

CheckBoxes are widely used on web pages to offer *multiple choices* of an option to the user. But when a **quality engineer** works with a checkbox, what choices does he/she have? In this tutorial, we will look deep into details on how we can locate and automate actions and validations on a **Checkbox in Selenium WebDriver**. A **CheckBox** on an **HTML** page provides various unique properties that can identify and automate their behavior in **Selenium WebDriver**. In this tutorial, we will understand the intricacies of **CheckBoxes** and how the same can be automated using **Selenium WebDriver** by covering the details under the following topics:

What is a CheckBox?

How to handle a CheckBox using Selenium WebDriver?

How to locate and select a checkbox in Selenium using the ID locator?

And, how to locate and select a checkbox in Selenium using the XPath locator?

Also, how to locate and select a checkbox in Selenium using the CSS Selector locator?

How to perform validations on a CheckBox using Selenium WebDriver?

How to use the isSelected() method for validation of a checkbox?

And, how to use the isDisplayed() method for validation of a checkbox?

How to use the isEnabled() method for validation of a checkbox?

What is a CheckBox?

The checkbox is a GUI element that allows the user to make certain choices for the given options. Users may get a list of choices, and the checkbox records the choices made by the user. The checkbox allows users to select either single or multiple choices out of the given list.

We can define a checkbox in **HTML** using **<input type="checkbox">** tag. Any **locator strategy** that uses **DOM** for locating web elements should use this tag and properties for recognizing the checkbox.

Apart from **"Checked "** and **"Unchecked"**, sometimes applications provide a **tri-state/intermediate**, which we generally use when a neutral answer needs to provide for an option or which is an *auto-selection for a parent* when the child checkbox is selected. Based on these, *checkboxes* will generally have the following states:



Checked




Tristate / Indeterminate



Unchecked

To understand more about *CheckBoxes*, let's consider the example of *checkboxes* (as highlighted below) given on the page "<http://www.demoqa.com/automation-practice-form>".



Student Registration Form

Name	<input type="text" value="First Name"/>	<input type="text" value="Last Name"/>
Email	<input type="text" value="name@example.com"/>	
Gender	<input type="radio"/> Male <input type="radio"/> Female <input type="radio"/> Other	
Mobile(10 Digits)	<input type="text" value="Mobile Number"/>	
Date of Birth	<input type="text" value="15 Jun 2020"/>	
Subjects	<input type="text"/>	
Hobbies	<input type="checkbox"/> Sports <input type="checkbox"/> Reading <input type="checkbox"/> Music	
Picture	<div>Select picture</div> <div><input type="button" value="Choose File"/> No file chosen</div>	
Current Address	<input type="text" value="Current Address"/>	

If we look at the **HTML** structure of the above element, we will find that it starts with the `<input type="checkbox">` node. We have highlighted the **HTML structure** of the **checkboxes** in the image below:

```
<div class="col-md-9 col-sm-12">
  <div class="custom-control custom-checkbox custom-control-inline">
    <input type="checkbox" id="hobbies-checkbox-1" class="custom-control-input" value="1">
  </div>
  <div class="custom-control custom-checkbox custom-control-inline">
    <input type="checkbox" id="hobbies-checkbox-2" class="custom-control-input" value="2">
  </div>
  <div class="custom-control custom-checkbox custom-control-inline">
    <input type="checkbox" id="hobbies-checkbox-3" class="custom-control-input" value="3">
    <label title="for=hobbies-checkbox-3" class="custom-control-label"> </label>
  </div>
</div>
```

As we can see, all the *checkboxes* are being created using the **HTML tag** `<input>` and have an attribute named **“type”**, which has a value **“checkbox”**, which signifies that the type of the input element is a checkbox.

Now, let's see how we can locate and perform specific actions on the **CheckBoxes** using **Selenium WebDriver**?

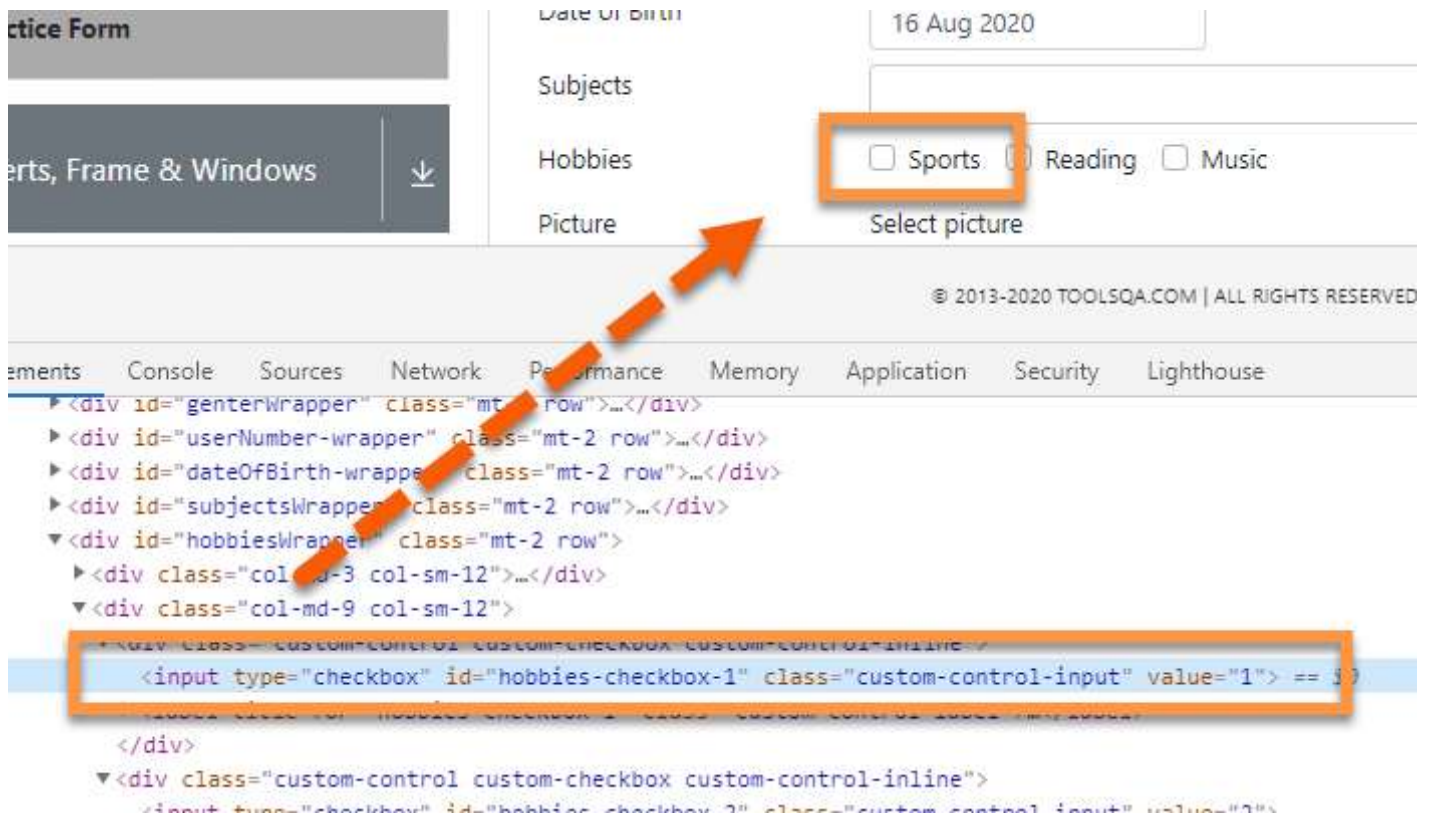
How to handle a CheckBox in Selenium WebDriver?

As we know, Selenium provides various **locator strategies**, and almost all of them can locate **CheckBoxes** using **Selenium WebDriver**. Let's understand a few of them:

How to locate and select a checkbox in Selenium using the ID locator?

If a checkbox has an **id** attribute that contains a unique value, then we can use the **ID locator** provided by the **Selenium WebDriver** for locating and selecting the element. To select a **checkbox**, the **click operation** needs to perform. So, once we locate the element, we need to perform a click to select it.

Let's have a look at the below **ToolsQA demo page** for understanding the details of the **checkbox** having **id** attribute:



In the above **DOM**, we can see that the **input** tag has an **id** attribute. Now, if we use the **ID locator** to recognize the element and perform the **click** operation, we will need to use the following **Selenium** code:

```
/**
 * Locating and Clicking a CheckBox By using ID
 */

driver.findElement(By.id("hobbies-checkbox-1")).click();
```

Using the above line of code, Selenium will locate the web element with "**id**" as "**hobbies-checkbox-1**" and will perform the **click** operation on that. The execution of the above line of code will lead to the following state on the web page:

Mobile(10 Digits)

Date of Birth

Subjects

Hobbies ☒ Sports ☐ Reading ☐ Music

Picture

No file chosen

Current Address

State and City

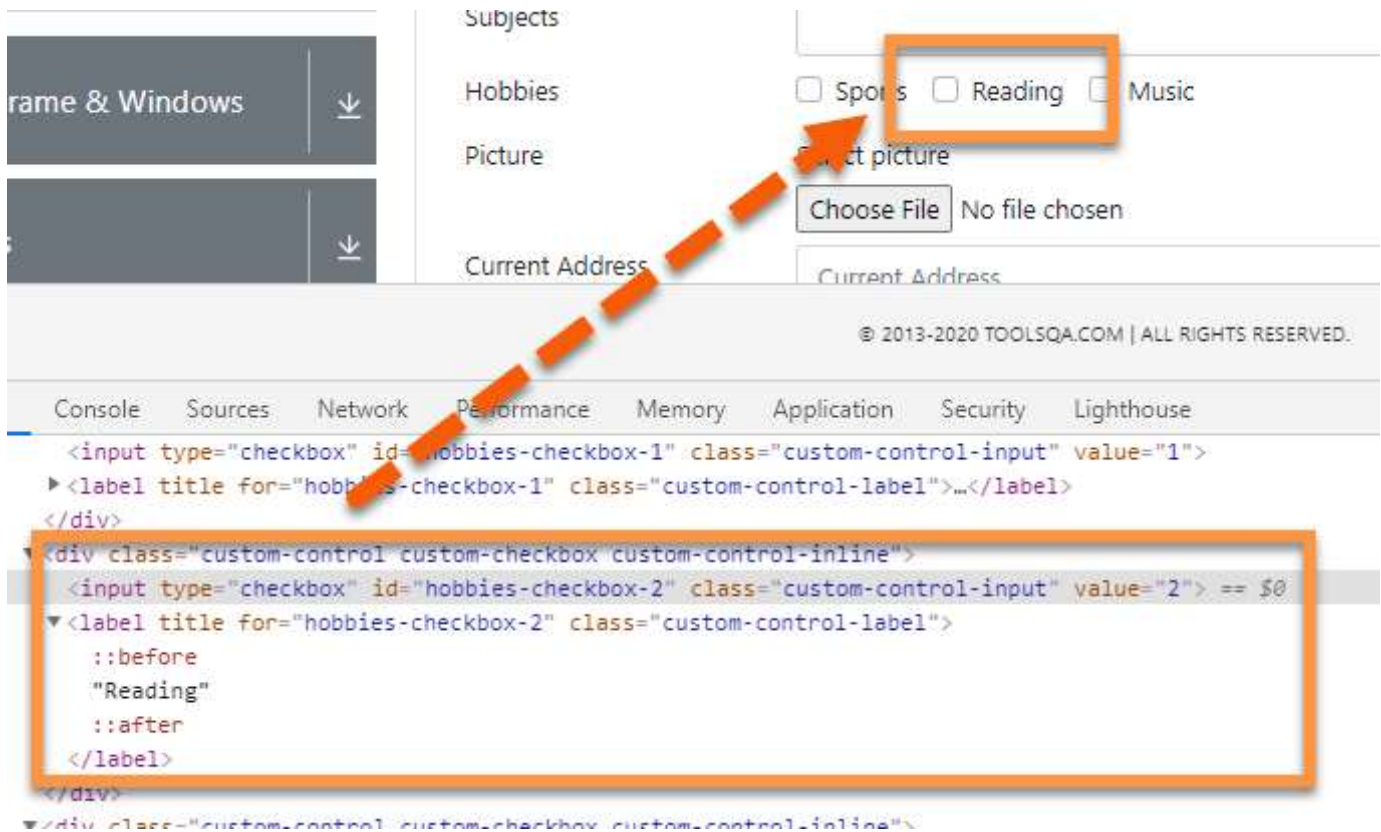
This way, we can select a **CheckBox** that has the unique **"id "** attribute and **check** the same by using the **"click"** operation.

How to locate and select a checkbox in Selenium using the XPath locator?

Selenium WebDriver can locate *CheckBoxes* using the XPath locator strategy, as it does the other web elements.

In the image given below, we can see the **HTML structure** of the highlighted checkbox in the **DOM**. We can use this to write the **XPath** and identify/locate the element.

Let's consider an example where we will try to *select multiple checkboxes using XPath*. We will choose both **"Sports "** and **"Reading "** checkbox to exhibit multiple selection scenarios.



Once we can uniquely identify the checkbox using **XPath**, we can use **Selenium WebElement** to perform a required operation such as **"click "** on the checkboxes. The code for *locating the CheckBoxes using Xpath* and *selecting the same* will look as follows:

```
//Selecting the first checkbox using XPath
```

```
driver.findElement(By.xpath("//label[text()='Sports']")).click();
```

```
//Selecting the second checkbox using Xpath
```

```
driver.findElement(By.xpath("//label[text()='Reading']")).click();
```

Note: In the above code, we have clicked on the label associated with the checkboxes. Generally, checkboxes can be checked/unchecked by clicking both the checkbox itself or the labels associated with the checkboxes.

Using the above line of code, Selenium will locate the web element with specified **XPath** and will perform the **click** operation on that. The execution of the above line of code will lead to the following state on the web page:

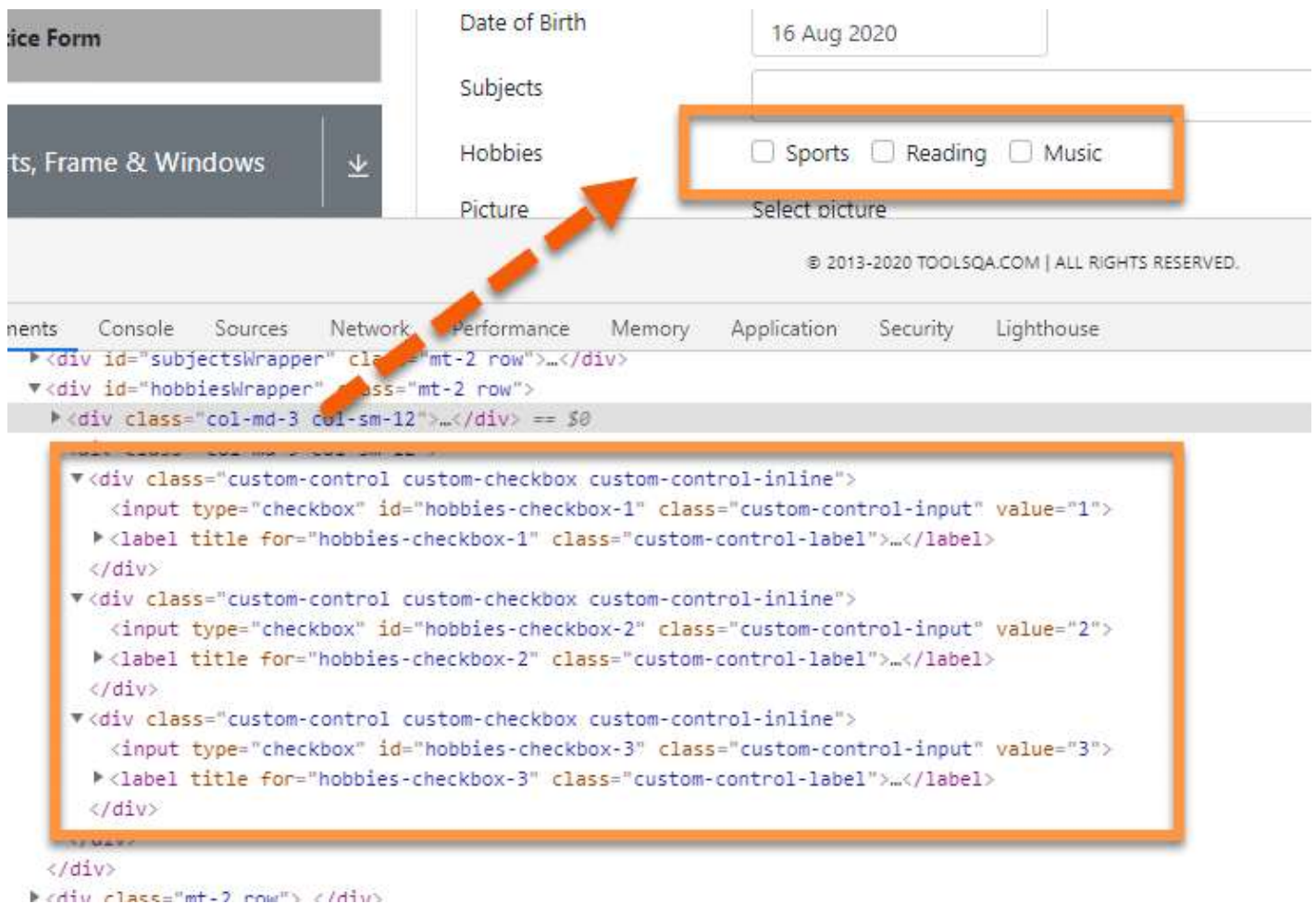
Mobile(10 Digits)	<input type="text" value="Mobile Number"/>
Date of Birth	<input type="text" value="16 Aug 2020"/>
Subjects	<input type="text"/>
Hobbies	<input checked="" type="checkbox"/> Sports <input checked="" type="checkbox"/> Reading <input type="checkbox"/> Music
Picture	Select picture <input type="button" value="Choose File"/> No file chosen
Current Address	<input type="text" value="Current Address"/>

So, this way, we can select a **CheckBox** by using a unique **XPath** and **select** the same by using the **"click "** operation.

How to locate and select a checkbox in Selenium using the CSS Selector locator?

As discussed in the previous section, sometimes it isn't easy to locate an element on the web page by using its identifier or even attribute value. The most efficient way is to use a **CSS Selector** to handle dynamic elements.

In the given image, we can see the element present in the **DOM** by highlighting it. We can use the given **HTML** structure to create a **CSS expression** that can recognize the element. We will try to select all three **checkboxes** using the **CSS selectors**.



To locate the *checkboxes*, we will first need to write the **CSS selector expression** for each element. Then we can use these in **Selenium WebElement** and then perform *select* or *click* operation.

As discussed earlier, the *checkboxes* can be selected either by locating and clicking the input element or can also be selected by clicking the label associated with the *checkboxes*. So, if we use the labels associated with the *checkboxes*, the below code snippet will locate the elements using CSS Selectors and will click on them for selecting the same:

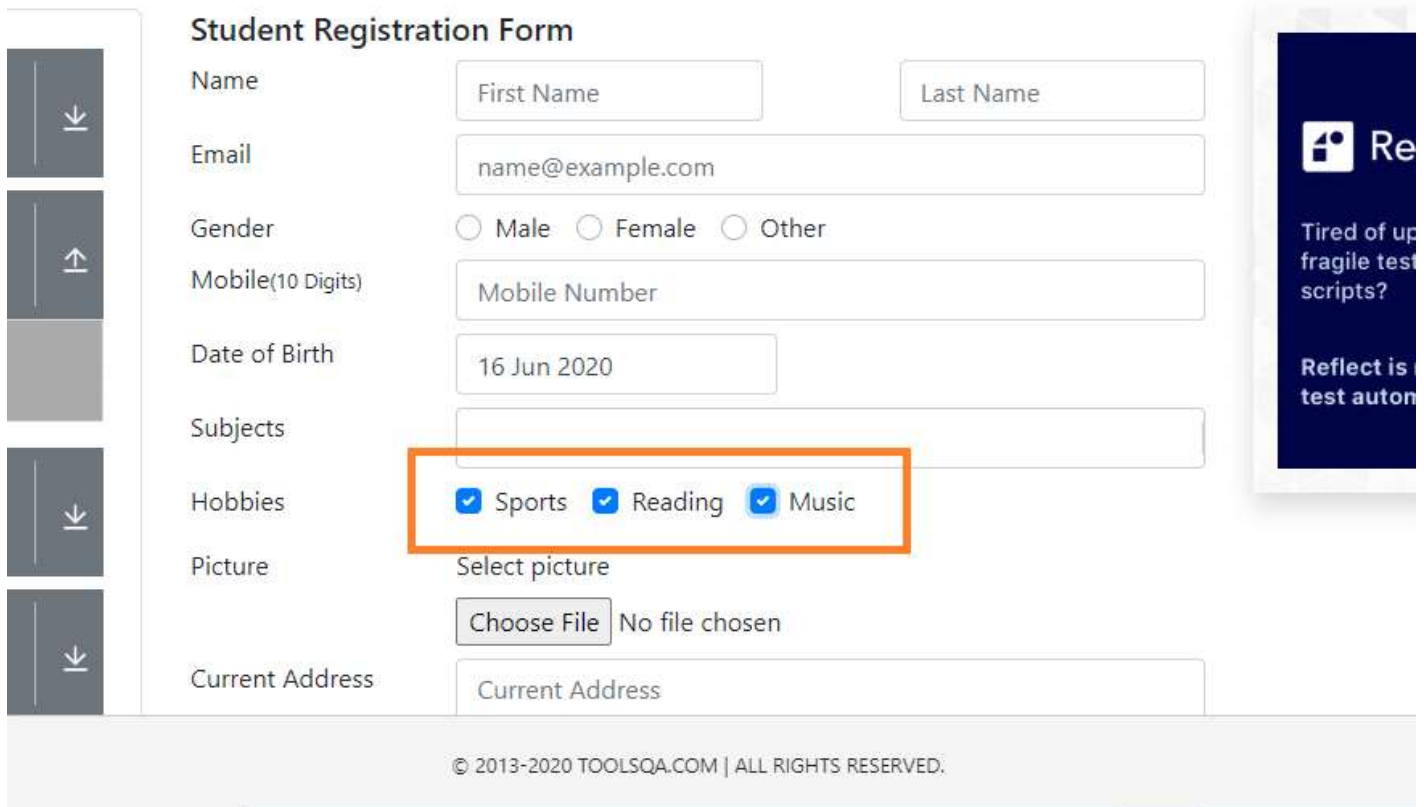
```
//Selecting the first checkbox
driver.findElement(By.cssSelector("label[for='hobbies-checkbox-1']")).click();

//Selecting the second checkbox
driver.findElement(By.cssSelector("label[for='hobbies-checkbox-2']")).click();

//Selecting the last check box
driver.findElement(By.cssSelector("label[for='hobbies-checkbox-3']")).click();
```

Using the above line of code, Selenium will locate the web element with specified **CSS Locator** and will perform the **click** operation on that. The execution of the above line of code will lead to the following state on the web page:

Practice Form



The screenshot shows a web form titled "Student Registration Form". On the left is a vertical sidebar with five buttons: a down arrow, an up arrow, a grey square, another down arrow, and a third down arrow. The form fields include: "Name" (split into "First Name" and "Last Name" text boxes), "Email" (text box with "name@example.com"), "Gender" (radio buttons for "Male", "Female", and "Other"), "Mobile(10 Digits)" (text box with "Mobile Number"), "Date of Birth" (text box with "16 Jun 2020"), "Subjects" (text box), "Hobbies" (checkboxes for "Sports", "Reading", and "Music", which are highlighted by an orange rectangle), "Picture" (a "Select picture" label and a "Choose File" button next to the text "No file chosen"), and "Current Address" (text box). On the right side of the form is a dark blue vertical banner with a white icon and the text "Re", "Tired of up fragile test scripts?", and "Reflect is test autom". At the bottom of the page is a footer with the text "© 2013-2020 TOOLSQA.COM | ALL RIGHTS RESERVED."

This way, we can select a **Checkbox** by using a unique **CSS Locator** and **check** the same by using the **"click "** operation.

How to perform validations on a CheckBox using Selenium WebDriver?

Selenium WebDriver provides certain methods that we can use for a *pre and post validation* of the states of a **CheckBox**. Few of these methods are:

isSelected(): Checks whether a checkbox is selected or not.

isDisplayed(): Checks whether a checkbox displays on the web page or not.

isEnabled(): Checks whether a checkbox is enabled or not

We can use these methods to validate the current state of the check boxes. E.g., to validate that after clicking the checkbox, whether we have checked or not, we can use the **"isSelected()"** method. In other words, it helps to validate the current state of the checkbox. Similarly, before clicking a checkbox, we can validate that whether is checkbox is displayed on the page and is enabled, then only click on the checkbox. So, such pre validations can be done using the **"isDisplayed()"** and **"isEnabled()"** methods.

Let's have a look at an example to understand how we can use all these validations. We will take the following *checkboxes* as an example:

Practice Form

Student Registration Form

Name: First Name, Last Name

Email: name@example.com

Gender: ☐ Male ☐ Female ☐ Other

Mobile(10 Digits): Mobile Number

Date of Birth: 15 Jun 2020

Subjects: [Text Input]

Hobbies: ☐ Sports ☐ Reading ☐ Music

Picture: Select picture, Choose File, No file chosen

Current Address: Current Address

How to use the `isSelected()` method to validate if the CheckBox is selected?

We can use the ***isSelected()*** method to validate the current state of the checkbox, whether we selected it or not. We can use it both in pre and post validation. E.g., we can perform the *click* operation on the checkbox when it is not already selected; otherwise, we can skip the operation, as shown by the below code snippet:

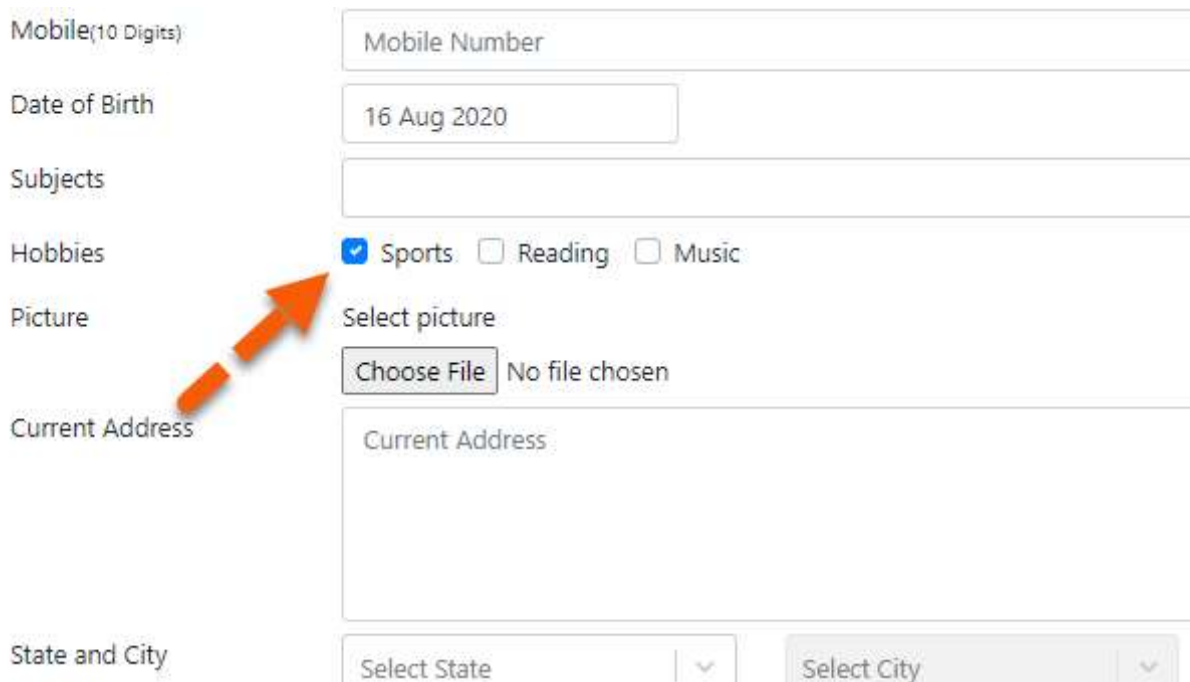
```
/**
 * Validate Checkbox isSelected method and click
 */

WebElement checkBoxElement = driver.findElement(By.cssSelector("label[for='hobbies-checkb
boolean isSelected = checkBoxElement.isSelected();

//performing click operation if element is not checked
if(isSelected == false) {
    checkBoxElement.click();
}
```

Once we ran this code, the code will first check whether the *checkbox* or not. Then an ***if condition*** will validate if the returned value is ***true*** or ***false***. In case it's false, i.e., the *checkbox will appear unchecked*. The code inside the ***if condition*** will execute, and the checkbox will check.

The output of the above code will be:



Mobile(10 Digits) Mobile Number

Date of Birth 16 Aug 2020

Subjects

Hobbies ☒ Sports ☐ Reading ☐ Music

Picture Select picture
Choose File No file chosen

Current Address

State and City Select State Select City

This way, we can do a conditional select of the checkbox by first checking the state of the *CheckBox*.

How to use isDisplayed() method to validate if the CheckBox is displayed?

Taking the same example as above, let's write a simple selenium code to validate if a given *checkbox displays* or not. If we can see it, then we will click and select it.

The code snippet for the scenario, as mentioned above, will be:

```
/**
 * Validate Checkbox using isDisplayed() and click
 */
WebElement checkBoxElement = driver.findElement(By.cssSelector("label[for='hobbies-checkb
boolean isDisplayed = checkBoxElement.isDisplayed();

// performing click operation if element is displayed
if (isDisplayed == true) {
    checkBoxElement.click();
}
```

The above test will validate if the given checkbox displays on the page or not. If it is displayed, then it will make a selection. The output of the above code will be the same as was in the case of **"isSelected()"** as the specified checkbox is displayed, and it will check using the click option.

How to use the isEnabled() method to validate if the CheckBox is enabled?

Assume we want to check a checkbox, but during the runtime, there may be a scenario where it is *disabled*. To handle such scenarios, Selenium offers a method called **"isEnabled()"**. As the name suggests, this method validates if the given web element is enabled or not. This method will return the boolean value based on the status of the element. It will return *'true'* if the element is in enabled status else *'false.'*

For example, we want first to check if the **"Sports"** checkbox is in enabled status or not. Then we will proceed with other actions.

Practice Form

Date Of Birth: 16 Aug 2020

Subjects

Hobbies: ☐ Sports ☐ Reading ☐ Music

Picture: Select picture

© 2013-2020 TOOLSQA.COM | ALL RIGHTS RESERVED

Selenium IDE Console:

```

<div id="genderWrapper" class="mt-2 row">...</div>
<div id="userNumber-wrapper" class="mt-2 row">...</div>
<div id="dateOfBirth-wrapper" class="mt-2 row">...</div>
<div id="subjectsWrapper" class="mt-2 row">...</div>
<div id="hobbiesWrapper" class="mt-2 row">
  <div class="col-md-3 col-sm-12">...</div>
  <div class="col-md-9 col-sm-12">
    <div class="custom-control custom-checkbox custom-control-inline">
      <input type="checkbox" id="hobbies-checkbox-1" class="custom-control-input" value="1"> == 1
    </div>
  </div>
</div>
  
```

The below code snippet will click on the checkbox only when enabled:

```

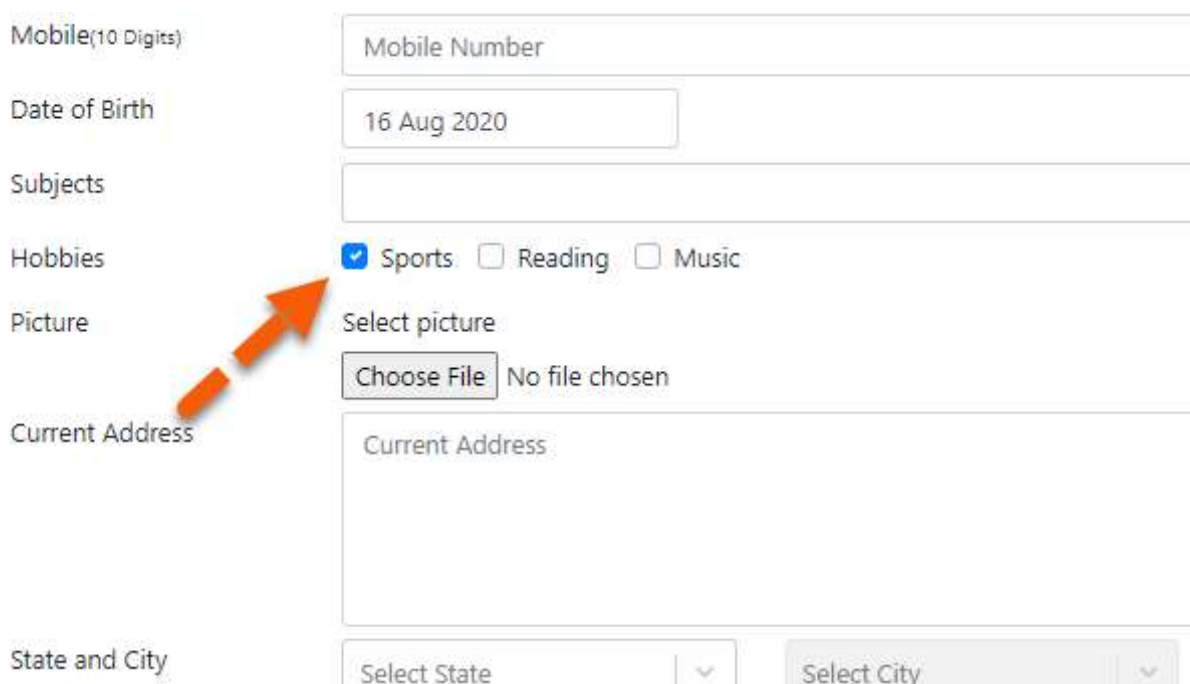
/**
 * Validate checkbox using isEnabled() and then click
 */

```

*/

```
WebElement checkBoxElement = driver.findElement(By.cssSelector("label[for='hobbies-checkb  
boolean isEnabled = chckBxEnable.isEnabled();  
  
// performing click operation if element is enabled  
if (isEnabled == true) {  
    checkBoxElement.click();  
}
```

The above code will first check if the element is in an enabled state or not; if enabled, it will perform the click operation. If disabled, it will not perform any operation. As the checkbox enables, it will click the checkbox and will show the output as shown below:



Mobile(10 Digits) Mobile Number

Date of Birth 16 Aug 2020

Subjects

Hobbies ☒ Sports ☐ Reading ☐ Music

Picture Select picture
Choose File No file chosen

Current Address

State and City Select State Select City

The above code will first check if we have enabled the element; if it is, it will perform the click operation. If disabled, no operation will perform.

Let's now try to write a single test, which will perform all the operations as mentioned above and validations on the radio buttons at the **ToolsQA Demo site**:

```
package TestPackage;  
  
import java.util.List;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.WebElement;  
import org.openqa.selenium.chrome.ChromeDriver;  
  
public class CheckBoxOperations {
```



```

public static void main(String[] args) {

    String exePath = "C:\\Selenium\\chromedriver\\chromedriver.exe";
    System.setProperty("webdriver.chrome.driver", exePath);
    WebDriver driver = new ChromeDriver();
    driver.get("https://www.demoqa.com/automation-practice-form");
    driver.manage().window().maximize();

    /**
     * Validate isSelected and click
     */

    WebElement checkBoxSelected = driver.findElement(By.cssSelector("label[for=checkbox1]"));
    boolean isSelected = checkBoxSelected.isSelected();

    // performing click operation if element is not selected
    if(isSelected == false) {
        checkBoxSelected.click();
    }

    /**
     * Validate isDisplayed and click
     */
    WebElement checkBoxDisplayed = driver.findElement(By.cssSelector("label[for=checkbox2]"));
    boolean isDisplayed = checkBoxDisplayed.isDisplayed();

    // performing click operation if element is displayed
    if (isDisplayed == true) {
        checkBoxDisplayed.click();
    }

    /**
     * Validate isEnabled and click
     */
    WebElement checkBoxEnabled = driver.findElement(By.cssSelector("label[for=checkbox3]"));
    boolean isEnabled = checkBoxEnabled.isEnabled();

    // performing click operation if element is enabled
    if (isEnabled == true) {
        checkBoxEnabled.click();
    }

}
}

```

This code snippet can directly execute by creating a new class in your eclipse project. In the new class, you can copy-paste all these codes to run. You can also learn in detail about how to create and execute selenium tests from our tutorial ["Creating Selenium Tests"](#).

Key Takeaways

CheckBoxes are denoted by `<input type="checkbox">` HTML tag

We can identify the Checkbox elements by locator strategies such as id, XPath, CSS selector, or any other selenium locator that can uniquely identify the checkbox

We can select a checkbox either by using the `click()` method on the input node or on the label node that represents the checkbox.

Selenium also offers validation methods like `isSelected`, `isEnabled`, and `isDisplayed`. We can use these methods to make sure checkboxes are in the correct status before performing any operation.