NEXTJS

Renderização REACT no lado do Servidor

Feito com `Marp` plugin do `VS Code`

Por Marianne Salomão ☻ ☺ Cloud Engineer na IBM - LinkedIn:

O que é o NextJS?

- O Next.js é um framework React, criado pela Vercel, que permite a construção de interfaces Web adicionando inúmeras funcionalidades em cima do React.

Um dos motivos de existir do NextJS:

- Em aplicações React tradicionais, toda a interface e chamada à API é feita pelo lado do client (browser), então quando um motor de busca ou crawler tenta indexar uma página, ira esperar que todo o Javascript e a chamada a API esteja carregada para ser feita toda construção da nossa página.
- Então essa busca retorna vazia sem informações relevantes para que a nossa aplicação seja indexada. Esse é um dos principais problemas que o Next.js procura solucionar.
- Com este framework, podemos renderizar nossas páginas no lado do servidor utilizando o SSR (Server Side Rendering).

Principais características do NextJS

Construindo aplicações utilizando o Next.js, é possível:

- Renderização estática pelo lado do servidor e do lado do client para desempenho de páginas e SEO (Melhor indexação para os motores de busca do Google);
- Construção do servidor utilizando o React para conexão com o banco de dados, já que o Next.js utiliza o NodeJs como seu interpretador Javascript padrão;

- Aplicações em React mais performática e uma melhor indexação de conteúdos pelos motores de busca do Google.
- Sistema de roteamento da aplicação muito interessante e intuitivo, baseado nas páginas do projeto (Com suporte para rotas dinâmicas);
- Suporte integrado para CSS e SASS ou qualquer biblioteca CSS-in-JS;
- Permite construir rotas de API (endpoints) com funções serveless.

- Renderização no servidor (Server Side Rendering – SSR)
- Geração de estáticos (Static Site Generation – SSG)
- CSS-in-JS (Vem com a estrutura em Styled-JSX, mas podemos utilizar qualquer outro, por exemplo: Styled Components, Emotion entre outros)
- Zero Configuration (Já tem incluso: Rotas, Hot Reloading e Code Splitting entre outros)
- Completamente extensível (Controle completo do Babel/Webpack, plugins)
- Otimizado para produção

FUNDAMENTOS NEXTJS

Mas então, o que é um Framework Web?

Um Framework Web é um sistema opinativo com estrutura e ferramentas já definidas. Eles nos ajudam no desenvolvimento rápido e seguro de aplicações.

TIPOS DE APLICAÇÕES WEB

Static Site Generation (SSG) (HTML / CSS / JS):

- Pode ser puro, você escrevendo HTML, CSS e JS ou pode usar um gerador estático, que por exemplo o GatsbyJS faz, basicamente ele pega os dados de uma API, passa durante um processo e cria os arquivos estáticos, onde no final são arquivos HTML, CSS e JS.

Vantagens do tipo Static Site Generation (SSG):

- Uso quase nulo do servidor.
- Como já foi renderizado antes os arquivos HTML, CSS e JS, o servidor não terá nenhum processo para rodar.
- Pode ser servidor em uma CDN (Content Delivery Network).
- Consequentemente terá um cache melhor, pois não precisará pegar todas vez os dados, tendo no cache retornará dele e pronto.
- Melhor performance entre todos.
- As duas opções acima explicam o motivo.
- Flexibilidade para usar em qualquer servidor.

Desvantagens do tipo Static Site Generation (SSG)

- O tempo de Build pode ser bem alto.
- Caso seu projeto tenha muitas páginas, precisará criar todas elas, essa ação é o processo do Build. Consequentemente o Build pode ser bem demorado. Algo que atrapalha e muito.
- Dificuldades para escalar em aplicações grandes.
- Você terá que criar todas as páginas antes, e se a aplicação for muito grande, você terá bastante dificuldades para controlar isso.
- Dificuldades para realizar atualizações constantes.

