# Test Automation Lecture 10 – Selectors & Locating elements


IT Learning & Outsourcing Center

Lector: Milen Strahinski
Skype: strahinski
E-mail: milen.strahinski@pragmatic.bg
Facebook: http://www.facebook.com/LamerMan
LinkedIn: http://www.linkedin.com/pub/milen-strahinski/a/553/615

www.pragmatic.bg

# Summary - overall

- Tools for element locating –Firefox Inspector, IE Developer Tools, Chrome Developer Tools
- Locating elements by id, name, DOM, xPath, CSS, etc. for both Selenium IDE & WebDriver
- Preferred locators for speed of execution

# Selenium IDE - DEPRECATED

- In order to use the nice “Find” button of the Selenium IDE to test your locators you will need to first install Mozilla Firefox version 54 (this is Win64 version): https://ftp.mozilla.org/pub/firefox/releases/54.0/win64/en-US/Firefox%20Setup%2054.0.exe

- Make sure to STOP immediately the auto-updater of the browser by going to Options -> Advanced -> Update -> Click the radio button “Never check for updates (not recommended: security risk)”

- To check your current version of Firefox, go to Help menu -> click on “About Firefox”

# Alternatives

- Eskry – a Chrome addon used for testing CSS and Xpath locators

https://chrome.google.com/webstore/detail/eskry/hfklgljgigfgbdklkehjikeedmfmkjaf

- CSS Selector Tester – a Chrome addon used for CSS locators - https://chrome.google.com/webstore/detail/css-selector-tester/bbklnaodgoocmcdejoalmbjihhdkbfon

# Preliminary points

- Before proceed further, ensure we have the following tools installed:
  - **Firefox Inspector:** Already built into Mozilla Firefox browser. (F12)
  - **IE Developer Tools:** This is built into IE7, IE8 and IE9 that we can launch by pressing F12. It also has a number of features that Firebug has.
  - **Google Chrome Developer Tools:** This, like IE, is built into the browser and will also allow you to find the elements on the page and be able to work out its XPath. (F12)
  - **Selenium IDE addon:** This is a Mozilla Firefox browser addon used for writing and recording automated test scripts. http://docs.seleniumhq.org/download/
  - If "Find" in Selenium IDE does not work, make sure to install also plugin - https://addons.mozilla.org/en-US/firefox/addon/test-results-selenium-ide/versions/

Let's play a little with these tools!

