Selenium Introduction , History , First Script and Locators

Selenium History

2004

Jason Huggins

In 2004 Jason Huggins, an engineer at Thoughtworks (Chicago) was working on a web application that required frequent testing.

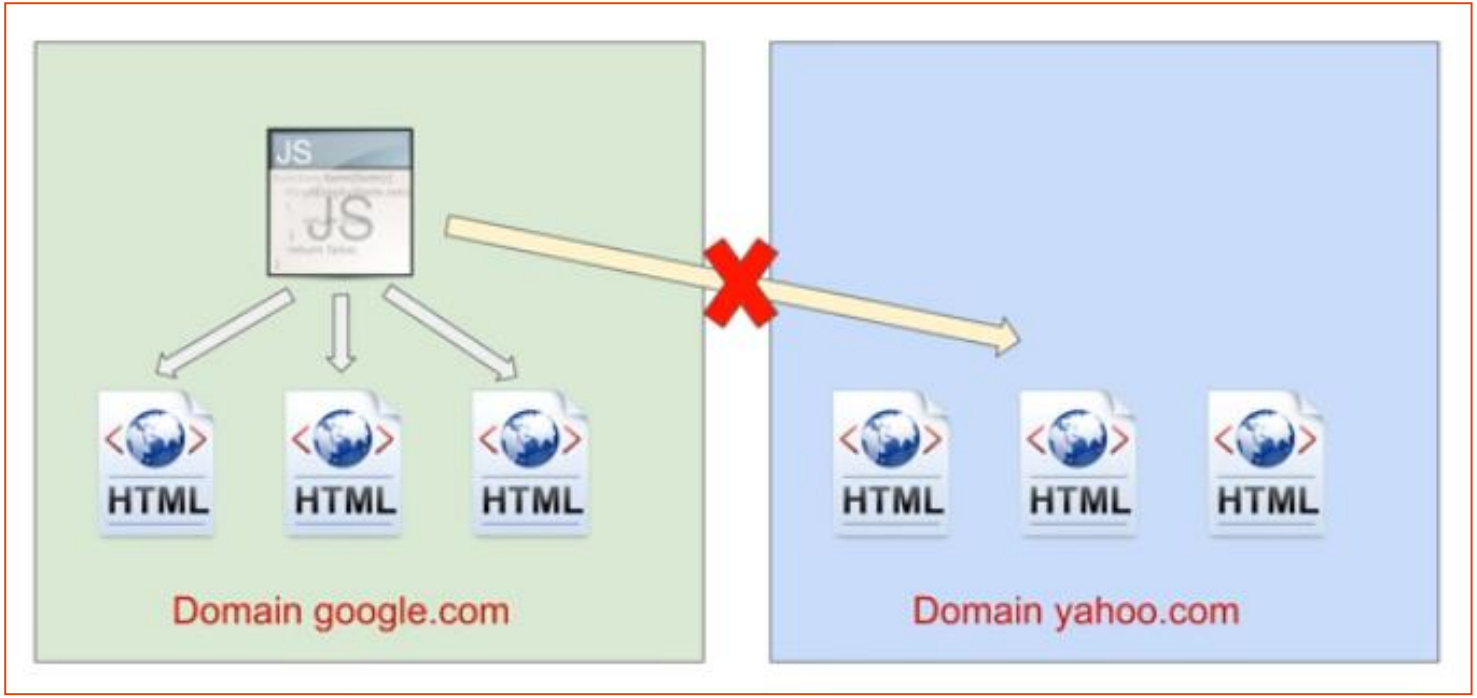
Manual testing was becoming hard as it was time-consuming and inefficient.

He wrote a Javascript program that could interact with the browser and do actions.

He called it "**Javascript Test Runner**"

He started giving demos about this tool and soon there were discussions to make it open source as a re-usable testing framework.

