Frontend: É a parte do desenvolvimento web que representa as interfaces das páginas web, onde o usuário irá interagir.

Backend: É a parte do desenvolvimento web onde se concentra a maior parte das regras de negócio da aplicação, assim como efetivação das ações do usuário, como registro em banco de dados, autenticação, controle de acessos e muitas outras.

React: O **React** é a biblioteca mais popular do **JavaScript** e é usada para construir uma interface de usuário (IU). Ela oferece uma resposta excelente para o usuário adicionar comandos usando um novo método de renderizar sites e, implementa modelos como SPA (Single Page Application).

SPA: Single Page Application, é o modelo onde todas as ações da aplicação devem acontecer em somente uma página, tem como principal característica a inexistência de recarregamento e redirecionamentos na página web.

Dom: Document Object Model, é uma representação num formato Classe/Objeto usado por navegadores Web para representar as páginas Web. Como toda página na web é um documento, os navegadores precisam fazer a leitura e a interpretação desses documentos para refletir seus conteúdos na tela do seu dispositivo num formato padrão. O DOM é a interface que representa a forma ou estrutura com que o documento será lido e exibido por todos os navegadores das redes.

Virtual Dom (React Dom): Modificar o DOM é um processo demorado e aí entra o Virtual Dom. Ele é uma representação do DOM mantida em memória. Assim, quando precisamos fazer alguma alteração, ela é feita no Virtual DOM, que é bem mais rápido que o DOM. Com isso ele analisa todos os lugares que serão afetados e sincroniza com o DOM em um processo chamado Reconciliação. A vantagem disso é que essa análise permite que haja o menor número possível de acessos ao DOM, melhorando muito a performance das aplicações. Lembrando que o Virtual DOM não é algo do navegador, e sim, um conceito implementado por bibliotecas como o React.

JSX: Criado pela equipe de desenvolvimento do React, o JSX é uma forma de criar elementos para serem utilizadas como templates de aplicações React. Basicamente, os elementos criados com o JSX são bem similares com código HTML e fornecem aos desenvolvedores uma forma mais simples e intuitiva de criar os componentes de uma aplicação. Porém, apesar de muito similar ao HTML, o JSX não é interpretado pelo navegador.

React Hooks: Implementados na versão 16.8 do React. O React Hooks é uma nova proposta que irá nos permitir utilizar estado, ciclo de vida, entre outras funcionalidades, sem a necessidade de escrevermos componentes com classe. Alguns dos hooks que existem por padrão no react são: useState (Declara um novo

estado da aplicação), useEffect (efeito colateral), useRef (gera uma referência a um componente), useCallback (memoriza uma função callback que serve para evitar loops infinitos no uso do useEffect, tem a mesma estrutura que o useEffect)

Estado(state): Em termos simples, ‘estado’ é todo dado que varia com o tempo. Essa mudança de valor pode, ou não, ser fruto da interação do usuário com a aplicação.

Componentes(components): Em vias diretas, componentes é o nome que se dá a “pedaços de frontend”. Ou seja, ao invés de construir uma página gigantesca com diversas funções e itens (listas, formulários, etc), se constrói pequenos componentes com responsabilidades únicas e que podem ser reutilizados em outras páginas. Tem como principal característica de ser um JSX (Função que retorna HTML) que é interpretada pelo Render do Virtual Dom do React.

Class Component: Formato anterior a criação dos hooks do react, onde os componentes eram criados no formato de classes, onde todo o estado do componente era um atributo da classe.

Functional Component: Formato atual do react onde os componentes são criados por funções que retornam o HTML que ela renderiza, onde todo o estado do componente é declarado usando o hook useState e, demais ações que estão ligadas a estado e são essenciais para o funcionamento do react também são usadas por meio de algum hook.

Callback: Uma função callback é uma função passada a outra função como argumento, que é então invocado dentro da função externa para completar algum tipo de rotina ou ação.

Detalhamentos do React:

1. **Princípio da Imutabilidade**: O princípio da imutabilidade diz que: os elementos em React são imutáveis, ou seja, não podem mudar seus atributos uma vez que criados. Eles representam sua interface em um determinado ponto do tempo.
2. **Reconciliação**: É o processo onde o React Dom (Virtual Dom) faz toda vez que acontece uma atualização no estado da aplicação, onde o React Dom irá processar todas as atualizações que aconteceu no estado da aplicação, gera todo o HTML da página, em seguida ele compara tudo que ele gerou de novo com tudo que está atualmente renderizado para o usuário, e fará a efetivação das alterações no DOM somente nos pontos onde realmente houve uma mudança no HTML da aplicação.
3. **Renderização**: É o ato de exibir informações na tela. No react esse processo fica em responsabilidade do React Dom, onde ele aplica a reconciliação e efetivação das alterações no DOM do navegador.

