Complete Next.js Cheat Sheet - Beginner to Advanced

Table of Contents

Getting Started

Installation & Setup

```
# Create new Next.js app
npx create-next-app@latest my-app
cd my-app

# With TypeScript
npx create-next-app@latest my-app --typescript

# With specific template
npx create-next-app@latest my-app --example with-tailwindcss

# Start development server
npm run dev
# or
yarn dev
# or
pnpm dev
```

Essential Dependencies

```
# Core dependencies (usually included)
npm install next react react-dom
```

```
# TypeScript support
npm install --save-dev typescript @types/react @types/node

# Styling
npm install tailwindcss postcss autoprefixer
npm install styled-components
npm install @emotion/react @emotion/styled

# State Management
npm install zustand
npm install @reduxjs/toolkit react-redux

# Forms
npm install react-hook-form
npm install formik yup

# HTTP Client
npm install axios
npm install swr

# Authentication
npm install next-auth
npm install @auth0/nextjs-auth0

# Database
npm install prisma @prisma/client
npm install mongoose
```

Project Structure

App Router Structure (Next.js 13+)

```
my-app/
├── app/
│   ├── globals.css
│   ├── layout.tsx
│   ├── page.tsx
│   ├── loading.tsx
│   ├── error.tsx
│   ├── not-found.tsx
│   ├── about/
│   │   └── page.tsx
│   ├── blog/
│   │   ├── page.tsx
│   │   └── [slug]/
│   │       └── page.tsx
│   └── api/
│       ├── auth/
│       └── users/
├── components/
```

```
├── lib/
├── public/
├── styles/
└── next.config.js
```

Pages Router Structure (Legacy)

```
my-app/
├── pages/
│   ├── _app.tsx
│   ├── _document.tsx
│   ├── index.tsx
│   ├── about.tsx
│   ├── blog/
│   │   ├── index.tsx
│   │   └── [slug].tsx
│   └── api/
│       ├── auth/
│       └── users.ts
├── components/
├── lib/
├── public/
├── styles/
└── next.config.js
```

Pages and Routing

App Router (Next.js 13+)

Basic Page Creation

```
// app/page.tsx (Home page)
export default function HomePage() {
  return <h1>Welcome to Next.js!</h1>
}

// app/about/page.tsx
export default function AboutPage() {
  return <h1>About Us</h1>
}

// app/blog/[slug]/page.tsx (Dynamic route)
export default function BlogPost({ params }: { params: { slug: string } }) {
  return <h1>Blog Post: {params.slug}</h1>
}
```

Layout Components

```
// app/layout.tsx (Root layout)
import './globals.css'

export const metadata = {
  title: 'My App',
  description: 'Generated by Next.js',
}

export default function RootLayout({
  children,
}: {
  children: React.ReactNode
}) {
  return (
    <html lang="en">
      <body>
        <nav>Navigation</nav>
        <main>{children}</main>
        <footer>Footer</footer>
      </body>
    </html>
  )
}

// app/blog/layout.tsx (Nested layout)
export default function BlogLayout({
  children,
}: {
  children: React.ReactNode
}) {
  return (
    <div className="blog-container">
      <aside>Blog Sidebar</aside>
      <div>{children}</div>
    </div>
  )
}
```

Special Files

```
// app/loading.tsx
export default function Loading() {
  return <div>Loading...</div>
}

// app/error.tsx
'use client'
```

```
export default function Error({
  error,
  reset,
}: {
  error: Error & { digest?: string }
  reset: () => void
}) {
  return (
    <div>
      <h2>Something went wrong!</h2>
      <button onClick={() => reset()}>Try again</button>
    </div>
  )
}

// app/not-found.tsx
export default function NotFound() {
  return (
    <div>
      <h2>Not Found</h2>
      <p>Could not find requested resource</p>
    </div>
  )
}
```

Pages Router (Legacy)

Basic Pages

```
// pages/index.tsx
export default function Home() {
  return <h1>Home Page</h1>
}

// pages/about.tsx
export default function About() {
  return <h1>About Page</h1>
}

// pages/blog/[slug].tsx
import { useRouter } from 'next/router'

export default function BlogPost() {
  const router = useRouter()
  const { slug } = router.query

  return <h1>Blog Post: {slug}</h1>
}
```

Custom App and Document

```
// pages/_app.tsx
import type { AppProps } from 'next/app'
import '../styles/globals.css'

export default function App({ Component, pageProps }: AppProps) {
  return (
    <div>
      <nav>Navigation</nav>
      <Component {...pageProps} />
    </div>
  )
}

// pages/_document.tsx
import { Html, Head, Main, NextScript } from 'next/document'

export default function Document() {
  return (
    <Html lang="en">
      <Head />
      <body>
        <Main />
        <NextScript />
      </body>
    </Html>
  )
}
```

Navigation

Link Component

```
import Link from 'next/link'

// Basic link
<Link href="/about">About</Link>

// Dynamic link
<Link href={`/blog/${post.slug}`}>Read More</Link>

// External link
<Link href="https://example.com" target="_blank">
  External Link
</Link>

// With custom styling
<Link href="/about" className="nav-link">
  About
</Link>
```