# Selenium IDE locators

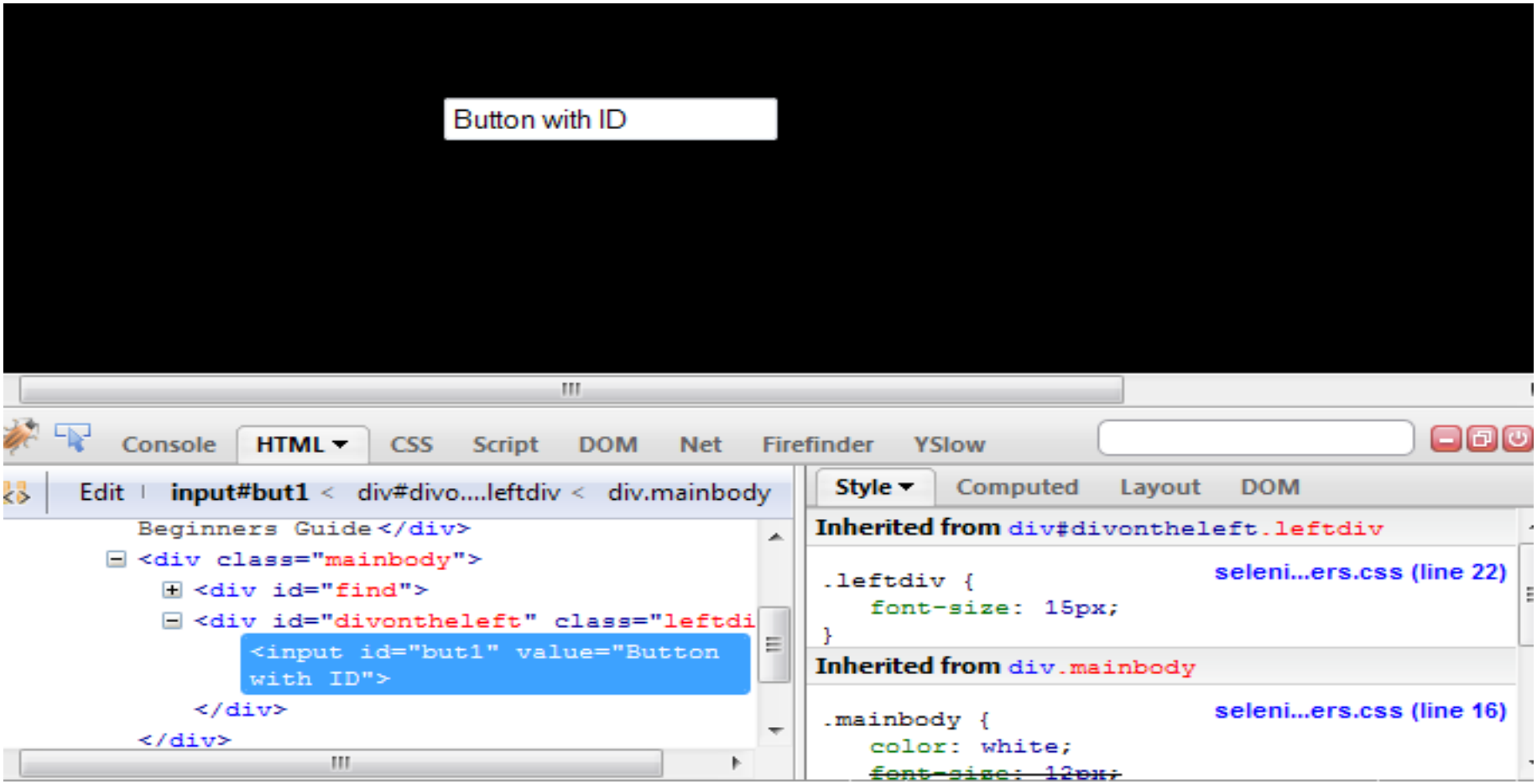


Focusing on these locators using Selenium IDE:

- Locate elements by ID
- Locate elements by Name
- Locate elements by Link
- Locate elements by CSS
- Locate elements by DOM
- Locate elements by Xpath

