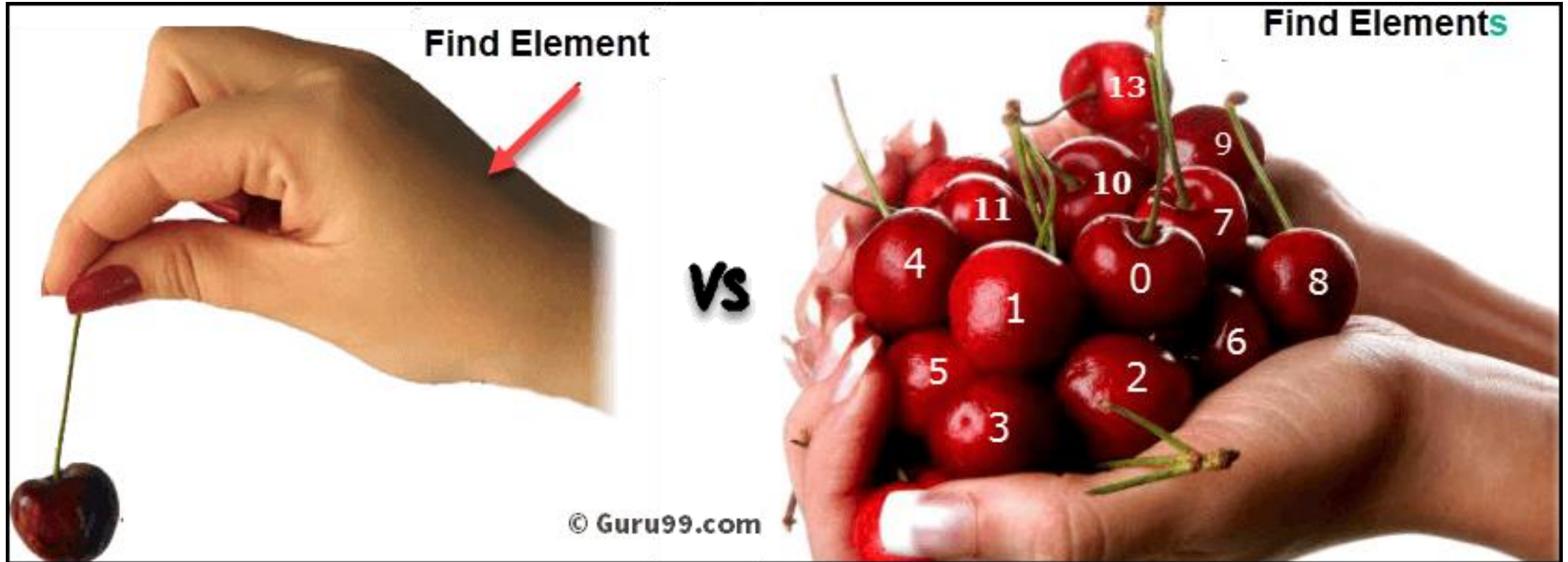


| Tag | Description |
|--|---|
| <code><html> ... </html></code> | Declares the Web page to be written in HTML |
| <code><head> ... </head></code> | Delimits the page's head |
| <code><title> ... </title></code> | Defines the title (not displayed on the page) |
| <code><body> ... </body></code> | Delimits the page's body |
| <code><h n> ... </h n></code> | Delimits a level <i>n</i> heading |
| <code> ... </code> | Set ... in boldface |
| <code><i> ... </i></code> | Set ... in italics |
| <code><center> ... </center></code> | Center ... on the page horizontally |
| <code> ... </code> | Brackets an unordered (bulleted) list |
| <code> ... </code> | Brackets a numbered list |
| <code> ... </code> | Brackets an item in an ordered or numbered list |
| <code> </code> | Forces a line break here |
| <code><p></code> | Starts a paragraph |
| <code><hr></code> | Inserts a horizontal rule |
| <code></code> | Displays an image here |
| <code> ... </code> | Defines a hyperlink |

Locators – what it is?

