Tutorials

for Pricing
Teams

Home Guide Front End Testing: A Beginner's Guide

Front End Testing: A Beginner's Guide

By Shreya Bose, Community Contributor - March 16, 2023

First, the front end is any part of a software application that a user interacts with. Essentially, it is all aspects of software visible to clients. This usually includes GUI elements such as menus, buttons, and forms – anything end-users can see when navigating the application. It can also include aspects like page load speed – factors contributing to overall user experience.

Table of Contents

Guide Categories GET A DEMO FREE TRIAL

na[illegible]igation is easy enough if the page is loading fast enough, etc.

FrontEnd Testing aims to test functionalities and verify that a website or app’s presentation layer is bug or error-free. This must be done after every update to ensure recent changes have not degraded any UI aspect.

Difference between Front End Testing and Back End Testing

- **Front End Testing** verifies the user-facing interface of a software application. It requires knowledge about requirements related to user experience but not about the database and the back-end mechanics.
- **Back End Testing** checks the efficacy of the functionality on the server and database side of the software. This is necessary to ensure that interdependent components interact accurately to efficiently produce, process, and store data. It requires knowledge of databases such as SQL Server, MySQL, Oracle, DB2, etc.

Why do you need Front End Web Testing?

Identifying Performance Issues

Web applications possess three functional layers – clients, servers, and resources (or information systems). Front End Testing handles the client layer, AKA the part of software presented to the client.

It isn’t uncommon for devs to focus on the server and resource layers, as they are the foundation on which the software is built. However, Front End testing is essential for QA teams to analyze software behavior from end users’ perspectives. Detecting client-side performance issues that may sabotage critical workflows and

- A website or app must be tested on multiple browser-device-OS combinations to ensure it behaves as expected on each one.

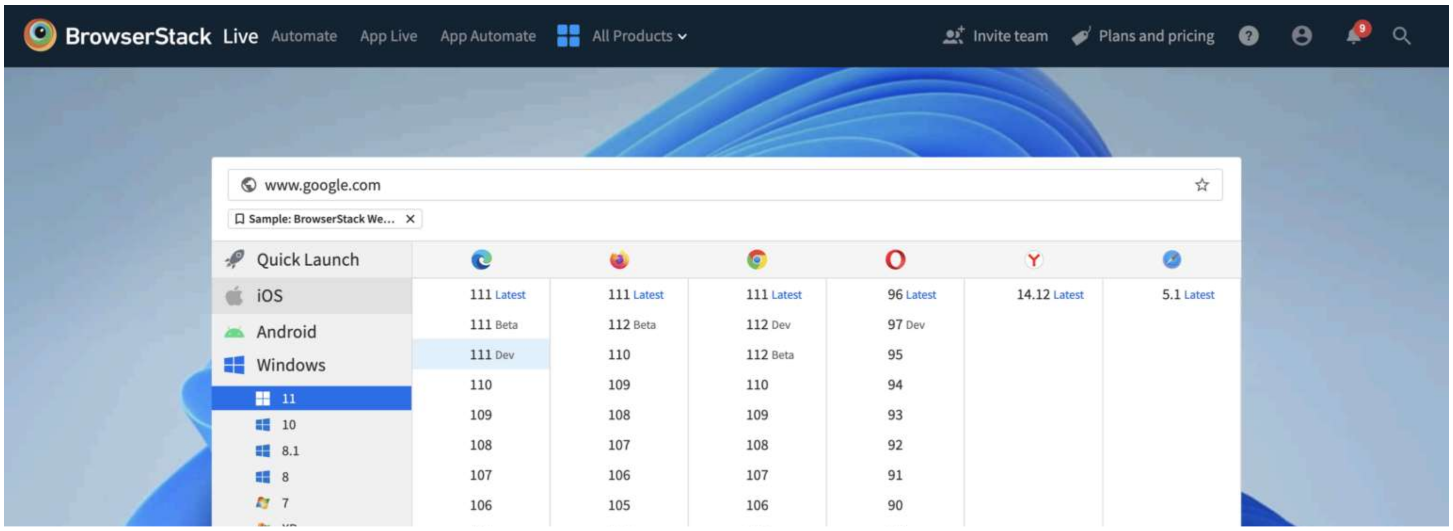
- The only way to ensure this is to perform comprehensive cross browser testing across real browsers and devices.