Same Origin Policy Issue



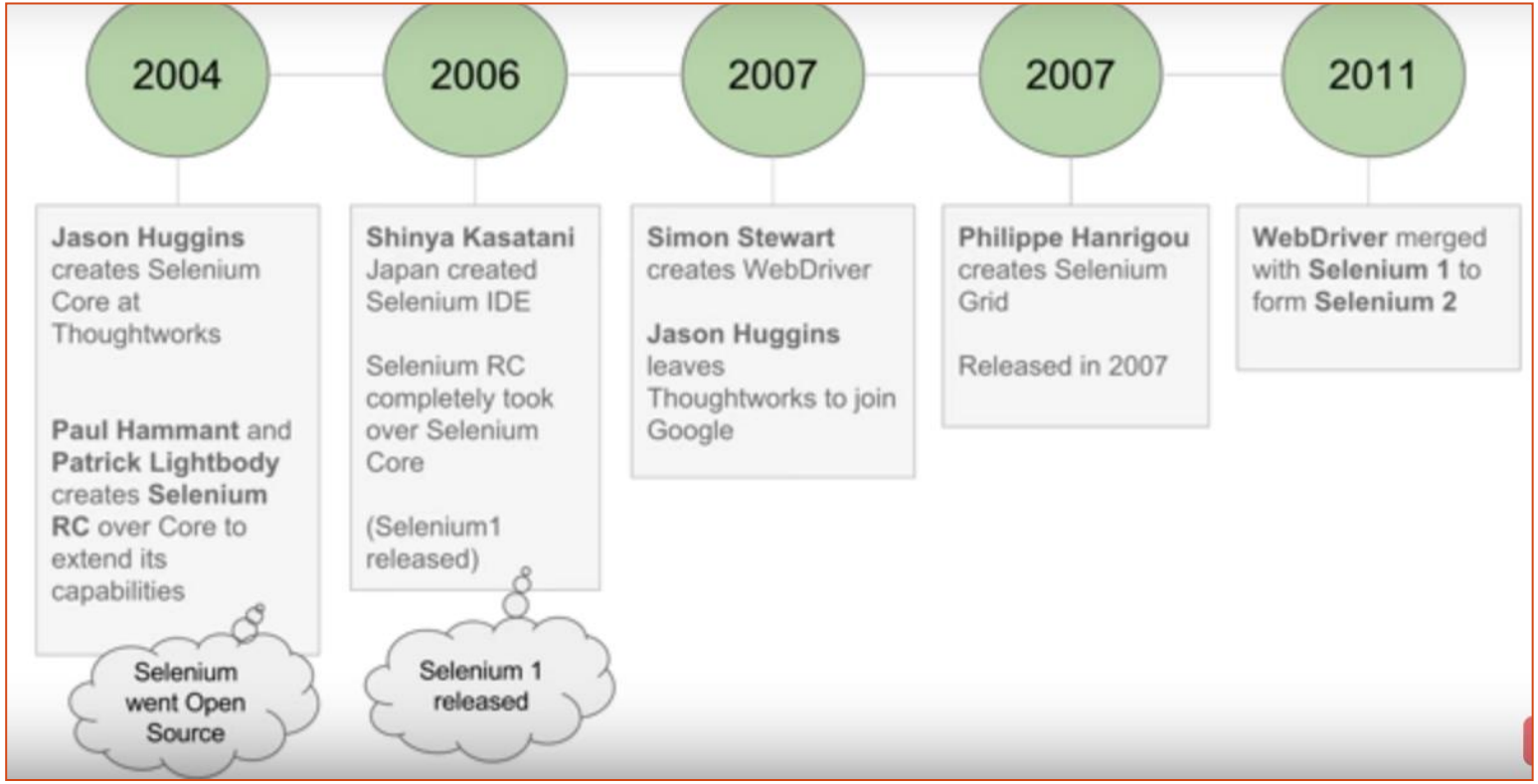
Selenium History

Selenium History

During the time Selenium was being created, there was a another popular web testing tool QTP developed by Mercury Interactive.

In one of the emails Jason Huggins jokingly said, "you need **Selenium** supplements to cure **Mercury** poisoning".
The name caught on from there.

