Curso de Programação Web Front-End: To-Do-List

Introdução

Bem-vindo ao curso de Programação Web Front-End! Este curso foi projetado para te ensinar a construir um site do zero, utilizando HTML, CSS e JavaScript. Vamos guiá-lo passo a passo, desde a configuração do ambiente de desenvolvimento até a criação de um site de lista de tarefas (to-do-list). Além das videoaulas, foi produzido este texto de apoio com dicas valiosas para programar com eficiência e boas práticas.

Objetivos do Curso:

1. Baixar e configurar as IDEs.
2. Aprender HTML.
3. Aprender CSS.
4. Aprender JavaScript.
5. Construir um site completo de lista de tarefas (to-do-list).

Como funcionará o curso?

O curso está dividido em módulos, cada um focado em uma etapa específica do desenvolvimento web. Abaixo, você encontra os detalhes de cada módulo, com explicações teóricas para apoiar seu aprendizado.

O que você encontrará aqui no curso:

Configuração das IDEs

1. O que são IDEs e por que usá-las

IDEs, ou Ambientes de Desenvolvimento Integrado, são ferramentas essenciais para programadores. Elas combinam várias funções necessárias para o desenvolvimento de software em uma única interface. As principais vantagens de usar uma IDE incluem:

Realce de Sintaxe: As IDEs destacam diferentes partes do código com cores específicas, ajudando a identificar erros de sintaxe rapidamente e tornando o código mais legível.

Autocompletar: Esta funcionalidade sugere e completa automaticamente palavras-chave e variáveis enquanto você digita, economizando tempo e reduzindo erros.

Debugging: As IDEs oferecem ferramentas integradas para depuração, permitindo que você execute o código passo a passo, examine variáveis e encontre bugs de forma eficiente.

Controle de Versão: Muitas IDEs integram sistemas de controle de versão como Git, facilitando a gestão de versões do código e a colaboração com outros desenvolvedores.

Interface Unificada: Com uma IDE, você tem acesso a um ambiente de trabalho coeso onde todas essas ferramentas estão disponíveis sem a necessidade de alternar entre diferentes programas.

Analogia para Entender IDEs:

Tentar programar sem uma IDE é como construir uma casa com apenas um martelo e pregos. Usar uma IDE é como ter uma caixa de ferramentas completa, onde cada ferramenta está facilmente acessível para a tarefa correta.

2. Instalando e navegando na IDE

- Passo a Passo da Instalação e Navegação na IDE: Começar com uma IDE envolve baixá-la do site oficial e seguir as instruções de instalação. Após instalado, aprender a navegar pela interface é crucial. Você aprenderá a criar novos projetos, organizar arquivos em pastas e configurar preferências. Por exemplo, ao criar um projeto, a IDE pedirá o nome do projeto e onde salvá-lo. Saber como abrir, editar e salvar arquivos é essencial para aproveitar todos os recursos da IDE.

HTML

1. Criando arquivos HTML, CSS e JavaScript

Fundamentos de Arquivos Web

A criação de arquivos HTML, CSS e JavaScript é a base de qualquer site. Cada tipo de arquivo desempenha um papel específico:

HTML (HyperText Markup Language): Define a estrutura do conteúdo da página. Ele cria a espinha dorsal do site, organizando o conteúdo em elementos como parágrafos, cabeçalhos, listas e links.

CSS (Cascading Style Sheets): Cuida da aparência e do layout da página. Ele estiliza o conteúdo HTML, definindo cores, fontes, espaçamentos e posicionamentos.

JavaScript: Adiciona interatividade ao site. Ele permite criar comportamentos dinâmicos, como responder a cliques de botões, validar formulários e manipular elementos da página.

Manter esses arquivos separados é uma prática importante que melhora a organização e a manutenção do código. Ao separar o conteúdo (HTML), a apresentação (CSS) e a lógica (JavaScript), você pode atualizar e modificar cada aspecto do site de forma independente e eficiente.

Referenciando CSS e JavaScript no HTML:

Para conectar os arquivos CSS e JavaScript ao HTML, usamos as tags <link> e <script>:

CSS: Para adicionar um arquivo CSS, usamos a tag <link> dentro do <head> do HTML.

```
<link rel="stylesheet" href="styles.css">
```

JavaScript: Para adicionar um arquivo JavaScript, usamos a tag <script>, geralmente no final do <body> para garantir que o HTML seja carregado antes de qualquer script.

```
<script src="scripts.js"></script>
```

2. Elementos básicos do HTML

Os elementos HTML são como blocos de construção que definem a estrutura e o conteúdo da sua página web. Aqui estão alguns dos elementos mais comuns e suas funções:

- Cabeçalhos (<h1> a <h6>): Usados para títulos e subtítulos. <h1> é o cabeçalho mais importante e <h6> o menos importante.

```
<h1>Este é um Título Principal</h1>

<h2>Este é um Subtítulo</h2>
```

- Parágrafos (<p>): Usados para blocos de texto.

Este é um parágrafo de exemplo.

- Listas Não Ordenadas (<ul>) e Ordenadas (<ol>): Usadas para agrupar itens em uma lista. Os itens são definidos com a tag <li>.

Item da lista não ordenada

Outro item

Item da lista ordenada

Outro item

Links (<a>): Usados para criar hyperlinks que navegam para outras páginas ou sites.

Clique aqui para visitar um site

Imagens (<img>): Usadas para incorporar imagens na página.

<img src="imagem.jpg" alt="Descrição da imagem">

Histórico do HTML e CSS:

Antes da introdução do CSS, os estilos eram aplicados diretamente nos elementos HTML usando atributos como bgcolor, font, align, etc. Isso resultava em um código

HTML muito confuso e difícil de manter. Com o CSS, a apresentação do conteúdo foi separada da estrutura, permitindo um código HTML mais limpo e semântico.

3. Construção inicial do to-do-list

- Construção da Estrutura HTML: Começaremos a construir a estrutura do to-do-list. Isso inclui criar um formulário para adicionar novas tarefas e uma lista onde as tarefas serão exibidas. Usaremos elementos como `<form>`, `<input>`, `<button>` e `<ul>` ou `<ol>`. Um HTML bem estruturado é fundamental para uma página semântica e acessível.

CSS

1. Estilizando a lista de tarefas

Introdução ao CSS

CSS (Cascading Style Sheets) é uma **linguagem usada para controlar a aparência visual de uma página web**. Enquanto o HTML fornece a estrutura do conteúdo, o CSS permite que os desenvolvedores definam estilos para elementos específicos, tornando a interface mais atraente e funcional. Para estilizar a lista de tarefas, vamos focar nos elementos de lista ordenada (<ol>) e não ordenada (<ul>).

Com CSS, podemos ajustar vários aspectos visuais, como margens, espaçamentos, bordas, fontes e tamanhos de texto. A propriedade list-style-type, por exemplo, permite alterar o estilo dos marcadores das listas, como círculos, discos ou números. Ajustar margin e padding ajuda a controlar o espaçamento interno e externo dos elementos de lista, melhorando o layout e a legibilidade. A escolha da font-family define a aparência do texto, podendo influenciar a sensação geral do design.

Ao definir estilos consistentes para as listas, garantimos uma aparência harmoniosa e

profissional, além de facilitar a manutenção do código. Um CSS bem estruturado e organizado torna a aplicação mais escalável e adaptável a futuras mudanças.

2. Estilizando itens da lista

Estilização Detalhada de Itens

Cada item da lista (<li>) pode ser estilizado individualmente, permitindo um controle preciso sobre a apresentação das tarefas. Utilizamos propriedades CSS como **margin, padding, border, background-color, font-size e color** para criar uma interface visualmente agradável e funcional.

A propriedade **margin** ajusta o espaço ao redor de cada item, evitando que os elementos fiquem muito próximos uns dos outros. O padding controla o espaçamento interno, garantindo que o texto não fique colado às bordas do item. Adicionar uma border pode destacar cada item, enquanto o background-color permite a alternância de cores para melhorar a legibilidade.

Definir o **font-size e color** adequados é crucial para a acessibilidade e a experiência do usuário. Tamanhos de fonte bem escolhidos garantem que o texto seja facilmente legível, enquanto as cores devem ter um bom contraste para serem visíveis por todos os usuários, incluindo aqueles com deficiências visuais.

A alternância de cores de fundo, utilizando a pseudo-classe **nth-child**, pode melhorar a distinção entre os itens, facilitando a visualização das tarefas. Um design cuidadoso e detalhado dos itens da lista contribui significativamente para a usabilidade e a estética da aplicação.

