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

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

Einordnung: 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 kurz erklärt

Chai bietet drei Stile für Assertions:

- **BDD:** `expect(...)` und `should` (lesbare, „natürliche“ Sprache)
- **TDD:** `assert(...)` (klassisch, funktionsorientiert)

In Bruno ist **`expect`** der gebräuchlichste Stil in Beispielen und Doku. `assert` ist ebenso möglich. `should` existiert in Chai, wird in Bruno jedoch seltener verwendet.

BDD-Stil (empfohlen für API-Spezifikationen)

Lesbarkeit vor Kürze: Gut für „verhaltensorientierte“ Checks (Antwortstatus, Struktur, Werte).

Beispiele Statuscode

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

Body-Struktur (Objekt)

```
test("Body enthält erwartete Felder", () => {
  const body = res.getBody(); // JSON wird automatisch geparst, falls
Content-Type JSON
  expect(body).to.have.property("title");
  expect(body).to.have.property("author");
});
```

Array-Länge & Teilmenge

```
test("Liste enthält genau 3 Bücher", () => {
  const list = res.getBody();
  expect(Array.isArray(list)).to.equal(true);
  expect(list.length).to.equal(3);
});
```

Deep Equality (Objekte/Arrays)

```
test("Buch entspricht dem erwarteten Objekt", () => {
  const book = res.getBody();
  expect(book).to.eql({
    title: "The Winds of Winter",
    author: "George R R Martin"
  });
});
```

> **Tipp:** Für tiefe Vergleiche nutze `eql`/`deep.equal`; für Primitive reicht `equal`.

TDD-Stil (assert)

Kürze vor Lesbarkeit: Prägnant, beliebt in Unit-Tests – auch in Bruno nutzbar, wenn du TDD gewohnt bist.

Beispiele

```
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 (optional)

Der `should`-Stil von Chai existiert, wird in Bruno jedoch nicht aktiv beworben. Wenn du ihn verwendest, achte darauf, dass er initialisiert ist; in der Praxis empfiehlt sich in Bruno **`expect`**.

BDD vs. TDD – wann was?

Nutze BDD (`expect`), wenn …

- Anforderungen fachlich/lesbar dokumentiert werden sollen (z. B. im Unterricht, Review mit Nicht-Entwicklern).
- du API-Verhalten/Szenarien formulierst („Wenn ich X aufrufe, **erwarte** ich Y“).
- du deklarative **Assertions** im UI mit wenigen Klicks ergänzen willst (Status, Header, Teilinhalte).

Nutze TDD (`assert`), wenn …

- du aus Unit-Test-Denke kommst und kurze, funktionale Checks bevorzugst.
- du sehr **präzise** Vergleiche und klassische Assertions magst (z. B. `assert.strictEqual`).
- du bestehende TDD-Snippets (aus anderen Projekten) nach Bruno überträgst.

> **Faustregel in Bruno:** Für Unterricht & Teams mit gemischter Erfahrung → **BDD/`expect`** (+ Assertions-Reiter). Für Entwickler-zentrierte, knappe Checks → **TDD/`assert`**.

Häufige API-Assertions in Bruno (Mini-Rezeptesammlung)

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 (einfacher Leistungs-Check)

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

Typische Stolpersteine

- **`equal` vs. `eql`**: Für komplexe Strukturen (Objekte/Arrays) **`eql`/`deep.equal`** nutzen, nicht `equal`.
- **Kontext „Tests“ vs. „Script“**: Chai-Hilfen sind im **Tests**-Kontext standardmäßig verfügbar; im **Script**-Kontext kann es je nach Version abweichen → Tests-Tab verwenden.
- **Parsing**: `res.getBody()` liefert bei JSON-Antworten ein bereits geparstes Objekt (ansonsten String) – prüfe ggf. Content-Type.

Weiterführende Links

- **Bruno – Testing (Einführung & Beispiele)**

– Assertions & `expect(res.getStatus()).to.equal(200)` u. a.

- **Bruno – Assertions (No-Code Assertions-Reiter)**
- **Bruno – JavaScript/Response-API (z. B. `res.getBody()`, `getHeader()`)**
- **Chai – Assertion-Stile** (BDD `expect`/`should`, TDD `assert`)

M450-LU08