# Nextron で画像ビューアーを作成

## 目次



---

## Nextron とは

**Nextron** は、Next.js と Electron を組み合わせたフレームワークです。Web 技術（React、Next.js）を使ってデスクトップアプリケーションを開発できます。

### 特徴

- ⚡ **高速開発**: Next.js の開発体験
- 🖥️ **クロスプラットフォーム**: Windows、macOS、Linux 対応
- 🎨 **豊富なUI**: Tailwind CSS などの CSS フレームワークが使える
- 🔧 **柔軟性**: Web 技術の豊富なエコシステムを活用

---

## 環境構築

### 1. Nextron プロジェクトの作成

```
# Tailwind CSS 付きのテンプレートでプロジェクトを作成
npx create-nextron-app . --example with-tailwindcss
```

### 2. 依存関係のインストール

```
npm install
```

### 3. 開発サーバーの起動

```
npm run dev
```

### 4. ビルド

```
npm run build
```

---

## 画像ビューアーの設計

### 機能要件

- 📁 **画像選択**: 複数画像の選択・読み込み
- 🖼️ **画像表示**: 美しい画像表示
- 🔍 **ズーム機能**: 画像の拡大・縮小
- ⬅️ ➡️ **ナビゲーション**: 画像の切り替え
- 📊 **情報表示**: 現在の画像番号とズーム率

### 技術スタック

- **フロントエンド**: React + Next.js
- **スタイリング**: Tailwind CSS
- **デスクトップ**: Electron
- **状態管理**: React Hooks (useState, useRef)

---

## 実装

### 1. 必要なインポート

```
import React, { useState, useRef } from 'react'
import Head from 'next/head'
```

### 2. 状態管理の設定

```
const [selectedImage, setSelectedImage] = useState<string | null>(null)
const [zoom, setZoom] = useState(1)
const [images, setImages] = useState<string[]>([])
const [currentIndex, setCurrentIndex] = useState(0)
const fileInputRef = useRef<HTMLInputElement>(null)
```

### 3. ファイル選択機能

```
const handleFileSelect = (event: React.ChangeEvent<HTMLInputElement>) => {
  const files = event.target.files
  if (files) {
    const newImages: string[] = []
    for (let i = 0; i < files.length; i++) {
      const file = files[i]
```

```
      if (file.type.startsWith('image/')) {
        const reader = new FileReader()
        reader.onload = (e) => {
          const result = e.target?.result as string
          newImages.push(result)
          if (newImages.length === files.length) {
            setImages(prev => [...prev, ...newImages])
            if (selectedImage === null) {
              setSelectedImage(newImages[0])
              setCurrentIndex(0)
            }
          }
        }
        reader.readAsDataURL(file)
      }
    }
  }
}
```

### 4. ナビゲーション機能

```
const nextImage = () => {
  if (images.length > 0) {
    const nextIndex = (currentIndex + 1) % images.length
    setCurrentIndex(nextIndex)
    setSelectedImage(images[nextIndex])
  }
}

const prevImage = () => {
  if (images.length > 0) {
    const prevIndex = currentIndex === 0 ? images.length - 1 :
currentIndex - 1
    setCurrentIndex(prevIndex)
    setSelectedImage(images[prevIndex])
  }
}
```

### 5. ズーム機能

```
const resetZoom = () => {
  setZoom(1)
}

const zoomIn = () => {
  setZoom(prev => Math.min(prev + 0.2, 3))
}

const zoomOut = () => {
```

```
    setZoom(prev => Math.max(prev - 0.2, 0.5))
  }
```

## 6. UI コンポーネント

```
  return (
    <React.Fragment>
      <Head>
        <title>画像ビューアー</title>
      </Head>

      <div className="min-h-screen bg-gray-100 p-4">
        {/* ヘッダー */}
        <div className="max-w-6xl mx-auto mb-6">
          <h1 className="text-3xl font-bold text-center text-gray-800 mb-4">
            📸 画像ビューアー
          </h1>

          {/* コントロールボタン */}
          <div className="flex justify-center gap-4 mb-6">
            <button
              onClick={openFileDialog}
              className="px-4 py-2 bg-blue-500 text-white rounded-lg
hover:bg-blue-600 transition-colors"
            >
              📁 画像を選択
            </button>
            <input
              ref={fileInputRef}
              type="file"
              multiple
              accept="image/*"
              onChange={handleFileSelect}
              className="hidden"
            />

            {images.length > 0 && (
              <>
                <button
                  onClick={prevImage}
                  className="px-4 py-2 bg-gray-500 text-white rounded-lg
hover:bg-gray-600 transition-colors"
                >
                  ⬅️ 前へ
                </button>
                <button
                  onClick={nextImage}
                  className="px-4 py-2 bg-gray-500 text-white rounded-lg
hover:bg-gray-600 transition-colors"
                >
                  次へ ➡
```

```
              </button>
            </>
          )}
        </div>

        {/* ズームコントロール */}
        {selectedImage && (
          <div className="flex justify-center gap-2 mb-4">
            <button
              onClick={zoomOut}
              className="px-3 py-1 bg-green-500 text-white rounded
hover:bg-green-600 transition-colors"
            >
              🔍 -
            </button>
            <button
              onClick={resetZoom}
              className="px-3 py-1 bg-yellow-500 text-white rounded
hover:bg-yellow-600 transition-colors"
            >
              🔄 リセット
            </button>
            <button
              onClick={zoomIn}
              className="px-3 py-1 bg-green-500 text-white rounded
hover:bg-green-600 transition-colors"
            >
              🔍 +
            </button>
            <span className="px-3 py-1 bg-gray-200 rounded">
              {Math.round(zoom * 100)}%
            </span>
          </div>
        )}
      </div>

      {/* メインコンテンツエリア */}
      <div className="max-w-6xl mx-auto">
        {selectedImage ? (
          <div className="bg-white rounded-lg shadow-lg p-4">
            {/* 画像表示エリア */}
            <div className="flex justify-center items-center min-h-[400px]
overflow-auto">
              <img
                src={selectedImage}
                alt="Selected image"
                style={{
                  transform: `scale(${zoom})`,
                  transition: 'transform 0.2s ease-in-out'
                }}
                className="max-w-full max-h-full object-contain"
              />
            </div>
```

```
              {/* 情報表示 */}
              <div className="mt-4 text-center text-gray-600">
                <p>画像 {currentIndex + 1} / {images.length}</p>
                <p>ズーム: {Math.round(zoom * 100)}%</p>
              </div>
            </div>
          ) : (
            /* 初期表示 */
            <div className="bg-white rounded-lg shadow-lg p-8 text-center">
              <div className="text-6xl mb-4">📸</div>
              <h2 className="text-2xl font-semibold text-gray-700 mb-4">
                画像を選択してください
              </h2>
              <p className="text-gray-500 mb-6">
                画像ファイルをドラッグ&ドロップするか、ボタンをクリックして画像を選択してください
              </p>
              <button
                onClick={openFileDialog}
                className="px-6 py-3 bg-blue-500 text-white rounded-lg hover:bg-blue-600 transition-colors text-lg"
              >
                📁 画像を選択
              </button>
            </div>
          )}
        </div>
      </div>
    </React.Fragment>
  )
```

---

## 機能の詳細

### 📁 画像選択機能

- **FileReader API**: ローカルファイルを Base64 形式で読み込み
- **複数選択**: `multiple` 属性で複数画像を同時選択可能
- **ファイルタイプ制限**: `accept="image/*"` で画像ファイルのみ選択

### 🖼️ 画像表示

- **レスポンシブ**: `object-contain` でアスペクト比を保持
- **スムーズアニメーション**: CSS トランジションでズーム効果
- **オーバーフロー対応**: `overflow-auto` でスクロール可能

### 🔍 ズーム機能

- **範囲制限**: 0.5倍〜3倍の範囲でズーム
- **段階的ズーム**: 0.2刻みでズーム調整
- **リアルタイム表示**: ズーム率をパーセンテージで表示

### ⬅️➡️ ナビゲーション

- **循環ナビゲーション**: 最後の画像から最初の画像へ
- **状態管理**: 現在の画像インデックスを管理
- **条件付き表示**: 画像がある場合のみナビゲーションボタンを表示

---

## 学んだこと

### 🎯 React Hooks の活用

- **useState**: 複数の状態を効率的に管理
- **useRef**: DOM 要素への直接アクセス
- **カスタムフック**: 再利用可能なロジックの分離

### 🎨 Tailwind CSS の威力

- **ユーティリティファースト**: 高速なスタイリング
- **レスポンシブデザイン**: 簡単なレスポンシブ対応
- **カスタマイズ性**: 豊富なカスタマイズオプション

### ⚡ Electron の利点

- **Web 技術**: 既存の Web 開発スキルを活用
- **クロスプラットフォーム**: 一度の開発で複数 OS 対応
- **ネイティブ機能**: ファイルシステムアクセスなど

### 🔧 開発体験

- **ホットリロード**: リアルタイムでの変更確認
- **デバッグ**: Chrome DevTools でのデバッグ
- **ビルド**: 簡単なビルドプロセス

---

## 今後の拡張案

### 🚀 追加機能

- **ドラッグ&ドロップ**: ファイルのドラッグ&ドロップ対応
- **キーボードショートカット**: 矢印キーでのナビゲーション
- **画像編集**: 回転、反転、フィルター機能
- **スライドショー**: 自動再生機能
- **サムネイル表示**: 画像一覧のサムネイル表示

### 🎨 UI/UX 改善

- **ダークモード**: テーマ切り替え機能
- **アニメーション**: より滑らかなトランジション
- **アクセシビリティ**: キーボードナビゲーション対応

### 🔧 技術的改善

- **パフォーマンス**: 大きな画像の最適化
- **メモリ管理**: 画像の効率的な読み込み
- **エラーハンドリング**: より堅牢なエラー処理

---

## まとめ

Nextron を使った画像ビューアーの開発を通じて、以下のことを学びました：

1. **Nextron の基本**: Electron + Next.js の組み合わせ
2. **React Hooks**: モダンな React 開発手法
3. **Tailwind CSS**: 効率的なスタイリング
4. **ファイル操作**: ブラウザでのファイル処理
5. **状態管理**: 複雑な状態の管理方法

このプロジェクトは、デスクトップアプリケーション開発の良い入門として機能します。Web 技術の知識を活かしながら、ネイティブアプリのような体験を提供できます。

次回は、より高度な機能（画像編集、スライドショーなど）を追加して、さらに実用的なアプリケーションにしていきたいと思います！

---

**技術スタック**: Nextron, React, Next.js, Electron, Tailwind CSS
**開発期間**: 1日
**難易度**: 初級〜中級