QUALITÄT UND QUALITÄTS-SICHERUNG

IN DER FRONTEND- UND WEB-ENTWICKLUNG

Contao Konferenz 2025 | Janosch Oltmanns

JANOSCH OLTMANNS

- Leitung Softwareentwicklung bei OVIS IT
- Aktiv für und mit Contao Open Source CMS
- Performance-Enthusiast aus Leidenschaft
- *@JanoschOltmanns*

KEY FACTS

GRUNDLAGEN CHECKS VS. TESTS AUTOMATISIERUNG
CODE STYLE END-TO-END PLAYWRIGHT IN DER PRAXIS
ACCESSIBILITY GITHUB-ACTIONS

QUALITÄT UND QUALITÄTS-SICHERUNG

IN DER FRONTEND- UND WEB-ENTWICKLUNG

TESTING GRUNDLAGEN
(AKA. *THEORIE*)

TEST VS. CHECK

TEST VS. CHECK

	Checks	**Tests**
Ziel	Einhaltung von Code-Standards und Regeln	Sicherstellen der fachlichen und technischen Korrektheit
Ebene	Quellcode, Syntax, Stil	Laufzeitverhalten, Logik, Interaktion
Methode	Statische Analyse (ohne Ausführung des Codes)	Ausführung des Codes
Ergebnis	Verstöße, Warnungen, Ja/Nein	Bestanden/Nicht bestanden, detaillierte Fehler
Dauer	Sehr schnell, oft Sekunden	Langsam, je nach Testtyp Sekunden bis Minuten
Aussage	*„Der Code ist ~~sauber~~ korrekt und konsistent!“*	*„Die Software funktioniert wie vorgesehen!“*

LINTER, STYLE-CHECKS

TOOLS

PHPSTAN PSALM PHP_CODESNIFFER PHP CS FIXER ESLINT
STYLELINT PRETTIER SONARQUBE ...

STYLELINT

VERMEIDUNG VON FEHLERN

- falsche Anweisungen (bsp. im CSS-Grid)
- doppelte Selektoren
- falsche Deklarationen
- falsche Properties

STYLELINT

EINHALTUNG VON KONVENTIONEN (BEST-PRACTICES)

- Verbot von Einheiten
- Namenskonventionen für bsp. Klassen, Custom Properties
- Anzahl oder Verwendung von ID-Selektoren
- Nutzung moderner Funktionen
- …

STYLELINT

- https://stylelint.io/
- https://github.com/JanoschOltmanns/ck2025-stylelint

STYLELINT INITIALISIERUNG

```
$ yarn init -2
$ yarn add stylelint --dev
$ yarn add stylelint-config-standard --dev
$ yarn add stylelint-config-standard-scss --dev
$ yarn add stylelint-config-sass-guidelines --dev
$ yarn add postcss --dev
$ yarn add postcss-scss --dev
```

STYLELINT KONFIGURATION

```
// stylelint.config.mjs

/** @type {import('stylelint').Config} */
export default {
  extends: ["stylelint-config-sass-guidelines"],
  rules: {},
  "ignoreFiles": []
};
```

https://stylelint.io/user-guide/configure/

STYLELINT KONFIGURATION

```
// package.json

{
  "name": "demo-stylelint",
  "packageManager": "yarn@4.10.2",
  "devDependencies": {
    "postcss": "^8.5.6",
    "postcss-scss": "^4.0.9",
    "stylelint": "^16.24.0",
    "stylelint-config-sass-guidelines": "^12.1.0"
  },
  "scripts": {
    "lint": "stylelint \"frontend/layout/**/*.{css,scss}\"",
    "lint:fix": "stylelint \"frontend/layout/**/*.{css,scss}\" --fix"
  }
}
```

