

Upload


Upload file by selecting or dragging.

When To Use

Uploading is the process of publishing information (web pages, text, pictures, video, etc.) to a remote server via a web page or upload tool.

- When you need to upload one or more files.
- When you need to show the process of uploading.
- When you need to upload files by dragging and dropping.

Examples

 Click to Upload

Upload by clicking

Classic mode. File selection dialog pops up when upload button is clicked.

```
import React from 'react';
import { UploadOutlined } from '@ant-design/icons';
import type { UploadProps } from 'antd';
import { Button, message, Upload } from 'antd'

const props: UploadProps = {
  name: 'file',
  action: 'https://www.mocky.io/v2/5cc8019d3000000800000000',
  headers: {
    authorization: 'authorization-text',
  },
  onChange(info) {
    if (info.file.status !== 'uploading') {
      console.log(info.file, info.fileList);
    }
    if (info.file.status === 'done') {
      message.success(`${info.file.name} file uploaded successfully`);
    } else if (info.file.status === 'error') {
      message.error(`${info.file.name} file upload failed.`);
    }
  },
};

const App: React.FC = () => (
  <Upload {...props}>
    <Button icon={<UploadOutlined />}>Click to Upload</Button>
  </Upload>
);

export default App;
```

 Upload

xxx.png

 yyy.png zzz.png

Default Files

Use `defaultFileList` for uploaded files when page init.

```
import React from 'react';
import { UploadOutlined } from '@ant-design/icons';
import type { UploadProps } from 'antd';
import { Button, Upload } from 'antd';
```

```
const props: UploadProps = {
  action: 'https://www.mocky.io/v2/5cc8019d3000000000000000',
  onChange({ file, fileList }) {
    if (file.status !== 'uploading') {
      console.log(file, fileList);
    }
  },
  defaultFileList: [
    {
      uid: '1',
      name: 'xxx.png',
      status: 'uploading',
      url: 'http://www.baidu.com/xxx.png',
      percent: 33,
    },
    {
      uid: '2',
      name: 'yyy.png',
      status: 'done',
      url: 'http://www.baidu.com/yyy.png',
    },
    {
      uid: '3',
      name: 'zzz.png',
      status: 'error',
      response: 'Server Error 500', // custom
      url: 'http://www.baidu.com/zzz.png',
    },
  ],
};
```

```
const App: React.FC = () => (
  <Upload {...props}>
    <Button icon={<UploadOutlined />}>Upload</Button>
  </Upload>
);

export default App;
```

+

Upload

+

Upload

Avatar

Click to upload user's avatar, and validate size and format of picture with `beforeUpload` .

The return value of function `beforeUpload` can be a Promise to check asynchronously. [demo](#)

```
import React, { useState } from 'react';
import { LoadingOutlined, PlusOutlined } from '@ant-design/icons';
import { message, Upload } from 'antd';
import type { UploadChangeParam } from 'antd/lib/upload';
import type { RcFile, UploadFile, UploadProps } from 'rc-upload';

const getBase64 = (img: RcFile, callback: (url: string) => void) => {
  const reader = new FileReader();
  reader.addEventListener('load', () => callback(reader.result as string));
  reader.readAsDataURL(img);
};
```

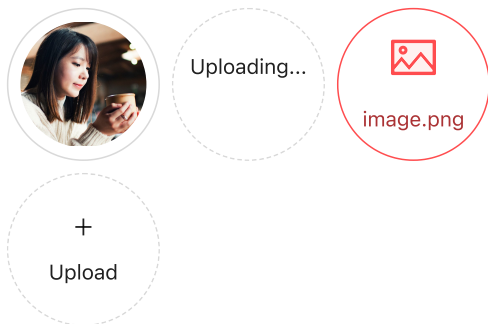
```
const beforeUpload = (file: RcFile) => {
  const isJpgOrPng = file.type === 'image/jpeg';
  if (!isJpgOrPng) {
    message.error('You can only upload JPG/PNG!');
  }
  const isLt2M = file.size / 1024 / 1024 < 2;
  if (!isLt2M) {
    message.error('Image must smaller than 2MB!');
  }
  return isJpgOrPng && isLt2M;
};
```

```
const App: React.FC = () => {
  const [loading, setLoading] = useState(false)
  const [imageUrl, setImageUrl] = useState<str
```

```
const handleChange: UploadProps['onChange'] =
  if (info.file.status === 'uploading') {
    setLoading(true);
    return;
  }
  if (info.file.status === 'done') {
    // Get this url from response in real world
    getBase64(info.file.originFileObj as RcFile)
      .then(res => {
        setLoading(false);
        setImageUrl(url);
      });
  }
};
```

```
const uploadButton = (
  <div>
    {loading ? <LoadingOutlined /> : <PlusOutlined />}
    <div style={{ marginTop: 8 }}>Upload</div>
  </div>
);

return (
```



