

esolutions

Nun wollen wir unsere Skills testen

Einführung

Die Idee dieser Slides ist keine Komplettlösung. Eher eine Aufgabensammlung, welche teilweise mit dem Trainer erarbeitet wird und teilweise auch im Nachgang. Je nach eigenem Ermessen.

Wie testen wir?

Zwei Dinge, die wir gelernt haben sind:

Testen aus Benutzer-Sicht; aus den best practices

Tests nach high complexity und high value auswählen.

Wie testen wir?

Aber was heisst das nun konkret?

Meist gibt es zwei Arten wie getestet wird:

- Seitenweise (index, contact, information,...)
- User-Flow

Beide haben Vorteile. Seitenweise ist bei einem Fehler sehr einfach nachvollziehbar. User-Flow konzentriert sich auf die (aus unserer Sicht) wichtigsten Abläufe.

Report mit Screenshot



```
1 // 21_reports/tests/01_screenshot/screenshot.test.ts
2
3 import { test } from '@playwright/test';
4
5 test('home page screenshot', async ({ page }, testInfo) => {
6     await page.goto('https://checklyhq.com');
7
8     const homeScreenshot = await page.screenshot();
9     testInfo.attach('Home Page', {
10         body: homeScreenshot,
11         contentType: 'image/png',
12     });
13
14     await page.click('nav a[href="/customers/"]');
15     const customersScreenshot = await page.screenshot();
16     testInfo.attach('Customers Page', {
17         body: customersScreenshot,
18         contentType: 'image/png',
19     });
20 });
```

Report mit Screenshot



```
1 npx playwright test
2
3 npx playwright show-report
```

Reporters

Playwright Test verfügt über einige integrierte Reporter für unterschiedliche Anforderungen und die Möglichkeit, benutzerdefinierte Reporter bereitzustellen.

Der einfachste Weg, integrierte Reporter auszuprobieren, ist die Übergabe der Kommandozeilenoption `--reporter`.



```
1 npx playwright test --reporter=line
```

```
2
```

Reporters

Playwright Test verfügt über einige integrierte Reporter für unterschiedliche Anforderungen und die Möglichkeit, benutzerdefinierte Reporter bereitzustellen.

Der einfachste Weg, integrierte Reporter auszuprobieren, ist die Übergabe der Kommandozeilenoption `--reporter`.



```
1 import { defineConfig } from '@playwright/test';  
2  
3 export default defineConfig({  
4   reporter: 'line',  
5 });
```


Multi-Reporters

Man kann auch mehrere Reporter gleichzeitig verwenden.

Zum Beispiel kann man „list“ für eine schöne Terminalausgabe und „json“ für eine umfassende json-Datei mit den Testergebnissen verwenden.

```
1 import { defineConfig } from '@playwright/test';  
2  
3 export default defineConfig({  
4   reporter: [  
5     ['list'],  
6     ['json', { outputFile: 'test-results.json' }]  
7   ],  
8 });
```

List-Reporter

Der **List-Reporter** unterstützt die folgenden Konfigurationsoptionen und Umgebungsvariablen:

Environment Variable Name	Reporter Config Option	Description	Default
<code>PLAYWRIGHT_LIST_PRINT_STEPS</code>	<code>printSteps</code>	Whether to print each step on its own line.	<code>false</code>
<code>PLAYWRIGHT_FORCE_TTY</code>		Whether to produce output suitable for a live terminal. If a number is specified, it will also be used as the terminal width.	<code>true</code> when terminal is in TTY mode, <code>false</code> otherwise.
<code>FORCE_COLOR</code>		Whether to produce colored output.	<code>true</code> when terminal is in TTY mode, <code>false</code> otherwise.

