Introduction to Cypress.io Testing Library

Alex Sawsienowicz // alex.sawsienowicz@venafi.com // Jet Jaguars

If you want to follow along...

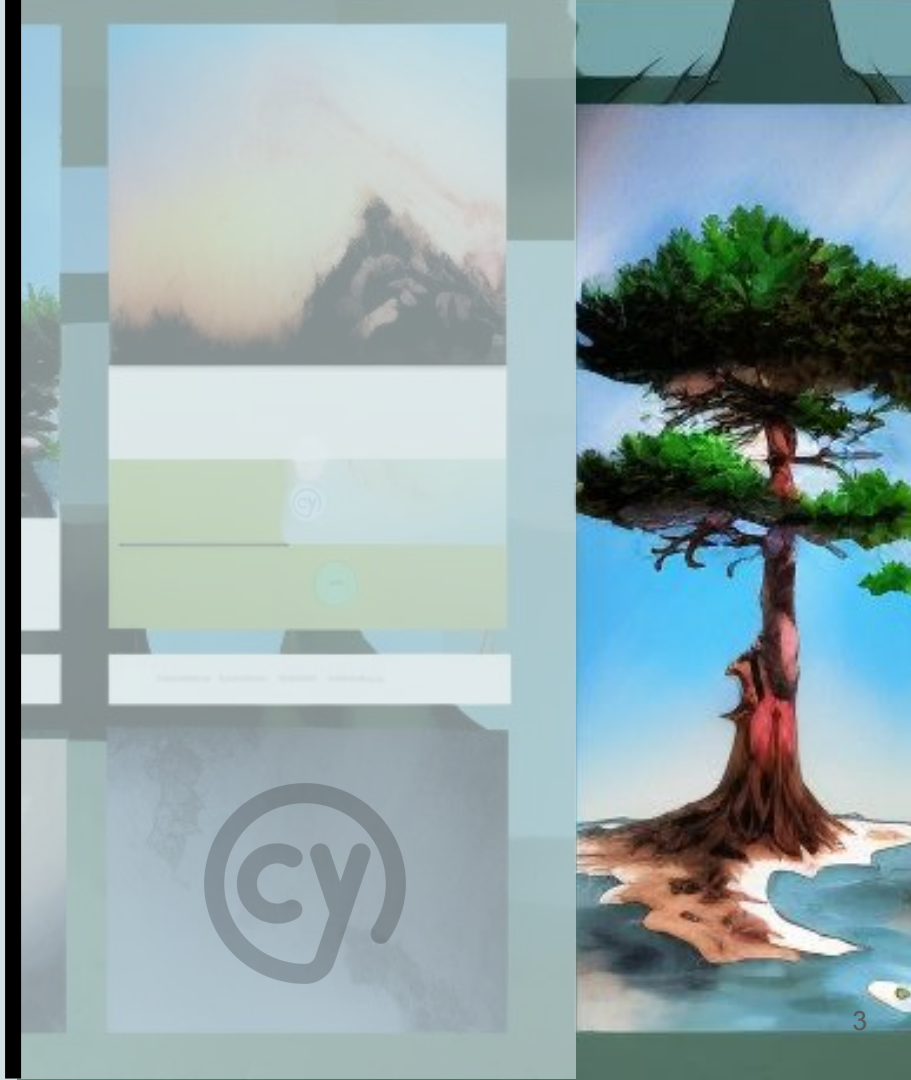
GitHub

https://github.com/alexsaw/cypress_intro

Medium

https://medium.com/@alex.sawsienowicz

Contents



Familiarization Demo

Installation and Setup

Cypress is a JavaScript project so you'll need node and npm installed, first.

Go to a new project directory and run the following:

```
$ npm install cypress
```

Now, run the following command to initialize your new Cypress project:

```
$ npx cypress open
```

Example Test

```
describe('Post Resource', () => {

 it('Creating a New Post', () => {
   cy.visit('https://www.example.com/posts/new') // 1.

   cy.get("input.post-title") // 2.
     .type("My First Post"); // 3.

   cy.get("input.post-body") // 4.
     .type("Hello, world!"); // 5.

   cy.contains("Submit") // 6.
     .click(); // 7.

   cy.get("h1") // 8.
     .should("contain", "My First Post");
   });

});
```

Running Tests

```
cypress
 |-- downloads
 |-- e2e
     |-- sampleTest.cy.js
 |-- fixtures
 |-- support
 |-- utils
 |-- videos
```

```
$ npm cypress open
```

or

```
$ npx cypress run -spec “cypress/e2e/sampleTest.cy.js”
```

Chaining and Accessing Subjects

```
describe('Post Resource', () => {
 it('Creating a New Post', () => {
   cy.visit('https://www.example.com/posts/new')

   cy.get("input.post_title")
     .find("p.post_body")
     .invoke("text")
     .then((post_text)=>{

       // locate a string with regex...
       if(post_text.includes("content:")) {
         let pattern = /content:.*/
         let content = pattern.exec(post_text)[0].replace("content: ", "")
         // ...then append it to a file
         cy.writeFile(
           'cypress/fixtures/content.json',
           content,
           { flag: 'a+' }
         )
      })

 })
});
```

Chaining and Accessing Subjects

```
describe('Post Resource', () => {
 it('Creating a New Post', () => {
   cy.visit('https://www.example.com/posts/new')

   cy.get("input.post_title")
     .find("p.post_body")
     .invoke("text")
     .then((post_text)=>{

       // locate a string with regex...
       if(post_text.includes("content:")) {
         let pattern = /content:.*/
         let content = pattern.exec(post_text)[0].replace("content: ", "")
         // ...then append it to a file
         cy.writeFile(
           'cypress/fixtures/content.json',
           content,
           { flag: 'a+' }
         )
      })

 })
});
```

Chaining and Accessing Subjects

```
describe('Post Resource', () => {
 it('Creating a New Post', () => {
   cy.visit('https://www.example.com/posts/new')

   cy.get("input.post_title")
     .find("p.post_body")
     .invoke("text")
     .then((post_text)=>{

       // locate a string with regex...
       if(post_text.includes("content:")) {
         let pattern = /content:.*/
         let content = pattern.exec(post_text)[0].replace("content: ", "")
         // ...then append it to a file
         cy.writeFile(
           'cypress/fixtures/content.json',
           content,
           { flag: 'a+' }
         )
      })

 })
});
```

Chaining and Accessing Subjects

```
describe('Post Resource', () => {
 it('Creating a New Post', () => {
   cy.visit('https://www.example.com/posts/new')

   cy.get("input.post_title")
     .find("p.post_body")
     .invoke("text")
     .then((post_text)=>{

       // locate a string with regex...
       if(post_text.includes("content:")) {
         let pattern = /content:.*/
         let content = pattern.exec(post_text)[0].replace("content: ", "")
         // ...then append it to a file
         cy.writeFile(
           'cypress/fixtures/content.json',
           content,
           { flag: 'a+' }
         )
       })

 })
});
```

Assertion Testing

Element contains specific text:

```
cy.get("h1").should("contain", “Dashboard”)
```

Retry until a spinner is hidden:

```
cy.get('[data-testid="loading"]').should("not.exist")
```

Result of get has correct number of elements:

```
cy.get("li.selected").should("have.length", 3)
```

Verify that a radio button has been checked:

```
cy.get(":radio").find("be.checked")
```

Assertion Testing

Element contains specific text:

```
cy.get("h1").should("contain", “Dashboard”)
```

Retry until a spinner is hidden:

```
cy.get('[data-testid="loading"]').should("not.exist")
```

Result of get has correct number of elements:

```
cy.get("li.selected").should("have.length", 3)
```

Verify that a radio button has been checked:

```
cy.get(":radio").find("be.checked")
```

Assertion Testing

Element contains specific text:

```
cy.get("h1").should("contain", “Dashboard”)
```

Retry until a spinner is hidden:

```
cy.get('[data-testid="loading"]').should("not.exist")
```

Result of get has correct number of elements:

```
cy.get("li.selected").should("have.length", 3)
```

Verify that a radio button has been checked:

```
cy.get(":radio").find("be.checked")
```

Assertion Testing

Element contains specific text:

```
cy.get("h1").should("contain", “Dashboard”)
```

Retry until a spinner is hidden:

```
cy.get('[data-testid="loading"]').should("not.exist")
```

Result of get has correct number of elements:

```
cy.get("li.selected").should("have.length", 3)
```

Verify that a radio button has been checked:

```
cy.get(":radio").find("be.checked")
```

Assertion Testing

Element contains specific text:

```
cy.get("h1").should("contain", “Dashboard”)
```

Retry until a spinner is hidden:

```
cy.get('[data-testid="loading"]').should("not.exist")
```

Result of get has correct number of elements:

```
cy.get("li.selected").should("have.length", 3)
```

Verify that a radio button has been checked:

```
cy.get(":radio").find("be.checked")
```

Testing with Plugins

```
describe('Post Resource', () => {
 it('Creating a New Post', () => {
   cy.visit('https://www.example.com/posts/new')

   cy.get("input.post_title")
     .find("p.post_body")
     .invoke("text")
     .then((post_text)=>{

       // insert record object to your MongoDB collection
       let record = {post: post_text}

       cy.insertOne(
          record,
          {
            database: 'some_database',
            collection: 'some_collection'
          })
         .then(response => {
           cy.log(response);
         });
     })
 })
});
```

Testing with Plugins

```
describe('Post Resource', () => {
 it('Creating a New Post', () => {
   cy.visit('https://www.example.com/posts/new')

   cy.get("input.post_title")
     .find("p.post_body")
     .invoke("text")
     .then((post_text)=>{

       // insert record object to your MongoDB collection
       let record = {post: post_text}

       cy.insertOne(
          record,
          {
            database: 'some_database',
            collection: 'some_collection'
          })
         .then(response => {
           cy.log(response);
         });
     })
 })
});
```

Testing with Plugins

```
describe('Post Resource', () => {
 it('Creating a New Post', () => {
   cy.visit('https://www.example.com/posts/new')

   cy.get("input.post_title")
     .find("p.post_body")
     .invoke("text")
     .then((post_text)=>{

       // insert record object to your MongoDB collection
       let record = {post: post_text}

       cy.insertOne(
          record,
          {
            database: 'some_database',
            collection: 'some_collection'
          })
         .then(response => {
           cy.log(response);
         });
     })
 })
});
```

Testing with Plugins

```
describe('Post Resource', () => {
 it('Creating a New Post', () => {
   cy.visit('https://www.example.com/posts/new')

   cy.get("input.post_title")
     .find("p.post_body")
     .invoke("text")
     .then((post_text)=>{

       // insert record object to your MongoDB collection
       let record = {post: post_text}

       cy.insertOne(
          record,
          {
            database: 'some_database',
            collection: 'some_collection'
          })
         .then(response => {
           cy.log(response);
         });
     })
 })
});
```

```
$ npm i cypress-mongodb
```

Testing with Plugins

```
describe('Post Resource', () => {
 it('Creating a New Post', () => {
   cy.visit('https://www.example.com/posts/new')

   cy.get("input.post_title")
     .find("p.post_body")
     .invoke("text")
     .then((post_text)=>{

       // insert record object to your MongoDB collection
       let record = {post: post_text}

       cy.insertOne(
          record,
          {
            database: 'some_database',
            collection: 'some_collection'
          })
         .then(response => {
           cy.log(response);
         });
     })
 })
});
```

```
$ npm i cypress-mongodb
```

```
# cypress.config.js

env: {
   mongodb: {
      uri: “mongodb+srv://…”
   }
}
```

```
# cypress/support/e2e.js

const mongo = require(“cypress-mongo”);
mongo.addCommands();
```

Executing Code in Node with cy.task()

```
# cypress.config.js

setupNodeEvents(on, config) {
 on("task", {
   runVenProxy({ file, token, environment_name, controller }) {
     venProxy = spawn(
       `./VenProxy --token=${token} --controller=${controller} > cypress/fixtures/${file} 2>&1`,
       [],
       {
         shell: true,
       }
     );
     console.log("[INFO] VenProxy started successfully");
     console.log(`[INFO] VenProxy controller API FQDN used for ${environment_name}:${controller}`);
     return venProxy;
   },
   runSampleConnector() {
     sampleConnector = spawn(`./sample-connector`, [], {
       shell: true,
     });
     console.log("[INFO] sample-connector started successfully");
     return sampleConnector;
   },
```

Executing Code in Node with cy.task()

```
# cypress.config.js

setupNodeEvents(on, config) {
 on("task", {
   runVenProxy({ file, token, environment_name, controller }) {
     venProxy = spawn(
       `./VenProxy --token=${token} --controller=${controller} > cypress/fixtures/${file} 2>&1`,
       [],
       {
         shell: true,
       }
     );
     console.log("[INFO] VenProxy started successfully");
     console.log(`[INFO] VenProxy controller API FQDN used for ${environment_name}:${controller}`);
     return venProxy;
   },
   runSampleConnector() {
     sampleConnector = spawn(`./sample-connector`, [], {
       shell: true,
     });
     console.log("[INFO] sample-connector started successfully");
     return sampleConnector;
   },
```

Executing Code in Node with cy.task()

```
# cypress.config.js

setupNodeEvents(on, config) {
 on("task", {
   runVenProxy({ file, token, environment_name, controller }) {
     venProxy = spawn(
       `./VenProxy --token=${token} --controller=${controller} > cypress/fixtures/${file} 2>&1`,
       [],
       {
         shell: true,
       }
     );
     console.log("[INFO] VenProxy started successfully");
     console.log(`[INFO] VenProxy controller API FQDN used for ${environment_name}:${controller}`);
     return venProxy;
   },
   runSampleConnector() {
     sampleConnector = spawn(`./sample-connector`, [], {
       shell: true,
     });
     console.log("[INFO] sample-connector started successfully");
     return sampleConnector;
   },
```

Executing Code in Node with cy.task()

```
# cypress.config.js

setupNodeEvents(on, config) {
 on("task", {
   runVenProxy({ file, token, environment_name, controller }) {
     venProxy = spawn(
       `./VenProxy --token=${token} --controller=${controller} > cypress/fixtures/${file} 2>&1`,
       [],
       {
         shell: true,
       }
     );
     console.log("[INFO] VenProxy started successfully");
     console.log(`[INFO] VenProxy controller API FQDN used for ${environment_name}:${controller}`);
     return venProxy;
   },
   runSampleConnector() {
     sampleConnector = spawn(`./sample-connector`, [], {
       shell: true,
     });
     console.log("[INFO] sample-connector started successfully");
     return sampleConnector;
   },
```

Executing Code in Node with cy.task()

```
# cypress.config.js

setupNodeEvents(on, config) {
 on("task", {
   runSampleConnector() {
     sampleConnector = spawn(`./sample-connector`, [], {
       shell: true,
     });
     console.log("[INFO] sample-connector started successfully");
     return sampleConnector;
   },
```

Executing Code in Node with cy.task()

```
# cypress.config.js

setupNodeEvents(on, config) {
 on("task", {
   runSampleConnector() {
     sampleConnector = spawn(`./sample-connector`, [], {
       shell: true,
     });
     console.log("[INFO] sample-connector started successfully");
     return sampleConnector;
   },
```

Executing Code in Node with cy.task()

```
# cypress.config.js

setupNodeEvents(on, config) {
 on("task", {
   runSampleConnector() {
     sampleConnector = spawn(`./sample-connector`, [], {
       shell: true,
     });
     console.log("[INFO] sample-connector started successfully");
     return sampleConnector;
   },
```

```
# testWithTask.cy.js

it("Starts Sample Connector", () => {
 cy.wait(5000);
 cy.task("runSampleConnector");
});
```

Executing Code in Node with cy.task()

```
# cypress.config.js

setupNodeEvents(on, config) {
 on("task", {
   runSampleConnector() {
     sampleConnector = spawn(`./sample-connector`, [], {
       shell: true,
     });
     console.log("[INFO] sample-connector started successfully");
     return sampleConnector;
   },
```

```
# testWithTask.cy.js

it("Starts Sample Connector", () => {
 cy.wait(5000);
 cy.task("runSampleConnector");
});
```

Executing Code in Node with cy.task()

```
# cypress.config.js

setupNodeEvents(on, config) {
 on("task", {
   runSampleConnector() {
     sampleConnector = spawn(`./sample-connector`, [], {
       shell: true,
     });
     console.log("[INFO] sample-connector started successfully");
     return sampleConnector;
   },
```

```
# testWithTask.cy.js

it("Starts Sample Connector", () => {
 cy.wait(5000);
 cy.task("runSampleConnector");
});
```

```
project_root
  |- cypress
  |    |- downloads
  |    |- e2e
  |    |   |- sampleTest.cy.js
  |    |- fixtures
  |    |- support
  |    |- utils
  |    |- videos
  |- cypress.config.js
```

Executing Code in Node with cy.task()

```
# cypress.config.js

setupNodeEvents(on, config) {
 on("task", {
   runSampleConnector() {
     sampleConnector = spawn(`./sample-connector`, [], {
       shell: true,
     });
     console.log("[INFO] sample-connector started successfully");
     return sampleConnector;
   },
```

```
# testWithTask.cy.js

it("Starts Sample Connector", () => {
 cy.wait(5000);
 cy.task("runSampleConnector");
});
```

```
project_root
  |- cypress
  |    |- downloads
  |    |- e2e
  |    |   |- sampleTest.cy.js
  |    |- fixtures
  |    |- support
  |    |- utils
  |    |- videos
  |- cypress.config.js
```

Testing Network Requests

```
it('cy.request() - make an XHR request', () => {

// https://on.cypress.io/request
 cy.request('https://jsonplaceholder.cypress.io/comments')
   .should((response) => {
     expect(response.status).to.eq(200)

     // the server sometimes gets an extra comment posted from another machine
     // which gets returned as 1 extra object
     expect(response.body).to.have.property('length').and.be.oneOf([500, 501])
     expect(response).to.have.property('headers')
     expect(response).to.have.property('duration')
   })

})
```

Testing Network Requests

```
it('cy.intercept() - route responses to matching requests', () => {

  // Listen to POST to comments
  cy.intercept('POST', '**/comments').as('postComment')

  // the button is clicked in scripts.js
  cy.get('.network-post').click()
  cy.wait('@postComment').should(({ request, response }) => {
    expect(request.body).to.include('email')
  })

})
```