Pictures with picture-circle type [🔗](#)

Alternative display for picture-card.

```
import React, { useState } from 'react';
import { PlusOutlined } from '@ant-design/ico
import { Modal, Upload } from 'antd';
import type { RcFile, UploadProps } from 'antc
import type { UploadFile } from 'antd/es/uploc
```

```
const getBase64 = (file: RcFile): Promise<stri
  new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = () => resolve(reader.resul
    reader.onerror = (error) => reject(error);
  });
```

```
const App: React.FC = () => {
  const [previewOpen, setPreviewOpen] = useStc
  const [previewImage, setPreviewImage] = use$
  const [previewTitle, setPreviewTitle] = use$
  const [fileList, setFileList] = useState<Up1
    {
      uid: '-1',
      name: 'image.png',
      status: 'done',
      url: 'https://zos.alipayobjects.com/rmsr
    },
    {
      uid: '-xxx',
      percent: 50,
      name: 'image.png',
      status: 'uploading',
      url: 'https://zos.alipayobjects.com/rmsr
    },
    {
      uid: '-5',
      name: 'image.png',
      status: 'error',
    },
  ],
  []);
```

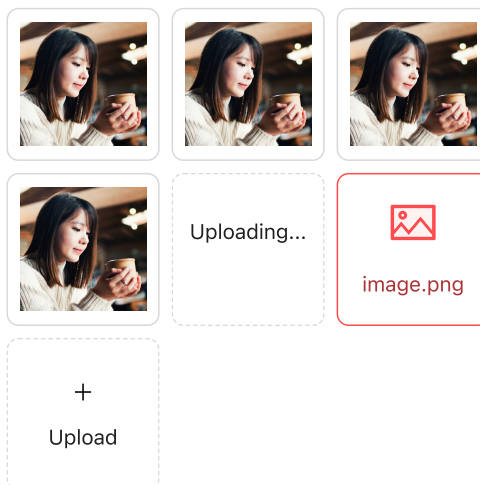
```
const handleCancel = () => setPreviewOpen(fc
```

```
const handlePreview = async (file: UploadFil
  if (!file.url && !file.preview) {
    file.preview = await getBase64(file.orig
  }
```

```
    setPreviewImage(file.url || (file.preview
    setPreviewOpen(true);
    setPreviewTitle(file.name || file.url!.sut
  });
```

```
const handleChange: UploadProps['onChange']
  setFileList(newFileList);
```

```
const uploadButton = (
```



Pictures Wall [🔗](#)

After users upload picture, the thumbnail will be shown in list. The upload button will disappear when count meets limitation.

```
import React, { useState } from 'react';
import { PlusOutlined } from '@ant-design/ico
import { Modal, Upload } from 'antd';
import type { RcFile, UploadProps } from 'antc
import type { UploadFile } from 'antd/es/uploc
```

```
const getBase64 = (file: RcFile): Promise<stri
  new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = () => resolve(reader.resul
    reader.onerror = (error) => reject(error);
  });
```

```
const App: React.FC = () => {
  const [previewOpen, setPreviewOpen] = useStc
  const [previewImage, setPreviewImage] = use$
  const [previewTitle, setPreviewTitle] = use$
  const [fileList, setFileList] = useState<Up1
    {
      uid: '-1',
      name: 'image.png',
      status: 'done',
      url: 'https://zos.alipayobjects.com/rmsr
    },
    {
      uid: '-2',
      name: 'image.png',
      status: 'done',
      url: 'https://zos.alipayobjects.com/rmsr
    },
    {
      uid: '-3',
      name: 'image.png',
      status: 'done',
      url: 'https://zos.alipayobjects.com/rmsr
    },
    {
      uid: '-4',
      name: 'image.png',
      status: 'done',
      url: 'https://zos.alipayobjects.com/rmsr
    },
    {
      uid: '-xxx',
      percent: 50,
    },
  ],
  []);
```



Click or drag file to this area to upload

Support for a single or bulk upload. Strictly prohibited from uploading company data or other banned files.

Drag and Drop

You can drag files to a specific area, to upload.

Alternatively, you can also upload by selecting.

We can upload several files at once in modern browsers by giving the input the `multiple` attribute.

```
import React from 'react';
import { InboxOutlined } from '@ant-design/icons';
import type { UploadProps } from 'antd';
import { message, Upload } from 'antd';

const { Dragger } = Upload;

const props: UploadProps = {
  name: 'file',
  multiple: true,
  action: 'https://www.mocky.io/v2/5cc8019d3000000000000000',
  onChange(info) {
    const { status } = info.file;
    if (status !== 'uploading') {
      console.log(info.file, info.fileList);
    }
    if (status === 'done') {
      message.success(`${info.file.name} file uploaded successfully`);
    } else if (status === 'error') {
      message.error(`${info.file.name} file upload failed`);
    }
  },
  onDrop(e) {
    console.log('Dropped files', e.dataTransfer.files);
  },
};

const App: React.FC = () => (
  <Dragger {...props}>
    <p className="ant-upload-drag-icon">
      <InboxOutlined />
    </p>
    <p className="ant-upload-text">Click or drag file to this area to upload</p>
    <p className="ant-upload-hint">Support for a single or bulk upload. Strictly prohibited from uploading company data or other banned files.</p>
  </Dragger>
);

export default App;
```

📁 Upload

📎 xxx.png

Complete control over file list

You can gain full control over fileList by configuring

`fileList`. You can accomplish all kinds of

customized functions. The following shows two circumstances:

1. limit the number of uploaded files.
2. read from response and show file link.

```
import React, { useState } from 'react';
import { UploadOutlined } from '@ant-design/icons';
import type { UploadProps } from 'antd';
import { Button, Upload } from 'antd';
import type { UploadFile } from 'antd/es/upload/interface';

const App: React.FC = () => {
  const [fileList, setFileList] = useState<UploadFile[]>([
    {
      uid: '-1',
      name: 'xxx.png',
      status: 'done',
      url: 'http://www.baidu.com/xxx.png',
    },
  ]);

  const handleChange: UploadProps['onChange'] = (info) => {
    let newFileList = [...info.fileList];

    // 1. Limit the number of uploaded files
    // Only to show two recent uploaded files,
    newFileList = newFileList.slice(-2);

    // 2. Read from response and show file link
    newFileList = newFileList.map((file) => {
      if (file.response) {
        // Component will show file.url as link
        file.url = file.response.url;
      }
      return file;
    });

    setFileList(newFileList);
  };

  const props = {
    action: 'https://www.mocky.io/v2/5cc8019d3000000000000000',
    onChange: handleChange,
    multiple: true,
  };

  return (
    <Upload {...props} fileList={fileList}>
      <Button icon={<UploadOutlined />}>Upload</Button>
    </Upload>
  );
};

export default App;
```

⬇ Select File

Start Upload

Upload manually [↗](#)

Upload files manually after `beforeUpload` returns

`false`.

```
import React, { useState } from 'react';
import { UploadOutlined } from '@ant-design/icons';
import { Button, message, Upload } from 'antd';
import type { RcFile, UploadFile, UploadProps }
```

```
const App: React.FC = () => {
  const [fileList, setFileList] = useState<UploadFile[]>();
  const [uploading, setUploading] = useState(false);
```

```
  const handleUpload = () => {
    const formData = new FormData();
    fileList.forEach((file) => {
      formData.append('files[]', file as RcFile);
    });
    setUploading(true);
    // You can use any AJAX library you like
    fetch('https://www.mocky.io/v2/5cc8019d3000000800000000', {
      method: 'POST',
      body: formData,
    })
      .then((res) => res.json())
      .then(() => {
        setFileList([]);
        message.success('upload successfully.')
      })
      .catch(() => {
        message.error('upload failed.');
      })
      .finally(() => {
        setUploading(false);
      });
  };
};
```

```
const props: UploadProps = {
  onRemove: (file) => {
    const index = fileList.indexOf(file);
    const newFileList = fileList.slice();
    newFileList.splice(index, 1);
    setFileList(newFileList);
  },
  beforeUpload: (file) => {
    setFileList([...fileList, file]);
```

```
    return false;
  },
  fileList,
};
```

```
return (
  <>
    <Upload {...props}>
      <Button icon={<UploadOutlined />}>Select File</Button>
    </Upload>
    <Button
      type="primary"
      onClick={handleUpload}
      disabled={fileList.length === 0}
      loading={uploading}
      style={{ marginTop: 16 }}
    />
  </>
);
```

⬇ Upload Directory

Upload directory [↗](#)

You can select and upload a whole directory.

```
import React from 'react';
import { UploadOutlined } from '@ant-design/icons';
import { Button, Upload } from 'antd';
```

```
const App: React.FC = () => (
  <Upload action="https://www.mocky.io/v2/5cc8019d3000000800000000">
    <Button icon={<UploadOutlined />}>Upload Directory</Button>
  </Upload>
);
```

```
export default App;
```

⬇ Upload png only

Upload png file only [↗](#)

`beforeUpload` only prevent upload behavior

when return false or reject promise, the prevented

file would still show in file list. Here is the example

you can keep prevented files out of list by return

`UPLOAD.LIST_IGNORE`.

```
import React from 'react';
import { UploadOutlined } from '@ant-design/icons';
import type { UploadProps } from 'antd';
import { Button, message, Upload } from 'antd';
```

```
const props: UploadProps = {
  beforeUpload: (file) => {
    const isPNG = file.type === 'image/png';
    if (!isPNG) {
      message.error(`${file.name} is not a png`);
    }
    return isPNG || Upload.LIST_IGNORE;
  },
  onChange: (info) => {
    console.log(info.fileList);
  },
};
```

```
const App: React.FC = () => (
  <Upload {...props}>
    <Button icon={<UploadOutlined />}>Upload png only</Button>
  </Upload>
);
```

```
export default App;
```

