



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

E2E Testing With Cypress

Alex Thalhammer

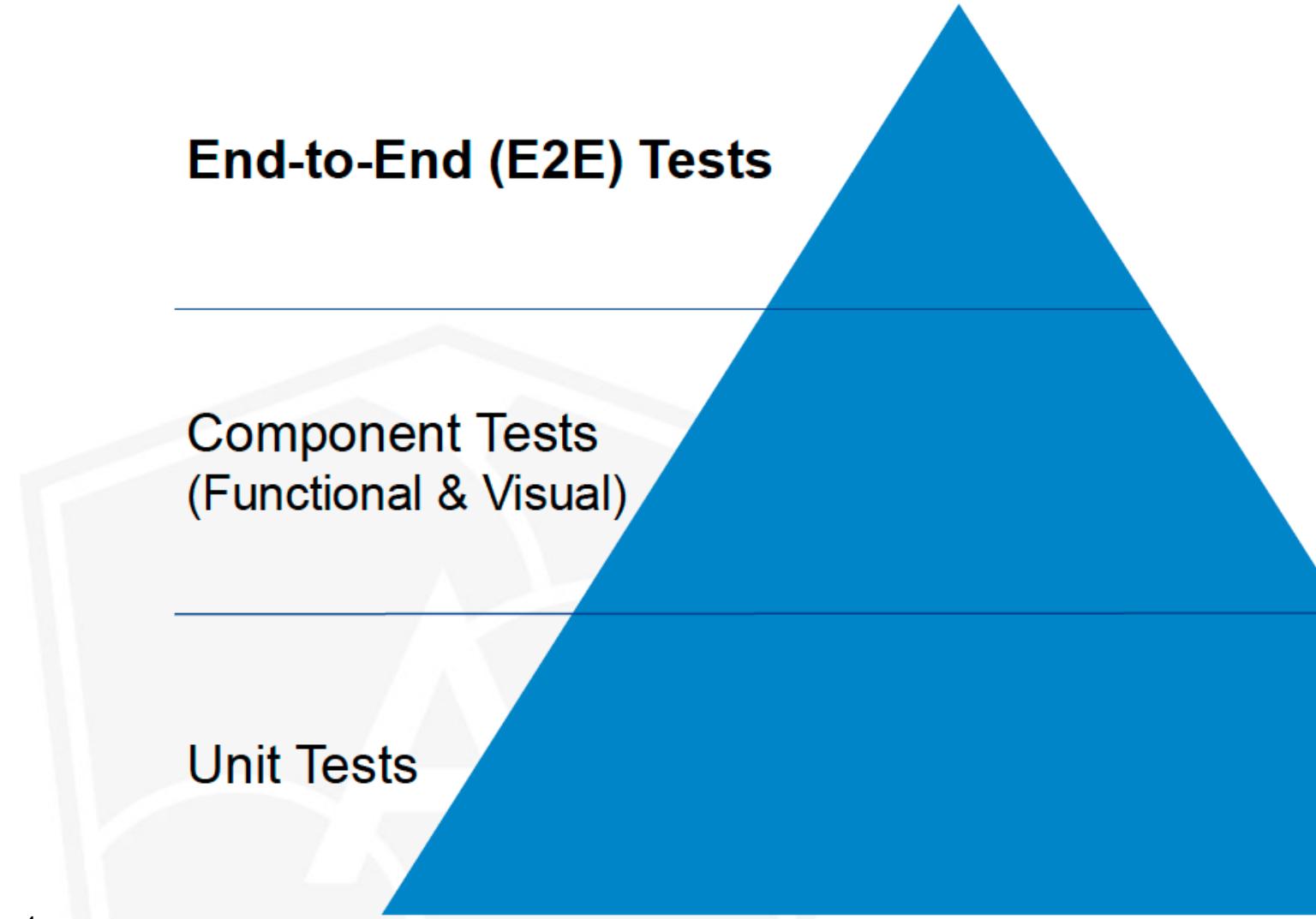
Outline

- Motivation
- E2E Testing with Cypress
- Cypress E2E Testing Lab
- Advanced topics

