Handout – Next.js Course: Session 2 Recap

📅 August 9th - Session Details

- **Video Recording:** Watch on YouTube
- **Code from Live Session:** GitHub Commit
- **Slides:** View Slides (PDF)
- **ISR Diagram:** Link GitHub

🎯 Learning Objectives for Session 2

By the end of this session, you should be able to:

1. Apply **best practices** when creating client components.
2. Understand **pros and cons** of using Next.js.
3. Work with **forms** and **data fetching**.
4. Use **Tailwind CSS** and **CSS Modules** without style conflicts.
5. Recognize when to choose **server** vs **client** components.

📌 Recap from Session 1

Before diving in, here's what we covered previously:

- Setting up a Next.js project and understanding folder architecture.
- Server vs client components basics.
- Fetching data and creating simple routes.
- Using CSS modules to avoid style leaks.

🚀 Pros & Cons of Next.js

Pros

- Excellent performance and load times.
- Great developer experience with built-in features.
- Highly scalable and easy to deploy.
- Large ecosystem and library support.
- SEO-friendly by default.

Cons

- Steeper learning curve compared to plain React.
- Opinionated structure, but still allows autonomy.

📚 *Further Learning:* Neil Cummings' Complete Guide to Building a Full-Stack App with Next.js

---

🛠️ Best Practices for Client Components

1. **Make client components the leaves**
   Client components should be as low-level as possible to keep performance benefits from server components.
2. **Avoid importing server components into client components**
   This breaks the separation and can cause unexpected behavior.
3. **Use a client component if you need:**
   - Event handlers (`onClick`, `onChange`, etc.)
   - State management (`useState`, `useReducer`)
   - Real-time interactions that happen in the browser.
4. **Fetching data**

    ○ Prefer fetching data in server components for performance, unless the data is user-specific and must be fetched client-side.

📚 *More on React Server Components:* [Moon Highway React Server Components Repo]

🎨 Styling with Tailwind and CSS Modules

- Use **Tailwind CSS** for utility-first rapid styling.
- Use **CSS Modules** for scoped, reusable styles.
- Avoid mixing styles from different modules in one component to prevent conflicts.

🔄 Next Built-in with Fetch - Incremental Static Regeneration (ISR) with `revalidate`

How it works:

1. When a page is requested, Next.js first checks the cache.
2. If the cached version is still fresh (within the `revalidate` time), it's served instantly.
3. If the cache is stale, the stale page is still served instantly while a background process fetches updated data.
4. Once fetching completes, the cache is updated for the next visitor.

This provides **fast load times** and **automatic updates** without a full rebuild.

✅ Things to Practice Before Next Session

- Refactor a component to follow client/server best practices.
- Style a component with both Tailwind and a CSS module without conflicts.
- Experiment with ISR by adding `revalidate` to your data fetching.