@wordpress/image-cropper

- Current features
- Future features
- Installation
- API
  - Components
  - Hooks
  - Types
  - Utilities
- Usage
  - Basic Implementation
  - Advanced Usage with Custom Reset State

An implementation of react-easy-crop.

Current features

- **Image Cropping**: Interactive aspect ratio-based crop area with drag and resize functionality
- **Rotation**: Rotate images in 90-degree increments
- **Zoom Control**: Zoom in/out with configurable min/max limits
- **Aspect Ratio**: Set and maintain specific aspect ratios
- **Flip Controls**: Horizontal and vertical image flipping
- **State Management**: Centralized state management with React Context
- **Custom Reset State**: Configure custom initial states for reset operations

Future features

- [ ] Freeform cropping

Installation

Install the module

```
npm install @wordpress/image-cropper --save
```

API

Components

- `ImageCropper` – The main cropping component that provides the cropping canvas.

```
interface ImageCropperProps {
    src: string; // Image source URL
    onLoad?: ( mediaSize: MediaSize ) => void; // Callback when image loads with media
dimensions
    minZoom?: number; // Minimum zoom level (default: 1)
    maxZoom?: number; // Maximum zoom level (default: 5)
}
```

- `ImageCropperProvider` – Context provider for state management. This component implements the React Context pattern to share cropper state across your application. It manages all internal cropper state including crop position, zoom levels, rotation, flip transformations, aspect ratios, and media dimensions. Any component that needs to read or modify cropper state must be a descendant of this provider. The provider also handles state persistence and provides methods for resetting to custom initial states.

Hooks

- `useImageCropper` – Provides access to all cropper state and methods.

```
import { useImageCropper } from '@wordpress/image-cropper';

const {
    // Unified state object
    cropperState, // Complete cropper state { crop, croppedArea, croppedAreaPixels, zoom,
```

```
rotation, aspectRatio, flip, mediaSize }
    setCropperState, // Set multiple state properties at once

    // Actions
    reset, // Reset all changes (uses resetState if set)
    setResetState, // Set custom reset state
    getCroppedImage, // Get cropped image as data URL

    // Reset state
    resetState, // Current reset state configuration
    isDirty, // Whether the state is dirty based on resetState or default settings
} = useImageCropper();
```

Types

- `ImageCropperState` – State interface for cropper data
- `ImageCropperProps` – Props interface for ImageCropper component
- `ImageCropperContextValue` – Context value interface
- `Flip` – Flip state interface
- `Point`, `Area`, `MediaSize` – Re-exported from react-easy-crop

Utilities

- `normalizeRotation` – Utility function to normalize rotation values to 0-360 degrees

```
import { normalizeRotation } from '@wordpress/image-cropper';

const normalized = normalizeRotation( -90 ); // Returns 270
const normalized2 = normalizeRotation( 450 ); // Returns 90
```

Usage

Basic Implementation

The image cropper provides the core cropping functionality without any built-in UI controls. You must implement your own tools using the `useImageCropper` hook.

```
import {
    ImageCropper,
    ImageCropperProvider,
    useImageCropper,
} from '@wordpress/image-cropper';

function ImageEditor() {
    return (
        <ImageCropperProvider>
            <div className="image-editor">
                <ImageCropper
                    src="https://example.com/image.jpg"
                    className="image-cropper"
                    onLoad={ ( mediaSize ) =>
                        console.log( 'Image loaded', mediaSize )
                    }
                />
                <ImageEditorTools />
            </div>
        </ImageCropperProvider>
    );
}

function ImageEditorTools() {
    const { cropperState, setCropperState, reset, setResetState } =
        useImageCropper();

    const { zoom, rotation, aspectRatio, flip, mediaSize, croppedArea } =
        cropperState;

    const handleSave = () => {
        console.log( 'Cropper state:', {
            crop: croppedArea,
            zoom,
            rotation,
            aspectRatio,
            flip,
        } );
        // Apply the crop state to your image
```

```
    };

    return (
        <div className="image-editor-tools">
            <button
                onClick={ () => setCropperState( { rotation: rotation + 90 } ) }
            >
                Rotate Right
            </button>
            <button
                onClick={ () => setCropperState( { rotation: rotation - 90 } ) }
            >
                Rotate Left
            </button>
            <button
                onClick={ () =>
                    setCropperState( {
                        flip: { ...flip, horizontal: ! flip.horizontal },
                    } )
                }
            >
                Flip Horizontal
            </button>
            <button
                onClick={ () =>
                    setCropperState( {
                        flip: { ...flip, vertical: ! flip.vertical },
                    } )
                }
            >
                Flip Vertical
            </button>
            <button onClick={ handleSave }>Apply Changes</button>
            <button onClick={ reset }>Reset</button>
        </div>
    );
}
```

Advanced Usage with Custom Reset State

You can configure a custom initial state that will be used when the reset function is called:

```
function ImageEditorWithCustomReset() {
    const { setResetState, reset } = useImageCropper();

    // Set custom reset state
    useEffect( () => {
        setResetState( {
            rotation: 90,
            zoom: 1.5,
            aspectRatio: 16 / 9,
            flip: { horizontal: false, vertical: false },
        } );
    }, [ setResetState ] );

    return (
        <div>
            { /* Your cropper components */ }
            <button onClick={ reset }>Reset to Custom State</button>
        </div>
    );
}
```

□□□ #2
□ □□□ □□□ 2026-10-31 18:22:15 UTC
□ □□□ □□□ 2026-10-31 18:22:17 UTC