

Frontend Testing automation

Focusing on

- Acceptance Testing
- Smoke Testing

Because

- Low effort needed
- Covers most basic functionality
- Makes annoying manual test scenarios less annoying
 - Doing a complete checkout

What we will use

- Javascript
 - Because Frontend
- Jest
 - As testing framework
- Puppeteer
 - For a Headless Chrome API in Node.js
- Jest-Puppeteer
 - To connect them both

package.json

```
{  
  "dependencies": {  
    "jest": "^24.9.0",  
    "jest-puppeteer": "^4.3.0",  
    "puppeteer": "^1.20.0",  
    "puppeteer-core": "^1.20.0"  
  }  
}
```

A first Test - cms-home

```
const website = require("./website.config.js");
```

```
describe('cms-home', () => {
```

```
  beforeAll(async () => {
```

```
    await page.goto(website.base_url);
```

```
  });
```

```
  it('should be titled "Madison Island"', async () => {
```

```
    await expect(page.title()).resolves.toMatch('Madison Island');
```

```
  });
```

```
});
```

Result

```
$jest --config=openmage\jest.config.js
```

```
PASS openmage/home.test.js
```

```
cms-home
```

```
  ✓ should be titled "Madison Island" (5ms)
```

```
Test Suites: 1 passed, 1 total
```

```
Tests:      1 passed, 1 total
```

```
Snapshots:  0 total
```

```
Time:       2.624s, estimated 3s
```

```
Ran all test suites.
```

Test - add ProductToCart

```
it('add Product to Cart', async () => {  
  await page.goto(website.base_url+'/accessories/jewelry/blue-horizons-bracelets.html');  
  await expect(page.title()).resolves.toMatch('Blue Horizons Bracelets');  
  await expect(page.title()).resolves.toMatch('Jewelry');  
  // Assert that a button containing text "Home" will be clicked  
  await expect(page).toClick('.add-to-cart-buttons .btn-cart', { text: 'Add to Cart' });  
  await page.waitForSelector('html');  
  const pathname = await page.evaluate(() => document.location.pathname);  
  expect(pathname).toBe('/checkout/cart/');  
  await expect(page.title()).resolves.toMatch('Shopping Cart');  
});
```


Check Product is in Cart

```
it('check Product to Cart', async () => {  
  await page.goto(website.base_url+'/checkout/cart/');  
  await expect(page).toMatchElement('#shopping-cart-table .product-name', { text: 'Blue Horizons Bracelets' });  
  await expect(page).toMatchElement('#shopping-cart-table .product-name', { text: 'Blue Horizons Bracelets' });  
});
```

Checkout process (M1)

```
it('Go through Checkout', async () => {
  await page.goto(website.base_url+'/checkout/cart/');
  await expect(page).toClick('.btn-proceed-checkout', { text: 'Proceed to Checkout' });
  await page.waitForSelector('html');
  await expect(page.title()).resolves.toMatch('Checkout');
  await expect(page).toClick('#checkout-step-login [value=guest]');
  await expect(page).toClick('#onpage-guest-register-button');

  await expect(page).toFill('#billing-new-address-form [name="billing[firstname]"]', 'Jamy');
  await expect(page).toFill('#billing-new-address-form [name="billing[lastname]"]', 'Jones');
  await expect(page).toFill('#billing-new-address-form [name="billing[email]"]', 'test@example.com');
  await expect(page).toFill('#billing-new-address-form [name="billing[street][]"]', 'Nowhere Street 42');
  await expect(page).toFill('#billing-new-address-form [name="billing[city]"]', 'somewhere');
  await expect(page).toSelect('#billing-new-address-form [name="billing[region_id]"]', 'Alaska');
  await expect(page).toFill('#billing-new-address-form [name="billing[postcode]"]', '12345');
  await expect(page).toFill('#billing-new-address-form [name="billing[telephone]"]', '0123456789');

  await expect(page).toClick('#opc-billing #billing-buttons-container button');
```

Checkout process (M1)

```
await expect(page).toClick('#opc-billing #billing-buttons-container button');
```

```
await page.waitForSelector('#opc-shipping_method.active');
```

```
await page.waitForSelector('#s_method_flatrate_flatrate');
```

```
await expect(page).toClick('#shipping-method-buttons-container button');
```

```
await page.waitForSelector('#opc-payment.active');
```

```
await expect(page).toClick('#payment-buttons-container button');
```

```
await page.waitForSelector('#opc-review.active');
```

```
await expect(page).toClick('#review-buttons-container .btn-checkout');
```

```
await page.screenshot({path: 'checkout_click.png'});
```

```
await page.waitFor(2000);
```

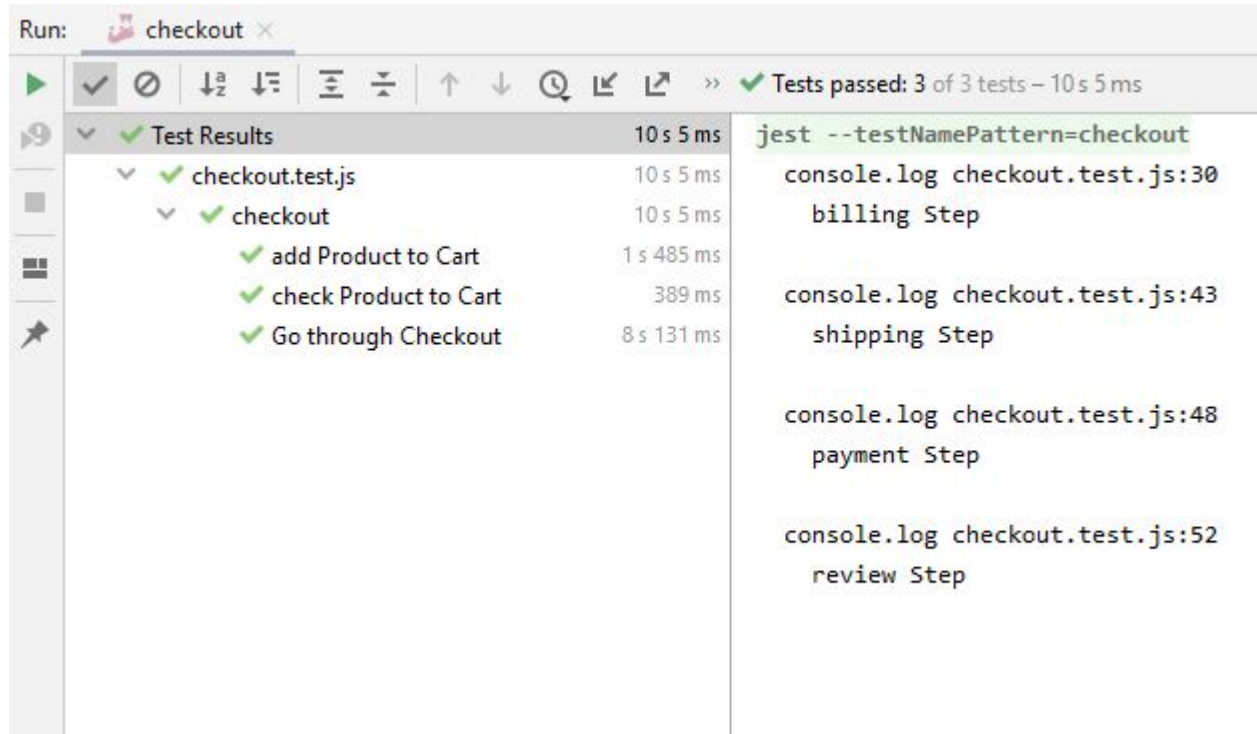
```
await page.waitForSelector('html');
```

```
await page.screenshot({path: 'checkout_after.png'});
```

```
await expect(page.title()).resolves.toMatch('Magento Commerce');
```

```
}, 50000);
```

With phpStorm integration



The screenshot shows the Run tool window in PhpStorm for a test named 'checkout'. The window is divided into two main sections: Test Results and Test Output.

Test Results:

Test Name	Duration
Test Results	10 s 5 ms
checkout.test.js	10 s 5 ms
checkout	10 s 5 ms
add Product to Cart	1 s 485 ms
check Product to Cart	389 ms
Go through Checkout	8 s 131 ms

Test Output:

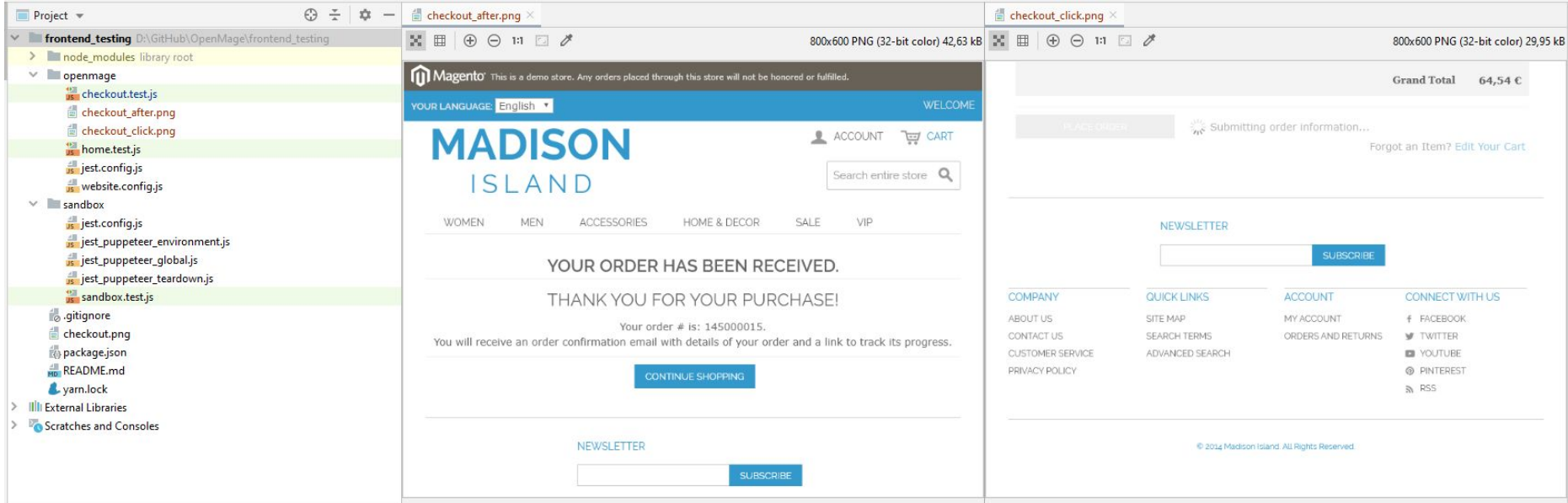
```
jest --testNamePattern=checkout
console.log checkout.test.js:30
  billing Step

console.log checkout.test.js:43
  shipping Step

console.log checkout.test.js:48
  payment Step

console.log checkout.test.js:52
  review Step
```

screenshots



Issues

- Example is unstable
 - Fails seemingly random
- Bad output in case of error
 - I got often “Node is either not visible or not an HTML Element”
 - Without line Number
 - Or any other hint, which did cause it

Links

- <https://github.com/GoogleChrome/puppeteer>
- <https://jestjs.io/docs/en/puppeteer>
- <https://www.jetbrains.com/help/phpstorm/running-unit-tests-on-jest.html#createRunConfigJest>
- <https://github.com/smooth-code/jest-puppeteer/tree/09703eaeeeeab553e13142153b55030db05611f7c/packages/expect-puppeteer/src/matchers>
- <https://pptr.dev/#?product=Puppeteer&version=v1.20.0&show=api-class-page>
- https://github.com/Flyingmana/frontend_testing_example/blob/master/openmage/checkout.test.js
-