3. Estilizando caixas de seleção

Estilização de Elementos Interativos

Caixas de seleção **(<input type="checkbox">)** permitem que os usuários marquem

tarefas como concluídas, tornando-as um elemento interativo crucial em uma lista de tarefas. CSS oferece várias propriedades para estilizar esses elementos, como **appearance, background-color, border**, e pseudo-classes como **:checked**.

A propriedade **appearance** pode ser usada para remover o estilo padrão do navegador, permitindo uma personalização completa. Ajustar o **background-color e border** ajuda a criar caixas de seleção visualmente agradáveis e coerentes com o design geral. Utilizar a pseudo-classe **:checked** permite definir estilos diferentes para caixas marcadas e não marcadas, proporcionando feedback visual imediato.

Efeitos de **hover e foco** são essenciais para melhorar a experiência do usuário. Eles indicam que o elemento é interativo e respondem ao comportamento do usuário, como mudanças de cor ou adição de sombras quando o usuário passa o cursor sobre a caixa de seleção. Esses efeitos tornam a interface mais intuitiva e responsiva.

Estilizar caixas de seleção de forma eficaz não só melhora a estética da aplicação, mas também a torna mais acessível e fácil de usar, resultando em uma melhor experiência geral para o usuário.

4. Estilizando botões de ação

Design de Botões

Botões de ação são elementos fundamentais para a interatividade do site. Eles são usados para adicionar, remover ou modificar tarefas na lista. Utilizamos CSS para definir cores, tamanhos, bordas e efeitos de hover e foco, criando botões funcionais e atraentes.

Propriedades como **background-color e colo**r são usadas para definir as cores de fundo e do texto do botão, garantindo que eles se destaquem e sejam facilmente identificáveis. **border-radius** pode ser aplicado para suavizar os cantos dos botões, conferindo um visual moderno e amigável. Ajustar **padding** garante que o texto dentro do botão tenha espaço adequado ao seu redor, melhorando a legibilidade e a

aparência.

Efeitos de **hover e foco** são importantes para fornecer feedback visual ao usuário. Quando um usuário passa o cursor sobre um botão, ele pode mudar de cor ou adicionar uma sombra, indicando que está pronto para ser clicado. Esses efeitos tornam os botões mais interativos e intuitivos.

A propriedade **cursor** pode ser usada para alterar o cursor do mouse para uma mão ao passar sobre o botão, indicando claramente que ele é clicável. Um design de botão bem pensado melhora significativamente a usabilidade da aplicação e a experiência do usuário.

5. Estilizando a entrada de texto

Customização de Campos de Entrada

A entrada de texto **(<input type="text">)** para adicionar novas tarefas deve ser estilizada para garantir que seja esteticamente agradável e fácil de usar. Utilizamos propriedades CSS para definir o tamanho, margens, bordas e fontes, criando um campo de entrada funcional e atraente.

Definir a largura (**width**) e altura (**height**) adequadas garante que o campo de entrada tenha o tamanho certo para o conteúdo e seja facilmente visível. O **padding** ajusta o espaçamento interno, proporcionando um espaço confortável para o texto. Propriedades como **border** e **border-radius** podem ser usadas para estilizar as bordas, dando ao campo de entrada um visual mais polido.

Escolher a **font-family** correta é essencial para a legibilidade, enquanto a cor (**color**) e o **background-color** do campo de entrada devem ser escolhidos para garantir um bom contraste e visibilidade. Utilizar margens (**margin**) ajuda a posicionar o campo de entrada adequadamente na página, garantindo que ele não fique muito próximo de outros elementos.

Estilizar o campo de entrada de texto de forma eficaz não só melhora a aparência geral

da aplicação, mas também facilita a interação do usuário, resultando em uma melhor experiência de uso.

6. Estilizando barra de progresso ou contador de tarefas

Feedback Visual com CSS

Barras de progresso ou contadores de tarefas fornecem feedback visual sobre o progresso do usuário, ajudando-os a entender rapidamente quanto já foi feito e quanto ainda falta fazer. Utilizamos CSS para definir a aparência desses elementos, ajustando cores, tamanhos e posicionamento.

Para uma barra de progresso, propriedades como **width**, **height** e **background-color** são essenciais. A **width** da barra interior pode ser ajustada dinamicamente com base no progresso, enquanto a **height** define a espessura da barra. Utilizar cores contrastantes para a barra de fundo e a barra de progresso torna o feedback visual mais claro.

Contadores de tarefas podem ser estilizados usando **font-size**, **color** e **margin** para garantir que sejam facilmente legíveis e bem posicionados na interface. A utilização de animações suaves pode melhorar ainda mais a experiência do usuário, mostrando o progresso de forma gradual e dinâmica.

Elementos de feedback visual como esses são essenciais para manter o usuário engajado e informado sobre seu progresso, melhorando a usabilidade e a satisfação geral com a aplicação.

7. Estilizando layout responsivo

Design Responsivo com CSS

Um **layout responsivo** se ajusta a diferentes tamanhos de tela, garantindo que a aplicação funcione bem em dispositivos móveis, tablets e desktops. Utilizamos media

queries **(@media)** para aplicar estilos específicos a diferentes resoluções de tela, ajustando tamanhos de fonte, espaçamentos e reorganizando elementos para garantir funcionalidade em qualquer dispositivo.

Media queries permitem que os desenvolvedores definam pontos de interrupção (**breakpoints**) onde o design precisa se ajustar. Por exemplo, podemos alterar o layout de uma coluna única em dispositivos móveis para várias colunas em telas maiores. Ajustar **font-size** e **padding** para diferentes tamanhos de tela melhora a legibilidade e a experiência do usuário.

Flexbox e **Grid** são duas técnicas CSS poderosas para criar **layouts responsivos**. **Flexbox** é ideal para **layouts unidimensionais**, permitindo que os elementos dentro de um contêiner se ajustem dinamicamente ao espaço disponível. **Grid** é útil para **layouts bidimensionais**, permitindo a criação de layouts complexos com linhas e colunas.

Garantir que o design seja responsivo é essencial para proporcionar uma experiência de usuário consistente e agradável, independentemente do dispositivo usado para acessar a aplicação.

8. Estilizando animações e transições

Animações e Transições Suaves

Animações e transições melhoram a experiência do usuário ao adicionar dinamismo e interatividade à interface. Utilizamos propriedades CSS como **transition**, **animation**, **keyframes** e **transform** para criar efeitos suaves ao adicionar, remover ou completar tarefas.

A propriedade **transition** permite a transição suave de uma propriedade CSS para outra, como mudar a cor de fundo de um botão ao passar o cursor sobre ele. **Animation** e **keyframes** permitem definir animações mais complexas, como mover um item da lista ao ser marcado como concluído.

Transform permite aplicar transformações 2D e 3D aos elementos, como rotacionar, escalar ou mover elementos.

Usar animações e transições com moderação é importante para não sobrecarregar o usuário e garantir que a interface permaneça rápida e responsiva.

9. Estilizando outros elementos

Personalização de Elementos Adicionais

Outros elementos, como etiquetas de prioridade, notas ou datas, podem ser estilizados para melhorar a funcionalidade do to-do-list. Utilizamos CSS para ajustar cores, tamanhos e posicionamento, destacando informações importantes e facilitando a organização das tarefas.

Etiquetas de prioridade podem ser estilizadas com cores diferentes para indicar níveis de urgência, ajudando os usuários a identificar rapidamente as tarefas mais importantes. Notas adicionais podem ser exibidas com fontes menores e cores distintas para diferenciar do texto principal.

Datas podem ser formatadas e posicionadas de forma clara, garantindo que os prazos sejam visíveis e fáceis de entender. Utilizar ícones ou outros elementos visuais também pode ajudar a destacar informações importantes, tornando a interface mais intuitiva e informativa.

A personalização desses elementos adicionais contribui para uma melhor organização e usabilidade da lista de tarefas, proporcionando ao usuário uma ferramenta mais eficaz e agradável de usar.

JavaScript

1. Introdução ao JavaScript

História e Importância do JavaScript

JavaScript é uma linguagem de programação usada para **tornar sites interativos e dinâmicos**. Foi criada em 1995 por Brendan Eich, enquanto trabalhava na empresa Netscape. No início, JavaScript era usado para coisas simples, como validar formulários e criar pequenas animações, mas ao longo do tempo, se tornou uma das linguagens mais populares do mundo.

