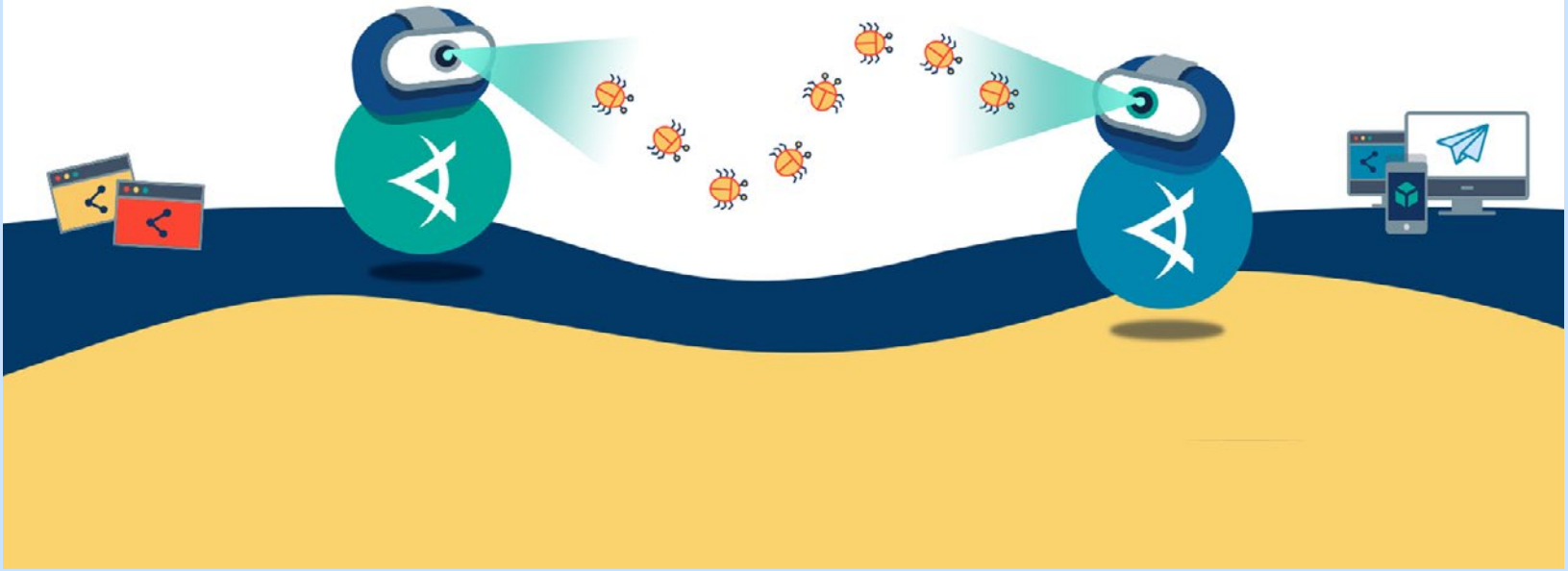
applitools

AI-Powered Visual Testing & Monitoring

Automated Visual Testing Best Practices Guide

By Angie Jones



Automated Visual Testing Best Practices Guide

Many of the top high-performing companies in software, financial services, and healthcare verticals are using Applitools Eyes for their visual test automation needs. We've talked with these companies to gain a better understanding of how they are using the product and to identify common practices that are working well for teams. This guide provides a set of best practices and our recommendations for implementing and maintaining automated visual tests with Applitools based on our customers' experiences.

Summary of Recommendations

Here is a summary of our top recommendations contained within this guide. Details on each of these, including when to detour from the recommendation can be found within the following sections of the guide.

1. **Use Eyes for both visual and functional assertions**
   The Applitools Eyes API should be used to complement or encompass functional assertions with less lines of code and more test coverage. **READ MORE**

2. **Verify the entire page**
   While there are various options to visually verify your application with Applitools Eyes, we recommend verifying the entire page of your application to obtain the most coverage from your visual testing. **READ MORE**

3. **Use the Strict match level**
   By default, Applitools Eyes uses the Strict match level algorithm (our recommended comparison method), which employs AI to compare your baseline image with the current image from a regression run. **READ MORE**

4. **Use Ignore annotation for data that is not relevant to the test**
   We recommend using our Ignore annotation to ignore parts of your application's page that are not pertinent to the test. **READ MORE**

5. **Programmatically apply annotations when possible**
   When annotations are needed, use fluent checks to programmatically apply them to baseline images before the test is executed. **READ MORE**

6. **Use steps to include multiple visual checkpoints in a single test**
   It's recommended to perform multiple visual checks where there are multiple visual states that need to be verified in a given scenario. **READ MORE**

7. **Group related tests into batches to enable easier management as well as automated maintenance of baselines**
   For related tests, it's recommended to use batches so that the tests display together on the Applitools Dashboard within an aggregated view. **READ MORE**

8. **Utilize a wrapper class to encapsulate visual checks**
   It's a good practice to create a wrapper class that isolates the Eyes API calls so that if there are any changes to the API, the required changes to the test code will be limited to an individual class. **READ MORE**

9. **Use Layout mode for early unverified localization testing**
   Before your localized app has been verified by a specialist, use Layout mode. After you're sure the content is correct, use Strict mode. **READ MORE**

10. **Tag visual tests and execute them any time the UI is under test**
    Utilize tagging capabilities of your test runner to easily include and exclude visual tests when executing. **READ MORE**

11. **Use the Ultrafast Grid to run cross-platform tests**
    Applitools' Ultrafast Grid enables you to execute your tests across many different browsers, viewports, and devices at a fraction of the time it takes with other solutions. **READ MORE**

12. **Gate your builds with relevant visual checks**
    With the accuracy of Eyes API, we feel confident recommending that you have your visual tests follow the same gating strategy used for your functional tests. **READ MORE**

13. **Visual checks can be used when using feature files**
    When practicing BDD, continue to write feature files without implementation detail and use the step definition code to execute visual checks where necessary. **READ MORE**

14. **Use the Applitools dashboard to collaborate**
    The Applitools Dashboard should be used to annotate screenshots of failures with bug regions, assign failures to developers to review, and remark on questionable changes. **READ MORE**

15. **Integrate Applitools with Slack for faster collaboration on failed visual tests**
    If your team already uses Slack for collaboration, we recommend integrating Applitools with Slack, which makes sharing and resolving failures easier. **READ MORE**

What Is Visual Testing

Automated visual testing is a technique to verify that your application appears to the user as you intended. The Applitools Eyes API can be used to complement or **encompass functional assertions** with less lines of code and more test coverage.

Applitools Eyes is both flexible and powerful, and makes a wonderful addition to any UI or mobile test automation toolkit.

Visual testing can be used whenever there's a need to verify the display of data on a web or mobile application.

Functional assertion libraries verify information in the **DOM**, but are incapable of verifying how that data is actually presented to the user.

For example, here's a visual bug on Instagram. Functional assertions that check if the text exists would pass because yes, it does indeed exist in the DOM. However, it misses the fact that all of the text is overlapping and therefore is unreadable by the user.

To avoid missing such bugs, Applitools should be used in your mobile and web UI tests.