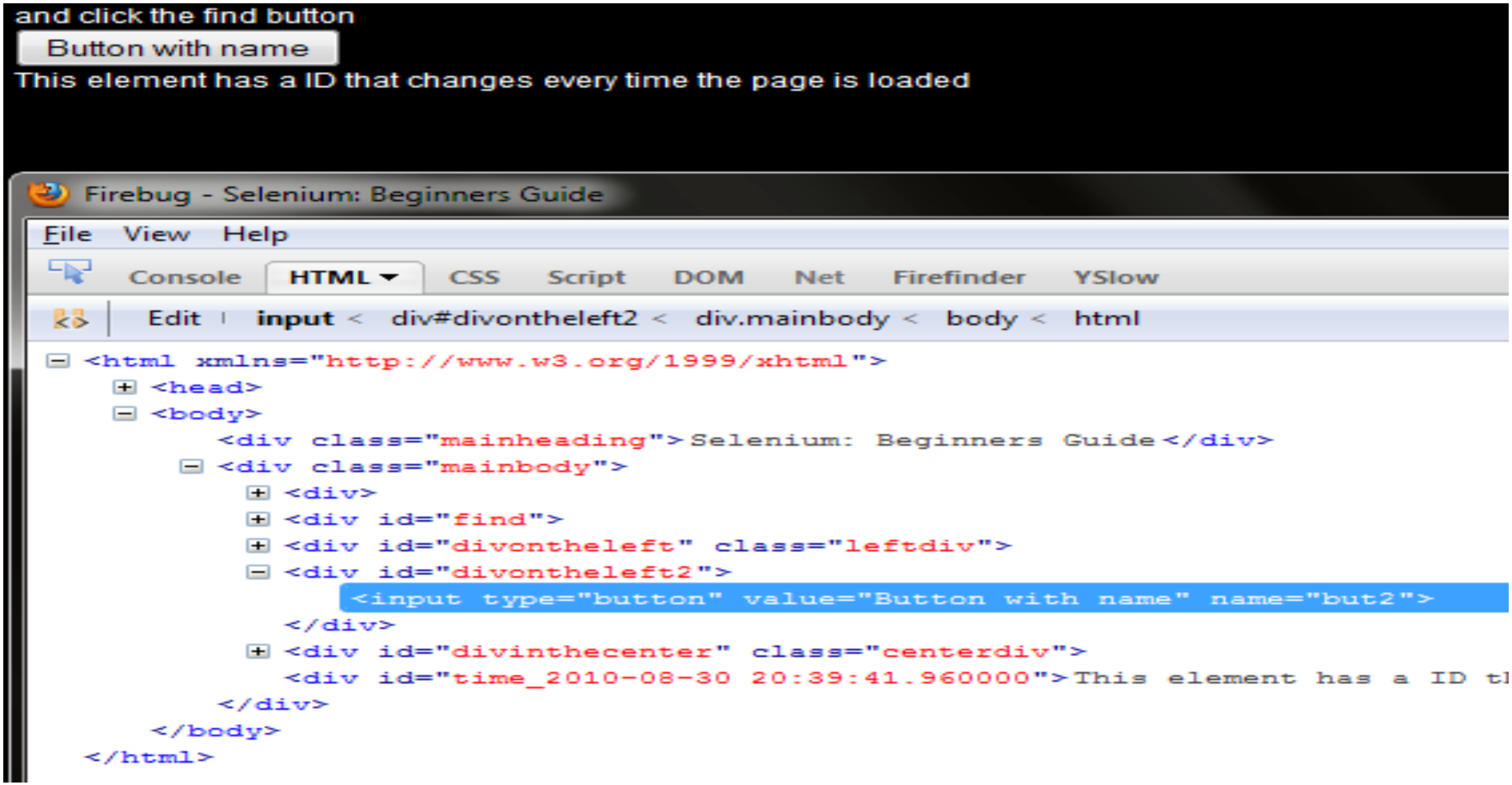
# Locating by ID

- Open http://pragmatic.bg/automation/example2.html



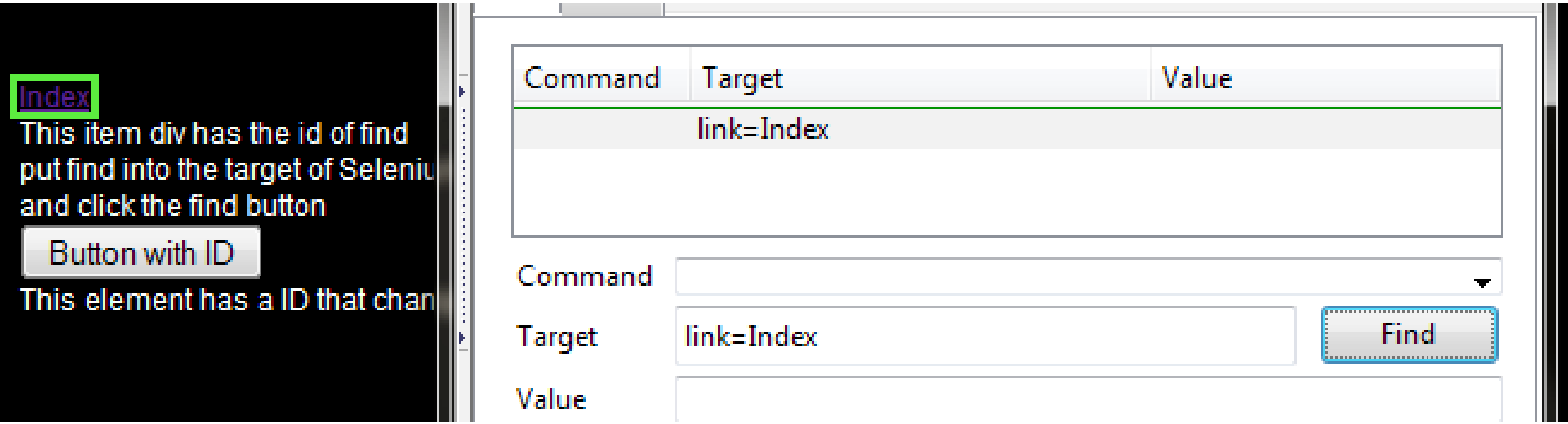
# Locating by *name*

- Open http://pragmatic.bg/automation/example2.html



# Locating by *link*

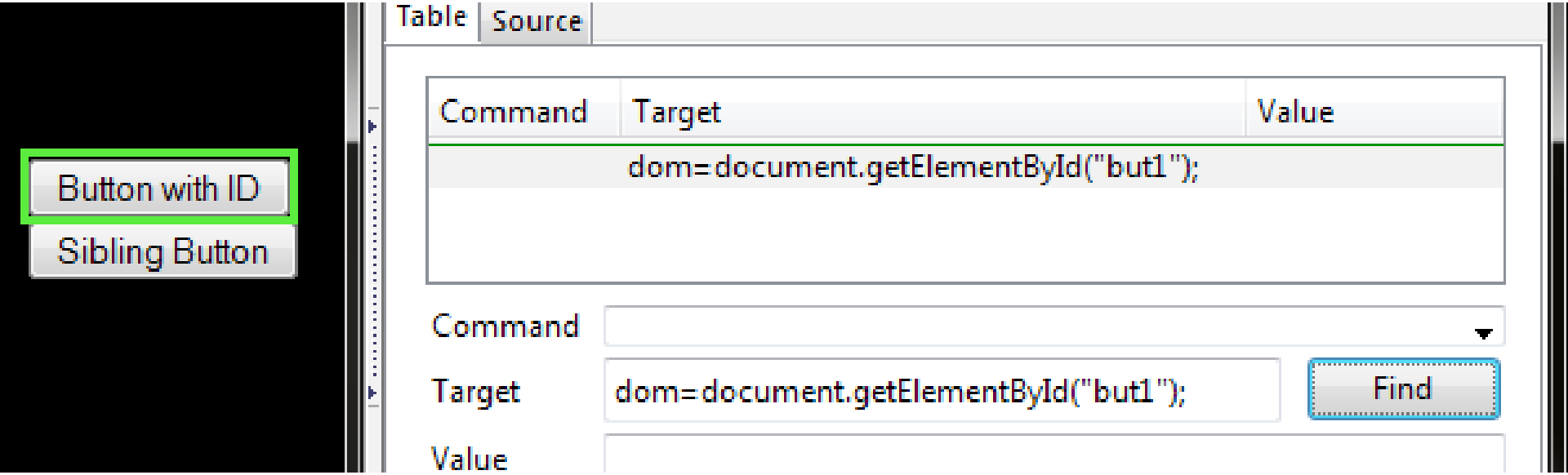
- Open http://pragmatic.bg/automation/example2.html