- Locator is a command that tells Selenium which GUI elements (say Text Box, Buttons, Check Boxes etc) its needs to operate on. Identification of correct GUI elements is a prerequisite to creating an automation script. But accurate identification of GUI elements is more difficult than it sounds. Sometimes, you end up working with incorrect GUI elements or no elements at all! Hence, Selenium provides a number of Locators to precisely locate a GUI element

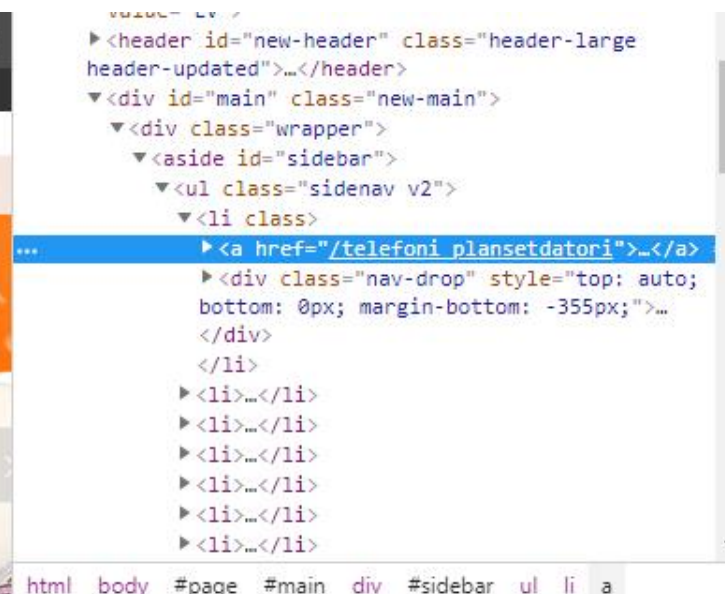
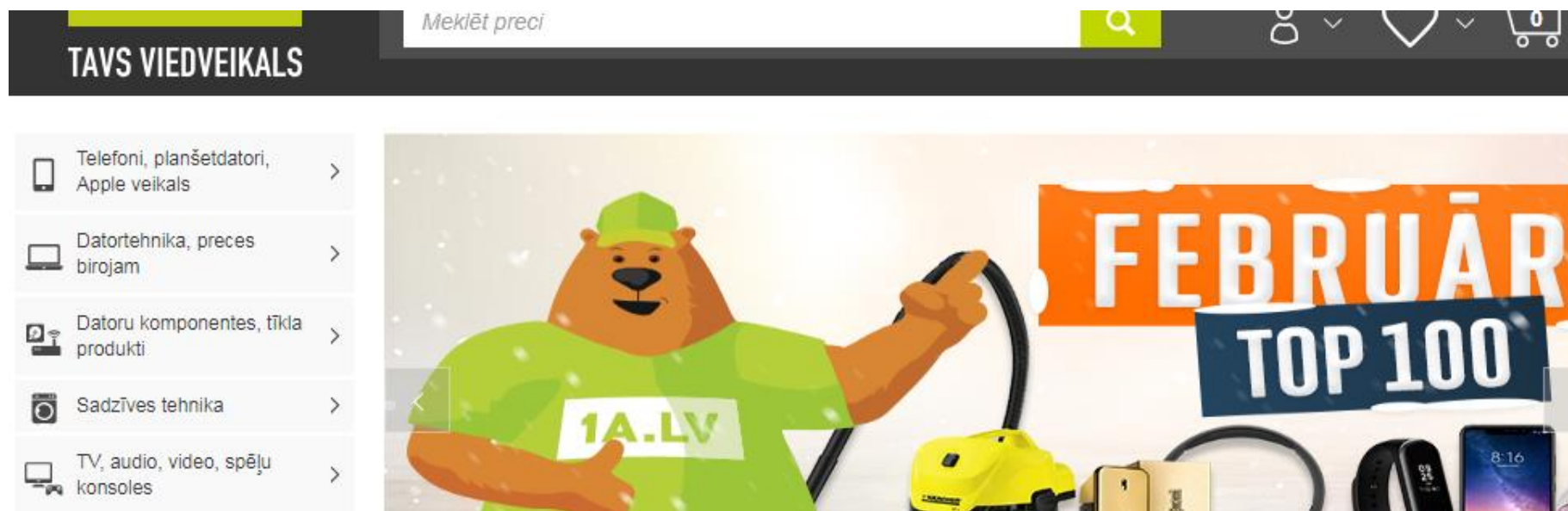
findElement vs findElements



Locators

- ID
- Name
- Link Text
- CSS Selector
 - Tag and ID
 - Tag and class
 - Tag and attribute
 - Tag, class, and attribute
 - Inner text

How to look?



Agenda

- Selenium Web driver
- Locators
- **Xpath**
- Css selector
- Iframe
- Practice

What is Xpath

- XPath stands for XML Path Language
- XPath uses "path like" syntax to identify and navigate nodes in an XML document
- XPath contains over 200 built-in functions
- XPath is a major element in the XSLT standard
- XPath is a W3C recommendation

XPath Path Expressions

XPath uses path expressions to select nodes or node-sets in an XML document.

These path expressions look very much like the path expressions you use with traditional computer file systems:



How XPath Works

The XPath specification is the foundation for a variety of specifications, including XSLT and linking/addressing specifications such as `xpointer`. So an understanding of XPath is fundamental to a lot of advanced XML usage. This section provides an introduction to XPath in the context of XSLT.

XPath Expressions

In general, an XPath expression specifies a pattern that selects a set of XML nodes. XSLT templates then use those patterns when applying transformations. (`xpointer`, on the other hand, adds mechanisms for defining a **point** or a **range** so that XPath expressions can be used for addressing).

The nodes in an XPath expression refer to more than just elements. They also refer to text and attributes, among other things. In fact, the XPath specification defines an abstract document model that defines seven kinds of nodes:

- Root
- Element
- Text
- Attribute
- Comment
- Processing instruction
- Namespace

The root element of the XML data is modeled by an **element** node. The XPath root node contains the document's root element as well as other information relating to the document.

Xpath what is is?

XPath Selenium Selectors

We can find the location of any element on a web page using XML path expressions. The basic syntax for XPath is shown below:

Syntax = `//tagname[@attribute='Value']`

Example = `//input[@id='user-message']`

The screenshot illustrates the application of XPath selectors in a web browser. The top part shows a web form with a label "Enter message", a text input field with the placeholder "Please enter your Message", and a "Show Message" button. The bottom part shows the FirePath extension's interface. The "XPath" tab is selected, and the selector `//input[@id='user-message']` is entered. Red dashed arrows point from the labels "tagname", "attribute", and "value" to the corresponding parts of the XPath expression: "input" for tagname, "@" for attribute, and "id" for value. The FirePath tool also displays the corresponding HTML code snippet, where the `<input id="user-message" class="form-control" placeholder="Please enter your Message" type="text"/>` line is highlighted, with red dashed arrows pointing from the labels to the respective parts of the HTML tag.

```
HTML CSS Script DOM Net Cookies FirePath
XPath: - //input[@id='user-message']


<label for="message">Enter message</label>
  <input id="user-message" class="form-control" placeholder="Please enter your Message" type="text"/>
</div>
<button class="btn btn-default" type="button" onclick="showInput();" >Show Message</button>
</form>

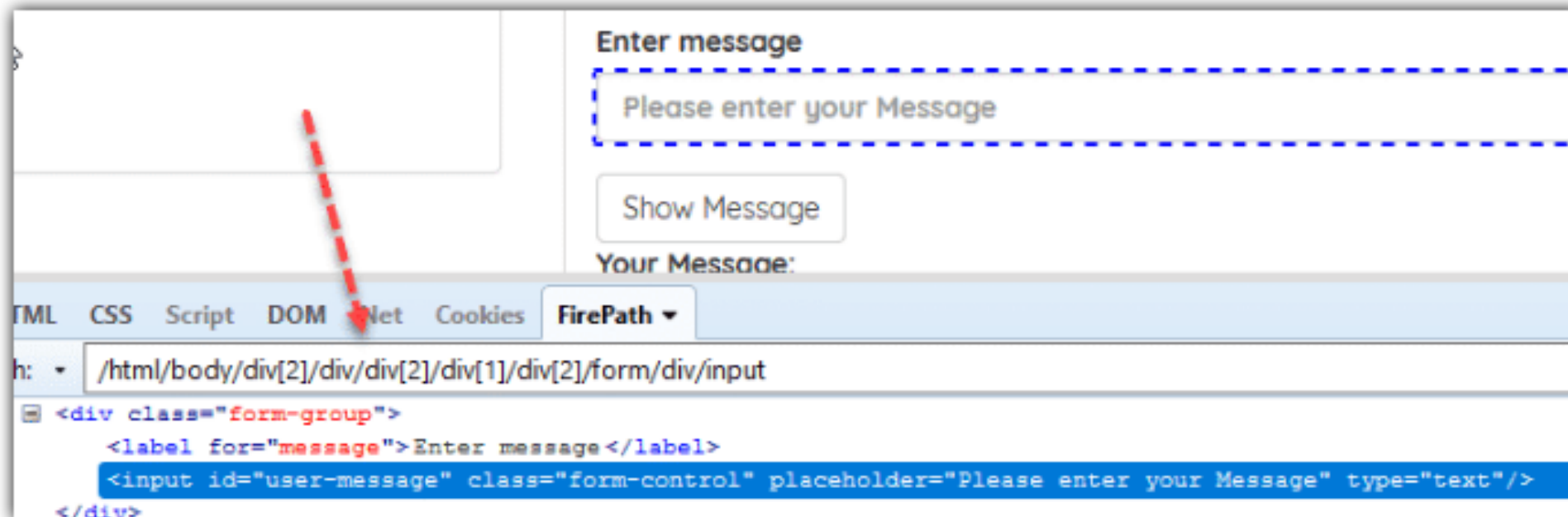

