ARGUS ACADEMY

TypeScript

ARGUS ACADEMY

www.argus-academy.com

Sobre a Argus Academy

A Argus Academy é uma plataforma completa de ensino digital que oferece cursos de alta qualidade em diversas áreas de TI, com ênfase especial em programação.

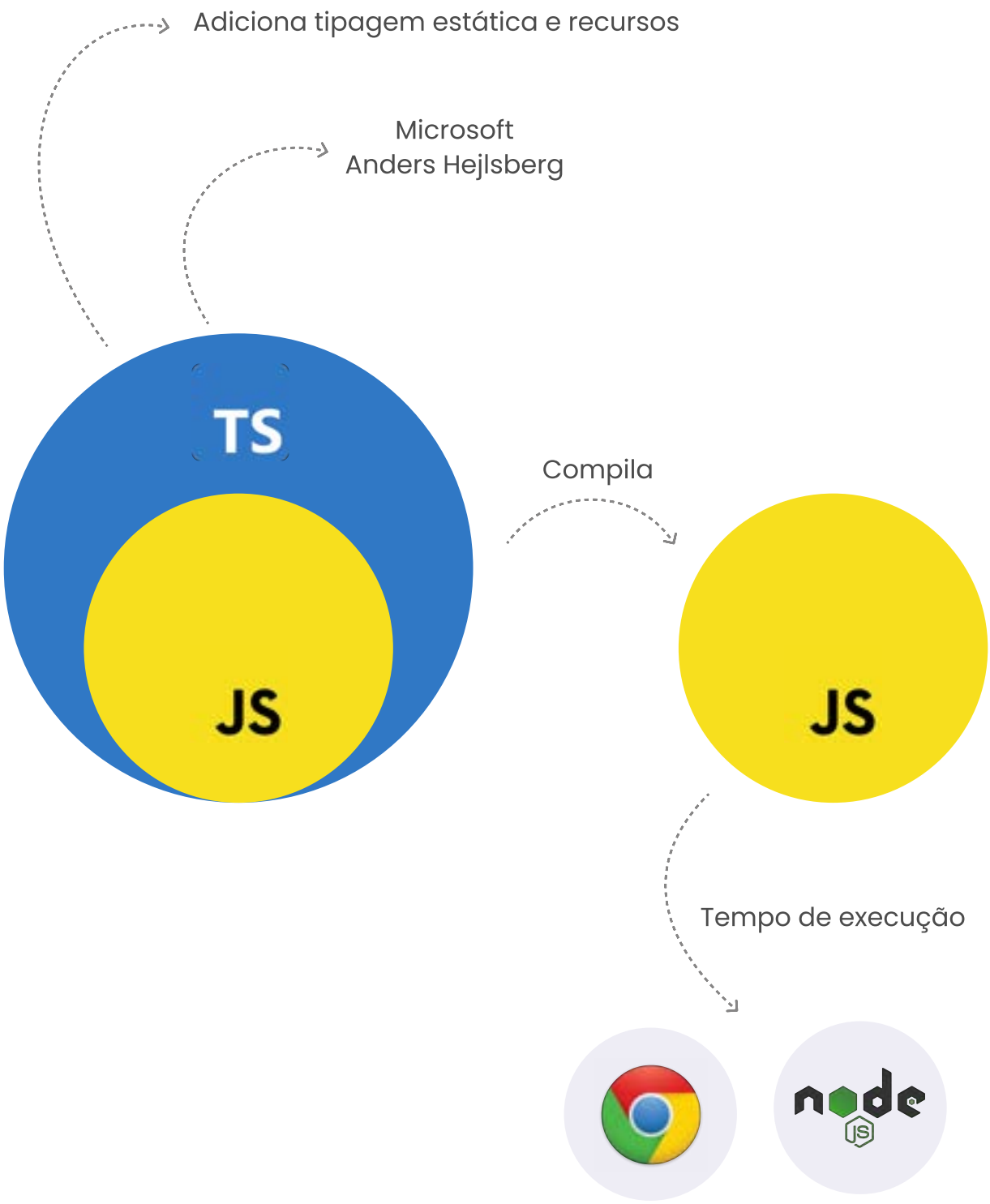
Nossa missão é democratizar o acesso à educação de qualidade, capacitando você para alcançar seus objetivos profissionais e pessoais.

Com que você irá aprender

Jorge Sant' Ana é programador, instrutor, entusiasta de tecnologia e fundador da Argus Academy e Argus Technology.

Formado em Tecnologia para Gestão de Negócios pela Fatec e Pós Graduado em Gestão de Projetos de TI pela Fundação Vanzolini.

O que é TypeScript?



Introdução ao NodeJS, NPM e VS Code

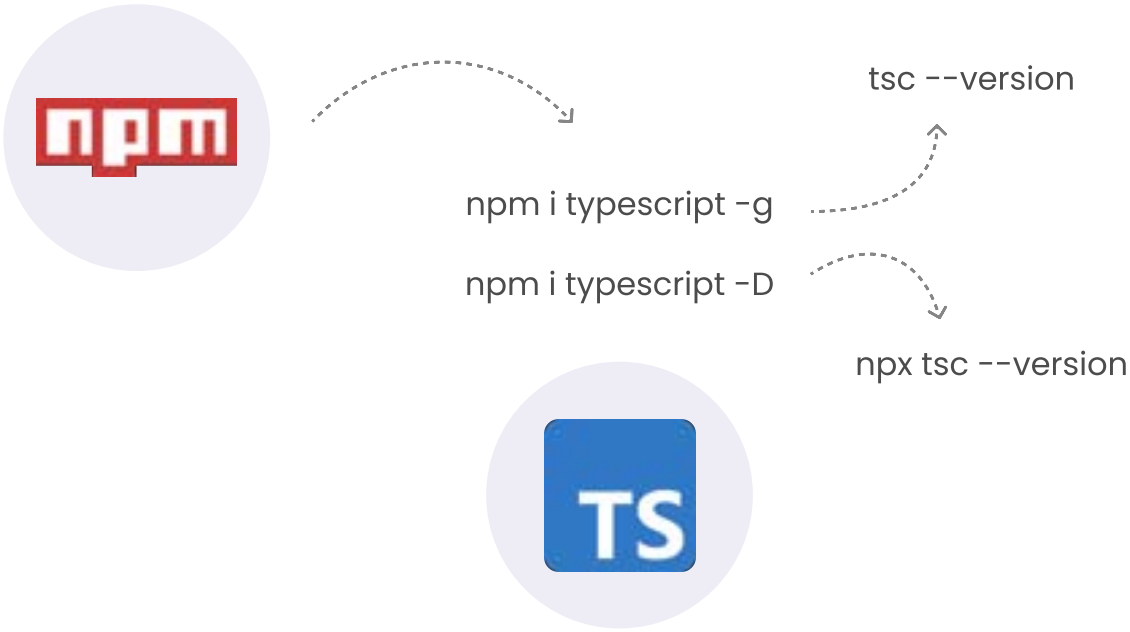
Plataforma de execução de código JavaScript

Gerenciador de pacotes do NodeJS

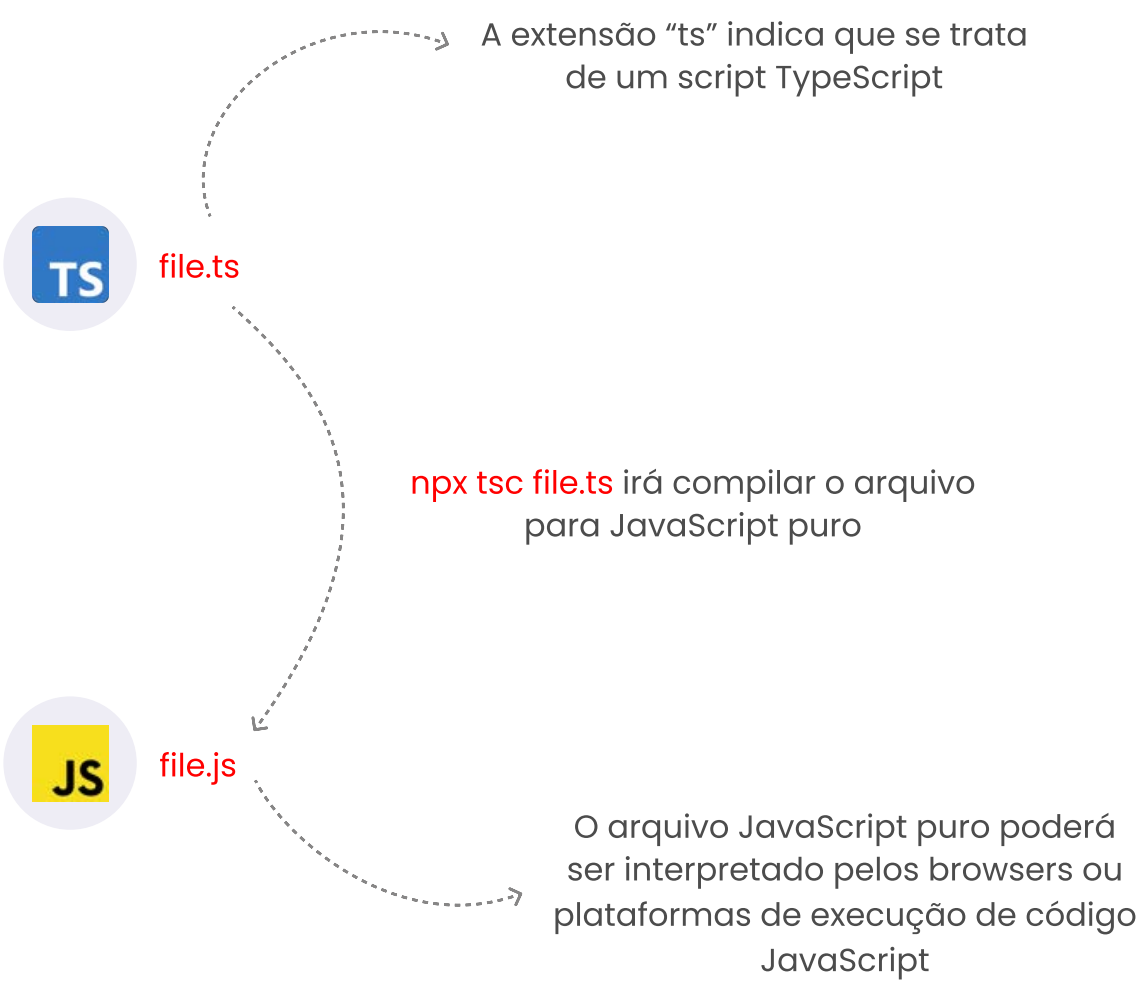
Editor de código fonte

Iniciando o projeto com o NPM

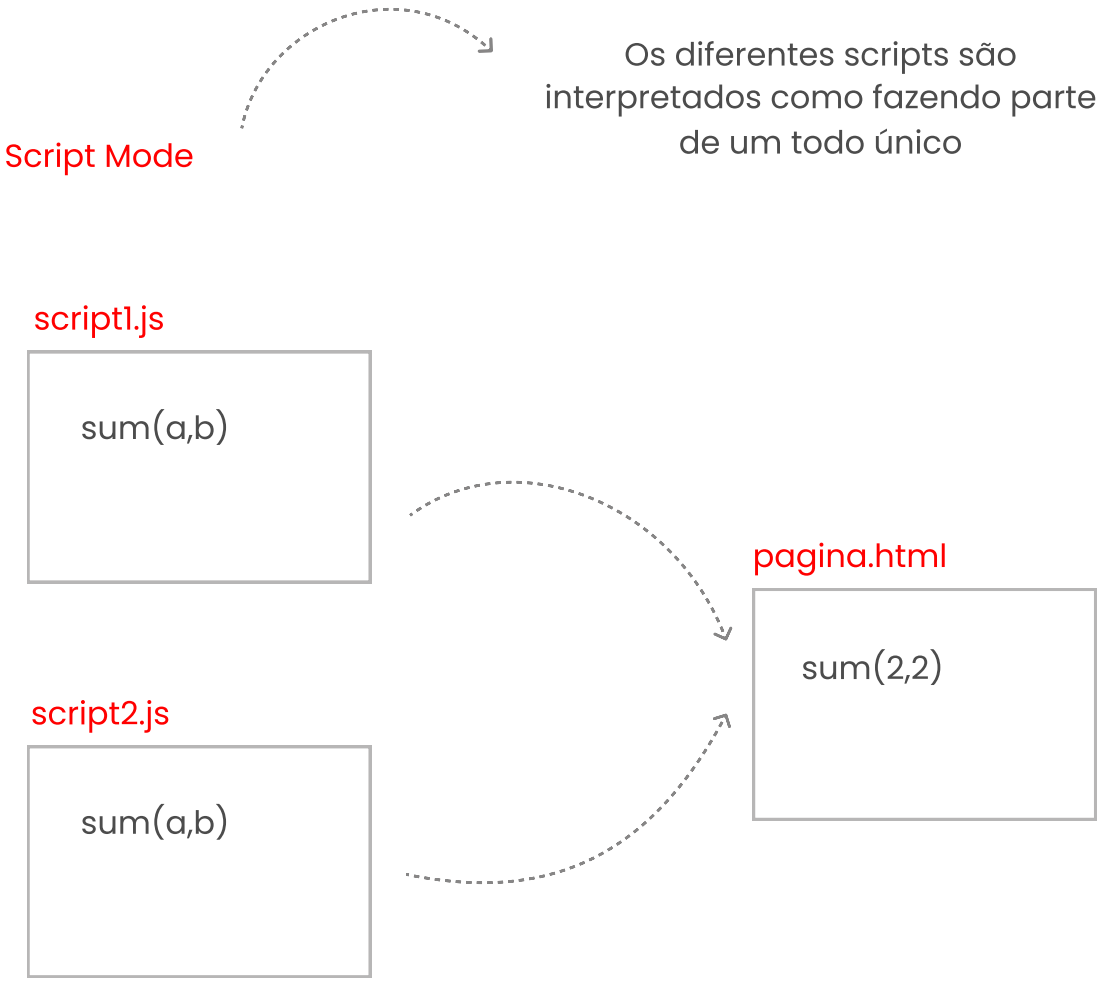
Instalando o TypeScript no projeto



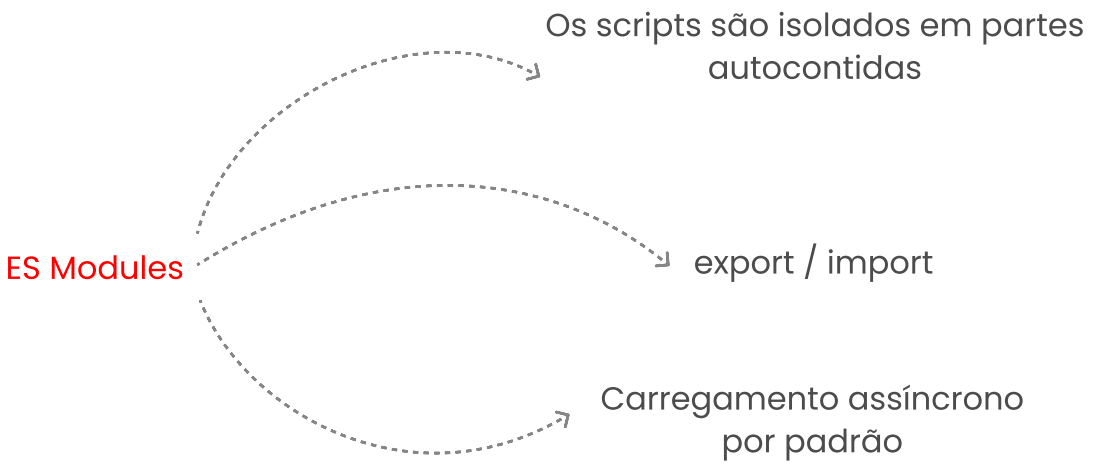
Compilando TypeScript para JavaScript

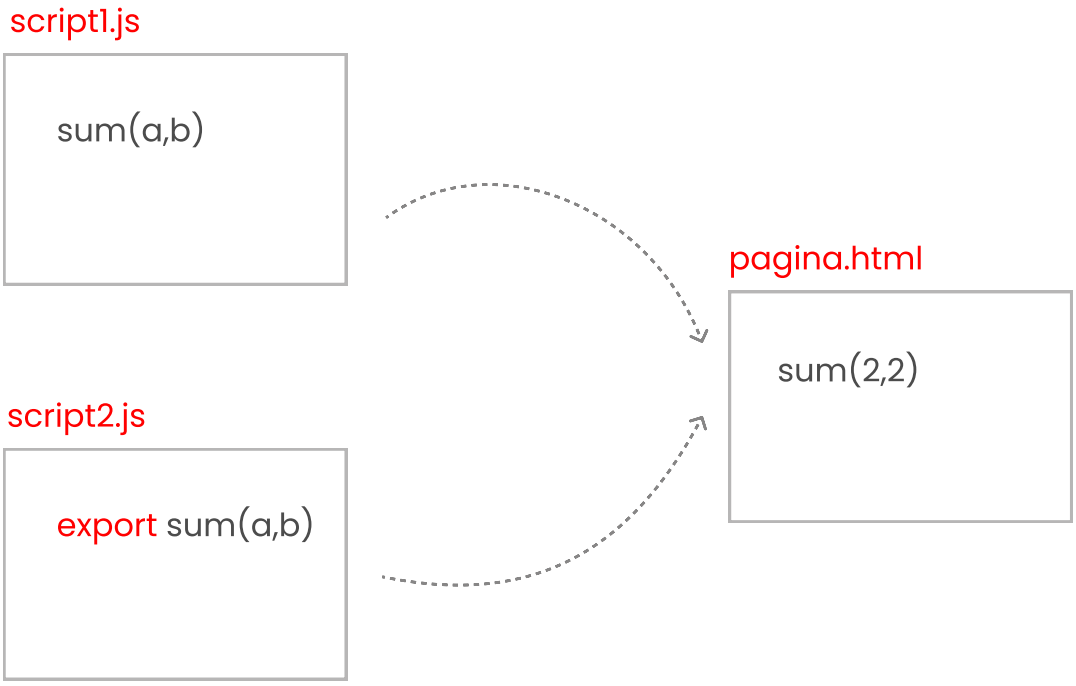


Script Mode

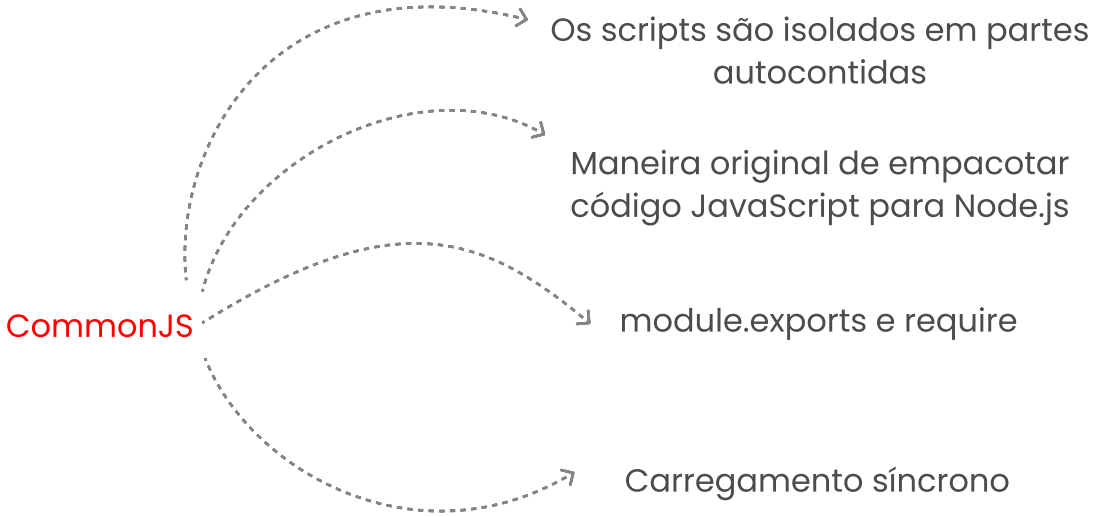

Script Mode
Os diferentes scripts são interpretados como fazendo parte de um todo único
script1.js
sum(a,b)
script2.js
sum(a,b)
pagina.html
sum(2,2)

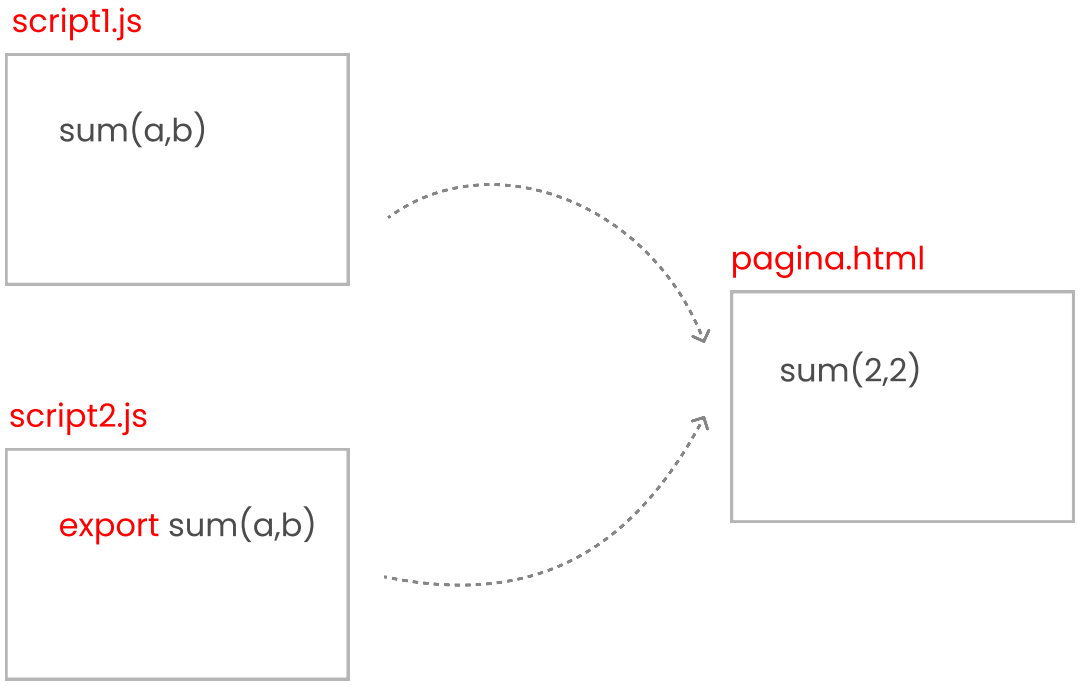
ES Modules


Os scripts são isolados em partes autocontidas
ES Modules
export / import
Carregamento assíncrono por padrão

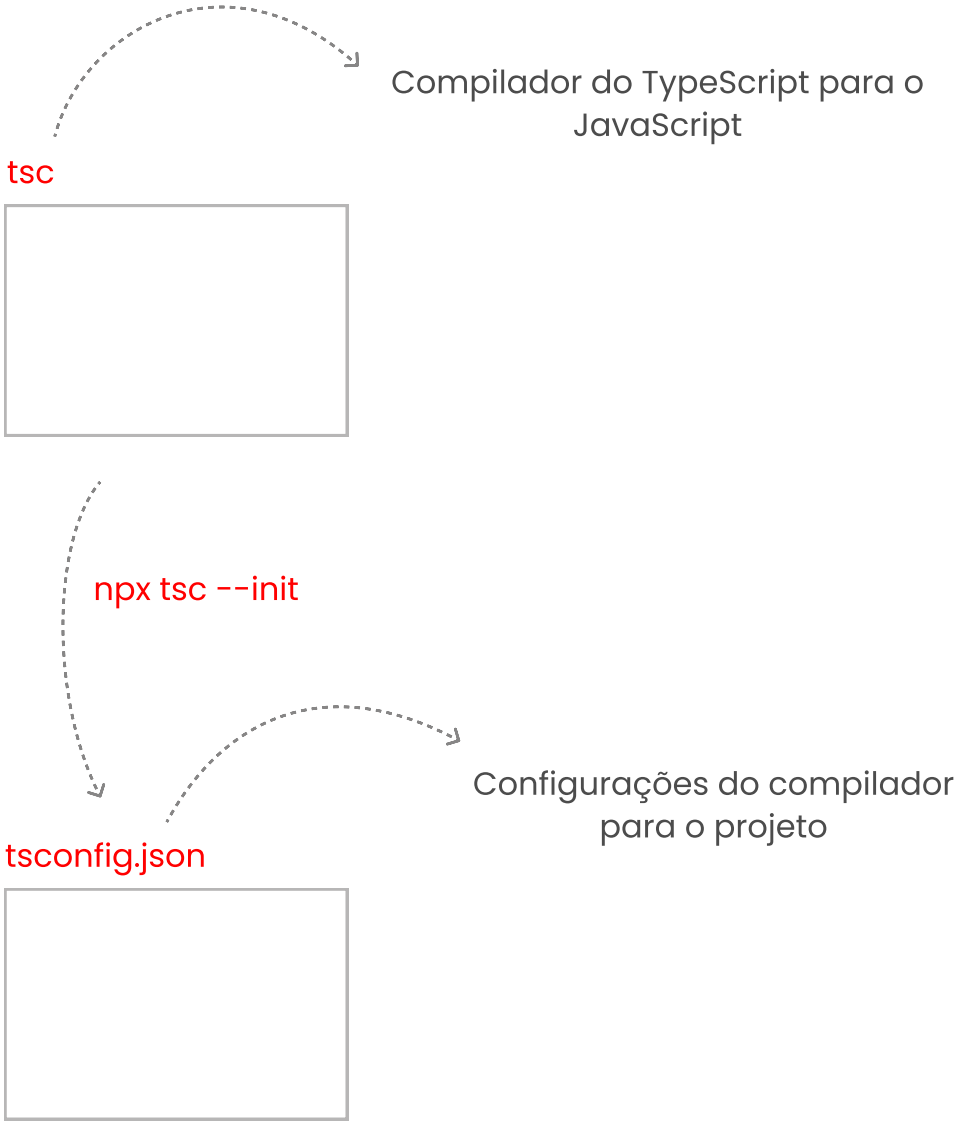

script1.js
sum(a,b)
pagina.html
sum(2,2)
script2.js
export sum(a,b)

CommonJS

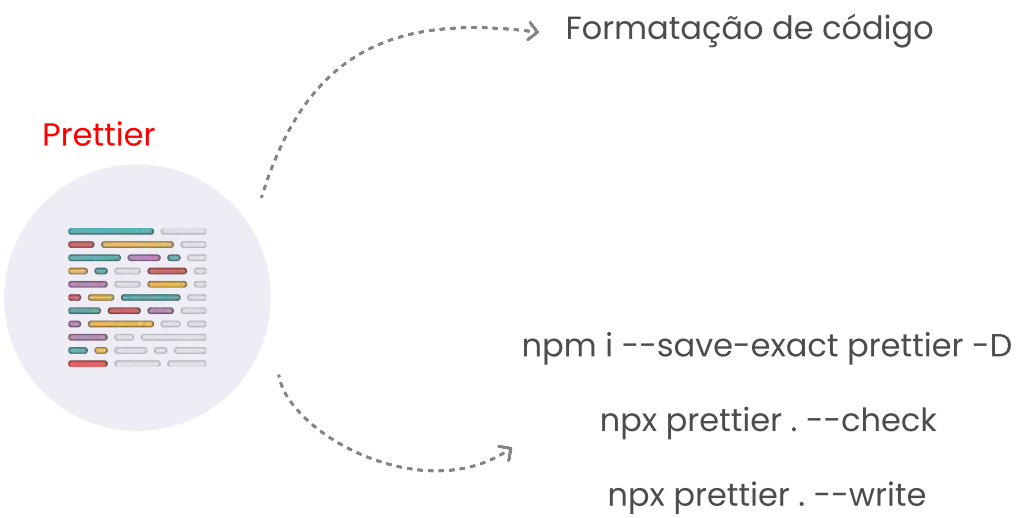

Os scripts são isolados em partes autocontidas
Maneira original de empacotar código JavaScript para Node.js
CommonJS
module.exports e require
Carregamento síncrono


script1.js
sum(a,b)
pagina.html
sum(2,2)
script2.js
export sum(a,b)

Iniciando o TypeScript no projeto (introdução ao tsconfig.json)



Prettier - Introdução, instalação e uso



VS Code - Format on Save e End of Line

Versionando o projeto com o Git

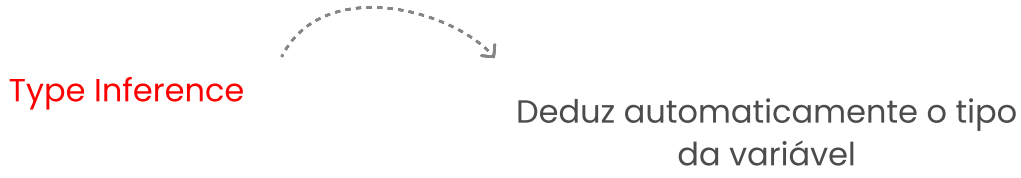
Git

git init

git add .

git commit -m"mensagem"

Type Inference



Type Annotation

Number	number
String	string
BigInt	bigint
Boolean	boolean
Symbol	symbol
Null	null
Undefined	undefined
Object	object

: type

```
export function display(product: string, price: number) {
  console.log(product, price);
}
```

Utilize a anotação de tipo apenas quando a inferência de tipo não for possível

Tipo Array

number

string

bigint

boolean

symbol

null

undefined

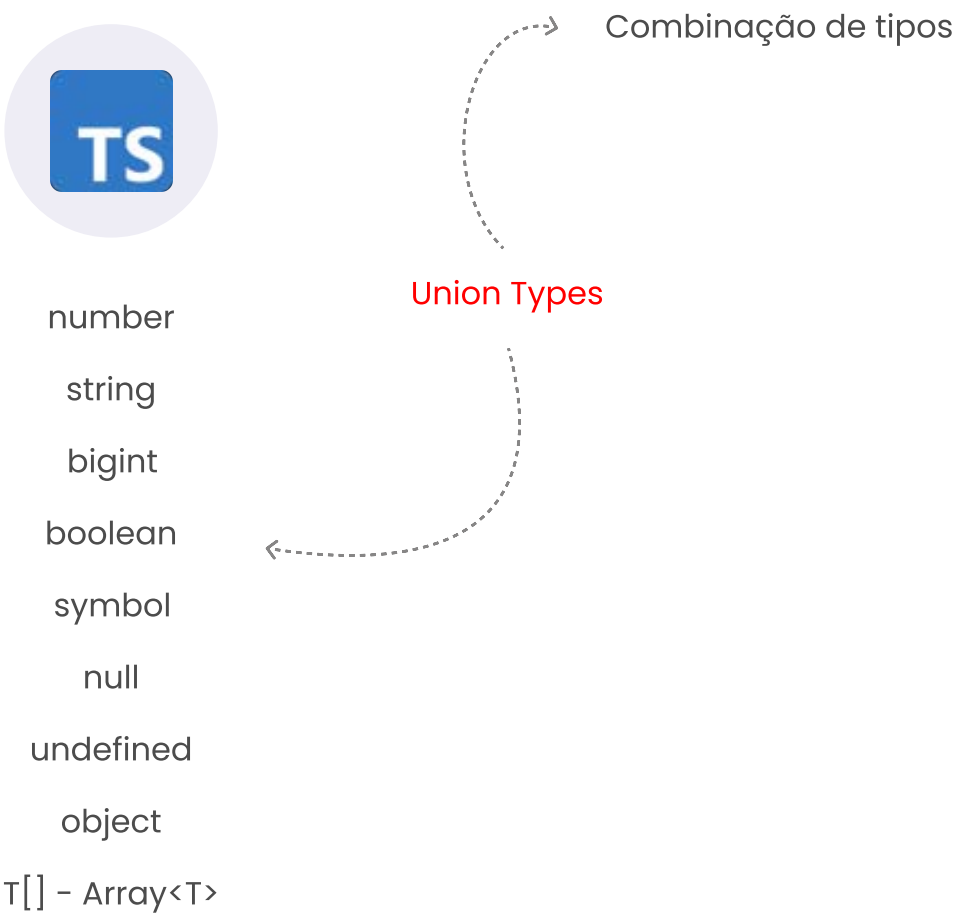
object

T[] - Array<T>

Interface para manipular tipos genéricos

Type Annotation convencional

Union Types


Combinação de tipos
TS
Union Types
number
string
bigint
boolean
symbol
null
undefined
object
T[] - Array<T>

Tipo Tuple

number

string

bigint

boolean

symbol

null

undefined

object

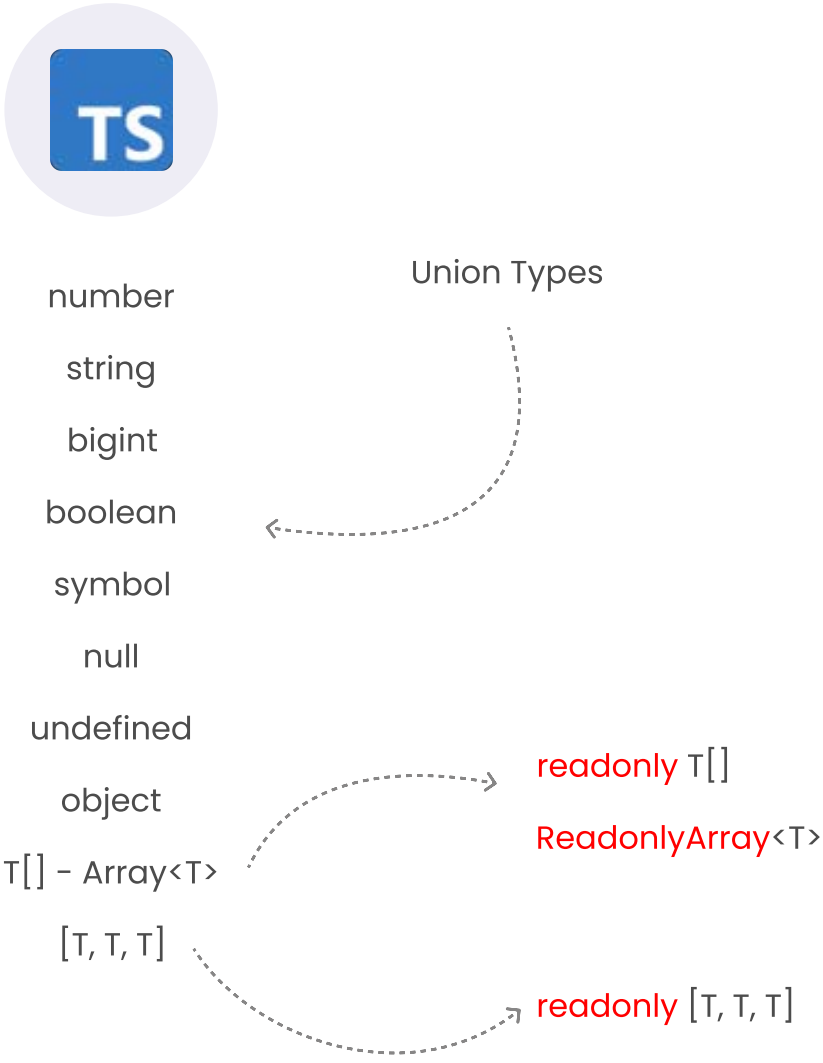
T[] - Array<T>

[T, T, T]

Union Types

Tuplas (tamanho e tipos fixos)

Readonly em Arrays e Tuples



Tipo Object - Inferência e Anotação de Tipo

number

string

bigint

boolean

symbol

null

undefined

object

T[] - Array<T>

[T, T, T]

Union Types

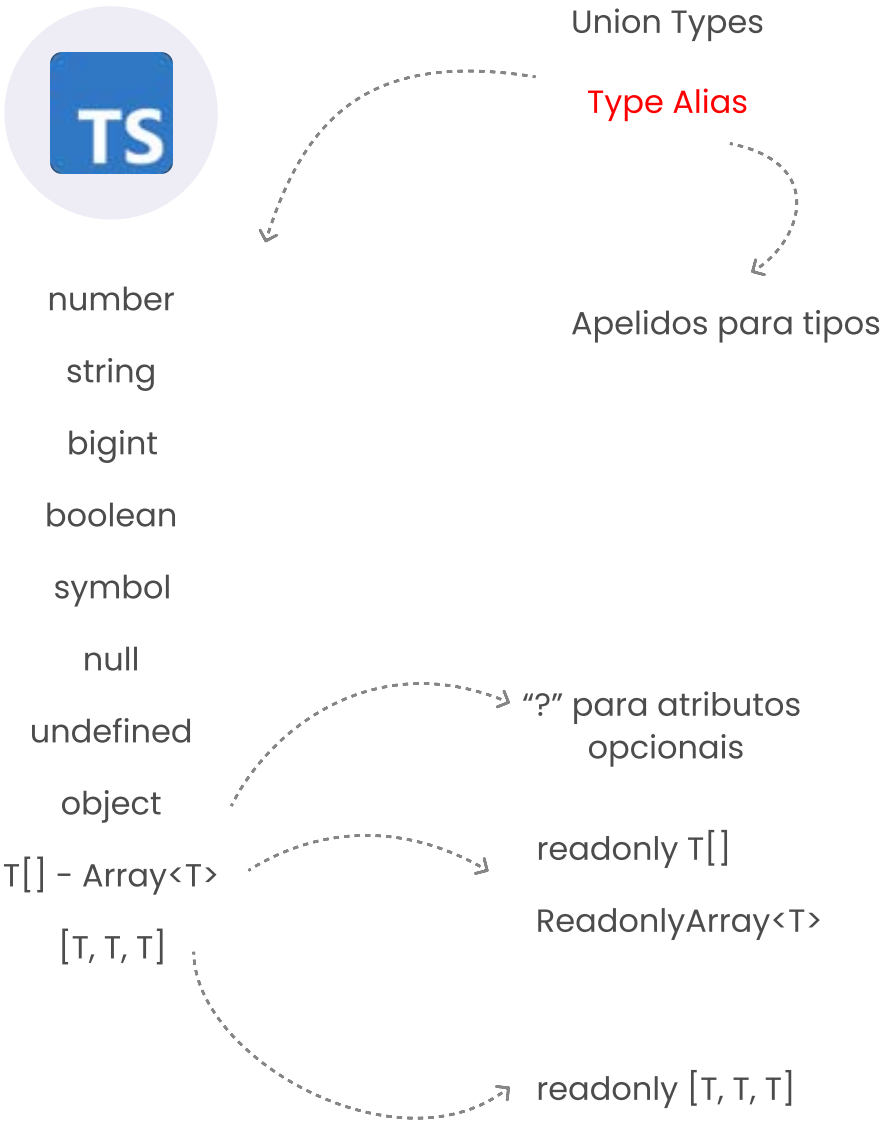
"?" para atributos opcionais

readonly T[]

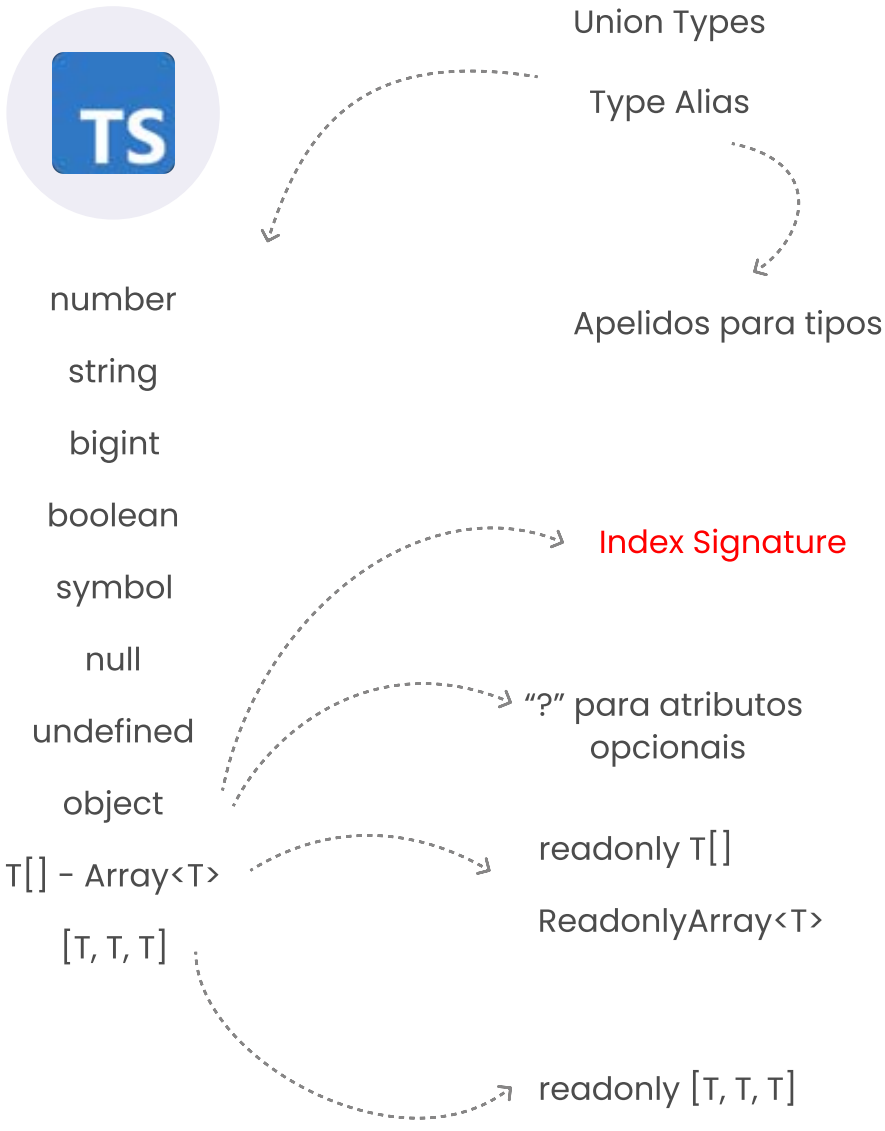
ReadonlyArray<T>

readonly [T, T, T]

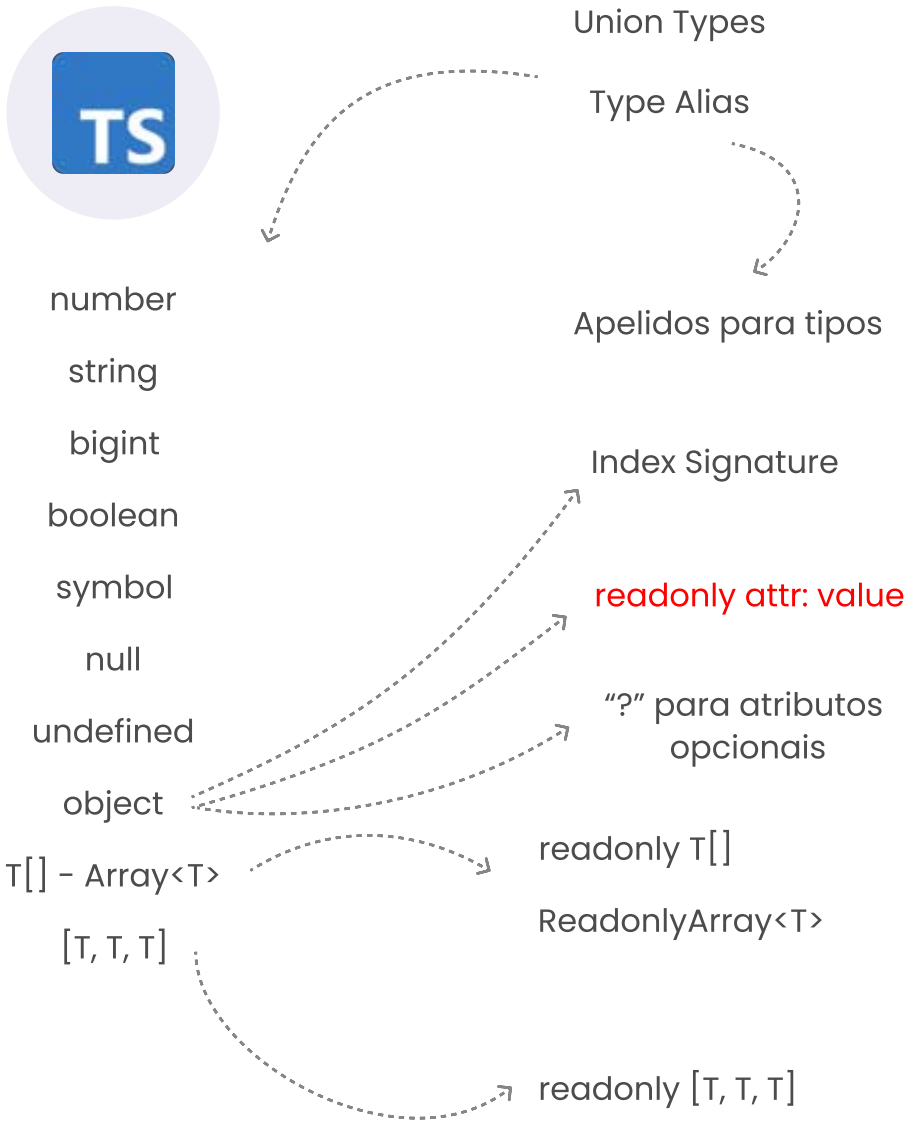
Type Alias


Union Types
Type Alias
Apelidos para tipos
number
string
bigint
boolean
symbol
null
undefined
object
T[] - Array<T>
[T, T, T]
"?" para atributos opcionais
readonly T[]
ReadonlyArray<T>
readonly [T, T, T]

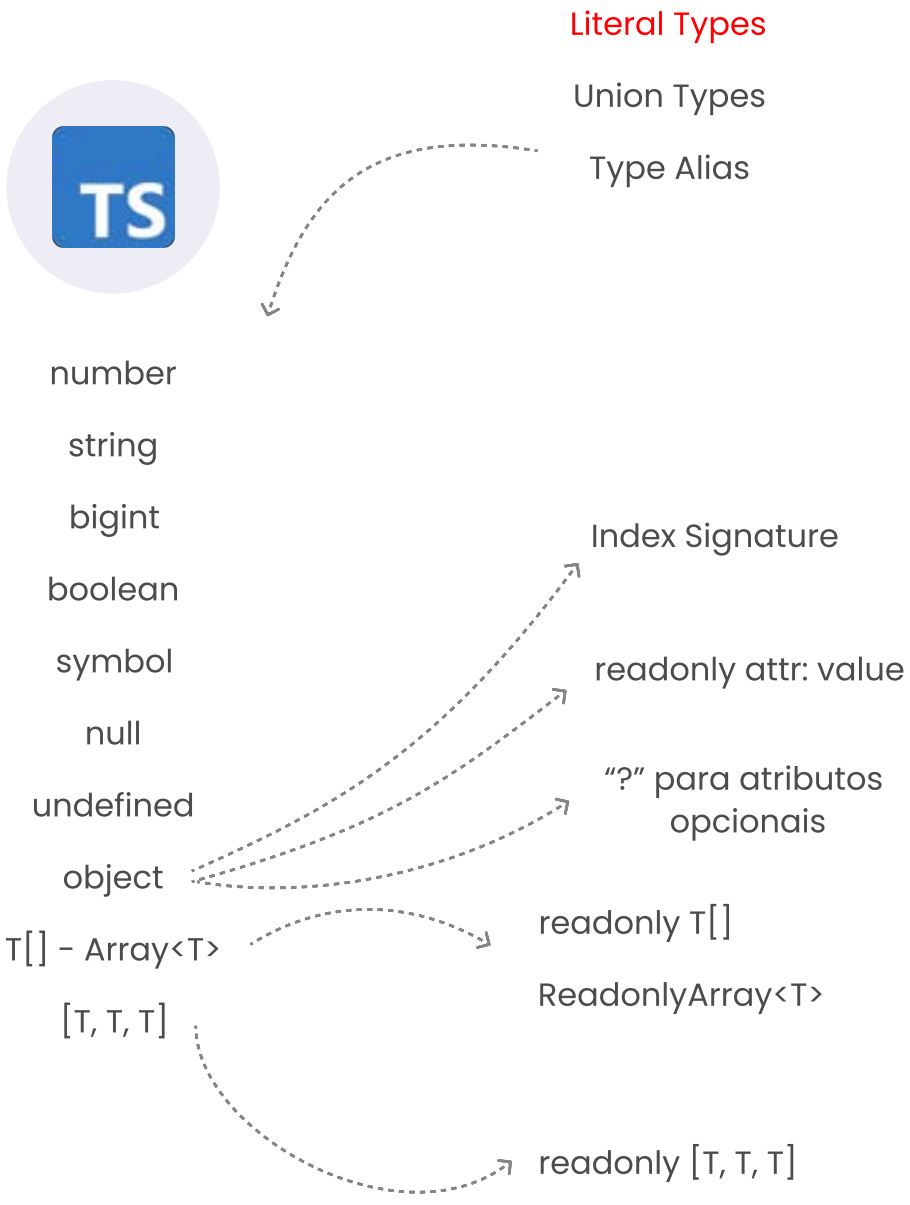
Tipo Object - Index Signature



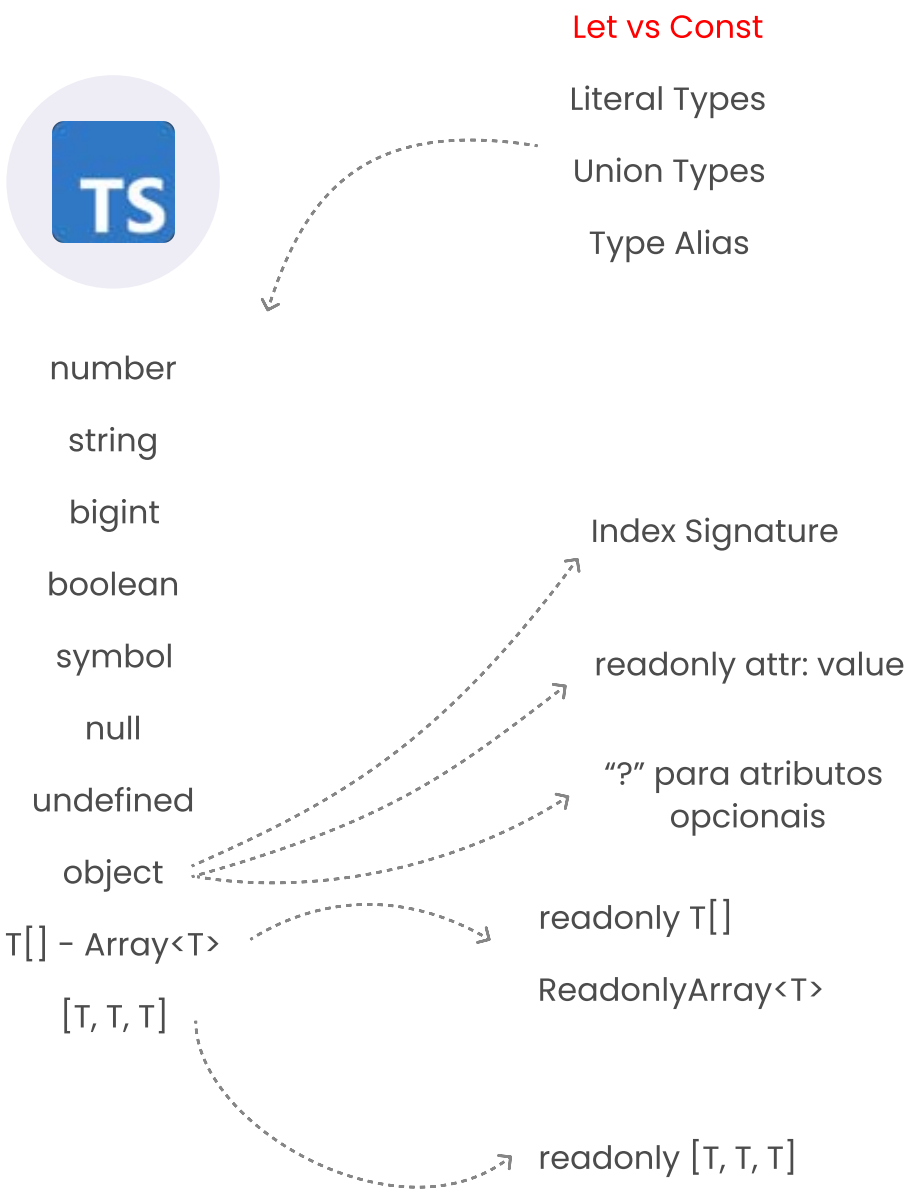
Tipo Object - Readonly



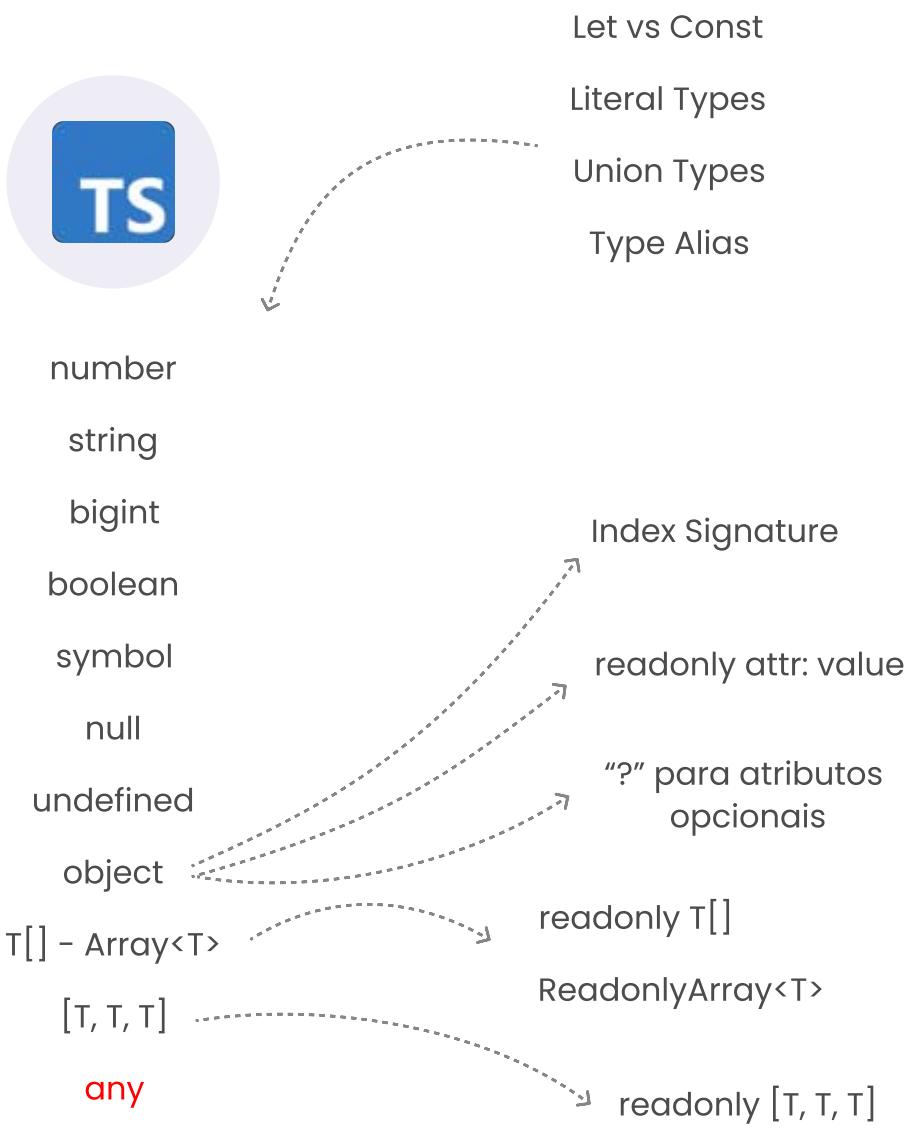
Literal Types



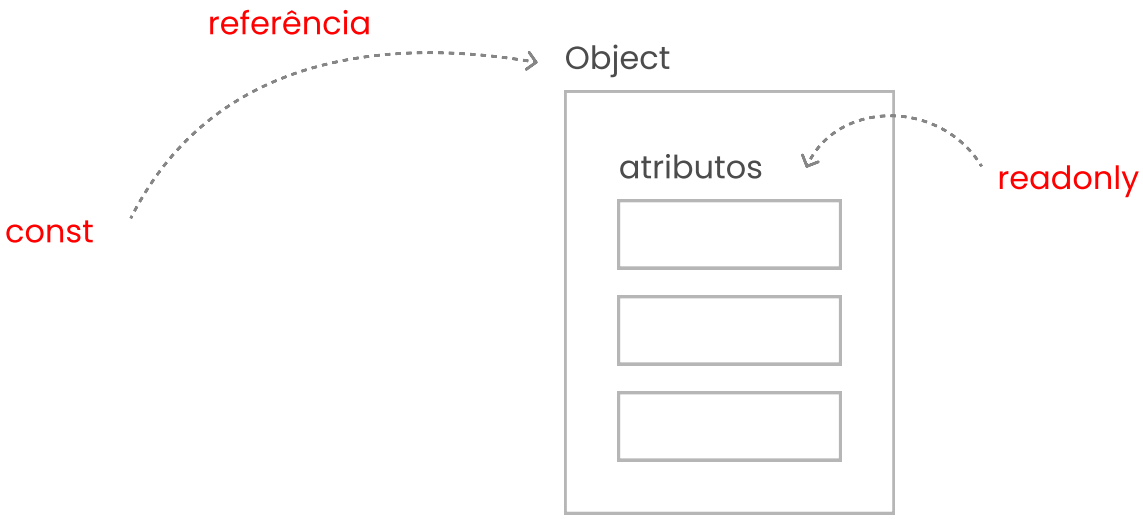
Let vs Const


Let vs Const
Literal Types
Union Types
Type Alias
TS
number
string
bigint
boolean
symbol
null
undefined
object
T[] - Array<T>
[T, T, T]
Index Signature
readonly attr: value
"?" para atributos opcionais
readonly T[]
ReadonlyArray<T>
readonly [T, T, T]

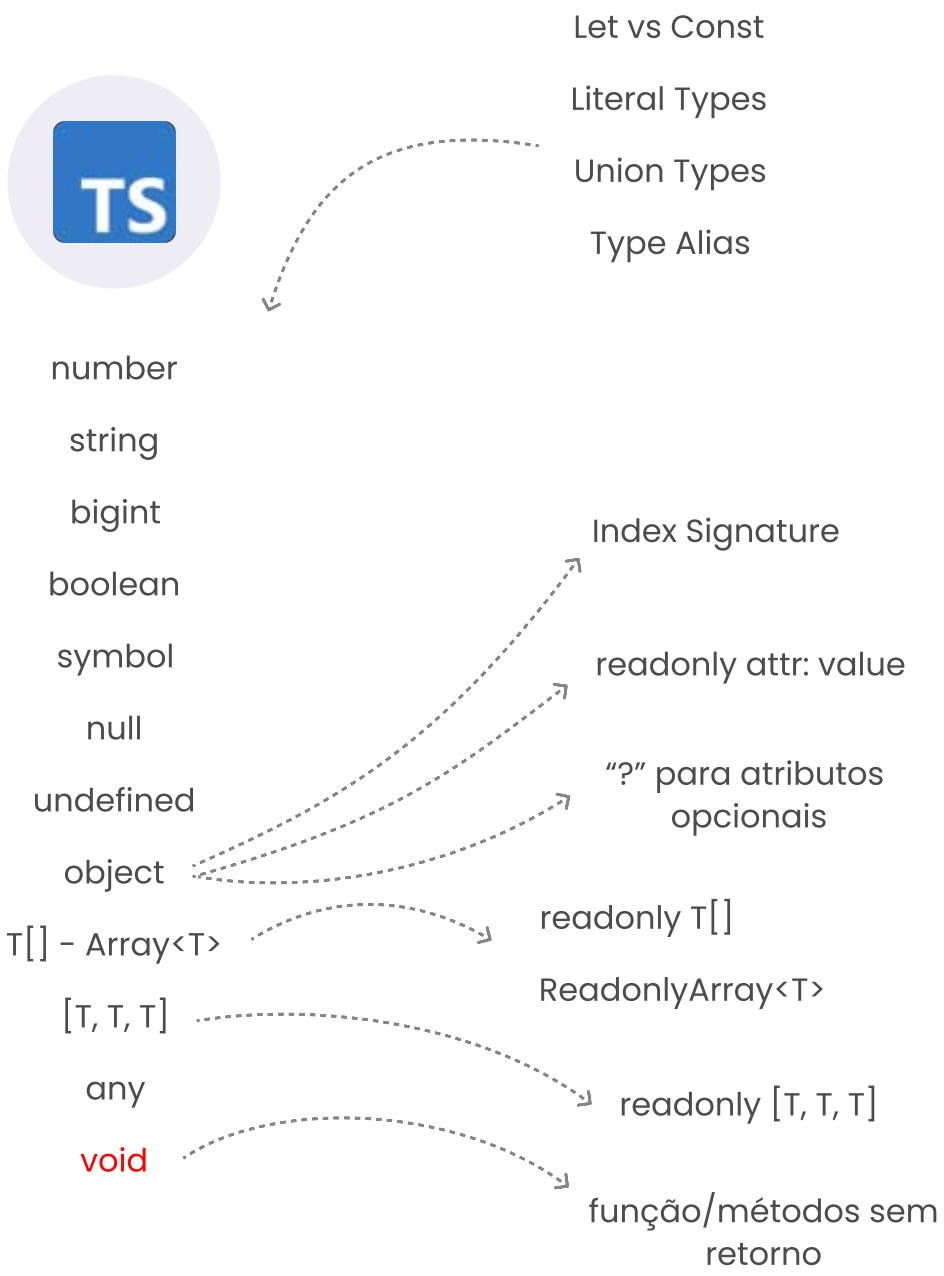
Tipo Any


Let vs Const
Literal Types
Union Types
Type Alias
number
string
bigint
boolean
symbol
null
undefined
object
T[] - Array<T>
[T, T, T]
any
Index Signature
readonly attr: value
"?" para atributos opcionais
readonly T[]
ReadonlyArray<T>
readonly [T, T, T]

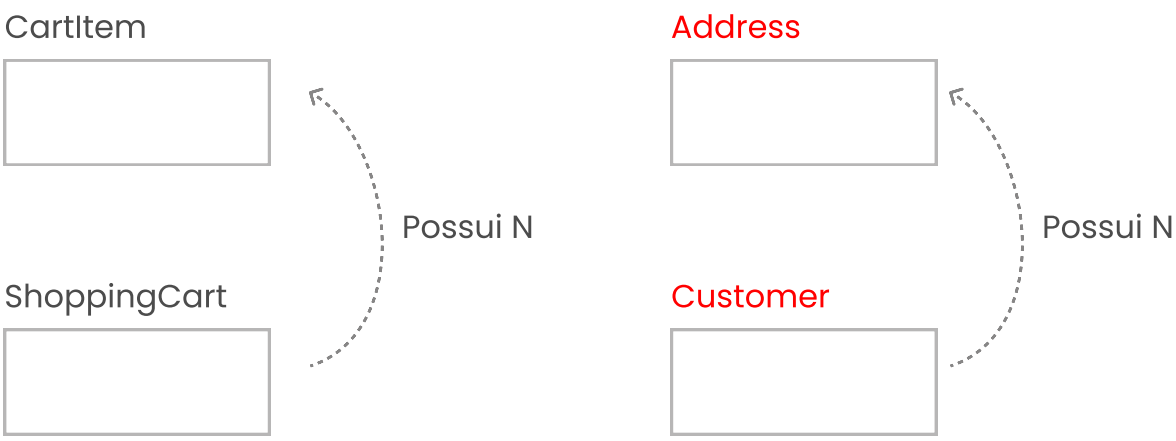
Const no Contexto de Objetos



Função - Tipo Void



Função - Tipo Return



Exige e determina o tipo do retorno

```
getPrincipalAddress(customer: Customer): Address | undefined
```

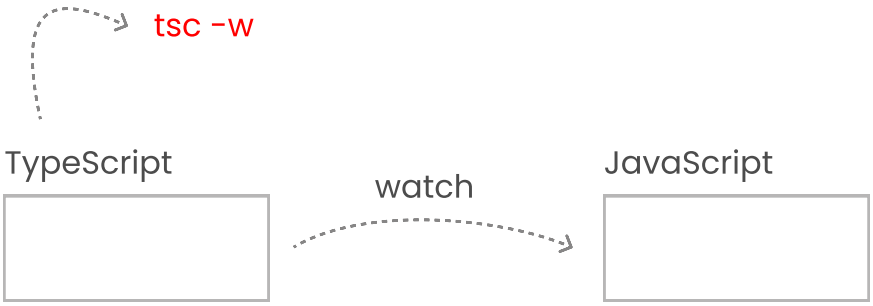
Função - Type Annotation

```
let calculateTotal: (sC: ShoppingCart) => number;
```

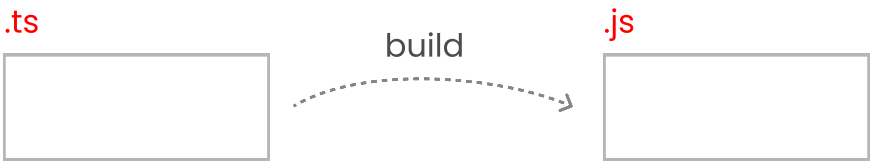
```
calculateTotal = function (shoppingCart: ShoppingCart): number {
```

Executando os scripts compilados no Browser

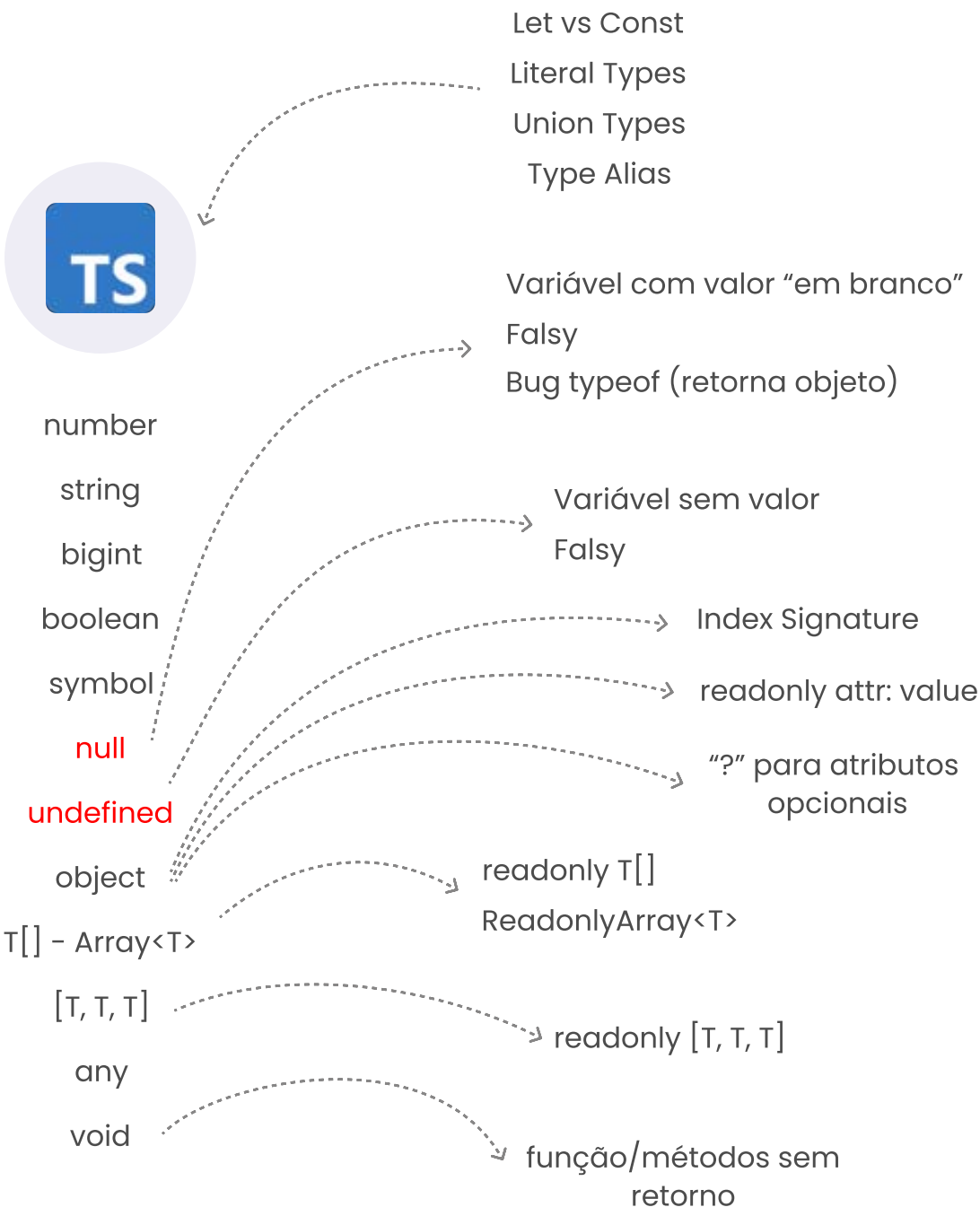
Compilação automática



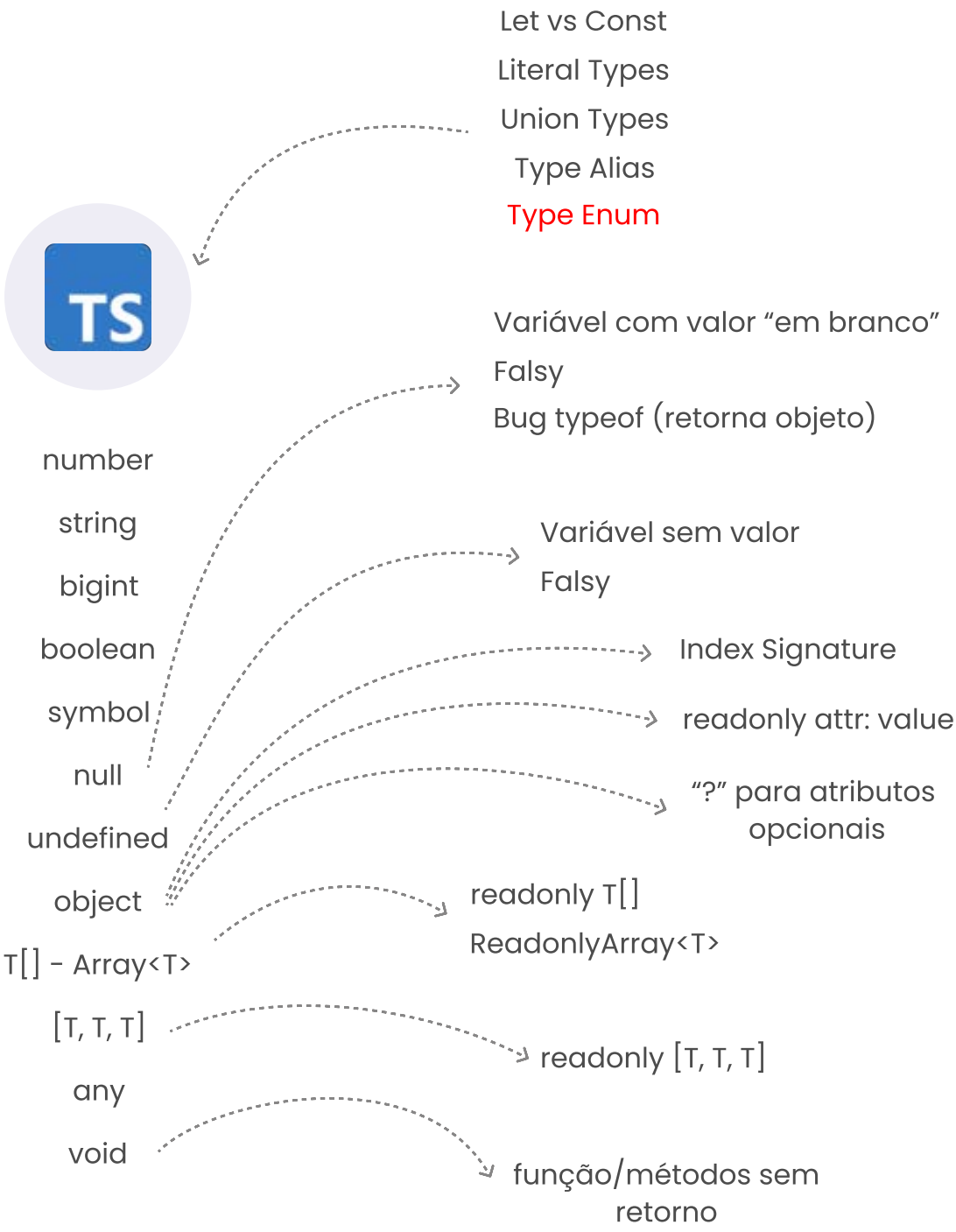
Extensões TS e JS Antes e Depois do Build



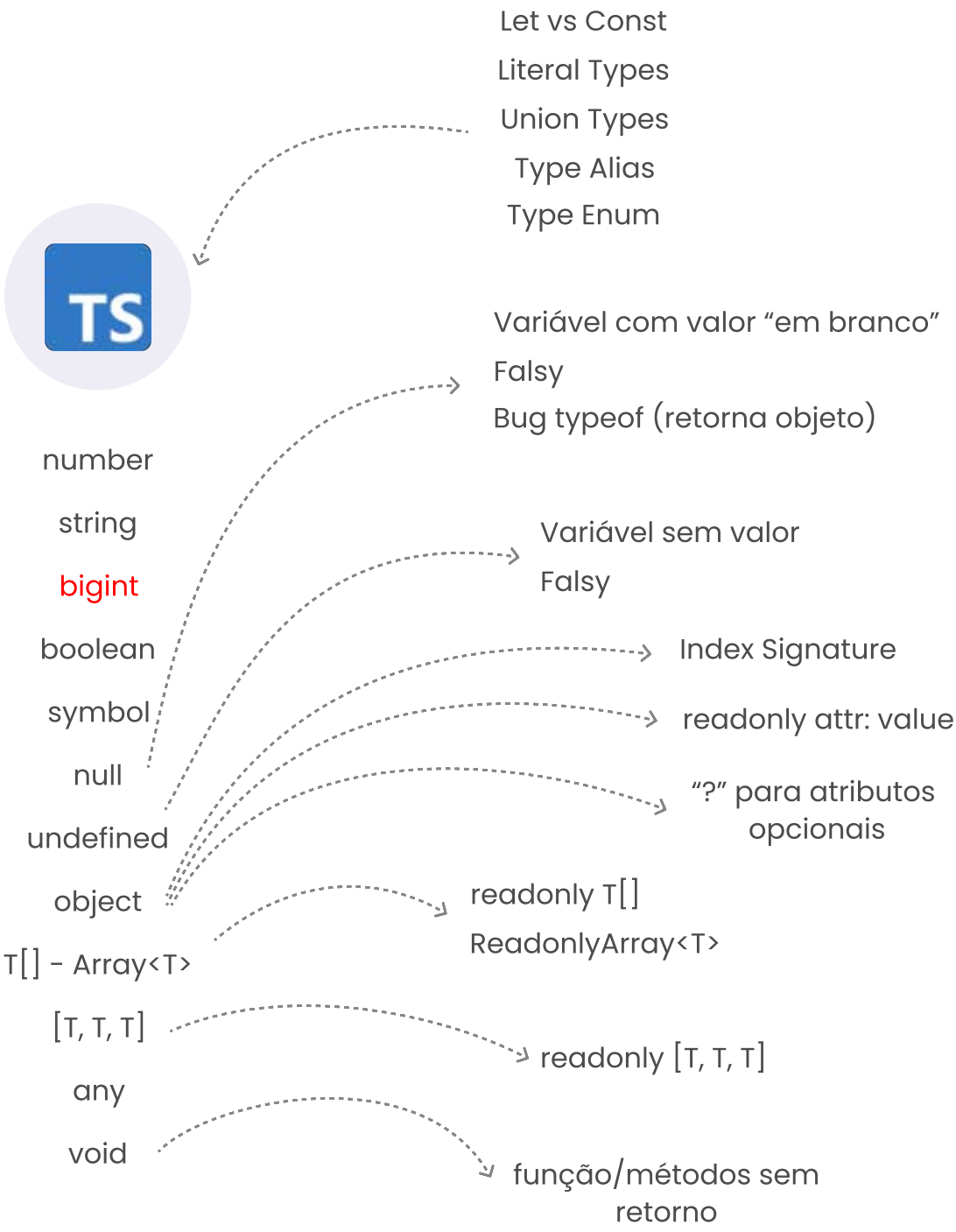
Tipos Null e Undefined



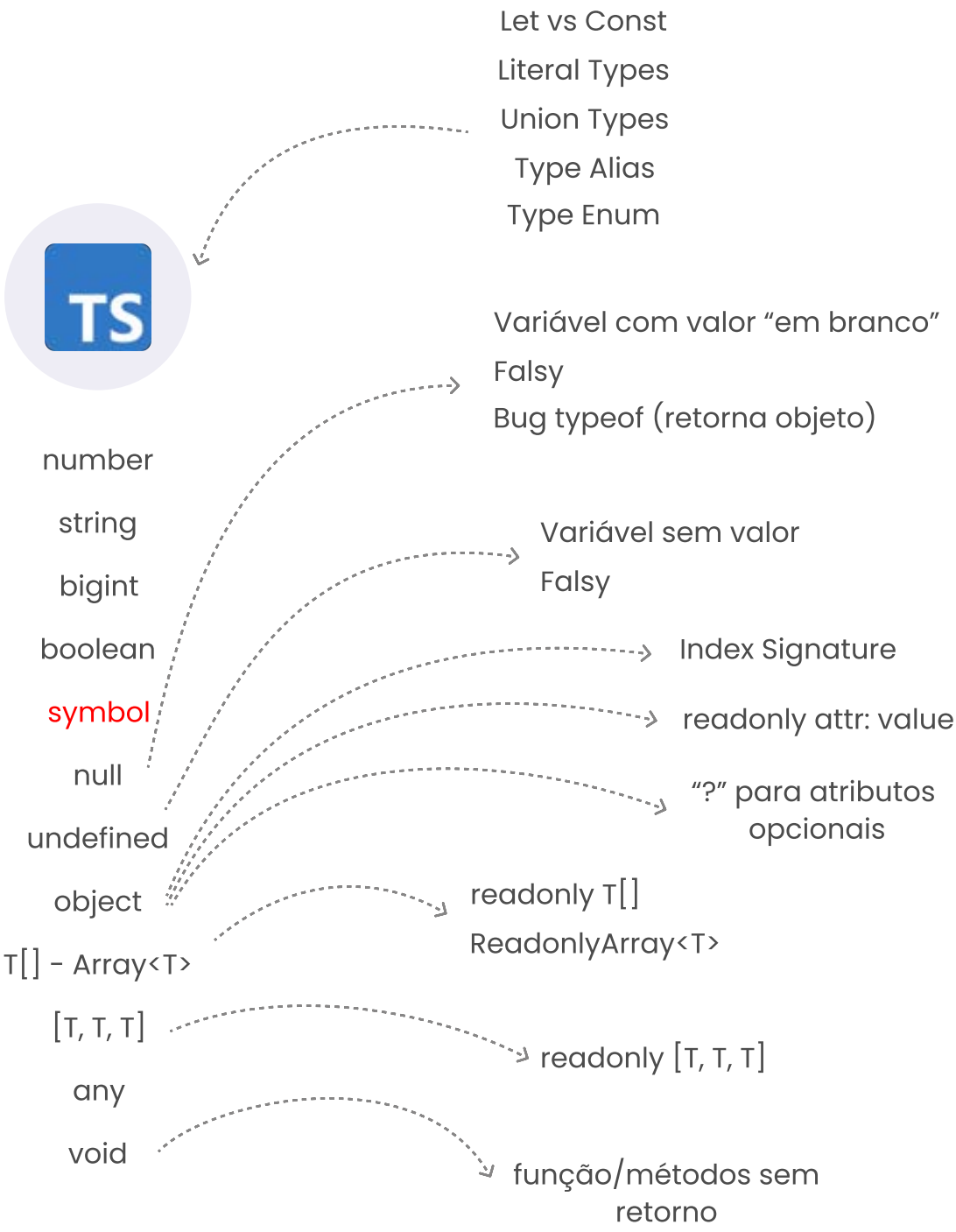
Tipo Enum


Let vs Const
Literal Types
Union Types
Type Alias
Type Enum
TS
Variável com valor “em branco”
Falsy
Bug typeof (retorna objeto)
number
string
bigint
boolean
symbol
null
undefined
object
T[] - Array<T>
[T, T, T]
any
void
Variável sem valor
Falsy
Index Signature
readonly attr: value
“?” para atributos opcionais
readonly T[]
ReadonlyArray<T>
readonly [T, T, T]
função/métodos sem retorno

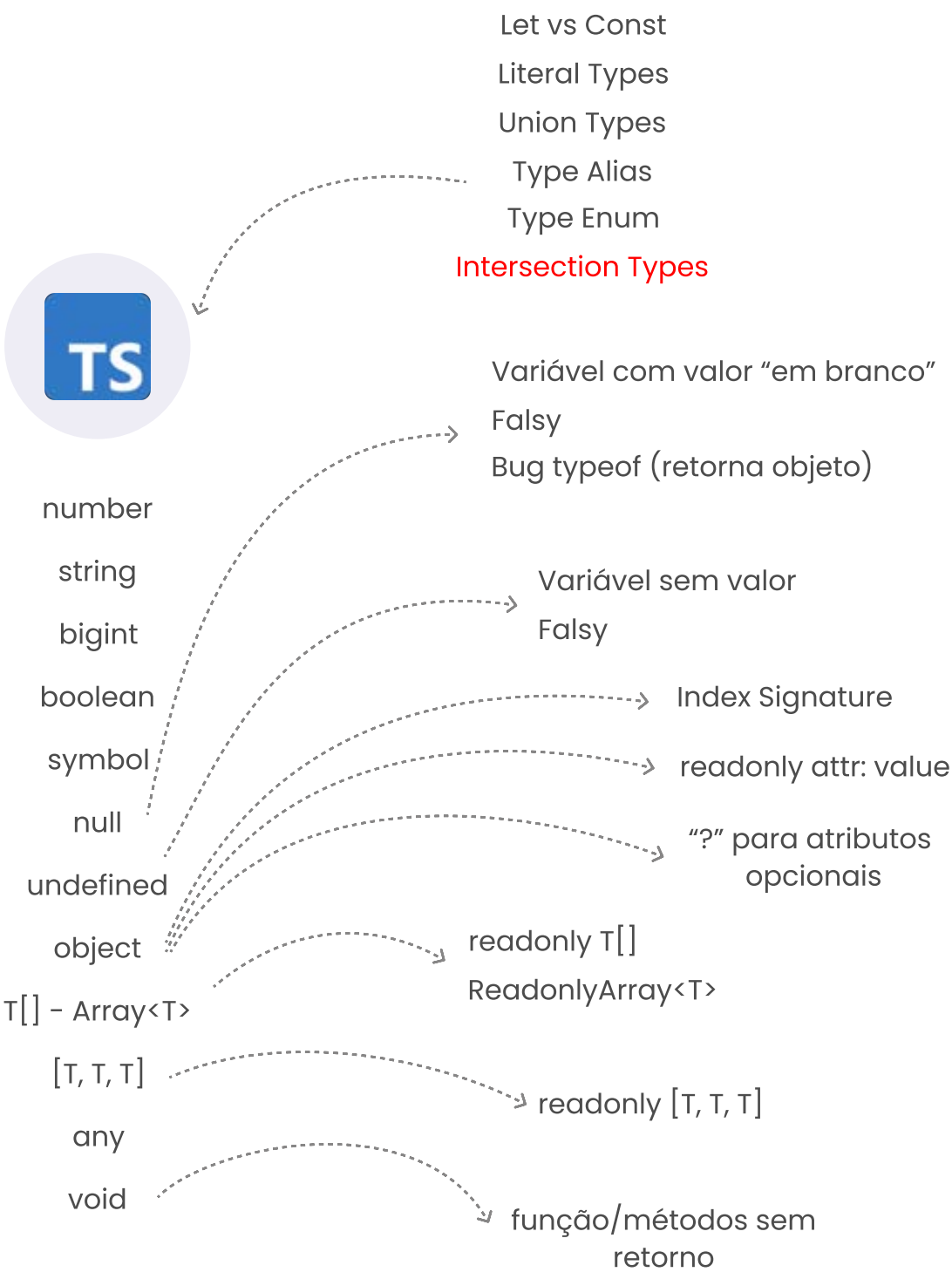
Tipo BigInt


Let vs Const
Literal Types
Union Types
Type Alias
Type Enum
TS
Variável com valor “em branco”
Falsy
Bug typeof (retorna objeto)
number
string
bigint
boolean
symbol
null
undefined
object
T[] - Array<T>
[T, T, T]
any
void
Variável sem valor
Falsy
Index Signature
readonly attr: value
“?” para atributos opcionais
readonly T[]
ReadonlyArray<T>
readonly [T, T, T]
função/métodos sem retorno

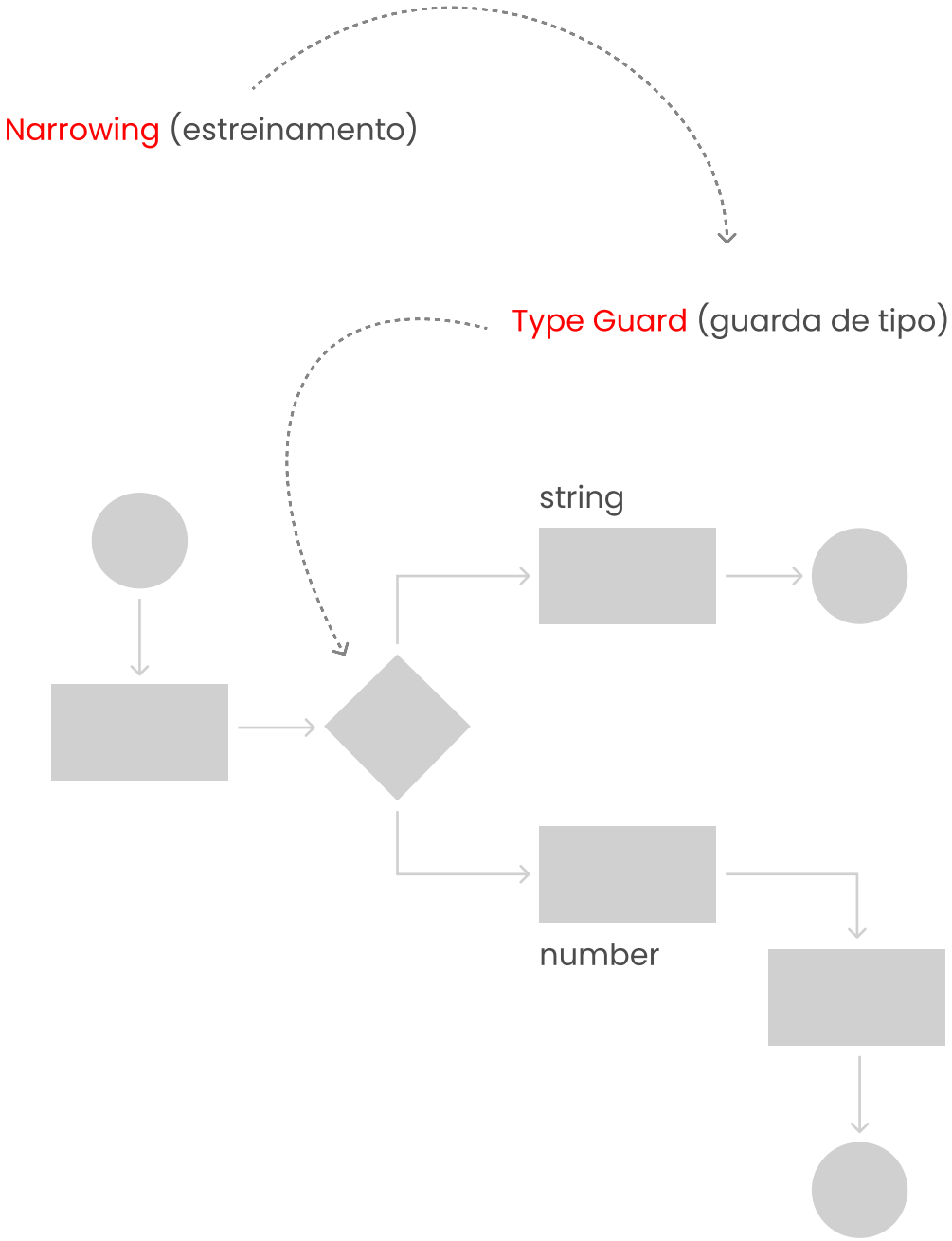
Tipo Symbol


Let vs Const
Literal Types
Union Types
Type Alias
Type Enum
TS
Variável com valor “em branco”
Falsy
Bug typeof (retorna objeto)
number
string
bigint
boolean
symbol
null
undefined
object
T[] - Array<T>
[T, T, T]
any
void
Variável sem valor
Falsy
Index Signature
readonly attr: value
“?” para atributos opcionais
readonly T[]
ReadonlyArray<T>
readonly [T, T, T]
função/métodos sem retorno

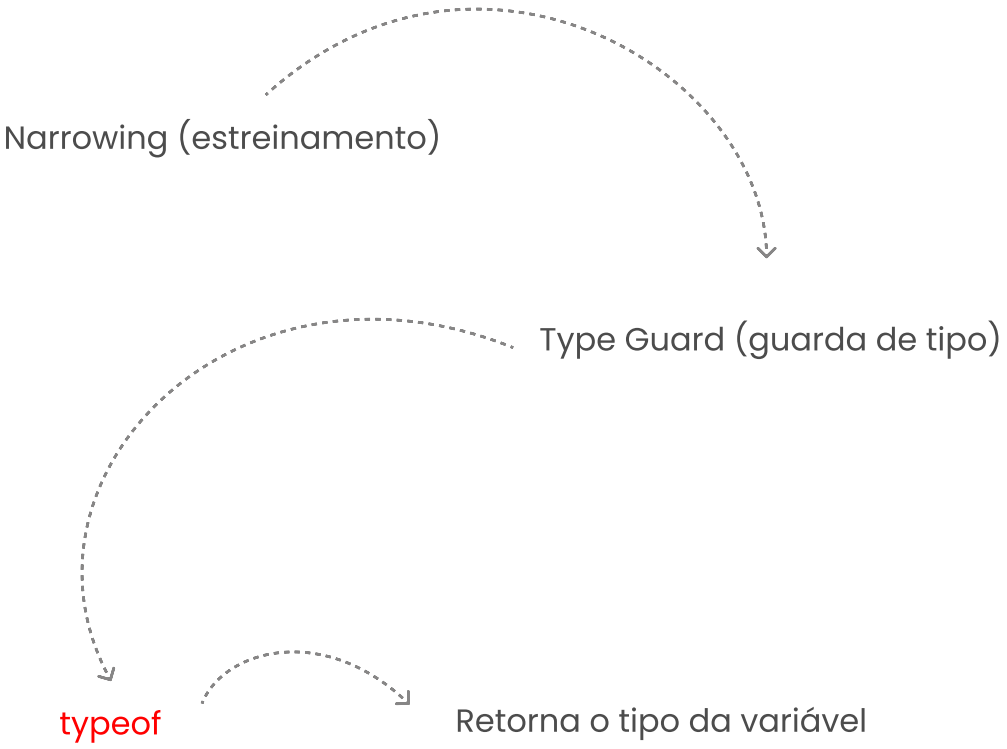
Intersection Types



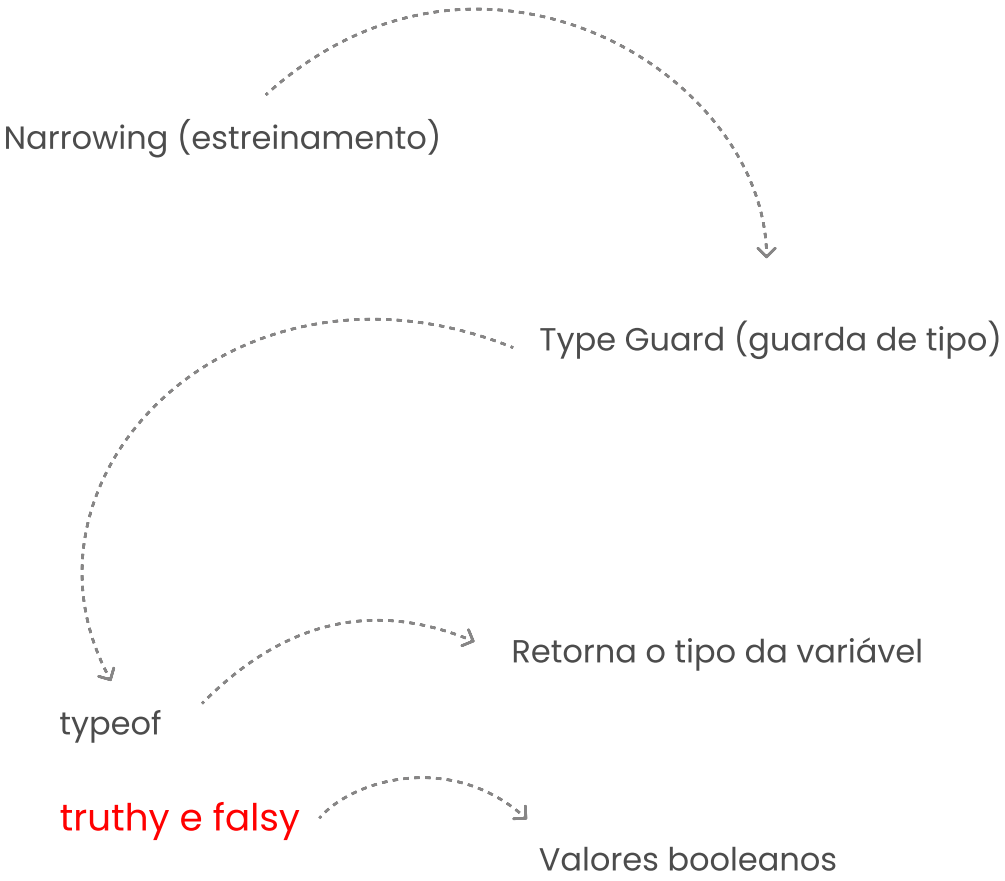
Type Guard


Narrowing (estreinamento)
Type Guard (guarda de tipo)
string
number

typeof


Narrowing (estreinamento)
Type Guard (guarda de tipo)
typeof
Retorna o tipo da variável

Valores Truthy e Falsy



Optional Chaining (?)

optional chaining (?) → Leitura segura de propriedades de objetos

Non-null Assertion (!)

non-null assertion (!)

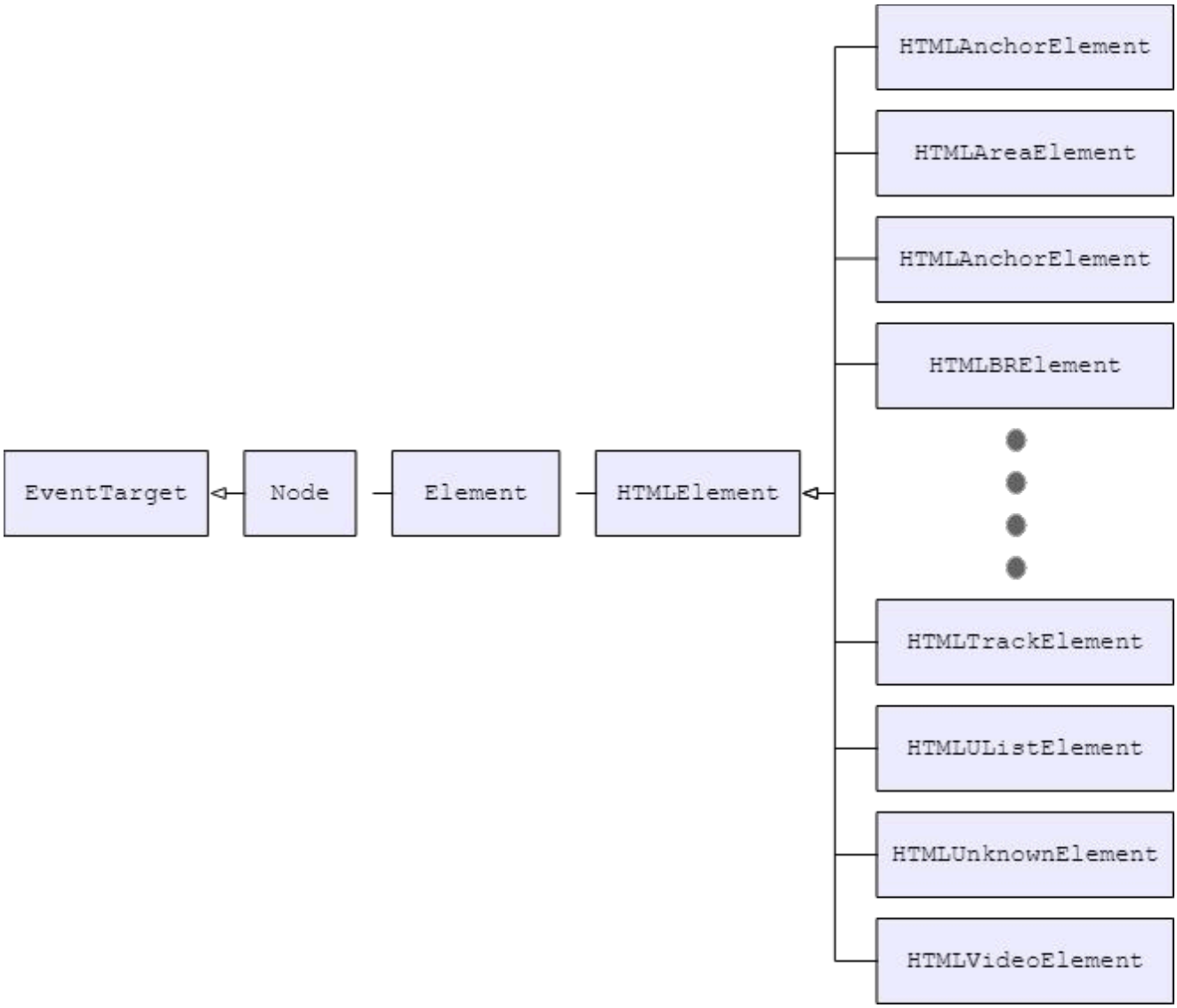
Indica que a variável ou expressão
não é null ou undefined

Type Assertion (as)

type assertion (as)

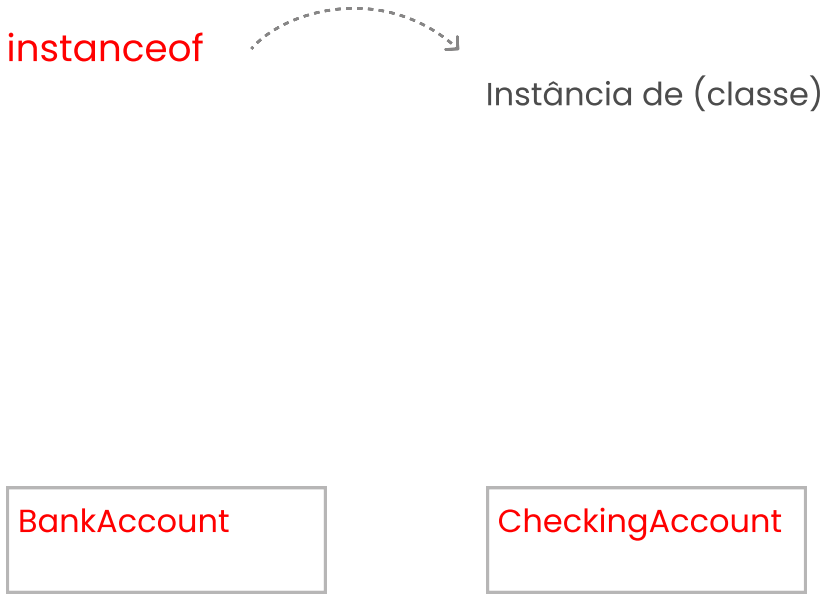
Definição de tipo mais específico

Interfaces dos elementos HTML

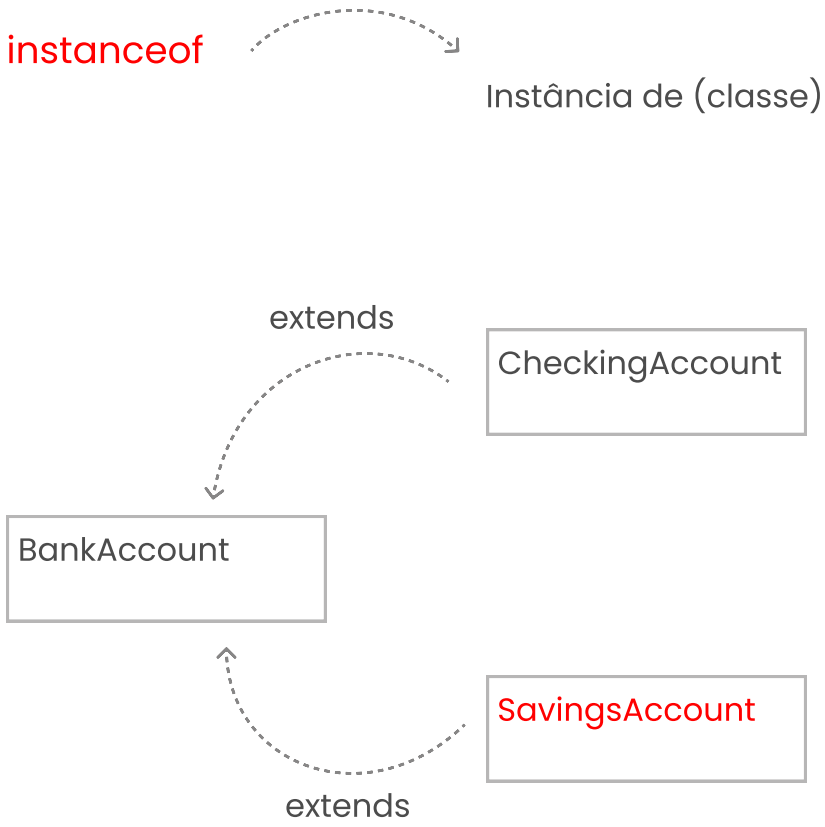


Fonte: https://developer.mozilla.org/en-US/docs/Web/API/HTML_DOM_API

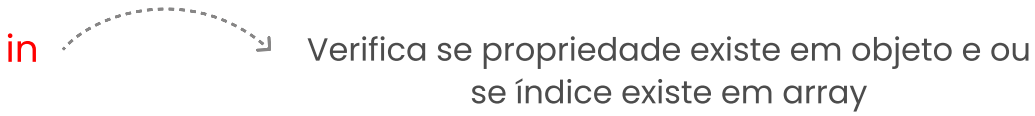
Verificando o Tipo do Objeto (Instanceof) Parte 1



Verificando o Tipo do Objeto (Instanceof) Parte 2

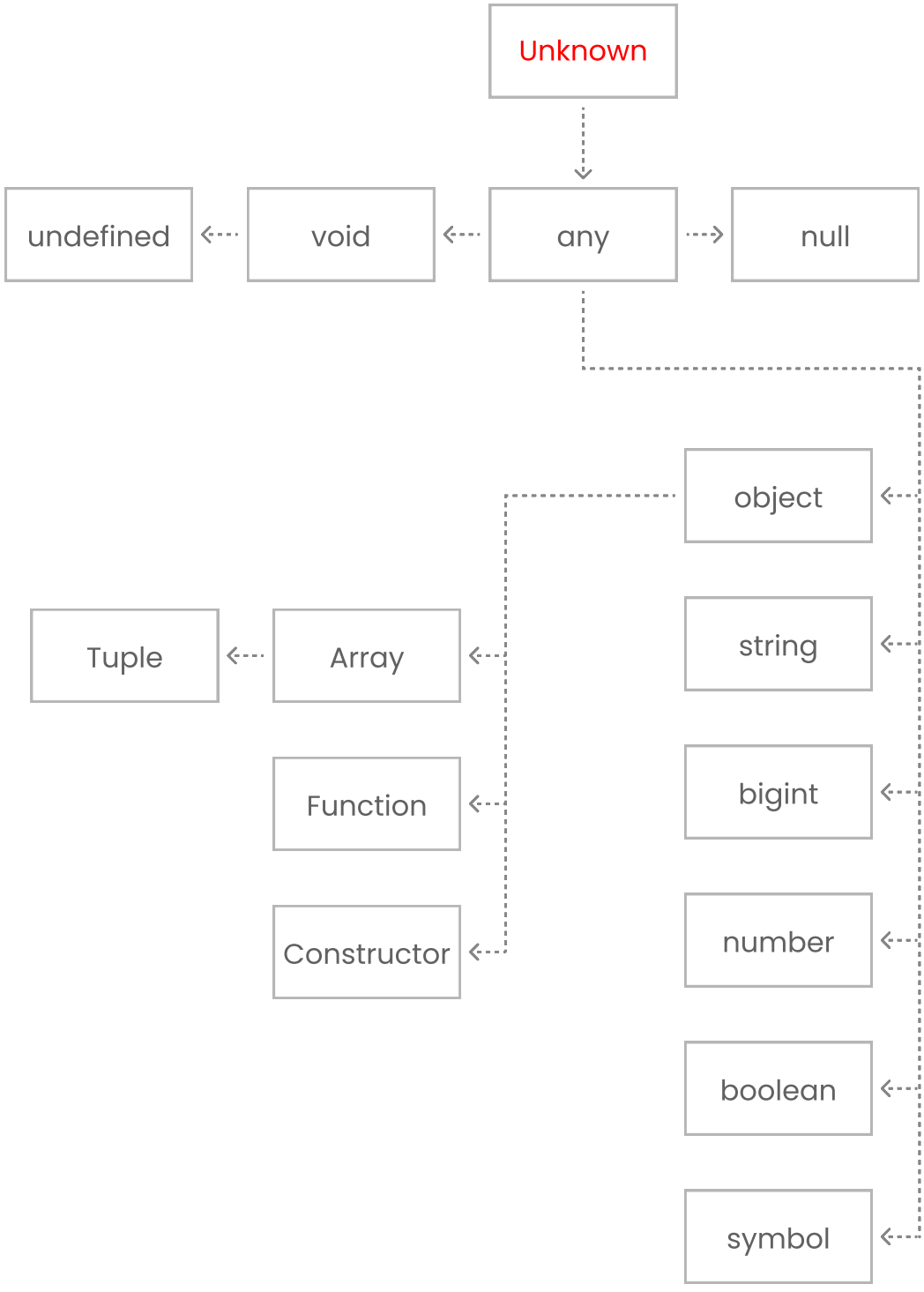


Verificando Propriedades de Objetos e Índices de Arrays (in)

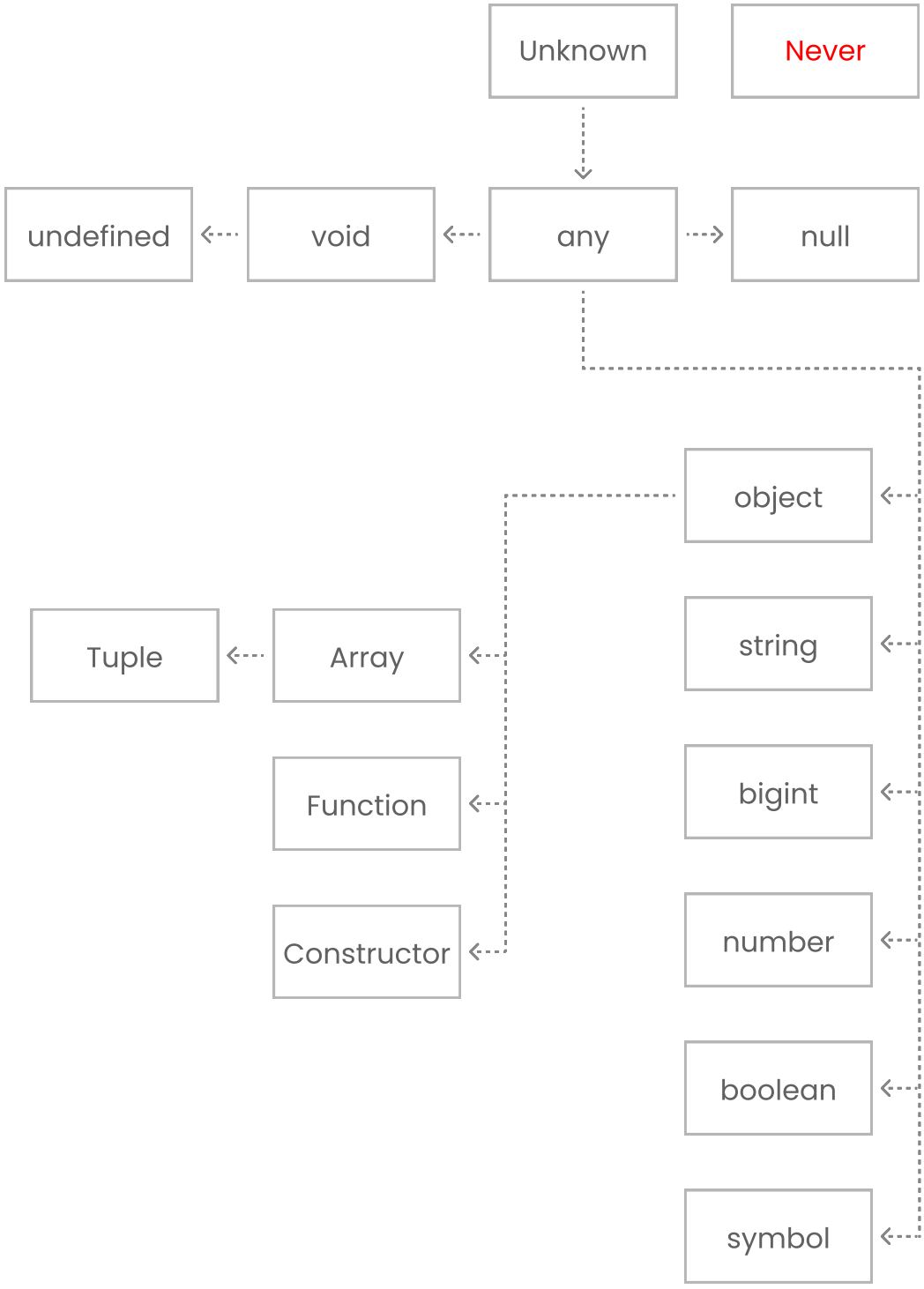


Array.isArray

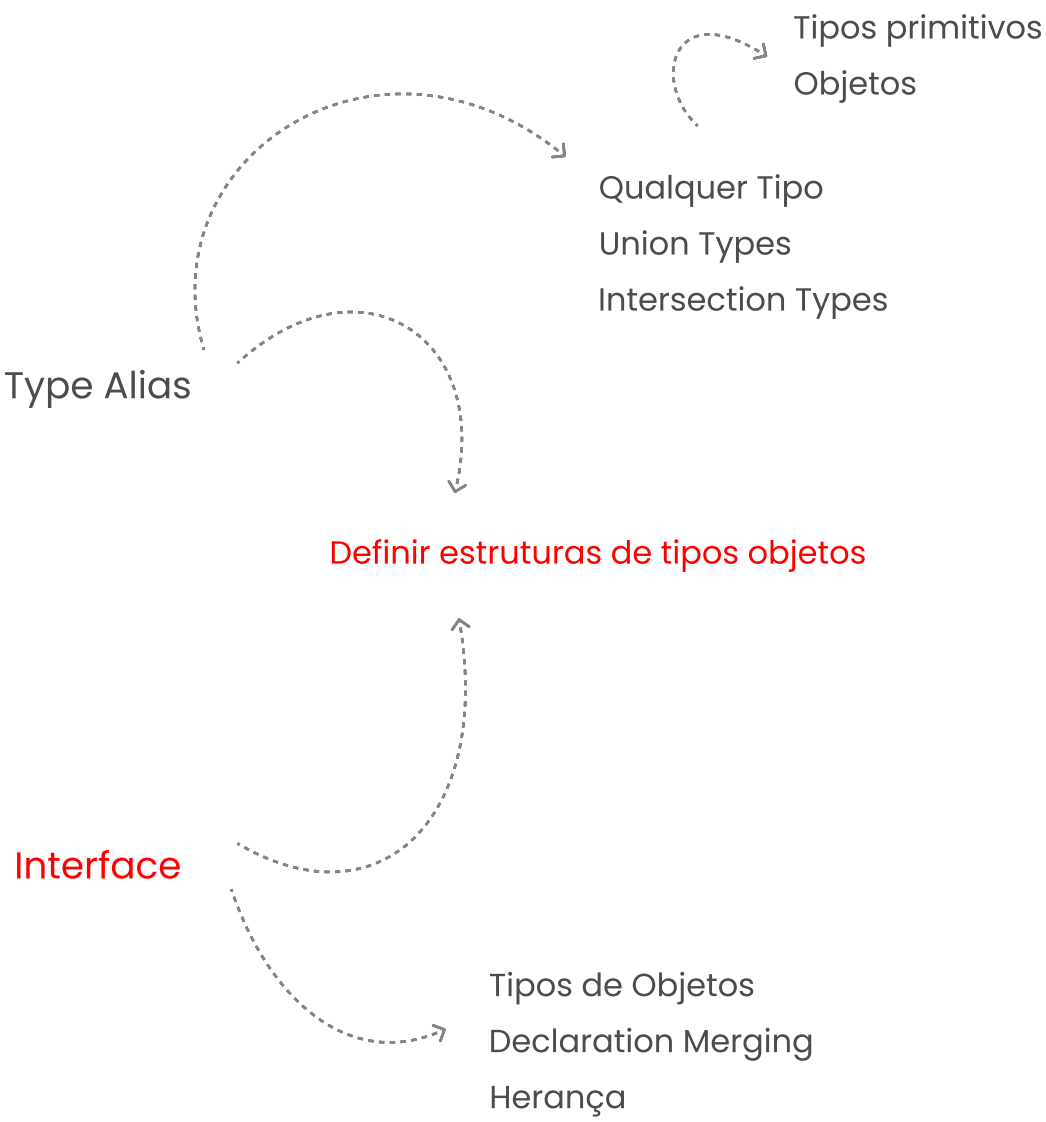
Hierarquia de Tipos e o Tipo Unknown



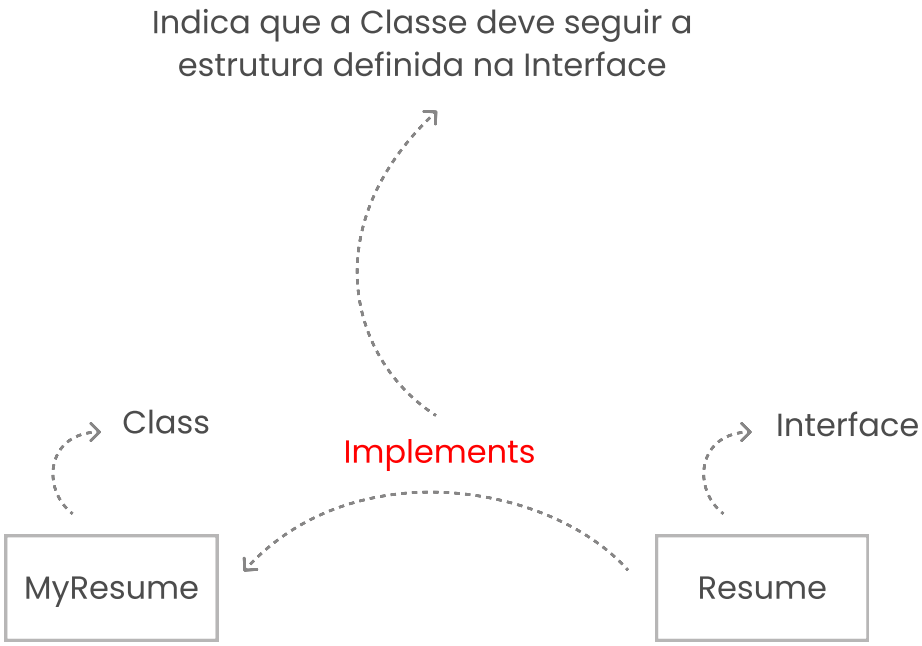
Hierarquia de Tipos e o Tipo Never



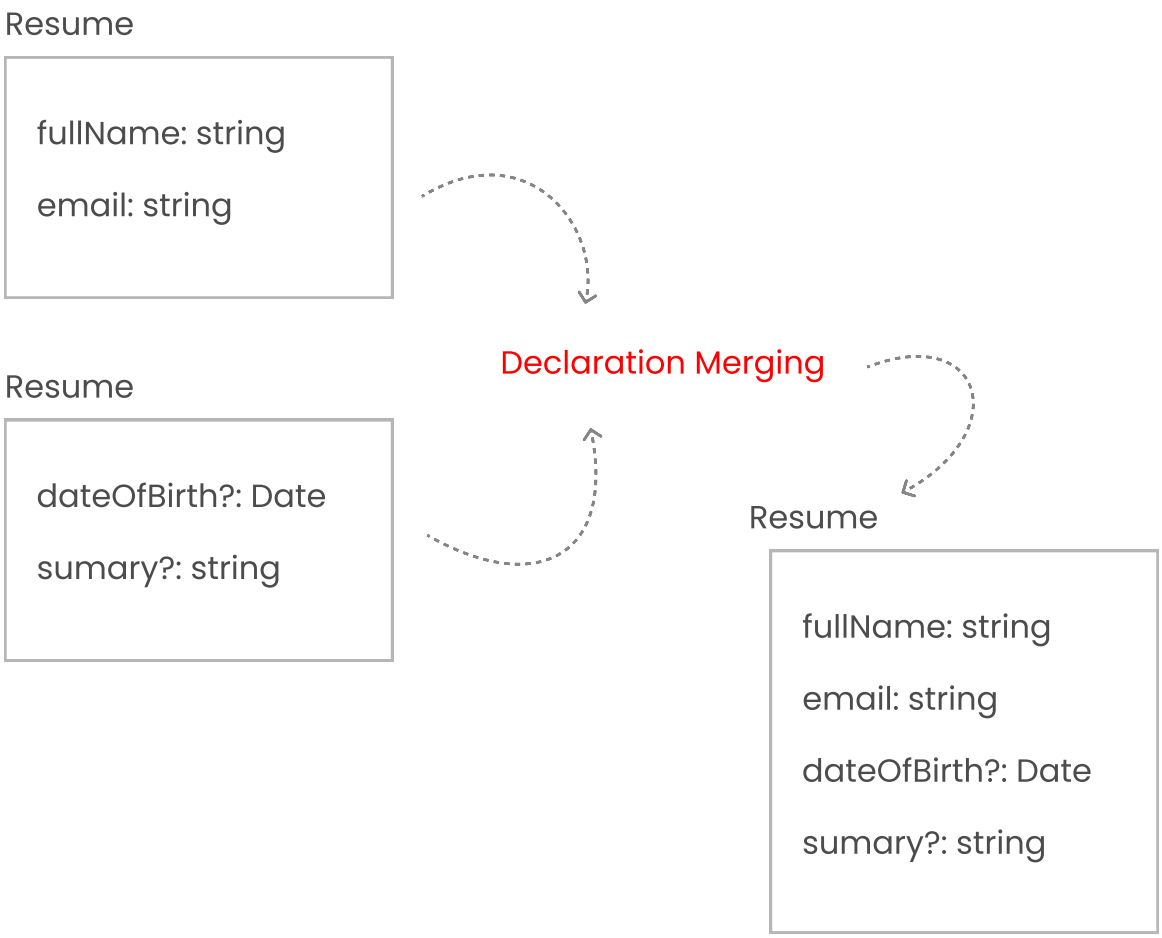
Introdução a Interfaces



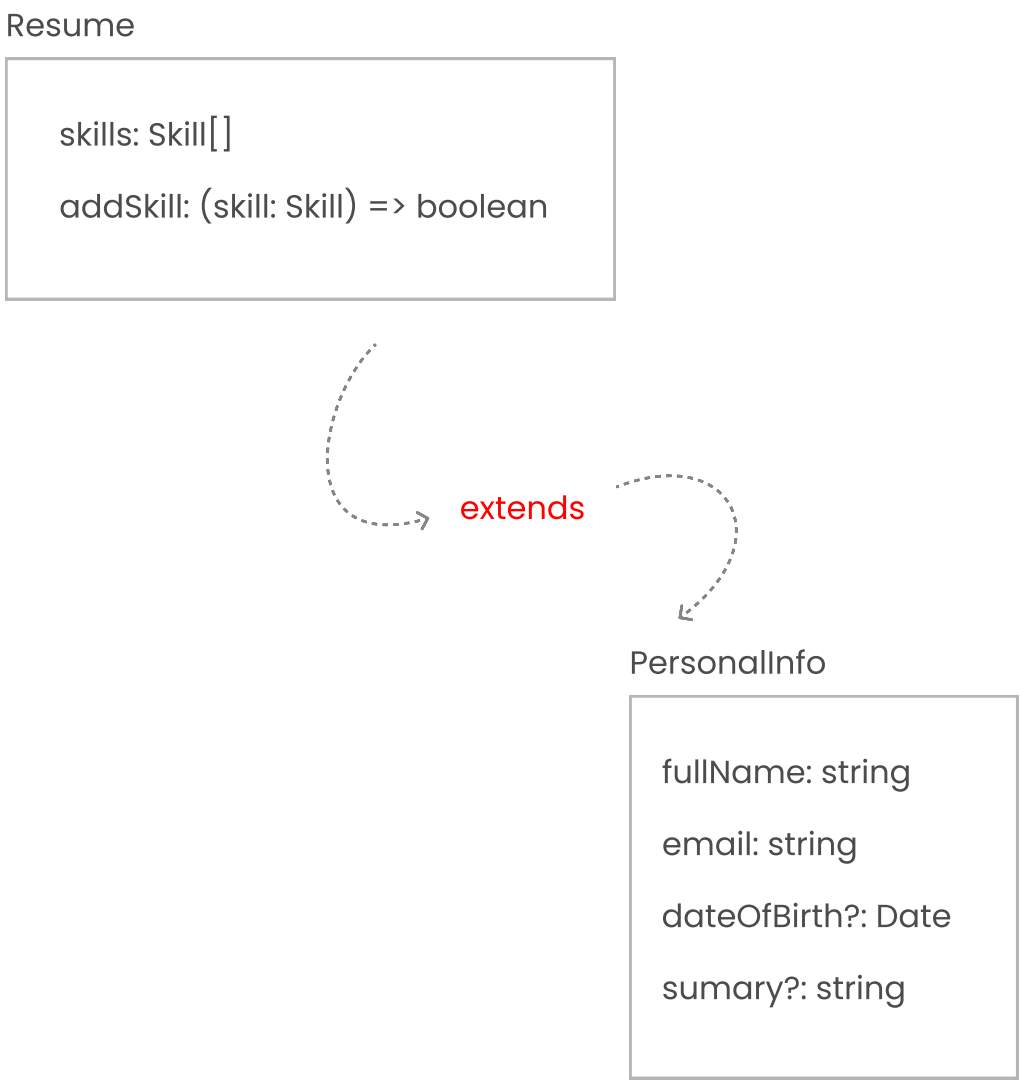
Implementando Interfaces (Implements)



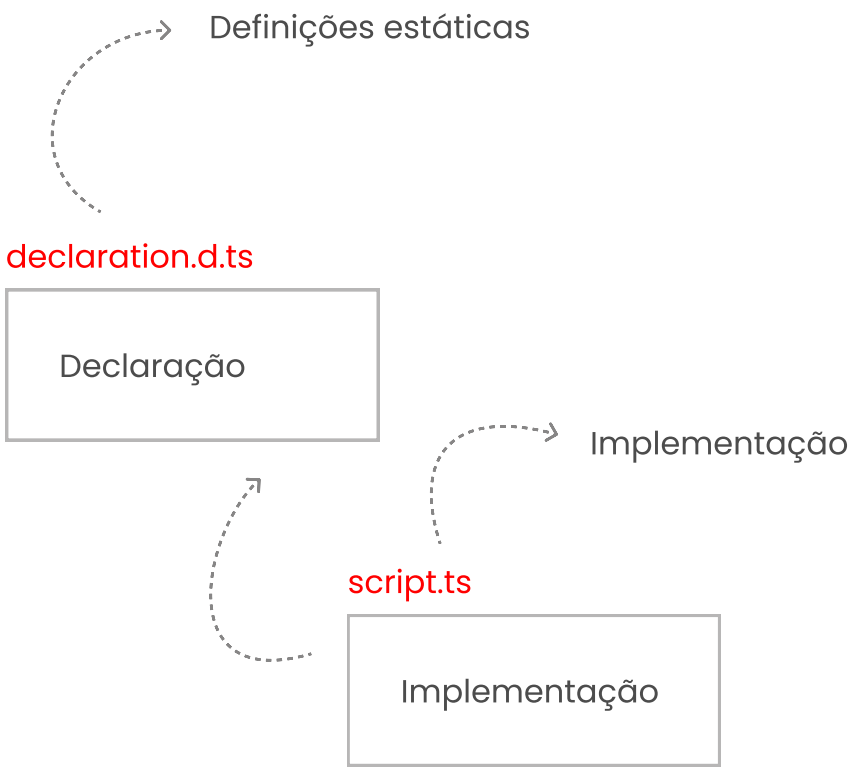
Merge de Múltiplas Declarações



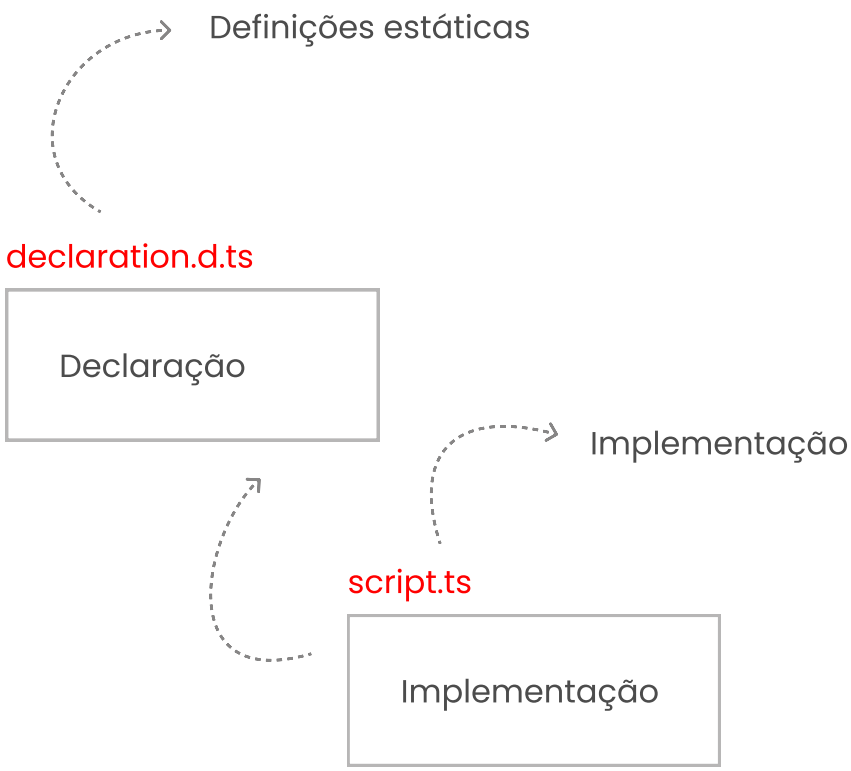
Estendendo Interfaces (Extends)



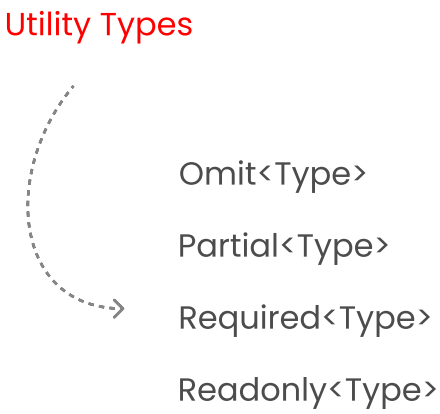
Arquivos de Declaração (Declaration Files)



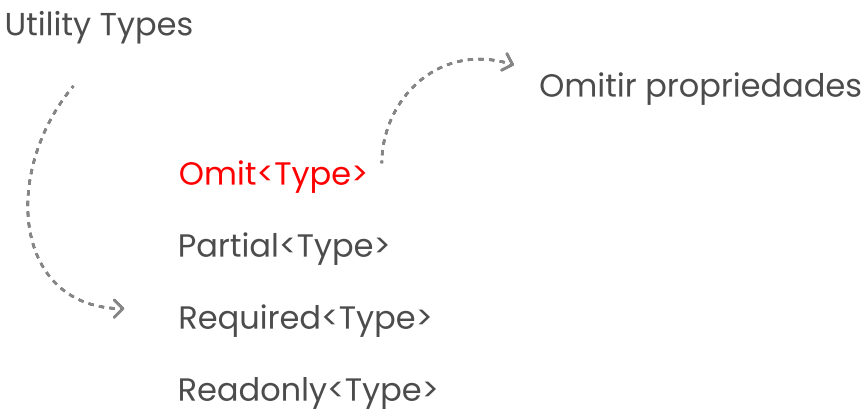
Gerando arquivos de declarações automaticamente



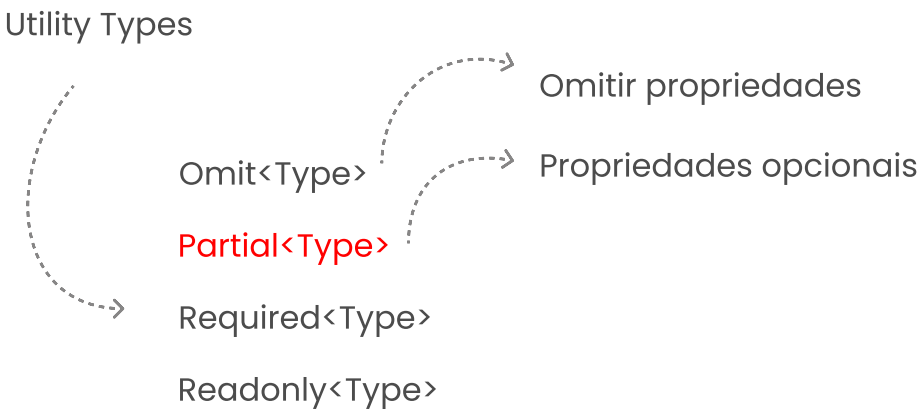
Introdução aos Tipos Utilitários (Utility Types)



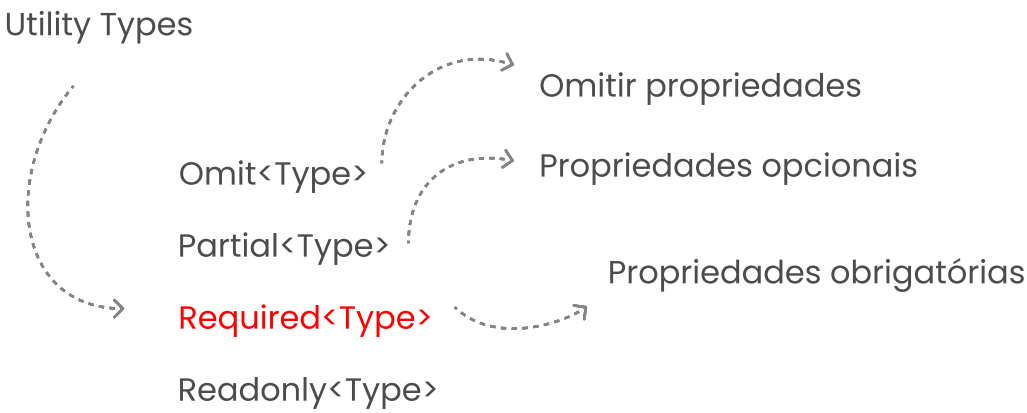
Omit<Type>


Utility Types
Omitir propriedades
Omit<Type>
Partial<Type>
Required<Type>
Readonly<Type>

Partial<Type>



Required<Type>



Type Predicate (is)

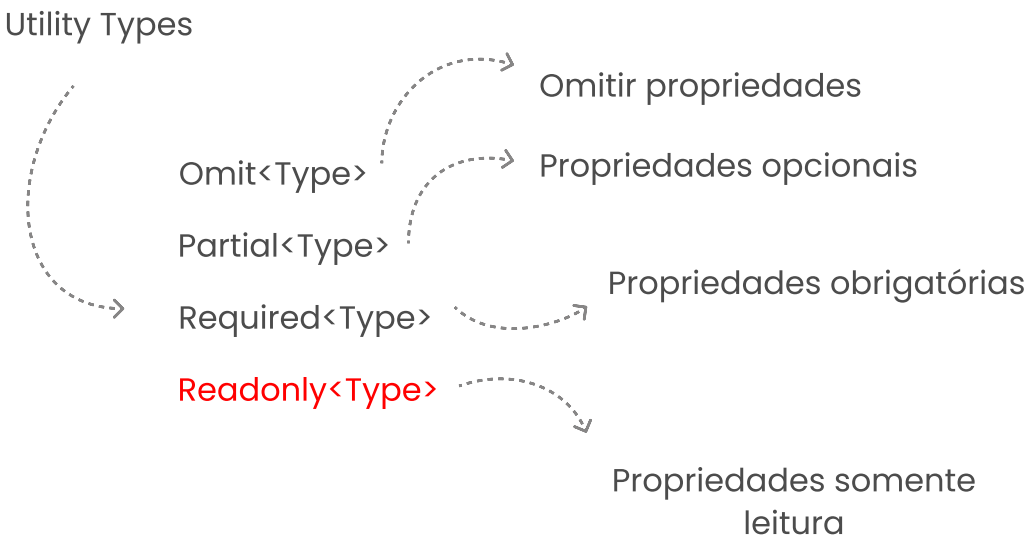
Type Predicate

Função especial que indica ao compilador do TypeScript que um valor é de um tipo específico

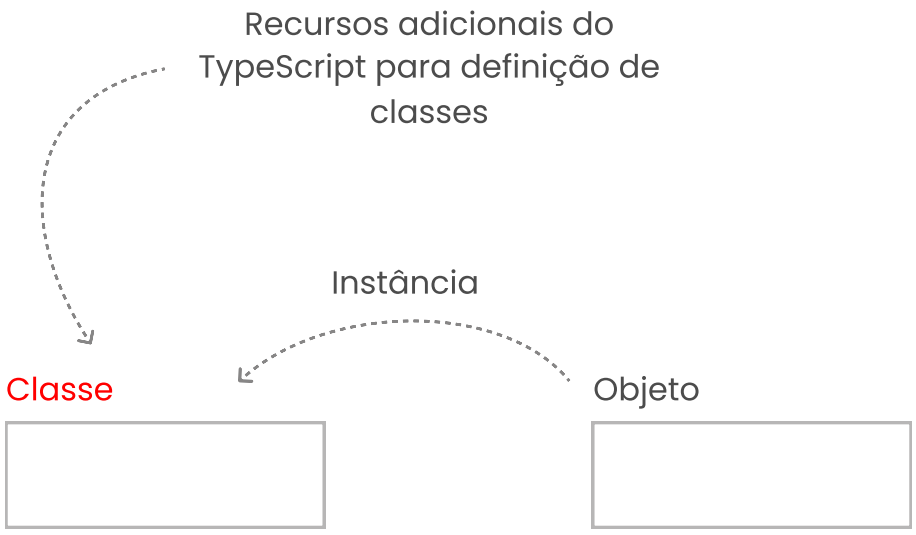
A função verifica o tipo do dado e retorna um valor booleano

```
function nomeDaFuncao(variavel: unknown): variavel is TipoEsperado {
  // lógica de verificação
}
```

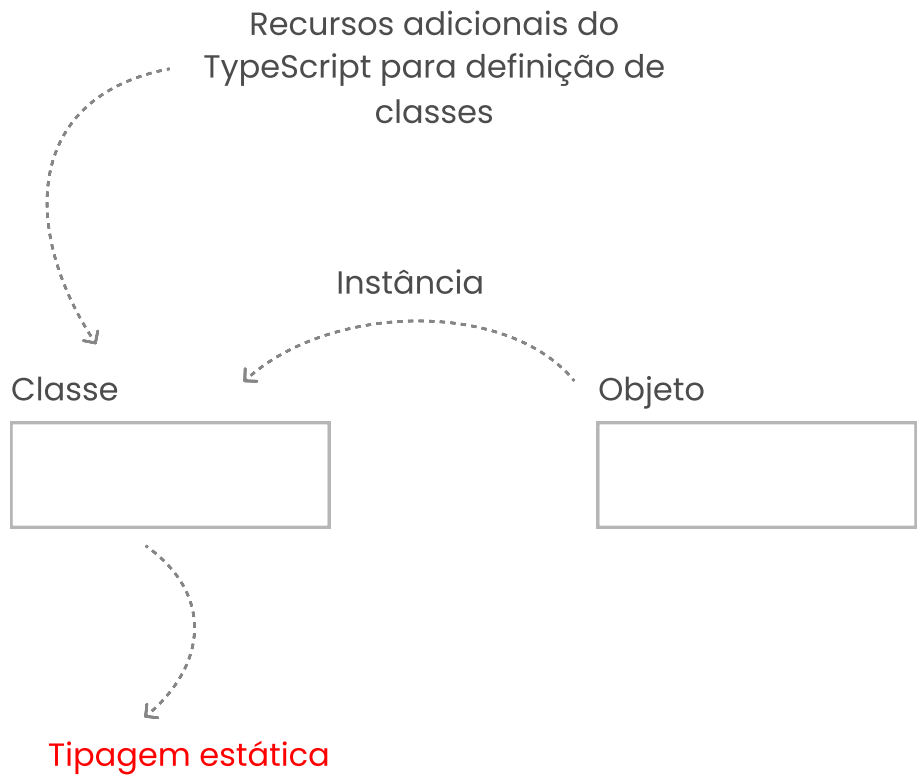
Readonly<Type>



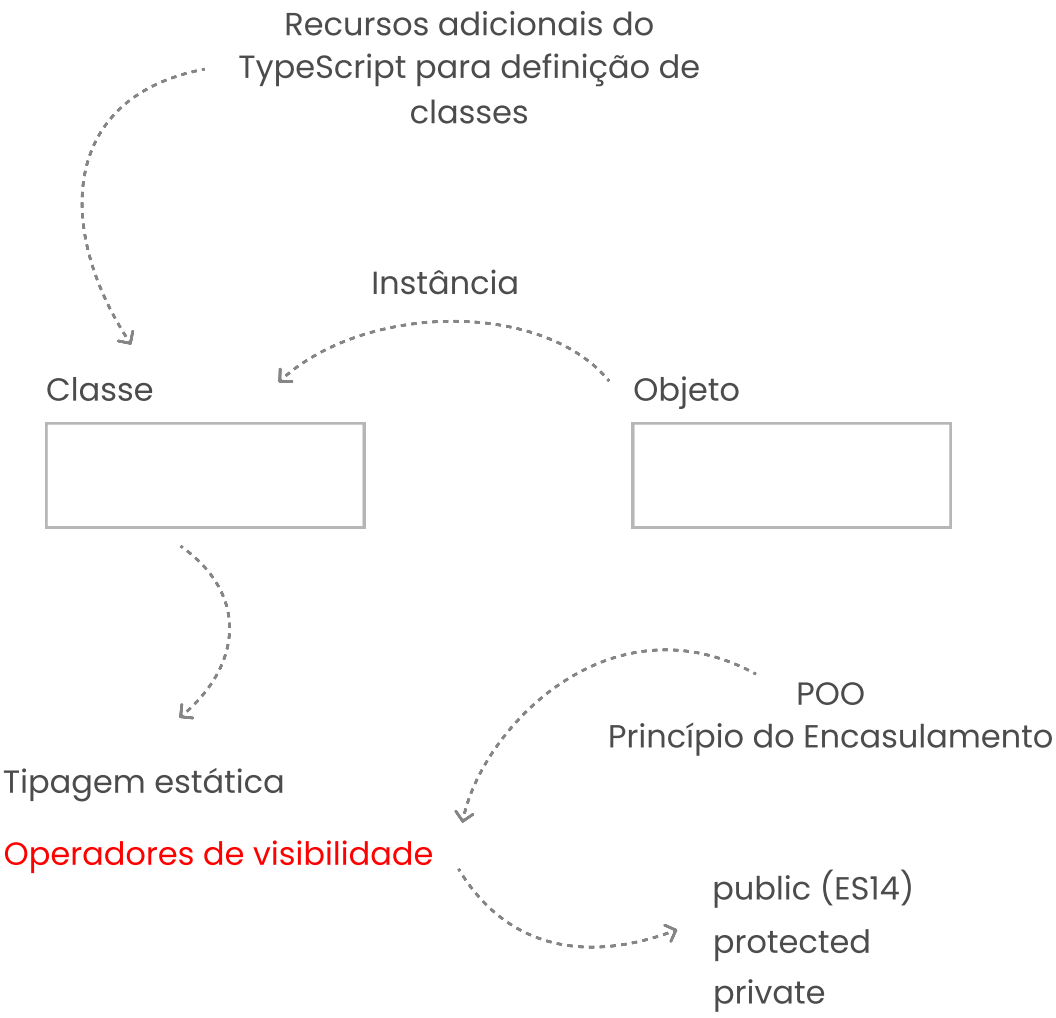
Recursos Adicionais para Classes



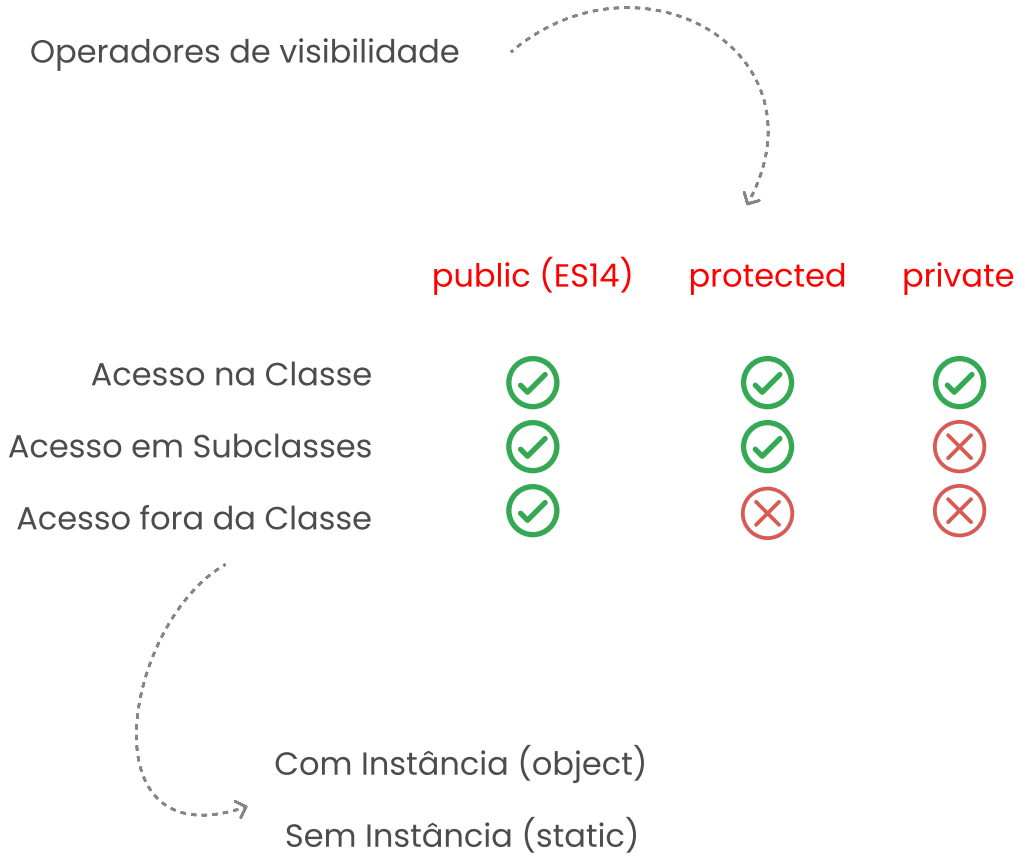
Tipagem Estática



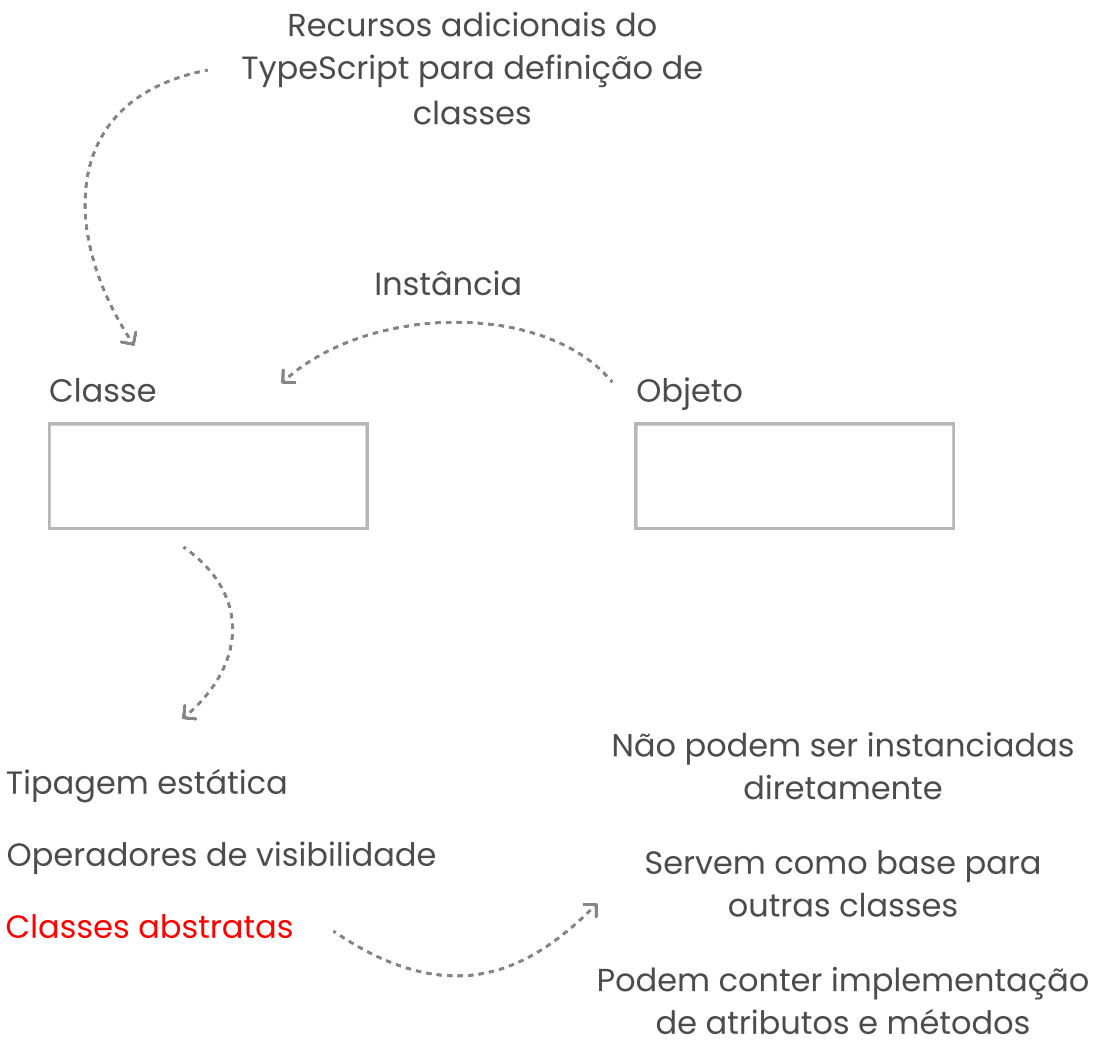
Operadores de Visibilidade



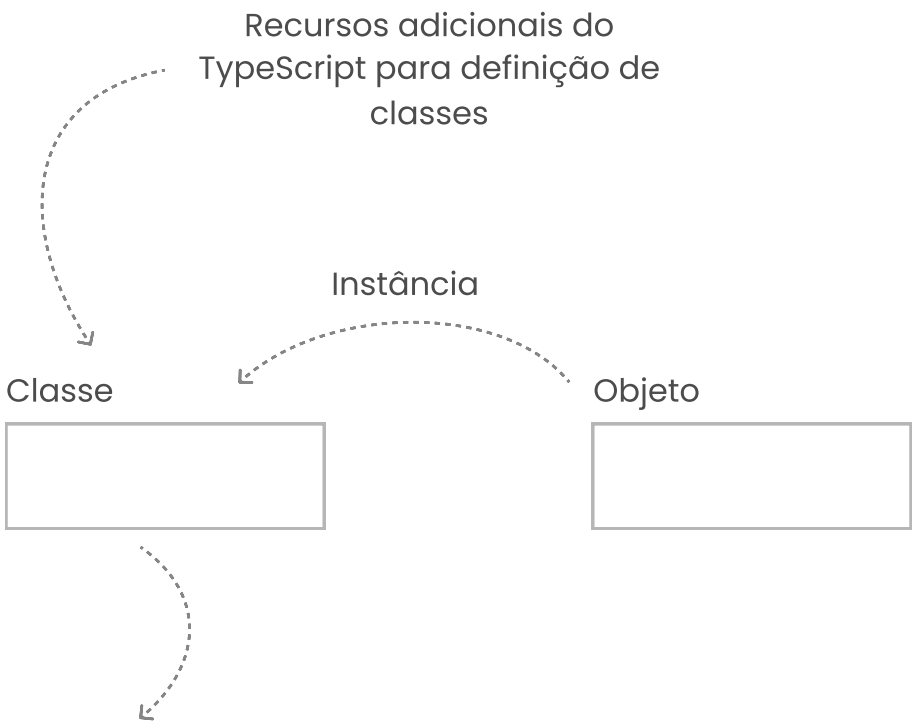
Operadores de Visibilidade Public, Protected e Private



Classes Abstratas



Atributos Opcionais



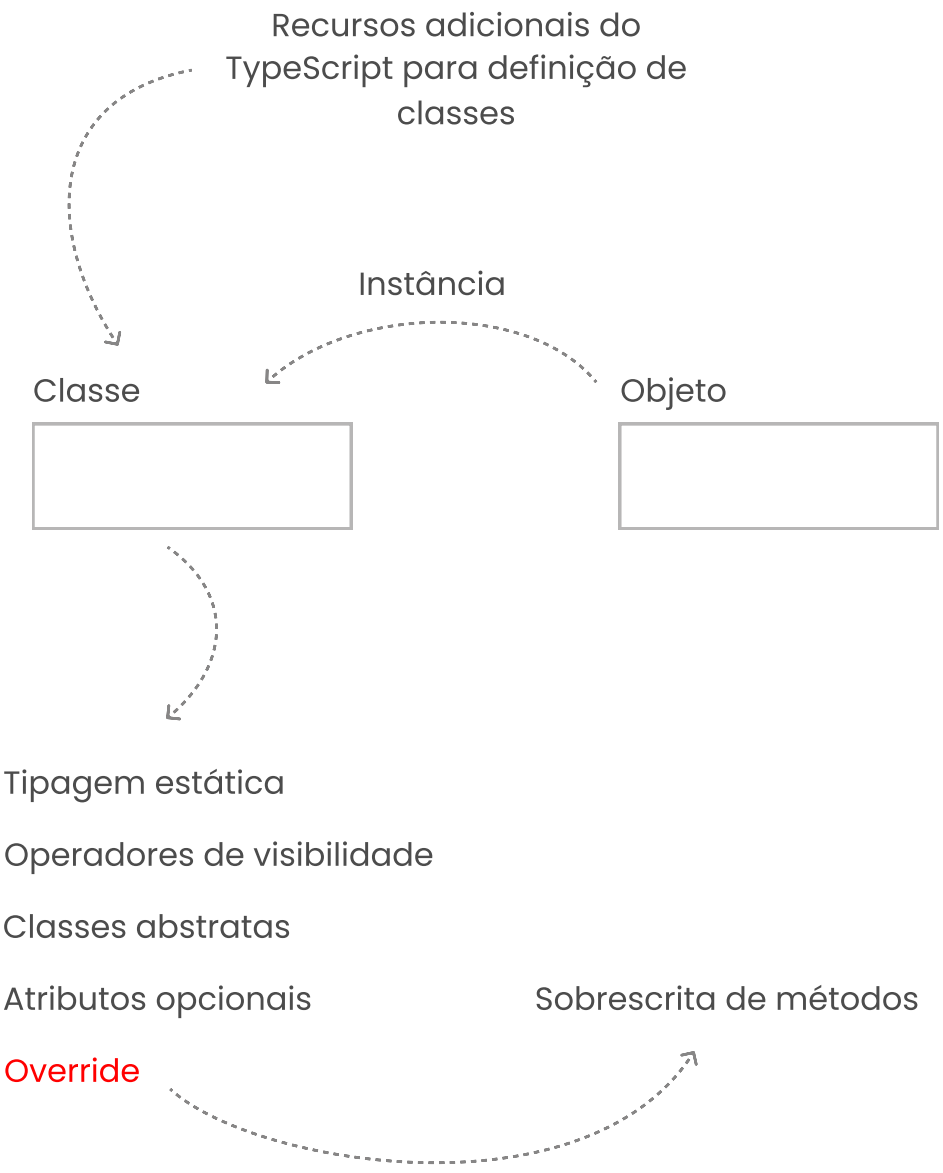
Tipagem estática

Operadores de visibilidade

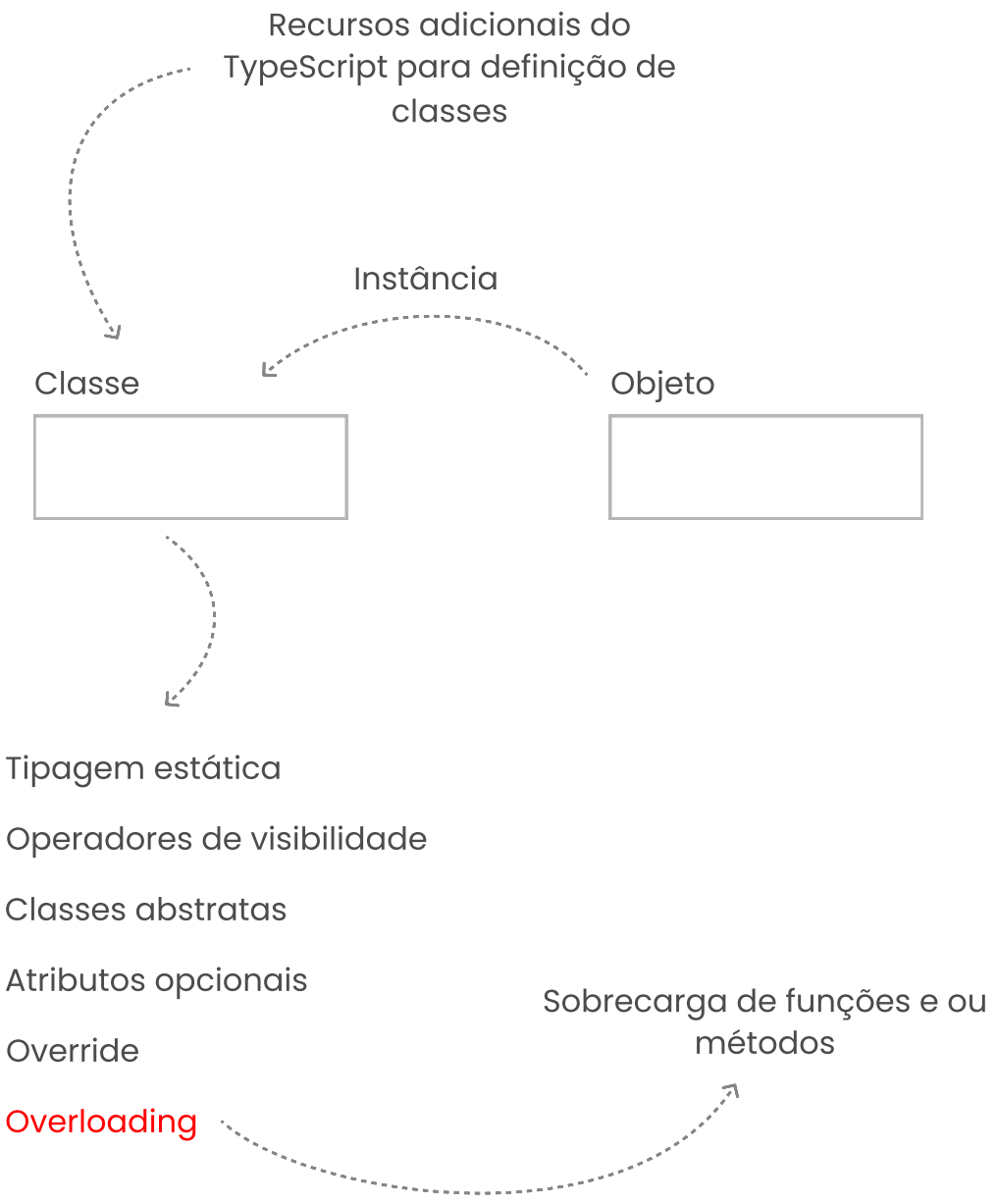
Classes abstratas

Atributos opcionais

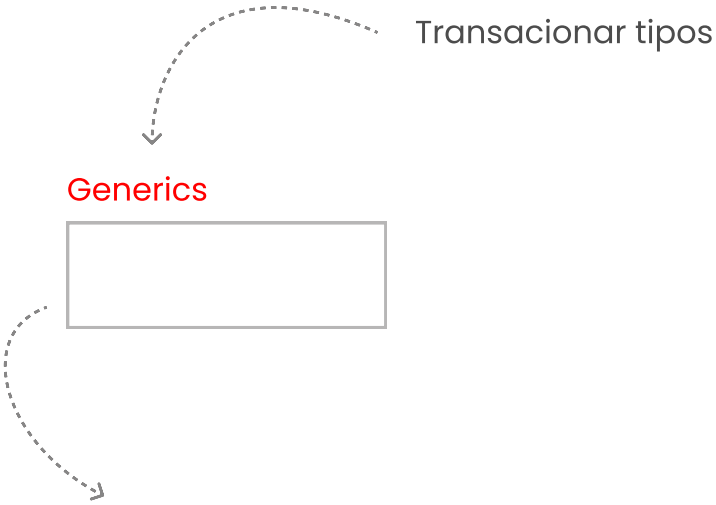
Sobrescrita de Métodos (Override)



Sobrecarga de Métodos (Overloading)

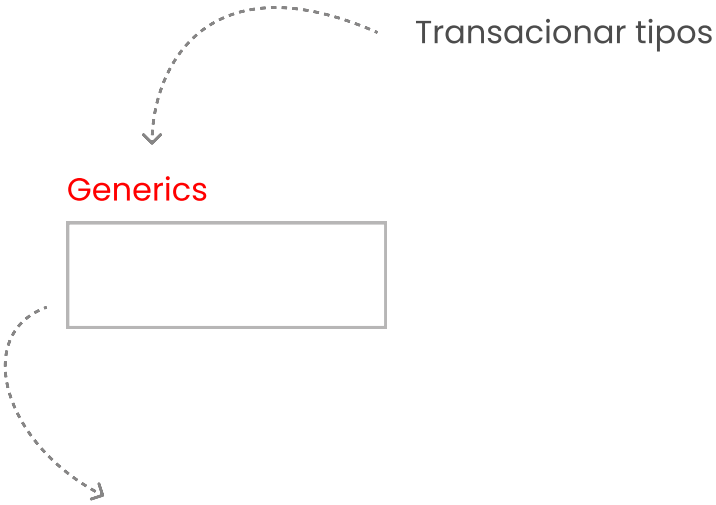


Introdução aos Generics



- Funções
- Métodos
- Classes
- Interfaces
- Type Alias
- Namespaces
- Enums
- Módulos
- Conditional Types
- Utility Types
- Tuples
- Decorators
- Promises

Generics em Funções



Funções

Métodos

Classes

Interfaces

Type Alias

Namespaces

Enums

Módulos

Conditional Types

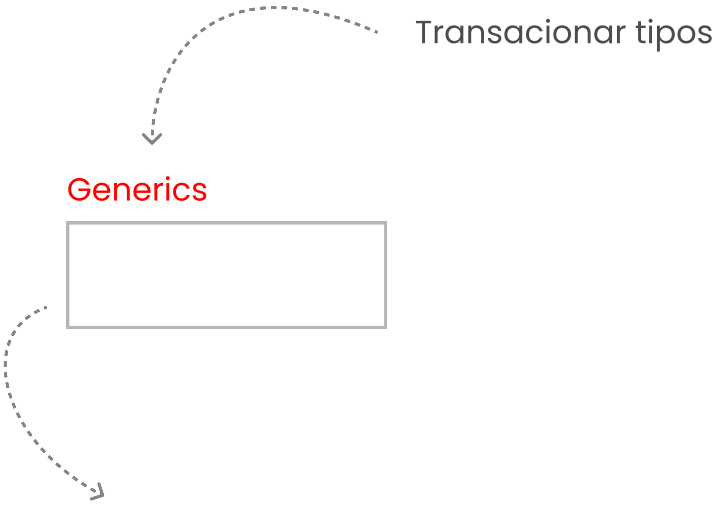
Utility Types

Tuples

Decorators

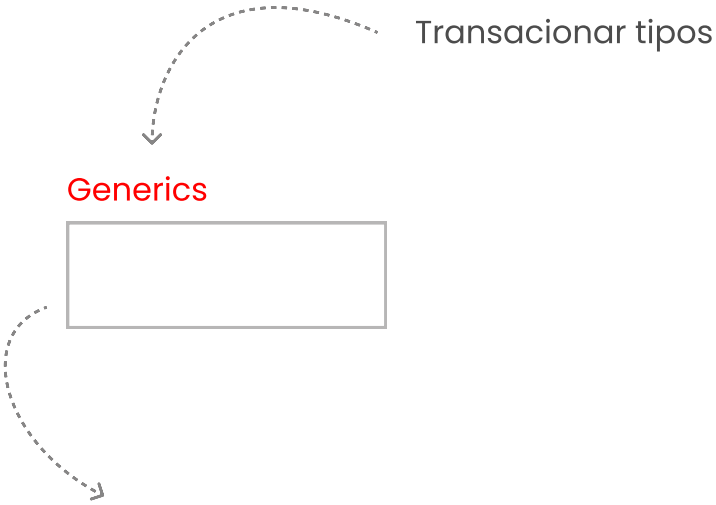
Promises

Generics em Métodos



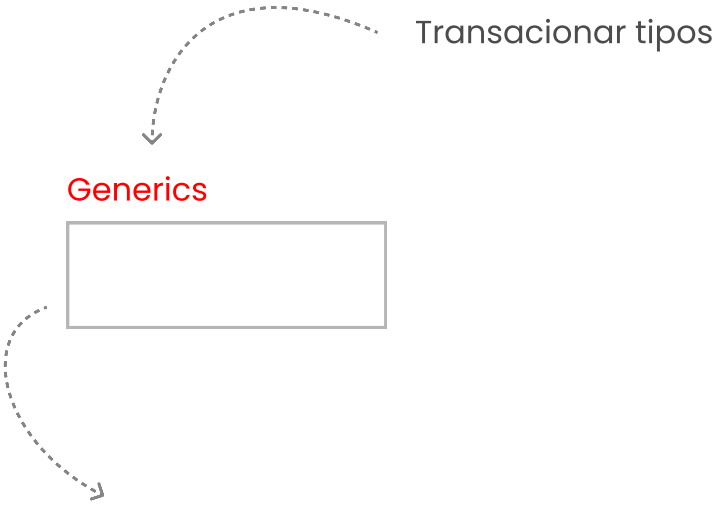
- Funções
- Métodos
- Classes
- Interfaces
- Type Alias
- Namespaces
- Enums
- Módulos
- Conditional Types
- Utility Types
- Tuples
- Decorators
- Promises

Generics em Classes



- Funções
- Métodos
- Classes
- Interfaces
- Type Alias
- Namespaces
- Enums
- Módulos
- Conditional Types
- Utility Types
- Tuples
- Decorators
- Promises

Generics em Type Alias e Interfaces



Funções

Métodos

Classes

Interfaces

Type Alias

Namespaces

Enums

Módulos

Conditional Types

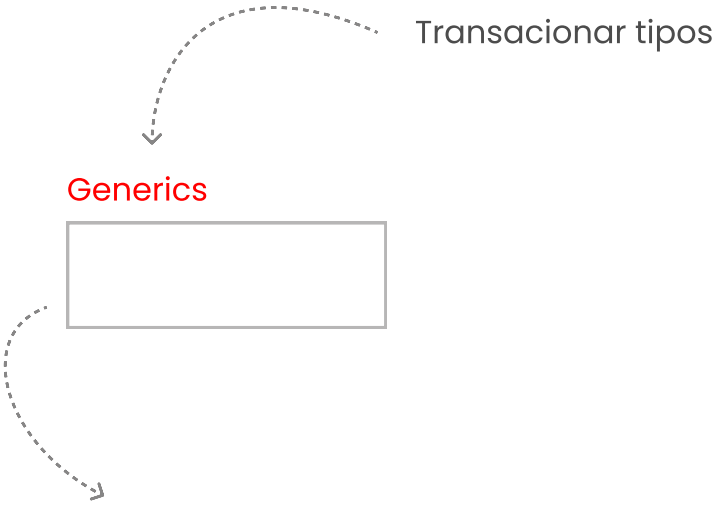
Utility Types

Tuples

Decorators

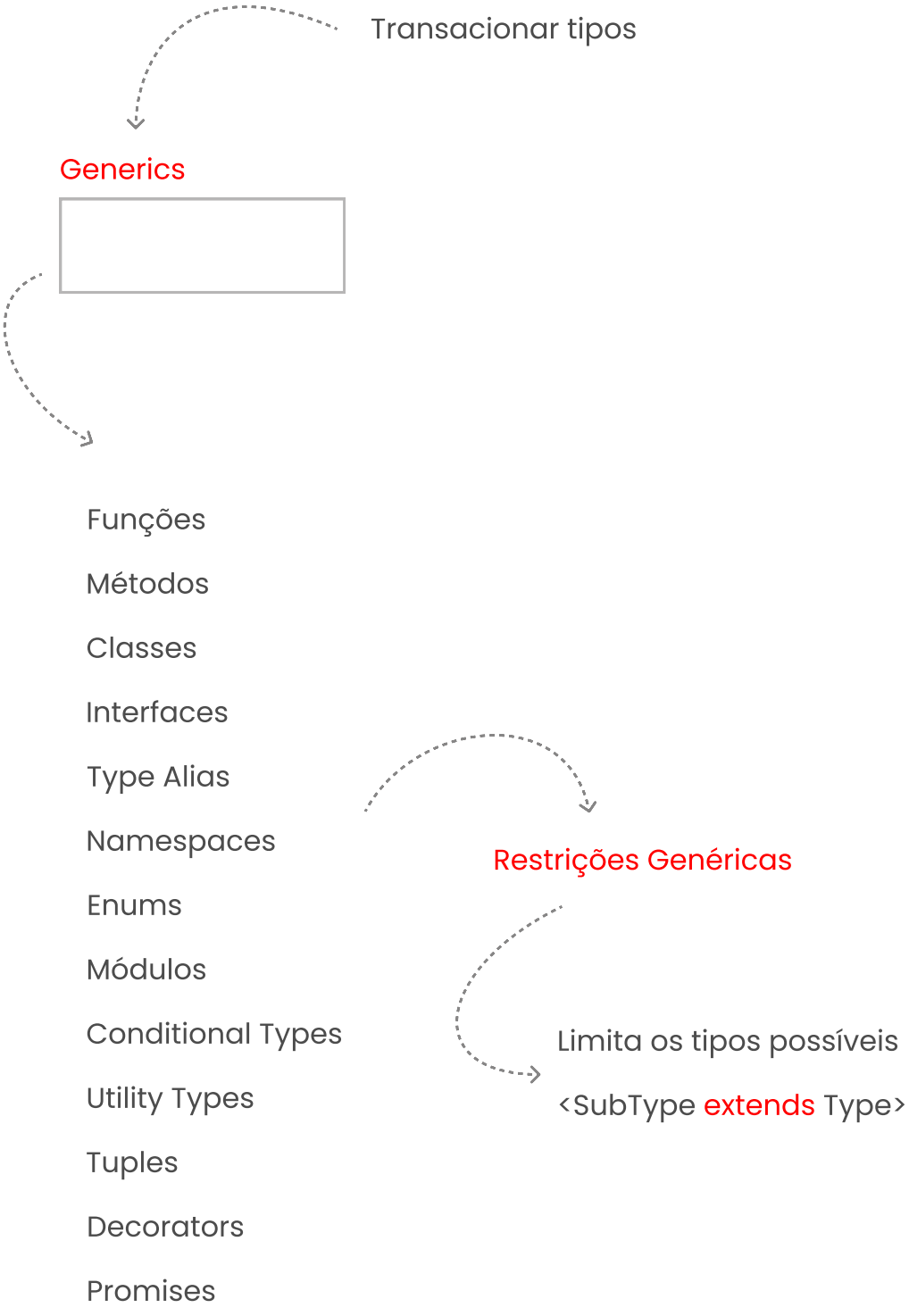
Promises

Generics em Promises

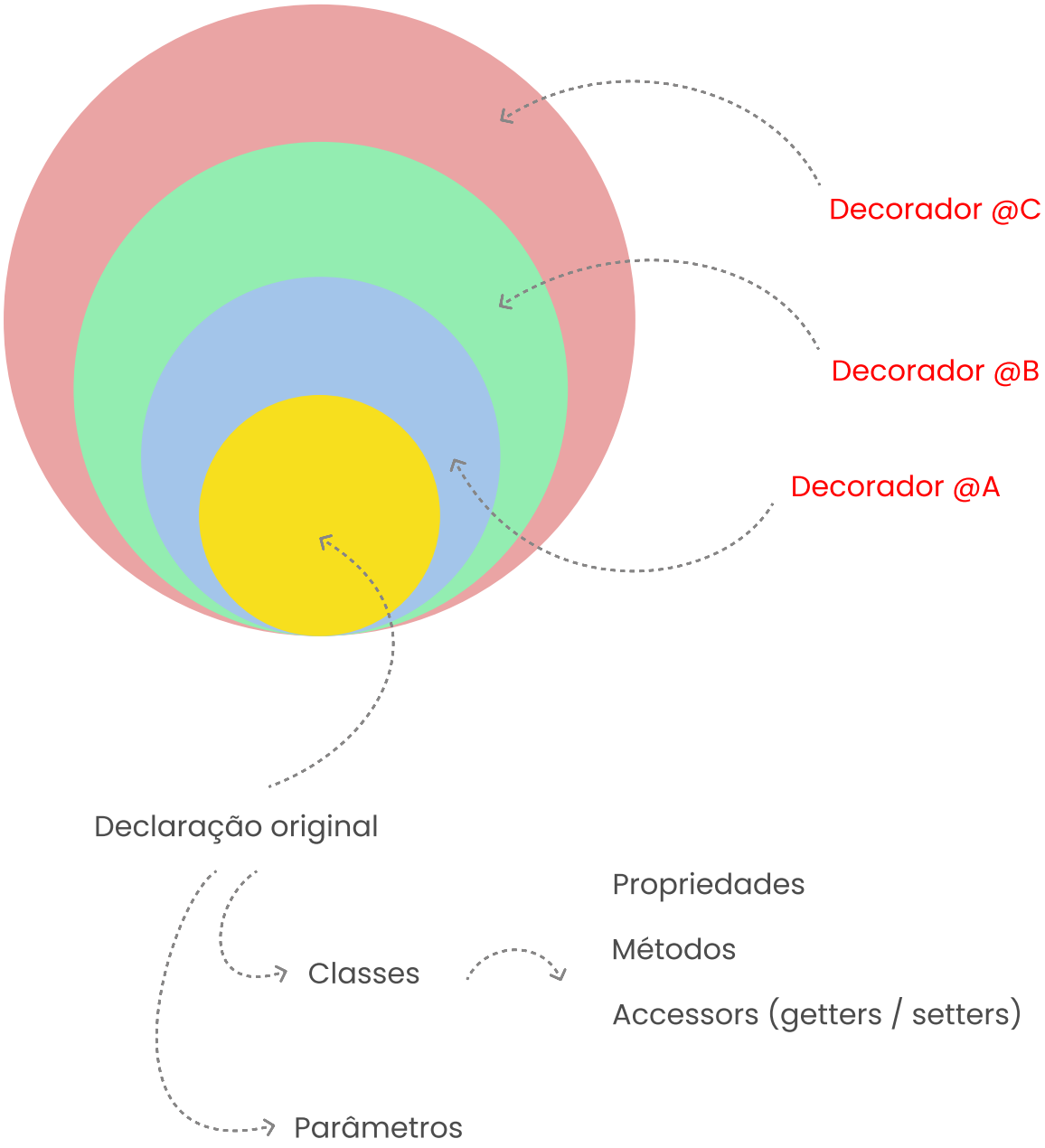


- Funções
- Métodos
- Classes
- Interfaces
- Type Alias
- Namespaces
- Enums
- Módulos
- Conditional Types
- Utility Types
- Tuples
- Decorators
- Promises

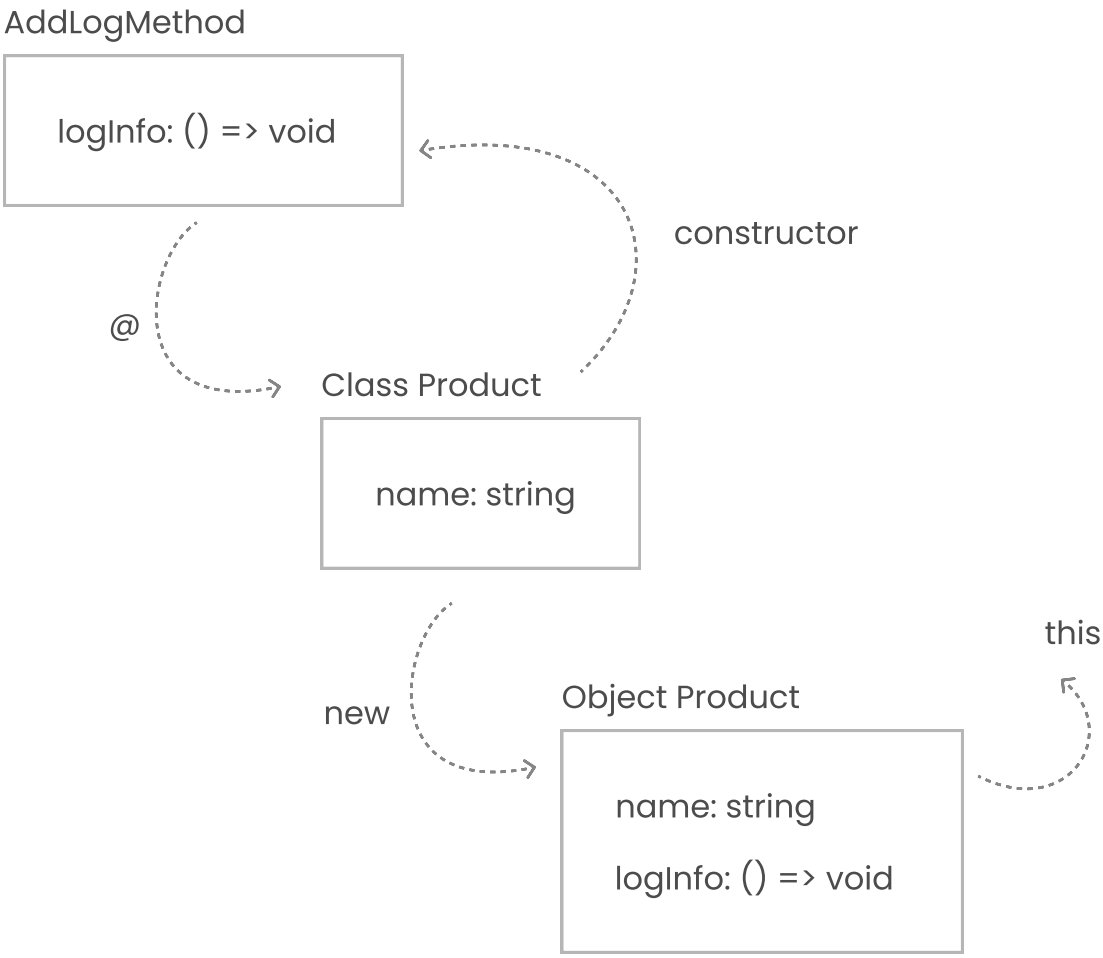
Restrições Genéricas



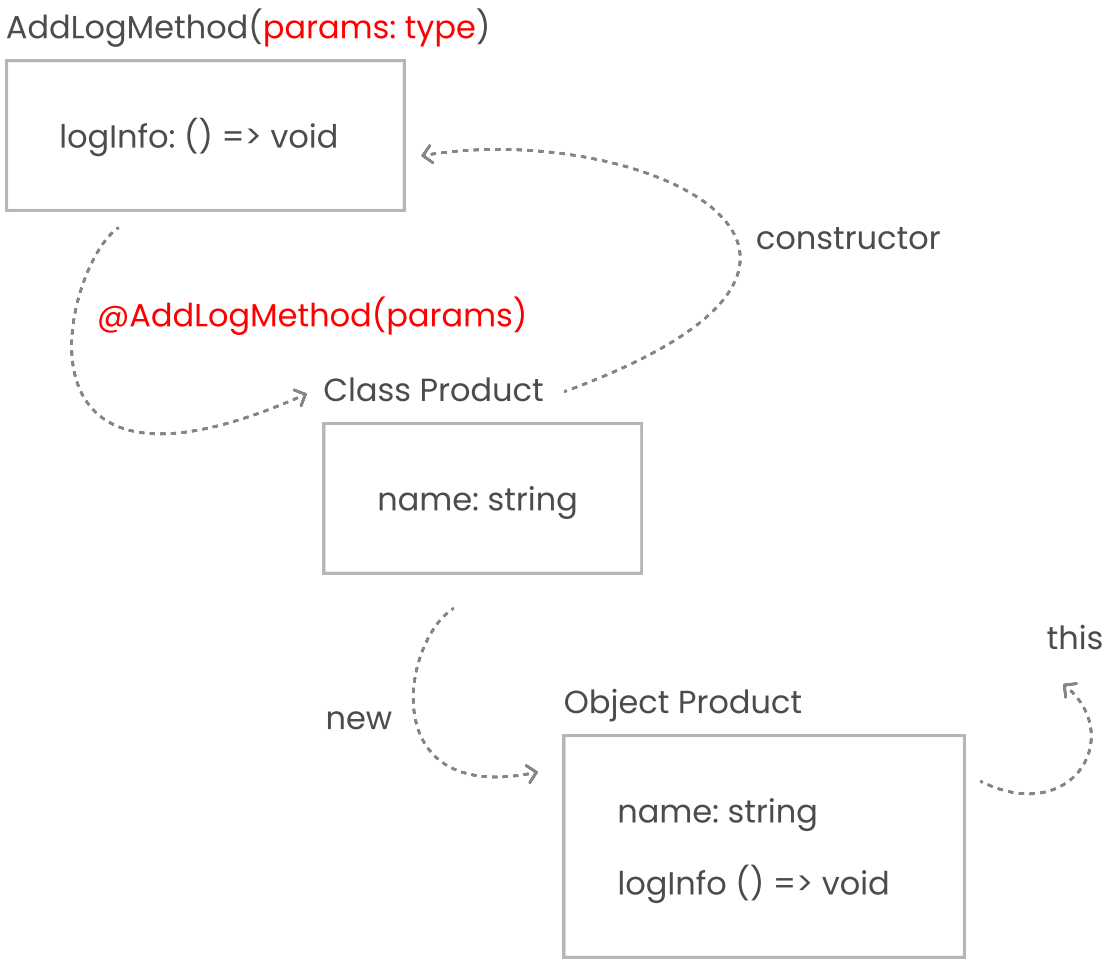
Introdução


Decorador @C
Decorador @B
Decorador @A
Declaração original
Classes
Propriedades
Métodos
Accessors (getters / setters)
Parâmetros

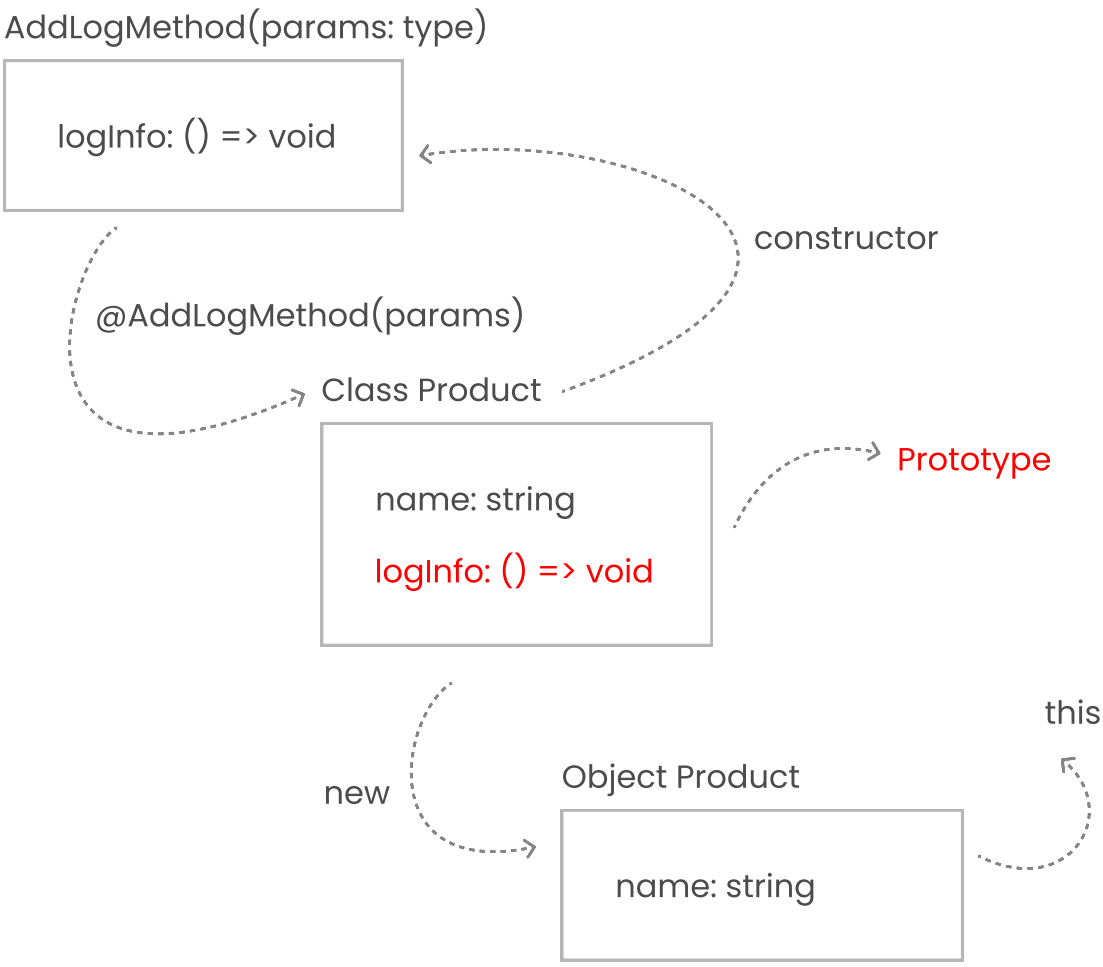
Decorando Classes (Class Decorator) - Parte 1



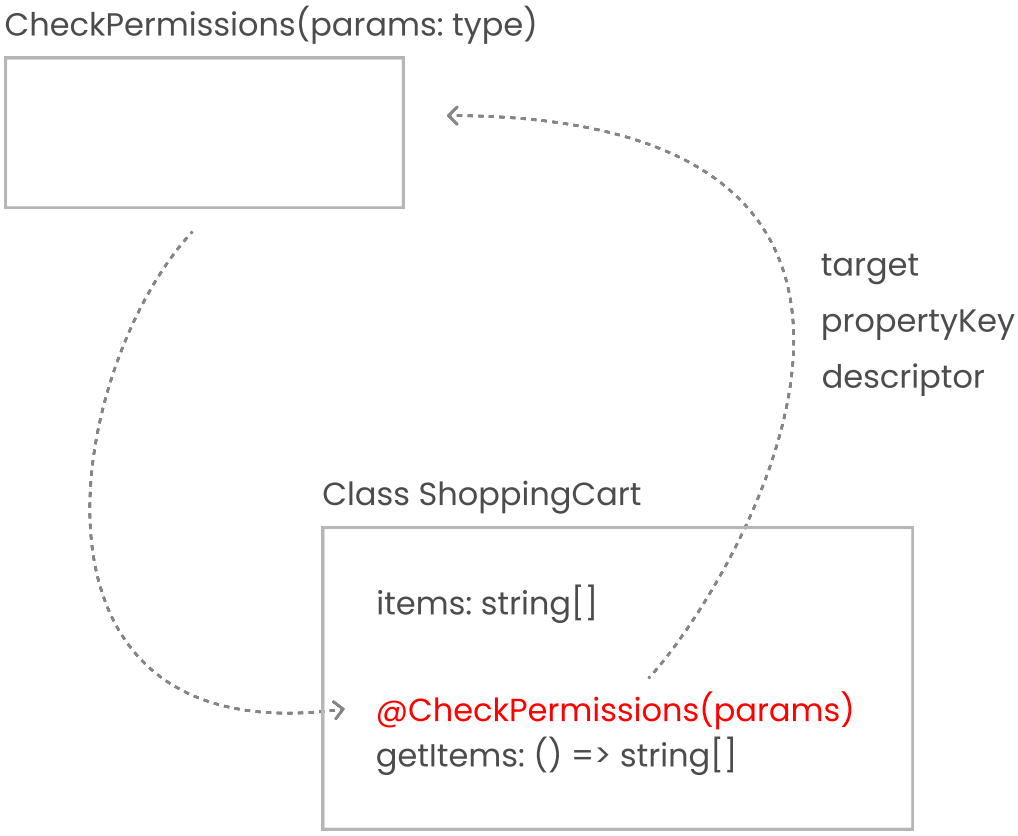
Encaminhando Parâmetros para Decoradores



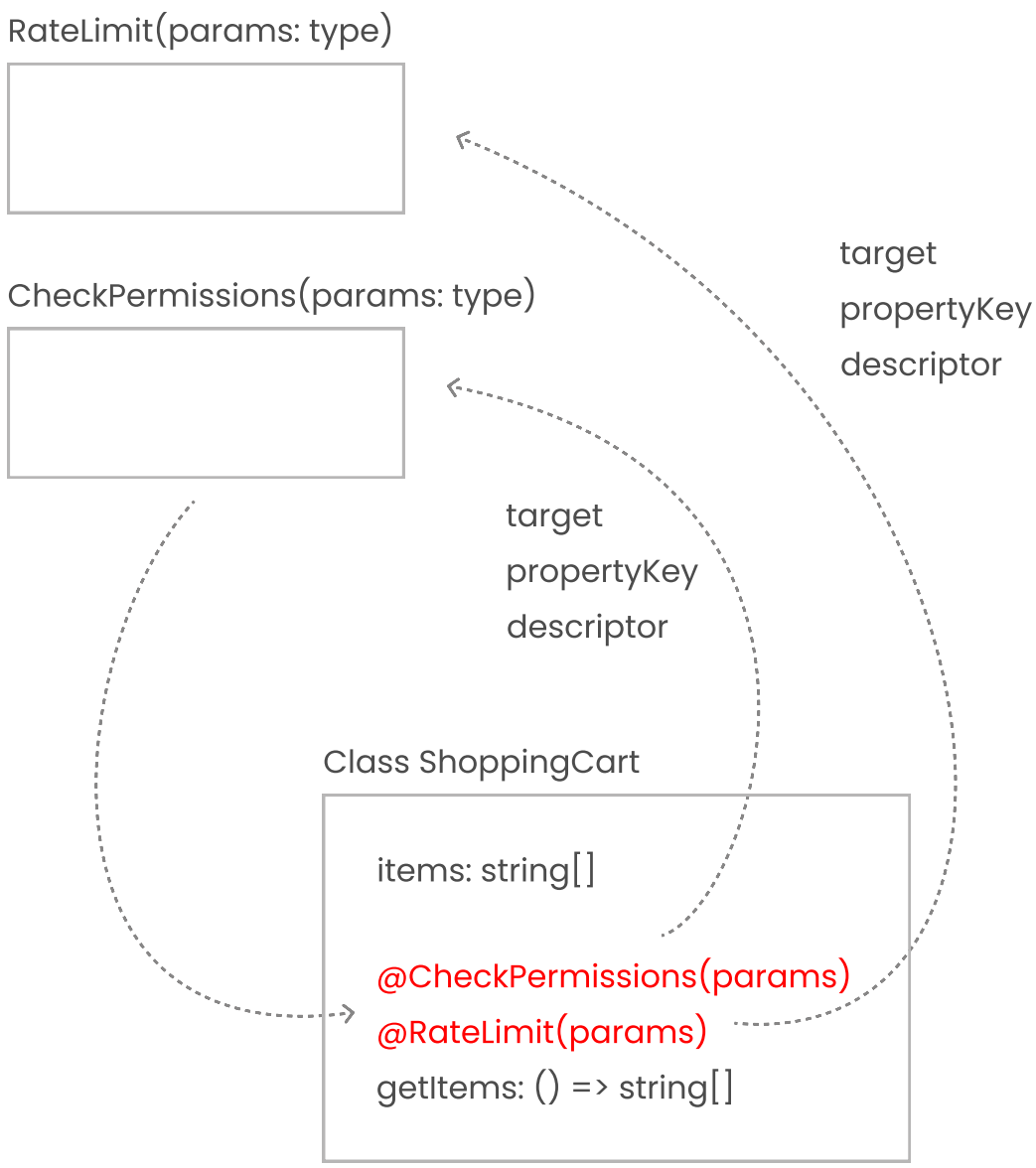
Decorando Classes (Class Decorator) - Parte 2



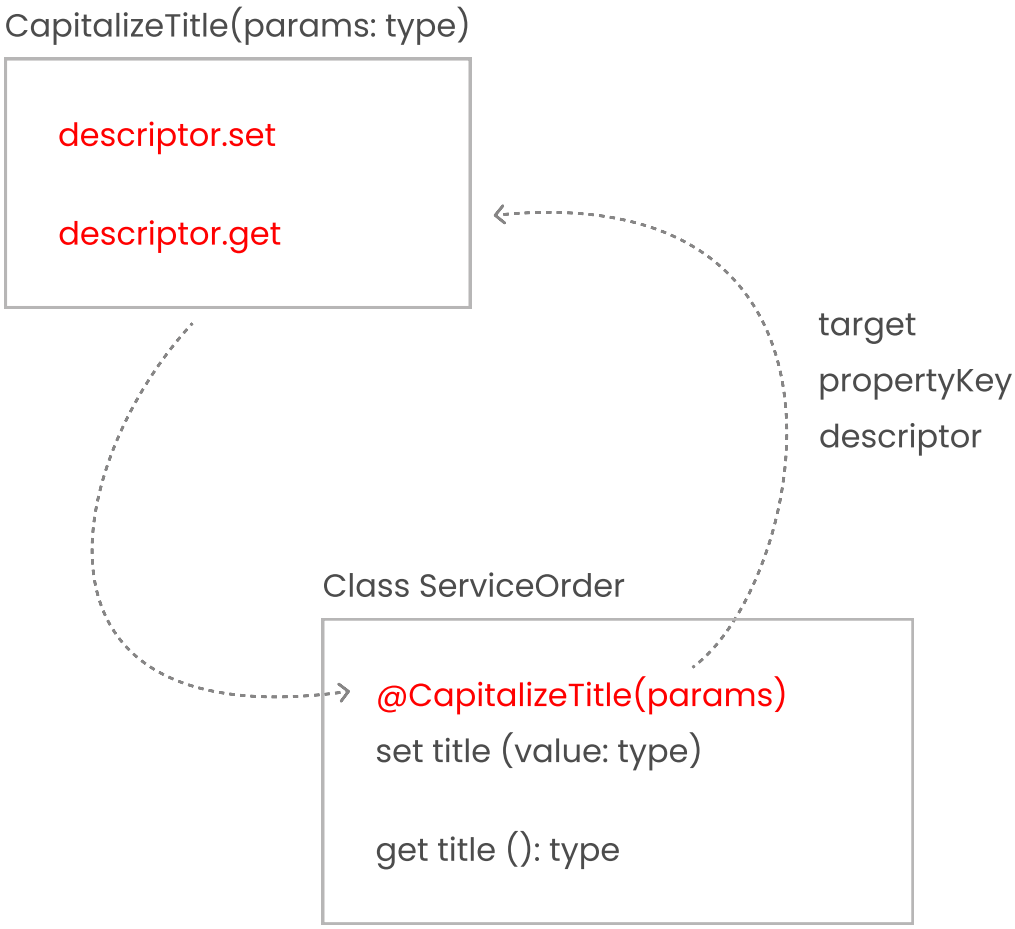
Decorando Métodos (Method Decorator)



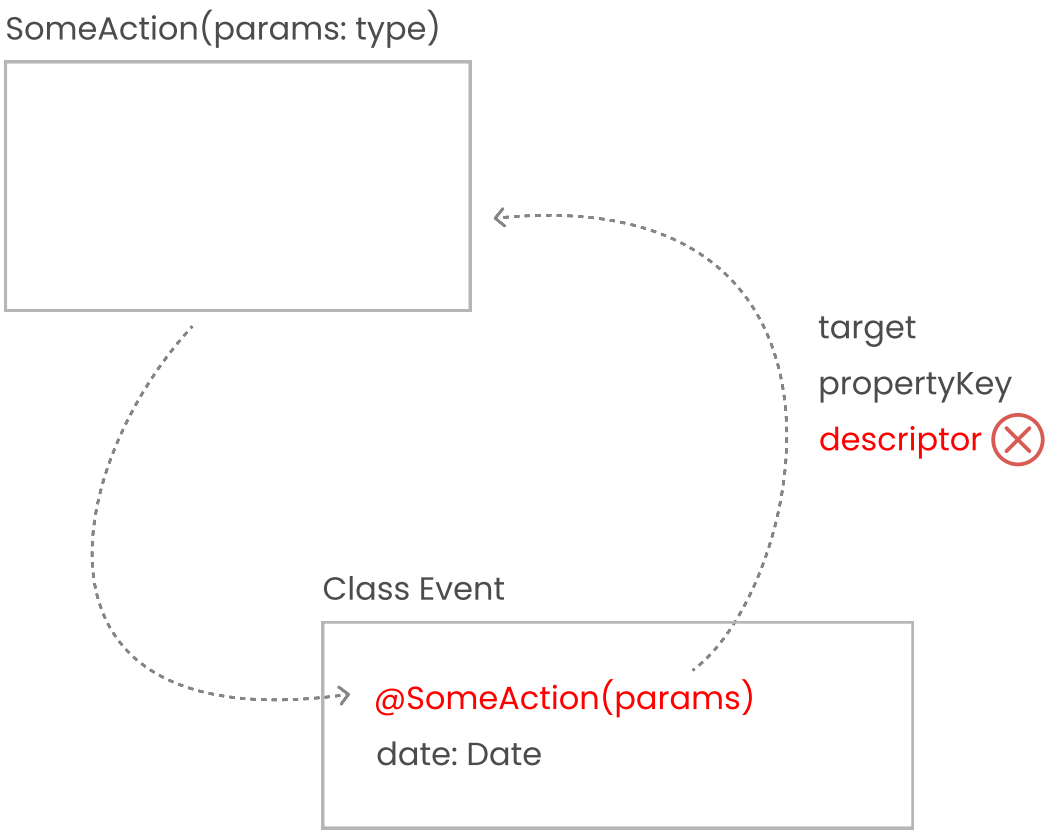
Múltiplos Decoradores



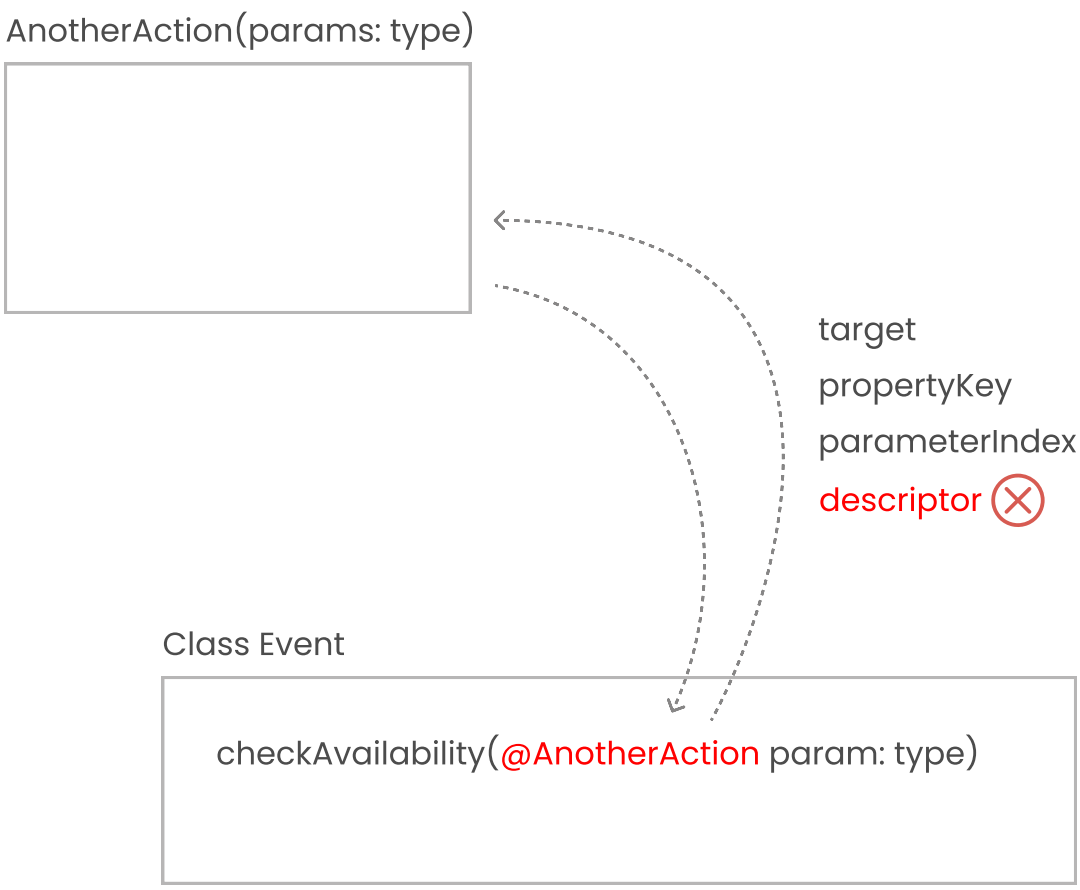
Decorando Métodos Assessores (Assessor Decorator)



Decorando Propriedades (Property Decorator)



Decorando Parâmetros (Parameter Decorator)



ARGUS ACADEMY

www.argus-academy.com