Cypress everywhere

Only one tool for all test levels?

Dehla Sokenou

Test and Quality Manager
Software Architect

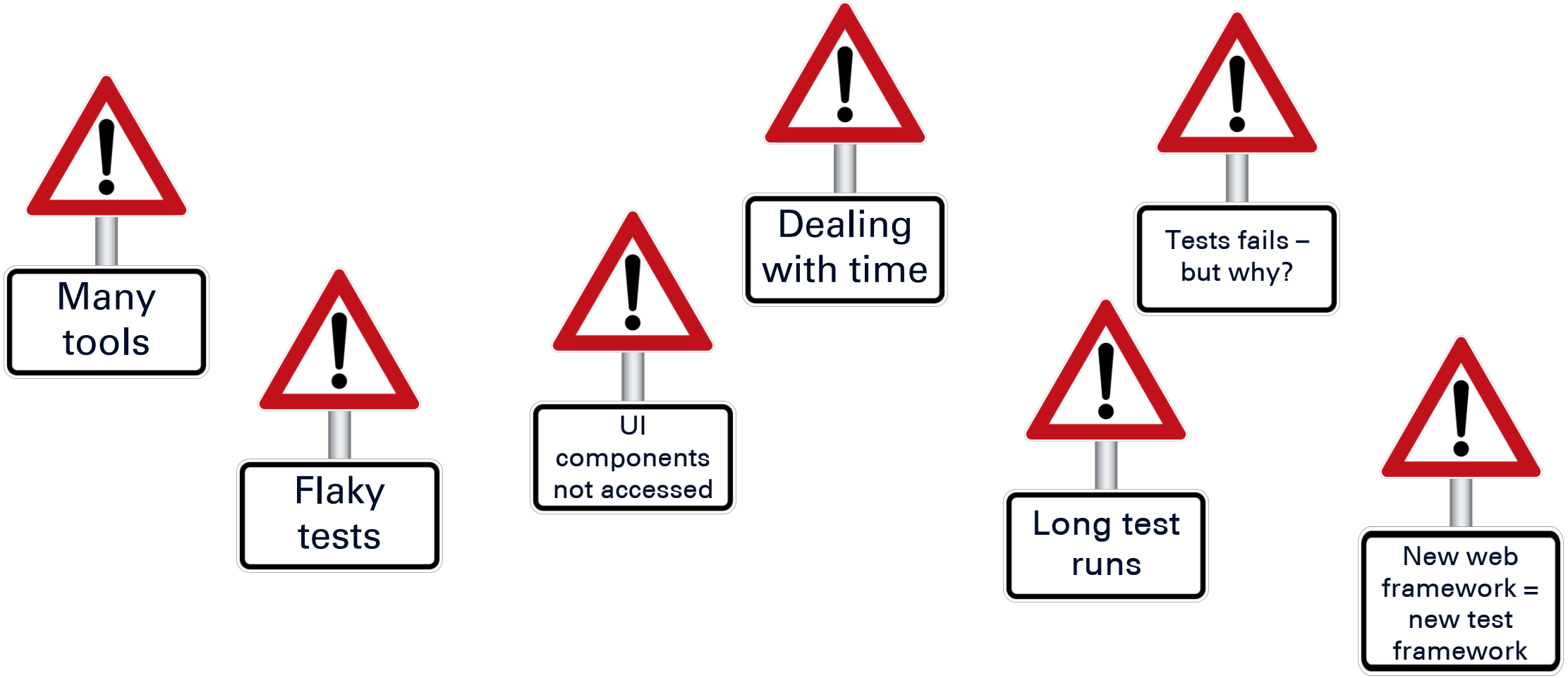
Spokesperson for the GI specialist group TAV

https://www.wps.de/

https://fg-tav.gi.de/

https://dpsq.de/

Do you also know this?



Ask a backend developer…

Unit tests?

xUnit!

Integration tests?

xUnit!

E2E tests?

xUnit!?

Ask a frontend developer…

Unit tests?

Jasmine/Karma?

Enzyme?

Jest?

Vitest?

Integration tests?

Jasmine/Karma?

Enzyme?

Jest?

Vitest?

E2E tests?

Cypress?

Selenium?

Playwright?

Couldn’t we make it better?

Maybe like this…?

Unit tests?

Cypress!

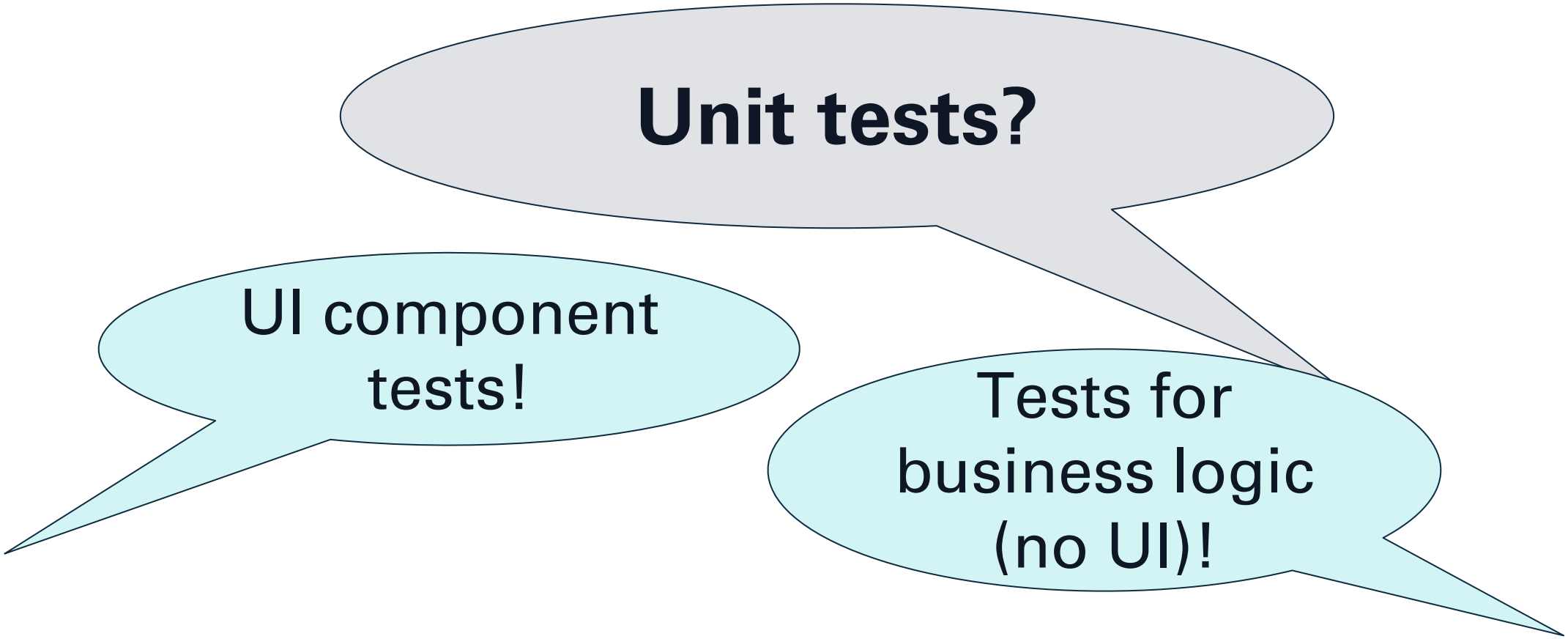
Integration tests?

Cypress!

E2E tests?

Cypress!

Unit-Tests in frontend…?



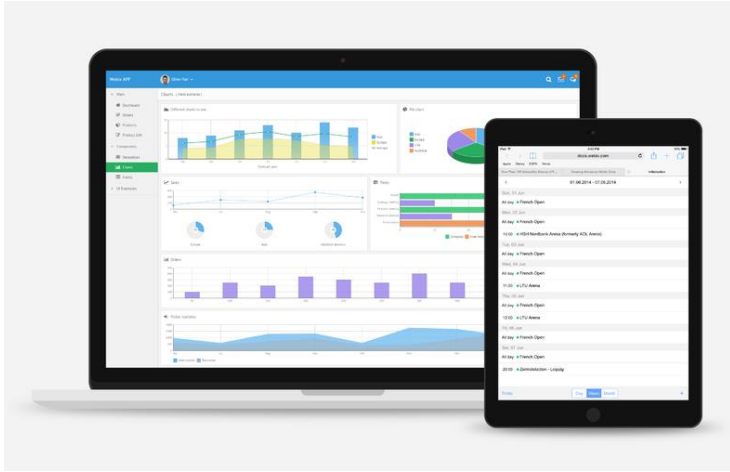
Cypress – a brief history

- 2014/15 Launched as an opponent to Selenium and other capture & replay tools, open source
- 2017 Version 1.0 → E2E Testing
- 2021 Version 7.0 → Component Testing (alpha)
- 2022 Version 10.0 → Component Testing (beta), gradual expansion of framework support (React, Vue, Angular, ...)

