Selenium WebDriver Basics

Setup Selenium + NUnit. Writing Selenium Tests. Interaction with Page Elements

Selenium WebDriver

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

You Have Questions?

Software University

sli.do

#QA-FrontEnd

Table of Contents

Selenium Web Driver

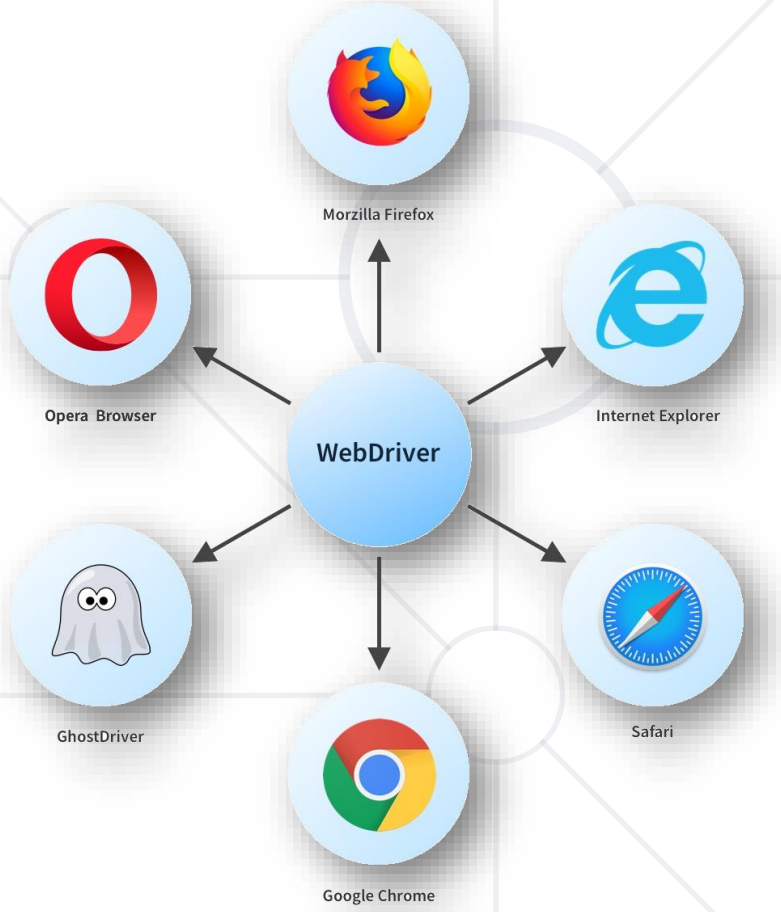
Web Browser Automation

Selenium – Overview

- **Selenium** automates Web browsers
  - Free, open-source test automation tool: https://selenium.dev
  - Very powerful: supports dynamic apps with AJAX and complex front-end frameworks (Angular, React, Vue.js, Meteor, Blazor, …)
- **Selenium WebDriver** automates browsers through standard API
  - Available for **C#**, **Java**, **Python**, **JavaScript** and other languages

Selenium WebDriver

- **Selenium WebDriver** automates browsers through standard API
  - Available for **C#**, **Java**, **Python**, **JavaScript** and other languages
  - It supports many browsers
  - Interacts with **HTML elements** on the web pages
    - **Sends text** to fields, **clicks on buttons**, etc.