```

DO NOT USE

Absolute XPath

- It is a direct way to locate an element.
- It is very brittle.
- Starts with single slash "/" that means starting to search from the root.

Example: `/html/body/div[2]/div/div[2]/div[1]/div[2]/form/div/input`



Common Xpath locators

Syntax = `//tagname[@attribute='Value']`

Example = `//input[@id='user-message']`

The screenshot illustrates the process of locating an element using XPath. The web page shows a form with the following structure:

- Enter message
- Please enter your Message (text input field)
- Show Message (button)
- Your Message:

The FirePath tool is open, showing the XPath locator `//input[@id='user-message']`. The HTML code for the form is displayed below, with the input field highlighted. Red arrows indicate the mapping between the XPath syntax and the HTML attributes:

- tagname**: Points to `input` in the XPath locator and the `<input>` tag in the HTML code.
- attribute**: Points to `@id` in the XPath locator and `id` in the HTML code.
- value**: Points to `'user-message'` in the XPath locator and `"user-message"` in the HTML code.

```
<div class="form-group">
  <label for="message">Enter message</label>
  <input id="user-message" class="form-control" placeholder="Please enter your Message" type="text"/>
</div>
<button class="btn btn-default" type="button" onclick="showInput();" >Show Message</button>
</form>
```

Agenda

- Selenium Web driver
- Locators
- Xpath
- **Css selector**
- Iframe
- Practice

CSS (Cascading Style Sheets)

- `body {`
- `background-color: lightblue;`
- `}`

- `h1 {`
- `color: white;`
- `text-align: center;`
- `}`

- `p {`
- `font-family: verdana;`
- `font-size: 20px;`
- `}`

- **CSS** stands for **Cascading Style Sheets**
- CSS describes **how HTML elements are to be displayed**
- Contains the rules for the presentation of HTML.
- CSS was introduced to keep the presentation information separate from HTML markup (content).
- CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
- External stylesheets are stored in **CSS files**



+



=

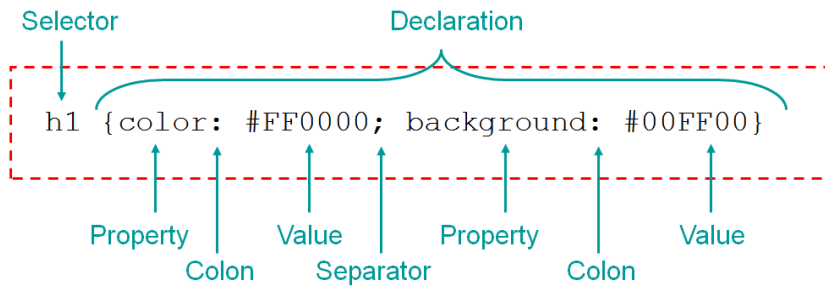


HTML

CSS

Web Page

CSS syntax overview



The element Selector

```
p {  
  text-align: center;  
  color: red;  
}
```

The class Selector

```
.className {  
  text-align: center;  
  color: red;  
}
```