# Locating by JavaScript DOM



- Open http://pragmatic.bg/automation/example2.html



# Locating by Xpath (part 1)




- Finding elements by direct Xpath
  xpath=/html/body/div[2]/div[3]/input – NEVER!
- Using XPath to find the nth element of a type
  xpath=//div[2]//input[2]
- Using element attributes in XPath queries
  xpath=//div[@class='classname']
- Doing a partial match on attribute content
  xpath=//div[contains(@id,'time_')]//..//input
  xpath=//div[starts-with(@id,'time_') and not(contains(@class,'left'))]
- Finding an element by the text it contains
  xpath=//div[text()='inner text']

# Locating by Xpath (part 2)



- Using XPath Axis to find elements

xpath=//input[@id='but1']/following-sibling::input[@type='button']

| Axis name | Result |
|---|---|
| ancestor | Selects all the ancestors (parent, grandparent, and so on) of the element |
| descendant | Selects all the descendants (children, grandchildren, and so on) of the element |
| following | Selects all elements that follow the closing tab of the current element |
| following-sibling | Selects all the siblings after the current element |
| parent | Selects the parent of the current element |
| preceding | Selects all elements that are before the current element |
| preceding-Sibling | Selects all of the siblings before the current element |

# Locating by CSS selectors (part 1)



- . searches for class attribute an element
  - <input class="right  blue  big" />
- # searches for id attribute of an element
  -  
- + searches for direct sibling, just like following-sibling in xpath
- A > B will only select B that are direct children to A (that is, there are no other elements in between). just like / in xpath
- A B will select any B that are inside A, even if there are other elements between them. Just like // in xpath

# Locating by CSS selectors (part 2)



- Using CSS class attributes in CSS selectors
  css=div.centerdiv
- Using element IDs in CSS selectors
  css=div#divinthecenter or css=#divinthecenter
- Using child nodes to find the element
  css=div.leftdiv > input or css=div.leftdiv  input
- Using sibling nodes to find the element
  css=input#but1 + br + input
- Finding elements by their attributes
  css=input[value='chocolate']
  css=input[id='but1'][value='Button with ID'] - AND
  css=input[id='but1'],[value='Button with ID'] - OR

# Locating by CSS selectors (part 3)



- Partial matches on attributes

| Syntax | Description |
|---|---|
| ^= | Finds the item starting with the value passed in. This is the equivalent to the XPath `starts-with`. |
| $= | Finds the item ending with the value passed in. This is the equivalent to the XPath `ends-with`. |
| *= | Finds the item which matches the attribute that has the value that partially matches. This is equivalent to the XPath `contains`. |

In the XPath section of this lecture, we had a look at the XPath //div[contains(@id,'time_')] which has a dynamic ID. The equivalent CSS selector would be div[id^='time_'] or div[id*='time_'].

# Locating by CSS selectors (part 3)



- Finding an element by its inner text

It uses :contains('text') - NOT RECOMMENDED!
(probably already deprecated in CSS3)

- Finding the nth element with CSS

Type css=div.leftdiv input:nth-of-type(2)
This will find the same as
xpath=//div[contains(@class,'leftdiv')]//input[2]

# Selenium WebDriver locators



- Focusing on these locators using Selenium WebDriver:
  - Locating an element using the findElement method
  - Locating element(s) using findElements method
  - Locating links
  - Locating elements by tag name
  - Locating elements using CSS selectors
  - Locating elements using Xpath
  - Locating elements using text
  - Locating elements using advanced CSS selectors
  - Using jQuery selectors
  - Locating table rows and cells
  - Locating child elements in a table

# Locating using findElement method (part 1)



- By ID

driver.findElement(By.id(<element ID>))

- By name

driver.findElement(By.name(<element name>))

- By class name

driver.findElement(By.className(<element class>))

- By tag name

driver.findElement(By.tagName(<htmltagname>))

# Locating using findElement method (part 2)



- By link text

driver.findElement(By.linkText(<linktext>))

- By partial link text

driver.findElement(By.partialLinkText(<linktext>))

- By CSS

driver.findElement(By.cssSelector(<css selector>))

- By xPath

driver.findElement(By.xpath(<xpath query expression>))

# Some findElement() examples (part 1)



- By ID:

```
<form name="loginForm">
<label for="username">UserName: </label>
<input type="text" id="username" /><br/>
<label for="password">Password: </label>
<input type="password" id="password" /><br/>
<input name="login" type="submit" value="Login" />
</form>
```

```
WebElement username = driver.findElement(By.id("username"));
WebElement password = driver.findElement(By.id("password"));
```

# Some findElement() examples (part 2)



- By name attribute:

```
<form name="loginForm">
<label for="username">UserName: </label>
<input type="text" name="username" /><br/>
<label for="password">Password: </label>
<input type="password" name="password" /><br/>
<input name="login" type="submit" value="Login" />
</form>
```

```
WebElement username = driver.findElement(By.name("username"));
WebElement password = driver.findElement(By.name("password"));
```

# Some findElement() examples (part 3)



- By class attribute:

```
<form name="loginForm">
<label for="username">UserName: </label>
<input type="text" class="username" /></br>
<label for="password">Password: </label>
<input type="password" class="password" /></br>
<input name="login" type="submit" value="Login" />
</form>
```

```
WebElement username =
    driver.findElement(By.className("username"));

WebElement password =
    driver.findElement(By.className("password"));
```

# Some findElement() examples (part 4)



- WebElement class supports find methods that find child elements:

```
WebElement div = driver.findElement(By.id("div1"));
WebElement topLink = div.findElement(By.linkText("top"));
```

- You can also a use a shortcut method in the following way:

```
WebElement topLink =
    driver.findElement(By.id("div1")).findElement(By.linkText("top"));
```

# Locating elements using findElements() (part 1)



- Selenium WebDriver provides the findElements() method, which enables the acquisition of a list of elements matching the specified search criteria. This method is useful when we want to work with a group of similar elements. For example, we can get all the links displayed on a page or get rows from a table, and so on.

# Locating elements using findElements() (part 2)



```
@Test
public void testFindElements()
{
        //Get all the links displayed on Page
        List<WebElement> links = driver.findElements(By.tagName("a"));

        //Verify there are four links displayed on the page
        assertEquals(4, links.size());

        //Iterate though the list of links and print
        //target for each link
        for(WebElement link : links)
        System.out.println(link.getAttribute("href"));
}
```

# Locating links

- Finding a link by its text

```
WebElement gmailLink = driver.findElement(By.linkText("GMail"));
assertEquals("http://mail.google.com/", gmailLink.getAttribute("href"));
```

- Finding a link by partial text

```
WebElement inboxLink = driver.findElement(By.partialLinkText("Inbox"));
System.out.println(inboxLink.getText());
```

# Locating elements by tag name



- Let's assume you have a single button element on a page. You can locate this button by using its tag in the following way:

```
WebElement loginButton = driver.findElement(By.tagName("button"));
loginButton.click();
```

- Take another example where we want to count how many rows are displayed in <table>. We can do this in the following way:

```
WebElement table = driver.findElement(By.id("summaryTable"));
List<WebElement> rows = table.findElements(By.tagName("tr"));
assertEquals(10, rows.size());
```

# Locating elements using CSS selectors (part 1)



- Finding elements with absolute path

  CSS absolute paths refer to the very specific location of the element considering its complete hierarchy in the DOM. Here is an example where the Username Input field is located using the absolute path. While providing absolute path, a space is given between the elements.

```
WebElement userName = driver.findElement(By.cssSelector("html body div div form input"));
```

OR

```
WebElement userName = driver.findElement(By.cssSelector("html > body > div > div > form > input"));
```

What do you think are the limitations of this strategy?

# Locating elements using CSS selectors (part 2)



- Finding elements with relative path

With relative path we can locate an element directly, irrespective of its location in the DOM. For example, we can locate the Username Input field in the following way, assuming it is the first <input> element in the DOM:

```
WebElement userName = driver.findElement(By.cssSelector("input"));
```

# Locating elements using CSS selectors (part 3)



- Finding elements using the Class selector

While finding elements using the CSS selector, we can use the Class attribute to locate an element. This can be done by specifying the type of HTML tag, then adding a dot followed by the value of the class attribute in the following way:

```
WebElement loginButton = driver.findElement(By.cssSelector("input.login"));
```

```
//This method is similar to the className() locator method.
WebElement loginButton = driver.findElement(By.cssSelector(".login"));
```

# Locating elements using CSS selectors (part 4)



- Finding elements using ID selector

  We can locate the element using the IDs assigned to elements. This can be done by specifying the type of HTML tag, then entering a hash followed by the value of the Class attribute, as shown:

```
WebElement userName =
driver.findElement(By.cssSelector("input#username"));
```

Shortcut:

```
WebElement userName =
driver.findElement(By.cssSelector("#username"));
```

This will return the username <input> element using its id attribute.

# Locating elements using CSS selectors (part 5)



- Finding elements using attributes selector

  Finding elements using attributes selector Apart from the class and id attributes, CSS selectors also enable the location of elements using other attributes of the element. In the following example, the Name attribute is used to locate an <input> element.

```
//Using the name attribute to locate an element is similar to the name() locator
//method of the By class.
WebElement userName =
driver.findElement(By.cssSelector("input[name=username]"));

WebElement previousButton =
driver.findElement(By.cssSelector("img[alt='Previous']"));

WebElement previousButton = driver.findElement(By.cssSelector("input[t
ype='submit'][value='Login']"));
```

# Locating elements using CSS selectors (part 6)



- Finding elements using Attributes Name Selector

  This strategy is a bit different from the earlier strategy where we want to locate elements based on only the specific attribute defined for them but not attribute values. For example, we want to lookup all the <img> elements which have alt attribute specified.

```
List<WebElement> imagesWithAlt =
driver.findElements(By.cssSelector("img[alt]"));
```

