

# Master Cypress in 15 minutes a day

Maurice de Beijer - @mauricedb

# Course Goals

# Course Goals

- Learn what Cypress is
  - And what it can do for you
- Understand what the best practices are
  - And what not to do
- See how Cypress differs from other End to End testing tools
  - Like Selenium, Nightwatch.js etc.

# Why this course?

- Cypress is not hard to use most of the time
  - Not knowing the best practices can trip you up
- Going beyond the basics can be tough

15 minutes

- Can I really do this in 15 minutes a day?

See you in the next video

# What is Cypress

And why use it?

# What is Cypress

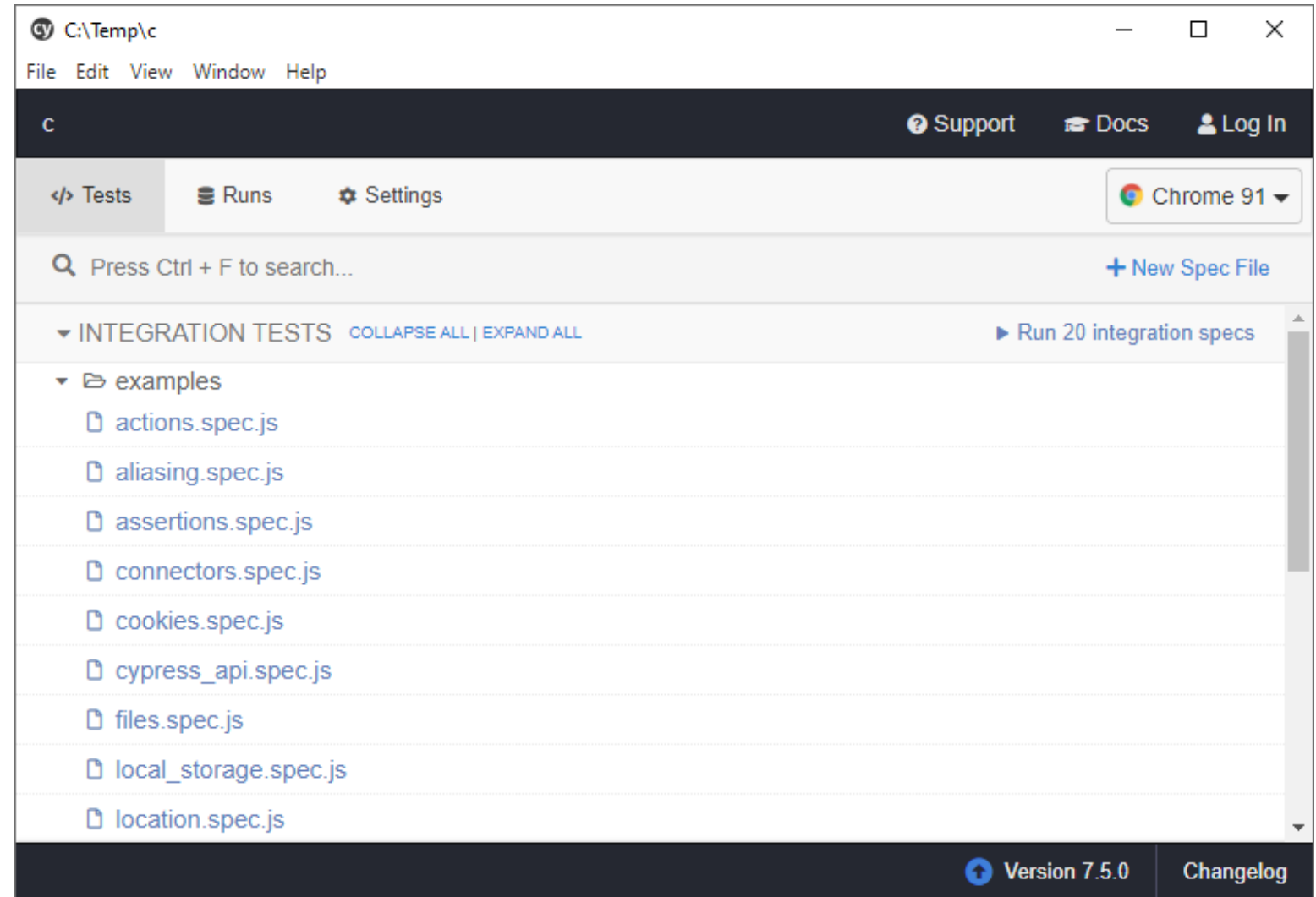
- Cypress is a front end testing tool
  - Built for the modern web
- Tests are easy to write
  - Using simple JavaScript
- Time Travel runner lets you inspect test at each step
  - Click on a step to see the state of the browser
- Automatic waiting for elements
  - Makes tests more resilient
- Video recording of running test
  - Helps debugging failing tests on the CI server



# What is Cypress

- Lets you fake the network
  - Or use the real backend as appropriate
- Automatically retry
  - Run flaky tests multiple times
- Many standard and 3<sup>rd</sup> party extensions
  - Or write your own in JavaScript
- Can run in Docker containers
  - Consistent cross platform behavior
- Runs inside the browser
  - Not based on Selenium

# Interactive Test Runner



# Interactive Test Runner

The screenshot displays the Cypress test runner interface. On the left, the 'Tests' panel shows a successful test run for 'Block Buster Film Reviews'. The test body includes the following steps:

- 1 visit /top-rated-movies
- 2 get #movie-278 a:first
- 3 -click
- 4 (page Load) --page Loaded--
- 5 (new url) https://block-buster-film-reviews.azureedge.net/movie/278
- 6 get h1
- 7 -assert expected <h1.bd-hd> to contain text The Shawshank Redemption (1994)

Below the test body, the test results show that the Shawshank Redemption has the expected cast and that the user can navigate to Morgan Freeman.

The main window displays the movie page for 'The Shawshank Redemption (1994)'. The page features a large image of the movie, a 'WATCH TRAILER' button, and a 'Rate This Movie' section with a star rating of 8.7/10. The page also includes sections for 'OVERVIEW', 'REVIEWS', 'CAST & CREW', 'MEDIA', and 'SIMILAR MOVIES'.

# Command Line Interface

```
Windows PowerShell x + -

Running: block-buster.spec.js (1 of 1)

Block Buster Film Reviews
  ✓ Has the correct title (810ms)
  ✓ Can open The Shawshank Redemption (344ms)
  ✓ The Shawshank Redemption has the expected cast (175ms)
  ✓ Can navigate to Morgan Freeman (377ms)

4 passing (2s)

(Results)

Tests:      4
Passing:    4
Failing:    0
Pending:    0
Skipped:    0
Screenshots: 0
Video:      true
Duration:   1 second
Spec Ran:   block-buster.spec.js

(Video)

- Started processing: Compressing to 32 CRF
- Finished processing: C:\Temp\c\cypress\videos\block-buster.spec.js.mp4 (1 second)




=====

(Run Finished)

Spec                                Tests  Passing  Failing  Pending  Skipped
✓ block-buster.spec.js              00:01    4        4        -        -        -
✓ All specs passed!                 00:01    4        4        -        -        -
```

# Pros and Cons of Cypress

## Pros

- Easy to get started
-  Fast execution
- Automatic waiting on elements
- Access to elements outside the DOM
- Support for FireFox and Chromium based browsers
- No driver dependencies
-  Time travel and debuggability
- Uses the Mocha test framework
- Open source
-  Mostly free

