



New Deterministic Agentic Workflow with AppliTools Visual AI MCP Server

Deploy AI-generated code with confidence

Adam Carmi – Co-Founder & CTO

Deterministic Guardrails for Agentic SDLC

- **Plain-English test steps for open-source testing frameworks**

Deterministic, LLM-free execution. No tokens. Less code. No locators.

- **Native Figma design baselines**

Streamline the design to dev handoff workflow. Detect design drifts using your regular regression tests

- **Giving AI Agents Eyes with Applitools MCP**

Allow coding agents to self-correct before declaring “done”. Let agents maintain baselines for you

Why deterministic guardrails?

Software development is changing fast

CODE AUTO COMPLETE ➡ AI ASSISTED DEVELOPMENT ➡ AI AGENTS ➡ AUTONOMOUS DEVELOPMENT?



We are here!

Human review is mandatory

The new bottleneck is **TRUST**

AI agents generate changes faster than
humans can review and verify them

The new bottleneck is **TRUST**

FOR DEVELOPERS

Endless code reviews and context switches

Cognitive load and burnout reduces the code review quality

Quality gates are skipped, and test code is mostly accepted unreviewed

FOR TESTERS

The rate and magnitude of (poorly tested) changes drastically increases

Impossible to narrow the scope of testing

More (AI generated) test code means more flakiness and longer test cycles

Independent studies confirm it*

**REVIEW TIME
INCREASED**

5x

Median PR review time

**QUALITY GATES
BYPASSED**

+242%

Incidents per PR

**PROD QUALITY
DECREASED**

+54%

Bugs per developer

* 22,000 developers studied (FarosAI AI Engineering Report, Mar 2026)

Deterministic Guardrails **break the trust bottleneck**

FASTER REVIEWS



Reduce the volume and complexity of changes

FEWER CYCLES



Help agents generate better code and catch issues earlier

FEWER BUGS



Increase test coverage

If we ask AI to **generate more playwright tests**,
will it solve the problem?

Functional testing alone **isn't enough**

What we need

Less code

More automated feedback

More coverage

Functional E2E testing

Bloated test suites, many textual assertions

Limited to expected functional correctness

Doesn't cover the appearance of the UI

THE DEMO APP

Plain-English test steps for open-source testing frameworks

Deterministic English test step execution

- **Author tests in plain English**
Easiest to author, review, understand and maintain. LLM friendly
- **Deterministic execution**
Consistent, fast and hallucination-free execution using a purpose-built DLM
- **No DOM locators required**
No locator authoring and tuning required. UI implementation changes no longer break tests
- **Less code, less complexity**
Implicit waits. Automatic tab and iframe switching. Custom control detection
- **Test the most dynamic apps**
Page stabilization. Find elements by text format, relative position, color, etc.

```
// Get the iframe that contains the dropdown
const iframe = page.frameLocator('iframe[name="iframeResult"]');

// Click the dropdown button to open it
await iframe.getByRole('button', { name: 'Dropdown' }).click();

// Wait for the dropdown menu to be visible
await iframe.locator('.dropdown-content').waitFor({ state: 'visible' });

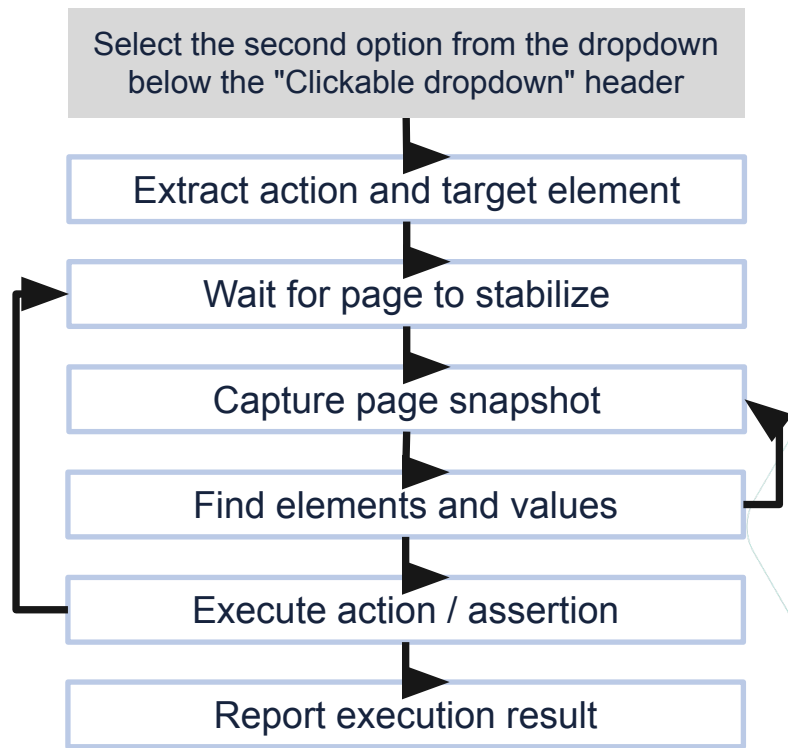
// Get all dropdown options (links)
const dropdownOptions = iframe.locator('.dropdown-content a');

// Click the 2nd option (index 1, since it's zero-based)
await dropdownOptions.nth(1).click();
```