  Boolean not() pseudo-class can also be used to locate elements not matching the specified criteria. For example, to locate all the <img> elements that do not have the alt attribute, the following method can be used:

```
List<WebElement> imagesWithoutAlt =
driver.findElements(By.cssSelector("img:not([alt])"));
```

# Locating elements using CSS selectors (part 7)



- Performing partial match on attribute values

| Syntax | Example | Description |
|---|---|---|
| ^= | `input[id^='ctrl']` | Starting with:<br>For example, if the ID of an element is `ctrl_12`, this will locate and return elements with `ctrl` at the beginning of the ID. |
| $= | `input[id$='_userName']` | Ending with:<br>For example, if the ID for an element is `a_1_userName`, this will locate and return elements with `_userName` at the end of the ID. |
| *= | `Input[id*='userName']` | Containing:<br>For example, if the ID of an element is `panel_login_userName_textfield`, this will use the `userName` part in the middle to match and locate the element. |

# Locating elements using Xpath (part 1)



- XPath, the XML path language, is a query language for selecting nodes from an XML document. All the major browsers support XPath as HTML pages are represented as XHTML documents in DOM.
- The XPath language is based on a tree representation of the XML document and provides the ability to navigate around the tree, selecting nodes using a variety of criteria.
- Selenium WebDriver supports XPath for locating elements using XPath expressions or queries.
- The least preferred strategy due to low performance.
- With XPath we can search elements backward or forward in the DOM hierarchy while CSS works only in a forward direction. This means that with XPath we can locate a parent element using a child element.

# Locating elements using Xpath (part 2)



- Finding elements with absolute path

Similar to CSS absolute paths, XPath absolute paths refer to the very specific location of the element, considering its complete hierarchy in the DOM. Here is an example where Username Input field is located using the absolute path. While providing absolute path a space is given between the elements.

```
WebElement userName =
driver.findElement(By.xpath("html/body/div/div/form/input"));
```

What do you think are the limitations of  this strategy?

# Locating elements using Xpath (part 3)



- Finding elements with relative path

  With relative path, we can locate an element directly irrespective of its location in the DOM. For example, we can locate the Username Input field in the following way, assuming it is the first <input> element in the DOM:

```
WebElement userName = driver.findElement(By.xpath("//input"));
```

# Locating elements using Xpath (part 4)



- Finding elements using index

  In the previous example, the XPath query will return the first <input> element that it finds in the DOM. There could be multiple elements matching the specified XPath query. If the element is not the first element, we can also locate the element by using its index in DOM. For example in our login form, we can locate the Password field which is the second <input> element on the page in the following way:

```
WebElement userName = driver.findElement(By.xpath("//input[2]"));
```

# Locating elements using Xpath (part 5)



- Finding elements using attributes values with Xpath
  Similar to CSS, we can also locate elements using their attribute values in XPath. In the following example, the Username field is located using the ID attribute:

```
WebElement userName =
driver.findElement(By.xpath("//input[@id='username']"));

WebElement previousButton =
driver.findElement(By.xpath("img[@alt='Previous']"));

WebElement previousButton =
driver.findElement(By.xpath("//input[@type='submit'][@value='Login']"));

WebElement previousButton = driver.findElement
(By.xpath("//input[@type='submit' or @value='Login']"));
```

# Locating elements using Xpath (part 6)



- Finding elements using attributes with Xpath

  This strategy is a bit different from the earlier strategy where we want to locate elements based only on the specific attribute defined for them but not attribute values. For example, we want to lookup all the <img> elements that have the alt attribute specified.

```
List<WebElement> imagesWithAlt =
        driver.findElements(By.xpath("//img[@alt]"));
```

# Locating elements using Xpath (part 7)



- Performing partial match on attribute values

| Syntax | Example | Description |
|---|---|---|
| `starts-with()` | `input[starts-with(@id,'ctrl')]` | Starting with:<br>For example, if the ID of an element is `ctrl_12`, this will locate and return elements with `ctrl` at the beginning of the ID. |
| `ends-with()` | `input[ends-with(@id,'_userName')]` | Ending with:<br>For example, if the ID of an element is `a_1_userName`, this will locate and return elements with `_userName` at the end of the ID. |
| `contains()` | `Input[contains(@id,'userName')]` | Containing:<br>For example, if the ID for an element is `panel_login_userName_textfield`, this will use the `userName` part in the middle to match and locate the element. |

# Locating elements using Xpath (part 8)



- Matching any attribute using a value

  For example, in the following XPath query, 'userName' is specified. XPath will check all the elements and their attributes to see if they have this value and return the matching element.

```
WebElement userName =
        driver.findElement(By.xpath("//input[@*='username']"));
```

