Lab Testing

Exercise 1: Test your component

You have two components that use your contactsService. We'll test InviteComponent and we'll leave out dependency injection for now.

1. In *app/+invite/*, create `invite.component.spec.ts`
2. Import your component, its dependencies and the test dependencies

```
import { beforeEach, describe, expect, it } from '@angular/core/testing';
import { InviteComponent } from './invite.component';
import { ContactsService } from './contacts.service';
```

3. Add a describe() with a beforeEach() and an it()

```
describe('InviteComponent', () => {
    let contactsService: ContactsService;
    let inviteComponent: InviteComponent;

    beforeEach(() => {

    });

    it('should retrieve a list of contacts', () => {

    });
});
```

4. In beforeEach(), instantiate the component and dependencies. Spy on the getContacts() function, we want to know whether this will be called

```
beforeEach(() => {
    contactsService = new ContactsService();
    spyOn(contactsService, 'getContacts');
});
```

5. In it(), instantiate the component and check whether contacts are being retrieved

```
it('should retrieve a list of contacts', () => {
    inviteCompoment = new InviteCompoment(contactsService);
    inviteComponent.ngOnInit();
    expect(contactsService.getContacts).toHaveBeenCalled();
});
```

Exercise 2: Dependency injection

1. import the necessary items for DI

```
import { provide } from '@angular/core';
import { beforeEach, describe, expect, it, inject } from
'@angular/core/testing';
import { TestComponentBuilder } from '@angular/compiler/testing';
```

2. Create a mock class representative of your service

```
class MockContactsService extends NavigateService {
    constructor() { super(null); }
}
```

3. In beforeEach(), create an instance of your mock class:

```
beforeEach(() => {
    mockContactsService = new MockContactsService();
    spyOn(mockContactsService, 'getContacts');
});
```

4. inject() the TestComponentBuilder, override the providers, let the builder instantiate your component and test whether the contacts have been retrieved

```
it('should retrieve a list of contacts',
    inject([TestComponentBuilder], (tcb: TestComponentBuilder) => {
    tcb.overrideProviders(InviteComponent, [
            provide(ContactsService, { useValue: mockContactsService })
        ])
        .createAsync(InviteComponent)
        .then(fixture => {
            expect(mockContactsService.getContacts).toHaveBeenCalled();
        });
}));
```

Exercise 3: Test your pipe

You have a pipe that transforms a Contact to a view-friendly string. Test this.