Tipos de União

Tipo de União TypeScript

O TypeScript permite um tipo flexível chamado any que pode ser atribuído a uma variável cujo tipo não é específico. Por outro lado, o TypeScript permite combinar tipos específicos como um tipo de união.

```
let answer: any;     // any type
let typedAnswer: string | number; // union type
```

Sintaxe do tipo de união TypeScript

O TypeScript permite criar um tipo de união que é uma composição de tipos selecionados separados por uma barra vertical, `|`.

```
let myBoolean: string | boolean;

myBoolean = 'TRUE';  // string type
myBoolean = false;   // boolean type
```

Limitação de tipo de união TypeScript

Como uma variável de um tipo de união pode assumir um dos vários tipos diferentes, você pode ajudar o TypeScript a inferir o tipo de variável correto usando a restrição de tipo. Para restringir uma variável a um tipo específico, implemente um protetor de tipo. Use o `typeof` operador com o nome da variável e compare-o com o tipo que você espera para a variável.

```
const choices: [string, string] = ['NO', 'YES'];
const processAnswer = (answer: number | boolean) => {
  if (typeof answer === 'number') {
    console.log(choices[answer]);
  } else if (typeof answer === 'boolean') {
    if (answer) {
      console.log(choices[1]);
    } else {
      console.log(choices[0]);
    }
  }
}
processAnswer(true);   // Prints "YES"
processAnswer(0);      // Prints "NO"
```

Tipo de União de Retorno da Função TypeScript

O TypeScript infere o tipo de retorno de uma função, portanto, se uma função retornar mais de um tipo de dados, o TypeScript inferirá que o tipo de retorno é uma união de todos os tipos de retorno possíveis. Se você deseja atribuir o valor de retorno da função a uma variável, digite a variável como uma união de tipos de retorno esperados.

```
const popStack = (stack: string[]) => {
  if (stack.length) {
    return stack[stack.length-1]; // return type is any
  } else {
    return null;      // return type is null
  }
};
let toys: string[] = ['Doll', 'Ball', 'Marbles'];
let emptyBin: string[] = [];
let item: string | null = popStack(toys); // item has union type
console.log(item);  // Prints "Marbles"
item = popStack(emptyBin);
console.log(item);  // Prints null
```

União TypeScript de tipos de matriz

O TypeScript permite declarar uma união de uma matriz de tipos diferentes. Lembre-se de colocar a união entre parênteses, `(...)` e anexar colchetes, `[]` após o parêntese de fechamento.

```
const removeDashes = (id: string | number) => {
  if (typeof id === 'string') {
    id = id.split('-').join('');
    return parseInt(id);
  } else {
    return id;
  }
}
// This is a union of array types
let ids: (number | string)[] = ['93-235-66', '89-528-92'];
let newIds: (number | string)[] = [];
for (let i=0; i < ids.length; i++) {
  newIds[i] = removeDashes(ids[i]); // Convert string id to
number id
}
console.log(newIds);  // Prints [9323566, 8952892]
```

Acesso à propriedade comum do tipo de união TypeScript

Como resultado do suporte a uma união de vários tipos, o TypeScript permite acessar propriedades comuns entre os tipos de membros sem nenhum erro.

```
let element: string | number[] = 'Codecademy';
// The .length property is common for string and array
console.log(element.length);      // Prints 10
// The .match method only works for a string type
console.log(element.match('my')); // Prints ["my"]

element = [3, 5, 1];
// The length property is common for string and array
console.log(element.length);      // Prints 3
// The .match method will not work for an array type
console.log(element.match(5));  // Error: Property 'match' does
not exist on type 'number[]'.
```

União TypeScript de tipos literais

Você pode declarar um tipo de união que consiste em tipos literais, como literais de string, literais numéricos ou literais booleanos. Isso criará tipos de união mais específicos e com estados distintos.

```
// This is a union of string literal types
type RPS = 'rock' | 'paper' | 'scissors' ;
const play = (choice: RPS): void => {
  console.log('You: ', choice);
  let result: string = '';
  switch (choice) {
    case 'rock':
      result = 'paper';
      break;
    case 'paper':
      result = 'scissors';
      break;
    case 'scissors':
      result = 'rock';
      break;
  }
  console.log('Me: ', result);
}
const number = Math.floor(Math.random()*3);
let choices: [RPS, RPS, RPS] = ['rock', 'paper', 'scissors'];
play(choices[number]);
```