ANGULAR ARCHITECTS
INSIDE KNOWLEDGE
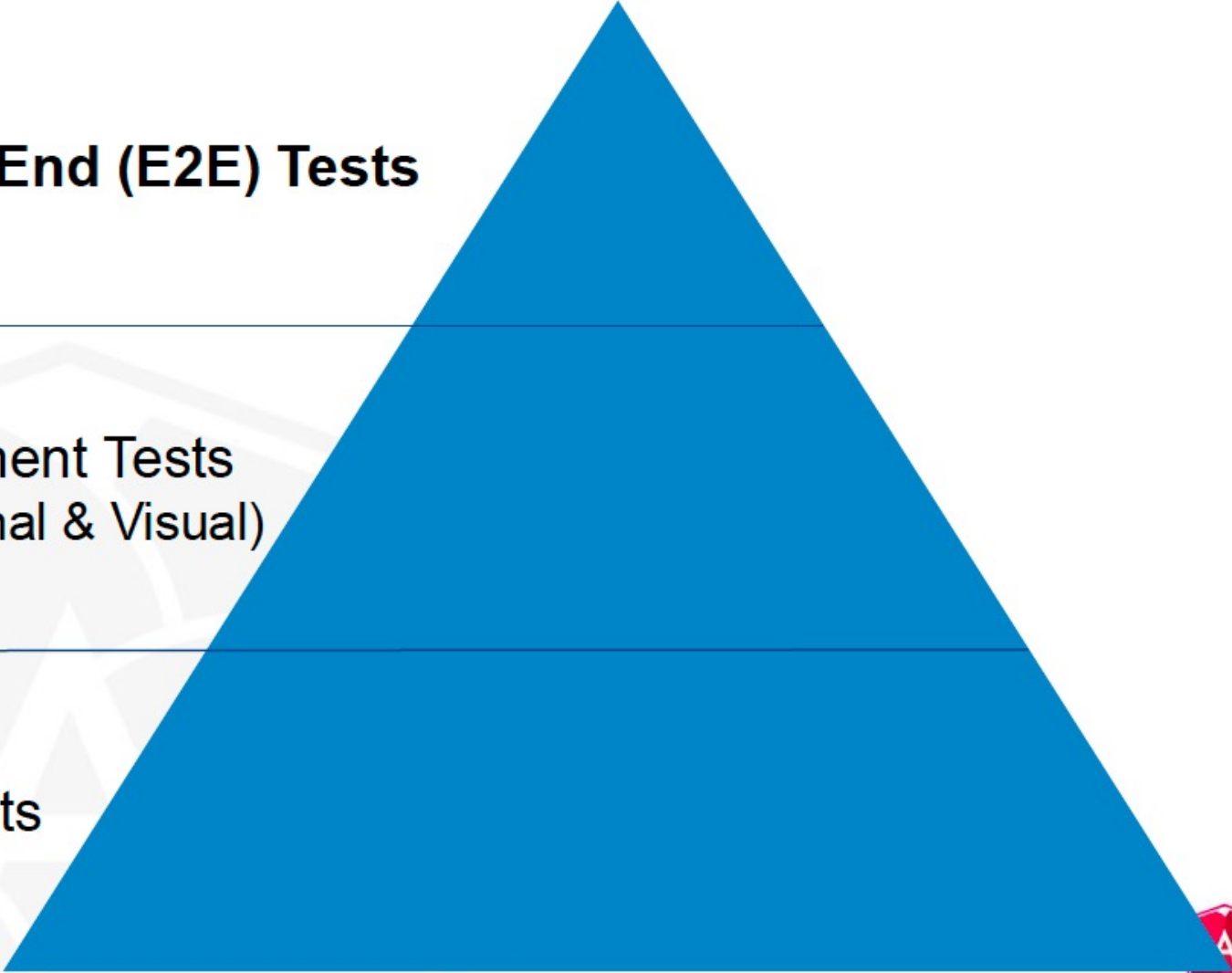
SOFTWARE ARCHITECT

# E2E Testing

Alex Thalhammer

# Testing pyramid

**End-to-End (E2E) Tests**

Component Tests
(Functional & Visual)