Programmatic Navigation

```
// App Router
import { useRouter } from 'next/navigation'

function MyComponent() {
  const router = useRouter()

  const handleClick = () => {
    router.push('/about')
    router.replace('/about') // No history entry
    router.back()
    router.forward()
    router.refresh()
  }
}

// Pages Router
import { useRouter } from 'next/router'

function MyComponent() {
  const router = useRouter()

  const handleClick = () => {
    router.push('/about')
    router.replace('/about')
    router.back()
    router.reload()
  }
}
```

Components

Client vs Server Components (App Router)

Server Components (Default)

```
// app/components/ServerComponent.tsx
// Server components run on the server
async function ServerComponent() {
  const data = await fetch('https://api.example.com/data')
  const json = await data.json()

  return (
    <div>
      <h1>Server Component</h1>
      <pre>{JSON.stringify(json, null, 2)}</pre>
    </div>
```

```
  )
}
```

Client Components

```
// app/components/ClientComponent.tsx
'use client'

import { useState, useEffect } from 'react'

export default function ClientComponent() {
  const [count, setCount] = useState(0)

  useEffect(() => {
    console.log('Client component mounted')
  }, [])

  return (
    <div>
      <h1>Client Component</h1>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>
        Increment
      </button>
    </div>
  )
}
```

Component Patterns

Higher-Order Component (HOC)

```
function withAuth<T extends object>(Component: React.ComponentType<T>) {
  return function AuthenticatedComponent(props: T) {
    const [isAuthenticated, setIsAuthenticated] = useState(false)

    if (!isAuthenticated) {
      return <div>Please log in</div>
    }

    return <Component {...props} />
  }
}

// Usage
const ProtectedPage = withAuth(MyPage)
```

Custom Hooks

```
// hooks/useCounter.ts
import { useState } from 'react'

export function useCounter(initialValue: number = 0) {
  const [count, setCount] = useState(initialValue)

  const increment = () => setCount(count + 1)
  const decrement = () => setCount(count - 1)
  const reset = () => setCount(initialValue)

  return { count, increment, decrement, reset }
}

// Usage in component
function Counter() {
  const { count, increment, decrement, reset } = useCounter(0)

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={increment}>+</button>
      <button onClick={decrement}>-</button>
      <button onClick={reset}>Reset</button>
    </div>
  )
}
```

Styling

CSS Modules

```
// styles/Home.module.css
.container {
  padding: 2rem;
  background-color: #f0f0f0;
}

.title {
  font-size: 2rem;
  color: #333;
}

// components/Home.tsx
import styles from '../styles/Home.module.css'

export default function Home() {
  return (
```

```
    <div className={styles.container}>
      <h1 className={styles.title}>Home Page</h1>
    </div>
  )
}
```

Tailwind CSS

```
// tailwind.config.js
module.exports = {
  content: [
    './pages/**/*.{js,ts,jsx,tsx}',
    './components/**/*.{js,ts,jsx,tsx}',
    './app/**/*.{js,ts,jsx,tsx}',
  ],
  theme: {
    extend: {
      colors: {
        primary: '#3b82f6',
        secondary: '#64748b',
      },
    },
  },
  plugins: [],
}

// Component usage
export default function Button({ children, variant = 'primary' }) {
  const baseClasses = 'px-4 py-2 rounded font-medium'
  const variantClasses = {
    primary: 'bg-blue-500 text-white hover:bg-blue-600',
    secondary: 'bg-gray-500 text-white hover:bg-gray-600',
  }

  return (
    <button className={`${baseClasses} ${variantClasses[variant]}`}>
      {children}
    </button>
  )
}
```

Styled Components

```
// components/StyledButton.tsx
import styled from 'styled-components'

const StyledButton = styled.button`
  background-color: ${props => props.primary ? '#3b82f6' : '#64748b'};
  color: white;
```

```
  padding: 0.5rem 1rem;
  border: none;
  border-radius: 0.25rem;
  cursor: pointer;

  &:hover {
    opacity: 0.8;
  }
`

export default function Button({ primary, children }) {
  return (
    <StyledButton primary={primary}>
      {children}
    </StyledButton>
  )
}
```

Global Styles

```
/* styles/globals.css */
@tailwind base;
@tailwind components;
@tailwind utilities;

* {
  box-sizing: border-box;
  padding: 0;
  margin: 0;
}

html,
body {
  max-width: 100vw;
  overflow-x: hidden;
  font-family: Inter, sans-serif;
}

@layer components {
  .btn-primary {
    @apply bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded;
  }
}
```

Data Fetching

App Router Data Fetching

Server Components (Recommended)

```
// app/posts/page.tsx
async function getPosts() {
  const res = await fetch('https://jsonplaceholder.typicode.com/posts', {
    cache: 'force-cache', // Default caching
  })

  if (!res.ok) {
    throw new Error('Failed to fetch data')
  }

  return res.json()
}

export default async function PostsPage() {
  const posts = await getPosts()

  return (
    <div>
      <h1>Posts</h1>
      {posts.map(post => (
        <div key={post.id}>
          <h2>{post.title}</h2>
          <p>{post.body}</p>
        </div>
      ))}
    </div>
  )
}
```

