Module 6:
Selenium WebDriver
Locators
RegEx basics

MODULE 4 : AGENDA

1

LECTURE 1

- Overview: what is WebDriver?
- How it works?
- Web DevTools
- Locators
- RegEx basics

2

LECTURE 2

- Page Object Pattern
- Page Factory Pattern

3

LECTURE 3

- Actions
- JavaScript executor
- Remote Execution
- Selenuim Grid

PART 1:
WHAT IS WEBDRIVER?

WHAT IS WEBDRIVER? WHAT IS SELENIUM?

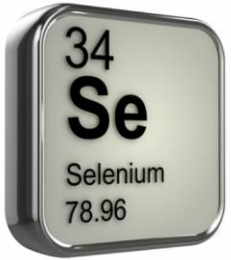
Selenium is a project name, within which a series of software products is developed:

- **Selenium IDE** (Firefox plugin, that records and reproduces user actions)
- **Selenium RC** (the previous version of the library for browser management = Selenium)
- **Selenium Server standalone** (lets you control the browser from a remote machine over the network)
- **Selenium WebDriver** (library for browser management)
- **Selenium Grid (**cluster consisting of several Selenium Servers)

Open source project (Apache 2.0 license)

WHAT IS WEBDRIVER? SELENIUM HISTORY

- Developed by **Jason Huggins** in **2004** at ThoughtWorks company
- In 2004, Paul Hammant joined the team and steered the development of the second mode of operation - Selenium Remote Control (**Selenium RC**)
- In 2005, Dan Fabulich and Nelson Sproul made a series of patches that transformed Selenium-RC into what it became best known for.
- In 2007, Huggins joined Google, where he continued with the development and stabilization of Selenium RC
- In 2007, Simon Stewart at ThoughtWorks developed **WebDriver**
- In 2008, Philippe Hanrigou (then at ThoughtWorks) made **Selenium Grid**
- In 2009, after a meeting between the developers at the Google Test Automation Conference Google Selenium RC and ThoughtWorks WebDriver projects were merged -> **Selenium WebDriver** (or **Selenium 2.0**) was born
- In 2016 Selenium 3.0 was released.
- Why the name is 'Selenium'? you can cure mercury poisoning by taking selenium supplements.



WHAT IS WEBDRIVER? SELENIUM FUTURE – 3.0

Selenium 3.0 – was released in 2016..
https://github.com/SeleniumHQ/selenium/wiki/Shipping-Selenium-3
https://www.joecolantonio.com/2016/05/31/selenium-3-sneak-peak/

W3C Working Draft 09 November 2015

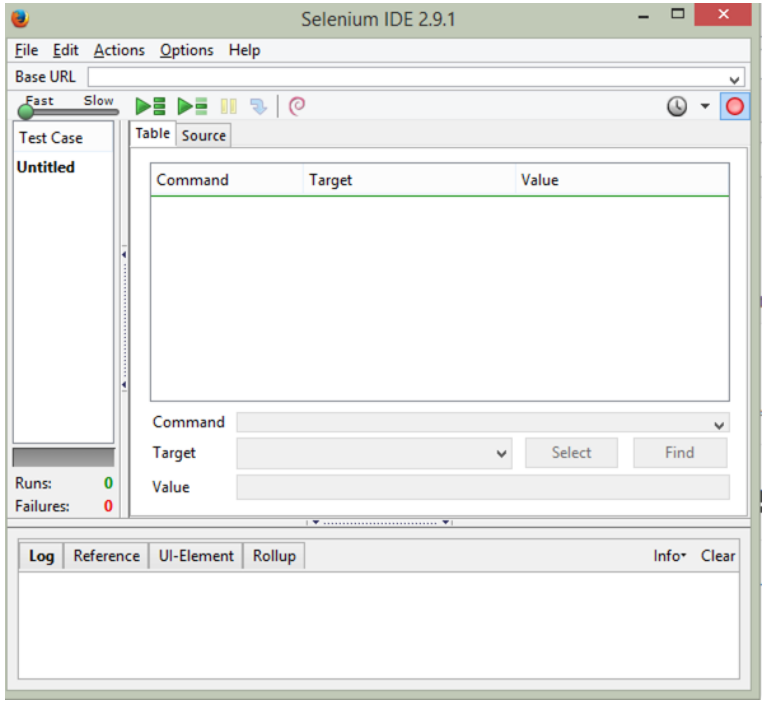
WebDriver is a remote control interface that enables introspection and control of user agents. It provides a platform- and language-neutral wire protocol as a way for out-of-process programs to remotely instruct the behavior of web browsers.

The standard forms part of the Web Testing Activity.

WHAT IS WEBDRIVER? SELENIUM IDE

Selenium IDE – Mozilla Firefox extension
http://www.seleniumhq.org/download/

Capabilities:
- record and playback simple tests
- simple assertions
- export generated code



Usually is used for
- Selenium automation get stared
- Create a small scenario for quick bug replication
- Auxiliary script to avoid routine during manual testing

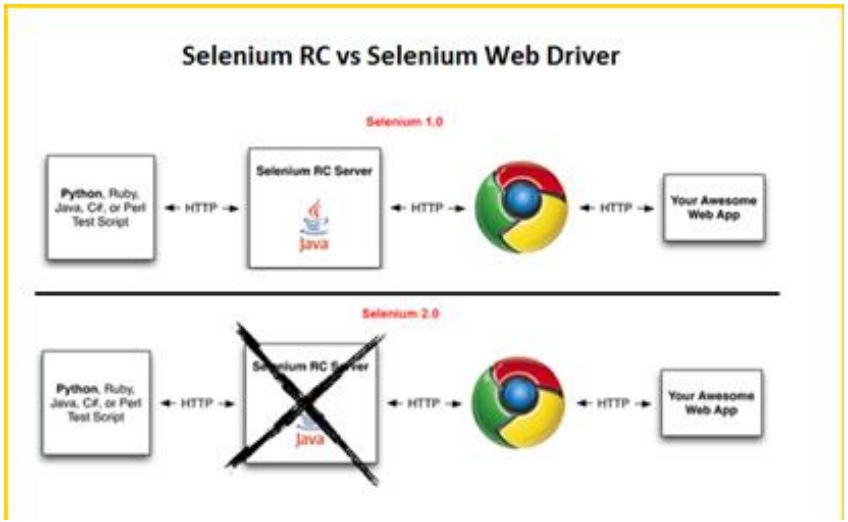
Doesn't require any programming skills
Has a lot of plugins, that may significantly extend Selenium IDE capabilities


Demo

SELENIUM WEBDRIVER VS. SELENIUM RC

Selenium RC sends commands to browsers via JavaScript (Selenium Core)

