



Upload – Actions – JSExecutor





Me waiting to
tell him that
this language
was created 5
years ago


Recruiter explaining
why they need 10
years of expertise of
a language for their
job

AGENDA: UPLOADS – DOWNLOADS

- How do we handle uploads using Selenium?
- How do we handle downloads using Selenium?
- How do we do advanced mouse and keyboard actions?



How do you handle **downloads** using Selenium?

- You don't. 
- You can't download a file using Selenium, because computer file structure will be out of scope for Selenium
- Selenium can **not** reach outside of the browser.

How do you handle **downloads** using **Java**?

- We can use regular Java classes to verify a file has been downloaded.
- Such as; **File** class or **FileInputStream** class.

```
File file = new File(pathname: "path/to/downloads/filename.extension");  
Assert.assertTrue(file.exists());
```

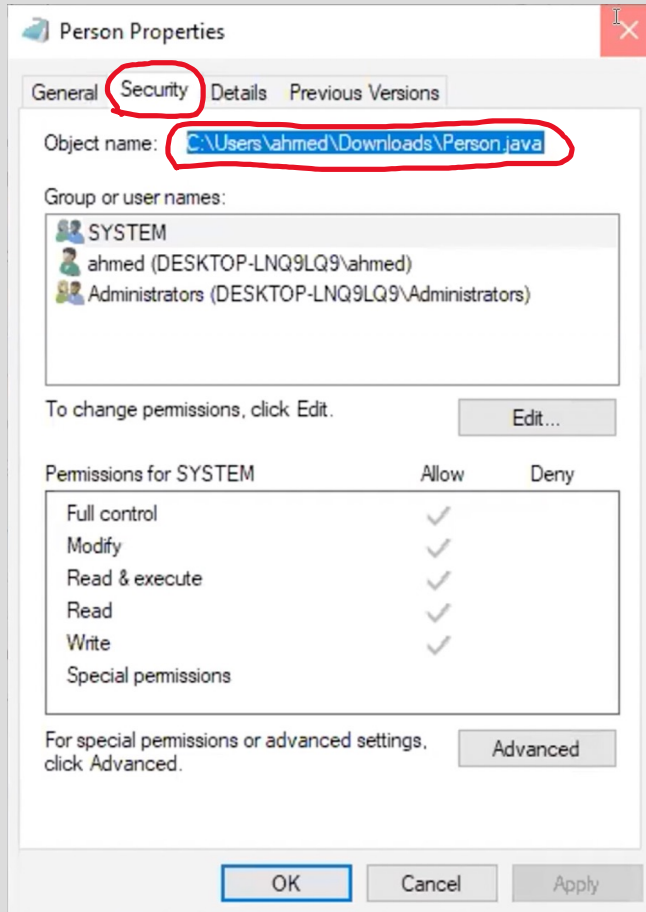


How do you handle uploads using Selenium?

- You can handle uploads using `sendKeys()` method.
- And the way to do it is, to locate the upload button, and send the `path of the file` using `sendKeys()`.
- Just `sendKeys()` without clicking



How to find the file path in Windows?



