Instalação de ferramentas

Olá Aluno :)

Este curso possui o objetivo de contemplarmos os conceitos básciso de React e de tecnologias usadas para o desenvolvimento web.

Além disso iremos trabalhar com tecnologias open-source, a fim da inserção do aluno dentro desse universo.

Dentro dele, iremos cobrir as seguintes tecnologias:

- **Git**: é um sistema de controle de versões distribuído, usado principalmente no desenvolvimento de software, mas pode ser usado para registrar o histórico de edições de qualquer tipo.
- **Javascript**: é uma linguagem de programação interpretada estruturada, de script em alto nível com tipagem dinâmica fraca e multiparadigma. Juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias da World Wide Web.
- **React**: Biblioteca para construção de Interfaces de Usuário (UI) criada pelo Facebook.
- **TypeScript**: É um superconjunto de JavaScript desenvolvido pela Microsoft que adiciona tipagem e alguns outros recursos a linguagem.
- **NodeJS** - Ambiente de desenvolvimento backend para construções de APIs modernas usando a linguagem JavaScript. Não iremos abordar isso em nosso curso mas precisaremos dele para o NPM
- **NPM** - Gerenciador de Pacotes do NodeJS

E para isso precisaremos instalar algumas ferramentas principais:

- Visual Studio Code (https://code.visualstudio.com/download)
- GIT (https://git-scm.com/downloads)
- Postman (https://www.postman.com/downloads/)
- NodeJS e NPM (https://nodejs.org/en/download/)

Alem disso utilizaremos as seguintes como apoio do curso:

- Notion (https://www.notion.so/)

- CodeSandbox (https://codesandbox.io/)
- GitHub (https://github.com/)

Introdução ao Desenvolvimento WEB

Equipe operacional

- Front-end

  O desenvolvedor Front-end está muito relacionado com a interface gráfica do projeto. Ou seja, é onde se desenvolve a aplicação com a qual o usuário irá interagir diretamente, seja em softwares, sites, aplicativos, etc. Portanto, é essencial que o desenvolvedor tenha uma preocupação com a experiência do usuário, pois suas ações impactam diretamente este.

- Back-end

  Profissionais de desenvolvimento back-end são responsáveis por tudo o que está “atrás dos panos” desse cenário de programação, toda a estrutura que suporta nossas ações nas máquinas com desenvolvimento de APIs, obtenção de dados.

- DevOps

  O profissional de Devops procura seguir uma série de práticas e processos para acelerar o processo de desenvolvimento de software, como o *deploy* da aplicação. Assim, ele permite que aconteça um bom lançamento do produto, como a entrega contínua de atualizações.

Equipe de gerenciamento

- Arquiteto

  O arquiteto de software é o profissional que está envolvido diretamente com mais questões estratégicas do que técnicas de uma operação, porque busca entender como as empresas podem ter uma performance melhor por meio do desenvolvimento de sistemas e aplicações. Basicamente, entrega para a equipe de desenvolvimento o que e como será feito.

- Analista de Qualidade

  É o responsável pelo planejamento, execução e análise dos resultados da garantia da qualidade de uma determinada entrega, assegurando que os produtos de trabalho e a execução dos processos pelos projetos de

desenvolvimento de software estejam em conformidade o esperado e requisitado pelo Gerente de Produto.

- Gerente de Produto

  O gerente de produto é a pessoa que identifica as necessidades do cliente e os objetivos de negócios mais amplos que um produto ou um recurso vão suprir, articula o que seria o sucesso para um produto e reúne uma equipe para transformar a sua visão em realidade.

- Gerente de Projeto

  O gerente de projetos deve assegurar que o projeto fique dentro do escopo, do custo e do prazo acordados, monitorar os indicadores dos projetos, obter, selecionar e adquirir recursos humanos, financeiros e materiais, coordenar as partes interessadas, gerenciar conflitos, comunicar decisões e resultados. É a função da gerencia de projetos que mais pode impactar as entregas de um projeto e identificar seus problemas.

Introdução ao GIT

Git é um sistema de controle de versão de **arquivos**, isso mesmo, você pode inclusive controlar versões de imagens, planilhas e qualquer tipo de arquivo. Através deles iremos desenvolver projetos na qual diversas pessoas podem contribuir simultaneamente, editando e criando novos arquivos e permitindo que os mesmos possam existir sem o risco de suas alterações serem sobrescritas.

Para usar o Git, basta baixá-lo aqui: https://git-scm.com/downloads

Siga esse **passo a passo** para adicionar a chave SSH ao seu repositório GIT

Commit

Antes de falarmos de commits precisamente, vamos falar do `git add`: Este comando adiciona arquivos em um lugar que chamamos de INDEX, que funciona como uma área do git no qual os arquivos possam ser enviados ao reposiório. É importante saber que ADD não está adicionando um arquivo novo ao repositório, mas sim dizendo que o arquivo (sendo novo ou não) está sendo preparado para entrar na próxima revisão do repositório.
EX:

- `git add .`: Adiciona todos os arquivos alterados
- `git add "index.html"`: Adiciona somente o arquivo desejado

Agora vamos ao **commit,** que significa pegar todos os arquivos que estão naquele lugar do INDEX que o comando `add` adicionou e criar uma revisão com um ID e um comentário, que será vista por todos.

- `git commit -m "comentário qualquer"`

Note que mesmo escrito "comentário qualquer", é importante que sua mensagem de commit seja breve e clara quanto ao o que você esta incluindo no seu commit, seja uma alteração de layout, código ou documentação.

- `git status` exibe o status do seu repositório atual, como todos os arquivos alterados e separados pelos que passaram por `add`

Branches

Suponha que o seu projeto seja um site HTML simples, e você deseja criar uma nova seção no seu código HTML, mas naquele momento você não deseja que estas alterações estejam disponíveis para mais ninguém, só para você. Isso é, você quer

alterar o projeto (incluindo vários arquivos nele), mas ainda não quer que isso seja tratado como “oficial” para outras pessoas, então você cria um branch (como se fosse uma cópia espelho) e então trabalha apenas nesse branch, até acertar todos os detalhes dele.

No git, o conceito de branch é algo muito simples e fácil de usar. Mas quando que temos que criar um branch? Imagine que o seu site está pronto, tudo funcionando perfeitamente, mas surge a necessidade de alterar algumas partes dele como forma de melhorá-lo. Além disso, você precisa manter estas alterações tanto no computador de casa quanto do trabalho. Com isso temos um problema, se você começa a alterar os arquivos em casa, para na metade da implementação, e precisa terminar no trabalho, como você iria comitar tudo pela metade e deixar o site incompleto?

Para isso existe o conceito de branch, que é justamente ramificar o seu projeto em 2, como se cada um deles fosse um repositório, e depois juntá-lo novamente (que veremos mais para frente).

Use o seguinte comando de terminal para criar um branch:

- `git checkout -b branch-nova`, onde branch-nova é nome da sua branch a sua escolha.

Mudar de branch:

- `git checkout master`, onde master é o destino e nome da branch já existente.

Pushes e Requests

Além disso, temos o `git push` que é usado para publicar todos os seus commits para o **repositório remoto**. Neste momento poderá ser solicitado a sua senha:

- `git push origin master`, onde "origin” é o seu repositorio remoto e "master" é a branch que esta subindo
- `git remote -v`, comando para listar todos os repositorios remotos na sua pasta local

Cada plataforma usa seu proprio nome para os Requests, no GitHub é Pull Request, no GitLab é Merge Request, mas ambos permitem que você troque as alterações feitas no código-fonte de uma branch e colabore com outras pessoas no mesmo projeto. Com eles você pode:

- Comparar as mudanças entre duas branches.
- Revisar e discutir as modificações propostas em linha.

- Live preview das alterações.
- Resolver conflitos pela interface de usuário.
- Solicitar aprovações de seus colegas.

Docs Importantes

- https://git-scm.com/
- https://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar/
- https://docs.gitlab.com/ee/user/project/merge_requests/

Introdução ao HTML

CodeSandbox link: https://codesandbox.io/s/introducao-ao-html-czq11c

HTML é um documento de texto simples estruturado com elementos.

Elementos são acompanhados de abertura e fechamento de Tags. Cada tag começa e termina com colchetes angulares (sinal de maior e menor, `<>`). Existem algumas tags vazias ou sem conteúdo que não podem incluir qualquer texto, como por exemplo a tag `<img />` ou a tag `<input />` .

Resumidamente, **elementos** são o conteúdo visível de uma página HTML e **tags** são a representação desses elementos no arquivo HTML.

```
<html>
  <head></head>
  <body></body>
</html>
```

Head e Body

A tag `<head>` serve para colocarmos informações gerais (chamados de metadados) na nossa pagina HTML, incluindo seu título e links para scripts e folhas de estilos.

A tag `<body>` do HTML representa o conteúdo da pagina, ou melhor, toda implementação de nossa aplicação visual será declarado como filho dela.

Ambas as tags são únicas e devem aparecer sempre uma após a outra obrigatoriamente. Toda implementação deve se ater em ser filho de uma das duas partes.

Folhas de estilos (CSS)

CSS (Cascading Style Sheets ou Folhas de Estilo em Cascata) é uma linguagem de estilo usada para descrever a apresentação de um documento escrito em HTML (incluindo várias linguagens em XML como SVG, MathML ou XHTML). O CSS descreve como elementos são mostrados na tela, no papel, na fala ou em outras mídias.

Assim como o HTML, o CSS não é realmente uma linguagem de programação. Também não é uma linguagem de marcação — é uma *linguagem de folhas de estilos*. Isso significa que o CSS permite aplicar estilos seletivamente a elementos em documentos HTML.

Por enquanto é o suficiente que você precisa saber sobre CSS, falaremos mais disso nas próximas aulas.

Scripts (Javascript)

JavaScript é uma **linguagem de programação** que permite implementar funcionalidades mais complexas em páginas web. Sempre que uma página web faz mais do que apenas mostrar informações estáticas para você - ela mostra em tempo real conteúdos atualizados, mapas interativos, animações gráficas em 2D/3D, vídeos, etc. - você pode apostar que o Javascript provavelmente está envolvido.

Docs Importantes

- https://developer.mozilla.org/pt-BR/docs/Glossario/HTML
- https://developer.mozilla.org/pt-BR/docs/Web/HTML
- https://developer.mozilla.org/pt-BR/docs/HTML/Introduction
- https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element

