IW-II
INTERFACE WEB II

Prof. Anderson Vanin

var, let e const

var

A palavra-chave **var** foi a forma original de declarar variáveis no JavaScript. Entretanto, seu escopo é global ou de função, o que pode levar a comportamentos inesperados. Exemplo:

```
var teste = "hey hi";

function minhaFuncao() {
     var ola = "hello";
}
console.log(ola); // erro: ola não está definido
```

let

let é, agora, a forma preferida de declaração de variáveis. Não é uma surpresa, já que ele é uma melhoria às declarações com **var**.

- **let** tem escopo de bloco

Um bloco é uma porção de código cercado por {}. Um bloco vive dentro dessas chaves. Tudo o que estiver cercado por chaves é um bloco.

Assim, uma variável declarada com **let** em um bloco estará disponível apenas dentro daquele bloco.

let

```
let greeting = "say Hi";
let times = 4;
if (times > 3) {
      let hello = "say Hello instead";
      console.log(hello);// dirá "say Hello instead"
}
console.log(hello) // hello não está definido
```

let

let pode ser atualizado, mas não declarado novamente.

Assim como **var**, uma variável declarada com **let** pode ser atualizada dentro de seu escopo. Diferente de **var**, no entanto, uma variável **let** não pode ser declarada novamente dentro de seu escopo.

```
let greeting = "say Hi";
greeting = "say Hello instead";
```

```
let greeting = "say Hi";
let greeting = "say Hello instead";
// erro: identificador 'greeting' já foi declarado
```

const

Variáveis declaradas com const mantêm valores constantes. Declarações com const compartilham algumas semelhanças com as declarações com let.

Declarações com const têm escopo de bloco

Assim como as declarações de let, as declarações de const somente podem ser acessadas dentro do bloco onde foram declaradas.

const não pode ser atualizado nem declarado novamente

Isso significa que o valor de uma variável declarada com const se mantém o mesmo dentro do escopo. Ela não pode ser atualizada nem declarada novamente.

Métodos de Interação com o Usuário

JavaScript fornece três funções principais para interação com o usuário:

- **Alert**
- **Confirm**
- **Prompt**

Alert

O método **alert** exibe uma caixa de diálogo com uma mensagem e um botão "OK".

Exemplo:

```
alert("Bem-vindo ao nosso site!");
```

Confirm

O método confirm exibe uma caixa de diálogo com uma mensagem e dois botões: "OK" e "Cancelar". Ele retorna true se o usuário clicar em "OK" e false se clicar em "Cancelar". Exemplo:

```
let resposta = confirm("Você deseja continuar?");
if (resposta) {
      alert("Você escolheu continuar.");
} else {
      alert("Você cancelou a ação.");
}
```

Prompt

O método prompt exibe uma caixa de entrada para o usuário digitar um valor.

Exemplo:

```
let nome = prompt("Qual é o seu nome?");
alert("Olá, " + nome + "!");
```

Exercícios

1. Exiba uma mensagem de boas-vindas ao usuário utilizando alert.
2. Pergunte ao usuário seu nome usando prompt e exiba um alerta com a resposta.
3. Pergunte ao usuário sua idade usando prompt e exiba no console se ele é maior ou menor de idade.
4. Crie um confirm perguntando se o usuário deseja prosseguir e mostre uma mensagem de acordo com a resposta.
5. Declare duas variáveis numéricas com let e peça ao usuário dois números via prompt, depois exiba a soma.
6. Peça ao usuário para digitar um número e exiba o dobro dele usando alert.
7. Crie um script que pergunte o nome e a idade do usuário e exiba uma mensagem personalizada com as informações fornecidas.

Por que usar parseFloat() ou parseInt()?

O parseFloat() e parseInt() são funções usadas em JavaScript para converter strings em números, permitindo realizar cálculos matemáticos corretamente.

O prompt() retorna os valores digitados como strings. Se tentarmos somar duas strings numéricas sem conversão, ocorrerá concatenação em vez de soma.