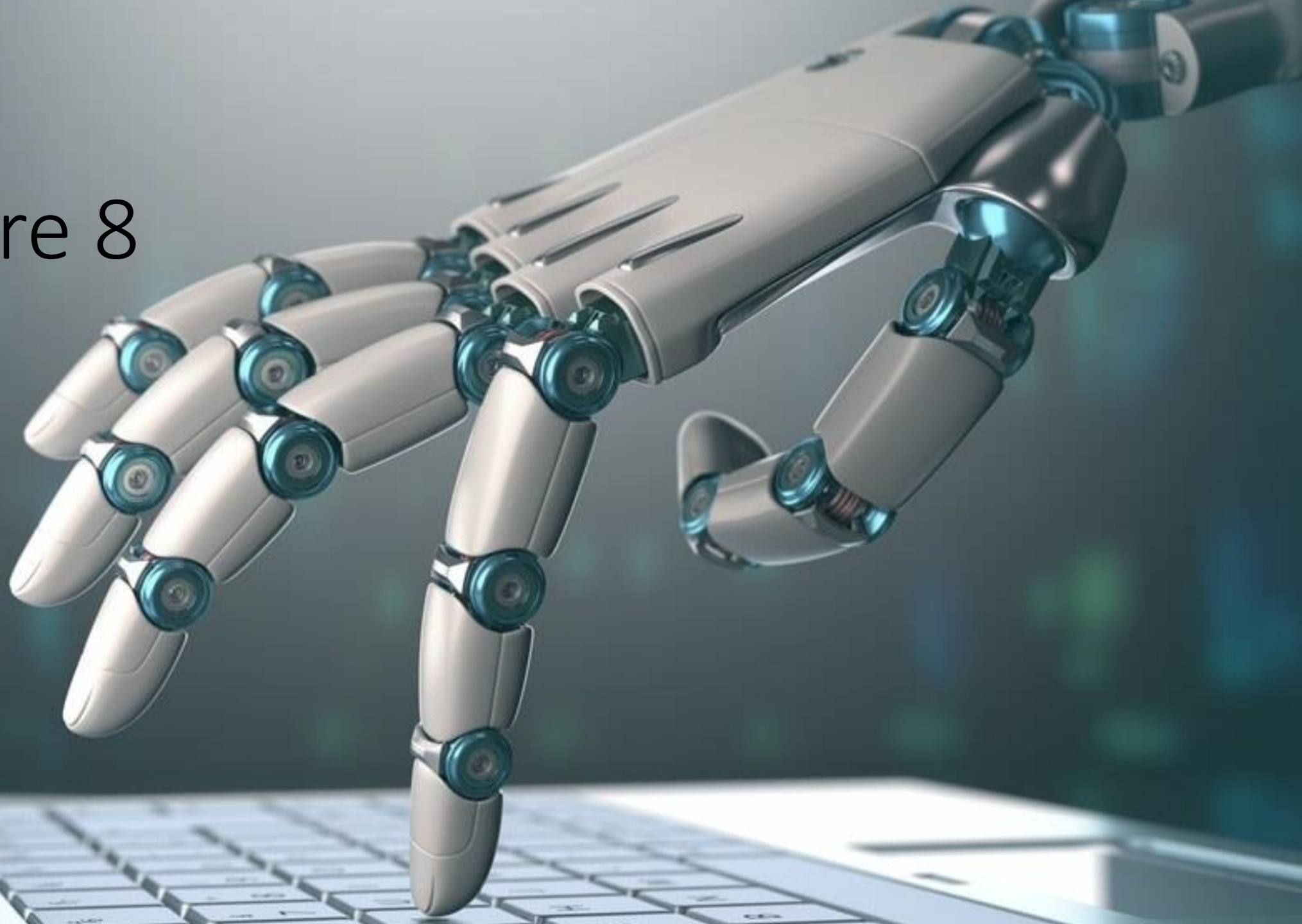


# Lecture 8



# Agenda

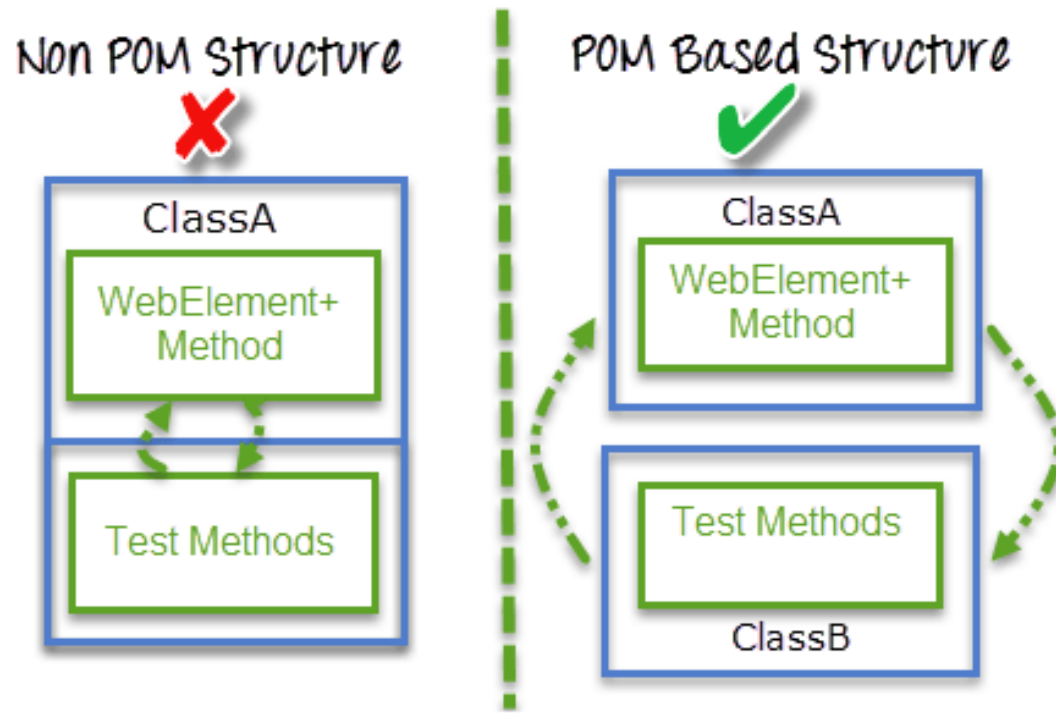
- **Page object model**
- Advanced locators css