- Testers need to check how the software renders and operates in real user conditions, for which they need to test on multiple unique browser-device-OS combinations.

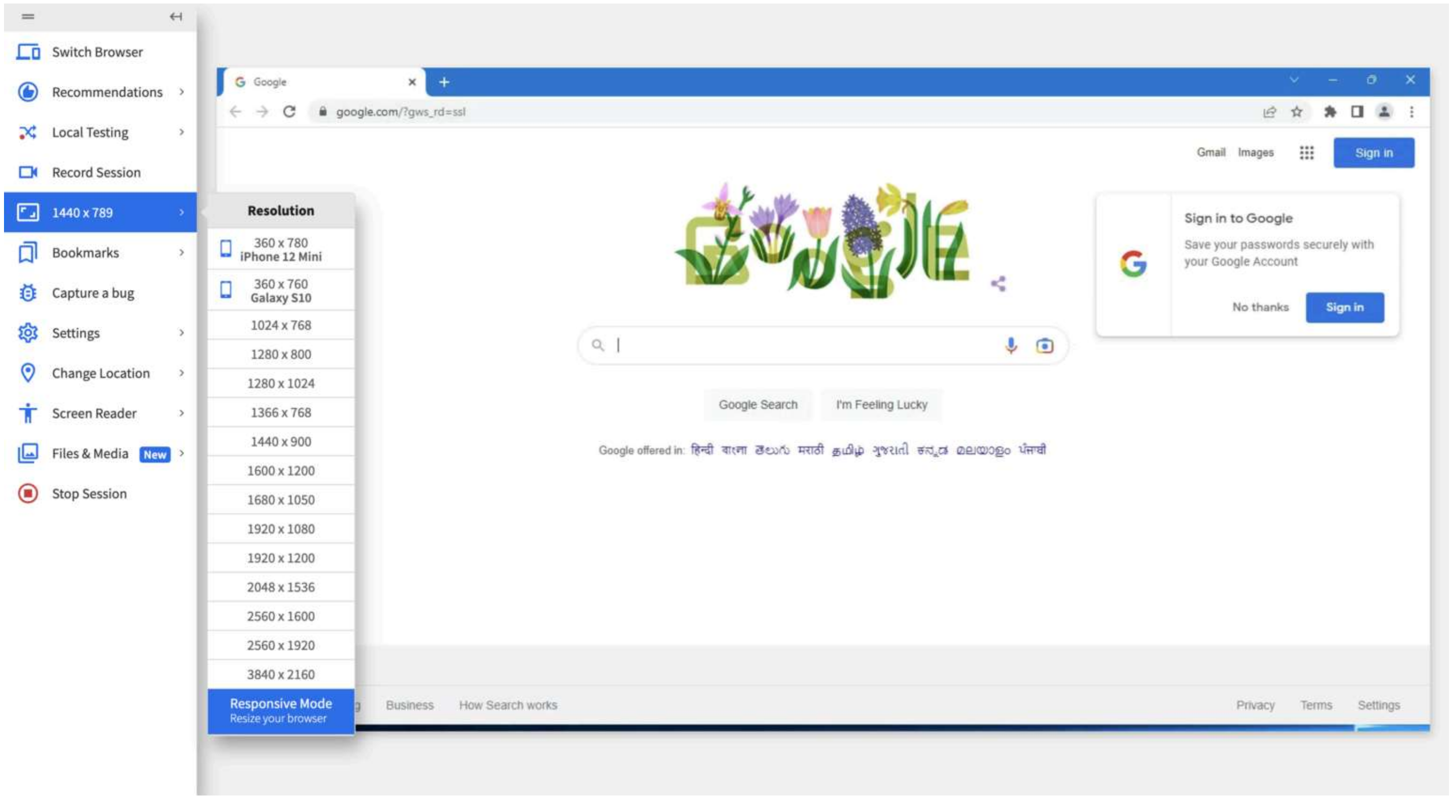
- QAs must run Front End tests on real browsers, devices, and operating systems.

