Introdução ao JavaScript

Conhecimentos	2. Plataformas de desenvolvimento: conceitos. Tipos. Características e especificações técnicas. 4. Lógica de Programação: Conceito de algoritmo. Algoritmos naturais e estruturados. Representaç
Data da aula	@17 de julho de 2024
Tipo	Atividade em Classe Aula Expositiva

Serviço Nacional de Aprendizagem Comercial

Departamento Regional de Pernambuco

Habilitação Profissional Técnica em Informática

UC9 - Desenvolver Algoritmos

Thiago Nogueira - Instrutor de Educação Tecnológica

thiago.nogueira@senac.pe.br
linkedin.com/tdn
(81) 9 9627-0419
@thiagoo._nogueiraa

Objetivos de aprendizado

1. Introdução ao JavaScript
2. Console
3. Variáveis
4. Operadores

Conceitos Iniciais

O que é JavaScript?

1. Também chamada de JS, é a linguagem de criação de scripts para a Web;

2. É utilizado por bilhões de páginas para:
    a. Adicionar funcionalidades
    b. Verificar formulários
    c. Comunicar com servidores

Console

O console é um painel que exibe mensagens importantes, como erros, para desenvolvedores. Grande parte do trabalho que o computador realiza com nosso código é invisível para nós por padrão. Se quisermos ver coisas aparecendo em nossa tela, podemos imprimir ou registrar diretamente em nosso console.

Em JavaScript, a palavra-chave `console` refere-se a um objeto, uma coleção de dados e ações, que podemos usar em nosso código. Palavras-chave são palavras incorporadas na linguagem JavaScript, de modo que o computador as reconhece e as trata de maneira especial.

Uma ação, ou método, que está incorporada no objeto `console` é o método `.log()`. Quando escrevemos `console.log()`, o que colocamos dentro dos parênteses será impresso, ou registrado, no console.

```
// Imprime o valor 5
console.log(5);
```

Comentários

À medida que escrevemos JavaScript, podemos escrever comentários em nosso código que o computador irá ignorar enquanto nosso programa for executado. Esses comentários existem apenas para leitores humanos.

Existem dois tipos de comentários de código em JavaScript:

- Um *comentário de uma única linha* comentará uma única linha e será indicado por duas barras `//` antes dela.

```
// Esse trecho será ignorado
console.log('imprimindo qualquer coisa...');
```

- Um *comentário de várias linhas* comentará várias linhas e será indicado com `/*` para iniciar e `*/` finalizar o comentário.

```
/*
Tudo isso está sendo ignorado
mesmo que tenha código escrito
console.log(123);
*/
```

Também pode-se usar para comentar algo no meio de uma linha de código:

```
console.log(/*IGNORED!*/ 5);  // Continua imprimindo só 5
```

Tipos de dados

Tipos de dados são as classificações que damos aos diferentes tipos de dados que usamos na programação. Em JavaScript, existem oito tipos de dados fundamentais:

- *Número* : Qualquer número, incluindo números com decimais: `4`, `8`, `1516`, `23.42`
- *BigInt* : Qualquer número maior que $2^53 - 1$ ou menor que $-(2^{53} - 1)$, com n anexado ao número: 1234567890123456n.
- *String* : Qualquer agrupamento de caracteres no seu teclado (letras, números, espaços, símbolos, etc.) entre aspas simples: `' ... '` ou aspas duplas `" ... "`.
- *Boolean* : Este tipo de dados possui apenas dois valores possíveis - `true` ou `false`.

- *Nulo* : Este tipo de dado representa a ausência intencional de um valor, e é representado pela palavra-chave `null` (sem aspas).
- *Indefinido* : este tipo de dados é indicado pela palavra-chave `undefined` (sem aspas). Também representa a ausência de um valor, embora tenha um uso diferente de `null`. `undefined` significa que um determinado valor não existe.
- Sy*mbol* : um recurso mais recente da linguagem, os símbolos são identificadores exclusivos, úteis em codificações mais complexas. Não há necessidade de se preocupar com isso por enquanto.
- *Object*: Coleções de dados relacionados.

