```
//THIS:
//A palavra reservada "this" no Javascript é semelhante a palavra reservada "self" em algumas linguagens de programação (como o Pytho
n por exemplo), essa palavra é usada para identificar quem é o objeto "chamador" de uma determinada execução, tarefa ou dados.
//Trazendo para o paradigma POO o "this" estaria se referindo ao objeto "instanciador" de determinada Classe.
//Lembre-se que o "this" vai pertencer sempre ao contexto chamador, e dentre eles podemos ter os seguintes contextos:
/*          - Escopo Global ou Window (quando estamos no DOM);
            - Escopo de Objetos;
            - Escopo de Array;
            - Escopo de Variável (Quando usamos arrow function ou bind);
 */

//A INFLUÊNCIA DO THIS SOBRE AS FUNCTION:
//Quando uma função é criada de forma normal usando "function" o this têm a característica de sempre referenciar ao contexto chamador
 da function, não importa o contexto...
function quemEhOThis() { //Veja que function está solta dentro do contexto global, o que significa no momento o This dessa função é o
 contexto global...
    return console.log(
    `Function comparada com o contexto Global: ${this === global}
Function comparada com o contexto Local da Chamada: ${this === this}`);
}
console.log("Function: Exemplo de this sendo chamado á partir de um contexto Global...")
quemEhOThis(); //Perceba que a chamada também está no contexto global, o que significa que ele deverá dar true no primeiro exemplo e
no segundo...



//INFLUÊNCIA DAS ARROW FUNCTION SOBRE O THIS:
//Todo arrow function tem por característica ter o seu this referenciado sobre o contexto léxico, ou seja, sobre o contexto de onde a
 arrow function foi escrita, mesmo que ela seja chamada á partir de outro contexto, ela ainda vai referenciar ao contexto de onde foi
 escrita...
let sempreLocal = () => console.log( //Note que o this será referenciado dentro do contexto da variável let...
    `Arrow Function comparada com o contexto Global: ${this === global}
Arrow Function comparada com o contexto Local da Chamada: ${this === this}`);
console.log("\nArrow Function: exemplo de this sendo chamado á partir de um contexto Global...")
sempreLocal();//Perceba que estamos chamado pelo contexto global, ainda assim, ela vai referenciar ao contexto da variável onde ela f
oi escrita, dando os resultados "false" quando comparada ao contexto global e "true quando comparada ao próprio contexto..."
```

```
//THIS DAS ARROW FUNCTION NÃO MUDA MESMO USANDO BIND OU APLICANDO A OUTRO CONTEXTO QUALQUER:
let comparaArrowFunction = (param) => console.log(this === param); //Perceba que criamos uma arrow function dentro de uma variável ch
amada "comparaArrowFunction" que recebe um contexto e compara com o this da arrow function retornando "true" para o this que for igua
l ao da arrow function e "false" para um this diferente...
console.log("\nArrow Function: Exemplos de comparação entre o this da arrow function e o this de outros contextos...")
console.log("Arrow function recebe \"global\" como parâmetro...");
comparaArrowFunction(global);
console.log("Arrow function recebe \"this\" como parâmetro...");
comparaArrowFunction(this);
console.log("Arrow function recebe \"module.exports\" como parâmetro...");
comparaArrowFunction(module.exports);
console.log("Arrow function é colocada dentro de um object...");
let obj = {};
obj.funcao = comparaArrowFunction;
console.log(obj);
console.log("Ainda assim, quando chamamos a arrow function comparando o this ao contexto do objeto, o resultado é...");
obj.funcao(obj);



//THIS EM ARRAYS:
//Quando um elemento está dentro de um array, ele passa automáticamente a fazer parte do contexto do array, ou seja, o seu this agora
 é o array...
const euSouThisAgora = [quemEhOThis, sempreLocal];
console.log("\nTestando em Arrays:")
euSouThisAgora[0](); //Note como o padrão mudou para quando chamávamos uma função pelo contexto global, agora ela pertence ao context
o do array...
euSouThisAgora[1](); //Note que no arrow function não houve mudança, afinal as arrow function são leais aos contextos onde foram cria
das...



//THIS EM OBJECTS:
```

```
//Quando um elemento está dentro de um object, ele passa automáticamente a fazer parte do contexto do object, ou seja, o seu this ago
ra é o object...
const euSouThisNow = {quemEhOThis, sempreLocal};
console.log("\nTestando em Objects:")
euSouThisNow.quemEhOThis(); //Note como o padrão mudou para quando chamávamos uma função pelo contexto global, agora ela pertence ao
contexto do object...
euSouThisNow.sempreLocal(); //Note que no arrow function não houve mudança, afinal as arrow function são leais aos contextos onde for
am criadas...



//USANDO THIS PARA ACESSAR UM VALOR QUE ESTÁ DENTRO DE UM OBJETO:
const pessoa = {
    saudacao: 'Bom dia!',
    falar() { //Perceba que temos uma função dentro de um objeto que retorna um outro atributo daquele objeto por através da palavra
reservada "this"...
        return console.log(this.saudacao); //como this foi criado dentro de um object ele referencia ao object, perceba que trocamos
o nome do objeto pelo this naturalmente...
    }
}
console.log("\nUsando 'this' para chamar um atributo dentro do mesmo object:")
pessoa.falar();



//THIS NÃO PODE SER USADO PARA CHAMAR UM ATRIBUTO QUE PERTENCE A OUTRO OBJETO:
const outroObject = pessoa.falar; //Perceba que passamos para a constante a função falar do objeto pessoa...
console.log("\nFalhando em capturar o valor do atributo de outro objeto usando o this referenciado em outro objeto:")
outroObject(); //Porém quando chamamos a função ela executa e não consegue encontrar o valor da variável "saudacao", pois a chamada d
a função é um "this.saudacao" que está no contexto do objeto "pessoa", e não no contexto da constante "outroObject", por isso a chama
da da função gera um valor "undefined"...



//AMARRANDO A REFERÊNCIA DE UM ELEMENTO POR ATRAVÉS DO bind():
```

```
const usandoBindNoutroObject = pessoa.falar.bind(pessoa); //Perceba que fizemos a mesma operação de cima, tentando capturar o valor d
o atributo "saudacao" por através da chave "falar" do objeto "pessoa", porém, dessa vez usamos o bind() para amarrar a chamada da fun
ção "falar" ao objeto "pessoa"...
console.log("\nConseguindo pegar o valor do atributo de outro objeto que usa this referenciado a ele mesmo com a ajuda do bind():");
usandoBindNoutroObject(); //Veja como o resultado é diferente quando usamos "bind()"...



//AMARRANDO A REFERÊNCIA THIS DE UMA FUNÇÃO USANDO CONSTANTE:
//Outro artifício tecnológico é usar constantes para amarrar o this de uma função ao contexto léxico da própria função, como acontece
 com as arrow function, veja como fazer isso.
function amarradaPorConstante() {
    this.nome = 'Gabriel';
    const thisAmarrado = this; //Perceba que atribuímos o this a uma constante chamada "thisAmarrado"...

    function dentro(){ //E dentro da nossa função construtora podemos referenciar a constante "thisAmarrado" como se fosse o this do
contexto léxico que chama a função construtora sem precisar usar bind para isso
        thisAmarrado.nome += ' Ferreira';
        console.log(thisAmarrado.nome);
    }

    dentro();
}
console.log("\nConseguindo chamando this á partir de uma constante a qual atribuímos o this de uma outra função:");
new amarradaPorConstante;



//USANDO BIND PARA AMARRAR O THIS DE UMA FUNÇÃO EXECUTADA DENTRO DO SETINTERVAL:
//setInterval também é uma função, por isso, devemos ter cuidado quando executamos um elemento que pertence a outra função com o uso
do this
function idadeDaPessoa() { //Perceba que temos aqui uma função construtora que possuí um atributo chamado "idade" de valor "0"...
    this.idade = 0;
}

console.log("\nUsando bind para chamar o this á partir do contexto léxico da função atribuída a um setInverval:");
//Agora, fora do contexto da função construtora, chamamos um setInterval globalmente...
```

```
setInterval(function() { //Dentro do setInterval criamos uma função anônima...
    this.idade++; //A função tem o papel de chamar um atributo idade referenciando ao objeto que lhe chamou e incrementar 1 ao valor desse objeto, porém, quem está chamando o objeto é contexto global que não possuí nenhum atributo "idade"
    console.log(this.idade); //Também queremos que os valores incrementados sejam impressos a cada espaçamento de tempo do setInterval...
}.bind(idadeDaPessoa()), 1000) //Como queremos que o setInterval chame o atributo idade á partir do escopo da função contrutora "idadeDaPessoa", nós utilizamos um bind que chama essa função (como ela é uma função construtora poderíamos chamá-
la também simplesmente instanciando a função, assim "new idadeDaPessoa")
```

RESULTADO...

```
[Running] node "c:\Users\User\Documents\JAVASCRIPT\ARQUIVOS_DAS_AULAS\064-This.js"
Function: Exemplo de this sendo chamado á partir de um contexto Global...
Function comparada com o contexto Global: true
Function comparada com o contexto Local da Chamada: true

Arrow Function: exemplo de this sendo chamado á partir de um contexto Global...
Arrow Function comparada com o contexto Global: false
Arrow Function comparada com o contexto Local da Chamada: true

Arrow Function: Exemplos de comparação entre o this da arrow function e o this de outros contextos...
Arrow function recebe "global" como parâmetro...
false
Arrow function recebe "this" como parâmetro...
true
Arrow function recebe "module.exports" como parâmetro...
true
Arrow function é colocada dentro de um object...
{ funcao: [Function: comparaArrowFunction] }
Ainda assim, quando chamamos a arrow function comparando o this ao contexto do objeto, o resultado é...
false

Testando em Arrays:
Function comparada com o contexto Global: false
Function comparada com o contexto Local da Chamada: true
Arrow Function comparada com o contexto Global: false
```

```
Arrow Function comparada com o contexto Local da Chamada: true

Testando em Objects:
Function comparada com o contexto Global: false
Function comparada com o contexto Local da Chamada: true
Arrow Function comparada com o contexto Global: false
Arrow Function comparada com o contexto Local da Chamada: true

Usando 'this' para chamar um atributo dentro do mesmo object:
Bom dia!

Falhando em capturar o valor do atributo de outro objeto usando o this referenciado em outro objeto:
undefined

Conseguindo pegar o valor do atributo de outro objeto que usa this referenciado a ele mesmo com a ajuda do bind():
Bom dia!

Conseguindo chamando this á partir de uma constante a qual atribuímos o this de uma outra função:
Gabriel Ferreira

Usando bind para chamar o this á partir do contexto léxico da função atribuída a um setInverval:
1
2
3
4
5
6
7
8
9
10
11

[Done] exited with code=1 in 11.922 seconds
```