WRITING END-TO-END TESTS WITH HEADLESS CHROME AND PUPPETEER

April 26th, 2019

@TaylorKrusen

I’m Taylor Krusen. I’m from Seattle. I work for Dropbox!

@TaylorKrusen

End-to-End Testing Using Puppeteer

What I'll cover

- End-to-end testing as a concept
- Headless Chrome and Puppeteer
- Using Puppeteer to create end-to-end tests

Why?

- Programmatically interact with a web page

Goals

- Explain end-to-end testing and why it’s valuable
- Demonstrate that Puppeteer is the best tool for the job

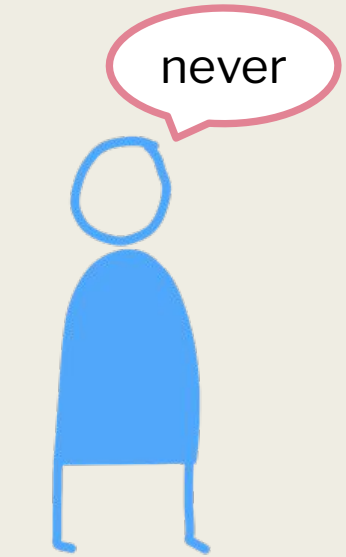
What is End-to-end Testing

- Does a feature behave as expected from a user's point of view?
- Can a User accomplish an action?

Web Apps Are Complicated


Kent C. Dodds
@kentcdodds
Follow
Still love this one. Unit testers be like:
"Looks like it's working"
gifbin.com
1:07 PM - 4 Aug 2015
3,010 Retweets 2,540 Likes
34
3.0K
2.5K

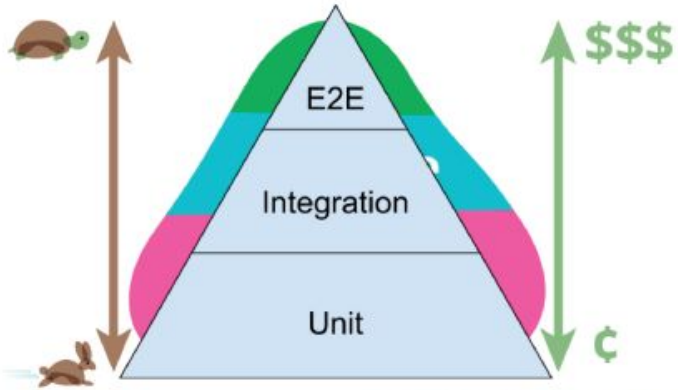
E2E !== Not a Replacement



End-to-end testing

Unit testing

Where do we focus our time?



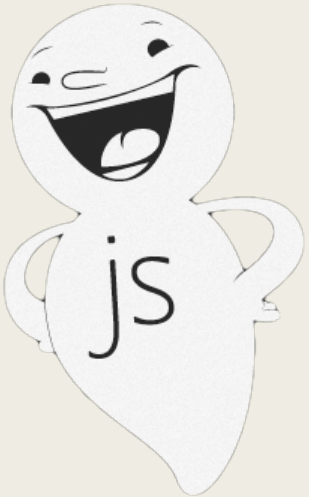
martinfowler.com/bliki/TestPyramid.html
testing.googleblog.com/2015/04/just-say-no-to-more-end-to-end-tests.html


INTERNET
EXPLORER
memegenerator.es

Headless Browsing

- Web page running without UI
- Program === User
- Scriptable

Headless Browsing Tools



Google enters the fray

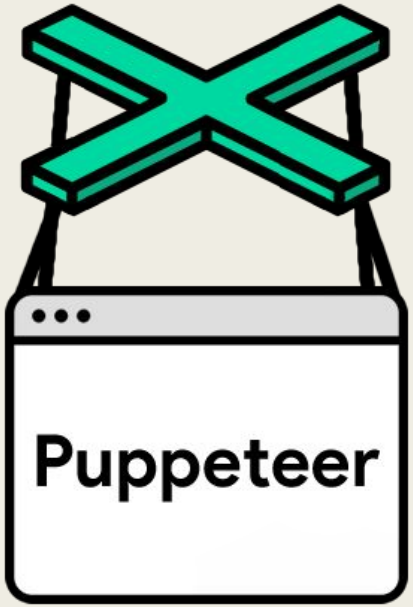
- Lead developer steps down in April 2017
- Google and Firefox both offer headless browsers
- Reasons: faster, more stable, less memory

