Javascript

Funções

Declaração de Funções

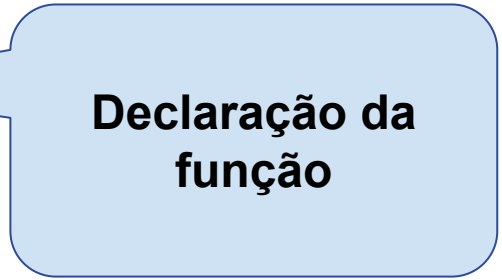
- Funções em Javascript são tipos especiais de valores.
- Veja a função abaixo:

```
function logar(teste) {
    console.log(teste);
}

logar("Teste 1");
logar("Teste 2");
```

Declaração de Funções

- Declarando e chamando funções:

```
function logar(teste) {
    console.log(teste);
}

logar("Teste 1");
logar("Teste 2");
```



Declaração de Funções

- Declarando e chamando funções:

```
function logar(teste, numeroVezes) {
    for (let i = 0; i < numeroVezes; i++) {
        console.log(i + 1, "-", teste);
    }
}
logar("Teste 1", 2);
logar("Teste 2", 3);
```

Parâmetros da função: podem ser passados mais de um parâmetro, separando-os por vírgula.

Declaração de Funções

- Declarando e chamando funções:

```
function logar(teste, numeroVezes) {
    let i;
    for (i = 0; i < numeroVezes; i++) {
        console.log(i + 1, "-", teste);
    }
}
logar("Teste 1", 2);
logar("Teste 2", 3);
```

Bloco de código da função:
é possível declarar novas variáveis, chamar funções, etc.

Declaração de Funções

- Declarando e chamando funções:

```
function logar(teste, numeroVezes) {
    let i;
    for (i = 0; i < numeroVezes; i++) {
        console.log(i + 1, "-", teste);
    }
}
logar("Teste 1", 2);
logar("Teste 2", 3);
```

Chamada da função: devem ser passados valores para os seus parâmetros, pois se isso não for feito, os mesmos terão o valor “undefined”.

Funções

- Escopo de variáveis e constantes:

```
function logar(teste, numeroVezes) {
    let i;
    for (i = 0; i < numeroVezes
        console.log(i + 1, "-",
    }
}
logar("Teste 1", 2);
logar("Teste 2", 3);
```

Variáveis e constantes declaradas dentro do bloco da função são visíveis somente dentro deste bloco.

Funções

- Escopo de variáveis e constantes:

```
const separador = "-";
function logar(teste, numeroVezes) {
    for (let i = 0; i < numeroVezes; i++) {
        console.log(i + 1, separador, teste);
    }
}
logar("Teste 1", 2);
logar("Teste 2", 3);
```

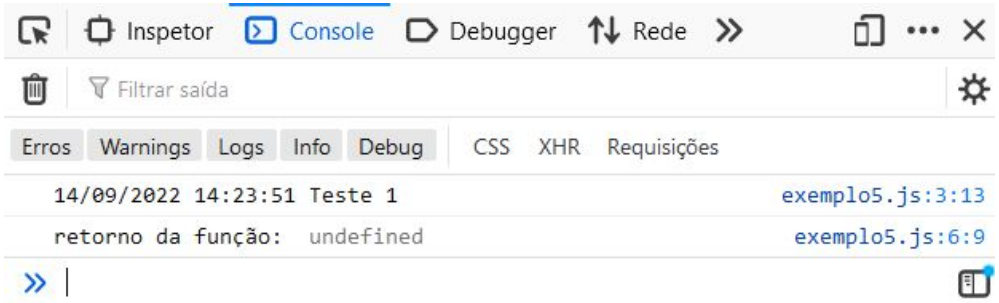
Variáveis e constantes declaradas fora de funções são conhecidas como variáveis/constantes externas ou globais.

Funções

- Retornando valores

```
function logar(mensagem) {
    const dataHora = new Date();
    console.log(dataHora.toLocaleString(), mensagem);
}
console.log("retorno da função: ", logar("Teste 1"));
```

Quando uma função não possui o comando return ou quando o comando return não possui uma expressão a ser retornada (return;), undefined é retornado.

