Image

A React component for displaying different types of images, including network images, static resources, temporary local images, and images from local disk, such as the camera roll.

This example shows fetching and displaying an image from local storage as well as one from network and even from data provided in the `'data:'` uri scheme.

> Note that for network and data images, you will need to manually specify the dimensions of your image!

Examples

Example Expo

```
import React from 'react';
import {View, Image, StyleSheet} from 'react-native';

const styles = StyleSheet.create({
  container: {
    paddingTop: 50,
  },
  tinyLogo: {
    width: 50,
    height: 50,
  },
  logo: {
    width: 66,
    height: 58,
  },
});

const DisplayAnImage = () => {
  return (
    <View style={styles.container}>
      <Image
        style={styles.tinyLogo}
        source={require('@expo/snack-static/react-native-logo.png')}
      />
      <Image
        style={styles.tinyLogo}
```

Preview My Device iOS Android Web

You can also add `style` to an image:

Example

Expo

```
import React from 'react';
import {View, Image, StyleSheet} from 'react-native';

const styles = StyleSheet.create({
  container: {
    paddingTop: 50,
  },
  stretch: {
    width: 50,
    height: 200,
    resizeMode: 'stretch',
  },
});

const DisplayAnImageWithStyle = () => {
  return (
    <View style={styles.container}>
      <Image
        style={styles.stretch}
        source={require('@expo/snack-static/react-native-logo.png')}
      />
    </View>
  );
};

export default DisplayAnImageWithStyle;
```

Preview My Device iOS Android Web

GIF and WebP support on Android

When building your own native code, GIF and WebP are not supported by default on Android.

You will need to add some optional modules in `android/app/build.gradle`, depending on the needs of your app.

```
dependencies {
  // If your app supports Android versions before Ice Cream Sandwich (API level 14)
  implementation 'com.facebook.fresco:animated-base-support:1.3.0'

  // For animated GIF support
  implementation 'com.facebook.fresco:animated-gif:2.5.0'

  // For WebP support, including animated WebP
```

```
    implementation 'com.facebook.fresco:animated-webp:2.5.0'
    implementation 'com.facebook.fresco:webpsupport:2.5.0'

    // For WebP support, without animations
    implementation 'com.facebook.fresco:webpsupport:2.5.0'
  }
```

> Note: the version listed above may not be updated in time. Please check `ReactAndroid/gradle.properties` in the main repo to see which fresco version is being used in a specific tagged version.

Reference

Props

View Props

Inherits View Props.

`accessible`

When true, indicates the image is an accessibility element.

TYPE	DEFAULT
bool	`false`

`accessibilityLabel`

The text that's read by the screen reader when the user interacts with the image.

TYPE
string

`alt`

A string that defines an alternative text description of the image, which will be read by the screen reader when the user interacts with it. Using this will automatically mark this element as accessible.

TYPE
string

`blurRadius`

blurRadius: the blur radius of the blur filter added to the image.

TYPE
number

> Tip: On IOS, you will need to increase `blurRadius` by more than `5`.

`capInsets` iOS

When the image is resized, the corners of the size specified by `capInsets` will stay a fixed size, but the center content and borders of the image will be stretched. This is useful for creating resizable rounded buttons, shadows, and other resizable assets. More info in the official Apple documentation.

TYPE
Rect

`crossOrigin`

A string of a keyword specifying the CORS mode to use when fetching the image resource. It works similar to crossorigin attribute in HTML.

- `anonymous` : No exchange of user credentials in the image request.
- `use-credentials` : Sets `Access-Control-Allow-Credentials` header value to `true` in the image request.

TYPE	DEFAULT
enum(`'anonymous'`, `'use-credentials'`)	`'anonymous'`

`defaultSource`

A static image to display while loading the image source.

TYPE
ImageSource

> **Note:** On Android, the default source prop is ignored on debug builds.

`fadeDuration` Android

Fade animation duration in milliseconds.

TYPE	DEFAULT
number	`300`

TYPE	DEFAULT

`height`

Height of the image component.

TYPE
number

`loadingIndicatorSource`

Similarly to `source`, this property represents the resource used to render the loading indicator for the image. The loading indicator is displayed until image is ready to be displayed, typically after the image is downloaded.

TYPE
ImageSource (`uri` only), number

`onError`

Invoked on load error.

TYPE
(`{nativeEvent: {error} }`) => void

`onLayout`

Invoked on mount and on layout changes.

TYPE
`({nativeEvent: LayoutEvent} => void`

`onLoad`

Invoked when load completes successfully.

Example: `onLoad={({nativeEvent: {source: {width, height}}}) => setImageRealSize({width, height})}`

TYPE
`({nativeEvent: ImageLoadEvent} => void`

`onLoadEnd`

Invoked when load either succeeds or fails.

TYPE
() => void

`onLoadStart`

Invoked on load start.

Example: `onLoadStart={() => this.setState({loading: true})}`

TYPE
() => void

`onPartialLoad` iOS

Invoked when a partial load of the image is complete. The definition of what constitutes a "partial load" is loader specific though this is meant for progressive JPEG loads.

TYPE
() => void

`onProgress`

Invoked on download progress.

TYPE
(`{nativeEvent: {loaded, total} }`) => void

`progressiveRenderingEnabled` Android

When `true`, enables progressive jpeg streaming - https://frescolib.org/docs/progressive-jpegs.

TYPE	DEFAULT
bool	`false`

`resizeMethod` Android

The mechanism that should be used to resize the image when the image's dimensions differ from the image view's dimensions. Defaults to `auto`.

- `auto`: Use heuristics to pick between `resize` and `scale`.

- `resize`: A software operation which changes the encoded image in memory before it gets decoded. This should be used instead of `scale` when the image is much larger than the view.

- `scale`: The image gets drawn downscaled or upscaled. Compared to `resize`, `scale` is faster (usually hardware accelerated) and produces higher quality images. This should be used if the image is smaller than the view. It should also be used if the image is slightly bigger than the view.

More details about `resize` and `scale` can be found at http://frescolib.org/docs/resizing.

TYPE	DEFAULT
enum(`'auto'`, `'resize'`, `'scale'`)	`'auto'`

`referrerPolicy`

A string indicating which referrer to use when fetching the resource. Sets the value for `Referrer-Policy` header in the image request. Works similar to `referrerpolicy` attribute in HTML.

TYPE	DEFAULT
enum(`'no-referrer'`, `'no-referrer-when-downgrade'`, `'origin'`, `'origin-when-cross-origin'`, `'same-origin'`, `'strict-origin'`, `'strict-origin-when-cross-origin'`, `'unsafe-url'`)	`'strict-origin-when-cross-origin'`

`resizeMode`

Determines how to resize the image when the frame doesn't match the raw image dimensions. Defaults to `cover`.

- `cover`: Scale the image uniformly (maintain the image's aspect ratio) so that

    - both dimensions (width and height) of the image will be equal to or larger than the corresponding dimension of the view (minus padding)
    - at least one dimension of the scaled image will be equal to the corresponding dimension of the view (minus padding)

- `contain`: Scale the image uniformly (maintain the image's aspect ratio) so that both dimensions (width and height) of the image will be equal to or less than the corresponding dimension of the view (minus padding).

- `stretch`: Scale width and height independently, This may change the aspect ratio of the src.

- `repeat`: Repeat the image to cover the frame of the view. The image will keep its size and aspect ratio, unless it is larger than the view, in which case it will be scaled down uniformly so that it is contained in the view.

- `center`: Center the image in the view along both dimensions. If the image is larger than the view, scale it down uniformly so that it is contained in the view.

TYPE	DEFAULT
enum(`'cover'`, `'contain'`, `'stretch'`, `'repeat'`, `'center'`)	`'cover'`

`source`

The image source (either a remote URL or a local file resource).

This prop can also contain several remote URLs, specified together with their width and height and potentially with scale/other URI arguments. The native side will then choose the best `uri` to display based on the measured size of the image container. A `cache` property can be added to control how networked request interacts with the local cache. (For more information see Cache Control for Images).

The currently supported formats are `png`, `jpg`, `jpeg`, `bmp`, `gif`, `webp`, `psd` (iOS only). In addition, iOS supports several RAW image formats. Refer to Apple's documentation for the current list of supported camera models (for iOS 12, see https://support.apple.com/en-ca/HT208967).

TYPE
ImageSource

`src`

A string representing the remote URL of the image. This prop has precedence over `source` prop.

Example: `src={'https://reactnative.dev/img/tiny_logo.png'}`

TYPE
string

`srcSet`

A string representing comma separated list of possible candidate image source. Each image source contains a URL of an image and a pixel density descriptor. If no descriptor is specified, it defaults to descriptor of `1x`.

If `srcSet` does not contain a `1x` descriptor, the value in `src` is used as image source with `1x` descriptor (if provided).

This prop has precedence over both the `src` and `source` props.

Example: `srcSet={'https://reactnative.dev/img/tiny_logo.png 1x, https://reactnative.dev/img/header_logo.svg 2x'}`

TYPE
string

`style`

TYPE
Image Style Props, Layout Props, Shadow Props, Transforms

`testID`

A unique identifier for this element to be used in UI Automation testing scripts.

TYPE
string

`tintColor`

Changes the color of all non-transparent pixels to the `tintColor`.

TYPE
color

`width`

Width of the image component.

TYPE
number

Methods

`abortPrefetch()` Android

```
static abortPrefetch(requestId: number);
```

Abort prefetch request.

Parameters:

NAME	TYPE	DESCRIPTION
requestId Required	number	Request id as returned by `prefetch()`.

`getSize()`

```
static getSize(
  uri: string,
  success: (width: number, height: number) => void,
  failure?: (error: any) => void,
): any;
```

Retrieve the width and height (in pixels) of an image prior to displaying it. This method can fail if the image cannot be found, or fails to download.

In order to retrieve the image dimensions, the image may first need to be loaded or downloaded, after which it will be cached. This means that in principle you could use this method to preload images, however it is not optimized for that purpose, and may in future be implemented in a way that does not fully load/download the image data. A proper, supported way to preload images will be provided as a separate API.

Parameters:

NAME	TYPE	DESCRIPTION
uri Required	string	The location of the image.
success Required	function	The function that will be called if the image was successfully found and width and height retrieved.

NAME	TYPE	DESCRIPTION
failure	function	The function that will be called if there was an error, such as failing to retrieve the image.

`getSizeWithHeaders()`

```
static getSizeWithHeaders(
  uri: string,
  headers: {[index: string]: string},
  success: (width: number, height: number) => void,
  failure?: (error: any) => void,
): any;
```

Retrieve the width and height (in pixels) of an image prior to displaying it with the ability to provide the headers for the request. This method can fail if the image cannot be found, or fails to download. It also does not work for static image resources.

In order to retrieve the image dimensions, the image may first need to be loaded or downloaded, after which it will be cached. This means that in principle you could use this method to preload images, however it is not optimized for that purpose, and may in future be implemented in a way that does not fully load/download the image data. A proper, supported way to preload images will be provided as a separate API.

Parameters:

NAME	TYPE	DESCRIPTION
uri Required	string	The location of the image.
headers Required	object	The headers for the request.
success Required	function	The function that will be called if the image was successfully found and width and height retrieved.
failure	function	The function that will be called if there was an error, such as failing to retrieve the image.

`prefetch()`

```
await Image.prefetch(url);
```

Prefetches a remote image for later use by downloading it to the disk cache. Returns a promise which resolves to a boolean.

Parameters:

NAME	TYPE	DESCRIPTION
url Required	string	The remote location of the image.
callback	function Android	The function that will be called with the `requestId`.

`queryCache()`

```
static queryCache(
  urls: string[],
): Promise<Record<string, 'memory' | 'disk' | 'disk/memory'>>;
```

Perform cache interrogation. Returns a promise which resolves to a mapping from URL to cache status, such as "disk", "memory" or "disk/memory". If a requested URL is not in the mapping, it means it's not in the cache.

Parameters:

NAME	TYPE	DESCRIPTION
urls Required	array	List of image URLs to check the cache for.

`resolveAssetSource()`

```
static resolveAssetSource(source: ImageSourcePropType): {
  height: number;
  width: number;
  scale: number;
  uri: string;
};
```

Resolves an asset reference into an object which has the properties `uri`, `scale`, `width`, and `height`.

Parameters:

NAME	TYPE	DESCRIPTION
source Required	ImageSource, number	A number (opaque type returned by `require('./foo.png')`) or an ImageSource.

Type Definitions

ImageCacheEnum iOS

Enum which can be used to set the cache handling or stategy for the potentially cached responses.

TYPE	DEFAULT
enum(`'default'`, `'reload'`, `'force-cache'`, `'only-if-cached'`)	`'default'`

- `default`: Use the native platforms default strategy.
- `reload`: The data for the URL will be loaded from the originating source. No existing cache data should be used to satisfy a URL load request.
- `force-cache`: The existing cached data will be used to satisfy the request, regardless of its age or expiration date. If there is no existing data in the cache corresponding the request, the data is loaded from the originating source.
- `only-if-cached`: The existing cache data will be used to satisfy a request, regardless of its age or expiration date. If there is no existing data in the cache corresponding to

a URL load request, no attempt is made to load the data from the originating source, and the load is considered to have failed.

ImageLoadEvent

Object returned in the `onLoad` callback.

TYPE
object

Properties:

NAME	TYPE	DESCRIPTION
source	object	The source object

Source Object

Properties:

NAME	TYPE	DESCRIPTION
width	number	The width of loaded image.
height	number	The height of loaded image.
uri	string	A string representing the resource identifier for the image.

ImageSource

TYPE
object, array of objects, number

Properties (if passing as object or array of objects):

NAME	TYPE	DESCRIPTION
uri	string	A string representing the resource identifier for the image, which could be an http address, a local file path, or the name of a static image resource.
width	number	Can be specified if known at build time, in which case the value will be used to set the default `<Image/>` component dimension.
height	number	Can be specified if known at build time, in which case the value will be used to set the default `<Image/>` component dimension.
scale	number	Used to indicate the scale factor of the image. Defaults to `1.0` if unspecified, meaning that one image pixel equates to one display point / DIP.
bundle ◀ iOS	string	The iOS asset bundle which the image is included in. This will default to `[NSBundle mainBundle]` if not set.
method	string	The HTTP Method to use. Defaults to `'GET'` if not specified.
headers	object	An object representing the HTTP headers to send along with the request for a remote image.
body	string	The HTTP body to send with the request. This must be a valid UTF-8 string, and will be sent exactly as specified, with no additional encoding (e.g. URL-escaping or base64) applied.
cache ◀ iOS	ImageCacheEnum	Determines how the requests handles potentially cached responses.

If passing a number:

- `number` - opaque type returned by something like `require('./image.jpg')`.

Is this page useful? 👍 👎

Edit this page

*Last updated on **Aug 17, 2023***