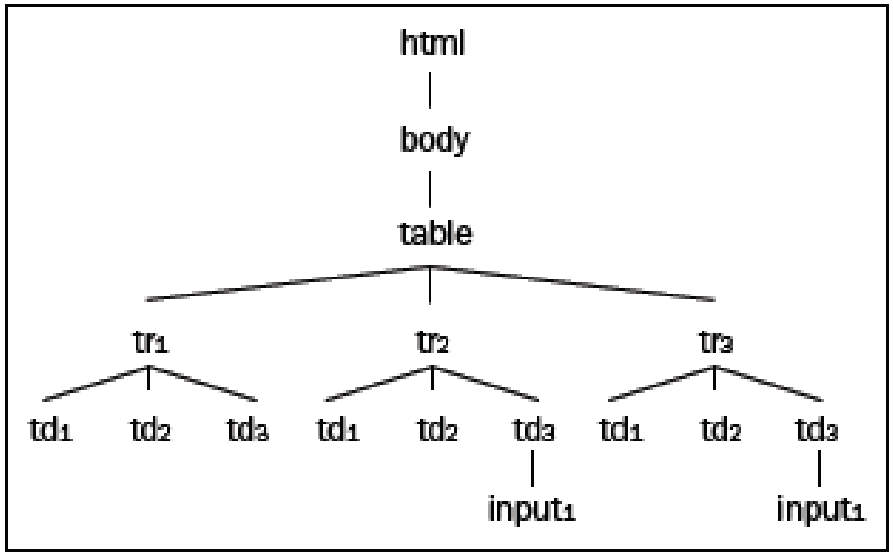
# Locating elements using Xpath (part 9)



- Locating elements with XPath axis

XPath axes help to locate elements based on the element's relationship with other elements in a document. Here are some examples for some common XPath axes used to locate elements from a <table> element. This can be applied to any other element structure from your application.

| Product | Price | Qty |
|---|---|---|
| Product 1 | $100 | 12 |
| Product 2 | $150 | 5 |



| Axis | Description | Example | Result |
|---|---|---|---|
| `ancestor` | Selects all ancestors (parent, grandparent, and so on) of the current node. | `//td[text()='Product 1']/ancestor::table` | This will get the table element. |
| `descendant` | Selects all descendants (children, grandchildren, and so on) of the current node. | `/table/descendant::td/input` | This will get the input element from the third column of the second row from the table. |
| `following` | Selects everything in the document after the closing tag of the current node. | `//td[text()='Product 1']/following::tr` | This will get the second row from the table. |
| `following-sibling` | Selects all siblings after the current node. | `//td[text()='Product 1']/following-sibling::td` | This will get the second column from the second row immediately after the column that has `Product 1` as the text value. |
| `preceding` | Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes, and namespace nodes. | `//td[text()='$150']/preceding::tr` | This will get the header row. |
| `preceding-sibling` | Selects all siblings before the current node. | `//td[text()='$150']/preceding-sibling::td` | This will get the first column of third row from the table. |

# Locating elements using text (part 1)



- You can use the *innerText* attribute (does not work with Firefox) or *textContent* attribute (for Firefox) in the following ways:

```
WebElement cell =
driver.findElement(By.cssSelector("td[innerText='Item 1']"));
```

```
WebElement cell =
driver.findElement(By.cssSelector("td[textContent='Item 1']"));
```

# Locating elements using text (part 2)



- Using XPath text function

  XPath provides the text() function which can be used to see if an element contains the specified text in the following way:

```
WebElement cell =
driver.findElement(By.xpath("//td[contains(text(),'Item 1')]"));
```

  Here we are using the contains function along with the text() function. The text() function returns the complete text from the element and the contains() function checks for the specific value that we have mentioned.

# Locating elements using text (part 3)



- Finding elements using exact text value in XPath

  With XPath, elements can be located by exact text value in the following way:

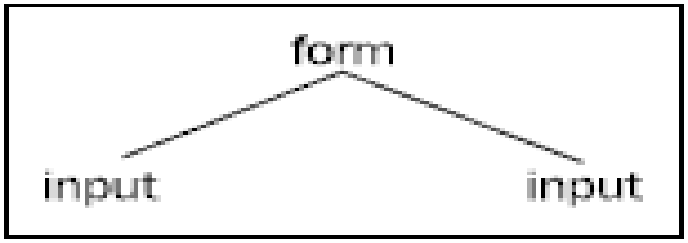
```
WebElement cell = driver.findElement(By.xpath("//td[.='Item 1']"));
```

  This will locate the <td> element matching with exact text.