Headless Chrome

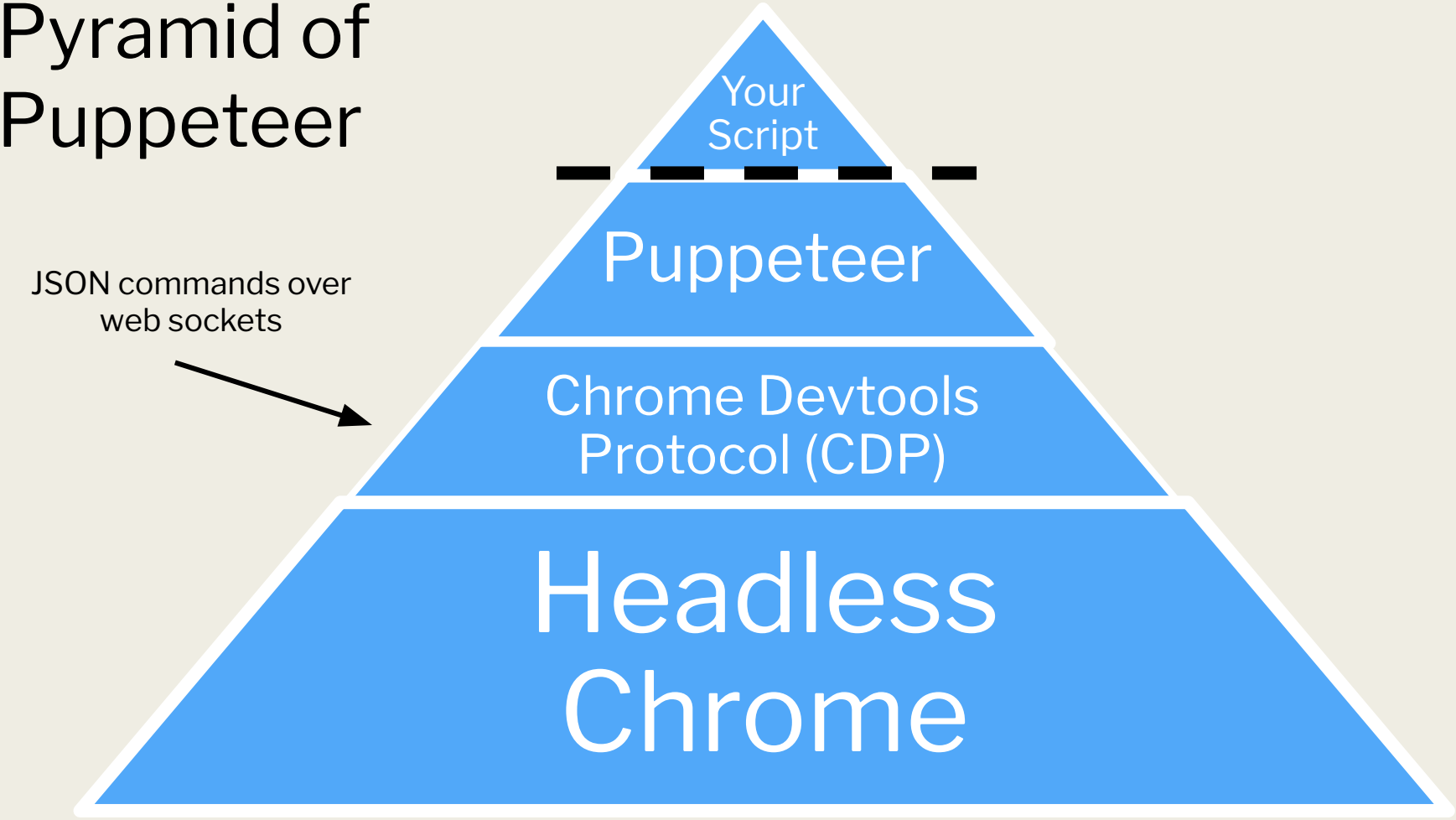
- Google Chrome's headless browser
- Shipped with Chrome 59 (~April 2017)
- Replaced PhantomJS as leader in headless browsing