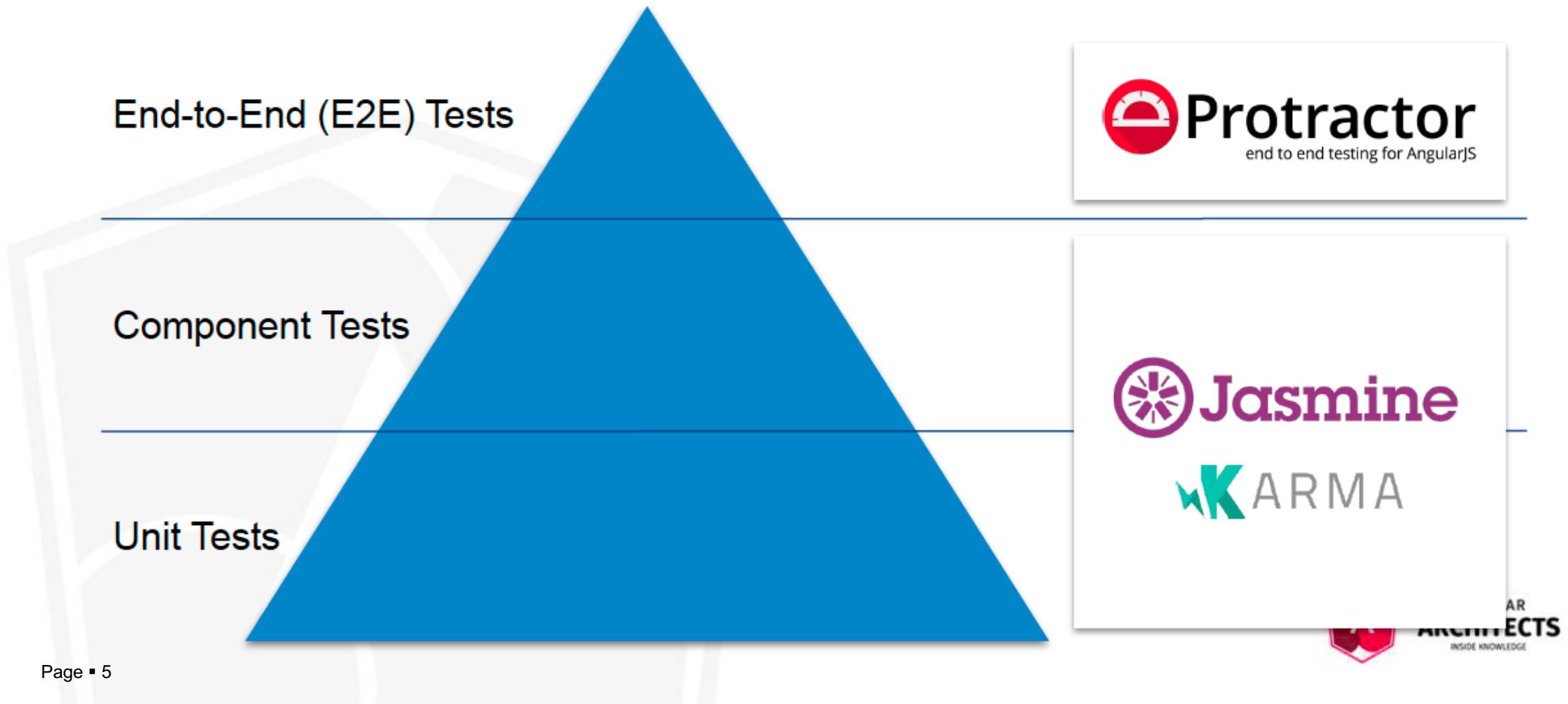
Motivation Testing

- Prevent bugs
- Enforce code quality
- Tests must be backed by Devs (require discipline)
- Writing Tests needs to be learned
- Tests must run fast, each has its own universe

Testing pyramid



Official version (until NG 11)

