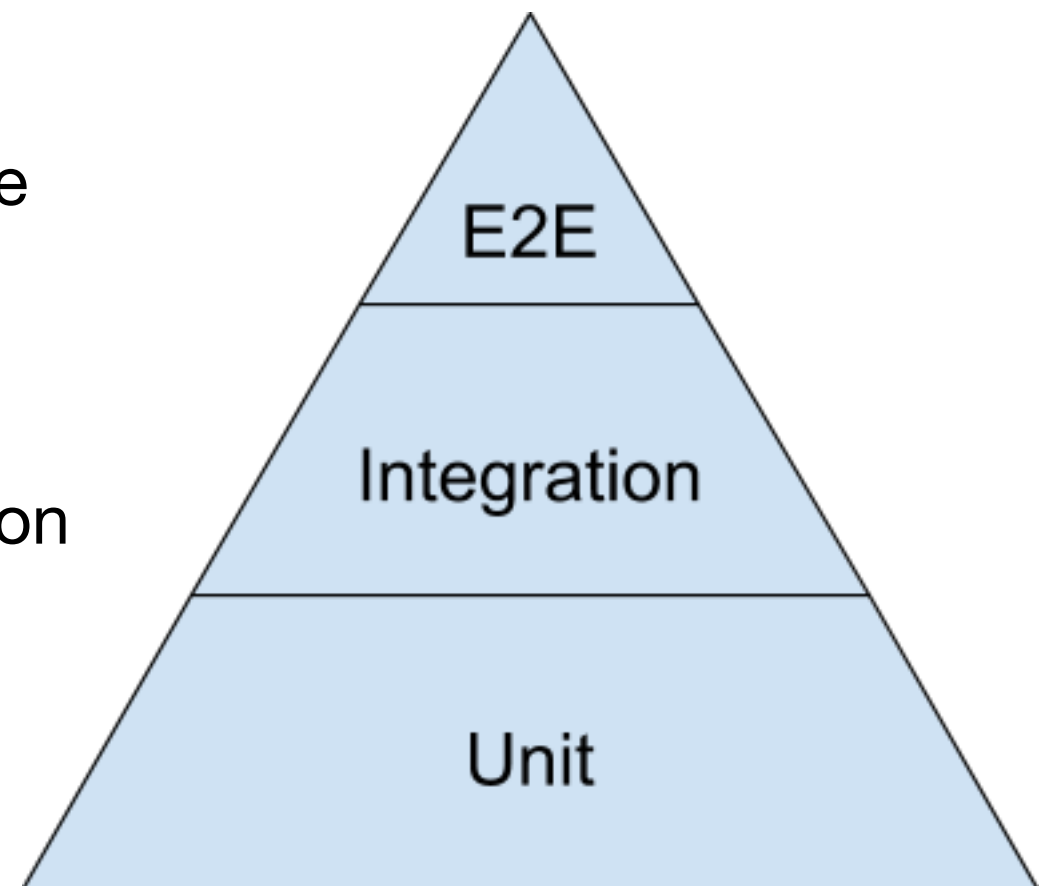


# Automation Testing

Jest & Puppeteer

# Introduction

- Unit Test: a piece of code or a module
  - Fast, reliable, isolates failures, White box
  - Jest, Mocha
- Integration testing: two or more module
  - Between Unit Test and E2E
- End-to-End Test (E2E): whole application
  - Simulates a Real User, Black box
  - Puppeteer, Selenium



# Jest

- From Facebook
- Fast and sandboxed
- Built-in code coverage reports
- Snapshot
- Powerful mocking library

# Getting Started

- `npm i -D jest babel-preset-env`
- `package.json`
  - `"scripts": { "test": "jest" },`
- `./node_modules/.bin/jest --init`
  - `jest.config.js`
- `.babelrc`

```
{
  "presets": [
    [
      "env",
      {
        "modules": false
      }
    ]
  ],
  "env": {
    "test": {
      "presets": [
        [
          "env"
        ]
      ]
    }
  }
}
```

# Simple example

- `math.js`

```
exports.add = function add(a, b) {  
  return a + b;  
};
```
- `math.test.js`

```
import { add } from '../src/core/math';  
  
describe('math', () => {  
  test('add', () => {  
    const a = 1;  
    const b = 2;  
    const c = add(a, b);  
    expect(c).toBe(3);  
  });  
});
```

# Using Matchers

- Common Matchers
  - toBe: object.is
  - toEqual: recursively checks every field of an object or array
- Truthiness: toBeNull toBeUndefined toBeDefined toBeTruthy toBeFalsy
- Numbers
  - toBeGreaterThan toBeGreaterThanOrEqual toBeLessThan toBeLessThanOrEqual
  - toBeCloseTo: for floating point number
- Strings: toMatch
- Arrays: toContain
- Exceptions: toThrow

# Testing Asynchronous Code

- Callbacks
- Promises
- ES2017 Async/Await

```
exports.fetchCallback = function fetchCallback(callback) {
  setTimeout(() => {
    callback('Done');
  }, 100);
};

exports.fetchPromise = function fetchPromise() {
  // eslint-disable-next-line no-unused-vars
  return new Promise((resolve, reject) => {
    setTimeout(resolve, 100, 'Done');
  });
};
```

```
test('fetchCallback', (done) => {
  backend.fetchCallback((data) => {
    expect(data).toBe('Done');
    done();
  });
});

test('fetchPromise', () => {
  expect.assertions(1);
  return backend.fetchPromise().then((data) => {
    expect(data).toBe('Done');
  });
});

test('fetchPromiseAsync', async () => {
  expect.assertions(1);
  const data = await backend.fetchPromise();
  expect(data).toBe('Done');
});
```

# Setup and Teardown

- `beforeAll(fn, timeout)`
- `afterAll(fn, timeout)`
- `beforeEach(fn, timeout)`
- `afterEach(fn, timeout)`
- `test.only(name, fn, timeout)`
- `test.skip(name, fn, timeout)`
- `test.each(table)(name, fn)`





# Code Coverage

- collectCoverage: true
- coverageDirectory: 'coverage'
- istanbul

## All files

61.54% Statements 8/13    100% Branches 8/8    44.44% Functions 4/9    61.54% Lines 8/13

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File ▲		Statements ▾		Branches ▾		Functions ▾		Lines ▾	
Entity.js		37.5%	3/8	100%	0/0	16.67%	1/6	37.5%	3/8
Util.js		100%	5/5	100%	0/0	100%	3/3	100%	5/5

# Mock

- Mock Functions
  - `mock.calls`
  - `mock.results`
  - `mock.instances`
- Mocking Modules

# Snapshot

- examples
- jsdom

# Testing Web Frameworks

- Vue.js
- AngularJS
- Angular

# Puppeteer

- From Google
- Headless Chrome Node API
- Useful for:
  - Crawler: Crawl a SPA and generate pre-rendered content
  - Automate tasks: form submission, keyboard and mouse emulation
  - UI Test: E2E Test
  - Auditing web performance

# Install

- `npm i -D puppeteer`
  - Proxy: `export http_proxy=http://127.0.0.1:1087;export https_proxy=http://127.0.0.1:1087;`
  - Include Chromium
- `npm i -D puppeteer-core`
  - Without Chromium
- examples
- API

# Config

- Turn off headless mode
  - `puppeteer.launch({headless: false});`
- Slow it down
  - `slowMo: 250 // slow down by 250ms`
- Uses a specific version of Chromium
  - `puppeteer.launch({executablePath: '/path/to/Chrome'});`