STYLELINT NUTZUNG

```
$ yarn lint
```

```
$ yarn lint:fix
```

STYLELINT GITHUB ACTION

```yaml
name: 'GitHub Actions: Linting with Stylelint'
run-name: ${{ github.actor }} is linting with GitHub Actions 🤦
on: [push]

jobs:
  Linting:
    runs-on: ubuntu-latest

    steps:

    - name: Install Node.js
      uses: actions/setup-node@v4
      with:
        node-version: 20

    - run: corepack enable

    - name: Checkout
      uses: actions/checkout@v4
      with:
```

https://github.com/JanoschOltmanns/ck2025-stylelint

UNIT-, INTEGRATION-, E2E-TESTS

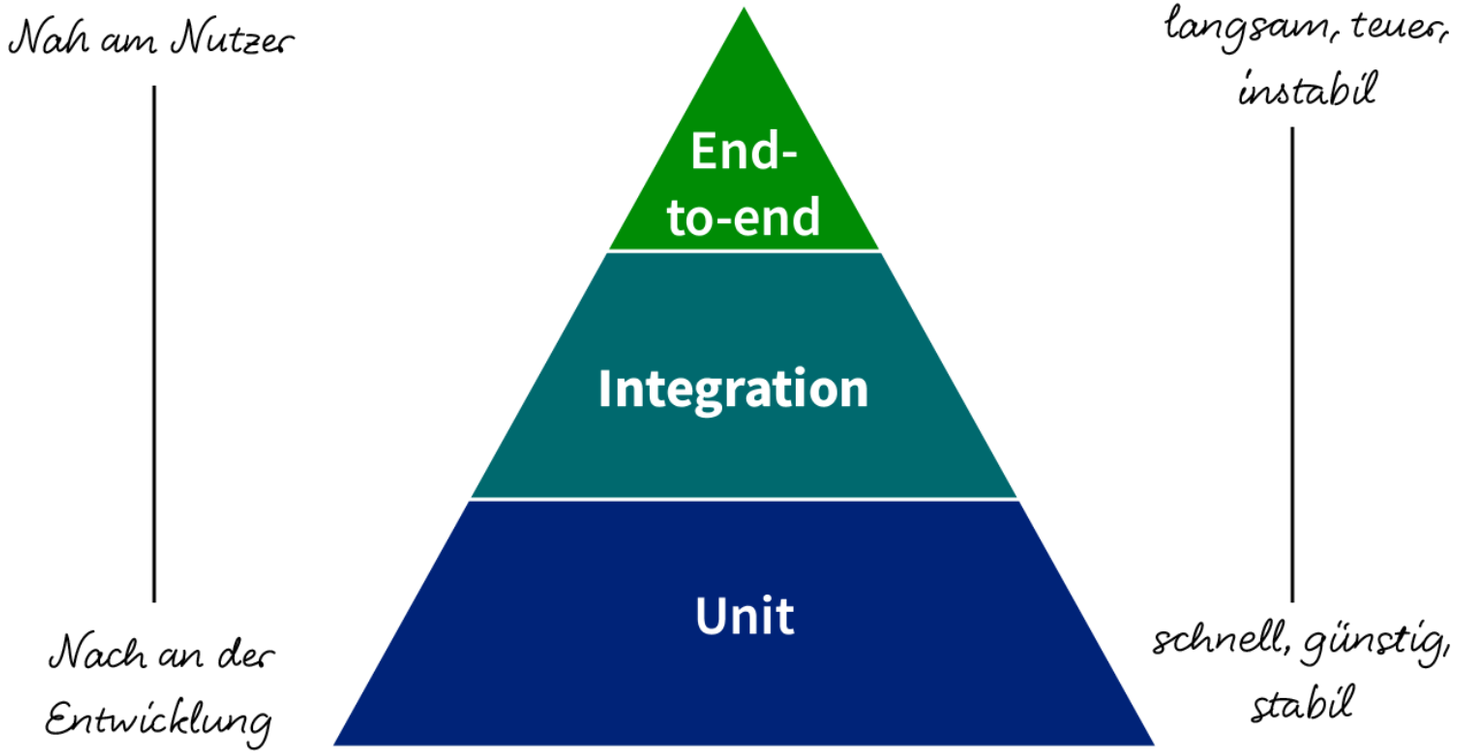
TESTARTEN

- Statische Analyse
- Unit Tests
- Integration Tests
- Funktionale Tests (aka. Application Tests)
- End-To-End Tests
- Visuelle Regressions Tests

