zh-CN

用数组生成一组标签，可以动态添加和删除。

en-US

Generating a set of Tags by array, you can add and remove dynamically.

```
import { Tag, Input, Tooltip } from 'antd';
import { PlusOutlined } from '@ant-design/icons';

class EditableTagGroup extends React.Component {
  state = {
    tags: ['Unremovable', 'Tag 2', 'Tag 3'],
    inputVisible: false,
    inputValue: '',
    editInputIndex: -1,
    editInputValue: '',
  };

  handleClose = removedTag => {
    const tags = this.state.tags.filter(tag => tag !== removedTag);
    console.log(tags);
    this.setState({ tags });
  };

  showInput = () => {
    this.setState({ inputVisible: true }, () => this.input.focus());
  };

  handleInputChange = e => {
    this.setState({ inputValue: e.target.value });
  };

  handleInputConfirm = () => {
    const { inputValue } = this.state;
    let { tags } = this.state;
    if (inputValue && tags.indexOf(inputValue) === -1) {
      tags = [...tags, inputValue];
    }
    console.log(tags);
    this.setState({
      tags,
      inputVisible: false,
      inputValue: '',
    });
  };

  handleEditInputChange = e => {
    this.setState({ editInputValue: e.target.value });
  };
  handleEditInputConfirm = () => {
    this.setState(({ tags, editInputIndex, editInputValue }) => {
      const newTags = [...tags];
      newTags[editInputIndex] = editInputValue;

      return {
        tags: newTags,
        editInputIndex: -1,
        editInputValue: '',
      };
    });
  };

  saveInputRef = input => {
    this.input = input;
  };

  saveEditInputRef = input => {
    this.editInput = input;
  };

  render() {
    const { tags, inputVisible, inputValue, editInputIndex, editInputValue } =
this.state;
    return (
      <>
        {tags.map((tag, index) => {
          if (editInputIndex === index) {
            return (
              <Input
                ref={this.saveEditInputRef}
                key={tag}
                size="small"
                className="tag-input"
                value={editInputValue}
                onChange={this.handleEditInputChange}
                onBlur={this.handleEditInputConfirm}
                onPressEnter={this.handleEditInputConfirm}
              />
            );
          }

          const isLongTag = tag.length > 20;

          const tagElem = (
            <Tag
              className="edit-tag"
              key={tag}
              closable={index !== 0}
              onClose={() => this.handleClose(tag)}
            >
```

```
                <span
                  onDoubleClick={e => {
                    if (index !== 0) {
                      this.setState({ editInputIndex: index, editInputValue: tag }, ()
=> {
                        this.editInput.focus();
                      });
                      e.preventDefault();
                    }
                  }}
                >
                  {isLongTag ? `${tag.slice(0, 20)}...` : tag}
                </span>
              </Tag>
            );
            return isLongTag ? (
              <Tooltip title={tag} key={tag}>
                {tagElem}
              </Tooltip>
            ) : (
              tagElem
            );
          })}
          {inputVisible && (
            <Input
              ref={this.saveInputRef}
              type="text"
              size="small"
              className="tag-input"
              value={inputValue}
              onChange={this.handleInputChange}
              onBlur={this.handleInputConfirm}
              onPressEnter={this.handleInputConfirm}
            />
          )}
          {!inputVisible && (
            <Tag className="site-tag-plus" onClick={this.showInput}>
              <PlusOutlined /> New Tag
            </Tag>
          )}
        </>
      );
    }
  }

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

```
.site-tag-plus {
  background: #fff;
  border-style: dashed;
}
```

```css
.edit-tag {
  user-select: none;
}
.tag-input {
  width: 78px;
  margin-right: 8px;
  vertical-align: top;
}
```