

A lot of the times, when we are filling in a form and press the submit button, an alert pops up, telling me, "**Contact Number is Required**". At times, I forget to check the terms and conditions box, and I get the same as a notification via a popup-up alert. Pressing the **OK** button on the alert box takes me back to the form, and the form does not submit until I have filled everything "**required**". These use cases show how necessary are the "**Alerts**" or "**Popups**" in Web Applications. In this article, we will understand these features in more detail and will understand how we can automate various types of **PopUps and Alerts in Selenium** by covering the points in the following topics:

What are Alerts/popups in Selenium?

Different types of Alerts/popups

What are the various kinds of alerts provided by Web Applications?

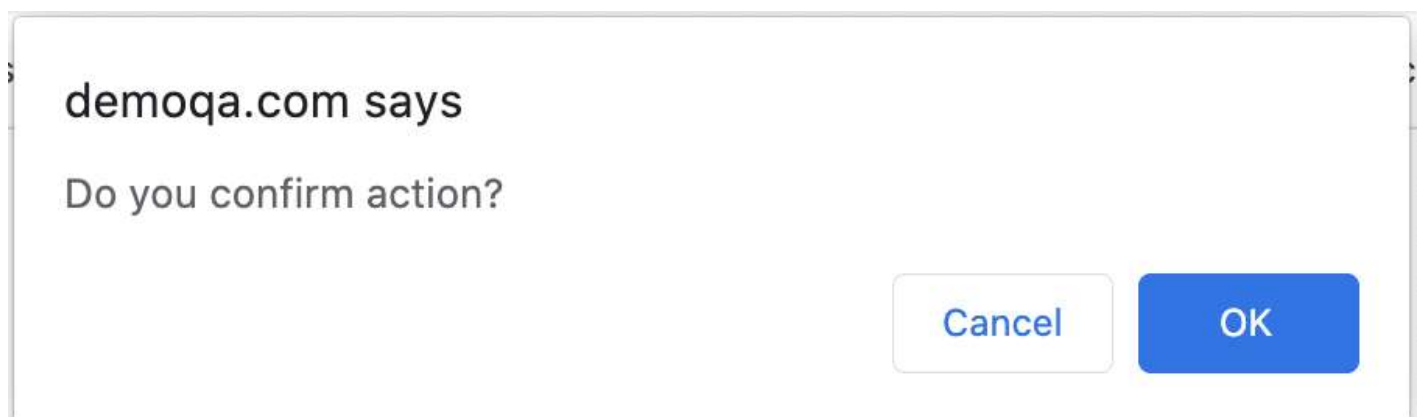
How to handle Alerts/popups using Selenium WebDriver?

How to handle unexpected Alerts using Selenium WebDriver?

What are Alerts/popups in Selenium?

Alerts are small popup boxes/windows which display the **messages/notifications** and notify the user with some information seeking some permission on certain kinds of operations. Additionally, we can also use them for warning purposes. Sometimes, the user can enter a few details in the alert box as well.

For Example, The alert box displayed below requires an action from the user to press **OK** and accept or press **Cancel** and dismiss the message box.