Puppeteer

- API for interacting with Headless Chrome
- Built and maintained by Chrome DevTools team
  - *Fast feedback loop*
- Bundles the latest version of Chromium



Pyramid of Puppeteer



Install Puppeteer

- Install via NPM (or yarn)
  - `$ npm install --save puppeteer`
  - *Automatically installs chromium*

- Node 7.6.0+ to use async await

Take a Screenshot

```
const puppeteer = require('puppeteer');

(async () => {
    const browser = await puppeteer.launch();
    const page = await browser.newPage();

    await page.goto('https://imgur.com/r/aww');
    await page.screenshot({path: 'cute_animals.png'});

    await browser.close();
})();
```

PDFs

```
const puppeteer = require('puppeteer');

(async () => {
    const browser = await puppeteer.launch();
    const page = await browser.newPage();

    await page.goto('https://www.kickstarter.com/', {waitUntil: 'networkidle2'});
    await page.pdf({path: 'kickstarter.pdf', format: 'Letter'});

    await browser.close();
})();
```

Click Tweet → Create Overlay → Screenshot

```
const puppeteer = require('puppeteer');

(async () => {
    const browser = await puppeteer.launch();
    const page = await browser.newPage();
    await page.goto('https://twitter.com/taylorkrusen');
    await page.$eval('.tweet', tweet => {
        tweet.click();
    })
    await page.waitForSelector('.permalink-tweet', {visible: true});
    const tweet_overlay = await page.$('.permalink-tweet');
    await tweet_overlay.screenshot({path: 'the_tweet.png'});
    await browser.close();
})();
```

Filling Out a Form

- Stand alone or have as part of Smartsheet demo?

Debugging

- Pass options to change Puppeteer behavior

```
const browser = await puppeteer.launch({
  // turn off headless mode
  headless: false,
  // slow down execution
  slowMo: 100,
  // stop execution and open chrome dev tools
  devtools: true
});
```

- Capture console logs from page

```
page.on('console', msg ⇒ console.log('PAGE CONSOLE:', msg.text()));
```

Moving Beyond Screenshots

Identify features to test

- Logging into an app
  - Open page in browser
  - Fill out and submit form
  - Login takes User to app
- Using the search bar
  - Click then type in search field
  - Submit query
  - Click result

Use a testing framework

- Jest-puppeteer

  https://github.com/smooth-code/jest-puppeteer

Jest-puppeteer

```
beforeAll(async () => {
    browser = await puppeteer.launch();
    page = await browser.newPage();
});

describe('Group your tests in a description block', () => {
    test('Info about a test', async () => {
        // awesome tests
    });
});

afterAll(() => {
    browser.close();
});
```

Testing Smartsheet

What features should we test?

- *Logging into the application*
- *Click home button to load home screen*
- *Searching for a sheet*
- *Opening a sheet from the search results*
- *Applying formatting to a cell*

smartsheet

Log In and Click Home

```
describe('Smartsheet end-to-end tests', () => {
    test('Log in, load home, use search feature', async () => {
        // Load home ⟶ click login
        await page.goto('https://smartsheet.com');
        await page.click('li.leaf.menu-mlid-23291');
        await page.waitForSelector('input#loginEmail');
        // Type email ⟶ click continue
        await page.type('input#loginEmail', config.Secret_UserName);
        await page.click('#formControl');
        // Type password ⟶ click login ⟶ load app
        await page.waitForSelector('input#loginPassword');
        await page.type('input#loginPassword', config.Secret_Password);
        await page.click('#formControl');
        await page.waitForSelector('.clsGMO',{timeout:0});
        // Click home button ⟶ load home
        await page.click('.clsDesktopTabTypeIcon');
        await page.waitForSelector('.clsGMO');
```