VS

```
await eyes.run("Select the second option from the dropdown");
```

The Deterministic Language Model stack



- A complete, 100% deterministic test automation stack
- English is the source of truth. No Playwright / Selenium test code generated under the hood
- Supports web apps and (coming soon) mobile apps
- Supports codeless (Autonomous) and coded tests (vis SDKs)

DEMO

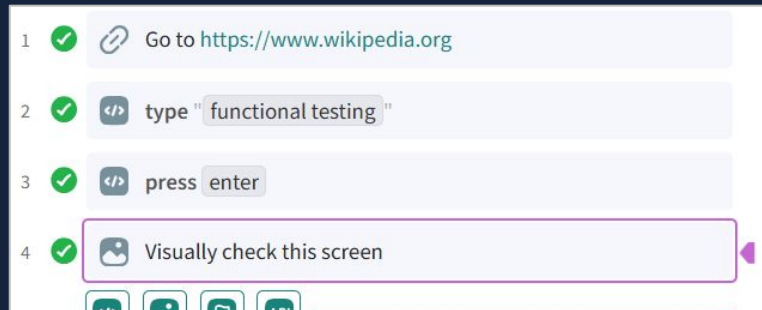
Introduction to **Visual Testing** (before we jump into native Figma baselines)

Visual testing focuses on **verifying the correctness of the UI**

Visual assertions powered by VisualAI

- Replace existing traditional textual assertions in your existing coded tests via an SDK (or codeless)
- Full page screenshots are captured and compared with pre-approved “baseline” images
- Assertions fail if differences are found along with links to a dashboard where findings can be reviewed and resolved

Codeless

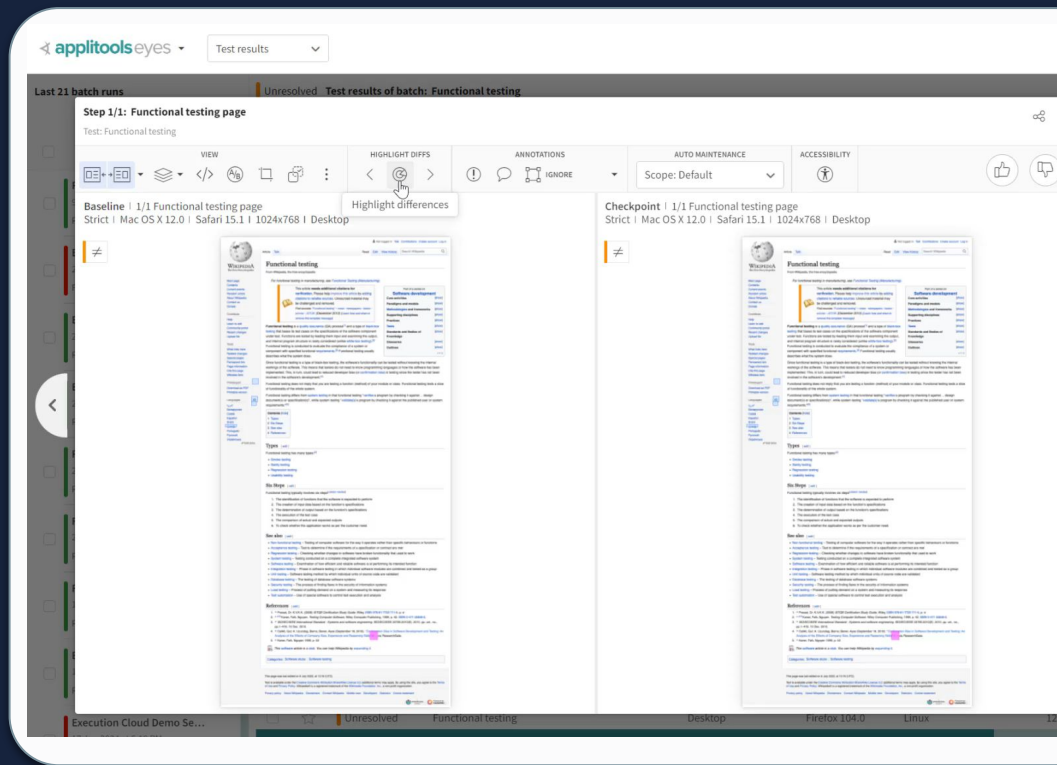


Codefull

```
test('Functional testing', async ({ page, eyes }) => {  
  await page.goto('https://wikipedia.org/');  
  
  const searchInput = await page.locator('#searchInput');  
  await searchInput.fill('Functional testing');  
  await searchInput.press('Enter');  
  
  await eyes.check('Functional testing article');  
});
```

Visual assertions powered by VisualAI

- Complete visual and functional coverage of an entire page or region
- Across any device, browser, form factor and operating system
- Reduces the amount of test code by up to 80%
- Catches unexpected defects
- No test code changes required following UI changes



DEMO

Native Figma design baselines

Streamline the design to dev handoff workflow

- **Avoid tedious manual implementation reviews**

Designer shares a link; developer writes a standard visual test mapped to it. UI viewport automatically adjusts

- **Tracks design and approver details**

No more arguing whether the implementation was approved, by whom, and by which version of the design

- **Automatically transitions to regression mode following approval**

Low fidelity and mockup design friendly

- **Automatically re-compares with the design if changed post approval**

Never miss a post-approval design change

DEMO

Appltools MCP Server

Most AI coding systems are **blind to the UI**

Even when AI writes the code, it **cannot** answer the following questions:

- Did I break the UI?
- Does this change look correct?
- Will it work across different devices and browsers?
- Did I unintentionally affect another part of the application?

Why **LLMs can't** reliably analyze UI screenshots

- **Inconsistency and guesswork**

Due to probabilistic token selection. Dismisses real issues as noise or as “expected”. Worsens at scale

- **Lacking coverage, context and precision**

Coarse image analysis. Insufficient baseline context to make decisions (e.g., dom, history, dynamic areas)

- **Speed and cost**

Multi-second “thinking cycles” and massive page snapshots. Minutes turn to hours. High execution cost.

- **Security and privacy concerns**

Sensitive or private information included in page snapshots may violate regulations

Applitools MCP: **Giving AI Agents Eyes**

- **Simplify setup and onboarding**
 - Setup and configuration. Converting functional tests to visual tests
- **Eliminate baseline maintenance overhead**
 - Accept / reject changes. Mask dynamic regions. Summarize test results for your approval
- **Faster and safer agentic development workflow**
 - Agents “see” the UI and can self-correct mistakes before declaring “done”

DEMO

WIKIPEDIA

The Free Encyclopedia

English

7,816,319+ articles

日本語

1,503,000+ 記事

Deutsch

3.595.663+ Artikel

Русский

2 103 000+ статей

Français

2 761 000+ articles

Español

2.116.000+ artículos

中文

1,537,000+ 条目 / 条目

Italiano

1.971.000+ voci

Polski

1 696 000+ haseł

Português

1.173.000+ artigos



EN

Read Wikipedia in your language



Wikipedia is hosted by the Wikimedia Foundation, a non-profit organization that also hosts a range of other projects.



Commons
Free media collection



Wikivoyage
Free travel guide



Wiktionary
Free dictionary

Deterministic Guardrails **break the trust bottleneck**

FASTER REVIEWS



English test steps are easier to review than test code and element locators

Visual assertions reduce the amount of test code by 80%

Review a summary and confirm baseline updates without leaving the IDE

FEWER CYCLES



Agents can “see” the UI and self-correct mistakes before declaring “done”

Stable test executions reduce automation maintenance cycles

Detect design drifts early, immediately as they occur

FEWER BUGS



Visual assertions drastically increase coverage across devices, browsers, operating systems and form factors

Visual assertions catch unexpected regressions

Questions?

Speak with a
Testing Specialist



<http://applitools.info/6qu>