ReactJS

ReactJS é uma biblioteca em javascript, não um framework. Framework que utilizam Reactjs são o NextJS, Remix, RedwoodJS....

**Conceito de SP**

SPA foi introduzido lá pelo ano de 2011 com a primeira ferramenta o AngularJS, que foi a ferramenta pioneira nesse conceito. E o que é SPA, nada mais do que uma sigla para Simgle Page Application, que é uma forma de trabalhar com a parte visual da aplicação de uma forma diferente do conceito tradicional com paginas estaticas

SSR e SPA são padrões de renderização (Render Patterns).

**Bundlers e Compalirs**

Compalirs nada mais são do que compiladores de código, ferramentas que converte o formato do nosso codigo para outro. E o que é converte o formato de código para outra, nada mais é que tranforma o seu código atual em um que o navegador antigo/desatualizado suporte. O que são Bundlers, webpack.

Site “Can I use”...

**Vite**

Vite é uma ferramente de front-end que visa facilitar a “modularização” de arquivos java script sem a necessidade de usar ferramentas exterans com webpack. Antes dos modulos estarem disponiveis nos navegadores, os dev não tinha a opção de criar mecanismo nativos para Javascript modular.

Vamos criar um projeto usando vite, para isso vou configurar uma pasta na area de trabalho que vai conter o projeto. O exemplo fica assim:



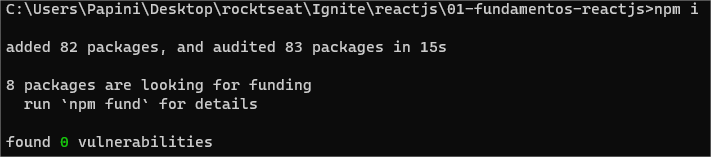
Acessando a pasta atraves do cmd, vou dar o comando que faz o download do vite, o comando fica assim:



Em seguinda deste comando, vai aparecer as configurações de instalação, nomear o projeto, em seguinda vai escolher se o projeto vai ser em JS, ou vai ter Typescript, como mostra a imagem:



Depois destas configurações, vamos instalar as dependeicas do projeto, e para isso vamos usar o comando npm i como mostra a imagem a seguir:

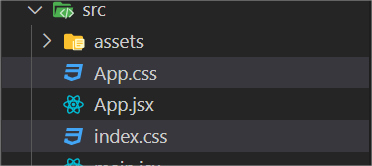


E depois só esxecutar o projeto com nosso comando npm run dev

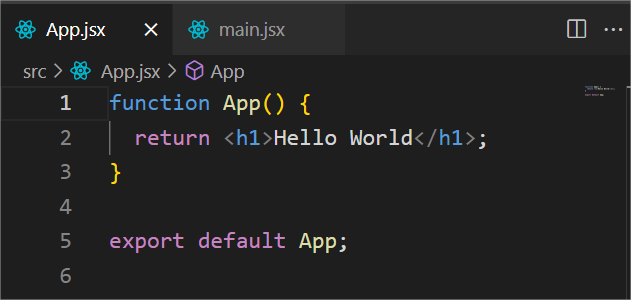


**Limpando a estrutura do projeto**

Inicialmente vamos eliminar os seguintes arquivos da pasta src, imagem abaixo para facilitar o entendimento:



A versão que estou usando é diferente das aulas, nas aulas existe outro arquivos além dos selecionados, mas por via das dúvidas os arquivos que tem que sobra na pasta src são o App.jsx e o main.jsx.

No arquivo main.jsx vamos abrir e tirar a importação do css. No arquivo App.jsx vamos retirar a importação do css e do svg, a variavel de estado, e a importação do useState(). O arquivo se encontra assim:

Explicação de como o ReactJS renderiza na tela, atraves da DOM que é Document Object Model que é a representação do nosso HTML atraves do Javascript.

**Componente**

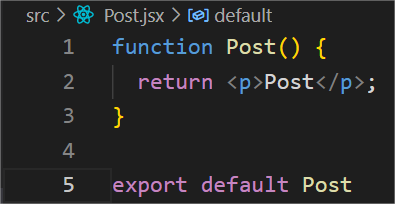
De uma forma simplista o componente é a ação de repetir enumeras vezes um trecho de codigo, quando digo esse “repetir enumeras vezes um trecho de código” quero falar com realçao em criar um pedaço de código e reutilizar esse codigo em diversas partes da aplicação, exmplo seria o header da aplicação, poderia-mos reutilizar o mesmo tanto na pagina home quanto na pagina sobre de um site, ou seja, reutilizar um trecho de código.

Uma parte da interface da aplicação que pode ser repetida varias vezes, e cada repetição que esse trecho tiver pode trazer informações diferentes, conforme a necessidade.