## Cons

- Only JavaScript or TypeScript
- No support for Safari
- Only a single browser
- Only a single tab

See you in the next video



# What is end to end testing

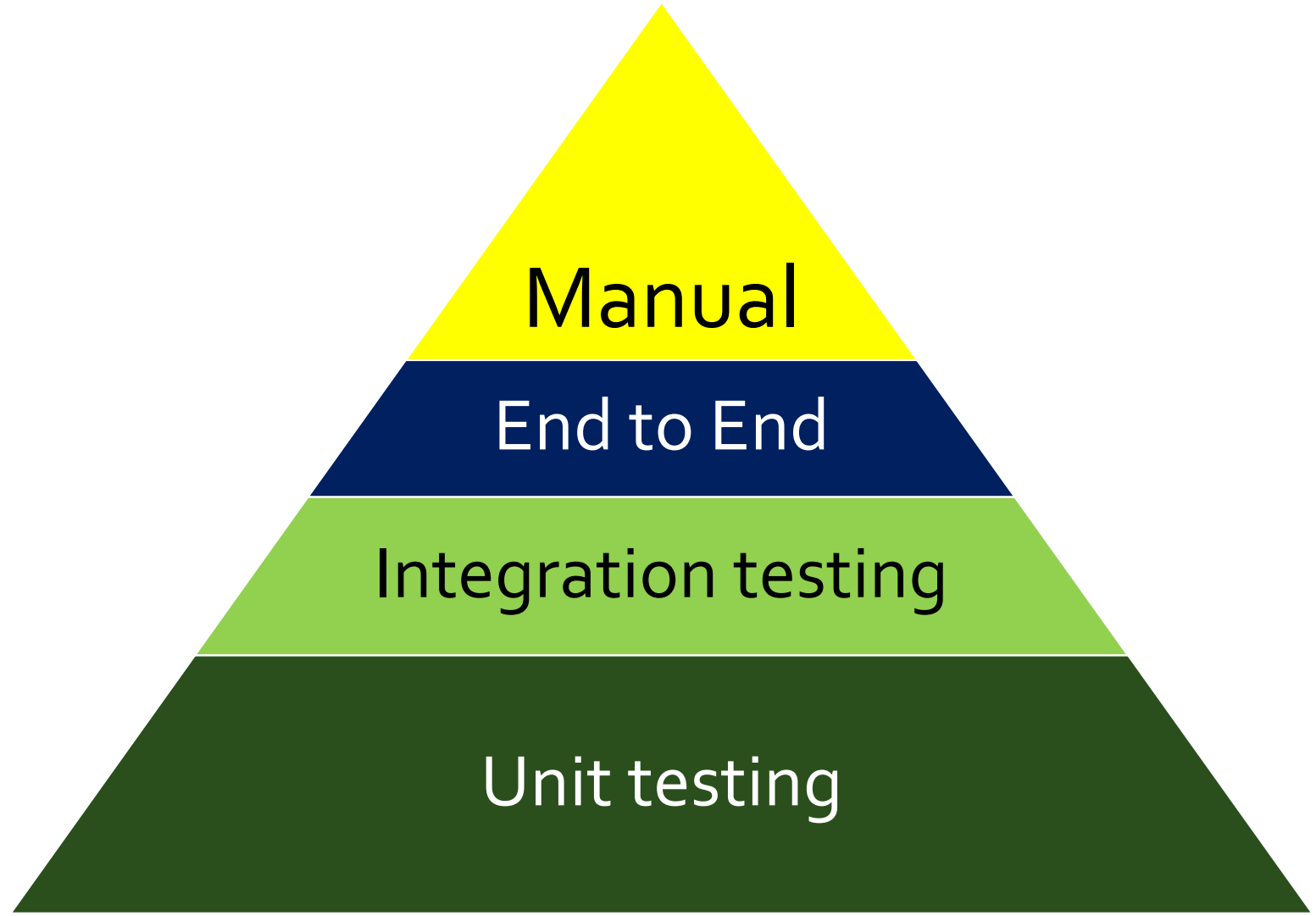
# What is testing

“Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test.”

-- Wikipedia --



# The traditional testing pyramid



# Unit Testing & Code Coverage

| File          | % Stmts | % Branch | % Funcs | % Lines | Uncovered Lines |
|---------------|---------|----------|---------|---------|-----------------|
| All files     | 98.92   | 94.36    | 99.49   | 100     |                 |
| yargs         | 99.17   | 93.95    | 100     | 100     |                 |
| index.js      | 100     | 100      | 100     | 100     |                 |
| yargs.js      | 99.15   | 93.86    | 100     | 100     |                 |
| yargs/lib     | 98.7    | 94.72    | 99.07   | 100     |                 |
| command.js    | 99.1    | 98.51    | 100     | 100     |                 |
| completion.js | 100     | 95.83    | 100     | 100     |                 |
| obj-filter.js | 87.5    | 83.33    | 66.67   | 100     |                 |
| usage.js      | 97.89   | 92.59    | 100     | 100     |                 |
| validation.js | 100     | 95.56    | 100     | 100     |                 |

Both windows  
are fine



via [reddit.com/r/programmerhumor](https://reddit.com/r/programmerhumor)

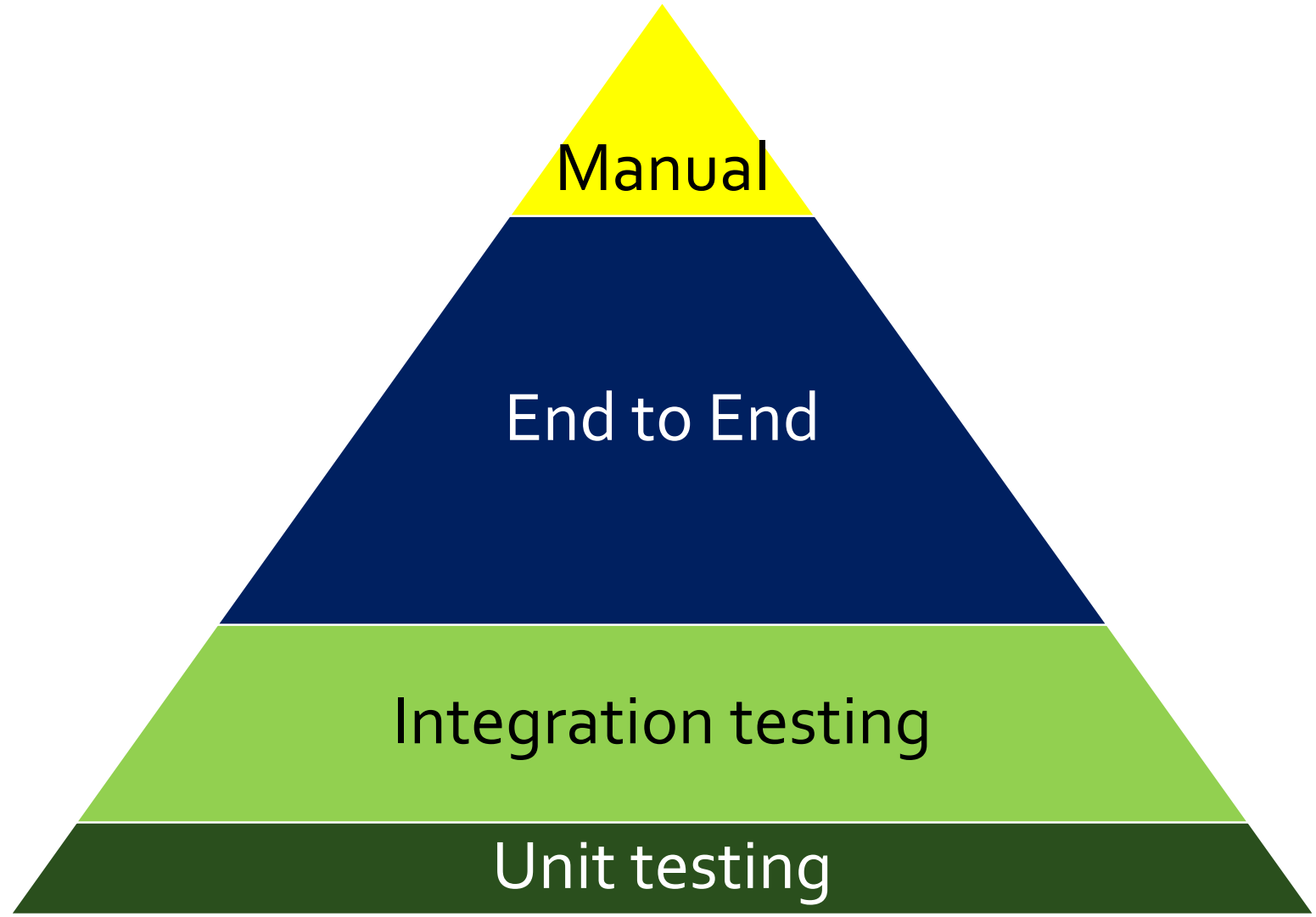
[Source](#)

