WebDriver driver = new FirefoxDriver() – Why we write in Selenium Scripts

Let’s see why can’t we use the following statement.

**WebDriver driver = new WebDriver();**

We cannot write our code like this because we cannot create Object of an Interface. WebDriver is an interface.

But we can use any of the following statements in our script

FirefoxDriver driver = new FirefoxDriver();

or

WebDriver driver = new FirefoxDriver();

**FirefoxDriver driver = new FirefoxDriver();**

The FirefoxDriver instance which gets created based on above statement will be only able to invoke FirefoxDriver and supports Firefox Browser only. Using this statement, we can run our scripts only on Firefox Browser.

To work with other browsers, we have to specifically create individual objects as below:

ChromeDriver driver = new ChromeDriver();  
InternetExplorerDriver driver = new InternetExplorerDriver();

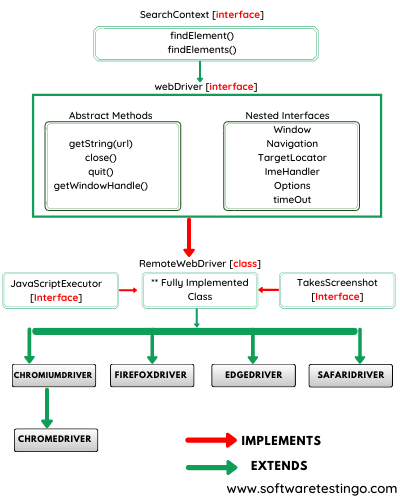
We don’t just run our scripts only on single browser. We use multiple browsers for [Cross Browser Compatibility](https://www.softwaretestingmaterial.com/cross-browser-testing/). We need the flexibility to use other browsers like ChromeDriver() to run on Chrome Browser and InternetExplorerDriver() to run on IE Browser and so on.

So, once you initiate a Firefox browser using FirefoxDriver driver = new FirefoxDriver(); same object cannot be used to initiate Chrome Browser (you have to rename it)

ChromeDriver driver = new ChromeDriver();

To solve this we use “Webdriver driver = new FirefoxDriver();” here we just have to replace FirefoxDriver to InternetExplorerDriver OR ChromeDriver. It is more easy method. Hence we use ChromeDriver driver = new ChromeDriver();

#### Selenium **WebDriver** Interface Hierarchy 🡪



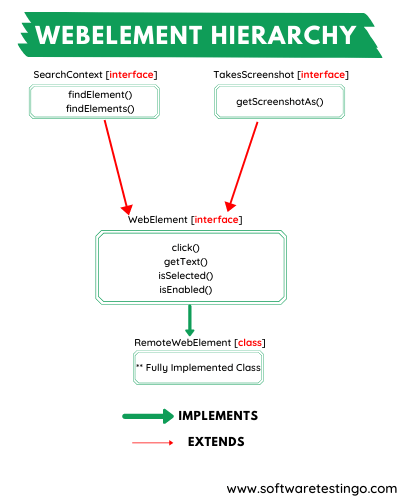
Let us explain the above hierarchy in details:

* **SearchContext** is the topmost interface of Webdriver which contains only two abstract method findElement() and findElements(). These two methods don’t have a method body.
* **WebDriver** also is an interface which extends SearchContext interface which has also so many abstract methods like close(), get(java.lang.String URL), quite(), navigate(), switchTo() and other so many methods for more details you can visit this [URL](https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/WebDriver.html)
* The next one is **RemoteWebDriver**, which is a fully implemented class where all abstract methods of WebDriver and SearchContext interface implemented. Also, two other interfaces JavascriptExecutor and TakesScreenshot abstract methods are implemented in RemoteWebDriver class.
* And Finally, browser-specific driver classes available like FirefoxDriver, ChromeDriver, IEDriver, SafariDriver, etc.

WebDriver driver = new ChromeDriver();

This means you are creating instance of ChromeDriver. In this case our driver object will access all the methods implemented in ChromeDriver class. Also it has access to all the methods of WebDriver and SearchContexts interfaces which is implemented in RemoteWebDriver as ChromeDriver class is extending.

## **WebElement interface Hierarchy 🡪**



* The webElement interface extends two other interfaces like SearchContext and TakesScreenshot interfaces. the webElement interface has so many useful methods that are frequently used during the automation. those methods like clear(), click(), getText(), submit() etc.
* RemoteWebELement is a class which implements all the abstract methods of the webElement interface
* When you try to search an element in DOM then you are going to use findElement() and findElements() then the remoteWebelement class is up-casting to webElement interface, and if you perform any operation like click() or submit(), then the overridden method of the remoteWebelement class is executed.