Selenium History

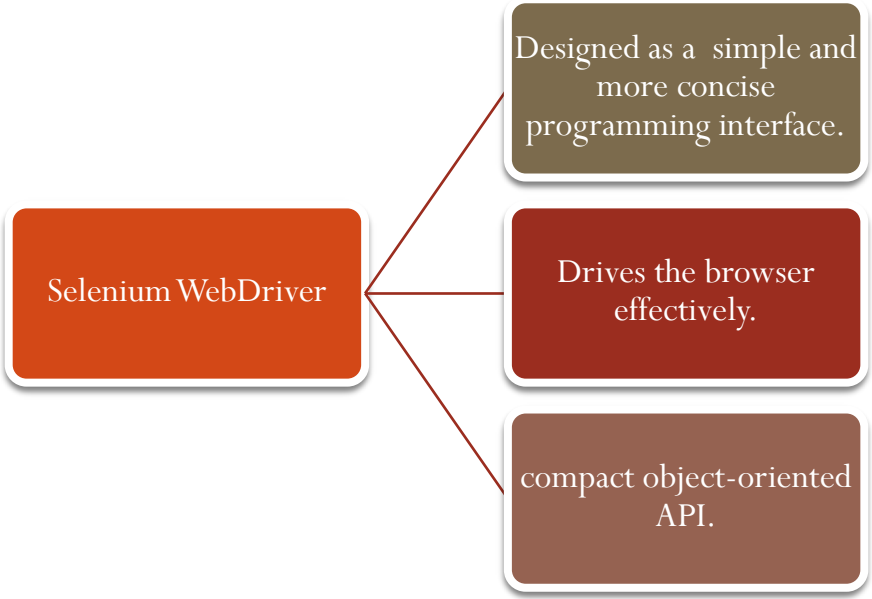


Selenium History

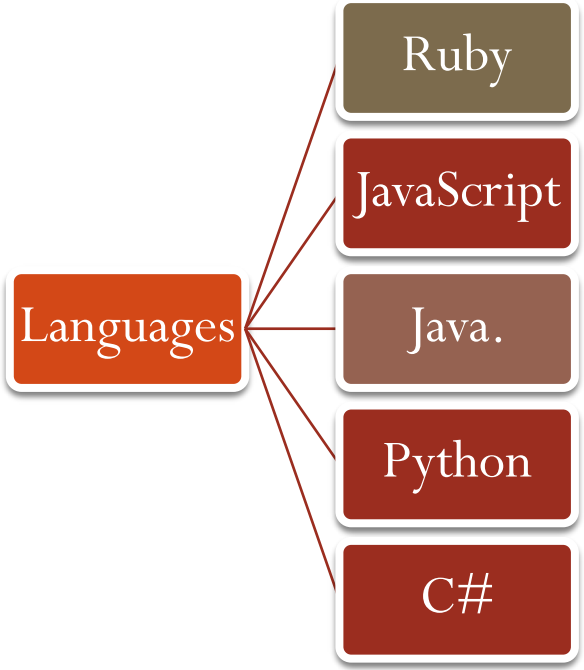
Selenium WebDriver Introduction

Selenium WebDriver

- Selenium History # https://selenium.dev/history/
- Selenium automates the browser



Selenium WebDriver – Supported Languages



Selenium WebDriver – Supported Browsers

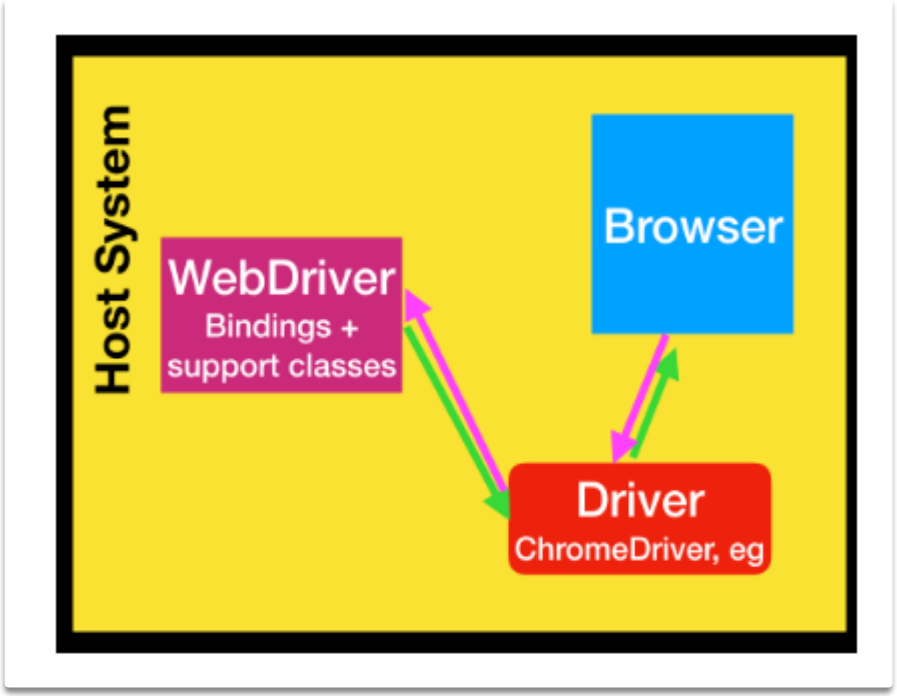
The Selenium framework officially supports the following browsers:

Browser	Maintainer	Versions Supported
Chromium	Chromium	All versions
Firefox	Mozilla	54 and newer
Internet Explorer	Selenium	6 and newer
Opera	Opera Chromium / Presto	10.5 and newer
Safari	Apple	10 and newer

Driver Name	Purpose	Maintainer
HtmlUnitDriver	Headless browser emulator backed by Rhino	Selenium project

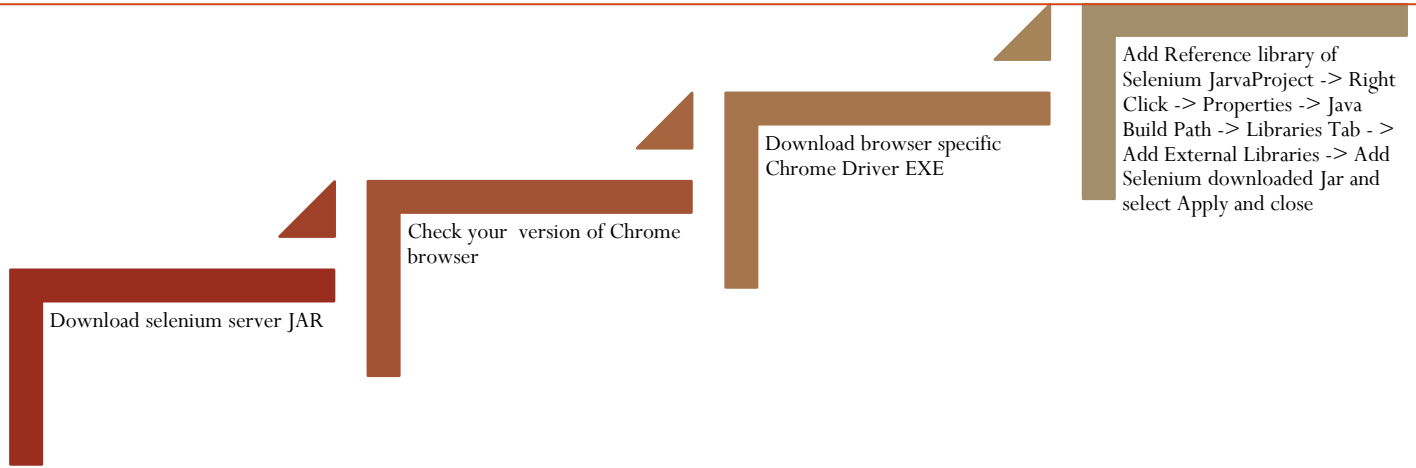
Selenium WebDriver – Architecture

- WebDriver talks to a browser through a driver.
- Communication is two way
- WebDriver passes commands to the browser through the driver, and receives information back via the same route



How to Download and Install Selenium

Selenium Jar – Download



Download and Install - Demo

First Selenium Script

1

```
System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");
```

Remote server that instructs the browser what to do by exposing Chrome's internal automation proxy interface.

