Zaawansowany React - Remix, Gatsby, NextJS (SSR), PWA, Astro, Island architecture

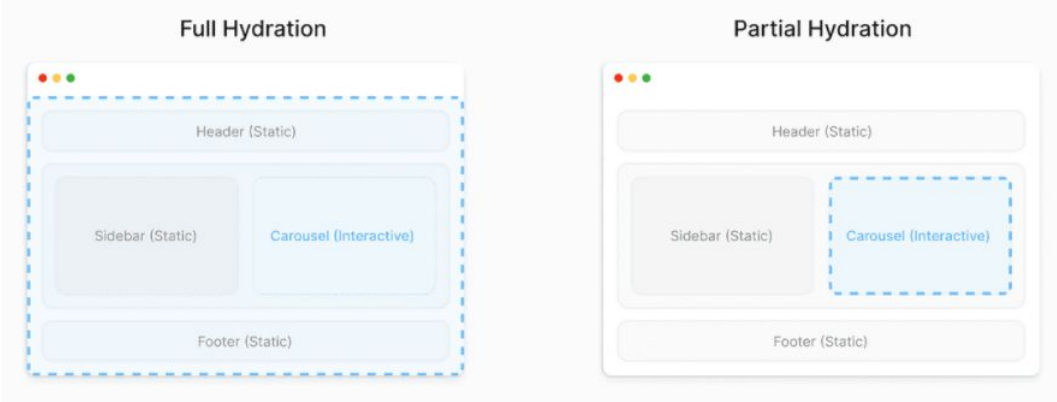
Michał Szymański
Bartosz Tonderski
Kewin Tarnowski
Konrad Siuzdak

Gatsby

- Framework oparty na React.js, który służy do tworzenia szybkich, nowoczesnych stron internetowych i aplikacji
- Wykorzystuje statyczne generowanie stron (SSG), co oznacza, że podczas procesu kompilacji tworzy statyczne pliki HTML/CSS/JavaScript dla wszystkich stron.
- Nie ma potrzeby generowania dynamicznego strony po stronie serwera, co przekłada się na szybkie czasy ładowania
- Automatycznie optymalizuje strony pod kątem wydajności. Do tego celu wykorzystuje między innymi ładowanie wstępne (preloading), łączenie plików (file concatenation), minifikację i kompresję plików.

Partial Hydration

- Pozwala na wybiórcze dodawanie interaktywności w przeciwieństwie do całkowicie statycznej aplikacji.
- Hydration - wykorzystywania client-side JS do dodawania stanu aplikacji oraz interaktywności do serwerowo HTML. Jeśli korzystamy z dużej biblioteki, która jest używana tylko podczas fazy SSR (renderowanie po stronie serwera), nie musimy jej wysyłać do klienta.
- Zwiększa miarę “Time to Interactive” przez przesyłanie mniejszej ilości JS do usera



Partial Hydration - działanie

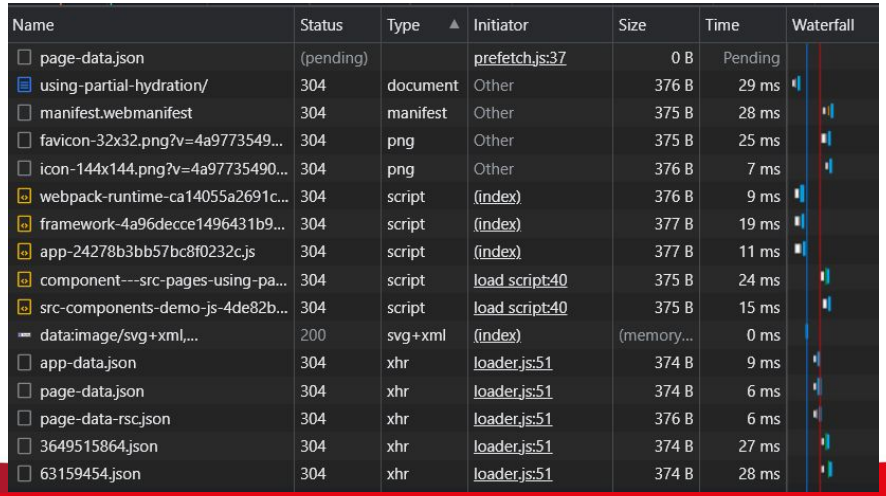
- Domyślnie komponenty w Gatsby są oznaczane jako server components, zaczynając od stron na najwyższym poziomie (np. src/pages).
- Dyrektywa *use clien*t przed definicją komponentu deklaruje go jako renderowany po stronie klienta
- Zamiast pobierania plików JS komponentu w przeglądarce, requestowane są pliki *page-data-rsc.json*. Zawierające opis UI, a komponenty po stronie klienta są dołączone jako odwołanie do bundle. Dlatego też propsy muszą być serializowalne, przez zapis do JSON. Widocznie zmniejsza to rozmiar komponentów w karcie Network.

```
"use client"

import React, { useCallback, useState } from "react"

export function Demo() {
  const [counter, setCounter] = useState(0)
  const onClick = useCallback(() => {
    setCounter(counter => counter + 1)
  }, [])

  return (
    <div style={{ marginTop: "10px", marginBottom: "10px" }}>
      <p style={{ margin: 0 }}>Current counter: {counter}</p>
      <button onClick={onClick}>Add counter</button>
    </div>
  )
}
```



