







**Notes on Project Organization in Next.js**

* **Organizing Components**
  + **Co-locating components**: Keeping a component inside its related page folder.
  + **Issue with co-location**: If needed elsewhere, a central **components folder** is still required, leading to an inconsistent structure.
  + **Preferred approach**: A dedicated **\_components** folder to store reusable UI elements while opting it out of routing.
* **Next.js Routing and Private Folders**
  + Any folder in **app/** can become a route unless opted out.
  + **Fix for unwanted routes**: Prefix folder names with \_ (e.g., \_components) to make them private.
  + Private folders do not create routes and remain at the top of the file structure.
* **Creating Additional Folders**
  + **\_styles/** → Contains global styles (e.g., globals.css).
  + **\_lib/** → Stores helper functions like API connections (e.g., Supabase functions).
  + **public/** → Stores static assets (e.g., images, logos).
* **Alternative Structure**
  + Option to create a **src/** folder at the root to store **app/** and **\_components/**.
  + Avoided for simplicity in small to medium projects.
* **Import Aliases in Next.js**
  + Prevents long relative imports (../../components/...).
  + Uses @/ as an alias for the root (@/app/...).
  + Simplifies imports across deeply nested structures

**Tailwind CSS Setup in Next.js**

* **Tailwind Installation**
  + Installed automatically during create-next-app setup.
  + tailwind.config.js file is already generated.
* **Activating Tailwind**
  + Import globals.css in the layout component:

js

CopyEdit

import "@/styles/globals.css";

* + globals.css contains Tailwind’s base styles (@tailwind base;, @tailwind components;, @tailwind utilities;).
* **Applying Tailwind Styles**
  + Uses **utility classes** directly in className prop.
  + No need for separate CSS files.
* **Example Styling**
  + Background color: bg-blue-800
  + Text color: text-gray-50
  + Minimum height: min-h-screen
* **Customizing Tailwind Theme**
  + Defined custom colors from colors.json in tailwind.config.js.
  + Used extend property to add new color values.
  + Example usage:

css

CopyEdit

bg-primary-950

text-primary-100

* **Tailwind IntelliSense**
  + Provides auto-completions for Tailwind classes.
  + Requires **Tailwind CSS IntelliSense** extension in VS Code.
* **Installing Heroicons**
  + Run:

sh

CopyEdit

npm install @heroicons/react

* + Used for adding icons with Tailwind styling.
* **Next.js Metadata Handling**:
  + Metadata, like the **title** and **description**, should be exported using a **metadata object** to avoid hardcoding in the HTML head.
  + **Title**: Exporting a **title** from a layout file sets the page title for all pages, unless overridden by individual pages.
    - **Example**: A global title "The Wild Oasis" appears on all pages by default.
    - To override, **metadata** can be defined on individual pages, e.g., on the **Cabins** page: title: 'Cabins'.
  + **Flicker Issue**: Sometimes, the title flickers to a default (like localhost) while the page loads, which may be a **Next.js bug**.
  + **Advanced Title Configuration**:
    - Use **template strings** in the metadata for more flexibility.
    - **Example**: { template: '%s - The Wild Oasis' }, where %s is replaced by the page-specific title.
    - If no title is defined, a **default title** like "Welcome - The Wild Oasis" is used.
  + **Description for SEO**:
    - Description can be defined in the **metadata object** (e.g., description: 'Luxurious cabin hotel located in the Italian Dolomites').
    - This automatically becomes the **meta description tag** in the HTML head.
    - If not overridden on individual pages, the global description is used.
  + **Favicon Setup**:
    - A **favicon.ico** or image file named **Icon** should be placed in the **app folder** (top-level folder).
    - The favicon image can be in any format (e.g., **PNG**), and Next.js automatically uses it without extra code.
  + **Next.js Font Handling**:
    - Next.js simplifies the process of adding custom fonts to the app.

