Automated end-to-end testing

with AngularJS and Protractor

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

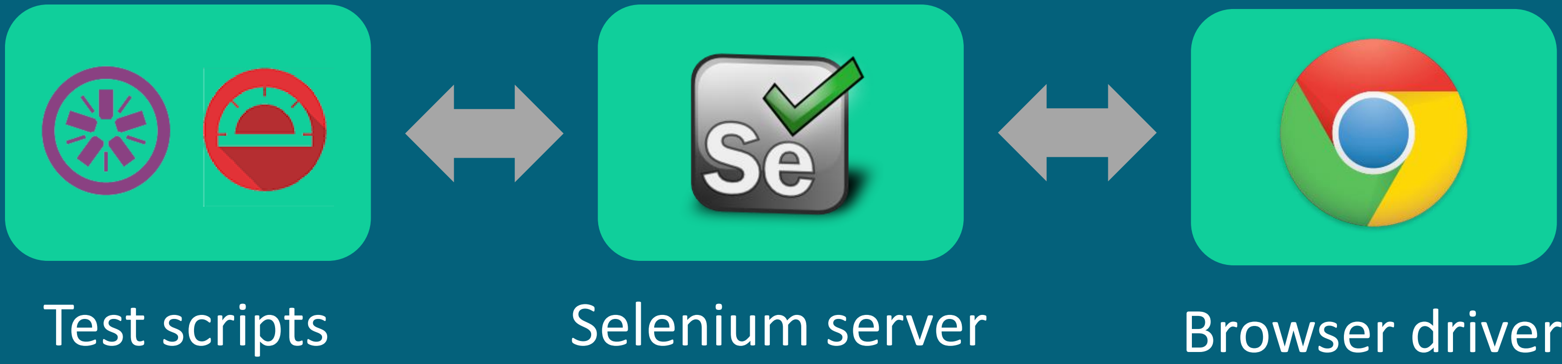
- Browser automation tool
- Performs the actions specified in test specs in the browser