Name	Status	Type	Initiator	Size	Time	Waterfall
page-data.json	(pending)		prefetch.js:37	0 B	Pending	
using-partial-hydration/	304	document	Other	376 B	29 ms	
manifest.webmanifest	304	manifest	Other	375 B	28 ms	
favicon-32x32.png?v=4a9773549...	304	png	Other	375 B	25 ms	
icon-144x144.png?v=4a97735490...	304	png	Other	376 B	7 ms	
webpack-runtime-ca14055a2691c...	304	script	(index)	376 B	9 ms	
framework-4a96decce1496431b9...	304	script	(index)	377 B	19 ms	
app-24278b3bb57bc8f0232c.js	304	script	(index)	377 B	11 ms	
component---src-pages-using-pa...	304	script	load script:40	375 B	24 ms	
src-components-demo-js-4de82b...	304	script	load script:40	375 B	15 ms	
data:image/svg+xml,...	200	svg+xml	(index)	(memory...	0 ms	
app-data.json	304	xhr	loader.js:51	374 B	9 ms	
page-data.json	304	xhr	loader.js:51	374 B	6 ms	
page-data-rsc.json	304	xhr	loader.js:51	376 B	6 ms	
3649515864.json	304	xhr	loader.js:51	374 B	27 ms	
63159454.json	304	xhr	loader.js:51	374 B	28 ms	

NextJS

- Framework React.js, oferuje wiele wbudowanych funkcji i ułatwień, które pomagają w budowaniu wydajnych, skalowalnych i nowoczesnych stron internetowych.
- Dzięki **SSR** (*server-site-rendering*), strony są generowane na serwerze przed wysłaniem ich do przeglądarki, co przyspiesza czas renderowania i poprawia indeksowalność przez wyszukiwarki.
- **CSR** (*client-site-rendering*) pozwala na interaktywność na stronie, gdy jest już widoczna w przeglądarce.
- **Dynamiczy routing** - łatwe definiowanie dynamicznych tras, które pozwalają na tworzenie stron zmiennych na podstawie danych, takich jak *id_product* czy *username*
- **Optymalizacja** przy pomocy automatycznego ładowanie (code splitting), minifikacji, buforowania statycznego i dynamicznego czy obsługi cache
- Wsparcie dla TypeScript i pluginów

NextJS- routing

- Istnieją dwie metody nawigacji między podstronami w NextJS:
  - Komponent <Link>
  - Hook useRouter()
- Link komponent rozszerza działalność taga <a> z HTML, zapewnia nawigację oraz prefetching - wczytywania trasy w tle przed jej odwiedzeniem. Renderowana wstępnie strona umieszczana jest w cache po stronie klienta. Dzięki temu nawigacja do wcześniej wczytanej trasy jest prawie natychmiastowa.

```
function MainNavigation() {
  return (
    <header className={classes.header}>
      <Link href='/'>

        <div className={classes.logo}>Next Auth</div>

      </Link>
      <nav>
        <ul>
          <li>
            <Link href='/auth'>Login</Link>
          </li>
          <li>
            <Link href='/profile'>Profile</Link>
          </li>
          <li>
            <button>Logout</button>
          </li>
        </ul>
      </nav>
    </header>
  );
}
```

NextJS- Optymalizacja

- Obrazu:
  - Rozmiaru: Automatyczna obsługa obrazy o odpowiednich rozmiarach dla każdego urządzenia, korzystając z nowoczesnych formatów obrazów, takich jak WebP i AVIF.
  - Stabilność wizualną: Automatycznie zapobiega przesunięciom układu podczas ładowania obrazów.
  - Szybsze ładowanie stron: Obrazy są ładowane tylko wtedy, gdy znajdują się w widoku przeglądarki dzięki natywnemu opóźnianiu ładowania (lazy loading), z opcjonalnymi zastępczymi wersjami rozmytymi (blur-up placeholders).
  - Elastyczność zasobów: Możliwość dynamicznego zmieniania rozmiaru obrazów, nawet dla obrazów przechowywanych na zdalnych serwerach.
- Skryptów:

```
import Script from 'next/script';
```

```
<Script src="https://example.com/script.js" />
```

  - Skrypt zewnętrzny jest pobierany, gdy użytkownik uzyskuje dostęp do danej ścieżki (np. dashboard/page.js) lub dowolnej zagnieżdżonej trasy (np. dashboard/settings/page.js). Next.js zapewni, że skrypt zostanie załadowany tylko raz, nawet jeśli użytkownik nawiguje między wieloma trasami w tym samym układzie.

Remix

- Framework React’a
- Służy do renderowania strony po stronie serwera
- Ogranicza użycie JavaScriptu po stronie klienta
- Pozwala na tworzenie jednolitych strona (za pomocą jednej aplikacji Remix)

Remix

- Zawiera nested Pages - pozwala na sprawny routing w oparciu o strukturę projektu
- Error boundaries - Jeśli wystąpi błąd strona załaduje się oprócz pojedynczego elementu, w którym błąd wystąpił
- Zoptymalizowany pod względem SEO - Search Engine Optimalization - ze względu na renderowanie po stronie serwera
- Style - umożliwia wczytanie stylów poszczególnych elementów z różnych plików css

Remix

- Dzięki ładowaniu zasobów na serwerze i wysyłaniu gotowych stron, strony w wielu wypadkach ładują się szybciej niż w przypadku next.js
- Strony statyczne oraz dynamiczne według pomiarów Remix'a są szybsze w ładowaniu zasobów
- Można wykorzystać baze danych (np.redis) do cachowania wyników do zapytań co dodatkowo może przyspieszyć wrażenia użytkownika
- Posiada możliwość preładowania strony. Wyszukując wyniku strona może zacząć ładować proponowane odpowiedzi

```
import { Form, PrefetchPageLinks } from "@remix-run/react";

function Search() {
  let [query, setQuery] = useState("");
  return (
    <Form>
      <input type="text" name="q" onChange={(e) => setQuery(e.target.value)}
      {query && <PrefetchPageLinks page={`/search?q=${query}`} />}
    </Form>
  );
}
```

Island Architecture - dlaczego?

- ładowanie skryptów JS jest wymagające wydajnościowo
- z jednej strony współczesne strony internetowe wymagają interaktywności, z drugiej spora część mogłaby być statyczna
- SEO(search engine optimization) lepiej radzi sobie ze statycznym HTML niż z JS

Island Architecture - cel

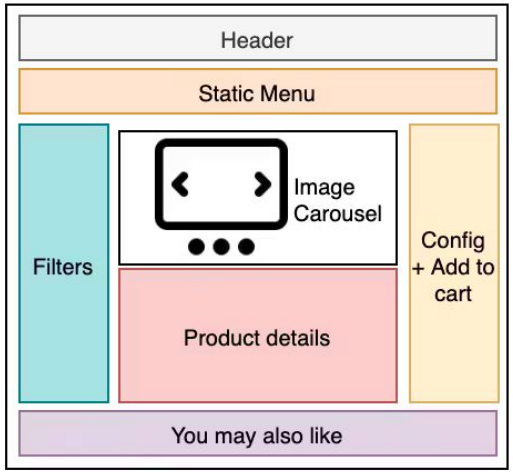
- Rozwiązanie(przynajmniej częściowe) wymienionych problemów
- Zmniejszenie objętości ładowanego JSa

Island architecture - Islands

- idea “islands of interactivity” - miejsca na stronie, które mogą być dostarczane niezależnie od reszty statycznego HTMLa
- wymagają hydratacji
- HTML + JS
- zdolne do hydratacji po renderowaniu

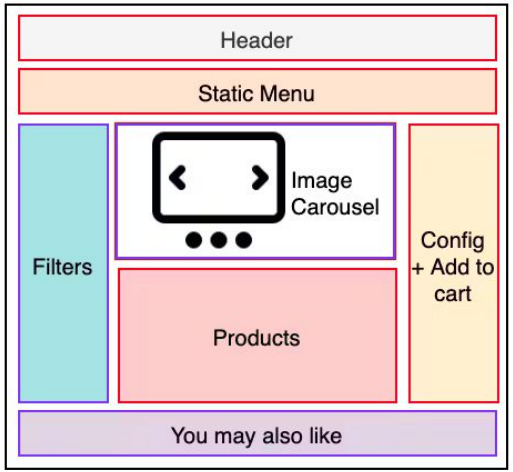
Island Architecture - porównanie

SSR



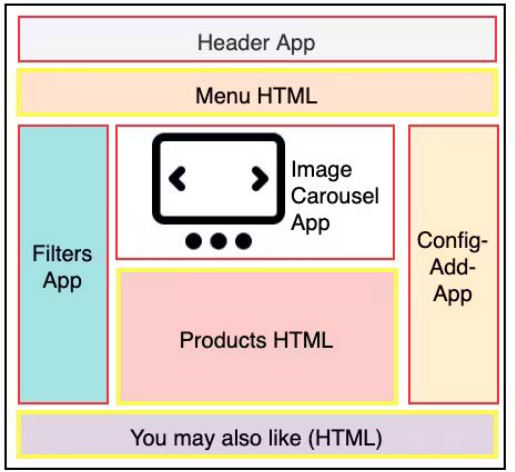
Render all components together and hydrate

Progressive Hydration



Render all components, hydrate key components first and then progressively hydrate others

Islands Architecture



Static components are server rendered HTML. Script is required only for interactive components

Island architecture - frameworki

- Marko
- **Astro**
- Eleventy + React

Astro - content focused

content rich(marketing, publishing, documentation, blogs, portfolios, ecommerce),

not web application(logged-in dashboards, inboxes, todo lists)

Astro - server first

- SSR over CSR
- MPA - MultiPageApp
- performance-oriented
- first-load performance is essential - TTI

Astro - easy to use

React, Preact, Svelte, Vue, Solid, Lit, and several others are all supported for creating new UI components in an Astro project

```
---
// Example: Mixing multiple framework components on the same page.
import MyReactComponent from '../components/MyReactComponent.jsx';
import MySvelteComponent from '../components/MySvelteComponent.svelte';
import MyVueComponent from '../components/MyVueComponent.vue';
---
<div>
  <MySvelteComponent />
  <MyReactComponent />
  <MyVueComponent />
</div>
```

PWA - Progressive Web App

Działanie:

Po wejściu na stronę typu PWA, zawartość aplikacji zapisywana jest w pamięci urządzenia. Dzięki zapisanym danym, ponowne wejście na stronę PWA może odbyć się również w trybie offline. To stanowi przewagę stron PWA nad innymi rodzajami stron internetowych.

PWA zostały zaprojektowane w celu dostarczania użytkownikom podobnego do aplikacji mobilnych doświadczenia, które można uzyskać za pośrednictwem przeglądarki internetowej.

PWA - atrybuty

- Responsywność
- Niezależność od łączności
- Bezpieczeństwo
- Aktualność
- Szybkość
- Reaktywność
- Możliwość powiadomień typu Push
- Możliwość instalacji

Porównanie PWA i aplikacji mobilnych

	Aplikacje PWA	Aplikacje Mobilne
Posiadają responsywny projekt	TAK	TAK
Możliwość bardzo szybkiej instalacji	TAK	NIE
Mogą wysyłać powiadomienia typu Push	TAK	TAK
Automatycznie aktualizowane poprzez mechanizm Service Worker	TAK	NIE
Rozpoznają lokalizację użytkownika	TAK	TAK
Konieczność pobrania	NIE	TAK
Możliwość podlinkowywania podstron	TAK	NIE
Działają w trybie offline	TAK	TAK
Działają na wszystkich systemach	TAK	NIE