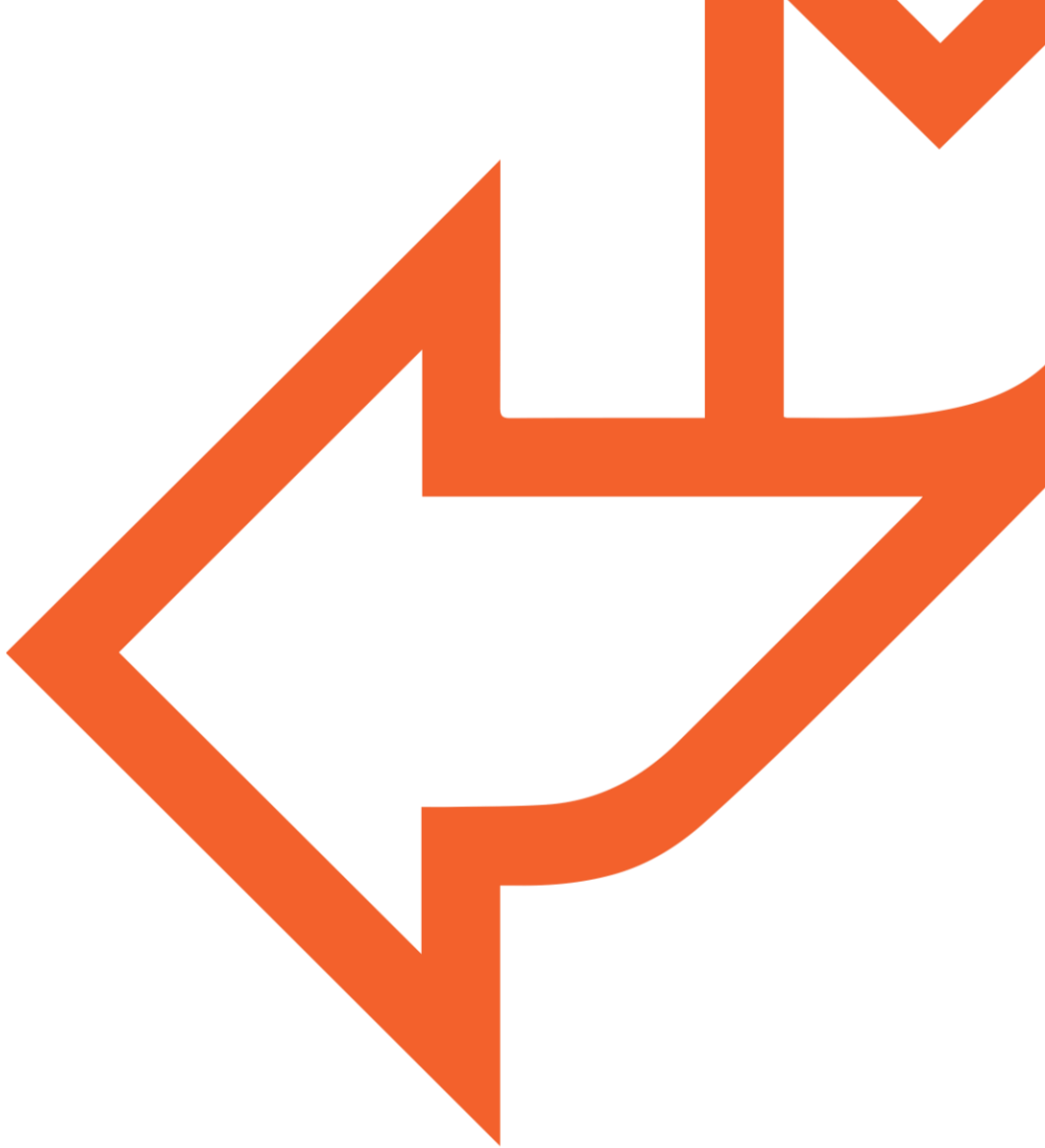
POWERING POTENTIAL

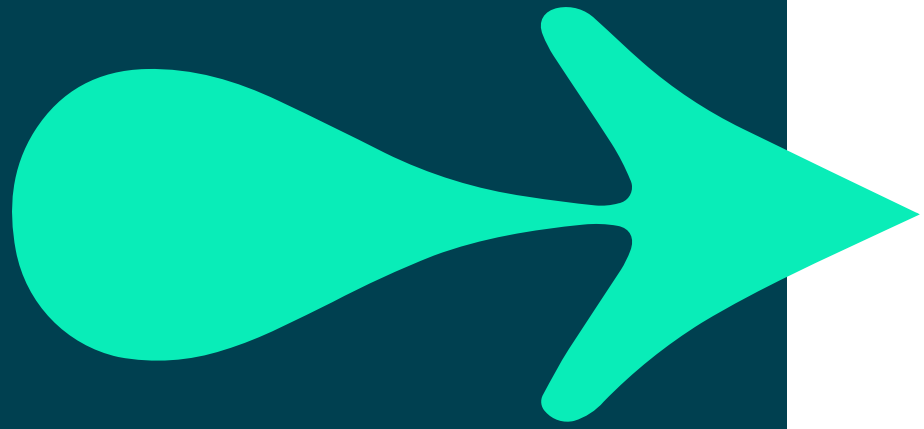


Selenium

Module 3 – Browser Manipulation

Navigation

NAVIGATION

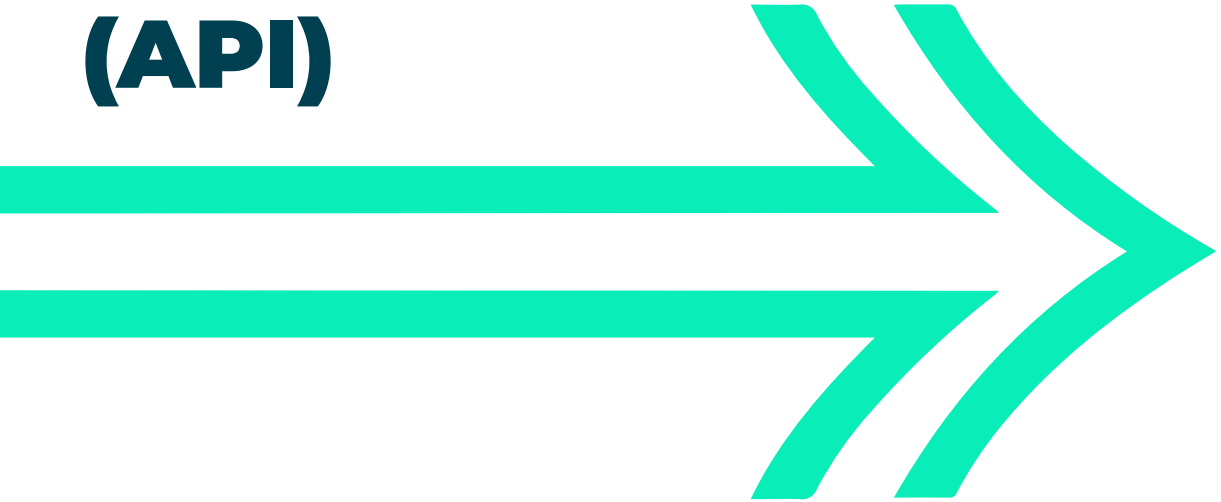


Navigating the web is an important part of testing. We have previously seen that the **WebDriver.get()** instance method accepts a URL as a string and opens that page in the browser.

The **WebDriver** has a more advanced **Navigation** object as an instance member, which can be used to navigate in the browser.

```
Navigation navigator = driver.navigate();
```

Navigation instance methods (API)

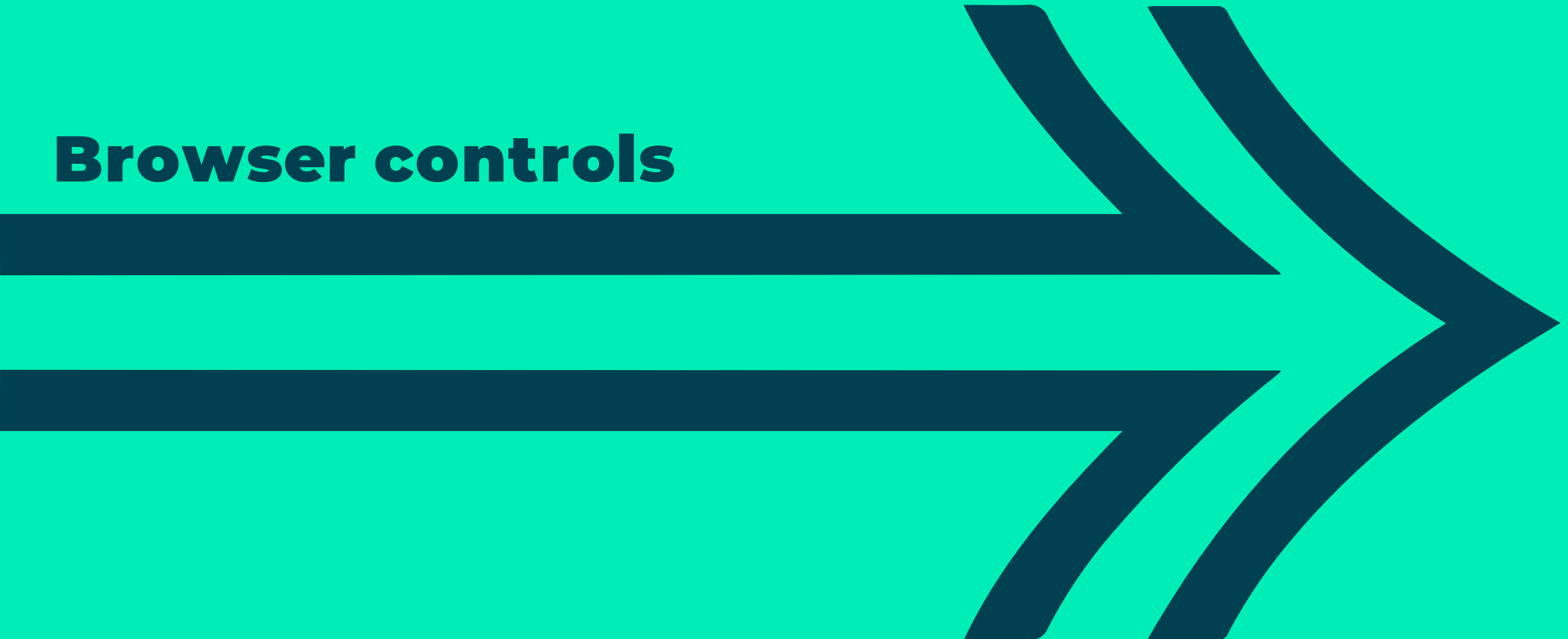


Method	Description
Navigation.to(String url)	Navigates to the supplied URL using a HTTP POST operation
Navigation.to(URL url)	Navigates using a URL object rather than a String
Navigation.back()	Moves back one page using the browser's history API
Navigation.forward()	Moves forward one page using the browser's history API, does nothing if on the latest page
Navigation.refresh()	Refreshes the current tab
Navigation.getTitle()	Gets the current tabs title

Exercise 2: Check the chapter titles

See Exercise 2 in your Selenium WebDriver exercise book.

Browser controls

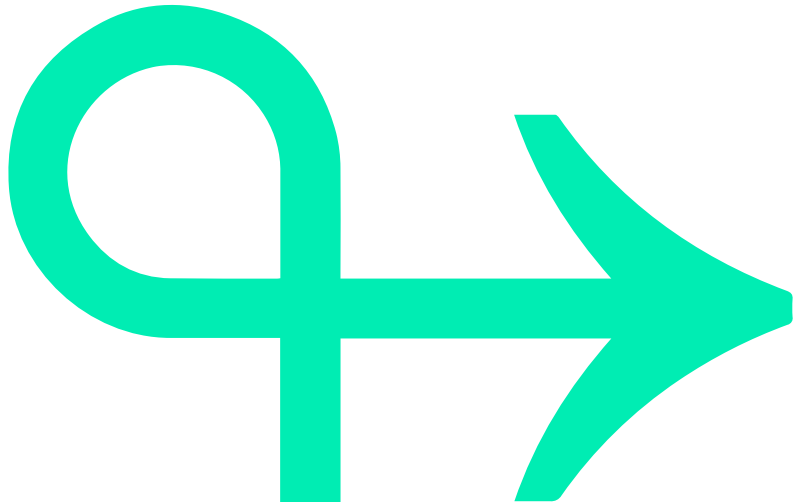


JavacriptExecutor

When using a web browser, the user can distinguish between many different parts of the system:

- The browser app itself
- The tabs open in the browser

Selenium quantifies a browser window or tab as just a window represented by a unique string identifier.



```
@Test
public void googleTest() throws InterruptedException {
    String originalTab = driver.getWindowHandle();
    driver.get("https://www.bing.com");

    ((JavascriptExecutor) driver).executeScript("window.open()");
    String newTab = driver.getWindowHandle();

    driver.switchTo().window(originalTab);

    driver.close();

    Thread.sleep(3000);
}
```

NewWindow API

The **NewWindow** API is a method available to call through a **TargetLocator** object, which is retrievable from a **WebDriver** instance:

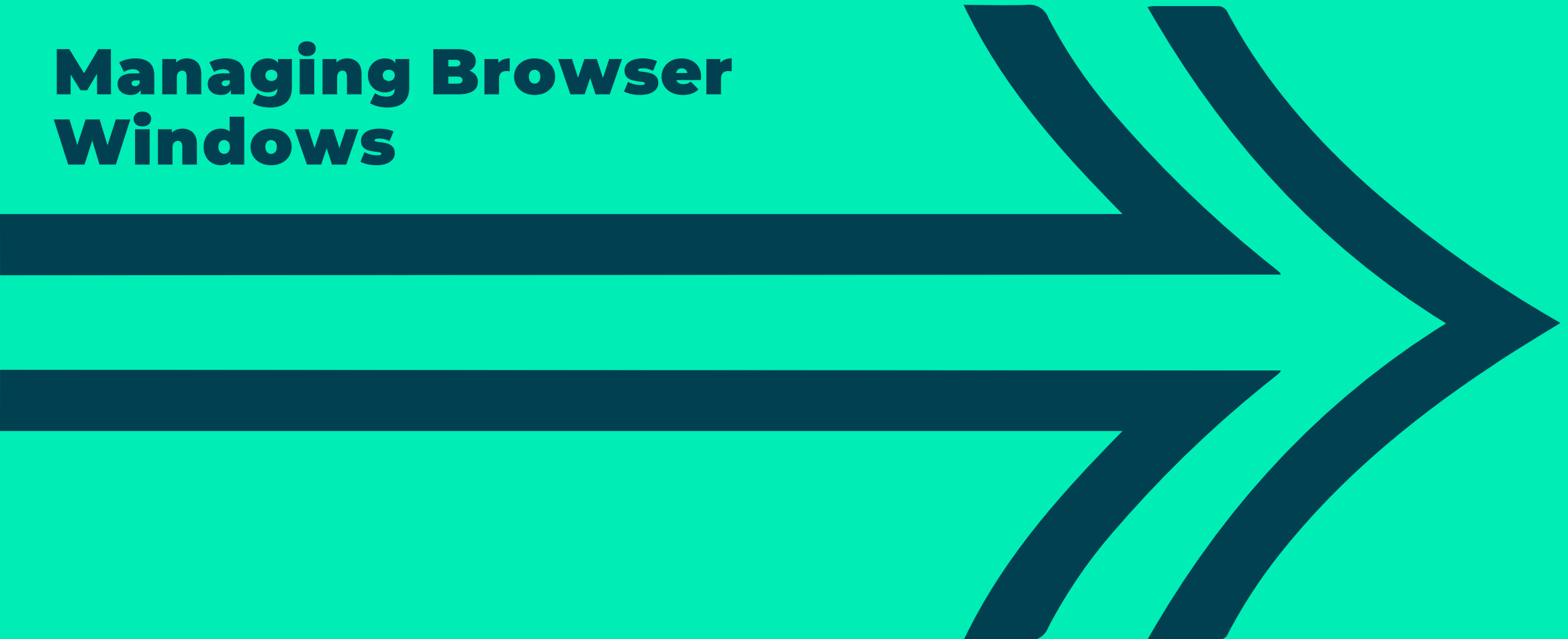
TargetLocator obj = driver.switchTo();

A **TargetLocator** is used to find frames or windows:

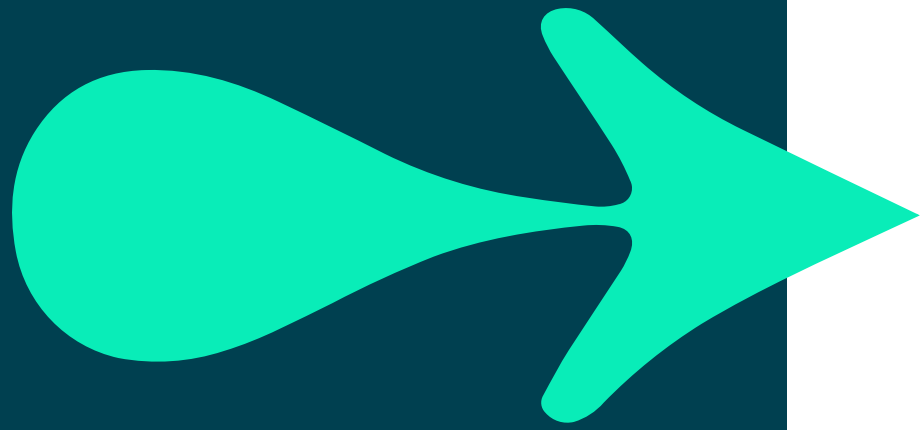
```
@Test
public void newWindowApiNavigationTest() throws InterruptedException {
    String originalTab = driver.getWindowHandle();
    driver.get("https://www.bing.com");

    driver.switchTo().newWindow(WindowType.TAB);
    driver.get("https://www.google.com");
    driver.close();

    driver.switchTo().window(originalTab);

    Thread.sleep(3000);
}
```

When calling **newWindow()**, either **WindowType.TAB** or **WindowType.WINDOW** can be specified as input to open either a new tab or window respectively.

Managing Browser Windows



THE WINDOW API



- The Window interface is used for controlling the current browser window.
- We can retrieve things like the size of our browser window or the coordinates of our window relative to the top left corner of the screen.
- The resolution of a screen can impact how content is displayed, ensure the Selenium WebDriver is correctly configured to deal with this by manipulating the browser window.

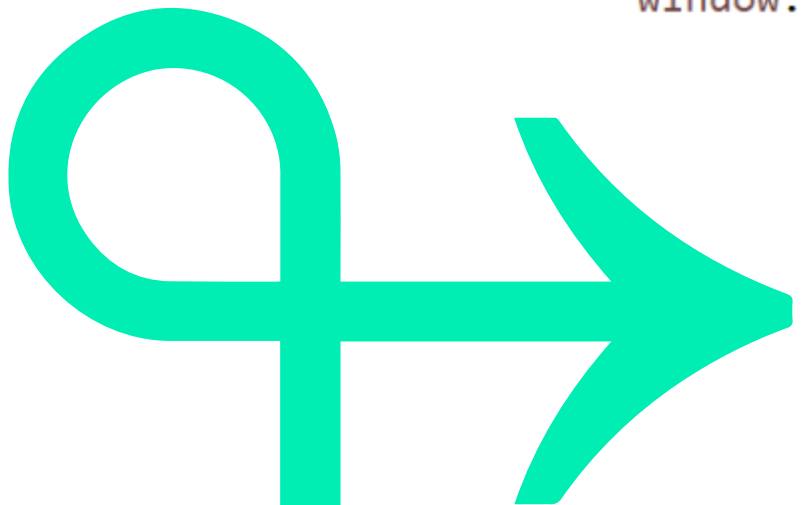
Window instance methods (API)

Method	Description
Window.getSize()	Gets the size of the current browser window as a **Dimension** object, this is not the size of the viewport but the browser itself.
Window.setSize(Dimension targetSize)	Sets the size of the current browser window, not the viewport.
Window.fullscreen()	Sets the browser to full screen mode, equivalent to F11 in most browsers.
Window.maximise()	Maximises the size of the current window.
Window.minimise()	Minimises the size of the current window.
Window.getPosition()	Returns a **Point** object representing the coordinates of the current window where the top left of the screen is the origin (0,0).
Window.setPosition(Point targetPosition)	Sets the position of the current window relative to the top left origin (0,0).

Managing a windows size

The **getSize()** method returns a **Dimension** object with the current size of the browser window, we can override this by providing a **Dimension** object to the **setSize()** method.

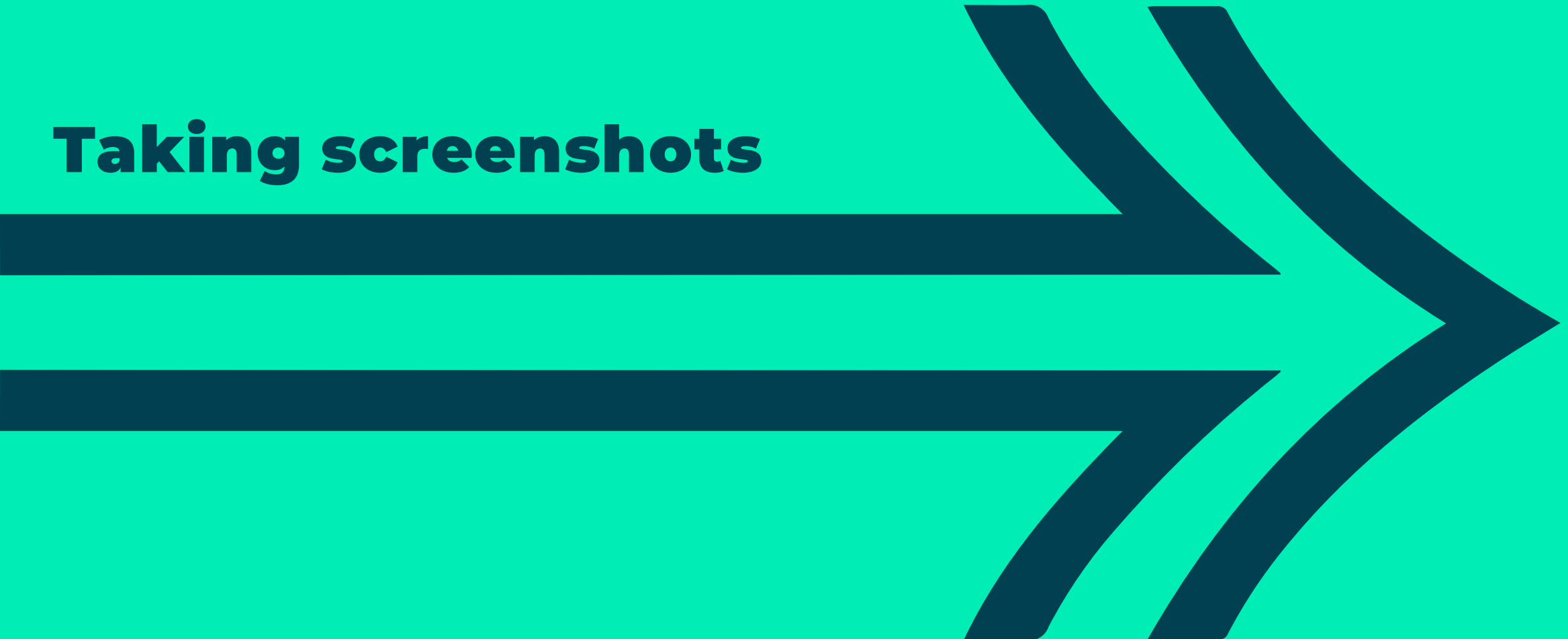
```
Window window = driver.manage().window();
Dimension oldWindowSize = window.getSize();
Dimension newWindowSize = new Dimension(1366, 768);
window.setSize(newWindowSize);
```



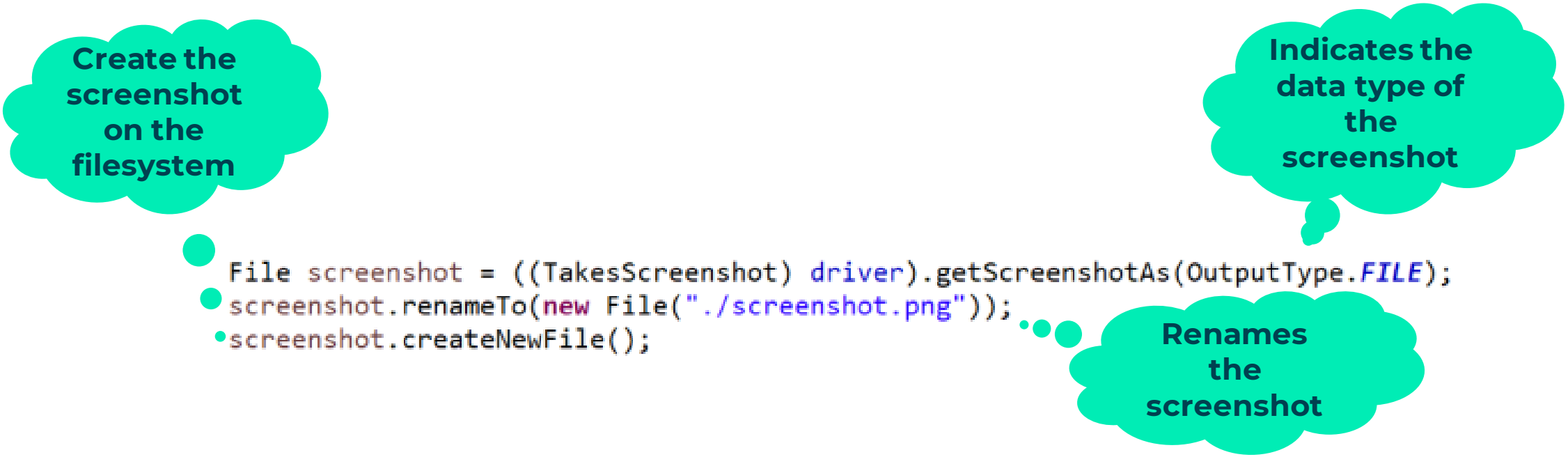
Exercise 3 & 4

See Exercises 3 and 4 in your Selenium WebDriver exercise book.

Taking screenshots



Taking a screenshot



Taking screenshots of web pages is often required during testing. This allows real humans to review the automated testing with greater certainty – it is also very useful for when exceptions occur, we can have a screenshot taken when a button fails to work for example.

The simplest way to take a screenshot is to typecast the **WebDriver** instance to a **TakesScreenshot** instance, then call its API.

Screenshot an element

Instances of **WebElement** have direct access to the **getScreenshotAs(OutputType type)** instance method. Call this method to take a screenshot of an element:

```
public void takeElementScreenshot(WebElement element, String path) {
    File screenshot = element.getScreenshotAs(OutputType.FILE);
    screenshot.renameTo(new File(path));
    screenshot.createNewFile();
}
```

This will take a screenshot of only the specified element instead of the whole web page.


Thank you!
Hope you enjoyed this learning journey.