Upload

xxx.png



yyy.png



zzz.png



Upload

xxx.png



yyy.png



zzz.png



Pictures with list style

If uploaded file is a picture, the thumbnail can be shown. `IE8/9` do not support local thumbnail show. Please use `thumbUrl` instead.

```
import React from 'react';
import { UploadOutlined } from '@ant-design/icons';
import { Button, Upload } from 'antd';
import type { UploadFile } from 'antd/es/upload'
```

```
const fileList: UploadFile[] = [
  {
    uid: '0',
    name: 'xxx.png',
    status: 'uploading',
    percent: 33,
  },
  {
    uid: '-1',
    name: 'yyy.png',
    status: 'done',
    url: 'https://zos.alipayobjects.com/rmsportal/...',
    thumbUrl: 'https://zos.alipayobjects.com/rmsportal/...',
  },
  {
    uid: '-2',
    name: 'zzz.png',
    status: 'error',
  },
];
```

```
const App: React.FC = () => (
  <>
    <Upload
      action="https://www.mocky.io/v2/5cc8019c213c237e6843e922"
      listType="picture"
      defaultFileList={fileList}
    />
    <Button icon={UploadOutlined}>Upload</Button>
  </>
);
```

Upload

Customize preview file

Customize local preview. Can handle with non-image format files such as video.

```
import React from 'react';
import { UploadOutlined } from '@ant-design/icons';
import type { UploadProps } from 'antd';
import { Button, Upload } from 'antd';
```

```
const props: UploadProps = {
  action: 'https://jsonplaceholder.typicode.com/posts/1',
  listType: 'picture',
  previewFile(file) {
    console.log('Your upload file:', file);
    // Your process logic. Here we just mock it
    return fetch('https://next.json-generator.com/api/json?...', {
      method: 'POST',
      body: file,
    })
      .then((res) => res.json())
      .then(({ thumbnail }) => thumbnail);
  },
};
```

```
const App: React.FC = () => (
  <Upload {...props}>
    <Button icon={UploadOutlined}>Upload</Button>
  </Upload>
);
```

```
export default App;
```

📁 Upload (Max: 1)

📁 Upload (Max: 3)

Max Count [🔗](#)

Limit files with `maxCount` . Will replace current one when `maxCount` is `1` .

```
import React from 'react';
import { UploadOutlined } from '@ant-design/icons';
import { Button, Space, Upload } from 'antd';

const App: React.FC = () => (
  <Space direction="vertical" style={{ width: 300px }}>
    <Upload
      action="https://www.mocky.io/v2/5cc8019d3000000800000000"
      listType="picture"
      maxCount={1}
    >
      <Button icon={<UploadOutlined />}>Upload</Button>
    </Upload>
    <Upload
      action="https://www.mocky.io/v2/5cc8019d3000000800000000"
      listType="picture"
      maxCount={3}
      multiple
    >
      <Button icon={<UploadOutlined />}>Upload</Button>
    </Upload>
  </Space>
);

export default App;
```

📁 Upload

Transform file before request [🔗](#)

Use `beforeUpload` for transform file before request such as add a watermark.

```
import React from 'react';
import { UploadOutlined } from '@ant-design/icons';
import type { UploadProps } from 'antd';
import { Button, Upload } from 'antd';

const props: UploadProps = {
  action: 'https://www.mocky.io/v2/5cc8019d3000000800000000',
  listType: 'picture',
  beforeUpload(file) {
    return new Promise((resolve) => {
      const reader = new FileReader();
      reader.readAsDataURL(file);
      reader.onload = () => {
        const img = document.createElement('img');
        img.src = reader.result as string;
        img.onload = () => {
          const canvas = document.createElement('canvas');
          canvas.width = img.naturalWidth;
          canvas.height = img.naturalHeight;
          const ctx = canvas.getContext('2d');
          ctx.drawImage(img, 0, 0);
          ctx.fillStyle = 'red';
          ctx.textBaseline = 'middle';
          ctx.font = '33px Arial';
          ctx.fillText('Ant Design', 20, 20);
          canvas.toBlob((result) => resolve(result));
        };
      };
    });
  },
};

const App: React.FC = () => (
  <Upload {...props}>
    <Button icon={<UploadOutlined />}>Upload</Button>
  </Upload>
);

export default App;
```


Photos:  Click to Upload

Aliyun OSS

Use Aliyun OSS upload example.

```
import React, { useEffect, useState } from 'react'
import { UploadOutlined } from '@ant-design/icons'
import type { UploadProps } from 'antd'
import { Button, Form, message, Upload } from 'antd'
import type { UploadFile } from 'antd/es/upload'

interface OSSDataType {
  dir: string;
  expire: string;
  host: string;
  accessId: string;
  policy: string;
  signature: string;
}

interface AliyunOSSUploadProps {
  value?: UploadFile[];
  onChange?: (fileList: UploadFile[]) => void;
}

const AliyunOSSUpload = ({ value, onChange }: AliyunOSSUploadProps): React.ReactNode => {
  // Mock get OSS api
  // https://help.aliyun.com/document_detail/31624.html
  const mockGetOSSData = () => ({
    dir: 'user-dir/',
    expire: '1577811661',
    host: 'http://www.mocky.io/v2/5cc8019d3000000980',
    accessId: 'c2hnb2RhaG9uZW==',
    policy: 'eG14aWhhaGFrdWt1ZGFkYQ==',
    signature: 'ZGFob25nc2hnbw==',
  });

  const init = async () => {
    try {
      const result = await mockGetOSSData();
      setOSSData(result);
    } catch (error) {
      message.error(error);
    }
  };

  useEffect(() => {
    init();
  }, []);

  const handleChange: UploadProps['onChange'] = (e) => {
    console.log('Aliyun OSS:', e.fileList);
    onChange?.(e.fileList);
  };

  const onRemove = (file: UploadFile) => {
    const files = (value || []).filter((v) => v.uid !== file.uid);

    if (onChange) {
      onChange(files);
    }
  };

  const getExtraData: UploadProps['data'] = {
    key: file.url,
    OSSAccessKeyId: OSSData?.accessId,
  };
}
```

 Upload

 xxx.png

 yyy.png

 zzz.png



custom action icon

Use `showUploadList` for custom action icons of files.

```
import React from 'react';
import { StarOutlined, UploadOutlined } from '@ant-design/icons';
import type { UploadProps } from 'antd';
import { Button, Upload } from 'antd';

const props: UploadProps = {
  action: 'https://www.mocky.io/v2/5cc8019d3000000980',
  onChange({ file, fileList }) {
    if (file.status !== 'uploading') {
      console.log(file, fileList);
    }
  },
  defaultFileList: [
    {
      uid: '1',
      name: 'xxx.png',
      status: 'done',
      response: 'Server Error 500', // custom
      url: 'http://www.baidu.com/xxx.png',
    },
    {
      uid: '2',
      name: 'yyy.png',
      status: 'done',
      url: 'http://www.baidu.com/yyy.png',
    },
    {
      uid: '3',
      name: 'zzz.png',
      status: 'error',
      response: 'Server Error 500', // custom
      url: 'http://www.baidu.com/zzz.png',
    },
  ],
  showUploadList: {
    showDownloadIcon: true,
    downloadIcon: 'Download',
    showRemoveIcon: true,
    removeIcon: <StarOutlined onClick={e} => {
      // custom remove icon
    }
  },
};

const App: React.FC = () => {
  <Upload {...props}>
    <Button icon={<UploadOutlined />}>Upload</Button>
  </Upload>
};

export default App;
```

📁 Click to Upload

📎 [image1.png](#)
📎 [image2.png](#)
📎 [image3.png](#)
📎 [image4.png](#)
📎 [image.png](#)



Drag sorting of uploadList [🔗](#)

By using `itemRender`, we can integrate upload with [dnd-kit](#) to implement drag sorting of uploadList.

```
import { UploadOutlined } from '@ant-design/icons';
import type { DragEndEvent } from '@dnd-kit/core';
import { DndContext, PointerSensor, useSensor, useSensors, arrayMove, SortableContext, useSortable, verticalListSortingStrategy, } from '@dnd-kit/sortable';
import { CSS } from '@dnd-kit/utilities';
import { css } from '@emotion/css';
import { Button, Upload } from 'antd';
import type { UploadFile, UploadProps } from 'antd/lib/upload';
import React, { useState } from 'react';

interface DraggableUploadListItemProps {
  originNode: React.ReactElement<any, string> | null;
  file: UploadFile<any>;
}

const DraggableUploadListItem = ({ originNode, const { attributes, listeners, setNodeRef, transform, id: file.uid, }}) => {

  const style: React.CSSProperties = {
    transform: CSS.Transform.toString(transform),
    transition: 'transform 0.2s ease-in-out',
    cursor: 'move',
  };

  // prevent preview event when drag end
  const className = isDragging ? 'css' : '';

  return (
    <div ref={setNodeRef} style={style} className={className} /* hide error tooltip when dragging */ {file.status === 'error' && isDragging ? 'error' : ''}>
      {originNode}
    </div>
  );
};

const App: React.FC = () => {
  const [fileList, setFileList] = useState<UploadFile[]>([
    {
      uid: '-1',
      name: 'image1.png',
      status: 'done',
    },
  ]);
};
```



+ Upload

Crop image before uploading [🔗](#)