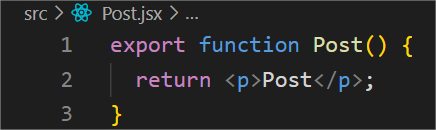
Componente em reactjs é uma função que retorna algum HTML.

JSX => JavaScript + XML (HTML), é uma arquivo que contem HTML dentro dele.

Default Export => Pode dar o nome ao componetne na importaçao e não na exportação.

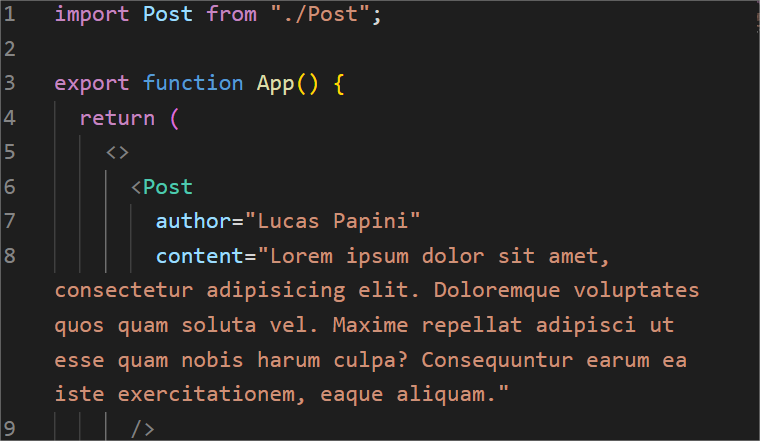


Named Export => Escrevemos a exportação juntamente na função que vai ser exportada. Segue um exemplo abaixo:

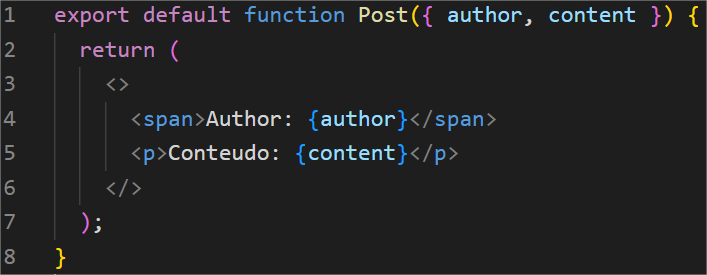


**Propriedades**

Propriedade nada mais são do que as informações que enviamos aos componentes, seria como os metodos que enviamos as funções, ou até mesmo podemos comparar com os atributos html. Para ilustra vamos adicionar as imagens abaixo:

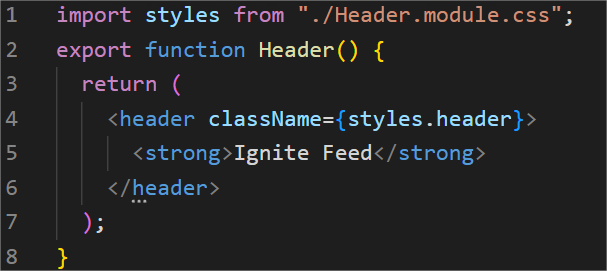
Componente recebendo as propriedades: 

Componente tratando as propriedades:

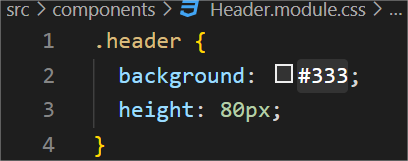


Observe que as props estão sendo desestruturadas, mas daria para usar no paramentro da função desta forma Post(props.author, props.content).

**CSS Modules**

Nesse primeiro contato com css modules, vamos adicionar o arquivo de estilo direto a um componente para exemplificar a funcionalidade. Conforme mostra imagem abaixo vamos crirar um componente e dentro do componente vamos fazer a chamada do css module, que é um pouco diferente do styled compoents.

A imagem acima mostra o componente Header criado, fazendo a chamada do arquivo css module. Para acessar a configuração do estilo é só observar a linha 4, nela estamos usando o estilo criado dentro do atributo header. A seguir a imagem do css module:



Vamos configurar o css Global para a aplicação, para isso vamos seguir os seguintes passos:

- Dentro do App.jsx coloque a seguinte importação: import ’./global.css

- Dentro da pasta SRC crie o arquivo global.css

- No global.css adicione as configurações padroes do css

1rem = 16px

0.25rem = 16px / 4

Dica de **componente**: Existe 2 grandes momentos que devemos criar um componentes, o primeiro é quando alguma coisa repete muito em tela. O segundo ponto aonde vai se criar um componente é quando consegue tirar algo de um componente maior sem que o componente maior para de funcionar, quando você consegue abstrari o componente maior.

O elemento que recebe o rem em seu font-size vai está recebendo a unidade relativa no tamanho do html da pagina.