A sturdy latch



[Source](#)

A better  
testing  
pyramid for  
the web



# End-to-end Testing

*“End-to-end testing is a methodology used to test whether the flow of an application is performing as designed from start to finish. The purpose of carrying out end-to-end tests is to identify system dependencies and to ensure that the right information is passed between various system components and systems.”*

-- Techopedia --

# End-to-end Testing

- Test an application's workflow from beginning to end
  - Including AJAX requests to the backend
- Based on workflows the end user actually executes
  - Using the same user interface, browser, network database etc.

See you in the next video



# Personal introduction



# Maurice de Beijer

Independent software developer and trainer

# The Netherlands









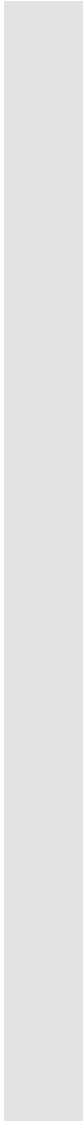





Happily married







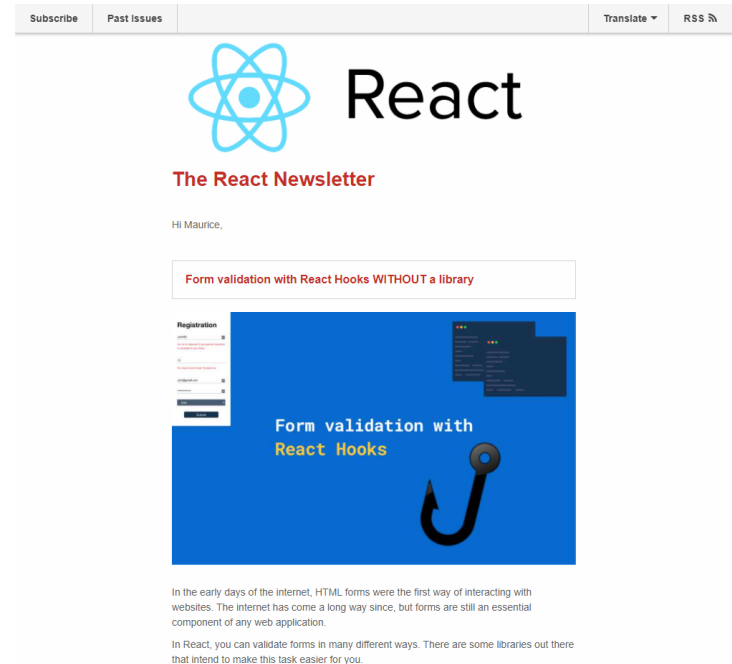
# Independent software developer & instructor

Since 1995



**Microsoft<sup>®</sup>  
Most Valuable  
Professional**

# The React Newsletter



See you in the next video

# Prerequisites

Install Node & NPM

Install the GitHub starter repository

# Following Along



```
1  /// <reference types="cypress" />
2
3  context('Block Buster Film Reviews', () => {
4
5    it('Has the correct title', () => {
6      cy.visit('/')
7      cy.get('h1').should('contain.text', 'Block Buster Film Reviews')
8    })
9
10   it('Can open The Shawshank Redemption', () => {
11     cy.visit('/top-rated-movies')
12     cy.get('#movie-278 a:first').click()
13     cy.get('h1').should('contain.text', 'The Shawshank Redemption (1994)')
14   })
15
16   it('The Shawshank Redemption has the expected cast', () => {
17     cy.visit('/movie/278')
18     cy.get('.col-md-8 > .mvcast-item > #cast-504 > p').should('have.text', '... Andy Dufresne')
19     cy.get('.col-md-8 > .mvcast-item > #cast-192 > p').should('have.text', '... Ellis Boyd "Red" Redding')
20     cy.get('.col-md-8 > .mvcast-item > #cast-192 > .cast-left').should('contain.text', 'Morgan Freeman')
21   })
22
23   it('Can navigate to Morgan Freeman', () => {
24     cy.visit('/movie/278')
25     cy.get('.col-md-8 > .mvcast-item > #cast-192 a').click()
26     cy.get('h1').should('contain.text', 'Morgan Freeman')
27   })
28 })
```

- Slides:
  - <http://theproblemsolver.nl/master-cypress-in-15-minutes-a-day.pdf>