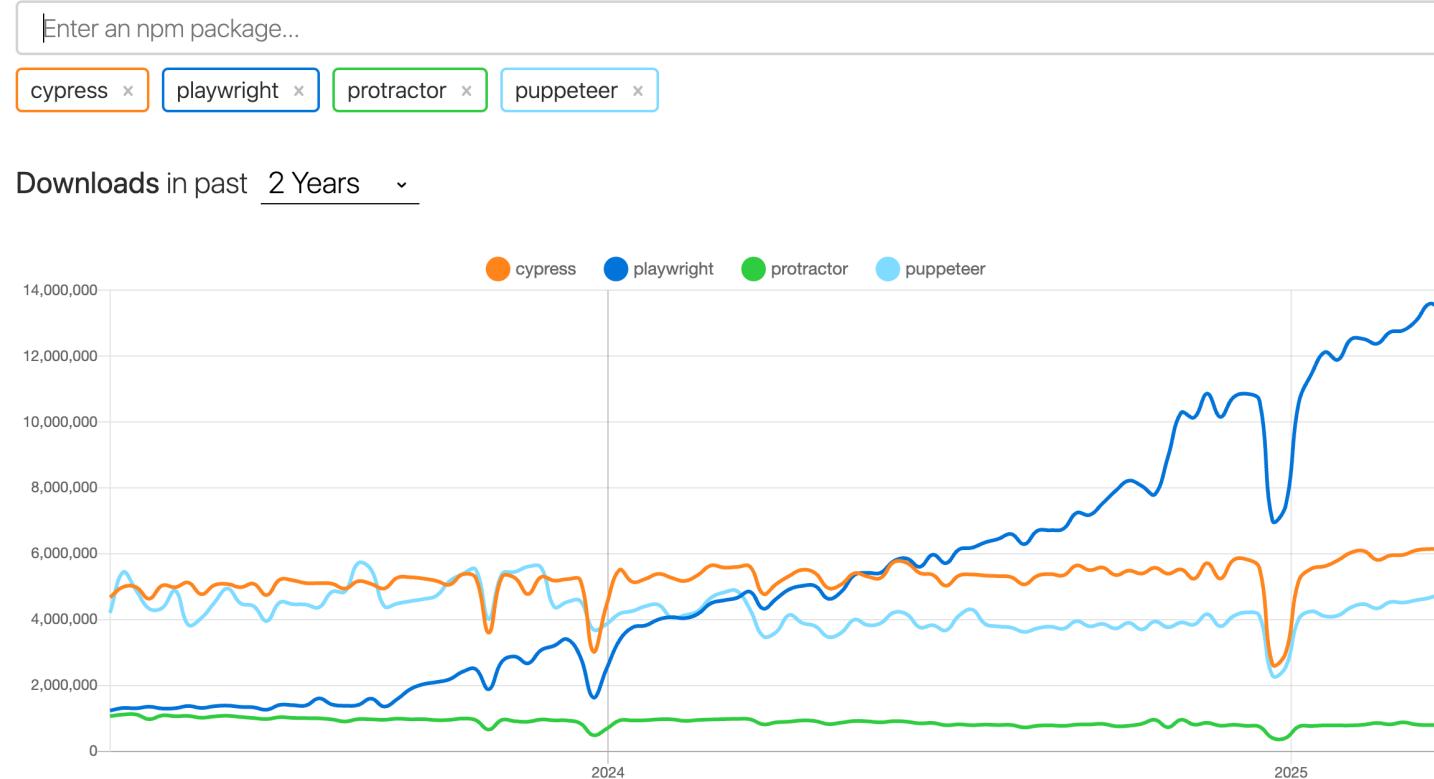


Our recommendation



NPM trends

cypress vs playwright vs protractor vs puppeteer



Intro



- Cypress (in most cases) perfect successor
- Migration is a rewrite

Motivation

- Great Developer Experience
- Good Documentation
- Easy Setup
- Internal "IDE"
- CI Features like Videorecording and Screenshots

Cypress Setup

- `npm i --save-dev / 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!



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

DEMO



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

LAB

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

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();
```

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

```
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? 😊