Line-Reporter

Der **Line-Reporter** ist übersichtlicher als der Listenreporter. Er verwendet eine einzige Zeile, um den letzten abgeschlossenen Test zu melden, und gibt Fehler aus, wenn sie auftreten. Vor allem für grosse Testsuites.



```
1 npx playwright test --reporter=line
```



```
1 import { defineConfig } from '@playwright/test';  
2  
3 export default defineConfig({  
4   reporter: 'line',  
5 });
```

Line-Reporter

Der **Line-Reporter** unterstützt die folgenden Konfigurationsoptionen und Umgebungsvariablen:

Environment Variable Name	Reporter Config Option	Description	Default
<code>PLAYWRIGHT_FORCE_TTY</code>		Whether to produce output suitable for a live terminal. If a number is specified, it will also be used as the terminal width.	<code>true</code> when terminal is in TTY mode, <code>false</code> otherwise.
<code>FORCE_COLOR</code>		Whether to produce colored output.	<code>true</code> when terminal is in TTY mode, <code>false</code> otherwise.

Weitere Reporter

Es gibt noch weitere Reporter:

- Dot
- HTML
- Blob
- JSON
- JUnit
- Github Actions annotations

Und sogar eigene können entworfen werden:

Eigene Reporter

my-awesome-reporter.ts

```
1 import type {
2   FullConfig, FullResult, Reporter, Suite, TestCase, TestResult
3 } from '@playwright/test/reporter';
4
5 class MyReporter implements Reporter {
6   onBegin(config: FullConfig, suite: Suite) {
7     console.log(`Starting the run with ${suite.allTests().length} tests`);
8   }
9
10  onTestBegin(test: TestCase, result: TestResult) {
11    console.log(`Starting test ${test.title}`);
12  }
13
14  onTestEnd(test: TestCase, result: TestResult) {
15    console.log(`Finished test ${test.title}: ${result.status}`);
16  }
17
18  onEnd(result: FullResult) {
19    console.log(`Finished the run: ${result.status}`);
20  }
21 }
22
23 export default MyReporter;
```

Eigene Reporter

my-awesome-reporter.ts



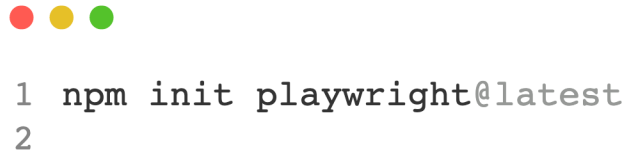
```
1 import { defineConfig } from '@playwright/test';  
2  
3 export default defineConfig({  
4   reporter: './my-awesome-reporter.ts',  
5 });
```



```
1 npx playwright test --reporter="./myreporter/my-awesome-reporter.ts"
```

1. Challenge

Die erste Challenge ist im Bereich Projekte einen neuen Ordner zu erstellen: esolutions.de und darin folgendes vorzubereiten:



```
1 npm init playwright@latest  
2
```

Darin bitte folgende Struktur erstellen:

tests/page

tests/flow

2. Challenge [page]

Wir wollen die Startseite abtesten. Gerne darf auch Codegen verwendet werden.

