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

Upload by clicking

```
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://660d2bd96ddfa2943b33731c.mockapi.io/api/upload',
  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;
```

Avatar

```
import React, { useState } from 'react';
import { LoadingOutlined, PlusOutlined } from '@ant-design/icons';
import { Flex, message, Upload } from 'antd';
import type { GetProp, UploadProps } from 'antd';

type FileType = Parameters<GetProp<UploadProps, 'beforeUpload'>>[0];

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

const beforeUpload = (file: FileType) => {
  const isJpgOrPng = file.type === 'image/jpeg' || file.type ===
'image/png';
  if (!isJpgOrPng) {
    message.error('You can only upload JPG/PNG file!');
  }
  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<string>();

  const handleChange: UploadProps['onChange'] = (info) => {
    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 FileType, (url) => {
        setLoading(false);
        setImageUrl(url);
      });
    }
  };

  const uploadButton = (
    <button style={{ border: 0, background: 'none' }} type="button">
```

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

  return (
    <Flex gap="middle" wrap>
      <Upload
        name="avatar"
        listType="picture-card"
        className="avatar-uploader"
        showUploadList={false}
        action="https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload"
        beforeUpload={beforeUpload}
        onChange={handleChange}
      >
        {imageUrl ? <img src={imageUrl} alt="avatar" style={{ width: '100%'
}} /> : uploadButton}
      </Upload>
      <Upload
        name="avatar"
        listType="picture-circle"
        className="avatar-uploader"
        showUploadList={false}
        action="https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload"
        beforeUpload={beforeUpload}
        onChange={handleChange}
      >
        {imageUrl ? <img src={imageUrl} alt="avatar" style={{ width: '100%'
}} /> : uploadButton}
      </Upload>
    </Flex>
  );
};

export default App;
```

Default Files

```
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://660d2bd96ddfa2943b33731c.mockapi.io/api/upload',
  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 error message to show
      url: 'http://www.baidu.com/zzz.png',
    },
  ],
};

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

export default App;
```

Pictures Wall

```
import React, { useState } from 'react';
import { PlusOutlined } from '@ant-design/icons';
import { Image, Upload } from 'antd';
import type { GetProp, UploadFile, UploadProps } from 'antd';

type FileType = Parameters<GetProp<UploadProps, 'beforeUpload'>>[0];
```

```
const getBase64 = (file: FileType): Promise<string> =>
  new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = () => resolve(reader.result as string);
    reader.onerror = (error) => reject(error);
  });

const App: React.FC = () => {
  const [previewOpen, setPreviewOpen] = useState(false);
  const [previewImage, setPreviewImage] = useState('');
  const [fileList, setFileList] = useState<UploadFile[]>([
    {
      uid: '-1',
      name: 'image.png',
      status: 'done',
      url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    },
    {
      uid: '-2',
      name: 'image.png',
      status: 'done',
      url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    },
    {
      uid: '-3',
      name: 'image.png',
      status: 'done',
      url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    },
    {
      uid: '-4',
      name: 'image.png',
      status: 'done',
      url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    },
    {
      uid: '-xxx',
      percent: 50,
      name: 'image.png',
      status: 'uploading',
      url:
```

```
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    },
    {
      uid: '-5',
      name: 'image.png',
      status: 'error',
    },
  ]);

  const handlePreview = async (file: UploadFile) => {
    if (!file.url && !file.preview) {
      file.preview = await getBase64(file.originFileObj as FileType);
    }

    setPreviewImage(file.url || (file.preview as string));
    setPreviewOpen(true);
  };

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

  const uploadButton = (
    <button style={{ border: 0, background: 'none' }} type="button">
      <PlusOutlined />
      <div style={{ marginTop: 8 }}>Upload</div>
    </button>
  );
  return (
    <>
      <Upload
        action="https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload"
        listType="picture-card"
        fileList={fileList}
        onPreview={handlePreview}
        onChange={handleChange}
      >
        {fileList.length >= 8 ? null : uploadButton}
      </Upload>
      {previewImage && (
        <Image
          wrapperStyle={{ display: 'none' }}
          preview={{
            visible: previewOpen,
            onVisibleChange: (visible) => setPreviewOpen(visible),
            afterOpenChange: (visible) => !visible && setPreviewImage(''),
          }}
          src={previewImage}
        />
      )}
    </>
  );
};

export default App;
```

Pictures with picture-circle type

```
import React, { useState } from 'react';
import { PlusOutlined } from '@ant-design/icons';
import { Image, Upload } from 'antd';
import type { GetProp, UploadFile, UploadProps } from 'antd';

type FileType = Parameters<GetProp<UploadProps, 'beforeUpload'>>[0];

const getBase64 = (file: FileType): Promise<string> =>
  new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = () => resolve(reader.result as string);
    reader.onerror = (error) => reject(error);
  });

const App: React.FC = () => {
  const [previewOpen, setPreviewOpen] = useState(false);
  const [previewImage, setPreviewImage] = useState('');
  const [fileList, setFileList] = useState<UploadFile[]>([
    {
      uid: '-1',
      name: 'image.png',
      status: 'done',
      url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    },
    {
      uid: '-xxx',
      percent: 50,
      name: 'image.png',
      status: 'uploading',
      url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    },
    {
      uid: '-5',
      name: 'image.png',
      status: 'error',
    },
  ]);

  const handlePreview = async (file: UploadFile) => {
    if (!file.url && !file.preview) {
      file.preview = await getBase64(file.originFileObj as FileType);
    }

    setPreviewImage(file.url || (file.preview as string));
    setPreviewOpen(true);
  };

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

  const uploadButton = (
    <button style={{ border: 0, background: 'none' }} type="button">
      <PlusOutlined />
      <div style={{ marginTop: 8 }}>Upload</div>
    </button>
  );
  return (
    <>
      <Upload
        action="https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload"
        listType="picture-circle"
        fileList={fileList}
        onPreview={handlePreview}
        onChange={handleChange}
      >
        {fileList.length >= 8 ? null : uploadButton}
      </Upload>
      {previewImage && (
        <Image
          wrapperStyle={{ display: 'none' }}
          preview={{
            visible: previewOpen,
            onVisibleChange: (visible) => setPreviewOpen(visible),
            afterOpenChange: (visible) => !visible && setPreviewImage(''),
          }}
          src={previewImage}
        />
      )}
    </>
  );
};

export default App;
```

Complete control over file list

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

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, and old ones will be
replaced by the new
    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://660d2bd96ddfa2943b33731c.mockapi.io/api/upload',
    onChange: handleChange,
    multiple: true,
  };
  return (
    <Upload {...props} fileList={fileList}>
      <Button icon={<UploadOutlined />}>Upload</Button>
    </Upload>
  );
};

export default App;
```

Drag and Drop

```
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://660d2bd96ddfa2943b33731c.mockapi.io/api/upload',
  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 directory

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

const App: React.FC = () => (
  <Upload action="https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload"
directory>
    <Button icon={<UploadOutlined />}>Upload Directory</Button>
  </Upload>
);

export default App;
```

Upload manually

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

type FileType = Parameters<GetProp<UploadProps, 'beforeUpload'>>[0];

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 FileType);
    });
    setUploading(true);
    // You can use any AJAX library you like
    fetch('https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload', {
      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 }}
      >
        {uploading ? 'Uploading' : 'Start Upload'}
      </Button>
    </>
  );
};

export default App;
```

Upload png file only

```
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 file`);
    }
    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;
```

Pictures with list style

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

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/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    thumbUrl:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
  },
  {
    uid: '-2',
    name: 'zzz.png',
    status: 'error',
  },
];

const App: React.FC = () => (
  <Upload
    action="https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload"
    listType="picture"
    defaultFileList={fileList}
  >
    <Button type="primary" icon={<UploadOutlined />}>
      Upload
    </Button>
  </Upload>
);

export default App;
```

Customize preview file

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

const props: UploadProps = {
  action: '//jsonplaceholder.typicode.com/posts/',
  listType: 'picture',
  previewFile(file) {
    console.log('Your upload file:', file);
    // Your process logic. Here we just mock to the same file
    return fetch('https://next.json-generator.com/api/json/get/4ytyBoLK8',
{
      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;
```

Max Count