- Advanced locators XPath
- Practice

# What is Page Object Model (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.

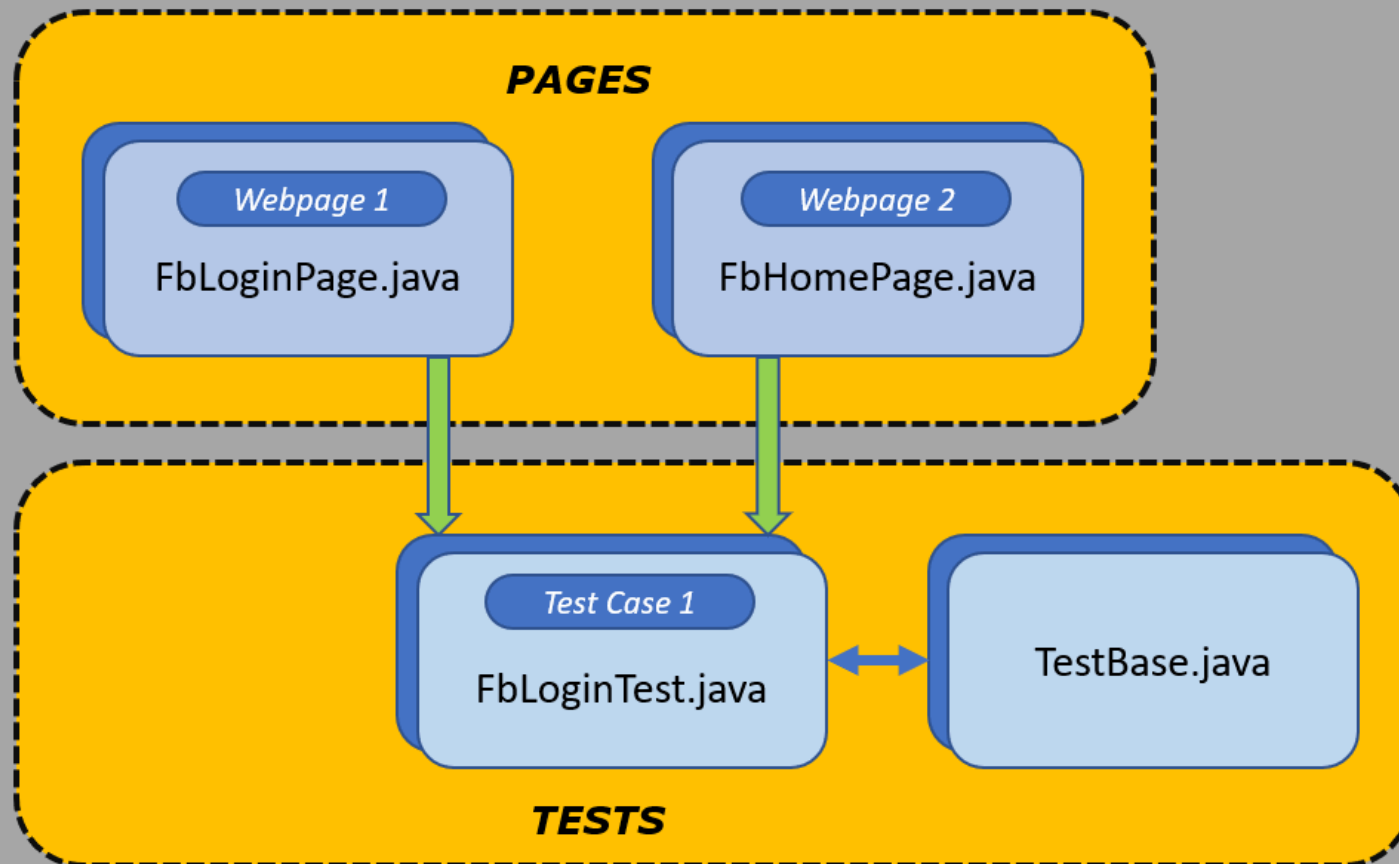
# Why POM?