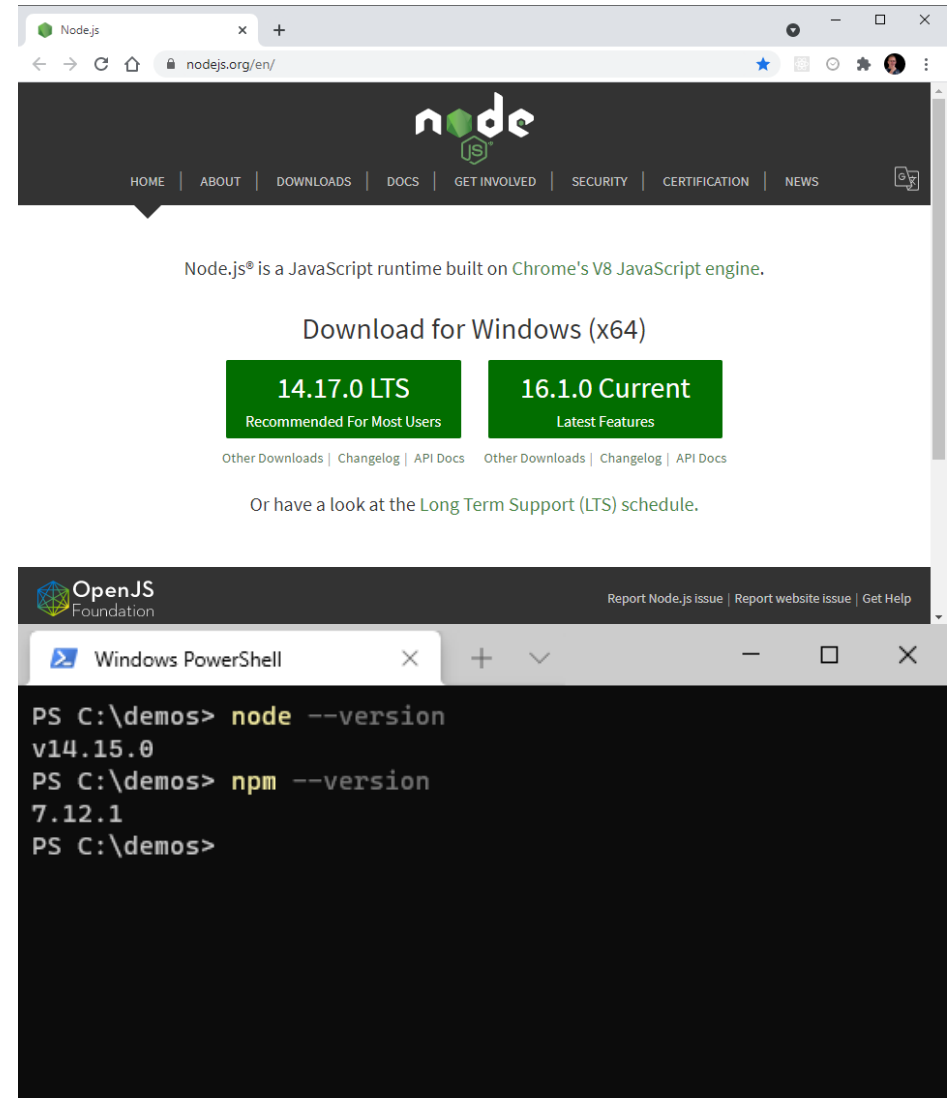
Type it out  
by hand?

*"Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now!"*

# Install Node.js



- Minimal:
  - Node version 12

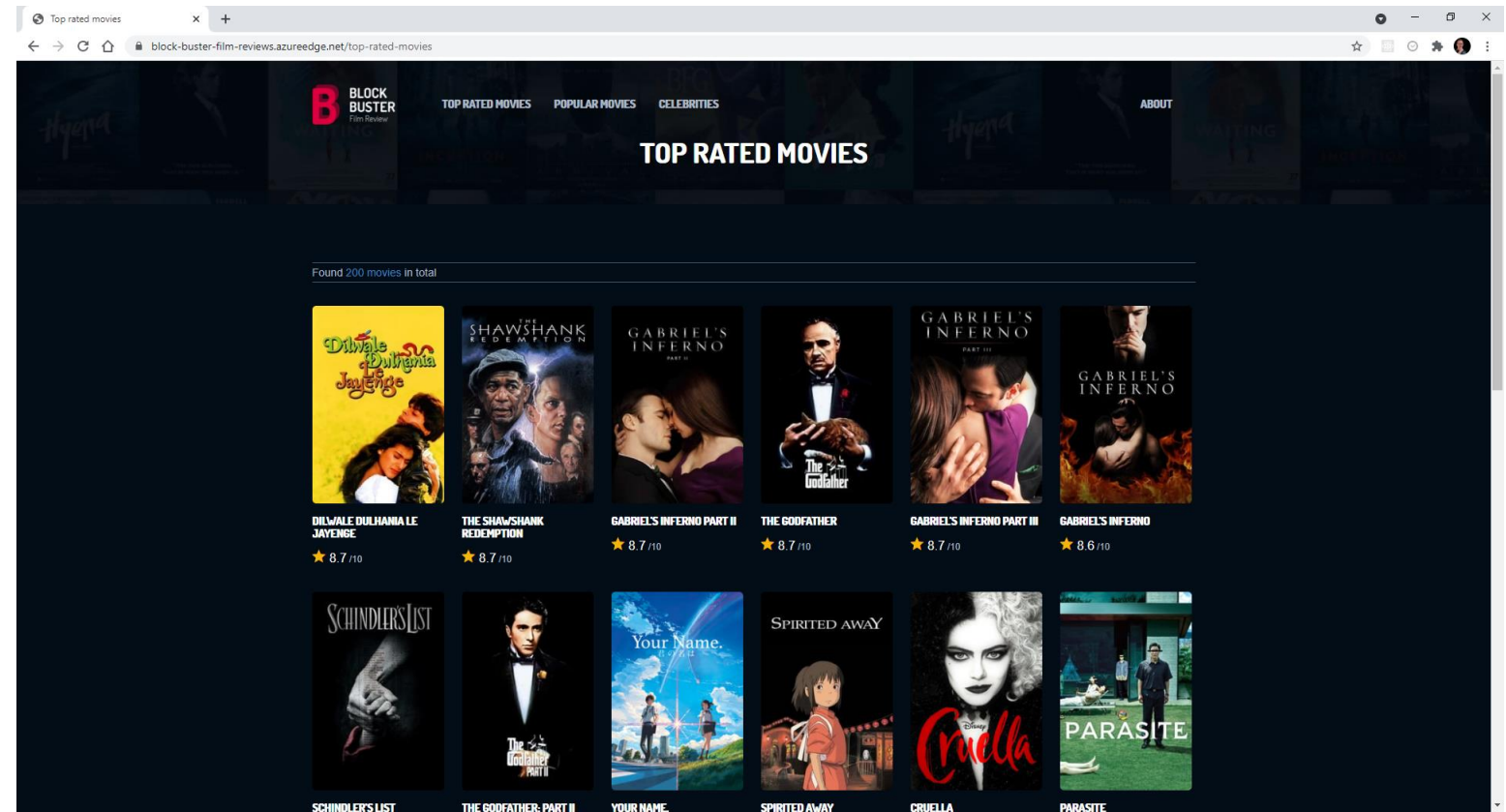




See you in the next video

# Creating a first Cypress test

# The Application



# Opening Cypress

- Start Cypress using “npx cypress open”
  - This will automatically download and run Cypress
- The first time it will create the Cypress a folder structure
  - Including a number of example tests

