

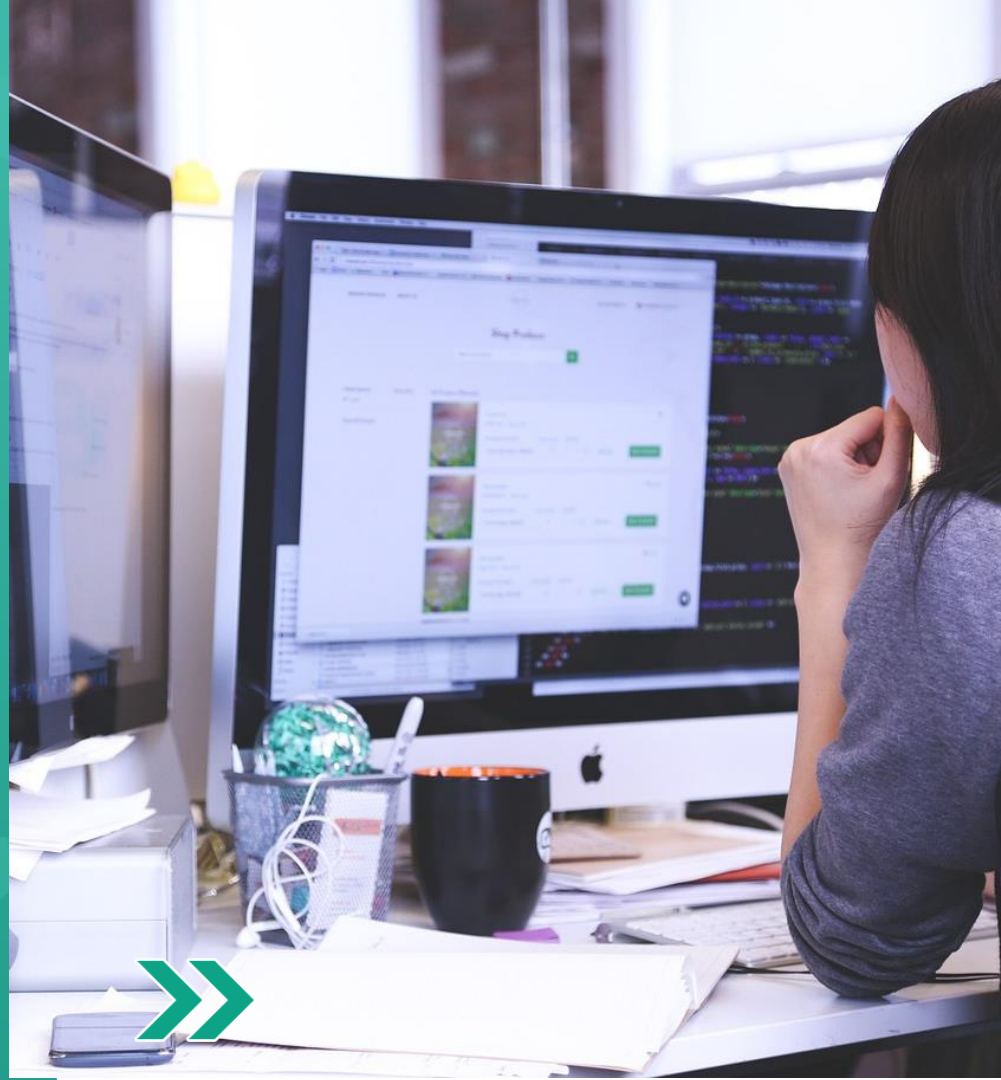


escola  
britânica de  
artes criativas  
& tecnologia

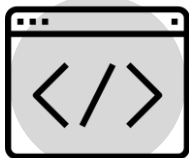
# Profissão: Engenheiro Front-End



# BOAS PRÁTICAS



# Fundamentos do JavaScript



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

- **Conheça o JavaScript**
- **Crie variáveis e constantes**
- **Explore os tipos de dados**
- **Crie repetições (laços)**
- **Use condicionais**



# Conheça o JavaScript

- **As principais funcionalidades das Dev Tools incluem:**
  - **Inspecção de elementos:** permite visualizar e modificar o HTML e o CSS de uma página, verificando como os elementos estão sendo renderizados e aplicando alterações em tempo real.
  - **Console:** oferece um console de JavaScript para visualizar e depurar erros, executar comandos e testar trechos de código.
  - **Network (Rede):** mostra as solicitações HTTP realizadas pelo navegador, permitindo analisar os recursos carregados, tempos de resposta, cabeçalhos e dados enviados e recebidos.



# Conheça o JavaScript

- **Sources (Fontes):** oferece uma visão detalhada dos arquivos JavaScript, permitindo definir pontos de interrupção para depurar o código passo a passo, visualizar pilhas de chamadas e monitorar variáveis.
- **Performance (Desempenho):** permite analisar o desempenho de uma página, identificando gargalos e otimizando o tempo de carregamento.
- **Application (Aplicação):** exibe informações sobre armazenamento em cache, cookies, bancos de dados locais e outras informações relacionadas à aplicação web.
- **Emulação de dispositivo:** possibilita simular dispositivos móveis e verificar como um *site* ou aplicativo é exibido em diferentes tamanhos de tela e orientações.



# Conheça o JavaScript

- Diferença entre `console.warn` e `console.error`:**  
 O tipo de mensagem que cada um deles representa e a maneira como são estilizados no console é a diferença entre ambos. O método `console.warn` é usado para exibir mensagens de **aviso** (warnings) no console. Já o `console.error` é usado para exibir mensagens de erro no console.
- Alert:**  
 A função `alert()` ela tem a limitação de que a caixa de diálogo é simples e não pode ser personalizada em termos de aparência ou estilo e ela bloqueia a execução do código, o que significa que o restante do código JavaScript não será executado até que o usuário interaja com a caixa de diálogo. Como alternativa, é comum usar outras abordagens, como exibir informações na página HTML, manipular o conteúdo de elementos HTML ou utilizar bibliotecas/frameworks JavaScript mais avançados para criar caixas de diálogo personalizadas.



# Crie variáveis e constantes



- **Camel Case (vantagens):**
  - **Legibilidade:** O camel case facilita a leitura e compreensão dos identificadores, pois a primeira letra minúscula e as letras maiúsculas no início de cada nova palavra ajudam a distinguir claramente as palavras individuais.
  - **Convenção padrão:** O camel case é amplamente utilizado na comunidade de desenvolvimento JavaScript, sendo considerado uma convenção padrão. Ao seguir essa convenção, você torna seu código mais consistente e mais fácil de entender para outros desenvolvedores.
- **Camel Case (variações):**
  - **Pascal case:** Semelhante ao camel case, mas a primeira letra de cada palavra é maiúscula. Geralmente usado para nomear classes e construtores em JavaScript.
  - **Snake case:** As palavras são separadas por um caractere de sublinhado (`_`). Exemplo: `minha_variavel`.
  - **Kebab case:** As palavras são separadas por um hífen (`-`). Exemplo: `minha-variavel`.

# Explore os tipos de dados

- **Typeof null:**  
Atente-se para o uso do typeof null, pois ele retorna "object", o que é considerado um erro histórico da linguagem JavaScript.





# Crie repetições (Laços)



## **Loops:**

Os *loops* podem ser poderosos, mas também podem levar a problemas de desempenho se não forem usados corretamente. Certifique-se de que seu loop tenha uma condição de parada adequada e não entre em um loop infinito acidentalmente.



# Use condicionais



## Use operadores de comparação apropriados:

Ao escrever condições, use os operadores de comparação apropriados, como `===` (igual estrito), `!==` (diferente estrito), `>` (maior que), `<` (menor que), `>=` (maior ou igual a) e `<=` (menor ou igual a). Isso garante que as comparações sejam feitas corretamente e evita erros comuns.



# Bons estudos!