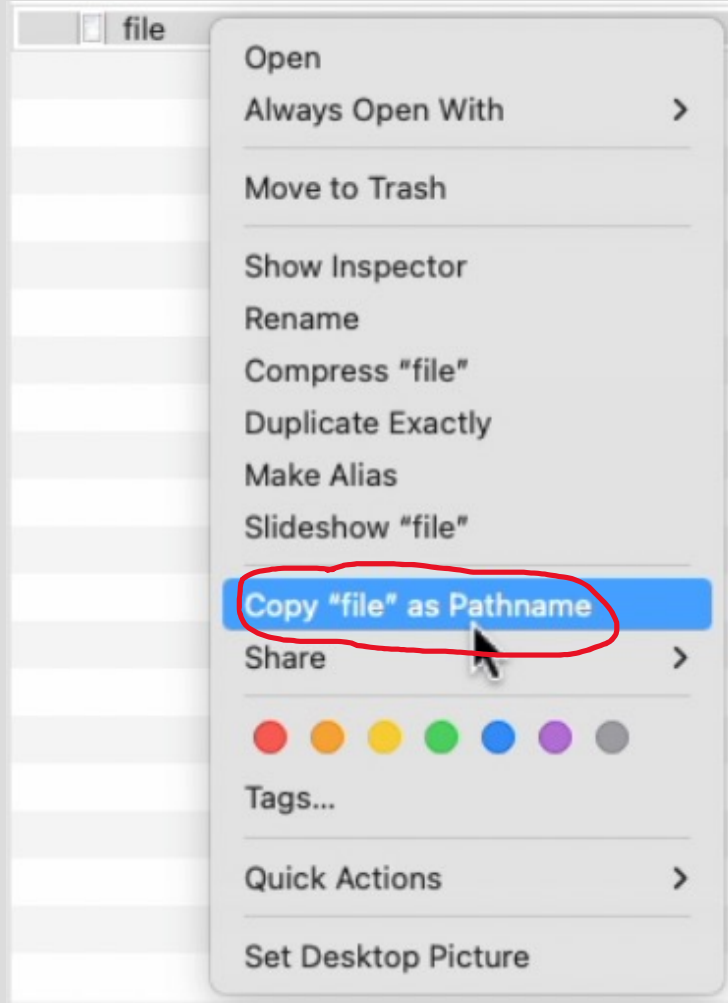
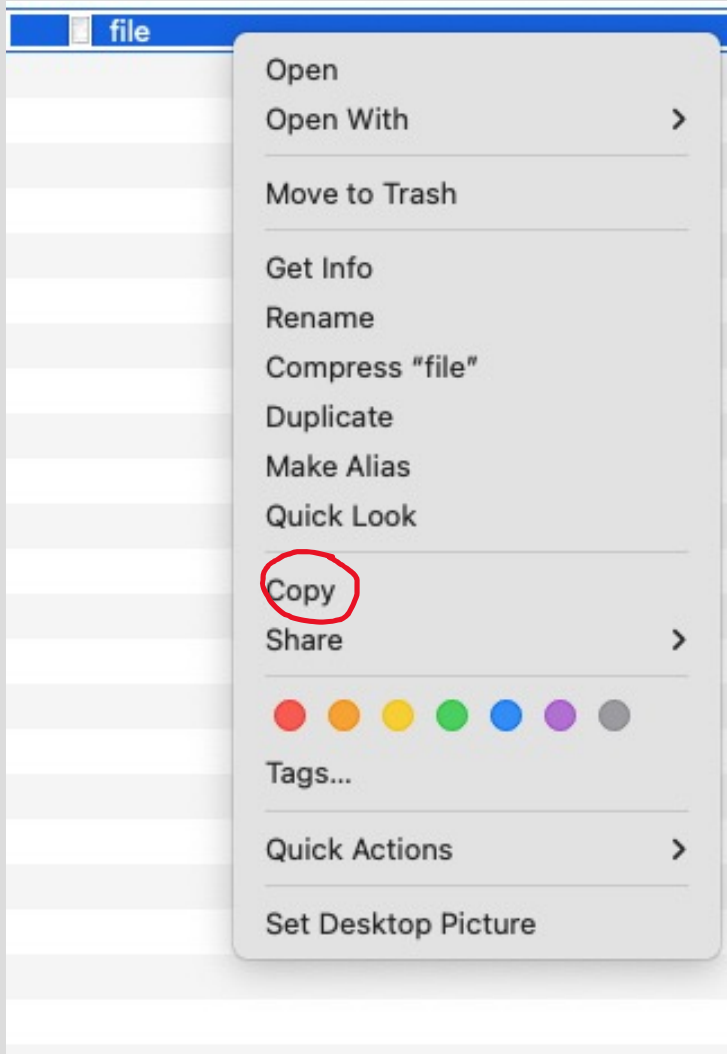
- Right click on the file you want to upload
- Select Properties
- Go to security
- Select all with mouse
- right click and copy path from "object name"
- Paste it in IntelliJ

C:\\Users\\yourName\\Downloads\\something.txt

C://Users/yourName/Downloads/something.txt



How to find the path in MacOS?



- Right click on the file you want to upload
- Press and **hold** **option** button from keyboard
- “Copy” will change to “Copy as path name”
- Select it, and paste in IntelliJ

Agenda: WebDriver API – Actions

- Actions overview
- Mouse actions
- File operations



After today's session, you should be able to:

- Do advanced mouse operations such as hover, right click etc.
- Do advanced Keyboard operations
- Chain several actions



Actions

- Actions class comes from Selenium library.
- The user-facing API for emulating complex user gestures.
- Class that provides ways to handle advanced mouse and keyboard events .