TESTARTEN

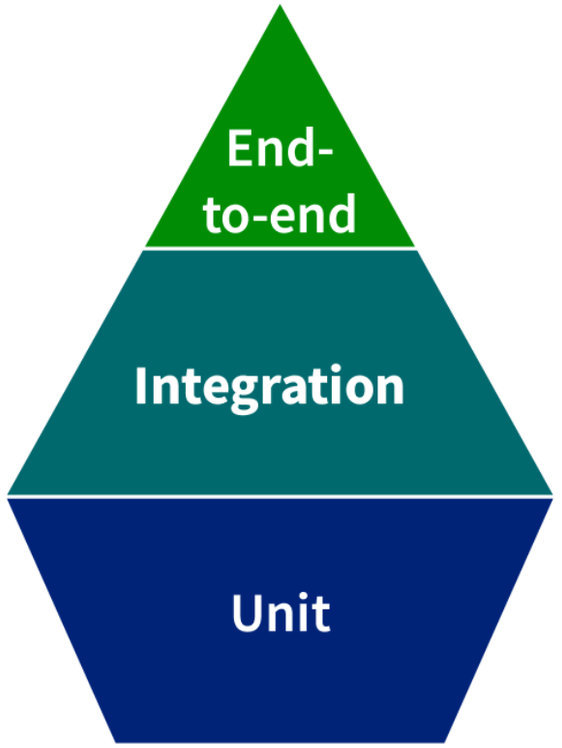
- **Automatisierte Tests**
  - Statische Analyse
  - Unit Tests
  - Integration Tests
  - Funktionale Tests (aka. Application Tests)
  - End-To-End Tests
  - Visuelle Regressions Tests
- **Manuelle Tests**