Tools for end-to-end testing

Jasmine

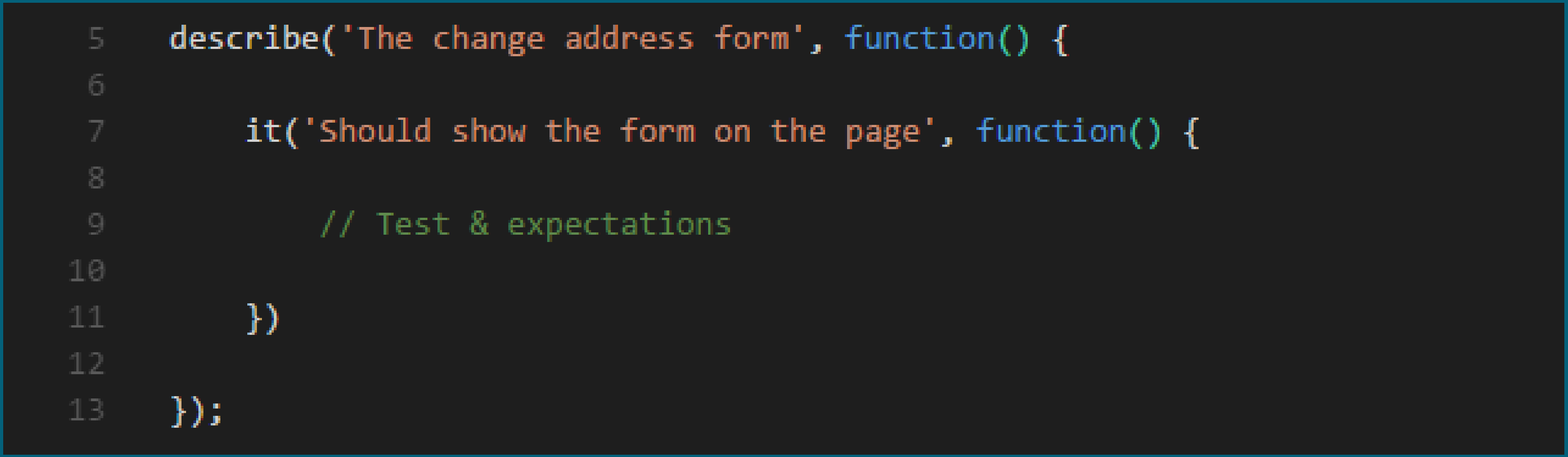
GRUNT

End-to-end test flow



Writing end-to-end tests

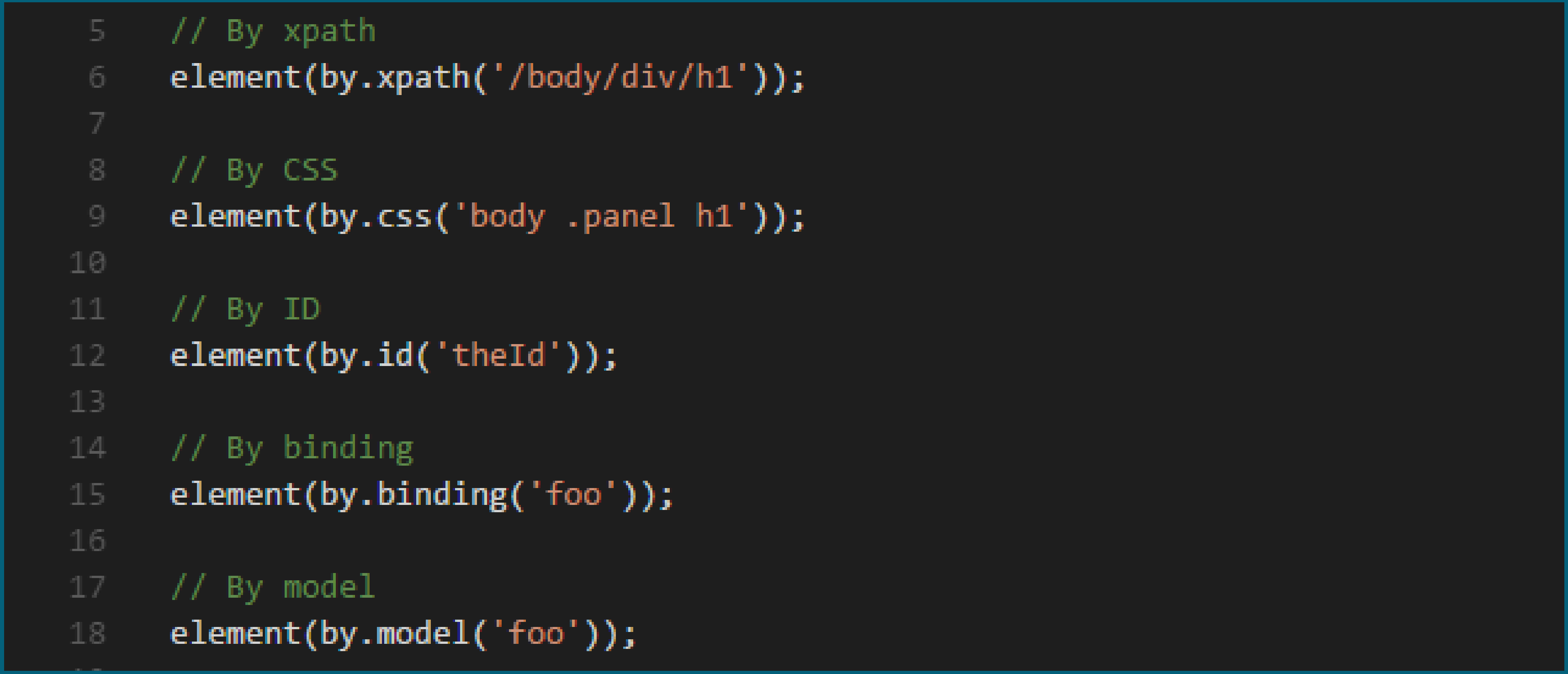
Tests are written the same way as unit tests



```
describe('The change address form', function() {

    it('Should show the form on the page', function() {

        // Test & expectations

    })

});
```