Search Sheets and Open Result

```
        // Click search ⟶ wait for overlay
        await page.click('.clsImageRenderingOptimization');
        await page.waitForSelector('.clsFloatingForm', {visible: true});
        // Type in search ⟶ wait for results
        const searchField = await page.$('.clsUserEnteredText');
        await searchField.type('burger');
        await searchField.press('Enter');
        await page.waitForSelector('.clsStandardListText');
        // Click first result ⟶ load page
        await page.click('.clsStandardListText');
        await page.waitForSelector('.clsGMO');
    });
});
```

The Data-* Attribute

```
<input data-test-id="user-email" type="email">
<button data-test-id="form-submit" type="submit">
```

```
await page.type('[data-test-id="user-email"]', 'hello@gmail.com');
await page.click('[data-test-id="form-subit"]');
```

Log In and Click Home

Login logic stays the same

```
describe('Smartsheet end-to-end tests', () => {
    test('Log in, load home, use search feature', async () => {
        // Load home --> click login
        await page.goto('https://smartsheet.com');
        await page.click('li.leaf.menu-mlid-23291');
        await page.waitForSelector('input#loginEmail');
        // Type email --> click continue
        await page.type('input#loginEmail', config.Secret_UserName);
        await page.click('#formControl');
        // Type password --> click login --> load app
        await page.waitForSelector('input#loginPassword');
        await page.type('input#loginPassword', config.Secret_
        await page.click('#formControl');
        await page.waitForSelector('.clsGMO',{timeout:0});
        // Click home button --> load home
        await page.click('.clsDesktopTabTypeIcon');
        await page.waitForSelector('.clsGMO');
```

Breaks here

BROKEN Search Sheets and Open Result

```
        // Click search ⟶ wait for overlay
        await page.click('.clsImageRenderingOptimization');
        await page.waitForSelector('.clsFloatingForm', {visible: true});
        // Type in search ⟶ wait for results
        const searchField = await page.$('.clsUserEnteredText');
        await searchField.type('burger');
        await searchField.press('Enter');
        await page.waitForSelector('.clsStandardListText');
        // Click first result ⟶ load page
        await page.click('.clsStandardListText');
        await page.waitForSelector('.clsGMO');
    });
});
```

FIXED Search Sheets and Open Result

```
        // Open side panel ⟶ click favorites
        await page.click('[data-client-id="ltn-1"]')
        await page.click('[data-client-id="ltn-4"]');
        // Choose first result ⟶ close side panel
        await page.click('[data-client-id="fnl-1"]');
        await page.waitForSelector('.clsGMO');
        await page.click('[data-client-id="ltn-10"]');
        //click search box and type ⟶ wait for overlay
        const searchField = await page.$('.clsSearchInput');
        await searchField.type('burger');
        await searchField.press('Enter');
        await page.waitForSelector('.clsStandardListText');
        // click first result of search ⟶ load sheet
        await page.waitForSelector('.clsStandardListText');
        await page.click('.clsStandardListText');
        await page.waitForSelector('.clsGMO');
```

The expect-puppeteer Syntax

Bonus step: click cell containing specific word and make bold

```
        // puppeteer-expect gives us better test syntax
        await expect(page).toMatch('onions');
        // Click cell with Laboratory in it ⟶ make bold
        await expect(page).toClick('.clsTD1', { text: 'Laboratory'});
        await expect(page).toClick('[data-client-id="tlb-40"]')
    });
});
```

Cool Stuff Puppeteer Can Do

- Puppeteer can run…
  - *In a Docker container*
  - *In a Serverless environment*
- Server side rendering
- Intercept network requests
- Capture performance info
- Test a chrome extension
- Run code in a page
- Emulate Chrome on mobile device

Thank you! Questions?

Twitter: @TaylorKrusen