"TypeScript for Automation QA" series is a complete roadmap designed to take you from the basics to advanced patterns, all from a tester's perspective.

idavidov.eu

2

TypeScript for
Automation QA

Core Types

```ts
//String - for URLs, locators and text messages
const blogURL: string = 'https://idavidov.eu/';
//Number - for item counts, prices, and status code
const statusCode: number = 200;
//Boolean - for element visibility and checkbox states
const toSubscribe: boolean = true;
//String Literals
const newsletterURL: string = `${blogURL}newsletter`;
```

3

TypeScript for
Automation QA

Arrays

```ts
//Arrays
const currency: string[] = ['USD', 'EUR', 'GBP']

// The .map() method returns a new array of strings
const currencyNames: string[] = await Promise.all(
    allOptions.map(option => option.textContent())
); // Output: ['USD', 'EUR', 'GBP']

// .forEach() performs an action for each item
currencyNames.forEach(name => {
  console.log(`Verifying currency: ${name}`);
});

// .filter() returns a new array with only the
matching items
const u_currency: string[] =
currencyNames.filter(name => name.includes('u'));
// u_currency is now ['USD', 'EUR']
```

idavidov.eu

4

TypeScript for
Automation QA

Objects

```ts
//Lets define Interface (set the structure of the object)
  interface Subsciber {
  id: number;
  email: string;
}
// Create a subscriber object that matches the interface
const firstSubscriber: Subsciber = {
  id: 1,
  email: 'firstSubscriber@example.com',
};
```

idavidov.eu

5

TypeScript for
Automation QA

Functions

```ts
// A named function for a common user action
async function navigateToHomePage() {
  await page.goto('https://idavidov.eu/');
  await expect(page.getByRole('heading', { name:
'Intelligent Quality' })).toBeVisible();
}

// The function expects an object with username and
password
async function login(username: string, password: string) {
  await page.getByLabel('Username').fill(username);
  await page.getByLabel('Password').fill(password);
  await page.getByRole('button', { name: 'Log In'
}).click();
}
```

idavidov.eu

6

TypeScript for
Automation QA

Loops

```ts
const productLocators = await page.locator('.product-item').all();

// Loop from the first element (index 0) to the last
for (let i = 0; i < productLocators.length; i++) {
  console.log(`Checking product #${i + 1}`);
  await expect(productLocators[i]).toBeVisible();
}

// Loop through each locator directly, no index needed
for (const locator of productLocators) {
  await expect(locator).toBeVisible();
}
```

idavidov.eu

7

TypeScript for
Automation QA

Conditions

```ts
async function registerUser(
  email: string,
  subscribeToNewsletter?: boolean,
) {
  await page.getByLabel('Email').fill(email);

  // If the subscribeToNewsletter parameter was passed as
true...
  if (subscribeToNewsletter) {
    // ...then click the newsletter checkbox.
    await page.getByLabel('Subscribe to
newsletter').check();
  }
}
```

idavidov.eu

8

TypeScript for
Automation QA

Custom Types

```ts
// Define a type that only allows these four specific
strings
type HttpMethod = 'GET' | 'POST' | 'PUT' | 'DELETE';

async function sendRequest(url: string, method:
HttpMethod) {
    // ... your API call logic here
    console.log(`Sending a ${method} request to ${url}`);
}

// ✅ This works perfectly:
sendRequest('/api/users', 'POST');

// 🚨 TypeScript stops you right here! No runtime
surprise.
// Argument of type '"FETCH"' is not assignable to
parameter of type 'HttpMethod'.
sendRequest('/api/users', 'FETCH');
```

idavidov.eu

9

TypeScript for
Automation QA

Classes

```ts
export class NavPage {
    constructor(private page: Page) {}

    // This code only runs when you call
`navPage.homePageLink`
    get homePageLink() {
        return this.page.getByRole('link', {
            name: 'Home',
        });
    }
}
```

idavidov.eu

10 TypeScript for Automation QA

Promises

```
const [publishActionPromise, responsePromise] = await
Promise.all([
    // Operation 1: Start publishing the article
    articlePage.publishArticle(title, description, body,
tags),

    // Operation 2: Start listening for the API response
SIMULTANEOUSLY
    page.waitForResponse('**/api/articles/'),
]);

// Now you can work with the results after both are
complete
const responseBody = await responsePromise.json();
```

idavidov.eu

11

TypeScript for
Automation QA

Enums

```typescript
export enum UserRole {
    ADMIN = 'admin',
    EDITOR = 'editor',
    VIEWER = 'viewer',
}

function assignRole(username: string, role: UserRole) {
    console.log(`Assigning role: ${role} to ${username}`);
}

// 1. No typos: TS would throw an error if UserRole.ADMNIN
existed.
// 2. Autocomplete: Your editor will suggest ADMIN,
EDITOR, or VIEWER.
assignRole('idavidov', UserRole.ADMIN);
```

idavidov.eu

Share in the comments the most unclear topic for you?

The whole tutorial can be found

👇

idavidov.eu