🎪 Pruebas de comportamiento

Cypress y Behavior Driven Development.

> *"Los ordenadores son muy buenos siguiendo instrucciones, pero muy malos leyendo mentes"*
>
> -- ✍️ Donald Knuth

- Agrupar y gestionar mejor las pruebas de comportamiento.
- **Convenios de nombrado**
- Además de testear, documentar la funcionalidad de la aplicación.

Funcionalidad

- **historia de usuario**: funcionalidad que resuelve un problema.
- Formalmente en *Agile* lo llaman **behavior-driven development (BDD)**.

> **Estructura y documenta muy bien tus pruebas funcionales.**

Role Feature Reason

- el rol (**quién**),
- la solución (**qué**)
- la razón (**por qué**)

```
FEATURE: have a web site with courses and a subscribing form
As a: visitor
I want to: view, navigate and subscribe
In order to: get information and be notified
```

Comportamiento

Se plantea un escenario, se simula la interacción del usuario y se comprueba el resultado esperado.

Context

Esta función es similar a `describe` pero se usa como un agrupador de nivel inferior.

```
describe('Funcionalidad que se pretende probar', () => {
  context('Escenario o situación prevista', () => {
    it('Lo que debería ocurrir', () => {});
  });
});
```

Arrange, Act, Assert

*AAA o **la triple A***: *Arrange, Act, Assert* que se traducen como: **preparar, actuar y comprobar**.

```
describe('Visiting the url https://www.bitademy.com', () => {
  // Arrange
  const sutUrl = 'https://www.bitademy.com';
  context('I visit it', () => {
    // Act
    before(() => cy.visit(sutUrl));
    // Assert
    it('should have an h2 on the hero header with text _Aprender a programar mejor_', () => {
      cy.get('#hero > div > div > div.cell.block-content > h2').should(
        'contain',
        'Aprender a programar mejor'
      );
    });
  });
});
```

Aceptación

La *triple AAA* nos ayuda dentro de nuestro código de prueba. Es *muy de programador*.

Las pruebas funcionales de integración e2e se asemejan a **pruebas de aceptación**.

Para acercarlas al usuario se deberían definir con sus palabras y con sus criterios.

Given When Then

- Es un convenio para **documentar la traza de la prueba**
- Se aplica a los textos que reciben y escriben las funciones
- **En lenguaje de usuario**.

GWT es una fórmula fácil de recordar y que nos narra en lenguaje humano lo que sucede por debajo. `Given, when, then` traducido como **dado, cuando, entonces** explica que **dado** un contexto, **cuando** se ejecuta una acción, **entonces** debería haber una consecuencia esperada.

```
describe('GIVEN: the url https://www.bitademy.com', () => {
  // Arrange
  const sutUrl = 'https://www.bitademy.com';
  context('WHEN: I visit it', () => {
    // Act
    before(() => cy.visit(sutUrl));
    // Assert
    it('THEN: should have an h2 on the hero header with text _Aprender a programar mejor_', () => {
      cy.get('#hero > div > div > div.cell.block-content > h2').should(
        'contain',
        'Aprender a programar mejor'
      );
    });
  });
});
```

Resumen

Orden y guía para no reinventar la rueda.

> tabla para organizar y documentar tus pruebas.

Organización	Documentación	Implementación
Arrange	GIVEN:	`describe()`
Act	WHEN:	`context() before()`
Assert	THEN:	`it()`
After		`after()`

Esto debería ayudarte a **demostrar que tu aplicación es aceptable**. Es decir, explicar qué hace, para quién lo hace y por qué lo hace.