The id Selector

```
#myId {  
  text-align: center;  
  color: red;  
}
```

Grouping Selectors

```
h1, h2 {  
  text-align: center;  
  color: red;  
}
```

| Property | Description | Example |
|-------------------|---|--|
| color | Sets the foreground color of an element | body{color: #FCC9814;} |
| background-color | Sets the background color of an element | body{background-color: green;} |
| background-image | Inserts a background image | body{background-image: url ("earth.gif");} |
| background-repeat | Specifies how background image will repeat itself. It may repeat horizontally (repeat-x) vertically (repeat-y) or (repeat) or may not be repeated (no-repeat) | body{background-repeat: repeat-x;} |

Cssselector what is is?

- As we all know, CSS stands for **Cascading Style Sheets**. By using CSS selectors, we can find or select HTML elements on the basis of their id, class or other attributes.

Why to use it?

Using CSS selectors to locate elements has some benefits:

- It's faster
- More readable
- And used more often

Why to use it?



Why Choose CSS Selectors for Selenium Automation?

Other Selectors

⇒ `By.cssSelector`

⇒ -----

`By.className("register")` ⇒ `".register"`

`By.tagName("table")` ⇒ `"table"`

`By.id("unique_id")` ⇒ `"#unique_id"`

`By.name("login")` ⇒ `"[name=login]"`

`By.xpath("//body/nav")` ⇒ `"body > nav"`

`By.xpath("//body//nav")` ⇒ `"body nav"`

How to use it?

The generic way of locating elements by their attribute in CSS Selectors is

```
css = element_name[<attribute_name>='<value>']
```

e.g. `css=input[name='first_name']`

CSS selectors

You can use **tag[attribute='value']** syntax.

(Note: For XPath we use **tag[@attribute='value']** for this.)

Example

Syntax: **input[id='user-message']**

The screenshot shows a web browser interface with a form on the right and its DOM structure in the FirePath panel at the bottom. The form contains a label 'Enter message', a text input field with the placeholder 'Please enter your Message', and a 'Show Message' button. The FirePath panel shows the DOM tree with the following structure:

```
<p> First, Let us try to be very simple with only one
+ <ul>
- <form id="get-input" method="post">
  - <div class="form-group">
    - <label for="message">Enter message</label>
    - <input id="user-message" class="form-contr
  - </div>
```

Red arrows point from the text 'user-message' in the CSS selector 'input[id='user-message']' to the 'id' attribute in the DOM tree. The selected element in the DOM tree is highlighted in blue.

XPath vs CSSselector

- CSS is faster and simpler than Xpath particularly in case of IE browser where Xpath works very slowly.

| | Cr 32 | Cr 32 | FF 26 | FF 26 | IE 8 | IE 8 | Op 12 | Op 12 |
|-----------------------------------|----------|----------|----------|----------|-----------|----------|----------|----------|
| | CSS | XPath | CSS | XPath | CSS | XPath | CSS | XPath |
| header_id_and_class | 1.132802 | 1.132802 | 0.829401 | 1.099982 | 8.550464 | 5.743214 | 1.859979 | 2.128673 |
| header_id_class_and_direct_desc | 1.132802 | 1.235784 | 0.922043 | 1.051453 | 9.914347 | 6.490025 | 2.358386 | 2.143988 |
| header_traversing | 1.214156 | 1.132800 | 0.873515 | 0.905867 | N/A | 7.388131 | 2.098045 | 2.317037 |
| header_traversing_and_direct_desc | 1.144128 | 1.212096 | 1.002925 | 0.922043 | N/A | 7.733464 | 2.408921 | 2.173082 |
| cell_id_and_class | 1.121472 | 1.719590 | 0.986748 | 0.938220 | 13.063517 | 7.812772 | 2.375230 | 2.274156 |
| cell_id_class_and_direct_desc | 1.258541 | 1.295923 | 1.002924 | 1.069246 | 12.334165 | 7.039037 | 2.430812 | 2.853636 |
| cell_traversing | 1.146393 | 1.271002 | 1.156597 | 1.085422 | N/A | 7.315236 | 2.705395 | 2.890695 |
| cell_traversing_and_direct_desc | 1.233619 | 1.196237 | 1.459092 | 1.174391 | N/A | 5.909296 | 2.377413 | 2.268085 |

XPATH VS CSS PATH CHEAT SHEET

| DESCRIPTION | XPATH | CSS PATH |
|-----------------------|---|--|
| Direct Child | <code>//div/a</code> | <code>div > a</code> |
| Child or Sub Child | <code>//div//a</code> | <code>div a</code> |
| Id | <code>//div[@id='idValue']//a</code> | <code>div#idValue a</code> |
| Class | <code>//div[@class='classValue']//a</code> | <code>div.classValue a</code> |
| Attribute | <code>//form/input[@name='username']</code> | <code>form input[name='username']</code> |
| Following Sibling | <code>//li[@class='first']/following-sibling::li</code> | <code>li.first + li</code> |
| Multiple Attributes | <code>//input[@name='continue' and @type='button']</code> | <code>input[name='continue'][type='button']</code> |
| nth Child | <code>//ul[@id='list']/li[4]</code> | <code>ul#list li:nth-child(4)</code> |
| First Child | <code>//ul[@id='list']/li[1]</code> | <code>ul#list li:first-child</code> |
| Last Child | <code>//ul[@id='list']/li[last()]</code> | <code>ul#list li:last-child</code> |
| Attribute Contains | <code>//div[contains(@title,'Title')]</code> | <code>div[title*="Title"]</code> |
| Attribute Starts With | <code>//input[starts-with(@name,'user')]</code> | <code>input[name^="user"]</code> |
| Attribute Ends With | <code>//input[ends-with(@name,'name')]</code> | <code>input[name\$="name"]</code> |
| With Attribute | <code>//div[@title]</code> | <code>div[title]</code> |

Agenda

- Selenium Web driver
- Locators
- Xpath
- Css selector
- **Iframe + Pop up and alerts**
- Practice

Iframe

```
<!DOCTYPE html>
<html>

  <head>
    <title>HTML Iframes</title>
  </head>

  <body>
    <p>Document content goes here...</p>
    <iframe src = "/html/menu.htm" width = "555" height = "200">
      Sorry your browser does not support inline frames.
    </iframe>
    <p>Document content also go here...</p>
  </body>

</html>
```


Iframe

The image shows a web browser window displaying a page titled "Embeds" with a sub-header "iFrame". The page content includes a Google search bar, a Gmail link, and a video player showing a "Video chat with a friend, ping a colleague, or give someone a ring" with a Google+ logo. Below the video is an "Audio" section. The browser's address bar shows the URL "127.0.0.1:64350/index.html".

On the left, a code editor (Brackets) shows the HTML code for the page. The code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport"
6     content="width=device-width, initial-
7     scale=1.0">
8   <link rel="stylesheet"
9     href="css/style.css">
10  <title>Form</title>
11 </head>
12 <body>
13   <div class="wrap">
14     <h1>Embeds</h1>
15     <h2>iFrame</h2>
16     <iframe src="https://gmail.com"
17       frameborder="2" width="100%"
18       height="300px"></iframe>
19     <h2>Audio</h2>
20     <h2>Video</h2>
21   </div>
22 </body>
23 </html>
24
```

A red box highlights the `<iframe>` tag in the code editor. A red arrow points from this box to the video player in the browser window. A red box with the text "The size is now changed" is also present, indicating the effect of the `width="100%"` and `height="300px"` attributes.

Iframe

```
driver.switchTo().frame("a077aa5e");
```

```
driver.switchTo().parentFrame();  
driver.switchTo().defaultContent();
```

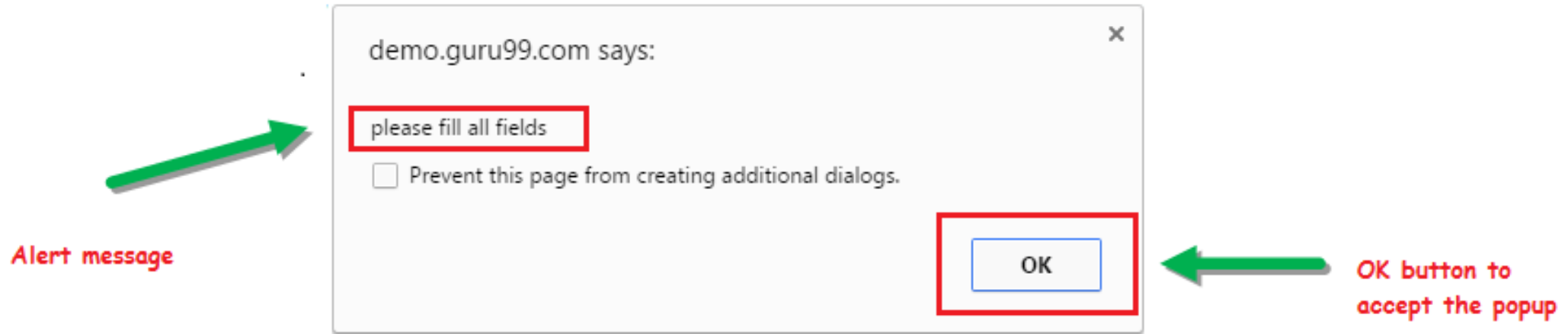
Iframe

```
/*for(int i=0; i<=size; i++){
    driver.switchTo().frame(i);
    int total=driver.findElements(By.xpath("html/body/a/img")).size();
    System.out.println(total);
    driver.switchTo().defaultContent(); //switching back from the iframe
}*/

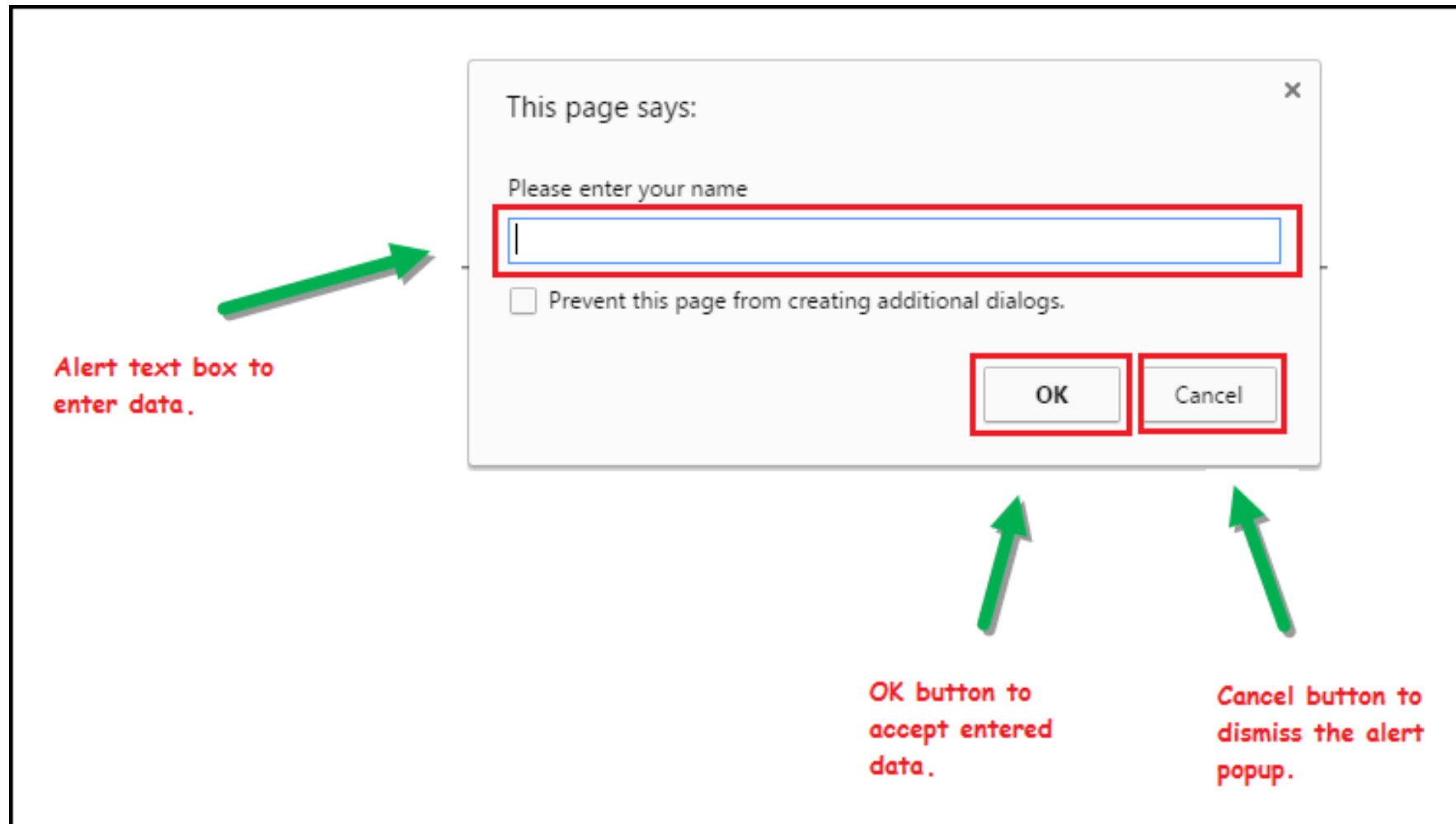
//Commented the code for finding the index of the element
driver.switchTo().frame(0); //Switching to the frame
System.out.println("*****We are switched to the iframe*****");
driver.findElement(By.xpath("html/body/a/img")).click();