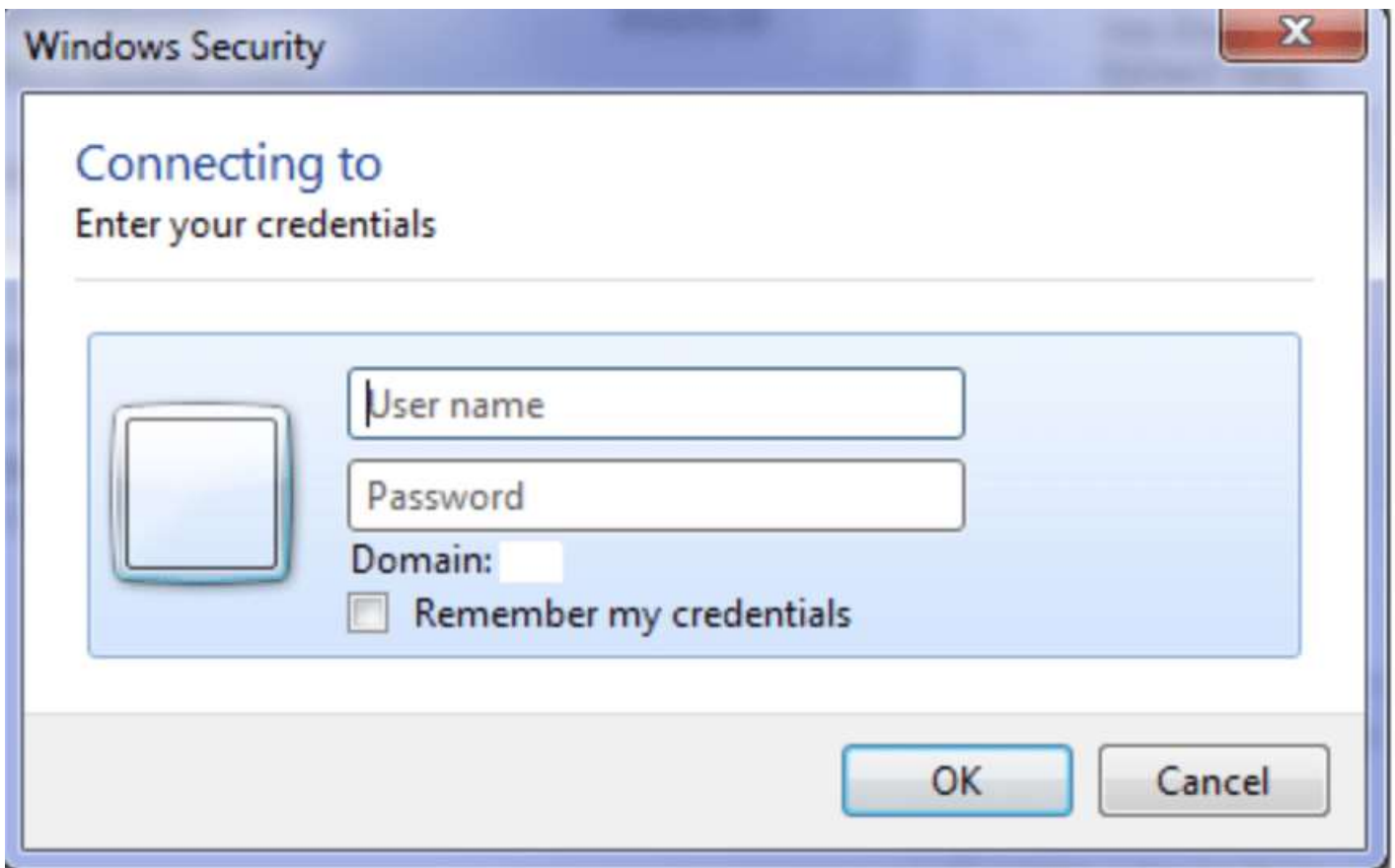


Application **alerts** shift your focus from the current browser window to a newly opened window and force the user to read the message displayed and act accordingly. Users will be blocked and will not be able to work further unless the alert message box gets handled.

What are the different types of Alerts/popups?

While automating any web application, **Selenium WebDriver** may encounter alerts that can either be **application dependant** or the **Operating system dependant** on which the user is working. Based on these categorizations, we can divide the alerts majorly into the following categories:

Windows/OS Alerts: Window-based alerts are system-generated alerts/popups. The developers invoke the operating system APIs to show these alerts/dialogue-boxes. Handling these alerts in Selenium is a little tricky and beyond the WebDriver's capabilities, as Selenium is an automation testing tool for web applications only, and we need third party utility to automate window based popups. A few of those utilities are **AutoIT** and **Robot Class** in Java. A sample operating system based alert will look as follows and are majorly called Dialog-Boxes:



Web/Javascript /Browser-based Alerts: Web/Browser based alerts are primarily called **Javascript alerts** and are those alerts that are browser dependant. These alerts are majorly called **Popups**.

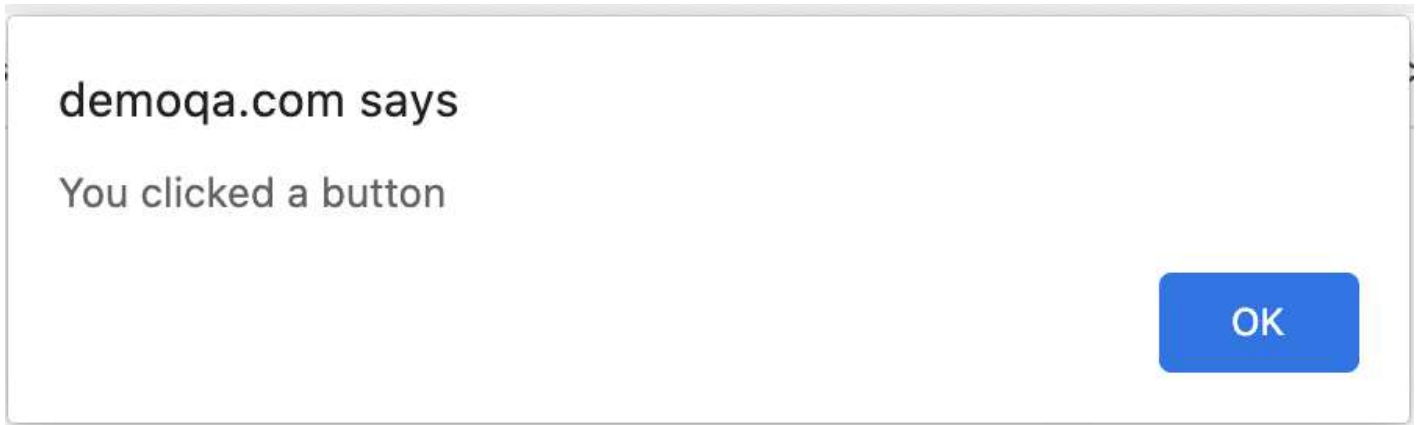
In this tutorial, we will be focusing majorly on *browser-based alerts*, as they are more prevalent in this web era and are compatible with automation using **Selenium WebDriver**.

Let's now see what the various web-alerts which we can see on multiple web applications are.