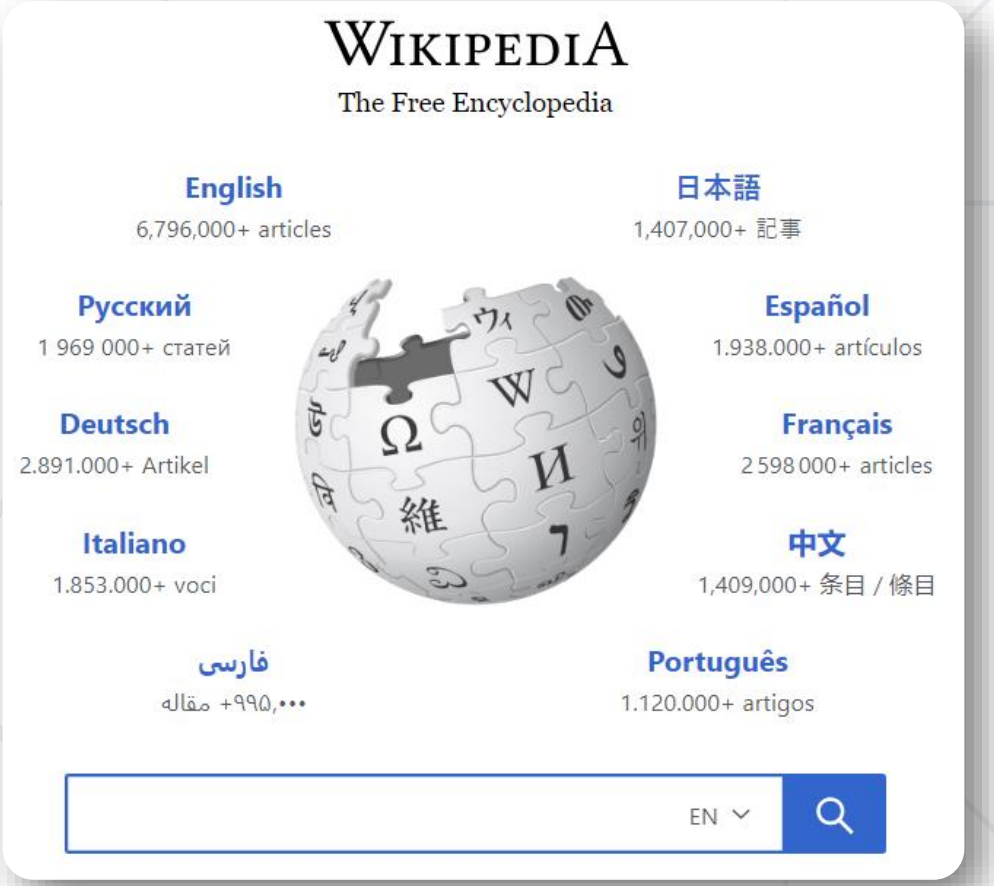


First Steps with Selenium WebDriver

Installation, Navigation

Problem: Open Wikipedia Web Site

- Create console application
- Install Selenium WebDriver
- Install Chrome Driver
- Add namespaces
- **Initialize** Selenium Web Driver
- Open a web site using Selenium: https://wikipedia.org
- Shut down the Web Driver



Installation

- Installing **Selenium WebDriver** from NuGet

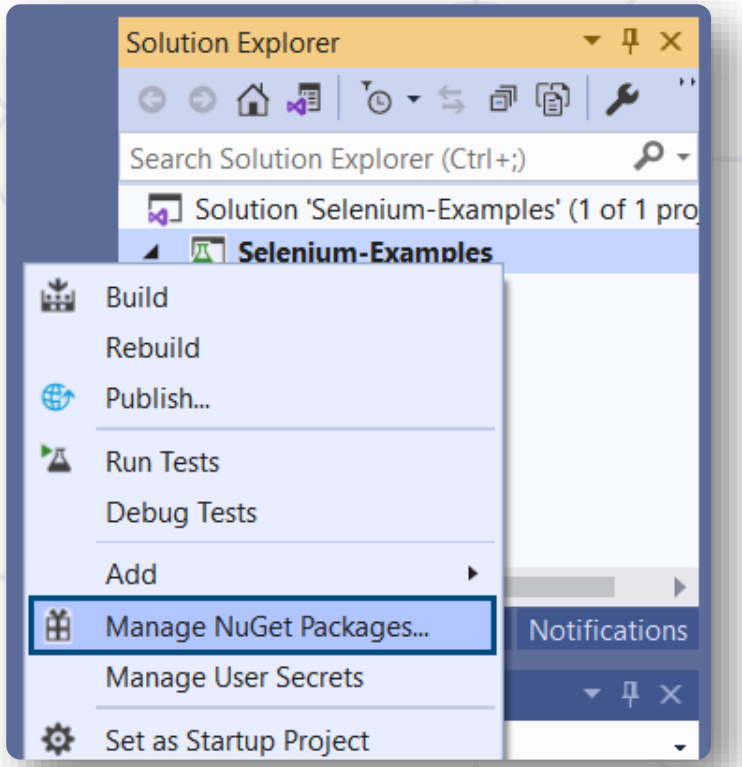
Selenium.WebDriver by selenium, **122M** downloads 4.28.0
.NET bindings for the Selenium WebDriver API

- Installing **ChromeDriver**

Selenium.WebDriver.ChromeDriver by jsakamoto, **63.2l** 132.0.6834.11000
Install Chrome Driver (Win32, macOS, macOS arm64, and Linux64) for Selenium WebDriver into your Unit Test Project.

