



W h i t e p a p e r

# Everything You Need To Know About **Cross Browser Testing**



# Contents

- Chapter 1 Introduction of Desktop Browsers
- Chapter 2 What is Cross Browser Testing and why do we need it?
- Chapter 3 Top browsers to test on for Cross Browser Testing in 2021
- Chapter 4 Challenges of Cross Browser Testing
- Chapter 5 Strategies To Perform Cross-Browser Testing
- Chapter 6 How is Selenium Grid useful for successful Cross Browser Testing?
- Chapter 7 How pCloudy can help execute Cross Browser Tests on-demand?

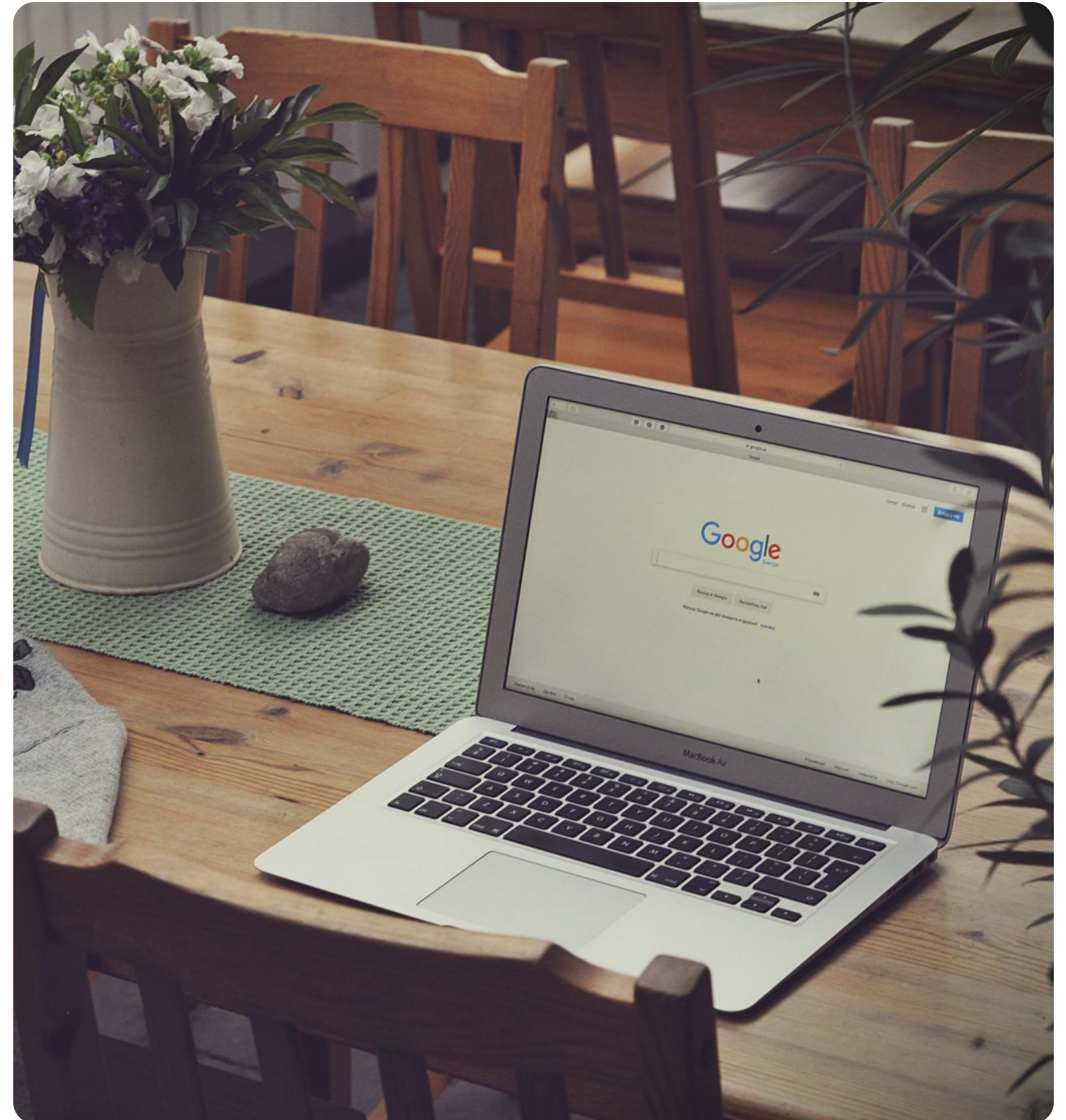




# Introduction of Desktop Browsers

Let's look back to the history in 1991 when the World Wide Web(WWW) was introduced and users around the world were able to access web pages filled with tons of valuable information for free. With such huge research and development, people were amazed with the fact that they were now able to share information and communicate with anyone from anywhere in the world.

Gradually, with time, we have been seeing the evolution of the internet and web pages. In the early days, people used to access web pages with a single classic browser, known as the Internet Explorer. But now, we have hundreds of browsers that are spread across the browser market. The development and usage of different browsers came as a surprise and depended majorly and on the regions and devices that are being used. However, with every new tech product launched in the market, there come a few limitations as well. With the introduction of multiple browsers and increasing user base in the world, the browser compatibility issues jacked up like no one's business. And to overcome browser compatibility issues, tech giants came up with the solution of Cross Browser Testing.





