zh-CN

一个带有远程搜索，防抖控制，请求时序控制，加载状态的多选示例。

en-US

A complete multiple select sample with remote search, debounce fetch, ajax callback order flow, and loading state.

```
import { Select, Spin } from 'antd';
import { SelectProps } from 'antd/es/select';
import debounce from 'lodash/debounce';

export interface DebounceSelectProps<ValueType = any>
  extends Omit<SelectProps<ValueType>, 'options' | 'children'> {
  fetchOptions: (search: string) => Promise<ValueType[]>;
  debounceTimeout?: number;
}

function DebounceSelect<
  ValueType extends { key?: string; label: React.ReactNode; value: string | number }
= any,
>({ fetchOptions, debounceTimeout = 800, ...props }: DebounceSelectProps) {
  const [fetching, setFetching] = React.useState(false);
  const [options, setOptions] = React.useState<ValueType[]>([]);
  const fetchRef = React.useRef(0);

  const debounceFetcher = React.useMemo(() => {
    const loadOptions = (value: string) => {
      fetchRef.current += 1;
      const fetchId = fetchRef.current;
      setOptions([]);
      setFetching(true);

      fetchOptions(value).then(newOptions => {
        if (fetchId !== fetchRef.current) {
          // for fetch callback order
          return;
        }

        setOptions(newOptions);
        setFetching(false);
      });
    };

    return debounce(loadOptions, debounceTimeout);
  }, [fetchOptions, debounceTimeout]);

  return (
    <Select<ValueType>
      labelInValue
      filterOption={false}
```

```
      onSearch={debounceFetcher}
      notFoundContent={fetching ? <Spin size="small" /> : null}
      {...props}
      options={options}
    />
  );
}

// Usage of DebounceSelect
interface UserValue {
  label: string;
  value: string;
}

async function fetchUserList(username: string): Promise<UserValue[]> {
  console.log('fetching user', username);

  return fetch('https://randomuser.me/api/?results=5')
    .then(response => response.json())
    .then(body =>
      body.results.map(
        (user: { name: { first: string; last: string }; login: { username: string }
}) => ({
          label: `${user.name.first} ${user.name.last}`,
          value: user.login.username,
        }),
      ),
    );
}

const Demo = () => {
  const [value, setValue] = React.useState<UserValue[]>([]);

  return (
    <DebounceSelect
      mode="multiple"
      value={value}
      placeholder="Select users"
      fetchOptions={fetchUserList}
      onChange={newValue => {
        setValue(newValue);
      }}
      style={{ width: '100%' }}
    />
  );
};

export default () => <Demo />;
```