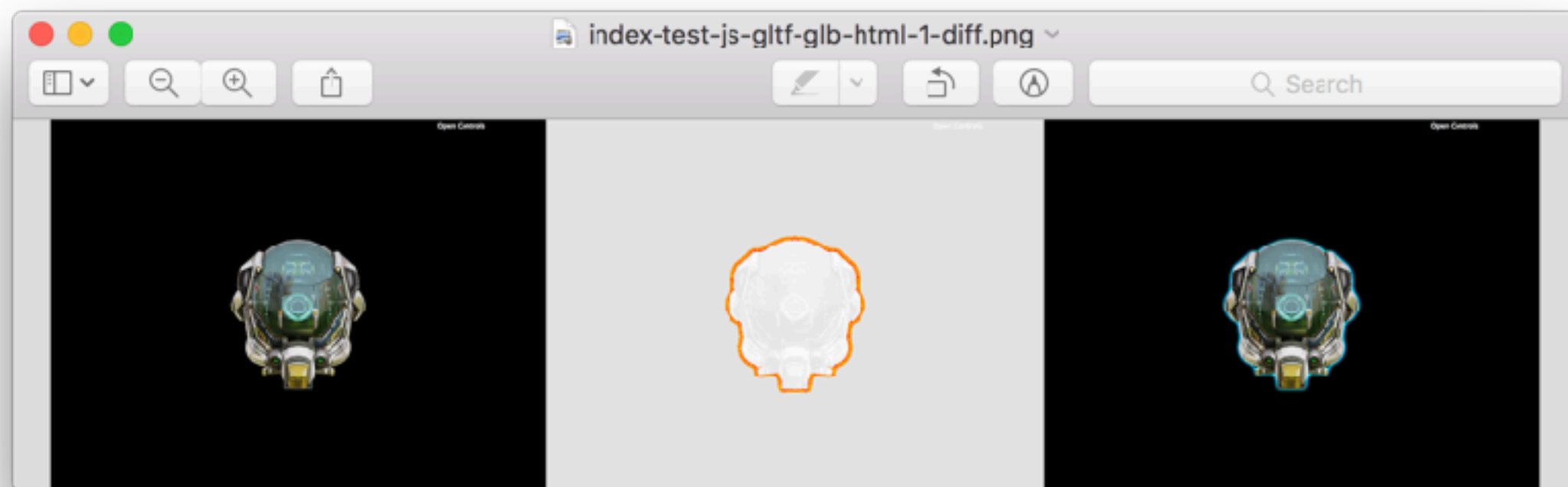
# Jest & Puppeteer

- jest-puppeteer
- `npm i -D jest-puppeteer`
- Setup
  - `globalSetup: './tool/puppeteer/setup.js',`
  - `globalTeardown: './tool/puppeteer/teardown.js',`
  - `testEnvironment: './tool/puppeteer/puppeteer_environment.js',`



# jest-image-snapshot

- `npm i -D jest-image-snapshot`
- `jest.config.js`
  - `setupTestFrameworkScriptFile: './test/jest-setup.js',`
  - `expect.extend({ toMatchImageSnapshot });`
- `expect(await page.screenshot()).toMatchImageSnapshot();`
- examples



# Tracing

- `page.tracing.start({ path: 'trace.json' });`
- `page.goto`
- `await page.tracing.stop();`

Summary			Bottom-Up	Call Tree	Event Log
Filter		Group by Category ▼			
Self Time		Total Time		Activity	
824.1 ms 79.8 %		824.1 ms 79.8 %		▼ Scripting	
233.0 ms 22.6 %		268.2 ms 26.0 %		▶ bind	
73.3 ms 7.1 %		87.2 ms 8.5 %		▶ createImageBitmap	
62.9 ms 6.1 %		99.9 ms 9.7 %		▶ calculateTangent	
39.7 ms 3.8 %		39.7 ms 3.8 %		▶ getContext	
30.4 ms 2.9 %		545.4 ms 52.8 %		▼ Function Call	
23.6 ms 2.3 %		132.9 ms 12.9 %		Event	

# Tips for writing e2e tests

- Test features, not implementation
  - data-testid: glue between implementation and user interaction

- Stick to the happy path features

- use unit test for edge case

- Use async/await for asynchronous things

- cleaner than callback or promise

- DON'T use arbitrary wait times

- Use a fake data generator like faker

- Faker.js

## A bad example

```
test('can logout', async () => {  
  await page.click('#menu div > a')  
  sleep 500  
})
```

## A good example

```
test('can logout', async () => {  
  await page.click('[data-testid="userMenuButton"]')  
  await page.waitForSelector('[data-testid="userMenuOpen"]')  
  await page.click('[data-testid="logoutLink"]')  
  await page.waitForSelector('[data-testid="userLoginForm"]')  
})
```

# Crawler

- [A Guide to Automating & Scraping the Web with JavaScript](#)
- [Getting started with Puppeteer and Chrome Headless for Web Scraping](#)
- [A Deep Dive Guide for Crawling SPA with Puppeteer and Troubleshooting](#)
- [headless-chrome-crawler](#)
- Proxy
  - `puppeteer.launch({args: [ '--proxy-server=socks5://localhost:1086' ]});`
  - System http&https proxy, or socks proxy

# Other

- Selenium
  - Runs in many browsers and operating systems
  - Can be controlled by many programming languages and testing frameworks
- React
  - Enzyme, jsdom, Jest
- Angular
  - Karma
- Cypress
  - Trade-offs
- Travis CI: continuous integration service that integrates with your Github repository to automatically run your tests when the code is pushed

# Further reading

- [An Overview of JavaScript Testing in 2018](#)
- [Testing Your Frontend Code](#)
- [End-to-end Tests that Don't Suck with Puppeteer](#)
- [User Interface Testing with Jest and Puppeteer](#)
- [Making your UI tests resilient to change](#)
- [Just Say No to More End-to-End Tests](#)
- [Top 15 UI Test Automation Best Practices You Should Follow](#)
- [Test website performance with Puppeteer](#)