

Automatiser facilement vos tests E2E web avec Gravity et Cypress

<https://tinyurl.com/gravitycypress>

About



Bruno Legeard

Chief Scientist /
Co-founder of Smartesting

bruno.legeard@smartesting.com



Julien Botella

Gravity Product Manager

julien.botella@smartesting.com



Abbas AHMAD

Test Automation Engineer
ISTQB Trainer

abbas.ahmad@uaestqb.ae

Agenda



- Introduction
- Cypress for test automation
- Usage-centric Testing
- Gravity
- Discussion/Questions

06.12.22 • Sophia Antipolis

Telecom
Valley

Soirée du
<Test Logiciel>

Introduction

gravity

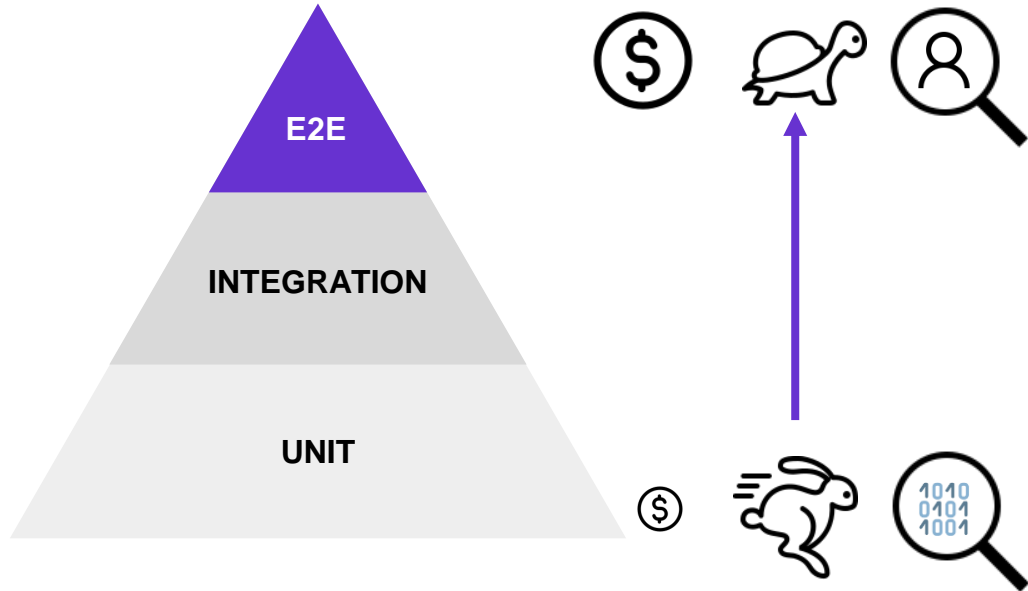
Introduction

Delivering value consistently, reliably

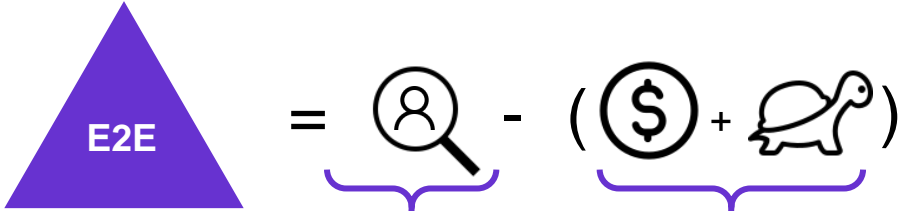
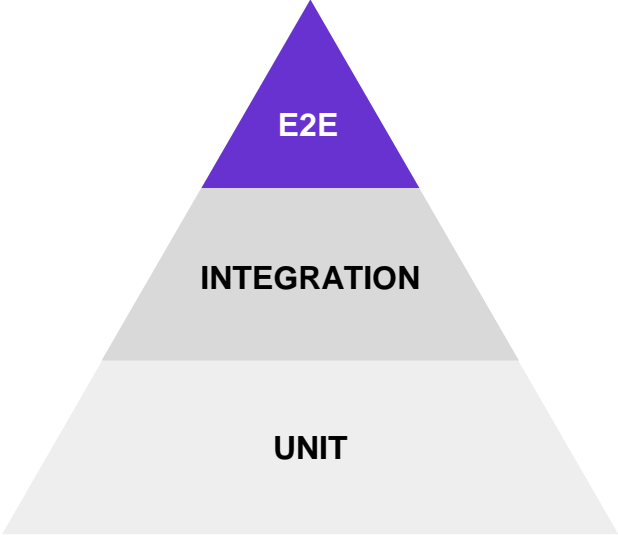
- A **software vendor** (company, team...) must **regularly deploy** new versions to :
 - Stay ahead of the competition (innovations)
 - Increase customer satisfaction (functional improvements, fixes...)

- **Agile** and **DevOps** practices aim at accelerating the pace of releases. They require a significant effort devoted to testing activities, especially **automated functional regression tests**
 - How to find the best compromise between **reactivity** and **quality**?
 - How to **reduce the effort** of creating and maintaining automated functional regression tests?
 - How to ensure **sufficient test coverage** of **key paths**?

Test pyramid



End-to-end tests



How to **target** the **right** tests ?

How to **reduce** their **production** and **maintenance** ?

Reduce production and maintenance

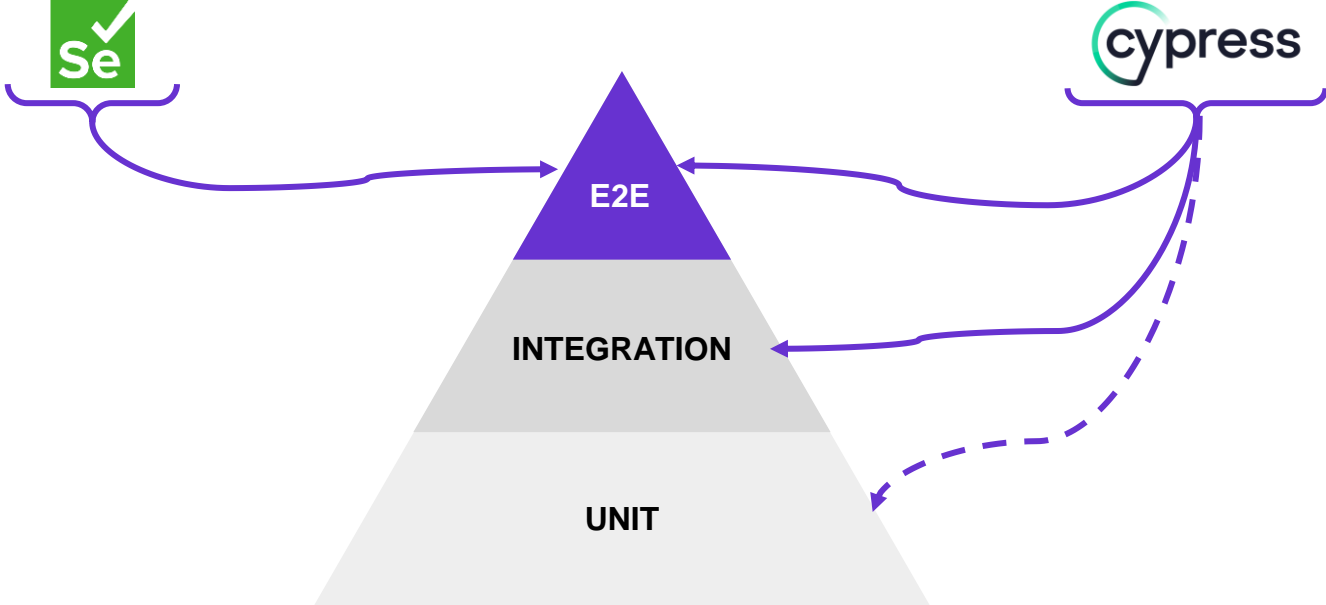


$$= \text{🔍} - (\text{💰} + \text{🐢})$$






















Cypress for test automation

Cypress vs Selenium: Same but different!



Cypress vs Selenium: Same but different!



	Selenium 	Cypress 
Speed		
Wait for element		 
Video		
Execute JS		 
Screenshots		
Parallel execution	 	
Several browsers		

Cypress for test automation



PREREQUISITE

- Node.js [v16.17.0](#)
 - Uninstall other versions
 - Install Node.js v16.17.0
- Yarn
 - Open a terminal
 - Type command: *npm install --global yarn*
- Git (Optional)



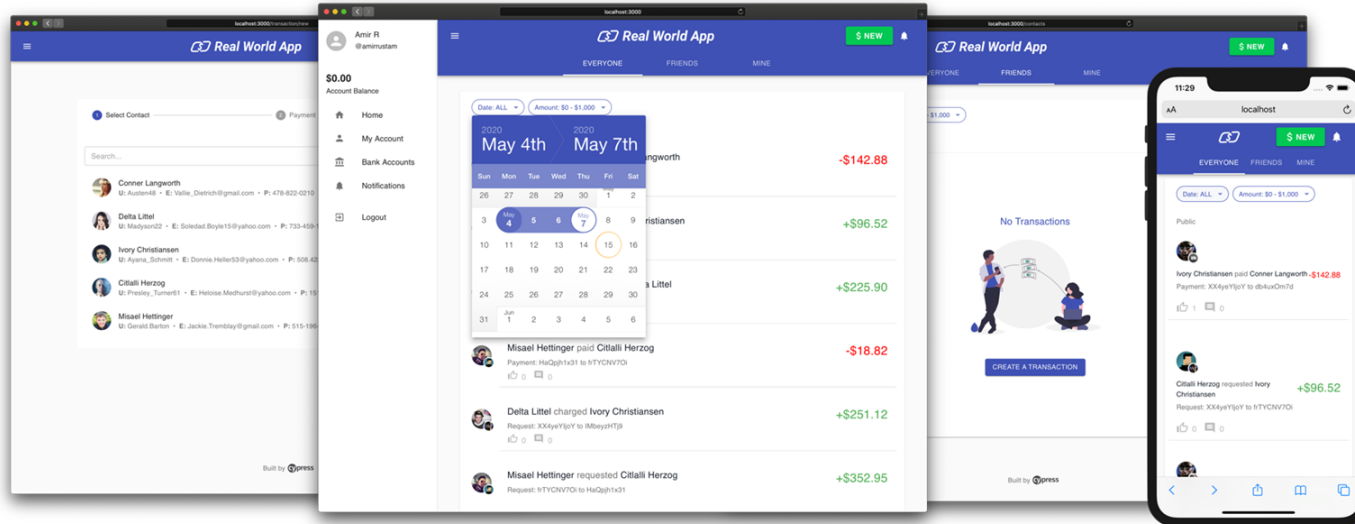
Cypress for test automation



Real World App: the **System Under Test** (SUT)

Application is purely for demonstration and educational purposes.

Real World App



Cypress for test automation



Installing the SUT

1. Clone/Download the [Real World App](#) git repository
2. In a terminal access the Real World App folder “cypress-realworld-app”
3. Command: ***yarn install***

Running the SUT

1. ***yarn dev***

```
start:react] Compiled successfully!  
start:react]  
start:react] You can now view cypress-realworld-app in the browser.  
start:react]  
start:react]   http://localhost:3000  
start:react]  
start:react] Note that the development build is not optimized.  
start:react] To create a production build, use yarn build.
```

1. <http://localhost:3000>



Cypress for test automation



Installing Cypress

Using *npm*:

1. Create an empty directory and access it via a terminal
2. Install Cypress in that directory: ***npm install cypress --save-dev***

```
C:\Users\aaahmad\Cypress\atelierCypress>npm install cypress --save-dev
added 165 packages, and audited 166 packages in 2m

28 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Opening Cypress

Using *npx*:

1. In a terminal, access your folder where you previously installed Cypress
2. Run Cypress: ***npx cypress open***

Note: Cypress is not installed globally on your PC.
It is a single install per project/directory.


Cypress for test automation



Cypress Launchpad

Select End to End Testing: **“E2E Testing”**


Welcome to Cypress!
[Review the differences between each testing type →](#)



E2E Testing

Build and test the entire experience of your application from end-to-end to ensure each flow matches your expectations.

Not Configured



Component Testing

Build and test your components from your design system in isolation in order to ensure each state matches your expectations.

Not Configured





Cypress for test automation



On Configuration files click on the “Continue” button

Configuration files

We added the following files to your project:

-  [cypress.config.js](#)
The Cypress config file for E2E testing.
-  [cypress\support\e2e.js](#)
The support file that is bundled and loaded before each E2E spec.
-  [cypress\support\commands.js](#)
A support file that is useful for creating custom Cypress commands and overwriting existing ones.
-  [cypress\fixtures\example.json](#)
Added an example fixtures file/folder

[Continue](#)

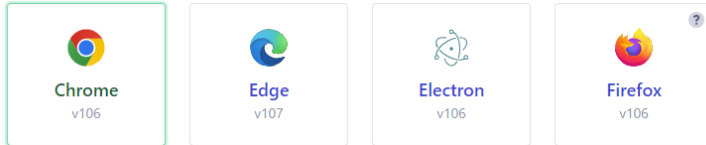
Cypress for test automation



When prompted, select a browser
Then select “Create new empty spec” and run the spec

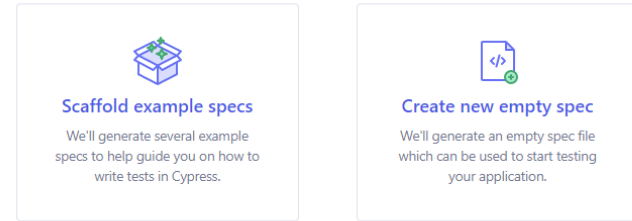
Choose a browser

Choose your preferred browser for E2E testing.



Create your first spec

Since this project looks new, we recommend that you use the specs and tests that we've written for you to get started.



Cypress for test automation



Let's start writing Cypress tests!

With your favorite IDE, open the folder where you installed Cypress

A screenshot of a code editor interface. The left sidebar shows a file explorer with a tree view containing folders like 'downloads', 'e2e', 'fixtures', 'support', 'node_modules' and files like 'cypress.config.js', 'package-lock.json', and 'package.json'. The 'e2e' folder is expanded, and 'spec.cy.js' is selected. The main editor area shows the content of 'spec.cy.js' with the following code:

```
1 describe('empty spec', () => {  
2   Open Cypress | Set ".only"  
3   it('passes', () => {  
4     cy.visit('https://example.cypress.io')  
5   })  
6 })
```

Cypress for test automation



We will now edit the existing spec file in order to test our Real World App.

Test objective: sign in to the application and make a new payment

Actions:

1. Edit the test suite description
2. Edit default test case name
3. Edit "*cy.visit(...)*" to point to your application

```
JS spec.cy.js ●
cypress > e2e > JS spec.cy.js > ...
1 describe('Cypress real World App Testing', () => {
  Open Cypress | Set *.only
2   it('Visit Real World App, Sign In and make a new payment', () => {
3     //Open Browser on Real World Application
4     cy.visit('http://localhost:3000/')
5   })
6 }
```



Cypress for test automation



The screenshot shows the Cypress test runner interface. On the left, the 'Specs' panel displays a test suite named 'Cypress real World App Testing' with a single spec file 'spec.cy.js'. The test suite is expanded to show a single test: 'Visit Real World App, Sign In and make a new payment'. The test body contains a single command: 'visit http://localhost:3000/'.

The main area shows a browser window at 'http://localhost:3000/signin'. The browser displays a sign-in form for 'Real World App'. The form includes:



- A 'Username' input field with a cursor.
- A 'Password' input field.
- A 'Remember me' checkbox.
- A blue 'SIGN IN' button.
- A link: [Don't have an account? Sign Up](#)
- Footer text: 'Built by Cypress'.



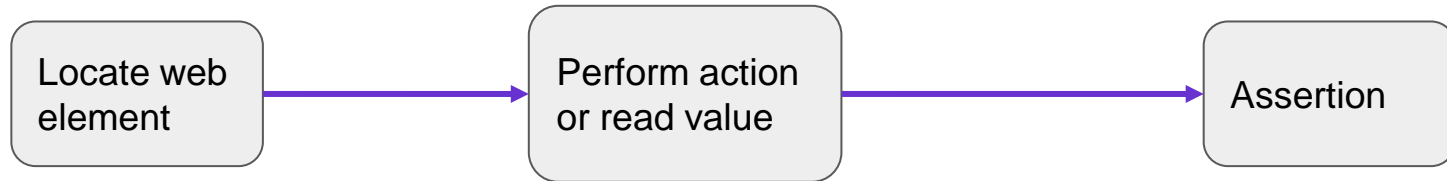
Cypress for test automation



For web applications testing we need testability characteristics:

- Observability 
- Controllability 

Test workflow:



cy.get: select based on HTML tag attrs ...

cy.contains: select based value within opening and closing tags

...

.click: click on an element

.check: check a checkbox or radio

.select: Select option from a dropdown menu

...

.should: command used for asserting tests

...

Cypress for test automation



Real World App preconfigured user:

Username: **Allie2**
Password: **s3cret**

Add assertions when ever needed:

```
//Verify that the displayed page is the Sign in page  
cy.get('h1:first').should('have.text', 'Sign in')
```

```
//Enter user sign in information  
cy.get('#username').type('Allie2')  
cy.get('#password').type('s3cret')
```

```
//click on button 'Sign in'  
cy.get('button').contains('Sign In').click()
```

```
//Verify user connected is signed in  
cy.get('[data-test="sidenav-username"]').then(($el)=> {  
    expect($el.get(0).textContent, 'Username').to.equal('@Allie2')  
})
```

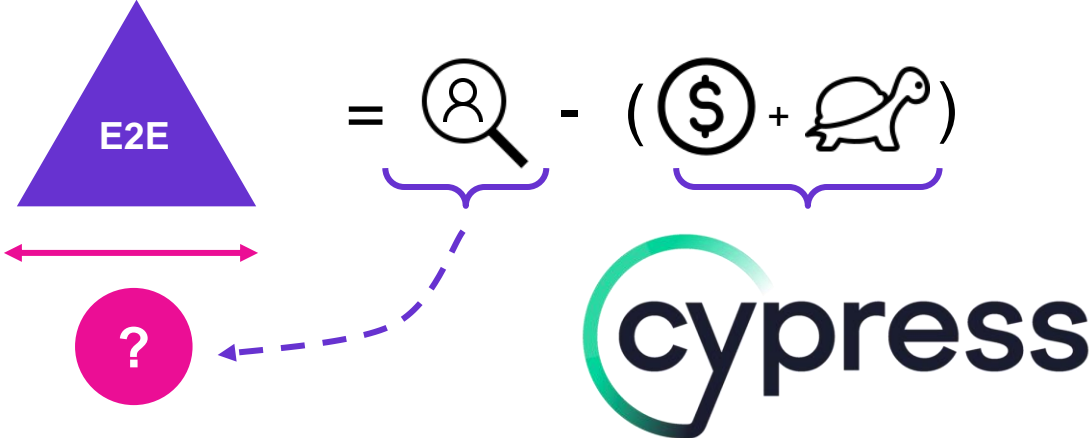
To go further: [Cypress documentation](#)

Download complete solution:
<https://pastebin.com/uKQfVke6>



Target the **right** tests with Usage-centric testing

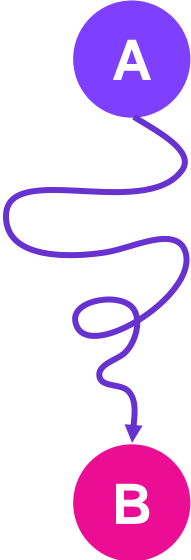
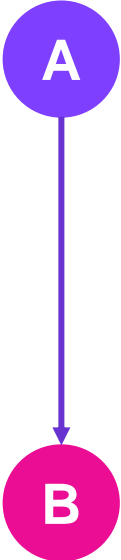
Target the right tests



Usage-centric Testing: Theory VS Reality

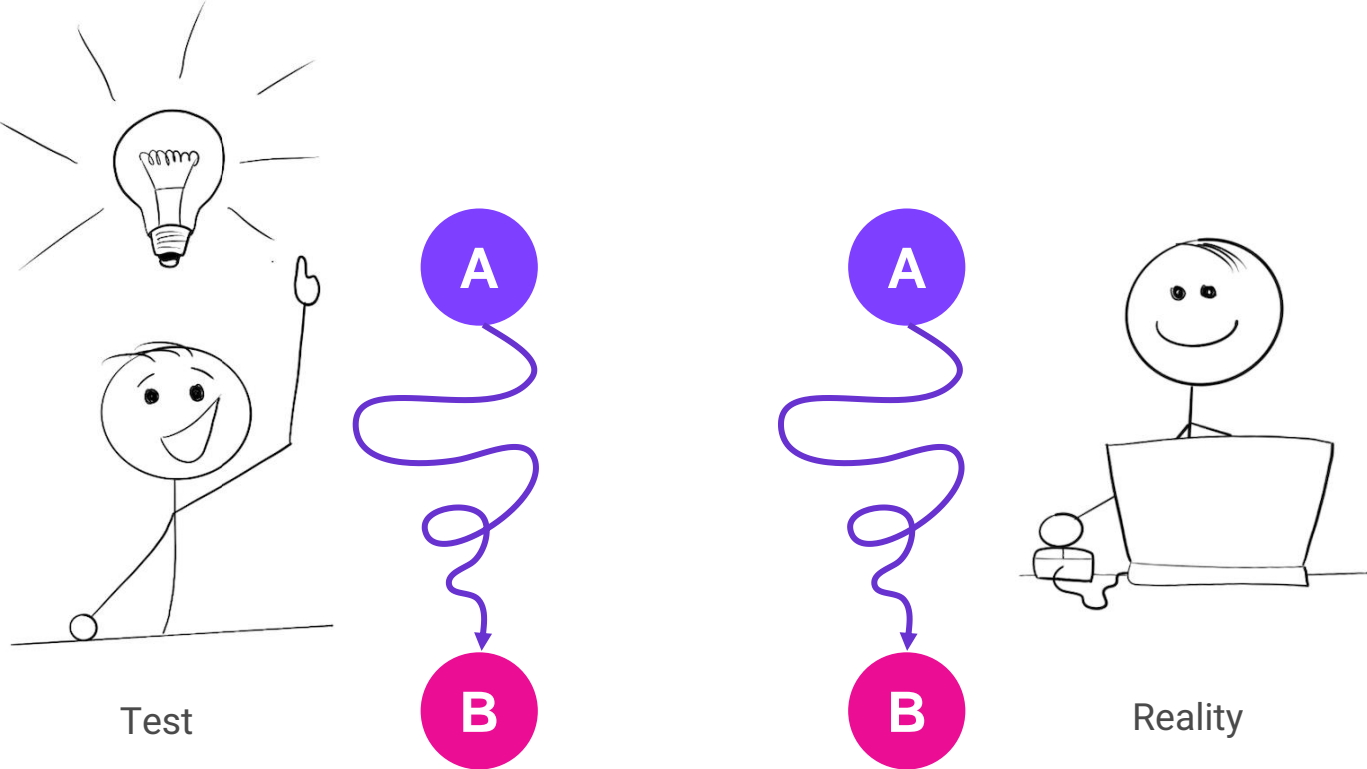


Theory



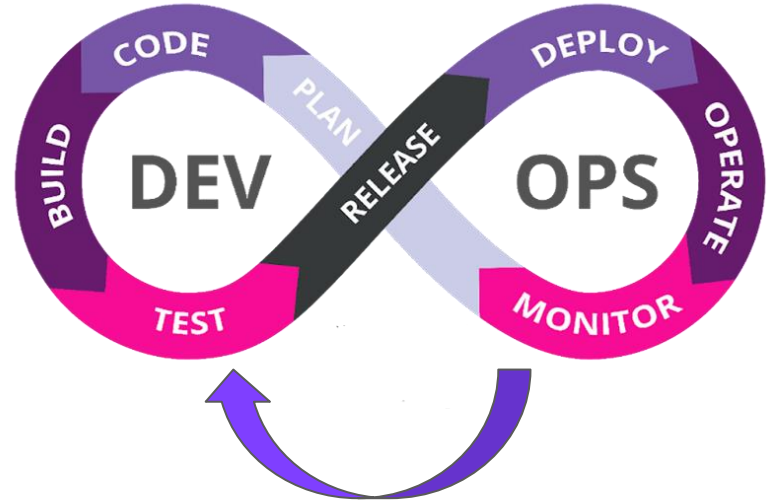
Reality

Usage-centric Testing: Test = Reality



Usage-centric Testing

- Agile & DevOps approach
- Based on usage in production
- Strengthen automated testing
→ *shift right*



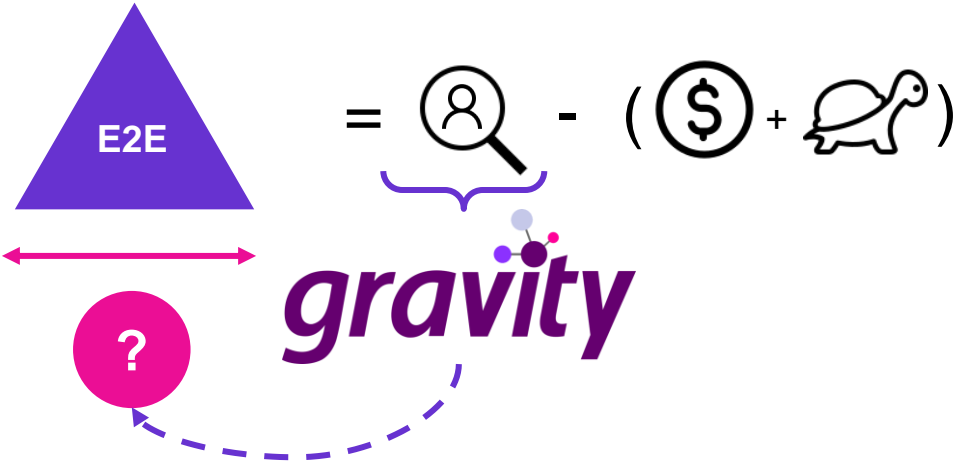
The logo for 'gravity' features the word in a white, lowercase, italicized sans-serif font. Above the 'i' in 'gravity', there is a white icon consisting of four circles of varying sizes connected by thin lines, resembling a molecular structure or a network diagram.

gravity

Helps **Agile & DevOps** teams to deliver high-quality software **faster**, by enabling them **select** and **design** end-to-end tests accordingly to **HOW** their users truly interact with their web application

We call it a **Usage-centric Testing** platform

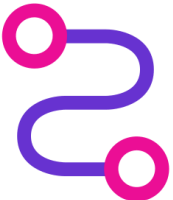
Gravity



Gravity



Collect usage data



Browse user sessions



Evaluate test coverage



Generate test scripts

Gravity

Production



Gravity
Data collector 

Qualification



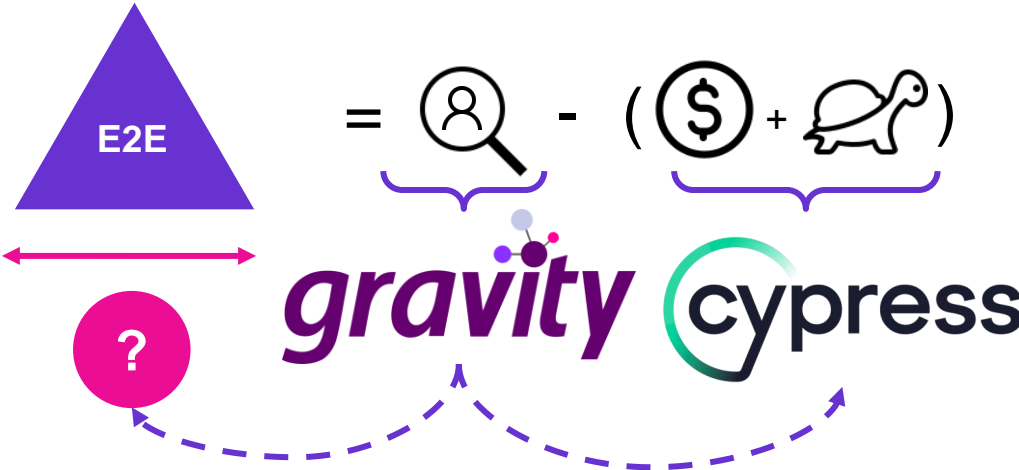
Gravity
Data collector 





Gravity, step-by-step

Gravity + Cypress



Start with Gravity ?

Try **Gravity** for **free** at : <https://gravity-testing.com>

The image is a composite of three screenshots. The top-left screenshot shows the Gravity website with a navigation bar (Features, Pricing, Resources, Contact), a 'Try for free' button, and a 'Login' link. The main content area features the headline 'Test what your users actually do' and a sub-headline 'Design automated tests suites tailored to your product actual usage'. Below this are two buttons: 'Create free account' and 'Schedule a demo'. A video player is embedded in the center, showing the Gravity logo and the same headline. The top-right screenshot shows a Cypress test runner interface with a sidebar menu (Demo Cypress, Sessions, Usages, Configuration) and a list of test runs for 'Login (valid with click)'. The main area displays a 'Login (with enter)' form with fields for 'label Username' and 'label Password', and a 'Supprimer l'usage' button. The bottom-right screenshot shows a browser view of the login form with a 'TROUVÉ DANS' (Found in) section listing sessions for 'UTILISATEUR' (17.2%) and 'TEST' (1).



Discussion / Questions

Thanks for watching!

Try Gravity for free at <https://gravity-testing.com>

Contact us :

bruno.legeard@smartesting.com

julien.botella@smartesting.com

abbas.ahmad@uaestqb.ae



gravity