Trilha de aprendizagem 01

Objetivo: Construir e usar objetos em JavaScript.

Introdução

Objetos são tipos de dados abstratos usados em programação. Representam coisas do mundo real, pessoas, animais, lugares, bem como conceitos: conta bancária, empréstimo, estrutura de arquivos.

As linguagens que implementam a técnica de programação orientada a objetos, em sua maioria, utilizam o conceito de classes para criação de objetos. A Classe vai modelar os objetos, ela é uma forma de demonstrar todas as características e comportamento que os objetos terão.

JavaScript não é uma linguagem baseada em classes, embora seja possível defini-las. Em JavaScript é possível criar objetos de forma literal, ou seja, sem o uso de classes.

Links de referência:

https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects

https://www.w3schools.com/js/js_objects.asp

Como definir um objeto JavaScript

- Usando um objeto literal
- Usando a new palavra-chave
- Usando um construtor de objeto
- Usando a forma declarativa com *class*

1 – Abra o editor de código JavaScript (VsCode). Crie um novo projeto.

2 – Crie uma página *html* e insira o seguinte código nela:

// Criação literal de objetos.

```
<script>

    // Criação de um objeto literal
    const pessoa = {primeiroNome: "Junior",
                    ultimoNome: "Bandeira",
                     idade:"46",
                    corOlho:"azul"};

    // Demonstração dos dados do ojbeto
    document.getElementById("demo").innerHTML =
    pessoa.primeiroNome + " tem " + pessoa.idade + " anos.";

</script>
```

3 – O próximo passo é criar objetos utilizando a palavra-chave *new*. Continue a tarefa no arquivo criado anteriormente, apenas acrescente o seguinte código:

// Criação de objetos usando a palavra new.

```
// Criação de um objeto com a palavra chave new
        const pessoa2 = new Object();

// Atribuição de dados as propriedades do objeto
        pessoa2.primeiroNome = "Carlos";
```

```
        pessoa2.ultimoNome = "Drummont";
        pessoa2.idade = 50;
        pessoa2.corOlhos = "blue";

       // Demonstração dos dados do objeto
        document.getElementById("demo2").innerHTML =
        pessoa2.primeiroNome + " tem " + pessoa2.idade + " anos.";
```

Essa é a forma de criar objetos com a palavra-chave new com a função construtora Object(), não definida pelo programador;

4 – Na sequência vamos aperfeiçoar o objeto pessoa com a **adição de um método**.

```
   nomeCompleto:  function() {
              return this.primeiroNome + " " + this.ultimoName;
          }

document.getElementById("demo").innerHTML = pessoa.nomeCompleto();
```

Observe que no exemplo a palavra reservada `this` está sendo utilizada. O `this` se refere ao objeto atual em que o código está sendo escrito.

Veja o código completo do arquivo .html:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Objeto literal - JavaScript</title>
</head>
<body>
    <p id="demo"></p>
    <p id="demo2"></p>
     <script>
         // Criacao de um objeto literal
         const pessoa = {primeiroNome: "Junior",
                         ultimoNome: "Bandeira", idade:"46", corOlho:"azul",
         nomeCompleto: function() {
             return this.primeiroNome + " " + this.ultimoNome;
          }
         };

         document.getElementById("demo2").innerHTML = pessoa.nomeCompleto();

         // Demonstracao dos dados do ojbeto
         document.getElementById("demo").innerHTML =
         pessoa.primeiroNome + " tem " + pessoa.idade + " anos.";
     </script>
</body>
</html>
```

5 – A próxima tarefa é criar um objeto pessoa com um número maior de atributos e métodos.

```
let pessoa1 = {
  nome: ["Bob", "Smith"],
  idade: 32,
  sexo: "masculino",
  interesses: ["música", "esquiar"],
  bio: function () {
    alert(
      this.nome[0] +
        " " +
        this.nome[1] +
        " tem " +
        this.idade +
        " anos de idade. Ele gosta de " +
        this.interesses[0] +
        " e " +
        this.interesses[1] +
        ".",
    );
```

```
  },
  saudacao: function () {
    alert("Oi! Eu sou " + this.nome[0] + ".");
  },
};
```

6 – Vamos criar esse objeto em um arquivo separado. Vamos chamar de *objetos.js.*

7 – Em seguida vamos chamar o arquivo dentro do código html:

```
<script src="objetos.js"></script>
```

8 – Depois de inserir o arquivo com o código de criação do objeto, agora é hora de acessar os dados dele.

```
<script>
        document.getElementById("demo3").innerHTML = pessoa1.idade;
</script>
```

9 – Mostre mais dados.

```
<script>
        document.getElementById("demo3").innerHTML = pessoa1.idade +
               " "+pessoa1.interesses[1];
           pessoa1.bio();
 </script>
```

10 – Em algumas circunstâncias é necessário criar muitos objetos do mesmo tipo, por exemplo, várias pessoas, várias vendas, vários produtos. Nesses casos existe um recurso importante que pode ser usado, tratam-se dos **métodos construtores**. Veja como funciona:

// Criação de objetos usando método construtor definido pelo programador:

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
}
```

11 – Coloque o código do método construtor no arquivo objetos.js.

12 – Em seguida use o método para criar algumas pessoas.

```
const myFather = new Person("John", "Doe", 50, "blue");
const myMother = new Person("Sally", "Rally", 48, "green");
const mySister = new Person("Anna", "Rally", 18, "green");

const mySelf = new Person("Johnny", "Rally", 22, "green");
```

13 – Mostre a idade de algumas pessoas criadas.

```
   document.getElementById("demo").innerHTML =
   "My father is " + myFather.age + ". My mother is " + myMother.age + ".";
```

14 – A partir do ECMAScript 6 (ES6), JavaScript introduziu a sintaxe de classes, que é uma forma mais declarativa e simplificada de criar objetos e lidar com herança. Por exemplo:

// Criação de objetos usando class:

```
class Carro {
    constructor(marca, modelo) {
        this.marca = marca;
        this.modelo = modelo;
    }
```

```
}

let meuCarro = new Carro("Toyota", "Corolla");
```

15 – Veja no código a seguir como é possível implementar herança utilizando classes em JavaScript:

```
// Definindo a classe Animal (classe base)
class Animal {
    constructor(nome) {
        this.nome = nome;
    }

    fazerSom() {
        console.log("Fazendo algum som...");
    }
}

// Definindo a classe Cachorro que herda de Animal
class Cachorro extends Animal {
    constructor(nome, raca) {
        super(nome); // Chama o construtor da classe pai (Animal)
        this.raca = raca;
    }

    latir() {
        console.log("Au au!");
    }

    toString() {
        return `${this.nome} é um ${this.raca}`;
    }
}

// Exemplo de uso das classes
let meuCachorro = new Cachorro("Rex", "Labrador");
console.log(meuCachorro.toString()); // Saída: Rex é um Labrador
meuCachorro.fazerSom();              // Saída: Fazendo algum som...
meuCachorro.latir();                 // Saída: Au au!
```

16 – Também é possível criar objetos a partir de outros objetos. O método **Object.create()** cria um novo objeto, utilizando um outro objeto existente como protótipo para o novo objeto a ser criado.
O método **Object.create**, recebe dois parâmetros. O primeiro parâmetro é um objeto obrigatório que serve como protótipo para o novo objeto a ser criado. O segundo parâmetro é um objeto opcional que contém as propriedades a serem adicionadas ao novo objeto.