Writing end-to-end tests

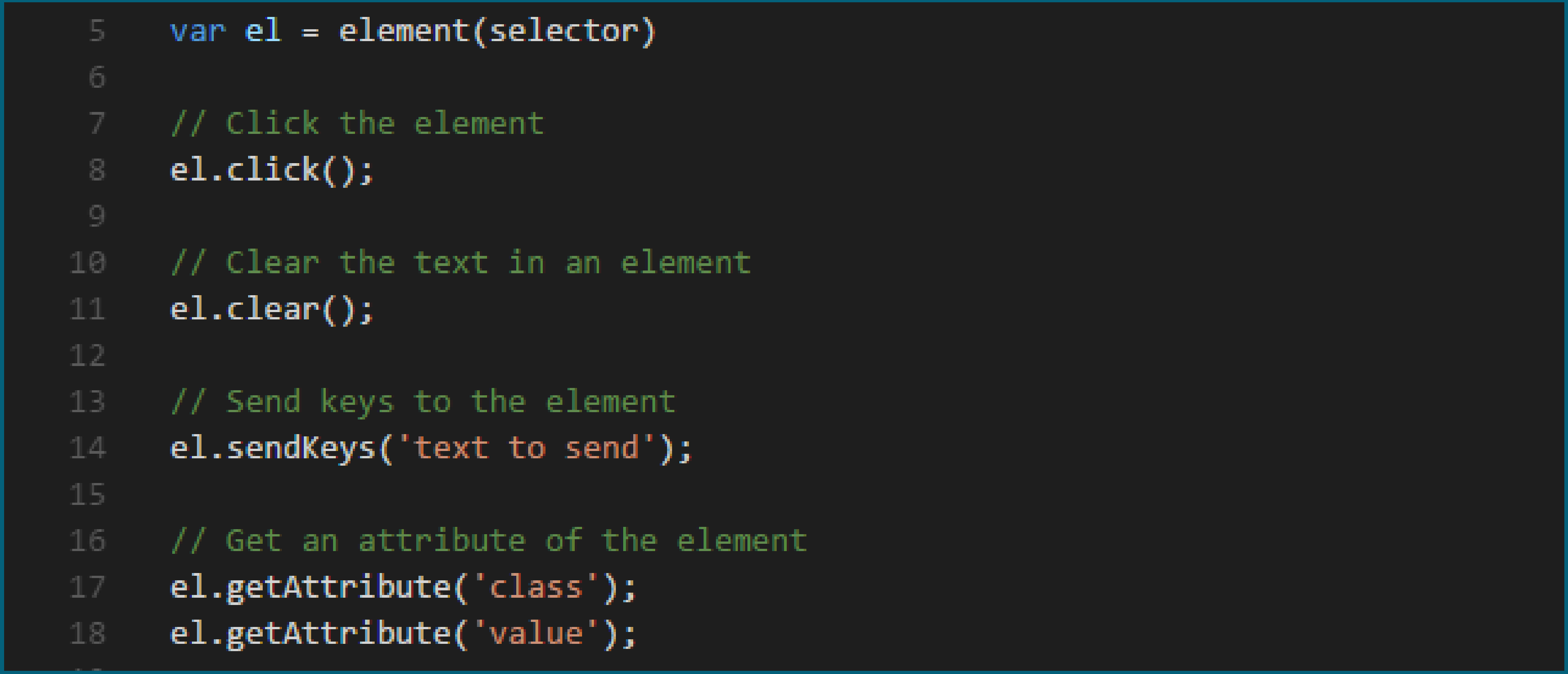
Selectors for elements



```
// By xpath
element(by.xpath('/body/div/h1'));

// By CSS
element(by.css('body .panel h1'));

// By ID
element(by.id('theId'));

// By binding
element(by.binding('foo'));

// By model
element(by.model('foo'));
```

Writing end-to-end tests

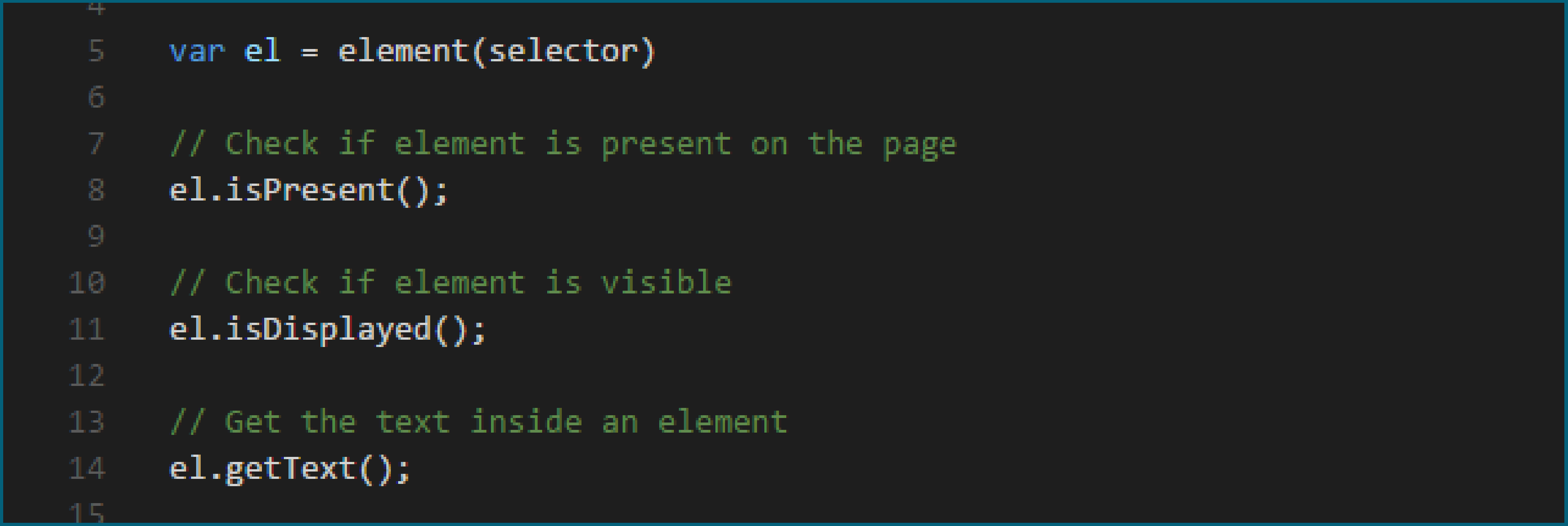
Actions for elements



```javascript
var el = element(selector)

// Click the element
el.click();

// Clear the text in an element
el.clear();

// Send keys to the element
el.sendKeys('text to send');

// Get an attribute of the element
el.getAttribute('class');
