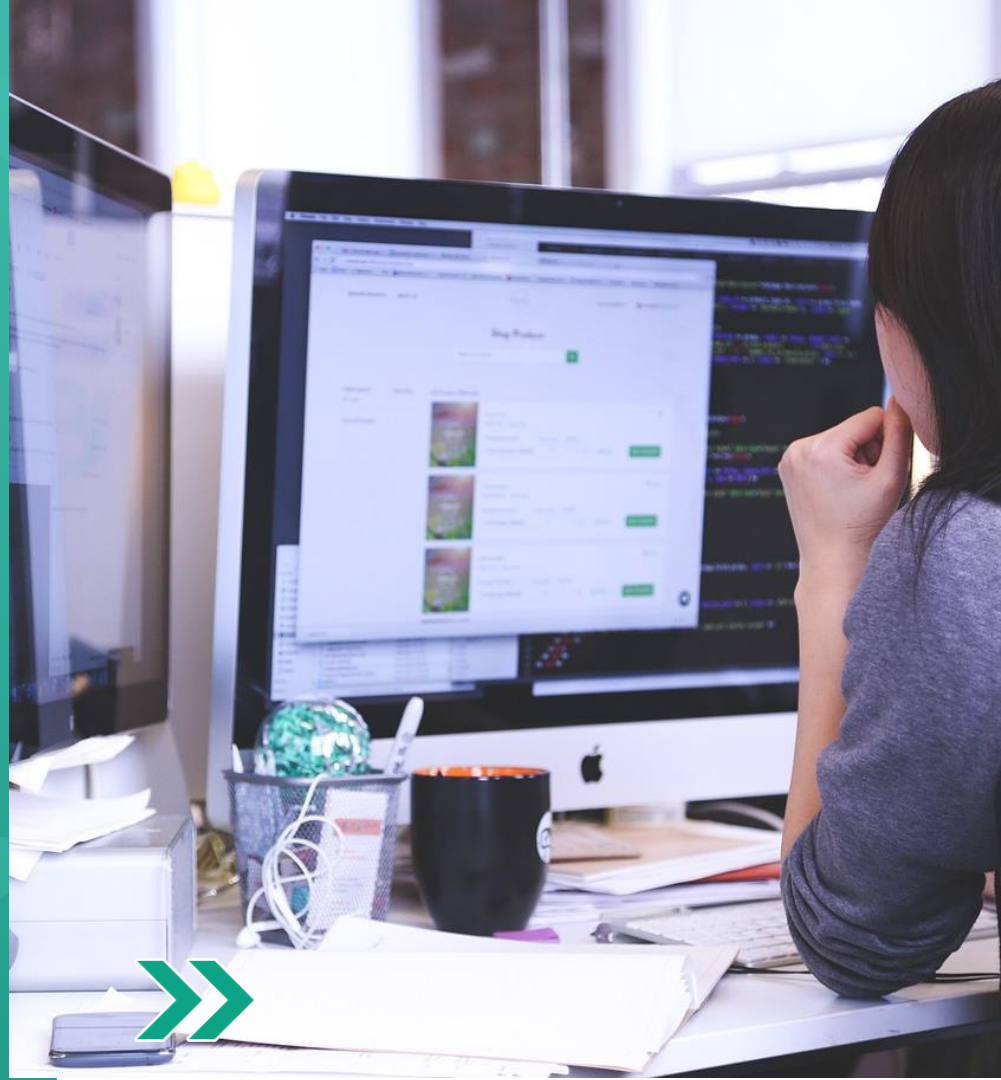
escola britânica de artes criativas & tecnologia

Profissão: Engenheiro Front-End

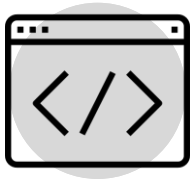
escola britânica de artes criativas & tecnologia

BOAS PRÁTICAS



Testes E2E com Cypress

- **Configure o Cypress**
- **Selecione elementos**
- **Manipule formulários**
- **Explore recursos extras do Cypress**



Confira boas práticas da comunidade de Front-End por assunto relacionado às aulas.

Configure o Cypress

- **Cypress:**
  O Cypress oferece suporte para testar em outros navegadores usando integrações com ferramentas de terceiros. Além disso, ele oferece uma visualização em tempo real dos testes à medida que são executados, permitindo que os desenvolvedores observem todas as etapas dos testes em um navegador em execução.

Selecione elementos

Acompanhe algumas dicas úteis sobre o uso da função it no Cypress.

- **Organize seus testes usando describe:**
  Use o describe para agrupar seus testes de acordo com cenários ou funcionalidades relacionadas. Isso ajuda a manter a estrutura do código organizada e facilita a localização de testes específicos.

- **Escreva descrições claras e concisas:**
  Ao definir o caso de teste usando it, certifique-se de usar descrições claras e concisas que expressem o que está sendo testado. Isso facilita a compreensão do propósito do teste.

- **Crie testes pequenos e focados:**
  É recomendável criar testes pequenos e focados em apenas um aspecto ou comportamento específico da aplicação. Testes menores são mais fáceis de manter e depurar.

Selecione elementos

Acompanhe algumas dicas úteis sobre o uso da função it no Cypress.

- **Evite testes dependentes de estado:**
  Procure evitar criar testes que dependam do estado de outros testes. Cada teste deve ser independente e não deve confiar em resultados anteriores para funcionar corretamente.

- **Verifique o console durante o teste:**
  Durante o desenvolvimento dos testes, é possível usar cy.log() para imprimir informações no console do Cypress. Isso pode ajudar a entender o que está acontecendo durante a execução do teste.

Selecione elementos

Caracteres:
É importante evitar o uso de caracteres que se tornaram classes ou IDs, pois isso pode causar problemas de seleção e estilização dos elementos. Quando um caractere é usado como classe ou ID, ele é interpretado como um seletor CSS, e alguns caracteres têm significados especiais em CSS que podem causar comportamentos inesperados. Para isso, evite o uso de caracteres especiais como "#" ou "." como parte dos nomes de classes e IDs e use hífens para separar palavras: É comum usar hífens para separar palavras em nomes de classes e IDs, o que torna o código mais legível e seguindo as boas práticas.

Manipule formulários

- **Funções do suíte de teste:**
  Você pode usar os blocos beforeEach e afterEach para executar código antes e depois de cada caso de teste. Isso é útil para configurar o estado inicial do teste ou fazer limpezas após cada execução.

Explore recursos extras do Cypress

Acompanhe algumas dicas sobre o uso do **Mochawesome** para criar relatórios personalizados de testes automatizados.

- **Configure o Mocha para usar o Mochawesome:**
  Você precisará configurar o Mocha para gerar relatórios no formato Mochawesome. Isso pode ser feito adicionando a opção --reporter mochawesome ao executar os testes via linha de comando ou configurando-o no arquivo mocha.opts ou mocha.config.js.

- **Personalize a saída do relatório:**
  O Mochawesome permite personalizar a aparência e o conteúdo do relatório gerado. Você pode personalizar as cores, o título do relatório, adicionar informações extras e até mesmo criar temas personalizados. Consulte a documentação do Mochawesome para obter detalhes sobre como personalizar a saída do relatório.

- **Adicione screenshots para falhas:**
  Para obter relatórios mais informativos, você pode adicionar capturas de tela dos testes que falharam. Isso pode ser especialmente útil para identificar problemas e depurar falhas com mais facilidade. Existem bibliotecas e plugins disponíveis que podem ajudá-lo a tirar e adicionar automaticamente screenshots aos relatórios de teste.

Explore recursos extras do Cypress

Acompanhe algumas dicas sobre o uso do **Mochawesome** para criar relatórios personalizados de testes automatizados.

- **Grave vídeos para casos de teste complexos:**
  Em alguns casos, especialmente para testes mais complexos e de longa duração, pode ser útil gravar vídeos da execução dos testes. Esses vídeos podem ser úteis para revisar a execução dos testes, identificar problemas de usabilidade ou compartilhar resultados com outras equipes.

- **Integre com outras ferramentas:**
  O Mochawesome pode ser facilmente integrado com outras ferramentas de automação de testes, como o Cypress, para melhorar a experiência de teste e relatório. Verifique se há plugins ou extensões disponíveis para a ferramenta que você está usando para facilitar a geração de relatórios Mochawesome.

Explore recursos extras do Cypress

Acompanhe algumas dicas sobre Cypress – **queries**.

- **Após a seleção de um elemento podemos encadear outras funções, como as ações e outra seleções:**
  - **First**: utilizado para selecionar o primeiro item de uma lista, por exemplo: cy.get("li").first()
  - **Find**: utilizado para fazer uma busca dentro do elemento selecionado, por exemplo: cy.get("header").find(".logo")
  - **Last**: utilizado para selecionar o último item de uma lista, por exemplo: cy.get("li").last()
  - **Parent**: utilizado para selecionar o elemento pai, por exemplo: cy.get(".logo").parent() // header
  - **Title**: utilizado para recuperar o conteúdo da tag title.

Explore recursos extras do Cypress

Acompanhe algumas dicas sobre Cypress – **ações**.

Com as ações podemos interagir com os elementos da tela, clicando, digitando, marcando checkboxes e selecionando itens em um select.
Utilizamos as ações encadeando com os seletores.

- **Click**: utilizado para clicar em elementos, por exemplo: cy.get("#btn-enviar").click(),
- **Type**: utilizado para digitar em um input ou textarea, por exemplo: cy.get("#github-user").type("ogiansouza")
- **Select**: utilizado para selecionar uma opção dentro de uma tag select, por exemplo: cy.get("#escolaridade").select("ensino-superior") ou cy.get("#escolaridade").select("Ensino Superior"), Podemos selecionar com base no valor ou no texto da opção.
- **Check**: utilizado para selecionar um input radio ou checkbox, por exemplo: cy.get("#termoscondicoes"). check(),

Explore recursos extras do Cypress

Acompanhe algumas dicas sobre Cypress – **asserções**.

- As asserções com o Cypress são escritas a partir da função should, que deve ser encadeada a um seletor, a partir dela criamos as expectativas do teste, por exemplo: cy.get("#saudacao").should("have.text", "Olá")

Para conhecer a lista das principais asserções, acesse o site https://docs.cypress.io/guides/references/assertions.

escola britânica de artes criativas & tecnologia

Bons estudos!