This approach is called **Page Object Model(POM)**. It helps make the code **more readable, maintainable, and reusable**.



# How it looks like

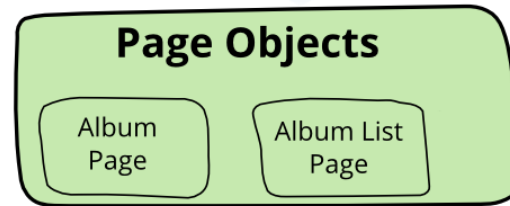
## Page Object Model Design Pattern



# How it looks like

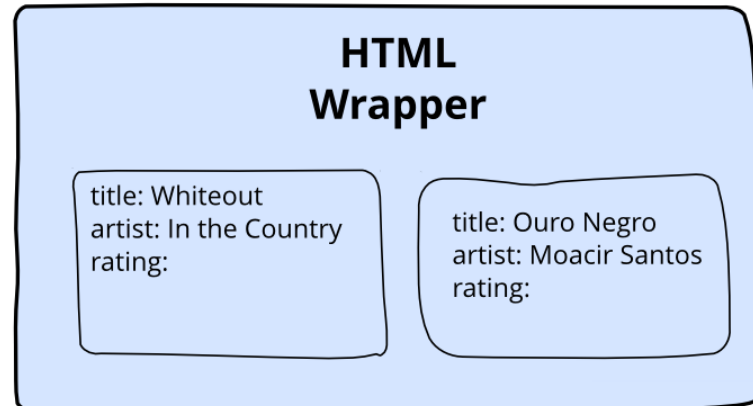
this API is about  
the application

`selectAlbumWithTitle()`  
`getArtist()`  
`updateRating(5)`

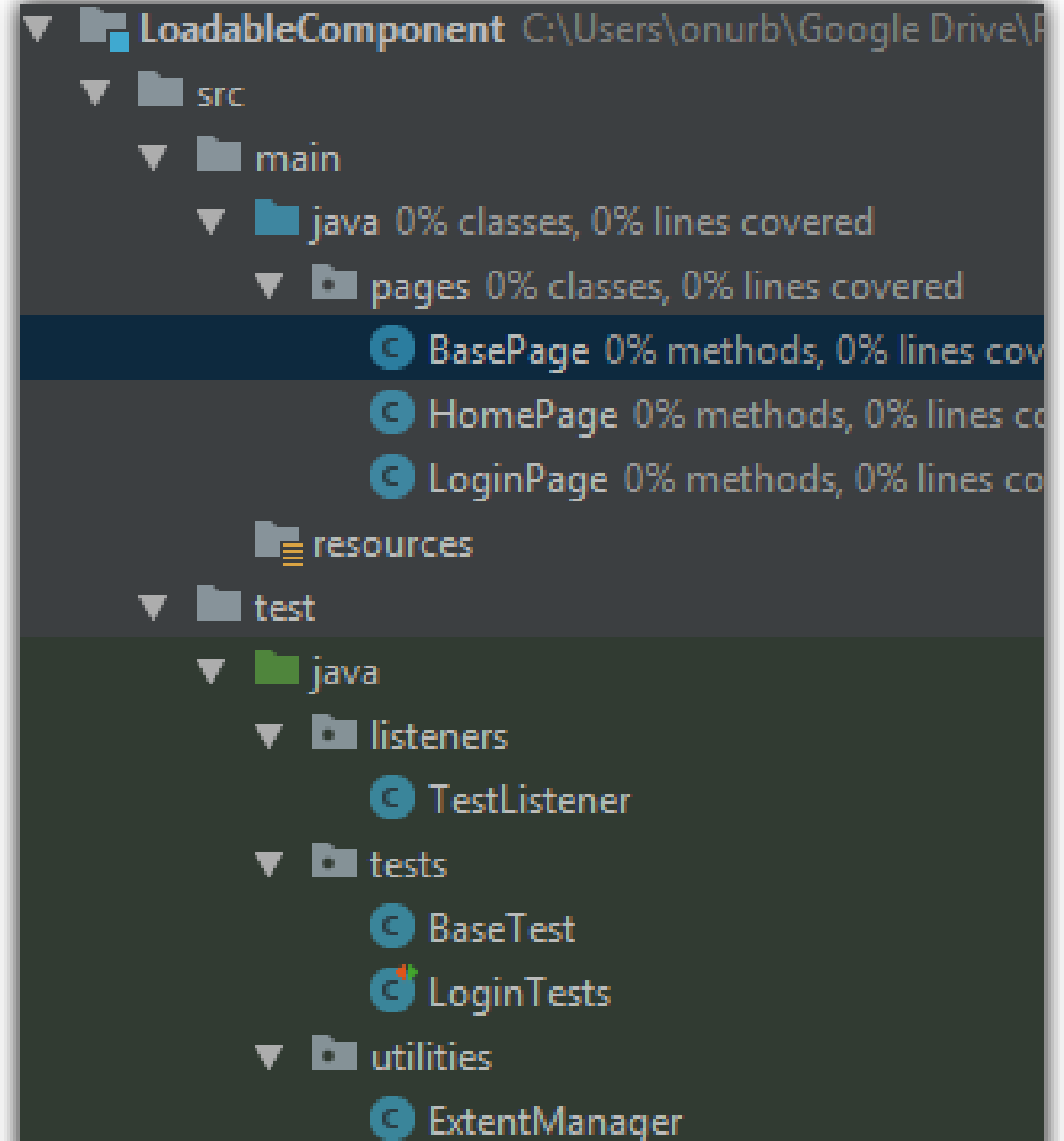


this API is  
about HTML

`findElementsWithClass('album')`  
`findElementsWithClass('title-field')`  
`getText()`  
`click()`  
`findElementsWithClass('ratings-field')`  
`setText(5)`



# Page Object Example



# Page Object Example

```
public class MainPage {  
  
    private static WebDriver driver;  
  
    private By minPrice = By.name("topt[8][min]");  
    private By maxPrice = By.name("topt[8][max]");  
    private By minYear = By.name("topt[18][min]");  
    private By maxEngine = By.name("topt[15][max]");  
    private By selectColor = By.name("opt[17]");  
  
    public void navigateToCategory(String category) { driver.findElement(By.linkText(category)).click(); }  
  
    public void searchByMinAndMaxPrice(String min, String max) {  
        driver.findElement(minPrice).sendKeys(min);  
        driver.findElement(maxPrice).sendKeys(max);  
    }  
}
```



# POM sample

```
public class Guru99Login {  
    WebDriver driver;  
    By user99GuruName = By.name("uid");  
    By password99Guru = By.name("password");  
    By titleText = By.className("barone");  
    By login = By.name("btnLogin");  
  
    public Guru99Login(WebDriver driver){  
        this.driver = driver;  
    }  
    //Set user name in textbox  
    public void setUsername(String strUserName){  
        driver.findElement(user99GuruName).sendKeys(strUserName);  
    }  
}
```

1 Page class in object repository

Find Web Element

Performing operation on Web element

2

3

# Issue with locating locator by Webdriver

```
driver.manage().window().maximize();
```

```
JavascriptExecutor js = (JavascriptExecutor) driver;  
js.executeScript("script: window.scrollTo(0,500)");
```

# Selector Debugging

### Registration page

Name

Email

Password

The screenshot shows the Chrome DevTools interface. The top bar includes tabs for Elements, Console, Sources, and Network. The Elements panel is open, displaying the DOM tree. The selected element is the first text input field, which has the following attributes: `<input type="text" id="regUsername" name="username" class="username" autocomplete="off">`. The Styles pane is open, showing the default user agent styles for the `input` element, including `display: block;` and `margin: 8px;`. A visual representation of the box model is shown on the right, with a blue box for the element (884 x 517) and an orange box for the margin (8px). The bottom of the interface shows the Console and What's New tabs.