el.getAttribute('value');
```

Writing end-to-end tests

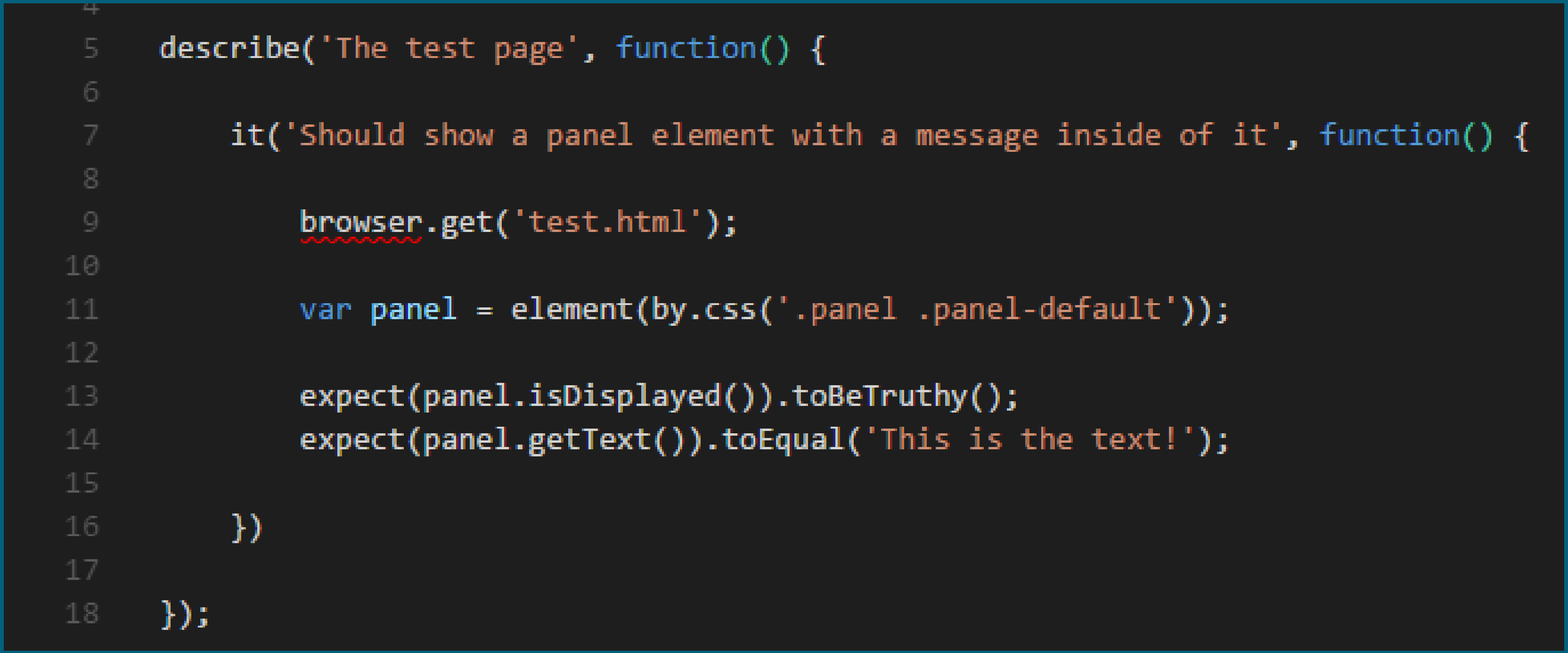
Actions for elements



```
var el = element(selector)

// Check if element is present on the page
el.isPresent();

// Check if element is visible
el.isDisplayed();

// Get the text inside an element
el.getText();
```

Writing end-to-end tests

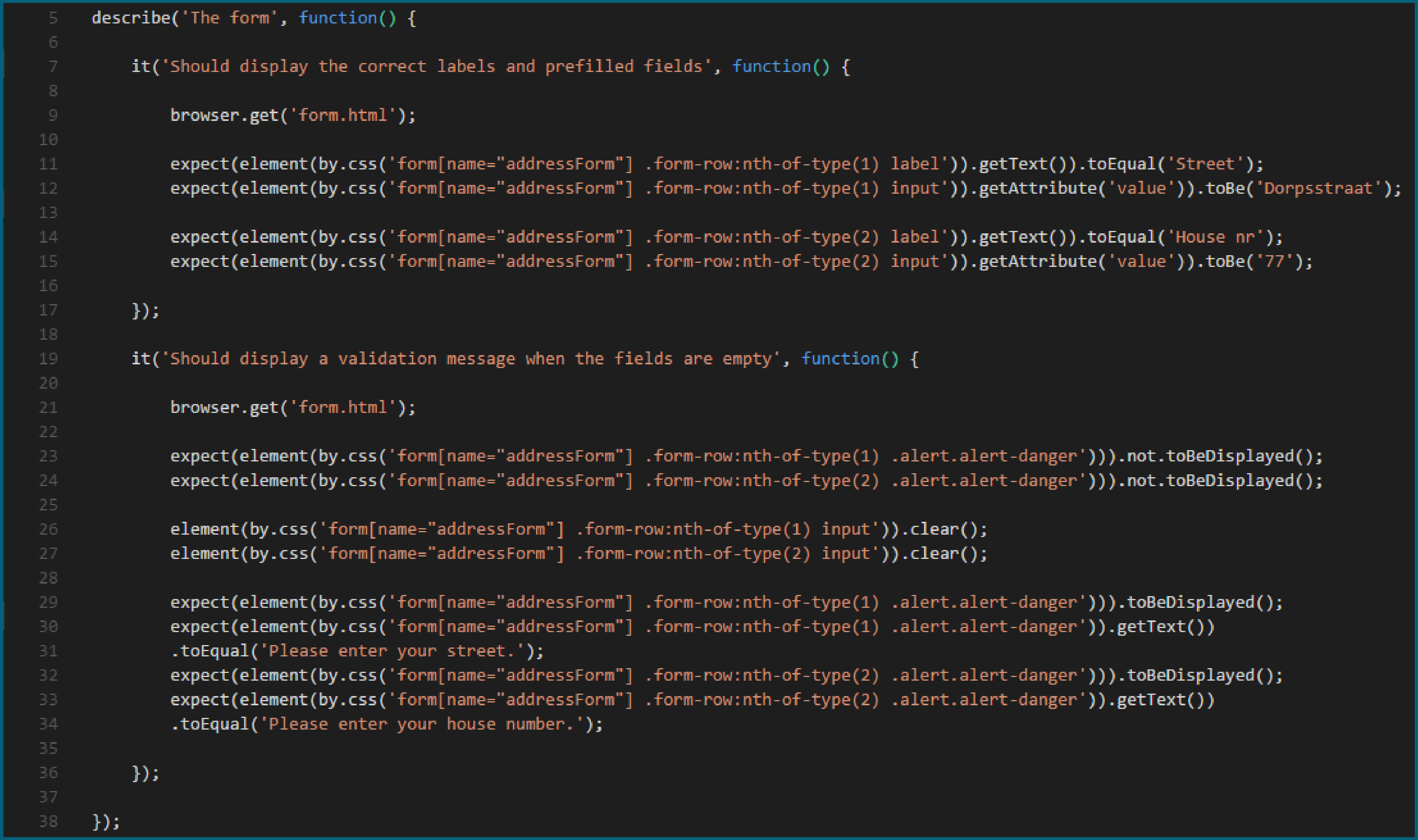
Example of a spec



```
describe('The test page', function() {

    it('Should show a panel element with a message inside of it', function() {

        browser.get('test.html');

        var panel = element(by.css('.panel .panel-default'));

        expect(panel.isDisplayed()).toBeTruthy();
        expect(panel.getText()).toEqual('This is the text!');

    })

});
```