TEST PYRAMIDE



TEST DIAMANT

Nah am Nutzer

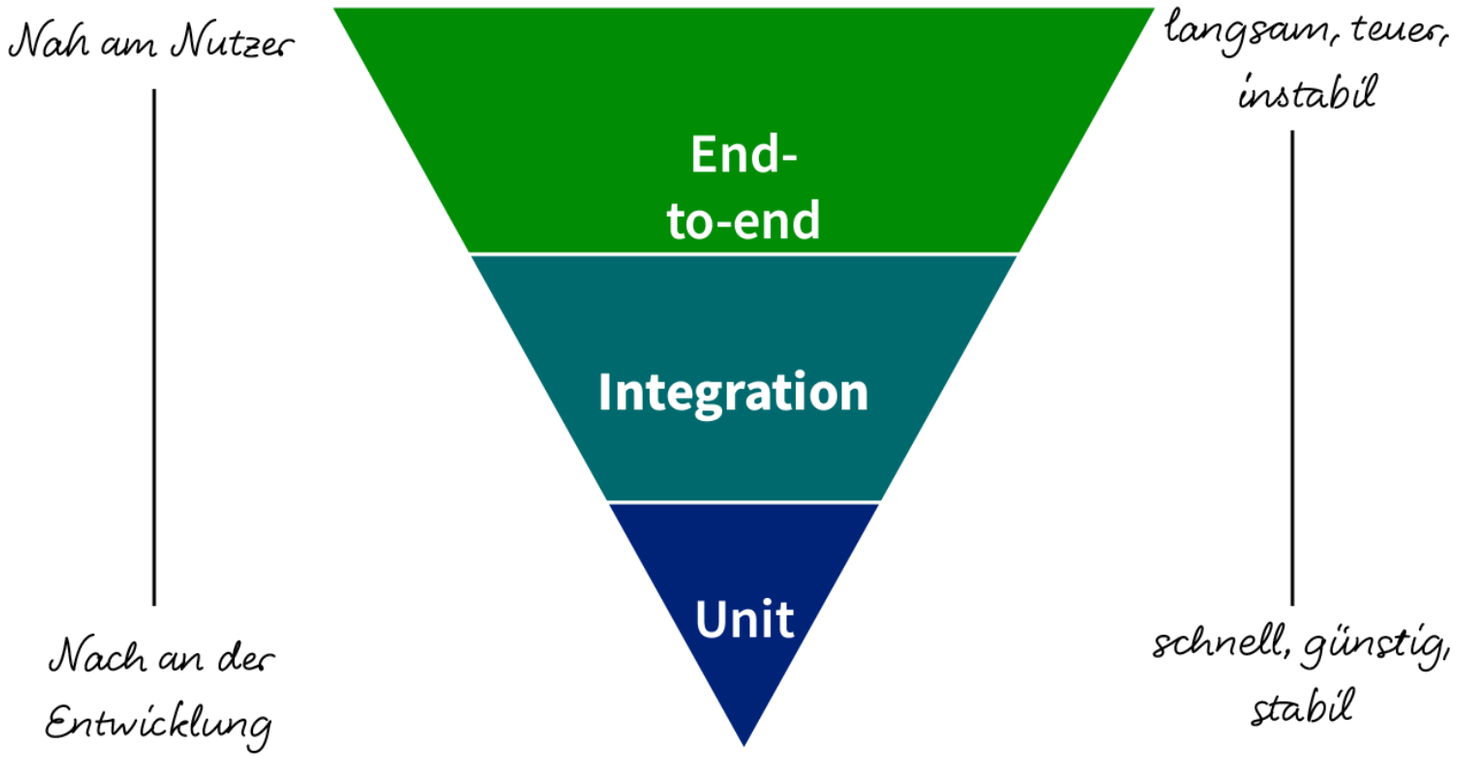


langsam, teuer, instabil

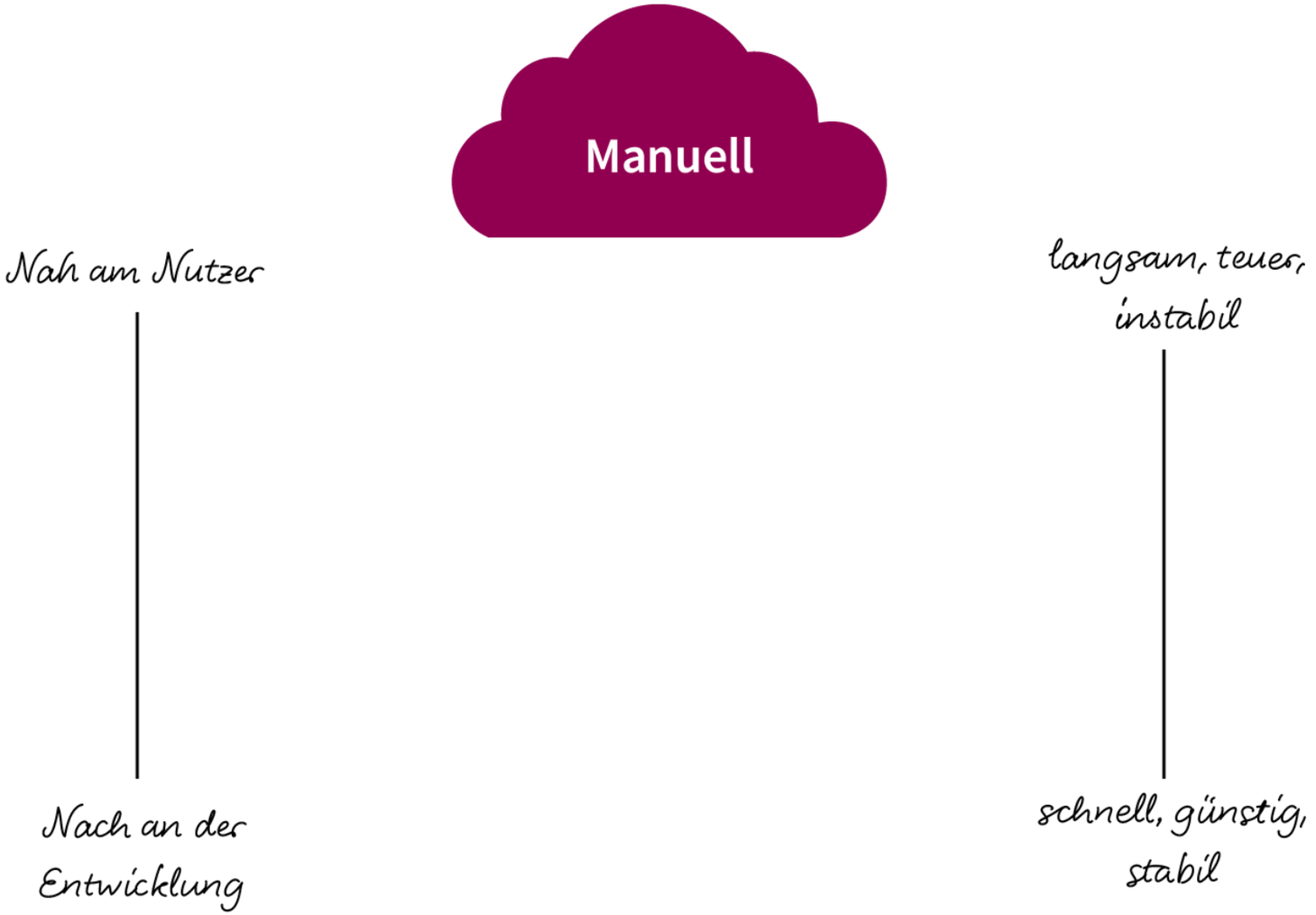
Nach an der Entwicklung