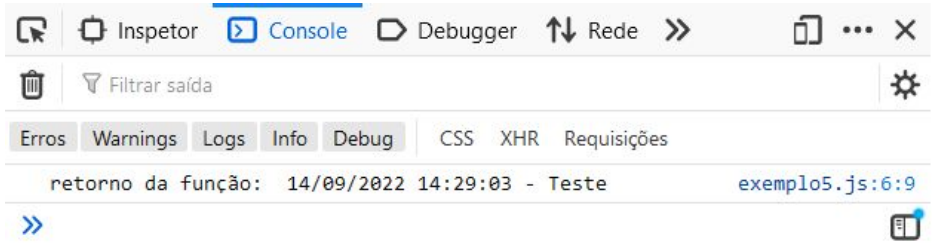


Funções

- Retornando valores

```
function logar(mensagem) {
    const dataHora = new Date();
    return dataHora.toLocaleString() + " - " + mensagem;
}
console.log("retorno da função: ", logar("Teste"));
```

Quando uma função possui o comando return, o valor retornado pode ser armazenado em variáveis ou utilizado em expressões.



Funções

- Retornando valores

```
function logar(mensagem) {
    const dataHora = 
    return (
        dataHora.toLocaleString() + " - " + mensagem
    );
}
console.log("retorno da função: ", logar("Teste"));
```

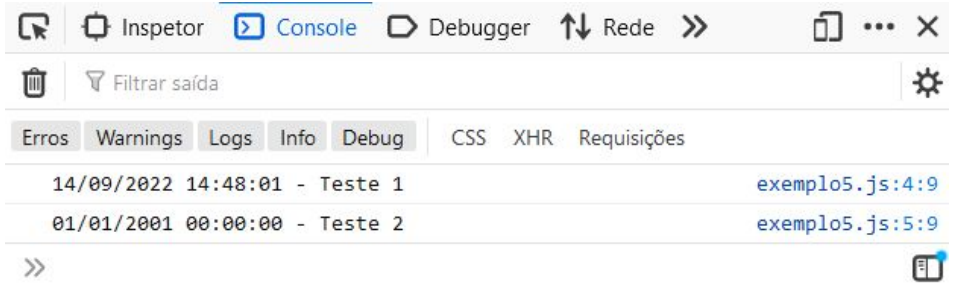
Atenção:
Se a expressão a ser retornada for muito grande, utilize parênteses para indicar que o retorno possuirá mais de uma linha. O comado return sozinho em uma linha será interpretado como (return;).

Funções

- Valor padrão para parâmetro

É possível definir um valor padrão para casos em que determinado parâmetro não receba valor.

```
function logar(mensagem, dataHora = new Date()) {
    return dataHora.toLocaleString() + " - " + mensagem;
}
console.log(logar("Teste 1"));
console.log(logar("Teste 2", new Date(2001, 0, 1)));
```

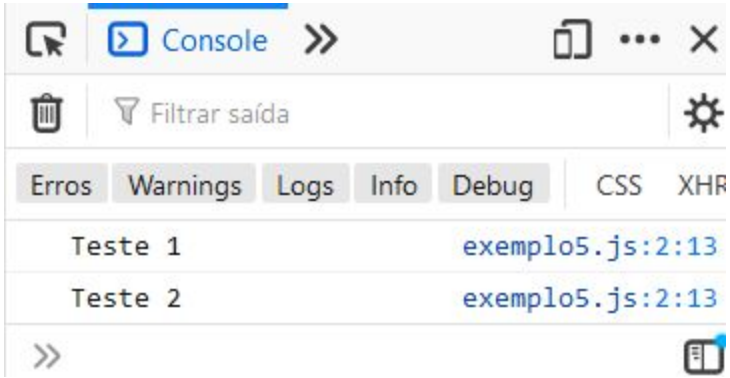


Funções como expressões

- Atribuindo função a variável

```
function logar(mensagem) {
    console.log(mensagem);
}
const log = logar;
log("Teste 1");
logar("Teste 2");
```

Uma função pode ser atribuída a uma variável ou a uma constante.



Funções como expressões

- Atribuindo função a variável

```
function logar(mensagem, dataHora = new Date()) {
    return dataHora.toLocaleString() + " - " + mensagem;
}
let log = logar;
console.log(log("Teste 1"));
console.log(log);
```

A chamada à função está condicionada à adição dos parênteses após o seu nome ou o nome da sua variável. Nesta linha não adicionamos os parênteses e, por isso, a função foi logada.

Funções como expressões

- Outra forma de declarar uma função

```
const logar = function (mensagem, dataHora = new Date()) {
    return dataHora.toLocaleString() + " - " + mensagem;
};
console.log(logar("Teste 1"));
```

Terminamos o comando com “;” (ponto e vírgula) por se tratar de uma atribuição de um valor à uma constante ou variável.

Arrow Functions

- Exemplo de arrow function:

```
const logar = (mensagem, dataHora = new Date()) => {
    return dataHora.toLocaleString() + " - " + mensagem;
};
console.log(logar("Teste 1"));
```

- Note as diferenças:

```
const logar = function (mensagem, dataHora = new Date()) {
    return dataHora.toLocaleString() + " - " + mensagem;
};
console.log(logar("Teste 1"));
```

Arrow Functions

- Exemplo de arrow function:

```
const logar = (mensagem, dataHora = new Date()) => {
    return dataHora.toLocaleString() + " - " + mensagem;
};
co
```

A palavra reservada “function” não é necessária para indicar que se trata de uma função.

Arrow Functions

- Exemplo de arrow function:

```
const logar = (mensagem, dataHora = new Date()) => {
    return dataHora.t      leString() + " - " + mensagem;
};
console.log(
```

Os parâmetros são passados normalmente.

Arrow Functions

- Exemplo de arrow function:

```
const logar = (mensagem, dataHora = new Date()) => {
    return dataHora.toLocaleString() + " - " + men
};
console.log(logar("Teste 1")
```

O operador “=>” é utilizado para indicar qual será o corpo da função.

Arrow Functions

- Exemplo de arrow function:

```
const logar = (mensagem, dataHora = new Date()) => {
    return dataHora.toLocaleString() + " - " + mensagem;
};
```

No corpo da função é possível programar normalmente.

Arrow Functions

- Arrow function que imprime “Olá mundo!” no console:

```
const ola = () => {
    console.log("Olá mundo!");
};
ola();
```

Arrow Functions

- Arrow function que imprime “Olá mundo!” no console:

```
const ola = () => {
    consol      ("Olá mundo!");
};
ola(
```

Mesmo quando não se recebe parâmetros, é necessário abrir e fechar os parênteses.

Arrow Functions

- Arrow function que imprime uma mensagem passada por parâmetro no console:

```
const ola = (mensagem) => {
     console.log(mensagem);
};
ola("Olá mundo!");
```

Arrow Functions

- Arrow function que imprime uma mensagem passada por parâmetro no console:

```
const ola = mensagem => {
    console.log(mensagem);
};
ola("Olá mundo!");
```

Quando somente um parâmetro é recebido, os parênteses podem ser retirados

Arrow Functions

- Arrow function que concatena duas strings adicionando um espaço no meio:

```
const contatenar = (str1, str2) => {
     return str1 + " " + str2;
};
console.log(concatenar("Olá", "mundo!"));
```

Arrow Functions

- Arrow function que concatena duas strings adicionando um espaço no meio:

```
const contatenar = (str1, str2) => str1 + " " + str2;


console.log(concatenar("Ol
```

Quando o retorno for uma única linha, as chaves podem ser retiradas, assim como o comando “return”.

Arrow Functions

- Arrow function que retorna o quadrado de um número recebido por parâmetro:

```
const elevarAoQuadrado = x => x * x;


console.log(elevarAoQuadrado(5));
```

Arrow Functions

- Arrow function que retorna o quadrado de um número recebido por parâmetro:

```
const elevarAoQuadrado = x => x * x;


cons
```

Nome da variável que irá referenciar a função.

Arrow Functions

- Arrow function que retorna o quadrado de um número recebido por parâmetro:

```
const elevarAoQuadrado = x => x * x;


console.log(eleva
```

Parâmetro recebido.

Arrow Functions

- Arrow function que retorna o quadrado de um número recebido por parâmetro:

```
const elevarAoQuadrado = x => x * x;


console.log(elevarAoQua
```

Expressão cujo resultado será retornado.

Fim de material

Dúvidas?