cap.05

1.O que é o DOM e como ele é manipulado em TypeScript?

           R: DOM é  uma interface de programação para documentos HTML, ele representa a estrutura de uma página da web, o DOM com TypeScript tem a vantagem de adicionar segurança e clareza ao código por meio de tipagem, isso torna o processo de desenvolvimento mais confiável e produtivo.

02.Qual a diferença entre a manipulação de DOM em JavaScript e TypeScript?

           R:está na tipagem estática de TypeScript, que oferece mais segurança e previsibilidade. Já em  JavaScript, a tipagem é dinâmica, o que pode levar a erros não detectados até o tempo de execução, como acessar propriedades de elementos nulos ou indefinidos. Já em TypeScript, você especifica os tipos dos elementos, o que permite que o compilador detecte erros antes da execução.

03.Como você tiparia um evento de clique no TypeScript?

          R:Para tipar um evento de clique em TypeScript, pode utilizar o tipo específico MouseEvent, que é o tipo padrão para eventos relacionados à cliques. Isso fornece autocompletar e validação de tipos para as propriedades do evento, como target, clientX, clientY.

04.Qual o benefício de usar as para afirmar o tipo de um elemento do DOM?

          R: Para afirmar tipos em elementos do DOM melhora a segurança do código, evitando erros de execução e facilitando a codificação com a sugestões de completar e acesso a propriedades específicas de cada tipo de elemento DOM, com isso torna o desenvolvimento em TypeScript mais eficiente.

05.Como a tipagem estática pode ajudar na manipulação de eventos?

            R: A tipagem estática em TypeScript ajuda na manipulação assim fornecendo verificação de tipos em tempo de compilação, isso garante que as propriedades e métodos corretos do evento sejam usados, diminuindo os erros de execução. Além disso, oferece validação assegurando que você manipule os eventos de maneira correta, como MouseEvent em cliques ou KeyboardEvent para eventos de teclado. Com a tipagem estática, você tem um código mais seguro.

     6.O que é um MouseEvent e como ele é usado?

        R: O MouseEvent permite capturar e manipular interações do usuário com o mouse, fornecendo informações detalhadas sobre a posição do cursor, botões pressionados e teclas modificadoras, ele é usado para criar interações dinâmicas em páginas web.

07.  Como você pode integrar jQuery em um projeto TypeScript?

       R:Para integrar jQuery em um projeto TypeScript, é preciso Instalar jQuery e os tipos, Importar o jQuery no código TypeScript e utilizar jQuery normalmente.

08.Qual a vantagem de utilizar bibliotecas JavaScript populares com TypeScript?

  R: A principal vantagem de utilizar bibliotecas com TypeScript é obter tipagem estática e uma melhor experiência de desenvolvimento com a detecção de erros antecipada, completar inteligente, e maior segurança ao código.

09.Quais são as melhores práticas para manipular o DOM com TypeScript?

     R: Para as melhores práticas são, usar tipos específicos para elementos do DOM, como HTMLInputElement, em vez de HTMLElement, Tratar possíveis retornos null, verificando se os elementos existem antes de manipular, preferir  addEventListener para eventos use funções reutilizáveis para evitar duplicação de código, utilize “as” para afirmar tipos, separe a lógica do DOM da lógica de negócios para melhor organização.

Essas práticas ajudam a melhorar a segurança e eficiência do código.

10.Como as interfaces de TypeScript podem ser usadas para melhorar o código em bibliotecas como React?

R: Elas podem ser usadas para melhorar o código em bibliotecas como React como Tipagem de “props” e estado evitando erros, Segurança em eventos: Fornece tipos corretos como cliques e alterações, componentes genéricos: Permite criar componentes reutilizáveis e flexíveis para diferentes tipos de dados, Composição de interfaces: Facilita a criação de props, Melhora na refatoração: Simplifica mudanças e garante que o código se mantenha consistente, essas práticas tornam o código mais seguro, fácil de manter.

Cap.06

1. Qual a importância da comunicação HTTP para aplicações web modernas?

            R :A comunicação HTTP é essencial para aplicações web, pois:Conecta ao cliente e servidor, Interoperabilidade: APIs Segurança (HTTPS), Manutenção de estado, Aplicações dinâmicas, Integração externa.

* Como você vê a evolução do HTTP na construção de APIs e serviços? R: podemos esperar maior convergência de protocolos como HTTP com APIs mais interativas e personalizável, além de maior automação no provisionamento de segurança e escalabilidade, o HTTP continua evoluindo para suportar a crescente demanda por APIs rápidas, seguras e escaláveis, adaptando-se às necessidades de serviços distribuídos e aplicações modernas.

1. Em quais situações você escolheria usar GET em vez de POST, e por quê?

          R: Usar GET quando escolher para  obter dados sem alterar o estado do servidor, quando a operação é mais apropriada, o URL pode ser compartilhado.

Usar Post para enviar ou modificar dados, quando você precisa alterar o estado no servidor, lidar com dados sensíveis ou complexos.

* Como as semânticas de cada método HTTP impactam o design da sua API?

                    R: o uso correto das semânticas dos métodos HTTP não só melhora a experiência de desenvolvedores e usuários, mas também garante a aceitação com as melhores práticas de desenvolvimento, segurança, e eficiência..

1. Como os parâmetros de rota e query params podem melhorar a flexibilidade de uma API?

            R: Quando usados adequadamente, aumentam a flexibilidade, permitindo que ela atenda a uma variedade de situações de uso. Parâmetros de rota oferecem clareza e estrutura.

.Você consegue pensar em um exemplo prático onde ambos seriam utilizados?

      R: Exemplo que poderiam ser utilizados é a combinação de parâmetros de rota e query params que  permite que a API forneça acesso a recursos específicos e ao mesmo tempo em que oferece flexibilidade para customizar a resposta de acordo com filtros ou critérios opcionais.

1. Qual é a função dos headers em uma requisição HTTP?

          R: A função dos headers tem um um papel essencial na comunicação entre cliente e servidor, fornecendo informações adicionais que orientam como os dados devem ser manipulados, protegidos e apresentados, eles aumentam a flexibilidade e a eficiência das interações na web, ajudando a adaptar as respostas e requisições às necessidades e características específicas de cada situação.

* Como a falta de headers apropriados poderia afetar a comunicação entre cliente e servidor?

                  R: A falta de headers apropriados pode causar desde falhas simples na interpretação de dados até graves vulnerabilidades de segurança. Eles desempenham papéis essenciais na comunicação, autenticidade, segurança, compatibilidade e performance.

1. Quais são as vantagens de usar bibliotecas como Axios ou Fetch em comparação com XMLHttpRequest?

           R: Axios se destaca por sua simplicidade, suporte automático a JSON, tratamento integrado de erros, e funcionalidade de cancelamento. O Fetch API, embora mais moderno, ainda requer mais código para lidar com alguns cenários, mas é mais leve e integrado nativamente aos navegadores.

* Há situações onde você ainda utilizaria XMLHttpRequest? Por quê?

                        R: Ele pode ser útil em situações onde a compatibilidade com navegadores antigos, a  clareza de implementação ou o controle detalhado são críticos. No entanto, para novas aplicações ou onde for possível, a recomendação geralmente é optar por Fetch API ou Axios pela facilidade.

1. Como a segurança deve ser considerada ao enviar dados sensíveis em requisições HTTP?

              R: Proteger dados sensíveis em requisições HTTP requer uma abordagem que combine criptografia, autenticação, autorização. As diretrizes, podem minimizar os riscos e garantir que as informações sensíveis sejam transmitidas de maneira segura, protegendo tanto o cliente quanto o servidor contra possíveis ataques.

* Quais práticas você implementaria para proteger dados durante a comunicação?

                      R: Implementar práticas de segurança cria uma defesa em profundidade que protege dados sensíveis durante a comunicação. Logo uma das ferramentas que são essenciais usar para a segurança protegendo a comunicação é HTTPS, validação de entrada, autenticação e autorização, minimização e dados sensíveis, Implementação de Headers de Segurança, Criptografia de Dados, Auditoria e Monitoramento e Conscientização do Usuário.

1. Como você poderia utilizar os métodos HTTP em um projeto de API que desenvolve?

          R: o uso correto dos métodos HTTP é essencial para criar uma interface clara, intuitiva e eficiente. Cada método HTTP tem suas semânticas específicas, que ajudam a descrever a intenção da operação, algumas diretrizes são:  get, post,put,delete, head,etc

* Pense em um projeto em que você precisaria implementar diferentes métodos. Quais seriam?

                      R: a API utiliza GET para recuperação de dados, POST para criação, PUT e PATCH para atualização de tarefas, e DELETE para exclusão. O uso de OPTIONS garante que o cliente saiba quais métodos são suportados. Essa estrutura permite um gerenciamento eficiente e claro das tarefas.

1. Qual a diferença entre requisições síncronas e assíncronas?

           R: A  Síncronas bloqueiam a execução até a resposta, já assíncronas continuam a execução em segundo plano.

* Como a natureza assíncrona das requisições HTTP impacta a experiência do usuário?

                      R: HTTP melhora a experiência do usuário ao garantir que a interface permaneça responsável, permitindo interações contínuas e carregamento de dados, tendo resultados em uma experiência mais suave e satisfatória.

1. Como você trataria erros em requisições HTTP?

            R: Ao tratar erros em requisições HTTP, é importante  identificar erros corretamente, fornecer feedback ao usuário, implementar lógica de repetição para erros temporários, testar regularmente a forte da aplicação.

* Quais estratégias você poderia implementar para garantir uma boa experiência em caso de falhas?

                     R: Uma das estratégias cruciais é implementando mensagens de erros claras, carregamento offline, experiência alternativa, redirecionamento de fluxo, ações de navegação, monitoramento e alertas e teste regular.

1. Que práticas de design de API você considera essenciais para uma boa documentação e usabilidade?

           R: Para garantir uma boa documentação e usabilidade em uma API, algumas práticas de design são essenciais como: clareza,documentação detalhada, uso de https, api, autenticação e segurança, testes interativos, feedbacks de erros e exemplo de casos de uso.

* O que você incluiria em uma documentação de API para facilitar o uso por desenvolvedores externos?

                         R: Para facilitar o uso de uma API por desenvolvedores externos, a documentação deve ser abrangente, clara e fácil de navegar, os elementos que eu incluiria seriam Introdução, Guia de Início rápido,Autenticação,exemplos de requisições e respostas, formatos de dados,tratamento de erros,Suporte e Mudanças