Hoje, JavaScript é essencial para o desenvolvimento web. Ele permite que desenvolvedores manipulem a estrutura de um site (**DOM - Document Object Model**), respondam a ações dos usuários (como cliques e digitação), e se comuniquem com servidores para buscar ou enviar dados sem precisar recarregar a página inteira. Além disso, com a criação do **Node.js**, JavaScript também pode ser usado para programar no servidor, permitindo que desenvolvedores usem a mesma linguagem tanto no **front-end** (a parte do site que você vê) quanto no **back-end** (a parte do site que lida com dados e lógica).

2. Lógica do to-do-list: adição e remoção de tarefas

Implementação da Lógica com JavaScript

Para fazer um to-do list (lista de tarefas) funcionar, precisamos usar JavaScript para adicionar e remover tarefas da lista. Vamos usar várias funções e métodos de JavaScript para interagir com os elementos HTML da nossa página.

Selecionar Elementos HTML: Primeiro, precisamos pegar os elementos que vamos manipular, como o campo de entrada de texto, o botão de adicionar e a lista onde as tarefas vão aparecer. Fazemos isso usando métodos como document.querySelector.

Adicionar Tarefas: Quando o usuário digita uma nova tarefa e clica no botão para adicionar, JavaScript cria um novo item na lista. Usamos createElement para criar um novo item de lista (<li>) e appendChild para adicionar esse item à lista existente.

Remover Tarefas: Para remover tarefas, adicionamos um botão de exclusão em cada item da lista. Quando o usuário clica nesse botão, JavaScript remove o item da lista usando removeChild.

3. Estilização dinâmica com JavaScript

Estilização Dinâmica e Personalização

JavaScript também pode mudar a aparência dos elementos da página em resposta às ações do usuário, como cliques e passagens do mouse.

Adicionar e Remover Classes: Usamos classList.add e classList.remove para mudar a classe de um elemento e, assim, aplicar diferentes estilos CSS. Por exemplo, podemos adicionar uma classe para destacar uma tarefa concluída.

Alterar Atributos: Podemos usar setAttribute para mudar atributos de um elemento HTML, como alterar o texto que aparece em um campo de entrada.

Modificar Estilos Inline: Embora não seja recomendado para todos os casos, podemos mudar estilos diretamente em um elemento com a propriedade style, como mudar a cor de fundo de um botão quando ele é clicado.

Eventos de Mouse e Teclado: JavaScript permite capturar eventos como cliques e teclas pressionadas, permitindo criar interfaces mais interativas. Por exemplo, podemos mudar a cor de um item quando o mouse passa sobre ele.

Com JavaScript, podemos criar uma aplicação de to-do list onde os usuários podem adicionar e remover tarefas facilmente, e também podemos mudar a aparência dos elementos da página conforme os usuários interagem com eles. Aprender JavaScript é essencial para quem quer desenvolver sites modernos e interativos.

Boas Práticas de Programação

Importância das Boas Práticas de Programação: Seguir boas práticas de programação é fundamental para garantir que o código seja fácil de entender, manter e expandir. A clareza e a legibilidade do código permitem que outros desenvolvedores compreendam e colaborem no projeto. Comentários e documentação são cruciais para explicar a lógica e os detalhes técnicos. Estruturar e modularizar o código facilita a manutenção e a escalabilidade, permitindo que novas funcionalidades sejam adicionadas sem comprometer a estabilidade do sistema.

- Clareza e legibilidade do código: Escrever código fácil de ler e entender por outros desenvolvedores. Utilize indentação, nomes de variáveis descritivos e mantenha o código limpo e organizado.

- Comentários e documentação: Inclua comentários que expliquem partes complexas do código e mantenha uma documentação adequada.

- Estruturação e modularização do código: Divida o código em módulos e funções bem definidos. Cada módulo ou função deve ter uma responsabilidade específica.

- Manutenção e escalabilidade: Escreva código fácil de manter e escalar. Evite duplicação de código e utilize padrões de design apropriados.

Conclusão

Este curso proporcionará a você uma compreensão abrangente de como construir um site do zero utilizando as tecnologias web mais importantes. Ao final do curso, você será capaz de planejar, desenvolver e estilizar um site funcional, seguindo as melhores práticas de programação e utilizando ferramentas modernas de desenvolvimento web. Assista aos vídeos que preparamos para você!

Boa sorte e bons estudos!