Use [antd-img-crop](#) to crop image before uploading.

```
import { Upload } from 'antd';
import ImgCrop from 'antd-img-crop';
import type { RcFile, UploadFile, UploadProps } from 'antd/lib/upload';
import React, { useState } from 'react';

const App: React.FC = () => {
  const [fileList, setFileList] = useState<UploadFile[]>([
    {
      uid: '-1',
      name: 'image.png',
      status: 'done',
      url: 'https://zos.alipayobjects.com/rmsportal/jkjfKkskQggJLvzDyvvMbZqcLbfKq/.jpeg',
    },
  ]);

  const onChange: UploadProps['onChange'] = ({ fileList: newFileList }) => {
    setFileList(newFileList);
  };

  const onPreview = async (file: UploadFile) => {
    let src = file.url as string;
    if (!src) {
      src = await new Promise((resolve) => {
        const reader = new FileReader();
        reader.readAsDataURL(file.originFileObj);
        reader.onload = () => resolve(reader.result);
      });
    }
    const image = new Image();
    image.src = src;
    const imgWindow = window.open(src);
    imgWindow?.document.write(image.outerHTML);
  };

  return (
    <div>
      <ImgCrop rotationSlider>
        <Upload
          action="https://www.mocky.io/v2/5cc8019e323e4239858e9139"
          listType="picture-card"
          fileList={fileList}
          onChange={onChange}
          onPreview={onPreview}
        />
        {fileList.length < 5 && '+ Upload'}
      </ImgCrop>
    </div>
  );
};

export default App;
```

 Click to Upload

Customize Progress Bar

Use `progress` for customize progress bar.

```
import React from 'react';
import { UploadOutlined } from '@ant-design/icons';
import type { UploadProps } from 'antd';
import { Button, message, Upload } from 'antd'

const props: UploadProps = {
  name: 'file',
  action: 'https://www.mocky.io/v2/5cc8019d3000000000000000',
  headers: {
    authorization: 'authorization-text',
  },
  onChange(info) {
    if (info.file.status !== 'uploading') {
      console.log(info.file, info.fileList);
    }
    if (info.file.status === 'done') {
      message.success(`${info.file.name} file uploaded successfully`);
    } else if (info.file.status === 'error') {
      message.error(`${info.file.name} file upload failed.`);
    }
  },
  progress: {
    strokeColor: {
      '0%': '#108ee9',
      '100%': '#87d068',
    },
    strokeWidth: 3,
    format: (percent) => percent && `${parseFloat(percent).toFixed(2)}%`,
  },
};

const App: React.FC = () => (
  <Upload {...props}>
    <Button icon={<UploadOutlined />}>Click to Upload</Button>
  </Upload>
);

export default App;
```

```
    <upload
      action="https://www.mocky.io/v2/5cc8019d3000000000000000"
      fileList={fileList}
      onChange={onChange}
      itemRender={(originNode, file) => (
        <DraggableUploadListItem originNode={originNode} file={file} />
      )}
    >
      <Button icon={<UploadOutlined />}>Click to Upload</Button>
    </Upload>
  </SortableContext>
</DndContext>
);
};

export default App;
```

API

| Property | Description | Type | Default |
|----------|---|------------------------------------|---------|
| accept | File types that can be accepted. See input accept Attribute | string | - |
| action | Uploading URL | string (file) => Promise<string> | - |

| Property | Description | Type | Default |
|-----------------|--|--|---|
| beforeUpload | Hook function which will be executed before uploading. Uploading will be stopped with <code>false</code> or a rejected Promise returned. When returned value is <code>Upload.LIST_IGNORE</code> , the list of files that have been uploaded will ignore it. Warning: this function is not supported in IE9 | <code>(file, fileList) => boolean Promise<File> Upload.LIST_IGNORE</code> | - |
| customRequest | Override for the default xhr behavior allowing for additional customization and ability to implement your own XMLHttpRequest | <code>function</code> | - |
| data | Uploading extra params or function which can return uploading extra params | <code>object (file) => object Promise<object></code> | - |
| defaultFileList | Default list of files that have been uploaded | <code>object[]</code> | - |
| directory | Support upload whole directory (caniuse) | <code>boolean</code> | <code>false</code> |
| disabled | Disable upload button | <code>boolean</code> | <code>false</code> |
| fileList | List of files that have been uploaded (controlled). Here is a common issue #2423 when using it | <code>UploadFile[]</code> | - |
| headers | Set request headers, valid above IE10 | <code>object</code> | - |
| iconRender | Custom show icon | <code>(file: UploadFile, listType?: UploadListType) => ReactNode</code> | - |
| isImageUrl | Customize if render <code></code> in thumbnail | <code>(file: UploadFile) => boolean</code> | (inside implementation) |
| itemRender | Custom item of uploadList | <code>(originNode: ReactElement,</code> | - |

| Property | Description | Type | Default |
|-----------------------|--|--|--|
| | | <pre>file: UploadFile, fileList: object[], actions: { download: function, preview: function, remove: function }) => React.ReactNode</pre> | |
| listType | Built-in stylesheets, support for four types: <code>text</code> , <code>picture</code> , <code>picture-card</code> or <code>picture-circle</code> | string | <code>text</code> |
| maxCount | Limit the number of uploaded files. Will replace current one when <code>maxCount</code> is <code>1</code> | number | - |
| method | The http method of upload request | string | <code>post</code> |
| multiple | Whether to support selected multiple file. <code>IE10+</code> supported. You can select multiple files with CTRL holding down while multiple is set to be true | boolean | false |
| name | The name of uploading file | string | <code>file</code> |
| openFileDialogOnClick | Click open file dialog | boolean | true |
| previewFile | Customize preview file logic | <pre>(file: File Blob) => Promise<dataURL: string></pre> | - |
| progress | Custom progress bar | ProgressProps (support <code>type="line"</code> only) | <pre>{ strokeWidth: 2, showInfo: false }</pre> |
| showUploadList | Whether to show default upload list, could be an object to specify <code>showPreviewIcon</code> , <code>showRemoveIcon</code> , <code>showDownloadIcon</code> , <code>removeIcon</code> and <code>downloadIcon</code> individually | <pre>boolean { showPreviewIcon?: boolean, showDownloadIcon?: boolean, showRemoveIcon?: boolean, previewIcon?: ReactNode (file: UploadFile) => ReactNode, removeIcon?: ReactNode (file: UploadFile) => ReactNode,</pre> | true |

| Property | Description | Type | Default |
|------------------------------|---|--|--------------------|
| | | <code>downloadIcon?: ReactNode (file: UploadFile) => ReactNode }</code> | |
| <code>withCredentials</code> | The ajax upload with cookie sent | <code>boolean</code> | <code>false</code> |
| <code>onChange</code> | A callback function, can be executed when uploading state is changing, see onChange | <code>function</code> | <code>-</code> |
| <code>onDrop</code> | A callback function executed when files are dragged and dropped into upload area | <code>(event: React.DragEvent) => void</code> | <code>-</code> |
| <code>onDownload</code> | Click the method to download the file, pass the method to perform the method logic, do not pass the default jump to the new TAB | <code>function(file): void</code> | (Jump to new TAB) |
| <code>onPreview</code> | A callback function, will be executed when file link or preview icon is clicked | <code>function(file)</code> | <code>-</code> |
| <code>onRemove</code> | A callback function, will be executed when removing file button is clicked, remove event will be prevented when return value is false or a Promise which resolve(false) or reject | <code>function(file): boolean Promise</code> | <code>-</code> |

UploadFile

Extends File with additional props.

| Property | Description | Type | Default | Version |
|--------------------------|--------------------------|--|----------------|----------------|
| <code>crossOrigin</code> | CORS settings attributes | <code>'anonymous' 'use- credentials' ''</code> | <code>-</code> | 4.20.0 |
| <code>name</code> | File name | <code>string</code> | <code>-</code> | <code>-</code> |
| <code>percent</code> | Upload progress percent | <code>number</code> | <code>-</code> | <code>-</code> |

| Property | Description | Type | Default | Version |
|----------|---|---|---------|---------|
| status | Upload status. Show different style when configured | <div>error </div> <div>success </div> <div>done </div> <div>uploading </div> <div>removed</div> | - | - |
| thumbUrl | Thumb image url | string | - | - |
| uid | unique id. Will auto generate when not provided | string | - | - |
| url | Download url | string | - | - |

onChange

The function will be called when uploading is in progress, completed or failed.

When uploading state change, it returns:

```
{
  file: { /* ... */ },
  fileList: [ /* ... */ ],
  event: { /* ... */ },
}
```

- 1. `file` File object for the current operation.

```
{
  uid: 'uid',      // unique identifier, negative is recommend, to prevent interference with
  name: 'xx.png',  // file name
  status: 'done',  // options: uploading, done, error, removed. Intercepted file by beforeUplc
  response: '{"status": "success"}', // response from server
  linkProps: '{"download": "image"}', // additional html props of file link
  xhr: 'XMLHttpRequest{ ... }', // XMLHttpRequest Header
}
```

- 2. `fileList` current list of files
- 3. `event` response from server, including uploading progress, supported by advanced browsers.