Writing end-to-end tests

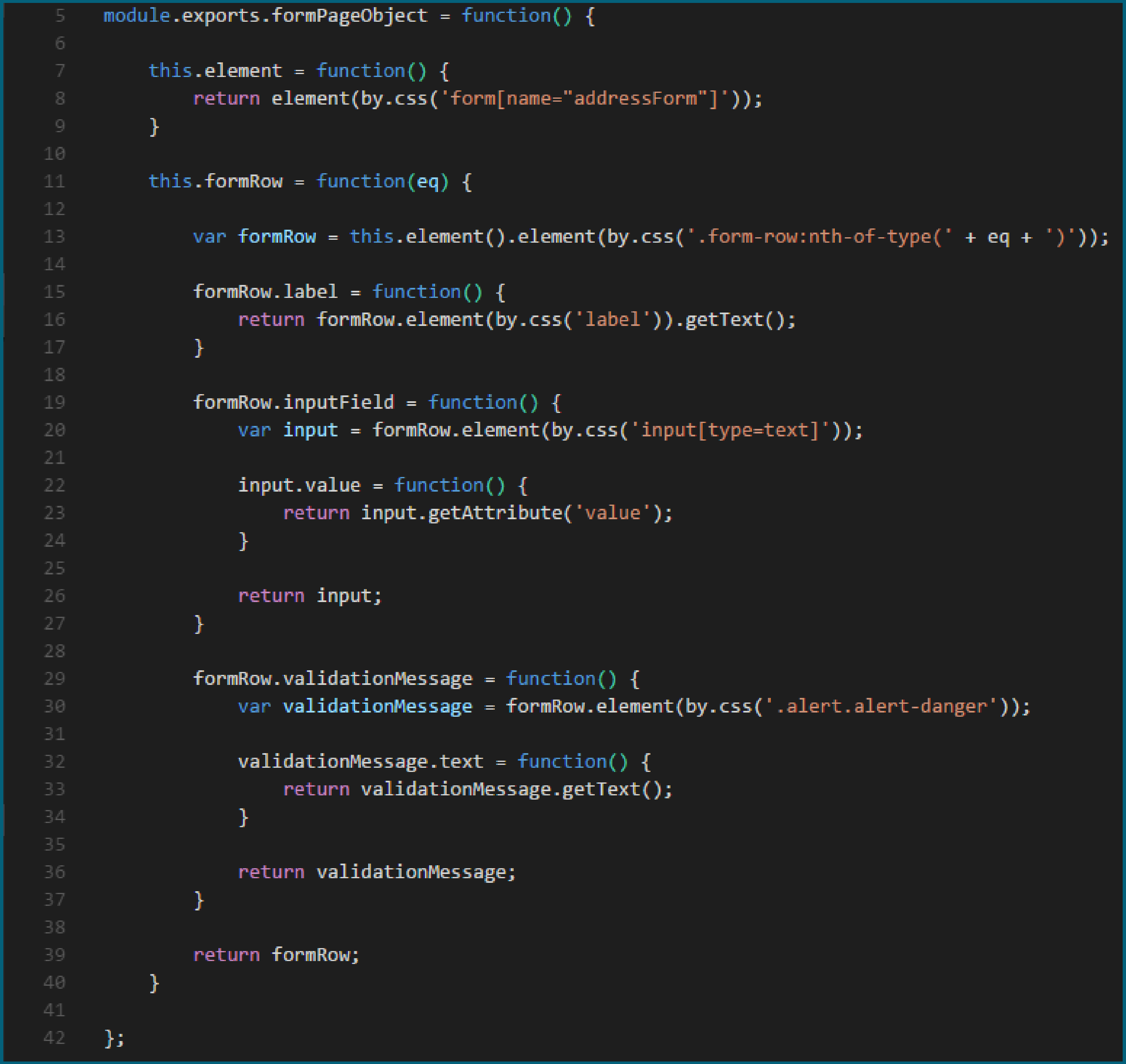
Things can get messy…



```
describe('The form', function() {

    it('Should display the correct labels and prefilled fields', function() {

        browser.get('form.html');

        expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) label')).getText()).toEqual('Street');
        expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) input')).getAttribute('value')).toBe('Dorpsstraat');

        expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) label')).getText()).toEqual('House nr');
        expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) input')).getAttribute('value')).toBe('77');

    });

    it('Should display a validation message when the fields are empty', function() {

        browser.get('form.html');

        expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) .alert.alert-danger'))).not.toBeDisplayed();
        expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) .alert.alert-danger'))).not.toBeDisplayed();

        element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) input')).clear();
        element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) input')).clear();

        expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) .alert.alert-danger'))).toBeDisplayed();
        expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(1) .alert.alert-danger')).getText())
        .toEqual('Please enter your street.');
        expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) .alert.alert-danger'))).toBeDisplayed();
        expect(element(by.css('form[name="addressForm"] .form-row:nth-of-type(2) .alert.alert-danger')).getText())
        .toEqual('Please enter your house number.');

    });

});
```