// Criação de objetos a partir de outros objetos:

```
// criação do objeto que representa uma organização:
const objOrg = { empresa: 'JMB Solutions' };

// criação do objeto funcionário a partir do objeto organização:
const funcionario = Object.create(objOrg, { nome: { valor: 'FuncionarioUm' } });

console.log(funcionario); // { empresa: " JMB Solutions " }
console.log(funcionario.nome); // "FuncionarioUm"
```

17 – Outra possibilidade é a criação de objetos a partir de mais de um objeto. O método **Object.assign()** é usado para copiar os valores de todas as propriedades próprias enumeráveis de um ou mais objetos *de origem* para um objeto de *destino*. Este método retornará o objeto de *destino*.

```
// criação dos objetos:
const objOrg = { empresa: 'JMB Solutions' }
const objCarro = { nomeCarro: 'Ford' }

const funcionario = Object.assign({}, objOrg, objCarro);
console.log(funcionario);
```

18 – O último conteúdo dessa trilha, diz respeito a construção de classes que possuem outras como atributo, veja o exemplo no código a seguir:

// Criação de objetos com outros objetos como atributo.

```
// Definindo a classe Cliente
class Cliente {
    constructor(nome, cpf) {
        this.nome = nome;
        this.cpf = cpf;
    }

    toString() {
        return `${this.nome} (${this.cpf})`;
    }
}

// Definindo a classe ContaBancaria que possui um Cliente
class ContaBancaria {
    constructor(numero, saldoInicial, cliente) {
        this.numero = numero;
        this.saldo = saldoInicial;
        this.cliente = cliente; // Cliente é um atributo da ContaBancaria
    }

    depositar(valor) {
        this.saldo += valor;
        console.log(`Depósito de ${valor} realizado. Novo saldo: ${this.saldo}`);
    }

    sacar(valor) {
        if (valor <= this.saldo) {
            this.saldo -= valor;
            console.log(`Saque de ${valor} realizado. Novo saldo: ${this.saldo}`);
        } else {
            console.log("Saldo insuficiente.");
        }
    }

    toString() {
        return `Conta ${this.numero}, Cliente: ${this.cliente}, Saldo: ${this.saldo}`;
    }
}

// Exemplo de uso das classes
let cliente1 = new Cliente("João da Silva", "123.456.789-00");
let conta1 = new ContaBancaria("001", 1000, cliente1);

console.log(conta1.toString()); // Saída: Conta 001, Cliente: João da Silva (123.456.789-
00), Saldo: 1000

conta1.depositar(500); // Saída: Depósito de 500 realizado. Novo saldo: 1500
conta1.sacar(200); // Saída: Saque de 200 realizado. Novo saldo: 1300

console.log(conta1.toString()); // Saída atualizada: Conta 001, Cliente: João da Silva
(123.456.789-00), Saldo: 1300
```

// EXERCÍCIOS:

19 – Exercícios para criação de Classes e objetos.

- Crie um objeto Produto: nome, valor unitário, marca.
- Crie um objeto Venda: código, data, Produto.
- Crie um arquivo .html onde sejam mostrados os dados da venda.

// Criação de forma literal:

```
let produto = {
  nome: 'Notebook',
  preco: 2500.00,
  quantidade: 1
};

let venda = {
  id: 12345,
  data: '2024-07-20',
  produto: produto, // Produto está sendo incluído dentro de Venda
  total: function() {
    return this.produto.preco * this.produto.quantidade;
  }
};

console.log(venda.id); // 12345
console.log(venda.data); // '2024-07-20'
console.log(venda.produto.nome); // 'Notebook'
console.log(venda.produto.preco); // 2500.00
console.log(venda.produto.quantidade); // 1
console.log(venda.total()); // 2500.00

venda.produto.quantidade = 2;

console.log(venda.produto.quantidade); // 2
console.log(venda.total()); // 5000.00
```

// Criação através de construtor de objetos

```
function Produto(nome, preco, quantidade)
  {
    this.nome = nome;
    this.preco = preco;
    this.quantidade = quantidade;
   };

    function Venda (id, data, produto) {
     this.id = id;
     this.data = data;
     this.produto = produto, // Produto está sendo incluído dentro de Venda
      total = function() {
       return this.produto.preco * this.produto.quantidade;
    }
  };
```

20 – Exercício: Construir e utilizar os objetos: Carrinho, produtos, venda.