```
import { test, expect } from '@playwright/test';

test('Interact with Tree View on LeafGround', async ({ page }) => {
  // Navigate to the Tree View page
  await page.goto('https://leafground.com/tree.xhtml');

  // Wait for the tree to be visible
  const treeLocator = page.locator('.ui-tree'); // Tree container
  await expect(treeLocator).toBeVisible();

  // Expand the first parent node
  const firstExpandIcon = page.locator('.ui-tree-toggler').first();
  await firstExpandIcon.click();

  // Verify child nodes are visible
  const childNodes = page.locator('.ui-treenode-content').nth(1); // Adjust index if needed
  await expect(childNodes).toBeVisible();

  // Expand deeper level (if exists)
  const secondExpandIcon = page.locator('.ui-tree-toggler').nth(1);
  await secondExpandIcon.click();

  // Retrieve all visible node texts
  const allNodes = await page.locator('.ui-treenode-content').allTextContents();
  console.log('Visible Tree Nodes:', allNodes);

  // Assert that expected nodes are present
  expect(allNodes).toContain('Documents'); // Example node
  expect(allNodes).toContain('Pictures');  // Example node
});
```

frames:

```
import {test,chromium} from '@playwright/test';

test('Working with frames',async()=>{
        const browser = await chromium.launch({ headless: false });
        // Launch browser in headed mode
        const page = await browser.newPage();

        await page.goto('https://ui.vision/demo/webtest/frames/');

        //find total frames
        const allf =await page.frames();
   console.log('Total frames: ',allf.length);

        //using framename or url of the page of frame that is frame object
        const frame1= await
page.frame({url:'https://ui.vision/demo/webtest/frames/frame_1.html'});
        //const frame1=await page.frame('name of frame if avaible') //if u want to use
name
        await frame1.fill("[name='mytext1']",'Welcome');
        await page.waitForTimeout(3000);

        //using frame locator
        const frameinput=await
page.frameLocator("frame[src='frame_1.html']").locator("[name='mytext1']");
        await frameinput.fill('Hello');
        await page.waitForTimeout(3000);


        const frame3 = await page.frame({ url:
'https://ui.vision/demo/webtest/frames/frame_3.html' }); if (frame3) {
 await frame3.locator("[name='mytext3']").fill('frame3');
 await page.waitForTimeout(3000);


 // Nested frame
const childFrames = frame3.childFrames();
 console.log('Total childframes: ', childFrames.length);


if (childFrames.length > 0) {
//await page.pause();
//*[@id="i6"]/div[3]/div
 await childFrames[0].locator("//*[@id='i6']/div[3]/div").check();
 await page.waitForTimeout(4000);
} else {
 console.log('No child frames found');}
 } else { console.log('Frame3 not found');
}




});
```
===

Write to file

```
import { test, chromium } from '@playwright/test';
const path =require('path');
const fs = require('fs');

test('login with test data', async () => {



 const browser = await chromium.launch({ headless: false });
          const page = await browser.newPage();

  // Navigate to Sauce Demo
  await page.goto('https://the-internet.herokuapp.com/');
  await page.waitForLoadState('load');

  const list = await page.locator('//li/a').count();
   const testimonials = [];

  for(let i=0; i<list; i++)
  {
   const text=await page.locator('//li/a').nth(i).textContent();
   testimonials.push(text?.trim());



  }
//fs.writeFileSync('../File/output.txt',testimonials.join('\n'), 'utf8');
// Ensure the directory exists
  const dirPath = path.join(__dirname, '../File');
  if (!fs.existsSync(dirPath)) {
    fs.mkdirSync(dirPath, { recursive: true });
  }

  // Write to file
  const filePath = path.join(dirPath, 'output.txt');
  fs.writeFileSync(filePath, testimonials.join('\n'), 'utf8');



});
```
===
=======

Fixture:

```
import { test as base, expect,Page ,chromium} from '@playwright/test';


type Fixtures = {
  loggedInPage: Page;
};


const test = base.extend<Fixtures>({
  loggedInPage: async ( {},use) => {
    const browser= await chromium.launch({headless:false})
    const context = await browser.newContext();
    const page = await context.newPage();
    await page.goto('https://the-internet.herokuapp.com/login');
    await page.fill('#username', 'tomsmith');
    await page.fill('#password','SuperSecretPassword!');
    await page.click('button[type="submit"]');
    await use(page);
  },
});

test('dashboard loads after login', async ({ loggedInPage }) => {
  await loggedInPage.goto('https://the-internet.herokuapp.com/secure');
  expect(await loggedInPage.isVisible('text= Secure Area')).toBeTruthy();
});
```

POM and Data Driven Testing:

```
 ├── tests/
│   └── login.spec.ts
 ├── pages/
│   └── LoginPage.ts
 ├── utils/
│   └── testData.ts
 ├── playwright.config.ts
```

pages/LoginPage.ts – Page Object Model

```
import { Page } from '@playwright/test';

export class LoginPage {
  constructor(private page: Page) {}

  async navigate() {
        await
this.page.goto('https://opensource-demo.orangehrmlive.com/web/index.php/auth/login');
  }

  async enterUsername(username: string) {
        await this.page.getByLabel('Username').fill(username);
  }

  async enterPassword(password: string) {
        await this.page.getByLabel('Password').fill(password);
  }

  async clickLogin() {
        await this.page.getByRole('button', { name: 'Login' }).click();
  }

  async isDashboardVisible(): Promise<boolean> {
        return await this.page.getByText('Dashboard').isVisible();
  }
}
```

utils/testData.ts – Data-Driven Testing

```
export const loginData = [
  { username: 'Admin', password: 'admin123', expected: true },
  { username: 'invalidUser', password: 'invalidPass', expected: false },
];
```

tests/login.spec.ts – Modular Test Using POM + Data-Driven

```
import { test, expect } from '@playwright/test';
import { LoginPage } from '../pages/LoginPage';
import { loginData } from '../utils/testData';

for (const data of loginData) {
  test(`Login test with username: ${data.username}`, async ({ page }) => {
          const loginPage = new LoginPage(page);

          await loginPage.navigate();
          await loginPage.enterUsername(data.username);
          await loginPage.enterPassword(data.password);
          await loginPage.clickLogin();

          const isDashboardVisible = await loginPage.isDashboardVisible();

          if (data.expected) {
          expect(isDashboardVisible).toBeTruthy();
          } else {
          await expect(page.getByText('Invalid credentials')).toBeVisible();
          }
  });
}
```

==
=======

Benefits

Feature   ----Benefit

POM   ----------Clean separation of UI logic

Modular Functions-------   Reusable methods for login actions

Data-Driven Testing-----Easy to scale with multiple test cases

Maintainability------   Easy to update selectors or flows in one place

pages/testData.json

```
[

{
   "username": "tomsmith",
            "password": "SuperSecretPassword!"
},
{
  "username": "invalidmith",
          "password": "SuperSecret"
}

]
```

reading data from json file

```
import {test,chromium} from '@playwright/test';
import testData from  '../pages/testData.json';

//test.describe('Combined API Tests', () => {
for(const arr of testData)
{
test(`login validation ${arr.username}`, async () => {


  const browser = await chromium.launch({headless:false}) ;
  const page= await browser.newPage();

  await page.goto('https://the-internet.herokuapp.com/login');
  console.log(arr.username);
  await page.fill('//*[@id="username"]',arr.username);
  console.log(arr.password)
  await page.fill('//*[@id="password"]',arr.password);
  await page.click('//*[@id="login"]/button/i');

  await page.waitForTimeout(2000);

});
}
```

==
=====

POM -pages+ts+testscript

pages/testdata.ts

```
export const loginData = [
  { username: 'tomsmith', password: 'SuperSecretPassword!'},
  { username: 'invalidUser', password: 'invalidPass' },
];
```

pages/LoginPage.ts

```
import { Browser, Page } from "@playwright/test";
```

```
export class LoginPage{


          private browser:Browser;
          private page:Page;



constructor(browser:Browser,page:Page)
{
this.browser=browser;
this.page =page;

}




async launchpage(url:string,username:string,password:string) {



await this.page.goto(url);
await this.page.fill('//*[@id="username"]',username);
await this.page.fill('//*[@id="password"]',password);
await this.page.click('//*[@id="login"]/button/i');
}



}
```

});

testscript-herokuPOM.spec.ts

```
import {test,chromium} from '@playwright/test';
//import testData from  '../pages/testData.json';
import {loginData} from '../pages/testdata';
import { LoginPage } from '../pages/LoginPage';

//test.describe('Combined API Tests', () => {
for(const arr of loginData)
{
test(`login validation ${arr.username}`, async () => {

const browser = await chromium.launch({headless:false});
const page = await browser.newPage();
  const obj = new LoginPage(browser,page);
  await
obj.launchpage('https://the-internet.herokuapp.com/login',arr.username,arr.password);

});
}
//});
```