**My playwright notes:**

* Playwright.config is like a test runner
* Package.json is like pom and has dependency info
* Package-lock is a supporting file for above
* Node\_modules has all the downloaded modules similar to jars in java

Base structure of a PW test in JS

test( 'Test Name' , function()

{

});

OR

test( 'Test Name' , ()=>

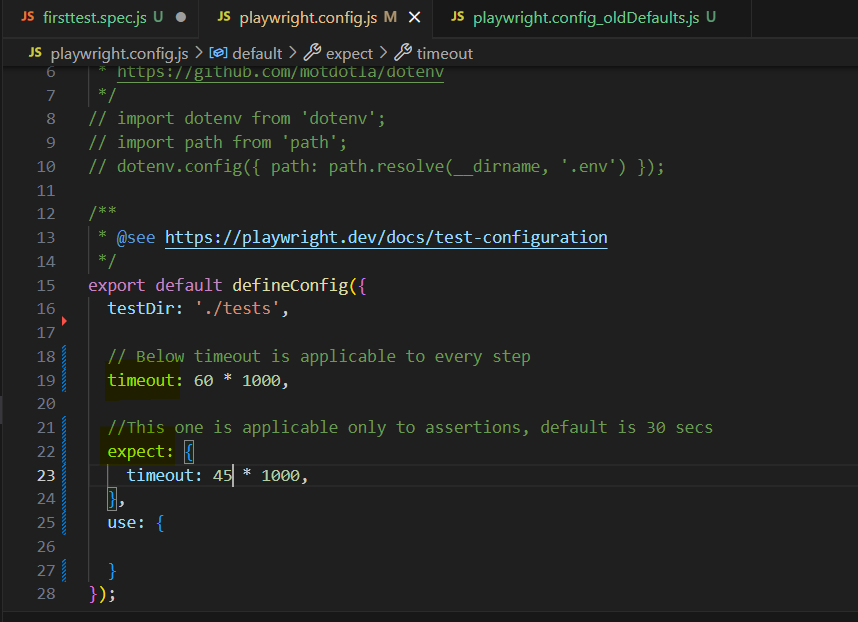
{

});

As JS/TS are asynchronous all the test steps get triggered parallelly, so to make them run one after the other we use **await** keyword in front of each step.

**Await** keyword on its own get ignored so to make it come into effect we need to use the **async** keyword ahead of the function theat we pass to out test.

* In newer versions of JS/TS the function keyword need not be mentioned and we can use a anonymous function instead ‘()=>’
* Browser is a global playwright fixture passed in curly braces to the test it is part of the npm playwright package and does not need a separate import.
* Fixtures are nothing but global variable available across your project.



* test**.only** will trigger only that test in the test file ignoring all others

For CSS selectors:

* If id is present then tagname#id
* If class is present then tagname.class (If multiple classes separated by spaces are present use dots in selector instead of space i.e tagName.class1.class2.class3)
* If need to use other attribute then tagname[attribute=’value’]
* To traverse parent to child, parentTag > childTag
* If need to write locator with text, tagname[text=’textStr’]

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Example description** |
| [[attribute]](https://www.w3schools.com/cssref/sel_attribute.php) | [lang] | Selects all elements with a lang attribute |
| [[attribute=value]](https://www.w3schools.com/cssref/sel_attribute_value.php) | [lang="it"] | Selects all elements with lang="it" |
| [[attribute~=value]](https://www.w3schools.com/cssref/sel_attribute_value_contains.php) | [title~="flower"] | Selects all elements with a title attribute containing the word "flower" |
| [[attribute|=value]](https://www.w3schools.com/cssref/sel_attribute_value_lang.php) | [lang|="en"] | Selects all elements with a lang attribute value equal to "en" or starting with "en-" |
| [[attribute^=value]](https://www.w3schools.com/cssref/sel_attr_begin.php) | [href^="https"] | Selects all elements with a href attribute value that begins with "https" |
| [[attribute$=value]](https://www.w3schools.com/cssref/sel_attr_end.php) | [href$=".pdf"] | Selects all elements with a href attribute value ends with ".pdf" |
| [[attribute\*=value]](https://www.w3schools.com/cssref/sel_attr_contain.php) | [href\*="keyboard"] | Selects all elements with a href attribute value containing the substring "keyboard" |

* page.context().waitForEvent("page") will start waiting for new page event for the timeout duration mentioned in the config file but that event will only get triggered when a click operation is done on a element that opens the page.
* The click operation for new page if used first and page.context().waitForEvent("page") is used after it, then it will not be able to capture it as the event has already happened.
* To ensure smooth transition to new window we need to run the page.context().waitForEvent("page") and the click step asynchronously wusing Promise.all() method.
* The input is an array of steps that will be triggered asynchronously and return type will be an array of newpages or a single new page that were opened based on the steps triggered in Promise.all().
* await newPage.close(); *// Close the new page if needed*
* await oldPage.bringToFront(); *// Optional: bring the old page to the front*

**For recording and auto generating playwright script use below command:**

* npx playwright codegen <your-app-url>

**For enabling screenshots and trace use below in playwright.config.js**