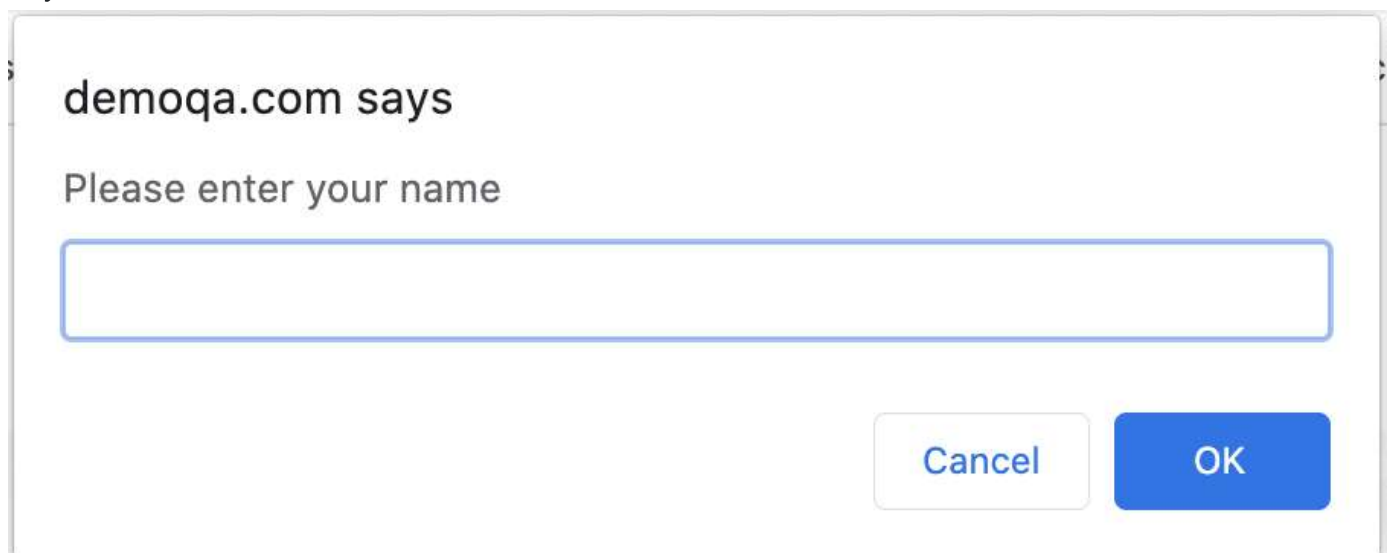
What are the various kinds of alerts provided by Web Applications?

As we discussed, there are various types of alerts that the developers can implement on web applications. Each of these alerts/popups needs different kinds of actions to handle these alerts. Let's see what these different types of alerts that the web applications provide are:

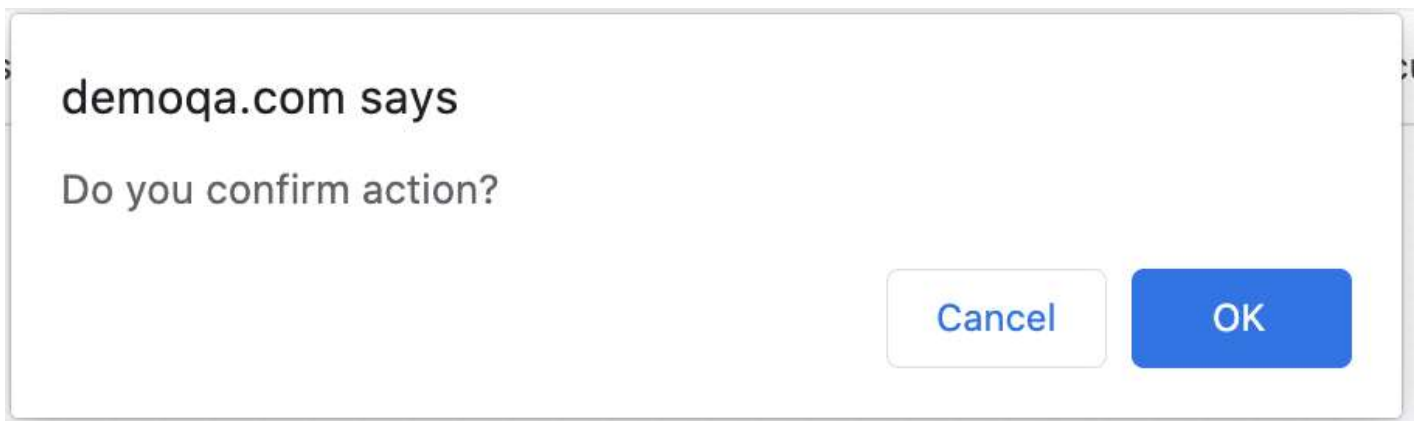
Simple alert: These alerts are just **informational** alerts and have an **OK** button on them. Users can click on the **OK** button after reading the message displayed on the alert box. A simple alert box looks like below:



Prompt Alert: In Prompt alerts, some **input requirement** is there from the user in the form of text needs to enter in the alert box. A prompt alert box is displayed like below, where the user can enter his/her username and press the **OK** button or **Cancel** the alert box without entering any details .



Confirmation Alert: These alerts **get some confirmation** from the user in the form of **accepting** or **dismissing** the message box. They are different from prompt alerts in a way that the user cannot enter anything as there is no text-box available. Users can only read the message and provide the inputs by pressing the **OK/Cancel** button.



Let's see how we can handle and automate these alerts using *Selenium WebDriver*.

How to handle Alerts/popups using Selenium WebDriver?

As we know, whenever we are executing any of the automation scripts using *Selenium WebDriver*, the *WebDriver* always has the focus on the main browser window and will run all the commands on the main browser window only. But, whenever an alert/popup appears, it opens up a new window. So, for *handling the Alerts using Selenium WebDriver*, the focus need to be shifted to the child windows opened by the Alerts. To switch the control from the parent window to the Alert window, the *Selenium WebDriver* provides the following command:

```
driver.switchTo( ).alert( );
```

Once we switch the control from the main browser window to the alert window, we can use the methods provided by ***Alert Interface*** to perform various required actions. For example, *accepting the alert, dismissing the alert, getting the text from the alert window, writing some text on the alert window*, and so on.