```
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: '100%' }} size="large">
    <Upload
      action="https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload"
      listType="picture"
      maxCount={1}
    >
      <Button icon={<UploadOutlined />}>Upload (Max: 1)</Button>
    </Upload>
    <Upload
      action="https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload"
      listType="picture"
      maxCount={3}
      multiple
    >
      <Button icon={<UploadOutlined />}>Upload (Max: 3)</Button>
    </Upload>
  </Space>
);
```

```
export default App;
```

Transform file before request

```
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://660d2bd96ddfa2943b33731c.mockapi.io/api/upload',
  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 as Blob));
        };
      };
    });
  },
};

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

export default App;
```

Aliyun OSS

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

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

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

// Mock get OSS api
// https://help.aliyun.com/document_detail/31988.html
const mockOSSData = () => {
  const mockData = {
    dir: 'user-dir/',
    expire: '1577811661',
    host: 'https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload',
    accessId: 'c2hhb2RhaG9uZw==',
    policy: 'eGl4aWhhaGFrdWt1ZGFkYQ==',
    signature: 'ZGFob25nc2hhbw==',
  };
  return Promise.resolve(mockData);
};

const AliyunOSSUpload: React.FC<Readonly<AliyunOSSUploadProps>> = ({ value,
onChange }) => {
  const { message } = App.useApp();

  const [OSSData, setOSSData] = useState<OSSDataType>();

  const init = async () => {
    try {
      const result = await mockOSSData();
      setOSSData(result);
    } catch (err) {
      if (err instanceof Error) {
        message.error(err.message);
      }
    }
  };

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

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

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

  const getExtraData: UploadProps['data'] = (file) => ({
    key: file.url,
    OSSAccessKeyId: OSSData?.accessId,
    policy: OSSData?.policy,
    Signature: OSSData?.signature,
  });

  const beforeUpload: UploadProps['beforeUpload'] = async (file) => {
    if (!OSSData) {
      return false;
    }

    const expire = Number(OSSData.expire) * 1000;

    if (expire < Date.now()) {
      await init();
    }

    const suffix = file.name.slice(file.name.lastIndexOf('.'));
    const filename = Date.now() + suffix;
    // @ts-ignore
    file.url = OSSData.dir + filename;

    return file;
  };
  const uploadProps: UploadProps = {
    name: 'file',
    fileList: value,
    action: OSSData?.host,
    onChange: handleChange,
    onRemove,
    data: getExtraData,
    beforeUpload,
  };

  return (
    <Upload {...uploadProps}>
      <Button icon={<UploadOutlined />}>Click to Upload</Button>
    </Upload>
  );
};

const Demo: React.FC = () => (
  <Form labelCol={{ span: 4 }}>
    <Form.Item label="Photos" name="photos">
      <AliyunOSSUpload />
    </Form.Item>
  </Form>
);

export default Demo;
```

custom show icon

Debug

```
import React, { useState } from 'react';
import {
  FileExcelTwoTone,
  FilePdfTwoTone,
  FileWordTwoTone,
  LoadingOutlined,
  PaperClipOutlined,
  PictureTwoTone,
  PlusOutlined,
} from '@ant-design/icons';
import { Image, Upload } from 'antd';
import type { GetProp, UploadFile, UploadProps } from 'antd';

type FileType = Parameters<GetProp<UploadProps, 'beforeUpload'>>[0];
```

```
const getBase64 = (file: FileType): Promise<string> =>
  new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = () => resolve(reader.result as string);
    reader.onerror = (error) => reject(error);
  });

const App: React.FC = () => {
  const [previewOpen, setPreviewOpen] = useState(false);
  const [previewImage, setPreviewImage] = useState('');
  const [fileList, setFileList] = useState<UploadFile[]>([
    {
      uid: '-2',
      name: 'pdf.pdf',
      status: 'done',
      url:
'http://cdn07.foxitsoftware.cn/pub/foxit/cpdf/FoxitCompanyProfile.pdf',
    },
    {
      uid: '-3',
      name: 'doc.doc',
      status: 'done',
      url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.doc
    },
    {
      uid: '-4',
      name: 'image.png',
      status: 'error',
    },
    {
      uid: '-5',
      name: 'pdf.pdf',
      status: 'error',
    },
    {
      uid: '-6',
      name: 'doc.doc',
      status: 'error',
    },
  ]);

  const handlePreview = async (file: UploadFile) => {
    if (!file.url && !file.preview) {
      file.preview = await getBase64(file.originFileObj as FileType);
    }

    setPreviewOpen(true);
    setPreviewImage(file.url || (file.preview as string));
  };

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

  const handleIconRender: UploadProps['iconRender'] = (file, listType) => {
    const fileSufIconList = [
      { type: <FilePdfTwoTone />, suf: ['.pdf'] },
      { type: <FileExcelTwoTone />, suf: ['.xlsx', '.xls', '.csv'] },
      { type: <FileWordTwoTone />, suf: ['.doc', '.docx'] },
      {
        type: <PictureTwoTone />,
        suf: ['.webp', '.svg', '.png', '.gif', '.jpg', '.jpeg', '.jfif',
'.bmp', '.dpg'],
      },
    ];
    // console.log(1, file, listType);
    let icon = file.status === 'uploading' ? <LoadingOutlined /> :
<PaperClipOutlined />;
    if (listType === 'picture' || listType === 'picture-card' || listType
=== 'picture-circle') {
      if (listType === 'picture-card' && file.status === 'uploading') {
        icon = <LoadingOutlined />; // or icon = 'uploading...';
      } else {
        fileSufIconList.forEach((item) => {
          if
(item.suf.includes(file.name.slice(file.name.lastIndexOf('.')))) {
            icon = item.type;
          }
        });
      }
    }
    return icon;
  };

  const uploadButton = (
    <button style={{ border: 0, background: 'none' }} type="button">
      <PlusOutlined />
      <div style={{ marginTop: 8 }}>Upload</div>
    </button>
  );
  return (
    <>
      <Upload
        action="https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload"
        listType="picture-card"
        fileList={fileList}
        onPreview={handlePreview}
        onChange={handleChange}
        iconRender={handleIconRender}
      >
        {fileList.length >= 8 ? null : uploadButton}
      </Upload>
      {previewImage && (
        <Image
          wrapperStyle={{ display: 'none' }}
          preview={{
            visible: previewOpen,
            onVisibleChange: (visible) => setPreviewOpen(visible),
            afterOpenChange: (visible) => !visible && setPreviewImage(''),
          }}
          src={previewImage}
        />
      )}
    </>
  );
};

export default App;
```

Custom action icon and extra info

```
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://660d2bd96ddfa2943b33731c.mockapi.io/api/upload',
  onChange({ file, fileList }) {
    if (file.status !== 'uploading') {
      console.log(file, fileList);
    }
  },
  defaultFileList: [
    {
      uid: '1',
      name: 'xxx.png',
      size: 1234567,
      status: 'done',
      response: 'Server Error 500', // custom error message to show
      url: 'http://www.baidu.com/xxx.png',
    },
    {
      uid: '2',
      name: 'yyy.png',
      size: 1234567,
      status: 'done',
      url: 'http://www.baidu.com/yyy.png',
    },
    {
      uid: '3',
      name: 'zzz.png',
      size: 1234567,
      status: 'error',
      response: 'Server Error 500', // custom error message to show
      url: 'http://www.baidu.com/zzz.png',
    },
  ],
  showUploadList: {
    extra: ({ size = 0 }) => (
      <span style={{ color: '#cccccc' }}>({(size / 1024 /
1024).toFixed(2)}MB)</span>
    ),
    showDownloadIcon: true,
    downloadIcon: 'Download',
    showRemoveIcon: true,
    removeIcon: <StarOutlined onClick={(e) => console.log(e, 'custom
removeIcon event')} />,
  },
};

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

export default App;
```

Drag sorting of uploadList

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

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

const DraggableUploadListItem = ({ originNode, file }:
DraggableUploadListItemProps) => {
  const { attributes, listeners, setNodeRef, transform, transition,
isDragging } = useSortable({
    id: file.uid,
  });

  const style: React.CSSProperties = {
    transform: CSS.Translate.toString(transform),
    transition,
    cursor: 'move',
  };

  return (
    <div
      ref={setNodeRef}
      style={style}
      // prevent preview event when drag end
      className={isDragging ? 'is-dragging' : ''}
      {...attributes}
      {...listeners}
    >
      {/* hide error tooltip when dragging */}
      {file.status === 'error' && isDragging ? originNode.props.children :
originNode}
    </div>
  );
};

const App: React.FC = () => {
  const [fileList, setFileList] = useState<UploadFile[]>([
    {
      uid: '-1',
      name: 'image1.png',
      status: 'done',
      url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    },
    {
      uid: '-2',
      name: 'image2.png',
      status: 'done',
      url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    },
    {
      uid: '-3',
      name: 'image3.png',
      status: 'done',
      url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    },
    {
      uid: '-4',
      name: 'image4.png',
      status: 'done',
      url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
    },
    {
      uid: '-5',
      name: 'image.png',
      status: 'error',
    },
  ]);

  const sensor = useSensor(PointerSensor, {
    activationConstraint: { distance: 10 },
  });

  const onDragEnd = ({ active, over }: DragEndEvent) => {
    if (active.id !== over?.id) {
      setFileList((prev) => {
        const activeIndex = prev.findIndex((i) => i.uid === active.id);
        const overIndex = prev.findIndex((i) => i.uid === over?.id);
        return arrayMove(prev, activeIndex, overIndex);
      });
    }
  };

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

  return (
    <DndContext sensors={[sensor]} onDragEnd={onDragEnd}>
      <SortableContext items={fileList.map((i) => i.uid)} strategy=
{verticalListSortingStrategy}>
        <Upload
          action="https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload"
          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;
```

Crop image before uploading

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

type FileType = Parameters<GetProp<UploadProps, 'beforeUpload'>>[0];

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

  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 as FileType);
        reader.onload = () => resolve(reader.result as string);
      });
    }
    const image = new Image();
    image.src = src;
    const imgWindow = window.open(src);
    imgWindow?.document.write(image.outerHTML);
  };

  return (
    <ImgCrop rotationSlider>
      <Upload
        action="https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload"
        listType="picture-card"
        fileList={fileList}
        onChange={onChange}
        onPreview={onPreview}
      >
        {fileList.length < 5 && '+ Upload'}
      </Upload>
    </ImgCrop>
  );
};

export default App;
```

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://660d2bd96ddfa2943b33731c.mockapi.io/api/upload',
  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;
```

Component Token

Debug

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

const props: UploadProps = {
  action: 'https://660d2bd96ddfa2943b33731c.mockapi.io/api/upload',
  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 error message to show
      url: 'http://www.baidu.com/zzz.png',
    },
  ],
};

const App: React.FC = () => (
  <ConfigProvider
    theme={{
      components: {
        Upload: {
          actionsColor: 'yellow',
        },
      },
    }}
  >
    <Upload {...props}>
      <Button icon={<UploadOutlined />}>Upload</Button>
    </Upload>
  </ConfigProvider>
);

export default App;
```

Debug Disabled Styles

Debug

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

const { Dragger } = Upload;

const fileList: UploadFile[] = [
  {
    uid: '-1',
    name: 'image.png',
    status: 'done',
    url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
  },
  {
    uid: '-2',
    name: 'image.png',
    status: 'done',
    url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
  },
  {
    uid: '-xxx',
    percent: 50,
    name: 'image.png',
    status: 'uploading',
    url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png
  },
  {
    uid: '-5',
    name: 'image.png',
    status: 'error',
  },
];
```