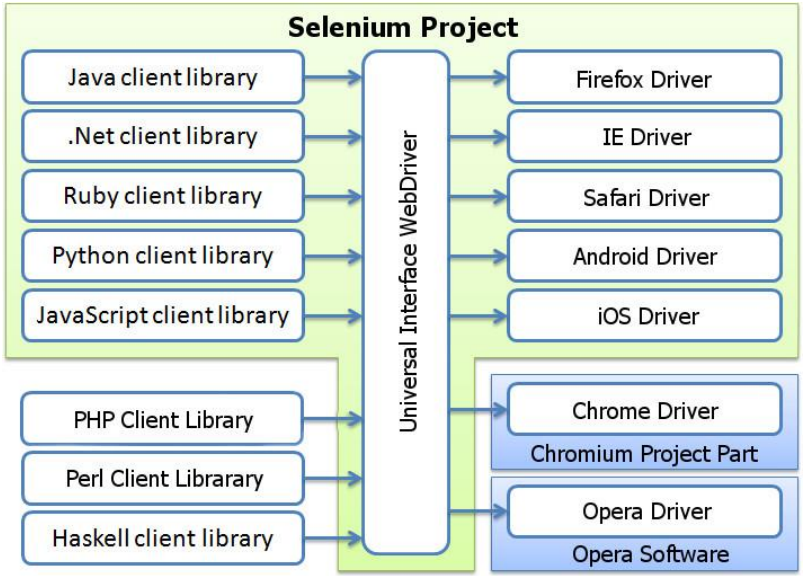
Selenium WebDriver sends commands to browsers using browser native interfaces



WHAT IS WEBDRIVER? DEFINITION

Selenium WebDriver, or Selenium 2.0 (widespread abbreviation - WebDriver) is a software library for browser management.

WebDriver is not a tool for test automation (such as TestComplete, QuickTest Pro, …).

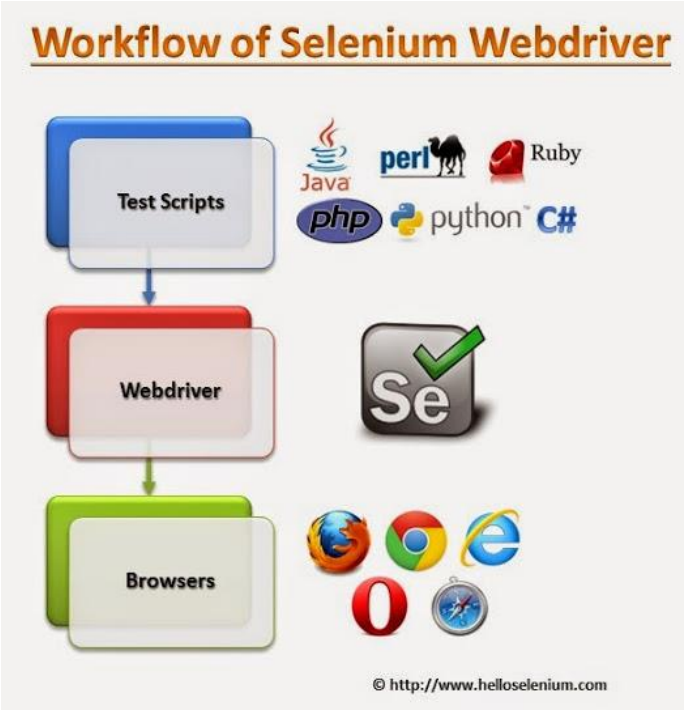


WHAT IS WEBDRIVER? WORKFLOW

1) An automation developer creates a scenario in any high level language (using a corresponding library)

1) The scenario is accepted by WebDriver and transformed into the language, that a particular browser (depends on WebDriver type) understands

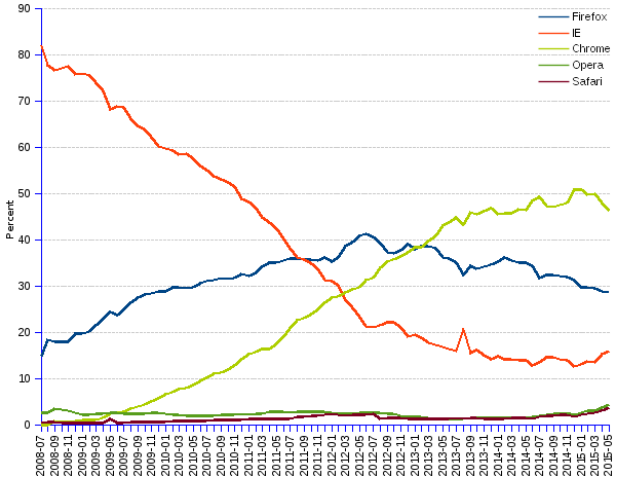
1) Browser performs the manipulations, that were described in the first step



SUPPORTED BROWSERS

Wide variety of browsers supported:

- Firefox (all the version, where Selenium IDE is available)
- IE (7-10, 11 – additional configuration required)
- Edge (not all major features yet implemented)
- Safari (5.1+)
- Opera (Opera Software)
- Chrome (Chromium)



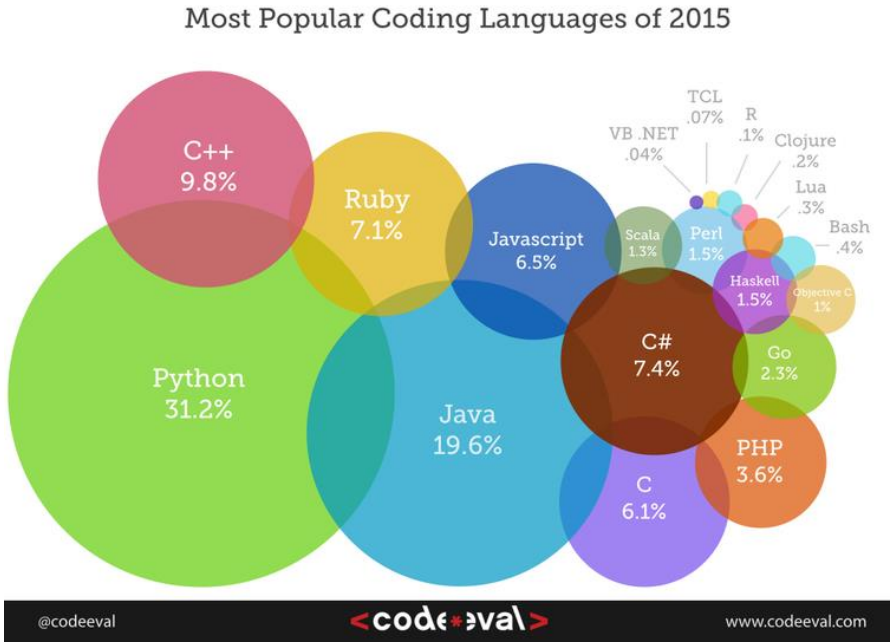
SUPPORTED PLATFORMS

- Windows
- Linux
- Mac OS
- OS X
- Solaris
- Android
- iOS

SUPPORTED PROGRAMMING LANGUAGES

Programming languages (+ frameworks)

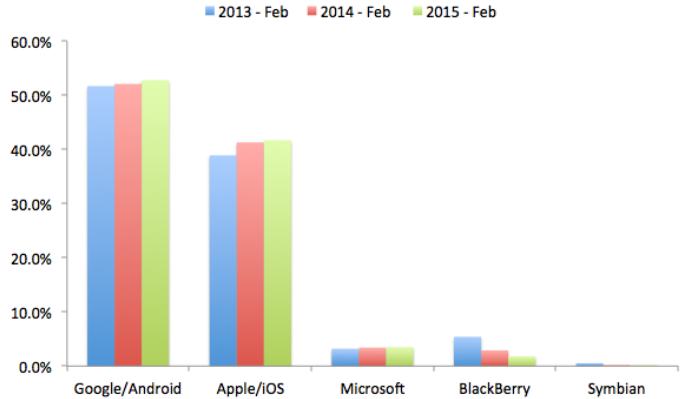
- Java (JUnit, TestNG)
- JavaScript (WebdriverJS, WebdriverIO, NightwatchJS)
- C# (NUnit)
- Objective-C
- Perl
- PHP (Behat + Mink)
- Python (unittest, pyunit, py.test, robot framework)
- Ruby (RSpec, Test::Unit)



WHAT IS WEBDRIVER? MOBILE SUPPORT

Mobile Drivers:

- Windows Phone
- Selendroid (real Android devices and emulators)
- Appium (iOS and WebDriver using WebDriver protocol)
- iOs-driver
- BlackBerry 10



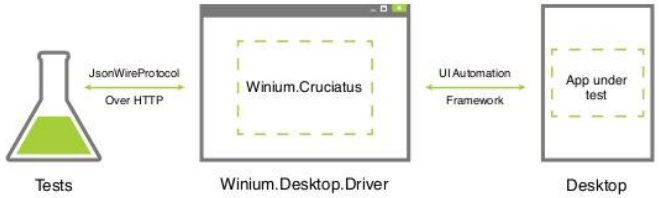
WHAT IS WEBDRIVER? WINIUM

Win-desktop + Selenium (http-client)

How to start with Winium:
1) download the latest driver;
2) Launch the driver with administrator privileges;
3) Select your favorite programming language and IDE;

How to get locators:
1) UISpy;
2) UI Automation Verify
(more handy + better performance)

Do not touch your mouse, then a case is started

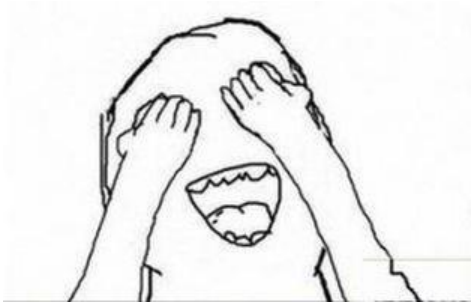


Locators:
- AutomationProperties.AutomationId
- Name
- ClassName

WHAT IS WEBDRIVER? HTMLUNIT DRIVER

The fastest and most lightweight implementation of WebDriver
Java based (for any language binding (other than java) the Selenium Server is required to use this driver)
without a GUI

It is known as Headless Browser Driver. It is same as Chrome, IE, or FireFox driver, but it does not have GUI so one cannot see the test execution on screen.



Pros:

- Fastest
- Platform independent (pure Java)
- Supports JavaScript

Cons:

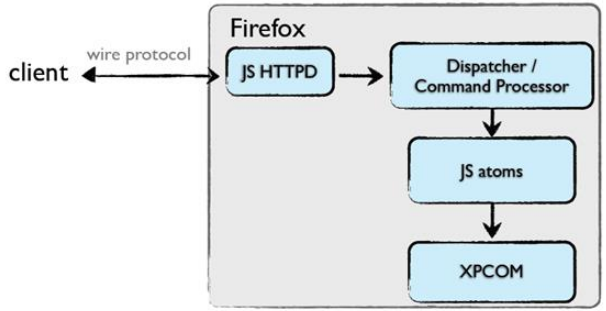
- Emulates other browsers' JavaScript behavior (Rhino JavaScript engine, that is used by HtmlUnit, is used by none of the popular browsers)
- Absence of the visual control (screenshots, etc.)

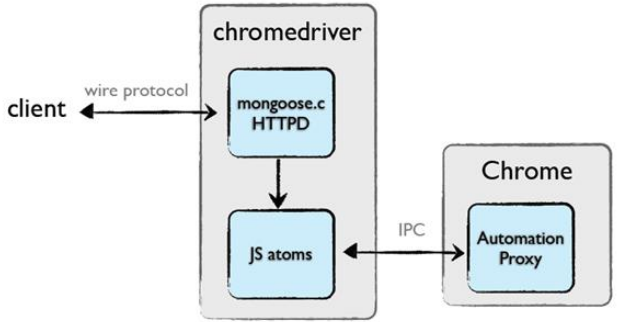
WHAT IS WEBDRIVER? PROS AND CONS

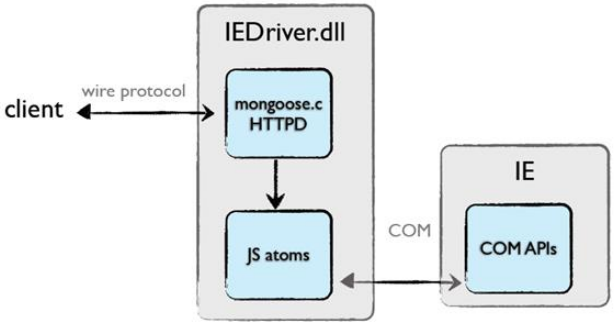
Pros	Cons
Selenium is pure open source, freeware and portable	Require expertise resources, well versed in frameworks
Supports a lot of languages (Java, C#, …)	Difficult to test image based applications
Supports many OS-s and browsers	Needs outside support for report generation
Can be integrated with Maven and ANT build tools	Does not support built in add-ins support
Can be integrated TestNG, NUnit, …	Does not provide any built in IDE
Can be integrated with Jenkins Hudson and other CI tools	Selenium Engineers are bit in scarcity these days
Can be used for mobile testing	Script creation time is bit high
Low CPU and RAM consumption	Partially supports for Dialog boxes
Faster test execution (in comparison with Selenium RC)	Some popular methods are missed - isElementPresent()
Doesn't require Selenium Server (unlike Selenium RC)	Does not support file upload facility

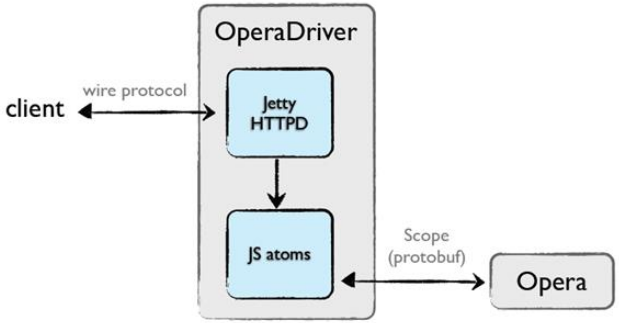
PART 2:
HOW IT WORKS?

