Conhecendo Arrow Functions

Uma escola precisa imprimir uma lista de todos os funcionarios e o código da turma, e para realizar essa ação temos a seguinte classe:

class ModuloDeImpressao {

constructor() {

this.\_codigo = 10;

}

imprime(nomes) {

nomes.forEach(function(nome){

console.log(`${this.\_codigo}: ${nome}`);

});

}

}

O problema é que após o teste inicial:

const professores = ['Elias', 'Yuri', 'Gabriel', 'Guilherme', 'Yan'];

const impressao = new ModuloDeImpressao();

impressao.imprime(professores);

Está sendo mostrado no console o seguinte erro:



O que vamos fazer é descobrir o porquê e como resolver esse erro.

**Entendendo o erro**

O JavaScript está nos dizendo que this.\_codigo é undefined, ou seja, não tem nenhum valor atribuído a ele. Isso ocorre porque **o this dentro de uma função tem um comportamento dinâmico**, isto é, variando de acordo com o contexto de execução.

No nosso caso, esse erro ocorre porque, no momento de execução do forEach, o contexto do this.\_codigo é o contexto da função passada para o forEach e não o da classe ModuloDeImpressao que contém o construtor com o código da turma.

**Contexto**

O que queremos dizer com **contexto** é que, durante a execução do forEach vamos acessar a propriedade onde está o **código da turma**, porém não vamos conseguir encontrar essa propriedade, porque em nenhum lugar dentro da função que o forEach está executando existe o this.\_codigo.

Ou seja, fora do contexto do forEach conseguimos acessar a propriedade this.\_codigo:

class ModuloDeImpressao {

constructor() {

this.\_codigo = 10;

}

imprime(nomes) {

console.log(this.\_codigo);

nomes.forEach(function(nome){

console.log(`${this.\_codigo}: ${nome}`);

});

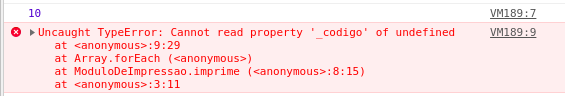
}

}

const impressao = new ModuloDeImpressao();

impressao.imprime(professores);

Como resultado obtemos:



Após instanciarmos a classe novamente, podemos perceber que conseguimos imprimir o código sem problemas, mas ainda estamos com erro. O que queremos é uma maneira de acessar o contexto da classe ModuloDeImpressao no momento em executarmos o laço forEach.

**Arrow Function**

A versão do ECMA Script 2015 do JavaScript, trouxe uma nova forma mais sucinta de trabalhar com funções chamada de **Arrow Functions**, por causa da sintaxe que lembra uma flecha: () =>.

Mas a Arrow Function não é só uma maneira menos verbosa de escrever uma função, ela tem uma característica em particular que vai nos auxiliar em nosso problema: **o escopo léxico**. Mas como assim o **escopo léxico**?

**Escopo Léxico**

Escopo léxico, significa que podemos acessar a propriedade **código** dentro do nosso forEach. O this não irá variar de acordo com o **contexto**. Agora, como o contexto do this.\_codigo é o da classe MóduloDeImpressão temos acesso a propriedade código:

nomes.forEach((nome) => {

console.log(`${this.\_codigo}: ${nome}`);

});

Agora que sabemos como funciona othis e a arrow function, vamos aplicar no nosso problema.

**Resolvendo o problema**

Agora utilizando uma arrow function em nossa classe, temos acesso ao construtor no momento em que entrarmos no laço doforEach, porque o this.\_codigo vai estar sempre no contexto da classe ModuloDeImpressao.

Depois de aplicar a técnica da arrow function, nosso código ficou assim:

class ModuloDeImpressao {

constructor() {

this.\_codigo = 10;

}

imprime(nomes) {

nomes.forEach(nome => {

console.log(`${this.\_codigo}: ${nome}`);

});

}

}

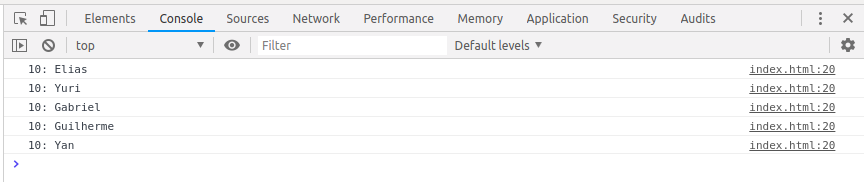
Quando instanciamos um novoModuloDeImpressao:

const professores = ['Elias', 'Yuri','Gabriel', 'Guilherme','Yan'];

const impressao = new ModuloDeImpressao();

impressao.imprime(professores);

Temos como resultado no console:



Ótimo, conseguimos resolver nosso problema com relação ao contexto do this! Agora já temos o código da turma e o nome dos professores.

**Outra solução**

Além das arrow functions, outra maneira que podemos solucionar esse problema é através do método **[bind](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Function/bind" \t "_blank)**, que vai fixar um valor para o this, assim ele não irá variar de acordo com o contexto em que está inserido.

class ModuloDeImpressao {

constructor() {

this.\_codigo = 10;

}

imprime(nomes) {

nomes.forEach(function(nome) {

console.log(`${this.\_codigo}: ${nome}`);

}.bind(this));

}

}

Como podemos ver, o bind vai passar um contexto para um função que não é dela, ou seja, o this.\_codigo dentro do forEach vai ter o contexto da classe ModuloDeImpressao.

**Para saber mais**

As [**Arrow functions**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions/Arrow_functions) tem outras propriedades interessantes como:

* **Retorno:** Não precisamos declarar explicitamente o retorno quando temos um bloco apenas de código, a última expressão realizada vai ser o retorno da arrow function.

const soma = (numero1, numero2) => numero1 + numero2;

soma(3,7) *// 10*

* **Nome:** As Arrows functions ganham nome da variável atribuida quando da criação

const arrow = () => {};

arrow.name; *// arrow*

* **Construtor:** Não é possível criar construtores com Arrow function

const Construtor = () => {};

new Construtor(); *// Constructor is not a constructor*

Se ficou interessado em como o Javascript funciona e como você pode utilizá-lo melhor, aqui na Alura temos uma [**formação em desenvolvimento Javascript**](https://www.alura.com.br/formacao-front-end). Nela, você verá como programar em Javascript, utilizar expressões regulares, dentre outras muitas coisas.

* [NOSSOS INSTRUTORES](https://www.alura.com.br/instrutores)