```
<html>
  <head>...</head>
  <body> == $0
    <div>...</div>
    <div id="register">
      <h3>Registration page</h3>
      <form method="post" action name="register">
        <label>Name</label>
        <input type="text" id="regUsername" name="username" class="username" autocomplete="off">
        <label>Email</label>
        <input type="text" id="regEmail" name="emailid" class="emailid" autocomplete="off">
        <label>Password</label>
        <input type="password" id="regPassword" name="userpassword" class="password" autocomplete="off">
        <input type="submit" class="button style1 style2 style3" name="submitregistrationform" value="Register">
      </form>
    </div>
  </body>
</html>
```

html body

input[class="username"] 1 of 1 ^ v Cancel

Styles Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls +

```
element.style {
}

body {
  user agent stylesheet
  display: block;
  margin: 8px;
}
```

margin 8  
border -  
padding -  
884 x 517  
8 8

Console What's New x

Highlights from the Chrome 72 update

# Agenda

- Page object model
- **Advanced locators css**
- Advanced locators XPath
- Practice

# XPATH VS CSS PATH CHEAT SHEET

DESCRIPTION	XPATH	CSS PATH
Direct Child	<code>//div/a</code>	<code>div &gt; 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>

# Locating First Child Element by Element Name

## Syntax:

CSS-of-Parent-Element **elementName:first-child**

Locating First element using **:first-child** selector.

- > form **input:first-child**

# Locating First Child Element by Element Name

The screenshot displays a web browser window with the URL `https://skptricks.github.io/learncoding/selenium-demo/login%20registration%20page/Register.html`. The page shows a "Registration page" with input fields for "Email", "Password", and a "Register" button. The Chrome DevTools interface is open, showing the DOM tree on the left and the Styles pane on the right. The DOM tree highlights the `<input type="text" id="regUsername" name="username" class="username" autocomplete="off">` element, which is the first child of the `form` element. The Styles pane shows the default user agent styles for the `input` element. Below the DevTools interface, a blue banner titled "Highlights from the Chrome 65 update" lists features like "Local overrides", "Changes tab", "New accessibility tools", "New audits", and "Code stepping updates". A large graphic for "new 65" is also visible.