- Considering their technical variances and quirks, the software must render perfectly on each device, browser, and browser version. Given that at least 63,000 possible browser-platform-device combinations are in popular usage, QA teams need access to a massive on-premise device lab (constantly updated with newer devices) to perform satisfactory cross browser compatibility testing.

Not every organization has the finances or the human resources to set up and maintain such a lab, and they don't have to. They can use BrowserStack to access a device cloud of 3000+ browsers and devices.



111 Latest	111 Latest	111 Latest	96 Latest	14.12 Latest	5.1 Latest
111 Beta	112 Beta	112 Dev	97 Dev		
111 Dev	110	112 Beta	95		
110	109	110	94		
109	108	109	93		
108	107	108	92		
107	106	107	91		
106	105	106	90		



Access BrowserStack Device Cloud

Verifying integration of Third-Party Services

With SaaS platforms becoming commonplace, most applications leverage third-party integrations to offer better services and heightened user experiences. However, faulty integrations are often a significant cause of software malfunction or, at least, unsatisfactory user journeys.

To prevent some damaged user experiences, Front End tests are mandatory. All third-party integrations must be tested (across browsers, devices, and platforms) to ensure they function seamlessly and within accepted

For example, in C#, consider a method as a unit (most minor component to be tested). In this case, the unit test would verify some features of the way in isolation from other forms i.e. the software at large. Unit Tests are usually categorized as state-based and interaction-based testing. The former checks whether the software produces expected results under specific conditions. The latter verifies if the software appropriately calls particular methods to accomplish its purpose.

2. Visual Regression Testing

Regression testing verifies that system changes do not interfere with existing features and/or code structure. There's a reason regression tests are part of almost every test suite in software development. It is common for devs to change or add a section of code and have it unintentionally disrupt something previously working just fine.

Visual Regression Testing applies the same logic but confines testing to the visual aspects of the software. In other words, it checks that code changes do not break any aspect of the software's visual interface. A visual regression test checks what the user will see after any code changes have been executed by comparing screenshots taken before and after code changes. This is why visual regression tests are also sometimes called visual snapshot tests.

3. Cross Browser Testing

As explained above, cross browser testing allows testers to check if a website works as expected when accessed via different browser-device-OS combinations. This applies to different versions of the same browser and assistive tools.

Apps must be tested on different device-OS combinations. Among other things, trying responsive design is a crucial aspect of cross-browser and cross-device testing. Testers can use BrowserStack's free responsive design checker to check their software's appearance on the latest real devices.

4. Integration Testing

Also Read: Unit Test vs Integration Test

5. Accessibility Testing

Accessibility Testing is a software testing technique that checks if every user on the internet can easily access a website or app, including individuals with disabilities or special needs. Often considered a sub-category of usability testing, it ensures that specific, unchangeable conditions do not prevent a person from accessing online resources as quickly as anyone else. You can run both automated and manual accessibility testing on BrowserStack. To run manual tests, look at how to use Screen Readers for Accessibility Testing.

Note: All accessibility testing on the BrowserStack real device cloud is executed on real browsers and devices—3000+ real browsers and devices.

6. Acceptance Testing

Acceptance testing determines if a software system meets all predetermined specifications. It evaluates if the system complies with business, technical, and aesthetic requirements so that business stakeholders and end-users can be satisfied alike.

Acceptance tests are generally divided into – **User Acceptance Testing** and **Business Acceptance Testing**. The former checks to ensure that the product meets all performance standards from the users' perspective, while the latter establishes that the software aligns with business goals and requirements.

Challenges of Automated FrontEnd Testing

- **Consistently Evolving UI:** In modern software, core libraries and third-party components must be visual appeal (has the new button covered up the existing menu?) Essentially, the developers' and testers' work on a specific software never ends.