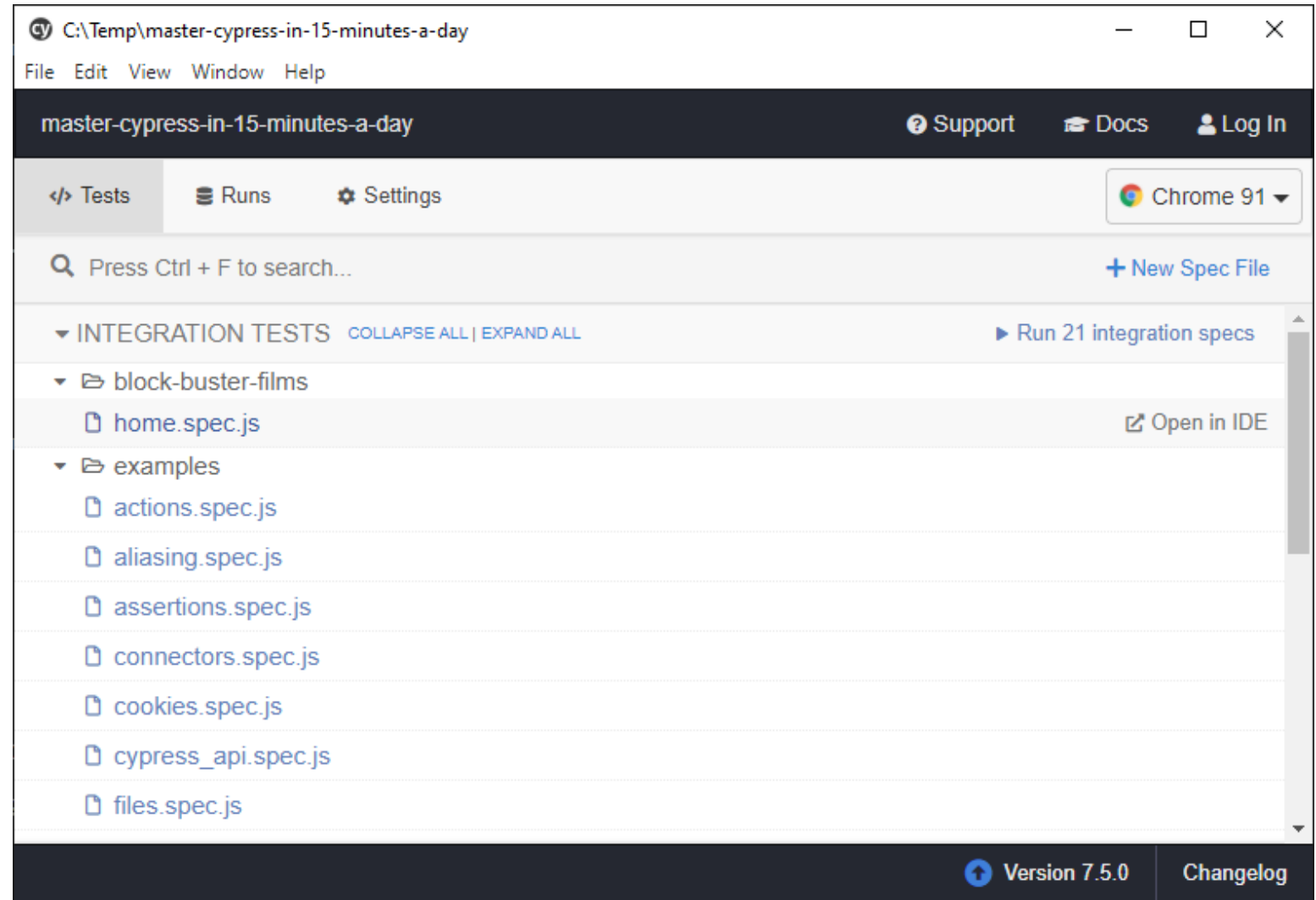
# Creating a first Cypress test

- Create a new spec file in the **/cypress/integration** folder
  - The spec file name must end with **spec.js**
- Call the **context()** function to create a group of specification
  - The **describe()** function is an alias for context()
- Call the **it()** function to create a End to End specification
  - Both **context()** and **it()** come from the Mocha test framework

# Creating a first Cypress test

```
JS home.spec.js X
cypress > integration > block-buster-films > JS home.spec.js > ...
1 context('Block Buster Film Reviews - Home Page', () => {
2   it('Has the correct title', () => {
3     cy.visit('https://block-buster-film-reviews.azureedge.net');
4
5     cy.title().should('equal', 'Block Buster Film Reviews');
6     cy.get('h1').should('have.text', 'Block Buster Film Reviews');
7   });
8 });
```

# Cypress test runner



# Running the test



master-cypress-in-15-minutes-a x +

Not secure | block-buster-film-reviews.azureedge.net/\_/#/tests/integration/block-buster-films/home.spec.js

Chrome is being controlled by automated test software.

< Tests 1 1 01.34 1000 x 660 (53%)

cypress/integration/block-buster-films/home.spec.js

Block Buster Film Reviews - Home Page

Has the correct title

TEST BODY

- 1 visit https://block-buster-film-reviews.azureedge.net/
- 2 title
- 3 - assert expected Block Buster Film Reviews to equal Block Buster Film Reviews
- 4 get h1
- 5 - assert expected <h1> to have text Block Buster Film Reviews

Block Buster Film Reviews



See you in the next video

# Adding a package.json

# Specifying the Cypress version

- Right now we are always using the **latest version of Cypress**
  - Instead of being explicit about what version to use
- Each developer/tester **has to know** to run “npx cypress open”
  - Maybe look for it in the documentation

# Adding a package.json

- When using Node.js the normal way is to **add a package.json**
  - And specify the dependencies with versions in there
  - As well as add important scripts to execute
- **Create a package.json** with the “npm init” command

# Adding dependencies

- **Add dependencies** with “npm install <<package>>”
  - Use the --save-dev option to register a development only package
  - Use “npm install cypress --save-dev”

# Adding scripts

- Add relevant scripts to the “scripts” section
  - “cypress open”
  - “cypress run”
- Execute with “npm run [script name]”

# The package.json



```
{ package.json U x
{ package.json > ...
1  {
2    "name": "master-cypress-in-15-minutes-a-day",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "cypress run",
8      "open": "cypress open"
9    },
10   "repository": {
11     "type": "git",
12     "url": "git+https://github.com/mauricedb/master-cypress-in-15-minutes-a-day.git"
13   },
14   "keywords": [
15     "cypress"
16   ],
17   "author": "Maurice de Beijer",
18   "license": "MIT",
19   "bugs": {
20     "url": "https://github.com/mauricedb/master-cypress-in-15-minutes-a-day/issues"
21   },
22   "homepage": "https://github.com/mauricedb/master-cypress-in-15-minutes-a-day#readme",
23   "devDependencies": {
24     "cypress": "^7.6.0"
25   }
26 }
```

See you in the next video



# Fixing a broken Cypress install

# A broken Cypress Installation

```
Windows PowerShell
PS C:\Repos\master-cypress-in-15-minutes-a-day> npm run open

> master-cypress-in-15-minutes-a-day@1.0.0 open
> cypress open

No version of Cypress is installed in: C:\Users\mauri\AppData\Local\Cypress\Cache\7.6.0\Cypress

Please reinstall Cypress by running: cypress install

-----

Cypress executable not found at: C:\Users\mauri\AppData\Local\Cypress\Cache\7.6.0\Cypress\Cypress.exe

-----

Platform: win32 (10.0.19042)
Cypress Version: 7.6.0
PS C:\Repos\master-cypress-in-15-minutes-a-day> |
```

# Cypress NPM Package

- The Cypress NPM package is not complete
  - It only contains part of the required bits
- The rest is installed into an application cache
  - Once for ever version used

# Cypress cache

## Binary cache

---

As of version `3.0`, Cypress downloads the matching Cypress binary to the global system cache, so that the binary can be shared between projects. By default, global cache folders are:

- **MacOS:** `~/Library/Caches/Cypress`
- **Linux:** `~/.cache/Cypress`
- **Windows:** `/AppData/Local/Cypress/Cache`

# Cypress cache

- Every version of Cypress is downloaded the first time it's required
  - And reused whenever needed
- Use the “cypress cache” command to manage the Cypress cache
  - `npx cypress cache path`
  - `npx cypress cache list [--size]`
  - `npx cypress cache prune`
  - `npx cypress cache clear`