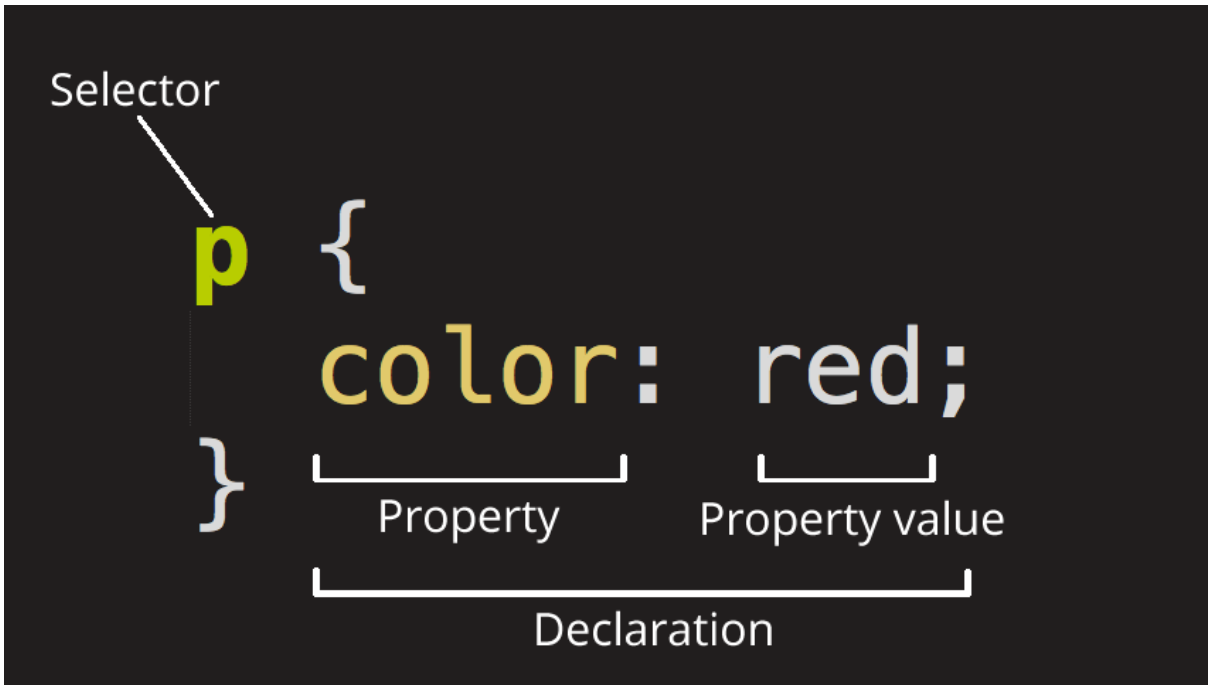
Introdução ao CSS

CodeSandbox Link: https://codesandbox.io/s/introducao-ao-css-037s8w

Como falamos anteriormente, CSS é uma linguagem de estilos que determina como os elementos aparecem numa página HTML. Para ser mais especifico, CSS pode mudar alem das cores, bordas e espaçamentos, com CSS podemos fazer animações como *fade* e até mesmo um *slide*.

Falando dessa forma ate parece que você pode fazer tudo sem Javascript, e até pode a união dos dois é na maioria das vezes o caminho melhor e mais fácil de aplicar algum **comportamento** num elemento.

Dito isso, as declarações de uma **regra CSS** possuem uma sintaxe:



Seletores

Esse ponto é o mais complexo da sintaxe do CSS, então vamos direto para a implementação mais difícil:

```
div#main input#address-name.is-open:not(:first-child) {}
```

Para começar vamos dividir isso em partes:

1. input → elemento HTML

2. #main → id do elemento div
3. input → elemento HTML
4. #address-name → id do elemento input
5. .is-open → classe do elemento input
6. :not → pseudo-classe
7. :first-child → pseudo-classe

Então se imaginarmos o elemento HTML que reproduziria o comportamento dessa regra CSS seria:

```
<div id="main">
  /* Essa tag não reproduz nada */
  <input id="address-name" class="is-open" />
  /* Essa tag reproduz a regra */
  <input id="address-name" class="is-open" />
</div>
```

Traduzindo isso tudo, `div#main` é a primeira parte (A) da regra e `input#address-name.is-open:not(:first-child)` é a segunda parte (B), ambas estão separadas por um espaço e isso significa que a regra so será aplicada se B for filho de A. Então se no nosso HTML não tivesse nenhum input, com o id “address-name”, a classe “is-open” e não fosse o primeiro filho do elemento pai, então essa regra nunca seria aplicada.

Propriedades e Valores

Como dito acima, as propriedades vão além de colocar uma cor, temos propriedades que alteram a posição e as que dão uma animação para um elemento. No começo tudo que temos a fazer é pesquisar, são realmente muitas propriedades CSS que existem e muitas outras que nem funcionam mais, porem tem algumas que nunca saem dos projetos e são usadas com frequência então vale a pena o estudo e entender melhor sobre:

- Espaçamentos
  - margin
  - padding
  - border
- Cores e fundos

    - color
    - background (e seus derivados)
- Fontes
    - font-size
    - font-family
    - font-weight
- Tamanhos
    - width (max e min)
    - height (max e min)
- Posições e renderizações
    - position
    - display
    - top, left, right e bottom (usadas em combinação de alguns valores do position)

Depois que entender um pouco mais sobre CSS e como ele funciona, usar bibliotecas CSS como Bootstrap, Tachyons e qualquer outra é simples.