Registration page

Email

Password

Register

```
<html>
  <#shadow-root (open)>
    <head>...</head>
    <body>
      <div>...</div>
      <div id="register">
        <h3>Registration page</h3>
        <form method="post" action name="register">
          <input type="text" id="regUsername" name="username" class="username" autocomplete="off">
          <label>Email</label>
          <input type="text" id="regEmail" name="emailid" class="emailid" autocomplete="off">
          <label>Password</label>
          <input type="password" id="regPassword" name="userpassword" class="password" autocomplete="off">
          <input type="submit" class="button style1 style2 style3" name="submitregistrationform" value="Register">
        </form>
      </div>
    </body>
  </html>
```

form input:first-child 1 of 1

Highlights from the Chrome 65 update

- Local overrides  
Override network requests and serve local resources instead.
- Changes tab  
Track changes that you make locally in DevTools via the Changes tab.
- New accessibility tools  
Inspect the accessibility properties and contrast ratio of elements.
- New audits  
New performance audits, a whole new category of SEO audits, and more.
- Code stepping updates  
Reliably step into web worker and asynchronous code.

new 65

# Locating Last Child Element by Element Name

## Syntax:

`<CSS of Parent Element> <space> <elementName>:last-child`

Locating last element using `:last-child` selector.

- `> form input:last-child`



# Locating First Child Element by Element Name

[Login](#)

Registration page

Email

Password

The screenshot shows the Chrome DevTools interface. The 'Elements' panel on the left displays the DOM tree. The 'register' div is expanded, showing a form with three input fields and a submit button. The first child element of the form, an input with id 'regUsername' and name 'username', is selected and highlighted in blue. The breadcrumb at the bottom of the Elements panel reads: 'html > body > div#register > form > input#regUsername.username'. The 'Styles' panel on the right shows the default user agent styles for the selected input element. Below the DevTools window, a blue banner titled 'Highlights from the Chrome 65 update' lists several new features: Local overrides, Changes tab, New accessibility tools, New audits, and Code stepping updates. To the right of the text is a graphic for Chrome 65, featuring the Chrome logo and the text 'new 65'.

```
<html>
  #shadow-root (open)
  <head>...</head>
  <body>
    <div>...</div>
    <div id="register">
      <h3>Registration page</h3>
      <form method="post" action name="register">
        <input type="text" id="regUsername" name="username" class="
        "username" autocomplete="off"> == $0
        <label>Email</label>
        <input type="text" id="regEmail" name="emailid" class="
        "emailid" autocomplete="off">
        <label>Password</label>
        <input type="password" id="regPassword" name="
        "userpassword" class="password" autocomplete="off">
        <input type="submit" class="button style1 style2 style3"
        name="submitregistrationform" value="Register">
      </form>
    </div>
  </body>
</html>
```

html > body > div#register > form > input#regUsername.username

form input:last-child 1 of 1 Cancel

### Highlights from the Chrome 65 update

- Local overrides**  
Override network requests and serve local resources instead.
- Changes tab**  
Track changes that you make locally in DevTools via the Changes tab.
- New accessibility tools**  
Inspect the accessibility properties and contrast ratio of elements.
- New audits**  
New performance audits, a whole new category of SEO audits, and more.
- Code stepping updates**  
Reliably step into web worker and asynchronous code.

# Locating By nth-child()

```
driver.findElement(By.cssSelector("ul > li:nth-child(1)")); >> home  
driver.findElement(By.cssSelector("ul > li:nth-child(2)")); >> posts  
driver.findElement(By.cssSelector("ul > li:nth-child(3)")); >> events
```

# Reminder

Selector	Example	Example description
<u>.class</u>	.intro	Selects all elements with class="intro"
<u>#id</u>	#firstname	Selects the element with id="firstname"
<u>*</u> <u>_</u>	*	Selects all elements
<u>element</u>	p	Selects all <p> elements
<u>element,element</u>	div, p	Selects all <div> elements and all <p> elements
<u>element element</u>	div p	Selects all <p> elements inside <div> elements
<u>element&gt;element</u>	div > p	Selects all <p> elements where the parent is a <div> element

[https://www.w3schools.com/csSref/css\\_selectors.asp](https://www.w3schools.com/csSref/css_selectors.asp)

# Locating Elements by its Attributes

## Web Element :

### Code

```
<input type="text" id="regUsername" name="username" class="username" autocomplete="off">
```

Following are the possible way to uniquely identify the "Name" Field.

- > input[id="regUsername"]
- > input[name="username"]
- > input[class="username"]

### Syntax:

element\_name[<attribute\_name>='<value>']

# Locating Elements by ID

Web Element :

Code

```
<input type="text" id="regUsername" name="username" class="username" autocomplete="off">  
<input type="text" id="regEmail" name="emailid" class="emailid" autocomplete="off">  
<input type="password" id="regPassword" name="userpassword" class="password" autocomplete="off">
```

Locating the **"Name"**, **"Email"** and **"Password"** field by its ID using below CSS Selector :

1. input#regUsername OR #regUsername
2. input#regEmail OR #regEmail
3. input#regPassword OR #regPassword

**Syntax:**

element\_name#id\_value

# Locating Elements by Class

## Web Element :

### Code

```
<input type="text" id="regUsername" name="username" class="username" autocomplete="off">  
<input type="text" id="regEmail" name="emailid" class="emailid" autocomplete="off">  
<input type="password" id="regPassword" name="userpassowrd" class="password" autocomplete="off">
```

Locating the **"Name"**, **"Email"** and **"Password"** field by its Class using below CSS Selector :

1. input.username OR .username
2. input.emailid OR .emailid
3. input.password OR .password

### Syntax:

element\_name.class\_value

# Locating elements by multiple classes

Code

```
<input type="submit" class="button style1 style2 style3" name="submitregistrationform" value="Regi
```

## Syntax:

elementName.class1.class2.class3 or .class1.class2.class3

Locating the input "**Button**" field by its multiple class using below CSS Selector :

1. input.button
2. input.button.style1
3. input.button.style1.style2
4. input.button.style1.style2.style3
5. .button.style1.style2.style3
6. .style1.style2.style3
7. .style2.style3
8. .button

# Locating Elements by Class and Attribute

Code

```
<input type="text" id="regUsername" name="username" class="username" autocomplete="off">
```

## Syntax:

elementName.class[attributeName='value']

Locating the "**Name**" field by its class and attribute element using below CSS Selector :

input.username[name="username"]

OR

.username[name="username"]



# Locating Elements with more than one Element

Code

```
<input type="text" id="regUsername" name="username" class="username" autocomplete="off">
```

**Syntax:**

elementName[attribute1='value1'][attribute2='value2']...[attributeN='valueN']

Locating the "**Name**" field with more than one element using below CSS Selector :

input[id="regUsername"][name="username"]

OR

input[id="regUsername"][name="username"][class="username"]

OR

[name="username"][class="username"]

OR

[id="regUsername"][name="username"][class="username"]

# Locating Elements by Prefix of the (starts-with) Attribute Value

Code

```
<input type="text" id="regUsername" name="username" class="username" autocomplete="off">
```

**Syntax:**

elementName[attributeName^='prefix-of-the-value']

Locating "**Name**" Field using start with prefix attribute value using CSS Selector :

input[name^="userna"]

OR

input[id^="regU"]

**NOTE :** ^= Match a prefix (This indicate the match prefix characters form attribute value)

# Locating Elements by Suffix of the (ends-with) Attribute Value

Code

```
<input type="text" id="regUsername" name="username" class="username" autocomplete="off">
```

## Syntax :

elementName[attributeName\$='suffix-of-the-value'] or \*[attributeName\$='suffix-of-the-value']

Locating "**Name**" Field using end with prefix attribute value using CSS Selector :

input[id\$="name"]

OR

input[class\$="name"]

**NOTE :** \$= Match a suffix (This indicate the match suffix characters form attribute value)

# Locating Elements containing part of the Attribute Value

Code

```
<input type="text" id="regUsername" name="username" class="username" autocomplete="off">
```

## Syntax:

elementName[attributeName\*='part-of-the-value']

Locating the Name Field with help of substring match :

input[id\*="ser"]

OR

input[class\*="ser"]

**NOTE :** \*= Match a substring (This indicate the match substring characters form attribute value)

# Agenda

- Page object model
- Advanced locators css
- **Advanced locators XPath**
- Practice

