```
ect(response.data.user.name.last).toBe("Phil");
ect(response.data.user.role).toBe("parent");
                                                                                                                                                expect(response.data.user.name.first).toBe("Uncle
                                                                                                                                                expect(response.data.user.name.last).toBe("Phil")
                                                                                                                                                expect(response.data.user.role).toBe("parent");
                                                              import * as BelAir from "../bel-air-academy";
                                                             otest("uncle phil", async () => {
                                                                                                                                                                                                                                 expect(response.data.u
                                                                         const response = await BelAir.request("philip");
                                                                                                                                                                                                                                 expect(response.data.u
                                                                         expect(response.data.user.name.first).toBe("Uncle") expect(response.data.user.name.data.user.name.first).toBe("Uncle") expect(response.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.user.name.data.
                                                                         expect(response.data.user.name.last).toBe("Phil");
irst).toBe("Uncle");
                                                                         expect(response.data.user.role).toBe("parent");
.ast).toBe("Phil");
                                                              11.
toBe("parent");
                                                               expect(response.data.user.name.first).toBe("Uncle");
irst).toBe("Uncle");
                                                               expect(response.data.user.name.last).toBe("Phil");
.ast).toBe("Phil");
                                                               expect(response.data.user.role).toBe("parent");
toBe("parent");
                                                                                                                                                                                           expect(response.data.user.name.fir
                                                                                                       Redundant
                                                                                                                                                                                           expect(response.data.user.name.las
                                                                                                                                                                                           expect(response.data.user.role).to
pect(response.data.user.name.first).toBe("Uncle");
```

ect(response.data.user.name.first).toBe("Uncle");

pect(response.data.user.name.last).toBe("Phil");

pect(response.data.user.role).toBe("parent");

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

expect(response.data.user.role).to

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