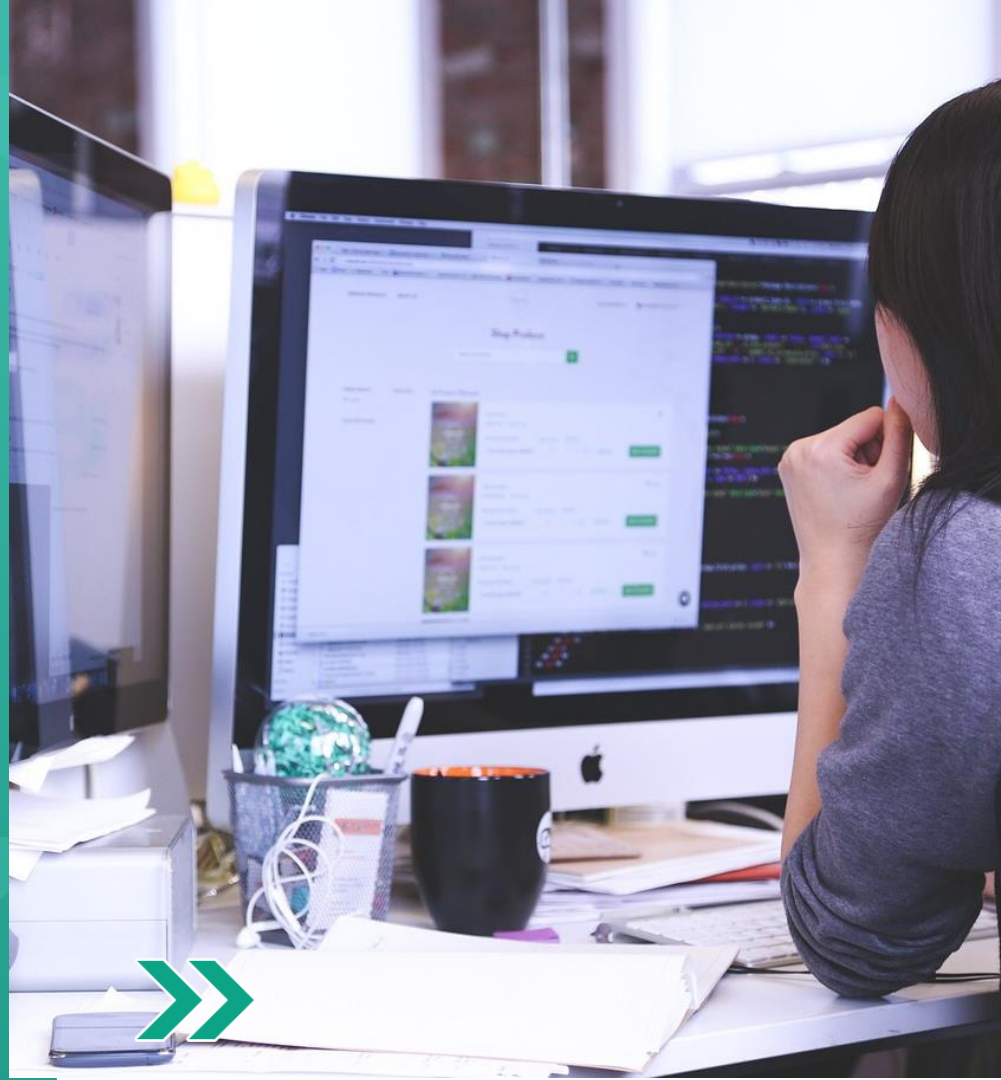
escola britânica de artes criativas & tecnologia

Profissão: Engenheiro Front-End

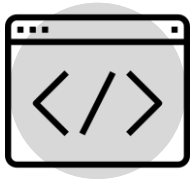
escola britânica de artes criativas & tecnologia

BOAS PRÁTICAS



Ajax e exceções

- Explore a comunicação entre Front-End e Back-End
- Faça requisições AJAX com XMLHttpRequest
- Faça requisições AJAX com fetch API
- Trate exceções
- Lance exceções



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

Explore a comunicação entre Front-End e Back-End

Os cookies podem ser configurados com diferentes atributos, como tempo de expiração, escopo de domínio e segurança. Além disso, os usuários têm controle sobre os cookies e podem ajustar as configurações de privacidade do navegador para aceitar, rejeitar ou remover cookies. Os cookies têm várias finalidades, acompanhe as principais.

- **Manter o estado da sessão:**
  Os cookies são frequentemente usados para rastrear e manter informações sobre a sessão do usuário. Isso permite que o site reconheça o usuário e mantenha as informações relacionadas à sessão, como o carrinho de compras em um site de comércio eletrônico.

- **Personalização e armazenamento de preferências:**
  Os cookies podem ser usados para armazenar as preferências do usuário, como idioma preferido, tema do site ou outras configurações personalizadas. Isso permite que o site se adapte às preferências do usuário em visitas futuras.

Explore a comunicação entre Front-End e Back-End

Os cookies podem ser configurados com diferentes atributos, como tempo de expiração, escopo de domínio e segurança. Além disso, os usuários têm controle sobre os cookies e podem ajustar as configurações de privacidade do navegador para aceitar, rejeitar ou remover cookies. Os cookies têm várias finalidades, acompanhe as principais.

- **Rastreamento e análise:**
  Os cookies também são usados para rastrear e coletar informações sobre a atividade do usuário no site. Isso pode incluir dados como páginas visitadas, cliques, tempo gasto no site e outras métricas. Essas informações podem ser usadas para análise, segmentação de anúncios ou personalização do conteúdo.

- **Autenticação e segurança:**
  Os cookies podem ser usados para autenticar usuários e garantir a segurança das interações entre o usuário e o site. Por exemplo, um cookie de autenticação pode ser usado para permitir que o usuário acesse áreas restritas do site sem precisar fazer login repetidamente.

Explore a comunicação entre Front-End e Back-End

Acompanhe agora algumas dicas sobre os códigos de status HTTP.

- **Conheça os códigos de status comuns:** Familiarize-se com os códigos de status mais comuns, como 200 (OK) para uma solicitação bem-sucedida, 404 (Not Found) para um recurso não encontrado e 500 (Internal Server Error) para erros internos do servidor. Entender o significado desses códigos facilita a depuração e solução de problemas.

- **Utilize os códigos de status apropriados:** Escolha os códigos de status que mais se adequam à situação específica. Use 201 (Created) ao criar um novo recurso com sucesso, 204 (No Content) quando uma solicitação não retorna conteúdo ou 403 (Forbidden) quando o acesso é negado devido a permissões insuficientes.

Explore a comunicação entre Front-End e Back-End

Acompanhe agora algumas dicas sobre os códigos de status HTTP.

- **Forneça mensagens descritivas:**
  Junto com o código de status, forneça mensagens descritivas na resposta HTTP para que os clientes possam entender o resultado da solicitação. Isso pode ajudar os desenvolvedores a entenderem a causa de um erro ou ação necessária.

- **Trate os erros adequadamente:**
  Ao lidar com erros no servidor, forneça códigos de status apropriados para indicar o tipo de erro ocorrido. Isso pode ajudar os clientes a tomar ações apropriadas com base nos códigos de status recebidos.

- **Evite o uso indevido de códigos de status:**
  Evite usar códigos de status de forma inadequada ou imprecisa. Por exemplo, não use o código 200 (OK) quando ocorrer um erro, pois isso pode confundir os clientes e dificultar a depuração.

Explore a comunicação entre Front-End e Back-End

Acompanhe agora algumas dicas sobre os códigos de status HTTP.

escola britânica de artes criativas & tecnologia

- **Considere os redirecionamentos:**
  Ao usar redirecionamentos, como os códigos de status 301 (Moved Permanently) ou 302 (Found), certifique-se de fornecer a nova localização do recurso no cabeçalho "Location" da resposta. Isso ajudará os clientes a seguir o redirecionamento corretamente.

- **Documente os códigos de status:**
  Mantenha uma documentação clara e atualizada sobre os códigos de status utilizados em sua API ou aplicação. Isso ajudará os desenvolvedores a entenderem como interpretar e lidar com os diferentes códigos de status retornados pelo seu serviço.

Faça requisições AJAX com XMLHttpRequest

- **Requisições AJAX usando XMLHttpRequest:** Essas requisições são uma forma tradicional de enviar e receber dados assincronamente entre um navegador e um servidor web. O objeto XMLHttpRequest é fornecido pela API do navegador e permite que os desenvolvedores realizem requisições HTTP assíncronas. O objeto XMLHttpRequest fornece vários métodos e propriedades para gerenciar requisições, definir cabeçalhos personalizados, enviar dados no corpo da requisição e muito mais. Você pode personalizar a requisição conforme necessário para atender às suas necessidades específicas.

Faça requisições AJAX com fetch API

Ao usar a Fetch API, é importante considerar a segurança, como tratar a proteção contra ataques de CSRF (Cross-Site Request Forgery) e XSS (Cross-Site Scripting), e adotar boas práticas de desenvolvimento e segurança. Acompanhe algumas dicas para usar a Fetch API de forma eficaz.

- **Conheça o básico da Fetch API:**
  Familiarize-se com a sintaxe e os principais conceitos da Fetch API, como a função fetch(), o uso de Promises, os métodos HTTP suportados e a manipulação de respostas. Entenda como lidar com os objetos Response e Request retornados pela Fetch API.

- **Lide com os diferentes tipos de resposta:**
  A Fetch API pode lidar com vários tipos de resposta, como JSON, texto, Blob, ArrayBuffer e outros. Certifique-se de usar os métodos apropriados, como .json(), .text(), .blob(), para extrair e manipular os dados retornados na resposta.

- **Trate erros adequadamente:**
  Use o método .catch() para capturar e tratar erros durante a requisição. Verifique se a resposta da requisição está no intervalo de códigos de status esperado (por exemplo, 200-299) usando o método .ok. Lançar um erro personalizado pode ajudar a lidar com casos de falha na requisição.

Faça requisições AJAX com fetch API

Ao usar a Fetch API, é importante considerar a segurança, como tratar a proteção contra ataques de CSRF (Cross-Site Request Forgery) e XSS (Cross-Site Scripting), e adotar boas práticas de desenvolvimento e segurança. Acompanhe algumas dicas para usar a Fetch API de forma eficaz.

- **Configure as opções da requisição:**
  A Fetch API permite que você configure várias opções da requisição, como headers, método, corpo da requisição e autenticação. Passe um objeto de configuração como segundo argumento para a função fetch() e configure as opções necessárias para sua requisição.

- **Gerencie o envio e cancelamento de requisições:**
  Se você precisar enviar várias requisições simultaneamente ou se precisar cancelar uma requisição em andamento, é importante entender como gerenciar e controlar as requisições usando Promises e cancelamento adequado.

- **Utilize funções auxiliares:**
  A Fetch API fornece funções auxiliares, como Headers, Request e Response, que podem ser úteis para construir e manipular cabeçalhos, criar requisições personalizadas e manipular respostas de maneira mais avançada.

Faça requisições AJAX com fetch API

Ao usar a Fetch API, é importante considerar a segurança, como tratar a proteção contra ataques de CSRF (Cross-Site Request Forgery) e XSS (Cross-Site Scripting), e adotar boas práticas de desenvolvimento e segurança. Acompanhe algumas dicas para usar a Fetch API de forma eficaz.

- **Considere o suporte do navegador:**
  Antes de usar a Fetch API, verifique a compatibilidade com os navegadores que você pretende suportar. Embora a Fetch API seja amplamente suportada em navegadores modernos, pode ser necessário usar um *polyfill* ou uma biblioteca alternativa para garantir a compatibilidade em navegadores mais antigos.

- **Explore bibliotecas e frameworks:**
  Embora a Fetch API seja poderosa por si só, considerar o uso de bibliotecas ou frameworks como Axios ou outras que simplifiquem ainda mais o uso de requisições assíncronas pode ser uma opção interessante, especialmente se você tiver requisitos específicos ou necessidades avançadas.

Trate exceções

Tratar as exceções de forma adequada é essencial para desenvolver um código confiável e resiliente. Acompanhe algumas dicas para tratar exceções de maneira eficaz.

- **Identifique os pontos críticos:**
  Identifique os trechos de código que podem gerar exceções e concentre-se em tratá-las em torno desses pontos críticos. Isso inclui operações de leitura/gravação de arquivos, chamadas de rede, manipulação de dados externos e qualquer outro código que possa apresentar comportamento imprevisível.

- **Forneça mensagens descritivas de erro:**
  Ao capturar exceções, forneça mensagens de erro descritivas que informem sobre o que deu errado. Isso ajuda na depuração e no entendimento dos erros durante a fase de desenvolvimento e facilita a identificação e correção de problemas.

- **Registre ou notifique os erros:**
  Além de tratar as exceções, é útil registrar ou notificar os erros para fins de monitoramento e solução de problemas. Use técnicas como registrar os erros em logs ou enviar notificações para uma equipe responsável para que possam tomar medidas corretivas.

Trate exceções

Tratar as exceções de forma adequada é essencial para desenvolver um código confiável e resiliente. Acompanhe algumas dicas para tratar exceções de maneira eficaz.

- **Limpe recursos adequadamente:**
  Se você estiver usando recursos externos, como conexões de banco de dados ou manipulação de arquivos, certifique-se de limpar ou fechar esses recursos adequadamente no bloco finally ou usando a construção try...finally. Isso garantirá que os recursos sejam liberados, mesmo em caso de exceção.

- **Considere o uso de bibliotecas de tratamento de erros:**
  Considere o uso de bibliotecas ou frameworks que facilitem o tratamento de erros, como bibliotecas de logging ou frameworks específicos de tratamento de erros, dependendo da linguagem ou ambiente em que você está desenvolvendo.

Lance exceções

O uso adequado do throw e o tratamento de exceções são considerações importantes para escrever um código JavaScript robusto e resiliente. Ao lançar exceções de forma adequada e capturá-las corretamente, você melhora a legibilidade, a manutenção e a confiabilidade do seu código.
Acompanhe algumas dicas sobre o uso adequado da palavra reservada throw em JavaScript.

- **Lançar exceções apropriadas:**
  Ao usar throw, certifique-se de lançar exceções apropriadas para indicar o tipo de erro ou problema que ocorreu. Use objetos de erro personalizados, como Error, ou subclasse Error, para fornecer informações detalhadas sobre o erro.

- **Forneça mensagens descritivas:**
  Ao lançar uma exceção, inclua uma mensagem de erro descritiva que ajude a identificar o problema. Isso é útil para depuração e para fornecer informações claras sobre o erro para os desenvolvedores que estão tratando a exceção.

- **Evite lançar exceções desnecessárias:**
  Embora o lançamento de exceções seja útil para sinalizar erros, evite lançar exceções desnecessárias. Use throw somente quando ocorrerem erros ou condições excepcionais que justifiquem uma interrupção do fluxo normal do programa.

Lance exceções

O uso adequado do throw e o tratamento de exceções são considerações importantes para escrever um código JavaScript robusto e resiliente. Ao lançar exceções de forma adequada e capturá-las corretamente, você melhora a legibilidade, a manutenção e a confiabilidade do seu código.
Acompanhe algumas dicas sobre o uso adequado da palavra reservada throw em JavaScript.

- **Documente as exceções lançadas:**
Ao escrever funções ou blocos de código que lançam exceções, documente adequadamente quais exceções podem ser lançadas. Isso ajudará outros desenvolvedores a entenderem o comportamento esperado e a lidar com as exceções adequadamente.

- **Mantenha a consistência:**
Mantenha a consistência no uso da palavra reservada throw em seu código. Use um estilo de lançamento de exceção consistente em todo o código para facilitar a leitura, a manutenção e o tratamento adequado de exceções.

escola britânica de artes criativas & tecnologia

Bons estudos!