Objetivos:

I. Declaração de variáveis na linguagem JavaScript;

II. Hoisting.

Observação: Para fazer os exercícios e exemplos recomenda-se o uso do VS Code ou da interface de programação online https://replit.com/.

> No JavaScript as instruções podem ser terminadas usando ponto e vírgula. Até aqui não adotamos o ponto e vírgula, porém constitui boa prática terminar as instruções usando ponto e vírgula.

I. Declaração de variáveis na linguagem JavaScript

Podemos declarar uma variável usando as seguintes notações:

```
var a = 1;     //usando a palavra reservada var
let b = 2;     //usando a palavra reservada let
const c = 3;   //usando a palavra reservada const
d = 4;         //sem o uso de uma palavra reservada
```

> A principal diferença entre `let` e `var` está no escopo da variável.

A variável declarada usando `let` possui o escopo do bloco atual, lembre-se que um bloco é criado usando um par de chaves `{}`. No exemplo a seguir a variável `a` definida dentro do bloco do `if` estará disponível apenas dentro dele:

```
let a = 1; //declarada fora do bloco
if( a == 1 ){
    let a = 2; //esta variável possui o escopo deste bloco
    console.log("Dentro do bloco:",a);
}
console.log("Fora do bloco:",a); //está lendo a variável definida fora do bloco
```

Saída do código:

```
Dentro do bloco: 2
Fora do bloco: 1
```

A variável declarada usando `var` possui o escopo da função ou escopo global de acordo com o local da declaração. No exemplo a seguir a variável `b` definida dentro do bloco do `if` estará disponível dentro da função:

```
function teste(){
    var b = 1;
    if( b == 1 ){
        var b = 2; //esta variável atualiza a variável b
```

```
        console.log("Dentro do if:",b);
    }
    console.log("Fora do if:",b);
}

teste();
```

Saída do código:

```
Dentro do if: 2
Fora do if: 2
```

A variável declarada sem as palavras reservadas `var`, `let` e `const` possui o escopo global do arquivo independentemente do local de sua declaração. No exemplo a seguir a variável `c` definida dentro do bloco do `if` estará disponível em qualquer parte do arquivo:

```
function teste(){
    if( true ){
        c = 3; //esta variável possui o escopo de todo o programa
    }
    console.log("Fora do if:",c);
}

teste();
console.log("Fora da função:",c);
```

Saída do código:

```
Fora do if: 3
Fora da função: 3
```

A variável declarada usando `const` possui o escopo do bloco atual assim como variáveis declaradas usando `let`. Como o nome sugere, `const` é usada para declarar variáveis cujo conteúdo não pode ser trocado. Porém, se a variável possuir o endereço/referência do conteúdo, assim como array ou objeto, daí os elementos do array e membros do objeto podem ser alterados ou removidos. Como exemplo:

```
const vet = []; //a variável vet possui o endereço do array
//não gera erro, pois estamos adicionando um elemento no array
// que a variável vet está apontando
vet[0] = 10;
vet = [10,20,30,40]; //gera erro, pois estaríamos trocando o conteúdo da variável
vet
console.log(vet); //não gera erro, pois é apenas a leitura da variável vet

const obj = {}; //a variável obj possui o endereço do objeto
obj.nome = 'Ana'; //o objeto passa a ter uma propriedade
obj = {}; //gera erro, pois estamos trocando o conteúdo da variável obj
console.log(obj);
```

II. Hoisting

Refere-se ao processo pelo qual o interpretador JavaScript move a declaração de funções e variáveis para o topo de seu escopo, antes da execução do código.

No exemplo a seguir a função `teste` está sendo chamada antes de sua definição. Como o código é interpretado de cima para baixo, então a instrução `teste()` deveria dar erro, pois a função ainda não existe no momento que a instrução de chamada da função foi executada.

```
teste(); //chamada da função teste

//declaração da função teste
function teste(){
    console.log("função")
}
```

No exemplo a seguir a variável `x` está sendo chamada antes de sua definição. Uma variável declarada como `var` pode ser chamada antes da sua declaração sem ser lançada uma mensagem de ReferenceError. No exemplo a seguir a variável `x` terá conteúdo `undefined` (não definido), porém não será lançada mensagem de erro.

```
console.log(x); //leitura da variável x. O conteúdo de x é undefined

var x = 10; //declaração da variável x
```

No exemplo anterior se a variável for declarada usando `let x = 10`, `const x = 10` ou `x = 10` será lançada a mensagem ReferenceError ao tentar ler a variável `x` antes da sua declaração.

Para mais detalhes sobre hoisting acesse https://developer.mozilla.org/en-US/docs/Glossary/Hoisting.

Exercícios

Exercício 1: Complete o código para exibir o número 10 ao rodar o programa.

```
um();

function um(){
    if(true){
        ________ x = 10;
    }
    console.log("x:", x);
}
```

Exercício 2: Qual é o valor exibido no console ao executar o código a seguir.

```
dois();

function dois(){
```

```
    let x = 10
    if(true){
        let x = 20;
    }
    console.log("x:", x);
}
```

Exercício 3: Complete o código para exibir o número 20 ao rodar o programa.

```
tres();

function tres(){
    ________ x = 10
    if(true){
        ________ x = 20;
    }
}

console.log("x:", x);
```

Exercício 4: Complete o código para que o programa não aceite a última atribuição.

```
_________ quatro = {};
quatro.nome = "Exercício";
quatro = [];  //apresentar erro aqui
```

Exercício 5: Complete o código para que seja impresso o valor `undefined` ao executar o código.

```
cinco();

function cinco(){
    console.log("x:", x);
    if(true){
        __________ x = 20;
    }
}
```