- You may also install **Gecko Driver** for Firefox

Selenium.WebDriver.GeckoDriver by jsakamoto, **6.41M** download 0.34.0
Selenium Gecko Driver (Win32, Win64, macOS, macOS arm64, and Linux64) (does not make



Solution: Open Wikipedia Web Site

- Add the **OpenQA.Selenium** namespaces

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
```

- **Initialize** Selenium Web Driver

```
var driver = new ChromeDriver();
```

- Navigate to URL:

```
driver.Url = "https://wikipedia.org";
```

- **Shut down** the Web Driver

```
driver.Quit();
```

Problem Extended

- Open Wikipedia web site using Selenium
- Print the page title
- **Search for Quality Assurance**
- Print the title of the search results page
- Shut down the Web Driver

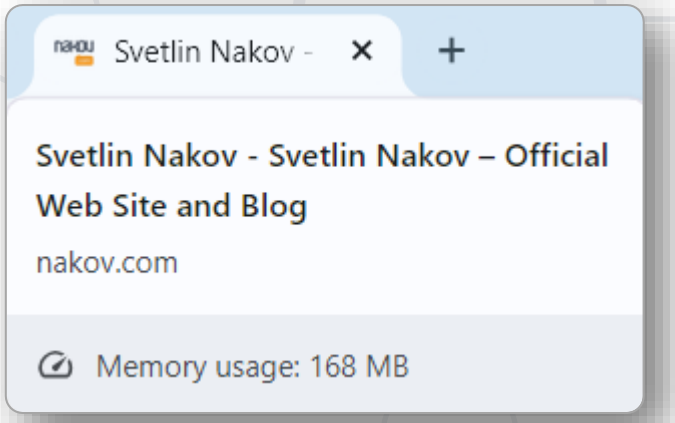
```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

// Create a new instance of ChromeDriver
var driver = new ChromeDriver();

// Navigate to the Wikipedia homepage
driver.Navigate().GoToUrl("https://wikipedia.org");

// Print the title of the main page to the console
Console.WriteLine("Main page title: " + driver.Title);

// Find the search input element by its ID
var searchBox = driver.FindElement(By.Id("searchInput"));

// Click on the search box to focus it
searchBox.Click();

// Type "QA" into the search box and press Enter
searchBox.SendKeys("Quality Assurance" + Keys.Enter);

// Print the title of the QA search results page to the console
Console.WriteLine("Quality Assurance page title: " + driver.Title);

// Close the browser and end the session
driver.Quit();
```

Selenium Tests in C# with NUnit

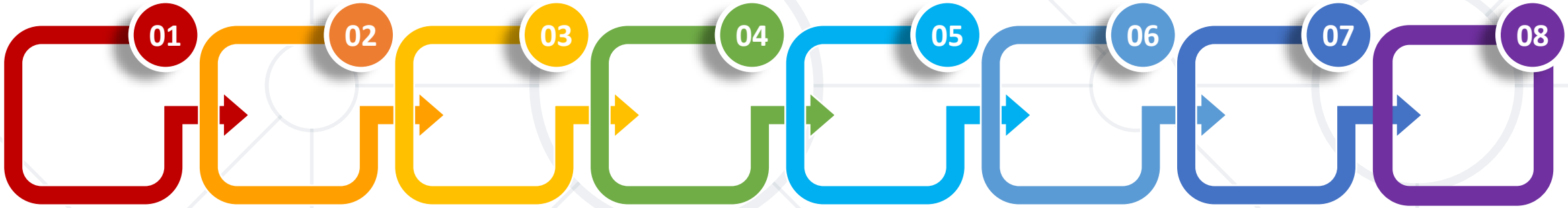
8 Basic Components

Selenium Tests in C#

- Setup **NUnit** test with **Selenium** in Visual Studio:
  - Create NUnit project in VS
  - Add the NuGet packages:
    - **Selenium.WebDriver**
    - **Selenium.WebDriver.Chrome**
  - Write the **SetUp()** and **TearDown()** methods
  - Write a **test method**



8 Basic Components in Selenium Tests



01	02	03	04	05	06	07	08
Start the session	Take action on browser	Request browser info	Waiting Strategy	Find an element	Take action on element	Request element info	End the session

Start

- **Start the Session:**

```
IWebDriver driver = new ChromeDriver();
```

  - `IWebDriver`: Interface that defines the methods and properties required to control a web browser
  - `new ChromeDriver()`: Creates a new instance of ChromeDriver which is a concrete implementation of IWebDriver

Take Action on Browser

- **Navigating to a web page**

```
driver.Navigate().GoToUrl("https://nakov.com");
```

  - The **Navigate()** method provides an object that allows browser navigation
  - The **GoToUrl** method navigates the browser to the specified URL

Request Browser Information

- **Request the Title of the Current Web Page**

```
var title = driver.Title;
```

  - Retrieves the title of the current web page and stores it in the "title" variable
  - There are various types of information that can be requested about the browser, including **window handles**, **browser size** and **position**, **cookies**, **alerts**, and more

Establish Waiting Strategy

- **Ensure the element is present on the page and ready for interaction**

```
driver.Manage().Timeouts().ImplicitWait =
                        TimeSpan.FromMilliseconds(500);
```

  - Implicit waits are **not the optimal solution**, but are the easiest to demonstrate
  - Synchronizing code with the browser's current state, requires advanced expertise to manage effectively in Selenium automation

Find an Element

- The majority of commands in Selenium sessions involve interacting with elements on a web page
- **Before you can interact with an element, you must first locate it**

```
var searchLink =
  driver.FindElement(By.ClassName("smoothScroll"));
```

- `FindElement` Method: Locates an element on the web page based on a specific strategy
- **Locating Strategies**: Can include locating by Name, TagName, Id, ClassName, CssSelector, XPath, etc.

Take Action on Element

- **There are 5 basic commands that can be executed on an element:**
  - `Click`: Simulates a mouse click on any web element
  - `SendKeys`: Allows to enter text into input fields
  - `Clear`: Removes any existing text or values from input
  - `Submit`: Mimics the behavior of clicking the submit button or pressing the "Enter" key
  - `Select`: Specific to dropdown menus or select elements, allowing to choose an option from the available choices

```
searchLink.Click();
```

Request Element Information

- **There are a number of details that can be queried about a specific element**
  - Tag Name
  - Size and Position
  - Is Displayed, Is Enabled
  - Get CSS Value
  - Get CSS Value
  - Text Content
  - Fetching Attributes or Properties

```
var placeholderText =
  driver.FindElement(By.Id("s")).GetAttribute("placeholder");
```

End the Session

- **Disposes** the **WebDriver instance**, ensuring proper cleanup of resources
- Calls **`Quit()`** internally while preventing unexpected exceptions
- Follows .NET best practices for managing disposable objects
- Recommended over **`Quit()`** in modern Selenium and .NET projects

```
driver.Dispose();
```

Selenium Test Example

```
[Test]
| 0 references
public void Test_NakovCom()
{
    driver.Url = "https://nakov.com";
    var windowTitle = driver.Title;
    Assert.That(windowTitle.Contains("Svetlin Nakov - Official Web Site"));
    Console.WriteLine(windowTitle);

    var searchLink = driver.FindElement(By.ClassName("smoothScroll"));
    Assert.That(searchLink.Text, Does.Contain("SEARCH"));
    Console.WriteLine(searchLink.Text);

    searchLink.Click();

    var message = driver.FindElement(By.Id("s"));
    var placeholderText = message.GetAttribute("placeholder");
    Assert.That(placeholderText, Is.EqualTo("Search this site"));
    Console.WriteLine(placeholderText);
}
```

Svetlin Nakov -

Svetlin Nakov - Svetlin Nakov – Official Web Site and Blog

nakov.com

Memory usage: 168 MB

Run | Debug

Test Detail Summary

Test_NakovCom

Source: FirstTest.cs line 17

Duration: 5.4 sec

Locating Elements

Locators in Selenium: Id, Name, CSS, XPath, Others

Locators

- A locator is a way to identify elements on a page
- It is the argument passed to the Finding element methods
  - **Basic Locators**
    - ID, Name, Tag Name, Class Name
  - **Link Text Locators**
    - Link Text, Partial Link Text
  - **Advanced Locators**
    - CSS selectors, XPath

Practice Page

Contact Form

○ Male ○ Female

First name:

Vincent

Last name:

Vega

Additional Information

Phone Number:

Newsletter:

☐

Submit

To know more about our programs, visit the official page Softuni Official Page

Elements Console Sources Network Performance Memory Application

```
<html>
▶ <head> ⋯ </head>
▼ <body>
    <style> .information { background-color: white; color: black; padding: 10px; }
    </style>
    <h2>Contact Form</h2>
    ▼ <form action="/action_page.php"> == $0
        <input type="radio" name="gender" value="m">
        "Male "
        <input type="radio" name="gender" value="f">
        "Female "
        <br>
        <br>
        <label for="fname">First name:</label>
        <br>
        <input class="information" type="text" id="fname" name="fname" value="Vincent">
        <br>
        <br>
        <label for="lname">Last name:</label>
        <br>
        <input class="information" type="text" id="lname" name="lname" value="Vega">
        <br>
        <br>
        <h3>Additional Information</h3>
        ▶ <div class="additional-info">⋯</div>
        <label for="newsletter">Newsletter:</label>
        <br>
        <input type="checkbox" name="newsletter" value="1">
        <br>
        <br>
        <input type="submit" value="Submit">
    </form>
    ▼ <p>
        "To know more about our programs, visit the official page "
        <a href="http://www.softuni.bg">Softuni Official Page</a>
```

Basic Locators

- **ID**
  - Locates elements whose ID attribute matches the search value
  - The most reliable locator, as it should be unique

```
driver.FindElement(By.Id("lname"));
```

- **Name**
  - Locates elements whose NAME attribute matches the search value
  - The Name attribute can be useful when ID is not available
  - Generally, also should be unique

```
driver.FindElement(By.Name("newsletter"));
```

Basic Locators

- **Tag Name**
  - Locates elements whose tag name matches the search value
  - Used to find elements of a particular type, like input, button, etc. Less specific than ID and Name

```
driver.FindElement(By.TagName("a"));
```

- **Class Name**
  - Compound class names are not permitted
  - Useful when elements are styled with class attributes, but be aware that multiple elements can share the same class name

```
driver.FindElement(By.ClassName("information"));
```

Link Text Locators

- **Link Text**
  - Locates anchor elements whose visible text matches the search value
  - Use this when you need to find a link by its exact text

```
driver.FindElement(By.LinkText("Softuni Official Page"));
```

- **Partial Link Text**
  - Locates anchor elements whose visible text contains the search value
  - If multiple elements are matching, only the first one will be selected
  - Use to find a link containing certain text

```
driver.FindElement(By.PartialLinkText("Official Page"));
```

Advanced Locators - CSS

- **CSS selectors** are a powerful way to find elements based on their attributes, classes, IDs, and more
- Offer more flexibility compared to basic locators
- If the element has an id, the locator is created as:
  - **`css = #id`**

```
driver.FindElement(By.CssSelector("#fname"));
```

- Otherwise the format is:
  - **`css = [attribute=value]`**

Attribute Selectors

- **Basic Attribute Selector**
  - Selects elements based on attribute value
  - Syntax: `[attribute='value']`

```
driver.FindElement(By.CssSelector("input[name='fname']"));
```

- **Contains Attribute Selector**
  - Selects elements containing a specified attribute value
  - Syntax: `[attribute*='value']`

```
driver.FindElement(By.CssSelector(
                        "input[class*='button']"));
```

Advanced CSS Selectors

- **Child Combinator**
  - Selects direct children
  - Syntax: **parent > child**

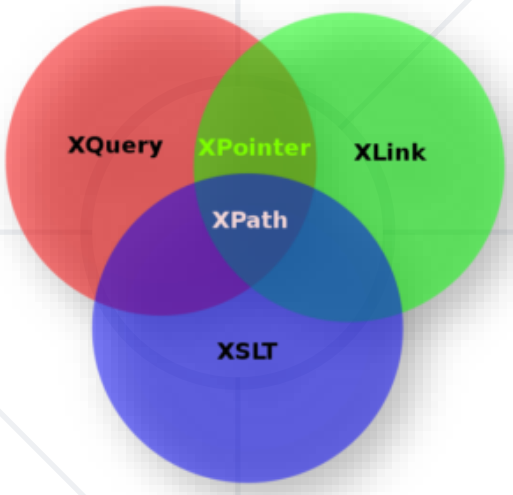
```
driver.FindElement(By.CssSelector(
        "div.additional-info > p > input[type='text']"));
```

- **Descendant Combinator**
  - Any element descendant of the specified parent (regardless of depth)
  - Syntax: **parent** (space) **child**

```
driver.FindElement(By.CssSelector(
"form div.additional-info input[type='text']"));
```

Advanced Locators - XPath

- **XPath** is a syntax for selecting parts of an XML (and HTML) documents
- XPath uses **path expressions** to navigate in the XML documents
- Expressions use a **combination** of **axes**, **operators**, **node types**, **functions**, and **predicates** to target elements




```
//div[@class='featured-box cloumnsize1']//h4[1]//b[1]
```

Absolute Path

- An absolute XPath expression contains the location of **all elements from the root node** (HTML), where the path starts, to the **desired element**
- It's **not flexible** and can break if the page structure changes
- **Always begins** with a single forward slash **/**

```
driver.FindElement(By.XPath("/html/body/form/input[1]"));
```

Relative Path

- **Starts** from a **specific element** and **navigates** through the DOM hierarchy to **locate** the **desired element**
- It's **more flexible** and resilient to changes in the page structure
- A relative path, or a double slash search, begins with double forward slashes **//**
- The double slashes signify a break in the absolute path

```
driver.FindElement(By.XPath("//input[@value='m']"));
```

Syntax for Relative XPath

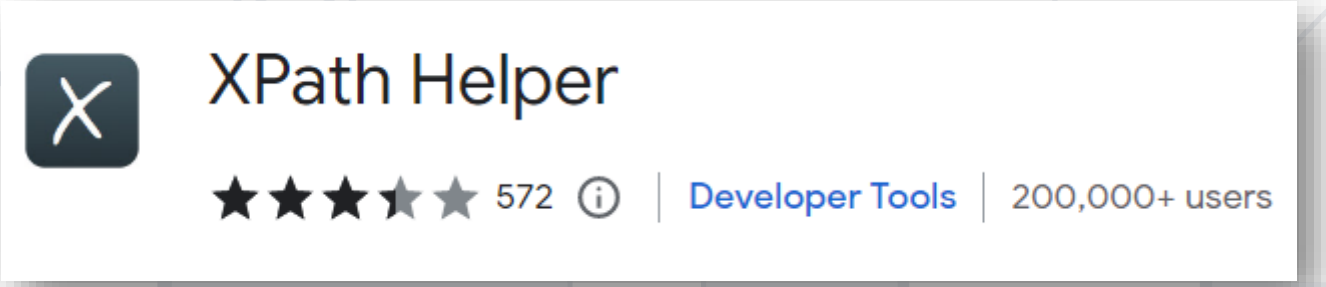
- The standard syntax for creating relative XPath is as follows:

```
//tagname[@attribute='value']
```

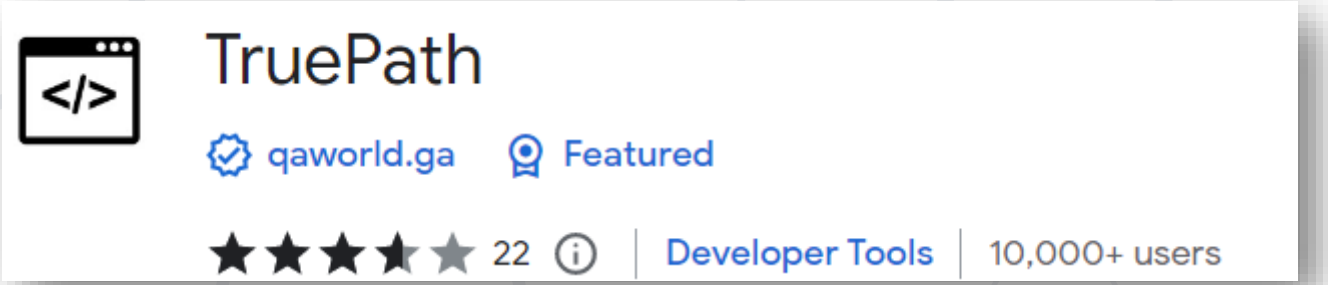
  - **//** : Select the current node
  - **tagname** : Tagname of the particular node
  - **@** : Select attribute
  - **attribute** : Attribute the name of the node
  - **value** : Value of the attribute

XPath Add-Ons

- **XPath Helper** add-on
  - https://cutt.ly/jjLqRZj



- **TruePath** add-on
  - https://cutt.ly/GjX4QPf



- **Ranorex Selocity** add-on
  - https://cutt.ly/f3AqrCl

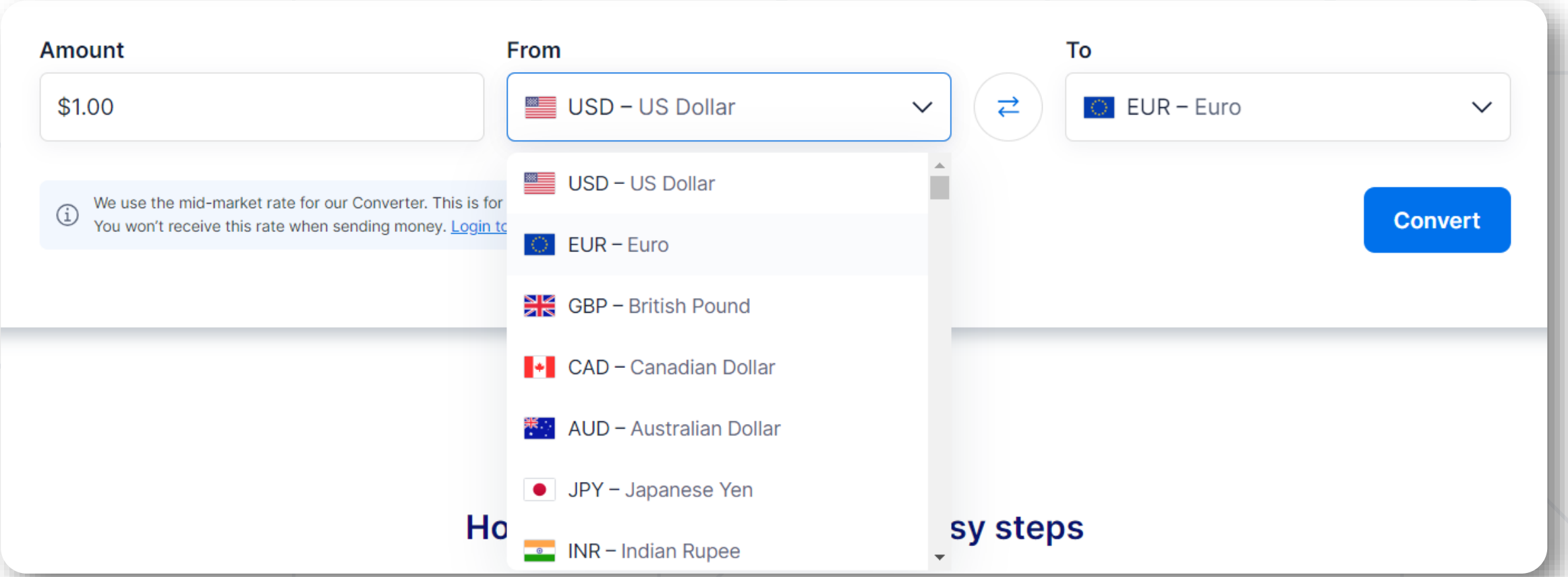
Best Location Practices

- The most reliable and efficient way to locate an element on a page are unique **HTML ID** or **Name**
- If unique IDs are not available, the **next best option** are well-written **CSS selectors**
- **XPath** is highly flexible and **can locate almost any element**, but is often **complex** and **hard to debug**
- **Link Text** is useful for locating link elements by their **visible text**
- **Tag Name** to be **used with caution**. Mainly useful when working with collections of elements

Inspecting Dynamic UI Elements

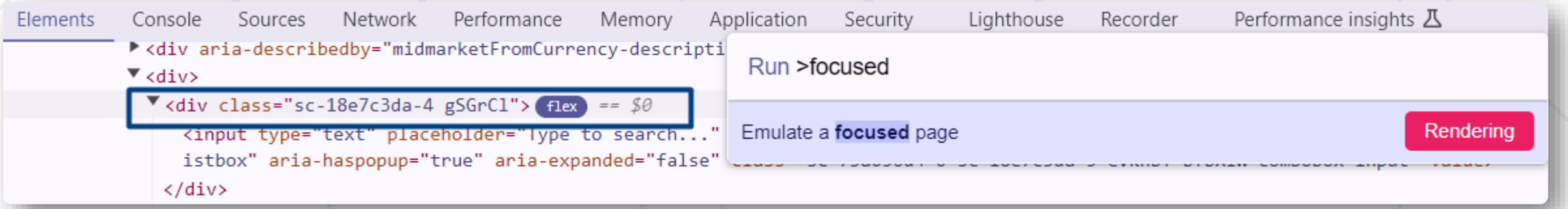
Disappearing Page Content

- https://www.xe.com/



Focusing Element

- **Dev Tools → Elements tab**
- Find the element with the drop-down menu an click on it