Caching Options

```
// No caching
fetch('https://api.example.com/data', { cache: 'no-store' })

// Revalidate every 60 seconds
fetch('https://api.example.com/data', { next: { revalidate: 60 } })

// Cache forever (default)
fetch('https://api.example.com/data', { cache: 'force-cache' })
```

Dynamic Data with Client Components

```
'use client'

import { useState, useEffect } from 'react'

export default function ClientPosts() {
```

```
  const [posts, setPosts] = useState([])
  const [loading, setLoading] = useState(true)

  useEffect(() => {
    async function fetchPosts() {
      try {
        const res = await fetch('/api/posts')
        const data = await res.json()
        setPosts(data)
      } catch (error) {
        console.error('Error fetching posts:', error)
      } finally {
        setLoading(false)
      }
    }

    fetchPosts()
  }, [])

  if (loading) return <div>Loading...</div>

  return (
    <div>
      {posts.map(post => (
        <div key={post.id}>{post.title}</div>
      ))}
    </div>
  )
}
```

Pages Router Data Fetching

getStaticProps (SSG)

```
// pages/posts.tsx
export async function getStaticProps() {
  const res = await fetch('https://jsonplaceholder.typicode.com/posts')
  const posts = await res.json()

  return {
    props: {
      posts,
    },
    revalidate: 60, // ISR - revalidate every 60 seconds
  }
}

export default function Posts({ posts }) {
  return (
    <div>
      {posts.map(post => (
```

```
        <div key={post.id}>{post.title}</div>
      ))}
    </div>
  )
}
```

getServerSideProps (SSR)

```
// pages/profile.tsx
export async function getServerSideProps(context) {
  const { req, res, params, query } = context

  // Access cookies, headers, etc.
  const token = req.cookies.token

  if (!token) {
    return {
      redirect: {
        destination: '/login',
        permanent: false,
      },
    }
  }

  const userData = await fetchUserData(token)

  return {
    props: {
      user: userData,
    },
  }
}
```

getStaticPaths (Dynamic SSG)

```
// pages/posts/[id].tsx
export async function getStaticPaths() {
  const res = await fetch('https://jsonplaceholder.typicode.com/posts')
  const posts = await res.json()

  const paths = posts.map(post => ({
    params: { id: post.id.toString() },
  }))

  return {
    paths,
    fallback: 'blocking', // or false, true
  }
```

```
}

export async function getStaticProps({ params }) {
  const res = await
fetch(`https://jsonplaceholder.typicode.com/posts/${params.id}`)
  const post = await res.json()

  return {
    props: {
      post,
    },
  }
}
```

SWR for Client-Side Data Fetching

```
import useSWR from 'swr'

const fetcher = (url: string) => fetch(url).then(res => res.json())

export default function Profile() {
  const { data, error, isLoading } = useSWR('/api/user', fetcher)

  if (error) return <div>Failed to load</div>
  if (isLoading) return <div>Loading...</div>

  return <div>Hello {data.name}!</div>
}

// With custom hook
function useUser(id: string) {
  const { data, error, isLoading } = useSWR(
    id ? `/api/user/${id}` : null,
    fetcher
  )

  return {
    user: data,
    isLoading,
    isError: error
  }
}
```

API Routes

App Router API Routes

Basic API Route

```
// app/api/users/route.ts
import { NextRequest, NextResponse } from 'next/server'

export async function GET(request: NextRequest) {
  const searchParams = request.nextUrl.searchParams
  const query = searchParams.get('query')

  // Fetch data from database
  const users = await fetchUsers(query)

  return NextResponse.json({ users })
}

export async function POST(request: NextRequest) {
  const body = await request.json()

  // Validate data
  if (!body.name || !body.email) {
    return NextResponse.json(
      { error: 'Name and email are required' },
      { status: 400 }
    )
  }

  // Create user
  const user = await createUser(body)

  return NextResponse.json({ user }, { status: 201 })
}
```

Dynamic API Routes

```
// app/api/users/[id]/route.ts
export async function GET(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  const id = params.id
  const user = await fetchUserById(id)

  if (!user) {
    return NextResponse.json(
      { error: 'User not found' },
      { status: 404 }
    )
  }

  return NextResponse.json({ user })
}
```

```
export async function PUT(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  const id = params.id
  const body = await request.json()

  const updatedUser = await updateUser(id, body)

  return NextResponse.json({ user: updatedUser })
}

export async function DELETE(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  const id = params.id
  await deleteUser(id)

  return NextResponse.json({ message: 'User deleted' })
}
```

Pages Router API Routes

Basic API Route

```
// pages/api/users.ts
import type { NextApiRequest, NextApiResponse } from 'next'

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponse
) {
  const { method } = req

  switch (method) {
    case 'GET':
      const users = await fetchUsers()
      res.status(200).json({ users })
      break

    case 'POST':
      const { name, email } = req.body

      if (!name || !email) {
        return res.status(400).json({ error: 'Name and email required' })
      }

      const user = await createUser({ name, email })
      res.status(201).json({ user })
      break
```

```
    default:
      res.setHeader('Allow', ['GET', 'POST'])
      res.status(405).end(`Method ${method} Not Allowed`)
  }
}
```