# Advanced Xpath selectors

## Relative XPath

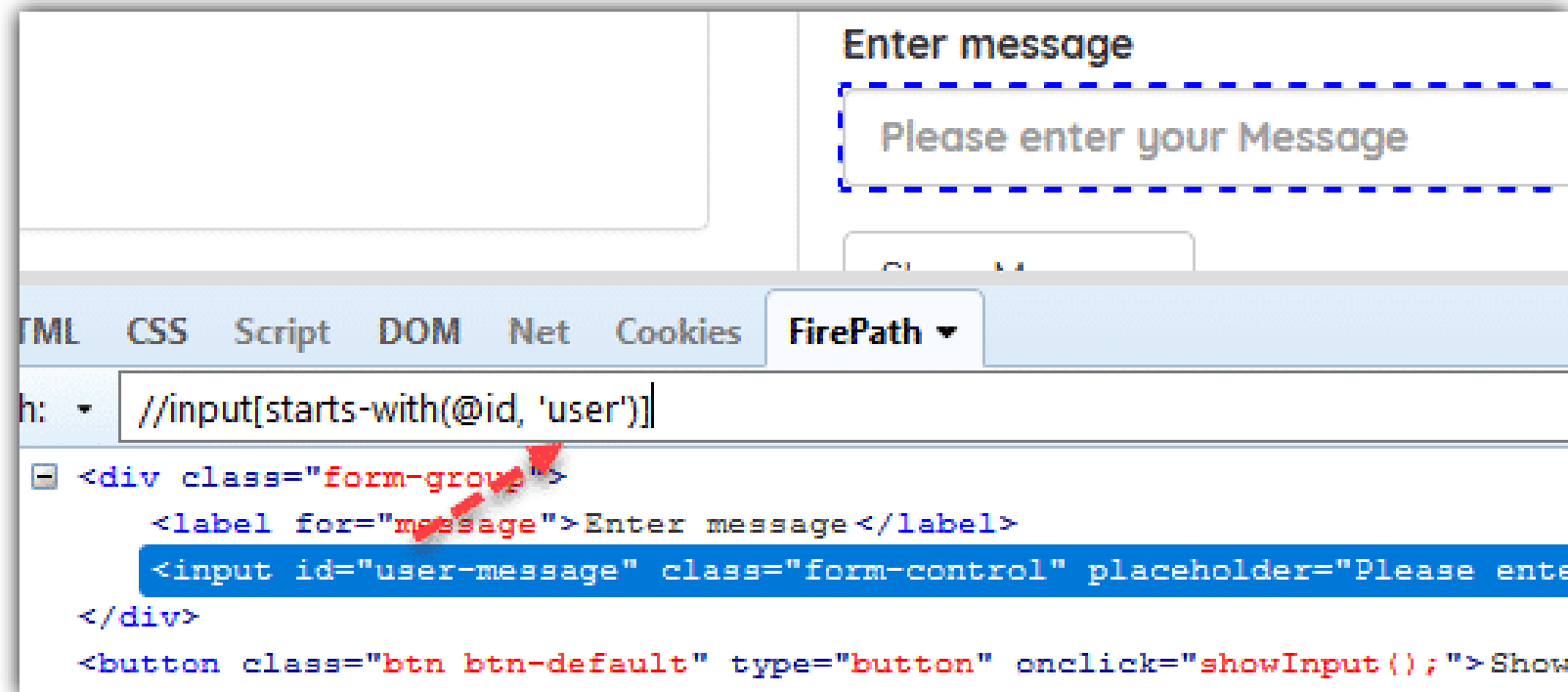
- Starts from the middle of the HTML DOM.
- Starts with a double slash “//” that means it can start to search anywhere in the DOM structure.
- Shorter than Absolute XPath.
- Less fragile.

**Example:** `//div[@class='form-group']/input[@id='user-message']`

# Locating Elements by Prefix of the (starts-with) Attribute Value

**Syntax:** `//tag[starts-with(@attribute, 'value')]`

**Example:** `//input[starts-with(@id, 'user')]`



The screenshot shows a web browser window with a form. The form has a label "Enter message" and a text input field with the placeholder text "Please enter your Message". The input field is highlighted with a dashed blue border. Below the browser window, the FirePath tool is open, showing the XPath query `//input[starts-with(@id, 'user')]` in the search bar. The results pane shows the following HTML structure:

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

A red arrow points from the `id="user-message"` attribute in the HTML to the `'user'` value in the XPath query.

# Locating Elements containing part of the Attribute Value

Syntax: `//tag[contains(@attribute, 'value')]`

Example: `//input[contains(@id, 'er-messa')]`



The screenshot shows a web browser interface with a form titled "Enter message". The form contains a text input field with the placeholder text "Please enter your Message" and a button labeled "Submit". The input field is highlighted with a dashed blue border. Below the browser window, the FirePath DOM inspector is open, showing the XPath `./input[contains(@id, 'er-messa')]` and the corresponding HTML structure. The HTML structure is as follows:

```
<form id="get-input" method="post">
  <div class="form-group">
    <label for="message">Enter message</label>
    <input id="user-message" class="form-control" placeholder="Please
  </div>
  <button class="btn btn-default" type="button" onclick="showInput();">
</form>
```

Red arrows point from the XPath expression to the `id` attribute of the `input` element in the HTML structure.



# Operator “or”

In this method, we use two interrogation conditions such as A and B and return a result-set as shown below:

A	B	Result
False	False	No Element
True	False	Returns A
False	True	Returns B
True	True	Returns Both

“or” is **case-sensitive**, you should not use capital “OR”.

**Syntax:** `//tag[XPath Statement-1 or XPath Statement-2]`

**Example:** `//*[@id='user-message' or @class='form-control']`

# Operator “and”

In this method, we use two interrogation conditions such as A and B and return a result-set as shown below:

A	B	Result
False	False	No Element
True	False	No Element
False	True	No Element
True	True	Returns Both

“and” is **case-sensitive**, you should not use capital “AND”.