[End of Notes]

* **Next.js Font Optimizations**:
  + Next.js includes **performance optimizations for fonts** out of the box.
  + Fonts can be **self-hosted** to improve **performance** and **privacy**, avoiding external server requests.
  + Self-hosting prevents **layout shifts** and enhances **page load speed**.
* **Font Hosting from Google**:
  + Google fonts can be **self-hosted** to improve website performance.
  + Using Google fonts directly from Google servers can lead to **privacy issues** (e.g., GDPR violations) and **slower load times**.
* **Steps to Add Google Fonts**:
  + **Import the Font**:
    - Use Next/font/Google to import fonts (e.g., **Josefin Sans**).
  + **Configure the Font**:
    - Define **subset** (e.g., latin for English).
    - Use the **display property**: swap displays text in a default font until the custom font loads.
    - Specify **font weight** if needed (optional for **variable fonts**).
  + **Apply the Font**:
    - The imported font returns a **class name** to be applied to the HTML body or specific elements.
    - Apply the font to the body or specific tags (e.g., headings).
* **Example Code**:

javascript

CopyEdit

import { Josefin\_Sans } from 'next/font/google';

const josefin = Josefin\_Sans({

subsets: ['latin'],

display: 'swap',

});

export default function Home() {

return (

<div className={josefin.className}>

<h1>Welcome</h1>

</div>

);

}

* **Benefits of Self-Hosting Fonts**:
  + **Improves performance** by reducing external HTTP requests.
  + **Enhances privacy** by avoiding external server communication.
  + Prevents **layout shifts** for better user experience.

[End of Notes]

**Notes on Improving Navigation and Root Layout in Next.js**

**Enhancing Navigation**

* Current navigation is an unstyled list of links.
* Replacing it with a pre-styled navigation component from starter files.
* Updating <a> elements to Next.js <Link> for proper routing.
  + **Syntax:** import Link from "next/link";
* The new navigation has:
  + Bigger text
  + Side-by-side arrangement
  + Improved styling

**Updating the Root Layout**

* **Header Component:**
  + Replaces manual <header> setup.
  + Imports logo and navigation.
  + Adds a subtle bottom border.
* **Removing the Footer:**
  + Not needed in the current design.
* **Centering Page Content:**
  + Setting max-w-7xl (2080px width) for main content.
  + Applying mx-auto to center it in the viewport.
  + Adding spacing:
    - Horizontal padding: px-8
    - Vertical padding: py-12

**Ensuring Full-Height Layout**

* **Issue:** Content does not occupy the full height of the viewport.
* **Solution:**
  + Wrap <main> in a parent <div>.
  + Make <body> a flex container (flex flex-col).
  + Set the inner <div> to flex-1 so it fills remaining space.
  + Now, the header stays fixed, and the main content expands dynamically.

**Final Adjustments**

* Ensuring layout consistency with predefined padding values.
* Adding the **antialiased** class for improved font rendering.

**Next Steps**

* Improve image performance in Next.js.

[End of Notes]

**Optimizing Images in Next.js**

* **Importance of Image Optimization**
  + Images significantly impact **page size** and **loading speed**.
  + Next.js provides an **Image component** to optimize images automatically.
* **Next.js Image Component (next/image)**
  + Replaces the native <img> tag.
  + Provides several optimizations **out of the box**:
    - **Automatic format conversion** (e.g., WebP).
    - **Responsive image sizing** using srcset.
    - **Prevents layout shifts** by requiring **explicit width and height**.
    - **Lazy loading** (loads images only when they enter the viewport).
* **Basic Usage**
  + Import the component:

jsx

CopyEdit

import Image from "next/image";

* + Replace <img> with <Image>:

jsx

CopyEdit

<Image src="/logo.png" width={128} height={128} alt="Logo" />

* + The srcset attribute ensures **responsive image loading**.
* **Alternative Usage: Static Image Import**
  + Import images directly into the component:

jsx

CopyEdit

import logo from "../public/logo.png";

* + Use the imported image:

jsx

CopyEdit

<Image src={logo} alt="Logo" />

* + This method allows **Next.js to analyze the image** automatically.
  + **No need to specify width and height**, but may affect resizing.
* **Adjusting Image Quality**
  + **Set quality prop** to control compression:

jsx

CopyEdit

<Image src={logo} quality={10} alt="Low-Quality Logo" />

* + Lower quality reduces file size but may cause blurriness.
  + Default quality is **not necessarily 100%**.
* **Benefits of Using next/image**
  + Eliminates **manual image optimization steps**.
  + Reduces **page load times** and improves **SEO**.
  + Provides **advanced image handling** with minimal effor

**Building the Homepage in Next.js**

**1. Setting Up the Page**

* Open page.js in the root **app** folder.
* Replace the existing JSX with the new homepage structure.
* **Ensure the button is a proper link** to navigate to the cabins page.

