

Claro. Vou transformar o conteúdo em um formato de estudo para celular, com blocos curtos, leitura vertical, bastante espaço visual e uma sequência progressiva. Também vou eliminar repetições desnecessárias para facilitar a revisão.



PLAYWRIGHT + TYPESCRIPT

Código básico explicado do ZERO

OBJETIVO

Ao terminar este estudo, você deverá conseguir olhar para este código:

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

test('basic test', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  const name = await page.innerText('.navbar__title');
  expect(name).toBe('Playwright');
});
```

e entender:

- o que cada palavra significa;
- o que cada símbolo faz;
- como o teste funciona;
- o que é page;
- o que é await;
- o que é expect;
- o que é toBe;
- o que é async;
- o que é const;
- o que é um CSS selector;
- como isso se relaciona com AAA.



1. A IDEIA GERAL

Antes de estudar cada palavra, entenda a história do código.

O teste faz isto:

IMPORTAR



CRIAR TESTE



ABRIR PÁGINA

↓
PEGAR TEXTO

↓
VERIFICAR TEXTO

Em linguagem humana:

"Abra o site do Playwright, encontre o título, pegue o texto e confirme que ele é 'Playwright'."



2. O CÓDIGO COMPLETO

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

test('basic test', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  const name = await page.innerText('.navbar__title');
  expect(name).toBe('Playwright');
});
```



3. PRIMEIRA LINHA

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

Leia assim:

"Importe test e expect do Playwright."

Agora vamos quebrar.



4. import

Inglês

import

Português

importar

O que faz?

Traz alguma coisa de outro módulo ou biblioteca para o seu código.

Imagine:

PLAYWRIGHT

↓

import

↓

SEU TESTE

Você está dizendo:

"Quero utilizar ferramentas do Playwright."



5. { test, expect }

```
{ test, expect }
```

Estamos escolhendo quais recursos queremos importar.

Nesse caso:

```
test  
expect
```

Portanto:

```
import { test, expect }
```

significa:

"Importe test e expect."



6. test

```
test
```

Português

teste

O que é?

É uma função fornecida pelo Playwright Test para criar um teste automatizado.

Exemplo:

```
test('login funciona', async ({ page }) => {  
  
});
```

Leia:

"Crie um teste chamado login funciona."



7. expect

expect

Português

esperar / esperar que

No contexto de testes, pense:

"Eu espero que isso seja verdadeiro."

Exemplo:

```
expect (name) .toBe ( 'Playwright' );
```

Leia:

"Eu espero que name seja Playwright."

expect é utilizado para fazer uma assertion.



8. from

from

Português

de

Temos:

```
import { test, expect } from ...
```

Leia:

"Importe test e expect DE..."



9. '@playwright/test'

```
'@playwright/test'
```

É o nome do pacote que fornece o Playwright Test.

Pense:

```
@playwright/test
```

↓

PACOTE

↓

ferramentas

↓

test

expect

fixtures

...

As aspas indicam que isso é um texto/nome de módulo.



10. ;

;

Nome

ponto e vírgula

Função

Indica o final da instrução.

Pense:

```
instrução
```

↓

;

```
fim
```

Em JavaScript/TypeScript, muitas vezes ele pode ser omitido, mas é comum utilizá-lo.



11. AGORA COMEÇA O TESTE

```
test('basic test', async ({ page }) => {
```

Essa linha é muito importante.

Vamos quebrar em pequenas partes.



12. test(...)

```
test(...)
```

Aqui estamos chamando a função test.

Anteriormente:

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

Agora:

```
test(...)
```

Estamos utilizando o que importamos.



13. (

```
test(
```

O parêntese indica que estamos passando informações para a função.

Essas informações são chamadas de:

argumentos

Exemplo:

```
test('basic test', ...)
```

O primeiro argumento é:

```
'basic test'
```



14. 'basic test'

```
'basic test'
```

Isso é uma:

string

String

É um texto.

Exemplos:

```
'Olá'  
'Playwright'  
'Login'  
'basic test'
```

Nesse código:

```
'basic test'
```

é o nome do teste.

Você poderia escrever:

```
test('login com sucesso', ...)
```

ou:

```
test('usuário consegue acessar o sistema', ...)
```



15. async

async

Português

assíncrono

É uma palavra muito importante no Playwright.

Ela indica que a função trabalha com operações que podem precisar ser aguardadas.

Por exemplo:

```
await page.goto(...)
```

O navegador precisa:

RECEBER COMANDO

↓

ABRIR PÁGINA



AGUARDAR



CONTINUAR

Memorize:

async → permite trabalhar com operações assíncronas.



16. page

page

Português

página

Representa uma página do navegador.

Imagine:

BROWSER



CONTEXT



PAGE

O page permite fazer coisas como:

```
await page.goto(...)
await page.click(...)
await page.fill(...)
await page.screenshot(...)
```

Memorize:

page = página do navegador.



17. ({ page })

({ page })

Aqui temos um conceito de JavaScript chamado:

destructuring

ou:

desestruturação

No Playwright, page é fornecido ao teste através de uma fixture.

Simplificando:

PLAYWRIGHT

↓

TEST

↓

FIXTURES

↓

PAGE

Então:

```
{ page }
```

pode ser entendido como:

"Quero receber a fixture page neste teste."



18. =>

=>

Nome

Arrow Function

Português

função de seta

Exemplo:

```
const somar = (a, b) => {  
  return a + b;  
};
```

No nosso código:

```
async ({ page }) => {
```

significa:

"Aqui começa a função que será executada pelo teste."



19. {

```
{
```

Abre o bloco de código.

Tudo dentro:

```
{
```

```
    ...
```

```
}
```

pertence à função.



20. PRIMEIRA AÇÃO

```
await page.goto('https://playwright.dev/');
```

Essa é a primeira ação do teste.

Vamos dividir:

```
await
```

```
+
```

```
page
```

```
+
```

```
.
```

```
+
```

```
goto
```

```
+
```

```
URL
```



21. await

```
await
```

Português

aguarde

Leia:

"Execute a operação e aguarde antes de continuar."

Exemplo:

```
await page.goto(...)
```

Mentalmente:

GOTO

↓

ABRIR PÁGINA

↓

AGUARDAR

↓

CONTINUAR

Memorize:

`await` = aguarde a operação terminar.



22. page

page

Já sabemos:

page representa a página do navegador.



23. .

`page.goto`

O ponto:

.

permite acessar uma propriedade ou método de um objeto.

Exemplo:

`page.goto`

Leia:

"Acesse o método goto do objeto page."

24. goto

`goto()`

Vem de:

go to

Português

ir para

No Playwright:

`page.goto()`

significa:

"Navegar para uma página."

Exemplo:

```
await page.goto('https://google.com');
```

Leia:

"Abra o Google."

25. URL

`'https://playwright.dev/'`

É uma URL.

Ela indica para qual endereço o navegador deve navegar.

Portanto:

```
await page.goto('https://playwright.dev/');
```

significa:

"Aguarde o navegador navegar para o site do Playwright."

26. PRIMEIRA FRASE COMPLETA

```
await page.goto('https://playwright.dev/');
```

Tradução mental:

"Aguarde o navegador abrir o site Playwright."



27. SEGUNDA LINHA

```
const name = await page.innerText('.navbar__title');
```

Essa linha possui vários conceitos importantes.



28. const

const

Português

constante

É utilizado para declarar uma variável que não será reatribuída.

Exemplo:

```
const nome = 'João';
```

Temos:

nome

↓

"João"

No nosso código:

```
const name = ...
```

Estamos criando uma variável chamada:

name



29. name

name

É simplesmente o nome escolhido para a variável.

Poderia ser:

```
const nome = ...
```

ou:

```
const texto = ...
```

ou:

```
const titulo = ...
```

O programador escolheu:

name



30. =

=

Nome

atribuição

Significa:

"Coloque o valor da direita dentro da variável da esquerda."

Exemplo:

```
const name = 'Playwright';
```

Mentalmente:

"Playwright"

↓

name



31. innerText

```
page.innerText()
```

Significado

texto interno/visível

O método busca o texto de um elemento.

No nosso caso:

```
page.innerText('.navbar__title')
```

significa:

"Pegue o texto do elemento encontrado pelo seletor `.navbar__title`."



32. `.navbar__title`

```
'.navbar__title'
```

Isso é um:

CSS Selector

O ponto inicial:

.

indica uma classe CSS.

Portanto:

```
.navbar__title
```

significa:

"Procure um elemento que tenha a classe `navbar__title`."



33. EXEMPLO HTML

Imagine que a página tenha:

```
<div class="navbar__title">
  Playwright
</div>
```

Temos:

`div`

↓

elemento

`class`

↓
navbar__title

texto

↓
Playwright

O selector:

`.navbar__title`

encontra esse elemento.



34. SEGUNDA FRASE COMPLETA

```
const name = await page.innerText('.navbar__title');
```

Tradução mental:

"Crie uma variável chamada name e coloque nela o texto encontrado no elemento que possui a classe navbar__title."

O resultado provavelmente será:

```
name = "Playwright"
```



35. AGORA: ASSERT

```
expect(name).toBe('Playwright');
```

Essa é a verificação.

Aqui aparece um dos conceitos mais importantes de testes:

ASSERT



36. AAA

AAA significa:

ARRANGE

ACT

ASSERT

Português:

ARRANGE
Preparar

ACT
Executar

ASSERT
Verificar

37. ARRANGE

```
await page.goto('https://playwright.dev/');
```

Estamos preparando o cenário.

ARRANGE



Preparar

38. ACT

```
const name = await page.innerText('.navbar__title');
```

Estamos executando uma ação.

ACT



Executar

39. ASSERT

```
expect(name).toBe('Playwright');
```

Estamos verificando o resultado.

ASSERT



Verificar

40. expect

expect

Português

esperar / esperar que

No contexto de testes:

"Eu espero que..."

Exemplo:

`expect (name)`

Leia:

"Eu espero que name..."



41. (name)

`expect (name)`

Queremos verificar o valor armazenado em:

`name`

Se anteriormente:

`name = "Playwright"`

então:

`expect (name)`

está verificando:

`"Playwright"`



42. .toBe()

`.toBe ()`

É um matcher de assertion.

Significado

deve ser exatamente igual a

Então:

```
expect (name) .toBe ('Playwright');
```

significa:

"Espero que name seja exatamente Playwright."

43. 'Playwright'

'Playwright'

É o valor esperado.

Temos:

VALOR ENCONTRADO

"Playwright"

VALOR ESPERADO

"Playwright"

Resultado:

"Playwright"

=

"Playwright"

 PASSOU

44. SE O VALOR FOSSE DIFERENTE

Imagine:

VALOR ENCONTRADO

"Playwright Test"

VALOR ESPERADO

"Playwright"

Então:

"Playwright Test"

≠

"Playwright"

 FALHOU

O Playwright informa que a assertion falhou.

 45. }

}

Fecha o bloco da função.

 46.)

)

Fecha a chamada da função.

 47. ;

;

Finaliza a instrução.

48. TRADUÇÃO DO CÓDIGO INTEIRO

Código:

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

test('basic test', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  const name = await page.innerText('.navbar__title');
  expect(name).toBe('Playwright');
});
```

Leia mentalmente assim:

Importe test e expect do Playwright.

↓

Crie um teste chamado "basic test".

↓

Receba uma página do navegador.



Abra o site do Playwright.



Pegue o texto do elemento `.navbar__title`.



Guarde o texto na variável `name`.



Verifique se o texto é exatamente "Playwright".



49. CÓDIGO ORGANIZADO EM AAA

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

test('basic test', async ({ page }) => {

  // ARRANGE
  await page.goto('https://playwright.dev/');

  // ACT
  const name = await page.innerText('.navbar__title');

  // ASSERT
  expect(name).toBe('Playwright');

});
```

Agora enxergue:

ARRANGE
Preparar



ACT
Executar



ASSERT
Verificar



50. DICIONÁRIO RÁPIDO

`import`

Português: importar

Função: trazer recursos de outro módulo.

`test`

Português: teste

Função: criar um teste automatizado.

`expect`

Português: esperar/verificar

Função: iniciar uma assertion.

`from`

Português: de

Função: indicar a origem do recurso importado.

`async`

Português: assíncrono

Função: permitir operações assíncronas.

`page`

Português: página

Função: representar uma página do navegador.

await

Português: aguarde

Função: esperar uma operação assíncrona.

goto

Português: ir para

Função: navegar para uma URL.

const

Português: constante

Função: declarar uma variável que não será reatribuída.

name

Português: nome

Função: nome da variável escolhida pelo programador.

innerText

Português: texto interno

Função: obter o texto de um elemento.

toBe

Português: deve ser

Função: verificar igualdade exata.



51. DICIONÁRIO DOS SÍMBOLOS

{ }

Nome: chaves

Podem representar:

- bloco de código;
- desestruturação;
- objetos.

()

Nome: parênteses

Usados para:

- argumentos;
- chamada de função;
- parâmetros.

' '

Nome: aspas simples

Representam texto.

Exemplo:

`'Playwright'`

=

Nome: atribuição

Coloca um valor em uma variável.

```
const name = 'Playwright';
```

•

Nome: ponto

Acessa métodos ou propriedades.

```
page.goto
```

;

Nome: ponto e vírgula

Finaliza uma instrução.

=>

Nome: arrow function

Cria uma função de seta.

@

Nome: arroba

Faz parte do nome do pacote:

```
@playwright/test
```



52. UM DETALHE IMPORTANTE

Existe uma diferença entre:

```
page.innerText('.navbar__title')
```

e:

```
page.getByRole(...)
```

O primeiro utiliza um CSS selector.

```
'.navbar__title'
```

O segundo utiliza um locator baseado no papel/acessibilidade.

★ 53. ABORDAGEM MAIS MODERNA

Em muitos casos, podemos preferir:

```
const heading = page.getByRole('heading', {  
  name: 'Playwright'  
});
```

```
await expect(heading).toBeVisible();
```

Em vez de:

```
const name = await page.innerText('.navbar__title');  
  
expect(name).toBe('Playwright');
```

Por quê?

Porque podemos localizar o elemento de uma maneira mais próxima de como o usuário o enxerga/interage.

🧠 54. NÃO DECORE — ENTENDA

Não tente decorar o código inteiro.

Memorize o fluxo:

```
IMPORT  
↓  
TEST  
↓  
PAGE  
↓  
AÇÃO
```

↓
RESULTADO
↓
EXPECT
↓
VERIFICAÇÃO



55. MAPA MENTAL

PLAYWRIGHT

```
|
|— import
|
|— test()
|
|— page
|   |— goto()
|   |— click()
|   |— fill()
|   |— screenshot()
|
|— locator
|
|— action
|
|— expect()
|   |— assertion
```



56. AAA — PARA MEMORIZAR

A
↓
ARRANGE
Preparar

A
↓
ACT
Executar

A
↓

ASSERT
Verificar

Frase para memorizar:

Prepare → Execute → Verifique

57. TESTE RÁPIDO

Sem olhar acima, tente responder:

1. O que significa `import`?

Resposta:

Importar recursos de outro módulo.

2. O que é `test`?

Resposta:

Função utilizada para criar um teste.

3. O que é `page`?

Resposta:

Objeto que representa uma página do navegador.

4. O que significa `await`?

Resposta:

Aguarde a operação assíncrona terminar.

5. O que faz `goto()`?

Resposta:

Navega para uma URL.

6. O que é `const`?

Resposta:

Declara uma variável que não será reatribuída.

7. O que faz `innerText()`?

Resposta:

Obtém o texto de um elemento.

8. O que faz `expect()`?

Resposta:

Inicia uma verificação/assertion.

9. O que significa `toBe()`?

Resposta:

Verifica se o valor é exatamente igual ao esperado.

10. O que significa AAA?

Resposta:

Arrange → Act → Assert.



58. O MAIS IMPORTANTE

Se você lembrar somente de uma coisa hoje, lembre disto:

ARRANGE



PREPARAR

ACT



EXECUTAR

ASSERT



VERIFICAR

E este código:

```
await page.goto('https://playwright.dev/');
```

significa:

"Abra a página e aguarde."

Este:

```
const name = await page.innerText('.navbar__title');
```

significa:

"Pegue o texto e guarde em name."

E este:

```
expect(name).toBe('Playwright');
```

significa:

"Verifique se name é exatamente Playwright."



RESUMO PARA REVISAR NO CELULAR

IMPORT

↓

trazer ferramentas

TEST

↓

criar teste

PAGE

↓

página do navegador

ASYNC

↓

operação assíncrona

AWAIT

↓

aguardar

GOTO

↓

navegar

CONST

↓

criar variável

INNER TEXT

↓

pegar texto

EXPECT

↓

verificar

TO BE

↓
deve ser igual

AAA

ARRANGE

↓
preparar

ACT

↓
executar

ASSERT

↓
verificar



FRASE FINAL

Playwright automatiza ações no navegador.

TypeScript é a linguagem usada para escrever o código.

AAA organiza o teste em Preparar → Executar → Verificar.

page controla a página.

await aguarda.

expect verifica.



Próximo conceito recomendado

Depois de dominar este código, o próximo passo deve ser estudar nesta ordem:

1. `async`
2. `await`
3. `page`
4. `locator`
5. `getByRole`
6. `getByText`
7. `getByLabel`
8. `click`
9. `fill`
10. `expect`
11. `toBeVisible`

- 12. `toHaveText`
- 13. AAA
- 14. Page Object Model

Essa sequência cria uma base muito mais sólida do que tentar decorar dezenas de comandos de Playwright de uma vez.