# LU08c - Chai-Assertions in Bruno (BDD & TDD)



Bruno nutzt die Chai-Bibliothek – du kannst also dieselbe Syntax wie in Chai für Assertions in deinen Tests verwenden. Zusätzlich bietet Bruno neben JS-Tests auch **deklarative Assertions** im eigenen Reiter.

## **Tests in Bruno**

Bruno kennt zwei Wege für automatische Prüfungen:

- Assertions-Reiter (No-Code/Low-Code): Bedingungen per Ausdruck/Operator/Wert definieren (z. B. response.status equals 200).
- Tests (JavaScript): Frei mit Chai formulieren ideal für komplexe Logik, Schleifen, dynamische Vergleiche etc.

Typischer Ablauf: Request ausführen → im Reiter Tests oder Assertions die Bedingungen definieren → erneut ausführen → Bruno zeigt Pass/Fail pro Bedingung.

## Chai-Assertion-Stile

Chai bietet drei Stile für Assertions:

- BDD: expect (...) und should (lesbar, "natürliche Sprache")
- **TDD**: assert (...) (klassisch, funktionsorientiert)

In Bruno ist expect der gebräuchlichste Stil in Beispielen und Doku. assert ist ebenso möglich. should existiert, wird aber selten genutzt.

## **BDD-Stil (empfohlen)**

Gut für verhaltensorientierte Tests, die API-Verhalten in natürlicher Sprache ausdrücken.

```
test("Status ist 200", () => {
  expect(res.getStatus()).to.equal(200);
});

test("Body enthält erwartete Felder", () => {
```

```
const body = res.getBody();
 expect(body).to.have.property("title");
 expect(body).to.have.property("author");
});
test("Liste enthält genau 3 Bücher", () => {
  const list = res.getBody();
 expect(Array.isArray(list)).to.equal(true);
 expect(list.length).to.equal(3);
});
test("Buch entspricht erwarteten Werten", () => {
  const book = res.getBody();
 expect(book).to.eql({
   title: "The Winds of Winter",
    author: "George R R Martin"
 });
});
```

## **TDD-Stil (assert)**

Kurz und präzise - beliebt in klassischen Unit-Tests.

```
test("Status ist 200 (assert)", () => {
  assert.equal(res.getStatus(), 200);
});

test("Body ist Array (assert)", () => {
  const body = res.getBody();
  assert.isArray(body, "Antwort ist kein Array");
  assert.equal(body.length, 3, "Erwartete Länge 3");
});
```

# "should"-Stil

Der should-Stil ist in Chai vorhanden, wird in Bruno aber kaum genutzt. Falls verwendet, muss er initialisiert werden. Im Unterricht empfehlen wir expect.

## **BDD oder TDD - was wann?**

Nutze **BDD** (expect), wenn ...

https://wiki.bzz.ch/ Printed on 2025/11/20 02:47

- Anforderungen lesbar dokumentiert werden sollen (z. B. im Unterricht oder Review mit Nicht-Entwicklern).
- API-Verhalten beschrieben wird (Wenn ich X aufrufe, erwarte ich Y).
- deklarative Assertions im UI ergänzt werden.

Nutze **TDD** (assert), wenn ...

- du sehr präzise, knappe Checks bevorzugst.
- du Unit-Test-Denke gewohnt bist.
- bestehende assert-Snippets nach Bruno überträgst.

**Faustregel:** Für Unterricht und API-Tests ist BDD/expect der Standard. assert eignet sich für Entwickler, die mit TDD vertraut sind.

# Häufige Assertions in Bruno

#### Header prüfen

```
test("Content-Type ist JSON", () => {
  const ct = res.getHeader("content-type");
  expect(ct).to.contain("application/json");
});
```

### Teilinhalt prüfen

```
test("Titel ist nicht leer", () => {
  const b = res.getBody();
  expect(b.title).to.be.a("string").and.not.empty;
});
```

#### Antwortzeit prüfen

```
test("Antwort < 800 ms", () => {
  expect(res.getResponseTime()).to.be.below(800);
});
```

# **Typische Stolpersteine**

- equal vs. eql: Für komplexe Strukturen (Objekte/Arrays) eql oder deep.equal nutzen.
- Kontext Tests vs. Script: Chai ist nur im Tests-Kontext garantiert verfügbar.
- Parsing: res.getBody() liefert bei JSON automatisch ein Objekt, sonst einen String.

## Weiterführende Links

- Bruno Docs: Tests (Chai)
- Bruno Docs: Assertions-Reiter
- Bruno Docs: JavaScript-API (res, bru, expect)
- Chai Assertion Styles (BDD & TDD)

#### M450-LU08



From:

https://wiki.bzz.ch/ - BZZ - Modulwiki

Permanent link:

https://wiki.bzz.ch/modul/m450/learningunits/lu08/chaibruno

Last update: 2025/09/18 09:52



https://wiki.bzz.ch/ Printed on 2025/11/20 02:47