Dynamic API Route

```
// pages/api/users/[id].ts
export default async function handler(req, res) {
  const { id } = req.query
  const { method } = req

  switch (method) {
    case 'GET':
      const user = await fetchUserById(id)
      if (!user) {
        return res.status(404).json({ error: 'User not found' })
      }
      res.status(200).json({ user })
      break

    case 'PUT':
      const updatedUser = await updateUser(id, req.body)
      res.status(200).json({ user: updatedUser })
      break

    case 'DELETE':
      await deleteUser(id)
      res.status(200).json({ message: 'User deleted' })
      break

    default:
      res.setHeader('Allow', ['GET', 'PUT', 'DELETE'])
      res.status(405).end(`Method ${method} Not Allowed`)
  }
}
```

Middleware

```
// middleware.ts
import { NextResponse } from 'next/server'
import type { NextRequest } from 'next/server'

export function middleware(request: NextRequest) {
  // Check authentication
  const token = request.cookies.get('token')
```

```
  if (!token && request.nextUrl.pathname.startsWith('/dashboard')) {
    return NextResponse.redirect(new URL('/login', request.url))
  }

  // Add custom headers
  const response = NextResponse.next()
  response.headers.set('X-Custom-Header', 'custom-value')

  return response
}

export const config = {
  matcher: [
    '/dashboard/:path*',
    '/api/:path*',
    '/((?!api|_next/static|_next/image|favicon.ico).*)',
  ],
}
```

State Management

React Built-in State

```
// useState
import { useState } from 'react'

function Counter() {
  const [count, setCount] = useState(0)

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  )
}

// useReducer
import { useReducer } from 'react'

const initialState = { count: 0 }

function reducer(state, action) {
  switch (action.type) {
    case 'increment':
      return { count: state.count + 1 }
    case 'decrement':
      return { count: state.count - 1 }
    case 'reset':
      return initialState
```

```
    default:
      throw new Error()
  }
}

function Counter() {
  const [state, dispatch] = useReducer(reducer, initialState)

  return (
    <div>
      Count: {state.count}
      <button onClick={() => dispatch({ type: 'increment' })}>+</button>
      <button onClick={() => dispatch({ type: 'decrement' })}>-</button>
      <button onClick={() => dispatch({ type: 'reset' })}>Reset</button>
    </div>
  )
}
```

Context API

```
// contexts/AuthContext.tsx
import { createContext, useContext, useState } from 'react'

type AuthContextType = {
  user: User | null
  login: (user: User) => void
  logout: () => void
}

const AuthContext = createContext<AuthContextType | undefined>(undefined)

export function AuthProvider({ children }: { children: React.ReactNode }) {
  const [user, setUser] = useState<User | null>(null)

  const login = (user: User) => {
    setUser(user)
    localStorage.setItem('user', JSON.stringify(user))
  }

  const logout = () => {
    setUser(null)
    localStorage.removeItem('user')
  }

  return (
    <AuthContext.Provider value={{ user, login, logout }}>
      {children}
    </AuthContext.Provider>
  )
}
```

```
export function useAuth() {
  const context = useContext(AuthContext)
  if (context === undefined) {
    throw new Error('useAuth must be used within an AuthProvider')
  }
  return context
}

// Usage in component
function Profile() {
  const { user, logout } = useAuth()

  return (
    <div>
      <h1>Welcome, {user?.name}</h1>
      <button onClick={logout}>Logout</button>
    </div>
  )
}
```

Zustand

```
// store/useStore.ts
import { create } from 'zustand'
import { persist } from 'zustand/middleware'

interface CounterState {
  count: number
  increment: () => void
  decrement: () => void
  reset: () => void
}

export const useCounterStore = create<CounterState>()(
  persist(
    (set) => ({
      count: 0,
      increment: () => set((state) => ({ count: state.count + 1 })),
      decrement: () => set((state) => ({ count: state.count - 1 })),
      reset: () => set({ count: 0 }),
    }),
    {
      name: 'counter-storage',
    }
  )
)

// Usage in component
function Counter() {
  const { count, increment, decrement, reset } = useCounterStore()
```

```
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={increment}>+</button>
      <button onClick={decrement}>-</button>
      <button onClick={reset}>Reset</button>
    </div>
  )
}
```

