Salesforce: New Task

---------------------

Steps :

1. Go to https://login.salesforce.com/?locale=in

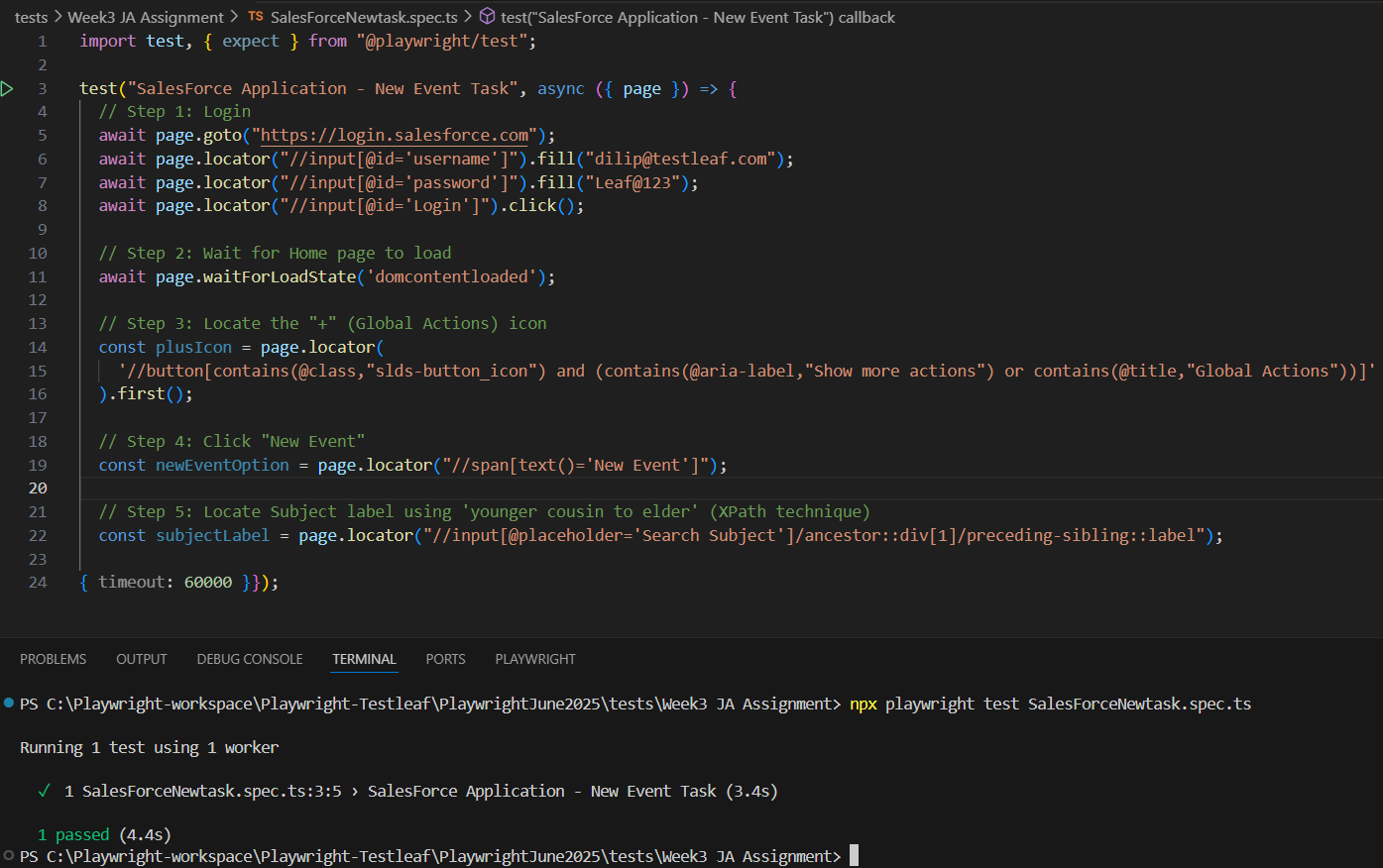
2. And login using credentials. [username : ravindran.ramdas@testleaf.com, password : RaviSalesTest]

3. Click on the "+" icon (6th icon from top right corner)

4. Choose and click "New Task" from dropdown.

5. Now get the xpath for "Subject" label web element.

Hint : New Task younger cousin to elder



leaftaps: First leadid

-----------------------

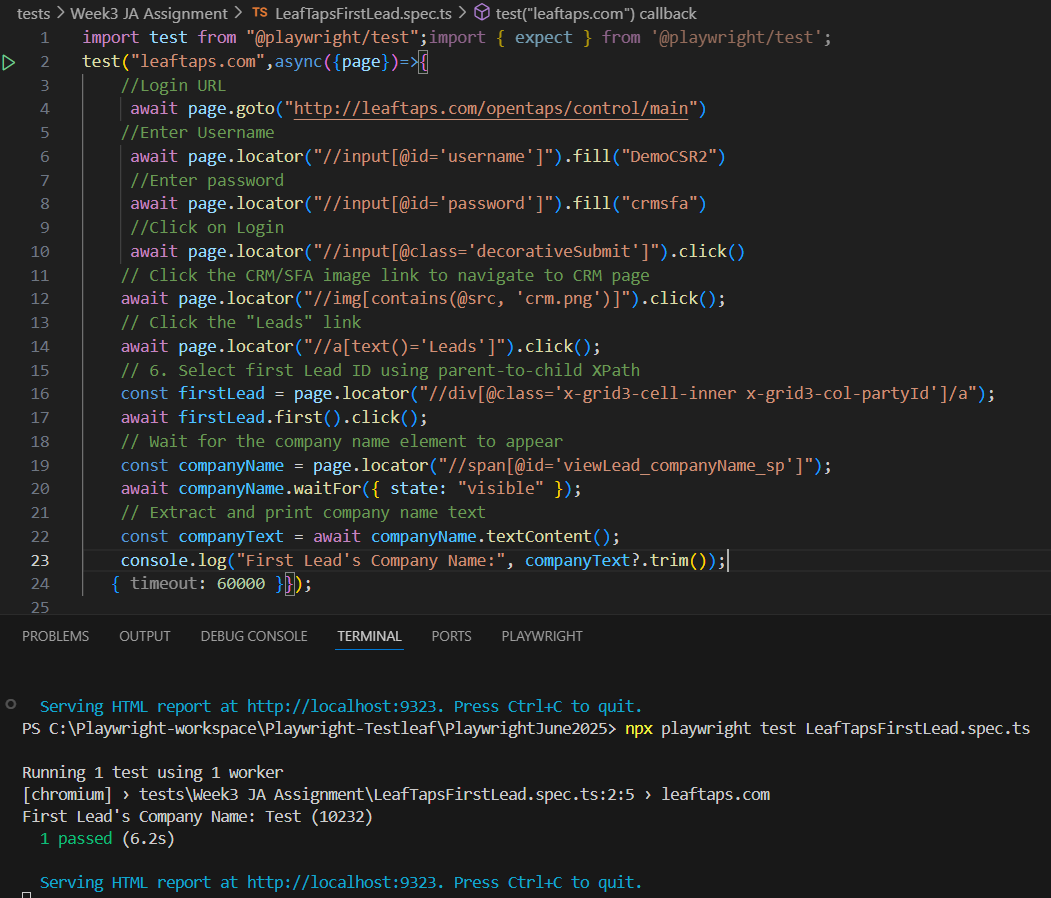
Steps :

1. Go to "http://leaftaps.com/opentaps/control/login" and login.

2. Click CRM/SFA .

3. Click "Leads" select the first lead from the list of leads

leaftaps: First leadid paren to child



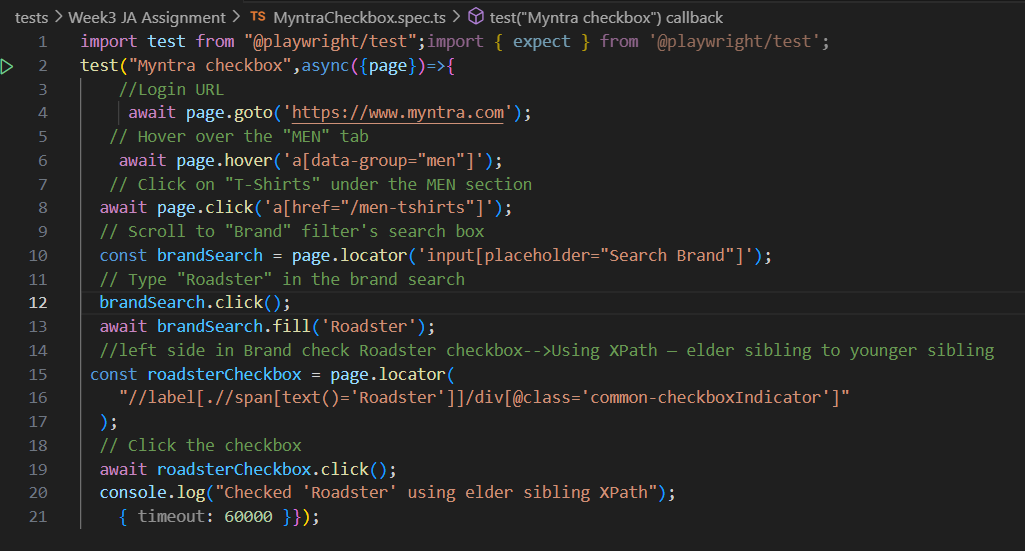
Myntra: Checkbox

-----------------

-> Go to mynthra.com choose "MEN" from the tab available.

-> Select the check box of "Roadster" under the brands

Myntra: Checkbox elder sibling to younger sibling

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Interview Questions Arrays:

Q1. What is the difference between map(), forEach(), and filter()?

**map()**

**Purpose:** Transforms each element in an array and returns a **new array** of the same length.

**Use case:** When you want to modify or transform the elements of an array.

**Returns:** A **new array** with the results of calling a function on every element.

javascript

CopyEdit

const nums = [1, 2, 3];

const doubled = nums.map(n => n \* 2);

console.log(doubled); // [2, 4, 6]

**🔄 forEach()**

**Purpose:** Executes a function once for each array element.

**Use case:** When you want to perform side effects (like logging or updating UI), not transformation.

**Returns:** **Nothing** (i.e., undefined).

javascript

CopyEdit

const nums = [1, 2, 3];

nums.forEach(n => console.log(n \* 2));

// Output:

// 2

// 4

// 6

**🔍 filter()**

**Purpose:** Filters elements based on a condition and returns a **new array** with only those that match.

**Use case:** When you want to **remove** elements that don't meet a condition.

**Returns:** A **new array** with elements that pass the test.

javascript

CopyEdit

const nums = [1, 2, 3, 4];

const evens = nums.filter(n => n % 2 === 0);

console.log(evens); // [2, 4]

**🧠 Summary Table:**

| **Method** | **Returns New Array** | **Can Modify Elements** | **Can Remove Elements** | **Side Effects** |
| --- | --- | --- | --- | --- |
| map() | Yes | Yes | 🚫 No | ❌ Avoid |
| forEach() | ❌ No | Yes (but not returned) | 🚫 No | Yes |
| filter() | Yes | 🚫 No | Yes | ❌ Avoid |

Q2. What is the difference between slice() and splice()?

**slice()**

**Purpose:** Creates a **shallow copy** of part of an array into a **new array** without modifying the original.

**Returns:** A **new array**

**Syntax:**

javascript

CopyEdit

array.slice(start, end);

* start: Index to begin extraction (inclusive)
* end: Index to end extraction (exclusive)

javascript

CopyEdit

const nums = [1, 2, 3, 4, 5];

const part = nums.slice(1, 4);

console.log(part); // [2, 3, 4]

console.log(nums); // [1, 2, 3, 4, 5] (unchanged)

**🛠️ splice()**

**Purpose:** Changes an array **in place** by removing, replacing, or adding elements.

**Returns:** An array of **removed elements**

**Syntax:**

javascript

CopyEdit

array.splice(start, deleteCount, item1, item2, ...)

* start: Index to begin changes
* deleteCount: Number of elements to remove
* item1, item2, ...: Elements to insert (optional)

javascript

CopyEdit

const nums = [1, 2, 3, 4, 5];

const removed = nums.splice(1, 2, 'a', 'b');

console.log(removed); // [2, 3]

console.log(nums); // [1, 'a', 'b', 4, 5] (modified)

| **Feature** | **slice()** | **splice()** |
| --- | --- | --- |
| **Modifies original?** | ❌ No | Yes |
| **Returns** | New array (copied portion) | Removed elements |
| **Use case** | Copy part of array | Add/remove/replace elements |
| **Destructive?** | ❌ No | Yes |

Q3. How do you convert an array to a string? And string to array?

**Array ➡️ String**

**join() method**

Converts an array to a string by **joining elements with a separator**.

const fruits = ['apple', 'banana', 'cherry'];

const result = fruits.join(', ');

console.log(result); // "apple, banana, cherry"

* Default separator is a comma , if not specified.
* You can use any string (e.g., space, hyphen) as a separator.

fruits.join(' - '); // "apple - banana - cherry"

**🔁 String ➡️ Array**

**split() method**

Splits a string into an array based on a **separator**.

const str = "apple, banana, cherry";

const arr = str.split(', ');

console.log(arr); // ["apple", "banana", "cherry"]

* The separator defines **how to break the string**.
* You can split on commas, spaces, characters, etc.

"hello world".split(' '); // ["hello", "world"]

"abc".split(''); // ["a", "b", "c"]

Q4. Find all pairs in an array whose sum is a given number

Function to find all pairs in an array that sum up to a given target

function findPairs(arr, target) {

// Loop through each element in the array

for (let i = 0; i < arr.length; i++) {

// Inner loop starts from the next element after i

for (let j = i + 1; j < arr.length; j++) {

// Check if the sum of arr[i] and arr[j] equals the target

if (arr[i] + arr[j] === target) {

// If so, print the pair to the console

console.log(`Pair found: (${arr[i]}, ${arr[j]})`);

} } }}

// Sample array of numbers

const numbers = [2, 4, 3, 5, 7, 8, 9];

// Target sum to find in the array

const targetSum = 10;

// Call the function with the array and target sum

findPairs(numbers, targetSum);

output:

Pair found: (2, 8)

Pair found: (3, 7)