Page Object Modeling and Re-Usability and Maintainability in Cypress Framework

▼ Custom Commands with Cypress

Cypress custom commands are described by users and not the default commands from Cypress. These customized commands are used to create the test steps that are repeated in an automation flow.

We can add and overwrite an already pre-existing command. They should be placed in the commands.js file within the support folder present in the Cypress project.

So, essentially this Cypress feature allow us to create customized Cypress commands for reusable pieces of code. We want to write custom commands for all functions that are globally repeating throughout the project.
For now, I came up with only one custom command that verifies if a text exists in the DOM. we can write it in commands.js file and use everywhere in our tests.

```
Cypress.Commands.add('textExists', (text) => {
    cy.contains(text).should('exist');
  });
```

▼ fixtures

We use this folder in the test structure to provide test data in dynamic way. Instead of writing test data in the Test files in a hard coded way, fixtures are preferred. Because these data might be needed throughout our test framework. And when these data need to be updated, we can do it from one location easily.

▼ Create a user.json file inside fixture folder and copy paste following for test purposes:

```
{
  "user1": {
```

```
    "firstName": "Harvey",
    "lastName": "Specter",
    "age": "40",
    "userEmail": "specter@example.com",
    "salary": "700000",
    "department": "legal"
  },
  "user2": {
    "username": "tomsmith",
    "password": "SuperSecretPassword"
  }
}
```

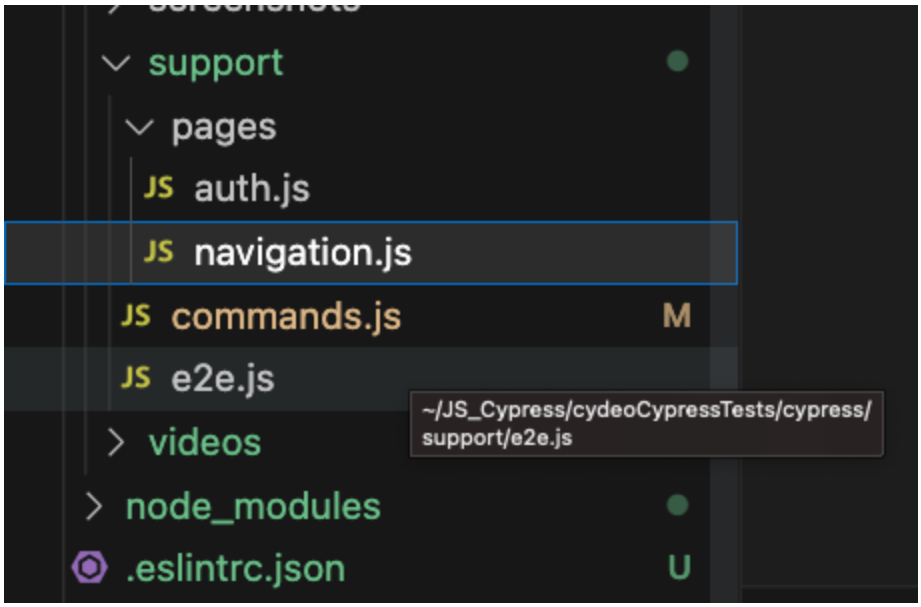
▼ Page Object Pattern

Page Object Model is a **design pattern** in the automation world which has been famous for its test maintenance approach and avoiding code duplication. **A page object** is a **class** that represents a page in the web application. Under this model, the overall web application
breaks down into logical pages.

Each page of the web application generally corresponds to one class in the page object, but can even map to multiple classes also, depending on the classification of the pages. This Page class will contain all the **locators** of the WebElements of that web page and will also contain **methods** that can perform operations on those WebElements.

▼ We will create pages inside support folder of Cypress Framework.

Let’s create auth.js and navigate.js files inside a folder named pages which will be inside support folder.



▼ **What are the benefits of using Page Object Pattern?**

As we have seen that the **Page Object Pattern** provides flexibility to writing code by splitting into different classes and also keeps the test scripts separate from the locators. Considering this few of the important benefits of the **Page Object Model** are:

Code reusability – The same page class can be used in different tests and all the locators and their methods can be re-used across various test cases.

Code maintainability – There is a clean separation between test code which can be our functional scenarios and page-specific code such as locators and methods. So, if some structural change happens on the web page, it will just impact the page object and will not
have any impacts on the test scripts.

From personal experience on enterprise projects, there is one advice I can give you - Sometimes you will have to test pages of web applications that are very complex, have multiple large features and require a massive amount of test cases.
In this case, you would NOT create only one page object class to store all page objects and methods for that page. What you will do to make code more maintainable is to separate page object classes by components (features) and

not by entire
page full of complex features. That is also a valid strategy, used mostly in enterprise projects.

Fun fact: this is also how developers structure their code, usually with components (frontend) or microservices(backend). This means breaking down enterprise app into smaller projects with large amount of smaller features/components/services etc.

▼ Some interview Questions for JS-Cypress

- synchronous vs async code?
- promises? states of promise?
- What is the difference between global and local variables in JS?
- What is the purpose of loops in JS? Name different types.
- What are some of the built-in methods in JavaScript?
- What are the purposes of fixtures, commands, config files in Cypress?
- Which type of testing Cypress can cover?
- How do you test iframes, webtables, alerts etc. (different webelements) in Cypress?