Redux Toolkit

```
// store/store.ts
import { configureStore } from '@reduxjs/toolkit'
import counterReducer from './counterSlice'

export const store = configureStore({
  reducer: {
    counter: counterReducer,
  },
})

export type RootState = ReturnType<typeof store.getState>
export type AppDispatch = typeof store.dispatch

// store/counterSlice.ts
import { createSlice, PayloadAction } from '@reduxjs/toolkit'

interface CounterState {
  value: number
}

const initialState: CounterState = {
  value: 0,
}

export const counterSlice = createSlice({
  name: 'counter',
  initialState,
  reducers: {
    increment: (state) => {
      state.value += 1
    },
    decrement: (state) => {
      state.value -= 1
    },
    incrementByAmount: (state, action: PayloadAction<number>) => {
      state.value += action.payload
    },
  },
})
```

```
export const { increment, decrement, incrementByAmount } = counterSlice.actions
export default counterSlice.reducer

// Usage in component
import { useSelector, useDispatch } from 'react-redux'
import { RootState } from '../store/store'
import { increment, decrement } from '../store/counterSlice'

function Counter() {
  const count = useSelector((state: RootState) => state.counter.value)
  const dispatch = useDispatch()

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => dispatch(increment())}>+</button>
      <button onClick={() => dispatch(decrement())}>-</button>
    </div>
  )
}
```

Authentication

NextAuth.js Setup

```
// app/api/auth/[...nextauth]/route.ts
import NextAuth from 'next-auth'
import GoogleProvider from 'next-auth/providers/google'
import CredentialsProvider from 'next-auth/providers/credentials'

const handler = NextAuth({
  providers: [
    GoogleProvider({
      clientId: process.env.GOOGLE_CLIENT_ID!,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET!,
    }),
    CredentialsProvider({
      name: 'credentials',
      credentials: {
        email: { label: 'Email', type: 'email' },
        password: { label: 'Password', type: 'password' }
      },
      async authorize(credentials) {
        // Validate credentials
        const user = await validateUser(credentials)

        if (user) {
          return {
            id: user.id,
```

```
            email: user.email,
            name: user.name,
          }
        }
        return null
      }
    })
  ],
  callbacks: {
    async jwt({ token, user }) {
      if (user) {
        token.id = user.id
      }
      return token
    },
    async session({ session, token }) {
      session.user.id = token.id
      return session
    },
  },
  pages: {
    signIn: '/auth/signin',
    signOut: '/auth/signout',
    error: '/auth/error',
  },
})

export { handler as GET, handler as POST }
```

Session Provider Setup

```
// app/providers.tsx
'use client'

import { SessionProvider } from 'next-auth/react'

export function Providers({ children }: { children: React.ReactNode }) {
  return <SessionProvider>{children}</SessionProvider>
}

// app/layout.tsx
import { Providers } from './providers'

export default function RootLayout({
  children,
}: {
  children: React.ReactNode
}) {
  return (
    <html lang="en">
      <body>
```

```
        <Providers>{children}</Providers>
      </body>
    </html>
  )
}
```

Using Authentication in Components

```
'use client'

import { useSession, signIn, signOut } from 'next-auth/react'

export default function Component() {
  const { data: session, status } = useSession()

  if (status === 'loading') return <p>Loading...</p>

  if (session) {
    return (
      <>
        <p>Signed in as {session.user?.email}</p>
        <button onClick={() => signOut()}>Sign out</button>
      </>
    )
  }

  return (
    <>
      <p>Not signed in</p>
      <button onClick={() => signIn()}>Sign in</button>
    </>
  )
}
```

Server-Side Authentication

```
// app/protected/page.tsx
import { getServerSession } from 'next-auth/next'
import { redirect } from 'next/navigation'

export default async function ProtectedPage() {
  const session = await getServerSession()

  if (!session) {
    redirect('/auth/signin')
  }

  return (
    <div>
```

```
      <h1>Protected Content</h1>
      <p>Welcome, {session.user?.name}!</p>
    </div>
  )
}
```

Custom Login Page

```
// app/auth/signin/page.tsx
'use client'

import { signIn, getProviders } from 'next-auth/react'
import { useState, useEffect } from 'react'

export default function SignIn() {
  const [providers, setProviders] = useState(null)

  useEffect(() => {
    const fetchProviders = async () => {
      const res = await getProviders()
      setProviders(res)
    }
    fetchProviders()
  }, [])

  return (
    <div className="min-h-screen flex items-center justify-center">
      <div className="max-w-md w-full space-y-8">
        <h2 className="text-center text-3xl font-bold">Sign in</h2>

        {providers && Object.values(providers).map((provider) => (
          <div key={provider.name}>
            <button
              onClick={() => signIn(provider.id)}
              className="w-full flex justify-center py-2 px-4 border border-
transparent rounded-md shadow-sm text-sm font-medium text-white bg-indigo-600
hover:bg-indigo-700"
            >
              Sign in with {provider.name}
            </button>
          </div>
        ))}
      </div>
    </div>
  )
}
```

Performance Optimization

Image Optimization

```
import Image from 'next/image'

// Basic usage
<Image
  src="/hero.jpg"
  alt="Hero image"
  width={800}
  height={600}
/>

// With priority loading
<Image
  src="/hero.jpg"
  alt="Hero image"
  width={800}
  height={600}
  priority
/>

// Fill container
<div style={{ position: 'relative', width: '100%', height: '400px' }}>
  <Image
    src="/hero.jpg"
    alt="Hero image"
    fill
    style={{ objectFit: 'cover' }}
  />
</div>

// Responsive images
<Image
  src="/hero.jpg"
  alt="Hero image"
  width={800}
  height={600}
  sizes="(max-width: 768px) 100vw, (max-width: 1200px) 50vw, 33vw"
/>
```