Os primeiros 7 desses tipos são considerados *tipos de dados primitivos* . Eles são os tipos de dados mais básicos da linguagem. *Os objetos* são mais complexos e você aprenderá muito mais sobre eles à medida que avançar nos conhecimentos da linguagem.

```
// imprime uma string
console.log('Uma string qualquer');

// imprime um número
console.log(44);
```

Operadores aritméticos

Um operador é um personagem que executa uma tarefa em nosso código. JavaScript possui vários *operadores aritméticos* integrados , que nos permitem realizar cálculos matemáticos em números. Incluem:

- Adicionar: +
- Subtrair: -
- Multiplicar: *
- Dividir: /
- Restante: %

```
console.log(3 + 4); // Imprime 7
console.log(5 - 1); // Imprime 4
console.log(4 * 2); // Imprime 8
console.log(9 / 3); // Imprime 3

console.log(11 % 3); // Imprime 2
console.log(12 % 3); // Imprime 0
```

Concatenação de Strings

Operadores não servem apenas para números! Quando um `+` operador é usado em duas strings, ele anexa a string direita à string esquerda:

```
console.log('bubba' + 'loo'); // Imprime'bubbaloo'
console.log('wo' + 'ah'); // Imprime 'woah'
```

Propriedades

Quando você introduz um novo dado em um programa JavaScript, o navegador o salva como uma instância do tipo de dados. Todos os tipos de dados têm acesso a propriedades específicas que são transmitidas a cada instância. Por exemplo, cada instância de string possui uma propriedade chamada `length` que armazena o número de caracteres dessa string. Você pode recuperar informações de propriedade anexando à string um ponto final e o nome da propriedade:

```
console.log('Hello'.length); // Imprime 5
```

O `.` é outra operador! Chamamos isso de *operador ponto* .

Métodos

Métodos são ações que podemos realizar. Os tipos de dados têm acesso a métodos específicos que nos permitem lidar com instâncias desses tipos de dados. JavaScript fornece vários métodos de string.

Chamamos ou *usamos* esses métodos anexando uma instância com:

- um ponto (o operador ponto)
- o nome do método
- abrindo e fechando parênteses

Por exemplo `'string de exemplo'.methodName()`.

```
console.log('hello'.toUpperCase()); // Retorna a string maiúscula - 'HELLO'
console.log('Hey'.startsWith('H')); // Verifica com qual inicial inicia a string - True
```

Objetos Integrados

Além do `console`, existem outros objetos incorporados no JavaScript. Esses objetos "embutidos" estão cheios de funcionalidades úteis.

Por exemplo, se você quiser realizar operações matemáticas mais complexas do que aritméticas, o JavaScript possui o objeto integrado `Math`.

A grande vantagem dos objetos é que eles possuem métodos! Vamos chamar o método .random() do objeto embutido `Math`:

```
console.log(Math.random()); // Imprime um valor aleatório entre 0 e 1
```

Para gerar um número aleatório entre 0 e 50, poderíamos multiplicar esse resultado por 50.

Atividades de Aprendizagem

Exercício 1

Escreva um script JavaScript que utilize o console para imprimir diferentes tipos de dados e operadores aritméticos.

1. Crie uma variável `x` com o valor `10` e uma variável `y` com o valor `5`.
2. Imprima no console o resultado das seguintes operações aritméticas entre `x` e `y`: adição, subtração, multiplicação e divisão.
3. Crie uma string chamada `greeting` com o valor `"Hello, World!"` e imprima-a no console.
4. Crie um comentário de linha única e um comentário de bloco no seu código.

Exercício 2

Escreva um script JavaScript que manipula diferentes tipos de dados e realiza concatenação de strings.

1. Crie uma variável `name` com o seu nome.
2. Crie uma variável `age` com a sua idade.
3. Crie uma variável `isStudent` com o valor booleano `true`.
4. Crie uma variável `info` que concatena `name`, `age` e `isStudent` em uma string descritiva e imprima-a no console.

Exercício 3

Utilize objetos integrados e métodos em JavaScript para realizar algumas operações úteis.

1. Utilize o método `.random()` do objeto Math para gerar um número aleatório entre 0 e 1 e imprima-o no console.
2. Utilize o método `.toUpperCase()` para transformar a string `"hello"` em letras maiúsculas e imprima-a no console.
3. Crie um objeto chamado `person` com propriedades `firstName`, `lastName` e `age`. Adicione um método `getFullName` que retorna o nome completo da pessoa.

4. Utilize o método `getFullName` do objeto `person` e imprima o nome completo no console.

Variáveis

Houve muitas mudanças introduzidas na versão ES6 do JavaScript em 2015. Uma das maiores mudanças foram duas novas palavras-chave e `let` , `const` para criar ou *declarar* variáveis . Antes do ES6, os programadores só podiam usar apalavra reservada `var` para declarar variáveis.

