Don’t miss to read the 10 of the latest Selenium testing interview questions and answers.

If you are a test automation developer preparing for a telephonic round, then it will help you greatly in your preparations.

For your information, today we picked questions from a wide range of topics. Also, many of these we added after listening to the feedback from our readers.

We hope this post to help you in clearing the quick chatter round.

Next, if you wish to think beyond the telephonic round, then you must go through the below article.

Prepare For The Selenium Interview.

Selenium Interview Questions and Answers.

You can come to read the above post after quickly reviewing the below questions and answers.

Top 10 Selenium Testing Interview Questions And Answers.

Selenium Testing Interview Questions.

Q-1. What Are The Primary Differences Between XPath And CSS Locators?

Ans. 1. First of all, the CSS selector responds faster than the XPath.
2. Unlike the XPath locators, CSS selectors exhibit the consistent behavior across different browsers.
3. It’s much easier to form a CSS selector than an XPath.

Q-2. When Is It Beneficial To Use XPath In Selenium Test Automation?

Ans. There are web pages which host dynamic elements for rendering runtime data. Such elements don't have fixed locators. We can utilize relative XPath expressions to take care of them.

Q-3. How Would You Locate Paragraph Elements Both Immediate And The Descendants Of A Div Element?

Ans. We can use both XPath and CSS to select elements inside a div.
Locate the first-level child.
In XPath, the forward slash "/" indicates an immediate child whereas a ">" sign represents it in CSS.

```
//div/p
div > p
```

Select the descendants of a div.
With the help of double slash <//>, we can form the XPath and use a space char in CSS to locate the descendants of a div.

```
//div//p
div p
```

Q-4. When Should You Use A Headless Driver In Selenium Testing? Also, How Will You Handle If A Test Fails Using The Headless Driver?

Ans. Test execution using headless driver is faster than running with a standard web driver. Mostly, we use it when we wish to run the tests in a silent mode. The likely scenarios where it can be used are when running the tests from the command line for nightly execution or launching them from a CI tool like Jenkins or Bamboo.
Since this driver doesn't have any built-in mechanism to handle errors, so the developers have to devise error handlers that can capture screenshots or log them to the console or output into a file.

Q-5. What Is The Application Of Java Reflection APIs In Selenium Webdriver?

Ans. We can use the concept of Reflection for implementing the Keyword and data-driven framework using Selenium Webdriver.
The keywords hold the name of Selenium methods to be called by the framework. The Reflection APIs provides the ability to translate the name into a method object which accepts arguments and invokes the actual Selenium API.

Q-6. How To Refresh A Web Page Using Selenium That Works On All Browsers?

Ans. There are two ways we can reload a web page in the browser. Both of these work on all type of browsers that Selenium supports.
With the help of <navigate> command.

```
driver.get("https://www.google.com/");
driver.navigate().refresh();
```

By calling a Javascript method.

```
driver.executeScript("history.go(0)");
```

Q-7. What Is Marionette, How Does It Work And When Should You Use It?

Ans. Marionette is the web driver module for Mozilla's Gecko engine. Among the test automation developers, it is also famous as the Gecko Driver.
It comprises two modules, first is the server which accepts requests and executes whereas the second is a client to forward commands to the server.

We should use Marionette to run UI tests in Firefox browser. A test automation developer should add its dependencies to the framework, import the required classes and call the methods to control the web page. It also returns the status of commands executed on the browser which can help in verifying the authenticity of the actions performed.

Q-8. What Is An StaleElementReferenceException In Selenium And How Do You Resolve It?

And. The stale element exception occurs if the element is in either of the following two states.
1. Removed or deleted altogether.
2. Detached from the DOM.

To fix this issue, we can try to access the element multiple times in a loop before finally throwing the stale element exception.

Q-9. What Is The Use Of The Page Object Model (POM) And Page Factory?

Ans.
Page Object Model.
It is a design pattern which refers to treat web pages as classes. Each member of the class points to an element that exists on the page. It offers an elegant way of creating test routines that have greater readability, less maintenance, as well as easy to extend.

Suggested Reading.
How to Implement Page Object Model using Selenium Webdriver?
Page Factory.
Page factory enables initialization of each element with a counterpart element on the target web page. It opens up the use of Annotations like @FindBy to define strategies for selecting elements.

Q-10. What Is The Best Way To Integrate The Page Object Model In A Selenium Webdriver Project?

Ans. While implementing the Page Object Model, we should define the page classes exactly in the format as mentioned here. The code below emphasizes that we should initialize the web elements using the Page factory's InitElements method.

```
public class Page
{
    public IWebDriver _browser;

    public Page(IWebDriver browser)
    {
        this._browser = browser;
        PageFactory.InitElements(_browser, this);
    }
}
```