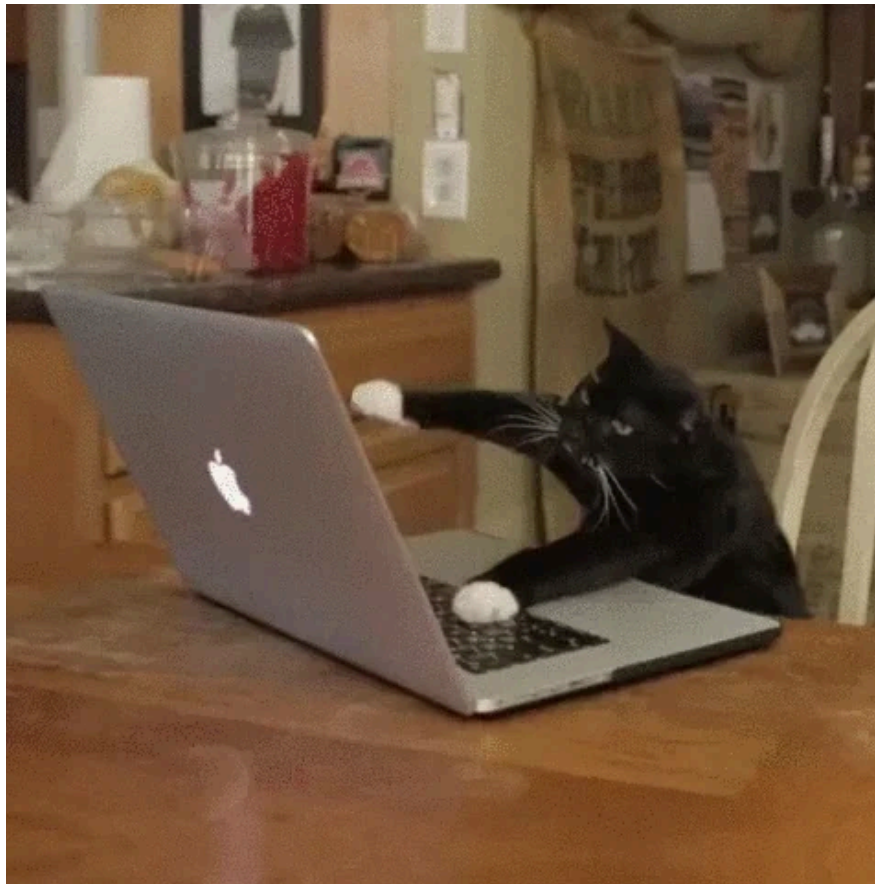
kittest

Framework-agnostic UI testing library, based on AccessKit.

By Lucas Meurer

giphy cat typing



Contents

- What is kittest?
- Query functions
- Query Types
- Helper methods
- Example test (with egui_kittest)
- How does a kittest integration look?
- Pros
- Cons
- What else could be part of kittest?

What is kittest?

- Opinionated UI testing library based on AccessKit
- Thin layer over accesskit_consumer
- Provides convenient api to
  - query the AccessKit node tree
  - trigger AccessKit events
    - e.g. `accesskit::Action::Default` via `Node::click`
  - trigger "Simulated" events (not sure on the naming here)
    - e.g. click at the node's center via `Node::simulate_click`

Query functions

- Inspired by the popular web-dev Testing Library
- Accessible by Default
- Implemented via `Queryable` trait

Query Types

- get_by_*
  - panic when node not found
- get_all_by_*
  - return an iterator of nodes
  - panic when no node found
- query_by_*
  - returning an option
  - panic when more than one node found
- query_all_by_*
  - return an iterator of nodes

Helper methods

- by_role
- by_name
- by_role_and_name
- […]
- via custom filter struct `By`
  - e.g. `harness.get(by().role(Role::CheckBox).name_contains("Check"))`

Example test (with egui_kittest)

```
let mut checked = false;
let app = |ctx: &Context| {
    CentralPanel::default().show(ctx, |ui| {
        ui.checkbox(&mut checked, "Check me!");
    });
};
```

How does a kittest integration look?

- Egui example integration
- for existing test frameworks (like masonry's test harness) kittest could be enabled via a feature, or as a separate crate

Pros

- Accessibility gets tested by default
  - You usually query nodes by their label
  - Thus, it’s e.g. ensured that all interactive elements have an accessible label
- There is one well-thought-out api that ui frameworks can rely on
- Hurdle to write a test harness for an ui framework gets reduced

Cons

- The "plumbing" for each ui framework still has to be done manually
  - Straightforward for egui
  - Might be more complicated for other ui frameworks
- You must use the accessibility labels, some people might prefer using e.g. id strings, those aren’t possible currently with AccessKit
- Nodes hold a reference to the accesskit tree, so they cannot be held across frames

Cons that could be resolved by making `Node` a trait

- AccessKit nodes might not 100% match what the ui framework provides
  - e.g. maybe the ui frameworks checkbox component only has two states while the kittest node will have AccessKits three states
- Currently, kittest only provides access to an accesskit node, not to the original masonry / egui / xilem widget

What else could be part of kittest?

- Image snapshot tests
  - Currently implemented in egui_kittest but could e.g. be released as kittest_image_snapshot
- GitHub action to set up an environment where wgpu will run
  - Should install swiftshader / llvmpipe / vulkan sdk
  - Should work with windows / macOS / linux
- Provide Mappings from the kittest event types to the winit types