

Program szkolenia:

Budowa nowoczesnych aplikacji w Next.js

Informacje:

| | |
|----------------------|--|
| Nazwa: | Budowa nowoczesnych aplikacji w Next.js |
| Kod: | react-next-js |
| Kategoria: | React |
| Czas trwania: | 3 dni |
| Forma: | 30% wykłady / 70% warsztaty |

Szkolenie przeznaczone jest dla programistów, którzy mają przynajmniej podstawową wiedzę z React.js i chcą rozszerzyć swoje umiejętności o budowanie bardziej złożonych aplikacji internetowych. W trakcie szkolenia uczestnicy poznają sposoby budowy skalowalnych aplikacji, które renderowane są po stronie serwera dbając jednocześnie o wydajność, jak i świetny User- Experience.

Zalety szkolenia:

- opanowanie i uporządkowanie najważniejszych zagadnień Next.js
- zrozumienie różnych sposobów renderowania aplikacji oraz ich optymalizacji
- wykorzystanie technik i dobrych praktyk używanych w komercyjnych projektach

Szczegółowy program:

1. Introduction to Next.js

- 1.1. Main advantages, use cases and evolution in the web development ecosystem
- 1.2. Differences between monoliths, frontend applications and Next.js
- 1.3. Methods of generating and serving content
- 1.4. Project structure and core principles
- 1.5. Differences between Next.js versions 12, 13, and 14.
- 1.6. Community and ecosystem around Next.js (plugins, tools)

2. Building Blocks

- 2.1. Dev server
- 2.2. Routing
- 2.3. Pages
- 2.4. Layouts
- 2.5. Rendering
- 2.6. Components
- 2.7. API Routes
- 2.8. Error Handling
- 2.9. Data fetching
- 2.10. Cache
- 2.11. Image Optimization

3. Routing

- 3.1. App Router
- 3.2. Defining Routes structure
- 3.3. Changes regarding the Pages Router

3.4. Route groups and dynamic Routes

3.5. Route handlers

3.6. Middleware and its role

3.7. Pages

4. Rendering

4.1. Static Rendering

4.2. Dynamic Rendering

4.3. Dynamic Pages

4.4. Streaming

4.5. Incremental Static Regeneration (ISR)

4.6. Hybrid Pages - SSR and SSG in the same application

4.7. Layouts

4.8. Nested layouts

4.9. Error pages

5. Server and Client Components

5.1. Motivation beyond two types of components

5.2. "use" directive

5.3. How Server Components work under the hood

5.4. Server Components lifecycle and use cases

5.5. Client components

5.6. Drawbacks and limitations

5.7. Compositions Patterns

5.8. Server actions

6. Fetching, Cache and Revalidation

6.1. Different methods of data retrieval

6.2. Fetching

6.3. Caching

6.4. Revalidating

6.5. Fetching Patterns

6.6. Mutations

6.7. Handling errors - best practices

6.8. Integration with headless CMS

7. Caching

7.1. Automatic Static Optimization, Static Site Generation, Static Rendering

7.2. Caching individual routes and Suspense boundaries

7.3. Server Component Payload (SCP)

7.4. Client component and SCP

7.5. Remote cache

7.6. Hydration and reconciliation

7.7. Router cache

7.8. Static and dynamic rendering

7.9. Cache invalidation

7.10. Trade-offs between different caching strategies

8. Good practices

8.1. React Context in Client Components

8.2. Images, Fonts and Styling

8.3. Builded functions (<https://nextjs.org/docs/app/api-reference/functions>)

8.4. React Query

8.5. GraphQL

8.6. Optimizations

9. Testing

9.1. What and how should be tested in Next.js applications?

9.2. Organization of testing environment

9.3. Unit testing

9.4. Integration tests

9.5. Mock API

9.6. Visual Regression Testing

9.7. E2E testing

10. Configuration and Deployment

10.1. Configuration and env variables - common use cases

10.2. Deployment and monitoring - common use cases