HOW IT WORKS? DIFFERENT BROWSERS









WEBDRIVER API

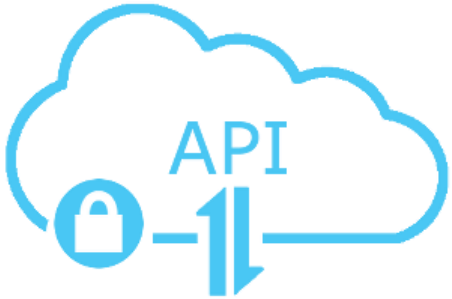
Interface WebDriver - represents an idealized web browser.

Methods can be used for:

- control of the browser itself;
- selection of WebElements
- debugging aids

Key methods:

- get(String url) - to load a new web page
- findElement(By by) – to find an element on the page



close()
findElements(By by)
getCurrentUrl()
getPageSource()

WEBDRIVER API. BROWSER INTERACTION

Java	C#	Description
close()	Close()	Close current window
findElement(By by)	FindElement(By by)	Find first WebElement By locator
findElements(By by)	FindElements(By by)	Find all WebElements By locator
get(String url)	Navigate().GoToURL(String url)	Open URL
getCurrentUrl()	GetCurrentUrl()	Get current URL
getPageSource()	GetPageSource()	Get Page source
getWindowHandle()	GetWindowHandle()	Get window descriptor
getWindowHandles()	GetWindowHandles()	Get window descriptors
manage()	Manage()	Managing the WebDriver
navigate()	Navigate()	Navigate backward and forward (History)
quite()	Quite()	Exit browser
switchTo()	SwitchTo()	Switch to frame or window
getTitle()	GetTitle()	Get current page title

WEBDRIVER API. WEBELEMENT

Interface to represent an HTML element

Provides methods to interact with web page elements:

- clear()
- click()
- findElement(By by)
- findElements(By by)
- getAttribute(String name)
- getCssValue(String propertyName)
- getLocation()
- getSize()
- getTagName()
- getText()
- isDisplayed()
- isEnabled()
- isSelected()
- sendKeys(CharSequence... keysToSend)
- submit()

WEBDRIVER API. NAVIGATION

Open Page URL

get(Strung url) / Get(String url)

```
driver.get("http://www.google.com");
```

Browser Navigation

```
driver.navigate().to("some_url");
driver.navigate().back();
driver.navigate().refresh();
driver.navigate().forward();
```

PART 3: LOCATORS

DOM

DOM (Document Object Model) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.

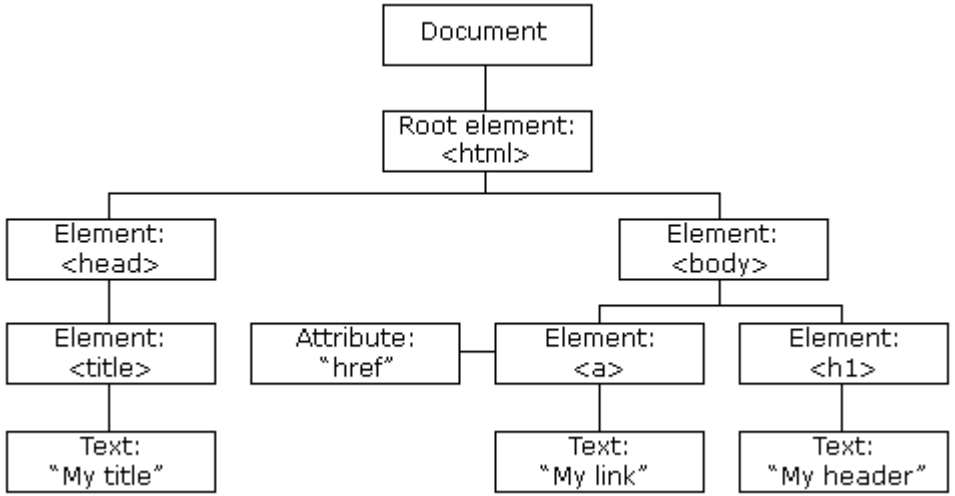
It is declared by W3C (World Wide Web Consortium).

HTML DOM

HTML DOM is a standard object model for HTML. It defines:

- The HTML elements as objects (nodes)
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

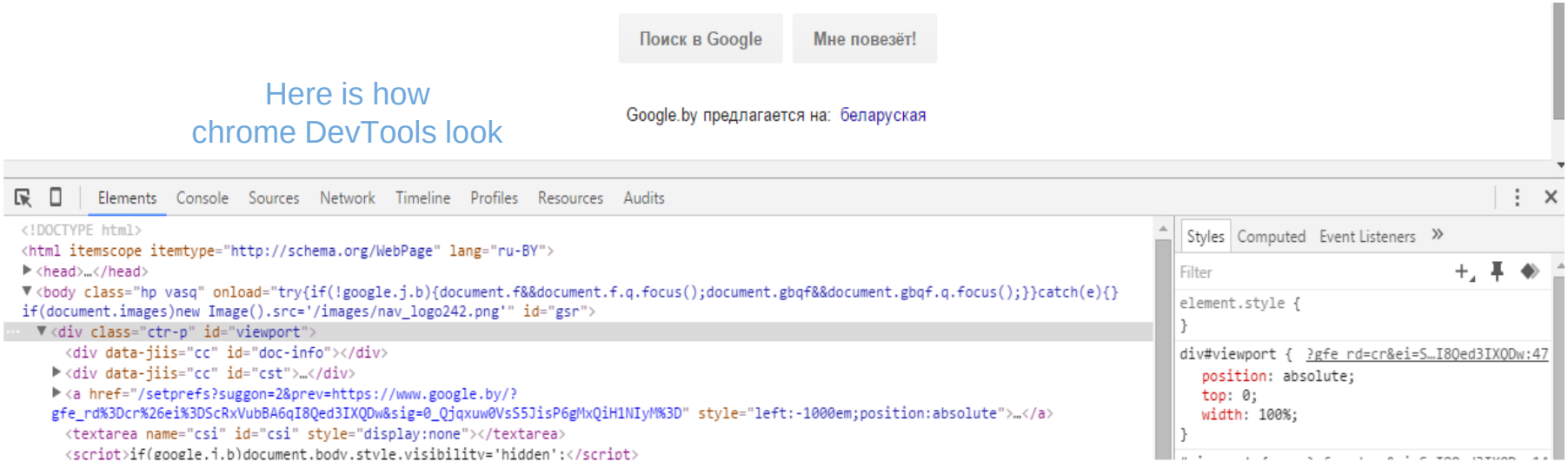
In other words: HTML DOM is a standard for how to get, change, add, or delete HTML elements.



