
A cluster of hexagonal icons in various shades of blue and cyan. The icons include a smartphone, a magnifying glass, a lightbulb, and a thumbs-up gesture.

Introdução ao *Typescript*

A cluster of hexagonal icons in various shades of blue and cyan. The icons include a speech bubble, a gear, a network diagram, and a large empty hexagon.

Decorative hexagonal icons surrounding the title, including a lightbulb, a thumbs up, a smartphone, a magnifying glass, a speech bubble, a gear, and a network diagram.



Material preparado

- Tipos implícitos e seus conceitos.

- Template dinâmico / Manipulação declarativa do DOM através de template.

- Desconstrução de Objetos/Arrays.

- Classes e seus conceitos + benefícios.

- Reuso de código através de herança.

- Interfaces, classes abstratas, generics.

- Um pouco de NodeJS.

- Express Framework.

- tsconfig.json e suas configurações.

- ES2015 -> arrow function, promises, async/await, import/export, autoload class, spread operator, escopo de função, valor padrão em variáveis.

- Criando seu primeiro servidor com Typescript.

- Criando sua própria REST com Typescript.

- Bônus -> módulos extras para seguir programando em Typescript.



Conceito

- ◇ O TypeScript é um superset da ES2015, ou seja, além de suportar os recursos da linguagem desta versão, adiciona outros. Um exemplo é o suporte ao modificador `private`, `protected`, `abstract`, `interface`.
- ◇ <https://www.typescriptlang.org/play/index.html>



Tipos Implícitos

Qual a ideia da tipagem de dados?

1

- Evitar erros em tempo de execução
- Checagens internas com relação aos dados trafegados internamente.
- Aumenta a produtividade.
- Suporte mais efetivo por parte de ide's.
- Produzir um código mais robusto.

Tipos de dados

- | | |
|-------------------|----------|
| - let, const | - type |
| - Boolean | - Number |
| - Never | - String |
| - Array | - Object |
| - Enum | - Any |
| - Type assertions | - Void |

Template Dinâmico

Sintax: `texto...`;

A idéia do template dinâmico é simplificar o processo de criação de um elemento DOM dentro do javascript.

2

Diferença entre Template String e Concatenação:

```
let nome = 'Robinson';  
let endereco = 'Testando';  
`<div> Meu nome é ${nome}  
  Meu endereço é ${endereco}  
</div>`;
```

```
"<div> Meu nome é" + nome + "\n" + "Meu  
endereço é" + endereco + "\n </div>";
```

Desconstrução de Objetos/Arrays.

3

funciona tanto para vetores, quanto para objetos. A ideia é que fique mais "claro" na hora de extrair dados de um objeto e/ou um vetor.

Classes

Vantagens

- Maximiza o reaproveitamento de código
- Dividir para conquistar
- Maior manutenibilidade
- Design do código mais arrojado

Desvantagens

- Maior espaço em disco
- Maior esforço (o tamanho do código é muito maior)

4

Herança

5

Com a **herança** é possível criar classes derivadas, subclasses, a partir de classes bases, superclasses. As subclasses são mais especializadas do que as suas superclasses, mais genéricas. A linguagem Typescript permite o uso de **herança** simples, mas não permite a implementação de **herança** múltipla.

Um pouco de nodeJS

6

Non-Blocking IO: O Node.js trabalha com o modelo de IO não-bloqueante, ou seja, nenhuma tarefa pesada de IO vai travar sua aplicação, pois elas serão executadas em background sem bloquear a aplicação e quando elas finalizarem você trata os resultados através dos callbacks das funções. Em uma aplicação de muita leitura de arquivos, o Node.js vai se destacar melhor nisso, pois vai ler os arquivos de forma assíncrona enquanto os demais processos da aplicação continuam rodando. Alguns exemplos práticos que geralmente travam aplicação são: manipulação de arquivos pesados, envio de emails e leitura de base de dados. Com IO não-bloqueante do Node.js essas tarefas são facilmente executadas em background e o retorno de sucesso ou falha dessas tarefas ocorrem através de uma função de callback.

Express Framework

7

O Express é um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel.

tsconfig.json e suas configurações

8

"include": diretório que o typescript usara para pegar os dados.

"outDir": resultado da compilação do typescript, quando é convertido pelo javascript.

"target": linguagem que está sendo usada.

"compileOnSave": somente é compilado quando o projeto está correto.

"buildOnSave": automaticamente já faz o build para js caso o projeto esteja correto.

"noImplicitAny": obriga o desenvolvedor a colocar any.

"removeComments": remove os comentários na hora da compilação para javascript, evitando o resultado ir para o JS

tsconfig.json e suas configurações

9

"preserveConstEnums": mantém os enums usados no typescript para o js.

"exclude": faz o oposto do include, e exclui os itens desejados na hora de compilador.

"typeRoots": declaração para rodar os types.

"strictNullChecks": verifica se alguma função retornará nulo.



Obrigado!

Perguntas?

Informações pessoais:

- ◇ <https://github.com/robinsonLuiz>
- ◇ <https://robinsonluiz.github.io/>
- ◇ robinson.acosta@compasso.com.br
- ◇ <https://www.linkedin.com/in/robinson-luiz-656a55164/>

