Selenium Checkboxes and Radio Buttons

In Selenium, **checkboxes** and **radio buttons** are handled using the `.click()` method just like other web elements. The key is to locate the correct input elements using `findElement()` or `findElements()`, and use conditions to determine their state using `.isSelected()`.

✅ 1. Handling Checkboxes in Selenium

✔️ HTML Example

```html
<input type="checkbox" id="java" name="language" value="Java">
<label for="java">Java</label>

<input type="checkbox" id="python" name="language" value="Python">
<label for="python">Python</label>
```

✔️ Java + Selenium Code

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

public class CheckboxExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();

        driver.get("https://example.com/checkboxpage");

        // Locate checkboxes
        WebElement javaCheckbox = driver.findElement(By.id("java"));
        WebElement pythonCheckbox = driver.findElement(By.id("python"));

        // Select Java if not selected
        if (!javaCheckbox.isSelected()) {
            javaCheckbox.click();
        }

        // Deselect Python if already selected
        if (pythonCheckbox.isSelected()) {
            pythonCheckbox.click();
        }

        driver.quit();
    }
}
```

📻 2. Handling Radio Buttons in Selenium

✔️ HTML Example

```html
<input type="radio" id="male" name="gender" value="Male">
<label for="male">Male</label>

<input type="radio" id="female" name="gender" value="Female">
<label for="female">Female</label>
```

✔️ Java + Selenium Code

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

public class RadioButtonExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();

        driver.get("https://example.com/radiobuttonpage");

        // Locate radio buttons
        WebElement maleRadio = driver.findElement(By.id("male"));
        WebElement femaleRadio = driver.findElement(By.id("female"));

        // Select Female if not already selected
        if (!femaleRadio.isSelected()) {
            femaleRadio.click();
        }

        driver.quit();
    }
}
```

🔍 Explanation

Method	Description
`.click()`	Clicks the checkbox or radio button.
`.isSelected()`	Returns **`true`** if the element is selected.
`findElement()`	Locates a single web element.
`findElements()`	Returns a list of matching elements.

🧠 Key Points

- Always use **`isSelected()`** to check current state before clicking.
- Radio buttons allow only **one** selection per group (based on **`name`** attribute).
- Checkboxes allow **multiple** selections.
- You can use **`getAttribute("checked")`** to confirm selection state if needed.

Would you like real-time interaction with dynamic checkboxes using loops or using Page Object Model (POM)?

Great! Let's take it further with **dynamic checkboxes using loops** and then implement the same using the **Page Object Model (POM)** in Selenium with Java.

✅ 1. Dynamic Checkboxes using Loops

✔️ HTML Sample (assume multiple checkboxes)

```html
<input type="checkbox" name="skills" value="Java"> Java<br>
<input type="checkbox" name="skills" value="Python"> Python<br>
<input type="checkbox" name="skills" value="Selenium"> Selenium<br>
<input type="checkbox" name="skills" value="DevOps"> DevOps<br>
```

✔️ Java + Selenium Code with Loop

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

import java.util.List;

public class DynamicCheckboxes {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com/checkboxform");

        // Locate all checkboxes with name='skills'
        List<WebElement> checkboxes = driver.findElements(By.name("skills"));

        // Loop through and select specific ones (e.g., Selenium and DevOps)
        for (WebElement checkbox : checkboxes) {
            String value = checkbox.getAttribute("value");

            if (value.equals("Selenium") || value.equals("DevOps")) {
                if (!checkbox.isSelected()) {
                    checkbox.click();
                }
            }
        }

        driver.quit();
    }
}
```

🧱 2. Page Object Model (POM) for Checkboxes and Radio Buttons

📄 Page Class: `CheckboxPage.java`

```java
package pages;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

import java.util.List;

public class CheckboxPage {
    WebDriver driver;

    // Constructor
    public CheckboxPage(WebDriver driver) {
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }

    // Elements using @FindBy
    @FindBy(name = "skills")
    List<WebElement> skillCheckboxes;

    @FindBy(id = "male")
    WebElement maleRadio;

    @FindBy(id = "female")
    WebElement femaleRadio;

    // Method to select specific checkboxes
    public void selectSkills(String... skillsToSelect) {
        for (WebElement checkbox : skillCheckboxes) {
            String value = checkbox.getAttribute("value");
            for (String skill : skillsToSelect) {
                if (value.equalsIgnoreCase(skill) && !checkbox.isSelected()) {
                    checkbox.click();
                }
            }
        }
    }

    // Method to select radio button
    public void selectGender(String gender) {
        if (gender.equalsIgnoreCase("male") && !maleRadio.isSelected()) {
            maleRadio.click();
        } else if (gender.equalsIgnoreCase("female") && !femaleRadio.isSelected()) {
            femaleRadio.click();
        }
    }
}
```

📄 Test Class: `CheckboxTest.java`

```java
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import pages.CheckboxPage;

public class CheckboxTest {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com/form");

        CheckboxPage checkboxPage = new CheckboxPage(driver);

        // Select multiple checkboxes
        checkboxPage.selectSkills("Java", "Selenium");

        // Select radio button
        checkboxPage.selectGender("Female");

        driver.quit();
    }
}
```

🧠 Benefits of Using POM

Feature	Benefit
`PageFactory`	Easy initialization of elements
Separation of Concerns	Easy maintenance and readability
Reusable Methods	Avoid code duplication
Scalable	Good for large test frameworks

Would you like me to show the **TestNG version with assertions**, or a **POM setup with Maven and reports** next?

Great! Let's prepare you for **interview questions** on **Checkboxes and Radio Buttons in Selenium**, including:

1. **Theory Questions**
2. **Practical Coding Questions**
3. **Real-time Scenario Questions**
4. **Bonus Tips & Mistakes to Avoid**

✅ 1. THEORY QUESTIONS (With Best Answers)

❓ Q1: How do you handle checkboxes in Selenium?

Answer:

> In Selenium, I handle checkboxes by locating the element using locators like `id`, `name`, or `xpath`, and then use `.click()` to toggle the selection. Before clicking, I check its current state using `.isSelected()` to avoid unnecessary interaction.

❓ Q2: What method checks if a checkbox or radio button is selected?

Answer:

> The `.isSelected()` method is used to verify whether a checkbox or radio button is currently selected. It returns a boolean (`true` or `false`).

❓ Q3: What is the difference between a checkbox and a radio button?

Answer:

> A checkbox allows **multiple selections**, whereas a radio button allows **only one selection** within a group defined by the same `name` attribute.

❓ Q4: How do you select multiple checkboxes using Selenium?

Answer:

> I use `findElements()` to collect all checkboxes with the same name or pattern and then iterate over them in a loop. I use conditions like `getAttribute("value")` or `.isSelected()` to select the desired options dynamically.

🧪 2. PRACTICAL CODING QUESTIONS

❓ Q5: Write a code snippet to select the checkbox with value "Selenium".

```java
List<WebElement> checkboxes = driver.findElements(By.name("skills"));
for (WebElement cb : checkboxes) {
    if (cb.getAttribute("value").equalsIgnoreCase("Selenium")) {
        if (!cb.isSelected()) {
            cb.click();
        }
    }
}
```

❓ Q6: Write code to ensure "Female" radio button is selected.

```java
WebElement femaleRadio = driver.findElement(By.id("female"));
if (!femaleRadio.isSelected()) {
    femaleRadio.click();
}
```

🔄 3. REAL-TIME SCENARIO QUESTIONS

❓ Q7: In your project, how did you handle dynamic checkboxes?

Answer:

> In one of my projects, the checkbox values were generated dynamically from the database. I used `findElements(By.xpath(...))` to collect all checkboxes, looped through them using `.getAttribute("value")`, and clicked only the required ones (like "AWS", "DevOps"). I made sure to avoid clicking already selected boxes using `.isSelected()`.

❓ Q8: What if clicking a checkbox fails due to overlay or not being in view?

Answer:

> In such cases, I first use `scrollIntoView()` with JavaScript to bring the checkbox into view:

```java
((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", checkbox);
```

> And then click. If there's an overlay, I check the DOM and use waits (`ExpectedConditions.elementToBeClickable`) before interaction.

⚠️ 4. BONUS: COMMON MISTAKES TO AVOID

Mistake	Why it Fails	Fix
Using **`click()`** without checking **`isSelected()`**	Might uncheck already selected checkbox	Always verify state first
Using wrong locator	Might click wrong element	Use accurate XPath/CSS
No wait for visibility	May throw **`ElementNotInteractableException`**	Use **`WebDriverWait`**
Clicking hidden elements	Not allowed in Selenium	Use JavaScript Executor if required

🚀 Extra Tip (for Experts)

If checkboxes or radio buttons are **custom-designed (e.g., hidden inputs)**, interact with their **label tag** or use JavaScript:

```java
WebElement label = driver.findElement(By.xpath("//label[@for='java']"));
label.click(); // this will check the checkbox even if input is hidden
```

Would you like mock interview-style Q&A or a quiz next?