HOW CAN WE SEE THE PAGE DOM?

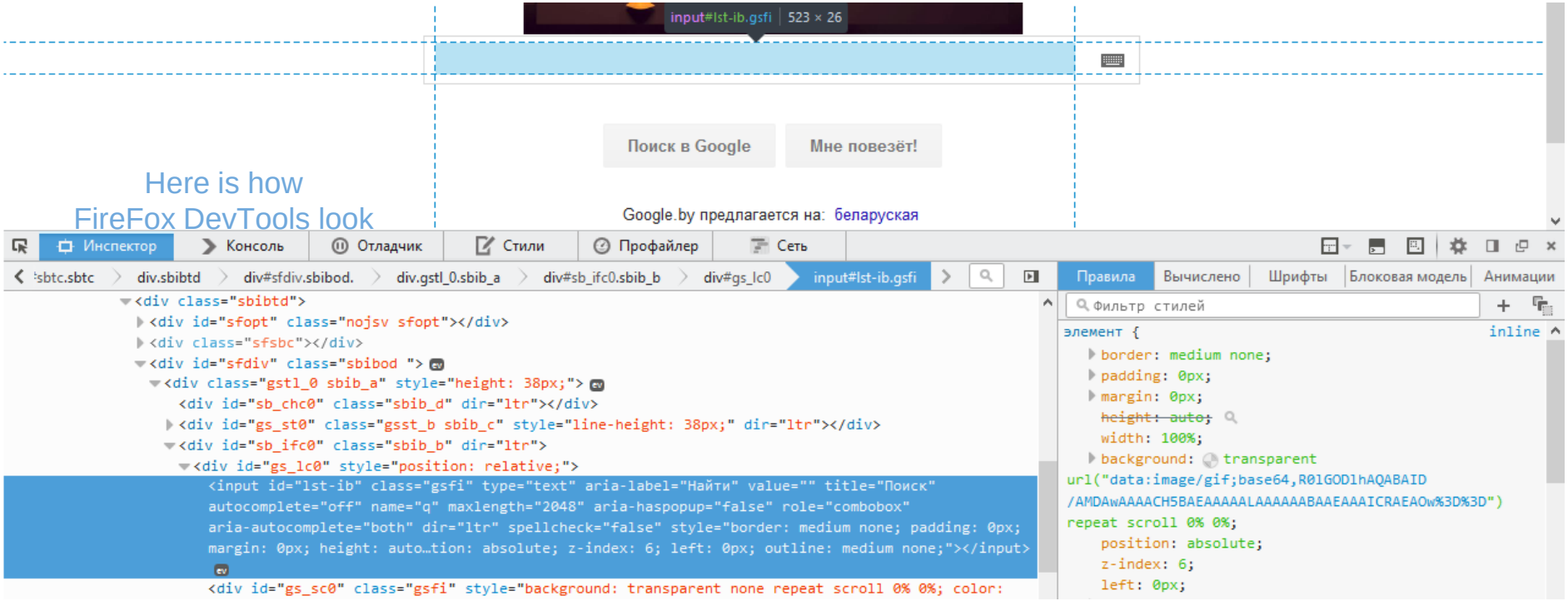
Using **DevTools** - web analyzing and debugging tools that are built into browser.

Here is how chrome DevTools look



HOW CAN WE SEE THE PAGE DOM?

Here is how FireFox DevTools look



HOW CAN WE SEARCH ELEMENTS USING LOCATORS?

Here is how
FirePath looks

Поиск в Google Мне повезёт!

Консоль HTML CSS Сценарий DOM Сеть Cookies FirePath

Top Window | Highlight | XPath: | //input[@name='btnK'] | Eval

```
<div class="tsf-p">
  <div id="logocont" class="nojsv logocont">
  <div class="sfibbbc">
  <div class="jsb" style="padding-top:18px">
    <center>
      <input type="submit" jsaction="sf.chk" name="btnK" aria-label="Поиск в Google" value="Поиск в Google"/>
      <input type="submit" jsaction="sf.lck" name="btnI" aria-label="Мне повезёт!" value="Мне повезёт!"/>
    </center>
  </div>
</div>
```

HOW DOES WEBDRIVER LOCATE WEB ELEMENTS IN DOM?

It can find elements by:

- Id
- Name
- Class
- Value
- XPath
- CSS Selector
- other ways

LOCATOR STRATEGIES

Strategy	HTML	Selenium Locator
Identifier	<form id="login"> <input name="username" type="text"/> </form>	identifier=login identifier=username
ID	<form id="login">	id=login
Name	<input name="username" type="text"/>	name=username
Xpath	<form id="login">	xpath=//form[@id='login']
Link Text	Cancel	link=Cancel
DOM	<form id="login">	dom=document.forms['loginForm']
CSS	<form id="login">	Css=form#loginForm
Tag Name	<form id="login">	tagName=form

LOCATOR CATEGORIES

Structure-based locators: locators that rely on the structure of the page to find elements:

1) XPath
2) DOM
3) CSS

Attributes-based locators: locators that relies on the attributes of the elements to locate them:

1) Identifier
2) ID
3) Name
4) Link
5) CSS

Consider this before choosing a locator strategy

BY CLASS

Java	C#	Description
By.id(String id)	By.Id(String id)	Unique id
By.linkText(String linkText)	By.LinkText(String linkText)	Link text
By.name(String name)	By.Name(String name)	Name attribute
By.partialLinkText(String linkText)	By.PartialLinkText(String linkText)	Part of link text
By.tagName(String name)	By.TagName(String name)	Search by HTML tag
By.xpath(String xpathExpression)	By.Xpath(String xpathExpression)	XPATH expression
By.className(Sting name)	By.ClassName(Sting name)	CSS class

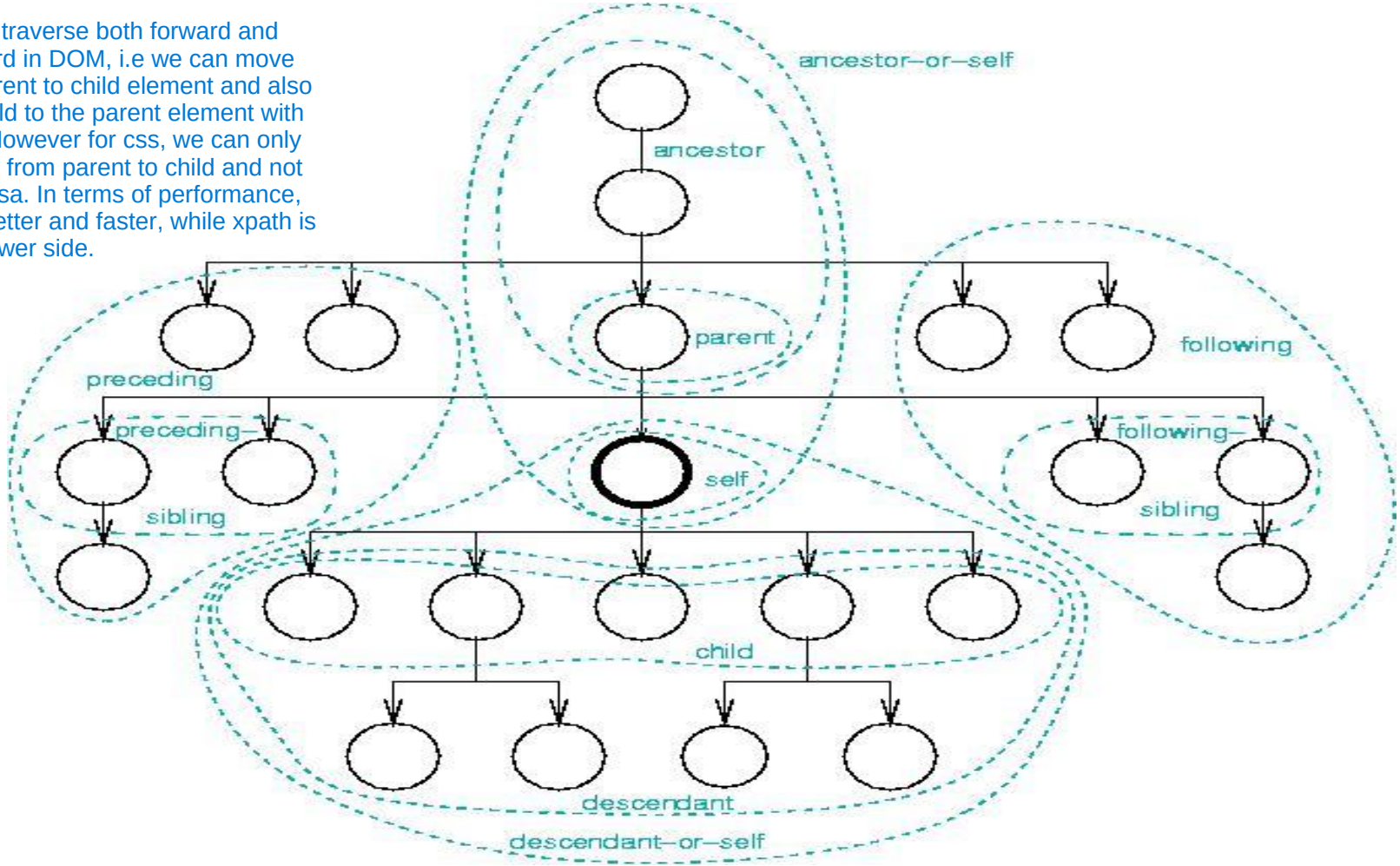
TERMS

XPath is ...

- a syntax
- a pattern
- a language

... for defining parts of an XML document.

We can traverse both forward and backward in DOM, i.e we can move from parent to child element and also from child to the parent element with xpath. However for css, we can only traverse from parent to child and not vice-versa. In terms of performance, css is better and faster, while xpath is on a slower side.



XPath AXES

AxisName	Result
ancestor::	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self::	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
attribute::	Selects all attributes of the current node
child::	Selects all children of the current node
descendant::	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self::	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
following::	Selects everything in the document after the closing tag of the current node
following-sibling::	Selects all siblings after the current node
namespace::	Selects all namespace nodes of the current node
parent::	Selects the parent of the current node
preceding::	Selects all nodes that appear before the current node in the document, except ancestors
preceding-sibling::	Selects all siblings before the current node
self::	Selects the current node

LOCATORS. XPATH

Xpath

- provides basic **facilities for manipulation** of strings, numbers and Booleans
- XPath models an XML document as a **tree of nodes**
- relative location path consists of a sequence of one or more **location steps** separated by “/”

A location step has three parts:

axis - specifies the tree relationship between the nodes selected by the location step and the context node
node test - specifies the node type and expanded-name of the nodes selected by the location step
zero or more **predicates** - use arbitrary expressions to further refine the set of nodes selected by the location step

./child::html/child::body/child::*/child::span[attribute::class=‘dummy_class’])

More examples:
child::para - selects the para element children of the context node
child::* - selects all element children of the context node
child::text() - selects all text node children of the context node

XPATH AXES SHORT SYNTAX

attribute::	@	
child::		(becomes
descendant::	/)	
parent::	/ (becomes //)	
self::	..	
	.	

“TRADITIONAL” XPATH LOCATOR

```
//input[@value='press me']
```

select all | 'input' nodes | in which | 'value' parameter | equals | 'press me'

```
.//input[@value='press me']
```

root node (not necessary) | select all | 'input' nodes | in which | 'value' parameter | equals | 'press me'

XPATH OPERATORS

Operator	Description	Example
|	Computes two node-sets	//book | //cd
+	Addition	6 + 4
-	Subtraction	6 - 4
*	Multiplication	6 * 4
div	Division	8 div 4
=	Equal	price=9.80
!=	Not equal	price!=9.80
<	Less than	price<9.80
<=	Less than or equal to	price<=9.80
>	Greater than	price>9.80
>=	Greater than or equal to	price>=9.80
or	or	price=9.80 or price=9.70
and	and	price>9.00 and price<9.90
mod	Modulus (division remainder)	5 mod 2

XPATH EXAMPLE – SOURCE XML

```xml
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>

<book>
  <title lang="en">Learning XML</title>
  <price>39.95</price>
</book>

</bookstore>
```

XPATH EXAMPLE – PREDICATES

Path Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='en']	Selects all the title elements that have a "lang" attribute with a value of "en"
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

LOCATORS. WILDCARD - SELECTING UNKNOWN NODES

Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

XPath Expression	Result
/bookstore/*	Selects all the child element nodes of the bookstore element
//*	Selects all elements in the document
//title[@*]	Selects all title elements which have at least one attribute of any kind

LOCATORS. SELECTING BY SEVERAL FILTERS

Path Expression	Result
//book/title | //book/price	Selects all the title AND price elements of all book elements
//title | //price	Selects all the title AND price elements in the document
/bookstore/book/title | //price	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document

MORE REALISTIC XPATH LOCATORS EXAMPLES

```
//button[@id='submit']

//div[not(@style='display:none')]/a

//*[text()='the visible text' and @class = 'class1']

//li[contains(@value, 'choice1')]

//span[count(div)>=3]

//*[@class='search-result']/descendant-or-self::*[@id='banana']

//a[ends-with(@href, '.pdf')]
```

XPATH - USEFUL LINKS

XPath syntax

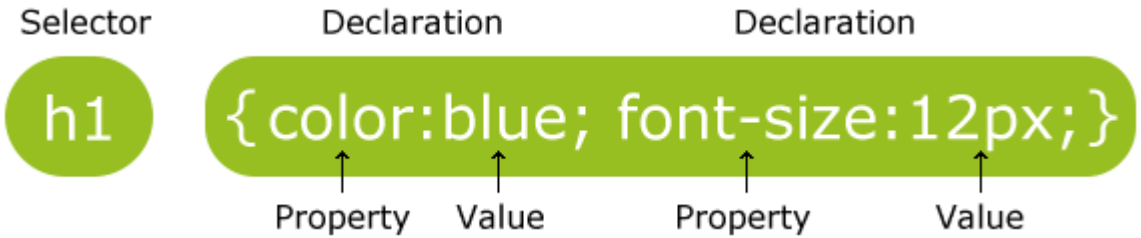
XPath axes

XPath operators

XPath functions

CSS SELECTORS

CSS Selectors work almost the same way, but use style attributes of elements.



They have more specific function and different syntax.

CSS SELECTOR - HOW DOES IT LOOK?

```
div#qwe3 p.intro>a[href^='https']:visited
```

search all 'div' nodes

where ID= 'qwe3'

empty space is equal to '//' in XPath

'p' nodes

where class= 'intro'

'>' is equal to '/' in XPath

'a' node

where 'href' parameter

begins with 'https'

and was already visited

it is pretty hard example :)

CSS SELECTOR EXAMPLES

```
1  <html>
2   <body>
3    <form id="loginForm">
4     <input class="required" name="username" type="text" />
5     <input class="required passfield" name="password" type="password" />
6     <input name="continue" type="submit" value="Login" />
7     <input name="continue" type="button" value="Clear" />
8    </form>
9  </body>
10 <html>
```

- form#loginForm (line number 3)
- input[name="username"] (line number 4)
- input.required[type="text"] (line number 4)
- input.passfield (line number 5)
- #loginForm input[type="button"] (line number 7)
- #loginForm input:nth-child(2) (line number 5)

CSS SELECTOR EXAMPLES

h1

div.tagline

li#intro a

div.result > a

li ~ ul

a[href$=".pdf"]

span.alpha div#id > a:visited

CSS SELECTORS - USEFUL LINKS

CSS Selectors syntax

CSS Selectors examples

CSS Selectors online tester

XPATH - CSS SELECTORS INTERCHANGEABILITY

div.ch #pnnext>span:nth-child(3)

//div[@class='ch']//*[@id='pnnext']/span[3]


Demo

WHAT IS BETTER TO USE IN TEST AUTOMATION?

CSS Selector

- works faster
- looks laconic
- has some unique features (like :visited)
- harder to use, read and understand
- can walk on DOM only down direction
- can not search by text

XPath

- the most frequently used
- easier to use, read and understand
- more powerful
- can walk on DOM in any directions
- has functions and operators

But it is better to know the both :)

LOCATORS: FREQUENTLY MADE MISTAKES

- using absolute, generated or too long path
  `//span[@id='tsf']/div[2]/div[3]/center/input[2]`
- using non-static properties
  `//th[@class='jfk-bubble chw-oc stv-enabled']`
- creating locators that return non-definite results
  `//a[contains(text(),'hello')]`
- using double quotes in values
- ignoring the frames
  `//input[@id='pass_input']`
- using difficult dependencies when simple location (id) could be used
  `//th[contains(@class, 'jfk-bubble')]`
  `//popup[@id='first_run_popup']/a`
- being sure that locator will never change

EXAMPLES. WEBDRIVER API + LOCATORS

void ***sendKeys(****String text****)*** / ***SendKeys(****String text****)*** - Entering text

```
element.sendKeys("sometext");
element.sendKeys("sometext", Keys.ARROW_DOWN);
```

void ***clear()*** / ***Clear()*** - clear content

```
element.clear();
```

String ***getAttribute****(String attributeName)* / ***GetAttribute****(String attributeName)* - getting attribute

```
        String var = webDriver.findElement(By.xpath("//input[@name='q']")).getAttribute("
maxlength")
        System.out.println(var);
```

void ***click()*** / ***Click()*** - clicking on element

```
driver.findElement(By.id("next_page")).click();
```

Regular Expressions Basics

Regular Expressions are special string masks for describing a search pattern in text or content.

It is frequently used in programming for parsing some specified values, controlling some parameters format, filtering some data, etc.

In this lection we will consider just basic techniques of RegEx (or RegExp).

RegEx - Syntax

Ordinary characters - mean themselves.

Special characters (metacharacters): **[ ] \ / ^ $. | ? * + - ( ) { }**

can have different meaning depends of context (position)

To use special symbol as itself, it should be escaped with **** .

Some ordinary characters become metacharacters when are escaped with **** :

d s w D S W

(not full list, but most frequently used chars)

REGEX - SYNTAX

.	matches any character except a newline
*	matches the preceding element zero or more times
+	matches the preceding element one or more times
?	matches the preceding element zero or one times
[ ]	denotes a set of possible character matches
[^]	matches every character except ones inside brackets
[a-z]	indicates a range - any between first and last chars
(a|x)	works like ‘or’ operator
{m,n}	matches the quantity of preceding element from m to n
\d	matches any digit char
\D	matches non-digit char
\w	matches any alphanumeric, digits character and ‘_’ (word)
\W	matches non-word char
\s	matches any whitespace char
\S	matches non-space char
?	turns preceding “greedy” matching to “lazy” (watch examples)

REGEX - EXAMPLES

home.	home1, home2, homez
h.*	h, homez, homespace.txt
a+	a, aa, aaaaaaaa
home4?	home4, home
[qwerty]	q,e,t,w,y (just one char)
[^123\s]+	5, aqw45, wer789!#
[a-e]+	abc, a, adeca
have (1|5) points?\?	have 1 point? , have 5 points?
cool!{1,3}	cool! , cool!! , cool!!!
\d+	1, 126, 156989479
\D+	Qwe , aaa#!$, o__O
\w+@\w+\.\w{3}	sanya_538@qwe.com
I have \d+ apples?	I have 1 apple, I have 3 apples
web\s?element	webelement, web element
Offer #\d+ was (not)?changed	Offer #333 was not changed Offer #123 was changed
	greedy and lazy matching:
<.+>	<a>
<.+?>	<a>