To handle *Javascript alerts*, *Selenium WebDriver* provides the package ***org.openqa.selenium.Alert*** and exposes the following methods:

Void accept(): This method clicks on the 'OK' button of the alert box.

```
driver.switchTo( ).alert( ).accept();
```

Void dismiss(): We use this method when the 'Cancel' button clicks in the alert box.

```
driver.switchTo( ).alert( ).dismiss();
```

String getText(): This method captures the message from the alert box.

```
driver.switchTo().alert().getText();
```

Void sendKeys(String stringToSend): This method sends data to the alert box.

```
driver.switchTo().alert().sendKeys("Text");
```

How to handle a Simple Alert using Selenium WebDriver?

To understand how we can handle the *Simple Alerts using Selenium WebDriver*, consider the following scenario:

In this scenario, we have used demoqa.com to illustrate simple alert handling.

1st Step: Launch the website "<https://demoqa.com/alerts>".

2nd Step: Click on the "**click me**" button, as highlighted by the arrow, to see the simple alert box.



Alerts

Click Button to see alert

On button click, alert will appear after 5 seconds

On button click, confirm box will appear

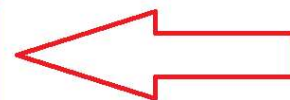
On button click, prompt box will appear

Click me

Click me

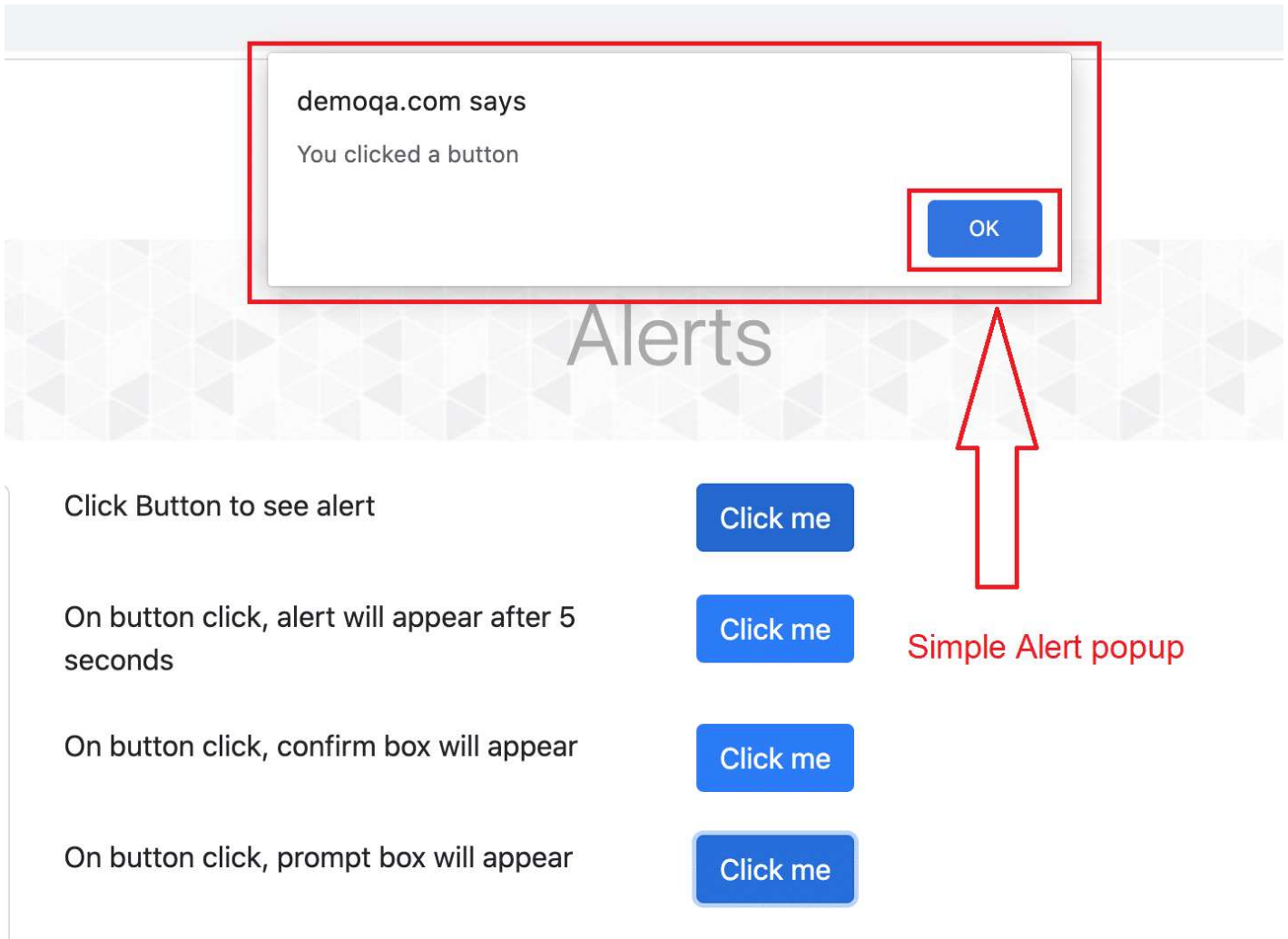
Click me

Click me



Click here to see the simple alert box

3rd Step: Simple alert box opens where the user has to accept it by hitting the **OK** button.