Mouse Events

Below are some of the mouse events we can do using Actions class:

- Double click
- Hover
- Drag and drop
- Context click (right click)
- Move to element



Keyboard Events

Below are some of the keyboard events using Actions:

- sendKeys
- keyUp
- keyDown
- Scroll up / down



Syntax

1. Create Actions object using WebDriver:

```
Actions actions = new Actions(driver)
```

2. Call methods provided in the Actions class:

```
actions.doubleClick()
```

3. Perform the action:

```
actions.doubleClick().perform()
```



Chaining Actions

1. Call multiple methods at the same time

```
actions.moveToElement(element).doubleClick()
```

2. Chain them into one using build method

```
actions.moveToElement(element).  
    doubleClick().build()
```

3. Perform the chained actions using perform

```
actions.moveToElement(element).  
    doubleClick().build().  
    perform();
```



Agenda: JavascriptExecutor

WHAT IS JAVASCRIPT EXECUTOR?

- It is a simple interface coming from Selenium library.
- It has only 2 methods in it.
- These 2 methods allow us to execute (inject) **JavaScript** code in our **Java** – Selenium code.



Agenda: JavascriptExecutor

WHY JAVASCRIPT EXECUTOR IS USEFUL IN CERTAIN SCENARIOS?

- JavaScript language runs natively in browsers.
- It helps to be able to pass JS code in our JAVA+Selenium code.



Agenda: JavascriptExecutor

WHAT CAN WE DO WITH JAVASCRIPT EXECUTOR?

We can:

- Interact with elements that are covered or invisible JavascriptExecutor
- Open empty tabs
- Scroll page in different way (up, down, left, right)
- Scroll to specific element
- Even sendKeys specifically using JavascriptExecutor.



JavascriptExecutor Methods

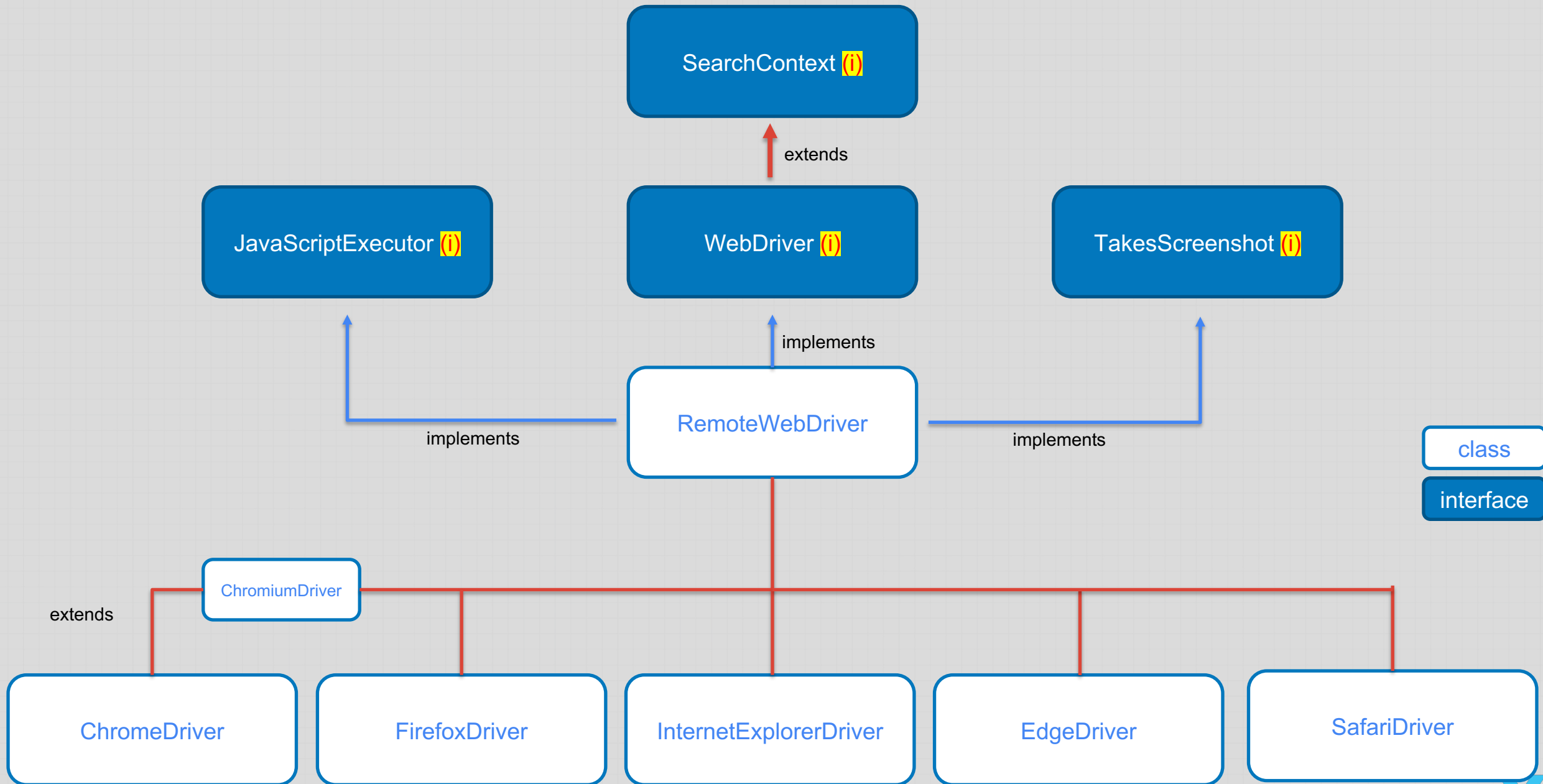
JavascriptExecutor only has 2 methods:

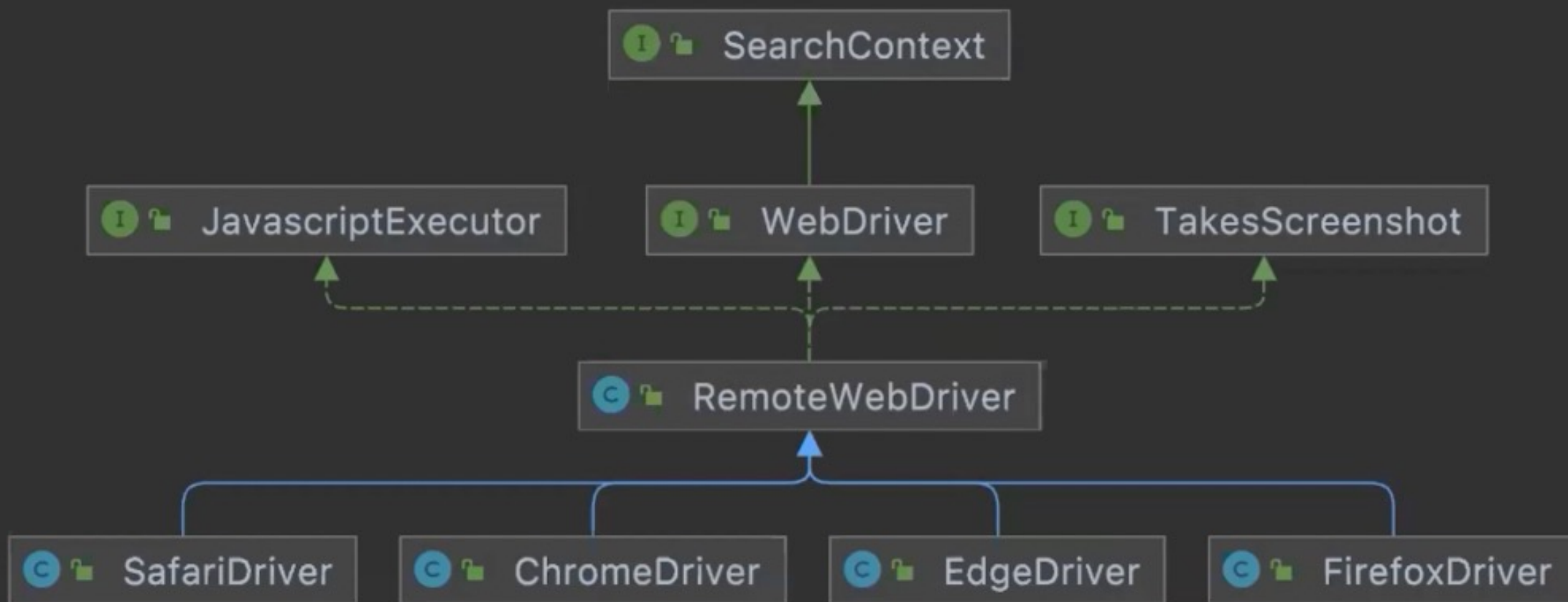
1. `executeAsynchScript()`
2. `executeScript()`