//Clicking the element in line with Advertisement
System.out.println("*****We are done*****");
}
}
```

Pops and Alerts



Pops and Alerts



Pops and Alerts

1) void dismiss() // To click on the 'Cancel' button of the alert.

```
driver.switchTo().alert().dismiss();
```

2) void accept() // To click on the 'OK' button of the alert.

```
driver.switchTo().alert().accept();
```

3) String getText() // To capture the alert message.

```
driver.switchTo().alert().getText();
```

4) void sendKeys(String stringToSend) // To send some data to alert box.

```
driver.switchTo().alert().sendKeys("Text");
```

Agenda

- Selenium Web driver
- Locators
- Xpath
- Css selector
- Iframe
- **Practice**

Reminder

- Setup Webdriver location
- Create driver
- Open url
- FindElement
- DO SOME MAGIC
- Close or Quit driver

Practice 1

Scenario demo

1. Go to <https://www.ss.com/lv/>
2. Click on “Suņi, kucēni”
3. Assert that opened page is correct
4. Enter “Vecums 1”
5. Check if Vecums 1 is entered

Practice 2

- Go to <https://www.ss.com/lv/>
- Open cars
- Check that cars page is opened
- Enter price 6000 - 10000
- Check if price parameter is selected
- year from 2001
- Check if year parameter is selected
- engine max 3.0
- Check if engine parameter is selected
- color – Balta
- Check if color parameter is selected
- Click submit

Practice 3

- Go to <https://www.facebook.com/> or any other social network
- Login with your credentials
- Press Login
- Open your profile

Homework - Obligatory

- Go to <https://www.aliexpress.com/>
 - In search write: tattoo
 - Press search
 - Set min price 10
 - Set max price 20
 - Press ok
-
- Hint! – Try to click twice on same window or use Tab key to get to correct price field

Bonus Task of PAIN

1. Open <https://220.lv>
2. Click on menu item “Auto preces”
3. Click on category “Riepas”
4. Verify page title is “PLAŠĀKAIS AUTO PREČU KLĀSTS ! IENĀC INTERNETA VEIKALĀ 220.LV ! | 220.lv”
5. Click on “Vasaras riepas”
6. Search tires by following input “Riepas platums 215 Profils 60 Ražotājs Bridgestone”
7. Add to cart



THANK YOU FOR YOUR ATTENTION

