

```
expect(addressForm.formRow(3).staticValue()).toContain('Postal code');  
expect(addressForm.formRow(4).label()).toContain('City');  
expect(addressForm.formRow(4).staticValue()).toContain('-');  
});
```

```
it('Should show an error when getting the address data fails', function() {
```

```
    browser.get(pageurl + '&getAddressResponse=ERROR');
```

```
    expect(addressForm.errorMessage().isDisplayed()).toBeTruthy();  
    expect(addressForm.errorMessage().getText()).toEqual('Error retrieving address data!');  
    expect(addressForm.errorMessage().getText()).toContain('Error retrieving address data!');
```

```
});
```

```
});
```

```
describe('Switching between view modes', function() {
```

```
    it('Should switch to edit mode when the \'Edit\' button is clicked', function() {
```

```
        browser.get(pageurl + '');
```

```
        expect(addressForm.btnSave().isPresent()).toBeFalsy();  
        expect(addressForm.btnCancel().isPresent()).toBeFalsy();  
        addressForm.btnEdit().click();
```

```
        expect(addressForm.btnEdit().isPresent()).toBeFalsy();  
        expect(addressForm.btnSave().isPresent()).toBeTruthy();  
        expect(addressForm.btnCancel().isPresent()).toBeTruthy();
```

Contents of today's session

- What is end-to-end testing?
- What tools to use
- Writing tests
- Using page objects
- Live code demo
 - Creating e2e tests for the app we build in the previous session

What is end-to-end testing?

End-to-end testing is a software testing method used to test whether the flow of an application is working as designed from start to finish.

Tools for end-to-end testing



Protractor

end to end testing for AngularJS

- Testing framework specifically created for Angular
- Provides selectors and actions for interacting with HTML elements on the page
- Configured through a js config file
- <http://angular.github.io/protractor/#/>

Tools for end-to-end testing

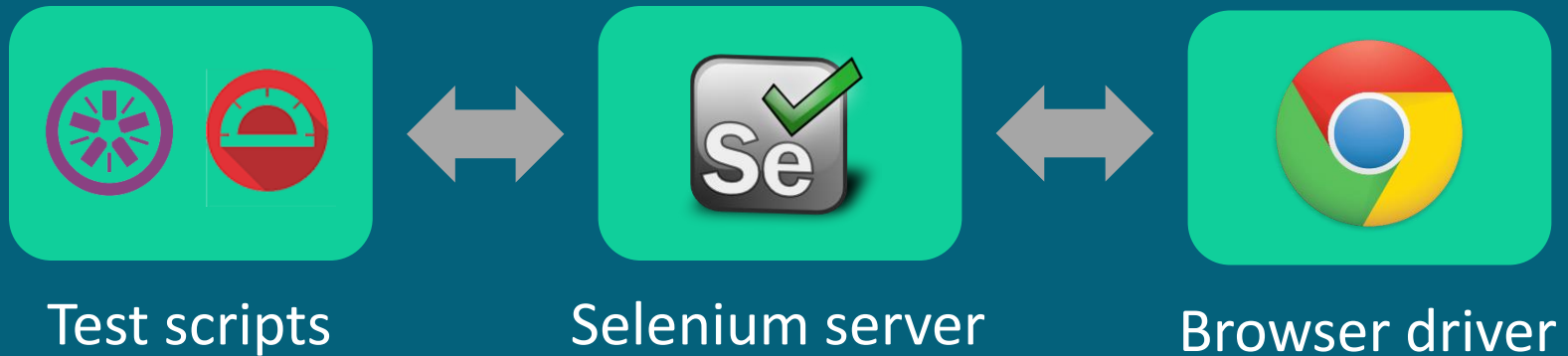


- Browser automation tool
- Performs the actions specified in test specs in the browser

Tools for end-to-end testing



End-to-end test flow



Writing end-to-end tests

Tests are written the same way as unit tests

```
5 describe('The change address form', function() {  
6  
7     it('Should show the form on the page', function() {  
8  
9         // Test & expectations  
10  
11     })  
12  
13 });
```


Writing end-to-end tests

Selectors for elements

```
5 // By xpath
6 element(by.xpath('/body/div/h1'));
7
8 // By CSS
9 element(by.css('body .panel h1'));
10
11 // By ID
12 element(by.id('theId'));
13
14 // By binding
15 element(by.binding('foo'));
16
17 // By model
18 element(by.model('foo'));
```

Writing end-to-end tests

Actions for elements

```
5  var el = element(selector)
6
7  // Click the element
8  el.click();
9
10 // Clear the text in an element
11 el.clear();
12
13 // Send keys to the element
14 el.sendKeys('text to send');
15
16 // Get an attribute of the element
17 el.getAttribute('class');
18 el.getAttribute('value');
```

Writing end-to-end tests

Actions for elements

```
4
5  var el = element(selector)
6
7  // Check if element is present on the page
8  el.isPresent();
9
10 // Check if element is visible
11 el.isDisplayed();
12
13 // Get the text inside an element
14 el.getText();
15
```

Writing end-to-end tests

Example of a spec

```
4  
5 describe('The test page', function() {  
6  
7     it('Should show a panel element with a message inside of it', function() {  
8  
9         browser.get('test.html');  
10  
11         var panel = element(by.css('.panel .panel-default'));  
12  
13         expect(panel.isDisplayed()).toBeTruthy();  
14         expect(panel.getText()).toEqual('This is the text!');  
15  
16     })  
17  
18 });
```

Writing end-to-end tests

Things can get messy...

```
5 describe('The form', function() {
6
7   it('Should display the correct labels and prefilled fields', function() {
8
9     browser.get('form.html');
10
11     expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) label')).getText()).toEqual('Street');
12     expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) input')).getAttribute('value')).toBe('Dorpsstraat');
13
14     expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) label')).getText()).toEqual('House nr');
15     expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) input')).getAttribute('value')).toBe('77');
16
17   });
18
19   it('Should display a validation message when the fields are empty', function() {
20
21     browser.get('form.html');
22
23     expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) .alert.alert-danger'))).not.toBeDisplayed();
24     expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) .alert.alert-danger'))).not.toBeDisplayed();
25
26     element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) input')).clear();
27     element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) input')).clear();
28
29     expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) .alert.alert-danger'))).toBeDisplayed();
30     expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) .alert.alert-danger')).getText())
31       .toEqual('Please enter your street. ');
32     expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) .alert.alert-danger'))).toBeDisplayed();
33     expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) .alert.alert-danger')).getText())
34       .toEqual('Please enter your house number. ');
35
36   });
37
38 });
```

Page objects

A **page object** is a Javascript object that contains the locators and functions for interacting with the application.

```
5  module.exports.formPageObject = function() {
6
7      this.element = function() {
8          return element(by.css('form[name="addressForm"]'));
9      }
10
11     this.formRow = function(eq) {
12
13         var formRow = this.element().element(by.css('.form-row:nth-of-type(' + eq + ')'));
14
15         formRow.label = function() {
16             return formRow.element(by.css('label')).getText();
17         }
18
19         formRow.inputField = function() {
20             var input = formRow.element(by.css('input[type=text]'));
21
22             input.value = function() {
23                 return input.getAttribute('value');
24             }
25
26             return input;
27         }
28
29         formRow.validationMessage = function() {
30             var validationMessage = formRow.element(by.css('.alert.alert-danger'));
31
32             validationMessage.text = function() {
33                 return validationMessage.getText();
34             }
35
36             return validationMessage;
37         }
38
39         return formRow;
40     }
41
42 };

```

Using page objects

Much better!

```
5 var FormPageObject = require('../pageObjects/formPageObject').formPageObject;
6 var addressForm = new FormPageObject();
7
8 describe('The form', function() {
9
10     it('Should display the correct labels and prefilled fields', function() {
11
12         browser.get('form.html');
13
14         expect(addressForm.formRow(1).label()).toEqual('Street');
15         expect(addressForm.formRow(1).inputField().value()).toBe('Dorpsstraat');
16
17         expect(addressForm.formRow(2).label()).toEqual('House nr');
18         expect(addressForm.formRow(2).inputField().value()).toBe('77');
19
20     });
21
22     it('Should display a validation message when the fields are empty', function() {
23
24         browser.get('form.html');
25
26         expect(addressForm.formRow(1).validationMessage()).not.toBeDisplayed();
27         expect(addressForm.formRow(2).validationMessage()).not.toBeDisplayed();
28
29         element(addressForm.formRow(1).inputField()).clear();
30         element(addressForm.formRow(2).inputField()).clear();
31
32         expect(addressForm.formRow(1).validationMessage()).toBeDisplayed();
33         expect(addressForm.formRow(1).validationMessage().text()).toEqual('Please enter your street.');
```


Live coding demo – setting up

<https://github.com/lvandiest/todo-list>

- Protractor config
- Starting selenium
- Running tests
- Writing tests