2

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

WebDriver driver = new ChromeDriver();
```

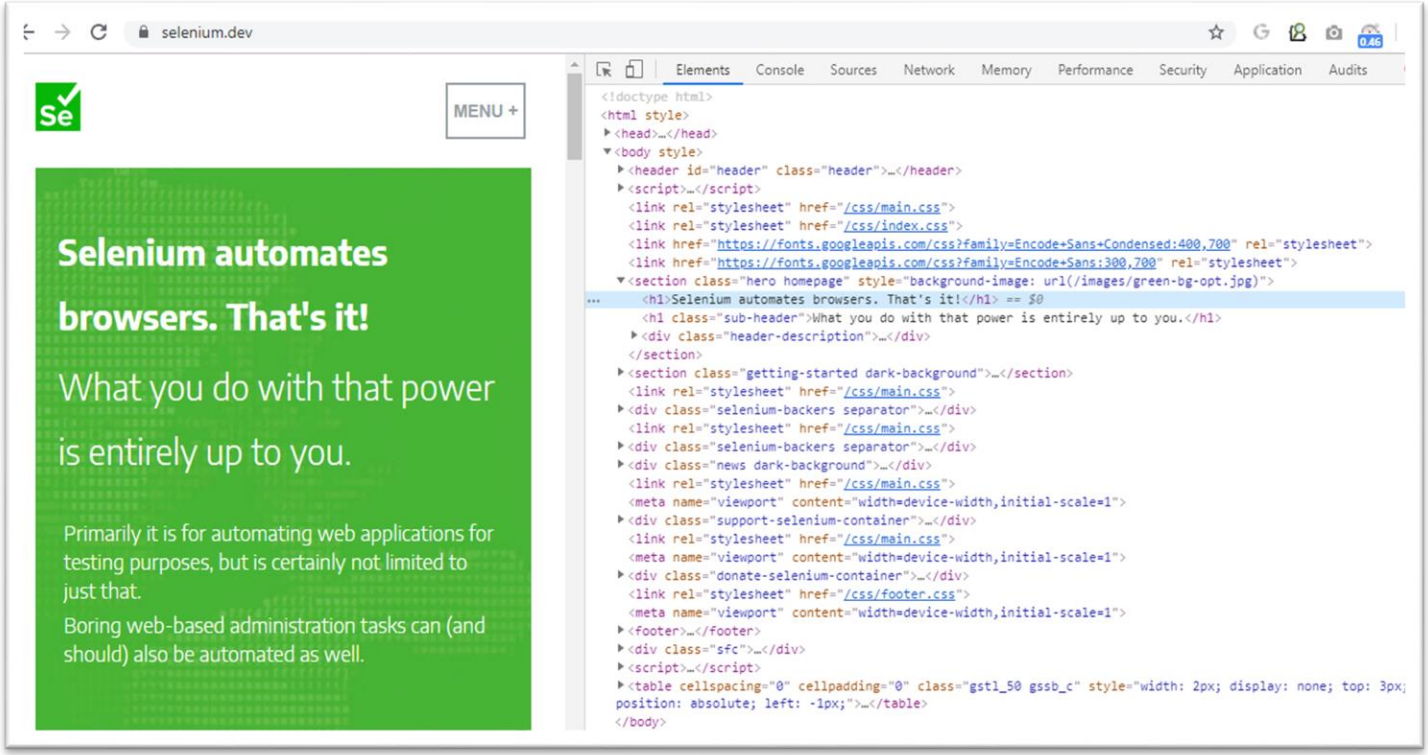
Interface

Class

Selenium Script - Demo

Locating Elements using selenium locators

DOM – Tree structured xml and Html document



DOM – Introduction