BUILD TIME

Data / Content / Templates

Static Site Generator

Web Site

REQUEST TIME

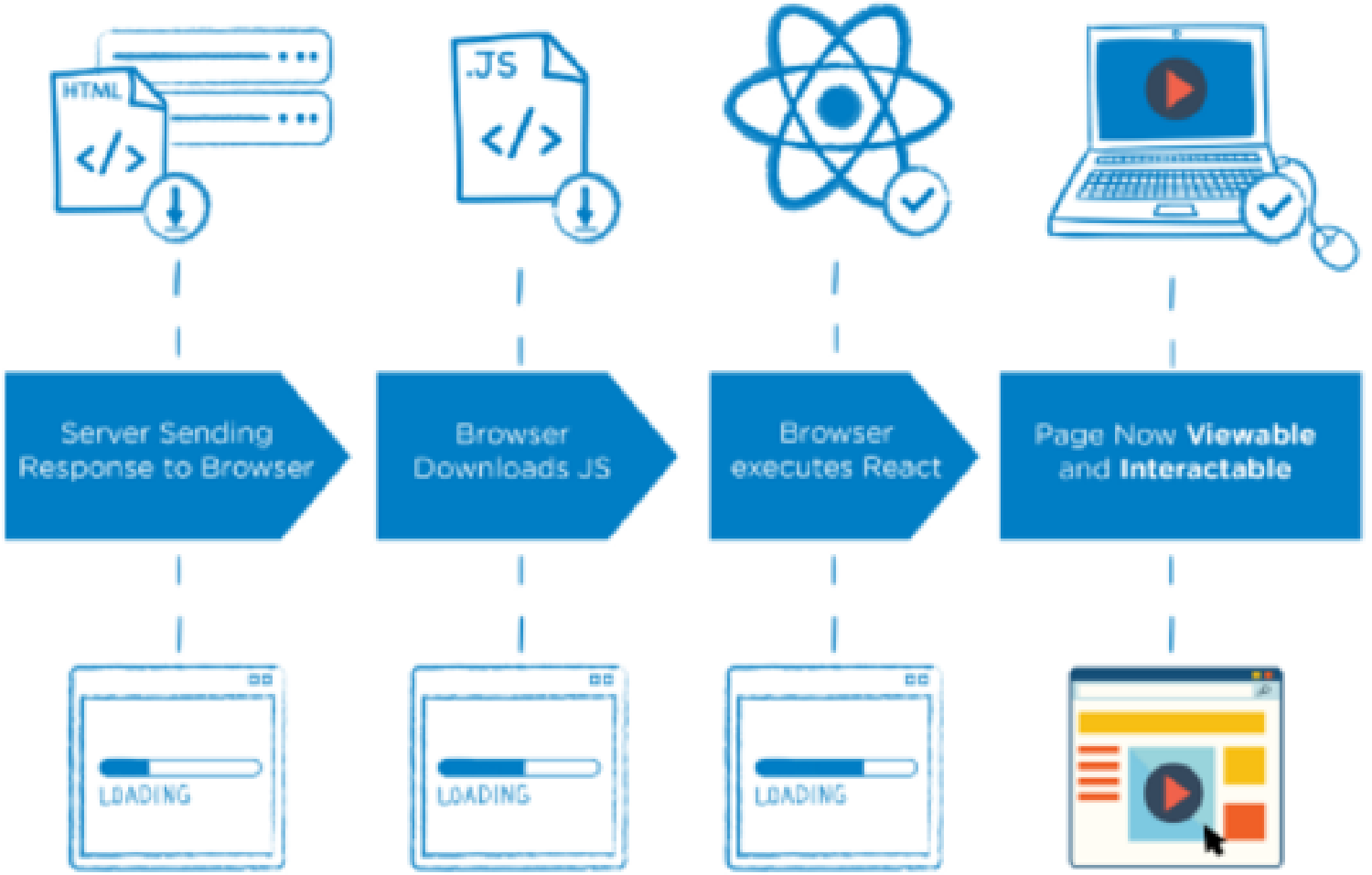
Users

Hosting

Client Site Rendering (CSR):

- Como o nome já diz, ele renderiza do lado do Client. Diferente do anterior, agora ele terá os arquivos HTML, CSS e JS. Onde o JavaScript será baixado no browser, após isso ele pegará os dados da API e vai renderizar os dados em tela.

CSR


HTML
.JS
Server Sending Response to Browser
Browser Downloads JS
Browser executes React
Page Now **Viewable** and **Interactable**
LOADING
LOADING
LOADING

Server Side Rendering (SSR):

- Ao contrário do anterior, esse renderiza os dados do lado do servidor. O usuário faz a chamada e com todos arquivos já prontos ele entrega para o Client e assim será renderizado os dados em tela.

Vantagens do tipo Server Side Rendering (SSR):

- Meta tags com previews mais adequados. As meta tags já estão com os previews certos, pois ele consegue fazer toda a parte descrita acima, que é pegar os dados como o: título, descrição, imagens e tudo mais. Logo as meta tags estarão corretas de acordo com aquela página em si.
- Melhor performance para o usuário. Isso quer dizer que o conteúdo será visto mais rápido. Como já entregamos tudo pronto e foi renderizado tudo no servidor não terá demora.