```
const App: React.FC = () => {
  const uploadButton = (
    <button
      style={{ color: 'inherit', cursor: 'inherit', border: 0, background:
'none' }}
      type="button"
    >
      <PlusOutlined />
      <div style={{ marginTop: 8 }}>Upload</div>
    </button>
  );

  return (
    <Space direction="vertical">
      <Upload disabled>Click Text to Upload</Upload>
      <Upload disabled>
        <Button icon={<UploadOutlined />}>Click to Upload</Button>
      </Upload>
      <Upload name="avatar" listType="picture-card" fileList={fileList}
disabled>
        {uploadButton}
      </Upload>
      <Upload name="avatar" listType="picture-circle" fileList={fileList}
disabled>
        {uploadButton}
      </Upload>
      <Dragger disabled>
        <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>
    </Space>
  );
};

export default App;
```

API

Common props ref： Common props

Property	Description	Type	Default
accept	File types that can be accepted. See input accept Attribute	string	-
action	Uploading URL	string | (file) => Promise<string>	-
beforeUpload	Hook function which will be executed before uploading. Uploading will be stopped with `false` or a rejected Promise returned. When returned value is `Upload.LIST_IGNORE`, the list of files that have been uploaded will ignore it. **Warning： this function is not supported in IE9**	(file, fileList) => boolean | Promise<File> | `Upload.LIST_IGNORE`	-
customRequest	Override for the default xhr behavior allowing for additional customization and the ability to implement your own XMLHttpRequest	function	-
data	Uploading extra params or function which can return uploading extra params	object | (file) => object | Promise<object>	-
defaultFileList	Default list of files that have been uploaded	object[]	-
directory	Support upload whole directory (caniuse)	boolean	false

disabled	Disable upload button	boolean	false
fileList	List of files that have been uploaded (controlled). Here is a common issue [#2423](#) when using it	[UploadFile](#)[]	-
headers	Set request headers, valid above IE10	object	-
iconRender	Custom show icon	(file: UploadFile, listType?: UploadListType) => ReactNode	-
isImageUrl	Customize if render <img /> in thumbnail	(file: UploadFile) => boolean	(inside implementation)
itemRender	Custom item of uploadList	(originNode: ReactElement, file: UploadFile, fileList: object[], actions: { download: function, preview: function, remove: function }) => React.ReactNode	-
listType	Built-in stylesheets, support for four types: `text`, `picture`, `picture-card` or `picture-circle`	string	`text`
maxCount	Limit the number of uploaded files. Will replace current one when `maxCount` is 1	number	-

method	The http method of upload request	string	`post`
multiple	Whether to support selected multiple files. `IE10+` 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	`file`
openFileDialogOnClick	Click open file dialog	boolean	true
previewFile	Customize preview file logic	(file: File | Blob) => Promise<dataURL: string>	-
