Roadmap Frontend

1. Como funciona a internet?

- Pesquisar artigos e vídeos de como funciona a internet.
- O que é HTTP.
- Funcionamento de um browser.
- DNS.
- Hosting.

2. Lógica de Programação

Antes de começar a estudar uma linguagem de programação ou seguir qualquer roadmap de desenvolvimento, é fundamental aprender lógica de programação. Ela vai treinar seu raciocínio para resolver problemas, usando estruturas como fluxogramas e pseudocódigos.

Sequência de estudos:

- **1.** Variáveis, Comandos de entrada e de saída
- **2.** Operadores Aritméticos
- **3.** Operadores Relacionais
- **4.** Operadores Lógicos
- **5.** Estruturas Condicionais
- **6.** Estruturas de Repetição
- **7.** Funções
- **8.** Vetores
- **9.** Matriz

3. Roadmap JavaScript

- **1.** Sintaxe e Variáveis
- **2.** Comandos de entrada e saída (Dica: biblioteca prompt-sync)
- **3.** Operadores Matemáticos
- **4.** Operadores de Comparação
- **5.** Operadores Lógicos
- **6.** Estruturas de Condição
- **7.** Funções
- **8.** Objetos
- **9.** Arrays
- **10.** Interação com Arrays
- **11.** Métodos de cada tipo de variável
- **12.** Datas
- **13.** Promises

Package Managers

Para facilitar o desenvolvimento de aplicações em JavaScript, utilizamos bibliotecas que tornam determinadas tarefas mais eficientes. Essas bibliotecas, ou *libs*, podem ajudar desde a manipulação de dados, formatação de moedas, chamadas de API, conexão com bancos de dados e outras funcionalidades lógicas, até a inclusão de pacotes visuais com componentes prontos para uso. Para gerenciar essas bibliotecas, usamos *package managers*, sendo os dois principais o NPM e o Yarn.

4. Roadmap Git e Github

- Inicializar um repositório (git init)
- Checkar o status (git status)
- Adicionar um arquivo pra stage area (git add)
- Adicionar a modificações ao histórico (commit)
- Visualizar histórico (git log)
- Verificar modificações nos arquivos (git diff)
- Renomear, remover e mover arquivos (rm, mv)
- Restaurar arquivos e alterar commits (restore, amend)
- Navegar para um commit passado (git checkout)
- Limpar working directory (git clean)
- Reverter um commit (git revert)
- Ignorar arquivos (gitignore)
- Branches (branch)

5. HTML

O HTML é responsável por estruturar o conteúdo da página web, organizando os elementos que serão exibidos. Já o CSS é a linguagem utilizada para estilizar esses elementos, definindo cores, layouts, fontes, espaçamentos e muito mais. Dominar bem os fundamentos dessas duas linguagens é essencial e deve ser uma das suas prioridades no início dos estudos em front-end. Para isso, recomendo focar nos seguintes tópicos:

Fundamentos de HTML: O que você precisa dominar:

1. Estrutura básica de um documento HTML

- Entenda a sintaxe padrão (`<!DOCTYPE html>`, `<html>`, `<head>`, `<body>`, etc.).
- **Tags de bloco (Block Elements):**
  - `<header>`
  - `<nav>`
  - `<div>`
  - `<section>`
  - `<footer>`
  - `<main>`
  - `<article>` / `<aside>`
- **Tags de texto e formatação:**
  - Títulos (`<h1>` a `<h6>`)
  - Parágrafos (`<p>`)
  - Negrito e itálico (`<strong>`, `<em>`)
  - Quebra de linha (`<br>`) e linha horizontal (`<hr>`)
- **Links e navegação:**
  - Hyperlink (`<a>`)
  - Âncoras e navegação interna
- **Imagens:**
  - Tag `<img>` e atributos (`src`, `alt`, `width`, `height`)
- **Botões:**
  - Tag `<button>` e tipos de botão (`submit`, `reset`, `button`)

- **Formulários e Inputs:**
  - `<form>`, `<input>`, `<label>`, `<textarea>`, `<select>`, `<option>`
- **Tabelas:**
  - Estrutura com `<table>`, `<thead>`, `<tbody>`, `<tr>`, `<th>`, `<td>`
- **Listas:**
  - Ordenadas (`<ol>`)
  - Não ordenadas (`<ul>`)
  - Itens de lista (`<li>`)

#CSS

Unidades de Medida e Tamanhos

Unidades: `px`, `em`, `rem`, `%`, `vh`, `vw`

Propriedades: `width`, `height`, `max-width`, `min-height`

Cores

Nomeadas, Hexadecimal (`#ffffff`), RGB/RGBA, HSL/HSLA
Propriedades: `color`, `background-color`, `border-color`

Estilização de Texto

Fontes: `font-family`, `font-size`, `font-weight`, `font-style`
Alinhamento e espaçamento: `text-align`, `line-height`, `letter-spacing`
Transformações e decoração: `text-transform`, `text-decoration`

Espaçamentos

`margin`, `padding`
Direcionais: `margin-top`, `padding-left`, etc.
Shorthand e espaçamento automático (`auto`)

Fundo (Background)

`background-color`, `background-image`
`background-size`, `background-position`, `background-repeat`

Bordas

`border-width`, `border-style`, `border-color`
Arredondamento: `border-radius`
Sombra: `box-shadow`

Posicionamento

Tipos de posicionamento:
`static`, `relative`, `absolute`, `fixed`, `sticky`
Flexbox:
`display: flex`
`flex-direction`, `justify-content`, `align-items`, `gap`
Grid Layout:
`display: grid`

`grid-template-columns`, `grid-template-rows`, `grid-gap`
`grid-area`, `place-items`, `justify-content`, `align-content`

6. Aplicações Básicas para Web

1. **Formulários**

Criação e estrutura com HTML (`<form>`, `<input>`, `<label>`, etc.)
Envio e validação de dados (client-side)
Integração com JavaScript para tratamento personalizado

2. **Eventos no JavaScript**
   Manipulação com `addEventListener()`
   Eventos comuns: `click`, `submit`, `change`, `input`, `keydown`, etc.
   Prevenção de comportamentos padrão (`event.preventDefault()`)

3. **Manipulação do DOM (Document Object Model)**
   Seleção de elementos: `getElementById`, `querySelector`, etc.
   Modificação de conteúdo: `innerHTML`, `textContent`, `value`
   Estilização e classes: `classList`, `style`, `setAttribute`
   Criação e remoção de elementos

4. **Comunicação com Servidores - Fetch API**
   Requisições assíncronas com `fetch()`
   Métodos: `GET`, `POST`, `PUT`, `DELETE`
   Tratamento de Promises com `.then()` e `.catch()`
   Uso com `async/await`

5. **Design Responsivo com Media Queries**
   Adaptação do layout para diferentes tamanhos de tela
   Sintaxe de media queries: `@media (max-width: 768px) { ... }`
   Uso combinado com Flexbox e Grid para melhor responsividade

7. React.js

1. **JSX (JavaScript XML)**
   Sintaxe que permite escrever HTML dentro do JavaScript
   Embutir expressões com `{ }`
   Diferenças em relação ao HTML (ex: `className`, `htmlFor`)

2. **Componentes**
   Componentes funcionais e **componentes de classe**
   Composição de componentes
   Props (propriedades): passagem de dados entre componentes

3. **State (Estado Local)**
   Gerenciamento de estado interno com `useState`
   Atualização e reatividade do componente
   Boas práticas com estados imutáveis

4. **React Hooks**
   `useState`, `useEffect`, `useRef`, `useContext`, `useMemo`, `useCallback`
   Regras de uso de hooks (somente em componentes funcionais)
   Custom hooks (hooks personalizados)

5. **Navegação com React Router (`react-router-dom`)**
   Criação de rotas com `<BrowserRouter>`, `<Routes>`, `<Route>`
   Navegação entre páginas com `<Link>` e `useNavigate`
   Rotas aninhadas e rotas dinâmicas (`/user/:id`)

6. **Gerenciamento de Estado Global**
   Context API
   Criação de contextos com `createContext`
   Compartilhamento de dados entre componentes
   Uso com `useContext`
   Redux
   Conceitos principais: `store`, `actions`, `reducers`
   Integração com `react-redux` (`Provider`, `useSelector`, `useDispatch`)
   Fluxo unidirecional de dados
   Zustand
   Gerenciamento de estado simples e leve
   Criação de stores com funções
   Uso direto com hooks personalizados

7. Chamadas de API em React

Axios – Biblioteca para requisições HTTP com suporte a interceptadores, baseURL, e tratamento de erros.
React Query – Gerenciamento de estado de dados assíncronos, com caching, refetching e sincronização automática.
useHttp – Hook personalizado para chamadas HTTP, geralmente usado para abstrair lógica repetitiva.
SWR – Estratégia de revalidação de dados do Vercel, focada em performance e simplicidade.
GraphQL com Apollo Client – Integração com APIs GraphQL, suporte a queries, mutations e cache inteligente.

---

8. CSS Moderno em React

Atomic Design – Organização dos componentes de UI em: átomos, moléculas, organismos, templates e páginas.
Styled Components – Estilização de componentes com CSS-in-JS, usando template literals.
Tailwind CSS – Framework utilitário de CSS com classes pré-definidas.
Emotion – Biblioteca CSS-in-JS leve e flexível.
Chakra UI – Conjunto de componentes acessíveis e estilizados prontos para uso.
Material UI (MUI) – Biblioteca com componentes seguindo o design system do Google Material Design.

---

9. Manipulação de Formulários

Yup – Biblioteca para validação de esquemas de dados, usada com React Hook Form e Formik.
Formik – Gerenciamento de formulários e estados, com integração a validações.
React Hook Form – Biblioteca leve e performática para criação e controle de formulários em React.

---

10. ESLint e Prettier

Ferramentas essenciais para padronização e qualidade do código:

ESLint – Analisa o código JavaScript/TypeScript para encontrar problemas e aplicar boas práticas.
Prettier – Formata automaticamente o código com base em regras configuradas, garantindo estilo consistente.

Recomendação: Sempre configure essas ferramentas em seus projetos para manter o código limpo, padronizado e evitar bugs desnecessários.

---

11. Cloud Básico para Deploy

Vercel – Ideal para projetos com Next.js, com deploy contínuo e integração com GitHub.
Netlify – Deploy de sites estáticos e SPAs, com suporte a funções serverless.
AWS S3 – Armazenamento de arquivos e hospedagem de sites estáticos com alta disponibilidade.

---

12. Sugestão de Tópicos Avançados

Para continuar sua evolução no front-end, explore os seguintes assuntos:

TypeScript – Superset do JavaScript com tipagem estática. Ideal para projetos em React.
Next.js – Framework para React com renderização híbrida (SSR, SSG, ISR) e rotas baseadas em arquivos.
CI/CD – Integração e entrega contínua de código.
GitHub Actions – Automatização de testes, builds e deploys via GitHub.
PWA (Progressive Web Apps) – Aplicações web com comportamento semelhante a apps nativos.
Testes em React:

- **Jest** – Testes unitários e de integração.
- **React Testing Library** – Testes focados na interação do usuário.
- **Cypress** – Testes de ponta a ponta (E2E).

Firebase – Backend como serviço (autenticação, banco de dados, storage, hosting).
Sentry – Monitoramento e rastreamento de erros em tempo real.

ROADMAP PARA PROGRAMADORES FRONT END