Font Optimization

```
// app/layout.tsx
import { Inter, Roboto_Mono } from 'next/font/google'

const inter = Inter({
  subsets: ['latin'],
  display: 'swap',
})
```

```
const robotoMono = Roboto_Mono({
  subsets: ['latin'],
  display: 'swap',
})

export default function RootLayout({
  children,
}: {
  children: React.ReactNode
}) {
  return (
    <html lang="en" className={inter.className}>
      <body>{children}</body>
    </html>
  )
}
```

Code Splitting

```
// Dynamic imports
import dynamic from 'next/dynamic'

const DynamicComponent = dynamic(() => import('../components/MyComponent'), {
  loading: () => <p>Loading...</p>,
  ssr: false, // Disable server-side rendering
})

// Conditional loading
const DynamicChart = dynamic(() => import('../components/Chart'), {
  loading: () => <p>Loading chart...</p>,
})

function Dashboard() {
  const [showChart, setShowChart] = useState(false)

  return (
    <div>
      <h1>Dashboard</h1>
      <button onClick={() => setShowChart(true)}>Show Chart</button>
      {showChart && <DynamicChart />}
    </div>
  )
}
```

Memoization

```
import { memo, useMemo, useCallback } from 'react'

// Component memoization
```

```
const ExpensiveComponent = memo(function ExpensiveComponent({ data }) {
  return (
    <div>
      {data.map(item => (
        <div key={item.id}>{item.name}</div>
      ))}
    </div>
  )
})

// Value memoization
function Component({ items, filter }) {
  const filteredItems = useMemo(() => {
    return items.filter(item => item.category === filter)
  }, [items, filter])

  const handleClick = useCallback((id) => {
    console.log('Item clicked:', id)
  }, [])

  return (
    <div>
      {filteredItems.map(item => (
        <div key={item.id} onClick={() => handleClick(item.id)}>
          {item.name}
        </div>
      ))}
    </div>
  )
}
```

Bundle Analysis

```
# Install bundle analyzer
npm install --save-dev @next/bundle-analyzer

# Add to next.config.js
const withBundleAnalyzer = require('@next/bundle-analyzer')({
  enabled: process.env.ANALYZE === 'true',
})

module.exports = withBundleAnalyzer({
  // Your Next.js config
})

# Run analysis
ANALYZE=true npm run build
```

Deployment

Vercel Deployment

```
# Install Vercel CLI
npm install -g vercel

# Deploy
vercel

# Environment variables
vercel env add NEXT_PUBLIC_API_URL
vercel env add DATABASE_URL
```

Docker Deployment

```
# Dockerfile
FROM node:18-alpine AS deps
WORKDIR /app
COPY package.json package-lock.json ./
RUN npm ci --only=production

FROM node:18-alpine AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .
RUN npm run build

FROM node:18-alpine AS runner
WORKDIR /app
ENV NODE_ENV production

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone ./
COPY --from=builder --chown=nextjs:nodejs /app/.next/static ./.next/static

USER nextjs
EXPOSE 3000
ENV PORT 3000

CMD ["node", "server.js"]
```

Environment Variables

```
# .env.local
NEXT_PUBLIC_API_URL=https://api.example.com
DATABASE_URL=postgresql://username:password@localhost:5432/mydb
```

```
NEXTAUTH_SECRET=your-secret-key
NEXTAUTH_URL=http://localhost:3000
```

Next.js Configuration

```
// next.config.js
/** @type {import('next').NextConfig} */
const nextConfig = {
  experimental: {
    appDir: true,
  },
  images: {
    domains: ['example.com', 'cdn.example.com'],
  },
  env: {
    CUSTOM_KEY: 'custom-value',
  },
  async redirects() {
    return [
      {
        source: '/old-path',
        destination: '/new-path',
        permanent: true,
      },
    ]
  },
  async rewrites() {
    return [
      {
        source: '/api/:path*',
        destination: 'https://api.example.com/:path*',
      },
    ]
  },
  async headers() {
    return [
      {
        source: '/(.*)',
        headers: [
          {
            key: 'X-Custom-Header',
            value: 'custom-value',
          },
        ],
      },
    ]
  },
}

module.exports = nextConfig
```

Advanced Features

Internationalization (i18n)

```
// next.config.js
module.exports = {
  i18n: {
    locales: ['en', 'fr', 'es'],
    defaultLocale: 'en',
    domains: [
      {
        domain: 'example.com',
        defaultLocale: 'en',
      },
      {
        domain: 'example.fr',
        defaultLocale: 'fr',
      },
    ],
  },
}
```

```
// pages/index.tsx
import { useRouter } from 'next/router'
import { GetStaticProps } from 'next'

export default function Home({ translations }) {
  const { locale, locales, asPath } = useRouter()

  return (
    <div>
      <h1>{translations.title}</h1>
      <p>Current locale: {locale}</p>
      <p>Available locales: {locales.join(', ')}</p>
    </div>
  )
}

export const getStaticProps: GetStaticProps = async ({ locale }) => {
  const translations = await import(`../locales/${locale}.json`)

  return {
    props: {
      translations: translations.default,
    },
  }
}
```

Progressive Web App (PWA)

```
// next.config.js
const withPWA = require('next-pwa')({
  dest: 'public',
  register: true,
  skipWaiting: true,
})

module.exports = withPWA({
  // Your Next.js config
})
```

```
// public/manifest.json
{
  "name": "My Next.js App",
  "short_name": "NextApp",
  "description": "A Next.js PWA",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#000000",
  "icons": [
    {
      "src": "/icons/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/icons/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

Streaming and Suspense

```
// App Router with Streaming
import { Suspense } from 'react'

async function SlowComponent() {
  await new Promise(resolve => setTimeout(resolve, 2000))
  return <div>Slow content loaded!</div>
}

export default function Page() {
  return (
```

```
      <div>
        <h1>Fast content</h1>
        <Suspense fallback={<div>Loading slow content...</div>}>
          <SlowComponent />
        </Suspense>
      </div>
    )
  }
```

Edge Runtime

```
// app/api/edge/route.ts
export const runtime = 'edge'

export async function GET() {
  return new Response('Hello from Edge Runtime!')
}
```

Server Actions (App Router)

```
// app/actions.ts
'use server'

import { revalidatePath } from 'next/cache'

export async function createPost(formData: FormData) {
  const title = formData.get('title') as string
  const content = formData.get('content') as string

  // Save to database
  await savePost({ title, content })

  // Revalidate the posts page
  revalidatePath('/posts')
}

// app/create-post/page.tsx
import { createPost } from '../actions'

export default function CreatePost() {
  return (
    <form action={createPost}>
      <input name="title" placeholder="Title" required />
      <textarea name="content" placeholder="Content" required />
      <button type="submit">Create Post</button>
    </form>
  )
}
```

Best Practices

File and Folder Structure

```
src/
├── app/
│   ├── (auth)/
│   │   ├── login/
│   │   └── register/
│   ├── dashboard/
│   │   ├── analytics/
│   │   └── settings/
│   └── globals.css
├── components/
│   ├── ui/
│   │   ├── Button.tsx
│   │   ├── Input.tsx
│   │   └── Modal.tsx
│   ├── forms/
│   └── layout/
├── lib/
│   ├── auth.ts
│   ├── db.ts
│   └── utils.ts
├── hooks/
│   ├── useAuth.ts
│   └── useLocalStorage.ts
├── types/
│   └── index.ts
└── utils/
    ├── constants.ts
    └── helpers.ts
```

Error Handling

```
// Global error boundary
'use client'

import { useEffect } from 'react'

export default function GlobalError({
  error,
  reset,
}: {
  error: Error & { digest?: string }
  reset: () => void
}) {
  useEffect(() => {
    // Log the error to an error reporting service
```

```
    console.error(error)
  }, [error])

  return (
    <html>
      <body>
        <h2>Something went wrong!</h2>
        <button onClick={() => reset()}>Try again</button>
      </body>
    </html>
  )
}

// Custom error page for specific routes
// app/dashboard/error.tsx
'use client'

export default function DashboardError({
  error,
  reset,
}: {
  error: Error & { digest?: string }
  reset: () => void
}) {
  return (
    <div className="error-container">
      <h2>Dashboard Error</h2>
      <p>{error.message}</p>
      <button onClick={reset}>Try again</button>
    </div>
  )
}
```

SEO Optimization

```
// app/blog/[slug]/page.tsx
import { Metadata } from 'next'

export async function generateMetadata({
  params,
}: {
  params: { slug: string }
}): Promise<Metadata> {
  const post = await getPost(params.slug)

  return {
    title: post.title,
    description: post.excerpt,
    openGraph: {
      title: post.title,
      description: post.excerpt,
```

```
      images: [post.image],
    },
    twitter: {
      card: 'summary_large_image',
      title: post.title,
      description: post.excerpt,
      images: [post.image],
    },
  }
}

// JSON-LD structured data
export default function BlogPost({ params }) {
  const post = await getPost(params.slug)

  const jsonLd = {
    '@context': 'https://schema.org',
    '@type': 'Article',
    headline: post.title,
    description: post.excerpt,
    author: {
      '@type': 'Person',
      name: post.author.name,
    },
    datePublished: post.publishedAt,
  }

  return (
    <>
      <script
        type="application/ld+json"
        dangerouslySetInnerHTML={{ __html: JSON.stringify(jsonLd) }}
      />
      <article>
        <h1>{post.title}</h1>
        <p>{post.content}</p>
      </article>
    </>
  )
}
```

Testing

```
// __tests__/components/Button.test.tsx
import { render, screen, fireEvent } from '@testing-library/react'
import Button from '../components/Button'

describe('Button', () => {
  it('renders button with text', () => {
    render(<Button>Click me</Button>)
    expect(screen.getByText('Click me')).toBeInTheDocument()
```

```
  })

  it('calls onClick handler when clicked', () => {
    const handleClick = jest.fn()
    render(<Button onClick={handleClick}>Click me</Button>)

    fireEvent.click(screen.getByText('Click me'))
    expect(handleClick).toHaveBeenCalledTimes(1)
  })
})

// __tests__/pages/api/users.test.ts
import { createMocks } from 'node-mocks-http'
import handler from '../pages/api/users'

describe('/api/users', () => {
  it('returns users for GET request', async () => {
    const { req, res } = createMocks({
      method: 'GET',
    })

    await handler(req, res)

    expect(res._getStatusCode()).toBe(200)
    const data = JSON.parse(res._getData())
    expect(data).toHaveProperty('users')
  })
})
```

Security Best Practices

```
// Content Security Policy
// next.config.js
const securityHeaders = [
  {
    key: 'Content-Security-Policy',
    value: "default-src 'self'; script-src 'self' 'unsafe-eval'; style-src 'self'
'unsafe-inline';",
  },
  {
    key: 'X-Frame-Options',
    value: 'DENY',
  },
  {
    key: 'X-Content-Type-Options',
    value: 'nosniff',
  },
]

module.exports = {
  async headers() {
```

```
    return [
      {
        source: '/(.*)',
        headers: securityHeaders,
      },
    ]
  },
}

// Input validation and sanitization
import { z } from 'zod'

const userSchema = z.object({
  name: z.string().min(1).max(50),
  email: z.string().email(),
  age: z.number().min(18).max(120),
})

export async function POST(request: Request) {
  try {
    const body = await request.json()
    const validatedData = userSchema.parse(body)

    // Process validated data
    const user = await createUser(validatedData)

    return NextResponse.json({ user })
  } catch (error) {
    if (error instanceof z.ZodError) {
      return NextResponse.json(
        { error: 'Invalid input', details: error.errors },
        { status: 400 }
      )
    }

    return NextResponse.json(
      { error: 'Internal server error' },
      { status: 500 }
    )
  }
}
```

Troubleshooting

Common Issues and Solutions

Hydration Errors

```
// Problem: Hydration mismatch
function MyComponent() {
```

```
  const [mounted, setMounted] = useState(false)

  useEffect(() => {
    setMounted(true)
  }, [])

  if (!mounted) {
    return null // or loading skeleton
  }

  return <div>{new Date().toLocaleDateString()}</div>
}

// Alternative: Use dynamic import with ssr: false
const DynamicComponent = dynamic(() => import('./MyComponent'), {
  ssr: false,
})
```

CORS Issues

```
// pages/api/cors-example.ts
import { NextApiRequest, NextApiResponse } from 'next'

export default function handler(req: NextApiRequest, res: NextApiResponse) {
  // Set CORS headers
  res.setHeader('Access-Control-Allow-Origin', '*')
  res.setHeader('Access-Control-Allow-Methods',
'GET,OPTIONS,PATCH,DELETE,POST,PUT')
  res.setHeader('Access-Control-Allow-Headers', 'X-CSRF-Token, X-Requested-With,
Accept, Accept-Version, Content-Length, Content-MD5, Content-Type, Date, X-Api-
Version')

  if (req.method === 'OPTIONS') {
    res.status(200).end()
    return
  }

  // Handle other methods
  res.status(200).json({ message: 'Hello World' })
}
```

Memory Leaks

```
// Cleanup useEffect
useEffect(() => {
  const timer = setInterval(() => {
    console.log('Timer tick')
  }, 1000)
```

```
  return () => {
    clearInterval(timer) // Cleanup
  }
}, [])

// Cleanup event listeners
useEffect(() => {
  const handleResize = () => {
    console.log('Window resized')
  }

  window.addEventListener('resize', handleResize)

  return () => {
    window.removeEventListener('resize', handleResize)
  }
}, [])
```

Performance Issues

```
# Check bundle size
npm run build

# Analyze bundle
npm install --save-dev webpack-bundle-analyzer
ANALYZE=true npm run build

# Check core web vitals
npm install --save-dev @next/bundle-analyzer
```

Debugging Tools

```
// Custom debug component
function DebugInfo({ data }) {
  if (process.env.NODE_ENV !== 'development') {
    return null
  }

  return (
    <div style={{
      position: 'fixed',
      bottom: 0,
      right: 0,
      background: 'black',
      color: 'white',
      padding: '1rem',
      fontSize: '12px',
      maxWidth: '300px',
```

```
        overflow: 'auto'
      }}>
        <pre>{JSON.stringify(data, null, 2)}</pre>
      </div>
    )
  }

  // Usage
  function MyComponent() {
    const [state, setState] = useState({ count: 0 })

    return (
      <div>
        <button onClick={() => setState({ count: state.count + 1 })}>
          Count: {state.count}
        </button>
        <DebugInfo data={state} />
      </div>
    )
  }
```

Useful Resources

Official Documentation

- Next.js Documentation
- React Documentation
- TypeScript Documentation

Community Resources

- Next.js Examples
- Awesome Next.js
- Next.js Discord

Tools and Libraries

- Tailwind CSS
- NextAuth.js
- Prisma
- Vercel
- SWR
- React Hook Form

This cheat sheet covers the essential concepts and patterns you'll need for Next.js development, from basic setup to advanced features. Keep it handy as a reference while building your applications!