schnell, günstig, stabil

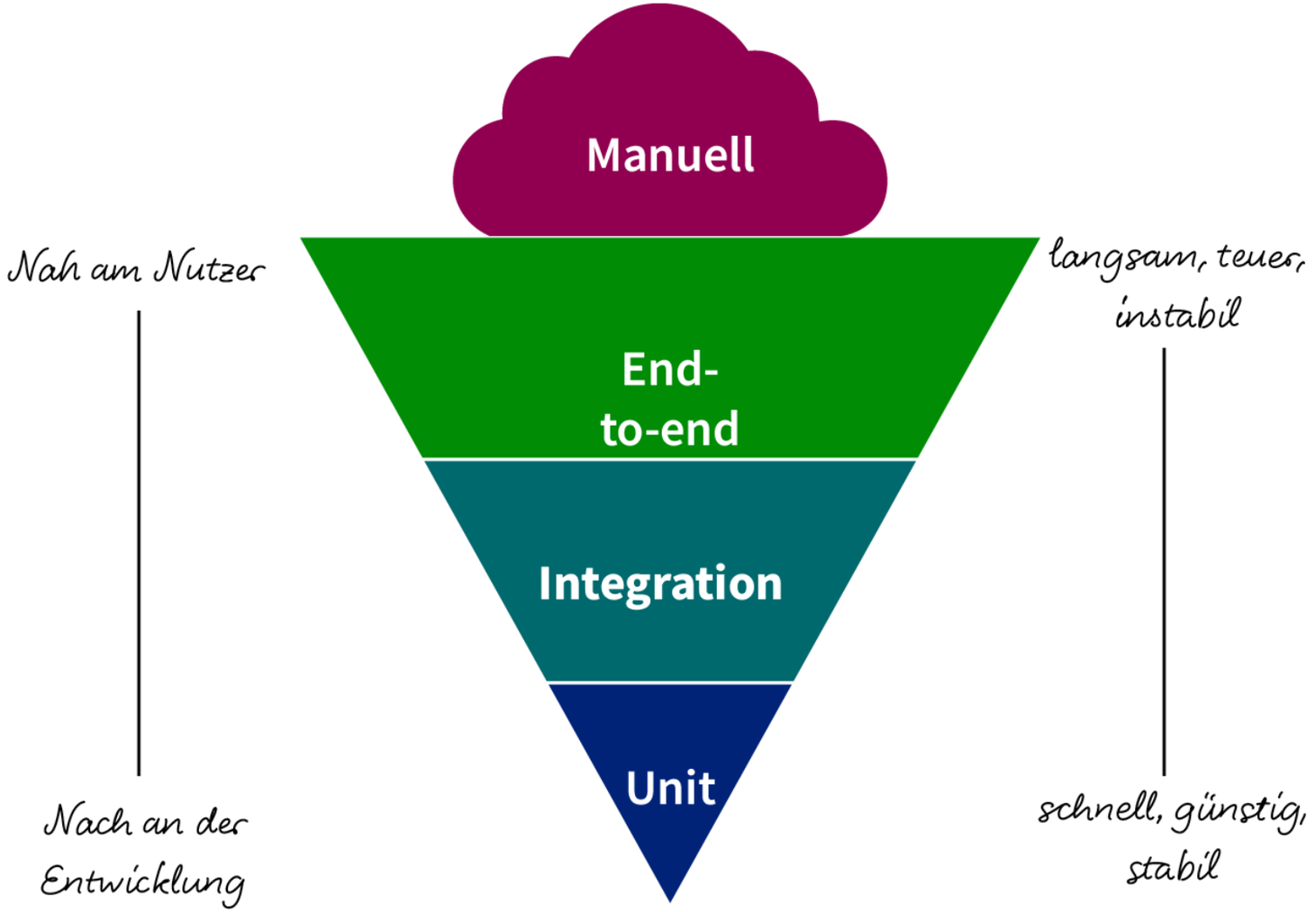
TEST PIZZA


Nah am Nutzer
Nach an der Entwicklung
End-to-end
Integration
Unit
langsam, teuer, instabil
schnell, günstig, stabil

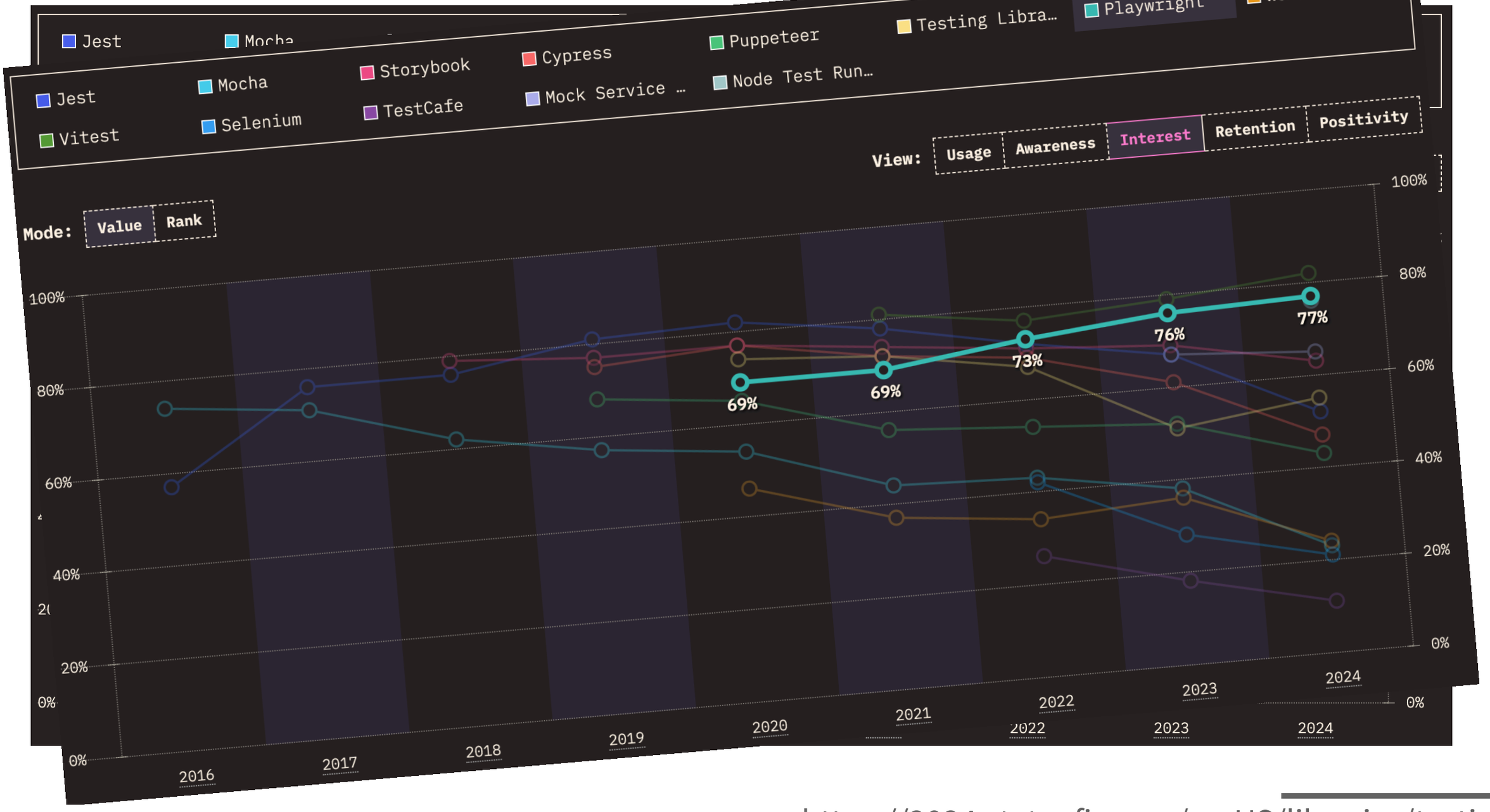
TEST WOLKE


Manuell
Nah am Nutzer
Nach an der Entwicklung
langsam, teuer, instabil
schnell, günstig, stabil