**2. Optimizing the Background Image**

* **Use Next.js Image component** instead of <img> for automatic optimization.
* Import the image statically:

jsx

CopyEdit

import bg from "../public/bg.png";

* Replace the <img> tag with:

jsx

CopyEdit

<Image src={bg} fill alt="Background Image" />

* The fill property makes the image **occupy the entire parent element** responsively.

**3. Styling the Image**

* Use **Tailwind CSS** for object fitting:

jsx

CopyEdit

className="object-cover"

* + Equivalent to CSS:

css

CopyEdit

object-fit: cover;

* Ensures the image **scales properly without distortion**.

**4. Improving the Image Behavior**

* Prevent unwanted image shifting:

jsx

CopyEdit

className="object-top"

* + Equivalent to CSS:

css

CopyEdit

object-position: top;

* + Keeps the image **anchored to the top** as the viewport resizes.
* Enable **blur placeholder** for smooth loading:

jsx

CopyEdit

placeholder="blur"

* + Shows a blurred version before the full image loads.
* Adjust **image quality** to save bandwidth:

jsx

CopyEdit

quality={80}

* + Reduces file size while keeping **visual fidelity**.

**5. Fixing Console Warning for fill Property**

* **Ensure parent container has position: relative**:

jsx

CopyEdit

className="relative"

* + Required when using fill to make the image **expand properly**.

**6. Next.js Image Optimization in Action**

* **On different screen sizes**, Next.js dynamically adjusts:
  + **Resolution** (e.g., 1000px vs. 3000px).
  + **File size** (e.g., 45KB instead of 7MB).
* This **saves bandwidth** and improves **performance**.

**7. Final Thoughts**

* The homepage **looks professional and polished**.
* Optimized image **reduces load time** and **improves UX**.
* Ready to move on to building the **Cabins** and **About** pages.

[End of Notes]

**Cabins Overview Page:**

* Display all cabins with individual **cabin carts** for each cabin.
* Users can click on a cabin cart to be directed to a **single cabin page** (to be implemented later).
* Implemented with a starter page and an empty **array of cabins** (will be updated later).
* **Cabin cart component** needs to be added for each cabin (currently without images).

**About Page:**

* Focus on responsive images.
* Images in the about page are large (1.1MB, 2,000px wide), which needs to be optimized for performance.
* Implemented using **Next.js image component** for responsiveness:
  + Static image import for responsiveness.
  + Images should scale based on container size, ensuring **fluid layout**.
* Use **image component** from Next.js:
  + Automatically renders as a regular **HTML img tag**, retaining **max width** of 100% for responsiveness.
* **Fill property**:
  + When source is dynamic (e.g., from a database), **use fill to make the image responsive within a container**, but without setting width/height.
  + Use object-fit: **cover in Tailwind CSS to ensure image fits container properly**.
* **Parent container size**:
  + **Set aspect ratio for the container (e.g., aspect-square for square images) to define the image size without specifying exact dimensions.**
* **Image Placeholder**:
  + Use **blur** placeholder for better user experience and SEO (no layout shift).
  + Placeholder will show while the image is loading.

**Key Techniques for Responsive Images:**

* **Aspect ratio** for container sizing instead of fixed width/height.
* **Static image import** works best for images in the project.
* For dynamic image sources, use the **fill** prop and adjust container styles.
* **Blur placeholder** improves UX and SEO metrics like **layout shift**.

**Nested Routes in Next.js**:

* **Folder structure** defines route segments in Next.js.
* **/account** is the primary route (guest area).
* **Nested routes**:
  + **/account/reservations** (nested route for reservations).
  + **/account/profile** (nested route for user profile).
* **Folder creation**:
  + Create reservations and profile folders inside /account.
  + Add page.js in each folder for the route to be accessible

**Notes on Nested Layouts in Next.js**

* **Nested Layouts** allow persistent UI elements across specific routes while maintaining page-specific content.
* Applied to the **account area** (/account, /account/reservations, /account/profile) to include a **side navigation**.

**Creating a Nested Layout**

* Next.js **layout.js** at the top level (app/layout.js) applies to all routes globally.
* For **nested layouts**, create another **layout.js** inside the specific folder (app/account/layout.js).
* This ensures the layout applies only to **routes under /account**.