Hoisting

Em JavaScript, toda variável é **"elevada/içada" (*hoisting*)** até o topo do seu contexto de execução. Esse mecanismo move as variáveis para o topo do seu escopo antes da execução do código.

```
var exibeMensagem = function() {
    mensagem = 'Teste';
    console.log(mensagem);
    var mensagem;
}
exibeMensagem(); // Imprime 'Teste'
```

Criar variáveis: var

Vamos considerar o exemplo:

```
var myName = 'Maria';
console.log(myName);
// Output: Maria
```

1. var, abreviação de variável, é uma palavra-chave do JavaScript que cria, ou declara, uma nova variável.
2. myName é o nome da variável. Escrever dessa forma é uma convenção padrão em JavaScript chamada *camelcase*. Nele, você agrupa palavras em uma só, a primeira palavra é em minúsculas, e cada palavra subsequente terá a primeira letra em maiúsculas. (por exemplo, *camelCaseTudo*).
3. = é o operador de atribuição. Ele atribui o valor ('Maria') à variável (myName).
4. Após a variável ser declarada, o valor da string `'Maria'` é impresso no console referenciando o nome da variável: `console.log(myName)`.

Existem algumas regras gerais para nomear variáveis:

- Os nomes das variáveis não podem começar com números.
- Os nomes das variáveis diferenciam maiúsculas de minúsculas, portanto, `myName` e `myname` seriam variáveis diferentes.

Criar variáveis: let

Foi pensando em trazer o escopo de bloco (tão conhecido em outras linguagens) que o ECMAScript 6 destinou-se a disponibilizar essa mesma flexibilidade (e uniformidade) para a linguagem.

```
var exibeMensagem = function() {
     if(true) {
         var escopoFuncao = 'Debi';
         let escopoBloco = 'Loide';

        console.log(escopoBloco); // Loide
    }
    console.log(escopoFuncao); // Debi
    console.log(escopoBloco);
}
```

```
exibeMensagem(); // Imprime 'Loide', 'Debi' e dá um erro
```

Veja que quando tentamos acessar uma variável que foi declarada através da palavra-chave `let` fora do seu escopo, o erro *Uncaught ReferenceError: escopoBloco is not defined* foi apresentado.

Portanto, podemos usar tranquilamente o `let`, pois o escopo de bloco estará garantido.

Criar variáveis: const

A `const` palavra-chave também foi introduzida no ES6 e é uma abreviação da palavra constante. Assim como com `var` e `let` você pode armazenar qualquer valor em uma `const` variável.

```
const myName = 'Gilberto';
console.log(myName); // Output: Gilberto
```

No entanto, uma `const` variável não pode ser reatribuída porque é *constante* . Se você tentar reatribuir uma `const` variável, obterá um arquivo `TypeError`.

Variáveis constantes *devem* receber um valor quando declaradas. Se você tentar declarar uma `const` variável sem valor, obterá um arquivo `SyntaxError`.

Operadores de Atribuição e Incremento

- i++ → i = i + 1
- i - - → i = i - 1
- i += 1,2,3...n → i = i +1,2,3...n
- i-= 1,2,3...n → i = i -1,2,3...n
- i*= 1,2,3...n → i = i *1,2,3...n
- i/= 1,2,3...n → i = i /1,2,3...n

```
let x = 20;
x -= 5; // Pode ser escrito como x = x - 5
console.log(x); // Output: 15

let y = 50;
y *= 2; // Pode ser escrito como y = y * 2
console.log(y); // Output: 100

let z = 8;
z /= 2; // Pode ser escrito como z = z / 2
console.log(z); // Output: 4
```

Interpolação de Strings

Na versão ES6 de JS foi inserida uma funcionalidade chamada de **template literals** ou template strings. Onde conseguimos realizar a interpolação, veja o código na prática:

```
const meuPet = 'caranguejeira';
console.log(`Eu tenho uma ${myPet}.`);
// Output: Eu tenho um pet caranguejeira.
```

- A sintaxe é formada por uma string entre crases (`)
- As variáveis precisam ser inseridas com um sifrão ($) e entre chaves ({nome})
- Esta é a sintaxe completa para você utilizar a interpolação
- Conseguimos aplicar quantas variáveis forem preciso, elas são convertidas para texto

Operador typeof

Ao escrever o código, pode ser útil acompanhar os tipos de dados das variáveis no seu programa. Se precisar verificar o tipo de dados do valor de uma variável, você pode usar o operador `typeof`.

```
const unknown1 = 'foo';
console.log(typeof unknown1); // Output: string

const unknown2 = 10;
console.log(typeof unknown2); // Output: number

const unknown3 = true;
console.log(typeof unknown3); // Output: boolean
```

Atividades de Aprendizagem

Exercício 1

1. Declare uma variável `a` usando `var` e atribua-lhe o valor `10`.
2. Declare uma variável `b` usando `let` e atribua-lhe o valor `20`.
3. Declare uma variável `c` usando `const` e atribua-lhe o valor `30`.
4. Antes da declaração da variável `a`, tente imprimir seu valor no console e observe o comportamento do hoisting.
5. Tente fazer o mesmo com as variáveis `b` e `c` antes de suas declarações e observe o comportamento.