Detalhamentos de alguns Hooks:

1. **useState**: Principal hook dentre os React Hooks, ele é responsável pelo estado da aplicação, partindo do princípio da imutabilidade que o react implementa, todas as mudanças que for acontecer na página devem ser declaradas anteriormente por um estado e, a declaração de um estado é disponibilizada pelo hook useState, que é uma função que recebe o estado inicial como parâmetro e retorna um array onde a primeira posição é uma variável que representa o estado em si e a segunda posição é uma função que serve para fazer a alteração do estado.
2. **useRef**: Retorna um objeto mutável que referencia a um componente durante todo o ciclo de vida desse componente, deve ser passado como propriedade no componente que se quer referenciar através da propriedade “ref”, onde o mesmo pode ser acessado pelo método “current” no objeto referenciado. Usado principalmente para evitar uma nova renderização para ações simples como por exemplo focar em um input.
3. **useEffect**: Hook que representa um efeito colateral, o useEffect é uma função que recebe dois parâmetros, o primeiro parâmetro é uma função callback que será executada toda vez que o useEffect for disparado e o segundo parâmetro é um array onde será colocado somente variáveis que quando sofrerem algum tipo de alteração devem dispara o useEffect.
   Muito cuidado ao usar o useEffect, seu uso descuidado pode gerar loops infinitos na página e renderizações infinitas, em tempo de desenvolvimento o react trava as renderizações para evitar o “crash” do navegador.
   Alguns casos importantes no uso do useEffect:

   a. O useEffect será disparado sempre que houver a primeira renderização da página ou componente onde está declarado ou quando alguma variável declarada no array do segundo parâmetro do useEffect mudar seu valor.
   b. useEffect sem o segundo parâmetro irá ser disparado a cada renderização da página ou componente onde está declarado.
   c. useEffect com o segundo parâmetro sendo um array vazio irá ser disparado somente uma vez, que é quando houver a renderização da página ou componente onde está declarado.
   d. useEffect com a função do primeiro parâmetro retornando uma outra função, também conhecido como efeito com limpeza, indica que toda vez que a página ou componente que já estão renderizados e, serão removidos do DOM ou seja quando o react for “desrenderizar” a página ou componente, o react irá executar a função que está sendo retornada no callback do useEffect. Obs.: Durante o uso do efeito enquanto a página ou componente estiver renderizado, o react não executará a função retornada no callback do efeito colateral, essa ação é semelhante a antiga função “componentWillUnmount” que era

usada nos formatos de class component onde os componentes e páginas eram classes javascript.

Fontes:

1. https://www.digitalhouse.com/br/blog/front-end-o-que-e-para-que-serve-e-como-aprender/
2. https://www.hostinger.com.br/tutoriais/o-que-e-react-javascript
3. https://www.devmedia.com.br/ja-ouviu-falar-em-single-page-applications/39009
4. https://www.hostinger.com.br/tutoriais/dom-o-que-e#O_Que_e_DOM
5. https://www.treinaweb.com.br/blog/o-que-e-dom-virtual-dom-e-shadow-dom
6. https://reactjs.org/docs/components-and-props.html
7. https://www.treinaweb.com.br/blog/o-que-e-jsx
8. https://www.zup.com.br/blog/react-hooks
9. https://reactjs.org/docs/hooks-intro.html
10. https://medium.com/@dimascyriaco_29717/entendendo-o-estado-no-react-ac1e5c32b0c0
11. https://dev.to/viniciusersouza/dissecando-react-parte-2-renderizacao-e-imutabilidade-3hh0#:~:text=Seguindo%20o%20princ%C3%ADpio%20da%20imutabilidade,um%20determinado%20ponto%20do%20tempo.
12. https://developer.mozilla.org/pt-BR/docs/Glossary/Callback_function
13. https://pt-br.reactjs.org/docs/hooks-effect.html
14. https://pt-br.reactjs.org/docs/hooks-reference.html