Page objects

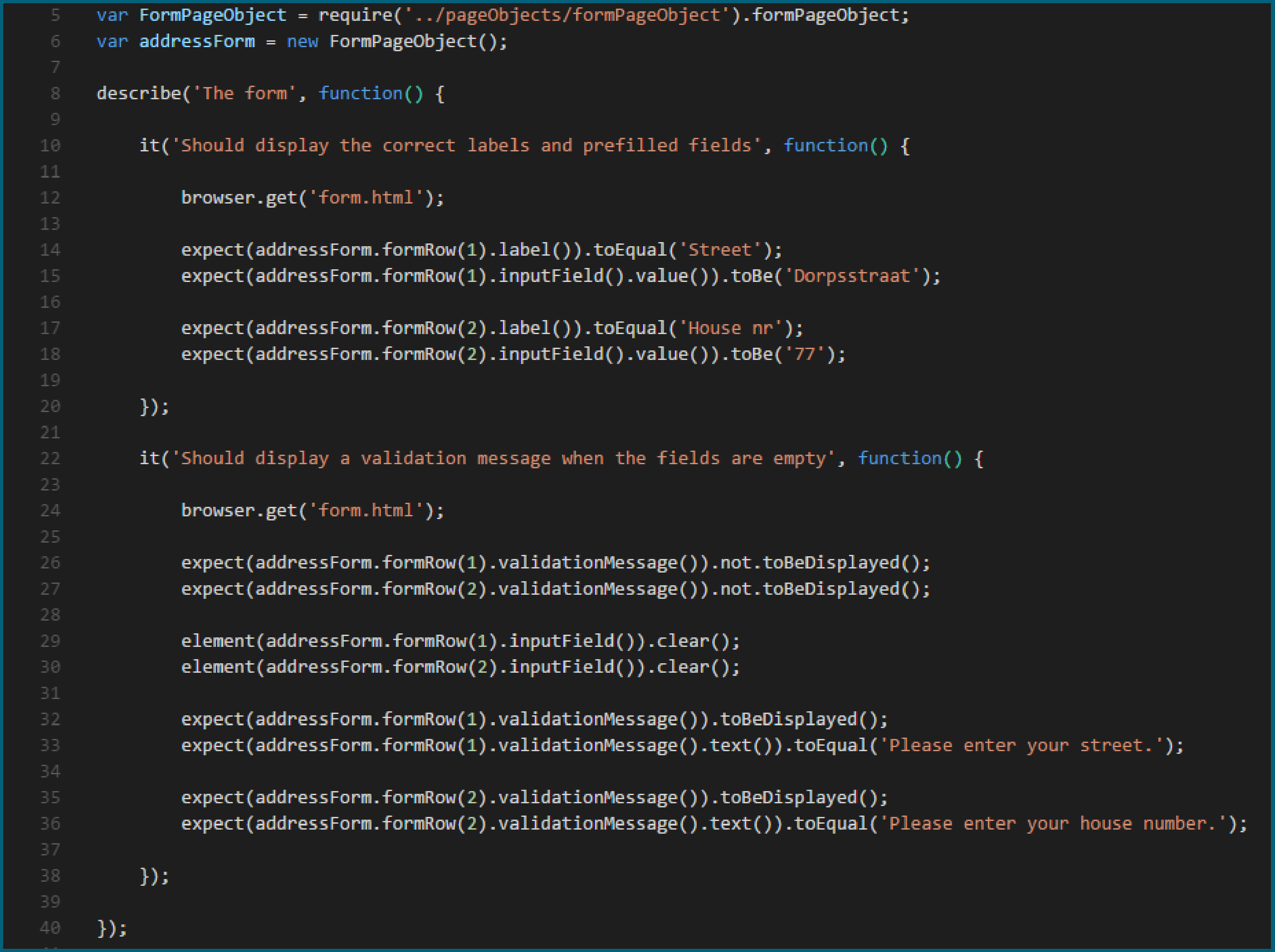
A **page object** is a Javascript object that contains the locators and functions for interacting with the application.



```
module.exports.formPageObject = function() {

    this.element = function() {
        return element(by.css('form[name="addressForm"]'));
    }

    this.formRow = function(eq) {

        var formRow = this.element().element(by.css('.form-row:nth-of-type(' + eq + ')'));

        formRow.label = function() {
            return formRow.element(by.css('label')).getText();
        }

        formRow.inputField = function() {
            var input = formRow.element(by.css('input[type=text]'));

            input.value = function() {
                return input.getAttribute('value');
            }

            return input;
        }

        formRow.validationMessage = function() {
            var validationMessage = formRow.element(by.css('.alert.alert-danger'));

            validationMessage.text = function() {
                return validationMessage.getText();
            }

            return validationMessage;
        }

        return formRow;
    }

};
```

Using page objects

Much better!



```javascript
var FormPageObject = require('../pageObjects/formPageObject').formPageObject;
var addressForm = new FormPageObject();

describe('The form', function() {

    it('Should display the correct labels and prefilled fields', function() {

        browser.get('form.html');

        expect(addressForm.formRow(1).label()).toEqual('Street');
        expect(addressForm.formRow(1).inputField().value()).toBe('Dorpsstraat');

        expect(addressForm.formRow(2).label()).toEqual('House nr');
        expect(addressForm.formRow(2).inputField().value()).toBe('77');

    });

    it('Should display a validation message when the fields are empty', function() {

        browser.get('form.html');

        expect(addressForm.formRow(1).validationMessage()).not.toBeDisplayed();
        expect(addressForm.formRow(2).validationMessage()).not.toBeDisplayed();

        element(addressForm.formRow(1).inputField()).clear();
        element(addressForm.formRow(2).inputField()).clear();

        expect(addressForm.formRow(1).validationMessage()).toBeDisplayed();
        expect(addressForm.formRow(1).validationMessage().text()).toEqual('Please enter your street.');

        expect(addressForm.formRow(2).validationMessage()).toBeDisplayed();
        expect(addressForm.formRow(2).validationMessage().text()).toEqual('Please enter your house number.');

    });

});
```

Live coding demo – setting up

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

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