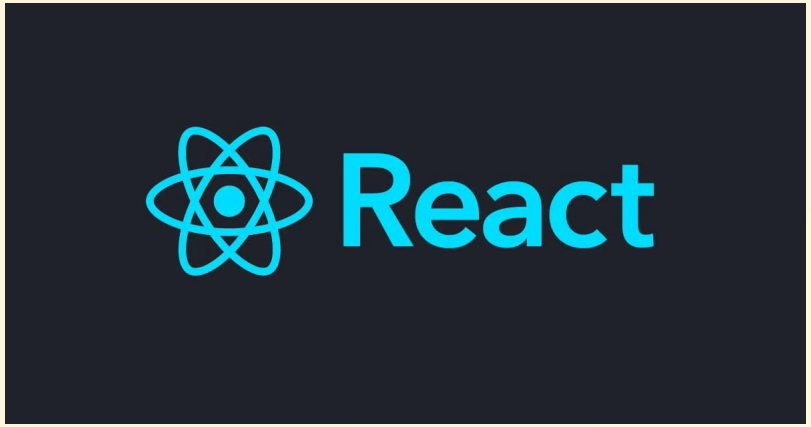
React/Next.js の最新トレンドに少しだけ触れてみる

本日のメニュー

- React とは何か
- Next.js とは何か
- React の課題について
- React Server Components が解決する React の課題
- Next.js の App Router について
- React Server Components と App Router のデモ

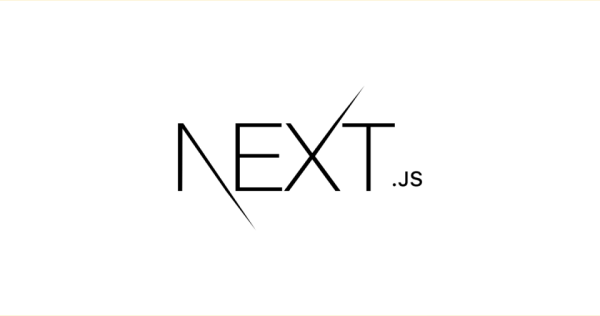
React とは何か

- React
  - UI を簡単に構築するための JavaScript ライブラリ
  - コンポーネントという概念を使ってパーツを組み立てる感覚で画面を構築できる



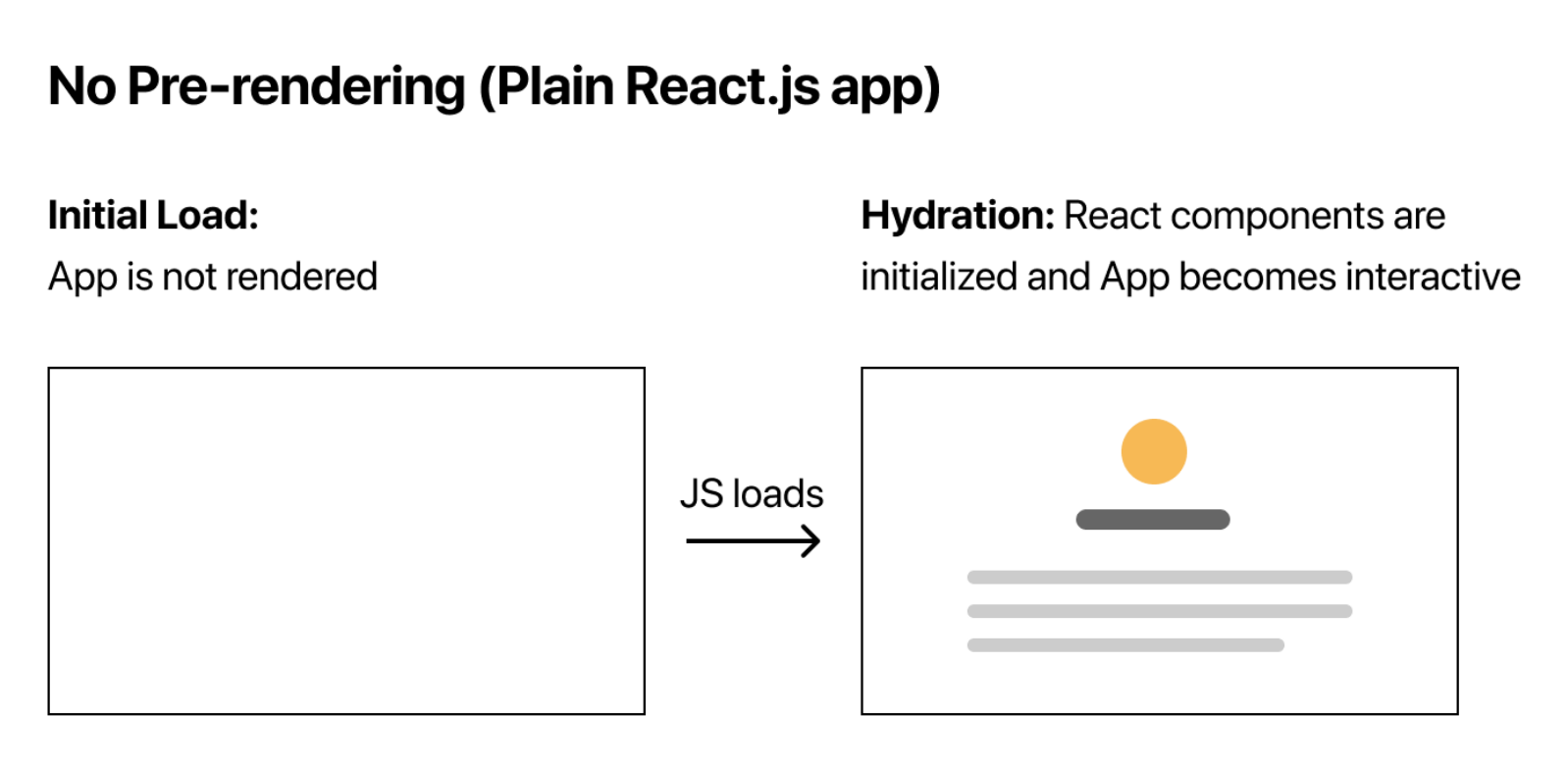
Next.js とは何か

- Next.js
  - React のフレームワーク
  - React の機能を拡張してより使いやすくしたもの
    - ex. ルーティング機能など



React の課題について

- クライアント側で全ての JavaScript をレンダリングする
  - クライアント側のパフォーマンス悪化が懸念



つまり

フロント側の負担が大きすぎる

そこで

React Server Components が登場！

React Server Components とは

- コンポーネントを「クライアント側でレンダリングされるコンポーネント」と「サーバー側でレンダリングされるコンポーネント」に分ける技術
- クライアントに送信される JS の量（クライアントでレンダリングされるコード量）が減るため、パフォーマンスの向上が期待されている

Next.js の App Router について

- Next.js には二つのモードがある
- Pages Router と App Router
- App Router が現在推奨されているモード
- App Router では、React Server Components が採用されている
- デフォルトだと、実装したコンポーネントは「サーバーコンポーネント」になる
  - Next.js の、「なるべくサーバー側に処理を寄せることで、パフォーマンスの改善を図りたい」という意図が読み取れる

React Server Components と App Router のデモ

- サーバー側で実行されるコンポーネントと、クライアント側で実行されるコンポーネント

まとめ

- React では、クライアント側の負担の増加が課題だった
- そこで登場したのが React Server Components
  - コンポーネントを「サーバーコンポーネント」と「クライアントコンポーネント」に分ける
  - クライアント側に送信する JS の量を減らすことに成功
- Next.js の App Router は React Server Components がベースになっている