Target platforms

- Cypress E2E is suitable for any type of web application
  In particular, it also supports modern single-page apps, where several other tools are weak

- Cypress Component Testing must be provided specifically for the web framework
  But then: uniform technology, regardless of whether you have an Angular, Vue, React, ... Project
  - Jest also supports various frameworks, but only unit, integration, component tests, no E2E



How does it work?

- Tests are implemented → at first glance a disadvantage, e.g. in comparison to Capture & Replay, BDD or keyword-driven testing tools
  - But: who actually uses the testing tool?

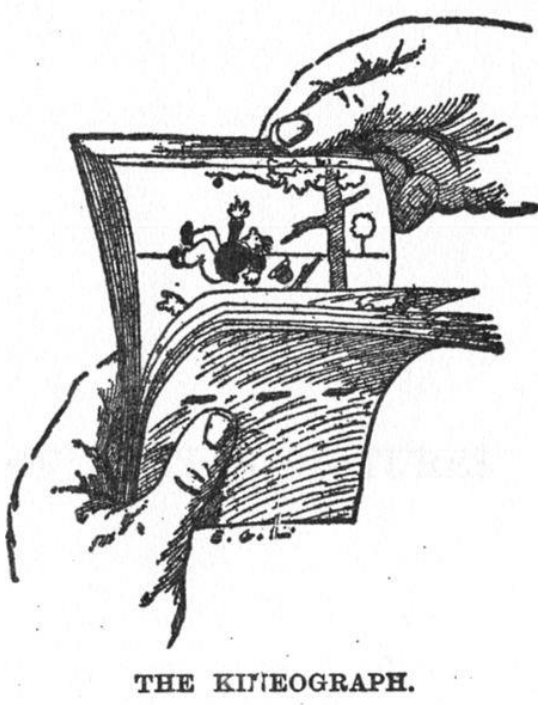
- Same technique can be reused at all testing levels (at least in frontend)
- Same technique can be reused for all common web frameworks (at least in frontend)

And (how) does that solve our problems?

Special features of Cypress – in general

Some characteristics of Cypress that make it special

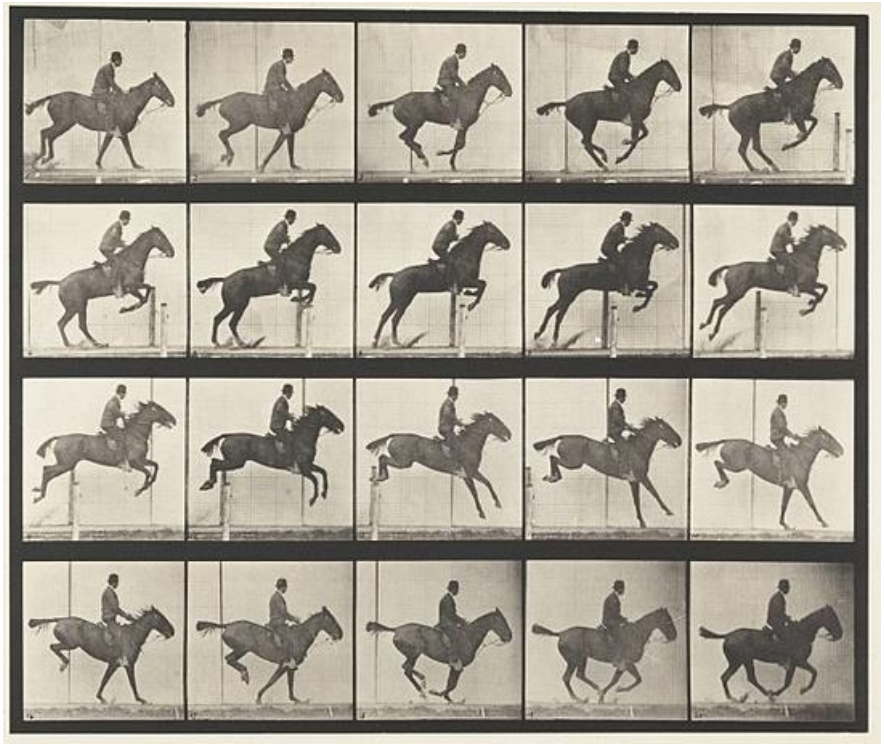
- “Time Travel” (a kind of flip book)
- Automatic screenshots / DOMs in case of errors
  → better error analysis
- Clock
- Mocks (spies, stubs)
- Automatic waiting
- Typing at (fast) user speed



Special features of Cypress – E2E tests

Some characteristics of Cypress that make it special

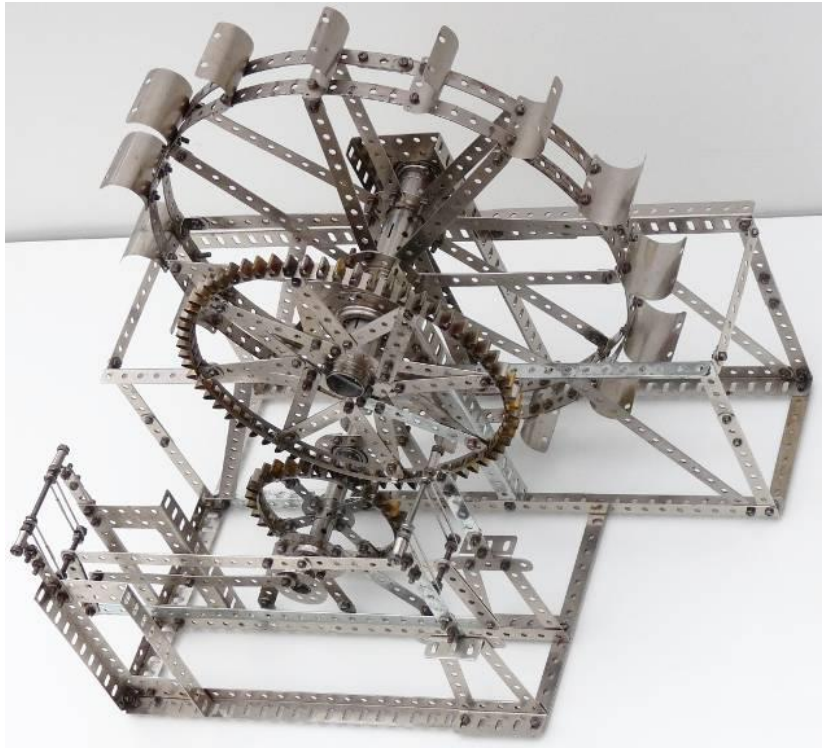
- Very stable, only few flaky tests
- Very fast
- Cross-browser testing (in real browser)
- Runs with the application in one environment and allows control from within
- Mocking of network traffic also in E2E tests



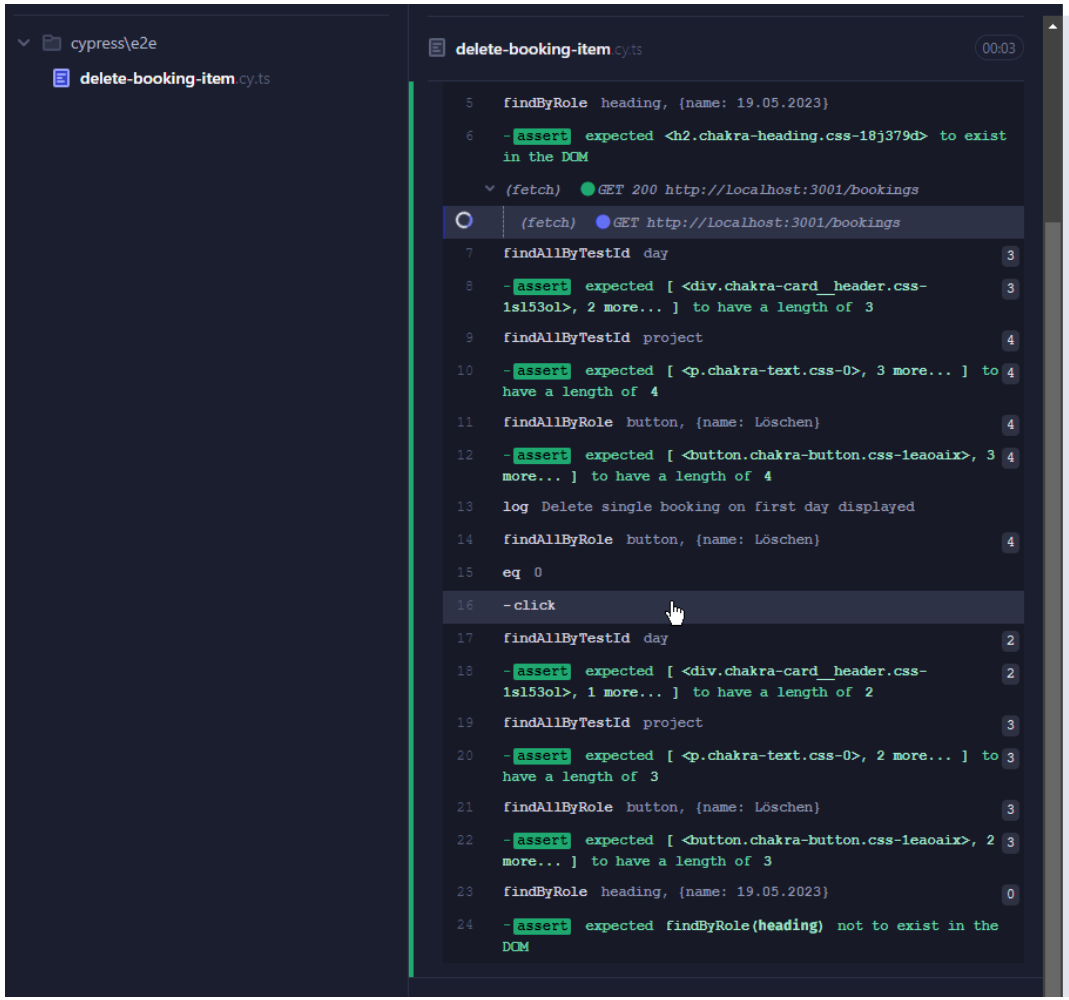
Special features of Cypress – Component and integration tests

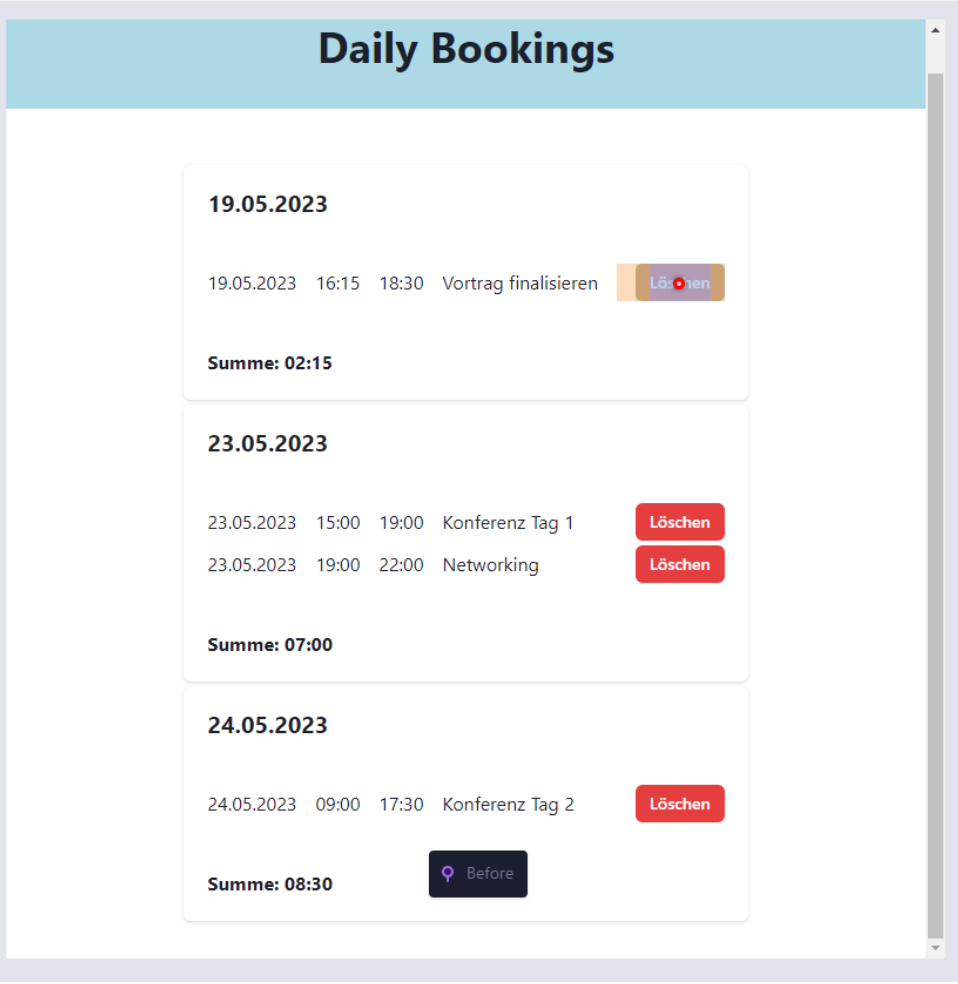
Some characteristics of Cypress that make it special

- Very stable, only few flaky tests, especially when having a lot of user interaction
- You can see what is being tested (like Karma, in contrast to Jest, for example)
- It's still fast (like Jest, unlike Karma, for example)
- Wide framework support (like Jest)