progress	Custom progress bar	ProgressProps (support `type="line"` only)	{ strokeWidth: 2, showInfo: false }
showUploadList	Whether to show default upload list, could be an object to specify `extra`, `showPreviewIcon`, `showRemoveIcon`, `showDownloadIcon`, `removeIcon` and `downloadIcon` individually	boolean | { extra?: ReactNode | (file: UploadFile) => ReactNode, showPreviewIcon?: boolean | (file: UploadFile) => boolean, showDownloadIcon?: boolean | (file: UploadFile) => boolean, showRemoveIcon?: boolean | (file: UploadFile) => boolean, previewIcon?: ReactNode | (file: UploadFile) => ReactNode, removeIcon?: ReactNode | (file: UploadFile) => ReactNode,	true

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

UploadFile

Extends File with additional props.

Property	Description	Type	Default	Version

crossOrigin	CORS settings attributes	`'anonymous'` | `'use-credentials'` | `''`	-	4.20.0
name	File name	string	-	-
percent	Upload progress percent	number	-	-
status	Upload status. Show different style when configured	`error` | `done` | `uploading` | `removed`	-	-
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 recommended, to prevent interference with internally generated id
      name: 'xx.png',   // file name
      status: 'done' | 'uploading' | 'error' | 'removed', // Intercepted file by beforeUpload doesn't have a status field.
      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 the server, including uploading progress, supported by advanced browsers.

Design Token

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 the content of the 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?

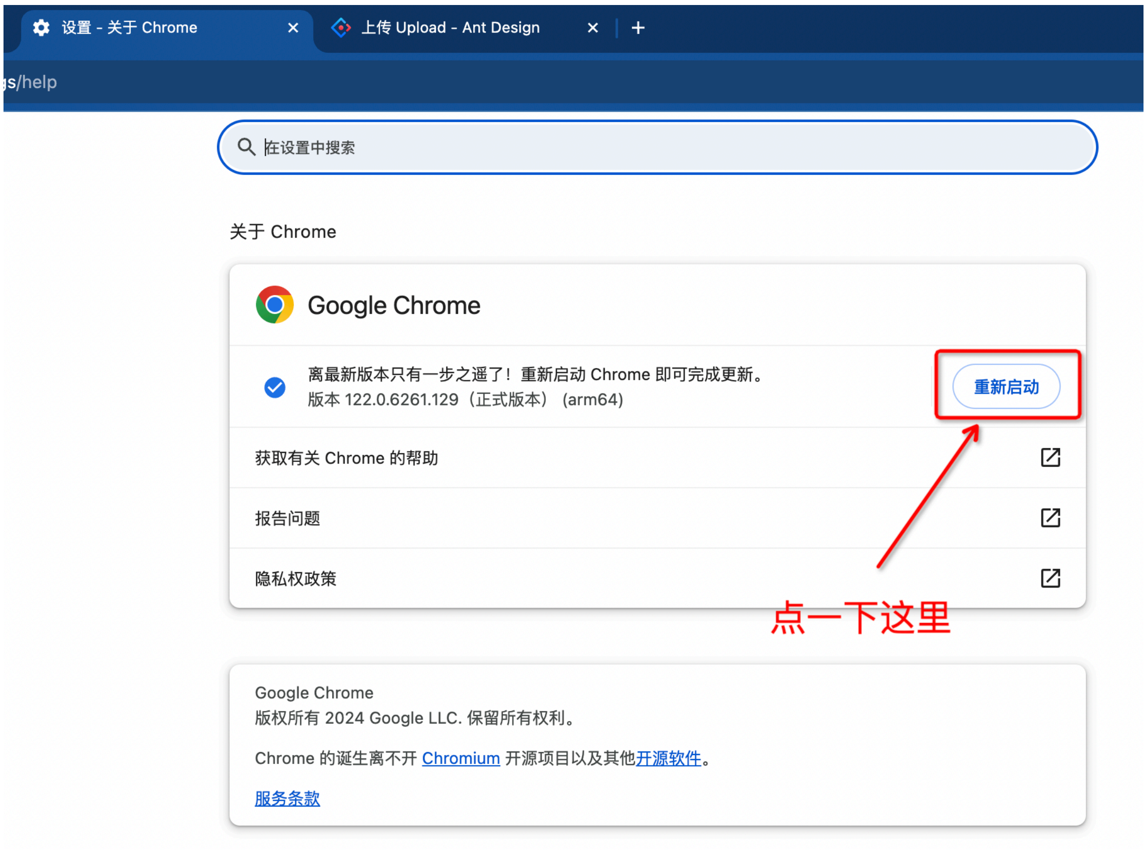
`onChange` will only trigger when the file is in the list, it will ignore any events removed from the list. Please note that there does exist a bug which makes an event still trigger even when the file is not in the list before `4.13.0` .

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 the next major version. Current version is compatible to get origin file by `info.file.originFileObj` . You can change this before a major release.

Why sometimes Chrome can not upload?

Chrome update will also break native upload. Please restart Chrome to finish the upload job.


设置 - 关于 Chrome

上传 Upload - Ant Design

gs/help

在设置中搜索

关于 Chrome

Google Chrome

离最新版本只有一步之遥了！重新启动 Chrome 即可完成更新。
版本 122.0.6261.129（正式版本） (arm64)

Google Chrome
版权所有 2024 Google LLC. 保留所有权利。

Chrome 的诞生离不开 Chromium 开源项目以及其他开源软件。

服务条款

Ref:

- #48007
- #32672
- #32913
- #33988

Can still select files when uploading a folder in Safari?

Inside the upload component, we use the `directory` and `webkitdirectory` properties to control only directories can be selected. However, in Safari's implementation it doesn't seem to work. See here. Please try passing an additional `accept` attribute that cannot match any files. For example:

```
accept: `.${'n'.repeat(100)}`;
```