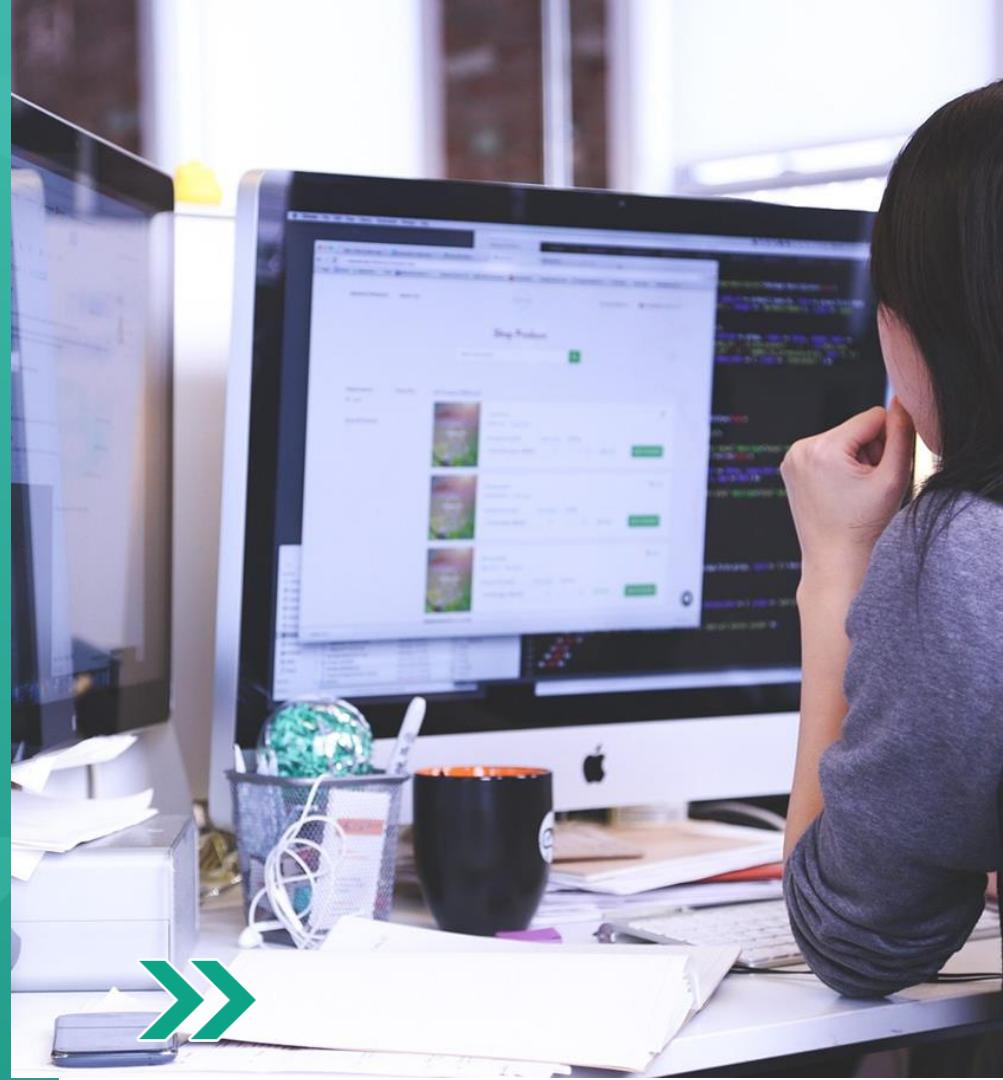


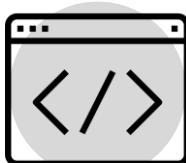
Profissão: Engenheiro Front-End



BOAS PRÁTICAS



Projeto 5 – Parte I



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

- **Faça a configuração inicial**
- **Construa o componente de filtro**
- **Parametrize os filtros**
- **Construa o card de tarefas**
- **Conheça Enums**



Faça a configuração inicial

Caracteres especiais em HTML

O uso de caracteres especiais HTML é fundamental para garantir que o texto seja exibido corretamente no navegador, especialmente quando se trata de caracteres que têm significados especiais na linguagem HTML, como <, >, ", ', &, entre outros. Acompanhe algumas dicas úteis sobre o uso de caracteres especiais HTML:



- **Entidades HTML:** Utilize entidades HTML para representar caracteres especiais. As entidades HTML são sequências de texto que começam com & e terminam com ;. Por exemplo:

- < representa o caractere <
- > representa o caractere >
- " representa o caractere "
- ' representa o caractere '
- & representa o caractere &

- **Evite o uso direto:** Evite usar caracteres especiais diretamente em seu código HTML, pois eles podem ser interpretados pelo navegador de maneira inesperada ou causar problemas de renderização.

- **Codificação de caracteres:** Certifique-se de que o arquivo HTML está codificado corretamente. Usar a codificação UTF-8 é uma prática comum para garantir que os caracteres especiais sejam tratados corretamente.

Faça a configuração inicial

Caracteres especiais em HTML

O uso de caracteres especiais HTML é fundamental para garantir que o texto seja exibido corretamente no navegador, especialmente quando se trata de caracteres que têm significados especiais na linguagem HTML, como <, >, ", ', &, entre outros. Acompanhe algumas dicas úteis sobre o uso de caracteres especiais HTML:



- **Formulários:** Em formulários HTML, certifique-se de que os caracteres especiais, como <, >, &, não causem problemas de segurança, especialmente em campos onde os dados serão enviados ao servidor. Considere usar recursos de segurança, como a validação dos dados do usuário e a sanitização do lado do servidor.
- **Citação de código ou texto:** Ao citar código ou texto em seu documento HTML, é uma boa prática envolver o conteúdo em elementos de bloco adequados, como <code> para código e <blockquote> para citações. Isso ajuda a manter a formatação e destaque corretamente os caracteres especiais.
- **Comentários:** Em comentários dentro do código HTML, evite usar sequências de caracteres especiais que possam fechar tags prematuramente ou causar erros de formatação.

Construa o componente de filtro

Viewport

Configurar corretamente a meta tag de viewport no HTML é fundamental para garantir que suas páginas da web sejam responsivas e se adaptem bem a diferentes dispositivos e tamanhos de tela. Acompanhe algumas dicas importantes sobre a configuração do viewport no HTML:



- **Definir a largura do viewport:** Use a propriedade width para definir a largura inicial do viewport. Definir width=device-width fará com que o viewport tenha a mesma largura do dispositivo, garantindo que o conteúdo seja exibido sem cortes laterais.
- **Definir o dimensionamento inicial:** A propriedade initial-scale permite definir o fator de escala inicial para o viewport. Definir initial-scale=1.0 evita o dimensionamento automático da página no carregamento, mantendo o tamanho original do conteúdo.
- **Evitar o dimensionamento da página:** Evite usar user-scalable=no para desativar o dimensionamento da página, a menos que seja estritamente necessário. Isso pode ser considerado uma má prática, pois impede que os usuários ajustem manualmente o zoom, o que pode ser frustrante para pessoas com deficiências visuais ou em dispositivos com tamanhos de tela atípicos.

Construa o componente de filtro

Viewport

Configurar corretamente a meta tag de viewport no HTML é fundamental para garantir que suas páginas da web sejam responsivas e se adaptem bem a diferentes dispositivos e tamanhos de tela. Acompanhe algumas dicas importantes sobre a configuração do viewport no HTML:



- **Escolher a unidade de largura:** Quando definir valores de largura no viewport, como width=device-width, lembre-se de que a unidade device-width refere-se à largura do dispositivo em pixels físicos. Isso é útil para dispositivos móveis, mas em telas maiores, como monitores de desktop, pode resultar em layouts muito amplos. Considere usar outras unidades, como porcentagens ou vw (viewport width), para obter mais controle sobre o layout.
- **Viewport mínimo e máximo:** Em alguns casos, pode ser útil definir um minimum-scale e/ou maximum-scale para limitar o dimensionamento da página, evitando que o conteúdo seja dimensionado em extremos indesejados.
- **Viewport para dispositivos com alta densidade de pixels:** Em dispositivos com alta densidade de pixels (como Retina displays), considere definir initial-scale e minimum-scale para valores maiores que 1 para evitar que o conteúdo fique muito pequeno e difícil de ler.

Parametrize os filtros

Utility types (Utilidades de tipo)

O `omit` é uma das várias utilidades de tipo (utility types) fornecidas pelo TypeScript, que tornam a manipulação e definição de tipos mais expressiva e flexível.

Acompanhe algumas dicas sobre como usar as utilidades de tipo.



- **Conheça as utilidades de tipo disponíveis:** O TypeScript oferece várias utilidades de tipo prontas para uso, como `Partial`, `Pick`, `Omit`, `Record`, `Exclude`, `Extract`, `Required`, `ReturnType`, entre outras. Familiarize-se com essas utilidades e suas funcionalidades para aproveitar ao máximo seus recursos.
- **Utilize `Partial` para tipos opcionais:** Quando você tiver um tipo com várias propriedades opcionais, use `Partial` para tornar todas as propriedades do tipo como opcionais de uma só vez. Isso pode facilitar a criação de objetos parciais durante o desenvolvimento.
- **Use `Pick` e `Omit` para extrair e remover propriedades:** `Pick` permite criar um novo tipo contendo apenas as propriedades específicas de outro tipo, enquanto `Omit` permite criar um novo tipo excluindo propriedades específicas do tipo original. Essas utilidades são úteis para restringir ou simplificar os tipos conforme necessário.

Parametrize os filtros

Utility types (Utilidades de tipo)

O `omit` é uma das várias utilidades de tipo (utility types) fornecidas pelo TypeScript, que tornam a manipulação e definição de tipos mais expressiva e flexível.

Acompanhe algumas dicas sobre como usar as utilidades de tipo.



- **Crie mapeamentos de tipos com Record:** O `Record` permite criar tipos com base em um conjunto de chaves e valores. Ele é útil quando você deseja definir um dicionário, mapeamento ou objeto com um conjunto predefinido de chaves.
- **Use Exclude e Extract para manipulação condicional de tipos:** Essas utilidades são úteis quando você precisa filtrar ou extrair tipos com base em condições ou predicados.
- **Combinando utility types:** Você pode combinar várias utilidades de tipo para obter resultados mais complexos e personalizados. Experimente combinações de utilidades para atender às necessidades específicas do seu código.

Construa o card de tarefas

Overflow

Acompanhe algumas dicas úteis sobre o uso do overflow:



● **Entenda os valores do overflow:** A propriedade overflow pode receber diferentes valores:

- visible: O comportamento padrão, o conteúdo pode se estender além dos limites do elemento.
- hidden: O conteúdo que excede o tamanho do elemento será cortado e não será visível.
- scroll: Será adicionada uma barra de rolagem, mesmo que o conteúdo não exceda o tamanho do elemento, tornando a rolagem sempre disponível.
- auto: Adiciona uma barra de rolagem somente se o conteúdo exceder o tamanho do elemento.

● **Cuidado com overflow: hidden:** Embora overflow: hidden seja útil para recortar o conteúdo que excede o tamanho do elemento, tenha cuidado para não cortar informações importantes ou elementos visíveis, especialmente em layouts responsivos.

Conheça Enums

Os enums são úteis quando você precisa representar um conjunto de valores relacionados e atribuir significado semântico aos valores, tornando o código mais legível e autoexplicativo. Acompanhe algumas dicas para usar os enums.



- **Atribua valores personalizados, se necessário:** Por padrão, os enums começam a atribuir valores numéricos a partir de 0 e aumentam em ordem crescente. No entanto, você pode atribuir valores personalizados aos membros do enum para garantir que eles se alinhem com requisitos específicos do seu sistema.
- **Evite valores duplicados:** Certifique-se de que cada membro do enum tenha um valor único. Valores duplicados podem levar a comportamentos inesperados e difíceis de depurar.
- **Use enums em vez de valores literais:** Em vez de usar valores literais (como strings ou números) diretamente em seu código, considere usar os membros do enum correspondentes. Isso tornará o código mais legível e menos propenso a erros de digitação.

Conheça Enums

Os enums são úteis quando você precisa representar um conjunto de valores relacionados e atribuir significado semântico aos valores, tornando o código mais legível e autoexplicativo. Acompanhe algumas dicas para usar os enums.



- **Cuidado com inversões de mapeamento:** Os enums do TypeScript permitem obter o valor numérico de um membro e vice-versa. No entanto, tenha cuidado com inversões de mapeamento, pois nem sempre é garantido que um valor numérico seja válido para um enum.
- **Utilize const enums quando a performance é importante:** Em alguns casos, quando a performance é uma preocupação, você pode usar const enums em vez de enums regulares. Os const enums são totalmente eliminados durante a compilação, e seus valores são substituídos diretamente no código, resultando em um código mais eficiente.
- **Considere enums string:** O TypeScript também suporta enums com valores string. Esses enums podem ser úteis quando você precisa representar um conjunto de valores que não sejam numéricos, como nomes de cores, tipos de status etc.

Conheça Enums

Os enums são úteis quando você precisa representar um conjunto de valores relacionados e atribuir significado semântico aos valores, tornando o código mais legível e autoexplicativo. Acompanhe algumas dicas para usar os enums.



- **Cuidado com inversões de mapeamento:** Os enums do TypeScript permitem obter o valor numérico de um membro e vice-versa. No entanto, tenha cuidado com inversões de mapeamento, pois nem sempre é garantido que um valor numérico seja válido para um enum.
- **Use maiúsculas:** por convenção, os membros de um enum são escritos em letras maiúsculas e separados por underscore (_) quando composto por mais de uma palavra. Isso torna o código mais legível e facilita a identificação rápida de que se trata de um enum.

Bons estudos!