Exercício 2

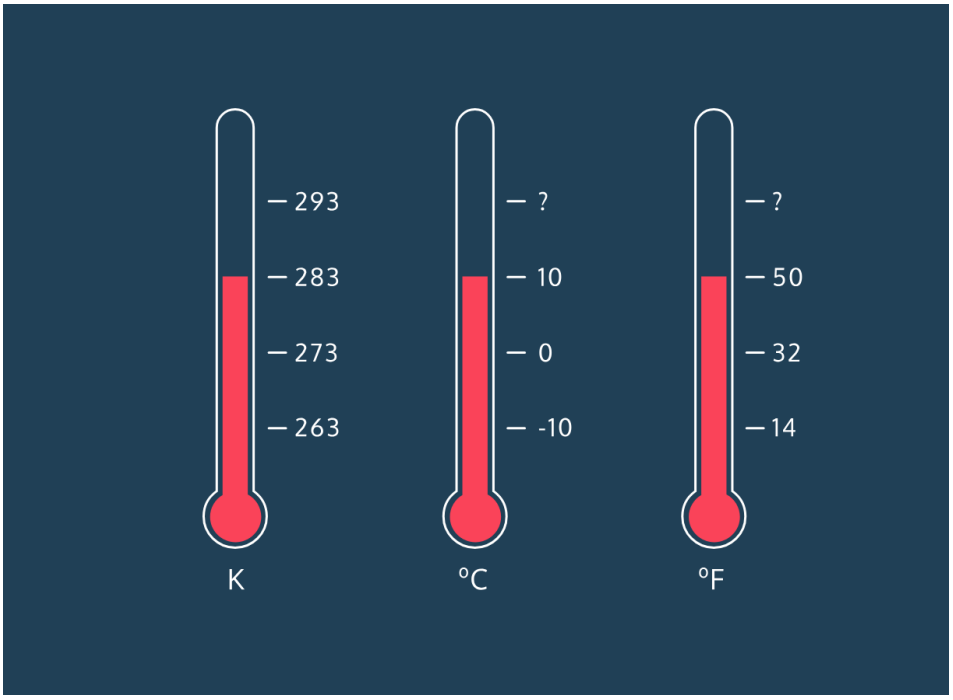
1. Declare uma variável `num` com o valor `5`.
2. Use o operador de atribuição para adicionar `10` a `num`.
3. Use o operador de incremento para aumentar o valor de `num` em 1.
4. Declare uma variável `result` e use a interpolação de strings para criar uma mensagem que inclui o valor de `num`.

Exercício 3

1. Declare uma variável `name` e atribua-lhe o valor `"Alice"`.
2. Declare uma variável `age` e atribua-lhe o valor `25`.
3. Declare uma variável `isStudent` e atribua-lhe o valor `true`.
4. Use o operador `typeof` para imprimir o tipo de cada uma dessas variáveis.
5. Altere o valor de `age` para um número decimal e imprima o novo tipo usando `typeof`.
6. Crie uma interpolação de strings que combine todas essas variáveis em uma mensagem descritiva.

Exercício 4

Dada a escala de temperaturas a seguir, faça um script que converte um valor Kelvin, salvo em uma variável, em Celsius e depois em Fahrenheit, imprimindo os valores dos três.



Exercício 5

Os cães amadurecem mais rapidamente do que os seres humanos. Costumamos dizer que a idade de um cão pode ser calculada em “anos caninos” para contabilizar seu crescimento em comparação com um ser humano da mesma idade. De certa forma, poderíamos dizer que o tempo passa rapidamente para os cães – 8 anos na vida de um ser humano equivalem a 45 anos na vida de um cão. Quantos anos você teria se fosse um cachorro?

Veja como você converte sua idade de “anos humanos” para “anos caninos”:

- Os primeiros dois anos de vida de um cão contam como 10,5 anos caninos cada.
- Cada ano seguinte equivale a 4 anos caninos.

Use JavaScript para converter sua idade humana em anos caninos.

Exercício 6

Faça um programa em Javascript que leia 3 números, some-os e exiba a média entre eles.