The image shows a web application interface with a section titled 'Alerts'. Below the title, there are four rows of text and buttons:

- Click Button to see alert
- On button click, alert will appear after 5 seconds
- On button click, confirm box will appear
- On button click, prompt box will appear

Each row has a blue button labeled 'Click me'. To the right of these buttons, there is a red arrow pointing upwards towards a simple alert box. The alert box is titled 'demoqa.com says' and contains the message 'You clicked a button'. It has a blue button labeled 'OK'.

Simple Alert popup

The below code snippet shows how we can handle the mentioned alert using *Selenium WebDriver*:

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

public class simplealert{
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "./src/resources/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.get("https://demoqa.com/alerts");
        driver.manage().window().maximize();
        // This step will result in an alert on screen
        driver.findElement(By.id("alertButton")).click();
        Alert simpleAlert = driver.switchTo().alert();
        simpleAlert.accept();
    }
}
```

Where,

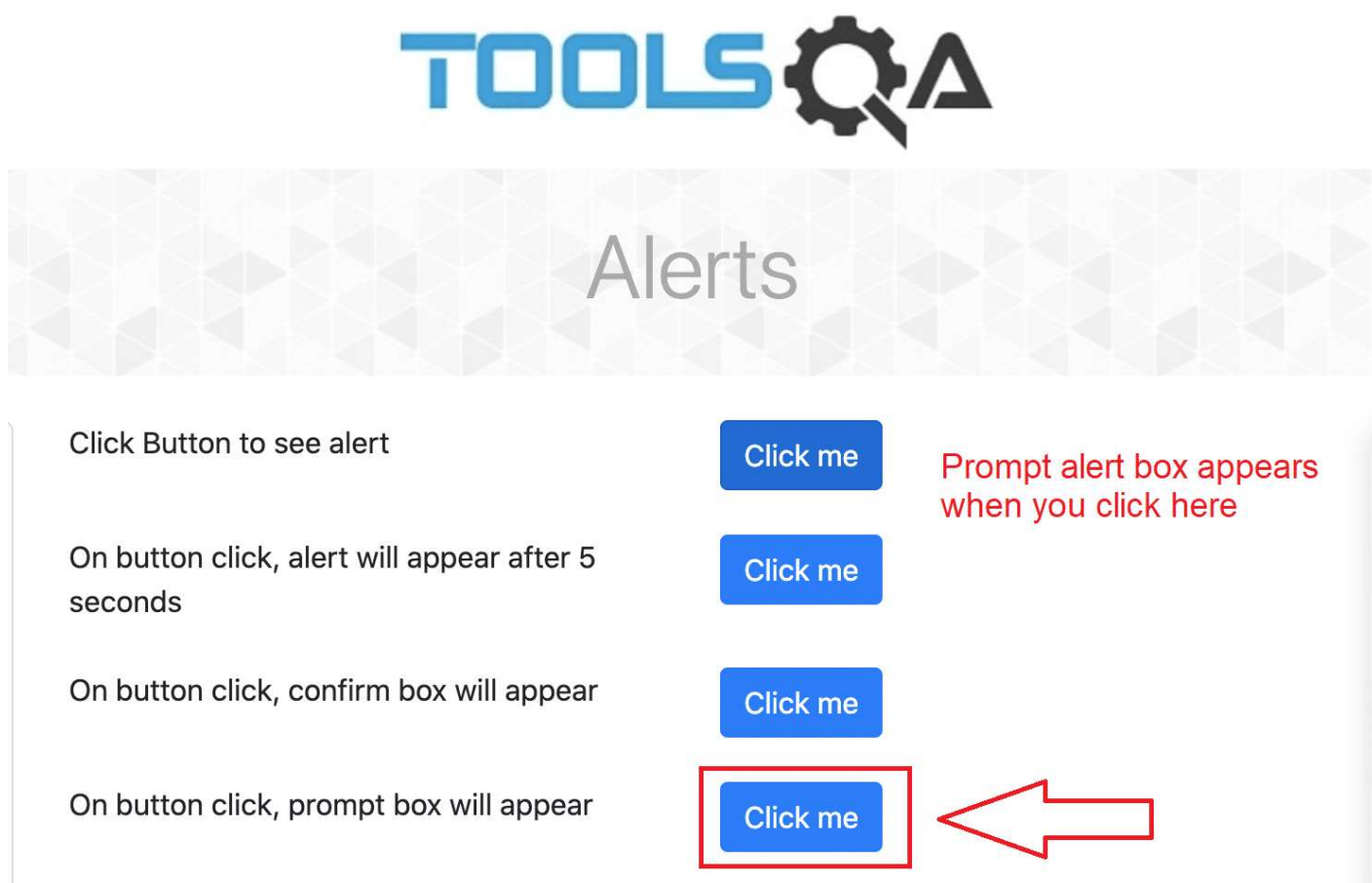
- . The creation of the WebDriver instance happens, and the launch of the browser opens the website.
- . Reference variable is created for Alert class which references to the alert by **Alert simpleAlert = driver.switchTo().alert();**.
- . To switch the control to the recently opened pop up window **Driver.switchTo().alert();** is used
- . Alert is accepted using **alert.accept();** method.

How to handle a Prompt Alert using Selenium WebDriver?

For understanding the handling the "**Prompt Alert**" using Selenium WebDriver, consider the following scenario:

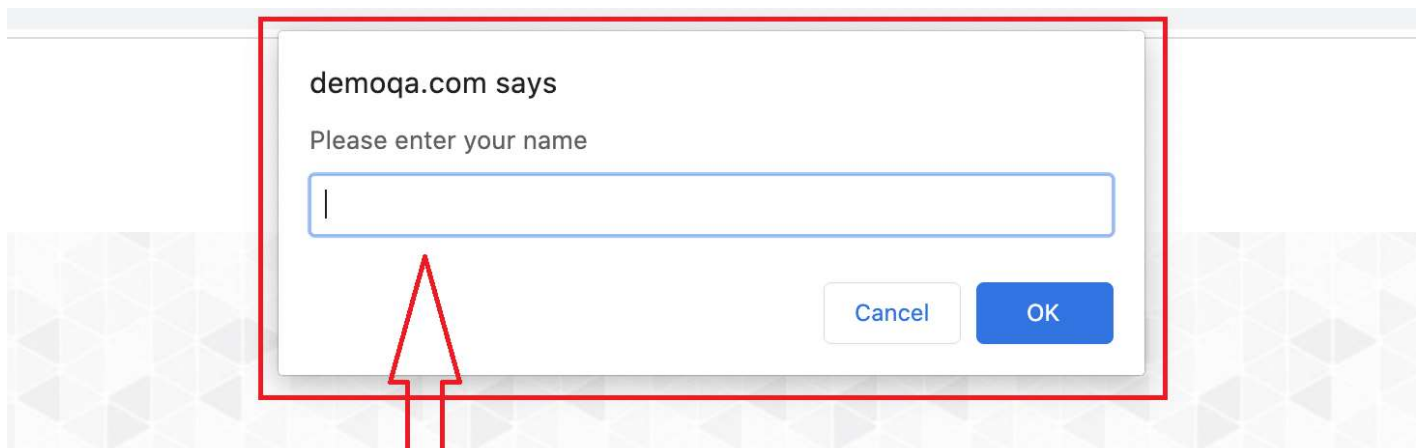
1st Step: Launch the website "<https://demoqa.com/alerts>".

2nd Step: Click on the "**click me**" button, as highlighted in the following screenshot, to see the prompt alert popup box.