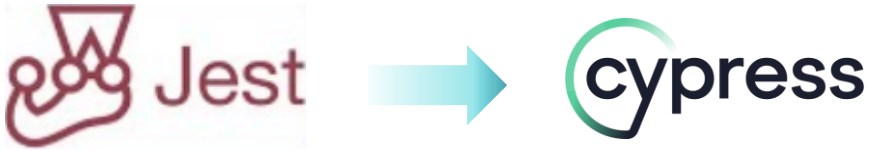
Code Insights


cypress\e2e
delete-booking-item.cy.ts
delete-booking-item.cy.ts
00:03
findByRole heading, {name: 19.05.2023}
-assert expected <h2.chakra-heading.css-18j379d> to exist in the DOM
(fetch) GET 200 http://localhost:3001/bookings
(fetch) GET http://localhost:3001/bookings
findAllByTestId day
-assert expected [<div.chakra-card__header.css-1sl53ol>, 2 more...] to have a length of 3
findAllByTestId project
-assert expected [<p.chakra-text.css-0>, 3 more...] to have a length of 4
findAllByRole button, {name: Löschen}
-assert expected [<button.chakra-button.css-1eaoaix>, 3 more...] to have a length of 4
log Delete single booking on first day displayed
findAllByRole button, {name: Löschen}
eq 0
-click
findAllByTestId day
-assert expected [<div.chakra-card__header.css-1sl53ol>, 1 more...] to have a length of 2
findAllByTestId project
-assert expected [<p.chakra-text.css-0>, 2 more...] to have a length of 3
findAllByRole button, {name: Löschen}
-assert expected [<button.chakra-button.css-1eaoaix>, 2 more...] to have a length of 3
findByRole heading, {name: 19.05.2023}
-assert expected findByRole(heading) not to exist in the DOM


Daily Bookings
19.05.2023
19.05.2023 16:15 18:30 Vortrag finalisieren Löschen
Summe: 02:15
23.05.2023
23.05.2023 15:00 19:00 Konferenz Tag 1 Löschen
23.05.2023 19:00 22:00 Networking Löschen
Summe: 07:00
24.05.2023
24.05.2023 09:00 17:30 Konferenz Tag 2 Löschen
Summe: 08:30
Before

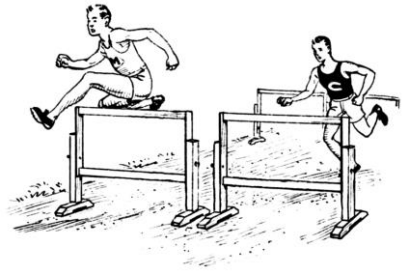
Demo

Migration of existing test suites

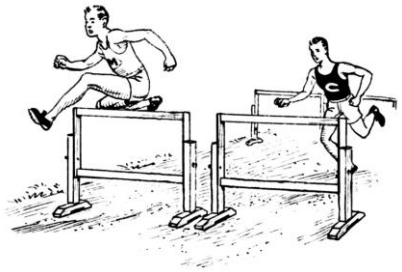


- Which tests are worth it?
  - We primarily migrated the flaky / long-running tests
  - All further tests directly in Cypress
- How much effort is that?
  - Teaser: 90% pure translation, 10% brainpower
  - Visual input, therefore faster when writing new tests
- What is the best way to proceed?
  - Separate the tests by naming them (what has already been converted, what still needs to be converted?)
- What is the current status of Cypress (e.g., in terms of API stability, false negatives)?

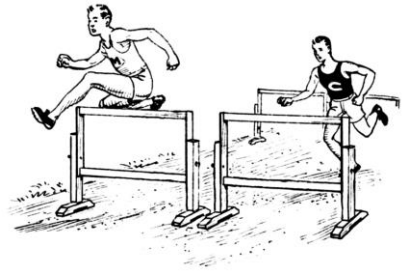
Migration of existing test suites: Pitfalls?



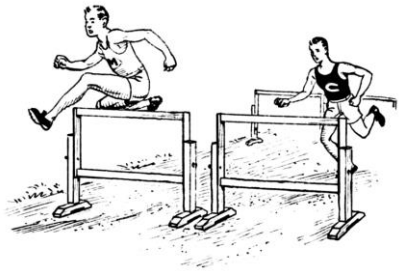
- Cypress UI and tests start up slowly



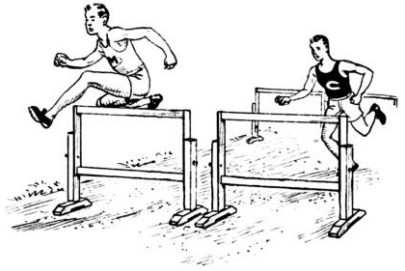
- Module Mocking



- Failing tests in Cypress UI



- Support in IDEs is sometimes insufficient (including debugging)



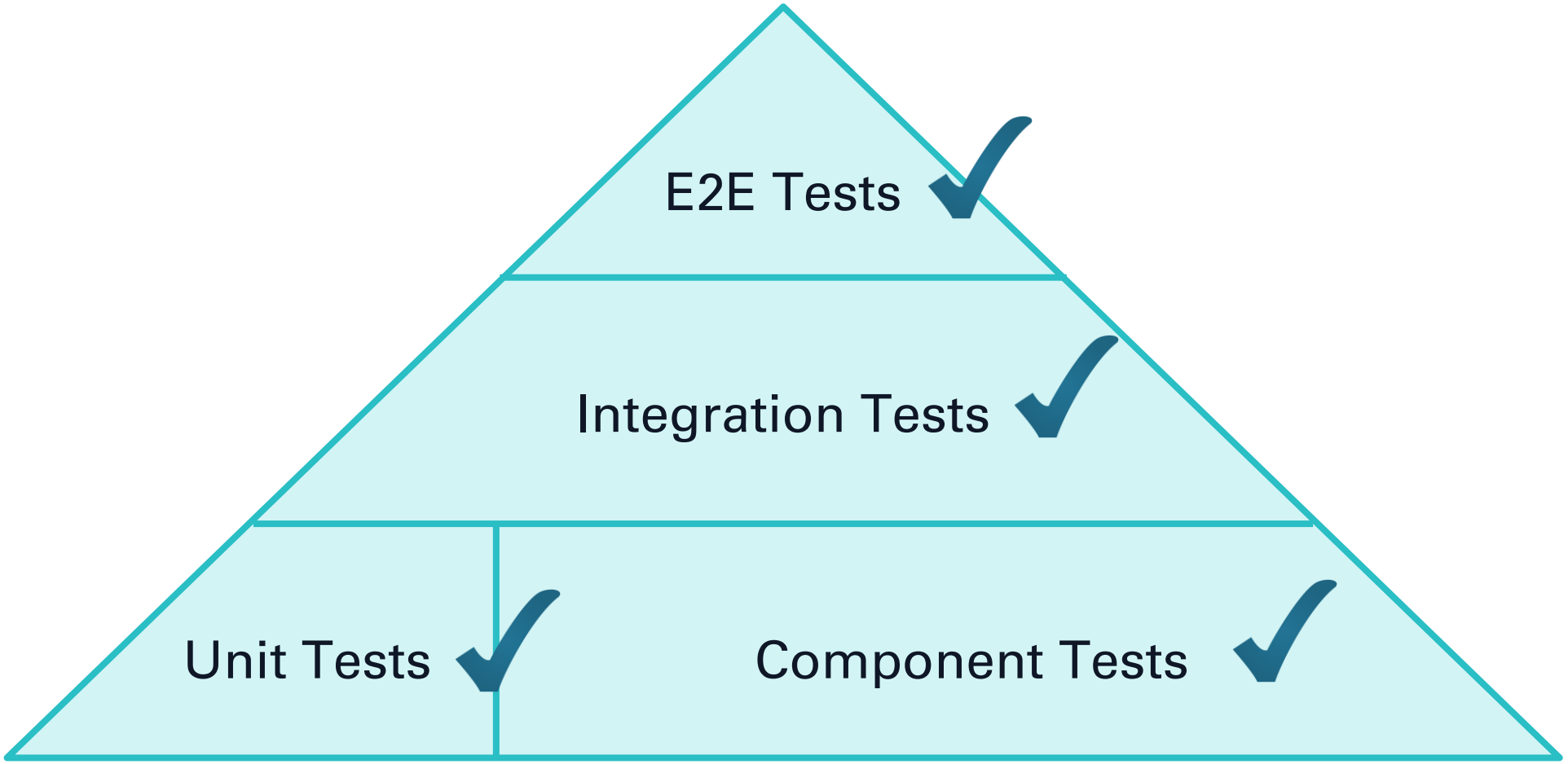
- Some E2E test scenarios not supported

But:

- Different way of working
  - Cypress UI in watch mode
  - Auto restart of the tests
  - The play button in the IDE is rarely missed, even if it is usually there now
- Faster start up with Vite

Evaluation

Is it now a tool for all test levels (in the frontend)?



