```
ect(response.data.user.name.first).toBe("Uncle");
                                                 expect(response.data.user.name.first).toBe("Uncle
ect(response.data.user.name.last).toBe("Phil");
                                                  expect(response.data.user.name.last).toBe("Phil")
ect(response.data.user.role).toBe("parent");
                                                 expect(response.data.user.role).toBe("parent");
ect(response.data.user.name.first).toBe("Uncle"
                                               expect(response.data.user.name.first).toBe("Uncle")
ect(response.data.user.name.last).toBe("Phil");
                                                expect(response.data.user.name.last).toBe("Phil");
ect(response.data.user.role).toBe("parent");
                                                expect(response.data.user.role).toBe("parent");
expect(response.data.user.name.first).toBe("Uncle expect(response.data.user.name.first).toBe("Uncle
expect(response.data.user.name.last).toBe("Phil")expect(response.data.user.name.last).toBe("Phil")
expect(response.data.user.role).toBe("parent"); expect(response.data.user.role).toBe("parent"):
irst).toBe("Uncle");
                                                                     expect(response.data.user.name
                     import * as BelAir from "../bel-air-academy";
.ast).toBe("Phil");
                                                                     expect(response.data.user.name
toBe("parent");
                                                                     expect(response.data.user.role
                     otest("uncle phil", async () => {
e.first).toBe("Uncle");
                                                                              expect(response.data.u
                         const response = await BelAir.request("philip");
e.last).toBe("Phil");
                                                                              expect(response.data.u
e).toBe("parent");
                         expect(response.data.user.name.first).toBe("Uncle")
                                                                              expect(response.data.u
                         expect(response.data.user.name.last).toBe("Phil")
irst).toBe("Uncle");
                                                                           expect(response.data.use
                         expect(response.data.user.role).toBe("parent");
.ast).toBe("Phil");
                                                                           expect(response.data.use
                     21.
toBe("parent");
                                                                           expect(response.data.use
                     expect(response.data.user.name.first).toBe("Uncle");
irst).toBe("Uncle");
                      expect(response.data.user.name.last).toBe("Phil");
                                                                           expect(response.data.use
.ast).toBe("Phil");
                     expect(response.data.user.role).toBe("parent");
                                                                           expect(response.data.use
toBe("parent");
                                                                           expect(response.data.use
ata.user.name.first).toBe("Uncle");
                                                                 expect(response.data.user.name.fir
                                    Redundant
ata.user.name.last).toBe("Phil");
                                                                expect(response.data.user.name.las
ata.user.role).toBe("parent");
                                                                 expect(response.data.user.role).to
                                                 expect(response.data.user.name.first).toBe("Uncle
pect(response.data.user.name.first).toBe("Uncle")
pect(response.data.user.name.last).toBe("Phil");
                                                 expect(response.data.user.name.last).toBe("Phil")
pect(response.data.user.role).toBe("parent");
                                                  expect(response.data.user.role).toBe("parent");
```

```
import * as BelAir from "../bel-air-academy";

etest("uncle phil", async () => {
    const response = await BelAir.request("philip");

    expect(response.data.user.name.first).toBe("Uncle");
    expect(response.data.user.name.last).toBe("Phil");
    expect(response.data.user.role).toBe("parent");
});
```

## Generic error messages