Design Token

▼ Global Token

| Token Name | Description | Type | Default Value |
|-------------|---|-------------------|---|
| colorBgMask | The background color of the mask, used to cover the content below the mask, Modal, Drawer and other components use this token | <div>string</div> | <div><input type="checkbox"/> rgba(0, 0, 0, 0.45)</div> |
| colorBorder | Default border color, used to separate different elements, such as: form separator, card separator, etc. | <div>string</div> | <div><input type="checkbox"/> #d9d9d9</div> |
| colorError | Used to represent the visual elements of the operation failure, such as the error | <div>string</div> | <div><input type="checkbox"/> #ff4d4f</div> |

| Token Name | Description | Type | Default Value |
|----------------------|---|--------|---|
| | Button, error Result component, etc. | | |
| colorErrorBg | The background color of the error state. | string | <code>#fff2f0</code> |
| colorFillAlter | Control the alternative background color of element. | string | <code>rgba(0, 0, 0, 0.02)</code> |
| colorPrimary | Brand color is one of the most direct visual elements to reflect the characteristics and communication of the product. After you have selected the brand color, we will automatically generate a complete color palette and assign it effective design semantics. | string | <code>#1677ff</code> |
| colorPrimaryHover | Hover state under the main color gradient. | string | <code>#4096ff</code> |
| colorText | Default text color which comply with W3C standards, and this color is also the darkest neutral color. | string | <code>rgba(0, 0, 0, 0.88)</code> |
| colorTextDescription | Control the font color of text description. | string | <code>rgba(0, 0, 0, 0.45)</code> |
| colorTextDisabled | Control the color of text in disabled state. | string | <code>rgba(0, 0, 0, 0.25)</code> |
| colorTextHeading | Control the font color of heading. | string | <code>rgba(0, 0, 0, 0.88)</code> |
| colorTextLightSolid | Control the highlight color of text with background color, such as the text in Primary Button components. | string | <code>#fff</code> |
| borderRadiusLG | LG size border radius, used in some large border radius components, such as Card, Modal and other components. | number | 8 |
| controlHeightLG | LG component height | number | 40 |
| controlItemBgHover | Control the background color of control component item when hovering. | string | <code>rgba(0, 0, 0, 0.04)</code> |
| fontFamily | The font family of Ant Design prioritizes the default interface font of the system, and provides a set of alternative font libraries that are suitable for screen display to maintain the readability and readability of the font under different platforms and browsers, reflecting the friendly, stable and professional characteristics. | string | -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, 'Helvetica Neue', Arial, 'Noto Sans', sans-serif, 'Apple Color Emoji', 'Segoe UI Emoji', 'Segoe UI Symbol', 'Noto Color Emoji' |
| fontSize | The most widely used font size in the design system, from which the text gradient will be derived. | number | 14 |
| fontSizeHeading2 | Font size of h2 tag. | number | 30 |
| fontSizeHeading3 | Font size of h3 tag. | number | 24 |
| fontSizeLG | Large font size | number | 16 |

| Token Name | Description | Type | Default Value |
|---------------------|--|--------|--------------------------------------|
| lineHeight | Line height of text. | number | 1.5714285714285714 |
| lineType | Border style of base components | string | solid |
| lineWidth | Border width of base components | number | 1 |
| margin | Control the margin of an element, with a medium size. | number | 16 |
| marginXL | Control the margin of an element, with an extra-large size. | number | 32 |
| marginXS | Control the margin of an element, with a small size. | number | 8 |
| marginXXS | Control the margin of an element, with the smallest size. | number | 4 |
| motionDurationMid | Motion speed, medium speed. Used for medium element animation interaction. | string | 0.2s |
| motionDurationSlow | Motion speed, slow speed. Used for large element animation interaction. | string | 0.3s |
| motionEaseInOut | Preset motion curve. | string | cubic-bezier(0.645, 0.045, 0.355, 1) |
| motionEaseInOutCirc | Preset motion curve. | string | cubic-bezier(0.78, 0.14, 0.15, 0.86) |
| padding | Control the padding of the element. | number | 16 |
| paddingSM | Control the small padding of the element. | number | 12 |
| paddingXS | Control the extra small padding of the element. | number | 8 |

FAQ

How do I implement upload server side?

- You can consult [jQuery-File-Upload](#) about how to implement server side upload interface.
- There is a mock example of [express](#) in rc-upload.

I want to display download links.

Please set property `url` of each item in `fileList` to control content of link.

How to use `customRequest` ?

See <https://github.com/react-component/upload#customrequest>.

Why will the `fileList` that's in control not trigger `onChange` `status` update when the file is not in the list?

`onChange` will only trigger when the file is in the list, it will ignore any events removed from the list. Please note that there does exist a bug which makes an event still trigger even when the file is not in the list before `4.13.0`.

Why does `onChange` sometimes return File object and other times return `{ originFileObj: File }`?

For compatible case, we return File object when `beforeUpload` return `false`. It will merge to `{ originFileObj: File }` in next major version. Current version is compatible to get origin file by `info.file.originFileObj`. You can change this before major release.

Why sometime Chrome can not upload?

Chrome update will also break native upload. Please restart chrome to finish the upload work. Ref:

- [#32672](#)
- [#32913](#)
- [#33988](#)