What to consider…

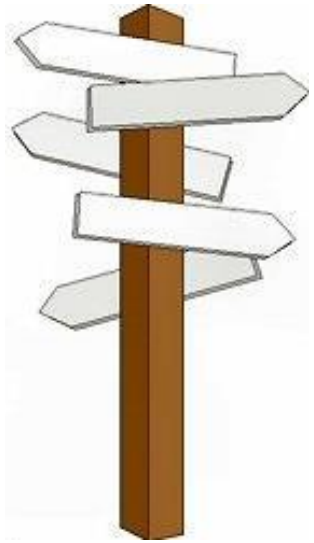
- When you decide on a tool, the following questions are also important:
  - Sustainability (is the project alive?)
  - Hard breaks between versions requiring a lot of adjustment?
  - Limitations of the open source version compared to the commercial version?
  - Not open source: Cypress Cloud with Flaky test management, parallelization and auto-rerun/auto-cancel
    (but there are workarounds)

Our conclusion

- Who could/should use Cypress Component Testing (and who shouldn’t)?
  - Yes, if projects with different frontend technologies are used
  - Yes, if the tests are very flaky
  - Maybe if you already use Cypress as an E2E tool
  - Maybe if tests are difficult to write without visual feedback
  - Maybe not if there are other UIs besides the web ones
  - No, if automated tests are currently running smoothly

Our conclusion

- What does Cypress show what a testing tool must be able to do?
  → Helpful for evaluations of other testing tools
  - Visual feedback, time travel and screenshots are helpful for error analysis
  - Tests should run in real browsers
    - Otherwise, you won't find browser-dependent errors
    - Otherwise, some tests will be slow or flaky because of the emulation
- Robustness of the tests is essential
- Support from common third-party libraries makes sense and makes porting tests easier
  - E.g. testing library



Appendix

Links

Test Solutions for Web frontend development

- Cypress: https://www.cypress.io/
- Cypress Dashboard (Open Source Variant): https://sorry-cypress.dev/
- Selenium: https://www.selenium.dev/
- Playwright: https://playwright.dev/
- WebDriverIO: https://webdriver.io/
- Jest: https://jestjs.io/
- Vitest: https://vitest.dev
- Karma / Jasmine: https://angular.io/guide/testing
- Testing Library: https://testing-library.com/

Image / photo credits (in the order of appearance)

Images / photos:

- https://www.cypress.io/
- https://commons.wikimedia.org/wiki/File:Strider_Linkage_Robot_Climbing.gif (CC BY-SA 4.0)
- https://commons.wikimedia.org/wiki/File:Zeichen_101_-_Gefahrstelle,_StVO_1970.svg (Public Domain)
- https://commons.wikimedia.org/wiki/File:Software_Developer_at_work_03.jpg (CC BY-SA 4.0)
- https://commons.wikimedia.org/wiki/File:Software_developer_at_work_02.jpg (CC BY-SA 4.0)
- https://commons.wikimedia.org/wiki/File:Zeichen_206_-_Halt!_Vorfahrt_gew%C3%A4hren!_StVO_1970.svg (Public Domain)
- https://commons.wikimedia.org/wiki/File:Pyramides_et_le_Sphinx_MET_DP113869.jpg (CC0 1.0)
- https://commons.wikimedia.org/wiki/File:JavaScript_UI_widgets_library_for_building_desktop_and_mobile_web_apps.png (CC BY-SA 4.0)
- https://commons.wikimedia.org/wiki/File:Victorinox_climber.jpg (CC BY-SA 4.0)
- https://commons.wikimedia.org/wiki/File:Linnet_kineograph_1886.jpg (Public Domain)
- https://commons.wikimedia.org/wiki/File:Horse_Daisy_jumping_a_hurdle,_saddled_with_a_rider_(rbm-QP301M8-1887-639).jpg (Public Domain)
- https://commons.wikimedia.org/wiki/File:Stabil_Modell_751_klein.jpg (CC BY-SA 4.0)
- https://jestjs.io/
- https://commons.wikimedia.org/wiki/File:Hurdle_(PSF).png (Public Domain)
- https://commons.wikimedia.org/wiki/File:Jack-in-the-box.jpg (Public Domain)
- https://commons.wikimedia.org/wiki/File:Confused_man.jpg (CC BY-SA 2.5)
- https://commons.wikimedia.org/wiki/File:BlackMarble_2016_rotating_globe_at_night_transparent.gif (Public Domain)

Licenses:

- CC0 1,0: https://creativecommons.org/publicdomain/zero/1.0/deed.en
- CC BY-SA 2.5: https://creativecommons.org/licenses/by-sa/2.5/deed.en
- CC BY-SA 4.0: https://creativecommons.org/licenses/by-sa/4.0/deed.en