Going Further with Basic Types

Brice Wilson

@brice_wilson www.BriceWilson.net

Overview

Destructuring assignments

Spread operator

Tuple types

Union types

Intersection types

Mixins

String literal types

Type aliases

Destructuring Arrays and Objects

destructuring assignment

The process of assigning the elements of an array or the properties of an object to individual variables.

Destructuring Arrays

```
let medals: string[] = ['gold', 'silver', 'bronze'];

let first: string = medals[0];
let second: string = medals[1];
let third: string = medals[2];

let [first, second, third] = medals;
```

Destructuring Objects

```
let person = {
  name: 'Audrey',
  address: '123 Main Street',
  phone: '555-1212'
};

let name: string = person.name;
let address: string = person.address;
let phone: string = person.phone;

let { name, address, phone } = person;
```

Demo

Destructuring arrays and objects

The Spread Operator

Spread Operator

```
let newBookIDs = [10, 20];

let allBookIDs = [1, 2, 3, ...newBookIDs];
```

Spread Operator

```
let newBookIDs = [10, 20];

let allBookIDs = [1, 2, 3, ...newBookIDs];

// [ 1, 2, 3, 10, 20 ]
```

Demo

Using the spread operator

Tuple Types

tuple type

A tuple type combines a set of numerically named properties with the members of an array type.

TypeScript Language Specification – Section 3.3.3

Version 1.8 - January 2016

```
let myTuple: [number, string] = [10, 'Macbeth'];
```

Tuple Types

Extension to arrays

Type of a fixed number of elements is specified

May contain different types

```
let myTuple: [number, string] = [10, 'Macbeth'];
```

Tuple Types

Extension to arrays

Type of a fixed number of elements is specified

May contain different types

```
let myTuple: [number, string] = [10, 'Macbeth'];
```

Tuple Types

Extension to arrays

Type of a fixed number of elements is specified

May contain different types

```
let myTuple: [number, string] = [10, 'Macbeth'];
myTuple[0] = 'Hamlet'; // ERROR
myTuple[1] = 20; // ERROR
myTuple[2] = 'Hamlet';
myTuple[2] = 20;
```

Tuple Types

Extension to arrays

Type of a fixed number of elements is specified

May contain different types

Demo

Creating and using tuple types

Combining Types

```
function PrintIdentifier(id: string | number) {
    // print things here
}
```

Union Types

Specify several valid types for a value

Vertical bar is used to separate valid types

```
function PrintIdentifier(id: string | number) {
    // print things here
}
```

Union Types

Specify several valid types for a value

Vertical bar is used to separate valid types

```
function CreateCoolNewDevice(): Phone & Tablet {
    // phablet is born
}
```

Intersection Types

Specify a value that will contain all members of several types

Ampersand is used to separate included types

```
function CreateCoolNewDevice(): Phone & Tablet {
    // phablet is born
}
```

Intersection Types

Specify a value that will contain all members of several types

Ampersand is used to separate included types

Demo

Using union and intersection types

Demo

Creating a mixin

String Literal Types and Type Aliases

String Literal Types

```
let empCategory: 'Manager';
```

String Literal Types

```
let empCategory: 'Manager';

let empCategory: 'Manager' = 'Manager';

let empCategory: 'Manager' = 'Non-Manager'; // ERROR

let empCategory: 'Manager' | 'Non-Manager' = 'Manager';
```

String Literal Types

```
let empCategory: 'Manager';

let empCategory: 'Manager' = 'Manager';

let empCategory: 'Manager' = 'Non-Manager'; // ERROR

let empCategory: 'Manager' | 'Non-Manager' = 'Manager';

let empCategory: 'Manager' | 'Non-Manager' = 'Non-Manager';
```

Type Aliases

```
let empCategory: 'Manager' | 'Non-Manager' = 'Manager';

type EmployeeCategory = 'Manager' | 'Non-Manager';
```

Type Aliases

```
let empCategory: 'Manager' | 'Non-Manager' = 'Manager';

type EmployeeCategory = 'Manager' | 'Non-Manager';

let empCategory: EmployeeCategory = 'Manager';
```

Type Aliases

```
let empCategory: 'Manager' | 'Non-Manager' = 'Manager';

type EmployeeCategory = 'Manager' | 'Non-Manager';

let empCategory: EmployeeCategory = 'Manager';

let empCategory: EmployeeCategory = 'Non-Manager';

let empCategory: EmployeeCategory = 'Intern'; // ERROR
```

Demo

Using string literal types and type aliases

Summary

Destructuring assignments

Doing more with arrays

Combining types

String literal types

Type aliases