# Locating elements using advanced CSS selectors (part 1)

- Finding child elements

  For example, to locate the Username Field in the login form, we can use the following selector. Here, > is used denote the parent and child relationship.




```
WebElement userName =
driver.findElement(By.cssSelector("form#loginForm > input"));
```

Similarly the nth-child() method can be used in the following way:

```
WebElement userName = driver.findElement
(By.cssSelector("form#loginForm :nth-child(2)"));
```

# Locating elements using advanced CSS selectors (part 2)



- Finding child elements

| Pseudo-class | Example | Description |
|---|---|---|
| `:first-child` | `form#loginForm :first-child` | This will locate the first element under the form, that is, the label for username. |
| `:last-child` | `form#loginForm :last-child` | This will locate the last element under the form, that is, the **Login** button. |
| `:nth-child(2)` | `form#loginForm :nth-child(2)` | This will locate the second child element under the form, that is, the **Username** field. |

- Finding sibling elements

| | | |
|---|---|---|
| `p + p` | `div#top5 > p + p` | Immediately following sibling. This will locate `Description for Product 2.` |
| `p + * + p` | `div#top5 > p + * + p` | Following sibling with one intermediary. This will locate `Description for Product 3.` |

# Locating elements using advanced CSS selectors (part 3)



- Using user action pseudo-classes

  Using the user action :focus pseudo-class, we can locate the element which has current input focus in the following way:

```
WebElement productDescription =
driver.findElement(By.cssSelector("input:focus"));
```

  This will locate any element that currently has the input focus. You can also locate elements using :hover and :active pseudo-classes.

# Locating elements using advanced CSS selectors (part 4)

- Using UI state pseudo-classes

  Using UI state pseudo-classes, we can locate elements for various states such as control is enabled, disabled, and checked. The following table describes these in detail:

| Pseudo-class | Example | Description |
|---|---|---|
| `:enabled` | `input:enabled` | This will locate all the elements that are enabled for user input. |
| `:disabled` | `input:enabled` | This will locate all the elements that are disabled for user input. |
| `:checked` | `input:checked` | This will locate all the elements (checkboxes) that are checked. |

# Locating table rows and cells



- Lets take a detailed look in the TableExample.java and see how it works

# Locating child elements in a table (part 1)



## Your Shopping Basket

You have selected the following products so far:

| Product | Price incl. VAT | Quantity | Value incl. VAT |
|---|---|---|---|
|  Large Bags James Wellbeloved + JWB USB Stick Free! - Junior Large Breed Turkey & Rice (15 kg) 315777.29 | Was £47.89 **Now £42.90** | 1 | £42.90 |
| **Subtotal** | | | **£42.90** |
| **Discount** Only one coupon can be accepted with each order. | **10% New Customer Discount** change coupon | | **-£4.29** |
| **Shipping Fees** Please choose a delivery country so that we can calculate your shipping costs for you. | Great Britain | | £0.00 |
| | **Grand Total** All prices include tax. | | **£38.61** |

back to shop

proceed to checkout

# Locating child elements in a table (part 2)



GMail | Inbox (10 New E-mails)

| User Name | E-mail | Access |
|---|---|---|
| Nash | Nash@test.com | Admin ☑ Content Manager ☐ Browser ☐ |
| John | John@test.com | Admin ☐ Content Manager ☐ Browser ☑ |

```
<tr>
        <td>Nash</td>
        <td><a href="mailto:nash@test.com">Nash@test.com</a></td>
        <td>
                <div>
                        <label for="user128_admin">Admin</label>
                        <input type="checkbox" id="user128_admin"
                                checked="true"/>
                        <label for="user128_cm">Content Manager</label>
                        <input type="checkbox" id="user128_cm"/>
                        <label for="user128_browser">Browser</label>
                        <input type="checkbox" id="user128_browser"/>
                </div>
        </td>
</tr>
```

# Locating child elements in a table (part 3)



- The checkbox has dynamic IDs that we cannot correlate to a user. However, we can deal with such issues by using CSS selectors or XPath. In this example, we want to grant user Nash with admin access. This can be done using CSS selectors in the following way:

```
WebElement adminCheckBox = driver.findElement
(By.cssSelector("td:contains('Nash')+td+td>div>label:contains('Admin')+input"));

adminCheckBox.click();
```

- with Xpath:

```
WebElement adminCheckBox = driver.findElement
(By.xpath("//td[contains(text(),'Nash')]/followingsibling::td/descendant::div/label
[contains(text(),'Admin')]/following-sibling::input"));

adminCheckBox.click();
```

# How it works...



Parent, child, and sibling in CSS or XPath axes become a great help in correlating users with roles and developing a generic locator strategy for this feature. In simple terms, these strategies help to locate elements based on the element's relationship with other elements in a document.

Coming back to the problem, first we need to find a unique way to identify a user in the table. For this, we will locate a cell which contains username. We will locate this cell using its inner text in the following way:

| CSS | XPath |
|---|---|
| `td:contains('Nash')` | `//td[contains(text(),'Nash')]` |

Next, we need to find the cell which contains the child elements. This is the second cell from the cell containing username.

| CSS | XPath |
|---|---|
| `td:contains('Nash')+td+td` | `//td[contains(text(),'Nash')]/following-sibling::td/` |

In the next step, we need to locate the label with the correct option. The next sibling of this label will be the checkbox we are looking for.

| CSS | XPath |
|---|---|
| `td:contains('Nash')+td+td>div>label:contains('Admin')+input` | `//td[contains(text(),'Nash')]/following-sibling::td/descendant::div/label[contains(text(),'Admin')]/following-sibling::input` |

# Questions?

Questions?