Resumo sobre Generics no TypeScript

Generics no TypeScript são uma ferramenta que permite criar código reutilizável e flexível, aceitando diferentes tipos de dados sem comprometer a segurança de tipos. Com generics, é possível criar funções, classes e interfaces que funcionam com vários tipos, sem precisar especificá-los diretamente.

Exemplo básico de função genérica:

```
function identity<T>(arg: T): T {
    return arg;
}
```

Restrições em Generics:

Generics podem ser restritos a certos tipos usando extends. Isso significa que o tipo genérico pode aceitar apenas tipos que seguem certas regras.

Por exemplo, se quisermos criar uma função que aceite apenas objetos com uma propriedade length:

```
function logLength<T extends { length: number }>(arg: T): void {
    console.log(arg.length);
}
```

Aqui, T só pode ser usado com tipos que possuem uma propriedade length, como arrays ou strings.

Classes Genéricas:

Generics também podem ser aplicados a classes, permitindo a criação de estruturas flexíveis e reutilizáveis.

Exemplo:

```
class GenericNumber<T> {
    zeroValue: T;
    add: (x: T, y: T) => T;
}


let myGenericNumber = new GenericNumber<number>();
myGenericNumber.zeroValue = 0;
```

```
myGenericNumber.add = function (x, y) { return x + y; };
```

Nesse exemplo, a classe GenericNumber é genérica, mas ao instanciá-la, podemos definir o tipo específico, como number.

Benefícios do Generics:

1. Reutilização de código
2. Segurança de tipos
3. Flexibilidade