Как помогает TypeScript

Общее

- TS - ~~язык~~ надмножество JS, система типов и компилятор
- Open-source, разрабатывается Microsoft
- TypeScript, VS Code, Slack, Angular2+, Vue3, Ant.design ...
- Отдельная типизация - @types и .d.ts
- Некоторые редакторы используют TS тайпинги для IntelliSense

Интерфейс (тип) и рефакторинг


```
function greet(user) {
  console.log(user.name)
}

// Структура user изменилась (name -> firstName)

function greet(user) {
  console.log(user.name) // упадет в рантайме
}
```

Интерфейс (тип) и рефакторинг


```
interface User {
  id: string
  name: string
}

function greet(user: User) {
  console.log(user.name)
}
```

```
interface User {
  id: string
  firstName: string
}

function greet(user: User) {
  console.log(user.firstName)
}

function bye(user: User) {
```

```
  user.| // автокомплит!
  console.log(user.fullName) // TS предупреждает
}

function greet('wrong param') // ошибка
```

То же самое в реакт компонентах


```
type Props = {
  onChange: (value: string) => void
  value?: string
}

function Input(props: Props) {
  return <StyledInput {...props} />
}
```

```
<Input value="" onChange={val => { console.log(val) }} />
<Input value={1} /> // не хвататет onChange, value не того типа
```

Class-based компоненты


```
type Props = {
  name: string
}
type State = {
  value: string
```

```
}
class FormInput extends Component<Props, State> { // Дженерики!
  onChange = value => { this.setState({ value })}
  render() {
    const {name} = this.props // типизирован
    const {value} = this.state // типизирован
    return <>
      <Input value={value} onChange={onChange} />
      {value && value.split('').reverse().join('')}
      {value.reverse()} // TS2339: Property 'reverse' does not exist on type 'string'
    </>
  }
}
```

Дженерики


```
interface User {
  data: {
    id: string
```

```
interface Entity<T> {
  data: T
  permissions: string[]
```

```
    name: string
  },
  permissions: string[]
}
```

```
}

type UserData = {
  id: string
  name: string
}
type User = Entity<UserData>
```

Union of strings

```
type Props = {
    onChange: (value) => void
    value: string
    size?: 'small' | 'large'
}

function Input(props: Props) {
    return <StyledInput {...props} />
}

<Input size="smoll" /> // Ошибка
```

Переиспользование типов


```
type Props = {
    size?: 'small' | 'large' // нарушаем принцип DRY
}

class FormInput extends Component<Props, State> {}
```

Переиспользование типов


```
// Input.tsx
export type InputSize = 'small' | 'large'

type Props = {
    size?: InputSize
    ...
}

// FormInput.tsx
import { Input, InputSize } from './Input'

type Props = {
    size?: InputSize // уже лучше
}
```

Type intersection


```
// Input.tsx
export type InputProps = {
    size?: InputSize
}
type Props = InputProps & {
  value: string
  onChange: (value: string) => void
}

// FormInput.tsx
import { Input, InputProps } from './Input'
```

```
class FormInput extends Component<InputProps, State> {}
```

Наследование интерфейсов


```
// Input.tsx
export interface InputProps {
    size?: InputSize
}
interface Props extends InputProps {
  value: string
  onChange: (value: string) => void
}

// FormInput.tsx
```

```tsx
import { Input, InputProps } from './Input'

class FormInput extends Component<InputProps, State> {}
```

Условные типы


```tsx
// Input.tsx
export type InputProps = {
  value: string
  onChange: (value: string) => void
    size?: InputSize
}

// FormInput.tsx
```

```tsx
import { Input, InputProps } from './Input'

type Props = Omit<InputProps, 'value' | 'onChange'> // мой выбор
```

Массивы


```tsx
type User = {
  id: string
  name: string
}
```

```
function getUsers(): Users[] {...}

const users = getUsers()

users.forEach(el => el.|) // TS знает, что el имеет тип User
```

Index signature


```
{
  '1': {
    id: '1',
```

```
type User = {
  id: string
  name: string
```

```
    name: 'Foo'
  },
  '2': {
    id: '2',
    name: 'Bar'
  }
}
```

```
}

const users: { [key: string]: User } = {
  foo: { // не хватает поля name
    id: 'foo'
  },
  bar: {
    id: 'bar',
    unknown: '' // такого поля нет в User
  },
}
```

Mapped types и keyof typeof

```ts
export type Status = 'guest' | 'admin' | 'editor'
export const statuses: { [K in Status]: string } = {
  guest: 'Гость',
  admin: 'Админ',
  editor: 'Редактор'
}

// или так
export const statuses = {
  guest: 'Гость',
  admin: 'Админ',
  editor: 'Редактор'
}
export type Status = keyof typeof statuses
```

Значения полей объекта в union


```ts
const roles = {
  User: 'ROLE_USER',
  Admin: 'ROLE_ADMIN',
} as const

type Roles = typeof roles
// {
//   'User': 'ROLE_USER',
//   'Admin': 'ROLE_ADMIN'
// }

type RoleValues = Roles[keyof Roles]
// 'ROLE_USER | ROLE_ADMIN'
```

Иконки


```
Icon
|--icons
|   |--check.svg
|   |--close.svg
|   |--index.ts
|--Icon.tsx
```

```ts
// icons/index.ts
export { default as check } from './check.svg';
export { default as close } from './close.svg';

// Icon.tsx
import * as icons from './icons';
type Props = {
  icon: keyof typeof icons
}
export function Icon(props: Props) {
  const CurrentIcon = icons[props.icon];
  return <CurrentIcon />
}
```

```
// somewhere
<Icon icon="chek"/> // автокомплит и тайпчек по названию иконок
```

Рекомендации

- Редактор с хорошей поддержкой TS (VS Code, WebStorm)
- Strict Mode
- Избегайте `any` и `// @ts-ignore`, стремитесь к 100% покрытию типами
- Typecheck в гит хуках
- Попробуйте, если не пробовали =)

Ссылки

[TypeScript Deep Dive]

[Typescript Evolution]

Презентация

Github