- **Choosing the right Automation Tool:** Effecting, periodic front-end testing requires automation. Manual testers cannot be expected to run tests every time an upgrade is pushed. However, choosing an automation tool that can be effectively set up and empowered with test scripts to run requisite checks and verifications. However, with the plethora of automation testing tools available, it can be somewhat challenging to select what would work best for your team, given their skill sets and project requirements. However, Selenium Webdriver, Cypress, TestCafe, and Playwright are some preferred frameworks for front-end testing.

- **Detecting cross-browser and cross device issues:** With thousands of browser versions and devices used worldwide to access the internet, testers must cover a massive range to equip a site or app for real-world usage. This can be challenging since new devices and browser versions are constantly released. To keep up, teams need access to real browsers and devices. An in-house device lab takes significant financial and human resources to set up, maintain and upgrade. Using test infra hosted on the cloud is more accessible, as in BrowserStack.

Front End Testing Best Practices

- **Start with the testing pyramid:** For testers and teams just starting with Front End testing, it's a good idea to use it as a blueprint. That means: starting running unit tests, then moving on to integration testing, and finally, executing end-to-end testing. Once this structure is in place, achieving reasonably high test coverage will be relatively easy.

  Dive deeper into the testing pipeline, and add more tests once the testing pyramid comes into action. Once unit tests, integration tests, and end-to-end tests are completed, it will be easier to expand the testing scope and include things like acceptance testing, visual testing, etc.

- **Choose the right Front End testing tools:** Outside of providing real browsers and devices for test execution (and this is a mandatory feature), the ideal Front End testing tool should offer ways to make the process as seamless as possible.

  BrowserStack, for example, offers a range of debugging options, and pre-installed developer tools that testers can quickly access to identify and resolve bugs. It also offers integrations with a range of necessary tools spanning automation frameworks, CI/CD, build and playback, record and deploy, and much more.

  In other words, find a tool that provides comprehensive resources covering every step of the testing pipeline. The tool should be able to let QAs run tests, identify bugs, file and forward bugs to other team members, debug, and deploy.

Front End Testing ensures that users get the best possible experience when they access a website or app. It is a key part of any testing pipeline. Consequently, testers need to be meticulous in their planning, execution, and implementation of Front End testing.

Testing Tools Types of Testing

Was this post useful? Yes, Thanks Not Really

Front-End Testing Strategy: A Detailed Guide

This guide will walk you through the common principles on which a sound front-end test strategy can ...

Learn More

What is End To End Testing?

Tutorials

for Teams Pricing

Cypress End to End Testing: Tutorial

Testing End to End user flows enhances user experience. Check out the how to perform End to End test...

Learn More

Ready to try BrowserStack?

Over 6 million developers and 50,000 teams test on BrowserStack. Join them.

Guide Categories GET A DEMO FREE TRIAL

Tutorials

for Teams

Pricing

PRODUCTS

Live
Automate
Automate TurboScale Beta
Percy
App Live
App Automate
App Percy New
Test Management New
Test Observability New
Accessibility Testing New
Accessibility Automation New
App Accessibility Testing Beta
Low Code Automation Beta
Nightwatch.js
Enterprise

TOOLS

SpeedLab
Screenshots
Responsive

PLATFORM

Browsers & Devices
Data Centers
Real Device Features
Security

SOLUTIONS

Test on iPhone
Test on iPad
Test on Galaxy
Test In IE
Android Testing
iOS Testing
Cross Browser Testing
Emulators & Simulators
Selenium
Cypress
Android Emulators

RESOURCES

Test on Right Devices
Support
Status
Release Notes
Case Studies
Blog
Events
Test University Beta
Champions
Mobile Emulators
Guide

COMPANY

About Us
Customers
Careers We're hiring!
Open Source
Partners
Press

Guide Categories GET A DEMO FREE TRIAL

By clicking 'Accept All', you agree to the use of all types of cookies to enhance site navigation. For more information, please visit our cookie policy.

Tutorials

for Pricing
Teams