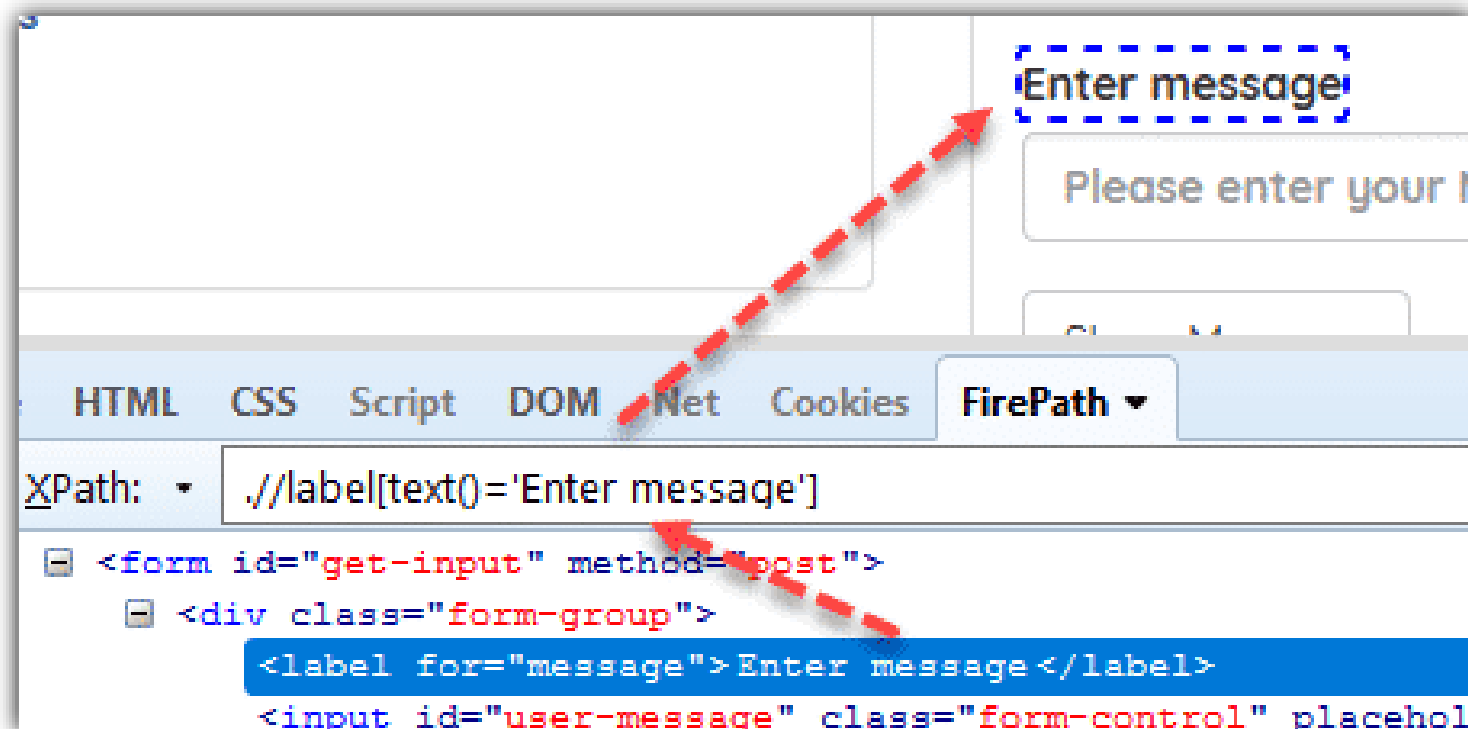
**Syntax:** `//tag[XPath Statement-1 and XPath Statement-2]`

**Example:** `//*[@id='user-message' and @class='form-control']`

# Locating element by text

Syntax: `//tag[text()='text value']`

Example: `//label[text()='Enter message']`



The screenshot displays a web application interface with a form. A label with the text "Enter message" is highlighted with a dashed blue border. A red dashed arrow points from this label to the XPath expression `./label[text()='Enter message']` in the Selenium IDE tool. The Selenium IDE interface shows the "XPath" tab selected, and the XPath expression is entered in the "XPath" field. Below the field, the corresponding HTML structure is shown, with the `<label for="message">Enter message</label>` element highlighted in blue. The HTML structure is as follows:

```
<form id="get-input" method="post">  
  <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>  
</form>
```

# Agenda

- Page object model
- Advanced locators css
- Advanced locators XPath
- **Practice**

# How to Make POM

- Create a separate class with somePagename.java
- Create variables that will be used as Objects for methods
- Create method where you will use these objects
- Tips for Current Tutorial:
  - Make driver static
  - Extend page class to your common or class which have Webdriver created

# Task 1 – POM for SS.lv

- Remake your ss.com project according to Page Object Model
- Both tasks – Dog and Car filter

## Task 2 – Login to Forum Cinemas

- Register some user for Forum Cinemas
- Based on POM Create Login page
- Write a test to login to ForumCinemas.lv

# Homework - Obligatory

- Create Profile POM page
- Write a test to login to ForumCinemas.lv
- Go To profile and Edit information – Edit all the fields on this page
- Validate that fields has been changed on this page (validation is done through assertions)



# Reference for Selectors

- <https://www.skptricks.com/2018/04/css-selectors-in-selenium-webdriver-with-example.html>
- <https://www.guru99.com/xpath-selenium.html>
- [https://www.w3schools.com/csSref/css\\_selectors.asp](https://www.w3schools.com/csSref/css_selectors.asp)
- <https://www.swtestacademy.com/xpath-selenium/>
- <https://www.softwaretestingmaterial.com/dynamic-xpath-in-selenium/>
- <https://www.techbeamers.com/websites-to-practice-selenium-webdriver-online/> - Demo websites for practice



**THANK YOU FOR YOUR ATTENTION**