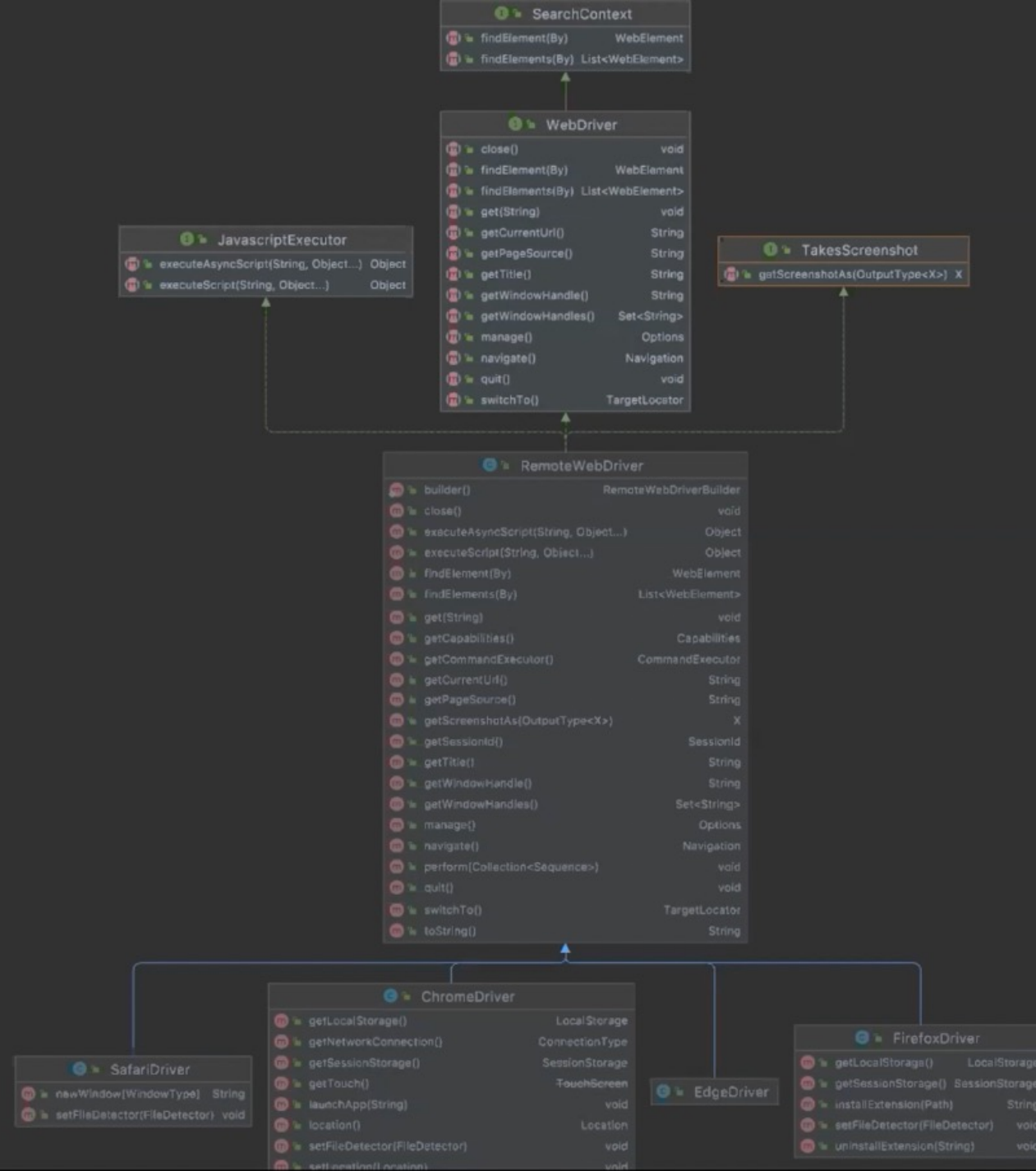
We pass the JavaScript function in the method as a String



Selenium classes and interfaces







Syntax

1. Create JavascriptExecutor object

```
JavascriptExecutor js = (JavascriptExecutor) Driver.getDriver();
```

2. Use the object and "executeScript" method to pass JS function.

```
js.executeScript(s: "window.scrollTo(0, 750)");
```

