

A brief analysis of BrowserStack's documentation

Souvik Sarkar · 2022-10-27

Based on my 30-minute analysis of BrowserStack's documentation and comparing with a competitor's documentation, I don't find anything that I strongly dislike. However, as a technical writer trained to pay attention to details, I have the following nitpicks:

1 Principles of minimalism are not followed

For example, in Getting started - Python - Prerequisites, the first bullet point is verbose.

Existing sentence:

```
You need to have BrowserStack Username and Access key, which you can find in your account settings. If you have not created an account yet, you can sign up for a Free Trial or purchase a plan.
```

Suggested sentence:

```
Note your BrowserStack Username and Access key from the account settings. Unregistered users can sign up for a trial account or purchase a plan.
```

The suggestion also demonstrates restructuring of sentences to emphasize user's actions, and brings more clarity.

2 Section headings are not in gerund form

For example, in Getting started - Python - Run your first test:

Existing heading:

```
Run your first test
```

Suggested heading:

```
Running your first test
```

Gerund forms of phrases are appropriate for highlighting a process, which might contain several steps. For the steps, we can use action oriented sentences in simple present tense, such as "Create a sample test code file".

3 Admonitions are misused

For example:

- The following Warning should ideally be a Caution, because the user needs to be mindful about adding the `driver.quit()` statement at the end of the code.
- Protip is not a standard admonition. Instead, use Tip.

Warning: The `driver.quit()` statement is required, otherwise the test continues to execute, leading to a timeout.

Figure 1: Warning admonition example

Protip: You can use our [capability builder](#) and select from a wide range of custom capabilities that BrowserStack supports.

Figure 2: Protip admonition example

4 Some unnecessary images

Images could have been used more sparingly to avoid maintenance problems. This is even more important for images that represent products such as GitHub, which are beyond BrowserStack's control.

In addition, images without appropriate `alt-text` hamper accessibility and SEO.

5 Doc site UI/UX can be improved

Modern documentation websites adopt a three-column approach: documentation suite navigation on the left, content in the middle, and current page ToC on the right. BrowserStack's documentation website mostly leaves the right column blank. As a result, the left navigation panel looks more crowded and needs frequent scrolling.

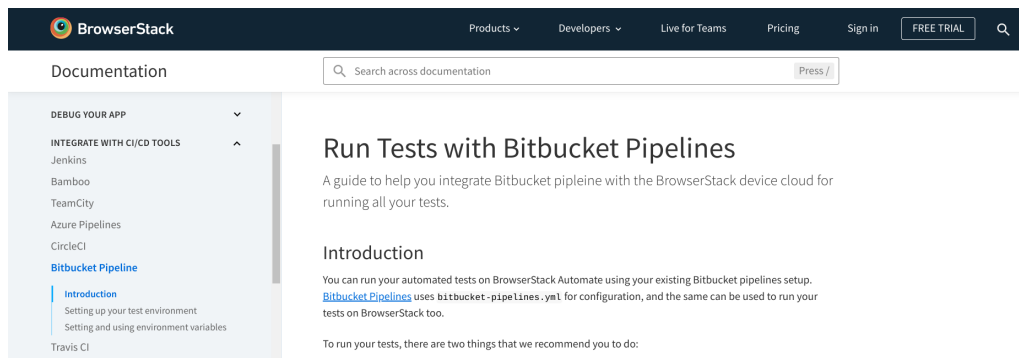


Figure 3: BrowserStack documentation site

6 General comment

- Some navigation links within the text/content are broken, especially the ones that navigate to different sections within the same article.
- In lots of occasions, the writing style deviates significantly from Google's documentation style guide. In fact, a seasoned writer can easily identify the documents contributed by developers, testers, product managers, or customer support. While this is not a big problem, the documentation experience can certainly be made more consistent.

7 Bonus

Modern test environments, especially in enterprise situations, are mostly containers running on clusters. If the onboarding docs describe procedures using containers, the onboarding experience and the documentation becomes more standardized and maintainable.

The screenshot shows the LambdaTest documentation page for a tutorial on running a first Selenium test. The page layout includes a top navigation bar with the LambdaTest logo and links for Docs, API Reference, FAQ, GitHub, and TestU Conf. A search bar is located in the top right corner. On the left side, there is a sidebar menu with categories like Selenium Testing, Languages and Frameworks, and Python. The main content area features the article title, a brief introduction, an objective section with a list of four goals, and a note about code samples. A right-hand sidebar contains a table of contents for the article.

LAMBDATEST Docs API Reference FAQ GitHub [TestU Conf](#)

Search

Introduction
 Selenium Testing
 Languages and Frameworks
 Java
 JavaScript
 CSharp
 Python
 unittest
 pytest
 Robot
 Behave
 Lettuce
 PHP
 Ruby
 Tesbo Framework
 Mobile Web Automation On Real Devices

Python with Selenium: Tutorial to Run Your First Test on LambdaTest

This post will help you in getting started with configuring and running your Python-based automation test scripts on [LambdaTest Selenium cloud platform](#).

Objective

By the end of this topic, you will be able to:

1. Set up an environment for testing your hosted web pages using **Python** with Selenium.
2. Understand and configure the core capabilities required for your Selenium test suite.
3. Test your locally hosted pages on LambdaTest platform.
4. Explore advanced features of LambdaTest.

Note: All the code samples in this documentation can be found in the [LambdaTest's Repository on GitHub](#). You can either download or clone the repository to quickly run your tests.

Objective
 Prerequisites For Running Selenium Python Scripts
 Installing Selenium Dependencies and Tutorial Repo
 Setting up Your Authentication
 Run Your First Test
 Sample Test Case
 Configuration of Your Test Capabilities
 Executing the Test
 Testing Locally Hosted or Privately Hosted Projects
 Additional Links

Figure 4: LambdaTest documentation site