## Cypress cache



```
Windows PowerShell
PS C:\Repos\master-cypress-in-15-minutes-a-day> npx cypress cache path
C:\Users\mauri\AppData\Local\Cypress\Cache
PS C:\Repos\master-cypress-in-15-minutes-a-day> npx cypress cache list --size
```

| version | last used      | size    |
|---------|----------------|---------|
| 6.0.1   | 26 minutes ago | 516.1MB |
| 6.5.0   | 23 minutes ago | 609.0MB |
| 6.9.1   | 20 minutes ago | 579.1MB |
| 7.0.1   | 17 minutes ago | 587.4MB |
| 7.3.0   | 12 minutes ago | 574.9MB |
| 7.5.0   | 14 minutes ago | 575.4MB |
| 7.6.0   | unknown        | 444.3MB |

```
PS C:\Repos\master-cypress-in-15-minutes-a-day> |
```

See you in the next video

# The most important commands



# cy.visit()

- Visit a URL in the browser
  - Takes the URL to navigate to
  - An optional options object with many settings
- <https://docs.cypress.io/api/commands/visit>

# cy.get()

- Queries the DOM for one or more elements
  - Uses the jQuery selector engine
- <https://docs.cypress.io/api/commands/get>

# .should()

- Check if a given condition is as expected
  - Based on the Chai NPM package
  - Often chain after a cy.get()
  - .and() is an alias
- Check for example
  - If an elements is visible and enabled
  - A value is equal to another
  - Many more...
- <https://docs.cypress.io/api/commands/should>

# Asynchronous

- Commands are asynchronous
  - They not executed immediately but placed in a queue

# Retry behavior

- Commands like `get()` and `should()` will retry upon failure
  - By default 4 seconds
- The retry timeout can be configured
  - Either globally or per command

# Chainable

- Most commands return a Chainable object

# Top rated movies Specs



```
JS top-rated-movies.spec.js U x
cypress > integration > block-buster-films > JS top-rated-movies.spec.js > ...
1  // top-rated-movies.spec.js created with Cypress
2  //
3  // Start writing your Cypress tests below!
4  // If you're unfamiliar with how Cypress works,
5  // check out the link below and learn how to write your first test:
6  // https://on.cypress.io/writing-first-test
7
8  context('Top rated movies', () => {
9    it('Has 20 movies per page', () => {
10      cy.visit('https://block-buster-film-reviews.azureedge.net/top-rated-movies')
11
12      cy.get('.movie-item-style-1').should('have.length', 24)
13    })
14
15    it('Has the correct title', () => {
16      cy.visit('https://block-buster-film-reviews.azureedge.net/top-rated-movies')
17
18      cy.title().should('equal', 'Top rated movies')
19      cy.get('h1')
20        .should('be.visible')
21        .should('have.text', 'Top rated movies')
22    })
23
24    it('The movie should be Dilwale Dulhania Le Jayenge', () => {
25      cy.visit('https://block-buster-film-reviews.azureedge.net/top-rated-movies')
26
27      cy.get('#movie-19404 > .mv-item-infor > h6 > a')
28        .should('have.text', 'Dilwale Dulhania Le Jayenge')
29    })
30  })
```

See you in the next video



# Adding IntelliSense

# IntelliSense

- **IntelliSense** in VS Code can be added with:
  - `/// <reference types="Cypress" />`
- Uses the **TypeScript type declarations**
  - From the Cypress NPM package

# IntelliSense



```
JS home.spec.js M x
cypress > integration > block-buster-films > JS home.spec.js > context('Block Buster Film Reviews') callback > it('Has the correct title - Home Page') callback

You, 2 minutes ago | 1 author (You)
1  /// <reference types="Cypress" />
2
3  context('Block Buster Film Reviews', () => {
4    it('Has the correct title - Home Page', () => {
5      cy.visit('https://block-buster-film-reviews.azureedge.net')
6
7      cy.title().should('equal', 'Block Buster Film Reviews')
8      cy.get('h1').should('have.text', 'Block Buster Film Reviews')
9    })
10 }
11
```

- have.text
- have.nested.property
- have.returned
- have.deep.nested.property
- have.length.gte
- have.length.lte
- have.lengthOf.gte
- have.lengthOf.lte
- have.always.returned
- have.length.greaterThan
- have.length
- have.length.above

See you in the next video

# Mocha hooks

# Mocha hooks

- Mocha provides **hooks** to run code **before and after tests**
  - The **before()** and **after()** run once per block
  - The **beforeEach()** and **afterEach()** run for every test
- Using the **beforeEach()** is very convenient
  - Use `cy.visit()` to load the right page

# Using beforeEach()



```
JS top-rated-movies.spec.js M x
cypress > integration > block-buster-films > JS top-rated-movies.spec.js > ...

10 context('Top rated movies', () => {
11
12   beforeEach(() => {
13     cy.visit('https://block-buster-film-reviews.azureedge.net/top-rated-movies')
14   })
15
16   it('Has 24 movies per page', () => {
17     cy.get('.movie-item-style-1').should('have.length', 24)
18   })
19
20   it('Has the correct title', () => {
21     cy.title().should('equal', 'Top rated movies')
22     cy.get('h1')
23       .should('be.visible')
24       .and('have.text', 'Top rated movies')
25   })
26
27   it('The movie should be Dilwale Dulhania Le Jayenge', () => {
28     cy.get('#movie-19404 > .mv-item-infor > h6 > a')
29       .should('have.text', 'Dilwale Dulhania Le Jayenge')
30   })
31 })
```

See you in the next video



# Cypress configuration

# Cypress configuration

- The `cypress.json` is used to store **global configuration values**
  - Base URL
  - Environment variables
  - Retries
  - Timeouts
  - Etc.
- <https://docs.cypress.io/guides/references/configuration#cypress-json>

# IntelliSense

- Add a **\$schema** key to get **IntelliSense**
  - <https://on.cypress.io/cypress.schema.json>

# Cypress.config()

- The **Cypress.config()** function
  - Retrieve a configuration value
  - Override a configuration value for a single test

# Command Line

- Use the `--config` option to **override config using the CLI**
  - Both with `cypress run` and `cypress open`

cypress.json



```
cypress.json M x
cypress.json > ...
You, seconds ago | 1 author (You)
1 {
2   "$schema": "https://on.cypress.io/cypress.schema.json",
3   "baseUrl": "https://block-buster-film-reviews.azureedge.net",
4   "defaultCommandTimeout": 10000,
5   "ignoreTestFiles": [
6     "**/1-getting-started/*",
7     "**/2-advanced-examples/*"
8   ]
9 }
```

See you in the next video

# Selecting DOM elements



# CSS Selectors

- Cypress uses the **jQuery selector engine**
  - Supports the standard CSS queries plus extension
- <https://api.jquery.com/category/selectors/>

## ✗ Anti-Pattern

- Using **highly brittle selectors** that are subject to change
- <https://docs.cypress.io/guides/references/best-practices#Selecting-Elements>

## ✓ Best Practice

- **Isolate selectors** from CSS or JS changes
  - ✗ Avoid: `cy.get('.btn.btn-large').click()`
- Using **data-\* attributes**
  - ✓ Use: `cy.get('[data-cy=submit]').click()`

# Recommendations

| Selector   | Recommended | Notes  |
|--|-------------|--|
| <code>cy.get('button').click()</code>            | ⚠ Never     | Worst - too generic, no context.                                     |
| <code>cy.get('.btn.btn-large').click()</code>    | ⚠ Never     | Bad. Coupled to styling. Highly subject to change.                   |
| <code>cy.get('#main').click()</code>             | ⚠ Sparingly | Better. But still coupled to styling or JS event listeners.          |
| <code>cy.get('[name=submission]').click()</code> | ⚠ Sparingly | Coupled to the <code>name</code> attribute which has HTML semantics. |
| <code>cy.contains('Submit').click()</code>       | ✅ Depends   | Much better. But still coupled to text content that may change.      |
| <code>cy.get('[data-cy=submit]').click()</code>  | ✅ Always    | Best. Isolated from all changes.                                     |

# .find()

- The `.find()` function **searches descendants** of a previous selection
  - Must be chained of a `cy.get()` or similar function
- <https://docs.cypress.io/api/commands/find>

# .within()

- **Don't** store the result of a command in a **variable**
  - ✗ You will not get a consistent result
- Use the `.within()` function
  - To **scope queries** to a previous result
- <https://docs.cypress.io/api/commands/within>

.eq()

- Use the eq( ) function to get a **specific element** from a **collection**
  - A negative index counts from the end
- <https://docs.cypress.io/api/commands/eq>

# home.spec.js



```
JS home.spec.js M X
cypress > integration > block-buster-films > JS home.spec.js > ...
13 context('The left navigation menu', () => {
14   it('The first navigation link points to Top rated movies', () => {
15     cy.get('.menu-left a:visible').first().should('have.text', 'Top rated movies')
16   })
17
18   it('The last navigation link points to Celebrities', () => {
19     cy.get('.menu-left').find('a:visible').last().should('have.text', 'Celebrities')
20   })
21
22   it('Nested searches', () => {
23     cy.get('.menu-left').within(() => {
24       cy.get('a:visible').first().should('have.text', 'Top rated movies')
25       cy.get('a:visible').last().should('have.text', 'Celebrities')
26     })
27   })
28 })
29
30 it('The navigation links have the correct text', () => {
31   cy.get('.menu-left a:visible').eq(0).should('have.text', 'Top rated movies')
32   cy.get('.menu-left a:visible').eq(1).should('have.text', 'Popular movies')
33   cy.get('.menu-left a:visible').eq(-1).should('have.text', 'Celebrities')
34 })
35 })
36
37 context.skip('Anti-patterns', () => {
38   it("Don't use variables", () => {
39     // This is an anti-pattern. Don't use!
40     const menuLeft = cy.get('.menu-left')
41     menuLeft.find('a:visible').first().should('have.text', 'Top rated movies')

```



See you in the next video

# Querying by text

# cy.contains()

- **Searches** for a DOM element **based on the text**
  - Based on a string or regular expression
- A string based search is **case sensitive**
  - Can be controlled with the `matchCase` option
- Can both be use as a **top level** or a **child command**

# cy.contains()

- Always **returns a single DOM element**
  - The first in case of multiple matches
- Sometimes **returns a higher level DOM element**
  - With buttons, links and labels
  - Can be controlled with a selector

home.spec.js



```
JS home.spec.js M x
cypress > integration > block-buster-films > JS home.spec.js > ...

38   it('Can navigate to Top rated movies', () => {
39     cy.contains('Top rated movies').click()
40     cy.title().should('equal', 'Top rated movies')
41   })
42
43   it('Can navigate to Popular movies', () => {
44     cy.contains(/^popular movies$/i).click()
45     cy.title().should('equal', 'Popular movies')
46   })
47
48   it('Can navigate to Celebrities', () => {
49     cy.get('.menu-left').contains('Celebrities').click()
50     cy.title().should('equal', 'Celebrities')
51   })
52 })
```

See you in the next video



# Aliases

# Aliases

- DOM queries can be **aliased for later reuse**
  - Alias a query:  
`cy.get('.menu-left a:visible').as('nav-links')`
  - Use the alias:  
`cy.get('@nav-links').eq(0).click()`
- Aliases can also be use with other concepts
  - AJAX requests, spies etc.
  - More about that in another video



# Aliases and Retries

- The query is executed when first defined
  - And the result reused later
  - Querying will happen again when assertions fail

Register an  
alias with `.as()`

```
JS home.spec.js M x
cypress > integration > block-buster-films > JS home.spec.js > ...
13   context('The left navigation menu', () => {
14     beforeEach(() => {
15       cy.get('.menu-left a:visible').as('nav-links')
16     })
17   })
```

# Using an alias with .get()

```
JS home.spec.js M x
cypress > integration > block-buster-films > JS home.spec.js > context('Block Buster Film Reviews') callback > context('The left navigation menu') callback
37     it('The navigation links have the correct text', () => {
38         cy.get('@nav-links').eq(0).should('have.text', 'Top rated movies')
39         cy.get('@nav-links').eq(1).should('have.text', 'Popular movies')
40         cy.get('@nav-links').eq(-1).should('have.text', 'Celebrities')
41     })
```

# The result



master-cypress-in-15-minutes-a x +

Not secure | block-buster-film-reviews.azureedge.net/\_/#/tests/integration/block-buster-films/home.spec.js

Chrome is being controlled by automated test software.

Tests 8 -- 1 01.69

https://block-buster-film-reviews.azureedge.net/ 1000 x 660 (82%)

cypress/integration/block-buster-films/home.spec.js

✓ The navigation links have the correct text

BEFORE EACH (1)

visit /

BEFORE EACH (2)

1 get .menu-left a:visible 1 nav-links

TEST BODY

1 get @nav-links 3

2 -eq 0

3 -assert expected <a> to have text Top rated movies

4 get @nav-links 3

5 -eq 1

6 -assert expected <a> to have text Popular movies

7 get @nav-links 3

8 -eq -1

9 -assert expected <a> to have text Celebrities

✓ Can navigate to Top rated movies

✓ Can navigate to Popular movies

✓ Can navigate to Celebrities

Anti-patterns

Resources DOM Snapshot (pinned) Cypress Course

Block Buster Film Reviews

TOP RATED MOVIES POPULAR MOVIES CELEBRITIES ABOUT

See you in the next video

# Interacting with Elements

# Common actions

- **Input actions:**
  - Typing into inputs
  - Checking checkboxes
  - Selecting options
  - Etc.
- **Mouse actions:**
  - Clicking
  - Scrolling
  - Etc.

See you in the next video



# Clicking DOM elements

# cy.click()

- With `cy.click()` you can **click on a DOM element**
  - All events are simulated by Cypress
- The element will **scroll into view**
- You can specify the **click position**
  - With coordinates or a position string
- Cypress checks if the **element is enabled and visible**
  - Use `{ force: true }` to override when needed
- <https://docs.cypress.io/api/commands/click>

top-rated-  
movies.spec.js



```
JS top-rated-movies.spec.js M x
cypress > integration > block-buster-films > JS top-rated-movies.spec.js > ...
31 |     it('Open Pulp Fiction', () => {
32 |         cy.get('#movie-680 .hvr-inner > a')
33 |             .click('bottomRight', { force: true })
34 |     })
35 | })
```

See you in the next video



Typing text

# cy.type()

- With **cy.type()** you can simulate typing
  - Works with `<input>` and `<textarea>`
- Special keys can be entered using **`{}` expressions**
  - Like `{enter}` or `{backspace}`
  - Or modifier keys like `{alt}` or `{ctrl}`
- There is a **10 ms delay** between key events
  - Makes the event flow more realistic
- <https://docs.cypress.io/api/commands/type>

# todo-list.spec.js

```
JS todo-list.spec.js U x
cypress > integration > demos > JS todo-list.spec.js > ...

3 context('To Do List', () => {
4   beforeEach(() => {
5     cy.visit('/')
6     cy.get('[data-testid=new-todo]').as('new-todo')
7     cy.get('[data-testid=btn-add-todo]').as('btn-add-todo')
8   })
9
10  it('Can add a new to-do with button', () => {
11    cy.get('@new-todo').type('A new to-do with button')
12    cy.get('@btn-add-todo').click()
13
14    cy.contains('li', 'A new to-do with button').should('be.visible')
15  })
}
```

# todo-list.spec.js

```
JS todo-list.spec.js U x
cypress > integration > demos > JS todo-list.spec.js > ...

17  it('Can add a new to-do with enter', () => {
18    cy.get('@new-todo').type('A new to-do with enter{enter}')
19
20    cy.contains('li', 'A new to-do with enter').should('be.visible')
21  })
22
23  it('Can add a new to-do in steps', () => {
24    cy.get('@new-todo')
25      .type('A ')
26      .type('to-do ')
27      .type('in ')
28      .type('steps')
29    cy.get('@btn-add-todo').click()
30
31    cy.contains('li', 'A to-do in steps').should('be.visible')
32  })
```



# settings.spec.js

```
JS settings.spec.js U x
cypress > integration > demos > JS settings.spec.js > ...

3 context('Settings', () => {
4   beforeEach(() => {
5     cy.visit('/')
6     cy.get('#view #settings').click()
7   })
8
9   it('Change the name to Jack Sparrow', () => {
10     cy.get('#firstName').clear().type('Jack')
11     cy.get('#lastName').type('{selectAll}Sparrow')
12     cy.get('#email').type('captain@black-pearl.ship')
13
14     cy.get('#result')
15       .should('contain.value', '"firstName": "Jack",')
16       .should('contain.value', '"lastName": "Sparrow",')
17       .should('contain.value', '"email": "captain@black-pearl.ship",')
18   })
19 })
```

todo-list.spec.js



```
JS todo-list.spec.js U x
cypress > integration > demos > JS todo-list.spec.js > ...

34  it('Can add {abc}', () => {
35    cy.get('@new-todo').type('Do {abc}', {
36      parseSpecialCharSequences: false
37    })
38    cy.get('@btn-add-todo').click()
39
40    cy.get('@new-todo').type('Do {{}}xyz{enter}')
41
42    cy.contains('li', 'Do {abc}').should('be.visible')
43  })
```

See you in the next video

# Radio buttons & checkboxes

# cy.check()

- With **cy.check()** you can check an item
  - Works with `<input type="radio">` and `<input type="checkbox">`
- <https://docs.cypress.io/api/commands/check>

settings.spec.js



```
JS settings.spec.js M x
cypress > integration > demos > JS settings.spec.js > ...
20  it('Change the gender to female', () => {
21    cy.get('#result').should('contain.value', '"gender": ""')
22
23    cy.get('#female').check()
24
25    cy.get('#result').should('contain.value', '"gender": "female"')
26  })
27
28  it('Select the newsletter', () => {
29    cy.get('#result').should('contain.value', '"newsletter": false')
30
31    cy.get('#newsletter').check()
32
33    cy.get('#result').should('contain.value', '"newsletter": true')
34  })
35  })
```

See you in the next video



# Select elements



# cy.select()

- With **cy.select()** you can select an option
  - Works with <select>
- Use either the **visible text or the value**
  - Use an array with multiselecting
- <https://docs.cypress.io/api/commands/select>

settings.spec.js



```
JS settings.spec.js M x
cypress > integration > demos > JS settings.spec.js > ...
38   it('Change the country to NL', () => {
39     cy.get('#result').should('contain.value', '"country": "US"')
40
41     cy.get('#country').select('Netherlands')
42
43     cy.get('#result').should('contain.value', '"country": "NL"')
44   })
45 })
```

See you in the next video



# Manipulating the value

`cy.select()`

settings.spec.js



See you in the next video

# Network Requests & Cypress



# Network Requests

- **Making network requests** with `cy.request()`
  - Dealing with cookies
  - Handling error responses
- **Intercepting network requests** with `cy.intercept()`
  - Waiting for request to have finished
  - Inspecting the request or response
  - Faking the response message

See you in the next video

# Continuous Integration

# Continuous Integration

- Run your Cypress tests as part of your CI process
  - Make sure all tests pass before merging a pull request
- Tests need to run against the local code in the PR
  - Run your application with a development server
- The steps can vary between different environments
  - But the same principal applies everywhere

See you in the next video

# Extending Cypress

# Extending Cypress

- Cypress can be extended with commands and tasks
- A custom command run in browser
  - Just like `cy.get()`
- A task runs in the Cypress backend Node process
  - Not limited by the browsers API

See you in the next video





# Cypress Testing Library

# Cypress Testing Library

*The more your tests resemble the way your software is used, the more confidence they can give you*

Kent C. Dodds

# Cypress Testing Library

- **Users search for a button** with the text “Save”
  - They don’t search for an element with the ID #btn-submit

# Cypress Testing Library

The screenshot shows the Cypress Testing Library documentation page in a web browser. The page has a dark green sidebar on the left with a navigation menu. The main content area is white with a dark green header. The header includes the title 'Cypress Testing Library' and a subtitle 'allows the use of dom-testing queries within Cypress end-to-end browser tests.' Below the header is a dark green box containing the command to install the library: `npm install --save-dev cypress @testing-library/cypress`. A link to the GitHub repository is provided. The 'Usage' section explains that the library extends Cypress's `cy` commands and provides an example of how to import the library into a project's `commands.js` file. A note mentions that `get*` queries are not supported and `find*` queries are preferred. The 'With TypeScript' section shows how to configure TypeScript for the library. The sidebar menu includes links to 'Getting Started', 'Core API', 'Frameworks', and 'Ecosystem'. The top of the page has a navigation bar with links to 'Help', 'Blog', and a search bar.

Cypress Testing Library | Testing | +  
testing-library.com/docs/cypress-testing-library/intro

Testing Library Docs Examples Help Blog Search

## Cypress Testing Library

Cypress Testing Library allows the use of dom-testing queries within Cypress end-to-end browser tests.

```
npm install --save-dev cypress @testing-library/cypress
```

- [Cypress Testing Library on GitHub](#)

### Usage

Cypress Testing Library extends Cypress's `cy` commands.

Add this line to your project's `cypress/support/commands.js`:

```
import '@testing-library/cypress/add-commands';
```

You can now use all of DOM Testing Library's `findBy`, `findAllBy`, `queryBy` and `queryAllBy` commands off the global `cy` object. See the [DOM Testing Library docs](#) for reference.

Note: the `get*` queries are not supported because for reasonable Cypress tests you need retryability and `find*` queries already support that. `query*` queries are no longer necessary since v5 and will be removed in v6. `find*` fully support built-in Cypress assertions (removes the only use-case for `query*`).

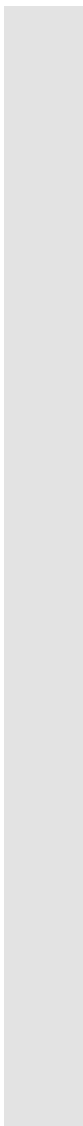
### With TypeScript

Typings should be added as follows in `tsconfig.json`:

```
{  "compilerOptions": {    "types": ["cypress", "@testing-library/cypress"]  }
```

See you in the next video

# Visual regression testing









See you in the next video







See you in the next video

Maurice de Beijer

@mauricedb

maurice.de.beijer  
@gmail.com

