TypeScript 的 React 支持

JSX 语法

JSX 是 React 库引入的一种语法，可以在 JavaScript 脚本中直接书写 HTML 风格的标签。

TypeScript 支持 JSX 语法，但是必须将脚本后缀名改成 `.tsx` 。

`.tsx` 文件中，类型断言一律使用 `as` 形式，因为尖括号的写法会与 JSX 冲突。

typescript

```
// 使用
var x = foo as any;

// 不使用
var x = <any>foo;
```

上面示例中，变量 `foo` 被断言为类型 `any` ，在 `.tsx` 文件中只能使用第一种写法，不使用第二种写法。

React 库

TypeScript 使用 React 库必须引入 React 的类型定义。

typescript

```
/// <reference path="react.d.ts" />
interface Props {
  name: string;
}
class MyComponent extends React.Component<Props, {}> {
  render() {
    return <span>{this.props.name}</span>;
```

```
  }
}
<MyComponent name="bar" />; // OK
<MyComponent name={0} />; // error, `name` is not a number
```

内置元素

内置元素使用 `JSX.IntrinsicElements` 接口。默认情况下，内置元素不进行类型检查。但是，如果给出了接口定义，就会进行类型检查。

typescript

```
declare namespace JSX {
  interface IntrinsicElements {
    foo: any;
  }
}
<foo />; // ok
<bar />; // error
```

上面示例中， `<bar />` 不符合接口定义，所以报错。

一种解决办法就是，在接口中定义一个通用元素。

typescript

```
declare namespace JSX {
  interface IntrinsicElements {
    [elemName: string]: any;
  }
}
```

上面示例中， 元素名可以是任意字符串。

组件的写法

typescript

```
interface FooProp {
  name: string;
  X: number;
```

```
  Y: number;
}
declare function AnotherComponent(prop: { name: string });
function ComponentFoo(prop: FooProp) {
  return <AnotherComponent name={prop.name} />;
}
const Button = (prop: { value: string }, context: { color: string }) => (
  <button />
);
```


限时抢

推荐机场 → 25元/月，500G 购买。

最后更新: 2023/8/13 15:25

Previous page
tsc 命令