**Curso** ATDEVTS - Programação | TypeScript

Duração: 14h

**Introdução**

Se pretende explorar as grandes vantagens do TypeScript e a forma como estende o JavaScript, este é o curso adequado. Tudo o que é possível criar com JavaScript é igualmente possível com o TypeScript, enquanto "superset", mas com inúmeras outras vantagens, incluindo suporte para bibliotecas JS, NPM, "tipagem" estática e muito mais. Este curso inicia com numa abordagem simples, evoluindo para conceitos mais avançados, como verificação de tipo, iteradores e manipulação de objetos e matrizes.

**Público-alvo**

Programadores que pretendam aumentar a sua eficiência com recurso a uma linguagem
• Seja livre e open source
• Permita gerar código JavaScript, de forma fortemente tipada, com programação orientada a objeto
• Seja um superset do JavaScript
• Mantenha toda compatibilidade que o JavaScript e agregue novas funcionalidades
• Uso de  classes
• Maior facilidade de identificação de erros, por ser "compilada"
• Tipagem das propriedades.
• Declaração automática de propriedades
• Herança entre classes
• Módulos
• Inferir tipos

**No final do curso**

No final deste curso os participantes terão os conhecimentos de que:
• Typescript é uma linguagem que permite facilitar o desenvolvimento de aplicações com JavaScript.
• Também acrescenta conceitos comuns como classes, módulos, interfaces, generics e tipagem estática para JavaScript.
• Trata-se de uma linguagem compilada, não é interpretada em tempo de execução. O compilador durante o processo de "build" do projeto converte os ficheiros TypeScript (com a extensão ".ts") em ficheiros JavaScript (com extensão ".js").
• Como usar o TypeScript para acelerar o desenvolvimento de projetos com JavaScript

**Pré-requisitos**

Para o melhor aproveitamento deste curso os participantes devem conhecer com algum detalhe como funciona a linguagem JavaScript e o DOM.

**Exames de certificação**

(não existem exames)

**Conteúdo programático em detalhe**

**Introdução**

• O que é TypeScript?
• Conceitos e arquitetura do TypeScript
• Porque usar TypeScript?

**Variáveis**

• Declaração de variáveis
• Declarar tipos indefinidos
• Variáveis “hoisting”
• O espectro do TypeScript é o mesmo que o do JavaScript
• Alterar espectro
• Os vários métodos de declaração de uma string
• Modelos com o tipo string
• O que é um número no TypeScript?
• Boolean, funções e objetos
• Evitar o uso de “any”
• Matrizes mutáveis e imutáveis
• “Undefined” Versus “null”
• Não devolver nada com o “Void”
• O tipo primitivo “never”
• “Unknown”: Uma forma interessante
• Tipos literais para como “primitivas”
• Símbolo e símbolo único
• “Casting” para alterar um tipo

**Comentários**

• Os comentários do TypeScript são como os de JavaScript com ligeiras exceções

**Enum**

• Enum com e sem valores
• Aceder a valores de um Enum
• Performance do Enum
• Expandir as funcionalidades de um Enum

**Tipos Genéricos**

• “Generic”
• Genérico e classes
• Genérico e “Constraints”
• Genérico com funções de construção
• Classe Externa Genérica
• Comparação Genérica
• Inferência Genérica
• Padrão Genérico
• Genérico e “keyof”

**Funções**

• Definição
• Funções nomeadas e anônimas
• Variáveis de função e inferência

- Tipo de retorno genérico, parâmetro opcional e valor padrão
- Funções em classes
- Relação de função e o “this”
- Função e tipos de retorno de inferência
- “Overload” de funções para expandir a definição base
- O tipo “string” e “overloads” possíveis

**Tipos mapeados**

- Definição e usos
- Dados imutáveis: apenas leitura
- Parcial
- Nullable
- Pick
- Omit
- Record
- Extract
- Exclude
- ReturnType
- Tipos Personalizados

**Objetos**

- Introdução aos muitos objetos do TypeScript
- O objeto Curly Braces {}
- Novo Objeto
- Objeto com nome em minúsculas vs nome em maiúsculas

**Assinatura de um índice**

- Definições e usos
- Índices de “string” ou números
- Membros do mesmo tipo
- Chaves com constantes e símbolos

**Variáveis “avançadas”**

- Cruzamento com Tipos, Interfaces e Genéricos
- Tipo Literal, Narrowing e Const
- União com Tipos e União Tagged
- Asserção Const para valores literais
- Tuplos para matrizes de tipo e comprimento
- Casting para alterar tipos
- keyof para validar o nome de um membro
- Como é que o TypeScript trata a variação
- Como restringir um tipo com o operador “in”
- O que é um tipo condicional?
- Inferência de TypeScript
- Conjunto e Dicionário

**Exceções**

- Criação de uma exceção
- Captura de exceções síncronas

• Captura de exceções assíncronas
• Funções de asserção

**Alias**

• Aliases com o comportamento estrutural do TypeScript
• Aliases com tipo
• Aliases com tipos genéricos e recursividade
• As diferenças entre aliases de tipo e interfaces
• “Branded” Aliases

**Comparação de tipos**

• Comparar Variáveis
• Verificação de tipo com typeof
• Verificação de tipo com instanceof
• Verificação de tipo e interface com um discriminador
• Verificação de tipo com interseções
• Verificação de tipo numa interface com proteção de tipo personalizada
• Encadeamento opcional e acesso ao elemento opcional
• Nullish Coalescing
• Funções de asserção

**Iteradores**

• Iteração as chaves de um objeto com For-In
• Iteração num objeto com o padrão for / while
• Iteração num Loop Assíncrono

**Manipular Objetos e Matrizes**

• Definir um Array
• Ignorar valores de uma matriz
• Destruturar um Array
• Destruir um objeto
• O operador de propagação e matrizes
• O operador de propagação e objetos
• O operador Bang

**Partilha de código**

• Namespace
• Módulo
• Módulo padrão
• Módulo “lazy loading”
• Importar atalhos
• Ficheiros de definição e ficheiros de definição global
• Localização de ficheiros de definição