```
<html style>
▶ <head>…</head>
▼ <body style>
  ▶ <header id="header" class="header">…</header>
  ▶ <script>…</script>
    <link rel="stylesheet" href="/css/main.css">
    <link rel="stylesheet" href="/css/index.css">
    <link href="https://fonts.googleapis.com/css?family=Encode+Sans+Condensed:400,700" rel="stylesheet'
    <link href="https://fonts.googleapis.com/css?family=Encode+Sans:300,700" rel="stylesheet">
  ▼ <section class="hero homepage" style="background-image: url(/images/green-bg-opt.jpg)">
      <h1>Selenium automates browsers. That's it!</h1> == $0
      <h1 class="sub-header">What you do with that power is entirely up to you.</h1>
    ▶ <div class="header-description">…</div>
    </section>
  ▶ <section class="getting-started dark-background">…</section>
    <link rel="stylesheet" href="/css/main.css">
  ▶ <div class="selenium-backers separator">…</div>
    <link rel="stylesheet" href="/css/main.css">
  ▶ <div class="selenium-backers separator">…</div>
  ▶ <div class="news dark-background">…</div>
    <link rel="stylesheet" href="/css/main.css">
    <meta name="viewport" content="width=device-width,initial-scale=1">
  ▶ <div class="support-selenium-container">…</div>
    <link rel="stylesheet" href="/css/main.css">
    <meta name="viewport" content="width=device-width,initial-scale=1">
  ▶ <div class="donate-selenium-container">…</div>
    <link rel="stylesheet" href="/css/footer.css">
    <meta name="viewport" content="width=device-width,initial-scale=1">
  ▶ <footer>…</footer>
  ▶ <div class="sfc">…</div>
  ▶ <script>…</script>
  ▶ <table cellspacing="0" cellpadding="0" class="gstl_50 gssb_c" style="width: 2px; display: none; top
  position: absolute; left: -1px;">…</table>
  </body>
</html>
```

- Document Object Model - Tree structured xml and Html document
- Tag Started – Tag Ended
- Parent and child Tags
- Attributes has values

Selenium Locators

Locator	Description
class name	Locates elements whose class name contains the search value (compound class names are not permitted)
css selector	Locates elements matching a CSS selector
id	Locates elements whose ID attribute matches the search value
name	Locates elements whose NAME attribute matches the search value
link text	Locates anchor elements whose visible text matches the search value
partial link text	Locates anchor elements whose visible text matches the search value
tag name	Locates elements whose tag name matches the search value
xpath	Locates elements matching an XPath expression

Selenium Locators Demo – id , name , Class ,

```
WebElement MenuButton= driver.findelement (By.id("dropdownButton"))

WebElement Searchbox= driver.findelement (By.name("search"))

WebElement Searchbox= driver.findelement (By.class("gsc-input"))
```

- findElement(By) method returns another fundamental object type, the WebElement
- WebDriver represents the browser
- WebElement represents a particular DOM node (a control, e.g. a link or input field, etc.)

Selenium Locators Demo – linkText, partial linkText

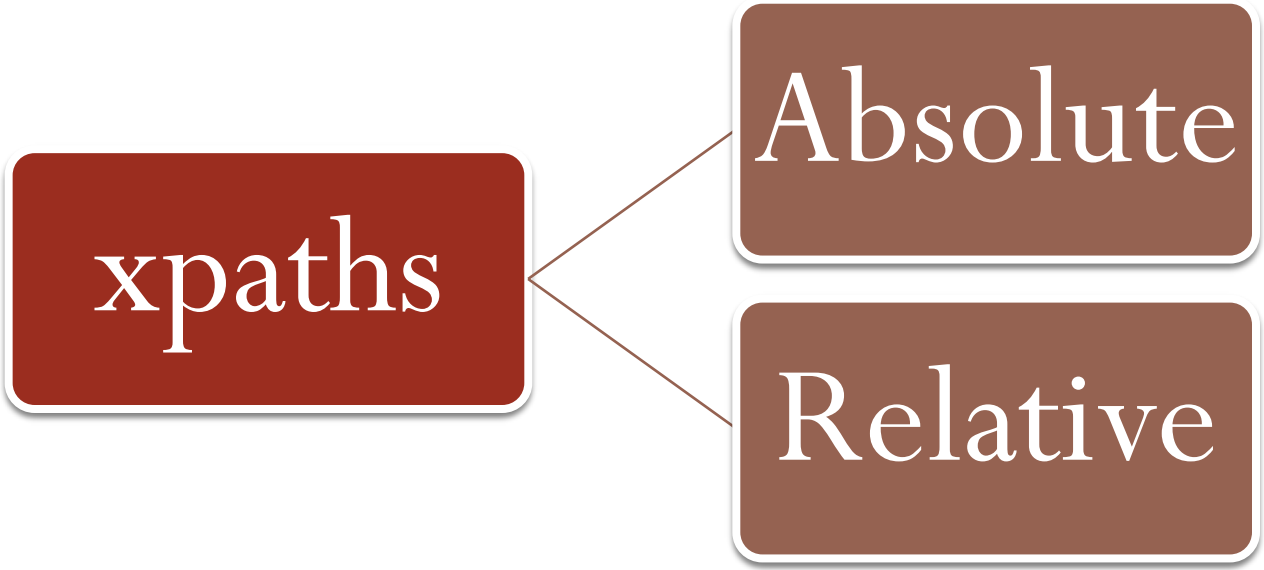
```
WebElement documentationlink= driver.findelement (By.linktext("Documentation"))

