

MANUAL INSTALAÇÃO SELENIUM

QA Aula 5

**Tech Fixe
prof. Hérzio Pinto**

**Carmen Cabral
08/11/2023**

Índice

O QUE É SELENIUM?	3
☞ SELENIUM WEBDRIVER X SELENIUM IDE	3
CARACTERÍSTICAS	3
Suporte a Múltiplos Navegadores	3
Suporte a Diferentes Linguagens de Programação	3
Gravação e Reprodução de Testes	3
Flexibilidade e Personalização	3
Integração com Frameworks de Teste	3
Ampla Comunidade e Suporte	3
Testes de Regressão e Continuidade	3
NPM: Node Package Manager	3
NPM: Instalar Pacotes	3
NPM: Scripts Personalizados	3
BAIXANDO A FERRAMENTA SELENIUM WEBDRIVER	4
Instalação / <i>Installation</i>	4
Driver	4
Baixar Driver geckodriver.exe do Firefox	4
Descompactar, Copiar e Colar na Pasta Windows, dentro da pasta C:	4
COMANDOS SELENIUM	4
CRIAR PASTA 'AulaSeleniumT3'	5
ABRIR VS CODE E ABRIR PASTA AulaSeleniumT3	5
No VS Code, abrir o terminal CTRL Ç	5
NO VS CODE, INSTALAR EXTENSÃO: Code Runner	6
NO VS CODE, INSTALAR EXTENSÃO: Material Icon Theme	7
NO NAVEGADOR, COPIAR O XPATH	8
ASSERTIONS	9
XPATH (XML Path Language)	9
Chai Asserts Library (Biblioteca)	9

O QUE É SELENIUM?

Ferramenta de automação de testes de software para testar aplicativos da web, que simula as ações que um usuário/utilizador faria.

👉 SELENIUM WEBDRIVER X SELENIUM IDE

CARACTERÍSTICAS

Suporte a Múltiplos Navegadores

Navegadores: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari e outros. Garantia de compatibilidade cruzada.

Suporte a Diferentes Linguagens de Programação

Linguagens: Java, Python, C#, JavaScript e outras.

Gravação e Reprodução de Testes

Os testes são gravados como sequências de ações do usuário e depois são reproduzidos automaticamente.

Flexibilidade e Personalização

Possui capacidade de criar verificações e validações personalizadas.

Integração com Frameworks de Teste

Frameworks: JUnit (para Java), TestNG, Mocha, Jasmine e outros para facilitar a execução e relatórios de testes.

Ampla Comunidade e Suporte

Fornece documentação e recursos online, o que torna o **Selenium** popular no mercado. Depois Cypress, Playwright, Nightwatch.

Testes de Regressão e Continuidade

Regressão: Garante que as alterações no código não quebrem funcionalidades existentes.

Continuidade: Garante que as funcionalidades continuem a funcionar após atualizações ou implantações.

NPM: Node Package Manager

Gerenciador de Pacotes do Node para Node.js. É a principal ferramenta para **instalar**, gerenciar e compartilhar bibliotecas e pacotes de código JavaScript. É amplamente utilizado nos testes automáticos.

NPM: Instalar Pacotes

O npm permite instalar pacotes e bibliotecas no JavaScript com o comando: **npm install nome-do-pacote**.

NPM: Scripts Personalizados

Permite definir scripts personalizados no arquivo/ficheiro **package.json**, que podem ser executados com comandos simples para automatizar tarefas de desenvolvimento, como **testes**, compilação e implantação.

BAIXANDO A FERRAMENTA SELENIUM WEBDRIVER

<https://www.npmjs.com/package/selenium-webdriver?activeTab=readme>

Instalação / Installation

Comando/Instrução: `npm install selenium-webdriver`

Será instalado no projeto, não na máquina. Terá que instalar de novo em cada projeto.

Driver

É o responsável por **controlar o navegador** e executar as ações especificadas no código de automação de teste. Será definido um navegador como driver e os testes rodarão nesse **navegador/driver**.

Atua como uma ponte de comunicação entre o código de teste e o navegador da web, traduzindo os comandos/instruções escritos(as) em linguagem de programação em **ações reais no navegador**.

Cada navegador requer seu próprio driver específico para garantir a compatibilidade e a capacidade de automação.

Navegador / Browser	Driver
Chrome	<code>chromedriver(.exe)</code>
Internet Explorer	<code>IEDriverServer.exe</code>
Edge	<code>MicrosoftWebDriver.msi</code>
Firefox	<code>geckodriver(.exe)</code>
Opera	<code>operadriver(.exe)</code>
Safari	<code>safaridriver</code>

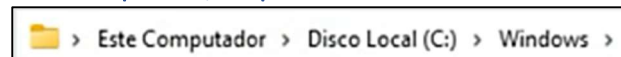
<https://www.npmjs.com/package/selenium-webdriver?activeTab=readme>

Baixar Driver geckodriver.exe do Firefox

<https://github.com/mozilla/geckodriver/releases/>

Para **download** versão para Windows, clique ao lado com **CTRL**: [geckodriver-v0.33.0-win64.zip](#)

Descompactar, Copiar e Colar na Pasta Windows, dentro da pasta C:



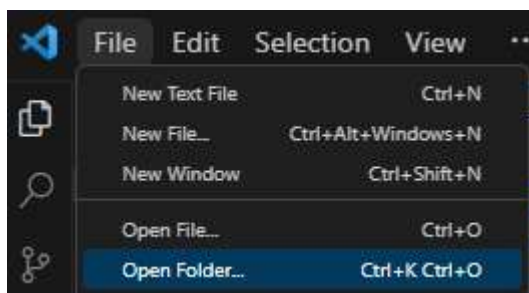
geckodriver.exe

COMANDOS SELENIUM

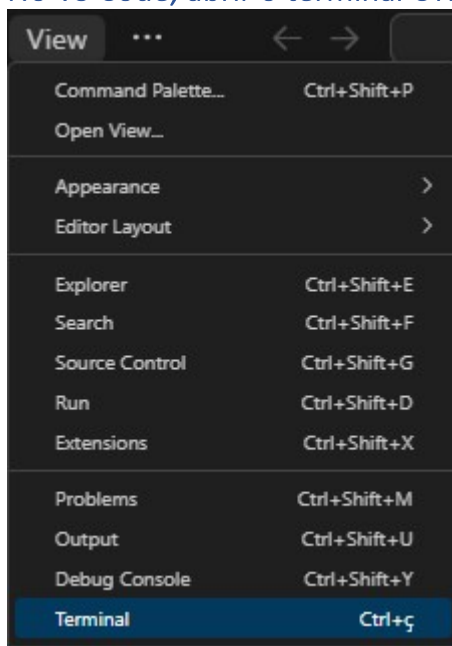
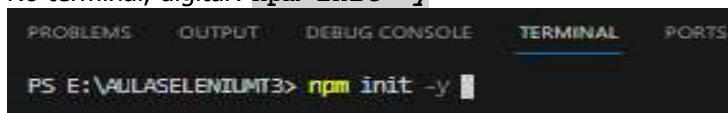
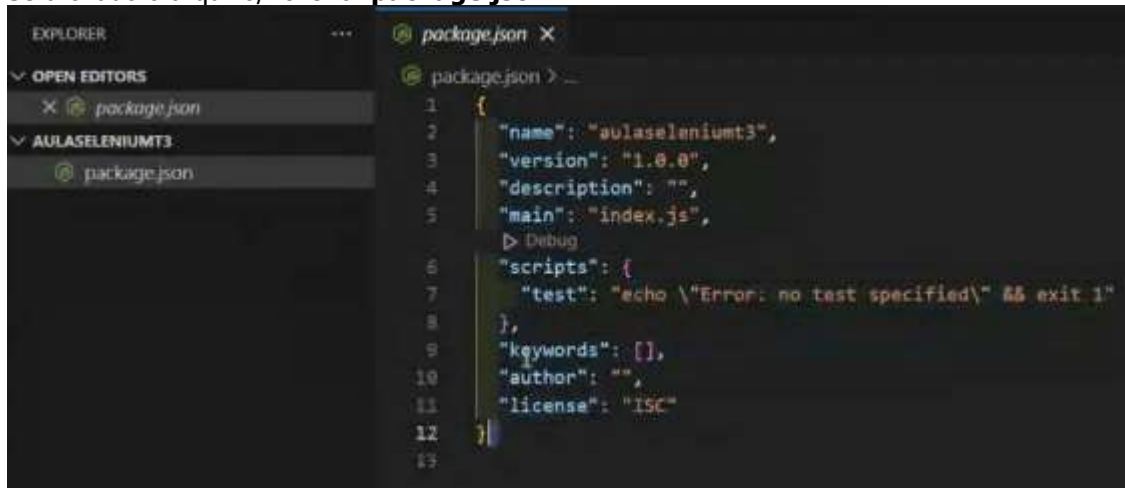
<code>driver.get(url)</code>	Navega para a URL especificada.
<code>driver.findElement(locator)</code>	Localiza um elemento na página com base no seletor especificado.
<code>element.sendKeys(text)</code>	Insere texto ou envia teclas para um elemento (como um campo de entrada).
<code>element.click()</code>	Clica em um elemento, como botão ou link.
<code>element.getText()</code>	Obtém o texto visível de um elemento da página.
<code>element.isDisplayed()</code>	Verifica se um elemento está visível na página.
<code>element.isSelected()</code>	Verifica se um elemento de seleção (como uma caixa de seleção) está marcado.
<code>element.getAttribute(attributeName)</code>	Obtém o valor de um atributo de um elemento.
<code>driver.navigate().back()</code>	Navega de volta para a página anterior.
<code>driver.navigate().forward()</code>	Navega para a página seguinte, após usar <code>navigate().back()</code> .
<code>driver.navigate().refresh()</code>	Atualiza a página atual.

CRIAR PASTA 'AulaSeleniumT3'

ABRIR VS CODE E ABRIR PASTA AulaSeleniumT3



No VS Code, abrir o terminal CTRL Ç

No terminal, digitar: `npm init -y`Será criado o arquivo/ficheiro: **package.json**

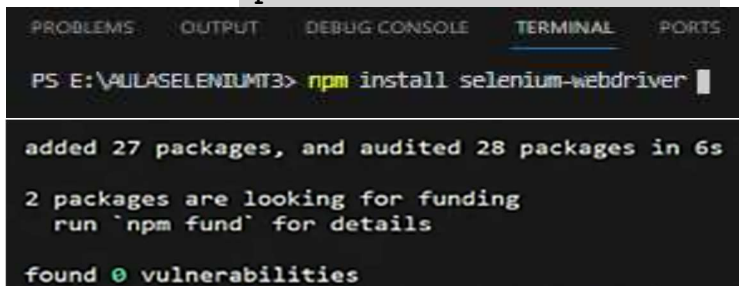
Ir ao site: <https://www.npmjs.com/package/selenium-webdriver?activeTab=readme>

Copiar `npm install selenium-webdriver`

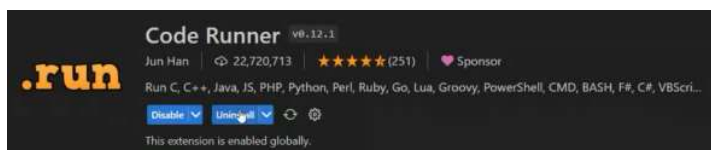
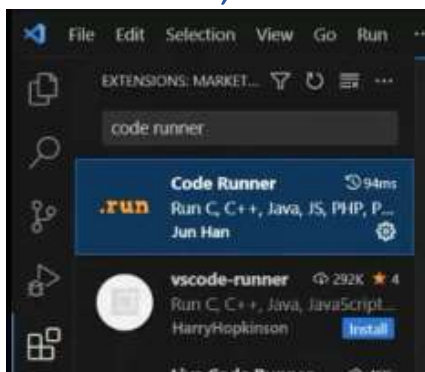
Clear para limpar o terminal.




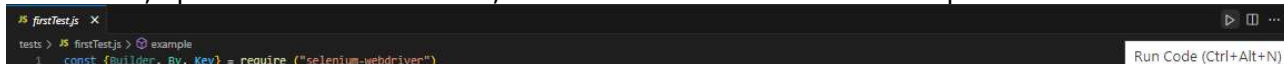
Colar no terminal o `npm install selenium-webdriver`



NO VS CODE, INSTALAR EXTENSÃO: Code Runner

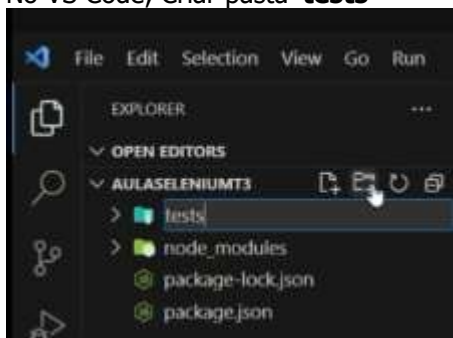


No VS Code, Após instalar **Code Runner**, habilita o botão executar no canto superior direito:  Run Code.

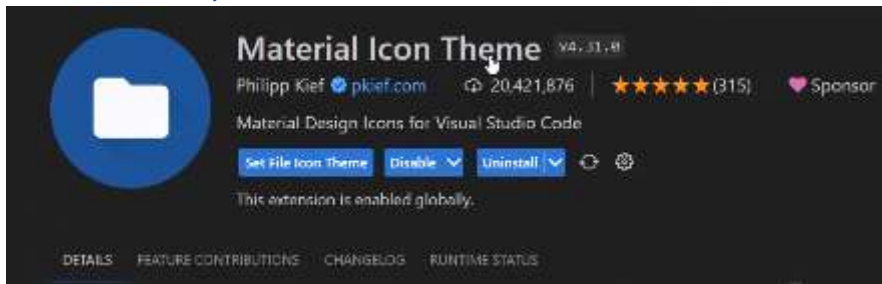


Evita digitar no terminal: `node tests/firstTest.js`, basta clicar no ícone .

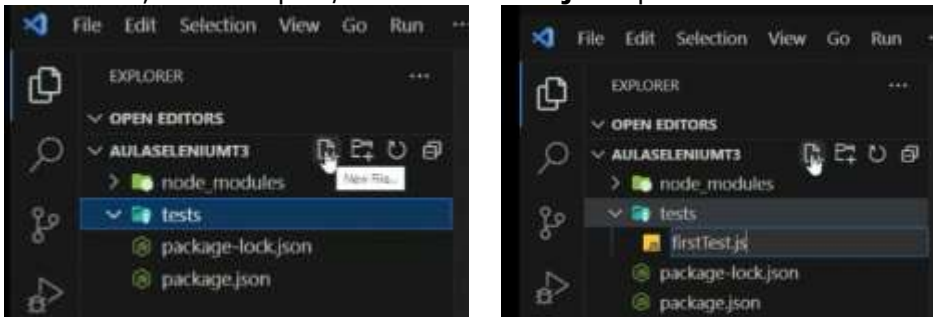
No VS Code, Criar pasta **'tests'**



NO VS CODE, INSTALAR EXTENSÃO: Material Icon Theme



No VS Code, Criar o arquivo/ficheiro **'firstTest.js'** na pasta **'tests'**:



No VS Code, Editar o arquivo **'firstTest.js'**

Função assíncrona – `async()`: o bloco de código que está dentro dela, só vai passar para a próxima linha de código se a primeira tiver sucesso. (Se primeira linha OK, passa para próxima.). Forma de visualizar o erro. Numa função normal, as linhas serão executadas com ou sem erro.

1ª etapa a ser feita antes de fazer um código: **Algoritmo**.

```
const {Builder, By, Key} = require ("selenium-webdriver")
const assert = require ("assert")
var should = require ("chai").should()
async function example() {
  //Declarar o navegador
  let driver = await new webdriver.Builder().forBrowser("firefox").build()

  //Navegar até o site
  await driver.get("https://herziopinto.github.io/lista-de-tarefas/")

  //Dar uma pausa na execução da função
  //await driver.sleep(5000)

  //Adicionar uma tarefa
  await driver.findElement(By.id("inputTask")).sendKeys("Aprender Selenium", Key.RETURN)

  //Assertion para consultar elemento
  let seleniumText = await driver.findElement(By.xpath("/html/body/div/section/ul/li/label")).getText().then(function(value){
    return value
  })

  //Assertion usando o node puro Vanilla JS
  // assert.strictEqual(seleniumText, "Aprender Selenium")

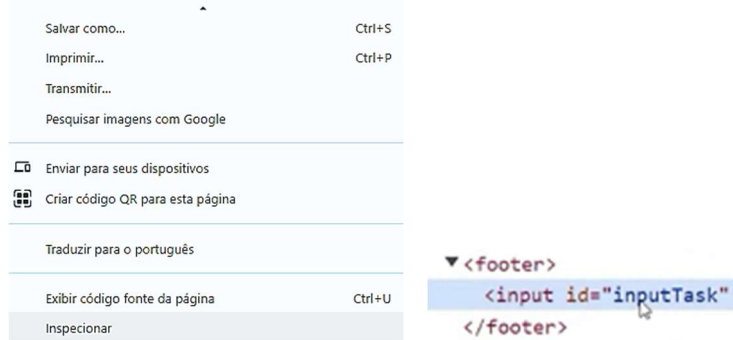
  //Assertion usando chai
  seleniumText.should.equal("Aprender Selenium")

  //Fechar o navegador
  await driver.quit()
}
example()
```

A expressão **`await`** faz a execução de uma função **`async`** pausar.

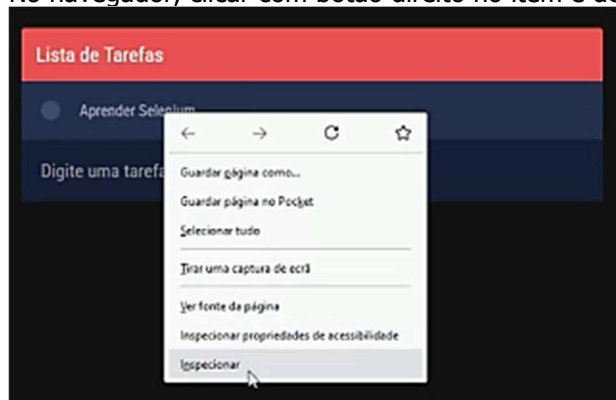
<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Operators/await>

No navegador, **inspecionar** página para copiar o id "inputTask":

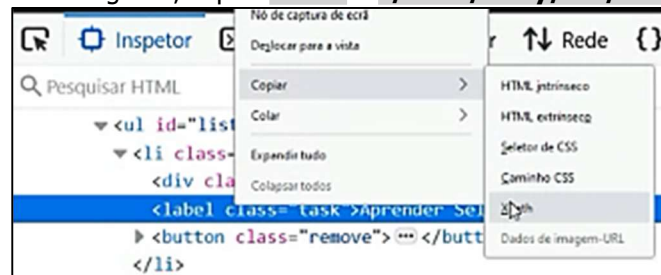


NO NAVEGADOR, COPIAR O XPATH

No navegador, clicar com botão direito no item e depois clicar em "inspecionar".



No navegador, copiar **xPath** – `"/html/body/div/section/ul/li/label"` – depois colar no **VS Code**.



DICA: FAZER PORTFÓLIO

- *Site: Worten.
- *Principal razão da Worten: vender / comprar um produto.
Efetuar uma compra; Adicionar ao carrinho; Preencher o formulário.
- *Colocar no Github.
- *Fazer documentação em PDF e colocar no projeto.

DICA: SITE-CURRÍCULO

- *Com **HTML e CSS; JavaScript**.
- *Colocar link para o **GitHub** e **LinkedIn**.

ASSERTIONS

Afirmações que se referem a verificações ou **validações** feitas dentro de um teste automatizado para garantir que o comportamento do aplicativo corresponda ao esperado.

Determinam se o teste foi bem sucedido ou se falhou conforme as condições definidas no código do teste.

Assertions são **instruções** que **verificarão** se as **condições** específicas são verdadeiras durante a execução do teste.

Há duas formas de fazer **Assertions**:

***Node puro / Vanilla JS / assert.strictEqual(var, "")**

```
const assert = require("assert")

assert.strictEqual(seleniumText, "Aprender Selenium")
```

***Chai / var.should.equal("")**

Precisa digitar **npm install chai** no terminal e digitar a linha abaixo dentro do arquivo **.js**.

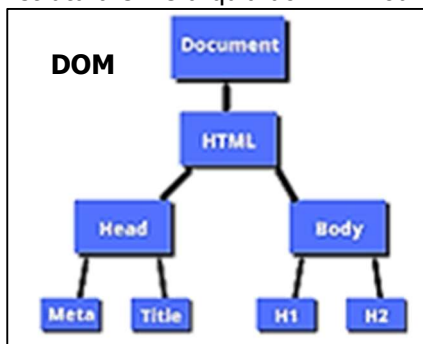
```
var should = require("chai").should()

seleniumText.should.equal("Aprender Selenium")
```

XPATH (XML Path Language)

Linguagem de consulta usada em **Selenium** para **localizar elementos** numa página web conforme **estrutura** e hierarquia no documento **HTML** ou **XML**. Em **Selenium** com **JavaScript**, localização de elementos: método **Xpath()** e seletores **CSS**.

Estrutura e hierarquia do HTML ou XML / **DOM** / **Document Object Model**:



Chai Asserts Library (Biblioteca)

<https://www.chaijs.com/>

Precisa digitar **npm install chai** no terminal.

```
PS E:\VULASELENIUMT3> npm install chai
```

Declarar variável 'should':

```
var should = require("chai").should()
```

Usar a variável 'seleniumText':

```
seleniumText.should.equal("Aprender Selenium")
```

Declarar variável 'seleniumText':

```
//Assertion
let seleniumText = await driver.findElement(By.xpath("/html/body/div/section/ul/li/label")).getText().then(function(value){
  return value
})
```