REGEX - USEFUL LINKS

The most user-friendly online parser

More powerful online parser

The most powerful online parser


Demo

BONUS PART: SPECIALS

SPECIALS. WAITS

Implicit wait

Java:

```
WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.get("http://somedomain/url_that_delays_loading");
WebElement myDynamicElement = driver.findElement(By.id("myDynamicElement"));
```

C#:

```
IWebDriver driver = new FirefoxDriver();
driver.Url = "http://somedomain/url_that_delays_loading";
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
IWebElement myDynamicElement = wait.Until<IWebElement>((d) =>
{
        return d.FindElement(By.Id("someDynamicElement"));
});
```

SPECIALS. WAITS

Explicit wait

Java:

```
WebDriver driver = new FirefoxDriver();
driver.get("http://somedomain/url_that_delays_loading");
WebElement myDynamicElement = (new WebDriverWait(driver, 10))
  .until(new ExpectedCondition<WebElement>(){
        @Override
        public WebElement apply(WebDriver d) {
                return d.findElement(By.id("myDynamicElement"));
        }});
```

C#:

```
IWebDriver driver = new FirefoxDriver();
driver.Url = "http://somedomain/url_that_delays_loading";
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
IWebElement myDynamicElement = wait.Until<IWebElement>((d) =>
{
            return d.FindElement(By.Id("someDynamicElement"));
});
```

SPECIALS. WAITS

User wait

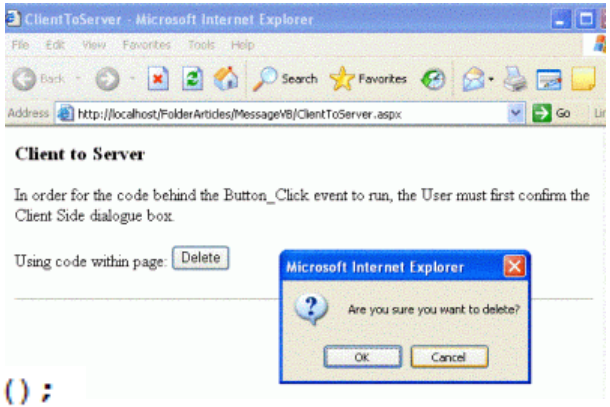
```
Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
    .withTimeout(30, TimeUnit.SECONDS)
    .pollingEvery(5, TimeUnit.SECONDS)
    .ignoring(NoSuchElementException.class);

WebElement nextPage = wait.until(new Function<WebDriver, WebElement>() {
  public WebElement apply(WebDriver driver) {
    return driver.findElement(By.id("next_page"));
  }
```

Thread.sleep(timeout)

(try to avoid, use for debug only)

```
Thread.sleep(5000);
driver.findElement(By.id("next_page")).click();
```

SPECIALS. POPUP WINDOWS




```
WebDriver driver = new FirefoxDriver();

/**
 * Do something
 */

Alert alert = driver.switchTo().alert();
System.out.println(alert.getText());
alert.accept();
```

SPECIALS. JAVASCRIPT

Enabling JavaScript

```
HtmlUnitDriver driver = new HtmlUnitDriver();
driver.setJavascriptEnabled(true);
```

JavaScript usage

```
WebDriver driver = new FirefoxDriver();
driver.get("http://ya.ru");
WebElement input = (WebElement) ((JavascriptExecutor) driver).
    executeScript("return document.getElementById('text');");
input.sendKeys("JavaScript");
```

SPECIALS. SELECT

Working with Lists:

```
WebElement select = driver.findElement(By.id("gender"));
List<WebElement> options = select.findElements(By.tagName("Male"));
for (WebElement option : options) {
   if("Germany".equals(option.getText()))
      option.click();
}
```

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

new Select(driver.findElement(By.id("gender"))).selectByVisibleText("Germany");
```

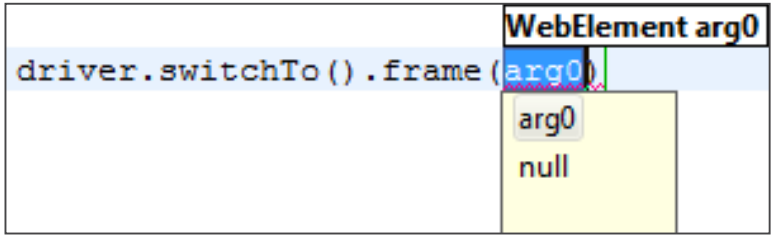
SPECIALS. ISELEMENTPRESENT()

```
public static WebDriver driver;

public static boolean isElementPresent(By locator) {
    try {
        driver.findElement(locator);
    } catch(NoSuchElementException e) {
        return false;
    }
    return true;
}
```

```
return driver.findElements(By(locator)).size() > 0;
```

SPECIALS. FRAMES

- Frame - element we can find by locator



- Get back to root content

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

SPECIALS. WORK WITH WINDOWS

```
driver.switchTo().window("windowName");
```

```
for (String handle : driver.getWindowHandles()) {
    driver.switchTo().window(handle);
}
```

```
Set<String> handles = driver.getWindowHandles();
driver.switchTo().window(handles.toArray(new String[handles.size()])[0]);
```

RESOURCES ABOUT WEBDRIVER

1) http://seleniumhq.org/projects/webdriver/ - official WebDriver definition
2) http://rus-linux.net/MyLDP/BOOKS/Architecture-Open-Source-Applications/Vol-1/selenium-01.html – Selenium project history in Russian
3) http://www.aosabook.org/en/selenium.html - Selenium project history in English language.
4) http://www.seleniumhq.org/about/history.jsp - Selenium project history in official WD resource
5) http://seleniumhq.org/download/ - download page for components
6) https://github.com/seleniumhq/selenium - alternative source
7) http://seleniumhq.org/docs/03_webdriver.html - basic information about Selenium WebDriver
8) http://toolsqa.com/selenium-webdriver/ - well supported and structured resource about automation via WebDriver. Contains many best practices.

SELENIUM WEBDRIVER. LECTURE 1. SUMMARY

- Selenium is an open source project
- WebDriver is a library for browser management (main product under Selenium)
- Selenium IDE is a Firefox extension for simple test record and playback
- There are drivers for all popular browsers and mobile platforms
- WebDriver 2.0 (3.0) will be a standard interface to control a browser (is ongoing)
- WebDriver API is easy and clear (13 methods)
- There are 8 locator strategies (every has its pros and cons – select the best for your project)
- Locator can be get with Xpath or CSS. Use your own judgement
- WebDriver supports explicit and implicit waits to manage interaction timeouts
- There are some tricky things with synchronization and windows/popups/frames handling

HOME TASK

Create automated tests for the AUT using:

- WebDriver + Java + TestNG

OR

- WebDriver + C# + NUnit

Thanks for attention

PRESENTER: SERGEY PORITSKIY