**Iteração**

É repetir alguma coisa, criar uma estruturação de repetição, iterar sobre um array. Para iterar um objeto em ReactJs devemos usar o .map e não o forEach, pois o forEach não tem retorno.

**Estado (useState)**

Estado são basicamente variaveis, que eu quero que componente monitore.

SpredOperato: copia os valores que já existe na variavel.

Imutabilidade -> as variaveis não sofrem mutação, nós criamos um novo valor (um novo espaço na memória).

**Entendendo closures no React**

Sempre que você atualizar uma informação, e essa informação depende do valor que ela tinha anteriormente o padrão a ser usar e o a seguir:

function handleLikeComment() {

  setLinkeCount((state) => {

   return state + 1;

  });

  setLinkeCount((state) => {

   return state + 1;

  });

 }

**Typescript**

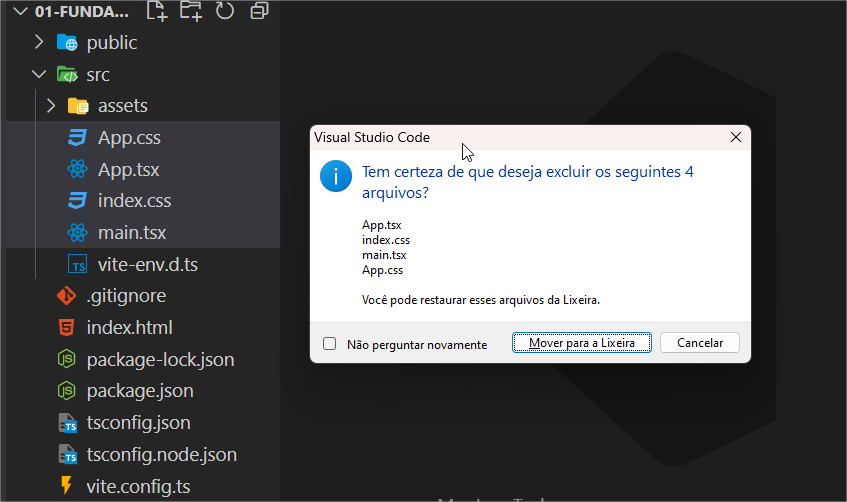
Typescript é um super set um conjunto ferramental em cima da linguagem Javascript para conseguir adicionar uma tipagem estatica em uma linguagem dinanmica.

Configurando um projeto com vite + typescript:

Primeiro vamos fazer uma instalçâo com o seguinte comando *npm create vite@latest*

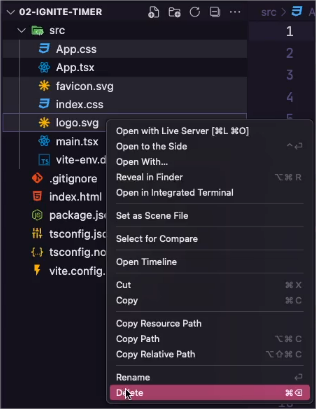
em seguida vamos apertar enter e depois dar o nome no projeto, em seguida vamos escolher qual biblioteca vamos usar e o bodyplate do typescript.

Configurando o projeto

Dentro do projeto vamos apagar quase todos os arquivos da pasta src, menos o arquivo *vite-env.d.ts*. Imagem a seguir demonstrando:

Instalando o Reactjs para o segundo projeto

Primeiro vamos na pasta que desejamos e lá iniciaremos o gitbash, no gitbash vamos dar o seguinte comando *npm create vite@lates*t latest é para pegar a última versão do reatcj. Após a instalação do projeto vamos limpar os arquivos que não usaremos.



nos arquivos main, App vamos retirar as importações que tem lá... Depois de remover as importações vamos instalar as dependecias faltantes do projeto *npm i* .

Dentro do arquivo App vamos mudar o tipo de exportação de export default para só export, segue exemplo abaixo:

para:



Na impotarção do componente no main, temos que mudar a chamada de import App para import { App }.

Styled components

Styled components é uma maneira de estilizar nossas aplicações em React utilizando um conceito chamado CSS in JS, como o próprio nome já diz é escrever nosso css com javascript. Para instalar vamos dar o seguinte comando *npm install styled-components* .

Como estamos usando typscript o styled components é uma daqules pacotes que não trazem as tipagens, então vamos ter que instalar para evitar erros. Para isso vamos dar o comando *npm i @types/styled-components -D*. INstalamos como dependecia de desenvolvimetno pq em produção nosso código vira 100% javascript e acaba não tendo necessidade de instalar sem ser uma dependencia.

Configurando o arquivo “styles.d.ts”

Esse arquivo vai conter somente codigos de definição do typescript, e nunca codigo js ou coisa assim. Esse arquivo é chamdo de “arquivo de definição de tipos” arquivos que só tem tipagem.