Unit Tests

# npm trends

# protractor vs cypress
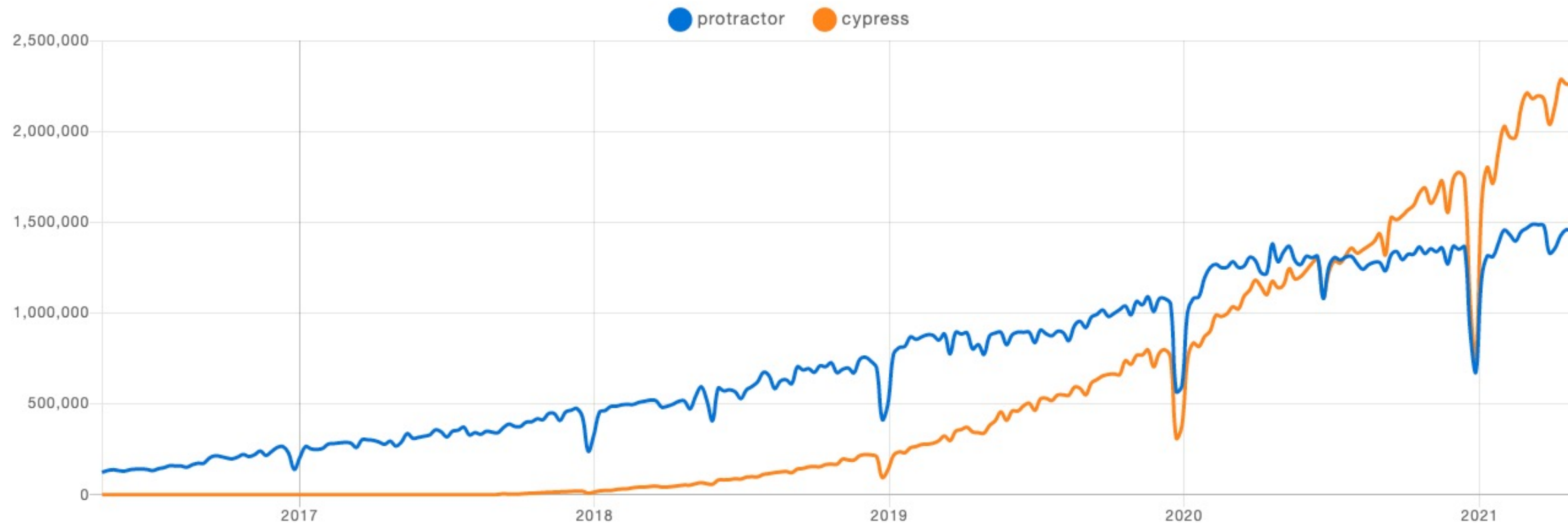
Enter an npm package...

protractor ×    cypress ×    + nightwatch    + webdriverio    + testcafe    + puppeteer

## Downloads in past 5 Years

● protractor    ● cypress



2,500,000

2,000,000

1,500,000

1,000,000

500,000

0

2017        2018        2019        2020        2021

# Intro

**cypress.io**

- Cypress (in most cases) perfect successor

- Migration is a rewrite

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# Motivation cypress.io

- Great Developer Experience

- Good Documentation

- Easy Setup

- Internal "IDE"

- CI Features like Videorecording and Screenshots

# Cypress Setup

- Cypress has no Angular integration (yet?)

- yarn add -D cypress

- add tsconfig.json to newly cypress directory

- cypress open (open the Cypress Dashboard) or

- cypress run (just runs the tests)

# Basic commands

- cy.visit(url: string)
  - Can only be run at the beginning (of a test)
  - Domain can't be changed

- cy.get(selector)
  - Uses jQuery style selectors
  - Runs asynchronously
  - Chainable
    - contains
    - click
    - type
    - …

# Assertions Implicit

- Behaves like a normal command

- Does waiting as well (asynchronicity)

- Good for single assertions

```
cy

  .get('h1')

  .should(

    'have.text',

    ' Unforgettable Holidays '

  );
```

# Assertions Explicit

- More verbose

- Good when more logic is involved

```
cy.get('h1').should(($h1) => {

    expect($h1).to.have

        .text(' Unforgettable Holidays ');

});
```

# Assertions are not required!

# Cypress – Demo

# Ready for the labs!

- Questions so far?

# Setup for Labs

- Prerequisites
  - NodeJS (I use version 14 because NG 12 currently demands it)
  - IDE: Visual Studio Code (free) or IntelliJ WebStorm
  - Chrome (or Chromium based)
  - Your repos checked out

- Exercises:
  - 09_e2e_testing.md or
  - 09_e2e_testing.pdf

# Advanced topics

# Page Object Model

- a page object should allow a software client
    - to do anything
    - see anything that a human can

- used for significant elements on a page

```
class Sidemenu {

  click(name: "Customers" | "Holidays"): Chainable {

    return cy.get("mat-drawer a").contains(name).click();

  }

}

export const sidemenu = new Sidemenu();
```

# Page Object Model - Demo

# Database Test Seed?

- Provide a Database specifically for Tests

- "One size fits all" approach

- Does not scale

- Better: Use Backend API for Arrange/Given Parts

# Without API access

- Reasons
  - Requires special privileges
  - External Data Source/Proxy
  - Unreasonable (complexity, efforts)

- Mocked API

```javascript
cy.server();

cy.route({

  method: 'GET',

  url: '/customer',

  response: [

    { id: 1, firstname: 'Hugo', name: 'Brandt', country: 'AT'},

    { id: 2, firstname: 'Natalia', name: 'Rusnov', country: 'RU'},

  ],

});

cy.visit('');
```

# Alternative: Arrange API Requests

- Call same endpoints as Angular

- Don't use the frontend directly!

```
cy.request({
  method: 'POST',
  url: '/holidays',
  body: {
    title: 'Pyramids',
    teaser: 'Visit the Ancient Wonders of Old Egypt',
    description: 'Fly to Cairo, get on a ship on the Nile and live like a Pharaoh'
  }
}).then(res => res.body.id);
```

# Best Case:  Dedicated Test API

- Backend provides special API for test mode

- Shortcuts possible, e.g.
    - merge chain of requests into one
    - Overcome Security Issues

- Best option, but of course expansive

# Now you know how to use Cypress

- Any questions left over? ☺