Angular Labs: End-to-End Testing with Cypress

Exercises

0. Add Cypress Run Script

Go to your package.json and add this run script.

```
"demo": "nx e2e flight-app-e2e --watch",
```

1. Add a sanity check

Clean up the existing file `app.spec.ts` file in apps/flight-app-e2e/src/integration and add the following first test (you may need the first line comment to keep your IDE calm).

```
/// <reference types="cypress" />

describe('flight-app', () => {
  beforeEach(() => cy.visit('/'));

  it('should do a sanity check', () => {
    cy.visit('/');
  });
});
```

Check if everything is okay by running cypress in watch mode.

```
npm run demo / yarn run demo
```

If the test passes good, else please contact me before you continue.

2. Now check the document encoding

Use cy.document to retrieve document information.

```
it('should have UTF-8 as charset', () => {
  cy.document().should('have.property', 'charset').and('eq', 'UTF-8')
});
```

Test your test by running cypress again.

3. Make an implicit Subject Assertion

We check if the last sidebar list item contains the text "Basket".

```
it('should do an implicit subject assertion', () => {
  cy.get('.sidebar-wrapper ul.nav li:last a').should('contain.text', 'Basket');
});
```

4. Test via an explicit Subject Assertion

We check the second list item now. It should contain "Flights". This time we also check the link.

```
it('should do an explicit subject assertion', () => {
  cy.get('.sidebar-wrapper ul.nav li:nth-child(2) a').should(($a) => {
    expect($a).to.contain.text('Flights');
    expect($a).to.have.attr('href', '/flight-booking/flight-search');
  });
});
```

5. Count the listed nav links

Count the listed sidebar nav links.

```
it('should count the nav entries', () => {
  cy.get('.sidebar-wrapper ul.nav li').should('have.length', '3');
});
```

6. Now we check our flight-search component

A click on search should get use three result flights.

```
it.only('should display 3 flights on search', () => {
  cy.get('flight-search button:first').contains('Search').click();

  cy.get('div.row > div').should('have.length', '3');
});
```

7. We try remove the first flight from the basket

Click on "Remove" should update the basket.

```
it.only('should remove flight from basket', () => {
  cy.get('flight-search button:first').contains('Search').click();

  cy.get('div.row > div:first button:first').contains('Remove').click();

  cy.get('flight-search div.card:last').contains('"3": false');
});
```

Finishing up

Now remove all ".only" from your tests and run cypress again. The result should be 7 passing tests.

Bonus

Ensure that the web application loads within a second

```
it('should load page below 1 second', () => {
  cy.visit('/', {
    onBeforeLoad: (win) => {
      win.performance.mark('start-loading');
    },
    onLoad: (win) => {
      win.performance.mark('end-loading');
    }
  })
    .its('performance')
    .then((p) => {
      p.measure('pageLoad', 'start-loading', 'end-loading');
      const measure = p.getEntriesByName('pageLoad')[0];
      expect(measure.duration).to.be.most(1000);
    });
});
```

Split up tests

Split up the test into 3 different spec files:

- app.spec.ts
- flight-search.spec.ts
- sidebar.spec.ts

```
npm run demo
```

With some luck you can directly jump from Cypress to the spec files in your IDE :-)

Think about something else to test and implement it

Here is your reference: https://docs.cypress.io/guides/references/assertions