- Make sure the **element is highlighted** in the HTML structure
- Click **control** + **shift** + **P** (Windows)
- Type the word "**focused**" + **Enter**
- Now clicking around in the console will not close the element



Show / Hide the Browser Windows

Headless and Visible Browsers

Headless Browser

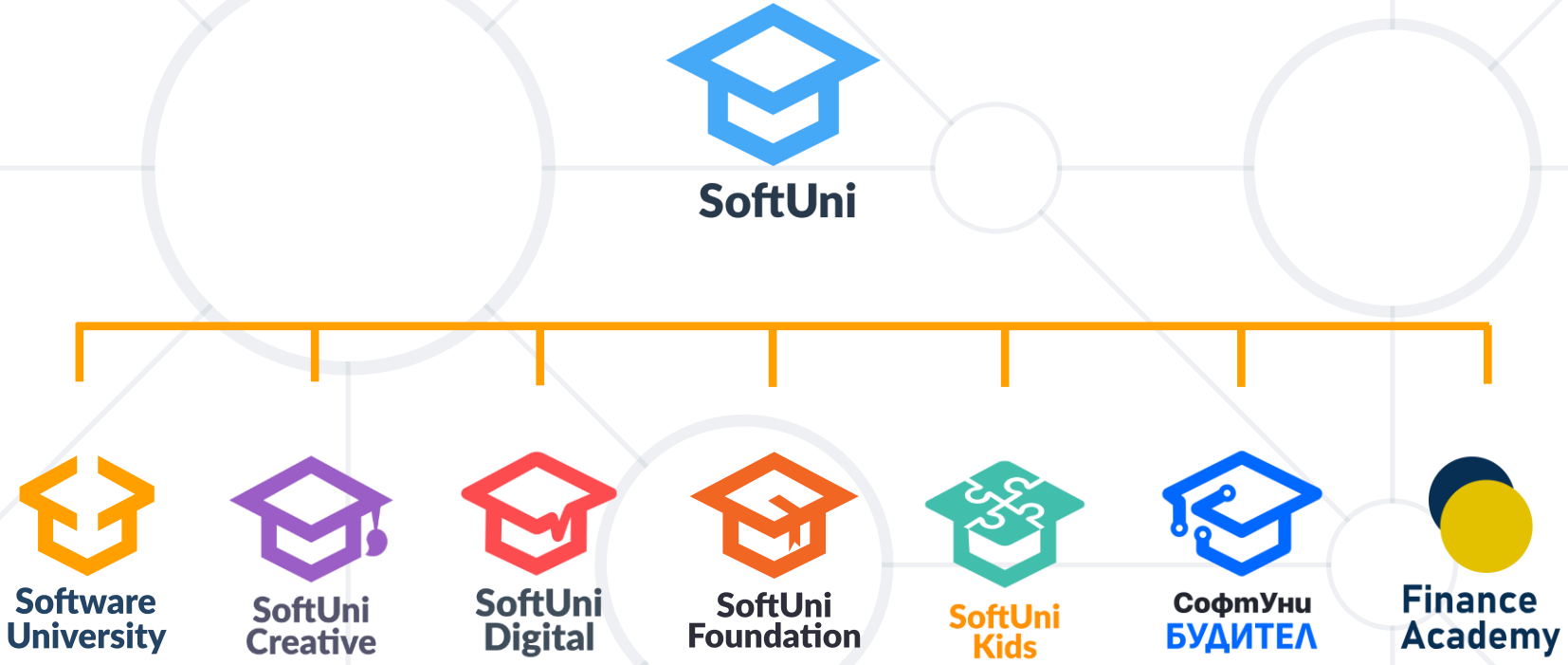
- **Headless** (invisible) browsers are used to speedup testing
  - Headless browsers can run server-side, without GUI
  - Run tests in a virtual environment, without specific browser installed
  - Run tests faster without rendering and displaying the content on a screen
- Starting **headless Chrome** with Selenium:

```
var options = new ChromeOptions();
options.AddArgument("--headless=new");
var driver = new ChromeDriver(options);
driver.Navigate().GoToUrl("https://selenium.dev");
```

Summary

- Selenium WebDriver setup
- Using NUnit + Selenium WebDriver in C#
- Test Components
- Selecting elements on the page
  - By ID, by Name, by CSS, by XPath, Others
- Headless Mode

Questions?



Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024 FOR A BETTER WORLD OF WORK

DRAFT KINGS THE CROWN IS YOURS

VIVACOM

INDEAVR Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg