

```
STATUS  
ACCOUNT  
STATUS  
class Banner  
attr_accessible :horiz, :link, :visible, :image, :position  
has_attached_file :image, styles: { vert: '220'  
before_create :assign_position  
ANALYS  
g"==f.type(a)},F=function(a){return p(a)&&a.indexOf("%")},l=function(a,d){var  
t(a,10)||0;d&&F(a)&&(e=b.getViewport()[d]/100);return Math.ceil(e)},x=function(a,b){return  
"x";f.extend(b,{version:"2.1.4",defaults:{padding:15,margin:20,width:800,  
0,minWidth:100,minHeight:100,maxWidth:9999,maxHeight:9999,autoSize:!0,autoHeight:!1,autoWidth:!1,autoResize  
center:!s,fitToView:!0,aspectRatio:0.5,topRatio:0.5,scrolling:"auto",wrapCSS:"",arrows:!0,close  
oseClick:!1,nextClick:!1,mouseWheel:!0,autoPlay:!1,playSpeed:3E3,preload:3,modal:!1,loop:!0,ajax:{dataType:  
ders:{'X-fancybox':!0}},iframe:{scrolling:"auto",preload:!0},swf:{wmode:"transparent",allowfullscreen:"true  
riptaccess:"always"},keys:{next:{13:"left",  
:"left",40:"up"},prev:{8:"right",33:"down",37:"right",38:"down"},close:[27],play:[32],toggle:[70]},directi  
left},prev:"right"},scrollOutside:!0,index:0,type:null,href:null,content:null,title:null,tpl:{wrap:'div  
ybox-wrap' tabIndex="-1"><div class="fancybox-skin"><div class="fancybox-outer"><div  
ybox-inner"></div></div></div></div>',image:'. Esse T é como um espaço reservado para o tipo de dado que você vai usar mais tarde. Por exemplo:

```
function caixaMagica<T>(item: T): T {
 return item;
}
```

Aqui, caixaMagica é uma função que pode trabalhar com qualquer tipo de dado. T é o tipo que você decidirá quando usar a função.

### Exemplos Práticos:

- Se você usar caixaMagica<number>(5), o T se torna um número e a função funciona com números.
- Se usar caixaMagica<string>('Olá'), agora o T é uma string e a função lida com strings.

**Mais de um tipo:** você também pode fazer uma caixa mágica que lida com mais de um tipo de coisa ao mesmo tempo. Por exemplo:

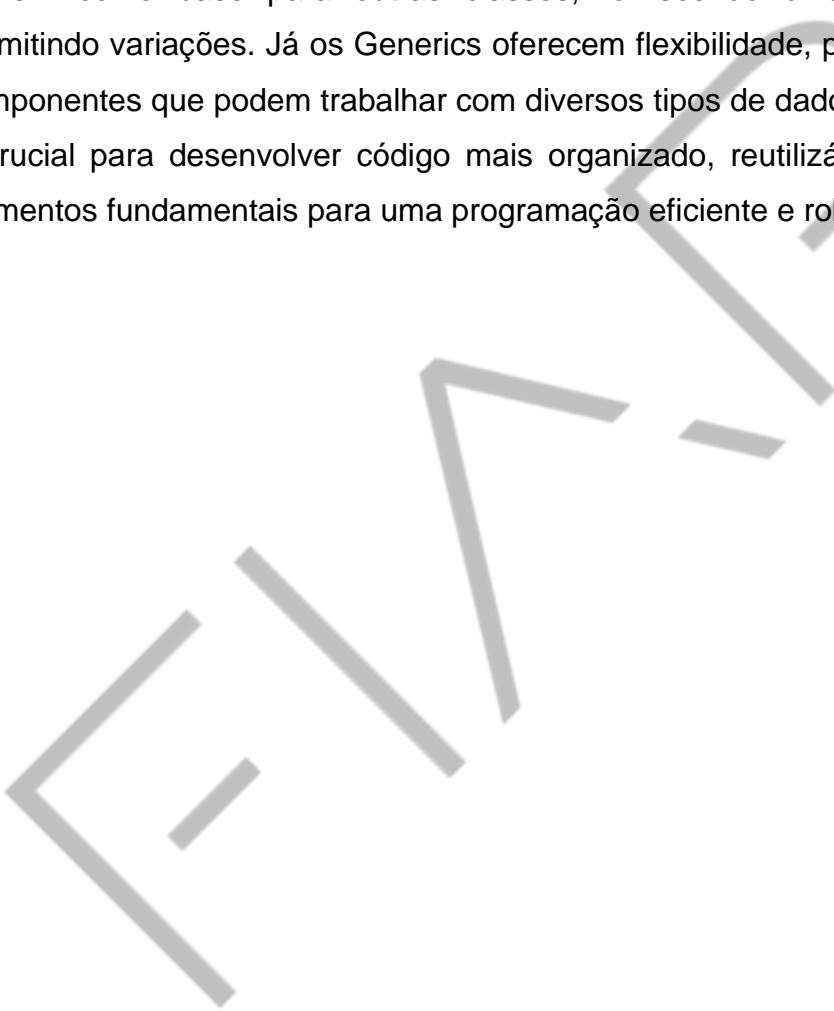
```
function caixaDupla<T, U>(item1: T, item2: U): [T, U] {
 return [item1, item2];
}
```

Aqui, caixaDupla pode guardar dois tipos diferentes de coisas ao mesmo tempo, como um número e uma string.

Em resumo, Generics em TypeScript são como ferramentas flexíveis que você pode ajustar para trabalhar com diferentes tipos de dados. Eles são úteis porque permitem que você escreva código mais geral e reutilizável, assim como uma caixa mágica que pode se ajustar para guardar qualquer coisa.

## O QUE VOCÊ VIU NESTA AULA?

Durante essa aula, focamos em conceitos essenciais da programação em TypeScript: interfaces, classes abstratas e Generics. Compreendemos que interfaces são como contratos para as classes, estabelecendo quais métodos devem ser implementados, mas sem ditar a implementação específica. As classes abstratas servem como base para outras classes, fornecendo uma estrutura comum e permitindo variações. Já os Generics oferecem flexibilidade, permitindo a criação de componentes que podem trabalhar com diversos tipos de dados. Esse conhecimento é crucial para desenvolver código mais organizado, reutilizável e fácil de manter, elementos fundamentais para uma programação eficiente e robusta.



## REFERÊNCIAS

ADRIANO, T. **Guia prático de TypeScript: Melhore suas aplicações JavaScript.** São Paulo: Casa do Código, 2021.

FELIX, R. **Programação Orientada a Objetos.** [s.l.]: Editora Pearson, 2016.

DevMedia. **Orientação a Objetos - simples assim!**. Disponível em: <<https://www.devmedia.com.br/orientacao-a-objetos-simples-assim/3254>>. Acesso em: 24 jan. 2024.



## PALAVRAS-CHAVE

**Palavras-chave:** Typescript. Desenvolvimento Básico. Superset.



```
STATUS
ACCOUNT
STATUS

class Banner
attr_accessible :horiz, :link, :visible, :image, :position
has_attached_file :image, styles: { vert: '220'
before_create :assign_position

ing==="f.type(a)},F=function(a){return p(a)&&0<a.indexOf("%")},l=function(a,d){var
int(a,10)||0;d&&F(a)&&(e=b.getViewport()[d]/100);return Math.ceil(e)},x=function(a,b){return
px"};f.extend(b,{version:"2.1.4",defaults:{padding:15,margin:20,width:800,
00,minWidth:100,minHeight:100,maxWidth:9999,maxHeight:9999,autoSize:!0,autoHeight:!1,autoWidth:!1,autoResize:
Center:!s,fitToView:!0,aspectRatio:!1,topRatio:0.5,leftRatio:0.5,scrolling:"auto",wrapCSS:"",arrows:!0,close
closeClick:!1,nextClick:!1,mouseWheel:!0,autoPlay:!1,playSpeed:3E3,preload:3,modal:!1,loop:!0,ajax:{dataType:
adlers:{'X-fancyBox':!0},iframe:{scrolling:"auto",preload:!0},swf:{wmode:"transparent",allowfullscreen:"true
criptaccess:"always"},keys:{next:{13:"left",
9:"left",40:"up"},prev:{8:"right",33:"down",37:"right",38:"down"},close:[27],play:[32],toggle:[70]},directi
"left",prev:"right"},scrollOutside:!0,index:0,type:null,href:null,content:null,title:null,tpl:{wrap:'<div
cybox-wrap" tabIndex="-1"><div class="fancybox-skin"><div class="fancybox-outer"><div
cybox-inner"></div></div></div>',image:'<img class="fancybox-image" src="{href}" alt=""
box-frame{rnd}" class="fancybox-iframe" frameborder="0"
are malicius screen'}

px;
x;
ME
px;
underline;
TIME
POS TECH
class Banner < ActiveRecord::Base
attr_accessible :horiz, :link, :visible, :image,
has_attached_file :image, styles: { vert: '220'
before_create :assign_position
protected
def assign_position
max = Banner.maximum(:position)
self.position = max ? max + 1 : 0
```