Vantagens do tipo Server Side Rendering (SSR):

- Menor processamento do lado do cliente.
- Tudo é renderizado no servidor. O browser do cliente não terá esforços para exibir o conteúdo em tela e assim tem uma vantagem para dispositivos mais fracos.

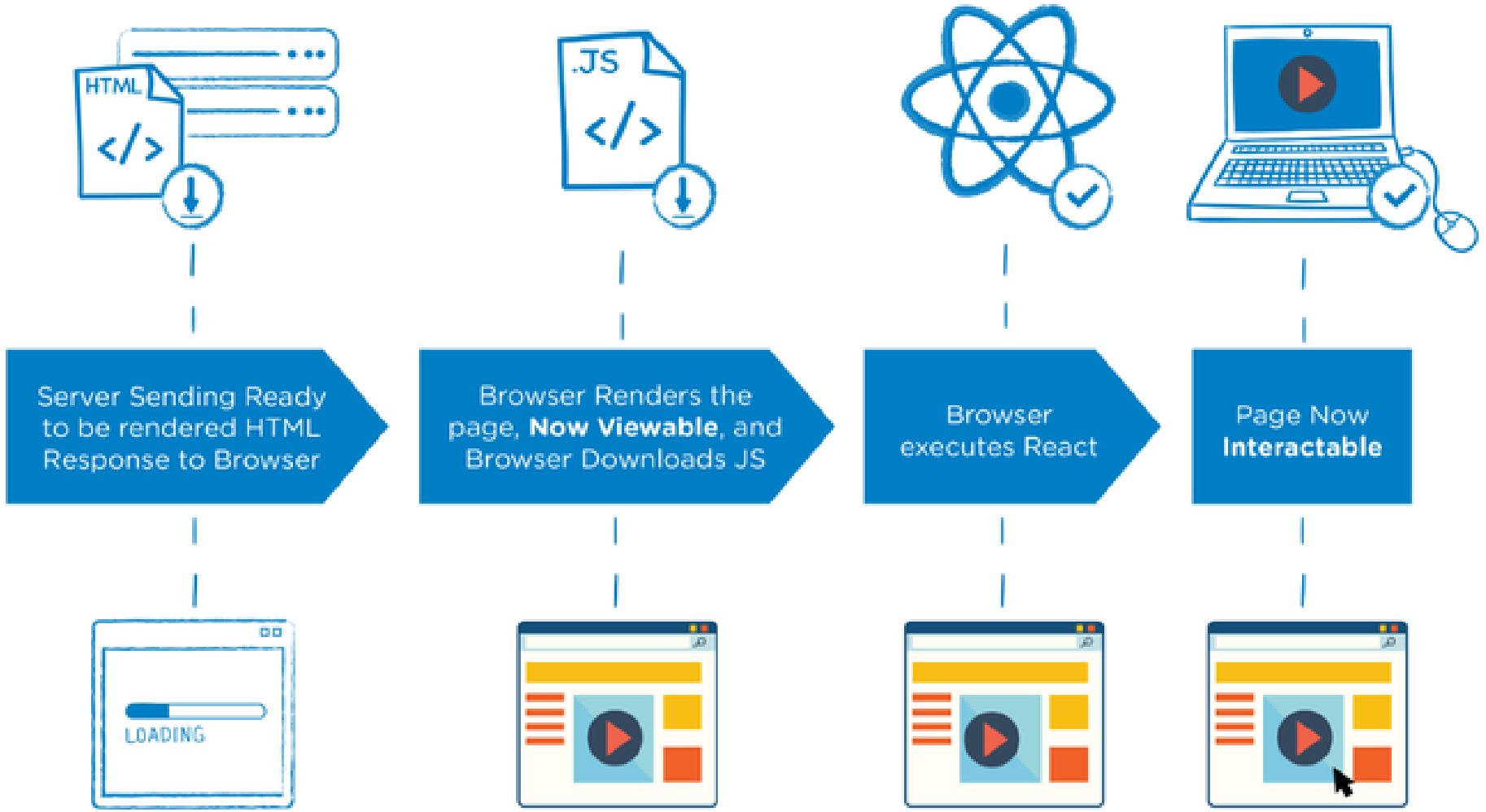
Desvantagens do tipo Server Side Rendering (SSR):

- TTFB (Time to first byte) maior.
- Acontece que o servidor irá preparar tudo antes para entregar o conteúdo. Então quando você acessa uma URL de um site, por exemplo, terá todo o processo feito antes pelo servidor.
- HTML maior.
- Já que ele entrega todo o HTML montado, consequentemente será maior. Essa podemos talvez considerar até como uma “vantagem”.

Desvantagens do tipo Server Side Rendering (SSR):

- Reload completo nas mudanças de rota.
- Toda vez que fazemos uma chamada, ele vai até o servidor e precisa recarregar para retornar o conteúdo. Porém dá para disfarçarmos esse Reload da tela criando uma situação com Micro Interação ou algo assim. Mas de qualquer forma ele precisará ir até o servidor para retornar o conteúdo.

SSR


HTML
.JS
Server Sending Ready to be rendered HTML Response to Browser
Browser Renders the page, **Now Viewable**, and Browser Downloads JS
Browser executes React
Page Now **Interactable**
LOADING

Single Page Application (SPA):

- A sigla SPA vem de Single Page Applications, ou Aplicações de Página Única, é a forma com que a página irá ser carregada.
- Estamos acostumados com aplicações onde as páginas são renderizadas do lado do servidor, independente da tecnologia utilizada. Isso traz um efeito: cada nova página que precisa ser carregada se traduz em uma nova requisição para o servidor, requisição esta para que o browser consiga carregar o HTML, o CSS e o JavaScript da nova página requisitada.

Single Page Application (SPA):

- Aplicações baseadas em frameworks SPA funcionam de maneira diferente. A aplicação seria “carregada” por inteiro na primeira requisição, onde todo o HTML, CSS e JavaScript necessários seriam carregados de uma vez. A partir deste momento, quando novas páginas precisassem ser carregadas, estas seriam carregadas através de rotinas JavaScript, retirando a necessidade de requisições para o servidor com a finalidade de obter o novo conteúdo a ser renderizado.

Vantagens do tipo Single Page Application (SPA):

- Permite páginas ricas em interações sem recarregar.
- Depois de baixado o App, tudo será feito no browser. Onde não precisará mais ficar dando reload na página.
- Site rápido após o load inicial.

Vantagens do tipo Single Page Application (SPA):

- Após feito o load inicial, foi baixado todo o JavaScript da aplicação. Depois de baixado ele já carregou todos arquivos que precisa para ser exibido em tela.
- Ótimo para Aplicações Web.
- Por exemplo: Facebook Web, Spotify Web, Twitter Web e tantos outros que utilizam o tipo SPA.
- Possui diversas bibliotecas para trabalhar.

Desvantagens do tipo Single Page Application (SPA):

- O Load inicial pode ser muito grande. Se seu arquivo JavaScript for muito grande, irá demorar mais para iniciar e ser exibido no site.
- Performance imprevisível. Pois cada página você terá arquivos JavaScript grandes e pequenos, onde acaba tendo algumas inconsistências.
- Como o arquivo JavaScript é carregado antes, para então fazer requisições na Api e assim poder renderizar. Acontece que ali você ainda não tem o conteúdo, você tem apenas o esqueleto do site e assim o Googlebot (Robô do Google) dificilmente encontrará o seu site.

Quando posso utilizar um ou outro?

Static Site Generation (Gatsby, NextJS)

- Quando a performance for muito importante.
- Site simples sem muita interação do usuário.
- Se você é a única pessoa que publica conteúdo.
- Se você faz pouca atualização no site.
- Se o site é simples com poucas páginas.

Exemplos: Portfólios, Blogs e Landing Pages.

Quando posso utilizar um ou outro?

Single Page Application (SPA)

- Quando a performance for muito importante.
- Site simples sem muita interação do usuário.
- Se você é a única pessoa que publica conteúdo.
- Se você faz pouca atualização no site.
- Se o site é simples com poucas páginas.

Exemplos: Facebook Web, Spotify Web, Twitter Web..

Quando posso utilizar um ou outro?

Server Side Rendering (SSR)

- Quando tem necessidade de um SPA, precisa de um loading mais rápido.
- Quando muda o conteúdo frequentemente.
- Quando trabalha com número muito grande de páginas.
- Quando precisa de uma boa indexação no Google.

Exemplos: E-Commerce e Sites de Notícias.

RESUMINDO TUDO ISSO...

Vimos sobre os três tipos de aplicação e entendemos o fluxogramas, vantagens e desvantagens de cada um. Entendemos que o **NextJS** suporta o **SSG, SSR e SPA**, essa é uma grande vantagem do NextJS.