JS Unit Testing

Lernziele

- [ ] Verstehen, was Unit Testing ist und wie es sich zu anderen Testmethoden verhält
- [ ] Wissen, wie man einen Unit Test schreibt, der die Ausgabe einer Funktion überprüft
- [ ] Verstehen, was Test Driven Development ist und wie es verwendet wird
- [ ] Wissen, wie man Unit Tests lokal ausführt (über die Kommandozeile)

Was sind Unit Tests

Bei der Entwicklung von Apps treten unvermeidlich Fehler im Code auf. Deshalb müssen Apps regelmäßig getestet werden, um sicherzustellen, dass sie wie erwartet funktionieren. Da manuelles Testen durch Klicken in der Benutzeroberfläche sehr zeitaufwändig und mühsam ist, werden Tests automatisiert.

Die Testautomatisierung kann auf verschiedene Weise durchgeführt werden. Eine gute Klassifizierung bietet die Testing Trophy.

Statisches Testen bezieht sich auf Linting und kann als eine Art Rechtschreibprüfung für deinen Code verstanden werden.

Unit Tests prüfen, ob eine einzelne Funktion wie vorgesehen funktioniert. Das Ziel ist es, jede einzelne Einheit unabhängig und isoliert von anderen Daten und äußeren Einflüssen zu testen. Der Test kann nach jeder Änderung automatisch ausgeführt werden, um sicherzustellen, dass die neuesten Änderungen die App nicht kaputt machen.

Test Driven Development (TDD)

Beim Test Driven Development (TTD) wird der Test zuerst geschrieben. Danach schreibst du die Funktion, die das gewünschte Ergebnis erzeugen soll.

Mit diesem Ansatz erhältst du so früh wie möglich Feedback, ob deine Arbeit in die richtige Richtung geht. Zu Beginn werden alle Tests fehlschlagen. Nach und nach werden immer mehr Tests erfolgreich sein. Die Implementierung deiner Funktion ist abgeschlossen, sobald alle Testläufe erfolgreich sind.

Wie man mit Jest testet

Tests werden in einer Datei neben dem Code platziert, den du testen möchtest, jedoch mit `.test.js` als Dateiendung.

```
calculator.js           <--- Code, der in deiner App verwendet wird
calculator.test.js      <--- Tests für diesen Code
```

Beim Schreiben von Tests erstellst du verschiedene Szenarien, die ein bestimmtes gewünschtes Ergebnis einer Funktion überprüfen. Jeder dieser Testfälle wird in eine Funktion namens test() eingebettet. Das erste Argument der `test()` Funktion ist eine Beschreibung des Testfalls in einfachem Englisch.

Unit Tests haben in der Regel eine ähnliche Struktur. Zuerst wird die zu testende Funktion aufgerufen und die erforderlichen Argumente übergeben. Danach wird das Ergebnis dieses Funktionsaufrufs an die expect() Funktion übergeben. Auf diese Weise können verschiedene matcher Funktionen wie `toBe()` oder `toEqual()` verwendet werden.

So wird das tatsächlich erzeugte Ergebnis der Funktion mit einem im Testfall definierten erwarteten Ergebnis verglichen.

```
import { add } from "calculator";

test("addiert die Zahlen 1, 2 und 3 korrekt", () => {
  const result = add(1, 2, 3);
  expect(result).toBe(6);
});

test("addiert die Zahlen 13, 28 und 42 korrekt", () => {
  const result = add(13, 28, 42);
  expect(result).toBe(83);
});
```

Tests lokal ausführen

Alle Tests ausführen:

```
npm run test
```

Beim Ausführen dieses Befehls siehst du das Ergebnis des Testlaufs: eine Liste aller enthaltenen Tests und ob sie erfolgreich waren oder fehlgeschlagen sind.

Ressourcen

- Jest
- Testing Trophy and Testing Classifications