# 🌲 Pruebas funcionales con Cypress

## Cypress para comprobación funcional interactiva.

> *"La calidad no es un acto, es un hábito"*
>
> -- ✍️ Aristóteles

Calidad es que algo funcione conforme a lo esperado

Sencillo, cómodo y agradable para ser un hábito

# Cypress

- framework para pruebas funcionales *e2e*.

## Instalar Cypress

```
yarn add cypress
npm i --save cypress
```

Comprobar en `cypress\integration`

## Test funcionales

Las **pruebas funcionales de aplicaciones web** simulan el comportamiento del usuario en ellos:

- hacer clic,
- escribir,
- desplazarse, etc.

> Aseguran que un escenario funciona desde el punto de vista de un usuario final.

## Probar funcionalidades web con Cypress

*Cypress* nos ofrece la conocida sintaxis `describe it`.

## Describe it

```
describe('Funcionalidad que se pretende probar', () => {
  // Código de preparación y actuación pruebas
  it('Lo que debería ocurrir', () => {
    // Comprobación mediante aserciones
  });
});
```

> **Presta atención al texto** del primer parámetro .

## Actuaciones y aserciones

```
/// <reference types="Cypress" />

// Actuaciones como un usuario
cy.visit('https://www.bitademy.com');
// Comprobaciones como un programador
cy.title().should('include', 'bitAdemy');
```

## Ejecutar las pruebas con Cypress

```
{
  "scripts": {
    "start": "cypress open",
    "test": "cypress run",
  }
}
```

## Hola mundo con Cypress

En el laboratorio

`cypress\integration\examples\0.0_hello-world.spec.js` y escribas esto en él.

```
/// <reference types="Cypress" />

describe('Visiting the url https://www.bitademy.com', () => {
  it('should have _bitAdemy_ on its title', () => {
    cy.visit('https://www.bitademy.com');
    cy.title().should('include', 'bitAdemy');
  });
});
```

Selecciona el `0.0_hello-world` y disfruta.

## Más acciones y comprobaciones

Ejemplo en `cypress\integration\examples\0.1_hello-world.spec.js`

Acciones comunes: clicks, navegar entre páginas, rellenar formularios... y comprobar contenidos.

Textos que deduce claramente **la intención del desarrollador**.

La documentación forma parte del programa. **No es un comentario, es un dato.**

### Before and after

*before beforeAll, after y afterAll* ejecutan la función que reciben como *callback* en los momentos adecuados. Sus nombres no dejan lugar a dudas.

**No reciben textos informativos.** Si quieres dejar rastro tienes que escribir en la consola o en un log. Si se complican es muy recomendable extraer funcionalidad a funciones con nombre.

Asegúrate de probarlo en el laboratorio