# What is Cross Browser Testing and why do we need it?

We already know that there are multiple browsers available in the market that people can directly download and install in their systems according to their requirement. Earlier, there used to be only one browser i.e. Internet Explorer (IE), hence, there were no compatibility issues, the web development was processed according to IE and web pages were accessed on IE itself.

But now, with time, the technology has changed which has introduced a number of browsers in the market, giving rise to the problem of web compatibility.

The aim of cross browser testing is to test web apps and websites on all targeted browsers. The kind of front-end testing that helps with validating that the site works as smoothly as on other comparative browsers is cross browser testing. Performing such tasks manually on a number of available browsers can be tedious, to overcome this and to add productivity in the cross browser testing process, Selenium has come up with Selenium Grid where we can run our automated tests parallelly on different environments.

Though now, we know a little bit about cross browser testing, there are a lot of things that highlight the significance of cross browser testing on which we will throw some light here.

Many times you might have noticed that the single specific feature of a web application is working fine on the chrome browser, but is failing to run properly on the firefox browser. Or maybe the font appears just fine on a chrome browser, but looks unpleasant on the firefox browser. Such issues are determined a failure in cross browser testing.

We should always be alert with such small and critical issues. Ultimately the product that is being developed would be used by the client and client won't think twice while switching on to the competitor product that works as smoothly on any desired browser. The only way to avoid such compatibility issues and to deliver a quality ensured product is to test the application across multiple browsers.

Once the quality assurance team runs cross browser tests from an Application Under Test (AUT), they get a better clarity on the following points:

- What's going wrong with a particular area of the application
- In which specific browser and browser version the issue is occurring
- Collection of consolidated logs from browser dev tools to analyse and share with the dev team for a quick fix

Once the web application passes the cross browser tests, the application gets enhanced in terms of UI and UX. As an edge scenario, cross browser testing can also increase the product's conversion rate. The most definite reason can be: the users keep on visiting and accessing the site if they find a good response in all aspects on their preferred browser.



Consider an instance where your organisation is developing a Human Resource Management System(HRMS) and meanwhile the sales team is selling the demo services of HRMS around the world. Now your targeted audience is the U.S, Europe, Asia and Africa, considering the population of these continents which is approximately 6.85 billion, do you think they will be using a specific browser despite the fact of having hundreds of browsers in the world? The answer is No. As a matter of fact, most of the development community use chrome for developing or debugging a web application.

Hence, it is the primary responsibility of a test engineer to uncover as many bugs as possible by testing the same application on all the widely used browsers to ensure that the product that is being delivered to the end user responds perfectly on any and every browser.



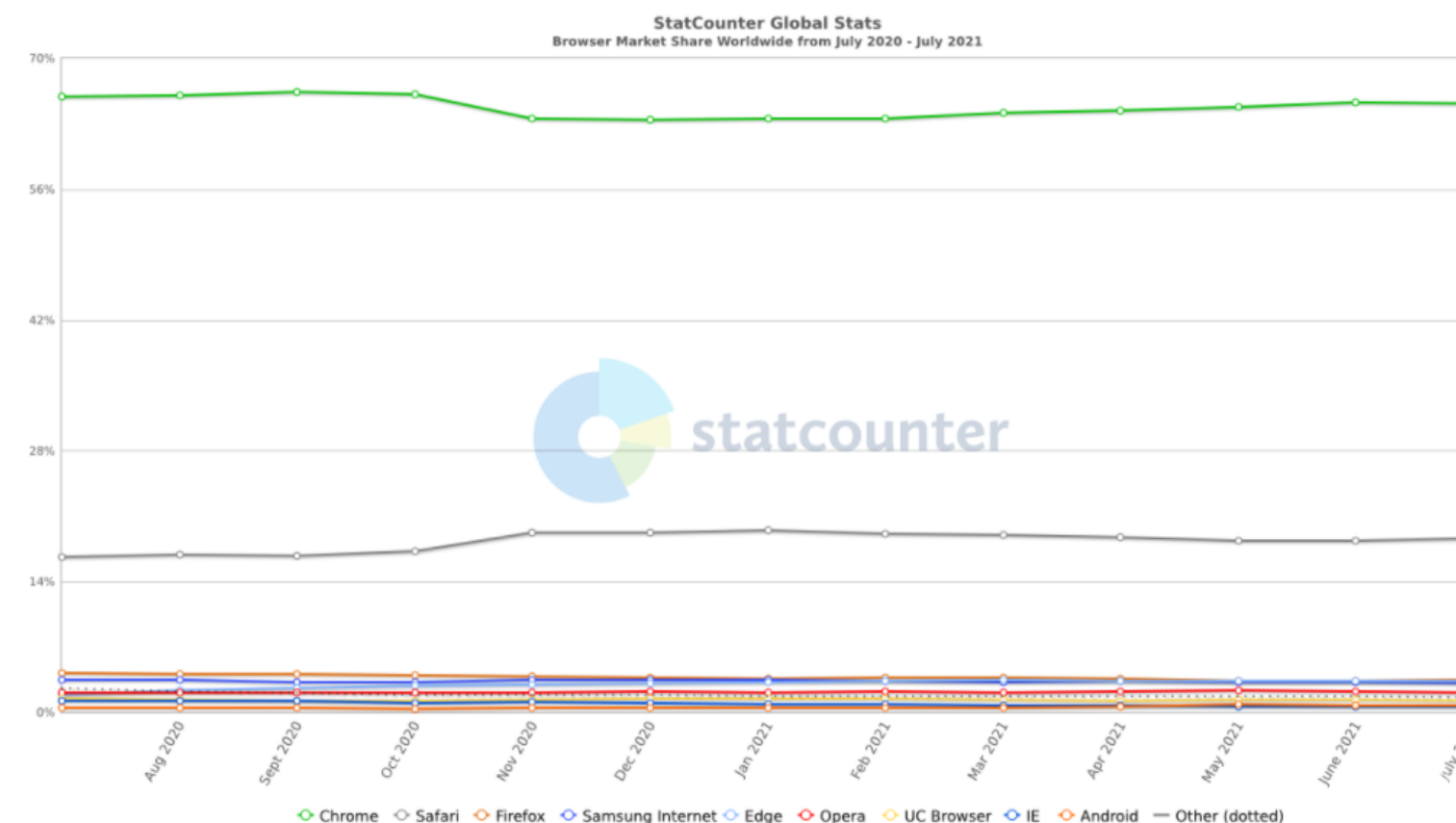
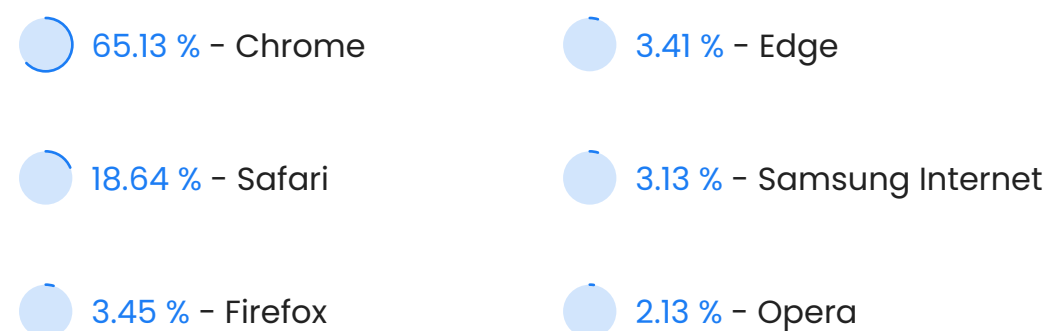


# Top browsers to test on for Cross Browser Testing in 2021

Different browsers are preferred by different users depending upon their geographical location and devices they use. It is next to impossible to test an application on all the preferred browsers manually, but still, do you think the browsers you prefer for testing are the right one?

Well, to figure this out, it is important to get familiarized with the stats of most used browsers in the world. This will not only help knowing the mostly preferred browser but also help you in prioritizing your browsers for testing.

As per statcounter global stats, the browser market share worldwide is pictured above and mentioned below in figures:



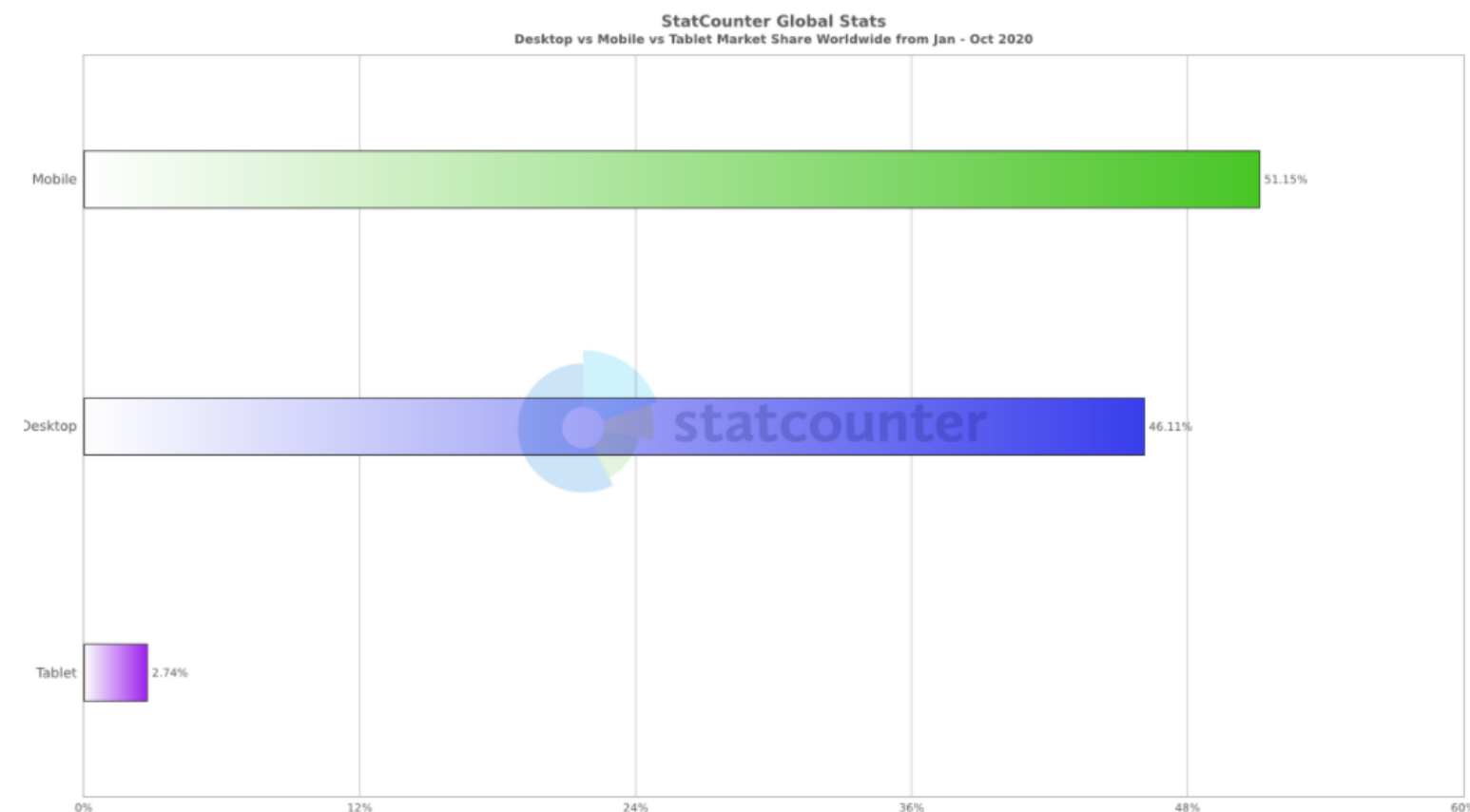
Using chrome browser has always been crystal clear, but it is important to know what the other preferred browsers are in order to test for compatibility. The above mentioned popular browsers can help in prioritizing the testing process based on the final browsers to be included for testing an application.

There are many other browsers that are not mentioned above but can be identified as critical markers for your application. For example, the outdated browser IE and legacy Edge might need to be included in your browser list in case your existing customer base or targeted audience prefers to use it.

Cross browser testing never specifies to only test an application on multiple browsers, but by the term cross browser testing, we actually mean to test an application against different environments. The testing environment can be a combination of devices, browsers, browser versions and operating systems.

It is good to have an understanding of the most used browsers in the world to prioritize the browser list to test, but ideally, that would not be enough for successful cross browser testing. You will also need to know where your application will most likely be used in terms of which devices will your application run on like is it on desktops, mobile or tablets, etc.

Below is the graph from statcounter global stats that picturize the usage of desktops, mobile and tablets.



From this graph we can see that usage of mobile phones around the world is leading with a market share of 51.15%, hence, along with desktops, it is equally important to test our applications on mobile browsers to achieve success in cross browser testing.



# Challenges of Cross Browser Testing

## with a **solution to overcome**

### 1 **Testing on too many browsers including the legacy ones can be hectic and tedious :**

This is one of the most prominent challenges of cross browser testing – to maintain a collection of multiple browsers with different versions. Not all organizations can afford to host a huge number of browser collections in-house. Let's take a quick peek at an example – Suppose you have successfully tested an application on the firefox browser and you assume that since it works fine on this browser, it will work just fine on all other firefox browsers. However, we need to know that there is no guarantee or assurance that the said application will work perfectly on the different versions of firefox being used by the client. The UI and UX has to be taken care of not only on latest browser versions but also on legacy versions plus on outdated browsers like IE.

The solution to overcome this challenge is to first get familiarized with your targeted browsers and browser versions. Once you have listed down all the preferred testing browsers and browser versions, the application can be automated and the suite can be executed on either cloud based cross browser testing platform or the in-house testing infrastructure.

### 2 **Testing too many combinations :**

Cross browser testing is not only about testing on multiple browsers, it also includes testing against different combinations of devices, browsers, browser versions and operating systems. Maintaining a numerous collection of devices, browsers, browser versions and operating systems in-house would be next to impossible. Even considering these combinations for manual testing can take a lot of time, effort and performance time.

The solution to overcome this challenge can be a cloud based cross browser testing platform that can offer multiple combinations of devices, browsers, browser versions and operating systems with a functionality to run automated tests on cloud. The biggest perk of moving to a cloud based service platform is to avoid the headache of setting and maintaining in-house testing infrastructure.

### 3 **Frequent updation of browsers :**

Browsers are getting evolved and updated quite often. Considering the example of a chrome browser, we probably get a newer version of chrome every two months. Few people prefer to keep chrome settings as auto-update, and for the people who never prefer to update, for those the chrome gets a force update after a certain wait time. Hence, gradually everyone has to use the updated browsers.

This prompts the testing team to test their application in every new browser version that is released along with the old browser versions already present to ensure that everything works smoothly on the targeted browser versions.



The solution to overcome this challenge is to keep your automation suite browser friendly. This means that the process of handling different browser versions should also be automated. We can use the library named “WebDriverManager” that can easily be integrated with the automation framework to take care of the automation suite in case the browser gets updated.

#### 4 In-house testing infrastructure setup can be a burden to maintain :

Setting up an in-house testing infrastructure might not be a feasible solution specially for the organization that has a huge product with a huge customer base.

Maintaining browsers, setting up new devices when launched in the market, upgrading the operating systems of laptops or mobiles, etc require a dedicated team which can cost a lot of expense to an organisation over time.

The best solution to overcome this challenge is to move to a cloud based platform that can offer stable and reliable infrastructure to run cross browser tests manually as well as through automation. The tests that fail via automation can be easily performed manually on the same cloud based platform to validate the false positive or false negative results.





# Strategies To Perform Cross-Browser Testing

Now that we understand a bit about cross browser testing and its challenges, let's now discuss some strategies to ease the process of cross browser testing.

## ● Prepare test scenarios and a list of targeted browsers

There are hundreds of browsers being used by millions of people around the world. Testing across all the browsers would definitely not be a possible task. Hence, it is important to target your browsers.

The possible solution to list down the targeted browsers can be: if you are developing a site for a specific client, simply and directly ask if there are any specific browsers that they would be using to access the site, if yes, add those browsers in your target list, if no, create a list by researching the popular browsers in the market worldwide or specific to clients target region.

The next step is to check whether the web application that is being developed will be used on only mobile devices or desktops or both. This strategy can help you define the combinations of devices and browsers to test.

Next, let's discuss the preparation of test scenarios. It is important to have smoke test cases that help in coverage of all the critical functionalities. Some of the important points to remember while creating test scenarios are as follows:

- The objective against which the application was built should be working. Basic functionalities that include login, payment gateway, navigation, queues etc should be working.
- Common issues found with cross browser testing like overlapping of elements or misalignment can ruin the UI and UX. Hence, along with the end-to-end flow testing, GUI test cases are also important.
- Modern web applications are known to be complicated. The responses of web pages and web elements may vary from browser to browser. It is important to take care of responsiveness across browsers while testing.

## ● Usage of virtual machines and emulators in the right manner

Cross platform testing has always been a part of cross browser testing as it includes testing of an application against combinations of different devices, different operating systems and browsers. Considering an example, there might be a possibility, the application that runs perfectly on Safari in a Windows OS might not work as smoothly as it works on Safari in the MacOS.

Setting up multiple operating systems can be a complicated task, hence, it is good to have few virtual machines (VM) hosted on tester's machines. Further, different operating systems can be installed on VM's along with the required browsers.





Though for the organisations that have a huge product with a huge customer base, the usage of virtual machines and emulators is not recommended as this will require a separate dedicated team for maintenance. As an alternative, such organisations can prefer moving on to the cloud based cross browser testing platform that provides a leverage to run automated tests against different environments parallelly to save execution time.

#### Automation of test scenarios

The product must be automated as much as possible. Nowadays, many of the organisations follow an agile model where development and testing are processed in parallel. As soon as the functionality gets deployed to the QA environment for testing the tested build has to be deployed on production to meet client expectations. Hence, organisations need QA to deliver their tasks earlier which is only possible when the regression and smoke test suite is automated.



## How is Selenium Grid useful for successful Cross Browser Testing?

Selenium is a robust and portable test automation framework that allows you to develop functional tests for your web application. Looking at the history of Selenium, it comes up with three major tools:

- **Selenium IDE:** It is a record and play back tool that auto creates testing scripts by multiple actions performed on a web application.
- **Selenium WebDriver:** It is a powerful testing framework that allows you to create scripts to automate your web testing process and execute on multiple browsers.
- **Selenium Grid:** It is a tool that is used for concurrent execution of multiple test cases on different browsers, machines and operating systems. This tool was developed to fulfil the needs of cross browser compatibility testing. Selenium Grid basically treats the master computer as a hub and distributes automated test cases among various slave machines called the nodes of the hubs.

Performing cross browser testing manually was one of the roadblocks in the process of testing as it required a lot of resources and time to test the same application features against different combinations of devices, browsers, browser versions and operating systems.

Thankfully, with the introduction of Selenium Grid, the cross browser tests execution time has drastically reduced. The Selenium Grid is highly scalable as a new node can be connected on demand with a hub whenever there is a need to have a new browser version for testing.

Cross browser testing with Selenium Grid can be highly valuable in speeding up the entire test automation process by simply taking the advantage of parallelism while executing the tests on multiple nodes. Though an in-house Selenium Grid infrastructure is a scalable approach, however, its scalability is limited to a certain extent. With the demand of new devices and operating systems, the in-house Selenium Grid might incur repetitive expenses to fulfill cross browser testing needs.

Setting up in-house Selenium Grid might cost organisations a lot of expense along with a dedicated team to maintain the infrastructure. For a budget friendly Selenium Grid, cloud based cross browser testing with Remote Selenium Grid is more preferable where you can simply avoid the headache of maintaining the infrastructure and can still run as many parallel sessions as you want. A cloud based Selenium Grid can save the infrastructure maintenance time and efforts and help one to focus on improving the overall test coverage which can further lead to better quality of product.





# How pCloudy can help execute Cross Browser Tests **on-demand?**

pCloudy is a continuous cloud based testing platform for desktop browser testing and mobile app testing. For desktop browser testing, it offers a Selenium Grid a.k.a pCloudy Browser Cloud where we can run our tests parallelly on different environments i.e, different browsers, multiple browser versions and different operating systems. Along with cloud based automation testing, it also offers manual testing on different cloud based real Mac and Windows machines where the user can naturally interact with different combinations of OS and browsers to test an application with a human feel.

Running automated tests on the pCloudy Selenium Grid provides us with various web app performance statistics, detailed logs, screenshots and videos. To perform cross browser testing on different browsers and OS, we can run our test parallelly on pCloudy browser cloud which would not only save the execution time but also help us compare the performance statistics of our application running on different environments.

Considering cross browser testing on Safari, let's develop a script to run our single test case parallelly on different safari versions.

```
import java.io.File;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;

import org.apache.commons.io.FileUtils;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class Blog {
    public String username = "ramit.dhamija@gmail.com";
    public String accesskey = "5TfF4UcNRbN3JhucQ";
    public static RemoteWebDriver driver = null;
    public String gridURL = "https://prod-browsercloud-in.pcloudy.com/seleniumcloud/wd/hub";
    public String expectedTitle = "Remote Mobile Web & Application Testing on Real Android Devices - pCloudy";

    @BeforeClass
    @Parameters(value = { "OS", "OSversion", "browser", "browserVersion" })
    public void setUp(String OS, String OSversion, String browser, String browserVersion) throws Exception {
        DesiredCapabilities capabilities = new DesiredCapabilities();
        capabilities.setCapability("pCloudy_Username", username);
        capabilities.setCapability("apiKey", accesskey);
        capabilities.setCapability("clientName", username);
        capabilities.setCapability("email", username);

        capabilities.setCapability("os", OS);
        capabilities.setCapability("osVersion", OSversion);
        capabilities.setCapability("browserName", browser);
        capabilities.setCapability("browserVersions", browserVersion);

        capabilities.setCapability("pCloudy_EnableVideo", "true");
        capabilities.setCapability("pCloudy_EnablePerformanceData", "true");
        capabilities.setCapability("pCloudy_EnableDeviceLogs", "true");

        try {
```

```

        try {
            driver = new RemoteWebDriver(new URL(gridURL), capabilities);
        } catch (MalformedURLException e) {
            System.out.println("Invalid grid URL");
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    @Test
    public void testSimple() throws Exception {
        try {
            driver.get("https://www.pcloudy.com/");

            WebElement loginButton = driver.findElement(By.xpath("//a[text()='Login']"));
            loginButton.click();

            String actualTitle = driver.getTitle();

            Assert.assertEquals(actualTitle, expectedTitle, "Login Page Title didn't matched with the expected title");
            System.out.println("Login Title matched");

            File screenshot = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

            try {
                FileUtils.copyFile(screenshot, new File("/home/ramit/Documents/loginPageScreenshot.png"));
            } catch (IOException e) {
                System.out.println(e.getMessage());
            }

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    @AfterClass
    public void tearDown() throws Exception {
        if (driver != null) {
            driver.quit();
        }
    }
}

```

#### TestNG.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://beust.com/testng/testng-1.0.dtd" >
<suite thread-count="2" name="cross browser test-safari"
        parallel="tests">

    <test name="Catalina-Safari-14">
        <parameter name="OS" value="Mac" />
        <parameter name="OSversion" value="Catalina" />
        <parameter name="browser" value="safari" />
        <parameter name="browserVersion" value="14" />

        <classes>
            <class name="main.java.src.Blog" />
        </classes>
    </test>

    <test name="Catalina-Mojave-14">
        <parameter name="OS" value="Mac" />
        <parameter name="OSversion" value="Mojave" />
        <parameter name="browser" value="safari" />
        <parameter name="browserVersion" value="13" />

        <classes>
            <class name="main.java.src.Blog" />
        </classes>
    </test>
</suite>

```

#### Code Walkthrough:

With the above script, we are running our parallel cross browser tests on two different environments. In @BeforeClass annotated method, we have added username and accessKey capability for authorization. As a part of capabilities, we have further defined our testing environment, we have used @Parameter annotation that would grab the testing environment values from TestNG.xml. To capture screenshots and logs of our automated tests, we have used the screenshot and logs capabilities provided by pCloudy.

In a @Test annotated method, we have written a short test case to verify the visibility of the pCloudy login page. Once the test case gets executed, the @AfterMethod annotated method quits the online safari browser testing.

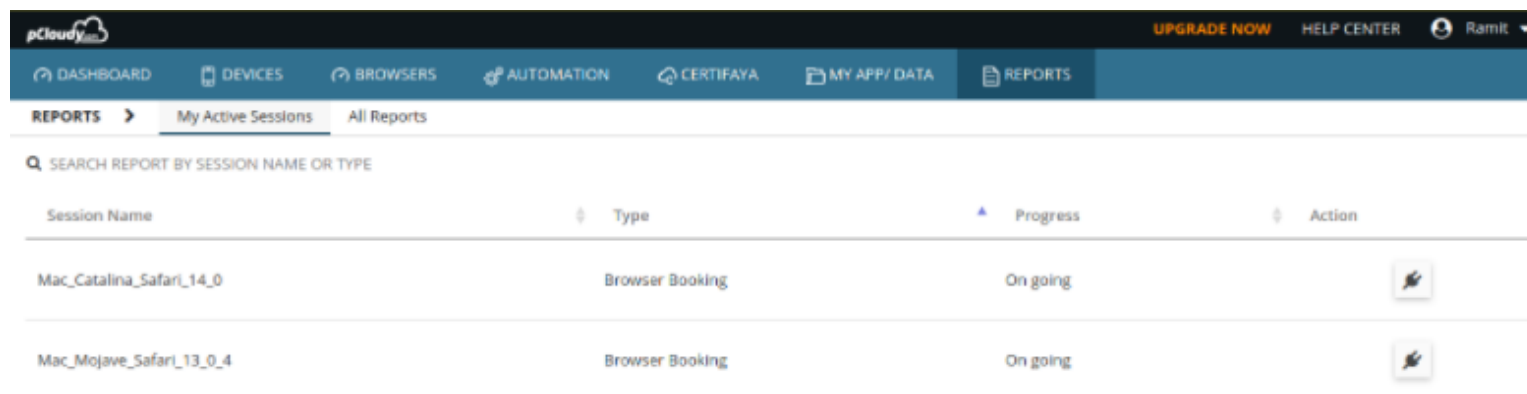
The TestNG.xml file is created to configure the execution of our test suite with different suite attributes and with a different testing environment. Since we are running 2 tests parallelly, we have passed thread-count as 2 and parallel as “tests” in suite tag. In each test tag, we have defined the parameter values that would be supplied to our java test class to run our test case in the required environment.

Using TestNG.xml, we are running single test case parallelly on below two environments:

- Mac-Catalina-Safari-14
- Mac-Mojave-Safari-13

### pCloudy Output :

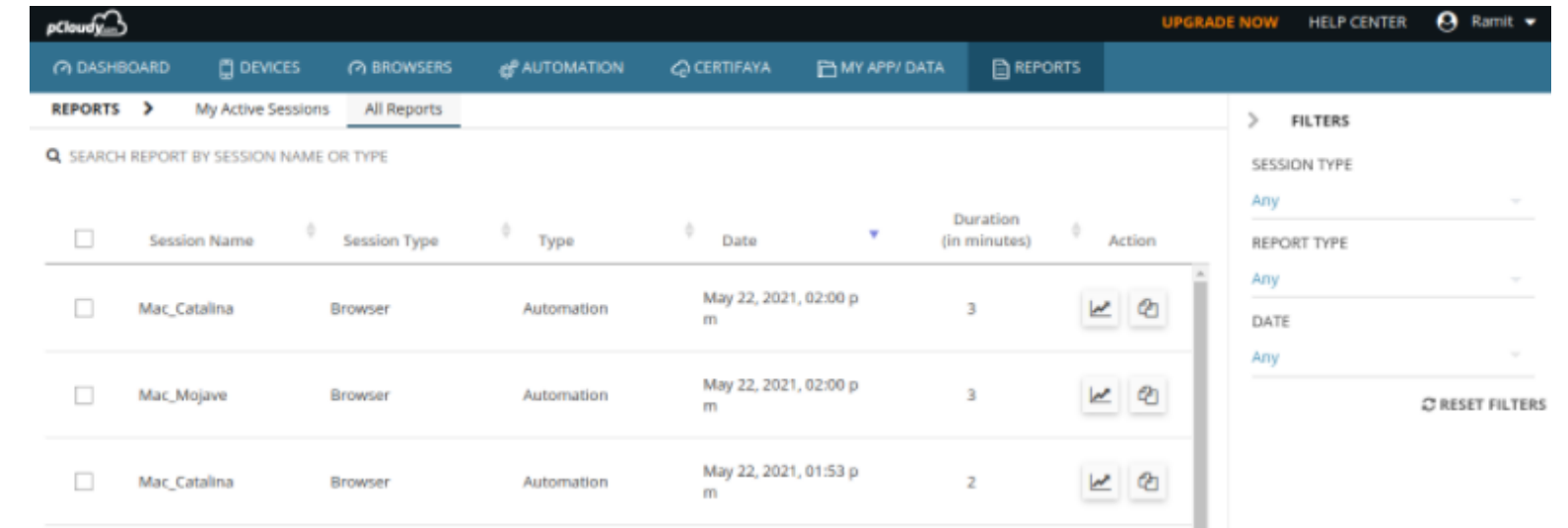
To view the current running status of your tests, direct to [pCloudy device page](#) and click on Reports-> My Active Sessions:



The screenshot shows the pCloudy web interface with the 'REPORTS' tab selected. Under 'My Active Sessions', there is a table with the following data:

Session Name	Type	Progress	Action
Mac_Catalina_Safari_14_0	Browser Booking	On going	[Icon]
Mac_Mojave_Safari_13_0_4	Browser Booking	On going	[Icon]

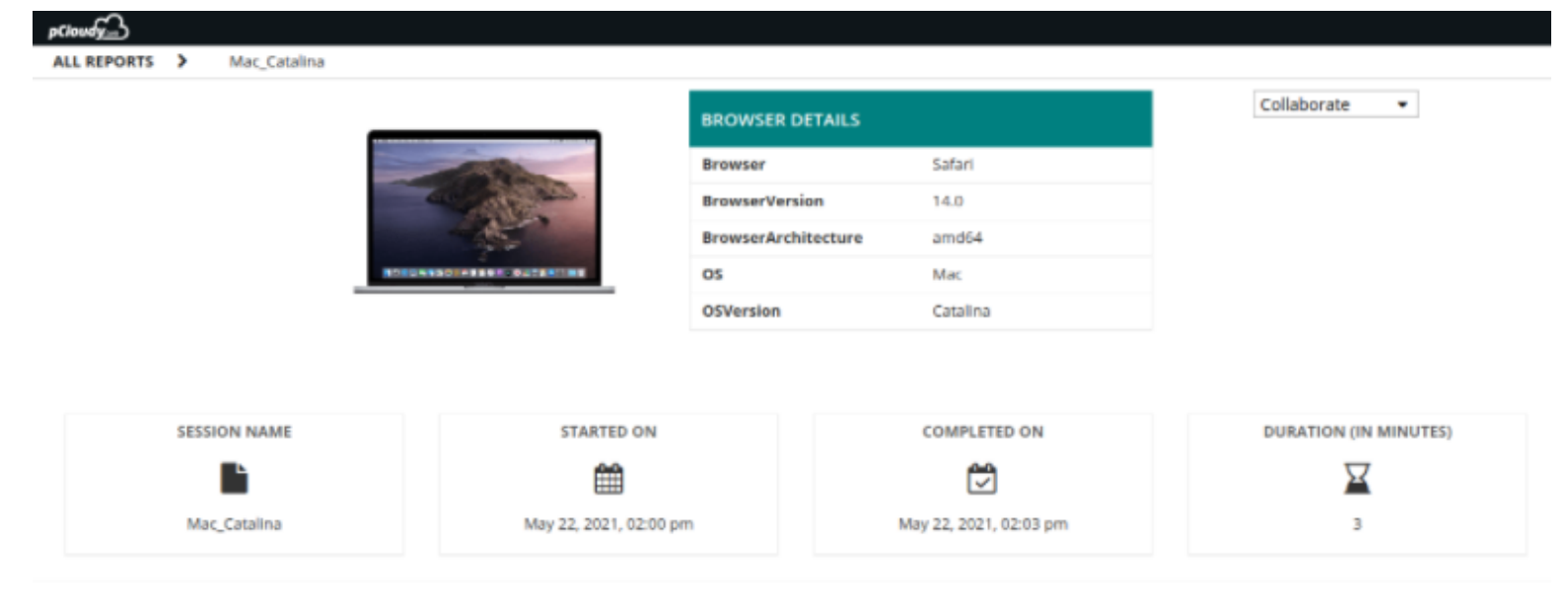
Direct to the [pCloudy device page](#) and click on Reports-> All Reports, to view the all the automated tests executed till date.



The screenshot shows the pCloudy web interface with the 'REPORTS' tab selected. Under 'All Reports', there is a table with the following data:

Session Name	Session Type	Type	Date	Duration (in minutes)	Action
Mac_Catalina	Browser	Automation	May 22, 2021, 02:00 p m	3	[Icon] [Icon]
Mac_Mojave	Browser	Automation	May 22, 2021, 02:00 p m	3	[Icon] [Icon]
Mac_Catalina	Browser	Automation	May 22, 2021, 01:53 p m	2	[Icon] [Icon]

To view the logs and snapshots of a specific test, direct to [pCloudy device page](#) and click on Reports-> All Reports and take a action on particular session name for which logs and screenshots are required:



The screenshot shows the pCloudy web interface with the 'ALL REPORTS' tab selected. The page displays details for a specific session named 'Mac\_Catalina'.

BROWSER DETAILS	
Browser	Safari
BrowserVersion	14.0
BrowserArchitecture	amd64
OS	Mac
OSVersion	Catalina

SESSION NAME	STARTED ON	COMPLETED ON	DURATION (IN MINUTES)
Mac_Catalina	May 22, 2021, 02:00 pm	May 22, 2021, 02:03 pm	3

## CONTACT US

Take a leap with



A comprehensive solution to increase the speed of your App Testing  
by enabling Continuous Testing



Device Lab



Browser Lab



Rapid Automation



Automation execution  
@ Scale



Test Analytics



DevOps