3rd Step: Prompt alert box opens where the user can enter text in the text box. After entering user can accept or dismiss the alert box.

Note: In Selenium Webdriver, locators like XPath, CSS, etc. identify and carry out operations on a web page. In case locators do not work, we can use JavaScriptExecutor to achieve the desired operation on a web element.



Click Button to see alert

Click me

Prompt Alert box where user will enter the text

On button click, alert will appear after 5 seconds

Click me

On button click, confirm box will appear

Click me

On button click, prompt box will appear

Click me

4th Step: When the user has successfully entered the text in the text box and accepted the alert, the parent window displays the operation performed "***You entered Test User***".

Alerts

Click Button to see alert

Click me

On button click, alert will appear after 5 seconds

Click me

On button click, confirm box will appear

Click me

On button click, prompt box will appear **You entered Test User**

Click me

Text entered by user in Prompt Alert box

Below code-snippet shows a sample code, which explains how to handle **prompt alerts** using *Selenium WebDriver*:

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

public class promptalert {
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", "./src/resources/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.get("https://demoqa.com/alerts");
        driver.manage().window().maximize();
        // This step will result in an alert on screen
        WebElement element = driver.findElement(By.id("promptButton"));
        ((JavascriptExecutor) driver).executeScript("arguments[0].click()", element);
        Alert promptAlert = driver.switchTo().alert();
        String alertText = promptAlert.getText();
        System.out.println("Alert text is " + alertText);
        //Send some text to the alert
        promptAlert.sendKeys("Test User");
        promptAlert.accept();
    }
}
```