Pseudo-classes

Uma **pseudo-classe** é uma palavra-chave adicionada a seletores que especifica um estado especial do elemento selecionado. No exemplo acima usamos duas pseudo-classes intercaladas, `:not()` e `:first-child`, aquela pode ser usada para especificar algum seletor indesejado na sua regra e esta é usada para indicar apenas o primeiro elemento filho.

Existem diversas pseudo-classes também mas se atenha ao conceito de que elas servem para designar estados, posições ou negações e afirmações que ficara muito mais fácil saber quando você não precisa criar uma classe para fazer algo que pseudo-classes ja fazem bem.

Docs importantes

https://developer.mozilla.org/pt-BR/docs/Web/CSS/Pseudo-classes

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Type_Class_and_ID_Selectors

https://www.w3schools.com/css/

Introdução ao Javascript

CodeSandbox Link: https://codesandbox.io/s/introducao-ao-js-4gc2yg

JavaScript (abreviado como "JS") é uma linguagem de programação dinâmica cheia de recursos que quando aplicada em um documento HTML que pode fornecer interatividade dinâmica em sites. Atualmente Javascript esta incluso na Web, Desktop, Backend e muito mais, se tornou uma linguagem de programação completamente escalável e tomou todos os cantos.

Neste curso vamos focar em HTML, então precisamos primeiramente incluir uma chamada de um arquivo Javascript e colocar um código nela. A tag `script` pode ser inserida dentro da tag `head`, porém por questões de performance, é aconselhável que fique no final do `body`, pois dessa forma temos a garantia de que o HTML que deveria existir para que o código o manipule exista.

```
<html>
  <head></head>
  <body>
    <main></main>
    <script src="./script.js"></script>
  </body>
</html>
```

```
// script.js
document.getElementsByTagName('body')[0].innerHTML = "Hello World"
```

Isso acontece porque o browser (navegador, Google Chrome, Edge, Safari entre outros) vai ler o HTML da pagina que é executada, sendo este a primeira coisa lida e entendida por ele. Após ter o HTML carregado o browser entende qual a próxima instrução a ser executada de cima para baixo, logo coisas como os arquivos CSS são lidos e processados e após isso o Javascript tambem.

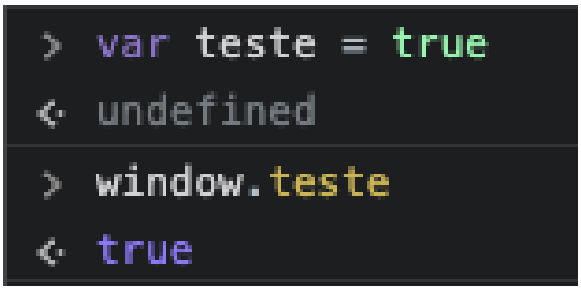
A `window`

O objeto window representa uma janela que contém um elemento DOM (Document Object Model); a propriedade document aponta para o documento DOM document carregado naquela janela.

Logo a janela do navegador é a base onde dados como o tamanho dela, ou funções como abrir e fechar uma aba, existem dentro desse objeto. Por padrão, qualquer

variável que criamos no escopo principal pode ser encontrado na window.

- `window.console` → `console.log()`
- `window.location` → `location.href`



https://developer.mozilla.org/pt-BR/docs/Web/API/Window

O `document`

Para cada página carregada no browser, existe um objeto Document, filho da `window`. A interface Document serve como um ponto de entrada para o conteúdo da Página (a árvore DOM, incluindo elementos como `<body>` e `<p>`) e provê funcionalidades globais ao documento (como obter a URL da página e criar novos elementos no documento).

`document.getElementById`

`document.querySelector`

`document.addEventListener`

https://developer.mozilla.org/pt-BR/docs/Web/API/Document

Docs Importantes

- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide