```
//ARROW FUNCTIONS:
//São funções resumidas do Javascript, a criação das arrow functions foi motivada pela criação de uma sintáxe simples que trouxesse a
 mesma funcionabilidade que uma função comum, e também, foi motivada pela ideia de criar funções cujo this obedecesse somente ao cont
exto léxico, ou seja, onde a função foi escrita.
//ATENÇÃO!! Functions arrow são sempre funções anônimas, ou seja, elas não tem nome. Por isso, arrows functions devem sempre ser decl
aradas sobre variáveis ou constantes.

//ARROW FUNCTION:
const somatoria = (a, b) => {//Perceba que a função arrow pode ser declarada usando o campo de parâmetros e a famosa seta do arrow, d
epois só precisamos abrir o corpo da função.
    return a + b;
}
console.log('\nFunção é uma arrow function...')
console.log(somatoria(15, 15)); //Como foi atribuída a uma variável, basta referenciá-la pelo nome da variável...



//ARROW RESUMIDA:
const adicao = (a, b) => a + b; //Perceba que a versão resumida da arrow function possui return implícito, não precisamos de declarar
 o seu return, porém, esse tipo de função resumida só funciona com funções que possuem uma única linha de código.
console.log('\nFunção é uma arrow function resumida...')
console.log(adicao(20, 20));



//ARROW VERSÃO AINDA MAIS RESUMIDA:
const nome = n => `Olá ${n}, tudo bem?`; //Quando a nossa função recebe um único parâmetro não precisamos usar os parênteses que deli
mitam o campo de parâmetros da função arrow.
console.log('\nFunção é uma arrow function ainda mais resumida...')
console.log(nome('Gabriel'));



//ARROW SEM PARÂMETROS:
//Toda funcão arrow por padrão recebe parâmetros, mas também podemos criar arrow functions sem parâmetros á receber, da seguinte form
a:
```

```
const ola = () => 'Olá!' //Perceba que a função ainda recebe os seus parênteses, mas vazios, isso mostra que essa função não precisa de ter parâmetros passados á ela.
const oi = _ => 'Oi!' //Algumas pessoas para simplificar mais ainda preferem usar o underline, que também é permitido em arrow functions para criar funções que não recebem parâmetro nenhum, essa é uma medida de simplificar ainda mais uma função que não recebe parâmetros...
console.log('\nArrow function sem parâmetros...')
console.log(ola());
console.log(oi());



//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 objeto que o chamou...
function quemEhOThis() {
    return console.log(
    `Function comparada com o contexto Global: ${this === global}
Function comparada com o contexto Local da Chamada: ${this === this}`);
}
console.log("\nFunction: Exemplo de this sendo chamado á partir de um contexto Global...")
quemEhOThis(); //Perceba que a chamada 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 dentro do contexto léxico, ou seja, onde a função foi criada, que no caso é dentro de uma variável...
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, 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);
```

RESULTADO...

```
[Running] node "c:\Users\User\Documents\JAVASCRIPT\ARQUIVOS_DAS_AULAS\067-Arrow_Functions.js"

Função é uma arrow function...
30

Função é uma arrow function resumida...
40

Função é uma arrow function ainda mais resumida...
Olá Gabriel, tudo bem?

Arrow function sem parâmetros...
Olá!
Oi!

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

[Done] exited with code=0 in 0.169 seconds
```