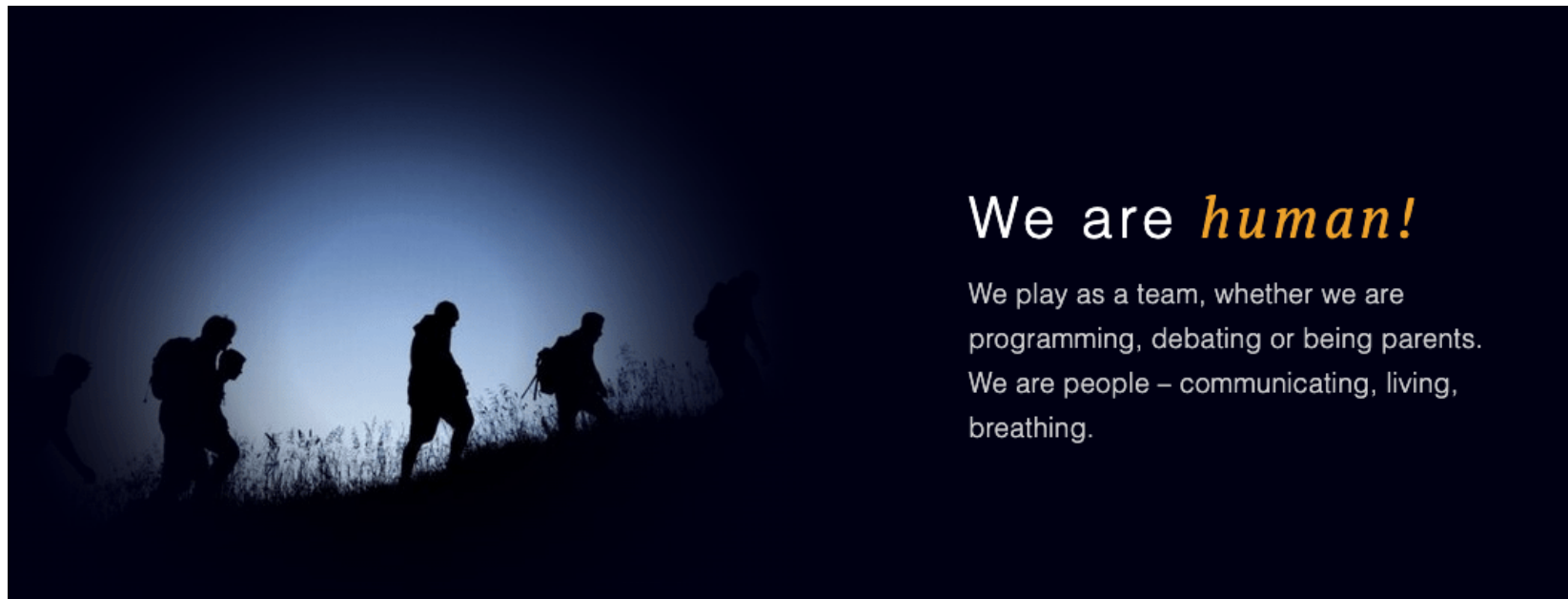
Wir wollen Audi + Text, Porsche + Text und Bentley + Text abtesten.
Zusätzlich wollen wir noch die Bilder testen:

- Haben einen alt-text
- sind verfügbar

Bonus: die Reihenfolge ist richtig :-)

3. Challenge [flow]

Wir schreiben einen Test, der von der Startseite aus folgenden Text findet und einen Screenshot erstellt im Report:



4. Challenge [flow]

Wir schreiben einen Test, der von der Startseite aus folgendes Bild findet:

Wichtig, dabei müssen die "clicks" Verwendet werden und der News-Artikel geöffnet.

**expect ist, wie viele
Mitarbeiter
sind es 2009 in der News?**

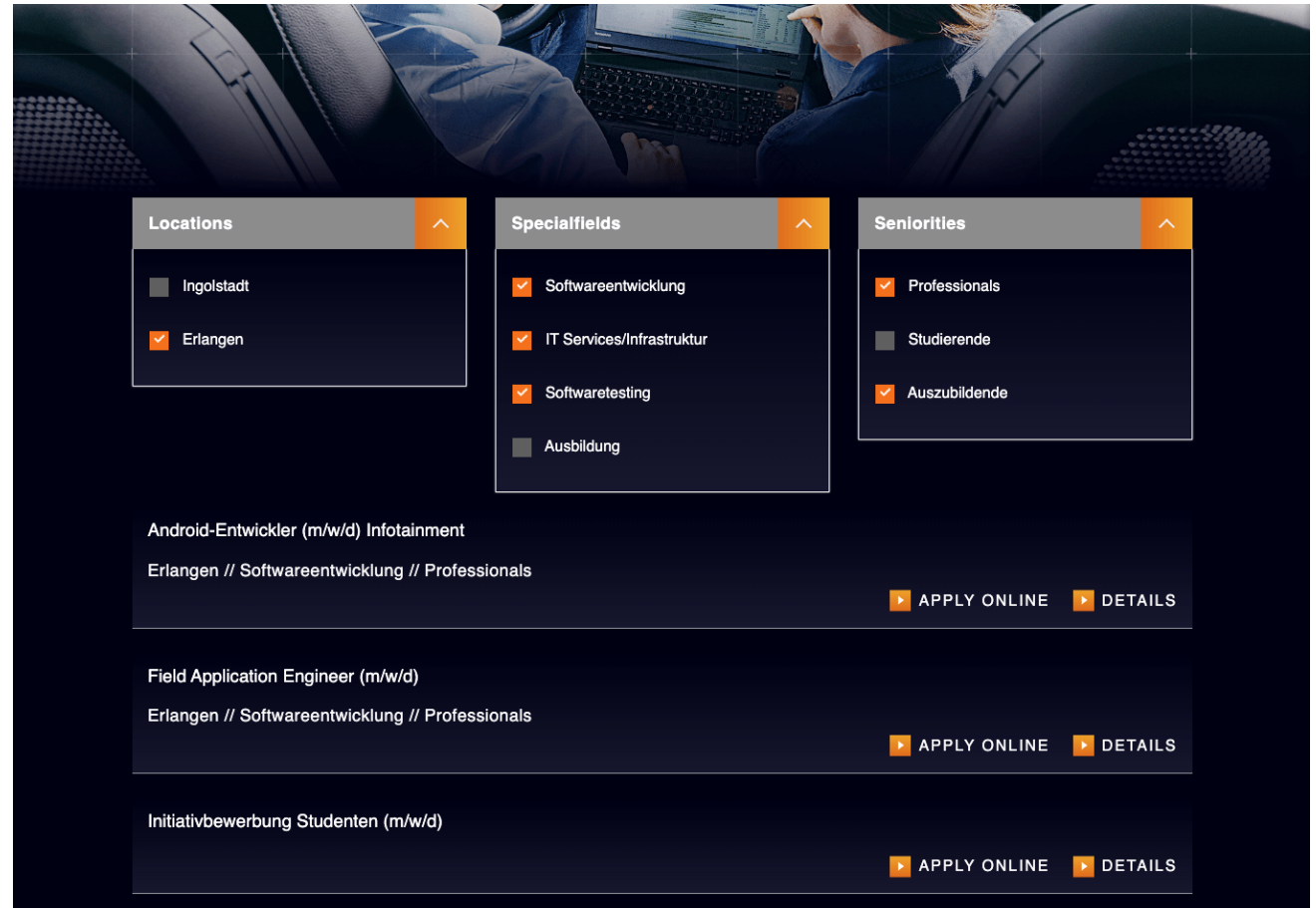
Business
milestones



5. Challenge [flow]

Nun sind wir auf Jobsuche:

von der Startseite aus.



6. Challenge [flow]

Secmail?!

User login

E-mail:

Password:

➔ Login

[Forgot your password?](#)

Login und weiter falls möglich?

6. Challenge [flow]

Secmail?!

User login

E-mail:

Password:

➔ Login

[Forgot your password?](#)

Login und weiter falls möglich?

7. Challenge [flow]

Bewerbung (ohne absenden...) - via Flow

<https://esolutions.onlyfy.jobs/application/de/apply/ed986ulciq70thd4d2l2z269vuxr1g>

Einfach bewerben

• Bewerbung aktuell möglich

Field Application Engineer (m/w/d)

e.solutions GmbH

Erlangen

Persönliche Angaben

Ihre Kontaktdaten

Anrede *

Anrede wählen... ▾

Vorname *

8. eigene Challenge

Weitere Seiten

Andere Webseiten

Weitere Tests?

Ende

Das war alles für dieses Kapitel