TEST EISWAFFEL



END-TO-END TESTS



https://2024.stateofjs.com/en-US/libraries/testing/

PLAYWRIGHT VS. CYPRESS

Feature	Playwright	Cypress
Sprache	JavaScript, TypeScript, Python, Java, C#	JavaScript, TypeScript
Test Runner	Funktioniert mit Jest, Mocha und anderen	Integrierter Test Runner
Betriebssysteme	Windows, macOS, Linux	Windows, macOS, Linux
Open Source	Ja	Ja
Paralleles Testen	Volle Parallelität, auch innerhalb von Spezifikationen	Parallel nur auf Spezifikationsebene
Architektur	Nutzt Browser-Kontexte zur Isolierung	Läuft direkt im Browser
Unterstützte Browser	Chromium, Firefox, WebKit	Chrome, Edge (Chromium-based)
Dokumentation	Starke Unterstützung durch Microsoft, Discord-Kanal	Starke Community, Slack-Unterstützung
Unterstützung echter Geräte	Begrenzt	Keine
Plugins	Benötigt individuelles Setup	Plugin-Ökosystem

www.checklyhq.com/learn/playwright/playwright-vs-cypress/

PLAYWRIGHT VS. CYPRESS

- Playwright empfiehlt die Nutzung von Accessibility-Locators wie bsp. `getByRole()`
- Playwright ist „Company-baked“
- Playwright hat **keine** Abhängigkeiten

WAS NÜTZT MIR DAS GANZE?

VISUELLE REGRESSION MIT PLAYWRIGHT

PLAYWRIGHT INITIALISIERUNG

```
$ yarn create playwright
```

[enter] + [enter] + [enter]

PLAYWRIGHT NUTZUNG

```
$ yarn playwright test
```

```
$ yarn playwright show-report
```

```
$ yarn playwright test --ui
```

```
$ yarn playwright test --update-snapshots
```

```
$ yarn playwright codegen your-awsome-url.de
```

VISUELLE REGRESSION MIT PLAYWRIGHT

```
import { test, expect } from '@playwright/test';

const baseUrl = 'https://www.odas.biz';
const urls = [];

for (const url of urls) {
  test(`Seite ${url} sieht aus wie erwartet`, async ({ page }) => {
    await page.goto(`${baseUrl}${url}`);

    expect(await page.screenshot({fullPage: true}))
      .toMatchSnapshot(`${url.replace(/\//g, '_')}.png`);
  });
}
```

ACCESSIBILITY TESTING

ACCESSIBILITY TESTING

Disclaimer

Automated accessibility tests can detect some common accessibility problems such as missing or invalid properties. But many accessibility problems can only be discovered through manual testing. We recommend using a combination of automated testing, manual accessibility assessments, and inclusive user testing.

https://playwright.dev/docs/accessibility-testing

PLAYWRIGHT ACCESSIBILITY

```
$ yarn add @axe-core/playwright
$ yarn add axe-html-reporter
```

PLAYWRIGHT ACCESSIBILITY

```
for (const url of urls) {
    test(`Seite ${url} hat keine automatisch erkennbaren Accessibility-Probleme`, async ({ pag
    ) => {
        await page.goto(`${baseUrl}${url}`);
        const accessibilityScanResults = await new AxeBuilder({ page }).withTags([
            'wcag2a', 'wcag2aa', 'wcag2aaa', 'wcag21a', 'wcag21aa', 'wcag22aa'
        ]).disableRules(['color-contrast', 'color-contrast-enhanced']).analyze();

        const folderPath = `${testInfo.config.rootDir}/${testInfo.titlePath[0]}-accessibility`
        const reportFileName = `${url.replace(/\//g, '_')}.html`;

        const reportHTML = createHtmlReport({
            results: accessibilityScanResults,
            options: {
                projectKey: 'I need only raw HTML'
```

NOCH FRAGEN?

RESSOURCEN

- web.dev/articles/ta-types
- web.dev/articles/ta-strategies
- stylelint.io/
- playwright.dev

VIELEN DANK!

@JanoschOltmanns

www.ackerprofi.de/jobs