Where,

- . Once the website launches, and we click the "**prompt button**", the prompt alert box opens.
- . To located the `WebElement` we have used javascript executor here `((JavascriptExecutor) driver).executeScript("arguments[0].click()", element);`. This method executes JavaScript in the context of the currently selected frame or window.
- . *It will read the text from the Alert box using `String alertText = promptAlert.getText();` the value stores in the String format.
- . `sendKeys();` the method enters the text in the Prompt alert box, and then accepts the alert using `alert.accept()` method. The below screen shows the text fetched from a prompt alert box.

How to handle a Confirmation Alert using Selenium WebDriver?

For understanding the handling the "**Confirmation Alert**" using Selenium WebDriver, consider the following scenario:

1st Step: Launch the website "<https://demoqa.com/alerts>".

2nd Step: Click on the "**click me**" button, as highlighted in the following screenshot, to see the confirmation alert box.



Alerts

Click Button to see alert

Click me

On button click, alert will appear after 5 seconds

Click me

On button click, confirm box will appear

Click me

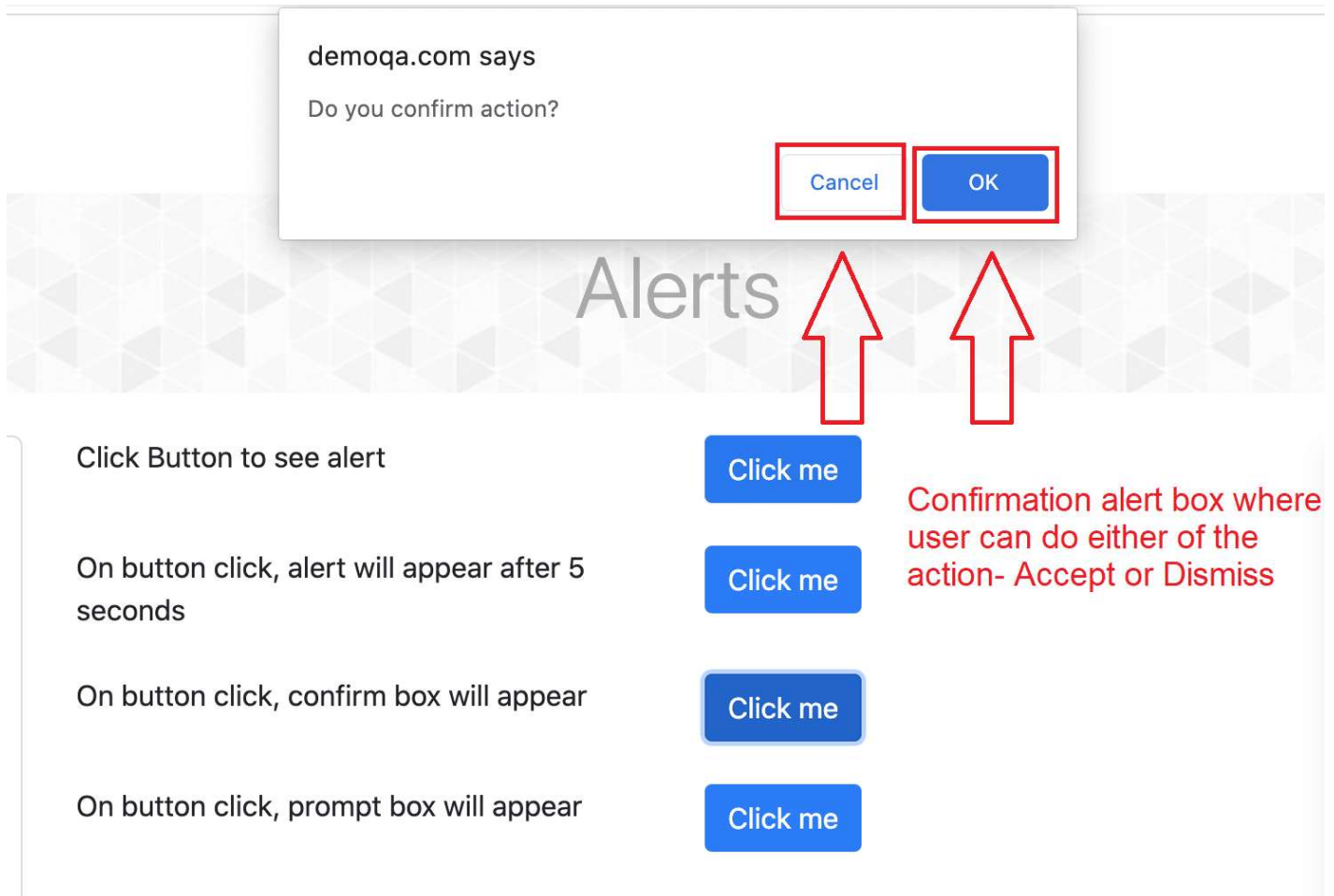
On button click, prompt box will appear

Click me

When user clicks here,
confirmation alert box opens



3rd Step: Once the confirmation box is open, the user can either accept or dismiss the alert box using **`Alert.accept()`**; or **`Alert.dismiss()`**; method.



The screenshot shows a confirmation alert box with the text "demoqa.com says" and "Do you confirm action?". It has two buttons: "Cancel" and "OK". Below the alert box, there is a section titled "Alerts" with a list of four items, each with a "Click me" button. Red arrows point from the "Click me" buttons to the "Cancel" and "OK" buttons in the alert box.

demoqa.com says
Do you confirm action?