WebElement documentationlink = driver.findelement (partiallinktext("Doc"))
```

Xpaths is selenium for Beginners

Selenium Locators Demo – Xpath Beginners

```
WebElement documentationlink= driver.findelement (By.xpath("XpathExpression"))
```



Selenium Locators Demo – Xpath – Beginners

Locating strategy	Syntax	Example
Using Text	//tagname [text()="TextWithingTag"]	//span[text()="About"]
Using attribute value	//tagname [@attributename="value"]	//input[@title="search"]
Following and index	//tagname[text()="Blog"]/following::tagname[index]	//a[text()="Blog"]/following::input[1]
Preseding and index	//tagname[text()="Blog"]/preceding::tagname[index]	//a[text()="Blog"]/preceding::input[1]
Using And	//tagname[@attributename="attributevalue" and @attributename="attributevalue"]	//input[@type="text" and @id="user-message"]
Using OR	//tagname[@attributename="attributevalue" or @attributename="attributevalue"]	//input[@type="text" or @id="user-message"]
Using contains	Xpath=//tagname[contains (@attribute, ‘value)]	//input[contains(@type,'text')]

Selenium Locators Demo – Xpath – Beginners

Locating strategy	Syntax	Example
Following and index	//tagname[text()="Blog"]/following-sibling::tagname[index]	//label[text()=' Total a + b = ']/following-sibling::span
Preseding and index	//tagname[text()="Blog"]/preceding-sibling::tagname[index]	//span[@id='displayvalue']/preceding-sibling::label
Child		//form[@id='gettotal']/child::div/label[text()='Enter a']
Parent		//label[text()='Enter a']/parent::div[1]
ancestor	GrandParent	//label[text()='Enter a']/ancestor::form
descendant	GrandChild	//form/descendant::label[text()='Enter message']

CSS Selecters is selenium for Beginners

Selenium Locators Demo – CSSSelecters Beginners

Strategy	Syntax	Example
Class Name	<.> < Class Name>	.form-control
ID Selector Conundrum	<#> < ID Name>	#user-message
Attribute Selector Conundrum	tag name [attribute key = ‘attribute value’]	button[type='button']
	OR [attribute key = ‘attribute value’]	[type='button']
‘Class or ID’ & Attribute Selector Conundrum	<HTML Tag><.> <Class Or ID> [<attribute key> = ‘<attribute value>’]	input.form-control[type='text']

Selenium Locators Demo – CSS Selecters Beginners

Strategy	Syntax	Example
SubString Match Conundrum		
^ Indicating a prefix match	<HTML Tag> [<Class Or ID> ^= <Class or ID Name>]	input[class^='form']
$ Indicating a suffix match	<HTML Tag> [<Class Or ID> $= <Class or ID Name>]	input[class$='-control']
* Indicating a substring match	<HTML Tag> [<Class Or ID> *= <Class or ID Name>]	input[class*='-control']
	<HTML Tag> <:> <contains> < (text) >	label:contains("^Enter message$")
Navigating Through Child Elements	Parentlocator > directchildlocator	div.form-group>label
	<HTML Tag> .class>childTag	
	<parent locator=""> "<Space> "<child locator="">	div.form-group label
Nth Child For Opting A Specific Value From A List	HTML Tag> <Clas or ID> <list> <:> <nth-child (number of desired item in the list)>	div.form-group input:nth-child(2)

Why Choose CSS Selectors Over Other Element Identifiers?

❖Faster Identification and reduced test execution time – Compared to XPath CSS selectors would tend to identify the elements better as most used browsers such as Chrome and Firefox are tuned for better performance with CSS selectors. Here is the link which provides performance stats for reference.

❖Availability of better documentation.

❖Enhanced readability.