

# Master Cypress in 15 minutes a day

Maurice de Beijer - @mauricedb

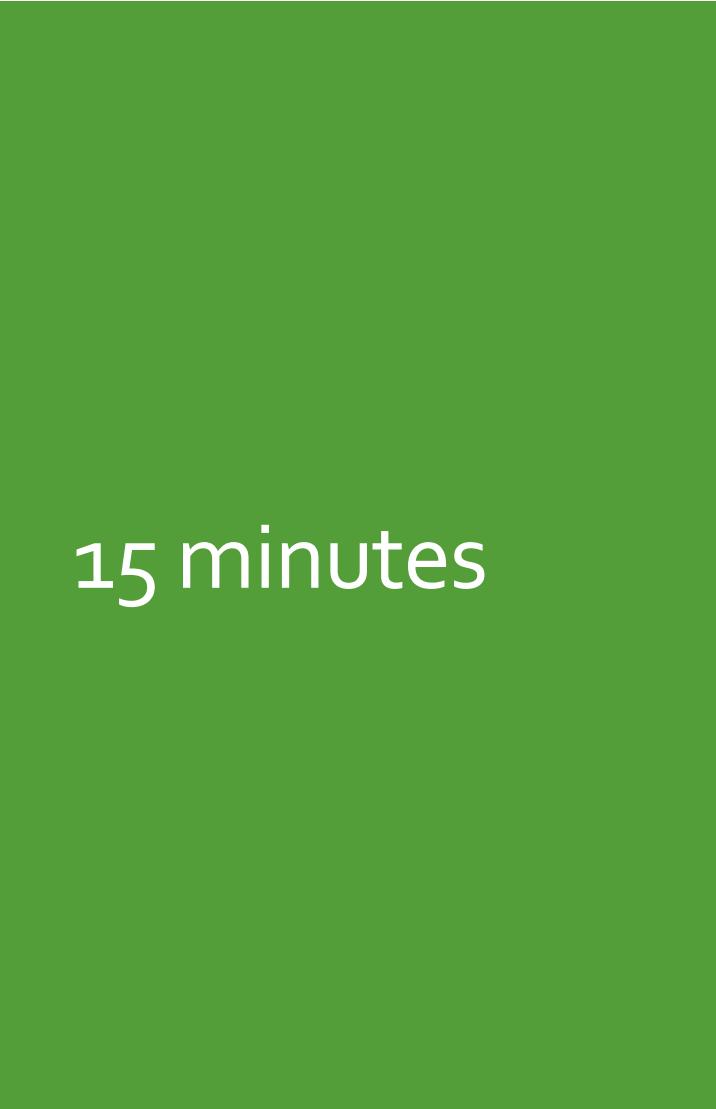
# Course Goals

# 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

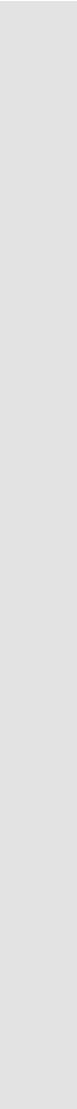


15 minutes

- 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

The screenshot shows the Cypress Interactive Test Runner interface. At the top, there's a header bar with the Cypress logo, the path "C:\Temp\c", and standard window controls (minimize, maximize, close). Below the header is a dark navigation bar with links for "File", "Edit", "View", "Window", and "Help". To the right of the navigation bar are "Support", "Docs", and "Log In" links. The main content area has tabs for "Tests" (which is selected), "Runs", and "Settings". A search bar at the top of the content area says "Press Ctrl + F to search...". On the right side of the search bar are "New Spec File" and "Run 20 integration specs" buttons. The main list area is titled "INTEGRATION TESTS" with "COLLAPSE ALL | EXPAND ALL" links. It contains a single expanded category "examples" which lists several spec files: "actions.spec.js", "aliasing.spec.js", "assertions.spec.js", "connectors.spec.js", "cookies.spec.js", "cypress\_api.spec.js", "files.spec.js", "local\_storage.spec.js", and "location.spec.js". At the bottom of the interface is a footer bar with "Version 7.5.0" and "Changelog" links.

# Interactive Test Runner

The screenshot shows a web browser window with the Cypress UI. On the left, the test runner interface displays a list of test cases and their results. One test case is expanded, showing the test code:

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

On the right, the browser displays the movie page for "The Shawshank Redemption" (1994). The page includes a large poster image, a rating of 8.7/10, and a five-star rating. Below the poster, there are tabs for OVERVIEW, REVIEWS, CAST & CREW, MEDIA, and SIMILAR MOVIES. The OVERVIEW tab is active, showing a brief plot summary and credits for Director, Writer, and Genres.

# 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.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

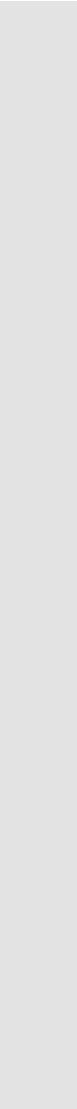
- Easy to get started
-  Fast execution
- Automatic waiting on elements
- Access to elements outside the DOM
- Support for FireFox and Chromium based browsers
- No driver dependencies
-  Time travel and debuggability
- Uses the Mocha test framework
- Open source
-  Mostly free

## Cons

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



See you in the next video





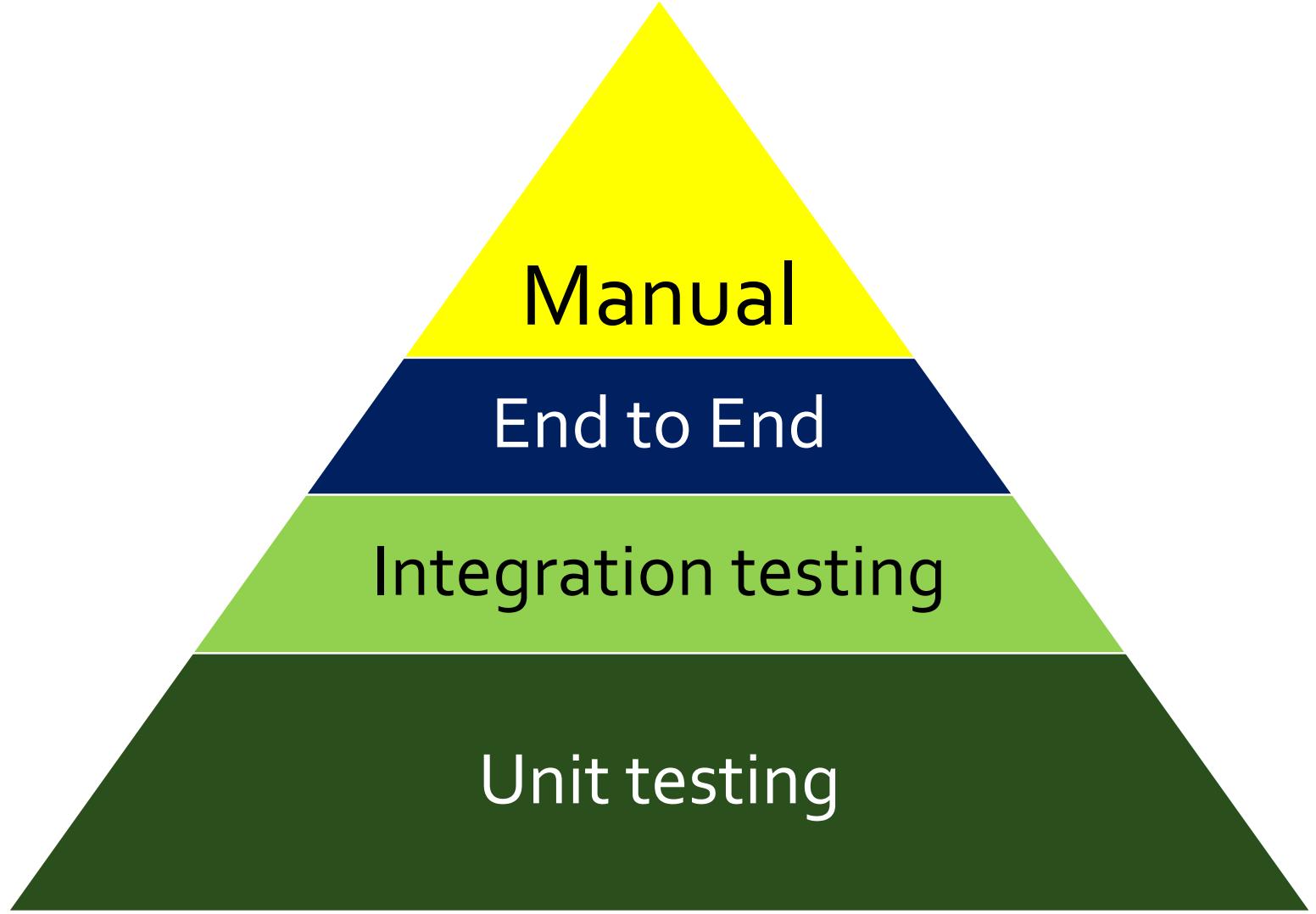
# What is end to end testing

# 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

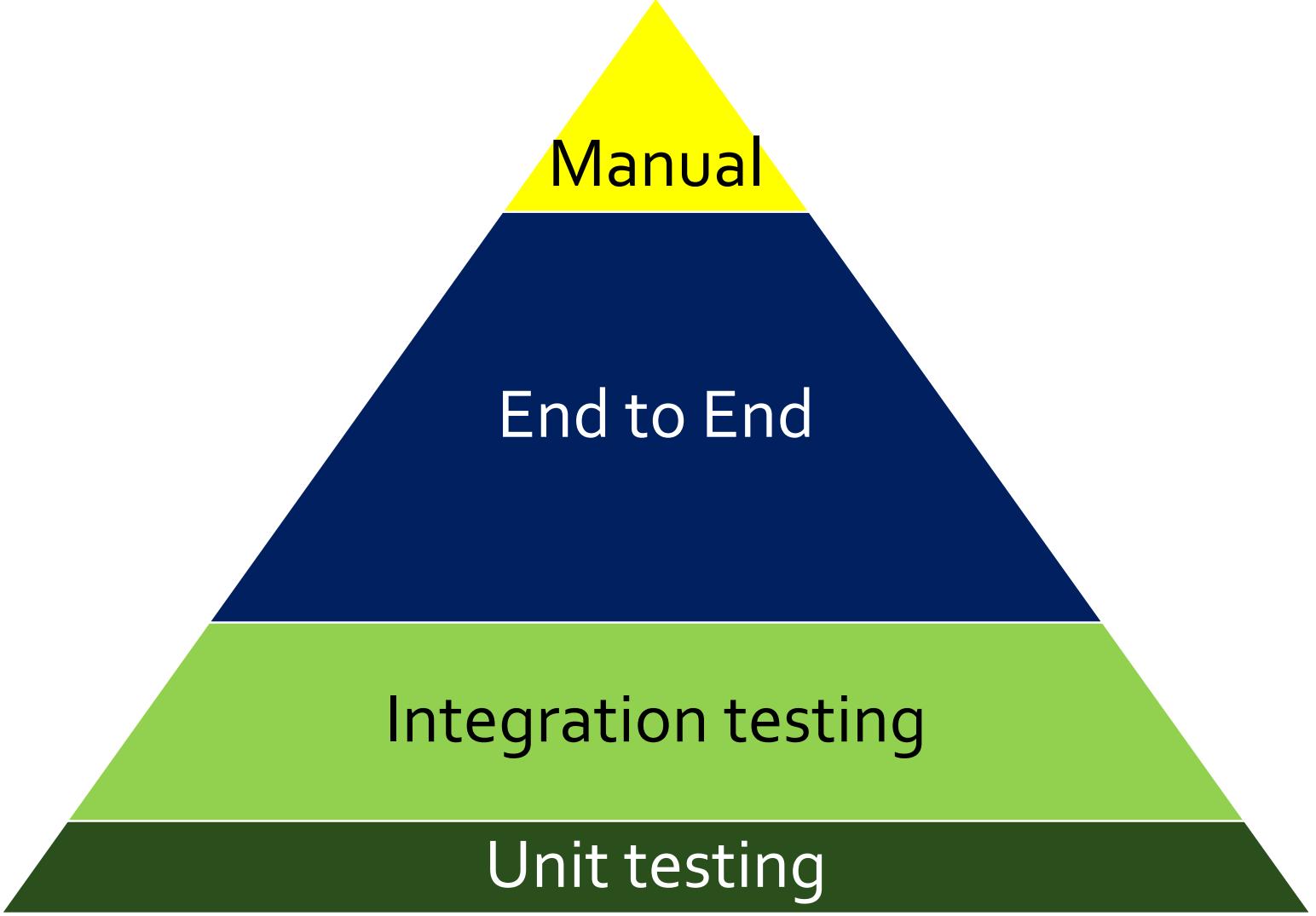
[Source](#)

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.”*

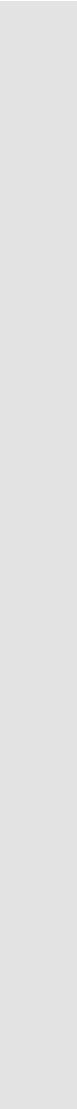
-- Techopedia --

# 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

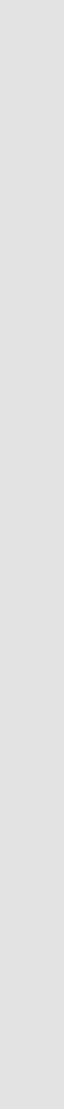








Happily married





# Independent software developer & instructor

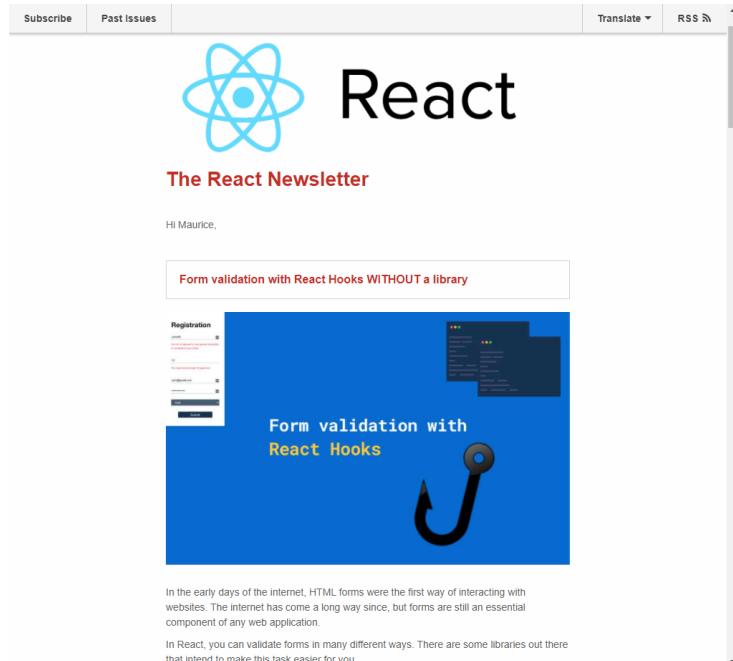
Since 1995





Microsoft®  
Most Valuable  
Professional

# The React Newsletter

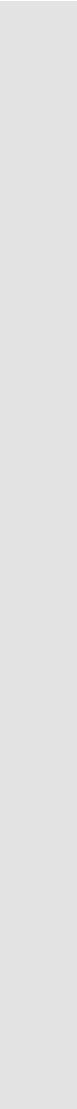


The screenshot shows an email from "The React Newsletter". The header includes links for "Subscribe", "Past Issues", "Translate", and "RSS". The main content features the React logo and the word "React". Below that is the heading "The React Newsletter". A personal greeting "Hi Maurice," follows. A section titled "Form validation with React Hooks WITHOUT a library" contains a screenshot of a registration form and a blue background with the text "Form validation with React Hooks" and a fishhook icon. A note at the bottom explains that while forms were the first way to interact with websites, they remain essential. It also mentions that in React, form validation can be done using hooks.





See you in the next video

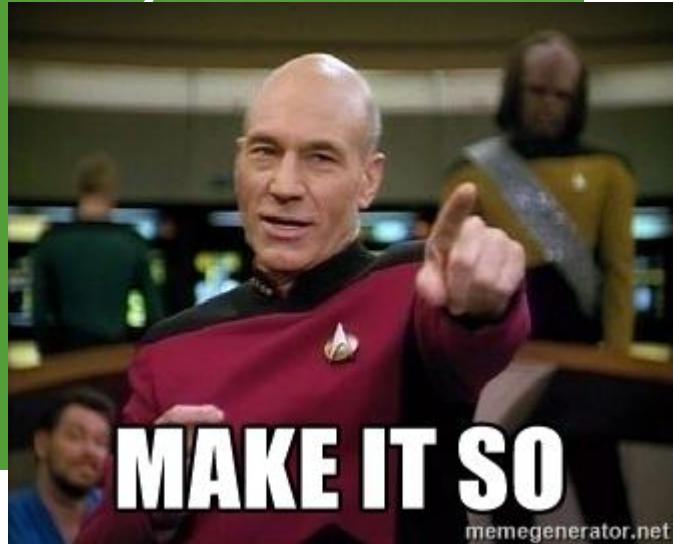


# Prerequisites

Install Node & NPM

Install the GitHub starter repository

# Following Along



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

```
JS block-buster.spec.js ✘
cypress > integration > JS block-buster.spec.js > ...
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 })
```



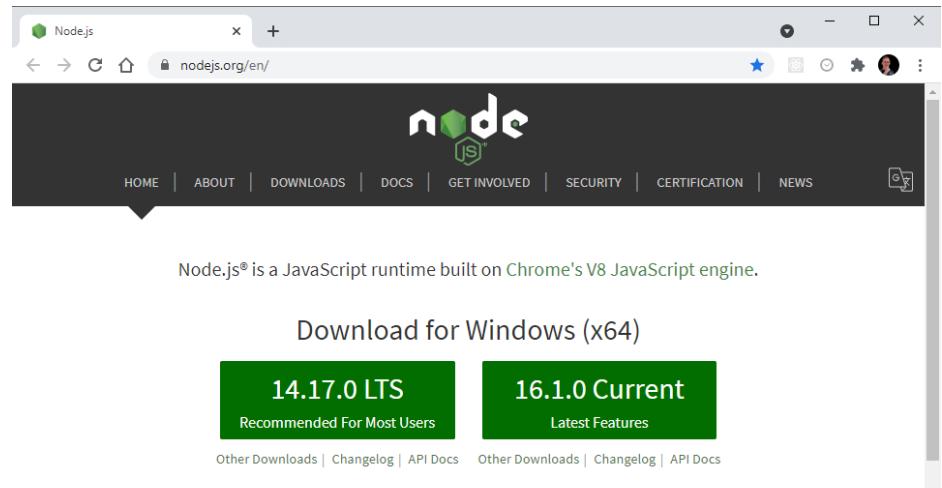
Type it out  
by hand?

*"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!"*

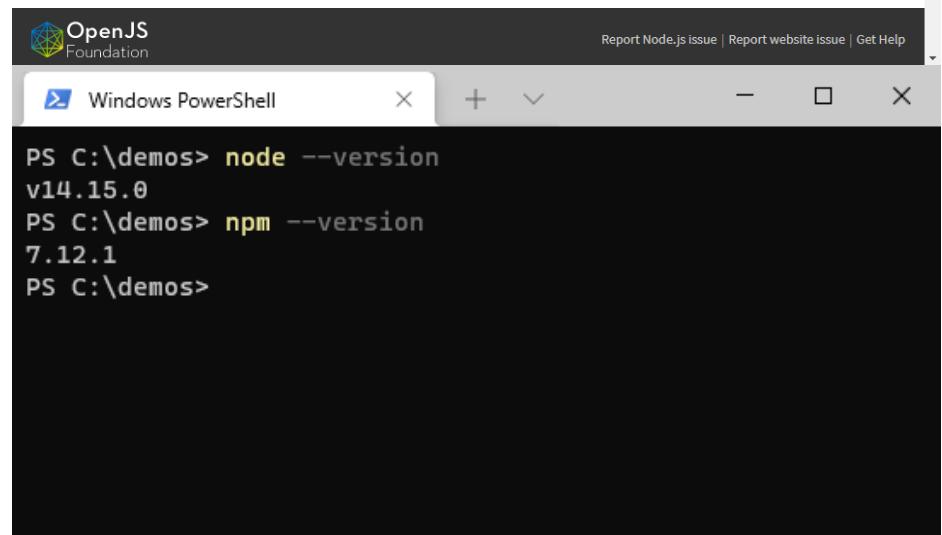
# Install Node.js



- Minimal:
  - Node version 12



The screenshot shows the official Node.js website at nodejs.org/en/. The page features the Node.js logo and navigation links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. A callout box highlights the "14.17.0 LTS" version as "Recommended For Most Users". Another button for "16.1.0 Current" is also visible. Below the download section, a note says "Or have a look at the Long Term Support (LTS) schedule." Navigation links for "Other Downloads", "Changelog", "API Docs", and "Report Node.js issue" are present.

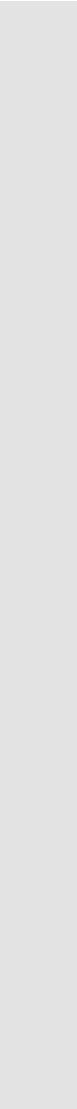


The screenshot shows a Windows PowerShell window titled "Windows PowerShell" running on Windows 10. The command "PS C:\demos> node --version" was run, outputting "v14.15.0". The command "PS C:\demos> npm --version" was also run, outputting "7.12.1". The OpenJS Foundation logo is visible in the top left corner of the slide.

```
PS C:\demos> node --version
v14.15.0
PS C:\demos> npm --version
7.12.1
PS C:\demos>
```

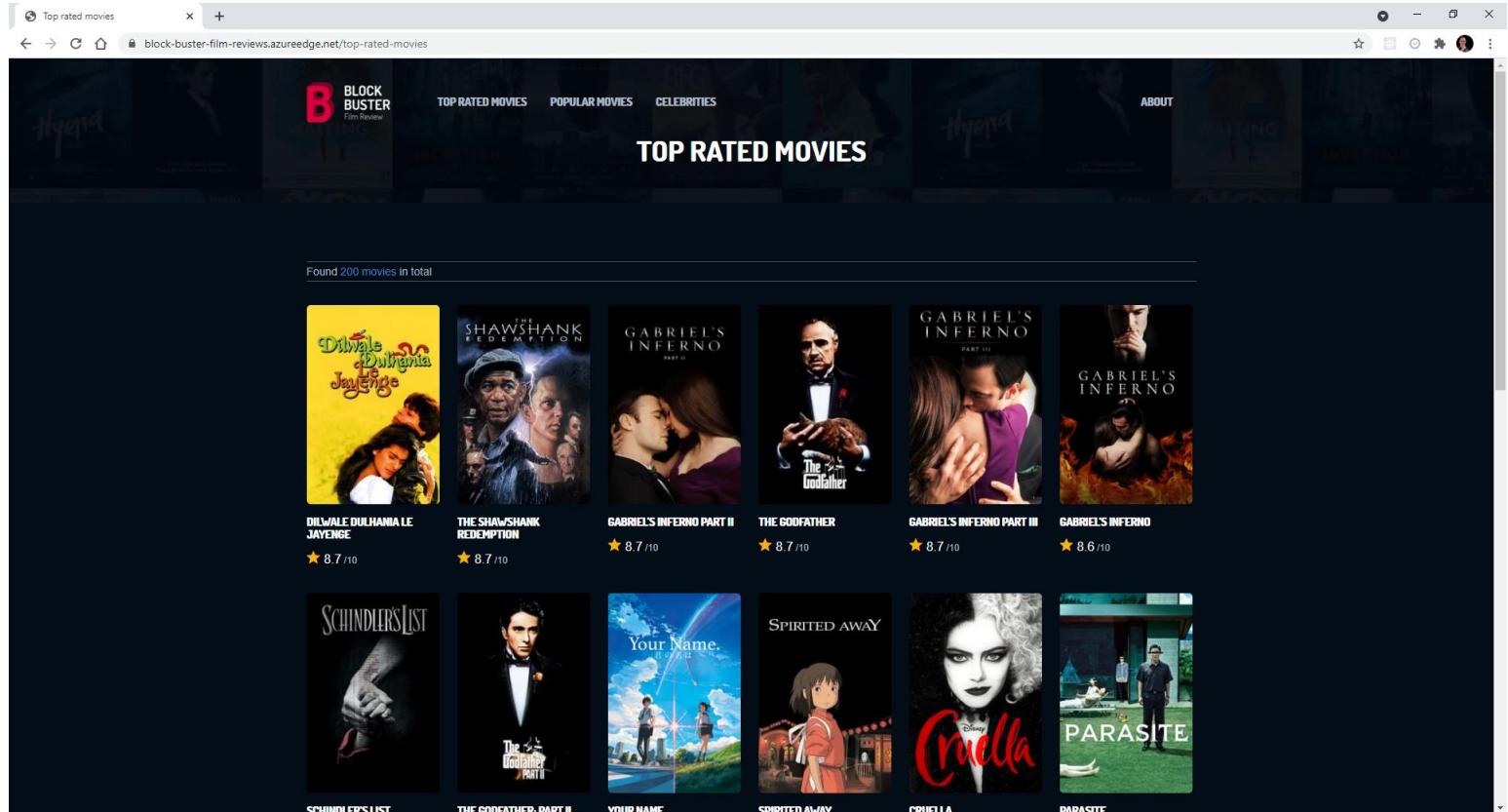


See you in the next video



# 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

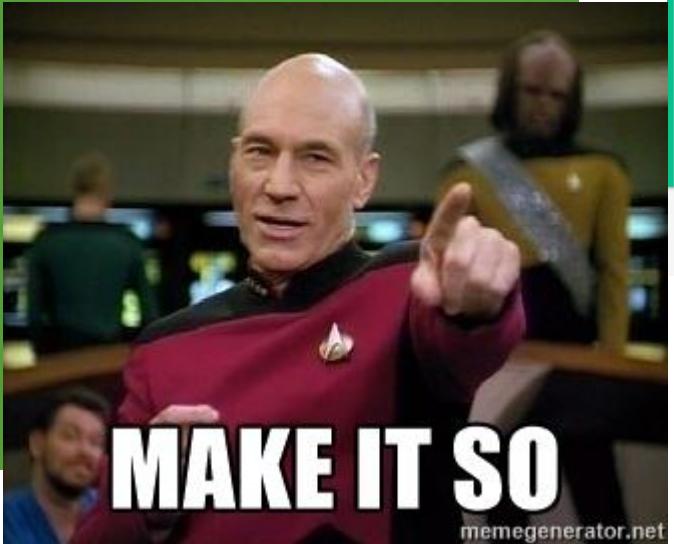
# Creating a first Cypress test

```
JS home.spec.js ×  
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

The screenshot shows the Cypress Test Runner application window. The title bar indicates the path: C:\Temp\master-cypress-in-15-minutes-a-day. The menu bar includes File, Edit, View, Window, and Help. The top right features links for Support, Docs, and Log In, along with a browser dropdown set to Chrome 91. The main interface has tabs for Tests (selected), Runs, and Settings. A search bar says "Press Ctrl + F to search...". On the right, there's a button for "+ New Spec File" and a link to "Run 21 integration specs". The left sidebar lists "INTEGRATION TESTS" with "COLLAPSE ALL | EXPAND ALL" options. Under "block-buster-films", the file "home.spec.js" is listed with an "Open in IDE" link. Under "examples", several files are listed: actions.spec.js, aliasing.spec.js, assertions.spec.js, connectors.spec.js, cookies.spec.js, cypress\_api.spec.js, and files.spec.js. At the bottom right, there are links for "Version 7.5.0" and "Changelog".

# Running the test



Chrome is being controlled by automated test software.

Tests    ✓ 1    ✖ 0    01.34    ⏴ ⏵ ⏵

https://block-buster-film-reviews.azureedge.net/ 1000 x 660 (53%) ⓘ

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

Block Buster Film Reviews - Home Page

Has the correct title

TEST BODY

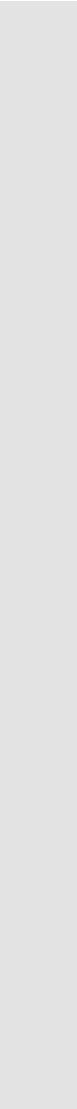
```
1 visit https://block-buster-film-reviews.azuree...
2 title
3 - assert expected Block Buster Film Reviews to
   equal Block Buster Film Reviews
4 get h1
5 - assert expected <h1> to have text Block Buster
   Film Reviews
```

B BLOCK BUSTER TOP RATED MOVIES POPULAR MOVIES CELEBRITIES ABOUT BLOCK BUSTER FILM REVIEWS

The screenshot shows a Cypress test runner window. The title bar says "master-cypress-in-15-minutes-a...". The address bar shows a "Not secure" warning and the URL "block-buster-film-reviews.azureedge.net/\_/#/tests/integration/block-buster-films/home.spec.js". The main area displays the test results: 1 test passed (✓) and 0 failed (✖). Below the results, the test code is shown, which visits the site and asserts that the title is "Block Buster Film Reviews". On the right, a screenshot of the "BLOCK BUSTER FILM REVIEWS" website is displayed, showing the homepage with the title and navigation menu.



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

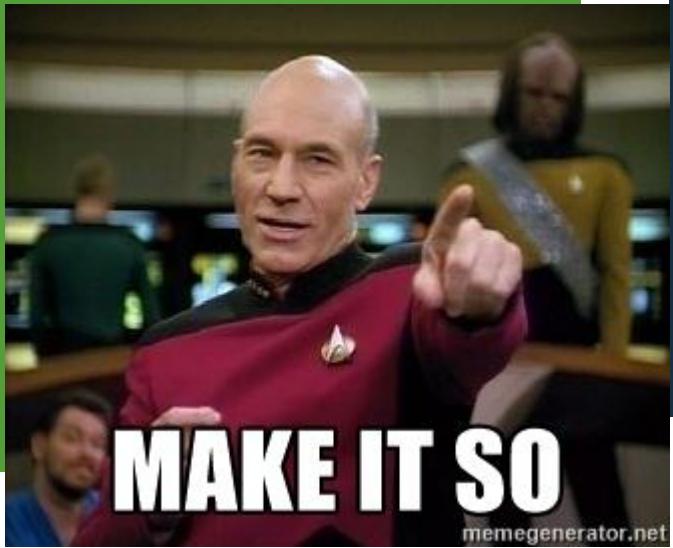
# Adding dependencies

- **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

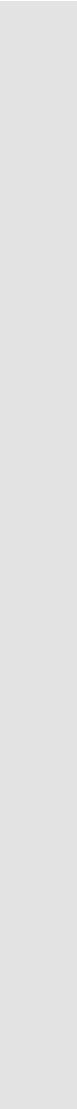


```
package.json x
package.json > ...
1  {
2    "name": "master-cypress-in-15-minutes-a-day",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    ▷ Debug
7    "scripts": {
8      "test": "cypress run",
9      "open": "cypress open"
10     },
11    "repository": {
12      "type":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

---

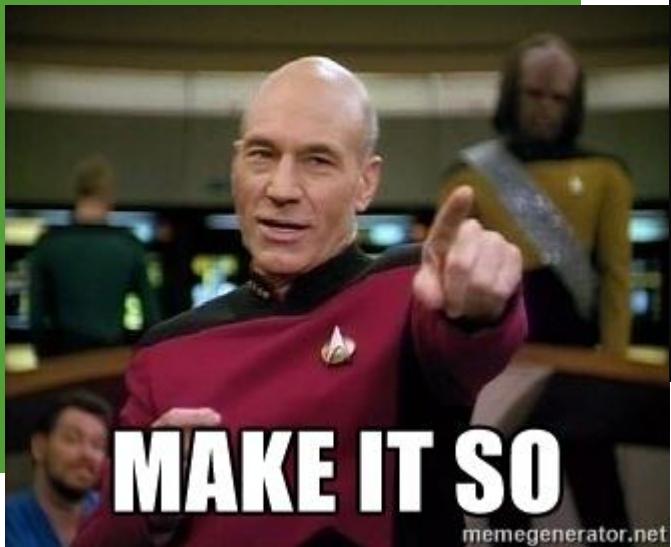
As of version [3.0](#), Cypress downloads the matching Cypress binary to the global system cache, so that the binary can be shared between projects. By default, global cache folders are:

- **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



MAKE IT SO

memegenerator.net

```
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

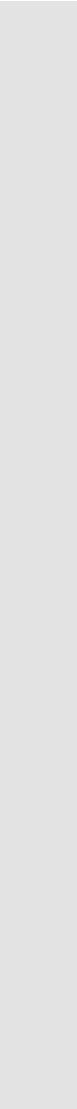
version      last used      size
6.0.1        26 minutes ago 516.1MB
6.5.0        23 minutes ago 609.0MB
6.9.1        20 minutes ago 579.1MB
7.0.1        17 minutes ago 587.4MB
7.3.0        12 minutes ago 574.9MB
7.5.0        14 minutes ago 575.4MB
7.6.0        unknown        444.3MB

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

cypress



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 element is visible and enabled
  - A value is equal to another
  - Many more...
- <https://docs.cypress.io/api/commands/should>

# Asynchronous

- Commands are asynchronous
  - They are 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

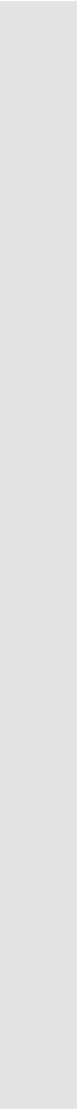
# Top rated movies Specs



```
JS top-rated-movies.spec.js ✘
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   })
})
```



See you in the next video

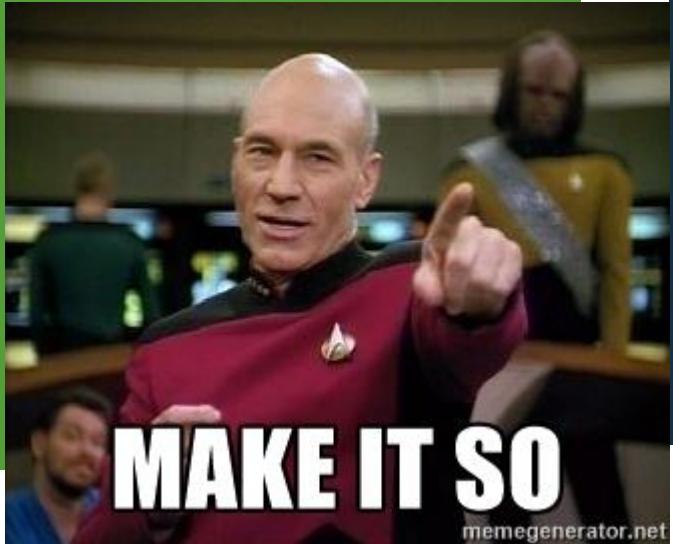


# Adding IntelliSense

# IntelliSense

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

# IntelliSense



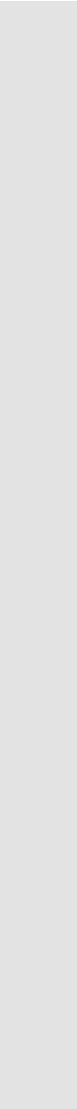
You, 2 minutes ago | 1 author (You)

```
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
```

The code is a Cypress test for a web application. It starts by defining a context named 'Block Buster Film Reviews'. Inside this context, there is an 'it' block with the title 'Has the correct title - Home Page'. The test first visits the URL 'https://block-buster-film-reviews.azureedge.net'. Then, it checks the title of the page using 'cy.title()' and asserts that it equals 'Block Buster Film Reviews'. Next, it uses 'cy.get('h1')' to select the main heading and asserts that its text is 'Block Buster Film Reviews'. A tooltip is shown over the 'have.text' method, listing various related methods such as 'have.nested.property', 'have.returned', and 'have.length.gte'.



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

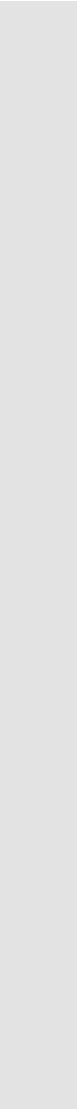
# Using beforeEach()



```
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   beforeEach(() => {
12     cy.visit('https://block-buster-film-reviews.azureedge.net/top-rated-movies')
13   })
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`

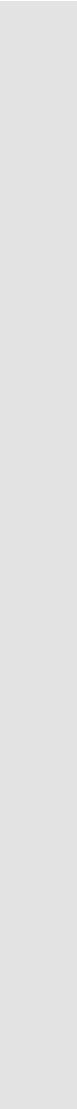
# cypress.json



```
{} 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>	<span style="color: red;">⚠ Never</span>	Worst - too generic, no context.
<code>cy.get('.btn.btn-large').click()</code>	<span style="color: red;">⚠ Never</span>	Bad. Coupled to styling. Highly subject to change.
<code>cy.get('#main').click()</code>	<span style="color: orange;">⚠ Sparingly</span>	Better. But still coupled to styling or JS event listeners.
<code>cy.get('[name=submission]').click()</code>	<span style="color: orange;">⚠ Sparingly</span>	Coupled to the <code>name</code> attribute which has HTML semantics.
<code>cy.contains('Submit').click()</code>	<span style="color: green;">✓ Depends</span>	Much better. But still coupled to text content that may change.
<code>cy.get('[data-cy=submit]').click()</code>	<span style="color: green;">✓ Always</span>	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>

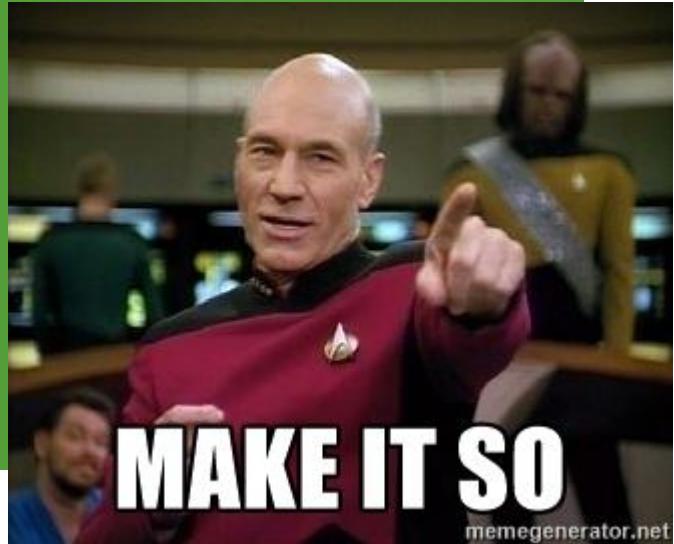
# .within()

- 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>

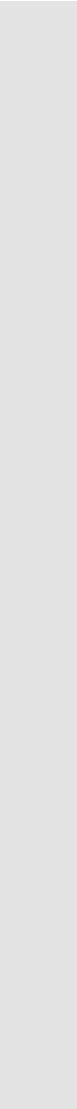
# home.spec.js



```
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 used 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

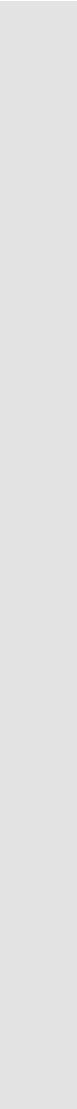
# home.spec.js



```
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 used 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

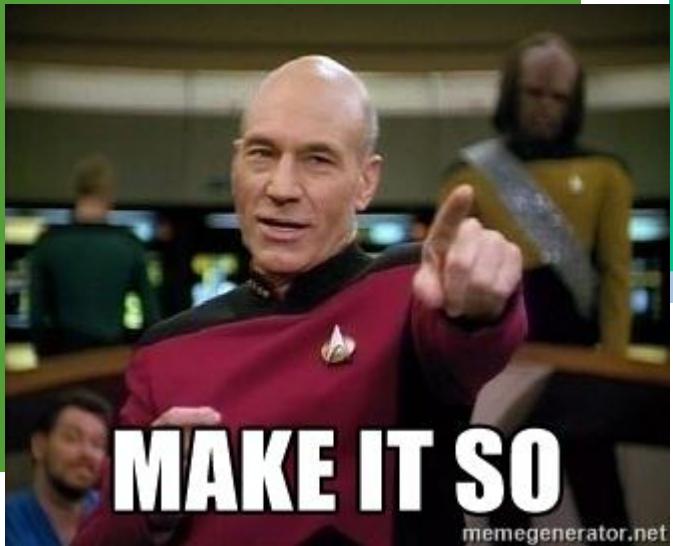
# Register an alias with .as()

```
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 |     })
  
```

# 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 ✘ 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 0 01.69

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

✓ The navigation links have the correct text

```
visit /
get .menu-left a:visible
get @nav-links
eq 0
assert expected <a> to have text Top rated movies
get @nav-links
eq 1
assert expected <a> to have text Popular movies
get @nav-links
eq -1
assert expected <a> to have text Celebrities
```

✓ Can navigate to Top rated movies  
✓ Can navigate to Popular movies  
✓ Can navigate to Celebrities

Anti-patterns

B BLOCK BUSTER Film Review TOP RATED MOVIES POPULAR MOVIES CELEBRITIES ABOUT

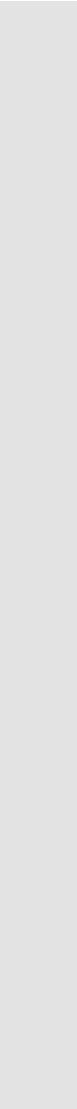
BLOCK BUSTER FILM REVIEWS

Resources DOM Snapshot (pinned) Cypress Course

A screenshot of a web browser window showing a Cypress test run. The test has passed with 8 green checkmarks and 0 red Xs. The test file is "cypress/integration/block-buster-films/home.spec.js". The test checks if the navigation links have the correct text. It visits the homepage, gets the ".menu-left a:visible" links, and asserts that the first link has text "Top rated movies", the second "Popular movies", and the third "Celebrities". Below the test results, there are three green checkmarks indicating successful navigation. The right side of the screen shows the "BLOCK BUSTER FILM REVIEWS" website with its navigation menu and some movie posters.



See you in the next video



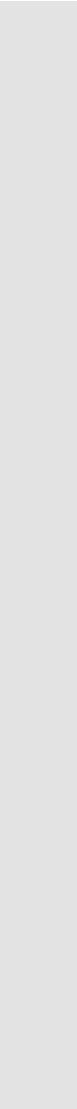
# Interacting with Elements

# Common actions

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



See you in the next video

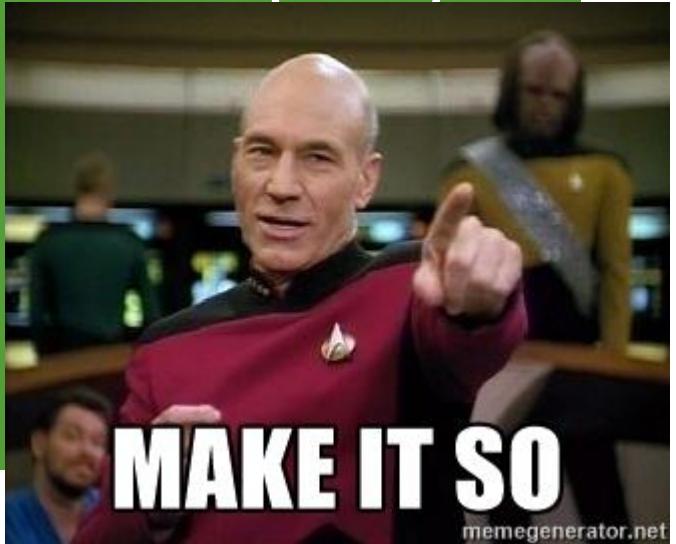


# Clicking DOM elements

# 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



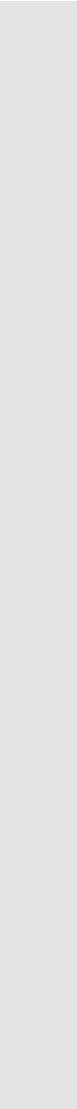
MAKE IT SO

memegenerator.net

```
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 ✘ 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

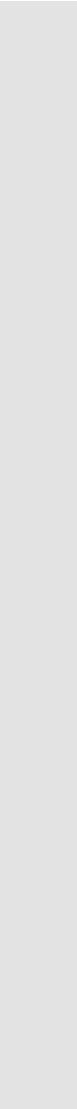


```
JS todo-list.spec.js ✘ X
cypress > integration > demos > JS todo-list.spec.js > ...
34  it('Can add {abc}', () => {
35    cy.get('@new-todo').type('Do {abc}', {
36      parseSpecialCharSequences: false
37    })
38    cy.get('@btn-add-todo').click()
39
40    cy.get('@new-todo').type('Do {{}xyz}{enter}')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





See you in the next video

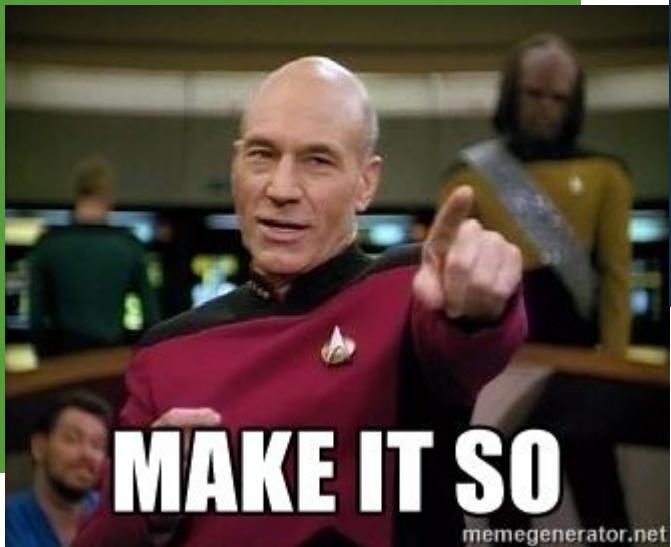


# Radio buttons & checkboxes

# 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>

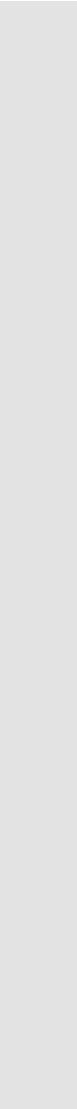
# settings.spec.js



```
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>

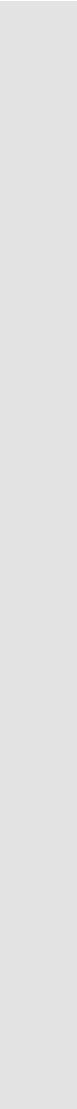
# settings.spec.js



```
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

# Manipulating Values

- Some input elements are **harder to manipulate** like an end user
  - For example:
    - <input type="date"/>
    - <input type="range"/>
- **Manipulate the value directly** using:
  - `cy.get("input[type=range]").invoke("val", 25).trigger("change")`
- Note: Sometimes you need to trigger the **input event**
  - Instead of **change event**

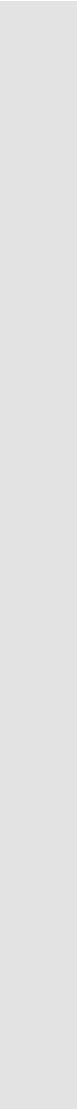
# settings.spec.js



```
JS settings.spec.js M X
cypress > integration > demos > JS settings.spec.js > ...
46  it('Change the happiness to 95%', () => {
47    cy.get('#result').should('contain.value', '"happiness": 80')
48
49    cy.get('#happiness').invoke('val', 95).trigger('input')
50
51    cy.get('#result').should('contain.value', '"happiness": 95')
52  })
53})
```



See you in the next video





# Validating the value

# Validating the value

- **Validating inputs** is normally done using the `.should()` function
- Many **validation options** you can use
  - Have a specific value
  - Is checked or not checked
  - Is empty
  - Etc.
- <https://docs.cypress.io/api/commands/should>

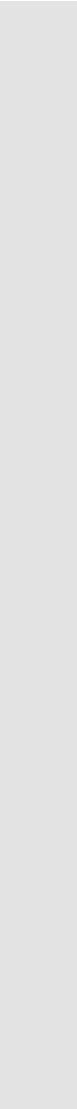
# settings.spec.js



```
JS settings.spec.js M X
cypress > integration > demos > JS settings.spec.js > ...
9   it('Change the name to Jack Sparrow', () => {
10     cy.get('#firstName')
11       .clear()
12       .type('Jack')
13       .should('have.value', 'Jack')
14     cy.get('#lastName')
15       .type('{selectAll}Sparrow')
16       .should('have.value', 'Sparrow')
17     cy.get('#email')
18       .type('captain@black-pearl.ship')
19       .should('have.value', 'captain@black-pearl.ship')
20   })
21
22   it('Change the gender to female', () => {
23     cy.get('#unknown')
24       .should('be.checked')
25
26     cy.get('#female')
27       .check()
28       .should('be.checked')
29
30     cy.get('#unknown')
31       .should('not.be.checked')
32   })
```



See you in the next video



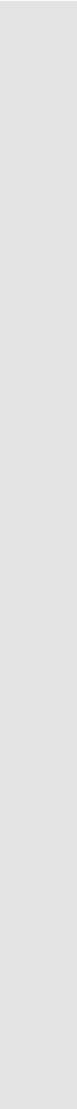
# Network Requests & Cypress

# 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



# Book Search

# Book Search

Book Search x +

book-search.azureedge.net/search/intitle/The+Hitchhiker's+Guide+to+the+Galaxy

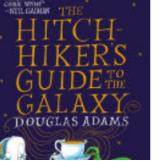
Book Search

The Hitchhiker's Guide to the Galaxy

Title

 [The Ultimate Hitchhiker's Guide to the Galaxy](#)  
... The Hitchhiker's Guide to the Galaxy One Thursday lunchtime the Earth gets unexpectedly demolished to make way for a new hyperspace bypass.  
[Douglas Adams](#)

 [The Hitchhiker's Guide to the Galaxy](#)  
The intergalactic adventures of Arthur Dent begin in the first volume of the 'trilogy of five', Douglas Adams' comedy sci-fi classic The Hitchhiker's Guide to the Galaxy.  
[Douglas Adams](#)

 [The Hitchhiker's Guide to the Galaxy: The Illustrated Edition](#)  
This beautifully illustrated edition of the New York Times bestselling classic celebrates the 42nd anniversary of the original publication—with all-new art by award-winning illustrator Chris Riddell.  
[Douglas Adams](#)

# GitHub

The screenshot shows a GitHub repository page for the project 'mauricedb/book-search'. The repository is public and has 1 branch and 0 tags. The master branch contains 30 commits. The repository was initialized using Create React App. Available scripts include 'yarn start' (Run the app in the development mode). The code is licensed under MIT. The repository has no description, website, or topics provided.

mauricedb/book-search

mauricedb / book-search (Public)

No description, website, or topics provided.

Code Issues Pull requests Actions Projects Wiki 0 Settings

master 1 branch 0 tags

mauricedb No default title e863002 42 minutes ago 30 commits

.github/workflows Deploy to Static Website on Azure Blob Storage 11 months ago

public 404 and title 11 months ago

src No default title 42 minutes ago

.gitignore Initialize project using Create React App 15 months ago

.prettierrc Add prettier 11 months ago

LICENSE Initial commit 15 months ago

README.md Initialize project using Create React App 15 months ago

package-lock.json ncu -u yesterday

package.json ncu -u yesterday

tsconfig.json ncu -u yesterday

This project was bootstrapped with [Create React App](#).

**Available Scripts**

In the project directory, you can run:

**yarn start**

Run the app in the development mode

# search.spec.js

```
JS search.spec.js 4, U X
cypress > integration > JS search.spec.js > ...
1  /// <reference types="cypress" />
2
3  describe('Search', () => {
4    beforeEach(() => {
5      cy.visit('/');
6    });
7
8    it('For books by Douglas Adams', () => {
9      cy.get('select').select('inauthor');
10   cy.get('input').type('Douglas Adams{enter}');
11
12   cy.contains('.card-title', "The Hitchhiker's Guide to the Galaxy").should(
13     'be.visible'
14   );
15 });
16});
```

# Cypress open



book-search

localhost:3000/\_/#/tests/integration/search.spec.js

Chrome is being controlled by automated test software.

Tests 1 01.22

cypress/integration/search.spec.js

Search

For books by Douglas Adams

BEFORE EACH

visit /

TEST BODY

get select inauthor

get input

- type Douglas Adams{enter}

(new url) http://localhost:3000/search/inauthor/Douglas...

(fetch) GET 200 https://www.googleapis.com/books/v1/volume...

contains .card-title, The Hitchhiker's Guide to the Ga...

- assert expected <h5.card-title> to be visible

(fetch) GET https://www.googleapis.com/books/v1/volumes?q=...

http://localhost:3000/search/inauthor/Douglas+Adams

1000 x 660 (96%)

Book Search

Douglas Adams

**Dirk Gently's Holistic Detective Agency**

From Douglas Adams, the legendary author of one of the most beloved science fiction novels of all time, The Hitchhiker's Guide to the Galaxy, comes a wildly inventive novel of ghosts, time travel, and one detective's mission to save ...

[Douglas Adams.](#)

**The Salmon of Doubt**

This sublime collection dips into the wit and wisdom of the man behind The Hitchhiker's Guide to the Galaxy, uncovering his unique comic musings on everything from his school-trousers to malt whisky and from the letter Y through to his own ...

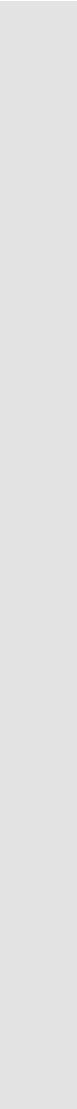
[Douglas Adams.](#)

**The Hitchhiker's Guide to the Galaxy: The Illustrated Edition**

This beautifully illustrated edition of the New York Times bestselling classic celebrates the 42nd anniversary of the original publication—with all-new art by award-winning illustrator Chris Riddell



See you in the next video

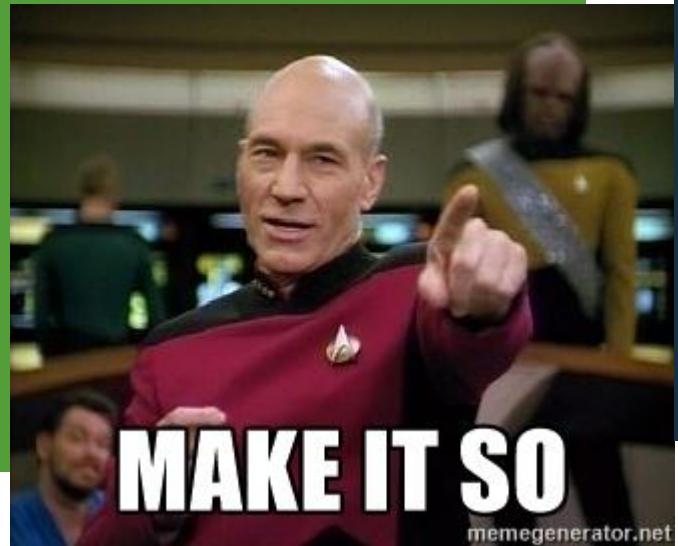


# Intercepting Requests

# cy.intercept()

- With `cy.intercept()` you can **spy on network requests**
  - Both XMLHttpRequest and fetch requests
- **Enables waiting for** network requests to **finish**
  - Before proceeding with a test
- **Intercept AJAX requests**
  - Assert the request message
  - Fake response messages
- **Intercept AJAX responses**
  - Delay a response
  - Use the response in the test body
- <https://docs.cypress.io/api/commands/intercept>

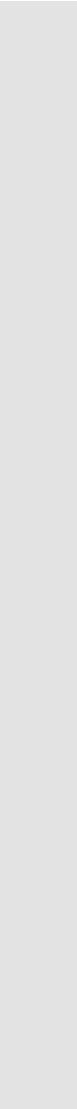
# cy.intercept()



```
JS search.spec.js M X
cypress > integration > JS search.spec.js > ...
8  it('intercept with a StringMatcher', () => {
9    cy.intercept('/books/v1/volumes?*').as('book-search');
10
11   cy.get('select').select('inauthor');
12   cy.get('input').type('Douglas Adams{enter}');
13
14   cy.contains('.card-title', "The Hitchhiker's Guide to the Galaxy").should(
15     'be.visible'
16   );
17 });
18
19 it('intercept with a RouteMatcher', () => {
20   cy.intercept({
21     hostname: 'www.googleapis.com',
22     method: 'get',
23     pathname: '/books/v1/volumes',
24   }).as('book-search');
25
26   cy.get('select').select('inauthor');
27   cy.get('input').type('Douglas Adams{enter}');
28
29   cy.contains('.card-title', "The Hitchhiker's Guide to the Galaxy").should(
30     'be.visible'
31   );
32 });
```



See you in the next video



# Cypress[minimatch()]

## Cypress.minimatch()

- Cypress uses the [minimatch](#) NPM package to match URL's
- Can be used to debug URL patterns
  - Available using Cypress.`minimatch()`
- <https://docs.cypress.io/api/utilities/minimatch>
- <https://github.com/isaacs/minimatch#minimatch>

# Cypress[minimatch()]



cypress/integration/search.spec.js

```
3 get      input
4 - type    Douglas Adams{enter}
(new url)  http://localhost:3000/search/inauthor:Douglas+Adams
(fetch)    GET 200 https://www.googleapis.com/books/v1/volumes?q=inauthor:Douglas+Adams (book-search)
5 contains .card-title, The Hitchhiker's Guide to the Galaxy
6 - assert expected <h5.card-title> to be visible
```

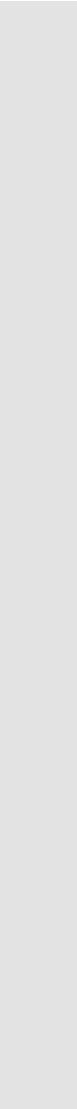
DOM

Console was cleared

Event: request  
Resource type: fetch  
Method: GET  
Url: <https://www.googleapis.com/books/v1/volumes?q=inauthor:Douglas+Adams&maxResults=25&langRestrict=en&filter=ebooks>  
Request went to origin?: yes  
Matched `cy.intercept()`: ► {RouteMatcher: {...}, RouteHandler Type: 'Spy', RouteHandler: undefined, Request: {...}, Response: {...}, ...}  
Request headers:  
► {sec-ch-ua: "Google Chrome";v="93", "Not;A Brand";v="99", "Chromium";v="93", Referer: 'http://localhost:3000/', sec-ch-ua-mobile: '?0', User-Agent: sec-ch-ua-platform: "Windows"}  
Response status code: 200  
Response headers: ► {date: 'Fri, 24 Sep 2021 07:08:02 GMT', content-encoding: 'gzip', x-content-type-options: 'nosniff', server: 'ESF', Vary: 'Or  
Response body: ► {kind: 'books#volumes', totalItems: 300, items: Array(25)}  
> Cypress.minimatch('/books/v1/volumes?q=inauthor:Douglas+Adams&maxResults=25&langRestrict=en&filter=ebooks', '/books/v1/volumes?\*', { matchBase: true })  
< true  
> Cypress.minimatch('/books/v1/volumes?q=inauthor:Douglas+Adams&maxResults=25&langRestrict=en&filter=ebooks', '/books/v1/volumes', { matchBase: true })  
< false  
> Cypress.minimatch('/books/v1/volumes?q=inauthor:Douglas+Adams&maxResults=25&langRestrict=en&filter=ebooks', '/volumes?\*', { matchBase: true })  
< false  
> Cypress.minimatch('/books/v1/volumes?q=inauthor:Douglas+Adams&maxResults=25&langRestrict=en&filter=ebooks', '/\*\*/volumes?\*', { matchBase: true })  
< false  
> Cypress.minimatch('/books/v1/volumes?q=inauthor:Douglas+Adams&maxResults=25&langRestrict=en&filter=ebooks', '\*\*/\*\*/volumes?\*', { matchBase: true })  
< true



See you in the next video



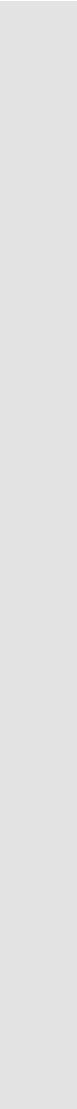
# cy.route() & cy.server()

# cy.route() cy.server()

- Both `cy.route()` and `cy.server()` have been **deprecated**
  - Use `cy.intercept()` instead!



See you in the next video



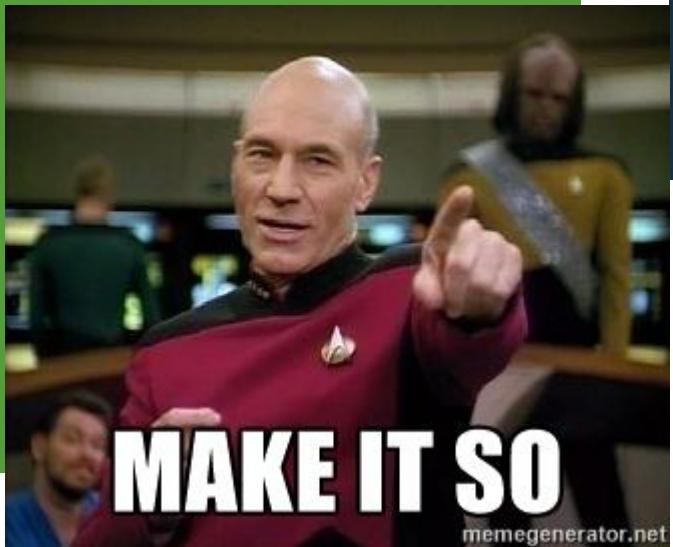


Waiting for a response

# cy.wait()

- Waits for a network request to complete
  - It can wait on multiple requests
- Fails the test if the requests doesn't complete in time
- <https://docs.cypress.io/api/commands/wait>

# cy.wait()

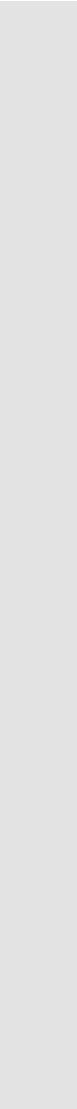


```
JS search.spec.js M X
cypress > integration > JS search.spec.js > ...
41 |   it('waiting for a fetch()', () => {
42 |     cy.intercept('/books/v1/volumes?*').as('book-search');
43 |
44 |     cy.get('select').select('inauthor');
45 |     cy.get('input').type('Douglas Adams{enter}');
46 |
47 |     cy.wait('@book-search');
48 |
49 |     cy.contains('.card-title', "The Hitchhiker's Guide to the Galaxy").should(
50 |       'be.visible'
51 |     );
52 |   });

```



See you in the next video



# Faking AJAX responses

# Faking AJAX responses

- Use a `staticResponse` to return a predetermined response
  - Preventing the original AJAX request
- Set the `statusCode`, `body` and/or `headers` options as needed
- The `fixture` option loads a file from the `cypress/fixtures` folder
  - The fixture location is configurable
- <https://docs.cypress.io/api/commands/intercept#staticResponse-It-code-gtStaticResponset-Code-gt>

# Faking AJAX responses

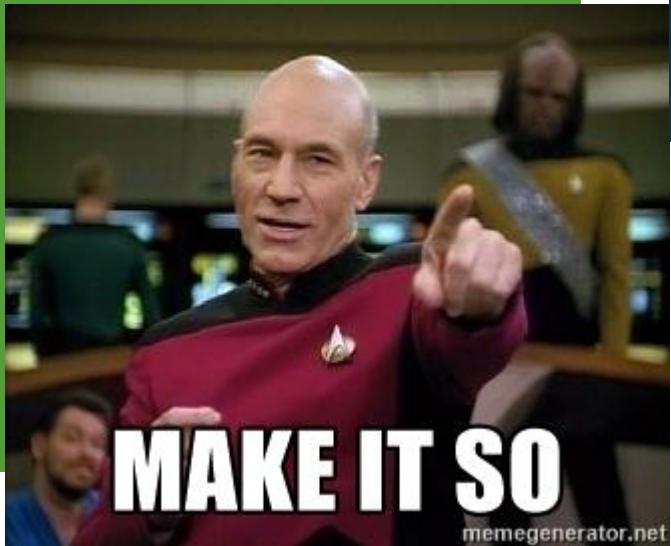
## Pro's

- Faster
- No dependencies on the backend services
- No need to reset the DB between tests
- No issues with external API's

## Con's

- Doesn't test the application as it will be used
- The API response might be updated

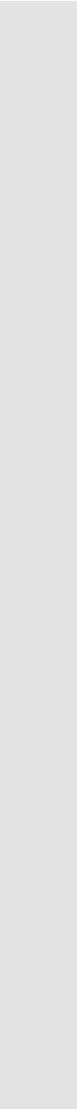
# search.spec.js



```
JS search.spec.js M X
cypress > integration > JS search.spec.js > ...
54 | it('respond with fake data', () => {
55 |   cy.intercept('/books/v1/volumes?*', { fixture: 'douglas-adams.json' }).as(
56 |     'book-search'
57 |   );
58 |
59 |   cy.get('select').select('inauthor');
60 |   cy.get('input').type('Douglas Adams{enter}');
61 |
62 |   cy.contains('.card-title', 'Book two').should('be.visible');
63 | });
```



See you in the next video



# Intercepting the response

# Intercepting the response

- Using `cy.wait()` with an AJAX request yields the response object
  - With a JSON body object if appropriate
- Use the response data to test the UI elements
- <https://docs.cypress.io/api/commands/wait#Yields>

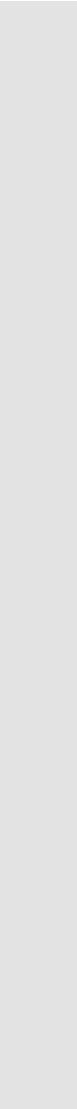
# search.spec.js



```
JS search.spec.js M X
cypress > integration > JS search.spec.js > ...
54 it('intercept and use the response', () => {
55   cy.intercept('/books/v1/volumes?*').as('book-search');
56
57   cy.get('select').select('inauthor');
58   cy.get('input').type('Douglas Adams{enter}');
59
60   cy.wait('@book-search').then((xhr) => {
61     const books = xhr.response.body.items;
62
63     books.forEach((book, index) => {
64       cy.get('.card')
65         .eq(index)
66         .within(() => {
67           cy.get('.card-title').should('have.text', book.volumeInfo.title);
68           cy.get('.card-author').should(
69             'have.text',
70             book.volumeInfo.authors.join(', '))
71         });
72     });
73   });
74 });
75 });
76 });
```



See you in the next video



# Making network requests

# cy.request()

- Using `cy.request()` you can make network requests from a test
  - Useful for testing your API's
  - Submitting URL encoded forms
  - Calling an API to seed test data
  - Etc...
- <https://docs.cypress.io/api/commands/request>

# Login example

```
JS request.spec.js M X
cypress > integration > JS request.spec.js > ...
6   it.skip('to login and retreive the token', () => {
7     cy.request('POST', '/login', {
8       username: 'maurice.de.beijer',
9       password: 'p@ssw0rd',
10    })
11    .its('body.token')
12    .then((token) => {
13      console.log(`Token: ${token}`);
14    });
15  });

```

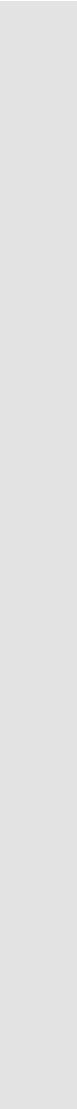
# request.spec.js



```
JS request.spec.js M X
cypress > integration > JS request.spec.js > ...
17 | it('to search for Douglas Adams', () => {
18 |   cy.request(
19 |     'https://www.googleapis.com/books/v1/volumes?q=inauthor:Douglas+Adams&maxR
20 |   ).as('response');
21 |
22 |   cy.get('@response').its('body.kind').should('equal', 'books#volumes');
23 |   cy.get('@response').its('body.totalItems').should('be.greaterThan', 350);
24 |   cy.get('@response').its('body.items').should('have.length', 25);
25 |   cy.get('@response').its('body.items.0.volumeInfo.title').should('exist');
26 |   cy.get('@response').its('body.items.0.volumeInfo.authors').should('exist');
27 |});
```



See you in the next video



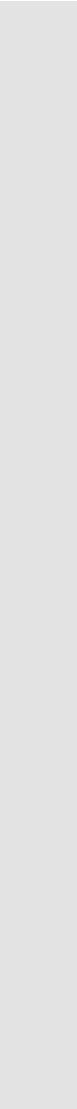
# Continuous Integration

# 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



# CI using GitHub Actions

# GitHub Actions

- Run Cypress as part of a **GitHub CI/CD pipeline**
  - Only deploy the application if the tests pass
- Run the development server on the **GitHub infrastructure**
  - Run the Cypress tests against this GitHub hosted server
- The **cypress-io/github-action** often makes this easy
  - Supports most of the common use cases
- <https://docs.cypress.io/guides/continuous-integration/github-actions>

# cypress-io.yml

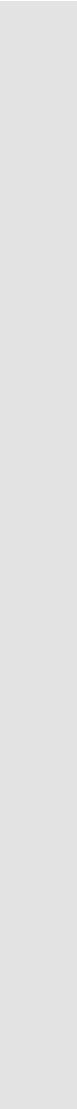


```
! cypress-io.yml ✘
.github > workflows > ! cypress-io.yml
1 name: Cypress CI Tests
2
3 on:
4   # Manual button click from the GitHub UI
5   workflow_dispatch:
6     push:
7       branches: [main]
8     pull_request:
9       branches: [main]
10
11 jobs:
12   start:
13     runs-on: ubuntu-latest
14     steps:
15       - name: Checkout Code
16         uses: actions/checkout@v2
17
18       - name: Run Cypress E2E Tests
19         uses: cypress-io/github-action@v2
20         with:
21           start: npm start
22           wait-on: 'http://localhost:3000'
23
24       - name: Upload Cypress Screenshots
25         uses: actions/upload-artifact@v2
26         if: failure()
27         with:
28           name: Cypress screenshots
29           path: cypress/screenshots
```





See you in the next video



# Cypress Continuous Integration Steps

# Cypress CI Steps

- The **basic steps** for a Cypress continuous integration process are:
  1. Get the latest version of the application
  2. Run `npm install`
  3. Start the web server
  4. Execute a `cypress run`
  5. Report the result

# Booting the web server

- **Booting the web server** can be tricky
  - It should not block the process
  - The Cypress tests should only start when the app is available
- The [start-server-and-test](#) NPM package is very useful

# cypress-io.yml

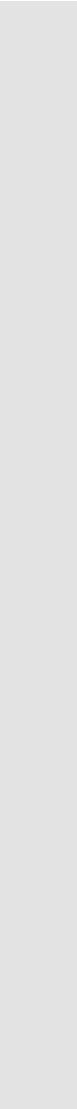


```
! cypress-io.yml M X
.github > workflows > ! cypress-io.yml
14   steps:
15     - name: Checkout Code
16       uses: actions/checkout@v2
17
18     - name: NPM Install
19       run: npm ci
20
21     - name: Install Start Server and Test
22       run: npm install start-server-and-test --no-save
23
24     - name: Run Application and Cypress E2E Tests
25       run: npx start-server-and-test start http://localhost:3000 cypress:run
26
27     - name: Upload Cypress Screenshots
28       uses: actions/upload-artifact@v2
29       if: failure()
30       with:
31         name: Cypress screenshots
32         path: cypress/screenshots
33
34     - name: Upload Cypress Videos
35       uses: actions/upload-artifact@v2
36       if: failure()
37       with:
38         name: Cypress videos
39         path: cypress/videos
```





See you in the next video



# Using a Docker container

# Docker containers

- Using Docker containers can help create a **consistent environment**
  - The same runtime locally and on the CI server
- Required when doing **visual regression testing**
  - Often a good approach
- <https://hub.docker.com/r/cypress/include>

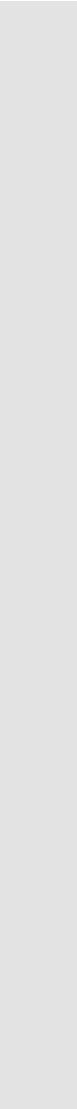
# cypress-io.yml



```
! cypress-io.yml x
.github > workflows > ! cypress-io.yml
11   jobs:
12     start:
13       runs-on: ubuntu-latest
14       container:
15         image: cypress/included:8.5.0
16         options: --ipc=host
17
18     steps:
19       - name: Checkout Code
20         uses: actions/checkout@v2
```



See you in the next video



# Flaky tests

# Flaky tests

- Flaky tests are tests that **sometimes fail** and sometimes pass without any change in code
- Cypress can **automatically retry failing tests** to detect flakiness
  - Helps prevent a complete CI build from failing
  - The underlying flaky test should still be fixed

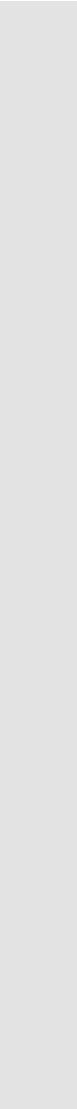
# cypress.json



```
① cypress.json M X
① cypress.json > ...
You, seconds ago | 1 author (You)
1 {
2   "$schema": "https://on.cypress.io/cypress.schema.json",
3   "baseUrl": "http://localhost:3000",
4   "retries": {
5     "runMode": 3,
6     "openMode": 1
7   },
8   "ignoreTestFiles": ["**/1-getting-started/*", "**/2-advanced-examples/*"]
9 }
```



See you in the next video

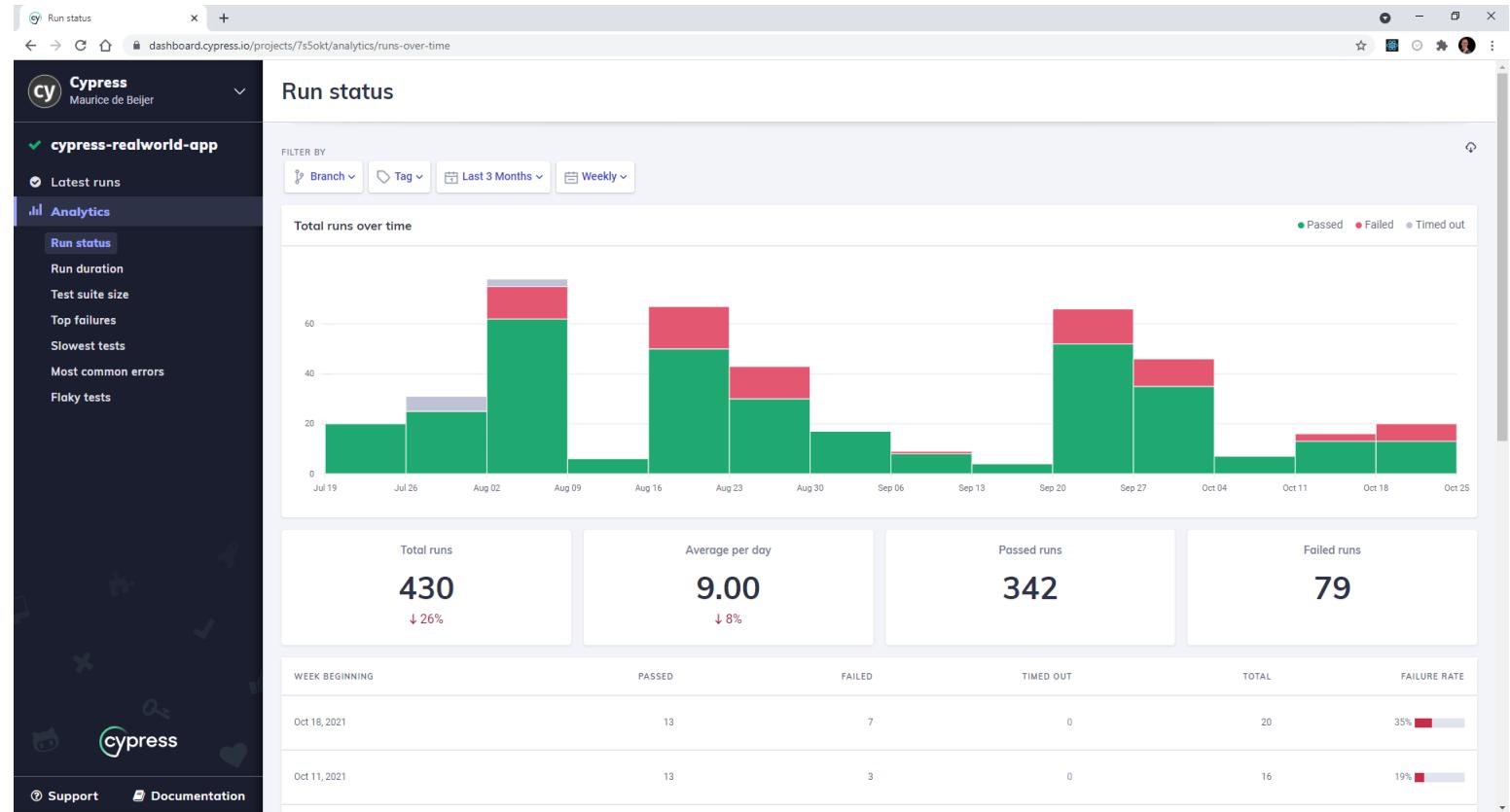


# The Cypress Dashboard

# Cypress Dashboard

- Makes it easy to **see what is happening** on the CI server
- **Saves a video** of the test running
- **Not required**, totally optional
  - This is a paid for option
  - With a limited free tier to get started

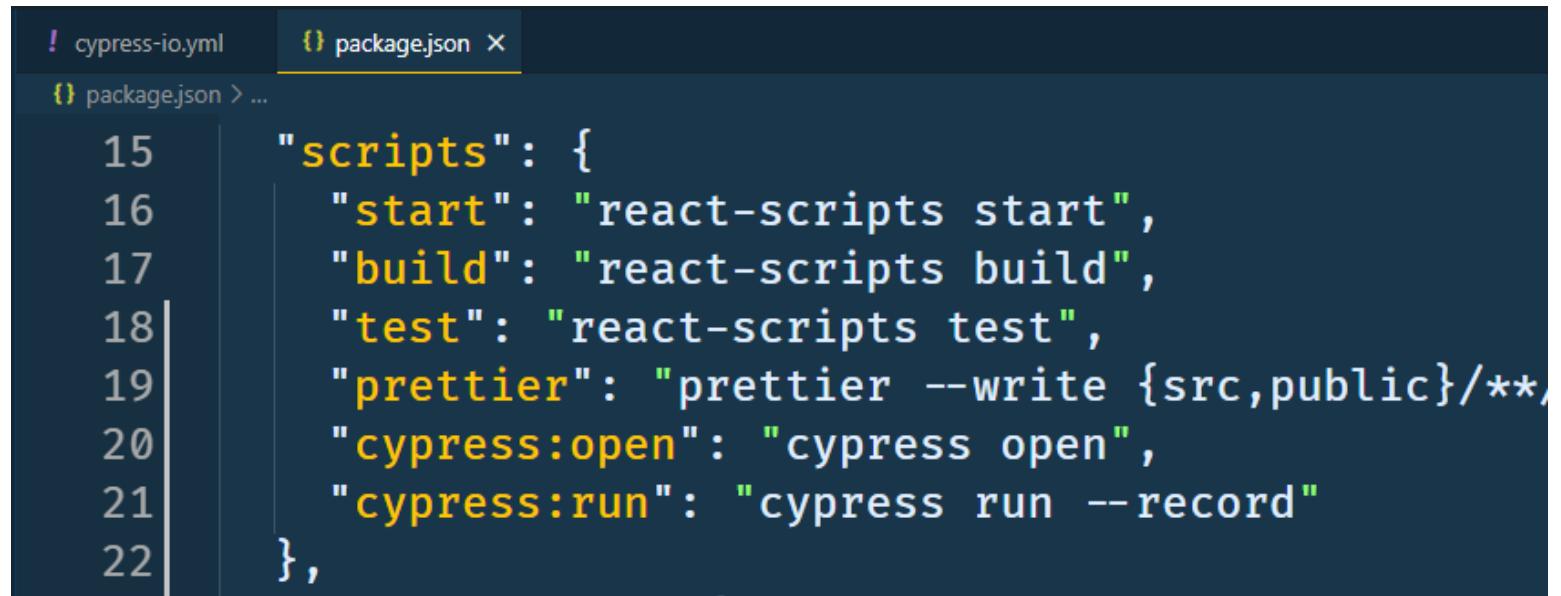
# Cypress Dashboard



# cypress-io.yml

```
! cypress-io.yml ✘ 0 package.json
.github > workflows > ! cypress-io.yml
11 jobs:
12   start:
13     runs-on: ubuntu-latest
14     container:
15       image: cypress/included:8.6.0
16       options: --ipc=host
17     env:
18       CYPRESS_PROJECT_ID: ${{ secrets.CYPRESS_PROJECT_ID }}
19       CYPRESS_RECORD_KEY: ${{ secrets.CYPRESS_RECORD_KEY }}
20
21   steps:
22     - name: Checkout Code
23       uses: actions/checkout@v2
24
25     - name: NPM Install
26       run: npm ci
27
28     - name: Install Start Server and Test
29       run: npm install start-server-and-test --no-save
30
31     - name: Run Application and Cypress E2E Tests
32       run: npx start-server-and-test start http://localhost:3000 cypress:run
```

# package.json



A screenshot of a code editor showing the `package.json` file. The editor has tabs for `cypress-io.yml` and `package.json`, with `package.json` being the active tab. The code editor shows the following JSON configuration:

```
15 "scripts": {  
16   "start": "react-scripts start",  
17   "build": "react-scripts build",  
18   "test": "react-scripts test",  
19   "prettier": "prettier --write {src,public}/**",  
20   "cypress:open": "cypress open",  
21   "cypress:run": "cypress run --record"  
22 },
```

# The dashboard

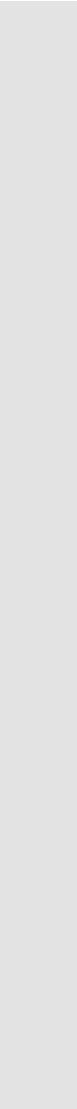


A screenshot of the Cypress dashboard interface. The top navigation bar includes "Book Search", a search bar, and a user profile icon. The main header displays the project name "ABL - The Problem" and the author "Maurice de Beijer". A prominent message says "100% Onboarding complete" with a note to invite teammates. Below this is a "Test Results" section showing 9 tests, all of which have passed ("3 specs passed"). The results list includes entries like "flaky.spec.js" (1 test, 123ms, Linux Debian - 10.11, Electron 93), "request.spec.js" (1 test, 981ms, Linux Debian - 10.11, Electron 93), and various "Search" and "Flaky tests" entries. The sidebar on the left contains links for "Book Search", "Latest runs", "Analytics", "Run status", "Run duration", "Test suite size", "Top failures", "Slowest tests", "Most common errors", "Flaky tests", and "Project settings". At the bottom are links for "Support" and "Documentation".





See you in the next video



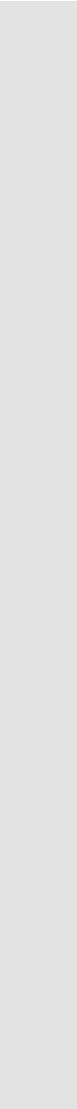
# Tip & Tricks

## Tip & Tricks





See you in the next video



# IntelliSense in Visual Studio Code

# IntelliSense

- Add a reference to a package types using **triple slash directives**
  - `/// <reference types="cypress" />`



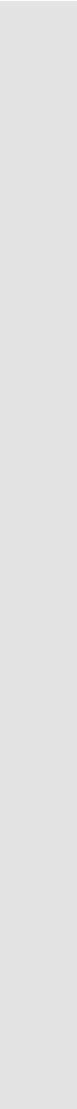
JS search.spec.js 4 ×

cypress > integration > JS search.spec.js > ...

```
1  /// <reference types="cypress" />
2
3  describe('Search', () => {
4    beforeEach(() => {
5      cy.visit('/');
6    });
}
```



See you in the next video



# Configuring ESLint

# Configuring ESLint

- Install the [eslint-plugin-cypress](#) NPM package
- Extend the **plugin:cypress/recommended** in your ESLint config

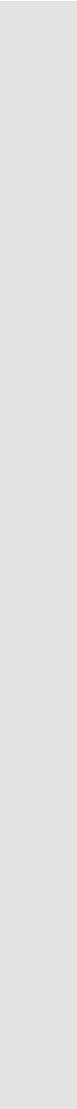
# package.json



```
{} package.json M x
{} package.json > ...
23   "eslintConfig": {
24     "extends": [
25       "react-app",
26       "plugin:cypress/recommended"
27     ]
28   },
29   "browserslist": {
30     "production": [
31       ">0.2%",
32       "not dead",
33       "not op_mini all"
34     ],
35     "development": [
36       "last 1 chrome version",
37       "last 1 firefox version",
38       "last 1 safari version"
39     ]
40   },
41   "devDependencies": {
42     "@testing-library/jest-dom": "^5.14.1",
43     "@testing-library/react": "^12.1.0",
44     "@testing-library/user-event": "^13.2.1",
45     "@types/jest": "^27.0.1",
46     "@types/node": "^16.9.1",
47     "@types/react": "^17.0.20",
48     "@types/react-dom": "^17.0.9",
49     "@types/react-router-dom": "^5.1.8",
50     "cypress": "^8.4.0",
51     "eslint-plugin-cypress": "^2.12.1",
```



See you in the next video



Maurice de Beijer

@mauricedb

maurice.de.beijer  
@gmail.com