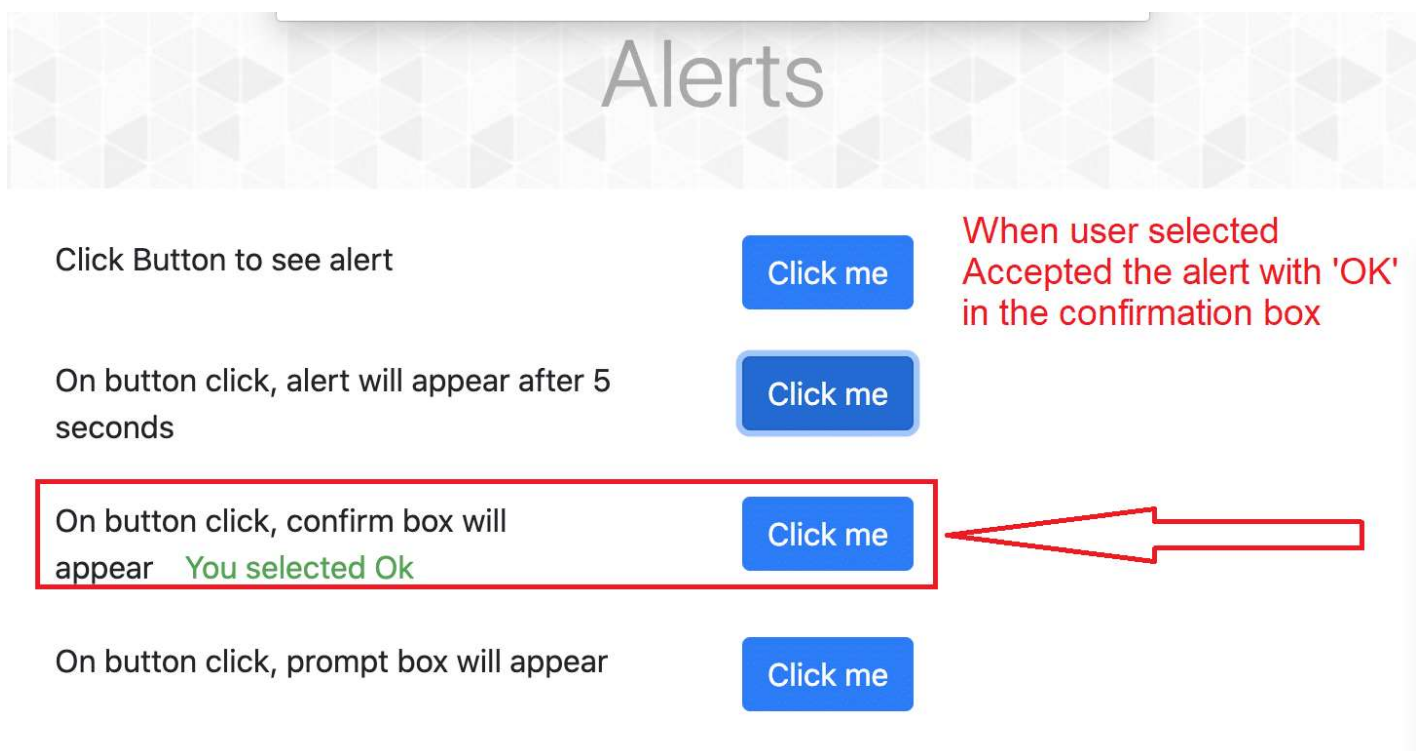
Cancel OK

Alerts

- Click Button to see alert [Click me](#)
- On button click, alert will appear after 5 seconds [Click me](#)
- On button click, confirm box will appear [Click me](#)
- On button click, prompt box will appear [Click me](#)

Confirmation alert box where user can do either of the action- Accept or Dismiss

4th Step: When the user has accepted the alert, then the parent window displays the operation performed on the confirmation alert box.



The screenshot shows the "Alerts" section with four items. The third item, "On button click, confirm box will appear", is highlighted with a red box. The text "You selected Ok" is displayed in green. A red arrow points from the "Click me" button of this item to the right. To the right of the buttons, there is a red text box stating "When user selected Accepted the alert with 'OK' in the confirmation box".

Alerts

- Click Button to see alert [Click me](#)
- On button click, alert will appear after 5 seconds [Click me](#)
- On button click, confirm box will appear [Click me](#) **You selected Ok**
- On button click, prompt box will appear [Click me](#)

When user selected Accepted the alert with 'OK' in the confirmation box

Below is the code snippet, which shows how to handle the **confirmation alerts** using *Selenium WebDriver*:

```
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class alerts {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "./src/resources/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.get("https://demoqa.com/alerts");
        driver.manage().window().maximize();
        // This step will result in an alert on screen
        WebElement element = driver.findElement(By.id("confirmButton"));
        ((JavascriptExecutor) driver).executeScript("arguments[0].click()", element);
        Alert confirmationAlert = driver.switchTo().alert();
        String alertText = confirmationAlert.getText();
        System.out.println("Alert text is " + alertText);
        confirmationAlert.accept();
    }
}
```

As we can see here, the maximum of the code is similar; the only difference is input from the user is not possible like a prompt alert box. Hence, we do not use it. We can either **accept or dismiss** the alert message box. Here we have accepted the alert using **Alert.accept();** method.

How to handle unexpected Alerts using Selenium WebDriver?

Sometimes when we are browsing any web application, **unexpected alerts** occur due to some error or some other reasons. This kind of alert does not display every time you open the site, but they surface only at random intervals. If you have created an automation test case for such pages and not handled this kind of alerts, then your script will fail immediately if such **unexpected alert pop up** is displayed.

We have to handle these unexpected alerts specifically, and for that, we can use the **try-catch block**.

Note: *If we write direct code(without try-catch) to accept or dismiss the alert, and If the alert does not appear, then our test case will fail to throw any exception in Selenium like timeout **Exception**. Try catch block can handle both situations.*

The following code demonstrates how we handle unexpected alerts in Selenium using a **try-catch** block.

```
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedCondition;
import org.openqa.selenium.support.ui.WebDriverWait;

public class alerts {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "./src/resources/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demoqa.com/alerts");

        try {
            driver.findElement(By.id("timerAlertButton")).click();
            WebDriverWait wait = new WebDriverWait(driver, 10);
            wait.until(ExpectedConditions.alertIsPresent());
            Alert simpleAlert = driver.switchTo().alert();
            simpleAlert.accept();
            System.out.println("Unexpected alert accepted");
        } catch (Exception e) {
            System.out.println("unexpected alert not present");
        }
        driver.quit();
    }
}
```

Let's understand this step by step:

- . Launch the browser and open the site "<https://demoqa.com/alerts>".
- . Locate the WebElement "**timerAlertButton**" and click on it
- . We have used **Explicit wait** here as the driver will wait for 10 seconds to see if an alert occurs and. It will check for the alert using the **try-catch block**. We use Explicit wait as **WebDriverWait wait = new WebDriverWait(driver, 10)** ; and **wait.until(ExpectedConditions.alertIsPresent());**
- . If the alert is present, then we will accept it using **driver.switchTo().alert().accept()**; method else it will go to catch block and print the message "**unexpected alert not present**".

Key Takeaways

Alerts are small popup windows that display the message/notifications and notify the user with some information or may ask for permission on certain kinds of operation.

There are two types of alerts: Web/Javascript/browser-based alerts and Windows/OS-based alerts in Selenium. Web-based alerts can further bifurcate into Simple alerts, Prompt alerts, and confirmation alerts.

These alerts are significantly visible in online application forms, banking websites, and social networking/Email service provider websites like Gmail. Every QA must have encountered such

alerts while automating the application under test.