Web Application Development

Session 11 – How Next.js Works

Outcomes

- In this lesson you will learn:
  - From Developer to Production
  - Compiling
  - Minifying
  - Bundling
  - Code Splitting
  - Build Time vs. Runtime
  - Client and Server
  - Rendering
  - CDN’s and the Edge

Development and Production Environments

- During development, the application is being built and run on the local machine.
- Going to production is the process of making the application ready to be deployed and consumed by users.

Development and Production Environments

- **The development stage** – Next.js optimises the developer experience with features (TypeScript, ESLint integration, Fast Refresh).
- **The Production stage** - Next.js optimizes for the end-users, and their experience using the application. It aims to transform the code to make it perform and accessible.

Development and Production Environments

- **Compiler** – Next.js handles much of these **code transformations** and underlying infrastructure to make it easier for the application to go to production.

Development and Production Environments

- Next.js has a compiler written in Rust, a low-level programming language, and SWC, a platform that can be used for **compilation, minification, bundling**.

Compiling

- Developers write code in languages that are more developer-friendly such as JSX, TypeScript, and modern versions of JavaScript.
- While these languages improve the efficiency and confidence of developers, they need to be compiled into JavaScript before browsers can understand them.

Compiling

- Compiling refers to the process of taking code in one language and outputting it in another language or another version of that language.

Minifying

- Developers write code that is optimized for readability by people.
- This code might contain extra information that is not necessary for the code to run, such as comments, spaces, indents, and multiple lines.

Minifying

- Minification is the process of removing unnecessary code formatting and comments without changing the code's functionality.
- The goal is to improve the application's performance by decreasing file sizes.

Bundling

- Developers break up applications into modules, components, and functions that can be used to build larger pieces of their application.
- Exporting and importing these internal modules, as well as external third-party packages, creates a complex web of file dependencies.

Bundling

- Bundling is the process of resolving the web of dependencies and merging (or ‘packaging’) the files (or modules) into optimized bundles for the browser, with the goal of reducing the number of requests for files when a user visits a web page.

Code Splitting

- Developers usually split their applications into multiple pages that can be accessed from different URLs.
- Each of these pages becomes a unique entry point into the application.

Code Splitting

- Code-splitting is the process of splitting the application’s bundle into smaller chunks required by each entry point.
- The goal is to improve the application's initial load time by only loading the code required to run that page.

Build Time and Runtime

- Build time (or build step) is the name given to a series of steps that prepare your application code for production
- The goal is to improve the application's initial load time by only loading the code required to run that page.

Build Time and Runtime

- At Build Time - Next.js will transform code into production-optimized files ready to be deployed to servers and consumed by users.
- These files include:
  - HTML files for statically generated pages
  - JavaScript code for rendering pages on the server
  - JavaScript code for making pages interactive on the client
  - CSS files

Build Time and Runtime

- Runtime – This refers to the period of time when the application runs in response to a user's request,
- After the application has been built and deployed.

Client and Server

- Client refers to the browser on a user's device that sends a request to a server for the application code.
- Then turns the response it receives from the server into an interface (Web page) the user can interact with.

Client and Server

- Server refers to the computer in a data center that stores the application code, receives requests from a client, does some computation, and sends back an appropriate response.

Rendering

- Convert the code written in React into the HTML representation of the UI.
- This process is called rendering.

Rendering

- Next.js pre-renders every page by default.
- Pre-rendering means the HTML is generated in advance, on a server, instead of having it all done by JavaScript on the user's device.

CDN’s and the Edge

- The Next.js application code can be distributed to origin servers, Content Delivery Networks (CDNs), and the Edge.

CDN's and the Edge

- Origin Servers - Refers to the main computer that stores and runs the original version of the application code.

CDN's and the Edge

- **Origin Servers** - Refers to the main computer that stores and runs the original version of the application code.
- **Different** from **CDN Servers**

CDN's and the Edge

- **CDN's (Content Delivery Networks)** – CDN's store static content (such as HTML and image files) in multiple locations around the world and are placed between the client and the origin server.
- When a new request comes in, the closest CDN location to the user can respond with the cached result.

CDN's and the Edge

- **The Edge -** is a generalized concept for the fringe (or edge) of the network, closest to the user.
- CDNs could be considered part of "the Edge" because they store static content at the fringe (edge) of the network.

CDN's and the Edge

- **The Edge**
- Edge servers can run small snippets of code.
- This means both caching and code execution can be done at the Edge closer to the user.

CDN's and the Edge

- **The Edge**
- By moving some work that was traditionally done client-side or server-side to the Edge, the application can perform better because it reduces the amount of code sent to the client, and part of the user's request does not have to go all the way back to the origin server
- The Goal is to reduce latency.

Reference

Next.js. (2023, Jun 30). What is Next.js. Retrieved from Learn Next.js: https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs