codecademy

Tipos de objetos avançados

Tipo de interface TypeScript

O TypeScript permite que você digite especificamente um objeto usando uma interface que pode ser reutilizada por vários objetos. Para criar uma interface, use a palavra- `interface` chave seguida do nome da interface e do objeto digitado.

```
interface Publication {
  isbn: string;
  author: string;
  publisher: string;
}
```

Interface TypeScript Definir somente objetos

No TypeScript, os aliases de tipo podem definir tipos compostos, como objetos e uniões, bem como tipos primitivos, como números e strings; interface, no entanto, só pode definir objetos. Interface é útil na digitação de objetos escritos para programas orientados a objetos.

```
// Type alias can define a union type
type ISBN = number | string;

// Type alias can define an object type
type PublicationT = {
  isbn: ISBN;
  author: string;
  publisher: string;
}

// Interface can only define an object type
interface PublicationI {
  isbn: ISBN;
  author: string;
  publisher: string;
}
```

Interface TypeScript para Classes

Para aplicar uma interface TypeScript a uma classe, adicione a palavra-chave `implements` após o nome da classe seguida pelo nome da interface. O TypeScript verificará e garantirá que o objeto realmente implemente todas as propriedades e métodos definidos dentro da interface.

```
interface Shape {
  area: number;
  computeArea: () => number;
}

const PI: number = 22/7 ;

// Circle class implements the Shape interface
class Circle implements Shape {
  radius: number;
  area: number;
  constructor(radius: number) {
    this.radius = radius;
    this.area = this.computeArea();
  }
  computeArea = (): number => {
    return PI * this.radius * this.radius;
  }
}

let target = new Circle(3);
console.log(target.area.toFixed(2));  // Prints "28.29"
```

Interface aninhada TypeScript

O TypeScript permite que os aliases de tipo e a interface sejam aninhados. Um objeto tipado com uma interface aninhada deve ter todas as suas propriedades estruturadas da mesma forma que a definição da interface.

```
// This is a nested interface
interface Course {
  description: {
    name: string;
    instructor: {
      name: string;
    }
    prerequisites: {
      courses: string[];
    }
  }
}

class myCourse implements Course {
  description = {
    name: '',
    instructor: {
      name: ''
    },
    prerequisites: {
      courses: []
    }
  }
}
```

Interfaces de aninhamento de TypeScript dentro de uma interface

Como as interfaces podem ser compostas, o TypeScript permite aninhar interfaces dentro de uma interface.

```
// Date is composed of primitive types
interface Date {
  month: number;
  day: number;
  year: number
}

// Passport is composed of primitive types and nested with another interface
interface Passport {
  id: string;
  name: string;
  citizenship: string;
  expiration: Date;
}

// Ticket is composed of primitive types and nested with another interface
interface Ticket {
  seat: string;
  expiration: Date;
}

// TravelDocument is nested with two other interfaces
interface TravelDocument {
  passport: Passport;
  ticket: Ticket;
}
```

Herança da interface TypeScript

Assim como as classes JavaScript, uma interface pode herdar propriedades e métodos de outra interface usando a palavra- `extends` chave. Os membros da interface herdada são acessíveis na nova interface.

```
interface Brand {
  brand: string;
}

// Model inherits property from Brand
interface Model extends Brand {
  model: string;
}

// Car has a Model interface
class Car implements Model {
  brand;
  model;
  constructor(brand: string, model: string) {
    this.brand = brand;
    this.model = model;
  }
  log() {
    console.log(`Drive a ${this.brand} ${this.model} today!`);
  }
}

const myCar: Car = new Car('Nissan', 'Sentra');
myCar.log(); // Prints "Drive a Nissan Sentra today!"
```

Assinatura do índice da interface TypeScript

Os nomes de propriedade de um objeto são considerados strings, mas também podem ser números. Se você não souber antecipadamente os tipos desses nomes de propriedade, o TypeScript permite usar uma assinatura de índice para especificar o tipo do nome da propriedade dentro de um objeto. Para especificar uma assinatura de índice, use colchetes, `[...]`, para cercar a notação de tipo do nome da propriedade.

```
interface Code {
  [code: number]: string;
}
const codeToStates: Code = {603: 'NH', 617: 'MA'};

interface ReverseCode {
  [code: string]: number;
}
const stateToCodes: ReverseCode = {'NH': 603, 'MA': 617};
```

Propriedades opcionais da interface TypeScript

O TypeScript permite especificar propriedades opcionais dentro de uma interface. Isso é útil em situações em que nem todas as propriedades do objeto têm valores atribuídos a elas. Para indicar se uma propriedade é opcional, anexe um `?` símbolo após o nome da propriedade antes dos dois pontos, `:`.

```
interface Profile {
  name: string;
  age: number;
  hobbies?: string[];
}

// The property, hobbies, is optional, but name and age are
required.
const teacher: Profile = {name: 'Tom Sawyer', age: 18};
```